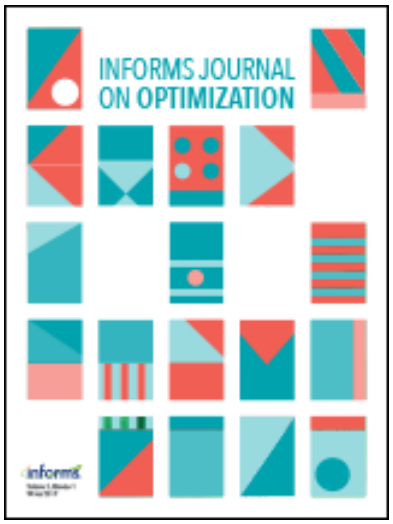


INFORMS Journal on Optimization

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Gradient Sampling Methods with Inexact Subproblem Solutions and Gradient Aggregation

Frank E. Curtis, Minhan Li

To cite this article:
Frank E. Curtis, Minhan Li (2022) Gradient Sampling Methods with Inexact Subproblem Solutions and Gradient Aggregation. INFORMS Journal on Optimization

Published online in Articles in Advance 01 Apr 2022

. https://doi.org/10.1287/ijoo.2022.0073

http://pubsonline.informs.org/journal/ijoo

INFORMS JOURNAL ON OPTIMIZATION
Articles in Advance, pp. 1–20
ISSN 2575-1484 (print), ISSN 2575-1492 (online)

Gradient Sampling Methods with Inexact Subproblem Solutions and Gradient Aggregation

Frank E. Curtis,[a,*] **Minhan Li**[a]

[a] Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, Pennsylvania 18015
*Corresponding author
Contact: frank.e.curtis@gmail.com, https://orcid.org/0000-0001-7214-9187 (FEC); mil417@lehigh.edu (ML)

Received: August 6, 2021
Revised: January 2, 2022
Accepted: January 8, 2022
Published Online in Articles in Advance: April 1, 2022

https://doi.org/10.1287/ijoo.2022.0073

Abstract. Gradient sampling (GS) methods for the minimization of objective functions that may be nonconvex and/or nonsmooth are proposed, analyzed, and tested. One of the most computationally expensive components of contemporary GS methods is the need to solve a convex quadratic subproblem in each iteration. By contrast, the methods proposed in this paper allow 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 computational effort required by a GS method. Additionally, a strategy is proposed for aggregating gradient information after a subproblem is solved (potentially 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 can also reduce the computational effort required by a GS method.

Funding: This work was supported by the National Science Foundation Division of Computing and Communication Foundations [Grants CCF–1618717, CCF–1740796].
Supplemental Material: The online companion is available at https://doi.org/10.1287/ijoo.2022.0073.

Keywords: nonsmooth optimization • nonconvex optimization • gradient sampling • inexact subproblem solutions • gradient aggregation

1. Introduction

The gradient sampling (GS) methodology (Burke et al. 2005) 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, here 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 and 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 computational savings through consistently reduced per-iteration costs.
- Bundle methods represent another important class of algorithms for nonsmooth minimization. Implementations of bundle methods can be made significantly more efficient through the use of *subgradient aggregation*, wherein one can compress the information from a QP solution such that a subsequent QP can be solved more 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 because the GS methodology requires the identification of points at which the objective function is differentiable or even continuously differentiable, at which gradients are to be evaluated when search directions are being computed.) 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 noticeable reductions in required computational effort.

1.1. Literature Review

The GS methodology was introduced by Burke et al. (2005); see also Burke et al. (2002). Shortly after, Kiwiel (2007) 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. Kiwiel (2010) also proposed and analyzed a derivative-free GS method aimed at avoiding the cost of gradient computations. Later, Curtis and Que (2013, 2015) showed 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 Curtis et al. (2020) for how to loosen the restrictions on the Hessian approximation scheme. A feature of the algorithms in all of these articles is that the analyses require that the convex QP subproblems for computing search directions be solved exactly in every iteration.

A method for reducing the costs associated with solving QPs in a GS method was proposed by Maleknia and Shamsi (2020). In this work, the authors argue that an "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 this one 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 by Maleknia and Shamsi (2020) 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 (Helou et al. 2017) and to solve constrained optimization problems (Curtis and Overton 2012, Tang et al. 2014, Hosseini and Uschmajew 2017). See Burke et al. (2020) for further discussion. Such extensions are beyond the scope of this article, wherein we focus on techniques for unconstrained minimization that ensure convergence from an 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 (Mifflin 1982; Kiwiel 1985a, b, 1996; Schramm and Zowe 1992; Hiriart-Urruty and Lemaréchal 1993; Lemaréchal et al. 1995; Lukšan and Vlček 1998; Haarala et al. 2004, 2007; Mifflin and Sagastizábal 2005; Ruszczynski 2006; Apkarian et al. 2008; Hare and Sagastizábal 2010). We also direct the reader's attention to more modern bundle methods that exploit inexact function value and derivative information; see, for example, de Oliveira et al. (2014), de Oliveira and Solodov (2016), van Ackooij and Frangioni (2018), and Hare et al. (2016). The technique employed in some bundle methods that is relevant for this paper is that of subgradient aggregation; see, for example, Kiwiel (1985a). The use of aggregation in this paper is similar although the surrounding convergence analysis is different because of the distinct differences in the convergence analyses of bundle and GS methodologies. For one thing, convergence analyses of GS methods are inherently probabilistic because of the random sampling of points.

1.2. Notation

We write $\mathbb{R}$ to denote the set of real numbers, $\mathbb{R}^n$ to denote the set of n-dimensional real vectors, and $\mathbb{R}^{m\times n}$ to denote the set of m-by-n-dimensional real matrices. For any symmetric $H \in \mathbb{R}^{n\times n}$, we write $H \succ 0$ to indicate that H is positive definite. For $H \in \mathbb{R}^{n\times n}$ such that $H \succ 0$, we define the corresponding norm $\|v\|_H := \sqrt{v^T H v}$ for any $v \in \mathbb{R}^n$. We write $\mathbb{N} := \{0, 1, 2, \dots\}$ to denote the set of nonnegative integers and use 1 to denote a vector of ones whose length is determined by the context in which it appears (e.g., through an inner product with a vector of known length). For any countable set $\mathcal{S}$, we write $|\mathcal{S}|$ to denote the cardinality of $\mathcal{S}$.

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. *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 GS algorithms, each of which is designed to produce an iterate sequence—that is, $\{x_k\}$ 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. Throughout, we refer to stationarity in the sense of Clarke (1983). Such stationarity for f can be defined as follows. By Rademacher's theorem under Assumption 1, the (Clarke) set of generalized gradients of f at $x \in \mathbb{R}^n$ is given by

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

see Clarke (1983, theorem 2.5.1). For $\epsilon \in [0,\infty)$, the set of ϵ-generalized gradients of f at $x \in \mathbb{R}^n$ is

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

One finds that $\eth_0 f(x) \equiv \eth f(x)$; see Goldstein (1977, corollary 2.5). A point $x \in \mathbb{R}^n$ is said to be ϵ-stationary for f if $0 \in \eth_\epsilon f(x)$ and is said to be stationary for f if $0 \in \eth f(x)$.

One could remove from Assumption 1 the assumption that f is bounded below, in which case the methods we propose would terminate finitely at a stationary point for f or, with probability one, generate iterates that either converge to (ϵ-)stationarity for f (see Theorems 1 and 2) or have objective values that diverge to $-\infty$. However, to focus on the more interesting setting, we include in Assumption 1 that f is bounded below, meaning this latter case cannot occur. For the algorithms that we propose to be well-posed, one only needs to assume 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. See Burke et al. (2020) for further discussion.

The first algorithm that we propose (see Algorithm 1) has a nested loop (with the "inner" loop being stated in Algorithm 2). 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; for example, $d_{k,j}$.

1.3. Outline

In Section 2, we propose and analyze an algorithm that employs inexact subproblem solutions. In Section 3, we propose gradient aggregation within a GS method and show that it can be used while maintaining the same guarantees as the method from Section 2. Numerical experiments employing both techniques are presented in Section 4. Concluding remarks are given in Section 5.

