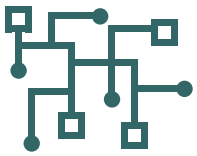

ISE

Industrial and
Systems Engineering

Gradient Sampling Methods with Inexact Subproblem Solves and Gradient Aggregation

Frank E. Curtis and Minhan Li

Department of Industrial and Systems Engineering, Lehigh University

COR@L Technical Report 20T-009

LEHIGH
UNIVERSITY

COR@L
COMPUTATIONAL OPTIMIZATION
RESEARCH AT LEHIGH

Gradient Sampling Methods with Inexact Subproblem Solves and Gradient Aggregation*

Frank E. Curtis[†1] and Minhan Li[‡1]

[1]Department of Industrial and Systems Engineering, Lehigh University

May 15, 2020

Abstract

Gradient sampling (GS) has proved to be an effective methodology for the minimization of nonsmooth, nonconvex objective functions. The most computationally expensive component of a contemporary GS method is the need to solve a convex quadratic subproblem in each iteration. In this paper, a strategy is proposed that allows the use of inexact solutions of these subproblems, which, as proved in the paper, can be incorporated without the loss of theoretical convergence guarantees. Numerical experiments show that by exploiting inexact subproblem solutions, one can consistently reduce the CPU time required by a GS method. Additionally, a strategy is proposed for aggregating gradient information after a subproblem is solved (inexactly), as has been exploited in bundle methods for nonsmooth optimization. It is proved that the aggregation scheme can be introduced without the loss of theoretical convergence guarantees. Numerical experiments show that incorporating this gradient aggregation approach further reduces the CPU time required by a GS method.

1 Introduction

The gradient sampling (GS) methodology has proved to be effective for solving nonsmooth, nonconvex minimization problems. Based on the conceptually simple idea of computing an approximate ϵ-steepest-descent direction at a point by finding the minimum-norm element of the convex hull of gradients evaluated at randomly generated nearby points, one can prove convergence to stationarity of a GS method under relatively loose assumptions. That said, the following are two ways in which implementations of GS methods could be more efficient:

- Each iteration of a GS method requires the solution of a convex quadratic subproblem (QP) for computing a search direction. The overall computational expense of a GS method could be reduced if one could terminate each call to a QP solver early, then employ the inexact QP solution as the search direction in the "outer" GS method. Such an inexact solution might cause a search direction to be less productive than if an exact QP solution were computed, meaning that more "outer" iterations may be required. However, as in other optimization algorithms that exploit inexact subproblem solutions, one might still obtain overall savings through reduced per-iteration costs.

- Bundle methods represent another important class of algorithms for nonsmooth minimization. It has been shown that implementations of bundle methods can be made significantly more efficient through

*This material is based upon work supported by the National Science Foundation under grant numbers CCF–1618717 and CCF–1740796.

[†]E-mail: frank.e.curtis@lehigh.edu

[‡]E-mail: mil417@lehigh.edu

the use of *subgradient aggregation*, wherein one can compress the information from a QP solution such that a subsequent QP can be solved rapidly. Implementations of GS methods could be made more efficient if such an idea could be incorporated.

In this paper, we propose enhancements to the GS methodology such that one can exploit inexact subproblem solutions and *gradient* aggregation. (We do not refer to *subgradient* aggregation since the GS methodology requires the identification of points at which the objective function is continuously differentiable, yielding gradients, when computing search directions.) We show techniques for exploiting these ideas that maintain the convergence guarantees of previously proposed GS methods. Implementations of our ideas in a C++ software package show that exploiting both inexact subproblem solutions and gradient aggregation can lead to consistently noticable reductions in the CPU time required by a GS method.

1.1 Literature Review

The GS methodology was introduced by Burke, Lewis, and Overton in [4]; see also [3]. Shortly after, in [23], Kiwiel showed an elegant convergence analysis of a GS method, and showed how the convergence guarantees could be maintained by multiple variations of the basic approach. Curtis and Que later showed in [9, 10] how one could sample gradients adaptively and introduce quasi-Newton Hessian approximations to maintain convergence guarantees while improving practical performance. (Here, as is common in the literature on quasi-Newton methods for solving nonsmooth optimization problems, we use the term "Hessian approximation" loosely; rather than as an approximate second-derivative matrix, it should merely be thought of as a matrix that approximates local changes in the gradient at points at which f is differentiable.) See also [11] for how to loosen the restrictions on the Hessian approximation scheme. A feature of the algorithms in all of these articles is that the analysis requires that the convex QP subproblems be solved exactly.

A recent article that proposes a method for reducing the costs associated with solving QPs in a gradient sampling method is [29]. In this work, the authors argue that a so-called "ideal" direction, which can be computed using a relatively inexpensive procedure, can be used in place of a QP solution when it is found to be sufficiently large in norm. The authors argue that convergence guarantees are maintained with this replacement, and show empirically that fewer QPs need to be solved. Our proposed approach is different from that in [29] in two main respects. First, rather than prescribe a formula for a particular direction that may be used, our algorithm involves conditions for an inexact QP solution that are more generic. This gives more computational flexibility to the algorithm. Second, whereas the algorithm in [29] still requires that some QPs be solved exactly—such as when the "ideal" direction is too small in norm, which occurs when approaching stationarity—our algorithm allows for inexact solutions of the QPs in all cases.

GS ideas have been extended in various ways, such as to attain good local convergence rate properties [16] and to solve constrained optimization problems [8, 18, 36]. See [5] for further discussion. Such extensions are beyond the scope of this article, wherein we focus on techniques for unconstrained minimization that ensure convergence from any arbitrarily chosen starting point. That said, our proposed enhancements could be employed in conjunction with these extensions.

Another prevailing methodology for solving nonsmooth optimization problems is the class of bundle methods, which have a long history [1, 20, 21, 22, 13, 14, 15, 17, 24, 27, 31, 32, 34, 35]. The technique employed in some bundle methods that is relevant for this paper is that of subgradient aggregation; see, e.g., [20]. The use of aggregation in this paper is similar, although the surrounding convergence analysis is different from that seen for bundle methods given the distinct differences in the convergence analyses of bundle and gradient sampling methodologies. For one thing, convergence analyses of gradient sampling methods are inherently probabilistic due to the random sampling of points during each iteration.

1.2 Notation

Throughout the paper, we consider the minimization problem

$$\min_{x \in \mathbb{R}^n} \; f(x), \tag{1}$$

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

Assumption 1.1. *The objective function f is bounded below over $\mathbb{R}^n$, locally Lipschitz on $\mathbb{R}^n$, and continuously differentiable on an open set $\mathcal{D}$ with full measure in $\mathbb{R}^n$.*

We propose gradient sampling algorithms, each of which is designed to produce an iterate sequence—i.e., $\{x_k\}$ with $x_k \in \mathbb{R}^n$ for all $k \in \mathbb{N}$—converging to stationarity of f, which is to say that any cluster point of $\{x_k\}$ is stationary for f. Here and throughout the paper, we refer to stationarity in the sense of Clarke [6]. Such stationarity for f can be defined as follows. By Rademacher's theorem, it follows under Assumption 1.1 that the subdifferential of f at $x \in \mathbb{R}^n$ is given by

$$\partial f(x) = \operatorname{conv}\left\{\lim_{k\to\infty} \nabla f(x_k) : \{x_k\} \to x \text{ with } x_k \in \mathcal{D} \text{ for all } k \in \mathbb{N}\right\}; \tag{2}$$

see, e.g., [6, Theorem 2.5.1]. For $\epsilon \in \mathbb{R}_{\geq 0}$, the ϵ-subdifferential of f at $x \in \mathbb{R}^n$ is

$$\partial_\epsilon f(x) = \operatorname{conv} \partial f(\mathbb{B}(x,\epsilon)), \quad \text{where} \quad \mathbb{B}(x,\epsilon) := \{\overline{x} \in \mathbb{R}^n : \|\overline{x} - x\|_2 \leq \epsilon\}. \tag{3}$$

One finds that $\partial_0 f(x) \equiv \partial f(x)$; see [12, Corollary 2.5]. A point $x \in \mathbb{R}^n$ is said to be ϵ-stationary for f if $0 \in \partial_\epsilon f(x)$ and is said to be stationary for f if $0 \in \partial f(x)$.

The first algorithm that we propose (see Algorithm 1 on page 24) has a nested loop. Iterations for the "outer" loop are indexed by $k \in \mathbb{N}$. We apply this iteration number subscript to other values—in addition to x_k—computed in the outer loop of the algorithm. Iterations for the "inner" loop are indexed by $j \in \mathbb{N}$. Quantities computed during the inner loop are denoted with a double-subscript to indicate outer and inner iteration numbers; e.g., $d_{k,j}$ is a vector corresponding to outer iteration $k \in \mathbb{N}$ and inner iteration $j \in \mathbb{N}$.

One could remove from Assumption 1.1 that f is bounded below, in which case the methods that we propose would terminate finitely at a stationary point for f or, with probability one, generate iterates that either converge to stationarity for f (see Theorems 2.1 and 3.1) or have objective values that diverge to $-\infty$. However, to focus on the more interesting setting, we include in Assumption 1.1 that f is bounded below. In addition, for the algorithms that we propose to be well posed, one need only have that f is (not necessarily continuously) differentiable in an open set with full measure in $\mathbb{R}^n$. However, a theoretical guarantee of convergence to stationarity requires that f be continuously differentiable over such a set, as we have included in Assumption 1.1. See [5] for further discussion.

1.3 Outline

In §2, we propose and analyze an algorithm that employs inexact subproblem solutions. In §3, we introduce the idea of gradient aggregation within a GS method and propose an algorithm that employs it while maintaining the convergence guarantees for the method presented in §2. Numerical experiments employing both techniques are presented in §4. Concluding remarks are given in §5.

2 GS Algorithm with Inexact Subproblem Solutions

We propose a gradient sampling algorithm that allows for the use of inexact subproblem solutions in each iteration. In this section, we present a statement of the proposed algorithm, then prove that iterates generated by the algorithm converge to stationarity with probability one. In our presentation, we focus on the components of the algorithm and analysis that are distinct from previous gradient sampling methods. Components that are not unique are summarized in Appendix A.

2.1 Algorithm Description

In iteration $k \in \mathbb{N}$ of our proposed algorithm, an iterate $x_k \in \mathcal{D}$ is available along with a sampling radius $\epsilon_k \in (0, \infty)$, a set of sample points

$$\mathcal{X}_k := \{x_{k,0}, x_{k,1}, \ldots, x_{k,p_k}\} \subset \mathbb{B}(x_k, \epsilon_k) \cap \mathcal{D} \text{ where } x_{k,0} \equiv x_k \text{ for some } p_k \in \mathbb{N},$$

and the corresponding matrix of gradients

$$G_k := \begin{bmatrix} \nabla f(x_{k,0}) & \nabla f(x_{k,1}) & \cdots & \nabla f(x_{k,p_k}) \end{bmatrix} \in \mathbb{R}^{n\times(p_k+1)}. \tag{4}$$

Given this matrix of gradients, a symmetric positive definite Hessian approximation H_k, and $W_k := H_k^{-1}$, the search direction is computed by approximately solving the primal-dual pair of quadratic optimization problems (QPs) given by

$$(P) := \left\{ \begin{array}{c} \min\limits_{(d,z)\in\mathbb{R}^{n+1}} \; z + \frac{1}{2}\|d\|^2_{H_k} \\ \text{s.t. } G_k^T d \le z\mathbb{1} \end{array} \right\} \quad \text{and} \quad (D) := \left\{ \begin{array}{c} \max\limits_{y\in\mathbb{R}^{p_k+1}} \; -\frac{1}{2}\|G_k y\|^2_{W_k} \\ \text{s.t. } \mathbb{1}^T y = 1,\; y \ge 0 \end{array} \right\}, \tag{5}$$

where, here and throughout the paper, we use $\mathbb{1}$ to denote a vector of ones. We assume that both H_k and W_k are available for all $k \in \mathbb{N}$. It is straightforward to maintain both approximations through the use of quasi-Newton techniques, as is done in our algorithms; see the subroutine stated as Algorithm 5.

Letting $(d_{k,*}, z_{k,*})$ denote the optimal solution of the primal subproblem (P) for each $k \in \mathbb{N}$, one finds that the solution component $d_{k,*}$ can be viewed as the minimizer of the piecewise quadratic function

$$\max_{i\in\{0,\ldots,p_k\}} \{\nabla f(x_{k,i})^T d\} + \tfrac{1}{2}\|d\|^2_{H_k}.$$

The optimal solution $y_{k,*}$ of the dual subproblem (D), on the other hand, can be viewed as the vector such that $G_k y_{k,*}$ is the least W_k-norm element of the convex hull of the columns of G_k, i.e., the W_k-projection of the origin onto this convex hull. The following lemma reveals important properties of these solutions.

Lemma 2.1. *For all $k \in \mathbb{N}$, either $(d_{k,*}, z_{k,*}) = (0, 0)$ and the origin lies in the convex hull of the columns of G_k, or $d_{k,*}$ is a direction of strict descent for f at x_k with*

$$\nabla f(x_k)^T d_{k,*} \le -d_{k,*}^T H_k d_{k,*} < 0. \tag{6}$$

In all cases, $d_{k,*} = -W_k G_k y_{k,*}$ and $\|G_k y_{k,*}\|_{W_k} = \|d_{k,*}\|_{H_k}$.

