2016

Randomized Algorithms for Nonconvex Nonsmooth Optimization

Xiaocun Que
Lehigh University

Randomized Algorithms for Nonconvex Nonsmooth Optimization

by

Xiaocun Que

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Industrial Engineering

Lehigh University

January 2016

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Date

Dissertation Advisor

Committee Members:

Frank E. Curtis, Committee Chair

Michael L. Overton

Katya Scheinberg

Ted K. Ralphs

Acknowledgements

First, I would like to thank my advisor Prof. Frank E. Curtis for his excellent guidance during my Ph.D. years. Second, I would like to thank the members of my dissertation committee, Prof. Michael L. Overton, Prof. Katya Scheinberg, and Prof. Ted K. Ralphs, for their insightful questions, comments, and suggestions. Third, I would like to thank the Computational Optimization Research Lab (COR@L) in the Industrial and Systems Engineering (ISE) Department at Lehigh University, because most of the numerical experiments presented in this thesis were performed using software and hardware in COR@L. Fourth, I would like to thank the National Science Foundation (NSF), because this work was financially supported in part by NSF grant DMS-1016291. Finally, I would like to thank my husband Xiong Xiong for his continuous support since we got married.

Contents

List of Tables

List of Figures

Abstract

Nonsmooth optimization problems arise in a variety of applications including robust control, robust optimization, eigenvalue optimization, compressed sensing, and decomposition methods for large-scale or complex optimization problems. When convexity is present, such problems are relatively easier to solve. Optimization methods for convex nonsmooth optimization have been studied for decades. For example, bundle methods are a leading technique for convex nonsmooth minimization. However, these and other methods that have been developed for solving convex problems are either inapplicable or can be inefficient when applied to solve nonconvex problems. The motivation of the work in this thesis is to design robust and efficient algorithms for solving nonsmooth optimization problems, particularly when nonconvexity is present.

First, we propose an adaptive gradient sampling (AGS) algorithm, which is based on a recently developed technique known as the gradient sampling (GS) algorithm. Our AGS algorithm improves the computational efficiency of GS in critical ways. Then, we propose a BFGS gradient sampling (BFGS-GS) algorithm, which is a hybrid between a standard Broyden-Fletcher-Goldfarb-Shanno (BFGS) and the GS method. Our BFGS-GS algorithm is more efficient than our previously proposed AGS algorithm and also competitive with (and in some ways outperforms) other contemporary solvers for nonsmooth nonconvex optimization. Finally, we propose a few additional extensions of the GS framework—one in which we merge GS ideas with those from bundle methods, one in which we incorporate smoothing techniques in order to minimize potentially non-Lipschitz objective functions, and one in which we tailor GS methods for solving regularization problems.

We describe all the proposed algorithms in detail. In addition, for all the algorithm variants, we prove global convergence guarantees under suitable assumptions. Moreover, we perform numerical experiments to illustrate the efficiency of our algorithms. The test problems considered in our experiments include academic test problems as well as practical problems that arise in applications of nonsmooth optimization.

Chapter 1

Introduction

This dissertation involves a study of the minimization of locally Lipschitz objective functions that may be both nonsmooth and nonconvex. Problems of this type arise in a variety of applications including robust control [27, 28, 58, 59], robust optimization [3, 23, 60], eigenvalue optimization [1], compressed sensing [12, 20], and decomposition methods for large-scale or complex optimization problems [5, 52]. Solutions of such problems often lie at points of nondifferentiability of the objective. This makes it imperative to design robust and efficient algorithms for the optimization of nonsmooth functions.

A variety of algorithms have been proposed for nonsmooth optimization. Many, however, are based on the assumption that the objective function is convex. For example, bundle methods [34], which rely on the ability to produce linear underestimators of the objective (i.e., cutting planes) are a leading technique for convex nonsmooth minimization. There are extensions to traditional bundle methods for solving nonconvex problems, but these methods are complex and we believe that alternative strategies may be better suited for handling nonconvexity.

The goal of the research outlined in this thesis is to develop, analyze, and implement efficient methods for solving nonsmooth optimization problems, particularly when nonconvexity is present. We study and propose extensions for a recently developed technique known as the gradient sampling (GS) algorithm [9, 39]. In contrast to bundle methods, GS handles nonconvexity without any extra algorithmic modifications, which makes it an attractive starting point for devising new methods for nonconvex optimization. The methods that we propose also incorporate quasi-Newton strategies, for which many have observed good practical performance, even when they are applied to solve nonsmooth problems [44].

After providing theoretical background on nonconvex optimization problems and algorithms that have been proposed for solving them, we begin the main part of this thesis by describing research that addresses some efficiency issues of GS. In Chapter 2, we propose an adaptive gradient sampling (AGS) algorithm, which improves the computational efficiency of GS by incorporating an adaptive sampling technique and Hessian updating strategies. Our numerical experiments illustrate that AGS outperforms GS in critical ways. In Chapter 3, we propose a BFGS Gradient Sampling (BFGS-GS) Algorithm, which is a hybrid between a standard Broyden-Fletcher-Goldfarb-Shanno (BFGS) and the GS method. The algorithm has been implemented in C++ and the results of numerical experiments are presented to illustrate the efficacy of the proposed numerical method.

The remainder of the thesis considers further extensions to the GS framework. In particular, in Chapter 4.1, we propose a bundle gradient sampling (BGS) algorithm that merges GS strategies with those of bundle methods so that the overall approach remains effective for convex problems and does not require algorithmic modifications to handle nonconvexity. We combine the two strategies into a single "bundle sampling" framework, provide theoretical convergence guarantees that are on par with those currently held by GS, and provide the results of numerical experiments to illustrate the computational performance of our new method. In Chapter 4.2, we propose a smoothing BFGS gradient sampling algorithm, which is based on the smoothing method and our BFGS-GS algorithm for nonsmooth optimization. A motivation for the smoothing approach is that it has theoretical convergence guarantees even when the problem functions are not Lipschitz. (This is more than can be said about the other algorithms in the thesis.) Numerical results are presented to illustrate that our algorithm is competitive with another recently proposed smoothing method for non-Lipschitz optimization. In Chapter 4.3, we tailor GS methods to solve regularized problems. Global convergence analysis is provided. Preliminary numerical experiments are performed to compare different algorithmic variations of GS with another algorithm proposed for solving regularization problems.

1.1 Theoretical Background

In this section, we provide essential definitions and background for the study of minimizing nonsmooth functions. We also outline notation that will be used throughout the thesis.

We consider the unconstrained problem

$$\min_{x\in\mathbb{R}^n} \; f(x) \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz and continuously differentiable in an open dense (see below) subset $\mathcal{D}$ of $\mathbb{R}^n$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is *Lipschitz continuous* [49] if there exists a constant $L > 0$ such that

$$\|f(x) - f(y)\|_2 \leq L\|x - y\|_2 \tag{1.2}$$

for all $x, y \in \mathbb{R}^n$. The constant L is called the Lipschitz constant which is independent of x and y. There are a variety of convenient features of Lipschitz continuous functions. In short, a Lipschitz continuous function is limited in how fast it can change. Also, for any two points on the graph of a Lipschitz continuous function, the absolute value of the slope of the line joining those two points is bounded above by a constant. Given a particular point x, a function $f : \mathbb{R}^n \to \mathbb{R}$ is *locally Lipschitz continuous* [49] at $x \in \mathbb{R}^n$ with a constant $L > 0$ if (1.2) holds for all y and z in a neighborhood of x.

A subset $\mathcal{D}$ of $\mathbb{R}^n$ is called *dense* [57] if any neighborhood of $x \in \mathbb{R}^n$ contains at least one point in $\mathcal{D}$. Note that this means that the closure of $\mathcal{D}$ is $\mathbb{R}^n$ and that the interior of the complement of $\mathcal{D}$ is the empty set. An important consequence of our assumption that f is continuously differentiable in such a set $\mathcal{D}$ is that there exist points at which f is differentiable in any arbitrarily small neighborhood of a given point x.

We now turn to notions of stationarity for locally Lipschitz functions that are essential for deriving optimality conditions for problem (1.1). We define the Euclidean ϵ-ball about x to be

$$\mathbb{B}_\epsilon(x) := \{\overline{x} : \|\overline{x} - x\|_2 \leq \epsilon\}. \tag{1.3}$$

Moreover, let $\operatorname{cl}\operatorname{conv} S$ denote the closure of the convex hull of $S \subseteq \mathbb{R}^n$. The multifunction

$$\mathbb{G}_\epsilon(x) := \operatorname{cl}\operatorname{conv} \nabla f(\mathbb{B}_\epsilon(x) \cap \mathcal{D}) \tag{1.4}$$

can then be seen as the closure of the convex hull of the gradients at all the points in the intersection of an ϵ-ball about x and the set $\mathcal{D}$ in which f is differentiable. Given these definitions, the *Clarke subdifferential* [15] of f at x can be expressed as the following:

$$\overline{\partial} f(x) = \bigcap_{\epsilon > 0} \mathbb{G}_\epsilon(x).$$

A point x is *stationary* for f if $0 \in \overline{\partial} f(x)$. The gradient sampling algorithm discussed in detail later on in this thesis makes use of an extension to the subdifferential, namely the ϵ-subdifferential introduced by Goldstein [24]. The Clarke ϵ-subdifferential is given by

$$\overline{\partial}_\epsilon f(x) := \operatorname{cl}\operatorname{conv} \overline{\partial} f(\mathbb{B}_\epsilon(x)),$$

and x is ϵ-stationary if $0 \in \overline{\partial}_\epsilon f(x)$. Observe from this definition that a reasonable strategy for computing a stationary point for f is to compute a sequence of ϵ-stationary points for $\epsilon \to 0$.

As previously mentioned, we are interested in the minimization of nonsmooth functions that may also be nonconvex. We do, however, make extensive comments pertaining exclusively to convex functions, and it is important to distinguish definitions of quantities that suppose convexity. A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* [53] if

$$f(\lambda x + (1 - \lambda) y) \leq \lambda f(x) + (1 - \lambda) f(y)$$

for any $x, y \in \mathbb{R}^n$ and $\lambda \in [0, 1]$. Namely, a function is said to be convex if the graph of the function lies below the line segment joining any two points of the graph. It is known that a real-valued convex function is guaranteed to be locally Lipschitz continuous at any x. Moreover, the subdifferential of a convex function f at x is the set

$$\partial f(x) = \{g \in \mathbb{R}^n \mid f(y) \geq f(x) + g^T(y-x) \ \ \forall y \in \mathbb{R}^n\}.$$

Each vector $g \in \partial f(x)$ is called a *subgradient* [53] of f at x.

1.2 Classical Algorithms

The nondifferentiability of the objective function f in (1.1) excludes the direct application of smooth gradient-based algorithms. Therefore, in this section we introduce some basic methods for solving nonsmooth optimization problems: the subgradient method, cutting plane method, proximal point method, bundle method, gradient sampling method, and smoothing method. For the first four algorithms in this section, note that all rely on the assumption that f is convex. This is important as, later on, we aim to design algorithms that do not make this assumption. Our descriptions of the first four algorithms in this section are based on the descriptions in [53]. The description of the gradient sampling (GS) method is based on the description in [9] and [39]. The description of the smoothing method is based on the description in [13].

1.2.1 Subgradient Method

As the gradient descent method is the most basic algorithm for smooth differentiable optimization, the subgradient method is the most basic method for nonsmooth problems. The approaches are nearly identical, but the idea behind the subgradient method is to replace the gradient of f at x with any arbitrary subgradient. Given an iterate x_k, an iteration of the subgradient method is given by

$$x_{k+1} = x_k - \alpha_k g_k \quad \text{for } k = 0, 1, 2, \cdots \tag{1.5}$$

where $g_k \in \partial f(x_k)$ is a subgradient of f at x_k and α_k is a positive step size.

There are some serious drawbacks of the subgradient method. First, note that a negative subgradient direction g_k is not necessarily a direction of descent of f from x_k. Therefore, under various step size selection rules, the sequence $\{f(x_k)\}$ in subgradient methods is not guaranteed to be nonincreasing. Moreover, some standard line search techniques (e.g., the Armijo or Wolfe conditions [49]) cannot be applied for choosing α_k.

There are certain step size selection rules that do guarantee global convergence of the method, but in many cases these rules are important only for their theoretical significance and are rarely used in practice due to their low efficiency. The key property of the subgradient method is that a small step in the direction negative to g_k will decrease the distance to the optimal solution set. This fact is used in the proofs of many convergence theorems [53].

Another drawback of the subgradient method is that a theoretically sound and practically robust termination condition is elusive in many applications. For one thing, the norm of an arbitrary subgradient does not necessarily become small in the neighborhood of an optimal point, meaning that termination conditions typical in smooth optimization do not generally apply for nonsmooth problems.

An important special case of the subgradient method is when g_k is always chosen to be the minimum-norm subgradient of f at x_k. In such cases, g_k is always a descent direction from any x_k that is not a minimizer of f; in particular, it defines the direction of steepest descent for f from x_k. Despite this nice feature, however, one finds that as in smooth optimization, algorithms based on steepest descent directions can be slow to converge in practice. Moreover, computing the steepest descent direction for a nonsmooth function f at any point is not always a viable option.

Perhaps the only clear advantage of the subgradient method comes from its simple structure.

1.2.2 Cutting Plane Method

Given an assumption of convexity, the idea behind cutting plane methods is to use subgradient inequalities to construct a convex piecewise linear approximation of the objective function at each iterate x_k. Specifically, given points $\{x_1, \dots, x_k\}$, suppose that values of the objective $\{f(x_1), \dots, f(x_k)\}$ and subgradients $\{g_1, \dots, g_k\}$ have been accumulated from previous iterations. We can then construct the following lower approximation of f at x_k:

$$m_k^{CP}(x) := \max_{1 \leq j \leq k} \{f(x_j) + g_j^T (x - x_j)\}. \tag{1.6}$$

The minimization of the model function (1.6) is called the master problem:

$$\min_{x \in \mathbb{R}^n} \; m_k^{CP}(x). \tag{1.7}$$

After solving (1.7), a new iterate x_{k+1} is obtained. The iterate x_{k+1}, objective value $f(x_{k+1})$ and subgradient g_{k+1} can then be added to the model to construct a new linear underestimator of f. Each linear piece $f(x_j) + g_j^T (x - x_j)$, added at each iteration, is called a cutting plane (or simply a cut). A key property of the master problem (1.7) is that, due to the convexity of f, its optimal value provides a lower bound for the optimal value of (1.1).

The master problem (1.7) can be written equivalently as the following linear optimization (LO) problem:

$$\begin{aligned} \min_{(x,z) \in \mathbb{R}^n \times \mathbb{R}} \quad & z \\ \text{s.t. } \; & f(x_j) + g_j^T (x - x_j) \leq z, \quad j = 1, \dots, k. \end{aligned} \tag{1.8}$$

In this formulation, a new constraint is added to the problem after each cutting plane is computed, which means the number of dual variables is increased by one after each iteration. This means that re-optimization by a dual method is an attractive option because it can start with a feasible solution obtained from a previous iteration.

The cutting plane method, when applied to general convex problems, is rather slow.

One reason for this is that there exist no reliable rules for removing the old cuts, even when they are inactive at a given solution (1.8). Usually, very many iterations are needed to achieve satisfactory accuracy in the solution. Only in the special case when the objective function is also piecewise linear and convex does the cutting plane method become consistently efficient. Cutting plane methods are important, however, as they form a basis for more effective techniques.

1.2.3 Proximal Point Method

Consider the function

$$h(w) := \min_{x \in \mathbb{R}^n} \left\{ f(x) + \tfrac{1}{2}\|x - w\|^2 \right\}. \tag{1.9}$$

This is known as the Moreau-Yosida regularization of the objective function $f(x)$. If f is convex, then it can be shown that $h(w)$ is convex and continuously differentiable. (The Moreau-Yosida regularization is often defined with a positive scalar weighting the proximal term $\frac{1}{2}\|x - w\|^2$. This weight can affect the practical performance of the method, but it is not necessary for our purposes here or in the subsection on bundle methods below.) The variable w can be thought of as a centering term. The goal of (1.9) is to minimize the true objective f as well as stay close to the center w_k. (The reason that we use w_k's as the iterates instead of the x_k's like we did in previous subsections is that we want to be consistent with the notation defined in bundle methods, which we will introduce in the next subsection.) In bundle methods, it is necessary to define two sequences: the iterates x_k's and the centering terms w_k's. The w_k's can also be thought of as the "best iterates attained so far".

Using the Moreau-Yosida regularization of $f(x)$, the proximal point method constructs the following iterative process. At iteration k, given w_k, the point $x(w_k)$ is computed as the solution of the problem (1.9). This then defines the iterative sequence

$$w_{k+1} = x(w_k), \quad k = 1, 2, \ldots. \tag{1.10}$$

Since (1.9) always has a solution, the proximal point method is well-defined. Moreover, by the construction of the Moreau-Yosida regularization, we have $h(w_k) \leq f(w_k)$ if we plug in a feasible solution $x = w_k$ to the problem (1.9) with $w = w_k$. We also have $f(w_{k+1}) \leq h(w_k)$ if we notice that $x = x(w_k)$ is the optimal solution to the problem (1.9) with $w = w_k$. Therefore, we have $f(w_{k+1}) \leq f(w_k)$, $k = 1, 2, \ldots$, namely, the sequence $f(w_k)$ is nonincreasing.

The proximal point method has its disadvantages. It does not appear to be very practical, because each iteration involves the solution of the optimization problem (1.9), which is not easy to solve because of the existence of the original objective function $f(x)$ in the objective of (1.9). However, the proximal point method is an important theoretical model of various highly efficient methods such as bundle methods, described next.

1.2.4 Bundle Method

Bundle methods are regarded as very effective and reliable methods for nonsmooth optimization. The basic idea of bundle methods is to approximate the subdifferential of the objective function by gathering subgradient inequalities from previous iterations into a bundle. This makes them similar to cutting plane methods in that they require the computation of one arbitrary subgradient and the objective value at each new iterate. A critical difference, however, is that the search direction is obtained by solving a specially designed quadratic optimization (QO) problem, not a LO problem. This helps bundle methods avoid some of the disadvantages of a straightforward cutting plane technique.

The bundle method we introduce in this section is a hybrid of the cutting plane method and the proximal point method that were introduced in previous sections. At iteration k, we define the following regularized master problem:

$$\min_{x \in \mathbb{R}^n} m_k^{BM}(x) + \tfrac{1}{2}\|x - w_k\|^2. \tag{1.11}$$

This problem is exactly the same as problem (1.9) except that we use a model $m_k^{BM}(x)$ instead of the true objective $f(x)$. The model $m_k^{BM}(x)$ is defined as the following, which is similar to the model (1.7) defined in the cutting plane method:

$$m_k^{BM}(x) := \max_{j \in J_k}\{f(x_j) + g_j^T(x - x_j)\}. \tag{1.12}$$

Here, similar to before, $g_j \in \partial f(x_j)$ are arbitrary subgradients computed during the iterations in the index set $J_k \subset \{1, \ldots, k\}$. We may think of J_k as being equal to $\{1, \ldots, k\}$, but note that under certain circumstances an index can be removed from J_k (i.e., a cutting plane can be removed from m_k^{BM}) without adversely affecting the performance of the algorithm.

Let x_{k+1} be the solution of the regularized master problem (1.11). If the model $m_k^{BM}(x)$ is exact in the sense that $m_k^{BM}(x) = f(x)$ for all $x \in \mathbb{R}^n$, then (1.11) would be identical to problem (1.9) defined for the proximal point method. We could then set $w_{k+1} = x_{k+1}$ as in the proximal point method to obtain the new w_{k+1}, and, in this manner, all steps would be descent steps. That is, all steps would be those where the objective function value has decreased. However, due to the fact that $m_k^{BM}(x)$ only approximates $f(x)$, the solution of (1.11) is different than the solution of (1.9). In particular x_{k+1} may not even be better than w_k in terms of minimizing f. This necessitates defining a condition under which the estimate of the optimal solution (i.e., w) is updated or remains the same.

For this purpose, we introduce a parameter γ used for updating w_k. If the ratio of the observed improvement in the objective value over the predicted improvement is greater than γ, namely,

$$\frac{f(w_k) - f(x_{k+1})}{f(w_k) - m_k(x_{k+1})} \geq \gamma, \tag{1.13}$$

then we set $w_{k+1} := x_{k+1}$. This is called a descent step as we obtain $f(w_{k+1}) < f(x_k)$. Otherwise, we set $w_{k+1} := w_k$. This is called a null step. Even though the objective has not improved due to a null step, by the addition of a new cut, it can be shown that the model m_{k+1}^{BM} is a sufficient improvement over m_k^{BM} in that, after a finite number of null steps, a descent step will be produced.

Similar to the cutting plane method, the regularized master problem (1.11) can be equivalently written as a problem with a quadratic objective function and linear constraints:

$$\begin{aligned} \min_{(x,z)\in\mathbb{R}^n\times\mathbb{R}} \quad & z + \tfrac{1}{2}\|x - w_k\|^2 \\ \text{s.t. } & f(x_j) + g_j^T(x - x_j) \le z, \quad j \in J_k. \end{aligned} \tag{1.14}$$

We provide a detailed description of a bundle method as Algorithm 1 below.

Algorithm 1 The Bundle Method

```
1: (Initialization): Choose a parameter $\gamma \in (0,1)$. Choose an initial $x_0 \in \mathcal{D}$, set $J_{-1} \leftarrow \emptyset$,
   $z_0 \leftarrow -\infty$ and $k \leftarrow 0$.
2: (Bundle addition): Compute $f(x_k)$ and $g_k \in \partial f(x_k)$. If $f(x_k) > z_k$, then $J_k \leftarrow$
   $J_{k-1} \cup \{k\}$; otherwise, $J_k \leftarrow J_{k-1}$.
3: (Step update): If $k = 0$ or if $f(x_k) \le (1-\gamma)f(w_{k-1}) + \gamma z_k$ (recall (1.13)), then set
   $w_k \leftarrow x_k$; otherwise set $w_k \leftarrow w_{k-1}$.
4: (Search direction computation): Solve the master problem (1.14) to obtain
   $(x_{k+1}, z_{k+1})$.
5: (Stationarity test): If $z_{k+1} = f(w_k)$, then stop; $w_k$ is an optimal solution.
6: (Bundle removal): Remove from $J_k$ some (or all) cuts whose Lagrange multipliers at
   the solution of the master problem (1.14) are 0.
7: (Iteration increment): Set $k \leftarrow k + 1$ and go to step 2.
```

We refer to [53] for the following convergence result for BM.

Theorem 1.2.1. *Suppose that the objective function f of problem* (1.1) *is convex and that problem has an optimal solution. Then, the sequence $\{w_k\}$ generated by the bundle method converges to an optimal solution of* (1.1).

We close this section with an illustrative example of the workings of the bundle method. Consider minimizing the following objective function:

$$f(x) = \max_{x\in\mathbb{R}}\{x^2, 2x\}. \tag{1.15}$$

Suppose we start with $x_0 = 1$. Then we have the following QO subproblem:

$$\begin{aligned} \min_{(x,z)\in\mathbb{R}\times\mathbb{R}} \quad & z + \tfrac{1}{2}(x-1)^2 \\ \text{s.t. } & 2 + 2(x-1) \le z. \end{aligned} \tag{1.16}$$

The blue curve in Figure 1.1 corresponds to the objective (1.15). In the plot on the left, the red line is the first cut $2 + 2(x - 1)$; the green line corresponds to the objective in an unconstrained reformulation of (1.16). After solving the QO subproblem (1.16), we move to $x_1 = -1$. Then we have the following QO subproblem:

$$\begin{aligned} \min_{(x,z)\in\mathbb{R}\times\mathbb{R}} \quad & z + \tfrac{1}{2}(x+1)^2 \\ \text{s.t.} \quad & 2 + 2(x-1) \le z \\ & 1 - 2(x+1) \le z. \end{aligned} \tag{1.17}$$

We can see from the plot on the right, another cut $1 - 2(x + 1)$ is added to the plot. After solving (1.17), we move closer to the optimal point. If we continue this process, the bundle method will find the optimal solution.

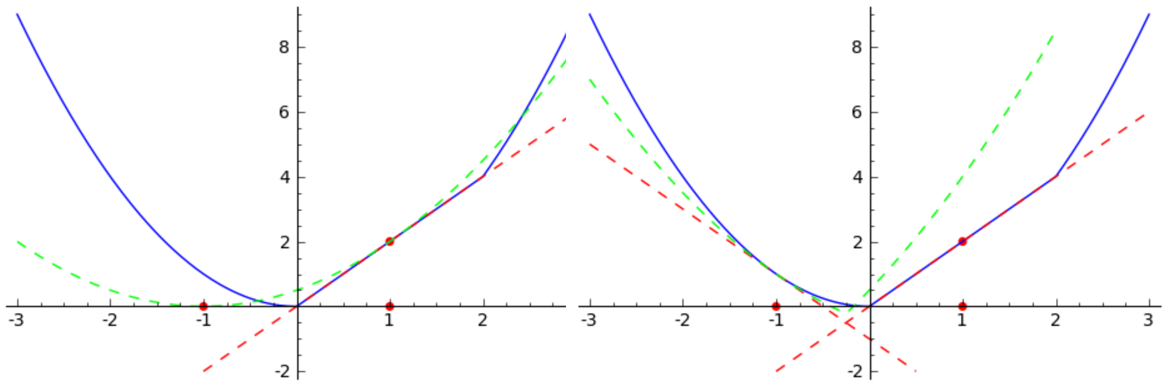


Figure 1.1: Illustration of the Bundle Method.

1.2.5 Gradient Sampling Method

The original GS algorithm was introduced and analyzed by Burke, Lewis, and Overton [9] for problems of form (1.1). Stronger theoretical results for a slightly revised version of GS were provided in [39], and further extensions have been considered for constrained problems [16] and problems for which only function evaluations are available [40].

The algorithmic structure of GS is very straightforward. At each iteration, we first sample a group of points around the current iterate and evaluate the gradient of f at the current iterate and at the sample points. The search direction is then set as the negative of the vector in the convex hull of the available gradients with smallest norm.

Finally, a backtracking line search is used to obtain a point with a lower objective value. The algorithm starts with an arbitrary positive initial sampling radius, updating it when appropriate to ensure that a stationary point for f is obtained.

More precisely, at a given iterate x_k and for a given sampling radius $\epsilon_k > 0$, the central idea behind gradient sampling techniques is to approximate $\mathbb{G}_{\epsilon_k}(x_k)$ (recall (1.4)) through the random sampling of gradients in $B_k := \mathbb{B}_{\epsilon_k}(x_k) \cap \mathcal{D}$. This set, in turn, approximates the Clarke ϵ_k-subdifferential since, at any x, $\mathbb{G}_{\epsilon}(x) \subset \overline{\partial}_{\epsilon} f(x)$ for any $\epsilon \geq 0$ and $\overline{\partial}_{\epsilon'} f(x) \subset \mathbb{G}_{\epsilon''}(x)$ for any $\epsilon'' > \epsilon' \geq 0$. If the computed search direction is large, then as it is easily shown to be a direction of descent for f, the line search easily produces a new iterate with an improved objective value. Otherwise, by locating x_k at which the minimum-norm element of $\mathbb{G}_{\epsilon_k}(x_k)$ is small, reducing the sampling radius, and then repeating the process for $\epsilon_k \to 0$, gradient sampling techniques locate stationary points of f by successively locating (approximate) ϵ_k-stationary points for decreasing values of ϵ_k.

We now give a detailed description of the GS algorithm. During iteration k, let $X_k := x_k \cup \overline{X}_k$ where x_k is the current iterate and $\overline{X}_k := \{\overline{x}_{k,1}, \ldots, \overline{x}_{k,\overline{p}}\}$ is composed of $\overline{p} \geq n+1$ points generated independently and uniformly in B_k. With

$$\mathcal{G}_k := \operatorname{conv}\{g_k, g_{k,1}, \ldots, g_{k,\overline{p}}\} \tag{1.18}$$

defined as the convex hull of the gradients at the points in $\overline{X}_k$, the search direction is set to be the negative of the minimum norm vector in $\mathcal{G}_k$, namely, $d_k = -\operatorname{Proj}(0|\mathcal{G}_k)$. This can be obtained via the solution of a QO problem. Specifically, in order to compare the QO of GS with the QO of AGS and BGS in later sections, we write the QO in a primal-dual form. Let

$$G_k := \begin{bmatrix} g_k & g_{k,1} & \cdots & g_{k,\overline{p}} \end{bmatrix} \tag{1.19}$$

denote the matrix whose columns are the gradients of f at the points in X_k. Then we have the following QO subproblem:

$$\begin{aligned}\min_{z,d}\ & z+\tfrac{1}{2}\|d\|^2\\ \text{s.t. } & f(x_k)e+G_k^Td\le ze.\end{aligned}\tag{1.20}$$

Here, e denotes a vector of ones whose length is determined by the context. The dual of (1.20) is given by

$$\begin{aligned}\max_{\pi}\ & -\tfrac{1}{2}\|G_k\pi\|^2\\ \text{s.t. } & e^T\pi=1,\ \pi\ge 0,\end{aligned}\tag{1.21}$$

The solution (z_k, d_k, π_k) of (1.20)–(1.21) has a relationship that $d_k = -G_k\pi_k$.

A specific GS algorithm is presented as Algorithm 2 below.

Algorithm 2 Gradient Sampling (GS) Algorithm

```
1: (Initialization): Choose a number of sample points to compute each iteration $\overline{p} > n+1$, sampling radius reduction factor $\psi \in (0,1)$, sufficient decrease constant $\eta \in (0,1)$, line search backtracking constant $\kappa \in (0,1)$, and tolerance parameter $\nu > 0$. Choose an initial iterate $x_0 \in \mathcal{D}$, set $X_{-1} \leftarrow \emptyset$, choose an initial sampling radius $\epsilon_0 > 0$, and set $k \leftarrow 0$.
2: (Sample set update): Set $X_k \leftarrow x_k \cup \overline{X}_k$, where $\overline{X}_k := \{\overline{x}_{k,1}, \dots, \overline{x}_{k,\overline{p}}\}$
3: (Search direction computation): Set $d_k \leftarrow -G_k\pi_k$, where $\pi_k$ solves (1.21).
4: (Stationarity test): If $\|d_k\| \le \epsilon_k \le \nu$, then stop. Otherwise, if $\|d_k\| \le \epsilon_k$, then set $x_{k+1} \leftarrow x_k$, $\alpha_k \leftarrow 1$, and $\epsilon_{k+1} \leftarrow \psi\epsilon_k$ and go to step 7.
5: (Backtracking line search): Set $\alpha_k$ as the largest value in $\{\kappa^0, \kappa^1, \kappa^2, \dots\}$ such that the following sufficient decrease condition
       $f(x_k + \alpha_k d_k) \le f(x_k) - \eta\alpha_k\|d_k\|^2.$   (1.22)
   is satisfied.
6: (Iterate update): Set $\epsilon_{k+1} \leftarrow \epsilon_k$. If $x_k + \alpha_k d_k \in \mathcal{D}$, then set $x_{k+1} \leftarrow x_k + \alpha_k d_k$. Otherwise, set $x_{k+1}$ as any point in $\mathcal{D}$ satisfying the following perturbed line search conditions
       $f(x_{k+1}) \le f(x_k) - \eta\alpha_k\|d_k\|^2$   (1.23a)
       and $\|x_k + \alpha_k d_k - x_{k+1}\| \le \min\{\alpha_k, \epsilon_k\}\|d_k\|.$   (1.23b)
7: (Iteration increment): Set $k \leftarrow k+1$ and go to step 2.
```

Note that after the search direction d_k is computed, a standard backtracking line search is performed to find a step size α_k. We set $x_{k+1} \leftarrow x_k + \alpha_k d_k$ for α_k chosen to satisfy (1.22), but in order to ensure that all iterates remain within the set $\mathcal{D}$, it may be necessary to perturb such an x_{k+1}; see [39] for the motivation of these perturbed line search conditions and a description of how, given α_k and d_k satisfying (1.22), an x_{k+1} satisfying (1.23) can be found in a finite number of operations. The chance seems to be very slim for the algorithm to come to the situation that $x_k + \alpha_k d_k \notin \mathcal{D}$. Therefore, while one may choose to skip this step in practice, it is necessary in establishing convergence guarantees.

The GS algorithm structure is very simple, though its convergence analysis in somewhat complicated by the stochastic nature of the algorithm. The convergence result is stated as the following. Please refer to §3 of [39] for the convergence proof.

Theorem 1.2.2. *Let $\{x_k\}$ be a sequence generated by GS with $\nu = 0$. Then, with probability 1, Algorithm 2 does not stop, and either $f(x_k) \to -\infty$, or $\epsilon_k \to 0$ and every cluster point of $\{x_k\}$ is stationary for f.*

We close this subsection with a description about how GS works on the same example (1.15) we mentioned before. Suppose we start with $x_0 = 1$. Let $\epsilon_0 = 2$ be the initial sampling radius and $\bar{p} = 2$ be the number of points sampled per iteration. At $k = 0$, suppose we generate two points $x_{0,1} = -1$ and $x_{0,2} = 1.5$. Then we have the following QO subproblem:

$$\begin{aligned} \min_{(x,z)\in\mathbb{R}\times\mathbb{R}} \quad & z + \tfrac{1}{2}(x-1)^2 \\ \text{s.t. } & 2 + 2(x-1) \le z \\ & 2 - 2(x+1) \le z. \end{aligned} \tag{1.24}$$

The blue curve in Figure 1.2 corresponds to objective (1.15). In the plot on the left, the green line corresponds to the objective in an unconstrainded reformulation of (1.24). After solving the QO subproblem (1.24), we stay at the same point $x_1 = 1$. However, we shrink the sampling radius $\epsilon_1 = 1$. In the plot on the right, if we sample again in the region $[0, 2]$, we will have the following QO subproblem:

$$\begin{aligned} \min_{(x,z)\in\mathbb{R}\times\mathbb{R}} \quad & z + \tfrac{1}{2}(x-1)^2 \\ \text{s.t. } & 2 + 2(x-1) \le z. \end{aligned} \tag{1.25}$$

If we repeat this process, we will find the optimal solution.

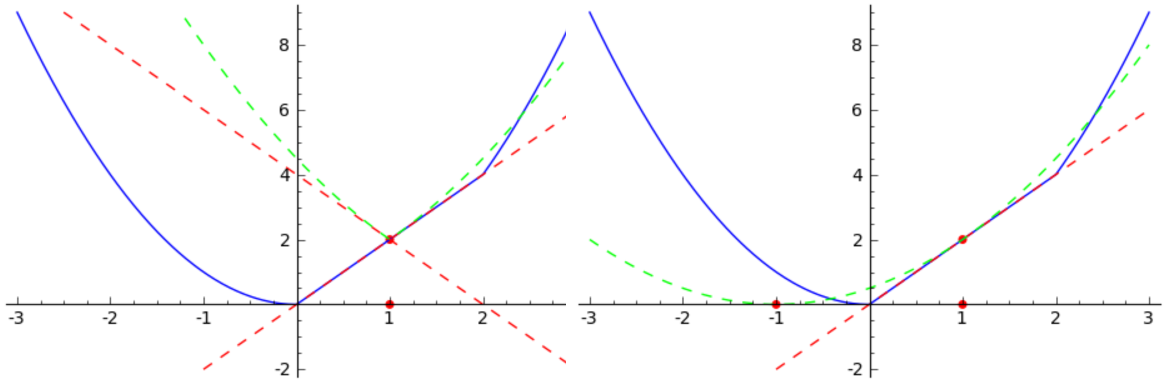


Figure 1.2: Illustration of the Gradient Sampling Algorithm.

1.2.6 Smoothing Method

The central idea behind a smoothing method is to use a parameterized smooth function to approximate the original nonsmooth objective function. The parameterization of the function is such that, if the smoothing parameter vanishes, then the original function is obtained. On the other hand, with a nonzero smoothing parameter, the given smoothed function can be minimized to produce an approximate minimizer of the original nonsmooth function, where any algorithm for smooth minimization can be used to minimize the smoothed function. By updating the smoothing parameter in an appropriate manner, one can show convergence to a minimizer of the original nonsmooth function.

In the smoothing method, we assume that the parameterized smoothing function satisfies the following assumption.

Assumption 1.2.3. *Let $f : R^n \to R$ be a continuous function with $\tilde{f} : R^n \times R^+ \to R$ its corresponding smoothing function. The smoothing function $\tilde{f}(\cdot, \mu)$ is continuously* *differentiable in $\mathbb{R}^n$ for any fixed $\mu > 0$, and we have*

$$\lim_{z \to x, \mu \to 0} \tilde{f}(z, x) = f(x)$$

for any $x \in \mathbb{R}^n$.

The smoothing method can be constructed by using the function and gradient value of the smoothing function, namely, $\tilde{f}$ and $\nabla_x \tilde{f}$. We present a description of a specific smoothing method as Algorithm 3 below.

Algorithm 3 The Smoothing Method

```
1: (Initialization): Choose a stationarity tolerance parameter $\nu > 0$, a smoothing param-
   eter reduction factor $\psi \in (0, 1)$, an initial iterate $x_0 \in \mathbb{R}^n$, and an initial smoothing
   parameter $\mu_0 > 0$. Set $k \leftarrow 0$.
2: (Inner iteration): Solve the following smooth optimization problem approximately to
   obtain an approximate solution $x_{k+1}$:
       $\min_x \tilde{f}(x, \mu_k)$
3: (Outer iteration): If $\|\nabla_x \tilde{f}(x_{k+1}, \mu_k)\| \geq \nu \mu_k$, then set $\mu_{k+1} \leftarrow \mu_k$; otherwise, choose
   $\mu_{k+1} \leftarrow \psi \mu_k$.
4: (Iteration increment): Set $k \leftarrow k + 1$ and go to step 2.
```

We refer to [13] for the following convergence result for the smoothing method.

Theorem 1.2.4. *The smoothing method produces an infinite sequence of iterates $\{x_k\}$ and either for some $\mu_k > 0$ we have $\tilde{f}(x_k, \mu_k) \to -\infty$ or $\{\mu_k\} \to 0$ and every cluster point of $\{x_k\}$ is stationary for f.*

The advantage of the smoothing method is that we can make use of many existing optimization algorithms for solving the smooth optimization problem in the inner iteration; and convergence to a stationary point of the original nonsmooth problem is guaranteed by updating the smoothing parameter, regardless of what algorithm used in the inner iteration. The efficiency of the smoothing method depends on the approximation function, the algorithm used for solving the smooth optimization problem in the inner iteration, and the updating strategy for the smoothing parameter.

The smoothing method also has its disadvantages. With the smoothing parameter approaching zero, the inner subproblem is smooth but may become very nonlinear. Even though any existing smooth optimization algorithm can be used to solve the inner subproblem, in practice they may not yield good solutions. Also, solving the inner subproblem exactly would be unnecessary and expensive; but on the other hand, there is no clear and general requirements about how approximately to solve it.

Chapter 2

An Adaptive Gradient Sampling Algorithm

We present an algorithm for the minimization of $f : \mathbb{R}^n \to \mathbb{R}$, assumed to be locally Lipschitz and continuously differentiable in an open dense subset $\mathcal{D}$ of $\mathbb{R}^n$. The objective f may be nonsmooth and/or nonconvex. The method is based on the gradient sampling algorithm (GS) of Burke, Lewis, and Overton [*SIAM J. Optim.*, 15 (2005), pp. 751-779]. It differs, however, from previously proposed versions of GS in that it is variable-metric and only $O(1)$ (not $O(n)$) gradient evaluations are required per iteration. Numerical experiments illustrate that the algorithm is more efficient than GS in that it consistently makes more progress toward a solution within a given number of gradient evaluations. In addition, the adaptive sampling procedure allows for warm-starting of the quadratic subproblem solver so that the average number of subproblem iterations per nonlinear iteration is also consistently reduced. Global convergence of the algorithm is proved assuming that the Hessian approximations are positive definite and bounded, an assumption shown to be true for the proposed Hessian approximation updating strategies.