2. GS Algorithm with Inexact Subproblem Solutions

We propose a GS algorithm that allows for the use of inexact subproblem solutions in each iteration. In this section, we present the proposed algorithm and then prove that iterates generated by it converge to (ϵ-)stationarity with probability one. In our presentation, we focus on the components of the algorithm and analysis that are distinct from previous GS methods. Details that are not new are provided in an online companion to this article.

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

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

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

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

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.

Letting $(d_{k,*}, z_{k,*})$ denote the optimal solution of (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

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

The optimal solution $y_{k,*}$ of (D) 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, that is, the W_k-projection of the origin onto this hull. The following lemma reveals important properties of these solutions.

Lemma 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*

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

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 (KKT) conditions for (5); see, for example, Curtis and Que (2013, equation (27); 2015, 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, sequences of inner iterates of a solver for the primal-dual subproblems (5). 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 algorithm requires that *both* primal and dual QP iterate sequences are generated. However, this is not an expensive requirement. After all, motivated by Lemma 1, one may choose for a given $y_{k,j} \in \mathbb{R}^{p_k+1}$ to set

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

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

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

Assumption 2. *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 $(k,j) \in \mathbb{N} \times \mathbb{N}$, one has*

$$G_k^T d_{k,j} \leq z_{k,j} \mathbb{1}, \ \mathbb{1}^T y_{k,j} = 1, \text{ and } \ y_{k,j} \geq 0,$$

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

Under Assumption 2, the primal and dual iterates satisfy weak duality with respect to (5) for all $(k,j) \in \mathbb{N} \times \mathbb{N}$. 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

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

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

Our algorithm with inexact subproblem solutions is stated as Algorithm 1, for which the details of the search direction computation are stated in Algorithm 2. The statement of Algorithm 1 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 (a feature required by the theoretical convergence analyses of all GS methods), sample set updates, and quasi-Newton updates are similar to those used in Curtis and Que (2015) and Curtis et al. (2020). Hence, we relegate them to the online companion. The algorithm also requires a subroutine for setting parameters related to the quasi-Newton updates that influence the line search subroutine. The approach is derived from properties of the self-correcting nature of quasi-Newton updating; see Byrd and Nocedal (1989). This subroutine is also provided in the online companion.

Each iteration of Algorithm 2 takes an approximate subproblem solution from the QP solver. The loop 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 Algorithm 1 reduces the sampling radius. On the other hand, if (11) holds along with either (12) or (13) (see Algorithm 2), then our analysis in the following section reveals that a sufficiently accurate QP solution yielding a direction of sufficient descent has been obtained. Condition (11) is motivated by Lemma 1, specifically (6), because $(d_k, y_k) = (-W_k G_k y_{k,j_\theta}, y_{k,j_\theta})$ 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) (see Algorithm 2), is explained in the following section.

Notice that an implementation of Algorithm 2 does not require storage and a search through all previous subproblem solutions when determining the indices in line 4. One only needs to store the best (in terms of objective values) primal and dual solution estimates during the loop and employ these values when checking for termination of the loop. Line 4 is only written in this manner for ease of exposition and to allow us to consider situations in which these inner iterations do not necessarily produce primal and dual subproblem solutions that have objective values that converge monotonically to the optimal value.

Algorithm 1 (Gradient Sampling Algorithm with Inexact Subproblem Solutions)

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

1: Set $W_0 \leftarrow H_0^{-1}$, $\mathcal{X}_0 \leftarrow \{x_0\}$, $p_0 \leftarrow 0$, G_0 by (4), and $\sigma_0 \leftarrow \sigma$.
2: Set $(\underline{\eta}, \mu) \in (0,1) \times (1,\infty)$ by Online Algorithm 4 (Parameter Selection).
3: Choose $\overline{\eta} \in (\underline{\eta}, 1)$.
4: **for all** $k \in \mathbb{N}$ **do**
5: **if** $\|\nabla f(x_k)\|_2 = 0$ **then**
6: **terminate** and **return** the stationary point x_k.
7: **end if**
8: Set y_{k,j_θ} by Algorithm 2 (Search Direction Computation).
9: Set $(d_k, y_k) \leftarrow (-W_k G_k y_{k,j_\theta}, y_{k,j_\theta})$.
10: Set $\alpha_k \geq 0$ by Online Algorithm 5 (Armijo–Wolfe Line Search).
11: **if** (10) (see Algorithm 2) holds with $y_{k,j_\theta} \equiv y_k$
12: set $\epsilon_{k+1} \leftarrow \psi \epsilon_k$ and $\sigma_{k+1} \leftarrow \sigma$;
13: **else if** $\alpha_k \geq \underline{\alpha}$
14: set $\epsilon_{k+1} \leftarrow \epsilon_k$ and $\sigma_{k+1} \leftarrow \sigma_k$;
15: **else**
16: set $\epsilon_{k+1} \leftarrow \epsilon_k$ and $\sigma_{k+1} \leftarrow \iota \sigma_k$.
17: **end if**
18: Set $x_{k+1} \in \mathcal{D}$ by Online Algorithm 6 (Iterate Perturbation).
19: Set (H_{k+1}, W_{k+1}) by Online Algorithm 7 (Approximation Updates).
20: Set $(\mathcal{X}_{k+1}, p_{k+1})$ by Online Algorithm 8 (Sample Set Update) and G_{k+1} by (4).
21: **end for**

Algorithm 2 (Search Direction Computation for Algorithm 1)

Require: $\nu \in (0,\infty)$; $(\rho, \kappa) \in (0,1)^2$

1: Set

$$\tau_k \leftarrow \sigma_k^2 + 2\sigma_k \in (0,1). \tag{8}$$

2: **for all** $j \in \mathbb{N}$ **do**
3: Set $(d_{k,j}, z_{k,j}, y_{k,j})$ satisfying Assumption 2 from the jth iterate from a solver for (5).
4: Set $j_q \leftarrow \arg\min_{i \in \{0,\ldots,j\}} q_k(d_{k,i}, z_{k,i})$ and $j_\theta \leftarrow \arg\max_{i \in \{0,\ldots,j\}} \theta_k(y_{k,i})$.
5: **if** $q_k(d_{k,j_q}, z_{k,j_q}) \geq 0$, **then** set $\lambda_{k,j_q} = \infty$
6: **else** set

$$\lambda_{k,j_q} \leftarrow \max\left\{1 - \frac{\sigma_k^2 + 2\sigma_k}{\frac{\theta_k(y_{k,0})}{q_k(d_{k,j_q}, z_{k,j_q})} - 1}, \rho\right\}. \tag{9}$$

7: **end if**

8: **if**

$$\max\{\|W_k G_k y_{k,j_\theta}\|_2,\ \|G_k y_{k,j_\theta}\|_2\} \leq \nu\epsilon_k, \tag{10}$$

9: **then terminate** and **return** y_{k,j_θ};
10: **else if**

$$-\nabla f(x_k)^T W_k G_k y_{k,j_\theta} \leq -\kappa y_{k,j_\theta}^T G_k^T W_k G_k y_{k,j_\theta} \tag{11}$$

11: **and either**

$$q_k(d_{k,j_q}, z_{k,j_q}) - \theta_k(y_{k,j_\theta}) \leq \tau_k(-q_k(d_{k,j_q}, z_{k,j_q})) \tag{12}$$

12: **or**

$$\theta_k(y_{k,j_\theta}) - \theta_k(y_{k,0}) \geq \lambda_{k,j_q}(q_k(d_{k,j_q}, z_{k,j_q}) - \theta_k(y_{k,0})) \tag{13}$$

13: **then terminate** and **return** y_{k,j_θ}.
14: **end if**
15: **end for**

2.2. Inexactness Conditions for the QP Solver

Convergence analyses of GS 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 Burke et al. (2005, lemma 3.1) and Kiwiel (2007, lemma 3.1) and is proved in a more general setting as Curtis and Que (2015, lemma 3.5). The conditions that we impose on inexact subproblem solutions are motivated by trying to ensure a property of this type but in an even slightly more general setting. 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 $\mathcal{S}$, that is,

$$P_W(\mathcal{S}) := \arg\min_{s \in \mathcal{S}} \|s\|_W. \tag{14}$$

Our new generalization of the lemma can be seen in Inequality (15), which does not require that a given vector $u \in \mathcal{S}$ is sufficiently close to the W-projection of the origin, but merely sufficiently close to a small enough neighborhood of this projection.