Proof. The properties follow from the Karush-Kuhn-Tucker optimality conditions for (5); see, e.g., [9, Eq. (27)] and [10, Lemma 2.2]. □ □

As our focus is on an algorithm that solves (5) approximately for all $k \in \mathbb{N}$, the statement of our algorithm is facilitated by defining, in each "outer" iteration (indexed by $k \in \mathbb{N}$), sequences of "inner" iterates of a solver for the primal-dual subproblems (5). For this, let $\{(d_{k,j}, z_{k,j})\}$ and $\{y_{k,j}\}$ be sequences of primal and dual iterates, respectively, generated when (5) is solved iteratively. Our gradient sampling algorithm requires that *both* primal and dual QP iterate sequences are generated. However, this should not be viewed as an expensive requirement. After all, motivated by Lemma 2.1, one may choose for a given $y_{k,j} \in \mathbb{R}^{p_k+1}$ to set

$$d_{k,j} \leftarrow -W_k G_k y_{k,j} \quad \text{and} \quad z_{k,j} \leftarrow \max_{i\in\{0,\ldots,p_k\}} \nabla f(x_{k,i})^T d_{k,j}, \tag{7}$$

in which case one need only generate a dual iterate sequence and a corresponding sequence of primal-feasible solutions can be obtained. In addition, to reduce computational expense, one does not need to evaluate (7) in each inner iteration; one might only evaluate it and check for termination only periodically and/or after an initial number of inner iterations have been performed. In any case, for the sake of generality, we define our algorithm to allow $\{d_{k,j}\} \neq \{-W_k G_k y_{k,j}\}$.

With respect to the QP solver, we merely assume that the following holds.

Assumption 2.1. *For all $k \in \mathbb{N}$, the primal and dual iterates when solving (5) satisfy $\{(d_{k,j}, z_{k,j}, y_{k,j})\} \to (d_{k,*}, z_{k,*}, y_{k,*})$. In addition, for all $j \in \mathbb{N}$, one has*

$$G_k^T d_{k,j} \le z_{k,j}\mathbb{1}, \quad \mathbb{1}^T y_{k,j} = 1, \quad \text{and} \quad y_{k,j} \ge 0,$$

i.e., $(d_{k,j}, z_{k,j}, y_{k,j})$ is primal-dual feasible for all $j \in \mathbb{N}$.

Under Assumption 2.1, the primal and dual iterates satisfy weak duality with respect to (5) for all $j \in \mathbb{N}$. In particular, defining the QP primal and dual objective functions $q_k : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ and $\theta_k : \mathbb{R}^{p_k+1} \to \mathbb{R}$, respectively, where

$$q_k(d, z) = z + \tfrac{1}{2}\|d\|^2_{H_k} \quad \text{and} \quad \theta_k(y) = -\tfrac{1}{2}\|G_k y\|^2_{W_k},$$

one has that $q_k(d_{k,j}, z_{k,j}) \geq \theta_k(y_{k,j})$ for all $j \in \mathbb{N}$.

Our algorithm with inexact subproblem solutions is stated as Algorithm 1. The statement of the algorithm focuses on its unique aspects related to the conditions that we require of inexact QP solutions. Other subroutines that we employ for the line search, iterate perturbation strategy (required by GS methods for theoretical purposes), sample set updates, and quasi-Newton updates are similar to those used in [10, 11]. Hence, we relegate them to Appendix A.

In Algorithm 1, each call to the QP solver terminates in one of two situations. If (10) holds, then one has obtained a dual iterate such that the corresponding convex combination of columns of G_k is sufficiently small in appropriate norms. In this case, one has identified that the current iterate may be sufficiently close to ϵ_k-stationarity, in which case the algorithm should reduce the sampling radius. On the other hand, if (11) holds along with either (12) or (13), then our analysis in the following subsection reveals that a sufficiently accurate QP solution yielding a direction of sufficient descent has been obtained. The condition (11) is motivated by Lemma 2.1, specifically (6), since $(d_k, y_k) = (-W_k G_k y_{k,j_\theta}, y_{k,j_\theta})$ yields

$$\nabla f(x_k)^T d_k = -\nabla f(x_k)^T W_k G_k y_k \overset{(11)}{\leq} -\kappa y_k^T G_k^T W_k G_k y_k = -\kappa d_k^T H_k d_k.$$

The role played by conditions (12) and (13), which make use of the values defined in (8) and (9), is explained in the following subsection.

Notice that an implementation of Algorithm 1 does not require storage and a search through all previous subproblem solutions when determining the indices in Line 8. One only needs to store the best (in terms of objective values) primal and dual solution estimates during the inner loop and employ these values when checking for termination of the inner loop. Line 8 is only written in this manner for ease of exposition, and to allow us to consider situations in which the inner iterations do not necessarily produce primal and dual subproblem solutions that have objective values that converge monotonically to the optimal value.

2.2 Inexactness conditions for the QP solver

Convergence analyses of gradient sampling methods rely on a fundamental property of any compact, convex set, call it $\mathcal{S} \subseteq \mathbb{R}^n$, that does not contain the origin. Intuitively, this property is that if $u \in \mathcal{S}$ is sufficiently close to the projection of the origin onto $\mathcal{S}$, then u makes a sufficiently acute angle (with respect to a given metric) with any $v \in \mathcal{S}$. Such a lemma appears as [4, Lemma 3.1] and [23, Lemma 3.1], and is proved in a more general setting as [10, Lemma 3.5]. Let us refer to a result of this type as an *angle lemma* for GS methods.

The conditions that we impose on inexact subproblem solutions are motivated by trying to ensure a property of this type, but in a more general setting than has previously been shown in the context of GS methods. Specifically, the lemma that we use is the following. In the lemma, we refer to the concept of a W-projection (with $W \succ 0$) of the origin onto a compact, convex set $\mathcal{S}$, i.e.,

$$P_W(\mathcal{S}) := \operatorname*{arg\,min}_{s \in \mathcal{S}} \|s\|_W. \tag{14}$$

The key generalization in the lemma can be seen in the inequality (16), which does not require that u is sufficiently close to the W-projection of the origin, but merely sufficiently close to a neighborhood of this projection. This generalization requires a substantial extension of the proof beyond that of [10, Lemma 3.5].

Lemma 2.2. *Suppose $\mathcal{S} \subseteq \mathbb{R}^n$ is a compact, convex set with $0 \notin \mathcal{S}$. For any $W \succ 0$, let $P_W(\mathcal{S})$ be the W-projection of the origin onto $\mathcal{S}$, and define*

$$\Omega_{\mathcal{S},W} := \max\{\|x\|_W : x \in \mathcal{S}\}/\|P_W(\mathcal{S})\|_W, \tag{15}$$

which is finite since $\mathcal{S}$ is compact. Consider any $(\beta, \varsigma) \in (0,1) \times (0,\infty)$ with

$$\beta(1+\sqrt{\varsigma^2+2\varsigma})^2 + \Omega_{\mathcal{S},W}\sqrt{\varsigma^2+2\varsigma} \in (0,1).$$

Then, there exists sufficiently small $\delta \in (0,\infty)$ such that, for any $(u,v) \in \mathcal{S} \times \mathcal{S}$ with

$$\|u\|_W \leq (1+\varsigma)\|P_W(S)\|_W + \delta, \tag{16}$$

it follows that $v^T W u > \beta \|u\|_W^2$.

Proof. To derive a contradiction, suppose that the implication is false, which is to say that for all $\delta \in (0,\infty)$ one can find $(u,v) \in \mathcal{S} \times \mathcal{S}$ with

$$\|u\|_W \leq (1+\varsigma)\|P_W(\mathcal{S})\|_W + \delta \;\text{ and }\; v^T W u \leq \beta \|u\|_W^2.$$

This means that one can define infinite sequences $\{u_i\} \subset \mathcal{S}$ and $\{v_i\} \subset \mathcal{S}$ such that

$$\|u_i\|_W \leq (1+\varsigma)\|P_W(\mathcal{S})\|_W + 1/i \;\text{ and }\; v_i^T W u_i \leq \beta \|u_i\|_W^2 \;\text{ for all }\; i \in \mathbb{N}. \tag{17}$$

For each $i \in \mathbb{N}$, let a_i be the point on the line segment $[P_W(\mathcal{S}), u_i]$ that is closest to u_i such that $\|a_i\|_W \leq \|P_W(\mathcal{S})\|_W + 1/i$ and let $b_i := u_i - a_i$. By convexity of $\mathcal{S}$, it follows that $\{a_i\} \subset \mathcal{S}$. The remainder of the proof involves proving certain properties of the sequences $\{a_i\}$ and $\{b_i\}$ that have been defined in this manner.

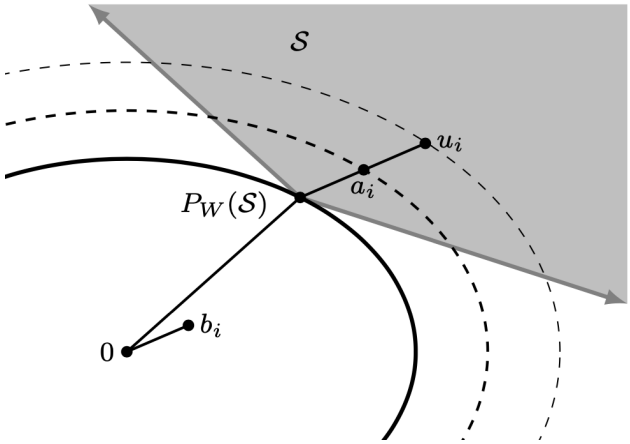

Figure 1: Illustration for generalized GS *angle lemma*, i.e., Lemma 2.2.

Let us prove an upper bound for $\|b_i\|_W$ with respect to $\|a_i\|_W$ for all $i \in \mathbb{N}$. There are two cases. If $\|u_i\|_W \leq \|P_W(\mathcal{S})\|_W + 1/i$, then $a_i = u_i$ and $b_i = 0$. Otherwise, $\|u_i\|_W > \|P_W(\mathcal{S})\|_W + 1/i$, which means $\|a_i\|_W = \|P_W(\mathcal{S})\|_W + 1/i$. This case is illustrated in Figure 1. In this case, the definitions of u_i and a_i imply

$$\begin{aligned} \|u_i\|_W &\leq \|P_W(\mathcal{S})\|_W + 1/i + \varsigma\|P_W(\mathcal{S})\|_W \\ &= \|a_i\|_W + \varsigma\|P_W(\mathcal{S})\|_W \leq (1+\varsigma)\|a_i\|_W. \end{aligned} \tag{18}$$

For all $i \in \mathbb{N}$, it follows by the definitions of a_i and b_i that

$$u_i - P_W(\mathcal{S}) = \upsilon_u b_i \text{ and } a_i - P_W(\mathcal{S}) = \upsilon_a b_i \text{ for some } (\upsilon_u, \upsilon_a) \in (0,\infty)^2.$$

Moreover, by [2, Proposition 1.1.8], the definition of $P_W(\mathcal{S})$ (as the W-projection of the origin onto $\mathcal{S}$), and the fact that $u_i \in \mathcal{S}$, one has that

$$0 \leq (u_i - P_W(\mathcal{S}))^T W P_W(\mathcal{S}) = \upsilon_u b_i^T W P_W(\mathcal{S}).$$

Combining these facts and $W \succ 0$, it follows that

$$\begin{aligned}\|u_i\|_W^2 = \|a_i + b_i\|_W^2 &= \|a_i\|_W^2 + \|b_i\|_W^2 + 2b_i^T W a_i \\ &= \|a_i\|_W^2 + \|b_i\|_W^2 + 2b_i^T W(P_W(S) + (a_i - P_W(S))) \\ &= \|a_i\|_W^2 + \|b_i\|_W^2 + 2b_i^T W P_W(S) + 2\upsilon_a b_i^T W b_i \\ &\geq \|a_i\|_W^2 + \|b_i\|_W^2,\end{aligned}$$

which along with (18) shows that

$$\begin{aligned}\|b_i\|_W &\leq \sqrt{\|u_i\|_W^2 - \|a_i\|_W^2} \\ &\leq \sqrt{(1+\varsigma)^2\|a_i\|_W^2 - \|a_i\|_W^2} = \sqrt{\varsigma^2 + 2\varsigma}\|a_i\|_W.\end{aligned} \tag{19}$$

Overall, one finds that the upper bound (19) holds for all $i \in \mathbb{N}$.

