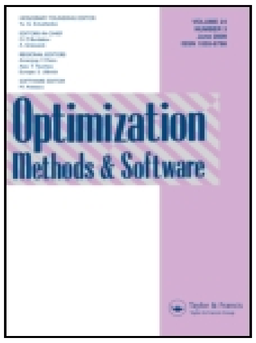


Optimization Methods and Software

ISSN: 1055-6788 (Print) 1029-4937 (Online) Journal homepage: http://www.tandfonline.com/loi/goms20

An adaptive gradient sampling algorithm for non-smooth optimization

Frank E. Curtis & Xiaocun Que

Optimization Methods & Software, 2013
Vol. 28, No. 6, 1302–1324, http://dx.doi.org/10.1080/10556788.2012.714781

An adaptive gradient sampling algorithm for non-smooth optimization

Frank E. Curtis* and Xiaocun Que

Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA

(*Received 7 October 2011; final version received 17 July 2012*)

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

Keywords: unconstrained optimization; non-smooth optimization; non-convex optimization; gradient sampling; line search methods; quadratic optimization; warm-starting

AMS Subject Classifications: 49M05; 65K05; 65K10; 90C26; 90C30

1. Introduction

The gradient sampling (GS) algorithm, introduced and analysed by Burke *et al.* [3,4], is a method for minimizing an objective function $f : \mathbb{R}^n \to \mathbb{R}$ that is locally Lipschitz and continuously differentiable in an open dense subset $\mathcal{D}$ of $\mathbb{R}^n$. The approach is widely applicable and robust [5,6,24], and it is intuitively appealing in that theoretical convergence guarantees hold with probability one without requiring algorithmic modifications to handle non-convexity.

The theoretical foundations for GS, as well as various extensions, are developing rapidly. Stronger theoretical results than in [4] for both the original algorithm and for various extensions were provided in [22], an extension of the ideas for solving constrained problems was presented in [8], and a variant using only gradient estimates derived via function evaluations appeared in [23]. Continued developments along these lines may allow GS techniques to one day be competitive with bundle methods [18,20] in terms of theoretical might and practical performance.

The main goal of this paper is to address three practical limitations of GS as it is presented in [4,22]. Consider the following remarks.

*Corresponding author. Email: frank.e.curtis@gmail.com

(1) GS produces approximate ϵ-steepest descent directions by evaluating the gradient of f at $n + 1$ (or more) randomly generated points during each iteration. This results in a high computational cost that is especially detrimental when search directions turn out to be unproductive.
(2) Each descent direction produced by GS is obtained by the solution of a quadratic optimization (QO) subproblem. As the subproblem data are computed afresh for every iteration, the computational effort required to solve each of these subproblems can be significant for large-scale problems.
(3) GS may behave, at best, as a steepest descent method. The use of second-order information of the problem functions may be useful, but it is not clear how to incorporate this information effectively in non-smooth regions.

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

We address remark (3) by proposing two novel strategies for updating approximations of second-order terms. The first strategy is similar to a limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) update typical in smooth optimization [27]. Our method is unique, however, in that we incorporate gradient information from sample points instead of that solely at algorithm iterates. We also control the updates so that bounds on the Hessian approximations required for our convergence analysis are obtained. The second strategy we propose – intended solely for non-convex problems – is entirely novel as far as we are aware. It also involves the incorporation of function information at sample points, but is based on the desire to produce model functions that overestimate the true objective f. Bounds required for our convergence analysis are also proved for this latter strategy. Our numerical experiments in Section 5 illustrate that our Hessian approximation strategies further enhance the algorithm's ability to progress towards a solution within a given amount of computational effort.

The paper is organized as follows. A description of our adaptive gradient sampling (AGS) algorithm is presented in Section 2. Our updating strategies for approximating second-order information are presented and analysed in Section 3. Global convergence of a generic AGS algorithm is analysed in Section 4. Numerical experiments comparing implementations of GS and variants of AGS on a large test set are presented in Section 5. This implementation involves a specialized QO solver that has been implemented by enhancing the method proposed in [21]; the details of this solver are described in Appendix. Finally, concluding remarks are provided in Section 6.