Lemma 2. *Suppose $\mathcal{S} \subseteq \mathbb{R}^n$ is a compact and convex set with $0 \notin \mathcal{S}$. For any $\beta \in (0,1)$ and $W \succ 0$, there exists $(\varsigma, \delta) \in (0,\infty)^2$ such that, for any $(u, v, \overline{\varsigma}, \overline{\delta}) \in \mathcal{S} \times \mathcal{S} \times (0,\varsigma] \times (0,\delta]$ with*

$$\|u\|_W \leq (1+\overline{\varsigma})\,\|P_W(\mathcal{S})\|_W + \overline{\delta} \tag{15}$$

(where $P_W(\mathcal{S})$ is defined in (14)), it follows that $v^T W u > \beta \|u\|_W^2$.

Proof. Consider arbitrary $\beta \in (0,1)$ and $W \succ 0$. To derive a contradiction, suppose the implication is false; that is, for any $(\varsigma, \delta) \in (0,\infty)^2$, there exists $(u, v, \overline{\varsigma}, \overline{\delta}) \in \mathcal{S} \times \mathcal{S} \times (0,\varsigma] \times (0,\delta]$ with

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

This means that there exist infinite sequences $\{u_i\} \subset \mathcal{S}$ and $\{v_i\} \subset \mathcal{S}$ such that

$$\|u_i\|_W \leq (1+1/i)\,\|P_W(\mathcal{S})\|_W + 1/i \text{ and } v_i^T W u_i \leq \beta \|u_i\|_W^2 \quad \text{for all } i \in \mathbb{N}. \tag{16}$$

Because $\mathcal{S}$ is compact, these sequences have convergent subsequences; hence, without loss of generality, one can assume that $\{u_i\} \to u$ and $\{v_i\} \to v$ for some $(u, v) \in \mathcal{S} \times \mathcal{S}$ with

$$v^T W u \leq \beta \|u\|_W^2 . \tag{17}$$

On the other hand, by the definition of $\{u_i\}$, it follows that $u = P_W(\mathcal{S})$, which is nonzero because $\mathcal{S}$ does not include the origin. Moreover, by Bertsekas (2009, proposition 1.1.8) and the definition of $P_W(\mathcal{S})$ (as the W-projection of the origin onto $\mathcal{S}$), one finds that

$$(0-u)^T W(v-u) \leq 0 \Leftrightarrow v^T W u \geq \|u\|_W^2 ,$$

which contradicts (17) because $\beta \in (0,1)$. □

Our goal now is to prove two lemmas that motivate the use of (12) and (13) as stopping conditions for the loop in Algorithm 2. Specifically, if $\theta_k(y_{k,*}) < 0$, each lemma shows that these conditions—(12) and (13), respectively, in the two lemmas—imply that

$$\begin{aligned} &0 > \theta_k(y_{k,j_\theta}) \geq (1+\sigma_k)^2 \theta_k(y_{k,*}) \\ \Leftrightarrow\ &0 < \|G_k y_{k,j_\theta}\|_{W_k} \leq (1+\sigma_k)\, \|G_k y_{k,*}\|_{W_k}. \end{aligned} \tag{18}$$

Importantly, these algorithmic conditions imply that (18) holds *without knowledge of* $y_{k,*}$. The inequalities in (18) are important because they, along with Lemma 2 (c.f. (15)), play a central role in our convergence analysis in Section 2.3 for Algorithm 1.

Lemma 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 Algorithm 2 (during outer iteration $k \in \mathbb{N}$), one finds that (12) holds. Then, (18) 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) (see Algorithm 2), 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 (18) holds as desired. □

When $\theta_k(y_{k,*}) < 0$, weak duality for (5) implies that (12) (see Algorithm 2) 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) (see Algorithm 2) 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 2 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 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 Algorithm 2 (during outer iteration $k \in \mathbb{N}$), one finds that $q_k(d_{k,j_q}, z_{k,j_q}) < 0$ and (13) holds. Then, (18) holds.*

Proof. By $q_k(d_{k,j_q}, z_{k,j_q}) < 0$ and weak duality for (5), one finds in (9) (see Algorithm 2) that

$$\frac{\theta_k(y_{k,0})}{q_k(d_{k,j_q}, z_{k,j_q})} \geq 1. \tag{19}$$

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 (18) 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 (19) holds strictly. Observing (9) (see Algorithm 2), one finds $\lambda_{k,j_q} \in (0,1)$. This fact, (13) (see Algorithm 2), 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

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

implies that

$$\begin{aligned} \theta_k(y_{k,j_\theta}) &\geq \lambda_{k,j_q}\theta_k(y_{k,*}) + (1-\lambda_{k,j_q})\theta_k(y_{k,0}) \\ &\geq \left(\lambda_{k,j_q} + (1-\lambda_{k,j_q})\frac{\theta_k(y_{k,0})}{q_k(d_{k,j_q}, z_{k,j_q})}\right)\theta_k(y_{k,*}). \end{aligned} \tag{20}$$

In addition, one finds that λ_{k,j_q} in (9) (see Algorithm 2) 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 (20) 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 and 2 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 (Online Algorithm 6) and the sample set update (Online Algorithm 8), 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 the online companion.

Lemma 5. *Algorithm 1 is well-posed; it either terminates finitely or, with probability one, it performs an infinite number of iterations. In any case, for all $k \in \mathcal{K}$, the following hold true.*

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

b. *(d_k, y_k) satisfies $\|d_k\|_{H_k} = \|G_k y_k\|_{W_k}$.*

c. *In line 10, Online Algorithm 5 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*

$$f(x_k) - f(x_k + \alpha_k d_k) > \underline{\eta}\alpha_k \max\{\|d_k\|_2^2, \|G_k y_k\|_2^2\} \tag{21a}$$

$$\text{and } v^T d_k \geq \overline{\eta}\nabla f(x_k)^T d_k, \text{ where } v \in \eth f(x_k + \alpha_k d_k), \tag{21b}$$

or at least (21a) holds (which is sufficient if deemed by Online Algorithm 5).

d. *In line 18, Online Algorithm 6 yields, with probability one, $x_{k+1} \in \mathcal{D}$ satisfying*

$$f(x_k) - f(x_{k+1}) \geq \underline{\eta}\alpha_k \max\{\|d_k\|_2^2, \|G_k y_k\|_2^2\}, \tag{22a}$$

$$\nabla f(x_{k+1})^T d_k \geq \overline{\eta}\nabla f(x_k)^T d_k, \tag{22b}$$

$$\text{and } \|x_k + \alpha_k d_k - x_{k+1}\|_2 \leq \min\{\alpha_k, \epsilon_k\}\min\{\|d_k\|_2, \|G_k y_k\|_2\}, \tag{22c}$$

or at least satisfying (22a) and (22c) (which is sufficient if deemed by Online Algorithm 6).

e. *If line 20 is reached and one finds that*

$$\|d_k\|_{H_k}^2 \geq \xi \|d_k\|_2^2 \text{ and } \alpha_k \geq \underline{\alpha}, \tag{23}$$

then Online Algorithm 8 yields $\mathcal{X}_{k+1} \leftarrow \{x_{k+1}\}$ and $p_{k+1} \leftarrow 0$; otherwise, with probability one,

$$\mathcal{X}_{k+1} \leftarrow (\{x_{k+1}\} \cup \mathcal{S}_{k+1} \cup (\mathcal{X}_k \cap \mathbb{B}(x_{k+1}, \epsilon_{k+1}))) \subset \mathbb{B}(x_{k+1}, \epsilon_{k+1}) \text{ with } p_{k+1} \geq \min\{p_k + 1, p\}.$$

Finally, let $\mathcal{K}_{H,W} := \{k \in \mathcal{K} : \alpha_k d_k =: s_k \neq 0\}$, which are the indices for which Online Algorithm 7 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 exists $(\underline{\mu}, \overline{\mu}) \in (0, \infty)^2$ with $\underline{\mu} \leq \overline{\mu}$ such that, for every $K \in \mathbb{N}$, the following hold for at least $\lceil \chi K \rceil$ values of $k \in \mathcal{K}_{H,W}$:

$$\underline{\mu}\|G_k y_k\|_2^2 \leq \|G_k y_k\|_{W_k}^2 \tag{24a}$$

$$\text{and } \|W_k G_k y_k\|_2^2 \leq \overline{\mu}\|G_k y_k\|_2^2. \tag{24b}$$

If $\mathcal{K}_{H,W}$ is finite, then such constants exist satisfying (24) for all $k \in \mathcal{K}$.