2.1 Introduction

The gradient sampling algorithm (GS), introduced and analyzed by Burke, Lewis, and Overton [8, 9], is a method for minimizing an objective function $f : \mathbb{R}^n \to \mathbb{R}$ that is locally Lipschitz and continuously differentiable in an open dense subset $\mathcal{D}$ of $\mathbb{R}^n$. The approach is widely applicable and robust [7, 11, 42], and it is intuitively appealing in that theoretical convergence guarantees hold with probability one without requiring algorithmic modifications to handle nonconvexity.

The theoretical foundations for GS, as well as various extensions, are developing rapidly. Stronger theoretical results than in [9] for both the original algorithm and for various extensions were provided in [39], an extension of the ideas for solving constrained problems was presented in [16], and a variant using only gradient estimates derived via function evaluations appeared in [40]. Continued developments along these lines may allow GS techniques to one day be competitive with bundle methods [34, 37] in terms of theoretical might and practical performance.

The main goal of this chapter is to address three practical limitations of GS as it is presented in [9, 39]. Consider the following remarks.

1. GS produces approximate ϵ-steepest descent directions by evaluating the gradient of f at $n+1$ (or more) randomly generated points during each iteration. This results in a high computational cost that is especially detrimental when search directions turn out to be unproductive.

2. Each descent direction produced by GS is obtained by the solution of a quadratic optimization subproblem (QO). As the subproblem data is computed afresh for every iteration, the computational effort required to solve each of these subproblems can be significant for large-scale problems.

3. GS may behave, at best, as a steepest descent method. The use of second order information of the problem functions may be useful, but it is not clear how to incorporate this information effectively in nonsmooth regions.

We address both remarks (1) and (2) by the *adaptive* sampling of gradients over the course of the optimization process. That is, rather than evaluate gradients at a completely new set of points during every iteration k, we maintain a history and reuse any recently stored gradients that were obtained in an ϵ-neighborhood of x_k. This reduces the per-iteration computational effort of gradient evaluations, and also provides a clear strategy for warm-starting the QO solver. That is, any gradients corresponding to *active* subproblem constraints during iteration $k-1$ that remain in the set of sample gradients are included in the initial active set when solving the QO during iteration k. We show in our numerical experiments that adaptive sampling allows the algorithm to make much more progress toward a solution within a fixed number of gradient evaluations.

We address remark 3 by proposing two novel strategies for updating approximations of second order terms. The first strategy is similar to a limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) update typical in smooth optimization [48]. Our method is unique, however, in that we incorporate gradient information from sample points instead of that solely at algorithm iterates. We also control the updates so that bounds on the Hessian approximations required for our convergence analysis are obtained. The second strategy we propose — intended solely for nonconvex problems — is entirely novel as far as we are aware. It also involves the incorporation of function information at sample points, but is based on the desire to produce model functions that overestimate the true objective f. Bounds required for our convergence analysis are also proved for this latter strategy. Our numerical experiments in §2.5 illustrate that our Hessian approximation strategies further enhance the algorithm's ability to progress toward a solution within a given amount of computational effort.

The chapter is organized as follows. A description of our Adaptive Gradient Sampling algorithm (AGS) is presented in §2.2. Our updating strategies for approximating second order information are presented and analyzed in §2.3. Global convergence of a generic AGS algorithm is analyzed in §2.4. Numerical experiments comparing implementations of GS and variants of AGS on a large test set are presented in §2.5. This implementation involves a specialized QO solver that has been implemented by enhancing the method proposed in

[38]; the details of this solver are described in the Appendix. Finally, concluding remarks are provided in §2.6.

The analysis in this chapter builds on that of Kiwiel in [39]. It should also be noted that ideas of "incremental sampling" and "bundling past information" were briefly mentioned by Kiwiel in [40]. However, our methods are unique from those appearing in these papers as adaptive sampling was not considered in [39], exact gradient information was not used in [40], and our algorithm involves Hessian approximations that were not considered in either article. Still, in addition to the original work by Burke, Lewis, and Overton [9], it is clear that the works of Kiwiel have been inspirational for the work in this chapter, not to mention the QO algorithm from [38] that has found a new area of applicability in the context of AGS. Finally, we mention that the idea of sampling function information about a given point to approximate the subdifferential has been around for decades; e.g., see [29].

2.2 Algorithm Description

Consider the unconstrained problem

$$\min_{x} \; f(x) \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz and continuously differentiable in an open dense subset $\mathcal{D}$ of $\mathbb{R}^n$. Letting $\operatorname{cl}\operatorname{conv} S$ denote the closure of the convex hull of a set $S \subseteq \mathbb{R}^n$ and defining the multifunction $\mathbb{G}_\epsilon(x) := \operatorname{cl}\operatorname{conv} \nabla f(\mathbb{B}_\epsilon(x) \cap \mathcal{D})$ where $\mathbb{B}_\epsilon(x) := \{\overline{x} : \|\overline{x} - x\| \leq \epsilon\}$ is the Euclidean ϵ-ball about x, we have the following representation of the Clarke subdifferential [15] of f at x:

$$\overline{\partial} f(x) = \bigcap_{\epsilon > 0} \mathbb{G}_\epsilon(x).$$

Similarly, the Clarke ϵ-subdifferential [24] is given by

$$\overline{\partial}_\epsilon f(x) := \operatorname{cl}\operatorname{conv} \overline{\partial} f(\mathbb{B}_\epsilon(x)).$$

A point x is stationary for f if $0 \in \overline{\partial} f(x)$ and ϵ-stationary if $0 \in \overline{\partial}_\epsilon f(x)$.

At a given iterate x_k and for a given sampling radius $\epsilon_k > 0$, the central idea behind gradient sampling techniques is to approximate $\mathbb{G}_{\epsilon_k}(x_k)$ through the random sampling of gradients in $\mathbb{B}_{\epsilon_k}(x_k) \cap \mathcal{D}$. This set, in turn, approximates the Clarke ϵ_k-subdifferential at x_k since, at any x, $\mathbb{G}_\epsilon(x) \subset \overline{\partial}_\epsilon f(x)$ for any $\epsilon \geq 0$ and $\overline{\partial}_{\epsilon'} f(x) \subset \mathbb{G}_{\epsilon''}(x)$ for any $\epsilon'' > \epsilon' \geq 0$. Thus, by locating x_k at which there is a small minimum-norm element of (an approximation of) $\mathbb{G}_{\epsilon_k}(x_k)$, reducing the sampling radius, and then repeating the process, gradient sampling techniques locate stationary points of f by repeatedly locating (approximate) ϵ_k-stationary points for $\epsilon_k \to 0$.

We now present a generic AGS algorithm of which GS is a special case. During iteration k, let $X_k := \{x_{k,0}, \ldots, x_{k,p_k}\}$ (with $x_{k,i} = x_k$ for some i) denote a set of points that have been generated in $B_k := \mathbb{B}_{\epsilon_k}(x_k) \cap \mathcal{D}$, let

$$G_k := \begin{bmatrix} g_{k,0} & \cdots & g_{k,p_k} \end{bmatrix} \tag{2.2}$$

denote the matrix whose columns are the gradients of f at the points in X_k, and let $H_k \in \mathbb{R}^{n \times n}$ be a positive definite matrix (i.e., $H_k \succ 0$). The main computational component of the generic algorithm is the solution of the following QO subproblem:

$$\begin{aligned} \min_{z,d}\ & z + \tfrac{1}{2} d^T H_k d \\ \text{s.t. } & f(x_k)e + G_k^T d \leq ze. \end{aligned} \tag{2.3}$$

Here, and throughout the chapter, e denotes a vector of ones whose length is determined by the context. Alternatively, one may solve the dual of (2.3), namely

$$\begin{aligned} \max_{\pi}\ & -\tfrac{1}{2}\pi^T G_k^T W_k G_k \pi \\ \text{s.t. } & e^T \pi = 1,\ \pi \geq 0, \end{aligned} \tag{2.4}$$

where $W_k := H_k^{-1} \succ 0$. The solution (z_k, d_k, π_k) of (2.3)–(2.4) has $d_k = -W_k G_k \pi_k$.

The only other major computational component of the algorithm is a backtracking line search, performed after the computation of the search direction d_k. For this purpose, we define the sufficient decrease condition

$$f(x_k + \alpha_k d_k) \leq f(x_k) - \eta \alpha_k d_k^T H_k d_k. \tag{2.5}$$

We set $x_{k+1} \leftarrow x_k + \alpha_k d_k$ for α_k chosen to satisfy (2.5), but in order to ensure that all iterates remain within the set $\mathcal{D}$, it may be necessary to perturb such an x_{k+1}; in such cases, we make use of the perturbed line search conditions

$$\begin{aligned} f(x_{k+1}) \leq&\ f(x_k) - \eta \alpha_k d_k^T H_k d_k & \text{(2.6a)} \\ \text{and } \|x_k + \alpha_k d_k - x_{k+1}\| \leq&\ \min\{\alpha_k, \epsilon_k\}\|d_k\|. & \text{(2.6b)} \end{aligned}$$

See [39] for motivation of these line search conditions and a description of how, given α_k and d_k satisfying (2.5), an x_{k+1} satisfying (2.6) can be found in a finite number of operations.

Our algorithmic framework, AGS, is presented as Algorithm 2.1 below. In the algorithm and our subsequent analysis, we suppose that iteration k involves setting an approximate Hessian H_k and computing a search direction by (2.3). Note, however, that the algorithm can be implemented equivalently by setting an approximate inverse Hessian W_k and computing an optimal solution to (2.4). In the latter case, the search direction is obtained by setting $d_k \leftarrow -W_k G_k \pi_k$ and the quantity $d_k^T H_k d_k$ can be replaced by the equal quantity $\pi_k^T G_k^T W_k G_k \pi_k$. Thus, in either case, H_k or W_k is needed for all k, but not both.

If $\overline{p} = p \geq n + 1$ and $H_k = I$ (or $W_k = I$) for all k, then Algorithm 4 reduces to GS as proposed in [39]; specifically, it reduces to the variant involving nonnormalized search directions in §4.1 of that paper. We use AGS, therefore, to refer to instantiations of Algorithm 4 where $\overline{p} < p$ with (potentially) variable H_k. Our numerical experiments in §2.5 illustrate a variety of practical advantages of AGS over GS, while the analysis in §2.4 shows that nothing is lost in terms of convergence guarantees when $\overline{p} < p$.

Algorithm 4 Adaptive Gradient Sampling (AGS) Algorithm

```
1: (Initialization): Choose a number of sample points to generate each iteration $\overline{p} \geq 1$, number of sample points required for a full line search $p \geq n+1$, sampling radius reduction factor $\psi \in (0,1)$, number of backtracks for an incomplete line search $u \geq 0$, sufficient decrease constant $\eta \in (0,1)$, line search backtracking constant $\kappa \in (0,1)$, and stationarity tolerance parameter $\nu > 0$. Choose an initial iterate $x_0 \in \mathcal{D}$, set $X_{-1} \leftarrow \emptyset$, choose an initial sampling radius $\epsilon_0 > 0$, and set $k \leftarrow 0$.
2: (Sample set update): Set $X_k \leftarrow (X_{k-1} \cap B_k) \cup x_k \cup \overline{X}_k$, where the sample set $\overline{X}_k := \{\overline{x}_{k,1}, \dots, \overline{x}_{k,\overline{p}}\}$ is composed of $\overline{p}$ points generated uniformly in $B_k$. Set $p_k \leftarrow |X_k| - 1$. If $p_k > p$, then remove the $p_k - p$ eldest members of $X_k \backslash \{x_k\}$ and set $p_k \leftarrow p$. Compute any unknown columns of $G_k$ defined in (2.2).
3: (Hessian update): Set $H_k \succ 0$ as an approximation of the Hessian of $f$ at $x_k$.
4: (Search direction computation): Compute $(z_k, d_k)$ solving (2.3).
5: (Sampling radius update): If $\min\{\|d_k\|^2, d_k^T H_k d_k\} \leq \nu\epsilon_k^2$, then set $x_{k+1} \leftarrow x_k$, $\alpha_k \leftarrow 1$, and $\epsilon_{k+1} \leftarrow \psi\epsilon_k$ and go to step 8.
6: (Backtracking line search): If $p_k < p$, then set $\alpha_k$ as the largest value in $\{\kappa^0, \kappa^1, \dots, \kappa^u\}$ such that (2.5) is satisfied, or set $\alpha_k \leftarrow 0$ if (2.5) is not satisfied for any of these values of $\alpha_k$. If $p_k = p$, then set $\alpha_k$ as the largest value in $\{\kappa^0, \kappa^1, \kappa^2, \dots\}$ such that (2.5) is satisfied.
7: (Iterate update): Set $\epsilon_{k+1} \leftarrow \epsilon_k$. If $x_k + \alpha_k d_k \in \mathcal{D}$, then set $x_{k+1} \leftarrow x_k + \alpha_k d_k$. Otherwise, set $x_{k+1}$ as any point in $\mathcal{D}$ satisfying (2.6).
8: (Iteration increment): Set $k \leftarrow k+1$ and go to step 2.
```

2.3 Hessian Approximation Strategies

In this section, we present novel techniques for choosing H_k or W_k in the context of AGS. We refer to H_k and W_k, respectively, as approximations of the Hessian and inverse Hessian of f at x_k. These are essentially accurate descriptions for our first strategy as we employ gradient information at sample points to approximate the Hessian or inverse Hessian of f at x_k, or more generally to approximate changes in ∇f about x_k. However, the descriptions are not entirely accurate for our second strategy as in that case our intention is to form models that overestimate f, and not necessarily to have $H_k d \approx \nabla f(x_k + d) - \nabla f(x_k)$ for all small $d \in \mathbb{R}^n$. Still, for ease of exposition, it will be convenient to refer to H_k and W_k as Hessian and inverse Hessian approximations, respectively, in that context as well.

A critical motivating factor in the design of our Hessian updating strategies is the following assumption needed for our global convergence guarantees in §2.4.

Assumption 2.3.1. *There exist $\overline{\xi} \geq \underline{\xi} > 0$ such that, for all k and $d \in \mathbb{R}^n$, we have*

$$\underline{\xi}\|d\|^2 \leq d^T H_k d \leq \overline{\xi}\|d\|^2.$$

For each of our updating strategies, we show that Assumption 2.3.1 is satisfied. We remark, however, that numerical experiments have shown that for nonsmooth problems it can be beneficial to allow Hessian approximations to approach singularity [44]. Thus, our numerical experiments include forms of our updates that ensure Assumption 2.3.1 is satisfied as well as forms that do not. Either of these forms can be obtained through choices of the user-defined constants defined for each update. Note also that the bounds we provide are worst case bounds that typically would not be tight in practice.

Both of the following strategies employ gradient information — and, in the latter case, function value information — evaluated at points in the sample set X_k. At each iteration, we reinitialize the approximations $H_k \leftarrow \mu_k I$ and $W_k \leftarrow \mu_k^{-1} I$ and apply a series of updates based on information corresponding to the sample set. Note that this is different from quasi-Newton updating procedures that initialize the (inverse) Hessian approximation only at the start of the algorithm. We have found in our numerical experiments that the value μ_k is critical for the performance of the algorithm. See §5 for our approach for setting μ_k. For now, all that is required in this section is that, for some constants $\overline{\mu} \geq \underline{\mu} > 0$ and all k, we have

$$\underline{\mu} \leq \mu_k \leq \overline{\mu}. \tag{2.7}$$

Note that, for simplicity, we discuss updates for H_k and W_k as if they are both computed during iteration k. However, as mentioned in §2.2, only one of the two matrices is actually needed in each iteration of AGS.

2.3.1 LBFGS Updates on Sampled Directions

We consider an updating strategy based on the well-known BFGS formula [6, 21, 22, 55]. During iteration k, the main idea of our update is to use gradient information at the points in X_k to construct H_k or W_k. We begin by initializing $H_k \leftarrow \mu_k I$ or $W_k \leftarrow \mu_k^{-1} I$ and then perform a series of (at most) $p_k + 1 \leq p + 1$ updates based on $d_{k,i} := x_{k,i} - x_k$ and $y_{k,i} := \nabla f(x_{k,i}) - \nabla f(x_k)$ for $i = 0, \ldots, p_k$. As at most $p_k + 1$ updates are performed, this strategy is most accurately described as a LBFGS approach for setting H_k and W_k [48]. In the end, after all $p_k + 1$ updates are performed, we obtain bounds of the type required in Assumption 2.3.1 where the constants $\overline{\xi} \geq \underline{\xi} > 0$ depend only on p and user-defined constants $\gamma > 0$ and $\sigma > 0$.

Suppose that updates have been performed for sample points 0 through $i - 1$ and consider the update for sample point i. We know from step 2 of AGS that

$$\|d_{k,i}\|^2 \leq \epsilon_k^2. \tag{2.8}$$

Moreover, we will require that

$$\begin{aligned} d_{k,i}^T y_{k,i} \geq&\ \gamma\epsilon_k^2 && \text{(2.9a)} \\ \text{and}\ \ \|y_{k,i}\|^2 \leq&\ \sigma\epsilon_k^2 && \text{(2.9b)} \end{aligned}$$

for the constants $\gamma > 0$ and $\sigma > 0$ provided by the user. We skip the update for sample point i if (2.9) fails to hold. (For instance, for some i we have $x_{k,i} = x_k$, meaning that $d_{k,i} = y_{k,i} = 0$ and (2.9a) is not satisfied. Indeed, it is possible that there is no i such that (2.9) holds, in which case the overall strategy yields $H_k = \mu_k I$ or $W_k = \mu_k^{-1} I$.) For ease of exposition, however, we suppose throughout the remainder of this subsection that no updates are skipped, this assumption not invalidating our main results, Theorem 2.3.3 and Corollary 2.3.4.

The update formulas for H_k and W_k for sample point i are the following:

$$\begin{aligned} H_k &\leftarrow H_k - \frac{H_k d_{k,i} d_{k,i}^T H_k}{d_{k,i}^T H_k d_{k,i}} + \frac{y_{k,i} y_{k,i}^T}{y_{k,i}^T d_{k,i}} && \text{(2.10a)} \\ W_k &\leftarrow \left(I - \frac{y_{k,i} d_{k,i}^T}{d_{k,i}^T y_{k,i}}\right)^T W_k \left(I - \frac{y_{k,i} d_{k,i}^T}{d_{k,i}^T y_{k,i}}\right) + \frac{d_{k,i} d_{k,i}^T}{d_{k,i}^T y_{k,i}}. && \text{(2.10b)} \end{aligned}$$

The following lemma reveals bounds on inner products with H_k and W_k after the updates for sample point i has been performed.

Lemma 2.3.2. *Suppose that after updates have been performed for sample points* 0 *through* $i-1$*, we have* $H_k \succ 0$ *and* $W_k \succ 0$*, and for any* $d \in \mathbb{R}^n$ *we have* $d^T H_k d \le \theta \|d\|^2$ *and* $d^T W_k d \le \beta \|d\|^2$ *for some* $\theta > 0$ *and* $\beta > 0$*. Then, after applying* (2.10)*, we maintain* $H_k \succ 0$ *and* $W_k \succ 0$ *and have*

$$d^T H_k d \le \left(\theta + \frac{\sigma}{\gamma}\right) \|d\|^2 \tag{2.11a}$$

$$\text{and} \quad d^T W_k d \le \left(2\beta \left(1 + \frac{\sigma}{\gamma^2}\right) + \frac{1}{\gamma}\right) \|d\|^2. \tag{2.11b}$$

We now have the following theorem revealing bounds for products with H_k.

Theorem 2.3.3. *For any* k*, after all updates have been performed via* (2.10a) *for sample points* 0 *through* p_k*, the following holds for any* $d \in \mathbb{R}^n$*:*

$$d^T H_k d \ge \left(2^{p+1}\left(1 + \frac{\sigma}{\gamma^2}\right)^{p+1} \mu_k^{-1} + \frac{1}{\gamma}\left(\frac{2^{p+1}\left(1+\frac{\sigma}{\gamma^2}\right)^{p+1} - 1}{2\left(1+\frac{\sigma}{\gamma^2}\right) - 1}\right)\right)^{-1} \|d\|^2; \tag{2.12a}$$

$$d^T H_k d \le \left(\mu_k + \frac{(p+1)\sigma}{\gamma}\right) \|d\|^2. \tag{2.12b}$$

We note that the following corollary follows by applying the Rayleigh-Ritz Theorem to the result of Theorem 2.3.3.

Corollary 2.3.4. *For any* k*, after all updates have been performed via* (2.10b) *for sample points* 0 *through* p_k*, the following holds for any* $d \in \mathbb{R}^n$*:*

$$d^T W_k d \le \left(2^{p+1}\left(1 + \frac{\sigma}{\gamma^2}\right)^{p+1} \mu_k^{-1} + \frac{1}{\gamma}\left(\frac{2^{p+1}\left(1+\frac{\sigma}{\gamma^2}\right)^{p+1} - 1}{2\left(1+\frac{\sigma}{\gamma^2}\right) - 1}\right)\right) \|d\|^2; \tag{2.13a}$$

$$d^T W_k d \ge \left(\mu_k + \frac{(p+1)\sigma}{\gamma}\right)^{-1} \|d\|^2. \tag{2.13b}$$

2.3.2 Updates to Promote Model Overestimation

During iteration k, the primal subproblem (2.3) is equivalent to the following:

$$\min_{d} \ m_k(d), \quad \text{where} \ \ m_k(d) := f(x_k) + \max_{x \in X_k}\{\nabla f(x)^T d\} + \tfrac{1}{2} d^T H_k d.$$

If $m_k(d) \geq f(x_k + d)$ for all $d \in \mathbb{R}^n$, then a reduction in f is obtained after a step along $d_k \neq 0$ computed from (2.3)–(2.4). Thus, it is desirable to choose H_k so that m_k overestimates f to guarantee that such reductions occur in AGS.

It is not economical to ensure through the choice of H_k that m_k overestimates f for any given $d \in \mathbb{R}^n$. However, we can promote overestimation by evaluating $f(x_{k,i})$ at each sample point $x_{k,i} = x_k + d_{k,i}$ and performing a series of updates of H_k to increase, when appropriate, the value of $m_k(d_{k,i})$. Specifically, we set

$$H_k \leftarrow M_{k,p_k}^T \cdots M_{k,0}^T (\mu_k I) M_{k,0} \cdots M_{k,p_k} \tag{2.14}$$

where $M_{k,i}$ is chosen based on information obtained along $d_{k,i}$. (Note that such an H_k can be obtained by initializing $H_k \leftarrow \mu_k I$ and updating $H_k \leftarrow M_{k,i}^T H_k M_{k,i}$ for $i = 0, \ldots, p_k$.) We choose $M_{k,i}$ in such a way that H_k remains well-conditioned and obtain bounds of the type required in Assumption 2.3.1 where $\overline{\xi} \geq \underline{\xi} > 0$ depend only on p and a user-defined constant $\rho \geq \frac{1}{2}$.

Suppose that updates have been performed for sample points 0 through $i-1$ and consider the update for sample point i. We consider $M_{k,i}$ of the form

$$M_{k,i} = \begin{cases} I + \frac{\rho_{k,i}}{d_{k,i}^T d_{k,i}} d_{k,i} d_{k,i}^T \succ 0 & \text{if } d_{k,i} \neq 0 \\ I & \text{if } d_{k,i} = 0 \end{cases} \tag{2.15}$$

where $d_{k,i} = x_{k,i} - x_k$ is the ith sample direction and the value for $\rho_{k,i}$ depends on the relationship between $f(x_{k,i})$ and the model value

$$m_k(d_{k,i}) = f(x_k) + \max_{x \in X_k}\{\nabla f(x)^T d_{k,i}\} + \tfrac{1}{2} d_{k,i}^T H_k d_{k,i}.$$

Specifically, if $m_k(d_{k,i}) \geq f(x_{k,i})$, then we choose $\rho_{k,i} \leftarrow 0$, which by (2.15) means that $M_{k,i} \leftarrow I$. Otherwise, we set

$$\rho_{k,i} = -1 + \sqrt{\frac{2\Delta_{k,i}}{d_{k,i}^T H_k d_{k,i}}} \tag{2.16}$$

where, for the constant $\rho \geq \frac{1}{2}$ provided by the user, we set

$$\Delta_{k,i} = \min\left\{f(x_{k,i}) - m_k(d_{k,i}) + \tfrac{1}{2}d_{k,i}^T H_k d_{k,i}, \rho d_{k,i}^T H_k d_{k,i}\right\}. \tag{2.17}$$

In this latter case when $m_k(d_{k,i}) < f(x_{k,i})$, we have $\Delta_{k,i} \geq \frac{1}{2}d_{k,i}^T H_k d_{k,i}$, implying that $\rho_{k,i} \geq 0$. Moreover, as (2.17) also yields $\Delta_{k,i} \leq \rho d_{k,i}^T H_k d_{k,i}$, it follows that $\rho_{k,i} \leq \sqrt{2\rho} - 1$. Thus, $\rho_{k,i} \in [0, \sqrt{2\rho}-1]$. Notice that in the process of performing the update with $d_{k,i} \neq 0$ and $\rho_{k,i}$ set by (2.16), we have from (2.15) that

$$\begin{aligned}
\tfrac{1}{2}d_{k,i}^T H_k d_{k,i} \leftarrow\ & \tfrac{1}{2}d_{k,i}^T M_{k,i}^T H_k M_{k,i} d_{k,i} \\
=\ & \tfrac{1}{2}d_{k,i}^T \left(I + \frac{\rho_{k,i}}{d_{k,i}^T d_{k,i}} d_{k,i} d_{k,i}^T\right)^T H_k \left(I + \frac{\rho_{k,i}}{d_{k,i}^T d_{k,i}} d_{k,i} d_{k,i}^T\right) d_{k,i} \\
=\ & \tfrac{1}{2}(1+\rho_{k,i})^2 d_{k,i}^T H_k d_{k,i} \\
=\ & \Delta_{k,i}.
\end{aligned}$$

Thus, by (2.17), if $\Delta_{k,i} = f(x_{k,i}) - m_k(d_{k,i}) + \frac{1}{2}d_{k,i}^T H_k d_{k,i}$, then the model value $m_k(d_{k,i})$ has been increased to the function value $f(x_{k,i})$. Otherwise, if $\Delta_{k,i} = \rho d_{k,i}^T H_k d_{k,i}$, then the model value is still increased since $\rho \geq \frac{1}{2}$.

The following lemma reveals useful bounds for inner products with $M_{k,i}$.

Lemma 2.3.5. *Let $M_{k,i}$ be defined by* (2.15)*. Then, for any $d \in \mathbb{R}^n$, we have*

$$\|d\|^2 \leq d^T M_{k,i}^T M_{k,i} d \leq (1+\rho_{k,i})^2 \|d\|^2. \tag{2.18}$$

We then have the following theorem revealing bounds for products with H_k.

Theorem 2.3.6. *For any k, with H_k defined by* (2.14)*, $M_{k,i}$ defined by* (2.15)*, and $\rho_{k,i} \in$*

$[0, \sqrt{2\rho} - 1]$ *for* $i = 0, \ldots, p_k$*, the following holds for any* $d \in \mathbb{R}^n$*:*

$$\mu_k \|d\|^2 \leq d^T H_k d \leq \mu_k (2\rho)^{p+1} \|d\|^2. \tag{2.19}$$

The approximation $W_k = H_k^{-1}$ for the inverse Hessian corresponding to (2.14) is

$$W_k \leftarrow M_{k,p_k}^{-T} \cdots M_{k,1}^{-T} (\mu_k^{-1} I) M_{k,1}^{-1} \cdots M_{k,p_k}^{-1} \tag{2.20}$$

where the Sherman-Morrison-Woodbury formula [25] reveals that for each $i = 0, \ldots, p_k$ we have

$$M_{k,i}^{-1} = \begin{cases} I - \frac{\rho_{k,i}}{(1+\rho_{k,i}) d_{k,i}^T d_{k,i}} d_{k,i} d_{k,i}^T \succ 0 & \text{if } d_{k,i} \neq 0 \\ I & \text{if } d_{k,i} = 0. \end{cases} \tag{2.21}$$

The following corollary follows by applying the Rayleigh-Ritz Theorem to the result of Theorem 2.3.6.

Corollary 2.3.7. *For any* k*, with* W_k *defined by* (2.20)*,* $M_{k,i}^{-1}$ *defined by* (2.21)*, and* $\rho_{k,i} \in [0, \sqrt{2\rho} - 1]$ *for* $0 = 1, \ldots, p_k$*, the following holds for any* $d \in \mathbb{R}^n$*:*

$$\mu_k^{-1} (2\rho)^{-p-1} \|d\|^2 \leq d^T W_k d \leq \mu_k^{-1} \|d\|^2. \tag{2.22}$$

We conclude this subsection by showing that the updating strategy described here is intended solely for nonconvex problems. That is, if f is convex, then the updates will maintain $H_k = \mu_k I$ and $W_k = \mu_k^{-1} I$.

Theorem 2.3.8. *Suppose* f *is convex. Then, for any* k*, the matrices* H_k *and* W_k *described in Theorems 2.3.6 and 2.3.7, respectively, satisfy* $H_k = \mu_k I$ *and* $W_k = \mu_k^{-1} I$*.*

2.4 Global Convergence Analysis

We make the following assumption about the objective function f of (2.1) throughout our global convergence analysis.

Assumption 2.4.1. *The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz and continuously differentiable in an open dense subset $\mathcal{D} \subset \mathbb{R}^n$.*

We also make Assumption 2.3.1 stated previously at the beginning of §2.3.

The result we prove is the following.

Theorem 2.4.2. *AGS produces an infinite sequence of iterates $\{x_k\}$ and, with probability one, either $f(x_k) \to -\infty$ or $\{\epsilon_k\} \to 0$ and every cluster point of $\{x_k\}$ is stationary for f.*

Our analysis follows closely that of Kiwiel in [39]. However, there are subtle differences due to the adaptive sampling procedure and the variable-metric Hessian approximations. Thus, we analyze the global convergence behavior of AGS for the sake of completeness.

We begin our analysis for proving Theorem 2.4.2 by showing that AGS is well-posed in the sense that each iteration terminates finitely. It is clear that this will be true as long as the backtracking line search in step 6 terminates finitely.

Lemma 2.4.3. *If $p_k < p$ in step 6, then $\alpha_k > 0$ is computed satisfying (2.5) or $\alpha_k \leftarrow 0$. If $p_k \geq p$ in step 6, then $\alpha_k > 0$ is computed satisfying (2.5).*

Proof. If $p_k < p$ in step 6, then the statement is obviously true since only a finite number of values of α_k are considered. Next, we consider the case when $p_k \geq p$. The Karush-Kuhn-Tucker conditions of (2.3) are

$$ze - f(x_k)e - G_k^T d \geq 0 \tag{2.23a}$$

$$\pi \geq 0 \tag{2.23b}$$

$$1 - \pi^T e = 0 \tag{2.23c}$$

$$H_k d + G_k \pi = 0 \tag{2.23d}$$

$$\pi^T (ze - f(x_k)e - G_k^T d) = 0. \tag{2.23e}$$

Let (z_k, d_k, π_k) be the unique solution of (2.23). Then, (2.23c)–(2.23e) and the fact that H_k is symmetric yield

$$z_k - f(x_k) = \pi_k^T G_k^T d_k = -d_k^T H_k d_k. \tag{2.24}$$

Plugging the above equality into (2.23a), we have

$$G_k^T d_k \leq z_k e - f(x_k)e = -(d_k^T H_k d_k)e.$$

In particular, as $\nabla f(x_k)$ is a column of G_k, we have

$$\nabla f(x_k)^T d_k \leq -d_k^T H_k d_k. \tag{2.25}$$

Since by step 5 we must have $d_k^T H_k d_k > 0$ in step 6, it follows that d_k is a direction of strict descent for f at x_k, so there exists $\alpha_k > 0$ such that (2.5) holds:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \eta \alpha_k \nabla f(x_k)^T d_k \leq f(x_k) - \eta \alpha_k d_k^T H_k d_k.$$

□

Lemma 2.4.3 reveals that the line search will yield $\alpha_k \leftarrow 0$ or $\alpha_k > 0$ satisfying (2.5). Our next lemma builds on this result and shows that there will be an infinite number of iterations during which the latter situation occurs.

Lemma 2.4.4. *There exists an infinite subsequence of iterations in which $\alpha_k > 0$.*

Proof. By step (5), if $\min\{\|d_k\|^2, d_k^T H_k d_k\} \leq \nu \epsilon_k^2$ an infinite number of times, then the result follows as the algorithm sets $\alpha_k \leftarrow 1$ for such iterations. Otherwise, to derive a contradiction, suppose there exists $k' \geq 0$ such that for $k \geq k'$, step 6 is reached and sets $\alpha_k \leftarrow 0$. By Lemma 2.4.3, this means that for $k \geq k'$, we have $p_k \leq p - 1$. However, by steps 7, 8, and then 2, it is clear that if $\alpha_k \leftarrow 0$, then $p_{k+1} = \min\{p, p_k + \overline{p}\}$, contradicting the conclusion that $\{p_k\}$ is bounded above by $p - 1$ for all $k \geq k'$. □

We now show a critical result about the sequence of decreases produced in f. A similar result was proved in [39].

Lemma 2.4.5. *The following inequality holds for all k:*

$$f(x_{k+1}) \leq f(x_k) - \tfrac{1}{2}\eta\underline{\xi}\|x_{k+1} - x_k\|\|d_k\|.$$

Proof. By the triangle inequality, condition (2.6b) ensures that

$$\|x_{k+1} - x_k\| \leq \min\{\alpha_k, \epsilon_k\}\|d_k\| + \alpha_k\|d_k\| \leq 2\alpha_k\|d_k\|. \tag{2.26}$$

Indeed, this inequality holds trivially if the algorithm sets $x_{k+1} \leftarrow x_k$ in step 5 or sets $\alpha_k \leftarrow 0$ in step 6, and holds by the triangle inequality if step 6 yields $x_{k+1} \leftarrow x_k + \alpha_k d_k$. Thus, by (2.5), (2.6), and (2.26), we find that for all k,

$$\begin{aligned} f(x_{k+1}) - f(x_k) &\leq -\eta\alpha_k d_k^T H_k d_k \\ &\leq -\eta\alpha_k\underline{\xi}\|d_k\|^2 \\ &\leq -\tfrac{1}{2}\eta\underline{\xi}\|x_{k+1} - x_k\|\|d_k\|, \end{aligned}$$

as desired. □

We now consider the ability of the algorithm to approximate the set $\mathbb{G}_{\epsilon_k}(x')$ when x_k is close to a given point x'. For this purpose, consider the following subproblem:

$$\inf_d \; q(d; x', \mathbb{B}_{\epsilon_k}(x'), H_k) \tag{2.27}$$

where

$$q(d; x', \mathbb{B}_{\epsilon_k}(x'), H_k) := f(x') + \sup_{x \in \mathbb{B}_{\epsilon_k}(x') \cap \mathcal{D}} \{\nabla f(x)^T d\} + \tfrac{1}{2} d^T H_k d.$$

Given a solution d' of (2.27), we have the following reduction in its objective:

$$\Delta q(d'; x', \mathbb{B}_{\epsilon_k}(x'), H_k) := q(0; x', \mathbb{B}_{\epsilon_k}(x'), H_k) - q(d'; x', \mathbb{B}_{\epsilon_k}(x'), H_k) \geq 0.$$

Similarly, writing (2.3) in the form

$$\min_d \; q(d; x_k, X_k, H_k)$$

(see [38]), we have the following reduction produced by the search direction d_k:

$$\Delta q(d_k; x_k, X_k, H_k) = q(0; x_k, X_k, H_k) - q(d_k; x_k, X_k, H_k) \geq 0.$$

We now show a result about the above reduction.

Lemma 2.4.6. *The following equality holds:*

$$\Delta q(d_k; x_k, X_k, H_k) = \tfrac{1}{2} d_k^T H_k d_k.$$

Proof. By the definition of q, we have $q(0; x_k, X_k, H_k) = f(x_k)$. Moreover, by (2.24), we have $q(d_k; x_k, X_k, H_k) = z_k + \frac{1}{2} d_k^T H_k d_k = f(x_k) - \frac{1}{2} d_k^T H_k d_k$. Therefore, $\Delta q(d_k; x_k, X_k, H_k) = \frac{1}{2} d_k^T H_k d_k$. □

The purpose of our next lemma is to show that for any desired level of accuracy (though not necessarily perfect accuracy), as long as x_k is sufficiently close to x', there exists a sample set X_k such that the reduction $\Delta q(d_k; x_k, X_k, H_k)$ produced by the solution d_k of (2.3) will be sufficiently close to the reduction $\Delta q(d'; x', \mathbb{B}_{\epsilon_k}(x'), H_k)$ produced by the solution d' of (2.27). For a given x' and tolerance ω, we define

$$\mathcal{T}_k(x', \omega) := \left\{ X_k \in \prod_{0}^{p_k} B_k : \Delta q(d_k; x_k, X_k, H_k) \leq \Delta q(d'; x', \mathbb{B}_{\epsilon_k}(x'), H_k) + \omega \right\}.$$

This set plays a critical role in the following lemma. A similar result was proved in [39], and in the context of constrained optimization in [16].

Lemma 2.4.7. *If $p_k \geq n + 1$, then for any $\omega > 0$, there exists $\zeta > 0$ and a nonempty set $\mathcal{T}$ such that for all $x_k \in \mathbb{B}_\zeta(x')$ we have $\mathcal{T} \subset \mathcal{T}_k(x', \omega)$.*

Proof. Under Assumption 2.4.1, there exists a vector d satisfying

$$\Delta q(d; x', \mathbb{B}_{\epsilon_k}(x'), H_k) < \Delta q(d'; x', \mathbb{B}_{\epsilon_k}(x'), H_k) + \omega$$

such that for some $g \in \operatorname{conv} \nabla f(\mathbb{B}_{\epsilon_k}(x') \cap \mathcal{D})$ we have

$$q(d; x', \mathbb{B}_{\epsilon_k}(x'), H_k) = f(x') + g^T d + \tfrac{1}{2} d^T H_k d.$$

Then, since $p_k \geq n + 1$, Carathéodory's theorem [51] implies that there exists

$\{y_0, \ldots, y_{p_k}\} \subset \mathbb{B}_{\epsilon_k}(x') \cap \mathcal{D}$ and a set of nonnegative scalars $\{\lambda_0, \ldots, \lambda_{p_k}\}$ such that

$$\sum_{i=0}^{p_k} \lambda_i = 1 \qquad \text{and} \qquad \sum_{i=0}^{p_k} \lambda_i \nabla f(y_i) = g.$$

Since f is continuously differentiable in $\mathcal{D}$, there exists $\zeta \in (0, \epsilon_k)$ such that the set

$$\mathcal{T} := \prod_{i=0}^{p_k} \text{int } \mathbb{B}_\zeta(y_i)$$

lies in $\mathbb{B}_{\epsilon_k - \zeta}(x')$ and the solution d_k to (2.3) with $X_k \in \mathcal{T}$ satisfies

$$\Delta q(d_k; x_k, X_k, H_k) \leq \Delta q(d'; x', \mathbb{B}_{\epsilon_k}(x'), H_k) + \omega.$$

Thus, for all $x_k \in \mathbb{B}_\zeta(x')$, $\mathbb{B}_{\epsilon_k - \zeta}(x') \subset \mathbb{B}_{\epsilon_k}(x_k)$ and hence $\mathcal{T} \subset \mathcal{T}_k(x', \omega)$. $\square$

We are now prepared to prove Theorem 2.4.2. Our proof follows closely that of [39, Theorem 3.3]. We provide a proof for the sake of completeness and since subtle changes to the proof are required due to our adaptive sampling strategy.