The analysis in this paper builds on that of Kiwiel in [22]. It should also be noted that ideas of 'incremental sampling' and 'bundling past information' were briefly mentioned by Kiwiel [23]. However, our methods are unique from those appearing in these papers as adaptive sampling was not considered in [22], exact gradient information was not used in [23], and our algorithm involves Hessian approximations that were not considered in either article. Still, in addition to the original work by Burke *et al.* [4], it is clear that the works of Kiwiel have been inspirational for the work in this paper, not to mention the QO algorithm from [21] that has found a new area of applicability in the context of AGS. Finally, we mention that the idea of sampling function information about a given point to approximate the subdifferential has been around for decades [15].

2. Algorithm description

Consider the unconstrained problem

$$\min_x f(x), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz and continuously differentiable in an open dense subset $\mathcal{D}$ of $\mathbb{R}^n$. Letting $\operatorname{cl}\operatorname{conv} S$ denote the closure of the convex hull of a set $S \subseteq \mathbb{R}^n$ and defining the multifunction $\mathbb{G}_\epsilon(x) := \operatorname{cl}\operatorname{conv} \nabla f(\mathbb{B}_\epsilon(x) \cap \mathcal{D})$ where $\mathbb{B}_\epsilon(x) := \{\bar{x} : \|\bar{x} - x\| \le \epsilon\}$ is the Euclidean ϵ-ball about x, we have the following representation of the Clarke subdifferential [7] of f at x:

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

Similarly, the Clarke ϵ-subdifferential [12] is given by

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

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

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

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

$$G_k := [g_{k,0} \quad \cdots \quad g_{k,p_k}] \tag{2}$$

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

$$\begin{aligned} \min_{z,d} \quad & z + \tfrac{1}{2} d^{\mathrm{T}} H_k d \\ \text{s.t.} \quad & f(x_k)e + G_k^{\mathrm{T}} d \le ze. \end{aligned} \tag{3}$$

Here, and throughout the paper, e denotes a vector of ones whose length is determined by the context. Alternatively, one may solve the dual of (3), namely

$$\begin{aligned} \max_{\pi} \quad & -\tfrac{1}{2} \pi^{\mathrm{T}} G_k^{\mathrm{T}} W_k G_k \pi \\ \text{s.t.} \quad & e^{\mathrm{T}} \pi = 1, \quad \pi \ge 0, \end{aligned} \tag{4}$$

where $W_k := H_k^{-1} \succ 0$. The solution (z_k, d_k, π_k) of (3)–(4) has $d_k = -W_k G_k \pi_k$.

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

$$f(x_k + \alpha_k d_k) \le f(x_k) - \eta \alpha_k d_k^{\mathrm{T}} H_k d_k. \tag{5}$$

We set $x_{k+1} \leftarrow x_k + \alpha_k d_k$ for α_k chosen to satisfy (5), but in order to ensure that all iterates remain within the set $\mathcal{D}$, it may be necessary to perturb such an x_{k+1}; in such cases, we make use of the perturbed line search conditions

$$f(x_{k+1}) \le f(x_k) - \eta\alpha_k d_k^{\mathrm{T}} H_k d_k \tag{6a}$$

and

$$\|x_k + \alpha_k d_k - x_{k+1}\| \le \min\{\alpha_k, \epsilon_k\}\|d_k\|. \tag{6b}$$

See [22] for motivation of these line search conditions and a description of how, given α_k and d_k satisfying (5), an x_{k+1} satisfying (6) can be found in a finite number of operations.

Our algorithmic framework, AGS, is presented as Algorithm 2.1. In the algorithm and our subsequent analysis, we suppose that iteration k involves setting an approximate Hessian H_k and computing a search direction by (3). Note, however, that the algorithm can be implemented equivalently by setting an approximate inverse Hessian W_k and computing an optimal solution to (4). In the latter case, the search direction is obtained by setting $d_k \leftarrow -W_k G_k \pi_k$ and the quantity $d_k^{\mathrm{T}} H_k d_k$ can be replaced by the equal quantity $\pi_k^{\mathrm{T}} G_k^{\mathrm{T}} W_k G_k \pi_k$. Thus, in either case, H_k or W_k is needed for all k, but not both.