Since $\mathcal{S}$ is compact, it follows that $\{a_i\}$, $\{b_i\}$, $\{u_i\}$, and $\{v_i\}$ have convergent subsequences. (In the case of $\{b_i\}$, note that each element is the difference between two points in $\mathcal{S}$, meaning that $\{b_i\}$ is contained in a bounded set.) Hence, one can assume that these sequences themselves are convergent; i.e., $\{a_i\} \to a$, $\{b_i\} \to b$, $\{u_i\} \to u$, and $\{v_i\} \to v$ for some $(a, b, u, v) \in \mathcal{S} \times \mathbb{R}^n \times \mathcal{S} \times \mathcal{S}$. From (17), the fact that $u = a + b$, and (19), one finds that these values satisfy

$$\begin{aligned}v^T W u \leq \beta\|u\|_W^2 = \beta\|a+b\|_W^2 &\leq \beta(\|a\|_W + \|b\|_W)^2 \\ &\leq \beta(\|a\|_W + \sqrt{\varsigma^2+2\varsigma}\|a\|_W)^2 \\ &= \beta(1+\sqrt{\varsigma^2+2\varsigma})^2\|a\|_W^2.\end{aligned} \tag{20}$$

On the other hand, by the definition of $\{a_i\}$, it follows that $a = P_W(\mathcal{S})$, which is nonzero since $0 \notin \mathcal{S}$. Again applying [2, Proposition 1.1.8], it follows that $v^T W u = v^T W a + v^T W b \geq \|a\|_W^2 + v^T W b$. In addition, with (15) and (19), one finds that

$$v^T W b \geq -\|v\|_W\|b\|_W \geq -\Omega_{\mathcal{S},W}\|a\|_W\sqrt{\varsigma^2+2\varsigma}\|a\|_W = -\Omega_{\mathcal{S},W}\sqrt{\varsigma^2+2\varsigma}\|a\|_W^2.$$

Hence, overall, one finds that

$$v^T W u \geq \|a\|_W^2 + v^T W b \geq (1 - \Omega_{\mathcal{S},W}\sqrt{\varsigma^2+2\varsigma})\|a\|_W^2. \tag{21}$$

The fact that $\beta(1+\sqrt{\varsigma^2+2\varsigma})^2 + \Omega_{\mathcal{S},W}\sqrt{\varsigma^2+2\varsigma} \in (0,1)$ implies $1 - \Omega_{\mathcal{S},W}\sqrt{\varsigma^2+2\varsigma} > \beta(1+\sqrt{\varsigma^2+2\varsigma})^2$, from which it follows that (21) contradicts with (20). □ □

Our goal now is to prove two lemmas that motivate the use of (12) and (13) as stopping conditions for the inner loop in Algorithm 1. Specifically, under the assumption that $\theta_k(y_{k,*}) < 0$, each lemma shows that these conditions—(12) and (13), respectively, in the two lemmas—imply that

$$\begin{aligned}&0 > \theta_k(y_{k,j_\theta}) \geq (1+\sigma_k)^2\theta_k(y_{k,*}) \\ \iff\ &0 < \|G_k y_{k,j_\theta}\|_{W_k} \leq (1+\sigma_k)\|G_k y_{k,*}\|_{W_k}.\end{aligned} \tag{22}$$

Importantly, these algorithmic conditions imply that (22) holds *without knowledge of* $y_{k,*}$. The inequalities in (22) are important since they, along with Lemma 2.2 (c.f. (16)), play a central role in our convergence analysis in §2.3 for Algorithm 1.

Lemma 2.3. *Suppose that, in iteration $k \in \mathbb{N}$ of Algorithm 1, one has $\theta_k(y_{k,*}) < 0$. In addition, suppose that, during iteration $j \in \mathbb{N}$ of the inner loop of outer iteration $k \in \mathbb{N}$, one finds that* (12) *holds. Then,* (22) *holds.*

Proof. By weak duality for (5), one has that

$$\begin{aligned} \theta_k(y_{k,*}) - \theta_k(y_{k,j_\theta}) &\leq q_k(d_{k,j_q}, z_{k,j_q}) - \theta_k(y_{k,j_\theta}) \\ \text{and} \quad -q_k(d_{k,j_q}, z_{k,j_q}) &\leq -\theta_k(y_{k,*}). \end{aligned}$$

Combined with (12) and (8), it follows that

$$\theta_k(y_{k,*}) - \theta_k(y_{k,j_\theta}) \leq \tau_k(-\theta_k(y_{k,*})) = (\sigma_k^2 + 2\sigma_k)(-\theta_k(y_{k,*})),$$

which shows that (22) holds, as desired. □ □

When $\theta_k(y_{k,*}) < 0$, weak duality for (5) implies that (12) can hold only if $q_k(d_{k,j_q}, z_{k,j_q}) < 0$. Hence, one does not need to check if $q_k(d_{k,j_q}, z_{k,j_q}) < 0$ holds before employing (12) as a stopping condition for the QP solver. By contrast, the next lemma shows that (13) should be used as a stopping condition for the QP solver only if $q_k(d_{k,j_q}, z_{k,j_q}) < 0$. Algorithm 1 ensures this by setting $\lambda_{k,j_q} \leftarrow \infty$ when $q_k(d_{k,j_q}, z_{k,j_q}) \geq 0$, and otherwise the lemma shows that $\lambda_{k,j_q} \in (0, 1)$.

Lemma 2.4. *Suppose that, in iteration $k \in \mathbb{N}$ of Algorithm 1, one has $\theta_k(y_{k,*}) < 0$. In addition, suppose that, during iteration $j \in \mathbb{N}$ of the inner loop of outer iteration $k \in \mathbb{N}$, one finds that $q_k(d_{k,j_q}, z_{k,j_q}) < 0$ and* (13) *holds. Then,* (22) *holds.*

Proof. By $q_k(d_{k,j_q}, z_{k,j_q}) < 0$ and weak duality for (5), one finds in (9) that

$$\frac{\theta_k(y_{k,0})}{q_k(d_{k,j_q}, z_{k,j_q})} \geq 1. \tag{23}$$

If $\theta_k(y_{k,0}) = q_k(d_{k,j_q}, z_{k,j_q})$, then $(d_{k,j_q}, z_{k,j_q}, y_{k,0})$ is a primal-dual solution of (5) and $\theta_k(y_{k,0}) = \theta_k(y_{k,j_\theta}) = \theta_k(y_{k,*})$, which means that (22) holds. Hence, we may proceed under the assumption that $\theta_k(y_{k,0}) < q_k(d_{k,j_q}, z_{k,j_q}) < 0$, which implies that (23) holds strictly. Observing the formula in (9), one finds that $\lambda_{k,j_q} \in (0, 1)$. This fact, (13), and weak duality for (5) imply

$$\begin{aligned} \theta_k(y_{k,j_\theta}) - \theta_k(y_{k,0}) &\geq \lambda_{k,j_q}(q_k(d_{k,j_q}, z_{k,j_q}) - \theta(y_{k,0})) \\ &\geq \lambda_{k,j_q}(\theta_k(y_{k,*}) - \theta(y_{k,0})) \geq 0, \end{aligned}$$

which along with $\lambda_{k,j_q} \in (0, 1)$ and the facts that $\theta_k(y_{k,*}) < 0$ and

$$\frac{\theta_k(y_{k,0})}{\theta_k(y_{k,*})} \leq \frac{\theta_k(y_{k,0})}{q_k(d_{k,j_q}, z_{k,j_q})} \implies \theta_k(y_{k,0}) \geq \frac{\theta_k(y_{k,0})}{q_k(d_{k,j_q}, z_{k,j_q})}\theta_k(y_{k,*})$$

implies that

$$\begin{aligned} \theta_k(y_{k,j_\theta}) &\geq \lambda_{k,j_q}\theta_k(y_{k,*}) + (1 - \lambda_{k,j_q})\theta_k(y_{k,0}) \\ &\geq \left(\lambda_{k,j_q} + (1 - \lambda_{k,j_q})\frac{\theta_k(y_{k,0})}{q_k(d_{k,j_q}, z_{k,j_q})}\right)\theta_k(y_{k,*}). \end{aligned} \tag{24}$$

In addition, one finds that λ_{k,j_q} in (9) satisfies

$$\lambda_{k,j_q} \geq 1 - \frac{\sigma_k^2 + 2\sigma_k}{\frac{\theta_k(y_{k,0})}{q_k(d_{k,j_q}, z_{k,j_q})} - 1} = \frac{\frac{\theta_k(y_{k,0})}{q_k(d_{k,j_q}, z_{k,j_q})} - (1 + \sigma_k)^2}{\frac{\theta_k(y_{k,0})}{q_k(d_{k,j_q}, z_{k,j_q})} - 1},$$

implying that

$$\lambda_{k,j_q} + (1 - \lambda_{k,j_q})\frac{\theta_k(y_{k,0})}{q_k(d_{k,j_q}, z_{k,j_q})} \leq (1 + \sigma_k)^2,$$

which along with (24) and the fact that $\theta_k(y_{k,*}) < 0$ shows that

$$\theta_k(y_{k,j_\theta}) \geq \left(\lambda_{k,j_q} + (1 - \lambda_{k,j_q})\frac{\theta_k(y_{k,0})}{q_k(d_{k,j_q}, z_{k,j_q})}\right)\theta_k(y_{k,*}) \geq (1 + \sigma_k)^2\theta(y_{k,*}),$$

as desired. □ □

2.3 Convergence Analysis

In this section, we show under Assumptions 1.1 and 2.1 that Algorithm 1 either terminates finitely with a stationary point for f or, with probability one, generates a sequence of iterates that converge to stationarity for f. Throughout this section, let $\mathcal{K}$ be the indices of the outer iterations performed by the algorithm before termination (if the algorithm ever terminates) or the failure of a subroutine (if a subroutine ever fails). The subroutines that may fail are the iteration perturbation procedure (Algorithm 4) and the sample set update (Algorithm 6), wherein failure means that a loop does not terminate. If such an event occurs in iteration k, then $\mathcal{K} = \{1, \ldots, k\}$. If the algorithm never terminates and no subroutine ever fails, then one simply has that the iterations performed are $\mathcal{K} = \mathbb{N}$.

We begin by showing that the algorithm is well posed along with important properties of the subroutines stated in Appendix A.

Lemma 2.5. *Algorithm 1 is well posed in the sense that it either terminates finitely or, with probability one, it performs an infinite number of outer iterations. In any case, for any $k \in \mathcal{K}$, the following hold true.*

(a) $H_k \succ 0$ and $W_k = H_k^{-1} \succ 0$.

(b) The inner loop terminates finitely with (d_k, y_k) satisfying $\|d_k\|_{H_k} = \|G_k y_k\|_{W_k}$.

(c) In Line 21, Algorithm 3 terminates finitely with $\alpha_k \geq 0$. If $p_k < p$, then $\alpha_k = 0$ or $\alpha_k \in [\underline{\alpha}, \overline{\alpha}]$. Otherwise, if $p_k = p$, then $\alpha_k \in (0, \overline{\alpha}]$. In any case, if $\alpha_k > 0$, then (39) holds, or at least (39a) holds.

(d) In Line 29, Algorithm 4 yields, with probability one, $x_{k+1} \in \mathcal{D}$ satisfying (40) or at least satisfying (40a) and (40c).

(e) If Line 31 is reached and (42) holds, then Algorithm 6 yields $\mathcal{X}_{k+1} \leftarrow \{x_{k+1}\}$ and $p_{k+1} \leftarrow 0$; otherwise, Algorithm 6 yields, with probability one,

$$\mathcal{X}_{k+1} \leftarrow (\{x_{k+1}\} \cup \mathcal{S}_{k+1} \cup (\mathcal{X}_k \cap \mathbb{B}(x_{k+1}, \epsilon_{k+1}))) \subset \mathbb{B}(x_{k+1}, \epsilon_{k+1})$$

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

Finally, let $\mathcal{K}_{H,W} := \{k \in \mathcal{K} : \alpha_k d_k =: s_k \neq 0\}$, which are the indices of iterations in which Algorithm 5 may yield $(H_{k+1}, W_{k+1}) \neq (H_k, W_k)$. If $\mathcal{K}_{H,W}$ is infinite, then for any $\chi \in (0, 1)$ there exist constants $(\underline{\mu}, \overline{\mu}) \in (0, \infty)^2$ such that, for every $K \in \mathbb{N}$, the following hold for at least $\lceil \chi K \rceil$ values of $k \in \mathcal{K}_{H,W}$:

$$\underline{\mu} \|G_k y_k\|_2^2 \leq \|G_k y_k\|_{W_k}^2 \tag{25a}$$

$$\text{and} \quad \|W_k G_k y_k\|_2^2 \leq \overline{\mu} \|G_k y_k\|_W^2. \tag{25b}$$

If $\mathcal{K}_{H,W}$ is finite, then such constants exist satisfying (25) for all $k \in \mathcal{K}$.

Proof. If the algorithm reaches iteration $k \in \mathcal{K}$ in which the condition in Line 3 holds, then the algorithm terminates finitely. In this case, all subroutines in iterations $\{0, 1, \ldots, k-1\}$ must have terminated successfully prior to termination. Moreover, in this case, (25) follows from the fact that only a finite number of iterations are performed and the following proof of part (a) of the lemma:

(a) The facts that $H_0 \succ 0$ and $W_0 \succ 0$ follow from the initialization of the algorithm. Now suppose that iteration 1 is reached. If $s_0 = 0$, then $H_1 \leftarrow H_0 \succ 0$ and $W_1 \leftarrow W_0 \succ 0$; otherwise, positive definiteness of H_1 and W_1 follows the fact that (41) implies $s_0^T v_0 > 0$ and from well-known properties of BFGS updating; see, e.g., [33, Chapter 6]. Inductively, positive definiteness of H_k and W_k for any $k \in \mathbb{N}$ follows by the same arguments.

This completes the proof of the lemma for the case when the algorithm reaches $k \in \mathcal{K}$ at which the condition in Line 3 holds. Hence, we may proceed under the assumption that this condition does not hold for any $k \in \mathcal{K}$.

Suppose that the algorithm reaches iteration $k \in \mathcal{K}$. To prove that, with probability one, it reaches iteration $k + 1$ (i.e., without failure of a subroutine), it suffices to prove parts (b)–(e) (since (a) has been proved above).