Proof. If $f(x_k) \to -\infty$, then there is nothing to prove, so suppose that

$$\inf_{k \to \infty} f(x_k) > -\infty.$$

Then, we have from (2.5), (2.6), and Lemma 2.4.5 that

$$\sum_{k=0}^{\infty} \alpha_k d_k^T H_k d_k < \infty, \quad \text{and} \tag{2.28a}$$

$$\sum_{k=0}^{\infty} \|x_{k+1} - x_k\| \|d_k\| < \infty. \tag{2.28b}$$

We continue by considering two cases, the first of which has two subcases.

Case 1: Suppose that there exists $k' \geq 0$ such that $\epsilon_k = \epsilon' > 0$ for all $k \geq k'$. According to step 5, this occurs only if

$$\min\{\|d_k\|^2, d_k^T H_k d_k\} > \nu \epsilon'^2 \text{ for all } k \geq k'. \tag{2.29}$$

In conjunction with (2.28), this implies $\alpha_k \to 0$ and $x_k \to x'$ for some x'. Moreover, the fact that $\alpha_k \to 0$ implies that there exists an infinite subsequence of iterations in which $p_k = p$. Indeed, if $p_k < p$ for all large k, then since $\alpha_k \to 0$, step 6 implies that $\alpha_k \leftarrow 0$ for all large k. However, as in the proof of Lemma 2.4.4, this leads to a contradiction as we eventually find $p_k = p$ for some k. Therefore, we can define $\mathcal{K}$ as the subsequence of iterations in which $p_k = p$ and know that $\mathcal{K}$ is infinite.

Case 1a: If x' is ϵ'-stationary for f, then for any $H_k \succ 0$, the solution d' to (2.27) satisfies $\Delta q(d'; x', \mathbb{B}_{\epsilon'}(x'), H_k) = 0$. Thus, with $\omega = \nu\epsilon'^2/2$ and $(\zeta, \mathcal{T})$ chosen as in Lemma 2.4.7, there exists $k'' \geq k'$ such that $x_k \in \mathbb{B}_\zeta(x')$ for all $k \geq k''$ and

$$\tfrac{1}{2} d_k^T H_k d_k = \Delta q(d_k; x_k, X_k, H_k) \leq \tfrac{1}{2}\nu\epsilon'^2 \tag{2.30}$$

whenever $k \geq k''$, $k \in \mathcal{K}$, and $X_k \in \mathcal{T}$. Together, (2.29) and (2.30) imply that $X_k \notin \mathcal{T}$ for all $k \geq k''$ with $k \in \mathcal{K}$. However, this is a probability zero event since for all such k the set X_k continually collects points generated uniformly from B_k, meaning that it will eventually include an element of the set $\mathcal{T}$ yielding (2.30).

Case 1b: If x' is not ϵ'-stationary, then for all $k \geq k'$, any α not satisfying the sufficient decrease condition (2.5) yields

$$f(x_k + \alpha d_k) - f(x_k) > -\eta\alpha d_k^T H_k d_k,$$

and along with (2.25) yields

$$f(x_k + \alpha d_k) - f(x_k) \leq -\alpha d_k^T H_k d_k + \alpha^2 L_k \|d_k\|^2.$$

Here, L_k is a finite upper bound for $(f'(x_k + \alpha d_k) - f'(x_k))/(\alpha\|d_k\|)$ on the interval $[x_k, x_k + \alpha d_k]$ whose existence follows from Assumption 2.4.1. Combining the above inequalities yields a lower bound on any α not satisfying (2.5), which, since step 6 invokes the backtracking factor κ, yields the bound

$$\alpha_k > \kappa(1-\eta) d_k^T H_k d_k / (L_k \|d_k\|^2).$$

However, with $\omega = \Delta q(d'; x', \mathbb{B}_{\epsilon'}(x'), H_k)$ (which is strictly positive since x' is not ϵ'-stationary) and $(\zeta, \mathcal{T})$ again chosen as in Lemma 2.4.7, there exists $k'' \geq k'$ such that $x_k \in \mathbb{B}_\zeta(x')$ for all $k \geq k''$ and

$$\Delta q(d_k; x_k, X_k, H_k) \leq 2\Delta q(d'; x', \mathbb{B}_{\epsilon'}(x'), H_k)$$

whenever $k \geq k''$, $k \in \mathcal{K}$, and $X_k \in \mathcal{T}$. Under Assumptions 2.4.1 and 2.3.1 and since $x_k \to x'$, we have that for all k sufficiently large, $L_k\|d_k\|^2 \leq L$ for some constant $L > 0$, implying that for all $k \geq k''$ with $k \in \mathcal{K}$ such that $X_k \in \mathcal{T}$, α_k is bounded away from zero. Together, this and the fact that $\alpha_k \to 0$ imply that $X_k \notin \mathcal{T}$ for all $k \geq k''$ with $k \in \mathcal{K}$. Again, this is a probability zero event.

Case 2: Suppose $\{\epsilon_k\} \to 0$ and $\{x_k\}$ has a cluster point x'. First, we show that

$$\liminf_{k \to \infty} \max\{\|x_k - x'\|, \|d_k\|\} = 0. \tag{2.31}$$

If $x_k \to x'$, then by construction in the algorithm, $\{\epsilon_k\} \to 0$ if and only if there exists an infinite subsequence $\mathcal{K}'$ of iterations where

$$\min\{1, \underline{\xi}\}\|d_k\|^2 \leq \min\{\|d_k\|^2, d_k^T H_k d_k\} \leq \nu\epsilon_k^2.$$

Thus, since $\{\epsilon_k\} \to 0$, we have

$$\lim_{k \in \mathcal{K}'} \|d_k\| = 0,$$

yielding (2.31). On the other hand, if $x_k \nrightarrow x'$, then we proceed by contradiction and suppose that (2.31) does not hold. Since x' is a cluster point of $\{x_k\}$, there is an $\epsilon' > 0$ and an index $k' \geq 0$ such that the set $K' := \{k : k \geq k', \|x_k - x'\| \leq \epsilon', \|d_k\| > \epsilon'\}$ is infinite. By (2.28b), this means

$$\sum_{k \in K'} \|x_{k+1} - x_k\| < \infty. \tag{2.32}$$

Since $x_k \nrightarrow x'$, there exists an $\epsilon > 0$ such that for all $k_1 \in K'$ with $\|x_{k_1} - x'\| \leq \epsilon'/2$ there is $k_2 > k_1$ satisfying $\|x_{k_1} - x_{k_2}\| > \epsilon$ and $\|x_k - x'\| \leq \epsilon'$ for all $k_1 \leq k \leq k_2$. Thus, by the triangle inequality, we have $\epsilon < \|x_{k_1} - x_{k_2}\| \leq \sum_{k=k_1}^{k_2-1} \|x_{k+1} - x_k\|$. However, for $k_1 \in K'$ sufficiently large, (2.32) implies that the right-hand side of this inequality must be strictly less than ϵ, a contradiction.

Finally, since for all k the elements of X_k lie in B_k, equation (2.31) and $\{\epsilon_k\} \to 0$ imply that the cluster point x' is stationary for f. □

2.5 An Implementation

We have implemented Algorithm 4 in Matlab along with the QO subproblem solver described in the Appendix. In this section, we describe the algorithm variations that we have tested, the test problems that we have solved, and the results of our numerical experiments. All tests were performed on a machine running Debian 2.6.32 with two 8-Core AMD Opteron 6128 2.0GHz processors and 32GB RAM.

Despite the fact that our algorithm has been presented with the approximations $\{H_k\}$, the QO solver in the Appendix only requires $\{W_k\}$, the inverse Hessian approximations. Thus, in this section, we refer only to W_k, and not to H_k.

2.5.1 Algorithm Variations

Given varying values for the input parameters, our implementation of Algorithm 4 yields the algorithm variations described below.

- `GS`. This is a basic gradient sampling algorithm with nonnormalized search directions [39, §4.1], obtained by choosing $\overline{p} = p \geq n + 1$ with $W_k = I$ for all k. We consider this variant of GS for comparison purposes as it is the most similar with the AGS variations described below. The global convergence analysis in §2.4 applies for this algorithm as long as Assumption 2.4.1 holds.

- `AGS`. This algorithm samples gradients adaptively as we choose $\overline{p} < p$, but it does not use either Hessian updating strategy as we choose $W_k = I$ for all k. The global convergence analysis in §2.4 applies for this algorithm as long as Assumption 2.4.1 holds.

- `AGS-LBFGS`. This algorithm is an enhanced version of `AGS` where the updating strategy in §2.3.1 is used to set W_k for all k. We choose $\gamma = 0.1$ and $\sigma = 100$ in the updates, which by Corollary 2.3.4 means that the global convergence analysis in §2.4 applies as long as Assumption 2.4.1 holds.

- `AGS-LBFGS-ill`. This algorithm is similar to `AGS-LBFGS`, except that we choose $\gamma = 0$ and $\sigma = \infty$ so that W_k may become ill-conditioned. The global convergence analysis in §2.4 does *not* apply for this method.

- `AGS-over`. This algorithm is an enhanced version of `AGS` where the updating strategy in §2.3.2 is used to set W_k for all k. We choose $\rho = 100$ in the updates, which by Corollary 2.3.7 means that the global convergence analysis in §2.4 applies as long as Assumption 2.4.1 holds.

- `AGS-over-ill`. This algorithm is similar to `AGS-over`, except that we choose $\rho = \infty$ so that W_k may become ill-conditioned. The global convergence analysis in §2.4 does *not* apply for this method.

We summarize the differing inputs for these six algorithm variations in Table 2.1.

Name	Samples per Iteration	Hessian updates	γ	σ	ρ
`GS`	$\overline{p} = p \geq n+1$	None	-	-	-
`AGS`	$\overline{p} < p \geq n+1$	None	-	-	-
`AGS-LBFGS`	$\overline{p} < p \geq n+1$	Strategy in §2.3.1	0.1	100	-
`AGS-LBFGS-ill`	$\overline{p} < p \geq n+1$	Strategy in §2.3.1	0	∞	-
`AGS-over`	$\overline{p} < p \geq n+1$	Strategy in §2.3.2	-	-	100
`AGS-over-ill`	$\overline{p} < p \geq n+1$	Strategy in §2.3.2	-	-	∞

Table 2.1: Summary of six algorithm variations used to test the adaptive sampling procedure in Algorithm 4 along with the Hessian approximation updating strategies described in §2.3.1 and §2.3.2.

Specific values for the input parameters mentioned above, as well as for the remaining parameters that were set consistently for all algorithm variations, were chosen as those that yielded the best overall results in our experiments. As recommended in [9, 39], we choose $p = 2n$ as the number of sample points required for a complete line search. (Note that this is also the number of sample points and sample gradients computed per iteration for `GS`.) For `AGS` and the remaining variants, we experimented with various values for $\overline{p}$, eventually finding that $\overline{p} = n/10$ yielded nice results. Our convergence analysis in §2.4 requires only $O(1)$ gradients per iteration, but we suggest that setting $\overline{p}$ as a fraction of n may generally yield a good balance between overall gradient evaluations and search direction quality. We set the line search backtracking constant to be $\kappa = 0.5$, sufficient decrease constant to be $\eta = 10^{-8}$, and number of backtracks in an incomplete line search to be $u = 7$. The initial sampling radius is chosen to be $\epsilon_0 = 0.1$ and $\psi = 0.1$ is set as the sampling radius reduction factor. We choose the stationarity tolerance parameter to be $\nu = 10$ and limit the number of gradient evaluations to $100n$ before terminating the algorithm.

The inverse Hessian approximations are initialized during iteration k as $W_k = \mu_k^{-1} I$. The scalar value μ_k itself is initialized at the start of a run of the algorithm as $\mu_0 = 1$ and is updated dynamically at the end of each iteration k based on the steplength α_k. Specifically, we set

$$\mu_{k+1} \leftarrow \begin{cases} \min\{2\mu_k, \overline{\mu}\} & \text{if } \alpha_k < 1 \\ \max\{\frac{1}{2}\mu_k, \underline{\mu}\} & \text{if } \alpha_k = 1 \end{cases}$$

where we choose $\underline{\mu} = 10^{-2}$ and $\overline{\mu} = 10^3$. This strategy decreases the eigenvalues of the initialized inverse Hessian if, during the current iteration, the line search had to backtrack from $\alpha_k = 1$, thus promoting a shorter search direction in iteration $k+1$. Similarly, if the current iteration yielded $\alpha_k = 1$, then the eigenvalues of the initialized inverse Hessian are increased to promote a longer search direction in iteration $k+1$.

We implemented Algorithm 4 along with the QO subproblem solver described in the Appendix. We set the subproblem optimality tolerance to 10^{-10} and maximum number of iterations to $\min\{1000, 2^{\max\{n, p_k\}}\}$.

2.5.2 Test Problems

We tested the algorithm variations with 26 nonsmooth minimization problems, some convex and some nonconvex. The first 20 of these problems were considered in [30] and the last 6 were considered in [56]. All problems are scalable in the sense that they can be defined to have different numbers of variables n. The first 10 problems, introduced in [31], are all nonsmooth at their respective minimizers: `MAXQ`, `MXHILB`, `CHAINED_LQ`, `CHAINED_CB3_I`, `CHAINED_CB3_II`, `ACTIVE_FACES`, `BROWN_FUNCTION_2`, `CHAINED_MIFFLIN_2`, `CHAINED_CRESCENT_I`, and `CHAINED_CRESCENT_II`. The first 5 of these problems are convex and the second 5 are nonconvex. The second 10 problems in our set, some of which are nonconvex, were introduced in the test library TEST29 [45]: `TEST29_2`, `TEST29_5`, `TEST29_6`, `TEST29_11`, `TEST29_13`, `TEST29_17`, `TEST29_19`, `TEST29_20`, `TEST29_22`, and `TEST29_24`. Of the 6 remaining problems, the first four were introduced in [43], the fifth was introduced in [26], and the sixth is a problem to minimize the Schatten norm [56]: `TILTED_NORM_COND`, `CPSF`, `NCPSF`, `EIG_PROD`, `GREIF_FUN`, and `NUC_NORM`.

2.5.3 Numerical Results

We chose $n = 50$ for all problems. The only exception was `EIG_PROD`, for which we choose $n = 64$, as the variables for this problem need to compose a square matrix. We ran each problem 10 times, the first time using a fixed initial point x_0' and the remaining nine times using a starting point generated randomly from a ball about x_0' with radius $\|x_0'\|$. (We choose $x_0' \neq 0$ for all problems, so the initial points for each run were unique.) For the first 20 problems, we choose x_0' as the initial point defined in [30]. For the remaining 6 problems, we choose $x_0' = e$. The input parameters we use for `TILTED_NORM_COND`, `CPSF`, `NCPSF`, and `NUC_NORM` are those used in [56]. The only remaining problem inputs that require specification are the matrices involved in `EIG_PROD` and `GREIF_FUN`. For the former we used the leading 8×8 submatrix of A from [1] and for the latter we used a randomly generated 10×10 symmetric positive definite matrix A (with the $n = 50$ variables composing a 10×5 matrix X so that the product $X^T AX$ is well defined).

The performance measures we considered were the final sampling radius and the average QO iterations per nonlinear iteration when the limit on gradient evaluations ($100n$) was reached. The first measure shows the progress toward optimality that the solver makes within a fixed gradient evaluation limit, and the second shows the benefit (or lack thereof in the case of GS) of warm-starting the QO solver. We put a lower bound of 10^{-12} on the final sampling radius ϵ. Thus, the performance profiles below are for $\log_{10}\max\{\epsilon, 10^{-12}\} + 13$ whose values lie in $\{1, 2, \ldots, 12\}$.

First, we compare the results obtained by applying the algorithms `GS` and `AGS` to the $26 \times 10 = 260$ test problems. Performance profiles [19] for the final sampling radius and average QO iterations are given in Figure 2.1. The profiles clearly illustrate the benefits of `AGS` over `GS`. Given the same limit on the number of gradient evaluations, `AGS` is able to perform many more nonlinear iterations than `GS` due to the fact that `AGS` requires many fewer gradient evaluations per nonlinear iteration. This allows `AGS` to make much more progress toward the solution, as evidenced by the final sampling radius consistently being much smaller.

One additional remark to make about the performance profiles in Figure 2.1 is that the number of average QO iterations is significantly fewer for `AGS` as compared to `GS`. This can be attributed to the fact that the subproblems in `AGS` are often smaller than those in `GS`, and when they are the same size as in `GS`, warm-starting the solver reduces the number of QO iterations required.

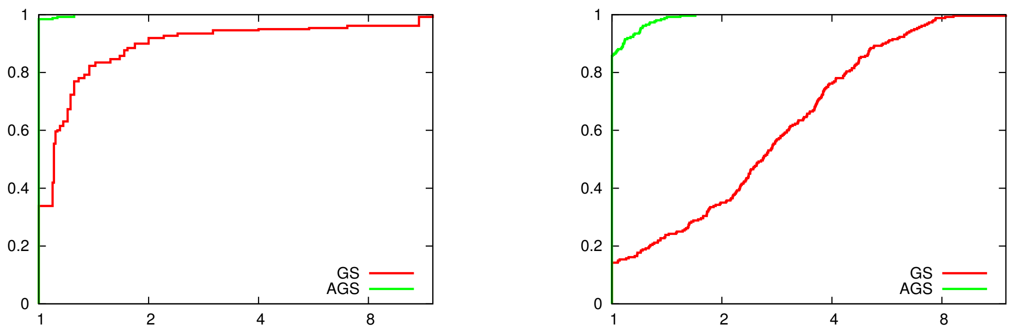


Figure 2.1: Performance profiles for the final sampling radius (left) and average QO iterations per nonlinear iteration (right) comparing algorithms `GS` and `AGS`.

Our second set of performance profiles illustrate the benefits of the Hessian updating strategies in §2.3 by comparing the results for `AGS-LBFGS` and `AGS-over` with those for

AGS. AGS-over performs better than AGS-LBFGS in terms of the final sampling radius, while it is dominated by AGS-LBFGS in terms of average QO iterations. Both AGS-over and AGS-LBFGS perform better than AGS no matter which performance measure is considered. We did not expect to see a reducton in average QO iterations when the inverse Hessian updates are employed, but in any case our experiments reveal that the updates bring benefits in terms of progress toward a minimizer.

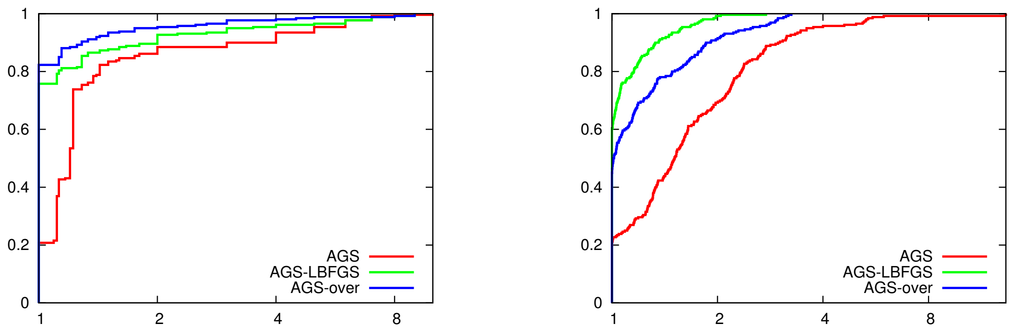
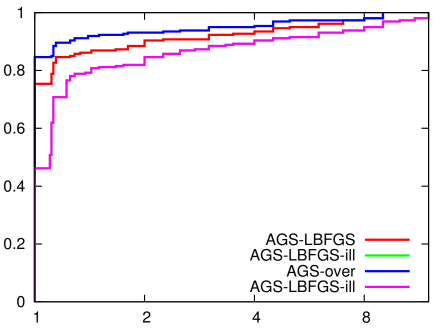


Figure 2.2: Performance profiles for the final sampling radius (left) and average QO iterations per nonlinear iteration (right) comparing algorithms AGS, AGS-LBFGS, and AGS-over.

We close this section with performance profiles comparing AGS-over and AGS-LBFGS with AGS-over-ill and AGS-LBFGS-ill, the latter two being variants for which our global convergence analysis in §2.4 does not apply. Despite the fact that in some situations it is believed that allowing Hessian approximations to tend to singularity can be beneficial [44], we do not see much of an impact in our numerical results. (In fact, there appears to be a disadvantage in terms of the final sampling radius when allowing ill-conditioning of AGS-LBFGS-ill.) There are at least a couple possible explanations for this phenomenon. First, due to the fact that we reinitialize W_k during each iteration and perform only a finite number of updates based on sample point information, our Hessian approximations may naturally remain better conditioned than those obtained by standard quasi-Newton updating techniques that continually build these matrices based on gradient information obtained at all previous algorithm iterates. Second, our input parameter choices may be generous enough that, e.g., AGS-over and AGS-over-ill produce similar Hessian approximations during most iterations.



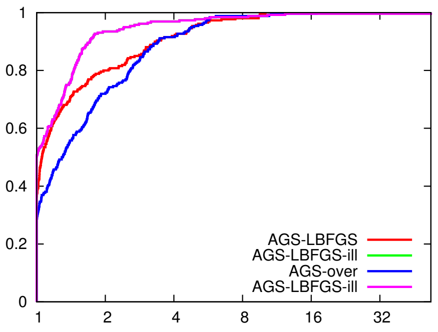


Figure 2.3: Performance profiles for the final sampling radius (left) and average QO iterations per nonlinear iteration (right) comparing algorithms `AGS-LBFGS`, `AGS-LBFGS-ill`, `AGS-over` and `AGS-over-ill`.

2.6 Conclusion

In this chapter, we have addressed major practical limitations of the rapidly-developing class of gradient sampling (GS) algorithms for nonsmooth optimization. Our proposed enhancements that attempt to correct for these limitations of GS take the form of an adaptive sampling procedure and variable-metric Hessian updating strategies. We have shown that our enhanced framework, AGS, maintains the global convergence guarantees of GS while providing many practical advantages. These advantages have been illustrated via numerical experiments on a diverse set of test problems without requiring tailored inputs for each test problem.

In addition to representing an enhanced version of GS, we believe that the development of AGS represents a step toward merging the algorithmic frameworks of gradient sampling and bundle methods. Indeed, by incorporating information obtained during previous iterations, the subproblems formed and solved in AGS closely resemble those typically found in bundle methods (after a “descent” or “serious” step has been made). We intend to investigate the marriage of gradient sampling and bundle method strategies in our future work.

Finally, we remark that there are interesting similarities between AGS and a forerunner of bundle methods, namely Wolfe’s conjugate subgradient method [61]. Wolfe’s method, a reasonably effective method for nondifferentiable convex optimization, is an extension of the conjugate gradient algorithm for minimizing differentiable functions. It is similar to

AGS in that the central idea behind both algorithms is to approximate the subdifferential at (or near) a nondifferentiable point via (sub)gradients at nearby points. In particular, both algorithms compute search directions by finding the minimum norm vector in the convex hull of (sub)gradients evaluated at these nearby points. A major difference, however, is that the (sub)gradients in Wolfe's method are the (sub)gradients and search directions obtained at previous iterates, whereas AGS employs random sampling and does not utilize previous search directions in place of gradients. Another important difference between AGS and Wolfe's method is that the latter has guarantees only for convex objective functions, whereas the former can also solve nonconvex problems. Still, despite these differences, we plan to investigate whether borrowing ideas from Wolfe's method, namely that of including previous search directions in the collection of sampled gradients, can help enhance the practical performance of AGS methods.

Chapter 3

A BFGS Gradient Sampling Algorithm

We present a line search algorithm for minimizing nonconvex and/or nonsmooth objective functions. The algorithm is a hybrid between a standard Broyden-Fletcher-Goldfarb-Shanno (BFGS) and an adaptive gradient sampling (GS) method. The BFGS strategy is employed as it typically yields fast convergence to the vicinity of a stationary point, and along with the adaptive GS strategy the algorithm ensures that convergence will continue to such a point. Under suitable assumptions, we prove that the algorithm converges globally with probability one. The algorithm has been implemented in C++ and the results of numerical experiments are presented to illustrate the efficacy of the proposed numerical method. Compared to the AGS algorithm proposed in the previous chapter, the BFGS-GS algorithm is even faster because it behaves like a BFGS method most of the time; and nothing is lost in terms of convergence guarantees.

3.1 Introduction

Our algorithm in this chapter is based on the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [6, 21, 22, 55]. Since its inception, this approach—arguably the most effective quasi-Newton method [49]—has been extremely popular for solving smooth optimization problems. This popularity stems from the fact that the method only requires first-order derivatives of the objective function, and yet can achieve a superlinear rate of local convergence. Moreover, many have witnessed good performance of BFGS when solving nonsmooth problems [30, 31], despite the fact that global convergence guarantees for the algorithm in this context are rather limited [44]. In order to overcome this theoretical deficiency, our algorithm enhances BFGS with an adaptive gradient sampling (GS) strategy adopted from the method in [18]. With this enhancement, as well as other practical features, we have designed an algorithm that exhibits good practical behavior, and for which we have established global convergence guarantees under suitable assumptions.

A feature critical to the practical performance of our algorithm is that, when it is applied to solve many problem instances, the algorithm reduces to an unadulterated BFGS strategy for the majority of the iterations. This feature is intentional, and is motivated by the encouraging results presented in [44]. Indeed, a straightforward BFGS algorithm applied to solve a nonsmooth, nonconvex optimization problem is often very effective in making progress toward a solution. However, it suffers from two important drawbacks: (*i*) it does not inherently offer termination conditions related to a stationarity measure that can theoretically be guaranteed to eventually be satisfied, meaning that there is no certain way of determining whether a solution has been reached, and (*ii*) guaranteeing global convergence appears to be difficult in general due to the fact that the (inverse) Hessian approximations may tend to singularity in the neighborhood of any solution point at which the objective function is not differentiable. (To address issue (*i*), the method in [44] employs a termination condition using a stationarity measure that is similar to the one we use in this paper for solution quality; see §3.4.3 for further discussion.) Overall, these deficiencies suggest that while BFGS may be able to converge to a neighborhood of a solution, enhancements—such as our adaptive GS procedure—may be needed to obtain high accuracy and provide the means to guarantee a certificate of stationarity (even though, in some cases, a straightforward BFGS approach can achieve a reasonable certificate of stationarity in practice).

The GS algorithm was introduced by Burke, Lewis, and Overton in [10]. Employing a strategy of randomly sampling gradients to approximate the ϵ-subdifferential of the objective about each iterate [8], the algorithm was motivated as a strategy for establishing global convergence guarantees when solving nonconvex, locally Lipschitz optimization problems. Enhancements to the algorithm have also been established over the past few years, both to improve the theoretical and practical behavior of the algorithm [18, 39] and extend the methodology to broader classes of problems [17, 32, 33, 40]. The main disadvantage of the algorithm, however, is that each iteration is significantly more expensive than that of an algorithm such as a BFGS method. Moreover, the algorithm in [10] does not employ variable-metric (inverse) Hessian approximations, and thus it may fail to fully capture the curvature information that makes an algorithm such as BFGS so effective. These disadvantages motivated the enhancements proposed in [18], though a drawback of the algorithm in that paper is that each iteration requires the sampling of gradient information in every iteration (along with a gradient evaluation at each iterate).

In summation, our proposed BFGS-GS algorithm possesses theoretical and practical advantages. It typically behaves as an unadulterated BFGS algorithm, and thus often converges to a neighborhood of a solution with a computational effort on the order of one gradient evaluation and one matrix-vector product per iteration. Throughout, the algorithm dynamically employs an adaptive GS strategy in order to provide a practical stationarity certificate as well as global convergence guarantees. Careful attention has been paid to the design of our line search, sample set update, and (inverse) Hessian approximation subroutines so that the algorithm attains this desirable behavior. For example, in certain situations, we replace a BFGS (inverse) Hessian approximation with a carefully constructed limited memory BFGS (L-BFGS [48]) approximation to ensure positive definiteness and boundedness. We have also implemented the algorithm in C++ and performed a variety of experiments to illustrate the efficacy of our proposed numerical method.

The remainder of the paper is organized into a few sections. In §3.2, we present our main algorithm, including its relevant subroutines for the line search, sample set update, and (inverse) Hessian approximation strategies. We then analyze the well-posedness and global convergence properties of the algorithm in §3.3, building on results proved during the algorithmic development in §3.2. An implementation of our algorithmic framework and the results of numerical experiments on a set of test problems is the subject of §3.4. Concluding remarks are provided in §3.5.

Notation and definitions

The sets of n-dimensional real, natural, and positive natural numbers are denoted by $\mathbb{R}^n$, $\mathbb{N}^n$, and $\mathbb{N}_+^n$, respectively, where $\mathbb{N} := \{0, 1, 2, \ldots\}$ and $\mathbb{N}_+ := \{1, 2, \ldots\}$. The ith element of a vector $x \in \mathbb{R}^n$ is written as x^i. We denote the closure and convex hull of a subset $S \subseteq \mathbb{R}^n$ as $\operatorname{cl} S$ and $\operatorname{conv} S$, respectively. The closed Euclidean ϵ-ball about $x \in \mathbb{R}^n$ is denoted as $\mathbb{B}_\epsilon(x) := \{\overline{x} \in \mathbb{R}^n : \|\overline{x} - x\|_2 \leq \epsilon\}$. The cardinality of a finite subset $S \subset \mathbb{R}^n$ is written as $|S| \in \mathbb{N}$. For a matrix W, we write $W \succ 0$ to indicate that W is real, symmetric, and positive definite. Given $W \succ 0$ and $x \in \mathbb{R}^n$, we define the "W-norm" of x as $\|x\|_W := \|W^{1/2}x\|_2$ so that $\|x\|_W^2 = x^T W x$. Given $W \succ 0$ and nonempty bounded $S \subseteq \mathbb{R}^n$, we define the (oblique) "W-projection" of the origin onto $\operatorname{cl}\operatorname{conv} S$ as $P_W(S)$, which is the unique solution of $\min_x \|x\|_W^2$ subject to $x \in \operatorname{cl}\operatorname{conv} S$. The quantities e and I respectively represent a vector of ones and an identity matrix whose sizes are determined by the context in which each quantity appears. For $\{a, b\} \subset \mathbb{R}^n$, we write $a \perp b$ to indicate that a and b are complementary, i.e., that $a^i b^i = 0$ for all $i \in \{1, \ldots, n\}$. We use a subscript for a quantity to denote the iteration number of an algorithm to which it corresponds; e.g., the value for a vector x in the kth iteration of an algorithm is written as x_k. If the limit of a sequence $\{a_k\}$ as $k \to \infty$ exists and equals a, then we write $\{a_k\} \to a$.

For a function $f : \mathbb{R}^n \to \mathbb{R}$, the sublevel set corresponding to a point $x \in \mathbb{R}^n$ is written as $\mathcal{L}_f(x) := \{\overline{x} \in \mathbb{R}^n : f(\overline{x}) \leq f(x)\}$. Such a function is locally Lipschitz over $\mathbb{R}^n$ if for every compact subset $S \subset \mathbb{R}^n$, there exists a constant $L_S \geq 0$ such that $|f(x) - f(y)| \leq L_S\|x - y\|_2$ for any $\{x, y\} \subseteq S$. If f is locally Lipschitz on $\mathbb{R}^n$, then the Clarke subdifferential [15] of f at x can be written as

$$\partial f(x) := \bigcap_{\epsilon > 0} \operatorname{cl}\operatorname{conv} \nabla f(\mathbb{B}_\epsilon(x) \cap \mathcal{D}),$$

and the Clarke ϵ-subdifferential [24] of f at x is

$$\partial_\epsilon f(x) := \operatorname{cl}\operatorname{conv} \partial f(\mathbb{B}_\epsilon(x)).$$

For such a function, a point $x \in \mathbb{R}^n$ is Clarke stationary if $0 \in \partial f(x)$, and is Clarke ϵ-stationary if $0 \in \partial_\epsilon f(x)$. For the sake of brevity, hereafter we drop the distinction "Clarke" from all of the terms defined here.

3.2 Algorithm Description

Consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \; f(x), \tag{3.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ satisfies the following assumption.

Assumption 3.2.1. *The objective function $f : \mathbb{R}^n \to \mathbb{R}$ of problem* (3.1) *is locally Lipschitz over $\mathbb{R}^n$ and continuously differentiable in an open, dense subset $\mathcal{D}$ of $\mathbb{R}^n$.*

While our convergence analysis of our algorithm rely on the properties of f assured under Assumption 3.2.1, we also believe that our algorithm could be a viable alternative to other approaches when f is not locally Lipschitz.

Given an initial iterate $x_0 \in \mathcal{D}$, our desire is to compute a solution of (3.1). However, since f may be nonconvex and/or nonsmooth, our algorithm is designed simply to locate a stationary point for f in the sublevel set $\mathcal{L}_f(x_0)$. More precisely, it is designed to compute a sequence of (approximately) ϵ_k-stationary points for a sequence $\{\epsilon_k\} \to 0$ that is set dynamically within the algorithm.

We present our algorithm in four subsections. The first subsection describes the main algorithm, at the heart of which is the search direction computation. We then discuss, in turn, the details of our line search, sample set generation scheme, and (inverse) Hessian approximation strategy. (As f may be nonsmooth, we use the term "Hessian" loosely as a matrix that approximates changes in ∇f about a given point in $\mathcal{D}$, changes that may be arbitrarily large relative to the distance between the given point and nearby points in $\mathcal{D}$.) Each of these latter algorithmic components are carefully constructed so that the main algorithm is well-posed and globally convergent to a stationary point (or points) of f under Assumption 3.2.1.

Our algorithm employs various user-specified parameters, which, for convenience, we enumerate upfront in Table 3.1. Our global convergence theory allows for any choices of these parameters in the given ranges, except for a restriction on the curvature threshold ξ and its relationship to other parameter values. This restriction, which is required due to a technical lemma revealed in the development of our algorithm, is given at the beginning of §3.3.

Table 3.1: User-specified constants for the proposed algorithm and subroutines

Parameter(s)	Range	Description
ν	$(0, \infty)$	Stationarity measure tolerance
ψ	$(0, 1)$	Sampling radius reduction factor
ξ	$(0, \infty)$	Model curvature threshold
$\underline{\eta} < \overline{\eta}$	$(0, 1)$	Armijo–Wolfe line search constants
$\underline{\alpha} \leq \overline{\alpha}$	$(0, \infty)$	Step size thresholds
γ	$(0, 1)$	Step size modification factor
$\underline{J} \leq \overline{J}$	$\mathbb{N}$	Iteration thresholds for line search
J	$\mathbb{N}$	Iteration threshold for iterate perturbation
p	$[n+1, \infty) \cap \mathbb{N}$	Sample set size threshold
$\underline{\mu} < 1 < \overline{\mu}$	$(0, \infty)$	(L-)BFGS updating thresholds
$\underline{w} \leq \overline{w}$	$(0, \infty)$	(L-)BFGS updating thresholds
m	$\mathbb{N}$	L-BFGS memory length

3.2.1 Main Algorithm

We now present our main algorithm, designed to converge to a stationary point of f in the sublevel set $\mathcal{L}_f(x_0)$. Ideally, such a point would be revealed as a cluster point of the iterate sequence $\{x_k\}$ obtained via a standard BFGS method, but since such a method generally has unknown convergence properties when employed to solve (3.1), our algorithm includes enhancements with which we guarantee global convergence with probability one. These enhancements are similar to those developed in the adaptive GS method proposed in [18], though are less expensive in the sense that, in many iterations, gradient sampling is not required.

At a given iterate $x_k \in \mathcal{D}$ and with a given inverse Hessian approximation of f at x_k, call it $W_k \succ 0$, a standard BFGS method computes a search direction as

$$d_k \leftarrow -W_k \nabla f(x_k). \tag{3.2}$$

However, in our approach, we incorporate gradient information at points in a set $X_k := \{x_{k,0}, \dots, x_{k,p_k}\}$ that has $x_{k,i} = x_k$ for some $i \in \{0, \dots, p_k\}$ and includes p_k other points from $B_k := \mathbb{B}_{\epsilon_k}(x_k) \cap \mathcal{D}$. (We refer to X_k as the sample set and p_k as the sample set size, even though $p_k = 0$ corresponds to $|X_k| = 1$.) With this information, we desire the search direction d_k that is the minimizer of a local piece-wise quadratic model of f at x_k; i.e., we desire d_k to be the solution of

$$\min_{d \in \mathbb{R}^n} \ q_k(d), \quad \text{where} \quad q_k(d) := \left(\max_{x \in X_k} \left\{ \nabla f(x)^T d \right\} + \tfrac{1}{2} \|d\|^2_{W_k^{-1}} \right). \tag{3.3}$$

Defining the matrix of gradients

$$G_k := \begin{bmatrix} g_{k,0} & \cdots & g_{k,p_k} \end{bmatrix} \text{ with } g_{k,i} := \nabla f(x_{k,i}) \text{ for all } i \in \{0, \dots, p_k\}, \tag{3.4}$$

the solution d_k of (3.3) can be obtained by solving the primal-dual pair

$$\left\{ \begin{array}{c} \min\limits_{(z,d) \in \mathbb{R}^{n+1}} \ z + \tfrac{1}{2}\|d\|^2_{W_k^{-1}} \\ \text{s.t. } G_k^T d \le ze \end{array} \right\} \qquad \left\{ \begin{array}{c} \max\limits_{y \in \mathbb{R}^{p_k+1}} \ -\tfrac{1}{2}\|G_k y\|^2_{W_k} \\ \text{s.t. } e^T y = 1, \ y \ge 0 \end{array} \right\}, \tag{3.5}$$

the primal-dual solution of which we denote as (z_k, d_k, y_k).

If the sample set has only one element, i.e., if $p_k = 0$ with $X_k = \{x_{k,0}\} = \{x_k\}$, then it is easily seen that our notation is consistent in that d_k from (3.5) is that in (3.2). Thus, henceforth we may refer to d_k as the solution of (3.3), knowing that if $p_k = 0$, then it can be obtained directly from (3.2), and otherwise it can be obtained by solving the former (i.e., primal) quadratic optimization subproblem (QP) in (3.5). In fact, if instead one solves the latter (i.e., dual) QP in (3.5) to obtain y_k, then the search direction can be obtained as $d_k \leftarrow -W_k G_k y_k$. This is the approach that we take in our implementation described in §3.4, and so it will be the approach used in the remainder of our discussion and analysis. Note that a benefit of this strategy is that the Hessian approximation W_k^{-1} need not be computed; i.e., we only need the matrix W_k appearing in (3.2) and (3.5) as all subsequent computations will be written in such a way that W_k^{-1} is not explicitly needed.