Algorithm 2.1 Adaptive Gradient Sampling (AGS) Algorithm

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

If $\bar{p} = p \ge n + 1$ and $H_k = I$ (or $W_k = I$) for all k, then Algorithm 2.1 reduces to GS as proposed in [22]; specifically, it reduces to the variant involving non-normalized search directions in Section 4.1 of that paper. We use AGS, therefore, to refer to instantiations of Algorithm 2.1 where $\bar{p} < p$ with (potentially) variable H_k. Our numerical experiments in Section 5 illustrate a variety of practical advantages of AGS over GS, while the analysis in Section 4 shows that nothing is lost in terms of convergence guarantees when $\bar{p} < p$.

3. Hessian approximation strategies

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

A critical motivating factor in the design of our Hessian updating strategies is the following assumption needed for our global convergence guarantees in Section 4.

Assumption 3.1 *There exist $\bar{\xi} \geq \underline{\xi} > 0$ such that, for all k and $d \in \mathbb{R}^n$, we have*

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

For each of our updating strategies, we show that Assumption 3.1 is satisfied. We remark, however, that numerical experiments have shown that for non-smooth problems it can be beneficial to allow Hessian approximations to approach singularity [25]. Thus, our numerical experiments include forms of our updates that ensure Assumption 3.1 is satisfied as well as forms that do not. Either of these forms can be obtained through choices of the user-defined constants defined for each update. Note also that the bounds we provide are worst-case bounds that typically would not be tight in practice.

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

$$\underline{\mu} \leq \mu_k \leq \bar{\mu}. \tag{7}$$

Note that, for simplicity, we discuss updates for H_k and W_k as if they are both computed during iteration k. However, as mentioned in Section 2, only one of the two matrices is actually needed in each iteration of AGS.

3.1 *LBFGS updates on sampled directions*

We consider an updating strategy based on the well-known BFGS formula [2,10,11,30]. During iteration k, the main idea of our update is to use gradient information at the points in X_k to construct H_k or W_k. We begin by initializing $H_k \leftarrow \mu_k I$ or $W_k \leftarrow \mu_k^{-1} I$ and then perform a series of (at most) $p_k + 1 \leq p + 1$ updates based on $d_{k,i} := x_{k,i} - x_k$ and $y_{k,i} := \nabla f(x_{k,i}) - \nabla f(x_k)$ for $i = 0, \ldots, p_k$. As at most $p_k + 1$ updates are performed, this strategy is most accurately described as a LBFGS approach for setting H_k and W_k [27]. In the end, after all $p_k + 1$ updates are performed, we obtain bounds of the type required in Assumption 3.1 where the constants $\bar{\xi} \geq \underline{\xi} > 0$ depend only on p and user-defined constants $\gamma > 0$ and $\sigma > 0$.

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

$$\|d_{k,i}\|^2 \le \epsilon_k^2. \tag{8}$$

Moreover, we will require that

$$d_{k,i}^{\mathrm{T}} y_{k,i} \ge \gamma \epsilon_k^2 \tag{9a}$$

and

$$\|y_{k,i}\|^2 \le \sigma \epsilon_k^2 \tag{9b}$$

for the constants $\gamma > 0$ and $\sigma > 0$ provided by the user. We skip the update for sample point i if (9) fails to hold. (For instance, for some i we have $x_{k,i} = x_k$, meaning that $d_{k,i} = y_{k,i} = 0$ and (9a) is not satisfied. Indeed, it is possible that there is no i such that (9) holds, in which case the overall strategy yields $H_k = \mu_k I$ or $W_k = \mu_k^{-1} I$.) For ease of exposition, however, we suppose throughout the remainder of this section that no updates are skipped, this assumption not invalidating our main results, Theorem 3.3 and Corollary 3.4.