(b) By part (a), one has $H_k \succ 0$ and $W_k \succ 0$, from which it follows that strong duality holds at the primal-dual optimal solution of (5). Since $\theta_k(y) \leq 0$ for all $y \in \mathbb{N}$, there are two cases to consider, namely, whether $\theta_k(y_{k,*}) = 0$ or $\theta_k(y_{k,*}) < 0$. First, suppose that $\theta_k(y_{k,*}) = 0$. Since $W_k \succ 0$, this implies that $G_k y_{k,*} = 0$. Under Assumption 2.1, we have that $y_{k,j} \to y_{k,*}$. This limit, the fact that $G_k y_{k,*} = 0$, and the facts that $W_k \succ 0$ and $\epsilon_k > 0$ together imply that (10) holds for some sufficiently large $j \in \mathbb{N}$. Now suppose that $\theta_k(y_{k,*}) < 0$. If (10) holds for any $j \in \mathbb{N}$, then the inner loop terminates and there is nothing left to prove; hence, we may proceed assuming that (10) does not hold for any $j \in \mathbb{N}$. Under Assumption 2.1, we have that $(d_{k,j}, y_{k,j}) \to (d_{k,*}, y_{k,*})$. This limit, continuity of q_k and θ_k, the fact that $\theta_k(y_{k,*}) < 0$, strong duality for (5), Lemma 2.1, and the fact that $\tau_k \in (0,1)$ imply that (11) and (12) will be satisfied for some sufficiently large $j \in \mathbb{N}$. Finally, the fact that $H_k = W_k^{-1}$ and at termination of the inner loop the algorithm yields $d_k = -W_k G_k y_k$ implies that $\|d_k\|_{H_k} = \|G_k y_k\|_{W_k}$, as desired.

(c–d) The proof follows in the same manner as that for [10, Lemma 2.3].

(e) The proof follows in the same manner as that for [10, Lemma 2.5].

Since we have shown that if the algorithm reaches iteration $k \in \mathcal{K}$, then it reaches iteration $k+1$ with probability one, it follows that, again with probability one, an infinite number of iterations are performed (without failure of a subroutine). Finally, with respect to the stated property of the sequence $\{(H_k, W_k)\}_{k \in \mathcal{K}_{H,W}}$, the proof follows in the same manner as that for [11, Corollary 3.2]. □ □

The next three lemmas are similar to results previously proved for GS methods. First, the following lemma is a simple consequence of the previous lemma (specifically, parts (c) and (e)) and the sample set update strategy, namely, Algorithm 6. A similar result was proved as [10, Lemma 3.3].

Lemma 2.6. *If $\mathcal{K} = \mathbb{N}$, then $\mathcal{K}_\alpha := \{k \in \mathbb{N} : \alpha_k > 0\}$ is infinite.*

Proof. Suppose $\mathcal{K} = \mathbb{N}$ and observe by Lemma 2.5(c) that $\alpha_k \geq 0$ for all $k \in \mathbb{N}$. In order to derive a contradiction, suppose that there exists $k_\alpha \in \mathbb{N}$ such that $\alpha_k = 0$ for all $k \in \mathbb{N}$ with $k \geq k_\alpha$. By Lemma 2.5(c), this means that $p_k \leq p - 1$ for all $k \geq k_\alpha$. However, with $\alpha_k = 0$, one finds that (42b) does not hold, which by Lemma 2.5(e) implies that $p_{k+1} \geq \min\{p_k + 1, p\}$. This implies the existence of some $k \geq k_\alpha$ such that $p_k \geq p$, which by Lemma 2.5(c) implies that $\alpha_k > 0$, a contradiction of the definition of k_α. □ □

The next lemma shows a useful upper bound on the objective function value at iteration $k+1 \in \mathcal{K}$; for a similar result, see, e.g., [10, Lemma 3.4].

Lemma 2.7. *If $k+1 \in \mathcal{K}$, then*

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

Proof. Suppose $k+1 \in \mathcal{K}$, which implies that $k \in \mathcal{K}$. Lemma 2.5(a) and (b) imply that $d_k = 0$ if and only if $G_k y_k = 0$. If $d_k = 0$ and $G_k y_k = 0$, then $x_{k+1} = x_k$ and the result follows trivially. Otherwise, in iteration $k \in \mathcal{K}$, Lemma 2.5(d) shows that x_{k+1} satisfies (40a) and (40c). The triangle inequality and (40c) imply

$$\begin{aligned}\|x_{k+1} - x_k\|_2 &\leq \min\{\alpha_k, \epsilon_k\} \min\{\|d_k\|_2, \|G_k y_k\|_2\} + \alpha_k \|d_k\|_2 \\ &\leq \alpha_k \|d_k\|_2 \min\{2, 1 + \|G_k y_k\|_2 / \|d_k\|_2\}.\end{aligned}$$

Hence, along with (40a), one finds that

$$\begin{aligned}f(x_{k+1}) - f(x_k) &\leq -\underline{\eta}\alpha_k \max\{\|d_k\|_2^2, \|G_k y_k\|_2^2\} \\ &= -\underline{\eta}\alpha_k \|d_k\|_2 \max\{\|d_k\|_2, \|G_k y_k\|_2^2 / \|d_k\|_2\} \\ &\leq -\underline{\eta}\|x_{k+1} - x_k\|_2 \left(\frac{\max\{\|d_k\|_2, \|G_k y_k\|_2^2 / \|d_k\|_2\}}{\min\{2, 1 + \|G_k y_k\|_2 / \|d_k\|_2\}} \right) \\ &\leq -\tfrac{1}{2}\underline{\eta}\|x_{k+1} - x_k\|_2 \max\{\|d_k\|_2, \|G_k y_k\|_2\},\end{aligned}$$

as desired. □ □

Now we enter the core theory of gradient sampling methods. At the heart of this theory is the closure of the convex hull of gradients at points of differentiability in an ϵ_k-neighborhood about a given point $\bar{x} \in \mathbb{R}^n$, namely,

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

as well as, for a given $k \in \mathcal{K}$, point $\bar{x} \in \mathbb{R}^n$, and threshold $\omega \in \mathbb{R}_{>0}$, the following subset of the Cartesian product of ϵ_k-balls about x_k defined as

$$\mathcal{T}_k(\bar{x}, \omega) := \left\{ \mathcal{X}_k \in \prod_0^{p_k} (\mathbb{B}(x_k, \epsilon_k) \cap \mathcal{D}) : \right.$$
$$\left. \|P_{W_k}(\operatorname{conv}(\{\nabla f(x)\}_{x \in \mathcal{X}_k}))\|_{W_k} \leq \|P_{W_k}(\mathcal{G}_k(\bar{x}))\|_{W_k} + \omega \right\}.$$

(Recall that $P_{W_k}(\cdot)$ has been defined in (14).) The following lemma, which follows [23, Lemma 3.2(i)], [9, Lemma 4.7], and [10, Lemma 3.6], shows that if the sample set size indictor p_k is sufficiently large and x_k is sufficiently close to $\bar{x}$, then for any $\omega \in \mathbb{R}_{>0}$ there exists a nonempty open subset of $\mathcal{T}_k(\bar{x}, \omega)$. This will be critical in our main result, where we need to show in certain situations that an element of this subset can be found through randomly sampling.

Lemma 2.8. *Let $\bar{x} \in \mathbb{R}^n$ and $\omega \in \mathbb{R}_{>0}$ be given. If $k \in \mathcal{K}$ and $p_k \geq n+1$, then there exists $\zeta > 0$ such that with $x_k \in \mathbb{B}(\bar{x}, \zeta)$ there is a nonempty open $\mathcal{T} \subseteq \mathcal{T}_k(\bar{x}, \omega)$.*

Proof. Using the metric defined by W_k, the proof follows the same argument of [23, Lemma 3.2(i)], which makes use of Carathéodory's theorem. □ □

We now present a convergence theorem for Algorithm 1. Much of the proof follows similar arguments as that for [10, Theorem 3.1], which we present for completeness. The new features are two-fold: (1) Our algorithm is even less conservative about the Hessian and inverse Hessian updates than the method in [10], so our convergence result relies on arguments about self-correcting properties of BFGS updating that we have stated in Lemma 2.5, which borrows from [11]; and (2) our inexactness conditions and our Lemma 2.2, which have not appeared before for GS methods, play critical roles in the proof of the theorem.

Theorem 2.1. *Suppose $\psi \in (0, 1)$. Algorithm 1 either terminates finitely with a stationary point for f or, with probability one, it performs an infinite number of outer iterations. In the latter case, with probability one, the sampling radius sequence satisfies $\{\epsilon_k\} \searrow 0$ and every cluster point of the iterate sequence $\{x_k\}$ is stationary for f.*

Proof. If Algorithm 1 terminates finitely with a stationary point for f, then there is nothing left to prove. Otherwise, by Lemma 2.5, it follows with probability one that an infinite number of outer iterations are performed, meaning $\mathcal{K} = \mathbb{N}$.

Going forward, we may assume that $\mathcal{K} = \mathbb{N}$. Under this assumption, our next aim is to prove that $\{\epsilon_k\} \searrow 0$ with probability one.

- **Case 1:** Suppose that $\mathcal{K}_d := \{k \in \mathbb{N} : d_k = 0\}$ is infinite. By Lemma 2.5(a) and (b), it follows that $G_k y_k = 0$ for all $k \in \mathcal{K}_d$. This fact, the fact that $|\mathcal{K}_d| = \infty$, and (10) imply that $\{\epsilon_k\} \searrow 0$, as desired.
- **Case 2:** Suppose that $\mathcal{K}_d := \{k \in \mathbb{N} : d_k = 0\}$ is finite. Let us proceed by supposing that there exists $k_\epsilon \in \mathbb{N}$ and a sampling radius $\epsilon \in (0, \infty)$ such that $\epsilon_k = \epsilon$ for all $k \in \mathbb{N}$ with $k \geq k_\epsilon$. Our aim is to show that the existence of such a pair (k_ϵ, ϵ) occurs with probability zero. From (10), it follows that

$$\max\{\|d_k\|_2, \|G_k y_k\|_2\} > \nu\epsilon \quad \text{for all} \quad k \geq k_\epsilon. \tag{26}$$

On the other hand, Assumption 1.1, Lemma 2.7, and (40a) imply that

$$\sum_{k=k_\epsilon}^{\infty} \|x_{k+1} - x_k\|_2 \max\{\|d_k\|_2, \|G_k y_k\|_2\} < \infty \tag{27a}$$

$$\sum_{k=k_\epsilon}^{\infty} \alpha_k \max\{\|d_k\|_2^2, \|G_k y_k\|_2^2\} < \infty. \tag{27b}$$

In conjunction with (26), the bound in (27a) implies that the iterate sequence $\{x_k\}$ is a Cauchy sequence, meaning $\{x_k\} \to \bar{x}$ for some $\bar{x} \in \mathbb{R}^n$. At the same time, with (26), the bound in (27b) implies that $\{\alpha_k\} \to 0$. We claim that this implies that $p_k = p$ for all sufficiently large $k \in \mathcal{K}_\alpha$, where $\mathcal{K}_\alpha$ is defined as in Lemma 2.6. Indeed, since $\{\alpha_k\} \to 0$, it follows by Lemma 2.5(c) that for sufficiently large $k \in \mathbb{N}$ either (i) $p_k < p$ and $\alpha_k = 0$ or (ii) $p_k = p$ and $\alpha_k > 0$. Combined with the fact that $|\mathcal{K}_d| < \infty$, it follows along with Lemma 2.6 that there exists an infinite number of iterations indexed by $k \geq k_\epsilon$ such that $\alpha_k d_k \neq 0$ and $p_k = p$, whereas all other iterations for sufficiently large $k \geq k_\epsilon$ yield $\alpha_k = 0$. Going forward, for ease of notation in the remainder of the proof of this case, since $x_{k+1} \leftarrow x_k$ and $(H_{k+1}, W_{k+1}) \leftarrow (H_k, W_k)$ whenever $\alpha_k = 0$, let us proceed without loss of generality under the assumption that $k_\epsilon = 0$ and $\epsilon = \epsilon_0$, and that $\alpha_k > 0$, $d_k \neq 0$, and $p_k = p$ for all $k \in \mathbb{N}$. Notice that under these conditions the set $\mathcal{K}_{H,W}$ defined in Lemma 2.5 equals $\mathbb{N}$. Correspondingly, for a given $\chi \in (0, 1)$, let $\mathcal{K}_\chi$ be the indices of iterations for which (25) holds; in particular, for $k \in \mathcal{K}_\chi$, one has from (25a)–(25b) that

$$\max\{\|d_k\|_2^2, \|G_k y_k\|_2^2\} \leq \mu \|G_k y_k\|_{W_k}^2, \quad \text{where} \quad \mu := \max\left\{\frac{\overline{\mu}}{\underline{\mu}}, \frac{1}{\underline{\mu}}\right\}. \tag{28}$$

Notice that since (10) does not hold for all $k \geq k_\epsilon$, it follows that either (12) or (13) holds for all $k \geq k_\epsilon$. Hence, by Lemmas 2.3 and 2.4, it follows that (22) holds for all $k \geq k_\epsilon$, meaning for all $k \geq k_\epsilon$ that

$$\|G_k y_k\|_{W_k} \leq (1 + \sigma_k) \|P_{W_k}(\operatorname{conv}(\{\nabla f(x)\}_{x \in \mathcal{X}_k}))\|_{W_k}. \tag{29}$$

– **Subcase 2a:** If $\bar{x}$ is ϵ-stationary, then $\|P_{W_k}(\mathcal{G}_k(\bar{x}))\|_{W_k} = 0$ for any $W_k \succ 0$. Therefore, with $\mu \in (0, \infty)$ defined in (28), $\omega = \nu\epsilon/(\sqrt{\mu}(1 + \sigma))$, and $(\zeta, \mathcal{T})$ chosen as in Lemma 2.8, it follows that there exists $k_\zeta \in \mathbb{N}$ with $k_\zeta \geq k_\epsilon$ such that $x_k \in \mathbb{B}(\bar{x}, \zeta)$ for all $k \geq k_\zeta$ and, with (29),

$$\begin{aligned} \max\{\|d_k\|_2, \|G_k y_k\|_2\} &\leq \sqrt{\mu} \|G_k y_k\|_{W_k} \\ &\leq \sqrt{\mu}(1 + \sigma_k) \|P_{W_k}(\operatorname{conv}(\{\nabla f(x)\}_{x \in \mathcal{X}_k}))\|_{W_k} \\ &\leq \sqrt{\mu}(1 + \sigma_k)\omega \leq \nu\epsilon \end{aligned} \tag{30}$$

whenever $k \geq k_\zeta$, $k \in \mathcal{K}_\chi$, and $\mathcal{X}_k \in \mathcal{T}$. Combining (26) and (30), it follows that $\mathcal{X}_k \notin \mathcal{T}$ for all $k \geq k_\zeta$ with $k \in \mathcal{K}_\chi$. However, this is a probability zero event since for all such k the set $\mathcal{X}_k$ will contain new points from $\mathbb{B}(x_k, \epsilon_k)$ that are generated independently whether or not $k \in \mathcal{K}_\chi$, meaning that with probability one there exists sufficiently large such k with $k \in \mathcal{K}_\chi$ and $\mathcal{X}_k \in \mathcal{T}$, which would yield (30).