Overall, there are two interpretations of the search direction d_k. First, it can be viewed, as in subproblem (3.3), as the minimizer of a local piece-wise quadratic model of f at x_k with gradient information sampled at the points in X_k. Second, it can be viewed, in terms of the dual QP in (3.5), as W_k times the negation of the oblique W_k-projection of the origin onto the convex hull of the gradients of f at the points in the sample set X_k, i.e., as $d_k = -W_k P_{W_k}(\{\nabla f(x)\}_{x \in X_k})$, which is to say that it is W_k times the negation of the minimum W_k-norm element in $\text{conv}\{\nabla f(x)\}_{x \in X_k}$. (Clearly, with $W_k = I$, the search direction reduces to the negation of the minimum Euclidean norm element in $\text{conv}\{\nabla f(x)\}_{x \in X_k}$, which is precisely the "nonnormalized search direction" interpretation described in [39, §4.1].) The former interpretation is perhaps more intuitively appealing as that of a search direction for an optimization algorithm, though we will make more use of the second interpretation in our global convergence analysis.

Once the pair (d_k, y_k) has been computed via (3.5), we either compute a null step size (to produce a null step, which may be necessary in some cases) or a positive step size $\alpha_k > 0$ such that the trial point $x_k + \alpha_k d_k$ yields a sufficiently lower objective value than that yielded by x_k. If $x_k + \alpha_k d_k \in \mathcal{D}$, then the next iterate x_{k+1} is set to be this trial point; otherwise a point $x_{k+1} \in \mathcal{D}$ in the vicinity of $x_k + \alpha_k d_k$ is computed that also yields sufficient decrease in f. (In fact, the step size α_k and new iterate x_{k+1} may be chosen also to satisfy a curvature condition to ensure that an unadulterated BFGS update will produce a positive definite Hessian approximation in the following iteration.) All of the details of these procedures are given in §3.2.2. Overall, starting with $x_0 \in \mathcal{D}$, we ensure that $\{x_k\} \subset \mathcal{D}$.

Once the pair (d_k, y_k), step size $\alpha_k \geq 0$, and next iterate $x_{k+1} \in \mathcal{D}$ have been computed, the remainder of the iteration involves setting the next sampling radius $\epsilon_{k+1} \in (0, \epsilon_k]$, sample set X_{k+1} (and related quantities), and inverse Hessian approximation W_{k+1}. In particular, the value to which the next sampling radius ϵ_{k+1} is set depends on whether or not the following inequalities hold:

$$\|G_k y_k\|_{W_k} \leq \nu\epsilon_k; \tag{3.6a}$$

$$\|G_k y_k\|_{W_k} \geq \xi\|d_k\|_2; \tag{3.6b}$$

$$\alpha_k > 0. \tag{3.6c}$$

The details pertaining to the updates of sample set and (inverse) Hessian approximation are the subjects of §3.2.3 and §3.2.4, respectively.

We now present our main algorithm, stated as Algorithm 5.

Algorithm 5 BFGS Gradient Sampling Algorithm

```
 1: Choose an initial iterate $x_0 \in \mathcal{D}$, inverse Hessian approximation $W_0 \succ 0$, and sampling
    radius $\epsilon_0 > 0$. Set the initial sample set $X_0 \leftarrow \{x_0\}$, sample set size $p_0 \leftarrow 0$, matrix of
    sample gradients $G_0$ as defined in (3.4), and iteration counter $k \leftarrow 0$.
 2: If $\nabla f(x_k) = 0$, then terminate and return the stationary point $x_k$.
 3: Compute a search direction $d_k \leftarrow -W_k G_k y_k$ where $y_k$ solves the dual QP in (3.5).
 4: Compute a step size $\alpha_k \geq 0$ via Algorithm 6 in §3.2.2.
 5: Compute a new iterate $x_{k+1} \in \mathcal{D}$ via Algorithm 7 in §3.2.2.
 6: If (3.6) holds, then set the new sampling radius $\epsilon_{k+1} \leftarrow \psi\epsilon_k$; otherwise, set $\epsilon_{k+1} \leftarrow \epsilon_k$.
 7: Compute a new sample set $X_{k+1}$ with $p_{k+1} \leftarrow |X_{k+1}| - 1$ via Algorithm 8 in §3.2.3.
 8: Compute the matrix of gradients $G_{k+1}$ as defined in (3.4).
 9: Compute a new inverse Hessian approximation $W_{k+1} \succ 0$ via Algorithm 9 in §3.2.4.
10: Set $k \leftarrow k + 1$ and go to Step 2.
```

We close this subsection by showing that if Algorithm 5 reaches Step 3 during iteration k, then it computes d_k as null or as a direction of strict descent for f from $x_k \in \mathcal{D}$. We state this result, which also proves an important relationship between the search direction d_k and the dual QP solution y_k, as it will be used to motivate algorithmic choices made in the following subsections. We provide a proof of this result for the sake of completeness; see also [18, Lemma 4.3].

Lemma 3.2.2. *If Algorithm 5 reaches Step 3 during iteration k, then it computes a search direction d_k that is zero or a direction of strict descent for f from $x_k \in \mathcal{D}$. In addition, the primal-dual solution (z_k, d_k, y_k) of (3.5) satisfies $\|G_k y_k\|_{W_k} = \|d_k\|_{W_k^{-1}}$.*

Proof. The primal and dual subproblems in (3.5) are both feasible. Moreover, since $W_k \succ 0$, they are strictly convex, so d_k and y_k are part of the unique tuple (z_k, d_k, y_k) satisfying the Karush-Kuhn-Tucker (KKT) conditions for (3.5):

$$d_k + W_k G_k y_k = 0; \tag{3.7a}$$

$$e^T y_k - 1 = 0; \tag{3.7b}$$

$$0 \leq y_k \perp z_k e - G_k^T d_k \geq 0. \tag{3.7c}$$

Equation (3.7a) implies $\|G_k y_k\|_{W_k} = \|d_k\|_{W_k^{-1}}$. Moreover, observing (3.7), we find

$$0 = y_k^T(z_k e - G_k^T d_k) \implies z_k = z_k e^T y_k = y_k^T G_k^T d_k = -\|G_k y_k\|_{W_k}^2,$$

which, along with the fact that (3.7c) states $z_k e - G_k^T d_k \geq 0$, implies that

$$G_k^T d_k \leq z_k e \leq -(\|G_k y_k\|_{W_k}^2)e.$$

In particular, as $\nabla f(x_k)$ is a column of G_k, we have

$$\nabla f(x_k)^T d_k \leq -\|G_k y_k\|_{W_k}^2 = -\|d_k\|_{W_k^{-1}}^2. \tag{3.8}$$

If $d_k = 0$, then there is nothing left to prove. Moreover, if $\nabla f(x_k) = 0$, then Algorithm 5 would have terminated in Step 2. Hence, from (3.8) and $W_k^{-1} \succ 0$, it follows that if Step 3 is reached and it produces $d_k \neq 0$, then $\nabla f(x_k)^T d_k < 0$. □

We remark that due to Lemma 3.2.2, the conditions in (3.6) could be written as

$$\xi\|d_k\|_2 \leq \|d_k\|_{W_k^{-1}} \leq \nu\epsilon_k \ \text{ and } \ \alpha_k > 0, \tag{2.6$'$}$$

from which it is clear that, in Algorithm 5, the sampling radius is decreased if and only if the step size is nonzero and the search direction has a W_k^{-1}-norm that is both relatively large compared to its Euclidean norm and relatively small compared to the current sampling radius.

3.2.2 Line Search

At an iterate $x_k \in \mathcal{D}$, Algorithm 5 either terminates in Step 2 or, by Lemma 3.2.2, it continues to Step 3 to produce a null or strict descent direction d_k for f from x_k. If $d_k = 0$, then we simply set α_k to its initial (positive) value, $x_{k+1} \leftarrow x_k$, and continue to the next step of the algorithm. If $d_k \neq 0$, then our line search aims to compute a step size $\alpha_k > 0$ such that $x_k + \alpha_k d_k$ yields an objective value that is sufficiently less than that yielded by x_k. In fact, it attempts to compute such a step size so that a curvature condition is also satisfied, as this would guarantee that an unadulterated BFGS update will yield $W_{k+1} \succ 0$; see §3.2.4. However, to ensure that the line search is well-posed under loose assumptions, this latter requirement is abandoned if such a step size is not computed within a predetermined number of line search iterations. We also terminate the search completely (and simply set $\alpha_k \leftarrow 0$ and $x_{k+1} \leftarrow x_k$) if the sample set X_k is not sufficiently large and, after a predetermined number of line search iterations, a sufficient decrease in f has not been obtained. This choice is motivated by the fact that if the sample set is not sufficiently large and a relatively large step size is not acceptable according to the line search conditions, then the algorithm may benefit by collecting more local gradient information before accepting a positive step size (which it can be seen to do by observing the sample set update in §3.2.3).

Given an iterate x_k and pair (d_k, y_k) from (3.5) with $d_k \neq 0$, we aim to compute a step size $\alpha_k > 0$ satisfying the following Armijo and curvature conditions, which together compose the well-known (weak) Wolfe line search conditions [49]:

$$f(x_k) - f(x_k + \alpha_k d_k) > \underline{\eta}\alpha_k \|G_k y_k\|_{W_k}^2; \tag{3.9a}$$

$$v^T d_k \geq \overline{\eta} \nabla f(x_k)^T d_k, \text{ where } v \in \partial f(x_k + \alpha_k d_k). \tag{3.9b}$$

(Technically, we are abusing this terminology as the traditional Armijo condition employs the negative directional derivative $-\nabla f(x_k)^T d_k$ in place of $\|G_k y_k\|_{W_k}^2$ in (3.9a). However, our abuse of this terminology is reasonable since, due to (3.8), the condition (3.9a) also ensures sufficient decrease in f from x_k after the step $\alpha_k d_k$.) If the resulting trial point satisfies $x_k + \alpha_k d_k \in \mathcal{D}$, then x_{k+1} is set to be this trial point; otherwise, we aim to compute $x_{k+1} \in \mathcal{D}$ satisfying

$$f(x_k) - f(x_{k+1}) \geq \underline{\eta}\alpha_k \|G_k y_k\|_{W_k}^2, \tag{3.10a}$$

$$\nabla f(x_{k+1})^T d_k \geq \overline{\eta} \nabla f(x_k)^T d_k, \tag{3.10b}$$

$$\text{and} \quad \|x_k + \alpha_k d_k - x_{k+1}\|_2 \leq \min\{\alpha_k, \epsilon_k\} \|d_k\|_2. \tag{3.10c}$$

Note that these conditions are also satisfied when $x_{k+1} \leftarrow x_k + \alpha_k d_k \in \mathcal{D}$, so we may refer to (3.10) as being satisfied whenever (3.9) holds and $x_{k+1} \leftarrow x_k + \alpha_k d_k$.

There are a variety of situations in which it may not be possible to compute a step size $\alpha_k > 0$ satisfying (3.9), or at least not within a predetermined number of iterations. For example, such a situation occurs when f is unbounded below along the ray $\{x_k + \alpha d_k : \alpha \geq 0\}$. However, even if f is bounded below over this ray, finite termination of a straightforward line search scheme may not be guaranteed without strengthening Assumption 3.2.1, or at least not without additional assumptions about f at x_k along d_k; see Lemma 3.2.4 below. Hence, we propose Algorithm 6 that guarantees finite termination by abandoning the curvature condition (3.9b) after a finite number of trial step sizes have been rejected. We also completely abandon the search for a positive step size (and set $\alpha_k \leftarrow 0$, $x_{k+1} \leftarrow x_k$, and eventually $\epsilon_{k+1} \leftarrow \epsilon_k$ due to (3.6c)) if X_k is not sufficiently large and the search has not been successful after a predetermined number of iterations. This truncation of the line search is required to prove our global convergence guarantees as it will result (by the method in §3.2.3) in additional gradient sampling about x_{k+1}.

After employing Algorithm 6 to compute a step size $\alpha_k \geq 0$, we employ Algorithm 7 to compute a new iterate $x_{k+1} \in \mathcal{D}$. If $\alpha_k = 0$, then $x_{k+1} \leftarrow x_k$, but if $\alpha_k > 0$, then x_{k+1} will satisfy the perturbed line search conditions (3.10), or at least (3.10a) and (3.10c). (If $\alpha_k > 0$, but (3.9b) does not hold, then we effectively ignore (3.10b) by immediately setting $j \leftarrow J + 1$ in Step 3 of Algorithm 7.)

We present the following lemma to show that our line search and iterate perturbation algorithms are well-posed. The lemma also delineates various situations that may result

Algorithm 6 Armijo-Wolfe Line Search

```
1: Take as input the quantities $(x_k, G_k, W_k, d_k, y_k)$ from Algorithm 5. Set the initial step size boundaries $l_0 \leftarrow 0$ and $u_0 \leftarrow \overline{\alpha}$, step size $\alpha_k \leftarrow \gamma\overline{\alpha}$, and iteration counter $j \leftarrow 0$.
2: If the step is null, i.e., $d_k = 0$, then terminate and return $\alpha_k$.
3: If the sample set is not sufficiently large in that $p_k < p$ and the upper iteration threshold has been surpassed in that $j > \overline{J}$, then set $\alpha_k \leftarrow 0$, terminate, and return $\alpha_k$.
4: If the lower iteration threshold has been surpassed in that $j > \underline{J}$, then reset $l_j \leftarrow 0$.
5: If the Wolfe conditions (3.9) hold, or if the Armijo condition (3.9a) holds and the lower iteration threshold has been surpassed in that $j > \underline{J}$, then terminate and return $\alpha_k$.
6: If the Armijo condition (3.9a) does not hold, then set $l_{j+1} \leftarrow l_j$ and $u_{j+1} \leftarrow \alpha_k$; otherwise, the curvature condition (3.9b) does not hold, so set $l_{j+1} \leftarrow \alpha_k$ and $u_{j+1} \leftarrow u_j$.
7: Set $\alpha_k \leftarrow (1-\gamma)l_{j+1} + \gamma u_{j+1}$.
8: Set $j \leftarrow j+1$ and go to Step 3.
```

Algorithm 7 Iterate Perturbation

```
1: Take as input the quantities $(x_k, \epsilon_k, G_k, W_k, d_k, y_k, \alpha_k)$ from Algorithm 5. Set the initial new iterate $x_{k+1} \leftarrow x_k + \alpha_k d_k$ and iteration counter $j \leftarrow 0$.
2: If the step or step size is null, i.e., $d_k = 0$ or $\alpha_k = 0$, then terminate and return $x_{k+1}$.
3: If the curvature condition (3.9b) does not hold, then set $j \leftarrow J+1$.
4: If $x_{k+1} \notin \mathcal{D}$, then continue to Step 5. Otherwise, if the (perturbed) Wolfe conditions (3.10) hold, or if the (perturbed) Armijo conditions (3.10a) and (3.10c) hold and the iteration threshold has been surpassed in that $j > J$, then terminate and return $x_{k+1}$.
5: Sample $x_{k+1}$ from a uniform distribution on $\mathbb{B}_{\min\{\alpha_k,\epsilon_k\}/j}(x_k + \alpha_k d_k)$.
6: Set $j \leftarrow j+1$ and go to Step 4.
```

after employing these two subroutines.

Lemma 3.2.3. *If Algorithm 5 reaches Step 4 during iteration k, then it either computes a null or positive step size α_k, where α_k is guaranteed to be positive if $p_k \geq p$. Moreover, if $\alpha_k > 0$, then the Wolfe conditions* (3.9)*, or at least the Armijo condition* (3.9a)*, is satisfied. Algorithm 5 then proceeds to Step 5, where with probability one it computes a new iterate $x_{k+1} \in \mathcal{D}$ with which the (perturbed) Wolfe conditions* (3.10)*, or at least the (perturbed) Armijo conditions* (3.10a) *and* (3.10c)*, are satisfied.*

Proof. If $d_k = 0$, then (3.9) holds for any value of $\alpha_k \geq 0$, so Algorithm 6 terminates in iteration $j = 0$ and returns $\alpha_k \leftarrow \gamma\overline{\alpha} > 0$. In this case, or if Algorithm 6 sets $\alpha_k \leftarrow 0$, then since $x_{k+1} \leftarrow x_k + \alpha_k d_k = x_k \in \mathcal{D}$ satisfies (3.10), it follows that Algorithm 7 terminates in iteration $j = 0$ and returns $x_{k+1} \leftarrow x_k + \alpha_k d_k$. Now suppose $d_k \neq 0$, from which it follows from Lemma 3.2.2 that d_k is a direction of strict descent for f from x_k. Without loss of generality, we may assume that Algorithm 6 performs at least $\underline{J}$ iterations, at which point it (re)sets $l_j \leftarrow 0$, and that it never sets $\alpha_k \leftarrow 0$. It then follows from the fact that $x_k \in \mathcal{D}$ and Lemma 3.2.2 that, after a finite number of additional iterations, $\alpha_k > 0$ will be produced at least satisfying the Armijo condition (3.9a). Turning to Algorithm 7, we may assume without loss of generality that at least J iterations will be performed, after which it follows from the strict inequality in (3.9a), the continuity of f, and Assumption 3.2.1 that, after a finite number of additional iterations and with probability one, a new iterate x_{k+1} will be produced satisfying (3.10a) and (3.10c). □

With additional assumptions about f, one could employ Algorithm 6 with the step size threshold set to $\overline{\alpha} \leftarrow \infty$ (assuming α_k is initialized to some finite value) and iteration thresholds set to $\underline{J} \leftarrow \infty$ and $\overline{J} \leftarrow \infty$ and still have a well-posed algorithm. To make this claim concrete, we present the following result, the proof of which follows from the results in [44, §4]; see also [41, 61, 46]. Although we do not wish to make the additional assumptions required in this lemma, we present this result to motivate the appeal of our line search strategy.

Lemma 3.2.4. *Suppose $f_k(\alpha) := f(x_k + \alpha d_k) - f(x_k)$ is bounded below and weakly lower semismooth [46] over $\{\alpha : \alpha > 0\}$. Then, if Algorithm 5 reaches Step 4 during iteration k and the function f_k is differentiable at all trial step sizes, Algorithm 6 with α_k initialized in $(0, \infty)$, $\overline{\alpha} \leftarrow \infty$, $\underline{J} \leftarrow \infty$, $\overline{J} \leftarrow \infty$, and Step 7 replaced by*

"7: If $u_{j+1} < \infty$, then set $\alpha_k \leftarrow (1-\gamma)l_{j+1} + \gamma u_{j+1}$; else, set $\alpha_k \leftarrow \alpha_k/\gamma$."

terminates finitely with $\alpha_k > 0$ satisfying (3.9).

3.2.3 Sample Point Generation

After the pair (d_k, y_k), step size α_k, and new iterate x_{k+1} have been computed, we are ready in Algorithm 5 to establish the new sample set X_{k+1}. As previously mentioned, we claim that the ideal behavior of the algorithm is that of an unadulterated BFGS method, at least when such a method continues to make sufficient progress in reducing the objective function f. Hence, if the curvature of W_k along $G_k y_k$ (equal, by Lemma 3.2.2, to the curvature of W_k^{-1} along d_k) is bounded below in that (3.6b) holds, and if the computed step size is sufficiently large in that

$$\alpha_k \geq \underline{\alpha}, \tag{3.11}$$

then we set the default value of $X_{k+1} \leftarrow \{x_{k+1}\}$ so that an unadulterated BFGS step will be computed in the following iteration. However, if either of these conditions does not hold, then we augment the sample set with points obtained from the previous sample set and some randomly generated in an ϵ_{k+1}-neighborhood about x_{k+1}. The details of our sample set update are stated in Algorithm 8, and the salient consequences of this strategy are provided in the following lemma.

Algorithm 8 Sample Set Update

```
1: Take as input the quantities $(x_{k+1}, \epsilon_{k+1}, G_k, W_k, d_k, y_k, \alpha_k)$ from Algorithm 5.
2: If the curvature of the inverse Hessian approximation is bounded below in that (3.6b)
   holds and the step size is sufficiently large in that (3.11) holds, then set $X_{k+1} \leftarrow \{x_{k+1}\}$
   and $p_{k+1} \leftarrow 0$, terminate, and return $(X_{k+1}, p_{k+1})$.
3: Set $X_{k+1} \leftarrow (X_k \cap B_{k+1}) \cup \{x_{k+1}\}$ and choose $\overline{p}_{k+1} \in \mathbb{N}_+$.
4: Set $\overline{X}_{k+1}$ as a collection of $\overline{p}_{k+1}$ points generated independently from a uniform dis-
   tribution over $\mathbb{B}_{\epsilon_{k+1}}(x_{k+1})$.
5: If $\overline{X}_{k+1} \not\subset \mathcal{D}$, then go to Step 4.
6: Set $X_{k+1} \leftarrow X_{k+1} \cup \overline{X}_{k+1}$ and $p_{k+1} \leftarrow |X_{k+1}| - 1$.
7: If $p_{k+1} > p$, then remove the $p_{k+1} - p$ eldest members of $X_{k+1} \backslash \{x_{k+1}\}$ and set $p_{k+1} \leftarrow$
   $p$.
8: Terminate and return $(X_{k+1}, p_{k+1})$.
```

Lemma 3.2.5. *If Algorithm 5 reaches Step 7 during iteration k, then it either sets* $X_{k+1} \leftarrow \{x_{k+1}\}$ *and* $p_{k+1} \leftarrow 0$*, or, with probability one, it produces*

$$X_{k+1} \leftarrow ((X_k \cap B_{k+1}) \cup \{x_{k+1}\} \cup \overline{X}_{k+1}) \subset B_{k+1}$$

and $p_{k+1} \geq \min\{p_k + 1, p\}$.

Proof. If (3.6b) and (3.11) hold, then the algorithm sets $X_{k+1} \leftarrow \{x_{k+1}\}$ and $p_{k+1} \leftarrow 0$, which is the first desirable result. Otherwise, the result follows by the construction of Algorithm 8, Assumption 3.2.1, and the fact that the points in $\overline{X}_{k+1}$ are generated independently and uniformly in $\mathbb{B}_{\epsilon_{k+1}}(x_{k+1})$. □

3.2.4 Hessian Approximation Strategy

Upon computing the pair (d_k, y_k), step size α_k, new iterate x_{k+1}, and new sample set X_{k+1}, the final main step of Algorithm 5 is to compute a new inverse Hessian approximation W_{k+1}. If the curvature along $G_k y_k$ determined by W_k is bounded below in that (3.6b) holds and if the step size is sufficiently large in that (3.11) holds, then we obtain $W_{k+1} \succ 0$ from $W_k \succ 0$ via a standard (damped) BFGS update. However, if one of these conditions does not hold, then we have reason to believe that a standard BFGS update may lead to an approximation whose ill-conditioning may be detrimental to the performance of the algorithm (or at least to our mechanisms for guaranteeing productive steps and/or verifying stationarity). Hence, in such cases, we set $W_{k+1} \succ 0$ by an L-BFGS strategy in which we monitor the updates so that the resulting matrix has a provably bounded condition number.

The algorithm presented in this section makes use of the quantities

$$s_k := x_{k+1} - x_k \quad \text{and} \quad t_k := \nabla f(x_{k+1}) - \nabla f(x_k) \quad \text{for all} \quad k \geq 0. \tag{3.12}$$

It also potentially uses the set of pairs $\{(s_{k-m+1}, t_{k-m+1}), \dots, (s_{k-1}, t_{k-1})\}$, where each element is defined similarly as in (3.12) for the previous $m-1$ iterations. We did not mention these pairs in our description of Algorithm 5, though it is obvious that these vectors may be stored in Algorithm 5 (having no effect on any other quantities or steps of Algorithm 5) for use in the algorithm in this subsection.

If $s_k = 0$ or $t_k = 0$, then we claim that we have obtained no useful curvature information from the step from x_k to x_{k+1}, so we set $W_{k+1} \leftarrow W_k$. Otherwise, if (3.6b) and (3.11) hold, then we damp the BFGS update by setting

$$r_k \leftarrow \delta_k s_k + (1 - \delta_k) W_k t_k, \tag{3.13}$$

where the scalar δ_k is defined by

$$\delta_k \leftarrow \begin{cases} 1 & \text{if } s_k^T t_k \geq \underline{\mu} t_k^T W_k t_k \\ (1-\underline{\mu}) t_k^T W_k t_k / (t_k^T W_k t_k - s_k^T t_k) & \text{if } s_k^T t_k < \underline{\mu} t_k^T W_k t_k, \end{cases} \tag{3.14}$$

then employ the standard BFGS formula with (s_k, t_k) replaced by (r_k, t_k) [49]:

$$W_{k+1} \leftarrow \left(I - \frac{r_k t_k^T}{r_k^T t_k}\right) W_k \left(I - \frac{t_k r_k^T}{r_k^T t_k}\right) + \frac{r_k r_k^T}{r_k^T t_k}. \tag{3.15}$$

On the other hand, if $s_k \neq 0$ and $t_k \neq 0$, but at least one of (3.6b) or (3.11) does not hold, then we employ a damped L-BFGS strategy, proceeding in the following manner. First, choosing a scalar $w_k > 0$, we initialize $W_{k+1}^{(k-m)} \leftarrow w_k I$. Then, for increasing j in the ordered set $\{k-m+1, \ldots, k\}$, we set

$$r_j \leftarrow \delta_j s_j + (1-\delta_j) W_{k+1}^{(j-1)} t_j, \tag{3.16}$$

where

$$\delta_j \leftarrow \begin{cases} 1 & \text{if } s_j^T t_j \geq \underline{\mu} t_j^T W_{k+1}^{(j-1)} t_j \\ (1-\underline{\mu}) t_j^T W_{k+1}^{(j-1)} t_j / (t_j^T W_{k+1}^{(j-1)} t_j - s_j^T t_j) & \text{if } s_j^T t_j < \underline{\mu} t_j^T W_{k+1}^{(j-1)} t_j, \end{cases} \tag{3.17}$$

and

$$W_{k+1}^{(j)} \leftarrow \left(I - \frac{r_j t_j^T}{r_j^T t_j}\right) W_{k+1}^{(j-1)} \left(I - \frac{t_j r_j^T}{r_j^T t_j}\right) + \frac{r_j r_j^T}{r_j^T t_j}. \tag{3.18}$$

Finally, we set $W_{k+1} \leftarrow W_{k+1}^{(k)}$. In this procedure, in order to guarantee that the resulting inverse Hessian approximation has a bounded condition number, for each j we skip the update in (3.18) (and simply set $W_{k+1}^{(j)} \leftarrow W_{k+1}^{(j-1)}$) unless

$$s_j \neq 0, \quad t_j \neq 0, \quad \text{and} \quad \max\{\|r_j\|_2^2, \|t_j\|_2^2\} \leq \overline{\mu} r_j^T t_j. \tag{3.19}$$

We formalize our inverse Hessian approximation strategy as Algorithm 9. We assume that the vectors in $\{(s_{-m+1}, t_{-m+1}), \ldots, (s_{-1}, t_{-1})\}$ are initialized to zero so that if the

L-BFGS strategy is employed in iteration $k < m$, then, as a consequence of the condition (3.19), at most k updates will be performed.

Algorithm 9 Inverse Hessian Approximation Update

```
1: Take as input the quantities $(x_k, x_{k+1}, \nabla f(x_k), \nabla f(x_{k+1}), W_k)$ from Algorithm 5 and
   the previously computed sequence $\{(s_{k-m+1}, t_{k-m+1}), \ldots, (s_{k-1}, t_{k-1})\}$.
2: Set $s_k$ and $t_k$ by (3.12).
3: If $s_k = 0$ or $t_k = 0$, then set $W_{k+1} \leftarrow W_k$, terminate, and return $W_{k+1}$.
4: If the curvature of the inverse Hessian approximation is bounded below in that (3.6b)
   holds and the step size is sufficiently large in that (3.11) holds, then set $r_k$, $\delta_k$, and
   $W_{k+1}$ by (3.13)–(3.15), terminate, and return $W_{k+1}$.
5: Choose $w_k \in [\underline{w}, \overline{w}]$ and initialize $W_{k+1}^{(k-m)} \leftarrow w_k I$.
6: for increasing $j \in \{k-m+1, \ldots, k\}$ do
7:     Set $r_j$ and $\delta_j$ by (3.16)–(3.17).
8:     if (3.19) holds then
9:         Set $W_{k+1}^{(j)}$ by (3.18).
10:    else
11:        Set $W_{k+1}^{(j)} \leftarrow W_{k+1}^{(j-1)}$.
12: Set $W_{k+1} \leftarrow W_{k+1}^{(k)}$, terminate, and return $W_{k+1}$.
```

In the remainder of this section, we prove properties of the inverse Hessian approximation W_{k+1} returned by Algorithm 9 during iteration k of Algorithm 5. First, we state the result that the damped BFGS update (3.15) yields $W_{k+1} \succ 0$. This fact is well known [49], so we state it without proof.

Lemma 3.2.6. *With $W_k \succ 0$, $s_k \neq 0$, and $t_k \neq 0$, the update* (3.15) *yields $W_{k+1} \succ 0$.*

Next, we prove a result about the update (3.18). We prove this result for completeness, but see also the similar result [18, Lemma 3.2].

Lemma 3.2.7. *Suppose that for $\overline{\theta} \geq \underline{\theta} > 0$ we have*

$$\underline{\theta}\|t\|_2^2 \leq t^T W_{k+1}^{(j-1)} t \leq \overline{\theta}\|t\|_2^2 \quad \text{for all} \quad t \in \mathbb{R}^n. \tag{3.20}$$

Then, with (r_j, s_j, t_j) satisfying (3.19)*, the update* (3.18) *yields $W_{k+1}^{(j)}$ such that*

$$(\underline{\theta}^{-1} + \overline{\mu})^{-1}\|t\|_2^2 \leq t^T W_{k+1}^{(j)} t \leq (2\overline{\theta}(1 + \overline{\mu}^2) + \overline{\mu})\|t\|^2 \quad \text{for all} \quad t \in \mathbb{R}^n. \tag{3.21}$$

Proof. Applying the Sherman-Morrison-Woodbury formula [25] to (3.18), we obtain

$$(W_{k+1}^{(j)})^{-1} \leftarrow (W_{k+1}^{(j-1)})^{-1} - \frac{(W_{k+1}^{(j-1)})^{-1} r_j r_j^T (W_{k+1}^{(j-1)})^{-1}}{r_j^T (W_{k+1}^{(j-1)})^{-1} r_j} + \frac{t_j t_j^T}{r_j^T t_j}. \tag{3.22}$$

The fact that (3.20) holds implies that $W_{k+1}^{(j-1)} \succ 0$, which along with the Rayleigh-Ritz theorem [35] implies that

$$t^T (W_{k+1}^{(j-1)})^{-1} t \leq \underline{\theta}^{-1} \|t\|_2^2 \ \text{ for all } \ t \in \mathbb{R}^n.$$

From this inequality, (3.19), (3.22), and the Cauchy-Schwarz inequality, we have

$$\begin{aligned} t^T (W_{k+1}^{(j)})^{-1} t &= t^T (W_{k+1}^{(j-1)})^{-1} t - \frac{(r_j^T (W_{k+1}^{(j-1)})^{-1} t)^2}{r_j^T (W_{k+1}^{(j-1)})^{-1} r_j} + \frac{(t_j^T t)^2}{r_j^T t_j} \\ &\leq t^T (W_{k+1}^{(j-1)})^{-1} t + \frac{(t_j^T t)^2}{r_j^T t_j} \\ &\leq (\underline{\theta}^{-1} + \overline{\mu}) \|t\|_2^2 \ \text{ for all } \ t \in \mathbb{R}^n. \end{aligned}$$

Hence, again applying the Rayleigh-Ritz theorem, we obtain the former inequality in (3.21). As for the latter inequality in (3.21), first note that from (3.19), we have

$$t^T \left(\frac{r_j r_j^T}{r_j^T t_j} \right) t = \frac{(r_j^T t)^2}{r_j^T t_j} \leq \overline{\mu} \|t\|_2^2. \tag{3.23}$$

Since (3.20) implies $W_{k+1}^{(j-1)} \succ 0$, we may write $W_{k+1}^{(j-1)} = N_j^T N_j$ for some nonsingular $N_j \in \mathbb{R}^{n \times n}$. Moreover, from (3.20), we have

$$t^T W_{k+1}^{(j-1)} t = \|N_j t\|_2^2 \leq \overline{\theta} \|t\|_2^2 \ \text{ for all } \ t \in \mathbb{R}^n,$$

which, along with (3.19), implies that

$$\left\| \frac{N_j t_j r_j^T t}{r_j^T t_j} \right\|_2^2 = \left(\frac{r_j^T t}{r_j^T t_j} \right)^2 \|N_j t_j\|_2^2 \leq \overline{\mu}^2 \overline{\theta} \|t\|_2^2. \tag{3.24}$$

Thus, from (3.24) and the fact that for any vectors a and b of equal length we have

$\|a-b\|^2 \leq 2(\|a\|^2 + \|b\|^2)$, it follows that

$$\left\|N_j\left(I - \frac{t_j r_j^T}{r_j^T t_j}\right)t\right\|_2^2 \leq 2\left(\|N_j t\|_2^2 + \left\|\frac{N_j t_j r_j^T t}{r_j^T t_j}\right\|_2^2\right) \leq 2\overline{\theta}(1+\overline{\mu}^2)\|t\|_2^2.$$

Overall, the above and (3.23) yield

$$t^T W_{k+1}^{(j)} t = \left\|N_j\left(I - \frac{t_j r_j^T}{r_j^T t_j}\right)t\right\|^2 + t^T\left(\frac{r_j r_j^T}{r_j^T t_j}\right)t \leq (2\overline{\theta}(1+\overline{\mu}^2) + \overline{\mu})\|t\|_2^2,$$

which is precisely the latter inequality in (3.21). □

We conclude this section with the following lemma revealing that for any k, the matrix W_{k+1} returned by Algorithm 9 is positive definite, and is also bounded in certain important situations; see the similar result [18, Theorem 3.3].

Lemma 3.2.8. *Algorithm 9 with input $W_k \succ 0$ yields W_{k+1} satisfying the following.*

(a) If $s_k = 0$ or $t_k = 0$, then $W_{k+1} \leftarrow W_k \succ 0$.

(b) If $s_k \neq 0$, $t_k \neq 0$, and (3.6b) and (3.11) hold, then $W_{k+1} \succ 0$.

(c) If $s_k \neq 0$ and $t_k \neq 0$, but at least one of (3.6b) or (3.11) does not hold, then $W_{k+1} \succ 0$ and for all $t \in \mathbb{R}^n$ we have

$$t^T W_{k+1} t \geq (\underline{w}^{-1} + m\overline{\mu})^{-1}\|t\|_2^2 \tag{3.25a}$$

$$t^T W_{k+1} t \leq \left(2^m\left(1+\overline{\mu}^2\right)^m \overline{w} + \overline{\mu}\left(\frac{2^m\left(1+\overline{\mu}^2\right)^m - 1}{2\left(1+\overline{\mu}^2\right) - 1}\right)\right)\|t\|_2^2. \tag{3.25b}$$

Proof. If $s_k = 0$ or $t_k = 0$, then, by Step 3, Algorithm 9 sets $W_{k+1} \leftarrow W_k \succ 0$, as desired. Otherwise, if (3.6b) and (3.11) hold, then, by Step 4, Algorithm 9 sets W_{k+1} by (3.15), which by Lemma 3.2.6 implies that $W_{k+1} \succ 0$, as desired.

All that remains is to consider the case when $s_k \neq 0$ and $t_k \neq 0$, but at least one of (3.6b) or (3.11) does not hold. In this case, by Steps 5–12, Algorithm 9 sets W_{k+1} by choosing $w_k \in [\underline{w}, \overline{w}]$, initializing $W_{k+1}^{(k-m)} = w_k I \succ 0$, applying (at most) m updates of the form (3.18) with quantities satisfying (3.19), and finally setting $W_{k+1} \leftarrow W_{k+1}^{(k)}$.

Since, by Lemma 3.2.7, each application of (3.18) takes the bounds in (3.20) and produces the wider bounds in (3.21), we may assume without loss of generality that all m updates are performed, i.e., that (3.19) holds for all $j \in \{k-m+1, \ldots, k\}$. Thus, starting with $\underline{\theta} = \overline{\theta} = w_k$, the result of Lemma 3.2.7 can be applied repeatedly for increasing $j \in \{k-m+1, \ldots, k\}$. In particular, as seen in the proof of Lemma 3.2.7, the upper bound corresponding to the inverse of the approximation increases by the constant factor $\overline{\mu} > 0$ with each update, so after m updates we obtain (3.25a). As for the upper bound (3.25b), by applying Lemma 3.2.7 for increasing $j \in \{k-m+1, \ldots, k\}$ we obtain for all $t \in \mathbb{R}^n$ that

$$\begin{aligned} t^T W_{k+1} t \leq\ & (2^m(1+\overline{\mu}^2)^m w_k + 2^{m-1}(1+\overline{\mu}^2)^{m-1}\overline{\mu} + \cdots + 2(1+\overline{\mu}^2)\overline{\mu} + \overline{\mu})\|t\|_2^2 \\ =\ & \left(2^m(1+\overline{\mu}^2)^m w_k + \overline{\mu}\left(\frac{2^m(1+\overline{\mu}^2)^m - 1}{2(1+\overline{\mu}^2)-1}\right)\right)\|t\|_2^2, \end{aligned}$$

which, since $w_k \in [\underline{w}, \overline{w}]$, implies that (3.25b) holds. □

LewiOver13

3.3 Global Convergence Analysis

In this section, we prove that Algorithm 5 is globally convergent from remote starting points. Specifically, with the restriction that

$$0 < \xi \leq \left(2^m\left(1+\overline{\mu}^2\right)^m \overline{w} + \overline{\mu}\left(\frac{2^m\left(1+\overline{\mu}^2\right)^m - 1}{2\left(1+\overline{\mu}^2\right)-1}\right)\right)^{-1} \tag{3.26}$$

the result that we prove is the following.

Theorem 3.3.1. *Algorithm 5 either terminates finitely with a stationary point for f or, with probability one, it produces an infinite sequence of iterates $\{x_k\}$. In the latter case, with probability one, either $\{f(x_k)\} \to -\infty$ or $\{\epsilon_k\} \to 0$ and every cluster point of the iterate sequence $\{x_k\}$ is stationary for f.*

We begin our analysis for proving Theorem 3.3.1 by summarizing the results of the previous section to prove that Algorithm 5 is well-posed.

Lemma 3.3.2. *Algorithm 5 is well-posed in the sense that it either terminates in Step 2 with a stationary point for f or, with probability one, it produces an infinite sequence of iterates $\{x_k\}$. In either case, for each k, the following hold true:*

(a) The primal-dual solution (z_k, d_k, y_k) of (3.5) *satisfies $\|G_k y_k\|_{W_k} = \|d_k\|_{W_k^{-1}}$ where d_k is either zero or is a direction of strict descent for f from $x_k \in \mathcal{D}$.*

(b) The step size α_k is nonnegative, and is positive if $p_k \geq p$. If $\alpha_k > 0$, then either the Wolfe conditions (3.9) *hold or at least the Armijo condition* (3.9a) *holds.*

(c) With probability one, $x_{k+1} \in \mathcal{D}$ is computed satisfying the (perturbed) Wolfe conditions (3.10) *or at least the (perturbed) Armijo conditions* (3.10a) *and* (3.10c)*.*

(d) If Step 6 is reached and (3.6) *holds, then $\epsilon_{k+1} \leftarrow \psi\epsilon_k$; otherwise, $\epsilon_{k+1} \leftarrow \epsilon_k$.*

(e) If Step 7 is reached and (3.6b) *and* (3.11) *hold, then $X_{k+1} \leftarrow \{x_{k+1}\}$ along with $p_{k+1} \leftarrow 0$; otherwise, with probability one,*

$$X_{k+1} \leftarrow ((X_k \cap B_{k+1}) \cup \{x_{k+1}\} \cup \overline{X}_{k+1}) \subset B_{k+1}$$

is generated and $p_{k+1} \geq \min\{p_k + 1, p\}$.