Proof. If the algorithm reaches iteration $k \in \mathcal{K}$ in which the condition in line 5 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, (24) follows from the fact that only a finite number of iterations are performed and the following proof of part (a) of the lemma.

a–d. 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 (EC.1) implies $s_0^T v_0 > 0$ and from well-known properties of Broyden–Fletcher–Goldfarb–Shanno (BFGS) updating; see, for example, Nocedal and Wright (2006, 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 5 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) (because part (a) has been proved already).

b–d. 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). Because $\theta_k(y) \le 0$ 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 $\theta_k(y_{k,*}) = 0$. Because $W_k \succ 0$, this implies that $G_k y_{k,*} = 0$. Under Assumption 2, 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) (see Algorithm 2) 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, 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 1, and the fact that $\tau_k \in (0,1)$ imply that (11) and (12) (see Algorithm 2) 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 Curtis and Que (2015, lemma 2.3).

e. The proof follows in the same manner as that for Curtis and Que (2015, lemma 2.5).

Because 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. 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 Curtis et al. (2020, 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, Online Algorithm 8. A similar result was proved as Curtis and Que (2015, lemma 3.3).

Lemma 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 5(c) that $\alpha_k \ge 0$ for all $k \in \mathbb{N}$. In order to derive a contradiction, suppose that there exists an index $k_\alpha \in \mathbb{N}$ such that $\alpha_k = 0$ for all $k \in \mathbb{N}$ with $k \ge k_\alpha$. By Lemma 5(c), this means that $p_k \le p-1$ for all $k \ge k_\alpha$. However, with $\alpha_k = 0$, one finds that (23) does not hold, which by Lemma 5(e) implies that $p_{k+1} \ge \min\{p_k + 1, p\}$. This implies the existence of some $k \ge k_\alpha$ such that $p_k \ge p$, which by Lemma 5(c) implies that $\alpha_k > 0$, a contradiction of the definition of the index 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, for example, Curtis and Que (2015, lemma 3.4).

Lemma 7. *If $k+1 \in \mathcal{K}$, then*

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

Proof. Suppose $k+1 \in \mathcal{K}$, which implies that $k \in \mathcal{K}$. Lemma 5(a and b) implies 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 5(d) shows that x_{k+1} satisfies (22a) and (22c). The triangle inequality and (22c) implies

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

Hence, along with (22a), one finds that

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

as desired. □

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

$$\mathcal{G}(\overline{x}, \epsilon_k) := \mathrm{cl}\,\mathrm{conv}\nabla \mathrm{f}(\mathbb{B}(\overline{x}, \epsilon_\mathrm{k}) \cap \mathcal{D}). \tag{25}$$

Along with this, for any $\omega \in (0, \infty)$, the subset of the Cartesian product of ϵ_k-balls about x_k is given by

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

both of which are defined with respect to each iteration number $k \in \mathbb{N}$ and a point $\overline{x} \in \mathbb{R}^n$. (In the definition of $\mathcal{T}_k(\overline{x}, \omega)$, recall that $P_{W_k}(\cdot)$ has been defined in (14).) The following lemma, which follows Kiwiel (2007, lemma 3.2(i)) and Curtis and Que (2013, lemma 4.7; 2015, lemma 3.6), shows that, if the sample set size indicator p_k is sufficiently large and x_k is sufficiently close to $\overline{x}$, then for any $\omega \in (0, \infty)$, there exists a nonempty open subset of $\mathcal{T}_k(\overline{x}, \omega)$. This will be critical in our main result, in which we need to show in certain situations that an element of this subset can be found through random sampling of points.

Lemma 8. *Let $\overline{x} \in \mathbb{R}^n$ and $\omega \in (0, \infty)$ be given. If $k \in \mathcal{K}$ and $p_k \geq n + 1$, then there exists $\zeta > 0$ such that, with $x_k \in \mathbb{B}(\overline{x}, \zeta)$, there is a nonempty open $\mathcal{T} \subseteq \mathcal{T}_k(\overline{x}, \omega)$.*

Proof. Using the metric defined by W_k, the proof follows the same argument of Kiwiel (2007, 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 Curtis and Que (2015, theorem 3.1), which we present for completeness. The new features are twofold: (1) our algorithm is even less conservative about the Hessian and inverse Hessian updates than the method in Curtis and Que (2015), so our convergence result relies on arguments about self-correcting properties of BFGS updating that we have stated in Lemma 5, which borrows from Curtis et al. (2020), and (2) our inexactness conditions and our Lemma 2, which have not appeared before for GS methods, play critical roles in the proof of the theorem.

Theorem 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 5, it follows with probability one that an infinite number of outer iterations are performed, meaning $\mathcal{K} = \mathbb{N}$. Because our desired conclusion only needs to hold with probability one, we may assume going forward that $\mathcal{K} = \mathbb{N}$. Under this assumption, our next aim is to prove that $\{\epsilon_k\} \searrow 0$ with probability one. We consider two cases.

- Case 1: Suppose that $\mathcal{K}_d := \{k \in \mathbb{N} : d_k = 0\}$ is infinite. By Lemma 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) (see Algorithm 2) imply that $\{\epsilon_k\} \searrow 0$.
- 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) (see Algorithm 2),

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

On the other hand, Assumption 1, Lemma 7, and (22a) 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 $\{x_k\} \to \overline{x}$ for some $\overline{x} \in \mathbb{R}^n$. At the same time, with (26), the bound in (27b) implies that $\{\alpha_k\} \to 0$. We claim this implies that $p_k = p$ for all sufficiently large $k \in \mathcal{K}_\alpha$, where $\mathcal{K}_\alpha$ is defined as in Lemma 6. Indeed, because $\{\alpha_k\} \to 0$, it follows by Lemma 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 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, because $x_{k+1} \leftarrow x_k$ 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 5 equals $\mathbb{N}$. Correspondingly, for a given $\chi \in (0,1)$, let $\mathcal{K}_\chi$ be the indices of iterations for which (24) holds; in particular, for $k \in \mathcal{K}_\chi$, one has from (24a) and (24b) that

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

Because (10) (see Algorithm 2) does not hold for any $k \geq k_\epsilon$, it follows that either (12) or (13) (see Algorithm 2) holds for all $k \geq k_\epsilon$. Hence, by Lemmas 3 and 4, it follows that (18) holds for all $k \geq k_\epsilon$, meaning for all $k \geq k_\epsilon$ that

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

Subcase 2a: If $\overline{x}$ is ϵ-stationary, then $\|P_{W_k}(\mathcal{G}(\overline{x}, \epsilon))\|_{W_k} = 0$ 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 8, it follows that there exists $k_\zeta \in \mathbb{N}$ with $k_\zeta \geq k_\epsilon$ such that $x_k \in \mathbb{B}(\overline{x}, \zeta)$ for all $k \geq k_\zeta$, and with (29),

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

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 because, for all such k, the set $\mathcal{X}_k$ will contain new points from $\mathbb{B}(x_k, \epsilon)$ that are generated independently whether $k \in \mathcal{K}_\chi$, 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 $\overline{x}$ is not ϵ-stationary, then it follows from Lemma 5(c) that α_k satisfies (21a) for all $k \in \mathbb{N}$. In particular, (21a) 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 (Clarke 1983, 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 $\tilde{g}_k \in \partial f(\tilde{x}_k)$ 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 8, there exists $k_\omega \geq k_\epsilon$ such that $x_k \in \mathbb{B}(\overline{x}, \min\{\zeta, \epsilon/3\})$ for $k \geq k_\omega$, and with (29),

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

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 with $\mathcal{S} = \mathcal{G}(\overline{x}, \epsilon)$, $\beta = \underline{\eta}\mu \in (0,1)$