– **Subcase 2b:** If $\bar{x}$ is not ϵ-stationary, then it follows from Lemma 2.5(c) that α_k satisfies (39a) for all $k \in \mathbb{N}$. In particular, (39a) holds either with $\alpha_k \geq \gamma\overline{\alpha}$ or with $\alpha_k < \gamma\overline{\alpha}$ such that

$$f(x_k + \gamma^{-1}\alpha_k d_k) - f(x_k) \geq -\underline{\eta}\gamma^{-1}\alpha_k \max\{\|d_k\|_2^2, \|G_k y_k\|_2^2\}. \tag{31}$$

In the latter case, Lebourg's mean value theorem [6, Theorem 2.3.7] implies the existence of a point $\tilde{x}_k \in [x_k, x_k + \gamma^{-1}\alpha_k d_k]$ and a corresponding subgradient of f at $\tilde{x}_k$, call it $\tilde{g}_k \in \partial f(\tilde{x}_k)$, such that

$$f(x_k + \gamma^{-1}\alpha_k d_k) - f(x_k) = \gamma^{-1}\alpha_k \tilde{g}_k^T d_k. \tag{32}$$

Combining (31), (32), and the fact that $d_k = -W_k G_k y_k$, one finds that

$$\tilde{g}_k^T W_k G_k y_k \leq \underline{\eta} \max\{\|d_k\|_2^2, \|G_k y_k\|_2^2\}. \tag{33}$$

On the other hand, for any $\omega \in (0, \infty)$ and $(\zeta, \mathcal{T})$ as in Lemma 2.8, there exists $k_\omega \geq k_\epsilon$ such that $x_k \in \mathbb{B}(\bar{x}, \min\{\zeta, \epsilon/3\})$ for $k \geq k_\omega$ and, with (29),

$$\begin{aligned} \|G_k y_k\|_{W_k} &\leq (1+\sigma_k)\|P_{W_k}(\operatorname{conv}(\{\nabla f(x)\}_{x \in \mathcal{X}_k}))\|_{W_k} \\ &\leq (1+\sigma_k)\|P_{W_k}(\mathcal{G}_k(\bar{x}))\|_{W_k} + (1+\sigma_k)\omega \end{aligned} \tag{34}$$

whenever $k \geq k_\omega$, $k \in \mathcal{K}_\chi$, and $\mathcal{X}_k \in \mathcal{T}$. Hence, for such k, it follows by Lemma 2.2 with $\mathcal{S} = \mathcal{G}_k(\bar{x})$, $\beta \in (0, \min\{1, \underline{\eta}\mu\})$, and σ_k such that

$$\beta(1 + \sqrt{\sigma_k^2 + 2\sigma_k})^2 + \Omega_{\mathcal{G}_k(\bar{x}), W_k} \sqrt{\sigma_k^2 + 2\sigma_k} \in (0, 1), \tag{35}$$

that there exists ω such that (34) implies

$$\begin{aligned} v^T W_k G_k y_k &> \underline{\eta}\mu \|G_k y_k\|_{W_k}^2 \\ &\geq \underline{\eta} \max\{\|d_k\|_2^2, \|G_k y_k\|_2^2\} \quad \text{for all} \quad g \in \mathcal{G}_k(\bar{x}). \end{aligned} \tag{36}$$

Notice that there exists $k_\sigma \geq k_\omega$ such that σ_k satisfies (35) for all $k \geq k_\sigma$ with $k \in \mathcal{K}_\chi$ since the fact that $\{\alpha_k\} \to 0$ and the construction of the algorithm implies that $\{\sigma_k\} \to 0$. Together, (33) and (36) imply that $\tilde{g}_k \notin \mathcal{G}_k(\bar{x})$ whenever $k \geq k_\sigma$, $k \in \mathcal{K}_\chi$, and $\mathcal{X}_k \in \mathcal{T}$. However, by the facts that $\mathbb{1}^T y_k = 1$ and $y_k \geq 0$, Assumption 1.1, and [6, Proposition 2.1.2], it follows for all $k \geq k_\sigma$ with $k \in \mathcal{K}_\chi$ that

$$\|d_k\|_2 = \|W_k G_k y_k\|_2 \leq \sqrt{\overline{\mu}} \|G_k y_k\|_2 \leq \sqrt{\overline{\mu}} L_{\mathbb{B}(\bar{x}, \epsilon)},$$

where $L_{\mathbb{B}(\bar{x}, \epsilon)} \in (0, \infty)$ is a Lipschitz constant for f over $\mathbb{B}(\bar{x}, \epsilon)$. This shows that $\{\|d_k\|_2\}_{k \in \mathcal{K}_\chi}$ is bounded. This fact, along with $\{\alpha_k\} \to 0$, implies that $\alpha_k \leq \gamma\epsilon/(3\|d_k\|_2)$ for all sufficiently large $k \in \mathcal{K}_\chi$, i.e., $\gamma^{-1}\alpha_k \|d_k\|_2 \leq \epsilon/3$ for all sufficiently large $k \in \mathcal{K}_\chi$. Along with the fact that $x_k \in \mathbb{B}(\bar{x}, \min\{\zeta, \epsilon/3\})$ implies $\|x_k - \bar{x}\|_2 \leq \epsilon/3$, it follows that $\tilde{x}_k \in \mathbb{B}(\bar{x}, 2\min\{\zeta, \epsilon/3\}/3)$ and hence $\tilde{g}_k \in \mathcal{G}_k(\bar{x})$ for all sufficiently large $k \in \mathbb{N}$. Overall, since $\tilde{g}_k \notin \mathcal{G}_k(\bar{x})$ whenever $k \geq k_\sigma$, $k \in \mathcal{K}_\chi$, and $\mathcal{X}_k \in \mathcal{T}$, yet $\tilde{g}_k \in \mathcal{G}_k(\bar{x})$ for all sufficiently large k, it follows that $\mathcal{X}_k \notin \mathcal{T}$ for all sufficiently large $k \in \mathcal{K}_\chi$. However, this is a probability zero event since $|\mathcal{K}_\chi| = \infty$ and the sample points are generated independently of whether $k \in \mathcal{K}_\chi$.

We have shown that $\{\epsilon_k\} \searrow 0$ with probability one. If $\{\epsilon_k\} \searrow 0$, then by (10) there exists an infinite index set $\mathcal{K}_\epsilon := \{k \in \mathbb{N} : \epsilon_{k+1} \leftarrow \psi\epsilon_k\}$ where

$$\max\{\|d_k\|_2, \|G_k y_k\|_2\} \leq \epsilon_k \quad \text{for all} \quad k \in \mathcal{K}_\epsilon.$$

The same argument as in [9, Theorem 4.2, Case 2], which borrows from [23, Theorem 3.3, part (iii)], shows all cluster points of $\{x_k\}$ are stationary for f. □ □

Our second convergence result, presented as the following corollary, considers the case when one chooses $\psi = 1$ so that the sampling radius remains that $\epsilon_0 \in \mathbb{R}_{>0}$ for all $k \in \mathcal{K}$. Similar results have appeared in the literature to prove a similar property of other GS methods; see, e.g., [23, Theorem 3.5].

Corollary 2.1. *Suppose $\psi = 1$. Algorithm 1 either terminates finitely with a stationary point for f or, with probability one, it performs an infinite number of outer iterations. In the latter case, with probability one, it either reaches iteration $k \in \mathbb{N}$ such that $0 \in \mathcal{G}_k(x_k)$ or every cluster point of the iterate sequence $\{x_k\}$ is ϵ_0-stationary for f.*

Proof. As in the proof of Theorem 2.1, if Algorithm 1 terminates finitely with a stationary point for f, then there is nothing left to prove. Otherwise, by Lemma 2.5, it follows with probability one that an infinite number of outer iterations are performed, meaning $\mathcal{K} = \mathbb{N}$. If the algorithm reaches iteration $k \in \mathbb{N}$ in which $0 \in \mathcal{G}_k(x_k)$, then there is nothing left to prove. Otherwise, following the arguments in the proof of Theorem 2.1, it follows that $\inf\{\|G_k y_k\|_2 : k \in \mathbb{N}\} > 0$ is a probability zero event. In the probability one event that $\inf\{\|G_k y_k\|_2 : k \in \mathbb{N}\} = 0$, the conclusion follows from the fact that $\partial_{\epsilon_0} f$ is closed. □ □

3 GS Algorithm with Gradient Aggregation

Our second algorithm adds a conceptually straightforward, but practically significant enhancement to Algorithm 1. In particular, we add a procedure for exploiting gradient aggregation that can significantly reduce the size of the subproblems to be solved approximately in each "outer" iteration of the algorithm. We remark that this enhancement to the GS methodology is only possible when one is able to employ inexact subproblem solutions. This is the case since the exact solution of a subproblem involving a "gradient aggregation vector" does not offer the exact solution of a subproblem involving individual gradients and no aggregation.

In this section, we present a statement of the proposed algorithm, then show that it offers the same convergence guarantees as does Algorithm 1.

3.1 Algorithm Description

Our algorithm with inexact subproblem solutions and gradient aggregation is stated as Algorithm 2. The algorithm borrows much from Algorithm 1; we have written it in such a manner that only its unique steps are stated. The main idea of the enhancement is the following. For any $k+1 \in \mathcal{K}$ such that $\alpha_k > 0$, the matrix of gradients G_{k+1} contains all points in the set $\mathcal{X}_{k+1}$, as in Algorithm 1. However, for any $k+1 \in \mathcal{K}$ such that $x_{k+1} = x_k$ since $\alpha_k = 0$, rather than solve a subproblem defined by gradients at all points in $\mathcal{X}_{k+1}$, the algorithm considers a subproblem in which the gradients defining the matrix G_k (which compose a submatrix of G_{k+1}) have been *aggregated* into a single "gradient aggregation vector" $G_k y_k$. The following lemma shows that a feasible point for the subproblem *that the algorithm considers* in iteration $k+1$ corresponds to a feasible point for the subproblem *that would be defined by all gradients in* G_{k+1}^{full}.

Lemma 3.1. *Consider $k \in \mathcal{K}$ such that $k \geq 1$ and $\alpha_{k-1} = 0$, meaning $G_k = G_k^{\text{agg}}$. For any $j \in \mathbb{N}$ such that $y_{k,j}$ is computed, this vector, which is feasible for the dual problem in (5), corresponds uniquely to a feasible point for the dual problem in (5) if G_k^{full} were used in place of $G_k = G_k^{\text{agg}}$.*

Proof. Consider any $j \in \mathbb{N}$ such that $y_{k,j}$ is computed. Let $[y_{k,j}]_1$ and $[y_{k,j}]_2$ denote the first and second elements of $y_{k,j}$, respectively, with the subvector of all remaining elements of $y_{k,j}$ being denoted as $[y_{k,j}]_{>2}$. One finds that

$$\begin{aligned}
&G_k^{\text{agg}} y_{k,j} \\
&= \nabla f(x_k)[y_{k,j}]_1 + (G_{k-1} y_{k-1})[y_{k,j}]_2 + [\nabla f(x)]_{x \in \mathcal{X}_k \setminus (x_k \cup \mathcal{X}_{k-1})}[y_{k,j}]_{>2} \\
&= \begin{bmatrix} \nabla f(x_k) & G_{k-1} & [\nabla f(x)]_{x \in \mathcal{X}_k \setminus (x_k \cup \mathcal{X}_{k-1})} \end{bmatrix} \begin{bmatrix} [y_{k,j}]_1 \\ y_{k-1}[y_{k,j}]_2 \\ [y_{k,j}]_{>2} \end{bmatrix} = G_k^{\text{full}} \begin{bmatrix} [y_{k,j}]_1 \\ y_{k-1}[y_{k,j}]_2 \\ [y_{k,j}]_{>2} \end{bmatrix},
\end{aligned}$$

where—since $\mathbb{1}^T y_{k-1} = 1$, $\mathbb{1}^T y_{k,j} = 1$, $y_{k-1} \geq 0$, and $y_{k,j} \geq 0$—it follows that

$$\mathbb{1}^T \begin{bmatrix} [y_{k,j}]_1 \\ y_{k-1}[y_{k,j}]_2 \\ [y_{k,j}]_{>2} \end{bmatrix} = 1 \quad \text{and} \quad \begin{bmatrix} [y_{k,j}]_1 \\ y_{k-1}[y_{k,j}]_2 \\ [y_{k,j}]_{>2} \end{bmatrix} \geq 0,$$

which proves the desired result. □ □

Theorem 3.1. *Suppose $\psi \in (0,1)$. Algorithm 2 either terminates finitely with a stationary point for f or, with probability one, it performs an infinite number of outer iterations. In the latter case, with probability one, the sampling radius sequence satisfies $\{\epsilon_k\} \searrow 0$ and every cluster point of the iterate sequence $\{x_k\}$ is stationary for f.*