(f) If Step 9 is reached, then $W_{k+1} \succ 0$, where if $s_k \neq 0$, $t_k \neq 0$, and at least one of (3.6b) *or* (3.11) *does not hold, then W_{k+1} satisfies the bounds in* (3.25)*.*

Proof. The result follows by the construction of Algorithms 5, 6, 7, 8, and 9 along with the results of Lemmas 3.2.2, 3.2.3, 3.2.5, 3.2.6, and 3.2.8. □

For simplicity in our analysis until our proof of Theorem 3.3.1 at the end of this section, we assume without loss of generality that Algorithm 5 produces an infinite iterate sequence $\{x_k\}$. Implicit in this assumption is that the procedures to compute x_{k+1} and X_{k+1} terminate finitely for all k, i.e., that these procedures may be considered deterministic. This is reasonable since, by Lemma 3.3.2, these procedures terminate finitely with probability one, and since there is nothing else that we aim to prove when they fail to terminate finitely.

In our next result, we prove that there exists an infinite subsequence of iterates in which the algorithm produces a positive step size.

Lemma 3.3.3. *There exists an infinite subsequence of iterations in which $\alpha_k > 0$.*

Proof. To derive a contradiction, suppose that there exists an iteration number k' such that for all $k \geq k'$ we have $\alpha_k = 0$. By Lemma 3.3.2(b), this must mean that for all $k \geq k'$ we have $p_k \leq p - 1$. However, with $\alpha_k = 0$, we have that (3.11) does not hold, which by Lemma 3.3.2(e) implies that the algorithm will set $p_{k+1} \geq \min\{p_k + 1, p\}$. This means that for some $k'' \geq k'$ we have $p_{k''} \geq p$, which contradicts the conclusion that $p_k \leq p - 1$ for all $k \geq k'$. □

We now prove a critical inequality for a subset of iterations.

Lemma 3.3.4. *If* (3.6b) *holds during iteration k, then*

$$f(x_{k+1}) \leq f(x_k) - \tfrac{1}{2}\underline{\eta}\xi\|x_{k+1} - x_k\|_2\|d_k\|_2.$$

Proof. By the reverse triangle inequality, (3.10c) ensures that

$$\|x_{k+1} - x_k\|_2 \leq \min\{\alpha_k, \epsilon_k\}\|d_k\|_2 + \alpha_k\|d_k\|_2 \leq 2\alpha_k\|d_k\|_2. \tag{3.27}$$

Thus, by (3.10a), (3.6b), and (3.27), we have

$$\begin{aligned} f(x_{k+1}) - f(x_k) &\leq -\underline{\eta}\alpha_k\|G_k y_k\|_{W_k}^2 \\ &\leq -\underline{\eta}\alpha_k\xi\|d_k\|_2^2 \\ &\leq -\tfrac{1}{2}\underline{\eta}\xi\|x_{k+1} - x_k\|_2\|d_k\|_2, \end{aligned}$$

as desired. □

We now prove a useful lemma on approximate least W-norm elements in certain types of sets of interest. For the lemma, recall that for $W \succ 0$ and nonempty bounded $S \subseteq \mathbb{R}^n$, we define $P_W(S)$ as the (oblique) W-projection of the origin onto $\operatorname{cl}\operatorname{conv} S$. The lemma can be seen as a variation of [39, Lemma 3.1].

Lemma 3.3.5. *Consider $W \succ 0$, a nonempty bounded set $S \subseteq \mathbb{R}^n$, and a constant $\beta \in (0,1)$. If $0 \notin \operatorname{cl}\operatorname{conv} S$, then there exists a constant $\kappa > 0$ such that for any $\{u,v\} \subseteq \operatorname{cl}\operatorname{conv} S$ the inequality $\|u\|_W^2 \leq \|P_W(S)\|_W^2 + \kappa$ implies $v^T W u > \beta \|u\|_W^2$.*

Proof. By definition, we have

$$P_W(S) := \underset{x \in \operatorname{cl}\operatorname{conv} S}{\arg\min} \; \|x\|_W^2,$$

which implies (e.g., see [4, Proposition 1.1.8]) that for all $v \in \operatorname{cl}\operatorname{conv} S$ we have

$$v^T W P_W(S) \geq \|P_W(S)\|_W^2. \tag{3.28}$$

We now prove the result by contradiction. If the result were false, then there exist sequences $\{u_i\} \subseteq \operatorname{cl}\operatorname{conv} S$ and $\{v_i\} \subseteq \operatorname{cl}\operatorname{conv} S$ satisfying $\|u_i\|_W^2 \leq \|P_W(S)\|_W^2 + 1/i$ and $v_i^T W u_i \leq \beta \|u_i\|_W^2$ for all $i \geq 0$. Then, $\{u_i\} \to \overline{u} = P_W(S) \neq 0$, and since S is bounded, it follows that $\operatorname{cl}\operatorname{conv} S$ is compact, meaning that we may assume that $\{v_i\} \to \overline{v} \in \operatorname{cl}\operatorname{conv} S$ such that $\overline{v}^T W \overline{u} \leq \beta \|\overline{u}\|_W^2$. On the other hand, we have from (3.28) that $v^T W \overline{u} \geq \|\overline{u}\|_W^2$ for all $v \in \operatorname{cl}\operatorname{conv} S$, a contradiction. □

Next, we prove a technical lemma pertaining to the discrepancy between two related measures of proximity to ϵ-stationarity. Given $x' \in \mathbb{R}^n$, we define

$$\mathcal{G}_k(x') := \operatorname{cl}\operatorname{conv} \nabla f(\mathbb{B}_{\epsilon_k}(x') \cap \mathcal{D}),$$

and, also given a tolerance $\omega > 0$, we define

$$\mathcal{T}_k(x', \omega) := \left\{ X_k \in \prod_0^{p_k} B_k : \|P_{W_k}(\{\nabla f(x)\}_{x \in X_k})\|_{W_k}^2 \leq \|P_{W_k}(\mathcal{G}_k(x'))\|_{W_k}^2 + \omega \right\}.$$

The purpose of the lemma is to show that for a sufficiently large sample set size p_k, an iterate x_k sufficiently close to x', and any $\omega > 0$, there exists a nonempty subset of $\mathcal{T}_k(x', \omega)$; see the similar result [39, Lemma 3.2(i)].

Lemma 3.3.6. *If $p_k \geq n+1$, then for any $\omega > 0$, there exists $\zeta > 0$ and a nonempty set $\mathcal{T}$ such that for all $x_k \in \mathbb{B}_\zeta(x')$ we have $\mathcal{T} \subseteq \mathcal{T}_k(x', \omega)$.*

Proof. Under Assumption 3.2.1, there exists $g \in \operatorname{conv} \nabla f(\mathbb{B}_{\epsilon_k}(x') \cap \mathcal{D})$ such that

$$\|g\|_{W_k}^2 \leq \|P_{W_k}(\mathcal{G}_k(x'))\|_{W_k}^2 + \omega.$$

Then, since $p_k \geq n+1$, Carathéodory's Theorem (e.g., see [4, Proposition 1.2.1]) implies the existence of a set of points $\{x'_0, \ldots, x'_{p_k}\} \subset \mathbb{B}_{\epsilon_k}(x') \cap \mathcal{D}$ and a set of nonnegative scalars $\{\lambda_0, \ldots, \lambda_{p_k}\}$ such that

$$\sum_{i=0}^{p_k} \lambda_i = 1 \quad \text{and} \quad \sum_{i=0}^{p_k} \lambda_i \nabla f(x'_i) = g.$$

Since f is continuously differentiable in $\mathcal{D}$, there exists $\zeta \in (0, \epsilon_k)$ such that

$$\mathcal{T} := \prod_{i=0}^{p_k} \operatorname{int} \mathbb{B}_\zeta(x'_i)$$

lies in $\mathbb{B}_{\epsilon_k - \zeta}(x')$ and for any $X_k \in \mathcal{T}$ we have

$$\|P_{W_k}(\{\nabla f(x)\}_{x \in X_k})\|_{W_k}^2 \leq \|P_{W_k}(\mathcal{G}_k(x'))\|_{W_k}^2 + \omega.$$

Thus, for all $x_k \in \mathbb{B}_\zeta(x')$, the fact that $\mathbb{B}_{\epsilon_k - \zeta}(x') \subset \mathbb{B}_{\epsilon_k}(x_k)$ and the above conclusion implies that $\emptyset \neq \mathcal{T} \subset \mathcal{T}_k(x', \omega)$. $\square$

We are now prepared to prove our main result.

Theorem 3.3.1. If Algorithm 5 terminates finitely with a stationary point for f, or if Algorithm 7 or 8 is called and fails to terminate finitely, then there is nothing left to prove. Otherwise, by Lemma 3.3.2, Algorithm 5 produces an infinite sequence of iterates $\{x_k\}$. In this case, if $\{f(x_k)\} \to -\infty$, then again there is nothing left to prove, so for the remainder of our analysis we suppose that an infinite iterate sequence $\{x_k\}$ is generated and that

$$\inf_{k \to \infty} f(x_k) > -\infty. \tag{3.29}$$

Our first main goal is to show that $\{\epsilon_k\} \to 0$. To prove this, we consider two cases.

Case 1: Suppose there exists an infinite iteration index set $\mathcal{K}$ such that (3.6b) and (3.11) hold for all $k \in \mathcal{K}$. Then, along with (3.10a), we have

$$f(x_{k+1}) - f(x_k) \leq -\underline{\eta}\alpha_k \|G_k y_k\|_{W_k}^2 \leq -\underline{\eta}\underline{\alpha}\xi^2 \|d_k\|_2^2 \quad \text{for all} \quad k \in \mathcal{K}.$$

Since f is bounded below by (3.29), this implies that

$$\lim_{k \in \mathcal{K}} \|d_k\|_2 = 0,$$

which, by Step 6 of Algorithm 5, implies that $\{\epsilon_k\} \to 0$.

Case 2: Suppose that at least one of (3.6b) or (3.11) does not hold for all sufficiently large k. By the construction of Steps 3–4 of Algorithm 9, it follows that this algorithm will set W_{k+1} satisfying (3.25) for all such k, and hence, with (3.26), it follows that for all sufficiently large k we have

$$\begin{array}{rrcll} & \xi\|d\|_2^2 & \leq & d^T W_{k+1}^{-1} d & \text{for all} \quad d \in \mathbb{R}^n, \\ \text{or, equivalently,} & t^T W_{k+1} t & \leq & \xi^{-1}\|t\|_2^2 & \text{for all} \quad t \in \mathbb{R}^n. \end{array} \tag{3.30}$$

Indeed, in this case, we may assume without loss of generality that these inequalities hold for all k. We now prove that $\{\epsilon_k\} \to 0$ with probability one by showing that the event that $\{\epsilon_k\}$ remains bounded away from zero has probability zero.

Suppose that there exists k' such that $\epsilon_k = \epsilon' > 0$ for all $k \geq k'$. From this fact, it follows that at least one of (3.6a), (3.6b), or (3.6c) does not hold for all $k \geq k'$. In fact, since (3.30) and Lemma 3.3.2(a) imply that (3.6b) holds for all k, we must have that (3.6a) or (3.6c) does not hold for all $k \geq k'$. However, by Lemma 3.3.3, we have the existence of the infinite iteration index set $\mathcal{K}_\alpha := \{k : k \geq k' \text{ and } \alpha_k > 0\}$. Thus, overall,

$$\|G_k y_k\|_{W_k} > \nu\epsilon' \quad \text{for all} \quad k \in \mathcal{K}_\alpha. \tag{3.31}$$

On the other hand, the fact that $\{f(x_k)\}$ is bounded below by (3.29), the sufficient decrease condition (3.10a), Lemma 3.3.4 (since (3.6b) holds for all k), and the fact that $\alpha_k = 0$ for all $k \geq k'$ with $k \notin \mathcal{K}_\alpha$ together imply that

$$\sum_{k=k'}^{\infty} \alpha_k \|G_k y_k\|_{W_k}^2 < \infty, \quad \text{and} \tag{3.32a}$$

$$\sum_{k=k'}^{\infty} \|x_{k+1} - x_k\|_2 \|d_k\|_2 < \infty. \tag{3.32b}$$

In conjunction with (3.31), the bound (3.32a) implies $\alpha_k \to 0$. Similarly, (3.32b), (3.31), Lemma 3.3.2(a), and (3.30) imply that $\{x_k\}$ is a Cauchy sequence, and hence $x_k \to x'$ for some $x' \in \mathbb{R}^n$. We claim that this implies the existence of an infinite iteration index set $\mathcal{K}_p := \{k : k \geq k' \text{ and } p_k = p\}$, for which Lemma 3.3.2(b) implies $\mathcal{K}_p \subseteq \mathcal{K}_\alpha$. Indeed, if $p_k < p$ for all large k, then, since $\alpha_k \to 0$, Step 3 of Algorithm 6 implies that $\alpha_k = 0$ for all large k. However, as in the proof of Lemma 3.3.3, this leads to a contradiction as we eventually find $p_k = p$ for some large k. Therefore, we can define $\mathcal{K}_p$ as stated and know $|\mathcal{K}_p| = \infty$. We continue by considering two subcases.

Subcase 2.a: If x' is ϵ'-stationary for f, then $\|P_{W_k}(\mathcal{G}_k(x'))\|_{W_k}^2 = 0$ for any $W_k \succ 0$. Thus, with $\omega = (\nu\epsilon')^2 > 0$ and $(\zeta, \mathcal{T})$ chosen as in Lemma 3.3.6, there exists $k'' \geq k'$ such that $x_k \in \mathbb{B}_\zeta(x')$ for all $k \geq k''$ and

$$\|G_k y_k\|_{W_k} = \|P_{W_k}(\{\nabla f(x)\}_{x \in X_k})\|_{W_k} \leq \nu\epsilon' \tag{3.33}$$

whenever $k \geq k''$, $k \in \mathcal{K}_p$, and $X_k \in \mathcal{T}$. Together, (3.31) and (3.33) imply that $X_k \notin \mathcal{T}$ for all $k \geq k''$ with $k \in \mathcal{K}_p$. However, this is a probability zero event since the construction of Algorithm 8 implies that for all such k the set X_k will contain newly generated points from B_k, meaning that with probability one there exists some sufficiently large k such that $X_k \in \mathcal{T}$, yielding (3.33).

Subcase 2.b: Now suppose that x' is not ϵ'-stationary for f. It follows from Lemma 3.3.2(b) (in particular, the Armijo condition (3.9a)) that for all k we have

$$f(x_k + \gamma^{-1}\alpha_k d_k) - f(x_k) \geq -\underline{\eta}\gamma^{-1}\alpha_k \|G_k y_k\|_{W_k}^2, \tag{3.34}$$

while Lebourg's Mean Value Theorem [15, Theorem 2.3.7] implies the existence of $\tilde{x}_k \in [x_k, x_k + \gamma^{-1}\alpha_k d_k]$ and a corresponding subgradient $v_k \in \partial f(\tilde{x}_k)$ such that

$$f(x_k + \gamma^{-1}\alpha_k d_k) - f(x_k) = \gamma^{-1}\alpha_k v_k^T d_k. \tag{3.35}$$

Together, (3.34), (3.35), and the fact that $d_k = -W_k G_k y_k$ imply

$$v_k^T W_k G_k y_k \leq \underline{\eta} \|G_k y_k\|_{W_k}^2. \tag{3.36}$$

Moreover, with $\omega > 0$ and $(\zeta, \mathcal{T})$ chosen as in Lemma 3.3.6, there exists $k''' \geq k'$ such that $x_k \in \mathbb{B}_\epsilon(x')$ with $\epsilon = \min\{\zeta, \epsilon'/3\}$ for all $k \geq k'''$ and

$$\|G_k y_k\|_{W_k}^2 = \|P_{W_k}(\{\nabla f(x)\}_{x \in X_k})\|_{W_k}^2 \leq \|P_{W_k}(\mathcal{G}_k(x'))\|_{W_k}^2 + \omega$$

whenever $k \geq k'''$, $k \in \mathcal{K}_p$, and $X_k \in \mathcal{T}$; hence, by Lemma 3.3.5, for such k we have

$$v^T W_k G_k y_k > \underline{\eta} \|G_k y_k\|_{W_k}^2 \text{ for all } v \in \mathcal{G}_k(x'). \tag{3.37}$$

Together, (3.36) and (3.37) imply that $v_k \notin \mathcal{G}_k(x')$ whenever $k \geq k'''$, $k \in \mathcal{K}_p$, and $X_k \in \mathcal{T}$. However, from the facts that $d_k = -W_k G_k y_k$ and $e^T y_k = 1$ (recall (3.5)), (3.30), Assumption 3.2.1, and [15, Proposition 2.1.2], we have for all $k \geq k'''$ that

$$\|d_k\|_2 = \|W_k G_k y_k\|_2 \leq \|W_k\|_2 \|G_k y_k\|_2 \leq \xi^{-1} L_{\mathbb{B}_\epsilon(x')},$$

where $L_{\mathbb{B}_\epsilon(x')} \geq 0$ is the Lipschitz constant for f over $\mathbb{B}_\epsilon(x')$; see the similar result [39, Lemma 4.1]. That is, $\{\|d_k\|_2\}$ is bounded for $k \geq k'''$. This, along with the fact that $\alpha_k \to 0$, implies that $\alpha_k \leq \gamma\epsilon'/(3\|d_k\|_2)$ for all sufficiently large k, i.e., $\gamma^{-1}\alpha_k\|d_k\|_2 \leq \epsilon'/3$ for all sufficiently large k. Combining this with the fact that $x_k \in \mathbb{B}_\epsilon(x')$ with $\epsilon = \min\{\zeta, \epsilon'/3\}$ implies $\|x_k - x'\| \leq \epsilon'/3$, we obtain that $\tilde{x}_k \in \mathbb{B}_{2\epsilon/3}(x')$ and hence $v_k \in \mathcal{G}_k(x')$ for all sufficiently large $k \geq k'''$. Overall, since $v_k \notin \mathcal{G}_k(x')$ whenever $k \geq k'''$, $k \in \mathcal{K}_p$, and $X_k \in \mathcal{T}$, yet $v_k \in \mathcal{G}_k(x')$ for all sufficiently large $k \geq k'''$, it follows that $X_k \notin \mathcal{T}$ for all sufficiently large $k \geq k'''$ with $k \in \mathcal{K}_p$. However, since $|\mathcal{K}_p| = \infty$, it follows as in the situation in Subcase 2.a that this is a probability zero event.

We have proved that the situations in Subcases 2.a and 2.b have probability zero, which implies that the event that there exists k' such that $\epsilon_k = \epsilon' > 0$ for all $k \geq k'$ has probability zero. This result and the proof of Case 1 shows that $\{\epsilon_k\} \to 0$ with probability one, as desired.

All that remains is to show that when $\{\epsilon_k\} \to 0$, all cluster points of $\{x_k\}$ are stationary for f. The proof is exactly that of [18, Theorem 4.2, Case 2]. $\square$

3.4 Implementation and Numerical Experiments

In this section, we describe a C++ implementation of our algorithm along with the results of numerical experiments that we performed to compare our code against other available software for solving problem (3.1). All of our experiments were performed on a machine running Debian 2.6.32 with two 8-Core AMD Opteron 6128 2.0GHz processors and 32GB of RAM.

3.4.1 An Implementation and Alternative Software

Hereafter, we refer to our implementation of Algorithm 5, along with all the subroutines described as Algorithms 6, 7, 8, and 9, as BFGS-GS. For convenience, BFGS-GS utilizes the linear algebra library ARMADILLO (version 4.300.0) [54]. A critical part of the implementation is the method for solving the QP (3.5), for which we implemented a specialized active set solver adapted from that proposed in [38]; further details for a similar Matlab implementation are discussed in [18, Appendix].

Recalling Table 3.1, the values of the input parameters used in our implementation are given in Table 3.2. (The format is consistent with that of Table 3.1 for ease of reference.) The only exception is that we do not set a value for the parameter J—i.e., the iteration threshold for iterate perturbation—since, in BFGS-GS, we do not check whether the iterates or sample points lie in the set $\mathcal{D}$. That is, at all steps in the algorithm (and subroutines) where one would normally check for a point's inclusion in $\mathcal{D}$, BFGS-GS determines that the point is indeed included. At such points, BFGS-GS assumes that a (sub)gradient of f is provided. Such an approach was also employed in the gradient sampling algorithms in [10, 17, 18], where it was argued—as we claim here in terms of our experiments—that, due to the presence of a GS strategy, this is a reasonable approach for practically handling nondifferentiability of f at certain points. We remark that our choice of a sample set size threshold of $p = 100$ was based on the fact that this value worked well in our tests, which all involved $n \approx 50$; see §3.4.2. We also remark that our model curvature threshold ξ does not satisfy the upper bound in (3.26); instead, we chose a relatively large value that worked well in our experiments.

Table 3.2: Summary of input parameters for algorithm BFGS-GS.

Parameter(s)	Value(s)	Description
ν	1	Stationarity measure tolerance
ψ	0.5	Sampling radius reduction factor
ξ	10^{-4}	Model curvature threshold
$\underline{\eta} < \overline{\eta}$	$10^{-8} < 0.9$	Armijo–Wolfe line search constants
$\underline{\alpha} \leq \overline{\alpha}$	$10^{-4} \leq 1$	Step size thresholds
γ	0.5	Step size modification factor
$\underline{J} \leq \overline{J}$	$5 \leq 10$	Iteration thresholds for line search
p	100	Sample set size threshold
$\underline{\mu} < 1 < \overline{\mu}$	$0.2 < 1 < 100$	(L-)BFGS updating thresholds
$\underline{w} \leq \overline{w}$	$10^{-4} \leq 1$	(L-)BFGS updating thresholds
m	100	L-BFGS memory length

We also use the following input parameters for BFGS-GS. We set the initial sampling radius to $\epsilon_0 \leftarrow 0.1$ as this value generally worked well in our experiments. For the QP solver for subproblem (3.5), we set an optimality tolerance of 10^{-8} and a maximum iteration limit of 10^3; i.e., the QP solver terminates once the ℓ_∞-norm of the residual of the KKT conditions (recall (3.7)) is less than this tolerance or the iteration counter exceeds this limit. (Regardless of the reason for termination of the QP solver, BFGS-GS uses the search direction yielded by the final QP solver iterate; i.e., BFGS-GS does not terminate if the QP solver fails to provide an accurate solution as determined by the optimality tolerance.) For Algorithm 8, we found a good choice to be $\overline{p}_{k+1} \leftarrow 5$ for all k. The initial inverse Hessian approximation corresponding to $k = 0$ is set as $W_k \leftarrow w_k I$, where the scalar w_k

is set as

$$w_k \leftarrow \frac{1}{\max\{1, \min\{10^4, \|\nabla f(x_k)\|_2\}\}}.$$

This is also the value for w_k employed in the L-BFGS strategy in Algorithm 9. Finally, BFGS-GS terminates when the iteration counter k exceeds 10^4 or when

$$\epsilon_k \leq \epsilon_f, \tag{3.38a}$$

$$\|G_k y_k\|_{W_k} \leq \epsilon_f, \tag{3.38b}$$

$$\|G_k y_k\|_{W_k} \geq \xi \|d_k\|_2, \tag{3.38c}$$

$$\text{and} \quad \alpha_k > 0, \tag{3.38d}$$

for some constant $\epsilon_f > 0$. (In our tests below, we consider $\epsilon_f \in \{10^{-4}, 10^{-6}\}$.) Reminiscent of (3.6) and (2.6′), these criteria require—recalling that Lemma 3.2.2 implies $\|G_k y_k\|_{W_k} = \|d_k\|_{W_k^{-1}}$—that the sampling radius has already been reduced to a sufficiently small value and the current step is sufficiently small while the curvature of the current Hessian approximation is sufficiently positive along d_k.

For comparison purposes, we ran implementations of three other algorithms for our numerical experiments. The first two are variants of the software available at [50], which we refer to as HANSO-BFGS and HANSO-DEFAULT. The former solver (obtained by setting `options.samprad = []`) employs a standard BFGS method with a weak Wolfe line search (see [44]), whereas the latter solver (obtained by leaving `options.samprad` at its default value) runs the same approach followed by the application of a GS method (as it is proposed in [10]) to obtain an improved solution. Despite the fact that these solvers are implemented in Matlab while BFGS-GS is implemented in C++, we believe our comparisons are appropriate, at least since we focus on performance measures other than CPU time (in terms of which one would expect a Matlab implementation to have a disadvantage). In particular, our method represents a technique for incorporating a GS strategy while optimizing with a BFGS-type approach, whereas HANSO-BFGS exhibits the behavior of a BFGS method (with no safeguarding for handling nonsmoothness) and HANSO-DEFAULT exhibits the behavior of an algorithm that switches from BFGS to a GS method. It is worthwhile to note that by switching to a GS method, HANSO-DEFAULT has theoretical convergence guarantees that are similar to the algorithm proposed in this paper, whereas the BFGS algorithm in HANSO-BFGS only has the convergence guarantees provided in [44], which are limited to only a few types of simple problems representing a small subset of the test set that we consider.

The third algorithm to which we compare our code is the Fortran 77 solver available (along with a mex-driver for Matlab users, which we used) at [36], which is an implementation of the limited memory bundle method proposed in [30, 31]. We refer to this solver as LMBM, and include it in our experiments so as to illustrate the performance of our approach compared to an alternative quasi-Newton method for solving nonconvex, nonsmooth optimization problems.

The input parameters for HANSO-BFGS, HANSO-DEFAULT, and LMBM (besides `options.samprad = []` for HANSO-BFGS) are left at their default values, except that we set maximum iteration and CPU time limits on par with that chosen for BFGS-GS. In particular, for HANSO-BFGS, we changed the maximum number of (BFGS) iterations to `options.maxit = 1e+4`. This value was also used for HANSO-DEFAULT, but it should be noted that, once its BFGS method terminates, HANSO-DEFAULT may do as many as 300 GS iterations—meaning that we allowed HANSO-DEFAULT to perform as many as 10^4 (BFGS) + 300 (GS) iterations. For LMBM, we changed the maximum number of iterations to `IPAR(2) = 1e+4` and set the maximum time limit to a large enough number that the solver never terminated due to a time limit in our tests. Overall, none of the solvers that we tested had a CPU time limit that led to termination in any of our experiments.

3.4.2 Test Problems

For our numerical experiments, we measured the performance of all algorithms on 26 nonsmooth minimization problems, some convex and some nonconvex. The first 20 of these problems were considered in [30] and the last six were considered in [56]. All problems are scalable in the sense that they can be defined to have different numbers of variables. The first ten problems—called `MAXQ`, `MXHILB`, `CHAINED_LQ`, CHAINED_CB3_I, CHAINED_CB3_II, ACTIVE_FACES, BROWN_FUNCTION_2, CHAINED_MIFFLIN_2, CHAINED_CRESCENT_I, and CHAINED_CRESCENT_II—can also be found in [31] and are all nonsmooth at their respective minimizers. The first five of these problems are convex and the remaining five are nonconvex. The next ten problems in our set, some of which are nonconvex, were introduced in the library TEST29 [45]. They are called TEST29_2, TEST29_5, TEST29_6, TEST29_11, TEST29_13, TEST29_17, TEST29_19, TEST29_20, TEST29_22, and TEST29_24. Of the six remaining problems in our set, the first four were introduced in [44], the fifth was introduced in [26], and the sixth is a problem to minimize the Schatten norm [56]. These problems are referred to as TILTED_NORM_COND, CPSF, NCPSF, EIG_PROD, GREIF_FUN, and NUC_NORM.

We chose $n = 50$ for all problems, except for the case of EIG_PROD that requires the number of variables to be the square of an integer, for which we choose $n = 64$. We ran each problem ten times with ten different starting points. For the first 20 problems, the first run was performed with the initial point x_0 stated in [30] while for the remaining nine runs we used a starting point that was randomly generated from a Euclidean ball about x_0 with radius $\|x_0\|_2$. (We remark that the initial points in [30] satisfy $x_0 \neq 0$ and that the initial points for each run were unique.) For the remaining six problems, we chose the initial point as a randomly generated point from a Euclidean ball about e with radius $\|e\|_2$.

The last six problems in our test set require input parameters; see [56]. Problems TILTED_NORM_COND, CPSF, and NCPSF require symmetric positive definite matrices with a specified condition number. To generate these, we used Matlab's built-in sprandsym function. Similarly, problem NUC_NORM requires an input matrix and vector, which we generated using Matlab's built-in randn function. For the matrix required in EIG_PROD, we used the leading 8×8 submatrix of A from [1]; see also the experiments in [10, 17]. For GREIF_FUN, we multiplied the transpose of a 10×10 matrix randomly generated by randn with the matrix itself to create a symmetric positive definite matrix A so that the $n = 50$ variables composing the 10×5 variable matrix X has the well defined sum $A + XX^T$.

We personally implemented all of the test problems in C++ for use by BFGS-GS. For the remaining solvers, we personally implemented the first 20 test problems in Matlab and obtained the remaining six from the website of [56].

3.4.3 Numerical Results

The purpose of presenting the results of our numerical experiments is to illustrate the efficiency and reliability of our BFGS-GS solver in comparison to HANSO-BFGS, HANSO-DEFAULT, and LMBM (with their default parameter settings) when run on the $26 \times 10 = 260$ problems in our test set. (That is, we tested our 26 problem formulations, each run with ten different starting points.) Since the codes are written in various languages and were run in different environments (i.e., compiled C++ code versus Matlab), we ignore CPU time and focus on the performance measures of iterations, function evaluations, and gradient evaluations required until termination. Despite the fact that we ignore CPU time, we claim that the per-iteration costs of the algorithms underlying BFGS-GS, HANSO-DEFAULT, and LMBM are all relatively similar—especially when averaged over all iterations that may be performed—so by being successful in terms of the performance measures that we consider, we claim that one should expect success in terms of CPU time if all codes were implemented in the same language and run in the same environment. By contrast, the average per-iteration cost of HANSO-BFGS is typically less than all other solvers (at least when ignoring computations performed to evaluate a stationarity measure). However, due to the fact that it is based on an algorithm that lacks theoretical convergence guarantees, one would expect HANSO-BFGS to be less reliable (at least when compared to the related method in HANSO-DEFAULT). Indeed, this is evident in our numerical results presented in this section.

When running our experiments with HANSO-BFGS, HANSO-DEFAULT, and LMBM, we observed that the default settings of these codes resulted in markedly different performance. In particular, the default settings of LMBM led to runs that terminated after many fewer iterations, function evaluations, and gradient evaluations as compared to HANSO-BFGS and HANSO-DEFAULT. However, when comparing solvers for nonsmooth optimization problems, one should not necessarily rely upon the termination conditions employed in a given implementation to have a sense of the quality of the provided solutions. As opposed to smooth optimization where one can simply observe the magnitude of the objective function gradient at the final iterate, stationarity measures for nonsmooth problems require information about the subdifferential (or ϵ-subdifferential) of the objective at the final iterate, which often can only be approximated. Hence, rather than focus solely on the performance measures mentioned above, we investigated further and found that the performance of LMBM as compared to HANSO-BFGS and HANSO-DEFAULT was not as good when considering a measure of quality of the provided solutions. (We define our quality measure later in this section.) Based on these observations, we could have adjusted the input parameters for all of the codes in order to ensure that solutions of similar quality were found before a given code was allowed to terminate. However, we found this to be difficult due to the numerous termination conditions employed in the codes; some are based on stationarity measures, but others are based on changes in the function values, failure to compute a direction of strict descent, etc. Hence, instead, we decided to leave the default inputs for these solvers, but present results for two separate runs of our code: one with the stationarity tolerance of $\epsilon_f \leftarrow 10^{-4}$ in (3.38) and one with $\epsilon_f \leftarrow 10^{-6}$. We show that with the former setting, our code—BFGS-GS(10^{-4})—was able to obtain solutions of similar quality as those obtained by LMBM, and could generally do so with fewer iterations, function evaluations, and gradient evaluations. On the other hand, with the latter setting, our code—BFGS-GS(10^{-6})—continued on to obtain solutions that often had similar quality as those obtained by HANSO-BFGS and HANSO-DEFAULT. Overall, our goal in presenting two sets of results for our code is to demonstrate the versatility of our software; it can quickly obtain solutions of reasonable quality, and, when desired, it can be forced to continue to obtain higher solution accuracy.

Table 3.3, below, summarizes the termination flags returned by all of the codes for all of the problems in our tests. We group the flags into three types:

(1) a stationarity measure tolerance was satisfied,

(2) the maximum iteration limit was reached, and

(3) other.

As previously mentioned, termination flags of the last type indicate termination based on various occurrences such as small changes in the objective, a failure to compute a direction of strict descent, etc. Overall, Table 3.3 reveals that with both termination tolerances our code was very successful in satisfying our termination criteria (3.38), whereas the other codes often terminated due to other reasons.

Flag	BFGS-GS(10^{-4})	BFGS-GS(10^{-6})	HANSO-BFGS	HANSO-DEFAULT	LMBM
(1)	253	229	68	68	20
(2)	7	31	31	19	0
(3)	0	0	161	173	240

Table 3.3: Counts of termination flag types

Next, we illustrate the performance of the algorithms in terms of iterations, function evaluations, and gradient evaluations via profiles in the style of Dolan and Moré [19]. Typically, when preparing such profiles, it is incumbent upon the user to decide when a particular run should be considered successful or unsuccessful. In our experiments, making this distinction was a difficult task due to the various termination flags returned by HANSO-BFGS, HANSO-DEFAULT, and LMBM. Indeed, if we only considered a termination flag of type (1) to be the indicator for a successful run, then the profiles would be skewed in favor of the codes that yielded such a flag most often (namely, ours), even though we often found that other runs also yielded good quality solutions (as we show later in this section). Hence, having presented the counts for the termination types in Table 3.3 above, we present performance profiles considering all runs by all codes to be successful. Despite the fact that this means, e.g., that a termination flag of type (2) is not considered a failure, we believe that the profiles are still meaningful since, for one thing, our iteration limit (namely, 10^4) was quite large; this means that if a code performed the maximum number of iterations, then this had an adverse affect for the code in the profile, as it would if such a run were considered a failure.

Figures 3.1, 3.2, and 3.3 are the performance profiles we obtained in terms of iterations, function evaluations, and gradient evaluations, respectively. Based on these profiles, we have a few observations, all of which should be considered along with the results in

Table 3.3 and the solution quality measures that we present later (see Table 3.4) to obtain a complete picture of the results of our experiments. First, the profiles reveal the observation that we made earlier about LMBM typically terminating after performing fewer iterations, function evaluations, and gradient evaluations as compared to HANSO-BFGS and HANSO-DEFAULT. (Recall that this motivated us to present two sets of results for our algorithm with different termination tolerances.) Second, the profiles reveal that BFGS-GS(10^{-4}) often outperforms LMBM in terms of all three measures; this is most interesting when one observes that these methods often obtained solutions of similar quality, as we show later. Third, the profiles reveal that BFGS-GS(10^{-6}) is more similar to HANSO-BFGS and HANSO-DEFAULT in terms of all three measures than is BFGS-GS(10^{-4}), so—in terms of our code and the solution quality results shown later—it is reasonable to compare the results of BFGS-GS(10^{-6}) to HANSO-BFGS and HANSO-DEFAULT.

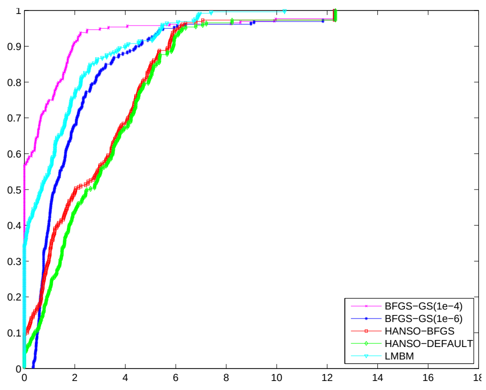


Figure 3.1: Performance profile for iterations

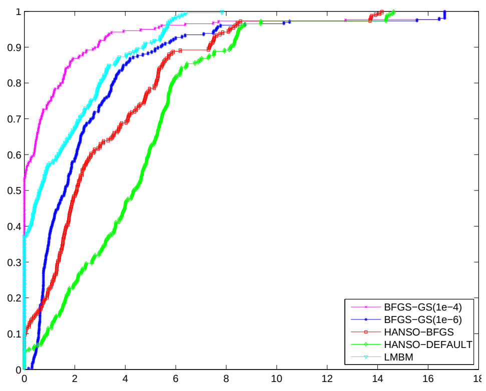


Figure 3.2: Performance profile for function evaluations

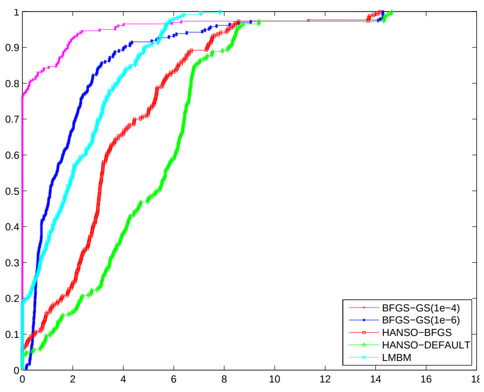


Figure 3.3: Performance profile for gradient evaluations

We are now prepared to consider the results of our experiments in terms of the quality of the provided solutions. For this purpose, we collected the final iterates provided by all of the codes on all of the problems in our test set. For each final iterate, say x_f, we randomly generated 10^3 points from a uniform distribution defined in a Euclidean 10^{-2}-ball about x_f. Then, using a Matlab implementation of our QP solver, we computed the minimum Euclidean norm element of the convex hull of the gradients of the objective evaluated at these points. The norm of this minimum norm vector represents a reasonable approximation of 10^{-2}-stationarity of x_f with respect to f. (As previously mentioned in §3.1, this type of measure was employed in [44] as a certificate of stationarity, except that, in that article, the authors employed iterates generated in the algorithm as opposed to randomly generated ones. We could have used iterates in this way as well, but we believe that by randomly generating the points—independent of the algorithm iterates—we obtain a fairer measure for comparing solution quality for the different codes.)

For each solver and each of the 26 problems in our original test set, Table 3.4 provides the geometric means of the norms of the minimum Euclidean norm vectors (as described in the previous paragraph) for the ten runs for each problem. We use geometric means as opposed to arithmetic means so that each mean is not skewed by one (or a few) large terms. Overall, one can see that, for all codes, results can vary in terms of this measurement of solution quality. All of the solvers are competitive, though, broadly speaking, the quality of the solutions provided by BFGS-GS(10^{-4}) and LMBM are not as good as those provided by BFGS-GS(10^{-6}), HANSO-BFGS, and HANSO-DEFAULT. In terms of HANSO-BFGS and HANSO-

DEFAULT, we believe that the improved solution quality is due to the termination criteria employed in the software. In particular, these algorithms check for stationarity in a similar way that we measure it here: they compute the minimum Euclidean norm element in the convex hull of gradients evaluated around a given iterate. By contrast, BFGS-GS (with both tolerances) and LMBM have termination criteria that are influenced by the employed quasi-Newton Hessian approximations. Due to this fact, we could include in BFGS-GS an extra step to measure stationarity using a Euclidean norm measure, but we chose not to do this in order to avoid extra computational expense (of perhaps generating additional sample points and solving a large QP) in our software. We feel that this is appropriate since, with its tightened stationarity tolerance, BFGS-GS(10^{-6}) is able to obtain solutions of similar quality as those yielded by HANSO-BFGS and HANSO-DEFAULT. (That being said, there are cases where BFGS-GS(10^{-6}) yields better or worse solutions. For example, for a few starting points, our solver performs poorly on problem 20.)