(where this inclusion is guaranteed by Online Algorithm 4), and $W = W_k$ that, for sufficiently small $\sigma_k \in (0, \sigma)$ and $\omega \in (0, \infty)$, one finds that (28) and (34) imply

$$v^T W_k G_k y_k > \underline{\eta}\, \mu \,\|\, G_k y_k \,\|_{W_k}^2$$
$$\geq \underline{\eta} \max\{\|d_k\|_2^2 \,,\, \|G_k y_k\|_2^2\} \text{ for all } \; v \in \mathcal{G}(\overline{x}, \epsilon). \tag{35}$$

There exists $k_\sigma \geq k_\omega$ such that σ_k is sufficiently small for all $k \geq k_\sigma$ with $k \in \mathcal{K}_\chi$ because $\{\alpha_k\} \to 0$ and the construction of the algorithm implies that $\{\sigma_k\} \to 0$. Together, (33) and (35) imply that $\tilde{g}_k \notin \mathcal{G}(\overline{x}, \epsilon)$ 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, and Clarke (1983, proposition 2.1.2), it follows for all $k \geq k_\sigma$ with $k \in \mathcal{K}_\chi$ that

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

where $L_{\mathbb{B}(\overline{x}, \epsilon)} \in (0, \infty)$ is a Lipschitz constant for f over $\mathbb{B}(\overline{x}, \epsilon)$. 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 $k \in \mathcal{K}_\chi$; that is, $\gamma^{-1}\alpha_k\,\|d_k\|_2 \leq \epsilon/3$ for all sufficiently large $k \in \mathcal{K}_\chi$. Along with the fact that $x_k \in \mathbb{B}(\overline{x}, \min\{\zeta, \epsilon/3\})$ implies $\|x_k - \overline{x}\|_2 \leq \epsilon/3$, it follows that $\tilde{x}_k \in \mathbb{B}(\overline{x}, 2\min\{\zeta, \epsilon/3\}/3)$, and hence, $\tilde{g}_k \in \mathcal{G}(\overline{x}, \epsilon)$ for all sufficiently large $k \in \mathbb{N}$. Overall, because $\tilde{g}_k \notin \mathcal{G}(\overline{x}, \epsilon)$ whenever $k \geq k_\sigma$, $k \in \mathcal{K}_\chi$, and $\mathcal{X}_k \in \mathcal{T}$, yet $\tilde{g}_k \in \mathcal{G}(\overline{x}, \epsilon)$ 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 because $|\mathcal{K}_\chi| = \infty$ 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) (see Algorithm 2), there exists an infinite index set $\mathcal{K}_\epsilon := \{k \in \mathbb{N} : \epsilon_{k+1} \leftarrow \psi\epsilon_k\}$, where

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

The same argument as in Curtis and Que (2013, theorem 4.2, case 2), which borrows from Kiwiel (2007, 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 $\epsilon_0 \in (0, \infty)$ for all $k \in \mathcal{K}$. Similar results have appeared in the literature to prove a similar property of other GS methods; see, for example, Kiwiel (2007, theorem 3.5).

Corollary 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 $0 \in \mathcal{G}(x_k, \epsilon_k)$ or every cluster point of the iterate sequence $\{x_k\}$ is ϵ_0-stationary for f.*

Proof. As in the proof of Theorem 1, if Algorithm 1 terminates finitely with a stationary point for f, then there is nothing left to prove. Otherwise, by Lemma 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 $0 \in \mathcal{G}(x_k, \epsilon_k)$, then there is nothing left to prove. Otherwise, following the arguments in the proof of Theorem 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 $\eth_{\epsilon_0} f$ 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 because 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 3. 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 $x_{k+1} = x_k$, because

$\alpha_k = 0$, 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}.

Algorithm 3 (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 5–19 of Algorithm 1 …]
  Set $(\mathcal{X}_{k+1}, p_{k+1})$ by Online Algorithm 8 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 $G_{k+1}^{\text{agg}} \leftarrow [\, \nabla f(x_{k+1}) \;\; G_k y_k \;\; [\nabla f(x)]_{x \in \mathcal{X}_{k+1} \setminus (x_{k+1} \cup \mathcal{X}_k)} \,]$.
  end if
end for

Lemma 9. *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

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

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

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

which proves the desired result. □

Theorem 2. *Suppose $\psi \in (0,1)$. Algorithm 3 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 1 holds regardless of whether $G_k = G_k^{\text{agg}}$ or $G_k = G_k^{\text{full}}$ because G_k has $\nabla f(x_k)$ as its first column in either case. The results of Lemmas 3 and 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

$$\begin{bmatrix} [y_k]_1 \\ y_{k-1}[y_k]_2 \\ [y_k]_{>2} \end{bmatrix} \tag{36}$$

as the dual vector as shown by Lemma 9. The arguments of Lemmas 5–8 and Theorem 1 now follow in the same manner as in Section 2 using G_k^{full} in place of G_k and y_k or (36) in place of the dual vector for all $k \in \mathcal{K}$. Crucial in 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 2. *Suppose $\psi = 1$. Algorithm 3 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 $0 \in \mathcal{G}(x_k, \epsilon_k)$ or every cluster point of the iterate sequence $\{x_k\}$ is ϵ_0-stationary for f.*

Proof. The proof follows from that of Theorem 2 in the same manner as the proof of Corollary 1 follows from that of Theorem 1. □

4. Numerical Experiments

In this section, we present the results of numerical experiments with implementations of our proposed algorithms. The main purpose of these experiments is to show that the introduction of inexactness and gradient aggregation can reduce the computational expense of an adaptive GS algorithm consistently and often substantially. As a sanity check, we also provide a comparison between our implementation of Algorithm 3 and a state-of-the-art code. All experiments were run on a Macbook Air with a 2.2 GHz Dual-Core Intel Core i7 processor running macOS 11.4.

We implemented our algorithms in the C++ software package NonOpt (Curtis 2021). 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. 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 Online Algorithm 6 always terminates in the first iteration; hence, the parameter $\overline{\ell}$ is not used. The initial point $x_0 \in \mathbb{R}^n$ in each run of the algorithm was chosen in a problem-dependent manner; see our discussions in the following sections about the initial points used.

NonOpt contains a dual active-set QP solver that we used for solving the QP subproblems. To reduce CPU time, during the solve of a given QP, the termination conditions (10)–(13) (see Algorithm 2) 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 solver iterations.

In our implementations, the outer iteration sequence terminates if

$$\max\{\|G_k y_k\|_\infty,\ \|W_k G_k y_k\|_\infty, \epsilon_k\} \leq 10^{-5} \tag{37}$$

or once an objective function value tolerance or CPU time limit is reached. These latter criteria are discussed in further detail in the subsequent sections.

We consider the performance of three implementations, to which we refer as follows:

• `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` except that,

Table 1. User-Specified Parameters for Our Implemented Algorithms and Subroutines

Parameter(s)	Range	Values	Description
ν	$(0, \infty)$	1	Stationarity measure tolerance
$\underline{\alpha} \leq \overline{\alpha}$	$(0, \infty)$	$10^{-20} \leq 100$	Step size thresholds
α_{init}	$(0, \infty)$	1	Initial step size
ρ	$(0, 1)$	0.01	Inexactness threshold bound
κ	$(0, 1)$	0.0001	Inexactness threshold
ψ	$(0, 1)$	0.1	Sampling radius reduction factor
ι	$(0, 1)$	0.5	Inexactness parameter reduction factor
$\underline{\eta} < \overline{\eta}$	$(0, 1)$	$10^{-10} < 0.9$	Armijo–Wolfe line search parameters
$\underline{p}$	$[n+1, \infty) \cap \mathbb{N}$	$10n$	Sample set size threshold
σ	$(0, \infty)$	10	Inexactness threshold reset value
γ	$(0, 1)$	0.5	Step size modification factor
$\underline{\phi} < 1 < \overline{\phi}$	$(0, \infty)$	$10^{-20} < 1 < 10^8$	BFGS updating thresholds
$\overline{\xi}$	$(0, \infty)$	10^{-20}	Curvature threshold
$\overline{p}$	$\mathbb{N}$	$0.01n$	Size of addition to sample set
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

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^{-10}.

- `GS-inexact`: An implementation of Algorithm 1.
- `GS-inexact-agg`: An implementation of Algorithm 3.