Proof. For all $k \in \mathbb{N}$, the result of Lemma 2.1 holds regardless of whether $G_k = G_k^{\text{agg}}$ or $G_k = G_k^{\text{full}}$ due to the fact that G_k has $\nabla f(x_k)$ as its first column in either case. The results of Lemmas 2.3 and 2.4 also continue to hold regardless of whether $G_k = G_k^{\text{agg}}$ or $G_k = G_k^{\text{full}}$, implying that the inner loop terminates finitely for all $k \in \mathcal{K}$. Now consider the pair $(d_k, y_k) = (-W_k G_k y_k, y_k)$ upon termination of the inner loop in iteration

$k \in \mathcal{K}$. If $G_k = G_k^{\text{full}}$, then the properties of (d_k, y_k) are the same as that in Algorithm 1. Otherwise, when $G_k = G_k^{\text{agg}}$, one may consider

$$\begin{bmatrix} [y_k]_1 \\ y_{k-1}[y_k]_2 \\ [y_k]_{>2} \end{bmatrix} \tag{37}$$

as the dual vector, as shown by Lemma 3.1. The arguments of Lemmas 2.5–2.8 and Theorem 2.1 now follow in the same manner as in §2 using G_k^{full} in place of G_k and y_k or (37) in place of the dual vector for all $k \in \mathcal{K}$. Crucial in the applicability of these arguments is that, if the sample set size indicator p_k ever exceeds p, then $G_k = G_k^{\text{full}}$ and the algorithm behaves as Algorithm 1 for such $k \in \mathcal{K}$. □ □

Corollary 3.1. *Suppose $\psi = 1$. Algorithm 2 either terminates finitely with a stationary point for f or, with probability one, it performs an infinite number of outer iterations. In the latter case, with probability one, it either reaches iteration $k \in \mathbb{N}$ such that $0 \in \mathcal{G}_k(x_k)$ or every cluster point of the iterate sequence $\{x_k\}$ is ϵ_0-stationary for f.*

Proof. The proof follows from that of Theorem 3.1 in the same manner as the proof of Corollary 2.1 follows from that of Theorem 2.1. □ □

4 Numerical Experiments

In this section, we present the results of numerical experiments with implementations of our proposed algorithms. The purpose of these experiments is to show that the introduction of inexactness and aggregation can consistently and substantially reduce the CPU times required by an adaptive GS algorithm. As a sanity check, we also show results from another state-of-the-art code. This code can be faster than our code, but at the expense of occasionally yielding less accurate solutions. All experiments were run on a 2015 Macbook Pro with a 2.7 GHz Dual-Core Intel Core i5 processor, although only one core was used for the experiments.

We implemented our algorithms in the C++ software package `NonOpt` [7]. For the parameters used in the algorithms and subroutines, we employed the values stated in Table 1. These values are used consistently across all of our experiments. A few points should be mentioned. First, as is typical in implementations of GS methods, our implementations assume that $x_k + \alpha_k d_k \in \mathcal{D}$ for all $k \in \mathbb{N}$, meaning that the loop in Algorithm 4 always terminates in the first iteration; hence, the parameter $\overline{\ell}$ is not used. Second, it should be highlighted that our choice of $\sigma \leftarrow 1$ is outside of the range required by our theoretical analysis. We chose this value since it yielded better results in our experiments by allowing the algorithm to employ more inexact solutions in early iterations. That said, after a few decreases, the value obeys our theory; after all, with $\iota \leftarrow 0.9$, the sequence $\{\sigma_k\}$ achieves values in $(0, \sqrt{2} - 1)$ as needed. Third, $x_0 \in \mathbb{R}^n$ was chosen in a problem-dependent manner; see the references given below in our discussion of the test problems used. Finally, for $\overline{p}$ (see Algorithm 6), we used $\overline{p}_1 \leftarrow 5$ for our implementation of Algorithm 1, whereas we used $\overline{p}_2 \leftarrow 3$ for our implementation of Algorithm 2, since we found better performance by differing this parameter in this manner.

`NonOpt` contains a dual active-set QP solver, which we used for solving the QP subproblems arising in the implementations of our algorithms. To reduce CPU time, during the solve of a given QP, the termination conditions (10)–(13) are not checked in every iteration of the QP solver. Instead, these conditions are checked only after $(p_k + 1)/4$ QP iterations have been performed, and after this threshold is reached, the conditions are checked only once every four QP iterations.

Following [10], the outer iteration sequence terminates in our implementations, with a message of success, once the stationarity radius and a dual solution satisfies

$$\max\{\|G_k y_k\|_\infty, \epsilon_k\} \leq 10^{-4}. \tag{38}$$

We consider the performance of three implementations, to which, for convenience in our discussion, we refer as follows:

Table 1: User-specified parameters for our proposed algorithms and subroutines, as well as the values used in our implementation.

Parameter(s)	Range	Values	Description
ν	$(0,\infty)$	$\sqrt{n}$	Stationarity measure tolerance
$\underline{\alpha} \leq \overline{\alpha}$	$(0,\infty)$	$10^{-10} \leq 1$	Stepsize thresholds
ρ	$(0,1)$	0.01	Inexactness threshold bound
κ	$(0,1)$	0.01	Inexactness threshold
ψ	$(0,1)$	0.5	Sampling radius reduction factor
ι	$(0,1)$	0.9	Inexactness parameter reduction factor
$\underline{\eta} < \overline{\eta}$	$(0,1)$	$10^{-12} < 0.9$	Armijo–Wolfe line search parameters
p	$[n+1,\infty) \cap \mathbb{N}$	$\min\{5000, 10n\}$	Sample set size threshold
σ	$(0, \sqrt{2}-1)$	1^*	Inexactness threshold reset value
γ	$(0,1)$	0.5	Stepsize modification factor
$\underline{\phi} < 1 < \overline{\phi}$	$(0,\infty)$	$10^{-20} < 1 < 100$	BFGS updating thresholds
ξ	$(0,\infty)$	10^{-4}	Curvature threshold
$\overline{p}_1$	$\mathbb{N}$	5	Size of addition to sample set (Alg. 1)
$\overline{p}_2$	$\mathbb{N}$	3	Size of addition to sample set (Alg. 2)
H_0	$\succ 0$	I	Initial Hessian approximation
ϵ_0	$(0,\infty)$	$\max\{0.01, 0.1\|\nabla f(x_0)\|_\infty\}$	Initial stationarity radius

- `GS-exact`: An implementation of an adaptive GS method in which the QP subproblems are solved "exactly" in each iteration; in particular, every aspect of this implementation is the same as that of `GS-inexact` (below), except that, when tasked to solve each QP subproblem, the QP solver is run until the ℓ_∞-norm of the KKT error for the QP is reduced below 10^{-12}.
- `GS-inexact`: An implementation of Algorithm 1.
- `GS-inexact-agg`: An implementation of Algorithm 2.

For test problems, we chose a set of 20 problems, some of which are convex and some nonconvex; see Table 2. The first ten problems come from [13] and the second ten come from [28]. In these sources, each problem is provided with an initial point $x_0 \in \mathbb{R}^n$, which were the initial points we used in our experiments. All of the problems are scalable in the sense that they are defined for any value of $n \in \mathbb{N}$. In order to achieve nontrivial differences in CPU times between the three algorithms, we chose n for each problem using the following scheme: (1) starting with $n = 100$, we observed the CPU time required before `GS-exact` terminated; (2) as long as the CPU time was under 30 seconds, we increased n by 30 until the CPU time required by `GS-exact` exceeded 30 seconds. This scheme led to the problem dimensionss indicated in Table 2.

The results obtained from each of our implemented algorithms can be found in Tables 3, 4, and 5. Each of the algorithms successfully reached an iteration $k \in \mathbb{N}$ satisfying (38) for all test problems, i.e., all runs of the three algorithms were successful. In each table, we provide for each problem: `obj`, the final objective value; `its`, the number of (outer) iterations until termination; `f evs`, the number of function evaluations until termination; `g evs`, the number of gradient evaluations until termination; `qp its`, the number of QP iterations until termination; and `CPU`, the number of CPU seconds until termination. In addition, in Tables 4 and 5, we provide `CPU diff`, which indicates the percentage reduction in CPU time when `CPU` is compared to that for `GS-exact`. In these columns, a negative percentage indicates that the algorithm required less CPU time than `GS-exact`.

Comparing the results for `GS-exact` and `GS-inexact` (Tables 3 and 4), one finds that the quality of the solutions in terms of the final objective values was comparable, whereas `GS-inexact` consistently required less CPU time, and often substantially less CPU time. In a few cases, `GS-inexact` required more CPU time, which can be attributed to the fact that in those few cases inexact subproblem solutions led to an increase in

Table 2: Test set with problem dimensions.

name	n
MaxQ	700
MxHilb	940
ChainedLQ	280
ChainedCB3_1	310
ChainedCB3_2	2410
ActiveFaces	2050
BrownFunction_2	340
ChainedMifflin_2	220
ChainedCrescent_1	4420
ChainedCrescent_2	280
Test29_2	610
Test29_5	880
Test29_6	400
Test29_11	160
Test29_13	310
Test29_17	640
Test29_19	430
Test29_20	220
Test29_22	1090
Test29_24	100

Table 3: Results for GS-exact.

name	obj	its	f evs	g evs	qp its	CPU
MaxQ	3.050E-07	3717	14216	5912	6859	30.55
MxHilb	1.820E-05	526	5597	4416	2006	40.79
ChainedLQ	-3.946E+02	268	4397	6631	60789	35.91
ChainedCB3_1	6.180E+02	337	4858	6046	45035	36.59
ChainedCB3_2	4.818E+03	95	591	292	598	33.88
ActiveFaces	3.083E-02	21	669	619	6173	37.15
BrownFunction_2	3.347E-03	233	3647	4843	31333	31.82
ChainedMifflin_2	-1.550E+02	482	9991	18229	160818	62.86
ChainedCrescent_1	5.197E-03	33	252	201	128	33.33
ChainedCrescent_2	1.258E-03	397	6441	9462	77608	47.18
Test29_2	4.840E-05	966	9390	7096	18357	38.39
Test29_5	9.194E-05	508	4311	2373	3736	39.07
Test29_6	2.263E-04	706	9305	9479	40356	33.13
Test29_11	1.913E+03	347	5216	7693	66261	35.65
Test29_13	1.747E+02	338	6313	10516	66438	41.17
Test29_17	3.961E-05	408	5341	4296	14739	42.63
Test29_19	6.247E-08	644	7561	9105	45696	43.67
Test29_20	1.339E-04	1777	21947	23897	110077	40.45
Test29_22	4.539E-05	40	574	377	10453	66.28
Test29_24	5.562E-05	2708	49258	89275	349192	76.25

the required number of outer iterations—and, perhaps more importantly, required total QP iterations. But this was rare compared to the cases when GS-inexact required less CPU time. On average, GS-inexact required 22.11% less CPU time than did GS-exact.

Going further to compare GS-exact and GS-inexact-agg (Tables 3 and 5), one finds again that the quality of the solutions obtained remained comparable, although GS-inexact-agg again required less CPU time consistently and often substantially. On average, the CPU time reduction offered by GS-inexact-agg compared to GS-exact is approximately 34.90%. This shows that the CPU time reduction offered by GS-inexact-agg compared to GS-inexact is also substantial.

To help put our results into perspective with respect to the state-of-the-art, we compare the performance of our fastest algorithm, GS-inexact-agg with the state-of-the-art code LMBM [19]; see also [13, 14]. This code comes with implementations of the first ten problems from our test set; for the remaining test problems, we obtained Fortran implementations from [26]. It should be said that GS-inexact-agg and LMBM have various differences, such as the fact that LMBM is a bundle method and uses limited-memory Hessian approximations, not (dense) Hessian approximations as used in GS-inexact-agg. LMBM also uses different termination criteria.