3.5 Conclusion

We have proposed an algorithm for solving nonconvex, nonsmooth optimization problems. The main features of the algorithm are that it typically behaves as an unadulterated BFGS method—and, hence, it often has very low per-iteration computational costs—but dynamically incorporates gradient sampling to ensure progress toward stationarity. We have proved that the algorithm has global convergence guarantees with probability one, and, on a set of test problems, we have shown that an implementation of it is competitive with—and in some ways outperforms—other available software for solving such problems.

We close this article by remarking that while the theoretical convergence guarantees of our algorithm in some cases rely on an L-BFGS strategy that ensures sufficiently positive definite and bounded Hessian approximations, one can consider a variant of our algorithm that allows these matrices to approach singularity and tend to unboundedness as $\{\epsilon_k\} \to 0$ while preserving our convergence guarantees. In particular, our convergence guarantees rely on the fact that for a given sampling radius, the method eventually satisfies our conditions for reducing this radius with probability one. Hence, one could allow our model

	BFGS-GS(10^{-4})	BFGS-GS(10^{-6})	HANSO-BFGS	HANSO-DEFAULT	LMBM
1	5.5115e-02	3.2624e-03	1.0931e-14	1.0931e-14	2.9769e-14
2	2.6008e-06	4.0027e-12	2.0981e-14	2.0981e-14	7.4413e-11
3	1.0032e-01	7.9324e-03	1.9953e-01	1.9953e-01	3.6743e-01
4	6.6657e-15	8.0674e-15	6.5293e-15	6.7015e-15	1.1089e-04
5	5.1371e-02	1.4784e-11	1.4116e-11	1.4116e-11	1.5361e-09
6	1.5343e-01	1.5343e-01	2.5912e-16	2.5912e-16	0.0000e+00
7	2.7203e-15	2.3324e-15	2.3766e-15	2.3766e-15	3.9072e-02
8	4.4343e+00	8.8031e-01	5.6539e+00	5.6539e+00	3.5372e+00
9	7.2550e-03	4.7572e-11	9.7894e-12	9.7894e-12	5.0344e-10
10	2.1219e+00	2.3921e+00	2.4665e+00	2.3562e+00	2.2678e+00
11	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	5.3619e-01
12	1.2268e-09	1.4630e-06	1.4011e-06	1.4011e-06	4.9237e-08
13	1.2418e-06	3.1166e-07	3.3843e-03	2.8826e-03	1.4463e-02
14	2.2987e+01	2.5143e+01	2.5657e+01	2.2958e+01	2.0717e+01
15	1.3441e+01	1.2830e+01	2.1905e+02	1.9625e+02	9.4820e-01
16	3.4085e-16	2.4520e-16	9.5007e-15	3.0778e-16	1.4537e-11
17	1.7747e-01	4.2699e-03	2.4028e-03	1.1057e-03	5.3772e-01
18	3.8821e-09	3.1459e-10	1.0375e-06	6.3327e-07	2.1108e-01
19	1.9198e-11	9.5458e-13	1.5087e-13	1.5087e-13	1.7270e-02
20	1.6003e+07	1.1290e+07	1.4061e+00	4.4723e-01	4.4147e+09
21	6.2899e-03	1.4594e-06	1.6764e-06	1.6764e-06	5.4393e-06
22	7.4602e-03	7.2946e-04	1.7976e-06	1.7976e-06	1.5031e-02
23	1.2722e-03	8.8772e-05	6.6831e-07	6.6831e-07	4.9783e-02
24	2.9884e-01	1.1171e-02	4.7271e-09	4.7271e-09	3.5170e-02
25	2.5222e-02	3.6621e-05	4.6504e-07	4.6504e-07	1.0049e-05
26	6.6878e-03	9.3793e-06	1.1201e-06	1.1201e-06	1.4454e-05

Table 3.4: For each solver and each test problem, the geometric means of stationarity measures

curvature threshold and lower (L-)BFGS updating threshold, namely ξ and $\underline{\mu}$, to decrease to zero along with the sampling radius (while respecting the requirements in Table 3.1 and (3.26)) and the upper (L-)BFGS updating threshold, namely $\overline{\mu}$, to correspondingly increase to ∞. Our theoretical convergence guarantees hold as long as these parameters remain fixed until the sampling radius is reduced. However, we decided not to propose this variant in the paper since it would require more complicated conditions and a slightly more complicated analysis, and we did not see any benefits of such a strategy in any numerical experiments that we performed.

Chapter 4

Algorithmic Extensions

In this chapter, we propose several algorithmic extensions of the gradient sampling (GS) framework. In Section 4.1, we propose a bundle gradient sampling (BGS) algorithm, which is a hybrid of the bundle method and the previously proposed adaptive gradient sampling (AGS) algorithm. In Section 4.2, we propose a smoothing BFGS gradient sampling algorithm, which is based on the smoothing method and the BFGS gradient sampling (BFGS-GS) algorithm proposed in the previous chapter. In Section 4.3, we tailor GS methods for solving regularization problems. For all the algorithmic extensions proposed in this chapter, global convergence analysis is provided; and numerical results are presented to illustrate the performance of the algorithms.

4.1 A Bundle Gradient Sampling Algorithm

In order to get a deeper understanding of the similarities and differences between BM and AGS, in this section we write the QO subproblems of BM and AGS in similar forms.

At iteration k of BM, suppose the previous iteration is a descent step (i.e., $w_k = x_k$). Moreover, let the search direction d be defined as a step from the current iterate $w_k = x_k$; that is, let $d = x - w_k = x - x_k$ in (1.14). Then, instead of minimizing over z and the next iterate x, here we are minimizing over z and the search direction d. The regularized master problem (1.14) can then be written as

$$\begin{aligned}&\min_{z,d}\ z + \tfrac{1}{2}\|d\|^2\\ &\text{s.t. } l_k + G_k^T d \leq ze.\end{aligned} \tag{4.1}$$

Here, G_k is defined as the matrix whose columns consist of the (sub)gradients of f at the points in $X_k := \{x_j : j \in J_k\}$ (where J_k is a subset of indices from previous iterates) and l_k is a vector with elements

$$l_{k,j} := f(x_j) + g_j^T(x_k - x_j),\ j \in J_k.$$

The dual of (4.1) has the following form:

$$\begin{aligned}&\max_{\pi}\ l_k^T\pi - \tfrac{1}{2}\|G_k\pi\|^2\\ &\text{s.t. } e^T\pi = 1,\ \pi \geq 0.\end{aligned} \tag{4.2}$$

Observing (4.1) and (4.2) and comparing their forms to that of (2.3) and (2.4) in AGS, we arrive at the following generic primal-dual QO subproblems:

$$(\text{P}) := \left\{\begin{aligned}&\min_{z,d}\ z + \tfrac{1}{2}d^T H_k d\\ &\text{s.t. } \xi_k + G_k^T d \leq ze\end{aligned}\right\} \qquad (\text{D}) := \left\{\begin{aligned}&\max_{\pi}\ \xi_k^T\pi - \tfrac{1}{2}\pi^T G_k^T W_k G_k \pi\\ &\text{s.t. } e^T\pi = 1,\ \pi \geq 0.\end{aligned}\right\}$$

Here, the similarities and differences between BM and AGS are apparent. With $\xi_k = l_k$ and $H_k = I$, we arrive at the QO subproblems for BM, whereas with $\xi_k = f(x_k)e$ we arrive at the QO subproblems for AGS. For convex functions, the constraints in the primal subproblem have the nice interpretation as cutting planes, i.e., local linear underestimators of the objective f. However, for nonconvex problems, a benefit in the AGS formulation is that the subproblem remains well-defined and leads to productive search directions.

An important point to make here is that with our extensions from GS to AGS, the sample set X_k in AGS has similar properties to that in BM. Specifically, X_k in AGS contains points obtained during previous iterations, whereas X_k in GS does not. We have added restrictions in AGS that the points in X_k correspond to points within an ϵ_k-ball at which f is differentiable, but we believe that in practice these restrictions are not as significant as the difference resulting from the differing choices of the vector ξ_k.

Consider the primal form of the subproblems of GS and BM, the only difference comes from the constant term in the constraints. In GS we have the constant term $f(x_k)$, while in BM we have the constant term $f(x_j)+g_j^T(x_k-x_j)$. This difference makes GS an approach that can handle both convex and nonconvex problems, and makes BM an efficient method for convex problems. An intuitive option is to consider both of the constant terms in the subproblem. In particular, we propose the following form of the subproblem:

$$\begin{aligned} &\min_{z,d}\ z+\tfrac{1}{2}d^T H_k d \\ &\text{s.t. } \min\{f(x_k), f(x_j)+g_j^T(x_k-x_j)\}+g_j^T d \le z, j \in J_k. \end{aligned} \tag{4.3}$$

The following example in Figure 4.1 is used to illustrate the subproblem (4.3). At iteration k, suppose we have a set of points $\{x_k, x_{k1}, x_{k2}\}$, where $x_k = 4.5$, $x_{k1} = 6$ and $x_{k2} = 2.5$. Suppose we also have the values of the corresponding objective functions $\{f(x_k), f(x_{k1}), f(x_{k2})\}$ and (sub)gradients $\{g_k, g_{k1}, g_{k2}\}$. Suppose the current approximation of the Hessian is $H_k = 0.2$. Based on (4.3), we construct a subproblem as the following one:

$$\begin{aligned} &\min_{z,d}\ z+\tfrac{1}{2}d^T H_k d \\ &\text{s.t. } f(x_k)+g_k^T d \le z \\ &\qquad f(x_k)+g_{k1}^T d \le z \\ &\qquad f(x_{k2})+g_{k2}^T(x_k-x_{k2})+g_{k2}^T d \le z. \end{aligned} \tag{4.4}$$

In nonconvex regions (for example, in the neighborhood of x_{k1}), the cutting plane $f(x_{k1})+g_{k1}^T(x_k-x_{k1})+g_{k1}^T d$ is no longer a local linear underestimator of the objective f. In this case, (4.4) acts like GS since $f(x_k) < f(x_{k1})+g_{k1}^T(x_k-x_{k1})$. In convex regions (for example, in the neighborhood of x_{k2}), (4.4) acts like BM since $f(x_k) > f(x_{k2})+g_{k2}^T(x_k-x_{k2})$.

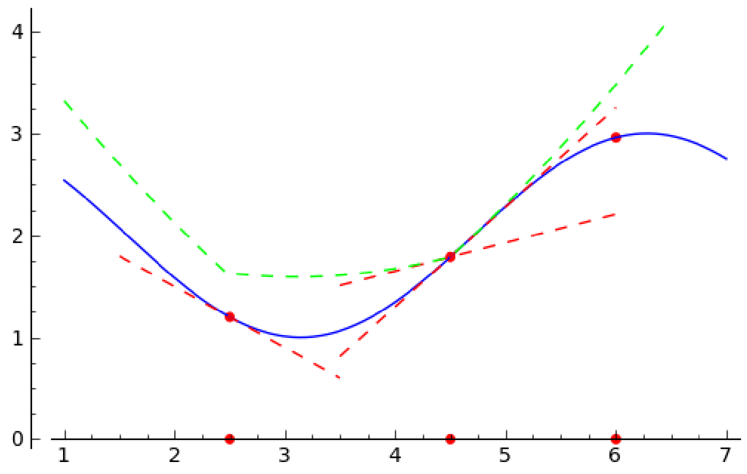


Figure 4.1: Illustration of bundle sampling method.

4.1.1 Algorithm Description

We now present a BGS algorithm of which AGS is a special case. At a given iterate $x_k \in \mathcal{D}$ with a given sampling radius $\epsilon_k > 0$, the sample ball $\mathbb{B}_{\epsilon_k}(x_k)$ is defined as the following Euclidean ball centered at x_k with radius ϵ_k: $\mathbb{B}_{\epsilon_k}(x_k) := \{x : \|x - x_k\| \leq \epsilon_k\}$. During iteration k, we generate the sample set

$$X_k := \{x_{k,0}, \ldots, x_{k,p_k}\}, \tag{4.5}$$

where $x_{k,0} := x_k$ and $x_{k,i}$ for $i = 1, \cdots, p_k$ are sampled uniformly and independently in $B_k := \mathbb{B}_{\epsilon_k}(x_k) \cap \mathcal{D}$, and then compute the gradient matrix

$$G_k := \begin{bmatrix} g_{k,0} & \cdots & g_{k,p_k} \end{bmatrix}, \tag{4.6}$$

where columns in G_k are the gradients of f at the points in X_k. Let $H_k \in \mathbb{R}^{n \times n}$ be a positive definite matrix (i.e., $H_k \succ 0$), and $W_k := H_k^{-1} \succ 0$. The main computational expense in an iteration of the method is to solve the following primal-dual QO subproblems to compute the search direction d_k:

$$(\mathrm{P}) := \left\{ \begin{array}{l} \min\limits_{z,d}\ z + \frac{1}{2} d^T H_k d \\ \text{s.t. } \xi_k + G_k^T d \le ze \end{array} \right\} \qquad (\mathrm{D}) := \left\{ \begin{array}{l} \max\limits_{\pi}\ \xi_k^T \pi - \frac{1}{2}\pi^T G_k^T W_k G_k \pi \\ \text{s.t. } e^T\pi = 1,\ \pi \ge 0. \end{array} \right\} \tag{4.7}$$

Here, ξ_k is a vector with elements

$$\xi_{k,i} := \min\{f(x_k), f(x_{k,i}) + g_{k,i}^T(x_k - x_{k,i})\}. \tag{4.8}$$

Note that either the primal or the dual (not both) needs to be solved; and the solution (z_k, d_k, π_k) of (4.7) has $d_k = -W_k G_k \pi_k$ and $z_k = \xi_k^T \pi - \pi_k^T G_k^T W_k G_k \pi_k$.

After the computation of the search direction d_k, a standard backtracking line search is performed to find a step size α_k satisfying the following sufficient decrease condition

$$f(x_k + \alpha_k d_k) \le f(x_k) - \eta \alpha_k d_k^T H_k d_k. \tag{4.9}$$

We set $x_{k+1} \leftarrow x_k + \alpha_k d_k$ if we have $x_k + \alpha_k d_k \in \mathcal{D}$; otherwise, in order to make sure that all iterates are differentiable, we perturb an $x_{k+1} \in \mathcal{D}$ satisfying the following perturbed line search conditions

$$\begin{aligned} f(x_{k+1}) \le&\ f(x_k) - \eta\alpha_k d_k^T H_k d_k && \text{(4.10a)} \\ \text{and } \|x_k + \alpha_k d_k - x_{k+1}\| \le&\ \min\{\alpha_k, \epsilon_k\}\|d_k\|. && \text{(4.10b)} \end{aligned}$$

See [9] and [39] for perturbation procedures that terminate finitely.

After computing the search direction d_k and the step size α_k, we need to test stationarity and update the sampling radius ϵ_k. At an ϵ-stationary point of AGS, we produce a small step for certain generated sample sets. However, at an ϵ-stationary point of BGS, we do not necessarily compute as small of a step. Therefore, we also solve the primal-dual QO subproblems in AGS and update the sampling radius ϵ_k by the solution. To distinguish the solution of the QO subproblems from AGS and that from BGS, we use (z_k^A, d_k^A, π_k^A) and (z_k^B, d_k^B, π_k^B) to denote the solutions, respectively.

The BGS algorithm is presented as Algorithm 10 below.

Algorithm 10 Bundle/Gradient Sampling (BGS) Algorithm

```
1: (Initialization): Choose a number of sample points to compute each iteration $\overline{p} \geq 1$, number of sample points required for a complete line search $p \geq n+1$, sampling radius reduction factor $\psi \in (0,1)$, number of backtracks for an incomplete line search $q \geq 0$, sufficient decrease constant $\eta \in (0,1)$, line search backtracking constant $\kappa \in (0,1)$, and tolerance parameter $\nu > 0$. Choose an initial iterate $x_0 \in \mathcal{D}$, set $X_{-1} \leftarrow \emptyset$, choose an initial sampling radius $\epsilon_0 > 0$, and set $k \leftarrow 0$.
2: (Sample set update): Set $X_k \leftarrow (X_{k-1} \cap B_k) \cup x_k \cup \overline{X}_k$, where the sample set $\overline{X}_k := \{\overline{x}_{k,1}, \ldots, \overline{x}_{k,\overline{p}}\}$ is composed of $\overline{p}$ points generated uniformly in $B_k$. Set $p_k \leftarrow |X_k| - 1$. If $p_k > p$, then remove the $p_k - p$ eldest members of $X_k$ and set $p_k \leftarrow p$. Compute any unknown columns of $G_k$ defined in (4.6).
3: (Hessian update): Set $H_k \succ 0$ and $W_k = H_k^{-1} \succ 0$, respectively, as approximations of the Hessian and inverse Hessian of $f$ at $x_k$.
4: (Search direction computation): Compute $(z_k^B, d_k^B, \pi_k^B)$ solving the QO subproblems (4.7) with $\xi_k$ defined as in (4.8).
5: (Sampling radius update): Compute $(z_k^A, d_k^A, \pi_k^A)$ solving the QO subproblems (4.7) with $\xi_k = f(x_k)e$ as in AGS. If $\min\{\|d_k^A\|^2, (d_k^A)^T H_k d_k^A\} \leq \nu\epsilon_k^2$, then set $x_{k+1} \leftarrow x_k$, $\alpha_k \leftarrow 1$, and $\epsilon_{k+1} \leftarrow \psi\epsilon_k$ and go to step 8.
6: (Backtracking line search): If $p_k < p$, then set $\alpha_k$ as the largest value in $\{\kappa^0, \kappa^1, \ldots, \kappa^q\}$ such that (4.38) is satisfied, or set $\alpha_k \leftarrow 0$ if (4.38) is not satisfied for any of these values of $\alpha_k$. If $p_k = p$, then set $\alpha_k$ as the largest value in $\{\kappa^0, \kappa^1, \kappa^2, \ldots\}$ such that (4.38) is satisfied.
7: (Iterate update): Set $\epsilon_{k+1} \leftarrow \epsilon_k$. If $x_k + \alpha_k d_k^B \in \mathcal{D}$, then set $x_{k+1} \leftarrow x_k + \alpha_k d_k^B$. Otherwise, set $x_{k+1}$ as any point in $\mathcal{D}$ satisfying (4.10).
8: (Iteration increment): Set $k \leftarrow k+1$ and go to step 2.
```

4.1.2 Global Convergence Analysis

We make the following assumption about the objective function f throughout our global convergence analysis.

Assumption 4.1.1. *The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz in $\mathbb{R}^n$ and continuously differentiable in an open dense subset $\mathcal{D} \subset \mathbb{R}^n$.*

We also make the following assumption about the Hessian approximation H_k throughout our global convergence analysis.

Assumption 4.1.2. *There exist $\overline{\xi} \geq \underline{\xi} > 0$ such that, for all k and $d \in \mathbb{R}^n$, we have*

$$\underline{\xi}\|d\|^2 \leq d^T H_k d \leq \overline{\xi}\|d\|^2.$$

The result we prove is the following.

Theorem 4.1.3. *BGS produces an infinite sequence of iterates $\{x_k\}$ and, with probability one, either $f(x_k) \to -\infty$ or $\{\epsilon_k\} \to 0$ and every cluster point of $\{x_k\}$ is stationary for f.*

We begin our analysis by showing that the search direction produced by solving the QO subproblem of BGS is a descent direction. This ensures that the backtracking line search is well defined, and BGS is well-posed in the sense that each iteration terminates finitely.

Lemma 4.1.4. *The search direction d_k^B produced by solving the QO subproblem* (4.7) *of BGS is a descent direction for f from x_k.*

Proof. The KKT conditions of (4.7) are

$$e^T \pi = 1, \;\; \pi \geq 0, \tag{4.11a}$$

$$ze + G_k^T W_k G_k \pi - \xi_k \geq 0, \tag{4.11b}$$

$$\pi^T (ze + G_k^T W_k G_k \pi - \xi_k) = 0, \tag{4.11c}$$

where z and $ze + G_k^T W_k G_k \pi - \xi_k$ are the Lagrange multipliers of the equality and inequality constraints of the dual form of (4.7).

Equation (4.11c) is equivalent to

$$\pi^T (ze + G_k^T W_k G_k \pi - \xi_k) = 0 \Leftrightarrow z = \pi^T \xi_k - \pi^T G_k^T W_k G_k \pi. \tag{4.12}$$

Plugging the expression of v from (4.12) into (4.11b), we have

$$G_k^T W_k G_k \pi - \xi_k + (\pi^T \xi_k - \pi^T G_k^T W_k G_k \pi) e \geq 0. \tag{4.13}$$

Since $d_k^B = -W_k G_k \pi_k^B$ and $\nabla f(x_k)$ is one column in G_k, we get

$$\nabla f(x_k)^T d_k^B \leq -f(x_k) + \pi^T \xi_k - \pi^T G_k^T W_k G_k \pi. \tag{4.14}$$

We know $\xi_{k,i} \leq f(x_k)$ for all i by the definition of ξ_k in (4.8), and so the convex combination of ξ_k is also no greater than $f(x_k)$, namely, $\pi^T \xi_k \leq f(x_k)$. Also, we have $G_k^T W_k G_k \succeq 0$ by Assumption 4.1.2. Therefore, we have

$$\nabla f(x_k)^T d_k^B < 0, \tag{4.15}$$

which means that the search direction produced by the QO subproblem of BGS is a descent direction.

□

Our next lemma shows that there exists an infinite number of iterations during which $\alpha_k > 0$.

Lemma 4.1.5. *There exists an infinite subsequence of iterations in which $\alpha_k > 0$.*

Proof. We refer to the proof of Lemma 4.4 in the AGS paper [18] for the proof here. It follows here since the assumptions about the objective function and search direction are the same as those in [18]. □

We now show a critical result about the sequence of decreases produced in f.

Lemma 4.1.6. *The following inequality holds for all k:*

$$f(x_{k+1}) \leq f(x_k) - \tfrac{1}{2}\eta\underline{\xi}\|x_{k+1} - x_k\|\|d_k^B\|.$$

Proof. We refer to the proof of Lemma 4.5 in the AGS paper [18] for the proof here. It also follows here since the assumptions about the objective function and search direction are the same as those in [18]. □

We now establish some properties of the set of sample gradients used to approximate the subdifferential. Consider the following subproblem, which is a variation of the subproblem defined in Section 2.4 of the AGS algorithm:

$$\inf_d \; q(d; x', \mathbb{B}_{\epsilon_k}(x'), H_k), \tag{4.16}$$

where

$$q(d; x', \mathbb{B}_{\epsilon_k}(x'), H_k) := \sup_{x \in \mathbb{B}_{\epsilon_k}(x') \cap \mathcal{D}} \{\min_x \{f(x'), f(x) + \nabla f(x)^T (x' - x)\} + \nabla f(x)^T d\} + \tfrac{1}{2} d^T H_k d.$$

Given a solution d' of (4.16), we have the following reduction in its objective:

$$\Delta q(d'; x', \mathbb{B}_{\epsilon_k}(x'), H_k) := q(0; x', \mathbb{B}_{\epsilon_k}(x'), H_k) - q(d'; x', \mathbb{B}_{\epsilon_k}(x'), H_k) \geq 0.$$

Similarly, writing (4.7) in the form

$$\min_d \; q(d; x_k, X_k, H_k),$$

where

$$q(d; x_k, X_k, H_k) := \max_{x \in X_k} \{\min_x \{f(x_k), f(x) + \nabla f(x)^T (x_k - x)\} + \nabla f(x)^T d\} + \tfrac{1}{2} d^T H_k d,$$

we have the following reduction produced by the search direction d_k:

$$\Delta q(d_k; x_k, X_k, H_k) = q(0; x_k, X_k, H_k) - q(d_k; x_k, X_k, H_k) \geq 0.$$

We now show a result about the above reduction. We use q^A and q^B to denote the subproblem objectives from AGS and that from BGS, respectively, to distinguish them.

Lemma 4.1.7. *Suppose we have the same X_k, G_k, H_k, W_k for the primal and dual QO subproblems of AGS and BGS. Then we have the following inequalities:*

$$(d_k^A)^T H_k d_k^A \leq (d_k^B)^T H_k d_k^B \leq (\sqrt{(d_k^A)^T H_k d_k^A} + D_k)^2 \tag{4.17a}$$

$$\Delta q^A(d_k^A; x_k, X_k, H_k) \leq \Delta q^B(d_k^B; x_k, X_k, H_k). \tag{4.17b}$$

Here, D_k is the diameter of the convex hull of the column vectors of the matrix $N_k G_k$, namely, $D_k = \sup\{\|x - y\| \;:\; x, y \in \operatorname{conv} \operatorname{col}(N_k G_k)\}$, where $\operatorname{col}(\cdot)$ denotes the set of column vectors of a matrix; and N_k is some matrix such that $W_k = N_k^T N_k$ (since $W_k \succ 0$).

Proof. We first prove inequality (4.17a). Consider the dual QO subproblem of AGS, which is a problem to find the vector with smallest norm in the convex hull of the column vectors of the matrix N_kG_k. Based on this interpretation, it clearly follows that $\|N_kG_k\pi_k^A\| \leq \|N_kG_k\pi_k^B\|$, namely $(d_k^A)^TH_kd_k^A \leq (d_k^B)^TH_kd_k^B$.

By the definition of D_k, we have $\|N_kG_k\pi_k^B - N_kG_k\pi_k^A\| \leq D_k$. Then, by the triangle inequality, we have $\|N_kG_k\pi_k^B\| \leq \|N_kG_k\pi_k^A\| + D_k$, namely, $(d_k^B)^TH_kd_k^B \leq (\sqrt{(d_k^A)^TH_kd_k^A} + D_k)^2$.

We now prove inequality (4.17b). By Lemma 2.4.6 in Chapter 2 of the AGS algorithm, we have $\Delta q^A(d_k^A; x_k, X_k, H_k) = \frac{1}{2}(d_k^A)^TH_kd_k^A$.

By the definition of the model q^B, we have $q^B(0; x_k, X_k, H_k) = \max_j\{\xi_{k,j}\} = f(x_k)$. Moreover, by (4.11c), we have $q^B(d_k^B; x_k, X_k, H_k) = z_k^B + \frac{1}{2}(d_k^B)^TH_kd_k^B = \xi_k^T\pi_k^B - \frac{1}{2}(d_k^B)^TH_kd_k^B$. Therefore, $\Delta q^B(d_k^B; x_k, X_k, H_k) = \frac{1}{2}(d_k^B)^TH_kd_k^B + f(x_k) - \xi_k^T\pi_k^B$.

We know $\xi_{k,i} \leq f(x_k)$ for all i by the definition of ξ_k in (4.8), and so the convex combination of ξ_k is also no greater than $f(x_k)$, namely, $\xi_k^T\pi_k^B \leq f(x_k)$. Also, we have $(d_k^B)^TH_kd_k^B \geq (d_k^A)^TH_kd_k^A$ by (4.17a), we then have $\Delta q^A(d_k^A; x_k, X_k, H_k) \leq \Delta q^B(d_k^B; x_k, X_k, H_k)$. □

For a given x' and tolerance ω, we define

$$\mathcal{T}_k(x', \omega) := \left\{ X_k \in \prod_0^{p_k} B_k : \Delta q^A(d_k^A; x_k, X_k, H_k) \leq \Delta q^A(d'; x', \mathbb{B}_{\epsilon_k}(x'), H_k) + \omega \right\}.$$

The purpose of the following lemma is to show that for an iterate x_k sufficiently close to x' and any tolerance $\omega > 0$, there exists a nonempty subset of $\mathcal{T}_k(x', \omega)$ if the sample set size $p_k \geq n + 1$; see the similar result [39, Lemma 3.2(i)].

Lemma 4.1.8. *If $p_k \geq n+1$, then for any $\omega > 0$, there exists $\zeta > 0$ and a nonempty set $\mathcal{T}$ such that for all $x_k \in B(x', \zeta)$ we have $\mathcal{T} \subseteq \mathcal{T}_k(x', \omega)$.*

We are now prepared to prove Theorem 4.1.3. Our proof follows closely that of [39, Theorem 3.3]. We provide a proof for the sake of completeness, and also because some changes to the proof are required due to the different QO subproblem used in BGS.

Proof. If $f(x_k) \to -\infty$, then there is nothing to prove, so suppose that

$$\inf_{k \to \infty} f(x_k) > -\infty.$$

Then, we have from (4.9), (4.10), and Lemma 4.1.6 that

$$\sum_{k=0}^{\infty} \alpha_k (d_k^B)^T H_k d_k^B < \infty, \quad \text{and} \tag{4.18a}$$

$$\sum_{k=0}^{\infty} \|x_{k+1} - x_k\| \|d_k^B\| < \infty. \tag{4.18b}$$

We continue by considering two cases, the first of which has two subcases.

Case 1: Suppose that there exists $k' \geq 0$ such that $\epsilon_k = \epsilon' > 0$ for all $k \geq k'$. According to step 5, this occurs only if

$$\min\{\|d_k^A\|^2, (d_k^A)^T H_k d_k^A\} > \nu\epsilon'^2 \text{ for all } k \geq k'. \tag{4.19}$$

Together with Lemma 4.1.7 and Assumption 4.1.2, we know that $\|d_k^B\|$ and $(d_k^B)^T H_k d_k^B$ are also bounded below for all $k \geq k'$.

In conjunction with (4.18), this implies $\alpha_k \to 0$ and $x_k \to x'$ for some x'. Moreover, the fact that $\alpha_k \to 0$ implies that there exists an infinite subsequence of iterations in which $p_k = p$. A similar argument is made in [18, Theorem 4.2]. Therefore, we can define $\mathcal{K}$ as the subsequence of iterations in which $p_k = p$ and know that $\mathcal{K}$ is infinite.

Case 1a: If x' is ϵ'-stationary for f, then for any $H_k \succ 0$, the solution d' to the subproblem q^A of (2.27) in AGS satisfies $\Delta q^A(d'; x', \mathbb{B}_{\epsilon'}(x'), H_k) = 0$. Thus, with $\omega = \nu\epsilon'^2/2$ and $(\zeta, \mathcal{T})$ chosen as in Lemma 4.1.8, there exists $k'' \geq k'$ such that $x_k \in \mathbb{B}_\zeta(x')$ for all $k \geq k''$ and

$$\tfrac{1}{2}(d_k^A)^T H_k d_k^A = \Delta q^A(d_k^A; x_k, X_k, H_k) \leq \tfrac{1}{2}\nu\epsilon'^2 \tag{4.20}$$

whenever $k \geq k''$, $k \in \mathcal{K}$, and $X_k \in \mathcal{T}$. Together, (4.19) and (4.20) imply that $X_k \notin \mathcal{T}$ for all $k \geq k''$ with $k \in \mathcal{K}$. However, this is a probability zero event since for all such k the set X_k continually collects points generated uniformly from B_k, meaning that it will eventually include an element of the set $\mathcal{T}$ yielding (4.20).

Case 1b: If x' is not ϵ'-stationary, then for all $k \geq k'$, any α not satisfying the sufficient decrease condition (4.9) yields

$$f(x_k + \alpha d_k^B) - f(x_k) > -\eta\alpha(d_k^B)^T H_k d_k^B,$$

and along with (4.14) yields

$$f(x_k + \alpha d_k^B) - f(x_k) \leq -\alpha(d_k^B)^T H_k d_k^B + \alpha^2 L_k \|d_k^B\|^2.$$

Here, L_k is a finite upper bound for $(f'(x_k + \alpha d_k^B) - f'(x_k))/(\alpha\|d_k^B\|)$ on the interval $[x_k, x_k + \alpha d_k^B]$ whose existence follows from Assumption 4.1.1. Combining the above inequalities yields a lower bound on any α not satisfying (4.9), which, since step 6 invokes the backtracking factor κ, yields the bound

$$\alpha_k > \kappa(1-\eta)(d_k^B)^T H_k d_k^B / (L_k \|d_k^B\|^2).$$

However, with $\omega = \Delta q^A(d'; x', \mathbb{B}_{\epsilon'}(x'), H_k)$ (which is strictly positive since x' is not ϵ'-stationary) and $(\zeta, \mathcal{T})$ again chosen as in Lemma 4.1.8, there exists $k'' \geq k'$ such that $x_k \in \mathbb{B}_\zeta(x')$ for all $k \geq k''$ and

$$\Delta q^A(d_k^A; x_k, X_k, H_k) \leq 2\Delta q^A(d'; x', \mathbb{B}_{\epsilon'}(x'), H_k)$$

whenever $k \geq k''$, $k \in \mathcal{K}$, and $X_k \in \mathcal{T}$. Under Assumptions 4.1.1 and 4.1.2 and since $x_k \to x'$, we have that for all k sufficiently large, $L_k\|d_k^B\|^2 \leq L$ for some constant $L > 0$, implying that for all $k \geq k''$ with $k \in \mathcal{K}$ such that $X_k \in \mathcal{T}$, α_k is bounded away from zero. Together, this and the fact that $\alpha_k \to 0$ imply that $X_k \notin \mathcal{T}$ for all $k \geq k''$ with $k \in \mathcal{K}$. Again, this is a probability zero event.

Case 2: Suppose $\{\epsilon_k\} \to 0$ and $\{x_k\}$ has a cluster point x'. We want to show that x' is stationary for $\tilde{f}$. The proof is exactly that of [18, Theorem 4.2, Case 2]. □

4.1.3 Numerical Experiments

We implement Algorithm 10 in Matlab and call the QO subproblem solver in Appendix A to solve the QO subproblem (4.7). In this section, we describe the algorithm variations that we have tested, the test problems that we have solved, and the results of our numerical experiments. All tests are performed on a machine running Debian 2.6.32 with two 8-Core AMD Opteron 6128 2.0 GHz processors and 32 GB RAM.

We consider two algorithm variations described below.

- `AGS` This is the adaptive gradient sampling algorithm which is obtained by the implementation of Algorithm 10 with the linear term ξ_k in the QO subproblems (4.7) defined as $\xi_k = f(x_k)e$.

- `BGS` This is the bundle gradient sampling algorithm which is obtained by the implementation of Algorithm 10 with the linear term ξ_k in the QO subproblems (4.7) defined as in (4.8).

Specific values for the input parameters of Algorithm 10 are set as the following. We choose $p = 2n$ as the number of sample points required for a complete line search; and $\overline{p} = 2n$ as the number of sample points to compute each iteration. We set the sampling radius reduction factor to be $\psi = 0.5$, number of backtracks for an incomplete line search to be $q = 7$, sufficient decrease constant $\eta = 10^{-8}$, line search backtracking constant $\kappa = 0.5$, and tolerance parameter $\nu = 1$. We choose the initial sampling radius to be $\epsilon_0 = 0.1$. We terminate Algorithm 10 either when the optimality conditions $\min\{\|d_k^A\|^2, (d_k^A)^T H_k d_k^A\} \leq \epsilon_k \leq 10^{-4}$ are satisfied, or when the maximum number of iterations 10^4 is reached. The QO subproblem solver is implemented as described in Appendix. We set the subproblem optimality tolerance to be 10^{-10} and maximum number of iterations to be 10^3. The Hessian and inverse Hessian approximations are set to be $H_k = W_k = I$.

We test algorithm variations `AGS` and `BGS` with the same 26 nonsmooth problems as described in Chapter 2 and Chapter 3. We choose $n = 50$ for all the 26 problems. The only exception is problem 24, for which we choose $n = 64$, as the variables for this problem need to compose a square matrix. We run each problem 10 times, each with different starting points. Please refer to previous chapters for details of starting points and parameters of the test problems.

The performance measures we consider are nonlinear iterations, function evaluations, gradient evaluations, and overall QO iterations. We present the numerical results by using performance profiles in Figure 4.2. From the performance profiles, we can see that `AGS` solved almost 90% of the test problems, while BGS only solved 80%. In terms of computational efficiency, `BGS` uses more function and gradient evaluations, and significantly more overall QO iterations because two QO subproblems are solved each iteration in BGS. The only major difference between `AGS` and `BGS` comes from the QO subproblems. The QO subproblem of `BGS` is motivated by the combination of the bundle method (BM) and the gradient sampling (GS) method. We have proved that the QO subproblem of `BGS` can produce a longer step than `AGS`; and therefore bigger predicted function value reduction. However, the longer step may not necessarily be a productive step. In particular, when the sampling radius is small, perhaps more backtracks are required to get sufficient decrease in terms of the true function value, resulting in more nonlinear iterations, function evaluations and gradient evaluations.

4.2 A Smoothing BFGS Gradient Sampling Algorithm

In this section, we propose a smoothing BFGS gradient sampling algorithm, which is based on the smoothing method and the BFGS-GS algorithm proposed in Chapter 3. A motivation for the smoothing approach is that it has theoretical convergence guarantees even when the problem functions are not Lipschitz. Numerical results are presented to illustrate that our algorithm is competitive with another recently proposed smoothing method [14] for non-Lipschitz optimization.

In the smoothing BFGS-GS algorithm, a sequence of parameterized smoothing functions is used to approximate the original nonsmooth objective function. The BFGS-GS algorithm is employed to solve the smooth but perhaps very nonlinear subproblems. By updating the smoothing parameter, the smoothing BFGS-GS algorithm can find a point satisfying the first order necessary condition of the original nonsmooth problem.

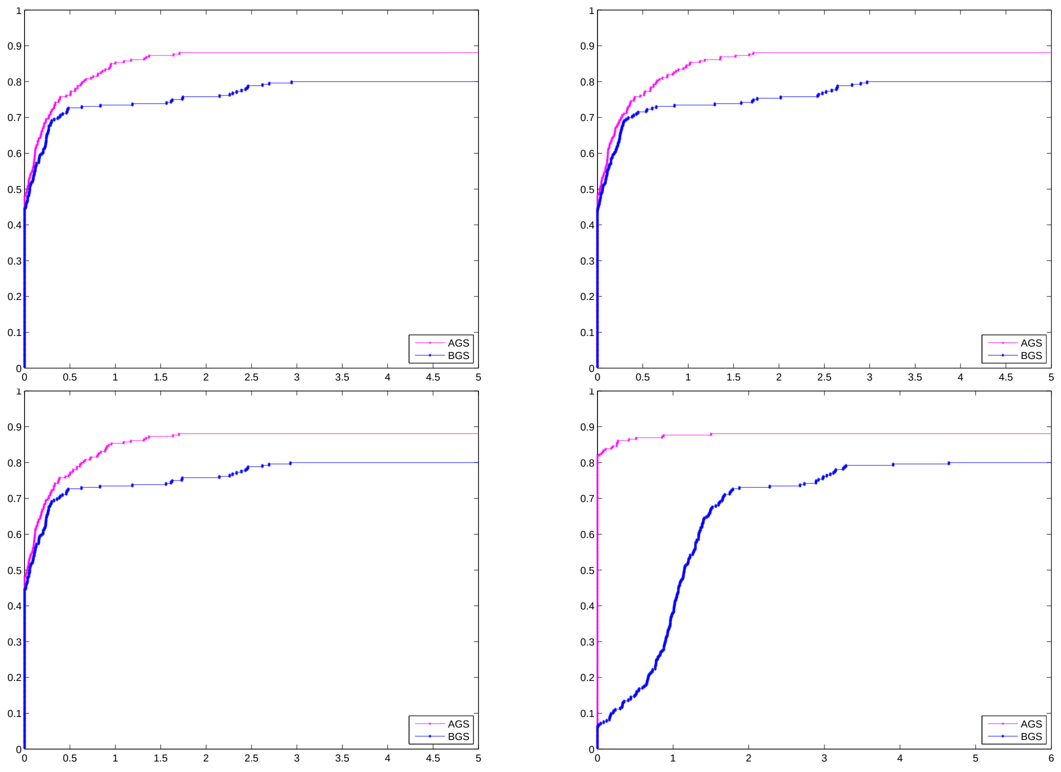


Figure 4.2: Performance profiles for nonlinear iterations (upper left), function evaluations (upper right), gradient evaluations (lower left), and overall QO iterations (lower right) comparing algorithms `AGS` and `BGS`.