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

$$H_k \leftarrow H_k - \frac{H_k d_{k,i} d_{k,i}^{\mathrm{T}} H_k}{d_{k,i}^{\mathrm{T}} H_k d_{k,i}} + \frac{y_{k,i} y_{k,i}^{\mathrm{T}}}{y_{k,i}^{\mathrm{T}} d_{k,i}}, \tag{10a}$$

$$W_k \leftarrow \left(I - \frac{y_{k,i} d_{k,i}^{\mathrm{T}}}{d_{k,i}^{\mathrm{T}} y_{k,i}}\right)^{\mathrm{T}} W_k \left(I - \frac{y_{k,i} d_{k,i}^{\mathrm{T}}}{d_{k,i}^{\mathrm{T}} y_{k,i}}\right) + \frac{d_{k,i} d_{k,i}^{\mathrm{T}}}{d_{k,i}^{\mathrm{T}} y_{k,i}}. \tag{10b}$$

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

LEMMA 3.2 *Suppose that after updates have been performed for sample points zero through $i-1$, we have $H_k \succ 0$ and $W_k \succ 0$, and for any $d \in \mathbb{R}^n$ we have $d^{\mathrm{T}} H_k d \le \theta \|d\|^2$ and $d^{\mathrm{T}} W_k d \le \beta \|d\|^2$ for some $\theta > 0$ and $\beta > 0$. Then, after applying (10), we maintain $H_k \succ 0$ and $W_k \succ 0$ and have*

$$d^{\mathrm{T}} H_k d \le \left(\theta + \frac{\sigma}{\gamma}\right) \|d\|^2, \tag{11a}$$

and

$$d^{\mathrm{T}} W_k d \le \left(2\beta\left(1 + \frac{\sigma}{\gamma^2}\right) + \frac{1}{\gamma}\right) \|d\|^2. \tag{11b}$$

Proof That BFGS updates of the form (10) maintain positive-definiteness of H_k and W_k if $d_{k,i}^{\mathrm{T}} y_{k,i} > 0$ is well known [28, Chapter 6]. Thus, as AGS has $\epsilon_k > 0$ for all k and (9a) ensures $d_{k,i}^{\mathrm{T}} y_{k,i} > 0$, it remains only to prove (11).

Since prior to the update we have $H_k \succ 0$ and (9a) with $\epsilon_k > 0$ ensures that $d_{k,i} \neq 0$, we have from (9) and (10a) that

$$\begin{aligned} d^{\mathrm{T}} H_k d &\leftarrow d^{\mathrm{T}} H_k d - \frac{(d_{k,i}^{\mathrm{T}} H_k d)^2}{d_{k,i}^{\mathrm{T}} H_k d_{k,i}} + \frac{(y_{k,i}^{\mathrm{T}} d)^2}{d_{k,i}^{\mathrm{T}} y_{k,i}} \\ &\leq d^{\mathrm{T}} H_k d + \frac{(y_{k,i}^{\mathrm{T}} d)^2}{d_{k,i}^{\mathrm{T}} y_{k,i}} \\ &\leq \left(\theta + \frac{\sigma}{\gamma}\right) \|d\|^2. \end{aligned}$$

This is precisely (11a). As for (11b), first note that from (8) and (9a), we have

$$d^{\mathrm{T}} \left(\frac{d_{k,i} d_{k,i}^{\mathrm{T}}}{d_{k,i}^{\mathrm{T}} y_{k,i}}\right) d = \frac{(d_{k,i}^{\mathrm{T}} d)^2}{d_{k,i}^{\mathrm{T}} y_{k,i}} \leq \frac{\epsilon_k^2 \|d\|^2}{\gamma \epsilon_k^2} = \frac{1}{\gamma} \|d\|^2. \tag{12}$$

Since prior to the update, we have $W_k \succ 0$, we may write $W_k = N_k^{\mathrm{T}} N_k$ for some non-singular $N_k \in \mathbb{R}^{n \times n}$. From the statement of the lemma, we have that for any d

$$d^{\mathrm{T}} W_k d = \|N_k d\|^2 \leq \beta \|d\|^2,$$

which, along with (8) and (9), yields

$$\begin{aligned} \left\| \left(\frac{d_{k,i}^{\mathrm{T}} d}{d_{k,i}^{\mathrm{T}} y_{k,i}}\right) N_k y_{k,i} \right\|^2 &= \left| \frac{d_{k,i}^{\mathrm{T}} d}{d_{k,i}^{\mathrm{T}} y_{k,i}} \right|^2 \|N_k y_{k,i}\|^2 \\ &\leq \left(\frac{\epsilon_k^2 \|d\|^2}{\gamma^2 \epsilon_k^4}\right) \beta \sigma \epsilon_k^2 \\ &= \beta \frac{\sigma}{\gamma^2} \|d\|^2. \end{aligned} \tag{13}$$