A comparison between LMBM with its default termination criteria, LMBM with modified termination criteria, and GS-inexact-agg can be seen in Table 6. The reason that we consider modified termination criteria for LMBM is that, with its default settings, only one of the 20 test problems (namely, MaxQ) is indicated as

Table 4: Results for GS-inexact.

name	obj	its	f evs	g evs	qp its	CPU	CPU diff
MaxQ	2.870E-07	3863	14676	6083	6121	27.75	-9.18%
MxHilb	2.000E-05	464	5872	4410	1835	40.50	-0.69%
ChainedLQ	-3.946E+02	247	3855	5510	54310	34.55	-3.81%
ChainedCB3_1	6.180E+02	336	4712	5444	31627	22.33	-38.98%
ChainedCB3_2	4.818E+03	94	614	213	387	22.94	-32.28%
ActiveFaces	3.083E-02	21	669	619	23	1.27	-96.58%
BrownFunction_2	3.131E-03	256	3917	4861	32461	32.34	1.65%
ChainedMifflin_2	-1.550E+02	442	9214	16351	127904	50.16	-20.21%
ChainedCrescent_1	4.627E-03	34	256	202	123	32.01	-3.96%
ChainedCrescent_2	1.065E-03	332	5735	8969	68458	44.08	-6.58%
Test29_2	4.961E-05	942	9384	6877	16719	38.35	-0.11%
Test29_5	6.824E-04	216	1455	800	1316	13.80	-64.67%
Test29_6	2.034E-04	703	9297	9493	35815	29.99	-9.48%
Test29_11	1.913E+03	433	5686	7806	98615	46.45	30.28%
Test29_13	1.747E+02	363	7356	12937	71641	46.94	14.03%
Test29_17	4.549E-05	410	5337	4237	13976	41.63	-2.34%
Test29_19	3.507E-08	580	6804	8239	37114	35.94	-17.71%
Test29_20	1.158E-04	425	6222	9015	42825	16.48	-59.25%
Test29_22	6.458E-05	36	526	347	355	2.91	-95.61%
Test29_24	4.351E-05	2197	34581	59774	287530	55.83	-26.77%

Table 5: Results for GS-inexact-agg.

name	obj	its	f evs	g evs	qp its	CPU	CPU diff
MaxQ	2.460E-07	3539	13171	4967	5387	26.01	-14.87%
MxHilb	1.115E-04	429	4184	2696	1826	31.90	-21.80%
ChainedLQ	-3.946E+02	229	5286	6861	58896	38.35	6.79%
ChainedCB3_1	6.180E+02	285	5698	6630	22419	16.57	-54.71%
ChainedCB3_2	4.818E+03	89	561	238	483	27.96	-17.48%
ActiveFaces	3.083E-02	21	669	619	23	1.22	-96.72%
BrownFunction_2	1.843E-03	238	4533	4872	19376	17.60	-44.68%
ChainedMifflin_2	-1.550E+02	516	12762	16994	187575	69.35	10.32%
ChainedCrescent_1	2.795E-03	24	141	66	71	18.90	-43.29%
ChainedCrescent_2	9.704E-04	315	6123	6851	29742	22.27	-52.80%
Test29_2	5.104E-05	1108	11368	6307	14573	30.91	-19.49%
Test29_5	7.822E-05	414	3768	1825	3340	33.84	-13.39%
Test29_6	2.326E-04	886	12819	10279	33892	28.40	-14.27%
Test29_11	1.913E+03	324	5563	5714	30196	8.62	-75.82%
Test29_13	1.747E+02	253	6419	8598	34949	27.47	-33.27%
Test29_17	5.141E-05	425	5662	3469	9700	27.78	-34.83%
Test29_19	1.050E-01	492	7941	8170	17696	15.14	-65.33%
Test29_20	1.329E-04	532	7971	7889	54621	18.85	-53.39%
Test29_22	4.850E-05	41	631	364	10405	45.62	-31.17%
Test29_24	3.826E-05	2413	42786	54909	337121	55.02	-27.84%

"solved" at termination (using a condition based on a measure of stationarity). For the other problems, termination occurs due to a small difference in objective function values between iterations. For many of the test problems, the quality of the solution obtained by LMBM is good, and in some cases better than that obtained by GS-inexact-agg. However, for a few of the test problems (see ChainedCrescent_1, Test29_2, Test29_6, Test29_19, and Test29_20), the solution quality is worse by orders of magnitude. Hence, for another comparison, we also ran LMBM with modified termination criteria, where we effectively turned off the stopping conditions based on differences in objective values between iterations. The results of these runs are also given in Table 5 ("modified term."). The results show that LMBM remains very fast on some problems, but significantly slower on others, and there remain some problems for which the final objective function value is not as good compared to that obtained by GS-inexact-agg. (To be fair, there also remain problems for which the solution obtained by GS-inexact-agg is not as good.)

5 Conclusion

We have proposed, analyzed, and tested two algorithms for minimizing locally Lipschitz objective functions. The algorithms are based on the gradient sampling methodology. The unique feature of the first algorithm

Table 6: Comparison with LMBM.

	LMBM (default term.)		LMBM (modified term.)		GS-inexact-agg	
name	CPU	obj	CPU	obj	CPU	obj
MaxQ	1.45	6.372E-06	1.42	6.372E-06	26.01	2.460E-07
MxHilb	10.14	6.516E-03	958.23	1.388E-04	31.90	1.115E-04
ChainedLQ	0.01	-3.945E+02	0.64	-3.946E+02	38.35	-3.946E+02
ChainedCB3_1	0.03	6.181E+02	2.57	6.181E+02	16.57	6.180E+02
ChainedCB3_2	0.01	4.818E+03	0.19	4.818E+03	27.96	4.818E+03
ActiveFaces	0.16	5.082E-11	0.92	1.710E-14	1.22	3.083E-02
BrownFunction_2	0.19	4.059E-09	0.32	1.409E-11	17.60	1.843E-03
ChainedMifflin_2	0.01	-1.549E+02	1.57	-1.550E+02	69.35	-1.550E+02
ChainedCrescent_1	0.08	8.288E-01	0.21	3.115E-09	18.90	2.795E-03
ChainedCrescent_2	0.05	1.136E-04	0.13	3.884E-08	22.27	9.704E-04
Test29_2	0.01	9.713E-01	301.70	9.713E-01	30.91	5.104E-05
Test29_5	26.74	2.599E-05	252.32	3.477E-06	33.84	7.822E-05
Test29_6	0.01	2.000E+00	49.60	2.000E+00	28.40	2.326E-04
Test29_11	0.02	1.915E+03	2.47	1.915E+03	8.62	1.913E+03
Test29_13	5.45	1.747E+02	1894.70	1.747E+02	27.47	1.747E+02
Test29_17	0.20	3.574E-03	64.48	2.046E-08	27.78	5.141E-05
Test29_19	0.02	1.000E+00	5.64	1.000E+00	15.14	1.050E-01
Test29_20	0.02	5.000E-01	0.07	5.000E-01	18.85	1.329E-04
Test29_22	0.01	1.664E-06	0.30	1.631E-06	45.62	4.850E-05
Test29_24	0.01	4.232E-02	0.04	3.913E-02	55.02	3.826E-05

is that it can allow *inexactness* in the subproblem solutions in each iteration while maintaining a theoretical convergence guarantee, which is new to the literature on gradient sampling methods. The unique feature of the second algorithm is that it can use inexact subproblem solutions and *aggregated* gradients in place of individual gradients in the subproblem definitions. Our numerical experiments show that employing inexactness and aggregation can each reduce the CPU time required.

References

[1] P. Apkarian, D. Noll, and O. Prot. A trust region spectral bundle method for nonconvex eigenvalue optimization. *SIAM Journal on Optimization*, 19(1):281–306, 2008.

[2] D. P. Bertsekas. *Convex Optimization Theory*. Athena Scientific, Nashua, NH, USA, 2009.

[3] J. V. Burke, A. S. Lewis, and M. L. Overton. Approximating Subdifferentials by Random Sampling of Gradients. *Math. Oper. Res.*, 27(3):567–584, 2002.

[4] 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.

[5] James V. Burke, Frank E. Curtis, Adrian S. Lewis, Michael L. Overton, and Lucas E. A. Simões. Gradient Sampling Methods for Nonsmooth Optimization. In *Numerical Nonsmooth Optimization*, chapter to appear. Springer, 2019.

[6] F. H. Clarke. *Optimization and Nonsmooth Analysis*. Canadian Mathematical Society Series of Monographs and Advanced Texts. John Wiley & Sons, New York, NY, USA, 1983.

[7] F. E. Curtis. NonOpt. `https://coral.ise.lehigh.edu/frankecurtis/nonopt/`, 2020.

[8] 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.

[9] Frank E. Curtis and Xiaocun Que. An Adaptive Gradient Sampling Algorithm for Nonsmooth Optimization. *Optimization Methods and Software*, 28(6):1302–1324, 2013.

[10] Frank E. Curtis and Xiaocun Que. A Quasi-Newton Algorithm for Nonconvex, Nonsmooth Optimization with Global Convergence Guarantees. *Mathematical Programming Computation*, 7(4):399–428, 2015.

[11] Frank E. Curtis, Daniel P. Robinson, and Baoyu Zhou. A Self-Correcting Variable-Metric Algorithm Framework for Nonsmooth Optimization. *IMA Journal of Numerical Analysis*, 10.1093/imanum/drz008, 2019.

[12] A. A. Goldstein. Optimization of Lipschitz continuous functions. *Mathematical Programming*, 13(1):14–22, 1977.

[13] N. 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.

[14] N. Haarala, K. Miettinen, and M. M. Mäkelä. Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Mathematical Programming*, 109(1):181–205, 2007.

[15] W. Hare and C. Sagastizábal. A redistributed proximal bundle method for nonconvex optimization. *SIAM Journal on Optimization*, 20(5):2442–2473, 2010.

[16] E. S. Helou, S. A. Santos, and L. E. A. Simões. On the Local Convergence Analysis of the Gradient Sampling Method for Finite Max-Functions. *Journal of Optimization Theory and Applications*, 175(1):137–157, 2017.

[17] 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.

[18] S. Hosseini and A. Uschmajew. A Riemannian Gradient Sampling Algorithm for Nonsmooth Optimization on Manifolds. *SIAM Journal on Optimization*, 27(1):173–189, 2017.

[19] N. Karmitsa. LMBM. `http://napsu.karmitsa.fi/lmbm`, accessed May, 2020.

[20] K. C. Kiwiel. A linearization algorithm for nonsmooth minimization. *Mathematics of Operations Research*, 10(2):185–194, 1985.

[21] K. C. Kiwiel. *Methods of Descent for Nondifferentiable Optimization*. Lecture Notes in Mathematics. Springer-Verlag, New York, NY, USA, 1985.

[22] K. C. Kiwiel. Restricted step and Levenberg-Marquardt techniques in proximal bundle methods for nonconvex nondifferentiable optimization. *SIAM Journal on Optimization*, 6(1):227–249, 1996.

[23] K. C. Kiwiel. Convergence of the Gradient Sampling Algorithm for Nonsmooth Nonconvex Optimization. *SIAM Journal on Optimization*, 18(2):379–388, 2007.

[24] C. Lemaréchal, A. Nemirovskii, and Yu. Nesterov. New variants of bundle methods. *Mathematical Programming*, 69(1):111–147, 1995.

[25] A. S. Lewis and M. L. Overton. Nonsmooth Optimization via Quasi-Newton Methods. *Mathematical Programming*, 141(1–2):135–163, 2013.

[26] L. Luksan. Test Problems in Fortran. `http://www.cs.cas.cz/~luksan/test.html`, accessed May, 2020.

[27] L. Lukšan and J. Vlček. A bundle-Newton method for nonsmooth unconstrained minimization. *Mathematical Programming*, 83(1):373–391, 1998.

[28] 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.

[29] Morteza Maleknia and Mostafa Shamsi. A gradient sampling method based on ideal direction for solving nonsmooth nonconvex optimization problems: convergence analysis and numerical experiments, 2019. https://arxiv.org/abs/1907.01321.

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

[31] R. Mifflin. A modification and an extension of Lemarechal's algorithm for nonsmooth minimization. In D. C. Sorensen and R. J.-B. Wets, editors, *Nondifferential and Variational Techniques in Optimization*, pages 77–90. Springer Berlin Heidelberg, Berlin, Heidelberg, 1982.

[32] R. Mifflin and C. Sagastizábal. A $\mathcal{VU}$-algorithm for convex minimization. *Mathematical Programming*, 104(2):583–608, 2005.

[33] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, New York, NY, USA, 2nd edition, 2006.

[34] A. Ruszczynski. *Nonlinear Optimization*. Princeton University Press, Princeton, NJ, USA, 2006.

[35] H. Schramm and J. Zowe. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM Journal on Optimization*, 2(1):121–152, 1992.

[36] C.-M. Tang, S. Liu, J.-B. Jian, and J.-L. Li. A Feasible SQP-GS Algorithm for Nonconvex, Nonsmooth Constrained Optimization. *Numerical Algorithms*, 65(1):1–22, Jan 2014.

A GS Algorithm Subroutines

In this appendix, we present subroutines needed for the adaptive gradient sampling algorithms with inexactness and aggregation proposed in this paper. These subroutines have been motivated and presented in previous articles, as mentioned in the following subsections. We include them here, along with brief descriptions of their properties, for ease of reference for the main body of the paper.

A.1 Line search

Given a descent direction for f at x_k, the line search is intended to find a stepsize satisfying the weak Armijo-Wolfe conditions; see (39) below and, e.g., [33]. However, as motivated in [10], the line search may terminate early if the sample set size indicator p_k is less than the prescribed integer $p \in [n+1, \infty)$ or may ignore the curvature condition (39b)—and switch to a "backtracking Armijo" line search—if both $p_k \geq p$ and a certain number of iterations of the line search have already been performed without termination. This potential switch to a backtracking Armijo line search may be needed due to potential nonsmoothness of f, since under Assumption 1.1 one can only show that an Armijo-Wolfe line search can "bracket" a stepsize satisfying the Armijo-Wolfe conditions (39); see [25, Theorem 4.7]. One could ensure finite termination of an Armijo-Wolfe line search, without having to switch to a backtracking Armijo line search as a backup, with a stronger assumption on f, such as it being weakly lower semismooth [30].