4.2.1 Algorithm Description

We consider the following unconstrained minimization problem:

$$\min_x \; f(x) := \theta(x) + \lambda \sum_{i=1}^{m} \phi(|d_i^T x|), \tag{4.21}$$

where $\theta : \mathbb{R}^n \to \mathbb{R}_+$, $\phi : \mathbb{R}_+ \to \mathbb{R}_+$, $\lambda \in \mathbb{R}_+$, and $d_i \in \mathbb{R}^n, i = 1, ..., m$. In particular, we are interested in problems of this form in which θ represents a data-fitting term while ϕ represents a penalty function designed to instill certain properties in the solution vector, such as sparsity. We assume that the objective function f has bounded level sets, the data fitting function θ is twice continuously differentiable, and the penalty function ϕ satisfies the following assumption. Note that the penalty function ϕ maybe non-convex, non-differentiable, and perhaps even non-Lipschitz.

Assumption 4.2.1. *(i) ϕ is differentiable in $(0, \infty)$, and ϕ' is locally Lipschitz continuous* *in* $(0,\infty)$. *(ii)* ϕ *is continuous at* 0 *with* $\phi(0)=0$, $\phi'(0^+)>0$, *and* $\phi'(t)\geq 0$ *for all* $t>0$.

To develop the smoothing BFGS-GS algorithm for (4.21), we construct a $\mathcal{C}^2$ smoothing function $\tilde{f}(\cdot,\mu)$ for the objective function $f(\cdot)$ in (4.21), where μ is the smoothing parameter. Since the first term $\theta(\cdot)$ of $f(\cdot)$ is twice continuously differentiable, we only need to construct a $\mathcal{C}^2$ smoothing function $\tilde{\phi}(\cdot,\mu)$ for the second term $\phi(\cdot)$. In particular, we assume $\tilde{\phi}(\cdot,\mu)$ satisfy Assumption 2.1 in [14].

The smoothing BFGS-GS algorithm is a line search algorithm. At each iteration, for a given smoothing parameter μ_k and a given iterate x_k, we desire the search direction d_k to be the minimizer of a quadratic model of $\tilde{f}(x_k,\mu_k)$, which is equivalent to solving the following primal and dual pair:

$$\left\{\begin{array}{c} \min\limits_{(z,d)\in\mathbb{R}^{n+1}} \ z+\frac{1}{2}\|d\|^2_{W_k^{-1}} \\ \\ \text{s.t. } G_k^T d \leq ze \end{array}\right\} \qquad \left\{\begin{array}{c} \max\limits_{y\in\mathbb{R}^{p_k+1}} \ -\frac{1}{2}\|G_k y\|^2_{W_k} \\ \\ \text{s.t. } e^T y = 1,\ y\geq 0 \end{array}\right\} \tag{4.22}$$

where G_k is the gradient matrix defined as the following:

$$G_k := \begin{bmatrix} g_{k,0} & \cdots & g_{k,p_k} \end{bmatrix}, \tag{4.23}$$

where $g_{k,i} := \nabla\tilde{f}(x_{k,i},\mu_k)$ for all $i\in\{0,\ldots,p_k\}$. Then we compute a positive step size $\alpha_k>0$ satisfying the well-known (weak) Wolfe line search conditions.

$$\tilde{f}(x_k,\mu_k)-\tilde{f}(x_k+\alpha_k d_k,\mu_k) \geq \underline{\eta}\alpha_k\|G_k y_k\|^2_{W_k}; \tag{4.24a}$$

$$\nabla\tilde{f}(x_k+\alpha_k d_k,\mu_k)^T d_k \geq \overline{\eta}\nabla\tilde{f}(x_k,\mu_k)^T d_k. \tag{4.24b}$$

Once the search direction d_k and step size $\alpha_k\geq 0$ have been computed, the remainder of the iteration involves setting the next sampling radius $\epsilon_{k+1}\in(0,\epsilon_k]$, smoothing parameter $\mu_{k+1}\in(0,\mu_k]$, sample set X_{k+1} (and related quantities), and inverse Hessian approximation W_{k+1}.

Define the following conditions for updating the sampling radius ϵ_{k+1}, sample set X_{k+1}, and smoothing parameter μ_{k+1}:

$$\|G_k y_k\|_{W_k} \leq \nu\epsilon_k; \tag{4.25a}$$

$$\|G_k y_k\|_{W_k} \geq \xi\|d_k\|_2; \tag{4.25b}$$

$$\alpha_k \geq \underline{\alpha}; \tag{4.25c}$$

$$\|\nabla\tilde{f}(x_k,\mu_k)\| \leq \nu\mu_k. \tag{4.25d}$$

We summarize parameters employed in our algorithm in Table 4.1.

Table 4.1: User-specified constants for the proposed algorithm

Parameter(s)	Range	Description
ν	$(0,\infty)$	Stationarity measure tolerance
ψ	$(0,1]$	Sampling radius reduction factor
ζ	$(0,1]$	Smoothing parameter reduction factor
ξ	$(0,\infty)$	Model curvature threshold
$\underline{\eta} < \overline{\eta}$	$(0,1)$	Armijo–Wolfe line search constants
$\underline{\alpha}$	$(0,\infty)$	Step size threshold
p	$[n+1,\infty)\cap\mathbb{N}$	Sample set size threshold
$\underline{\mu} < 1 < \overline{\mu}$	$(0,\infty)$	(L-)BFGS updating thresholds
$\underline{w} \leq \overline{w}$	$(0,\infty)$	(L-)BFGS updating thresholds
m	$\mathbb{N}$	L-BFGS memory length

We now present our main algorithm, stated as Algorithm 11.

4.2.2 Global Convergence Analysis

We first show that if Algorithm 11 reaches Step 2 during iteration k, then it computes d_k as null or as a direction of strict descent for f from $x_k \in \mathcal{D}$. We state this result, which also proves an important relationship between the search direction d_k and the dual QP solution y_k; see also [18, Lemma 4.3].

Lemma 4.2.2. *If Algorithm 11 reaches Step 2 during iteration k, then it computes a*

Algorithm 11 Smoothing BFGS Gradient Sampling Algorithm

```
1: Choose an initial iterate $x_0$, inverse Hessian approximation $W_0 \succ 0$, sampling radius $\epsilon_0 > 0$, and smoothing parameter $\mu_0 > 0$. Set the initial sample set $X_0 \leftarrow \{x_0\}$, sample set size $p_0 \leftarrow 0$, matrix of sample gradients $G_0 \leftarrow \nabla\tilde{f}(x_0, \mu_0)$ and iteration counter $k \leftarrow 0$.
2: Compute a search direction $d_k \leftarrow -W_k G_k y_k$ where $y_k$ solves the dual QP in the primal-dual pair (4.22).
3: Compute a step size $\alpha_k \geq 0$ satisfying the well-known (weak) Wolfe line search conditions (4.24).
4: Compute a new iterate $x_{k+1} \leftarrow x_k + \alpha_k d_k$.
5: If conditions (4.25a) and (4.25b) hold, then set the new sampling radius $\epsilon_{k+1} \leftarrow \psi\epsilon_k$; otherwise, set $\epsilon_{k+1} \leftarrow \epsilon_k$.
6: If the condition (4.25d) holds, then set the new smoothing parameter $\mu_{k+1} \leftarrow \psi\mu_k$ and reset the sampling radius $\epsilon_{k+1} \leftarrow \epsilon_0$; otherwise, set $\mu_{k+1} \leftarrow \mu_k$.
7: Compute a new sample set $X_{k+1}$ with $p_{k+1} \leftarrow |X_{k+1}| - 1$ as the following. If conditions (4.25b) and (4.25c) hold, then set $X_{k+1} \leftarrow \{x_{k+1}\}$ and $p_{k+1} \leftarrow 0$, terminate. Otherwise, set $X_{k+1} \leftarrow (X_k \cap B_{k+1}) \cup \{x_{k+1}\} \cup \overline{X}_{k+1}$ and $p_{k+1} \leftarrow |X_{k+1}| - 1$. Here, $\overline{X}_{k+1}$ is a collection of $\overline{p}_{k+1}$ points generated independently from a uniform distribution over $B_{k+1}$. If $p_{k+1} > p$, then remove the $p_{k+1} - p$ eldest members of $X_{k+1} \backslash \{x_{k+1}\}$ and set $p_{k+1} \leftarrow p$.
8: Compute the matrix of gradients $G_{k+1}$ defined in (4.23).
9: Compute a new inverse Hessian approximation $W_{k+1} \succ 0$ via Algorithm 9 in §3.2.4.
10: Set $k \leftarrow k + 1$ and go to Step 2.
```

search direction d_k that is zero or a direction of strict descent for $\tilde{f}$ from x_k. In addition, the primal-dual solution (z_k, d_k, y_k) of (4.22) *satisfies $\|G_k y_k\|_{W_k} = \|d_k\|_{W_k^{-1}}$.*

Proof. In the proof of [18, Lemma 4.3], we have the following inequality from the KKT conditions of the primal and dual subproblems:

$$\nabla\tilde{f}(x_k, \mu_k)^T d_k \leq -\|G_k y_k\|_{W_k}^2 = -\|d_k\|_{W_k^{-1}}^2. \tag{4.26}$$

If $d_k = 0$, then there is nothing left to prove. Hence, from (4.26) and $W_k^{-1} \succ 0$, it follows that if Step 2 is reached and it produces $d_k \neq 0$, then $\nabla\tilde{f}(x_k, \mu_k)^T d_k < 0$. □

We now prove a critical inequality for a subset of iterations; see also [18, Lemma 4.5].

Lemma 4.2.3. *If $\xi\|d_k\|_2 \leq \|G_k y_k\|_{W_k}$ holds during iteration k, then*

$$\tilde{f}(x_{k+1}, \mu_k) \leq \tilde{f}(x_k, \mu_k) - \underline{\eta}\xi\|x_{k+1} - x_k\|_2\|d_k\|_2.$$

Proof. By Step 4 of Algorithm 11, we have

$$\|x_{k+1} - x_k\|_2 = \alpha_k \|d_k\|_2. \tag{4.27}$$

Thus, by the sufficient decrease condition (4.24a), we have

$$\begin{aligned}\tilde{f}(x_{k+1}) - \tilde{f}(x_k) &\leq -\underline{\eta}\alpha_k \|G_k y_k\|_{W_k}^2 \\ &\leq -\underline{\eta}\alpha_k \xi \|d_k\|_2^2 \\ &= -\underline{\eta}\xi \|x_{k+1} - x_k\|_2 \|d_k\|_2.\end{aligned}$$

□

Given $x' \in \mathbb{R}^n$, we define

$$\mathcal{G}_k(x') := \operatorname{cl}\operatorname{conv} \nabla \tilde{f}(B(x', \epsilon_k), \mu_k),$$

and, also given a tolerance $\omega > 0$, we define

$$\mathcal{T}_k(x', \omega) := \left\{ X_k \in \prod_0^{p_k} B_k : \|P_{W_k}(\{\nabla \tilde{f}(x, \mu_k)\}_{x \in X_k})\|_{W_k}^2 \leq \|P_{W_k}(\mathcal{G}_k(x'))\|_{W_k}^2 + \omega \right\}.$$

The purpose of the following lemma is to show that for an iterate x_k sufficiently close to x' and any tolerance $\omega > 0$, there exists a nonempty subset of $\mathcal{T}_k(x', \omega)$ if the sample set size $p_k \geq n + 1$; see the similar result of Lemma 4.1.8.

Lemma 4.2.4. *If $p_k \geq n+1$, then for any $\omega > 0$, there exists $\zeta > 0$ and a nonempty set $\mathcal{T}$ such that for all $x_k \in B(x', \zeta)$ we have $\mathcal{T} \subseteq \mathcal{T}_k(x', \omega)$.*

The purpose of the following lemma is to show that if the sample set size $p_k \geq n + 1$, for an iterate x_k sufficiently close to a non-stationary point x', Algorithm 11 eventually computes a step size α_k that is bounded below, so that the iterates $\{x_k\}$ move away from the non-stationary point x'; see the similar result [39, Lemma 3.2(ii)].

Lemma 4.2.5. *Assuming $0 \notin \mathcal{G}_k(x')$ and the sample set size $p_k \geq n + 1$, pick $\omega > 0$ and then $(\zeta, \mathcal{T})$ as in Lemma 4.2.4. Suppose at iteration k of Algorithm 11, Step 3 is reached* *with $x_k \in B(x', \min\{\zeta, \epsilon_k/3\})$ and $X_k \in \mathcal{T}$. Then $\alpha_k \geq \gamma\epsilon_k/3\kappa$, where κ is the Lipschitz constant of f on $B(x', 2\epsilon_k)$.*

The following lemma is critical to the global convergence proof.

Lemma 4.2.6. *Consider the iterates $\{x_k\}$ and $\{\mu_k\}$ generated by applying Algorithm 11 to problem* (4.21)*. Then $\mathcal{K}_\mu$ defined in* (4.32) *is an infinite set.*

Proof. Suppose for contradiction that $\mathcal{K}_\mu$ defined in (4.32) is finite. Then there exists $\bar{\mu} > 0$ such that $\mu_k = \bar{\mu}$ and $\|\nabla \tilde{f}(x_k, \mu_k)\| > \nu\mu_k$ for all sufficiently large k.

Define the index sets

$$\mathcal{K}_\epsilon := \{k \mid \xi\|d_k\|_2 \leq \|G_k y_k\|_{W_k} \leq \nu\epsilon_k\}, \tag{4.28}$$

and

$$\mathcal{K}_d := \{k \mid \xi\|d_k\|_2 \leq \|G_k y_k\|_{W_k} \text{ and } \alpha_k \geq \underline{\alpha}\}. \tag{4.29}$$

Our first main goal is to show that $\{\epsilon_k\} \to 0$. To prove this, we consider two cases.

Case 1: Suppose $\mathcal{K}_d$ defined in (4.29) is an infinite set. Then, along with the sufficient decrease condition (4.24a), we have

$$\tilde{f}(x_{k+1}, \mu_k) - \tilde{f}(x_k, \mu_k) \leq -\underline{\eta}\alpha_k\|G_k y_k\|_{W_k}^2 \leq -\underline{\eta}\underline{\alpha}\xi^2\|d_k\|_2^2 \text{ for all } k \in \mathcal{K}_d.$$

From Assumption 2.1 in [14], we know $\tilde{f}(x, \mu) \geq f(x)$ which implies that the level set of $\tilde{f}$ is a subset of the level set of f. Since f has bounded level sets, $\tilde{f}$ has also bounded level sets for any given $\mu > 0$. This implies that

$$\lim_{k \in \mathcal{K}_d} \|d_k\|_2 = 0,$$

which, by Step 5 of Algorithm 11, implies that $\{\epsilon_k\} \to 0$.

Case 2: Suppose $\mathcal{K}_d$ defined in (4.29) is finite. Then either $\xi\|d_k\|_2 \leq \|G_k y_k\|_{W_k}$ or $\alpha_k \geq \underline{\alpha}$ does not hold for all sufficiently large k.

According to the inverse Hessian updating strategy, it follows that for all sufficiently large k we have W_{k+1}^{-1} and W_{k+1} be bounded. Indeed, in this case, we may assume without loss of generality that W_{k+1}^{-1} and W_{k+1} being bounded for all k.

We now prove that $\{\epsilon_k\} \to 0$ with probability one by showing that the event that $\{\epsilon_k\}$ remains bounded away from zero has probability zero.

Suppose that there exists k' such that $\epsilon_k = \epsilon' > 0$ for all $k \geq k'$. From this fact, it follows that either $\xi \|d_k\|_2 \leq \|G_k y_k\|_{W_k}$ or $\|G_k y_k\|_{W_k} \leq \nu \epsilon_k$ does not hold for all $k \geq k'$. In fact, since $\xi \|d_k\|_2 \leq \|G_k y_k\|_{W_k}$ holds for all k, we must have

$$\|G_k y_k\|_{W_k} > \nu \epsilon' \text{ for all } k \geq k'. \tag{4.30}$$

On the other hand, the fact that $\{\tilde{f}\}$ is bounded below, the sufficient decrease condition, and the inequality in Lemma 4.2.3 together imply that

$$\sum_{k=k'}^{\infty} \alpha_k \|G_k y_k\|_{W_k}^2 < \infty, \quad \text{and} \tag{4.31a}$$

$$\sum_{k=k'}^{\infty} \|x_{k+1} - x_k\|_2 \|d_k\|_2 < \infty. \tag{4.31b}$$

In conjunction with (4.30), the bound (4.31a) implies $\alpha_k \to 0$. Similarly, (4.31b) and (4.30) imply that $\{x_k\}$ is a Cauchy sequence, and hence $x_k \to x'$ for some $x' \in \mathbb{R}^n$. We claim that this implies the existence of an infinite iteration index set $\mathcal{K}_p := \{k : k \geq k' \text{ and } p_k = p\}$.

We continue by considering two subcases.

Subcase 2.a: Suppose x' is ϵ'-stationary for $\tilde{f}$. See the proof of [18, Theorem 3.1, Subcase 2.a], we have $X_k \notin \mathcal{T}$ for sufficiently large $k \in \mathcal{K}_p$. This is a probability zero event.

Subcase 2.b: Suppose x' is not ϵ'-stationary for $\tilde{f}$. See the proof of [18, Theorem 3.1, Subcase 2.b], we have $X_k \notin \mathcal{T}$ for sufficiently large $k \in \mathcal{K}_p$. This is also a probability zero event.

Now since we have $\{\epsilon_k\} \to 0$ and $\{x_k\}$ has a cluster point x', all that remains is to show that x' is stationary for $\tilde{f}$. The proof is exactly that of [18, Theorem 4.2, Case 2].

Then we have

$$\liminf_{k\to\infty} \|\nabla \tilde{f}(x_k, \mu_k)\| = 0,$$

which contracts with the supposition that $\mathcal{K}_\mu$ defined in (4.32) is finite.

□

We are now prepared to prove the global convergence theory.

Theorem 4.2.7. *Consider the iterates $\{x_k\}$ and $\{\mu_k\}$ generated by applying Algorithm 11 to problem* (4.21)*. Define the index set*

$$\mathcal{K}_\mu := \{k \mid \|\nabla \tilde{f}(x_k, \mu_k)\| \leq \nu\mu_k\}. \tag{4.32}$$

If $\mathcal{K}_\mu$ is an infinite set, then

$$\lim_{k\to\infty} \mu_k = 0, \tag{4.33a}$$

$$\liminf_{k\to\infty} \|\nabla \tilde{f}(x_k, \mu_k)\| = 0, \tag{4.33b}$$

and any accumulation point of $\{x_k\}$ satisfies the first-order necessary condition in Theorem 4.4 of [14].

Proof. We know by Lemma 4.2.6 that $\mathcal{K}_\mu$ is an infinite set, by Step 6 we have

$$\lim_{k\to\infty} \mu_k = 0$$

because Algorithm 11 generates a monotonically decreasing sequence $\{\mu_k\}_{k=0}^{\infty}$. Therefore, we also have

$$\liminf_{k\to\infty} \|\nabla \tilde{f}(x_k, \mu_k)\| = 0.$$

□

4.2.3 Numerical Experiments

In this section, we compare our smoothing BFGS-GS algorithm (`SBFGSGS`) with a smoothing trust region Newton method (`STR`) proposed in [14]. Both algorithms are implemented in Matlab. The QO subproblem solver in Appendix A is called to solve the QO subproblem (4.22) in `SBFGSGS`. The Fortran subroutine `GQTPAR` in [47] is called to solve the trust region subproblem in `STR`. For fair comparison, the termination conditions for both algorithms are either that the maximum iteration number is reached, or that the following optimality conditions are satisfied:

$$\mu_k \leq \nu \quad \text{and} \quad \|\nabla \tilde{f}(x_k, \mu_k)\| \leq \nu,$$

where $\nu > 0$ is a given optimality tolerance. In particular, we set the initial smoothing parameter to be $\mu_0 = 0.01$, the smoothing parameter reduction factor to be $\zeta = 0.1$, the optimality tolerance to be $\nu = 10^{-4}$, and the maximum iteration number to be 10^4 for both algorithms. We refer to [14] for specific values of other parameters of `STR`, and Chapter 3 for other parameter values of `SBFGSGS`.

We test both algorithms for six penalty functions of $\phi(\cdot)$ in (4.21): $\phi_1, ..., \phi_6$. The six penalty functions are called fraction penalty, log-penalty, ℓ_q penalty (or bridge penalty), hard thresholding penalty, smoothingly clipped absolute deviation penalty, and minimax concave penalty. Please refer to Section 1 of [14] for details of the six penalty functions. We consider the three test problems from Section 5 of [14]: prostate cancer, linear regression, logistic regression.

Example: Prostate Cancer

The prostate cancer problem studies the correlation between the level of prostate specific antigen (lpsa) and eight clinical measures. The problem is formulated as in (4.21) with the data-fitting function being $\theta(x) = \|Ax - b\|^2$ and the six penalty functions mentioned before: $\phi_1, ..., \phi_6$.

The data set (A and b) are available on the website `http://statweb.stanford.edu/`

`~tibs/ElemStatLearn/`. The data set consists of medical records of 97 patients, which is divided into two parts: a training set with 67 observations and a test set with 30 observations. The prediction error is defined as the mean squared errors (MSEs) over the test set.

In the first set of experiments, we apply both algorithms to solving the prostate cancer problem with specific parameter values for the six penalty functions. Final solution and MSE from `SBFGSGS` and `STR` are reported in Table 4.2 and Table 4.3, respectively. From the tables, we can see that the results returned by both algorithms are almost identical for this test problem.

	ϕ_1	ϕ_2	$\phi_3(q=1.0)$	$\phi_3(q=0.5)$	ϕ_4	ϕ_5	ϕ_6
λ	14.5	14.5	14.5	14.5	14.5	14.5	14.5
α	1.0	1.0	(-)	(-)	(-)	3.7	2.7
	0.6800	0.6800	0.5487	0.6461	0.5590	0.5487	0.5506
	0.2635	0.2635	0.2157	0.2752	0.2152	0.2157	0.2158
	0	0	0	0	0	0	0
	0.2107	0.2107	0.0909	0	0.0871	0.0909	0.0909
	0.3047	0.3047	0.1578	0.1277	0.1522	0.1578	0.1573
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0.2634	0.2634	0.0606	0	0.0554	0.0606	0.0600
MSE	0.5185	0.5185	0.4514	0.4283	0.4497	0.4514	0.4511

Table 4.2: Results for prostate cancer from algorithm `SBFGSGS`

	ϕ_1	ϕ_2	$\phi_3(q=1.0)$	$\phi_3(q=0.5)$	ϕ_4	ϕ_5	ϕ_6
λ	14.5	14.5	14.5	14.5	14.5	14.5	14.5
α	1.0	1.0	(-)	(-)	(-)	3.7	2.7
	0.6800	0.6800	0.5487	0.6461	0.5590	0.5487	0.5506
	0.2635	0.2635	0.2157	0.2752	0.2152	0.2157	0.2158
	0	0	0	0	0	0	0
	0.2107	0.2107	0.0909	0	0.0871	0.0909	0.0909
	0.3047	0.3047	0.1578	0.1277	0.1522	0.1578	0.1573
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0.2634	0.2634	0.0606	0	0.0554	0.0606	0.0600
MSE	0.5185	0.5185	0.4514	0.4283	0.4497	0.4514	0.4511

Table 4.3: Results for prostate cancer from algorithm `STR`

As in [14], we focus on the ℓ_q penalty function ϕ_3 in the second set of experiments, since ϕ_3 with $q = 0.5$ performs the best on the previous tables. Numerical results with $\lambda = 8$ and $q = 0.9, 0.8, ..., 0.3$ are presented in Table 4.4 and Table 4.5. Similarly here, the results returned by both algorithms are almost identical.

	$q = 0.9$	$q = 0.8$	$q = 0.7$	$q = 0.6$	$q = 0.5$	$q = 0.4$	$q = 0.3$
	0.5659	0.5827	0.6091	0.6202	0.6461	0.7254	0.6543
	0.2264	0.2257	0.2229	0.229	0.2752	0.2782	0.2838
	0	0	0	0	0	0	0
	0.1316	0.1227	0.1141	0.0982	0	0	0
	0.1879	0.1837	0.1914	0.1784	0.1277	0	0.1295
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0.0747	0.0531	0	0	0	0	0
MSE	0.4526	0.4458	0.4358	0.4332	0.4283	0.4895	0.4268

Table 4.4: Results for prostate cancer with penalty function ϕ_3 from algorithm SBFGSGS

	$q = 0.9$	$q = 0.8$	$q = 0.7$	$q = 0.6$	$q = 0.5$	$q = 0.4$	$q = 0.3$
	0.5659	0.5827	0.6091	0.6202	0.6461	0.656	0.6543
	0.2264	0.2257	0.2229	0.229	0.2752	0.2784	0.2838
	0	0	0	0	0	0	0
	0.1316	0.1227	0.1141	0.0982	0	0	0
	0.1879	0.1837	0.1914	0.1784	0.1277	0.1189	0.1295
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0.0747	0.0531	0	0	0	0	0
MSE	0.4526	0.4458	0.4358	0.4332	0.4283	0.4311	0.4268

Table 4.5: Results for prostate cancer with penalty function ϕ_3 from algorithm STR

Example: Linear Regression

Now we consider a linear regression problem (i.e., $\theta(x) = \|Ax - b\|^2$) also with the six penalty functions. Each rows of the matrix A is an eight-dimensional vector from multivariate normal distribution with covariance between a_i and a_j being $0.5^{|i-j|}$ $(1 \leq i, j \leq 8)$. Each component of the vector b is computed from the following data model:

$$b = a^T x + \sigma\epsilon,$$

where $x = (3, 1.5, 0, 0, 2, 0, 0, 0)$ and $\epsilon \sim N(0, 1)$. Let n be the sample size. Three sets of experiments are performed: $(n, \sigma) = (40, 3), (40, 1)$, and $(60, 1)$. For each pair (n, σ), the results are based on the average of randomly generated 100 runs. The sparsity of the solutions is measured by computing the average number of correct zeros and incorrect zeros. To reflect the quality of the solutions returned by the two algorithms, we report the median relative model error (MRME). The relative model error (RME) is defined as the following

$$RME(\bar{x}) = \frac{(\bar{x} - x^*)^T \sum(\bar{x} - x^*)}{(x_{LS} - x^*)^T \sum(x_{LS} - x^*)},$$

where x^* is the true solution, $\bar{x}$ is the solution returned by the two algorithms, and x_{LS} is the least squares solution. Problem parameters and numerical results are illustrated in Table 4.6 and Table 4.7, respectively. From the tables, we can see that both algorithms perform poorly with the first two penalty functions ϕ_1 and ϕ_2. However, both algorithms perform similarly well with the last four penalty functions.

Example: Logistic Regression

The logistic regression example is similar to the previous linear regression example except that, instead of using the linear regression function as the data-fitting function, this example uses the following logistic regression function:

$$\theta(x) = \sum_{i=1}^{n} \ln \frac{e^{-b_i(x^T a_i)}}{1 + e^{x^T a_i}},$$

where $x = (3, 1.5, 0, 0, 2, 0, 0, 0)$ is the same as in the previous example; the first six components of a are the same as before and the last two components of a are independently identically distributed as a Bernoulli distribution with probability of success 0.5; and the the vector b is computed by the model $b \sim$Beroulli$\{p(a^T x)\}$, where $p(u) = exp(u)/(1 + exp(u))$. Similar results from both algorithms are reported in Table 4.8 and Table 4.9.

Method	Parameters	MRME	Correct 0's	Incorrect 0's
	$n = 40, \sigma = 3$			
ϕ_1	$\lambda = 70.0, \alpha = 1.0$	1	0	0
ϕ_2	$\lambda = 70.0, \alpha = 1.0$	1	0	0
ϕ_3	$\lambda = 49.0, q = 0.5$	0.3966	4.72	0.34
ϕ_3	$\lambda = 50.0, q = 1.0$	1.0998	3.80	0.05
ϕ_4	$\lambda = 26.0$	0.6692	3.93	0.11
ϕ_5	$\lambda = 57.0, \alpha = 3.7$	1.0721	4.02	0.10
ϕ_6	$\lambda = 47.5, \alpha = 2.7$	1.0244	3.82	0.08
	$n = 40, \sigma = 1$			
ϕ_1	$\lambda = 41.5, \alpha = 1.0$	1	0.02	0
ϕ_2	$\lambda = 40.0, \alpha = 1.0$	1	0.02	0
ϕ_3	$\lambda = 19.0, q = 0.5$	0.2435	4.97	0
ϕ_3	$\lambda = 20.0, q = 1.0$	1.5653	4.04	0
ϕ_4	$\lambda = 8.5$	0.6044	4.15	0
ϕ_5	$\lambda = 17.5, \alpha = 3.7$	1.0181	3.88	0
ϕ_6	$\lambda = 19.5, \alpha = 2.7$	1.1236	4.08	0
	$n = 60, \sigma = 1$			
ϕ_1	$\lambda = 44.5, \alpha = 1.0$	1	0.07	0
ϕ_2	$\lambda = 44.5, \alpha = 1.0$	1	0.03	0
ϕ_3	$\lambda = 20.0, q = 0.5$	0.1290	4.97	0
ϕ_3	$\lambda = 22.5, q = 1.0$	0.7292	4.01	0
ϕ_4	$\lambda = 11.5$	0.5315	4.19	0
ϕ_5	$\lambda = 24.5, \alpha = 3.7$	1.1932	4.06	0
ϕ_6	$\lambda = 22.5, \alpha = 2.7$	0.8174	3.81	0

Table 4.6: Results of linear regression from algorithm `SBFGSGS`

4.3 Gradient Sampling for ℓ_1-Regularization

In this section, we propose an algorithm motivated by the gradient sampling (GS) idea for solving ℓ_1 regularization problems. Global convergence analysis is provided. Preliminary numerical experiments are performed to compare different algorithmic variations of GS with an iterative shrinkage-thresholding algorithms (ISTA).

4.3.1 Algorithm Description

Consider the unconstrained minimization problem

$$\min_x \; f(x) := f_s(x) + \mu\|x\|_1, \tag{4.34}$$

Method	Parameters	MRME	Correct 0's	Incorrect 0's
	$n = 40, \sigma = 3$			
ϕ_1	$\lambda = 70.0, \alpha = 1.0$	1	0.01	0
ϕ_2	$\lambda = 70.0, \alpha = 1.0$	1	0	0
ϕ_3	$\lambda = 49.0, q = 0.5$	0.4697	4.72	0.42
ϕ_3	$\lambda = 50.0, q = 1.0$	0.6195	3.88	0.07
ϕ_4	$\lambda = 26.0$	0.8197	3.92	0.08
ϕ_5	$\lambda = 57.0, \alpha = 3.7$	0.9466	3.94	0.06
ϕ_6	$\lambda = 47.5, \alpha = 2.7$	0.6581	3.74	0.05
	$n = 40, \sigma = 1$			
ϕ_1	$\lambda = 41.5, \alpha = 1.0$	1	0.04	0
ϕ_2	$\lambda = 40.0, \alpha = 1.0$	1	0.01	0
ϕ_3	$\lambda = 19.0, q = 0.5$	0.1384	4.99	0
ϕ_3	$\lambda = 20.0, q = 1.0$	1.0758	4.03	0
ϕ_4	$\lambda = 8.5$	0.4705	3.91	0
ϕ_5	$\lambda = 17.5, \alpha = 3.7$	0.8022	3.84	0
ϕ_6	$\lambda = 19.5, \alpha = 2.7$	1.0153	4.11	0
	$n = 60, \sigma = 1$			
ϕ_1	$\lambda = 44.5, \alpha = 1.0$	1	0	0
ϕ_2	$\lambda = 44.5, \alpha = 1.0$	1	0.02	0
ϕ_3	$\lambda = 20.0, q = 0.5$	0.1585	4.97	0
ϕ_3	$\lambda = 22.5, q = 1.0$	0.7413	3.99	0
ϕ_4	$\lambda = 11.5$	0.7410	3.99	0
ϕ_5	$\lambda = 24.5, \alpha = 3.7$	0.9800	3.99	0
ϕ_6	$\lambda = 22.5, \alpha = 2.7$	0.9432	4.04	0

Table 4.7: Results of linear regression from algorithm STR

where $f_s(x) : \mathbb{R}^n \to \mathbb{R}$ is a smooth function. Let $\mathcal{D}$ be the set where the function $\|x\|_1$ is smooth, i.e., $\mathcal{D} := \{x \in \mathbb{R}^n : x^i \neq 0, \ \forall i = 1, ..., n\}$, where x^i is the ith component of x. Let $\mathbb{B}_\epsilon(x)$ be the "box" centered at x with "radius" ϵ, i.e., $\mathbb{B}_\epsilon(x) := \{\bar{x} \in \mathbb{R}^n : \|\bar{x} - x\|_\infty \leq \epsilon\}$.

At a given iterate $x_k \in \mathcal{D}$ and for a given radius $\epsilon_k > 0$, let q_k be the number of components satisfying $|x_k^i| < \epsilon_k, \ \forall i = 1, ..., n$, and then $p_k = 2^{q_k}$ is the number of all possible gradients of the function $\|x\|_1$ evaluated at $x \in B_k := \mathbb{B}_{\epsilon_k}(x_k) \cap \mathcal{D}$. Let

$$G_k = G_k^s + G_k^1,$$

where

$$G_k^1 := \begin{bmatrix} g_{k,1} & \cdots & g_{k,p_k} \end{bmatrix} \tag{4.35}$$

Method	Parameters	MRME	Correct 0's	Incorrect 0's
ϕ_1	$\lambda = 11.0, \alpha = 1.0$	0.0028	0	0
ϕ_2	$\lambda = 10.5, \alpha = 1.0$	0.0020	0	0
ϕ_3	$\lambda = 7.5, q = 0.5$	0.0111	4.97	0.01
ϕ_3	$\lambda = 21.5, q = 1.0$	0.4637	4.88	0.03
ϕ_4	$\lambda = 9.5$	0.3549	4.87	0.02
ϕ_5	$\lambda = 21.0, \alpha = 3.7$	0.4557	4.88	0.03
ϕ_6	$\lambda = 20.0, \alpha = 2.7$	0.4226	4.88	0.04

Table 4.8: Results of logistic regression from algorithm SBFGSGS

Method	Parameters	MRME	Correct 0's	Incorrect 0's
ϕ_1	$\lambda = 11.0, \alpha = 1.0$	0.0026	0	0
ϕ_2	$\lambda = 10.5, \alpha = 1.0$	0.0025	0.02	0
ϕ_3	$\lambda = 7.5, q = 0.5$	0.0102	4.96	0.04
ϕ_3	$\lambda = 21.5, q = 1.0$	0.4728	4.83	0.02
ϕ_4	$\lambda = 9.5$	0.3555	4.89	0.03
ϕ_5	$\lambda = 21.0, \alpha = 3.7$	0.4478	4.83	0
ϕ_6	$\lambda = 20.0, \alpha = 2.7$	0.4204	4.89	0.03

Table 4.9: Results of logistic regression from algorithm STR

denote the matrix whose columns are all possible gradients of the function $\|x\|_1$ evaluated at $x \in B_k$, and

$$G_k^s := \begin{bmatrix} \nabla f_s(x_k) & \cdots & \nabla f_s(x_k) \end{bmatrix} \tag{4.36}$$

denote the matrix that also has p_k columns and each column is simply the gradient of the function f_s at the current iterate x_k.

One should notice that gradients in G_k^1 are n-dimensional vectors with different combinations of 1's and -1's. In Figure 4.3, the contour of the function $\|x\|_1$ and the "box" centered at x_k with "radius" ϵ_k are plotted. The current iterate is $x_k = (0.4, 0.2)^T$. The "radius" in the left plot is $\epsilon_k = 0.3$; and in the right plot $\epsilon_k = 0.5$. The corresponding gradient matrix G_k^1 is computed as the following:

$$G_k^1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{ and } G_k^1 = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix}.$$

Let $H_k \in \mathbb{R}^{n \times n}$ be a positive definite matrix (i.e., $H_k \succ 0$), and $W_k := H_k^{-1} \succ 0$.

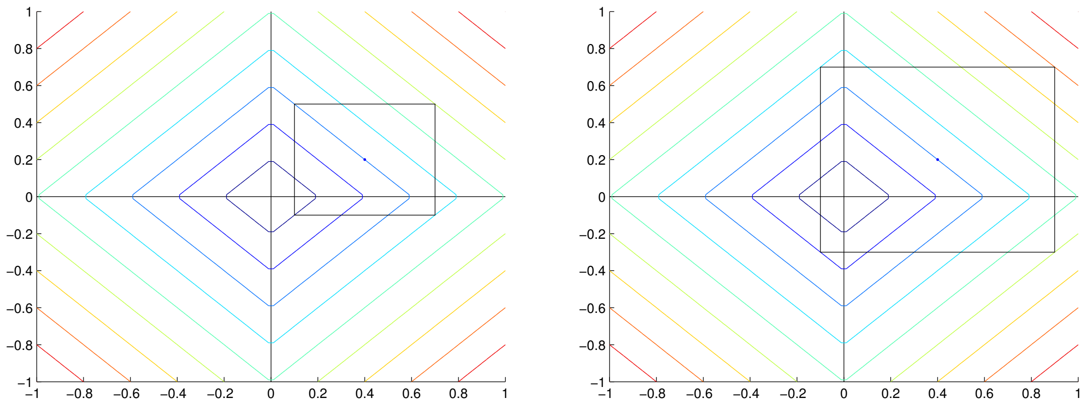


Figure 4.3: Illustration of the gradient matrix G_k^1.

The main computational expense in an iteration of the method is to solve the following primal-dual QO subproblems to compute the search direction d_k:

$$(\mathrm{P}) := \left\{ \begin{array}{l} \min\limits_{z,d} \; z + \frac{1}{2} d^T H_k d \\ \\ \text{s.t. } f(x_k)e + G_k^T d \le ze \end{array} \right\} \qquad (\mathrm{D}) := \left\{ \begin{array}{l} \max\limits_{\pi} \; f(x_k) - \frac{1}{2}\pi^T G_k^T W_k G_k \pi \\ \\ \text{s.t. } e^T\pi = 1, \;\; \pi \ge 0. \end{array} \right\} \tag{4.37}$$

Note that either the primal or the dual (not both) needs to be solved; and the solution (z_k, d_k, π_k) of (4.7) has $d_k = -W_k G_k \pi_k$ and $z_k = f(x_k) - \pi_k^T G_k^T W_k G_k \pi_k$.