4.1. Randomly Generated Test Problems

Our algorithms are designed to minimize objectives that may be nonconvex and/or nonsmooth. However, in order to conduct a controlled comparison between the aforementioned implemented algorithms, our main experiment involves randomly generated convex test problems of the form

$$\min_{x\in\mathbb{R}^n} g^T x + \frac{1}{2}x^T H x + \max\{Ax + b\}, \tag{38}$$

where $g \in \mathbb{R}^n$, $H \in \mathbb{R}^{n\times n}$ is symmetric and positive definite, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and the max is taken element-wise. (By employing convex as opposed to nonconvex test problems, we can be sure that the results of our experiments are not skewed by two algorithms converging to different local minimizers and other related circumstances.) The problems were constructed such that the unique global minimizer is always $x_* = 0$, the global minimum is always $f(x_*) = 0$, and the number of elements of the vector $Ax_* + b = b$ yielding the max, call it $m_{\mathcal{A}}$, is always predetermined. Specifically, each problem was constructed in the following manner. Observe that $x_* \in \mathbb{R}^n$ solves (38) if and only if there exists $(y_*, z_*) \in \mathbb{R}^m \times \mathbb{R}$ such that

$$\begin{aligned} g + Hx_* + A^T y_* &= 0 \\ 1 - y_*^T \mathbb{1} &= 0 \\ Ax_* + b - z_* \mathbb{1} &\le 0 \\ y_* &\ge 0 \\ y_*^T(Ax_* + b - z_* \mathbb{1}) &= 0, \end{aligned} \tag{39}$$

and, when $x_* = 0$, one finds that $f(x_*) = 0$ with the number of elements of $Ax_* + b = b$ yielding the max being $m_{\mathcal{A}}$ if and only if one finds that $m_{\mathcal{A}}$ elements of b are equal to zero with the rest being strictly negative. Hence, in the construction of each problem, the first $m_{\mathcal{A}}$ elements of b were set to zero and each of the remaining elements was set to the negative of a random value drawn from a χ^2_1 distribution. The matrix A was then set randomly with each element being drawn from a standard normal distribution. Next, the first $m_{\mathcal{A}}$ elements of y_* were set randomly with each element drawn from a uniform distribution over $[0, 1]$. The remaining elements were set to zero and then the entire vector was normalized so that $y_*^T \mathbb{1} = 1$. Next, $g \leftarrow A^T y_*$. Finally, $H \leftarrow \overline{H}^T \overline{H}$, where each element of $\overline{H}$ was drawn from a standard normal distribution. These steps were verified to ensure that $H \succ 0$ and $(0, y_*, 0)$ satisfies (39) with the desired value of $m_{\mathcal{A}}$.

When solving potentially nonconvex and/or nonsmooth optimization problems, termination conditions can be sensitive in practice; for example, one can find that the termination condition (37) may be satisfied relatively early for some problems, whereas for other problems, the magnification of small numerical errors can cause (37) to take longer to be satisfied. Hence, we added a condition for these experiments that terminates an algorithm whenever the objective value is less than a prescribed threshold of 10^{-3}. This is reasonable in these experiments because $f(x_*) = 0$ for all problems.

For the purposes of these experiments, 15 problems were generated; with $n = 1{,}000$ and $m = 500$, five problems were generated for each of the values $m_{\mathcal{A}} \in \{125, 250, 375\}$. In this manner, we provide results for a range of dimensions of the "$\mathcal{U}$-space" and "$\mathcal{V}$-space" at the minimizer; see, for example, Liu and Sagastizábal (2020). For each problem, each of the three implemented algorithms were run from the same randomly generated starting point; in particular, each element for the initial point was drawn from a standard normal distribution. Because GS methods are randomized, we ran each algorithm 10 times for each problem and provide averages of performance measures.

Results for `GS-exact`, `GS-inexact`, and `GS-inexact-agg` are provided in Tables 2–4, respectively. Averaged over the 10 runs for each algorithm and problem, we provide the required number of iterations (`iters`), required total number of QP solver iterations (`QP-iters`), required number of objective function evaluations (`funcs`), required number of objective gradient evaluations (`grads`), and final objective value (f). Because the total computational effort is roughly proportional to the total number of QP iterations, for `GS-inexact`, we provide the relative change in the required total number of QP iterations as compared with `GS-exact`, and for `GS-inexact-agg`, we provide the relative change as compared with `GS-inexact`. (This is a rough proxy for computational effort

Table 2. Results for `GS-exact` Averaged over 10 Runs

n	m	$m_{\mathcal{A}}$	iters	QP-iters	funcs	grads	f
1,000	500	125	5,059	22,126	39,484	37,939	+9.970765e−04
1,000	500	125	5,185	23,701	40,681	39,243	+9.969306e−04
1,000	500	125	5,559	25,350	44,227	40,406	+9.961246e−04
1,000	500	125	4,513	21,721	35,696	34,544	+9.986064e−04
1,000	500	125	6,660	31,010	53,104	50,547	+9.952435e−04
1,000	500	250	4,660	23,396	36,201	38,673	+9.970373e−04
1,000	500	250	4,700	24,705	36,514	39,828	+9.949415e−04
1,000	500	250	4,841	24,237	37,157	40,845	+9.967083e−04
1,000	500	250	4,324	23,389	33,250	38,513	+9.967001e−04
1,000	500	250	4,910	25,148	37,763	40,083	+9.975146e−04
1,000	500	375	3,958	22,479	29,712	33,767	+9.936715e−04
1,000	500	375	3,738	21,626	28,174	32,966	+9.967347e−04
1,000	500	375	5,540	31,959	42,412	48,309	+9.970079e−04
1,000	500	375	5,291	29,483	40,416	46,418	+9.970639e−04
1,000	500	375	3,956	25,199	30,153	36,355	+9.953953e−04

because the cost for each QP solver iteration can differ depending on the number of nonzero variables in the dual solution estimate. That said, we found it to be the best measure for comparison as opposed to CPU time, which can vary despite the algorithm being run with the same initial conditions, random number generator seeds, and so on.) In these statistics, a negative percentage indicates that `GS-inexact` (respectively, `GS-inexact-agg`) required fewer total QP solver iterations than `GS-exact` (respectively, `GS-inexact`); for example, a statistic of $-z\%$ indicates that the algorithm lowered the required total number of QP solver iterations by $z\%$.

Observe that between the termination condition (37) and the condition that the algorithm terminates if the objective value fell below 10^{-3}, one finds that the solutions obtained by all algorithms on all problems were comparable in quality with final objective values on the order of 10^{-3}. That said, one finds in the results in Tables 2–4 that inexactness and gradient aggregation reduce the total number of QP solver iterations consistently and often substantially. Interestingly, one also finds in many cases that `GS-inexact` and `GS-inexact-agg` also require fewer outer iterations. This was not necessarily expected and might not represent behavior that one should anticipate in general. That said, one explanation for this behavior is that requiring exact subproblem solutions may tend to produce shorter search directions, whereas by allowing inexactness in the subproblem solutions, the algorithm is able to take longer steps in each iteration. In any case, because of the reduced number of QP solver iterations required per outer iteration, one may expect a reduction in total computational effort for `GS-inexact` and `GS-inexact-agg` even if these algorithms were to require the same number, or even more, of outer iterations than `GS-exact`.

Table 3. Results for `GS-inexact` Averaged over 10 Runs

n	m	$m_{\mathcal{A}}$	iters	QP-iters	funcs	grads	f	change in QP-iters, %
1,000	500	125	5,598	19,170	45,902	44,969	+9.977302e−04	−13.36123
1,000	500	125	5,436	19,003	44,530	44,422	+1.023123e−03	−19.82204
1,000	500	125	6,015	19,754	49,600	45,475	+9.977170e−04	−22.07298
1,000	500	125	5,014	15,902	41,134	37,434	+9.984253e−04	−26.78950
1,000	500	125	7,206	22,222	59,453	52,540	+9.990196e−04	−28.33939
1,000	500	250	5,197	20,830	42,745	45,009	+9.978813e−04	−10.96660
1,000	500	250	5,150	20,822	42,436	44,944	+9.978613e−04	−15.71849
1,000	500	250	5,000	19,857	40,225	42,308	+9.988319e−04	−18.07192
1,000	500	250	4,711	19,706	38,700	42,438	+9.941688e−04	−15.74402
1,000	500	250	5,653	21,889	46,103	47,035	+9.976889e−04	−12.95639
1,000	500	375	4,316	19,615	34,423	36,914	+9.952806e−04	−12.74017
1,000	500	375	4,327	20,723	34,861	37,765	+9.973864e−04	−4.175375
1,000	500	375	6,316	31,207	52,132	58,984	+9.962637e−04	−2.353611
1,000	500	375	5,520	24,971	44,869	48,141	+9.981637e−04	−15.30415
1,000	500	375	4,517	21,625	36,977	41,138	+9.969319e−04	−14.18299

Note. The final column indicates the relative change in `QP-iters` compared with `GS-exact`.