Our line search subroutine is stated as Algorithm 3. Given $\underline{\eta} \in (0,1)$ and $\overline{\eta} \in (\underline{\eta}, 1)$, the Armijo-Wolfe conditions that we use are

$$f(x_k) - f(x_k + \alpha_k d_k) > \underline{\eta}\alpha_k \max\{\|d_k\|_2^2, \|G_k y_k\|_2^2\} \tag{39a}$$

$$\text{and} \quad v^T d_k \geq \overline{\eta} \nabla f(x_k)^T d_k, \text{ where } v \in \partial f(x_k + \alpha_k d_k). \tag{39b}$$

We remark that (39a) does not use the directional derivative of f at x_k along d_k, as is typical in the Armijo condition in the context of smooth optimization; rather, it uses squared norms of search direction quantities, which is common in GS algorithms when nonnormalized search directions are used; see, e.g., [23, Eq. (4.2)].

A.2 Iterate perturbation

Gradient sampling algorithms require that each iterate lies in the set of points over which the objective function f is continuously differentiable [5]. For our proposed algorithms, we employ Algorithm 4 to ensure that each iterate lies in the set $\mathcal{D}$ defined in Assumption 1.1. If, after the line search, 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, the iterate perturbation strategy in Algorithm 4 aims to compute $x_{k+1} \in \mathcal{D}$ satisfying

$$f(x_k) - f(x_{k+1}) \geq \underline{\eta}\alpha_k \max\{\|d_k\|_2^2, \|G_k y_k\|_2^2\}, \tag{40a}$$

$$\nabla f(x_{k+1})^T d_k \geq \overline{\eta} \nabla f(x_k)^T d_k, \tag{40b}$$

$$\text{and} \quad \|x_k + \alpha_k d_k - x_{k+1}\|_2 \leq \min\{\alpha_k, \epsilon_k\} \min\{\|d_k\|_2, \|G_k y_k\|_2\}. \tag{40c}$$

Algorithm 4 can fail if its **for** loop iterates infinitely. However, under Assumption 1.1, this is a probability zero event. In other words, it terminates finitely—meaning the subroutine runs successfully—with probability one.

A.3 Hessian and inverse Hessian approximation strategy

The Hessian approximation strategy employed in [10] is conservative in the sense that it might replace a BFGS approximation with an L-BFGS approximation in order to ensure that, in certain cases, the eigenvalues of the Hessian approximation are bounded above and below away from zero. For our purposes, we employ the less conservative strategy advocated in [11], which exploits the *self-correcting* properties of BFGS updating. The subroutine we use is stated in Algorithm 5.

A.4 Sample point genearation

With $x_{k+1} \in \mathcal{D}$ in hand, a GS algorithm turns to setting the set of sample points $\mathcal{X}_{k+1}$ and corresponding size indicator p_{k+1}. To limit the size of the sample set, which has the benefit of reducing the costs of subsequent QP solves, we follow the lead of [10], which sets $\mathcal{X}_{k+1} \leftarrow \{x_{k+1}\}$ if, for $(\xi, \underline{\alpha}) \in (0, \infty)^2$, one finds

$$\|d_k\|_{H_k}^2 \geq \xi \|d_k\|_2^2 \tag{42a}$$

$$\text{and} \quad \alpha_k \geq \underline{\alpha}. \tag{42b}$$

Otherwise, the sample set preserves points near x_{k+1} and augments it with randomly generated points, the hallmark of GS methods; see Algorithm 6.

Like for Algorithm 4, one finds that Algorithm 6 can fail if its **do-while** loop iterates infinitely. However, under Assumption 1.1, this occurs with probability zero. The subroutine runs successfully with probability one.

Algorithm 1 Gradient Sampling Algorithm with Inexact Subproblem Solutions

Require: $(\nu, \underline{\alpha}) \in (0,\infty)^2$; $(\rho, \kappa, \iota, \underline{\eta}) \in (0,1)^5$; $\psi \in (0,1]$; $\overline{\eta} \in (\underline{\eta}, 1)$; $p \in \mathbb{N}$ with $p \geq n+1$; $\sigma \in (0, \sqrt{2}-1)$; $x_0 \in \mathcal{D}$; $H_0 \succ 0$; $\epsilon_0 \in (0,\infty)$.

Set $W_0 \leftarrow H_0^{-1}$, $\mathcal{X}_0 \leftarrow \{x_0\}$, $p_0 \leftarrow 0$, G_0 by (4), and $\sigma_0 \leftarrow \sigma$.
for all $k \in \mathbb{N}$ **do**
  if $\|\nabla f(x_k)\|_2 = 0$ **then**
    terminate and **return** the stationary point x_k.
  end if
  Set
$$\tau_k \leftarrow \sigma_k^2 + 2\sigma_k \in (0,1). \tag{8}$$
  for all $j \in \mathbb{N}$ **do**
    Set $j_q \leftarrow \underset{i \in \{0,\ldots,j\}}{\arg\min}\ q_k(d_{k,i}, z_{k,i})$ and $j_\theta \leftarrow \underset{i \in \{0,\ldots,j\}}{\arg\max}\ \theta_k(y_{k,i})$.
    if $q_k(d_{k,j_q}, z_{k,j_q}) \geq 0$, **then** set $\lambda_{k,j_q} = \infty$
    else set
$$\lambda_{k,j_q} \leftarrow \max\left\{1 - \frac{\sigma_k^2 + 2\sigma_k}{\frac{\theta_k(y_{k,0})}{q_k(d_{k,j_q}, z_{k,j_q})} - 1}, \rho\right\}. \tag{9}$$
    end if
    if
$$\max\{\|W_k G_k y_{k,j_\theta}\|_2, \|G_k y_{k,j_\theta}\|_2\} \leq \nu\epsilon_k, \tag{10}$$
      then break;
    else if
$$-\nabla f(x_k)^T W_k G_k y_{k,j_\theta} \leq -\kappa y_{k,j_\theta}^T G_k^T W_k G_k y_{k,j_\theta} \tag{11}$$
      and either
$$q_k(d_{k,j_q}, z_{k,j_q}) - \theta_k(y_{k,j_\theta}) \leq \tau_k(-q_k(d_{k,j_q}, z_{k,j_q})) \tag{12}$$
      or
$$\theta_k(y_{k,j_\theta}) - \theta_k(y_{k,0}) \geq \lambda_{k,j_q}(q_k(d_{k,j_q}, z_{k,j_q}) - \theta_k(y_{k,0})) \tag{13}$$
      then break.
    end if
  end for
  Set $(d_k, y_k) \leftarrow (-W_k G_k y_{k,j_\theta}, y_{k,j_\theta})$.
  Set $\alpha_k \geq 0$ by Algorithm 3.
  if (10) holds (with $y_k \equiv y_{k,j_\theta}$)
    set $\epsilon_{k+1} \leftarrow \psi\epsilon_k$ and $\sigma_{k+1} \leftarrow \sigma$;
  else if $\alpha_k \geq \underline{\alpha}$
    set $\epsilon_{k+1} \leftarrow \epsilon_k$ and $\sigma_{k+1} \leftarrow \sigma_k$;
  else
    set $\epsilon_{k+1} \leftarrow \epsilon_k$ and $\sigma_{k+1} \leftarrow \iota\sigma_k$.
  end if
  Set $x_{k+1} \in \mathcal{D}$ by Algorithm 4.
  Set (H_{k+1}, W_{k+1}) by Algorithm 5.
  Set $(\mathcal{X}_{k+1}, p_{k+1})$ by Algorithm 6 and G_{k+1} by (4).
end for

Algorithm 2 GS with Inexact Subproblem Solutions and Gradient Aggregation

Require: [. . . same parameters and initial values as in Algorithm 1, except G_0 . . .]

Set G_0^{full} by (4), G_0^{agg} by (4), and $\alpha_{-1} \leftarrow 0$.
for all $k \in \mathbb{N}$ **do**
  if $\alpha_{k-1} > 0$ or $p_k \geq p$ **then**
    set $G_k \leftarrow G_k^{\text{full}}$;
  else
    set $G_k \leftarrow G_k^{\text{agg}}$.
  end if
  [. . . same as Line 3 through Line 30 of Algorithm 1 . . .]
  Set $(\mathcal{X}_{k+1}, p_{k+1})$ by Algorithm 6 and G_{k+1}^{full} by (4).
  if $\alpha_k > 0$ **then**
    set $G_{k+1}^{\text{agg}} \leftarrow G_{k+1}^{\text{full}}$;
  else
    set $G_{k+1}^{\text{agg}} \leftarrow \left[\nabla f(x_{k+1}) \quad G_k y_k \quad [\nabla f(x)]_{x \in \mathcal{X}_{k+1} \setminus (x_{k+1} \cup \mathcal{X}_k)}\right]$.
  end if
end for

Algorithm 3 Armijo-Wolfe Line Search

Require: $(\underline{\eta}, \overline{\eta}, \underline{\alpha})$ from outer algorithm; $\overline{\alpha} \in [\underline{\alpha}, \infty)$; $\gamma \in (0, 1)$.

Set $l \leftarrow 0$, $u \leftarrow \overline{\alpha}$, and $\alpha_k \leftarrow \gamma\overline{\alpha}$.
if $d_k = 0$ **then**
  terminate and **return** α_k.
end if
for all $\ell \in \mathbb{N}$ **do**
  if $p_k < p$ and $\alpha_k < \underline{\alpha}$ **then**
    set $\alpha_k \leftarrow 0$, **terminate**, and **return** α_k. [truncate and take null stepsize]
  end if
  if $\alpha_k < \underline{\alpha}$ **then**
    set $l \leftarrow 0$. [switch to backtracking Armijo line search]
  end if
  if (39) holds or both $\alpha_k < \underline{\alpha}$ and (39a) hold **then**
    terminate and **return** α_k. [success]
  end if
  if (39a) does not hold **then**
    set $u \leftarrow \alpha_k$;
  else
    set $l \leftarrow \alpha_k$.
  end if
  set $\alpha_k \leftarrow (1 - \gamma)l + \gamma u$.
end for

Algorithm 4 Iterate Perturbation

Require: $\overline{\ell} \in \mathbb{N}$.

Set $x_{k+1} \leftarrow x_k + \alpha_k d_k$.
if $\alpha_k = 0$ or $d_k = 0$ **then**
    terminate and **return** x_{k+1}.
end if
for all $\ell \in \mathbb{N}$ **do**
    if $x_{k+1} \in \mathcal{D}$ and either (40) holds or each of (40a), (40c), and $\ell > \overline{\ell}$ hold **then**
        terminate and **return** x_{k+1}
    end if
    Sample x_{k+1} from a uniform distribution over

$$\mathbb{B}\left(x_k + \alpha_k d_k, \frac{\min\{\alpha_k, \epsilon_k\} \min\{\|d_k\|_2, \|G_k y_k\|_2\}}{\ell \max\{\|d_k\|_2, \|G_k y_k\|_2\}}\right).$$

end for

Algorithm 5 Hessian and Inverse Hessian Approximation Updates

Require: $\underline{\phi} \in (0,1)$; $\overline{\phi} \in (1,\infty)$.

Set $s_k \leftarrow \alpha_k d_k$ and $\hat{v}_k \leftarrow x_{k+1} - x_k$.
if $s_k = 0$ **then**
    set $(H_{k+1}, W_{k+1}) \leftarrow (H_k, W_k)$.
else
    Compute ϑ_k as the smallest value in $[0,1]$ such that

$$v_k \leftarrow \vartheta_k s_k + (1 - \vartheta_k) y_k$$

    yields

$$\underline{\phi} \leq \frac{s_k^T v_k}{\|s_k\|_2^2} \quad \text{and} \quad \frac{\|v_k\|_2^2}{s_k^T v_k} \leq \overline{\phi}, \tag{41}$$

    then set

$$H_{k+1} \leftarrow \left(I - \frac{s_k s_k^T H_k}{s_k^T H_k s_k}\right)^T H_k \left(I - \frac{s_k s_k^T H_k}{s_k^T H_k s_k}\right) + \frac{v_k v_k^T}{s_k^T v_k}$$

$$\text{and} \quad W_{k+1} \leftarrow \left(I - \frac{v_k s_k^T}{s_k^T v_k}\right)^T W_k \left(I - \frac{v_k s_k^T}{s_k^T v_k}\right) + \frac{s_k s_k^T}{s_k^T v_k}.$$

end if
terminate and **return** (H_{k+1}, W_{k+1}).

Algorithm 6 Sample Set Update

Require: $\underline{\alpha}$ from outer algorithm; $\xi \in (0, \infty)$; $\overline{p} \in \mathbb{N}$ with $\overline{p} \geq 1$.

if (42) holds **then**
    set $\mathcal{X}_{k+1} \leftarrow \{x_{k+1}\}$ and $p_{k+1} \leftarrow 0$, **terminate**, and **return** $(\mathcal{X}_{k+1}, p_{k+1})$.
end if
do
    set $\mathcal{S}_{k+1}$ as a set of $\overline{p}$ points from a uniform distribution over $\mathbb{B}(x_{k+1}, \epsilon_{k+1})$.
while $\mathcal{S}_{k+1} \not\subset \mathcal{D}$
Set $\mathcal{X}_{k+1} \leftarrow \{x_{k+1}\} \cup (\mathcal{X}_k \cap \mathbb{B}(x_{k+1}, \epsilon_{k+1})) \cup \mathcal{S}_{k+1}$ and $p_{k+1} \leftarrow |\mathcal{X}_{k+1}| - 1$.
if $p_{k+1} > p$ **then**
    remove the $p_{k+1} - p$ eldest members of $\mathcal{X}_{k+1}$ (except $\{x_{k+1}\}$) and set $p_{k+1} \leftarrow p$.
end if
terminate and **return** $(\mathcal{X}_{k+1}, p_{k+1})$.