After the computation of the search direction d_k, a standard backtracking line search is performed to find a step size α_k satisfying the following sufficient decrease condition

$$f(x_k + \alpha_k d_k) \le f(x_k) - \eta \alpha_k d_k^T H_k d_k. \tag{4.38}$$

The Algorithm for ℓ_1 Regularization is presented as Algorithm 12 below.

4.3.2 Global Convergence Analysis

We make the following assumptions about the function f_s and the Hessian approximation H_k throughout our global convergence analysis.

Assumption 4.3.1. *The function $f_s : \mathbb{R}^n \to \mathbb{R}$ is bounded below, continuously differentiable, and the gradient ∇f_s is Lipschitz continuous.*

Algorithm 12 Algorithm for ℓ_1 Regularization (L1R)

```
1: (Initialization): Choose a sampling radius reduction factor $\psi \in (0,1)$, sufficient decrease constant $\eta \in (0,1)$, line search backtracking constant $\kappa \in (0,1)$, and tolerance parameter $\nu > 0$. Choose an initial iterate $x_0$, an initial sampling radius $\epsilon_0 > 0$, and set $k \leftarrow 0$.
2:  (Hessian update): Set $H_k \succ 0$ as an approximation of the Hessian of $f$ at $x_k$.
3:  (Search direction computation): Compute $(z_k, d_k)$ solving (4.37).
4:  (Sampling radius update): If $\min\{\|d_k\|^2, d_k^T H_k d_k\} \le \epsilon_k$, then set $x_{k+1} \leftarrow x_k$, $\alpha_k \leftarrow 1$, and $\epsilon_{k+1} \leftarrow \psi\epsilon_k$ and go to step 7.
5: (Backtracking line search): Set $\alpha_k$ as the largest value in $\{\kappa^0, \kappa^1, \kappa^2, \dots\}$ such that (4.38) is satisfied.
6: (Iterate update): Set $\epsilon_{k+1} \leftarrow \epsilon_k$ and $x_{k+1} \leftarrow x_k + \alpha_k d_k$.
7:  (Iteration increment): Set $k \leftarrow k+1$ and go to step 2.
```

Assumption 4.3.2. *There exist $\overline{\xi} \ge \underline{\xi} > 0$ such that, for all k and $d \in \mathbb{R}^n$, we have*

$$\underline{\xi}\|d\|^2 \le d^T H_k d \le \overline{\xi}\|d\|^2.$$

The result we prove is the following.

Theorem 4.3.3. *L1R produces infinite sequences of iterates $\{x_k\}$ and sampling radius $\{\epsilon_k\}$. The sequence of sampling radius $\{\epsilon_k\} \to 0$ and every cluster point of $\{x_k\}$ is stationary for f.*

We begin our analysis with the following Lemma 4.3.4 that shows the sufficient decrease condition (4.38) is well defined.

Lemma 4.3.4. *There exists $\alpha_k > 0$ that satisfies the sufficient decrease condition* (4.38).

Proof. We have showed

$$\nabla f(x_k)^T d_k \le -d_k^T H_k d_k. \tag{4.39}$$

Since $d_k^T H_k d_k > 0$, it follows that d_k is a direction of strict descent for f at x_k, so there exists $\alpha_k > 0$ such that (4.38) holds:

$$f(x_k + \alpha_k d_k) \le f(x_k) + \eta\alpha_k \nabla f(x_k)^T d_k \le f(x_k) - \eta\alpha_k d_k^T H_k d_k.$$

□

We now show a highly useful result about the sequence of decreases produced in f.

Lemma 4.3.5. *The following inequality holds for all k:*

$$f(x_{k+1}) \leq f(x_k) - \eta\underline{\xi}\|x_{k+1} - x_k\|\|d_k\|.$$

Proof. By Algorithm 12 and Assumption 4.3.2, we have

$$\|x_{k+1} - x_k\| = \alpha_k\|d_k\| \quad \text{and} \quad \underline{\xi}\|d\|^2 \leq d^T H_k d$$

which, along with the sufficient decrease condition (4.38), yield

$$\begin{aligned} f(x_{k+1}) - f(x_k) \leq& \ -\eta\alpha_k d_k^T H_k d_k \\ \leq& \ -\eta\alpha_k\underline{\xi}\|d_k\|^2 \\ \leq& \ -\eta\underline{\xi}\|x_{k+1} - x_k\|\|d_k\|, \end{aligned}$$

□

The next Lemma 4.3.6 shows that if two points x' and $\bar{x}$ are "close", the ϵ-subdifferential of $\|x\|_1$ at those two points can also be "close".

Lemma 4.3.6. *Let $\epsilon > 0$ and $x' \in \mathbb{R}^n$. Suppose $|x'_i| \neq \epsilon$ for all $i = 1, \dots, n$. There exists $\delta > 0$ and $\bar{x} \in \mathbb{R}^n$ such that, if $\|x' - \bar{x}\| < \delta$, then $\partial_\epsilon(\|\cdot\|_1)|_{x'} = \partial_\epsilon(\|\cdot\|_1)|_{\bar{x}}$.*

Proof. Pick $\delta = \delta_\epsilon(x') := \min\{||x'_i| - \epsilon| : 1 \leq i \leq n\}$. □

For the rest of analysis, we suppose $H_k = W_k = I$. Let $\epsilon > 0$ and $x \in \mathbb{R}^n$. Define

$$p_\epsilon(x) := \text{Proj}(-\nabla f_s(x)|\partial_\epsilon(\|\cdot\|_1)|_x).$$

Notice that the search direction d_k in step 3 of Algorithm 12 is characterized by

$$\begin{aligned} d_k &= -\text{Proj}(0|\nabla f_s(x_k) + \partial_{\epsilon_k}(\|\cdot\|_1)|_{x_k}) \\ &= -\nabla f_s(x_k) - \text{Proj}(-\nabla f_s(x_k)|\partial_{\epsilon_k}(\|\cdot\|_1)|_{x_k}) \\ &= -\nabla f_s(x_k) - p_{\epsilon_k}(x_k). \end{aligned}$$

In particular, notice that

$$d_k \neq -\operatorname{Proj}(0|\partial_{\epsilon_k} f(x_k)) \neq -\operatorname{Proj}(-\nabla f_s(x_k)|\partial_{\epsilon_k}(\|\cdot\|_1)|_{x_k}) \neq -\nabla f_s(x_k) - \operatorname{Proj}(0|\partial_{\epsilon_k}(\|\cdot\|_1)|_{x_k}).$$

Also, if x' is stationary for f, then $-\nabla f_s(x') \in \partial_{\epsilon'}(\|\cdot\|_1)|_{x'}$ for any $\epsilon' \geq 0$, then $p_{\epsilon'}(x') = -\nabla f_s(x')$ for any $\epsilon' \geq 0$. Therefore, if Algorithm 12 arrives at a stationary point x', the corresponding search direction computed is

$$d' = -\nabla f_s(x') - p_{\epsilon'}(x') = -\nabla f_s(x') + \nabla f_s(x') = 0.$$

The next Lemma 4.3.7 showes that if two points x' and $\bar{x}$ are "close", the projections from the vector $-\nabla f_s(x_k)$ to the corresponding ϵ-subdifferential can be also "close". This result will be useful as we mentioned the relationship between the projection and the search direction. If we are at a stationary point, the search direction is zero. If we are "close" to a stationary point, the search direction can be "close" to zero.

Lemma 4.3.7. *Let $\epsilon > 0$ and $x' \in \mathbb{R}^n$. Suppose $|x'_i| \neq \epsilon$ for all $i = 1, \dots, n$. There exists $\delta > 0$ and $\bar{x} \in \mathbb{R}^n$ such that, if $\|x' - \bar{x}\| < \delta$, then $\|p_\epsilon(x') - p_\epsilon(\bar{x})\| < L\delta$, where L is the Lipschitz constant of ∇f_s.*

Let d' and $\bar{d}$ be the corresponding search directions computed in step 3 of Algorithm 12 at points x' and $\bar{x}$, respectively, with radius $\epsilon > 0$. Then $\|d' - \bar{d}\| < 2L\delta$.

Proof. Pick δ in Lemma 4.3.6 such that $\|x' - \bar{x}\| < \delta$ and $\partial_\epsilon(\|\cdot\|_1)|_{x'} = \partial_\epsilon(\|\cdot\|_1)|_{\bar{x}}$. By the fact that projection is nonexpansive and the Assumption 4.3.1 that f_s has Lipschitz gradient, we have

$$\|p_\epsilon(x') - p_\epsilon(\bar{x})\| \leq \|\nabla f_s(x') - \nabla f_s(\bar{x})\| \leq L\|x' - \bar{x}\| < L\delta.$$

Similarly, we have

$$\|d' - \bar{d}\| = \|-\nabla f_s(x') - p_\epsilon(x') + \nabla f_s(\bar{x}) + p_\epsilon(\bar{x})\| \leq \|\nabla f_s(\bar{x}) - \nabla f_s(x')\| + \|p_\epsilon(\bar{x}) - p_\epsilon(x')\| < 2L\delta.$$

□

We are now prepared to prove Theorem 4.3.3.

Proof. By Assumption 4.3.1, we know f is bounded below. Then, we have from the sufficient decrease consition (4.38), and the inequality in Lemma 4.3.5 that

$$\sum_{k=0}^{\infty} \alpha_k d_k^T H_k d_k < \infty, \quad \text{and} \tag{4.40a}$$

$$\sum_{k=0}^{\infty} \|x_{k+1} - x_k\| \|d_k\| < \infty. \tag{4.40b}$$

We continue by considering two cases, the first of which has two subcases.

Case 1: Suppose for contradiction that there exists $k' \geq 0$ such that $\epsilon_k = \epsilon' > 0$ for all $k \geq k'$. According to step 4, this occurs only if

$$\|d_k\|^2 \geq \min\{\|d_k\|^2, d_k^T H_k d_k\} > \epsilon' \text{ for all } k \geq k'. \tag{4.41}$$

In conjunction with (4.40), this implies $\alpha_k \to 0$ and $x_k \to x'$ for some x'.

Case 1a: : If x' is stationary for f, then $d' = 0$. Let $\delta = \min\{\delta_{\epsilon'}(x'), \sqrt{\epsilon'}/(2L)\}$, where L is the Lipschitz constant of ∇f_s. Since $x_k \to x'$, there exists $k'' > k'$ such that $\|x' - x_k\| < \delta$ for all $k \geq k''$. By Lemma 4.3.7, we have

$$\|d_k - d'\| = \|d_k\| < 2L\delta \leq \sqrt{\epsilon'}, \tag{4.42}$$

whenever $k \geq k''$. This contradicts with (4.41).

Case 1b: : If x' is not stationary, then for all $k \geq k'$, by construction of the sufficient decrease condition (4.38)

$$f(x_k + \kappa^{-1}\alpha_k d_k) - f(x_k) > -\eta\kappa^{-1}\alpha_k d_k^T H_k d_k,$$

whereas Lebourg's mean value yields the existence of $\tilde{x}_k \in [x_k + \kappa^{-1}\alpha_k d_k, x_k]$ and $\tilde{v}_k \in$

$\partial f(\tilde{x}_k)$ such that

$$f(x_k + \kappa^{-1}\alpha_k d_k) - f(x_k) = \kappa^{-1}\alpha_k \tilde{v}_k^T d_k.$$

Combining the above two inequalities yields

$$\tilde{v}_k^T d_k > -\eta d_k^T H_k d_k. \tag{4.43}$$

Since $\tilde{v}_k \in \partial f(\tilde{x}_k) = \nabla f_s(\tilde{x}_k) + \partial(\|\cdot\|_1)|_{\tilde{x}_k} \subseteq \nabla f_s(\tilde{x}_k) + \partial_{\epsilon_k}(\|\cdot\|_1)|_{\tilde{x}_k}$ and $\alpha_k \to 0$, by Lemma 4.3.6, there exists $k'' > k'$ and $v_k \in \nabla f_s(x_k) + \partial_{\epsilon_k}(\|\cdot\|_1)|_{x_k}$ such that $\partial_{\epsilon_k}(\|\cdot\|_1)|_{\tilde{x}_k} = \partial_{\epsilon_k}(\|\cdot\|_1)|_{x_k}$ and

$$\|\tilde{v}_k - v_k\| \leq \|\nabla f_s(\tilde{x}_k) - \nabla f_s(x_k)\| \leq L\|\kappa^{-1}\alpha_k d_k\|. \tag{4.44}$$

Since $v_k \in \nabla f_s(x_k) + \partial_{\epsilon_k}(\|\cdot\|_1)|_{x_k}$ and recall that $d_k = -\operatorname{Proj}(0|\nabla f_s(x_k) + \partial_{\epsilon_k}(\|\cdot\|_1)|_{x_k})$, then we have

$$v_k^T d_k \leq -\|d_k\|^2. \tag{4.45}$$

Since we assume $H_k = W_k = I$, subtracting (4.45) from (4.43), and with (4.44) , we have

$$(1-\eta)\|d_k\|^2 < (\tilde{v}_k - v_k)^T d_k \leq \|\tilde{v}_k - v_k\|\|d_k\| \leq L\kappa^{-1}\alpha_k\|d_k\|^2,$$

namely,

$$\alpha_k \geq \kappa(1-\eta)/L.$$

whenever $k \geq k''$. This contradicts with that $\alpha_k \to 0$.

Case 2: Suppose $\{\epsilon_k\} \to 0$ and $\{x_k\}$ has a cluster point x'. We want to show that x' is stationary for $\tilde{f}$. The proof is exactly that of [18, Theorem 4.2, Case 2]. □

4.3.3 Numerical Experiments

In this section, we compare our proposed algorithm `L1R` with an iterative shrinkage-thresholding algorithm (`ISTA`) proposed in [2]. Both algorithms are implemented in Matlab. We terminate `L1R` when optimality conditions $\|d_k\| \leq \epsilon_k \leq 10^{-6}$ are satisfied. For fair comparison, we also implement a subroutine in `ISTA` to compute a search direction d_k by solving a QO subproblem, and then terminate `ISTA` when the same optimality conditions are satisfied. We consider four algorithm variations described below.

- `L1R-H`: `L1R` with H_k being the exact Hessian of f_s at x_k;
- `L1R-I`: `L1R` with $H_k = I$;
- `ISTA-constant`: `ISTA` with constant stepsize in Section 3 of [2];
- `ISTA-backtrack`: `ISTA` with backtracking in Section 3 of [2].

We solve the problem (4.34) with f_s being randomly generated convex quadratics. We test for problem dimension $n = 10, 20, 50$, each with 10 runs. We consider the number of iterations required to arrive at optimality as the performance measure. We present the numerical results in the performance profile in Figure 4.4. Based on the profile, we have two observations. First, `L1R-H` outperforms all other algorithm variations; and this makes sense since `L1R-H` makes use of the exact Hessian of the smooth function f_s. Second, `L1R-I` is competitive with `ISTA-backtrack` in terms of both efficiency and robustness.

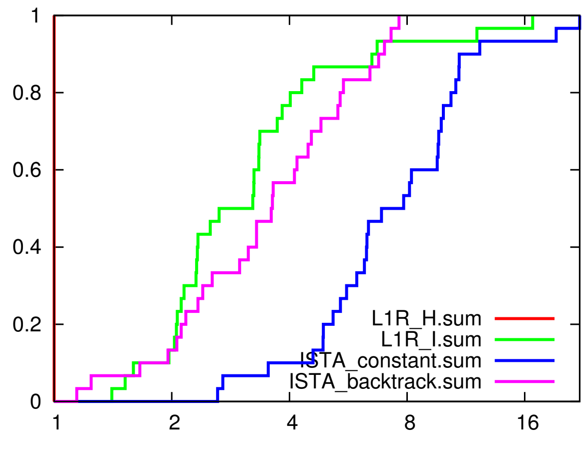


Figure 4.4: Performance profile comparing `L1R` and `ISTA`

Chapter 5

Conclusion

In this thesis, we consider unconstrained minimization problems in which the objective functions are not necessarily smooth or convex. This is an important and challenging class of optimization problems; and it is desirable to design efficient algorithms for solving this type of problems.

First, we propose an adaptive gradient sampling (AGS) algorithm, which is based on a recently developed technique known as the gradient sampling (GS) algorithm. Our AGS algorithm improves the computational efficiency of GS in critical ways. We achieve this goal by implementing two strategies: adaptive sampling and approximating the Hessian. We obtained two benefits by sampling gradients adaptively. The first benefit is that the number of gradient evaluations required in AGS is significantly smaller than that in GS. The second benefit is that the QO solver can be warm started because some of the gradients corresponding to the active constraints of the previous QO subproblem are still in the current sample set. We show in our numerical experiments that adaptive sampling allows the algorithm to make much more progress toward a solution within a fixed number of gradient evaluations.

We propose two strategies for approximating Hessian approximations. The first strategy is similar to a limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) updating strategy that is typical in smooth optimization, but differs in that we make use of gradient information at sample points as well as iterates. The basic idea of the second strategy is to produce model functions that overestimate the true objective function by incorporating the function information at sample points. We control both of the updating strategies so that the Hessian approximations are bounded, which is required for our convergence analysis. Our numerical experiments illustrate that our Hessian approximation strategies further enhance the algorithm's ability to progress toward a solution within a given amount of computational effort.

Second, we propose a BFGS gradient sampling (BFGS-GS) algorithm, which is a hybrid between a standard Broyden-Fletcher-Goldfarb-Shanno (BFGS) and the AGS method. The BFGS-GS algorithm is more efficient than our previously proposed AGS algorithm in that it typically behaves as an unadulterated BFGS algorithm for the majority of the iterations when applied to solve many problem instances. In addition, the BFGS-GS algorithm possesses not only practical but also theoretical advantages. Throughout, the algorithm dynamically employs the AGS strategy in order to provide a practical stationarity certificate as well as global convergence guarantees. Our numerical experiments show that our implementation of the BFGS-GS algorithm is competitive with (and in some ways) outperforms other available software.

Finally, we propose a few additional extensions of the GS framework—one in which we merge GS ideas with those from bundle methods, one in which we incorporate smoothing techniques in order to minimize potentially non-Lipschitz objective functions, and one in which we tailor GS methods for solving regularization problems. For all the proposed algorithm extensions, we write algorithm descriptions in detail. In addition, we prove global convergence guarantees under suitable assumptions. Moreover, we solve test problems to illustrate the performance of our algorithms.

In summary, we have proposed various randomized algorithms for nonsmooth nonconvex optimization, based on the gradient sampling idea. Possible directions to future work include the following: tune parameters to obtain better performance of our algorithmic extensions; perform more numerical experiments with practical application problems in nonsmooth optimization; extend the proposed algorithms to handle constraints.

Bibliography

[1] K. M. Anstreicher and J. Lee. A Masked Spectral Bound for Maximum-Entropy Sampling. In *MODA 7 - Advances in Model-Oriented Design and Analysis*, pages 1–10, Berlin, Germany, 2004.

[2] A. Beck and M. Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[3] A. Ben-Tal and A. Nemirovski. Robust Optimization - Methodology and Applications. *Mathematical Programming, Series B*, 92:453–480, 2002.

[4] D. P. Bertsekas. *Convex Optimization Theory.* Athena Scientific, Nashua, NH, USA, 2009.

[5] J. F. Bonnans, J. ch. Gilbert, C. Lemarechal, and C. A. Sagastizabal. *Numerical Optimization.* Springer-Verlag, Berlin, Germany and New York, NY, USA, 2006.

[6] C. G. Broyden. The Convergence of a Class of Double-Rank Minimization Algorithms. *Journal of the Institute of Mathematics and Its Applications*, 6(1):76–90, 1970.

[7] J. V. Burke, D. Henrion, A. S. Lewis, and M. L. Overton. HIFOO - A MATLAB Package for Fixed-order Controller Design and H-infinity Optimization. In *Proceedings of the IFAC Symposium on Robust Control Design*, Haifa, Israel, 2006.

[8] J. V. Burke, A. S. Lewis, and M. L. Overton. Approximating Subdifferentials by Random Sampling of Gradients. *Mathematics of Operations Research*, 27(3):567–584, 2002.

[9] J. V. Burke, A. S. Lewis, and M. L. Overton. A Robust Gradient Sampling Algorithm for Nonsmooth, Nonconvex Optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005.

[10] J. V. Burke, A. S. Lewis, and M. L. Overton. A Robust Gradient Sampling Algorithm for Nonsmooth, Nonconvex Optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005.

[11] J. V. Burke, A. S. Lewis, and M. L. Overton. Pseudospectral Components and the Distance to Uncontrollability. *SIAM Journal on Matrix Analysis and Applications*, 26(2):350–361, 2005.

[12] E. J. Candes and T. Tao. Near Optimal Signal Recovery from Random Projections: Universal Encoding Strategies. *IEEE Transactions on Informaton Theory*, 52:5406–5425, 2006.

[13] X. Chen. Smoothing Methods for Nonsmooth, Nonconvex Optimization. *Mathematical Programming, Series B*, 134(1):71–99, 2012.

[14] X. Chen, L. Niu, and Y. Yuan. Optimality Conditions and a Smoothing Trust Region Newton Method for Non-Lipschitz Optimization. *SIAM Journal on Optimization*, 23(3):1528–1552, 2013.

[15] F. H. Clarke. *Optimization and Nonsmooth Analysis*. Canadian Mathematical Society Series of Monographs and Advanced Texts. John Wiley & Sons, New York, NY, USA, 1983.

[16] F. E. Curtis and M. L. Overton. A Sequential Quadratic Programming Algorithm for Nonconvex, Nonsmooth Constrained Optimization. *SIAM Journal on Optimization*, in second round of review; original submission, 2009.

[17] F. E. Curtis and M. L. Overton. A Sequential Quadratic Programming Algorithm for Nonconvex, Nonsmooth Constrained Optimization. *SIAM Journal on Optimization*, 22(2):474–500, 2012.

[18] F. E. Curtis and X. Que. An Adaptive Gradient Sampling Algorithm for Nonsmooth Optimization. *Optimization Methods and Software*, 2012.

[19] E. Dolan and J. Moré. Benchmarking Optimization Software with Performance Profiles. *Mathematical Programming*, 91:201–213, 2002.

[20] D. L. Donoho. Compressed Sensing. *IEEE Transactions on Informaton Theory*, 52:1289–1306, 2006.

[21] R. Fletcher. A New Approach to Variable Metric Algorithms. *Computer Journal*, 13(3):317–322, 1970.

[22] D. Goldfarb. A Family of Variable Metric Updates Derived by Variational Means. *Mathematics of Computation*, 24(109):23–26, 1970.

[23] D. Goldfarb and G. Iyengar. Robust Portfolio Selection problems. *Mathematics of Operations Research*, 28:1–38, 2003.

[24] A. A. Goldstein. Optimization of Lipschitz Continuous Functions. *Mathematical Programming*, 13(1):14–22, 1977.

[25] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, third edition, 1996.

[26] C. Greif and J. Varah. Minimizing the Condition Number for Small Rank Modifications. *SIAM Journal on Matrix Analysis and Applications*, 29(1):82–97, 2006.

[27] S. Guenter, M. Sempf, P. Merkel, E. Strumberger, and C. Tichmann. Robust Control of Resistive Wall Modes Using Pseudospectra. *New Journal of Physics*, 11:1–40, 2009.

[28] S. Gumussoy, D. Henrion, M. Millstone, and M. L. Overton. Multiobjective Robust Control with HIFOO 2.0. In *Proceedings of the IFAC Symposium on Robust Control Design*, Haifa, 2009.

[29] A. M. Gupal. On a Minimization Method for Almost-Differentiable Functions. *Cybernetics*, 13(1):115–117, 1977.

[30] M. Haarala. *Large-Scale Nonsmooth Optimization: Variable Metric Bundle Method with Limited Memory.* PhD thesis, University of Jyväskylä, 2004.

[31] M. Haarala, K. Miettinen, and M. M. Mäkelä. New Limited Memory Bundle Method for Large-Scale Nonsmooth Optimization. *Optimization Methods and Software*, 19(6):673–692, 2004.

[32] W. Hare and M. Macklem. Derivative-Free Optimization Methods for Finite Minimax Problems. *Optimization Methods and Software*, 28(2):300–312, 2013.

[33] W. Hare and J. Nutini. A Derivative-Free Approximate Gradient Sampling Algorithm for Finite Minimax Problems. *Computational Optimization and Applications*, 56(1):1–38, 2013.

[34] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms II.* A Series of Comprehensive Studies in Mathematics. Springer-Verlag, New York, NY, USA, 1993.

[35] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1990.

[36] N. Karmitsa. Limited Memory Bundle Method. `http://napsu.karmitsa.fi/lmbm/`, 2014. Accessed: 2014-05-12.

[37] K. C. Kiwiel. *Methods of Descent for Nondifferentiable Optimization.* Lecture Notes in Mathematics. Springer-Verlag, New York, NY, USA, 1985.

[38] K. C. Kiwiel. A Method for Solving Certain Quadratic Programming Problems Arising in Nonsmooth Optimization. *IMA Journal of Numerical Analysis*, 6(2):137–152, 1986.

[39] K. C. Kiwiel. Convergence of the Gradient Sampling Algorithm for Nonsmooth Nonconvex Optimization. *SIAM Journal on Optimization*, 18(2):379–388, 2007.

[40] K. C. Kiwiel. A Nonderivative Version of the Gradient Sampling Algorithm for Nonsmooth Nonconvex Optimization. *SIAM Journal on Optimization*, 20(4):1983–1994, 2010.

[41] C. Lemaréchal. A View of Line-Searches. In *Optimization and Optimal Control: Proceedings of a Conference Held at Oberwolfach, March 16–22, 1980*, pages 59–78, Berlin, Germany, 1981.

[42] A. S. Lewis. Local Structure and Algorithms in Nonsmooth Optimization. In F. Jarre, C. Lemaréchal, and J. Zowe, editors, *Optimization and Applications*, pages 104–106. Mathematisches Forschungsinstitut Oberwolfach, Oberwolfach, Germany, 2005.

[43] A. S. Lewis and M. L. Overton. Nonsmooth Optimization via BFGS. `http://cs.nyu.edu/overton/papers/pdffiles/bfgs_inexactLS.pdf`, 2010. Accessed: 2015-04-20.

[44] A. S. Lewis and M. L. Overton. Nonsmooth Optimization via Quasi-Newton Methods. *Mathematical Programming*, 141(1–2):135–163, 2013.

[45] L. Lukšan, M. Tůma, M. Šiška, J. Vlček, and N. Ramešová. UFO 2002: Interactive System for Universal Functional Optimization. Technical Report 883, Institute of Computer Science, Academy of Sciences of the Czech Republic, 2002.

[46] R. Mifflin. An Algorithm for Constrained Optimization with Semismooth Functions. *Mathematics of Operations Research*, 2(2):191–207, 1977.

[47] J.J. Moré and D.C. Sorensen. *SIAM J. Sci. Stat. Comput.*

[48] J. Nocedal. Updating Quasi-Newton Matrices With Limited Storage. *Mathmatics of Computation*, 35(151):773–782, 1980.

[49] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, second edition, 2006.

[50] M. L. Overton. HANSO: Hybrid Algorithm for Non-Smooth Optimization. `http://www.cs.nyu.edu/faculty/overton/software/hanso/`, 2014. Accessed: 2014-05-12.

[51] R. T. Rockafellar. *Convex Analysis.* Princeton Landmarks in Mathematics and Physics. Princeton University Press, Princeton, NJ, USA, 1970.

[52] R. T. Rockafellar. Nonsmooth Optimization. In *Mathematical Programming: The State of the Art 1994*, pages 248–258, Ann Arbor, MI, USA, 1994.

[53] A. Ruszczynski. *Nonlinear Optimization*. Princeton University Press, first edition, 2006.

[54] C. Sanderson and R. Curtin. Armadillo C++ Linear Algebra Library. `http://arma.sourceforge.net/`, 2014. Accessed: 2014-05-12.

[55] D. F. Shanno. Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 24(111):647–656, 1970.

[56] A. Skajaa. Limited Memory BFGS for Nonsmooth Optimization. Master's thesis, New York University, 2010.

[57] L. A. Steen and J. A. Seebach. *Counterexamples in Topology*. Dover Publications, first edition, 1995.

[58] J. Vanbiervliet, B. Vandereycken, W. Michiels, S. Vandewalle, and M. Diehl. The Smoothed Spectral Abscissa for Robust Stability Optimization. *SIAM Journal on Optimization*, 20:156–171, 2009.

[59] J. Vanbiervliet, K. Verheyden, W. Michiels, and S. Vandewalle. A Nonsmooth Optimisation Approach for the Stabilisation of Time-delay Systems. *ESAIM: Control, Optimization and Calculus of Variations*, 14:478–493, 2008.

[60] G. A. Watson. Data Fitting Problems with Bounded Uncertainties in the Data. *SIAM Journal on Matrix Analysis and Applications*, 22:1274–1293, 2001.

[61] P. Wolfe. A Method of Conjugate Subgradients for Minimizing Nondifferentiable Functions. *Mathematical Programming Study*, pages 145–173, 1975.

Appendix A

QO Subproblem Solver

In this appendix, we discuss a specialized technique for solving the primal-dual pair (2.3) and (2.4) during step 4 of AGS. Specifically, we present an approach for solving (2.4) that follows the technique described in [38] for solving a similar subproblem. The differences are that, in our subproblem, there is no linear term in the objective, and we allow for the use of general positive definite Hessian approximations. We drop iteration number subscripts in this section. Now, subscripts are used to indicate column number of a matrix or element number(s) in a vector.

The benefits of our QO solver are that it can produce more accurate solutions than, say, an interior-point method, and we can easily warm-start the approach to take advantage of the fact that the columns of G often do not change drastically between iterations of AGS. The algorithm also carefully handles ill-conditioning in G, which is extremely important in our context as the columns of G come from the calculation of gradients of f at points that may be very close to one another.

By (2.23), we can write necessary and sufficient conditions for (2.4) as

$$g_j^T WG\pi - (\pi^T G^T WG\pi) \geq 0, \; j = 1, \ldots, q \tag{A.1a}$$

$$e^T \pi = 1, \; \pi \geq 0. \tag{A.1b}$$

A vector π satisfying these conditions is the unique optimal solution to (2.4), with which the unique optimal solution d to (2.3) can be computed as $d \leftarrow -WG\pi$.

It is clear from (A.1b) that any solution π to (A.1) must have at least one positive entry. Thus, the method commences with a nonempty estimate $\mathcal{A} \subseteq \{1, \ldots, q\}$ of the optimal *positive set*, i.e., the indices corresponding to positive values of π in the solution to (A.1). Denoting $\widehat{G}$ and $\widehat{\pi}$, respectively, as the ordered submatrix of G and the ordered subvector of π corresponding to the indices in the positive-set estimate $\mathcal{A}$, we begin with $\widehat{\pi} \geq 0$ solving

$$\min_{\pi} \ \tfrac{1}{2}\pi^T \widehat{G}^T W \widehat{G} \pi \quad \text{s.t.} \ \ e^T \pi = 1, \tag{A.2}$$

i.e., $\widehat{\pi} \geq 0$ where $(\widehat{\pi}, \widehat{v})$ is the unique solution to

$$\begin{bmatrix} \widehat{G}^T W \widehat{G} & e \\ e^T & 0 \end{bmatrix} \begin{bmatrix} \pi \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{A.3}$$

(If for a given $\mathcal{A}$ the solution $\widehat{\pi}$ of (A.2) does not satisfy $\widehat{\pi} \geq 0$, then $\mathcal{A}$ is replaced by $\{i\}$ for some $i \in \{1, \ldots, q\}$ and (A.2) is re-solved to obtain $\widehat{\pi}$ with $\widehat{\pi}_i = 1$ and $\widehat{\pi}_{\{1,\ldots,q\}\backslash\{i\}} = 0$.) Assuming always that the elements of the solution corresponding to $\{1, \ldots, q\}\backslash\mathcal{A}$ are set to zero, this solution is either optimal or, for some $j \notin \mathcal{A}$,

$$g_j^T W \widehat{G} \widehat{\pi} - (\widehat{\pi}^T \widehat{G}^T W \widehat{G} \widehat{\pi}) < 0. \tag{A.4}$$

An improvement in the objective of (2.4) can then be obtained by including j in $\mathcal{A}$. If the direct inclusion of j in $\mathcal{A}$ yields a new $\widehat{G}$ such that $\begin{bmatrix} e \ \widehat{G}^T \end{bmatrix}$ has full row rank, then $\mathcal{A}$ is simply augmented to include j. (Determining whether or not this matrix has full row rank can be done by solving the least-squares system

$$(\widehat{G}^T W \widehat{G} + ee^T)\widetilde{\pi} = e + \widehat{G}^T W g_j \tag{A.5}$$

and then determining whether $e^T \widetilde{\pi} = 1$ and $\widehat{G}\widetilde{\pi} = g_j$.) Otherwise, j is swapped with an appropriate element in $\mathcal{A}$ to avoid rank-deficiency. In either case, a new trial solution $(\overline{\pi}, \overline{v})$ is obtained by solving (A.3). If $\overline{\pi} > 0$, then $\widehat{\pi} \leftarrow \overline{\pi}$ becomes the new solution estimate and the above procedures are repeated. Otherwise, a step from $\widehat{\pi}$ in the direction of $\overline{\pi}$ is made until some element hits zero (say, corresponding to the $\overline{j}$th column of G), in which case $\overline{j}$ is removed from $\mathcal{A}$ and (A.3) is reformulated and re-solved for the new positive set estimate.

A complete description of our subproblem solver, ASQO, is presented as Algorithm 13 on page 135. The algorithm returns a vector $\widehat{\pi}$ corresponding to $\mathcal{A}$. This vector is to be permuted and augmented with zeros in the appropriate entries to construct the optimal π from which the optimal primal solution d is obtained.

Algorithm 13 Active-Set Quadratic Optimization Subproblem Solver (ASQO)

```
1: (Initialization) Choose $\mathcal{A}$ such that, with $\widehat{G}$ as the submatrix of $G$ corresponding to
   the indices in $\mathcal{A}$, the solution $(\widehat{\pi}, \widehat{v})$ of (A.3) has $\widehat{\pi} \geq 0$.
2: (Termination check) If (A.1a) holds, then terminate. Otherwise, choose an index
   $j \in \{1, \ldots, q\} \backslash \mathcal{A}$ such that (A.4) holds.
3: (Rank-deficiency check) Solve (A.5) for $\widetilde{\pi}$. If $\widetilde{\pi}^T[e\ \widehat{G}^T] = [1\ g_j^T]$, then go to step 5;
   otherwise, continue.
4: (Column augmentation) Append $j$ to $\mathcal{A}$, $g_j$ to $\widehat{G}$, and 0 to $\widehat{\pi}$. Go to step 6.
5: (Column exchange) Replace $\widehat{\pi}$ by $\widehat{\pi} - t\widetilde{\pi}$ where
       $$t = \min_i \{\widehat{\pi}_i / \widetilde{\pi}_i : \widetilde{\pi}_i > 0\}.$$
   Find some $i$ such that $\widehat{\pi}_i = 0$. Delete the $i$th index from $\mathcal{A}$, the $i$th column from $\widehat{G}$,
   and the $i$th component from $\widehat{\pi}$. Append $j$ to $\mathcal{A}$, $g_j$ to $\widehat{G}$, and $t$ to $\widehat{\pi}$.
6: (Subproblem solution) Solve (A.3) for $(\overline{\pi}, \overline{v})$. If $\overline{\pi} > 0$, then set $\widehat{\pi} = \overline{\pi}$ and go to step 2;
   otherwise, continue.
7: (Column deletion) Replace $\widehat{\pi}$ by $t\overline{\pi} + (1-t)\widehat{\pi}$ where
       $$t = \min\{1, \min_i \{\widehat{\pi}_i / (\widehat{\pi}_i - \overline{\pi}_i) : \overline{\pi}_i < 0\}\}.$$
   Find $i$ such that $\widehat{\pi}_i = 0$. Delete the $i$th index from $\mathcal{A}$, the $i$th column from $\widehat{G}$, and the
   $i$th component from $\widehat{\pi}$. Go to step 6.
```

Our implementation of ASQO actually maintains a Cholesky factorization of $(\widehat{G}^T W \widehat{G} + ee^T)$ that is updated during each iteration. Specifically, we maintain an upper triangular matrix R satisfying

$$R^T R = \widehat{G}^T W \widehat{G} + ee^T,$$

with which it can easily be verified that the solutions to (A.3) and (A.5) can be obtained, respectively, by solving

$$\left\{\begin{aligned} R^T r_1 =&\ e + \widehat{G}^T W g_j \\ R\widetilde{\pi} =&\ r_1 \end{aligned}\right\} \qquad \text{and} \qquad \left\{\begin{aligned} R^T r_2 =&\ e \\ R\overline{\pi} =&\ r_2/\|r_2\|^2. \end{aligned}\right\}.$$

The maintenance of R and calculation of the intermediate vectors above allows for sophisticated extensions of the rank-deficiency check in step (3) of ASQO so that it is less susceptible to numerical errors. We have implemented these extensions in our code, but suppress the details here as they are out of the scope of this thesis; see [38] for details.

Appendix B

Nonsmooth Test Problems

#	Name	Convexity	$f(x^*)$	Source
1	`MAXQ`	Y	0.0	[31]
2	`MXHILB`	Y	0.0	[31]
3	`CHAINED_LQ`	Y	$-\sqrt{2}(n-1)$	[31]
4	`CHAINED_CB3_I`	Y	$2(n-1)$	[31]
5	`CHAINED_CB3_II`	Y	$2(n-1)$	[31]
6	`ACTIVE_FACES`	N	0.0	[31]
7	`BROWN_FUNCTION_2`	N	0.0	[31]
8	`CHAINED_MIFFLIN_2`	N	$-34.795(n=50)$	[31]
9	`CHAINED_CRESCENT_I`	N	0.0	[31]
10	`CHAINED_CRESCENT_II`	N	0.0	[31]
#	Name	Convexity	$f(x^*)$	Source
11	`TEST29_2`	NA	NA	[45]
12	`TEST29_5`	NA	NA	[45]
13	`TEST29_6`	NA	NA	[45]
14	`TEST29_11`	NA	NA	[45]
15	`TEST29_13`	NA	NA	[45]
16	`TEST29_17`	NA	NA	[45]
17	`TEST29_19`	NA	NA	[45]
18	`TEST29_20`	NA	NA	[45]
19	`TEST29_22`	NA	NA	[45]
20	`TEST29_24`	NA	NA	[45]
#	Name	Convexity	$f(x^*)$	Source
21	`TILTED_NORM_COND`	NA	NA	[43]
22	`CPSF`	Y	NA	[43]
23	`NCPSF`	N	NA	[43]
24	`EIG_PROD`	NA	NA	[43]
25	`GREIF_FUN`	NA	NA	[26]
26	`NUC_NORM`	NA	NA	[56]

Table B.1: Number, name, convexity, $f(x^*)$, and source for nonsmooth test problems.

Biography

Xiaocun Que was born in Hubei, China. She received her B.S. in Logistics Engineering from Beijing University of Posts and Telecommunications (BUPT) in July 2008. She has been studying toward her Ph.D. in the Department of Industrial and Systems Engineering at Lehigh University since August 2009. She was awarded the RCEAS Emeritus Faculty Fellowship for the Fall 2009 and Spring 2010 semesters, and was awarded the Gotshall Fellowship for the Fall 2010 and Spring 2011 semesters. She worked as a Givens Associate in the Mathematics and Computer Science Division at Argonne National Laboratory in Summer 2013. She was a Rossin Doctoral Fellow for the Fall 2013 and Spring 2014 semesters. She is expected to receive her Ph.D. in Industrial Engineering in January 2016.