Table 4. Results for `GS-inexact-agg` Averaged over 10 Runs

n	m	$m_{\mathcal{A}}$	iters	QP-iters	funcs	grads	f	change in QP-iters, %
1,000	500	125	4,769	14,484	36,291	34,707	+9.981286e−04	−24.44093
1,000	500	125	4,905	14,965	37,272	35,982	+9.988238e−04	−21.24999
1,000	500	125	5,528	17,036	42,753	39,277	+9.969082e−04	−13.75838
1,000	500	125	4,600	13,978	35,343	32,215	+1.002537e−03	−12.09518
1,000	500	125	6,486	20,017	50,219	46,658	+1.624848e−03	−9.921205
1,000	500	250	4,581	15,239	34,511	35,282	+1.115497e−03	−26.84115
1,000	500	250	4,214	13,802	31,259	31,523	+2.096993e−03	−33.71098
1,000	500	250	4,407	13,665	31,907	31,061	+9.981417e−04	−31.18384
1,000	500	250	4,253	13,844	31,533	32,841	+9.984159e−04	−29.74861
1,000	500	250	4,950	16,218	36,826	37,262	+9.974769e−04	−25.90921
1,000	500	375	4,026	14,203	29,311	30,293	+9.966796e−04	−27.58802
1,000	500	375	3,754	12,767	27,003	27,208	+9.967023e−04	−38.39113
1,000	500	375	5,684	21,193	41,911	44,935	+9.982174e−04	−32.08651
1,000	500	375	5,133	17,739	37,718	38,859	+1.066837e−03	−28.96165
1,000	500	375	4,288	15,233	31,248	32,955	+9.956476e−04	−29.55487

Note. The final column indicates the relative change in `QP-iters` compared with `GS-inexact`.

4.2. Test Set Problems

To demonstrate that our implementation can be competitive with a state-of-the-art solver in terms of obtaining high-quality solutions within a reasonable CPU time limit, we performed an experiment to compare the performance of the state-of-the-art code `LMBM` (Karmitsa accessed 2021) and `GS-inexact-agg`. The experiments with `GS-inexact-agg` in the previous section were performed with full BFGS approximations, but the experiments in this section were performed with a limited-memory BFGS strategy with a history of 50 so that the algorithm would be more similar to `LMBM`, which uses limited memory approximations (with a history of seven). We also decreased the size of additions to the sample set to $\overline{p} \leftarrow \lceil \max\{1, 10^{-4}n\} \rceil$ (see Table 1) to increase speed for the CPU-time-limited experiments for solving the larger scale problems that are considered here.

We chose a set of 20 test problems for which `LMBM` has been tuned, some of which are convex and some of which are nonconvex. The first 10 problems come from Haarala et al. (2004) and the second 10 come from Lukšan et al. (2002). (`LMBM` comes with implementations of the first 10 problems; for the remaining test problems, we obtained Fortran implementations from Lukšan, accessed in 2021.) All of the problems are scalable in the sense that they can be defined for any dimension $n \in \mathbb{N}$. For any $n \in \mathbb{N}$, the aforementioned sources describe an initial point $x_0 \in \mathbb{R}^n$ for each problem. We followed these rules for determining the initial points used in our experiments.

The dimension of each problem was chosen using the following procedure to ensure that the CPU time required by `LMBM` to reach a good solution was nontrivial for all problems. Starting with n = 10,000, the dimension was decreased by 100 or increased by 1,000 in an iterative manner until the average CPU time required by `LMBM` (using its default termination conditions) over 10 runs was at least one second and at most 10 seconds. This led to the problem sizes shown in Table 5.

`LMBM` and `GS-inexact-agg` have many differences. For example, `LMBM` employs a bundle method and `GS-inexact-agg` employs a GS method. The termination criteria of the two codes are also very different; for example, besides observing termination criteria related to detecting stationarity, `LMBM` may terminate for various reasons related to the iterate and/or objective value not changing significantly between iterations. Hence, in order to offer a fair and illustrative comparison, after the problem sizes were determined using the aforementioned procedure, we ran both `LMBM` and `GS-inexact-agg` with their main termination conditions effectively disabled and a CPU time limit of 20 seconds. (Specifically, for `LMBM`, we set `RPAR(1)` = `RPAR(2)` = `RPAR(4)` = `RPAR(5)` = 10^{-20} and `RPAR(3)` = -10^{60}, and for `GS-inexact-agg`, we set the right-hand side of (37) to 10^{-20}. This means that `LMBM` was allowed to run at least as long as it would have run with its default termination conditions and that both codes were given the same CPU time limit.)

The results obtained by the codes are shown in Table 5. Both codes report the number of iterations (`iters`) and function evaluations (`funcs`) that were performed as well as the final objective value (f) obtained. Because the algorithms have various differences, it is not necessarily informative to compare the number of iterations or function evaluations required by the two methods. In addition, some of the test problems are nonconvex, meaning that it is possible in some instances for a solver to be attracted to a stationary point that is not a local/global minimizer. Nonetheless, one can compare final objective values to get a general sense of the performance of the

Table 5. Results for `LMBM` and `GS-inexact-agg` Within 20-Second CPU Time Limit

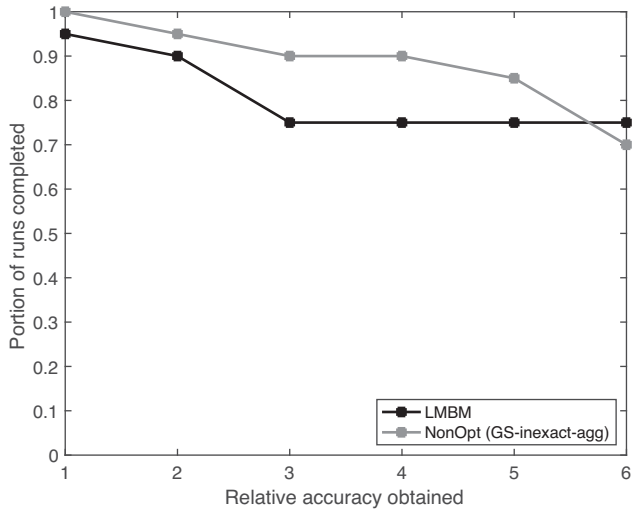
		LMBM			GS-inexact-agg		
name	n	iters	funcs	f	iters	funcs	f
MaxQ	1,500	48,234	49,913	+1.02076e−10	24,706	55,256	**+2.101717e−19**
MxHilb	2,000	510	1,668	+1.56265e−02	296	1,300	**+1.101971e−07**
Chained_LQ	11,000	688	7,149	−1.55549e+04	183	1,381	−1.555488e+04
Chained_CB3_1	10,000	582	5,473	**+1.99985e+04**	421	3,045	+2.000014e+04
Chained_CB3_2	42,000	205	417	+8.39980e+04	147	1,131	+8.399800e+04
ActiveFaces	10,000	8,481	8,604	**+0.00000e+00**	305	4,765	+1.869505e−10
Brown_Function_2	10,000	4,872	8,796	**+2.47374e−09**	285	2,651	+5.175338e−07
Chained_Mifflin_2	10,000	952	8,873	−7.07043e+03	143	929	−7.070059e+03
Chained_Crescent_1	21,000	337	1,844	**+1.90958e−10**	208	1,504	+2.205846e−10
Chained_Crescent_2	10,000	2,926	37,537	**+6.49194e−04**	157	1,097	+4.584530e−02
Test29_2	140,000	1,104	1,109	+9.99847e−01	81	408	**+9.992071e−01**
Test29_5	600	1,677	9,104	+4.77363e−06	1,701	10,803	**+1.651470e−07**
Test29_6	130,000	227	6,223	+2.00000e+00	43	361	+2.002375e+00
Test29_11	10,000	598	5,135	+1.20472e+05	265	1,888	+1.204895e+05
Test29_13	600	2,377	17,148	+3.39884e+02	2,103	17,411	+3.392142e+02
Test29_17	10,000	8,795	11,208	+2.12820e−03	342	3,071	**+3.940380e−05**
Test29_19	25,000	991	33,977	+1.00000e+00	91	640	+1.000004e+00
Test29_20	10,000	640	12,197	+5.00001e−01	181	1,400	+5.000012e−01
Test29_22	109,000	76	3,041	**+1.68293e−10**	50	646	+4.753889e−07
Test29_24	12,000	1,050	12,544	+3.91284e−02	254	2,485	**+1.302162e−03**

Note. Objective values in bold indicate that the final objective value is better than other alternative within the first three significant digits.

solvers with respect to which one finds that the results are generally comparable. For emphasis, we mark in bold text the final objective values that were better than the alternative within the first three significant digits. One finds that, in the cases when one final objective value was significantly better than the other using this threshold, each code yielded a lower objective value for six problems.

The results in Table 5 can be visualized in the form of an accuracy profile (Beiranvand et al. 2017), which for a given relative accuracy (with respect to the *initial* objective value), shows the number of problems in the test set for which each solver is able to attain the desired relative accuracy. To construct a profile, for each problem, we set the "optimal" value as the best objective value obtained by one of the solvers, and because `LMBM` only reports the final objective value to an accuracy of six digits, we used a maximal improvement value of six. The profile can be seen in Figure 1. It shows that `GS-inexact-agg` is able to "solve" more problems up to five digits of accuracy although `LMBM` is able to solve one more problem to six digits of accuracy. Overall, the profile shows that the solvers are competitive in terms of the quality of solutions obtained on this test set.

Figure 1. Accuracy Profile Corresponding to the Results in Table 5



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 is that it can allow *inexactness* in the subproblem solutions while maintaining convergence guarantees, 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 computational effort.

References

Apkarian P, Noll D, Prot O (2008) A trust region spectral bundle method for nonconvex eigenvalue optimization. *SIAM J. Optim.* 19(1): 281–306.

Beiranvand V, Hare W, Lucet Y (2017) Best practices for comparing optimization algorithms. *Optim. Engrg.* 18:815–848.

Bertsekas DP (2009) *Convex Optimization Theory* (Athena Scientific, Nashua, NH).

Burke JV, Lewis AS, Overton ML (2002) Approximating subdifferentials by random sampling of gradients. *Math. Oper. Res.* 27(3):567–584.

Burke JV, Lewis AS, Overton ML (2005) A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM J. Optim.* 15(3): 751–779.

Burke JV, Curtis FE, Lewis AS, Overton ML, Simões LEA (2020) Gradient sampling methods for nonsmooth optimization. Bagirov AM, Gaudioso M, Karmitsa N, Mäkelä MM, Taheri S, eds. *Numerical Nonsmooth Optimization* (Springer, Cham), 201–225.

Byrd RH, Nocedal J (1989) A tool for the analysis of quasi-Newton methods with application to unconstrained minimization. *SIAM J. Numerical Anal.* 26(3):727–739.

Clarke FH (1983) *Optimization and Nonsmooth Analysis.* Canadian Mathematical Society Series of Monographs and Advanced Texts (John Wiley & Sons, New York).

Curtis FE (2021) NonOpt. Accessed January 13, 2022, https://coral.ise.lehigh.edu/frankecurtis/nonopt/.

Curtis FE, Overton ML (2012) A sequential quadratic programming algorithm for nonconvex, nonsmooth constrained optimization. *SIAM J. Optim.* 22(2):474–500.

Curtis FE, Que X (2013) An adaptive gradient sampling algorithm for nonsmooth optimization. *Optim. Methods Software* 28(6):1302–1324.

Curtis FE, Que X (2015) A quasi-Newton algorithm for nonconvex, nonsmooth optimization with global convergence guarantees. *Math. Programming Comput.* 7(4):399–428.

Curtis FE, Robinson DP, Zhou B (2020) A self-correcting variable-metric algorithm framework for nonsmooth optimization. *IMA J. Numerical Anal.* 40(2):1154–1187.

de Oliveira W, Solodov M (2016) A doubly stabilized bundle method for nonsmooth convex optimization. *Math. Programming* 156:125–159.

de Oliveira W, Sagastizábal C, Lemaréchal C (2014) Convex proximal bundle methods in depth: A unified analysis for inexact oracles. *Math. Programming* 148:241–277.

Goldstein AA (1977) Optimization of Lipschitz continuous functions. *Math. Programming* 13(1):14–22.

Haarala N, Miettinen K, Mäkelä MM (2004) New limited memory bundle method for large-scale nonsmooth optimization. *Optim. Methods Software* 19(6):673–692.

Haarala N, Miettinen K, Mäkelä MM (2007) Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Math. Programming* 109(1):181–205.

Hare W, Sagastizábal C (2010) A redistributed proximal bundle method for nonconvex optimization. *SIAM J. Optim.* 20(5):2442–2473.

Hare W, Sagastizábal C, Solodov M (2016) A proximal bundle method for nonsmooth nonconvex functions with inexact information. *Comput. Optim. Appl.* 63(1):1–28.

Helou ES, Santos SA, Simões LEA (2017) On the local convergence analysis of the gradient sampling method for finite max-functions. *J. Optim. Theory Appl.* 175(1):137–157.

Hiriart-Urruty JB, Lemaréchal C (1993) *Convex Analysis and Minimization Algorithms II.* A Series of Comprehensive Studies in Mathematics (Springer-Verlag, New York).

Hosseini S, Uschmajew A (2017) A Riemannian gradient sampling algorithm for nonsmooth optimization on manifolds. *SIAM J. Optim.* 27(1): 173–189.

Karmitsa N (2021) LMBM. Accessed January 13, 2022, http://napsu.karmitsa.fi/lmbm.

Kiwiel KC (1985a) A linearization algorithm for nonsmooth minimization. *Math. Oper. Res.* 10(2):185–194.

Kiwiel KC (1985b) *Methods of Descent for Nondifferentiable Optimization.* Lecture Notes in Mathematics (Springer-Verlag, New York).

Kiwiel KC (1996) Restricted step and Levenberg-Marquardt techniques in proximal bundle methods for nonconvex nondifferentiable optimization. *SIAM J. Optim.* 6(1):227–249.

Kiwiel KC (2007) Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM J. Optim.* 18(2):379–388.

Kiwiel KC (2010) A nonderivative version of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM J. Optim.* 20(4): 1983–1994.

Lemaréchal C, Nemirovskii A, Nesterov Y (1995) New variants of bundle methods. *Math. Programming* 69(1):111–147.

Liu S, Sagastizábal C (2020) Gradient sampling methods for nonsmooth optimization. *Beyond First Order: VU-Decomposition Methods* (Springer International Publishing, Cham, Switzerland), 297–329.

Lukšan L (2021) Test problems in Fortran. Accessed January 13, 2022, http://www.cs.cas.cz/ luksan/test.html.

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

Lukšan L, Tuma M, Šiška M, Vlček J, Ramešová N (2002) UFO 2002: Interactive system for universal functional optimization. Technical Report 883, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic.

Maleknia M, Shamsi M (2020) A gradient sampling method based on ideal direction for solving nonsmooth optimization problems. *J. Optim. Theory Appl.* 187(3):181–204.

Mifflin R (1982) A modification and an extension of Lemarechal's algorithm for nonsmooth minimization. Sorensen DC, Wets RJB, eds. *Nondifferential and Variational Techniques in Optimization* (Springer, Berlin, Heidelberg), 77–90.

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

Nocedal J, Wright S (2006) *Numerical Optimization.* Springer Series in Operations Research, 2nd ed. (Springer, New York).

Ruszczynski A (2006) *Nonlinear Optimization* (Princeton University Press, Princeton, NJ).

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

Tang CM, Liu S, Jian JB, Li JL (2014) A feasible SQP-GS algorithm for nonconvex, nonsmooth constrained optimization. *Numerical Algorithms* 65(1):1–22.

van Ackooij W, Frangioni A (2018) Incremental bundle methods using upper models. *SIAM J. Optim.* 28(1):379–410.