Together, (13) and the fact that for any vectors a and b of equal length, we have $\|a - b\|^2 \leq 2(\|a\|^2 + \|b\|^2)$ show that

$$\begin{aligned} \left\| N_k \left(I - \frac{y_{k,i} d_{k,i}^{\mathrm{T}}}{d_{k,i}^{\mathrm{T}} y_{k,i}}\right) d \right\|^2 &\leq 2 \left(\|N_k d\|^2 + \left\| \left(\frac{d_{k,i}^{\mathrm{T}} d}{d_{k,i}^{\mathrm{T}} y_{k,i}}\right) N_k y_{k,i} \right\|^2 \right) \\ &\leq 2\beta \left(1 + \frac{\sigma}{\gamma^2}\right) \|d\|^2. \end{aligned}$$

Overall, (10b), the above equation, and (12) yield

$$\begin{aligned} d^{\mathrm{T}} W_k d &\leftarrow \left\| N_k \left(I - \frac{y_{k,i} d_{k,i}^{\mathrm{T}}}{d_{k,i}^{\mathrm{T}} y_{k,i}}\right) d \right\|^2 + d^{\mathrm{T}} \left(\frac{d_{k,i} d_{k,i}^{\mathrm{T}}}{d_{k,i}^{\mathrm{T}} y_{k,i}}\right) d \\ &\leq \left(2\beta \left(1 + \frac{\sigma}{\gamma^2}\right) + \frac{1}{\gamma}\right) \|d\|^2, \end{aligned}$$

which is precisely (11b). ■

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

THEOREM 3.3 *For any k, after all updates have been performed via (10a) for sample points zero through p_k, the following holds for any $d \in \mathbb{R}^n$:*

$$d^{\mathrm{T}} H_k d \geq \left(2^{p+1}\left(1+\frac{\sigma}{\gamma^2}\right)^{p+1} \mu_k^{-1} + \frac{1}{\gamma}\left(\frac{2^{p+1}(1+\sigma/\gamma^2)^{p+1}-1}{2(1+\sigma/\gamma^2)-1}\right)\right)^{-1} \|d\|^2, \tag{14a}$$

$$d^{\mathrm{T}} H_k d \leq \left(\mu_k + \frac{(p+1)\sigma}{\gamma}\right)\|d\|^2. \tag{14b}$$

Proof Since H_k and W_k are initialized to the positive- definite matrices $\mu_k I$ and $\mu_k^{-1} I$, respectively, we have that prior to the first update the bounds in Lemma 3.2 hold with $\theta = \mu_k$ and $\beta = \mu_k^{-1}$. Thus, the result of Lemma 3.2 can be applied repeatedly to produce upper bounds for $d^{\mathrm{T}} H_k d$ and $d^{\mathrm{T}} W_k d$ that hold for any $d \neq 0$ after all $p_k + 1$ updates have been performed. (Note that the bounds are trivially satisfied for $d = 0$.) Specifically, the upper bound for $d^{\mathrm{T}} H_k d/\|d\|^2$ increases by the constant factor σ/γ for every update, implying that after all $p_k + 1$ updates have been performed, we have

$$d^{\mathrm{T}} H_k d \leq \left(\mu_k + \frac{(p_k+1)\sigma}{\gamma}\right)\|d\|^2.$$

Then, since $p \geq p_k$, we have the upper bound in (14b).

As for the lower bound in (14a), we begin by letting $\delta := \left(1+\sigma/\gamma^2\right)$ and $\omega := 1/\gamma$. Then, by applying the result in Lemma 3.2 repeatedly for $i = 0, \dots, p_k$, we find that after all updates have been performed, we have the following upper bound for inner products with W_k:

$$\begin{aligned} d^{\mathrm{T}} W_k d &\leq (2^{p_k+1}\delta^{p_k+1}\mu_k^{-1} + 2^{p_k}\delta^{p_k}\omega + \cdots + 2\delta\omega + \omega)\|d\|^2 \\ &= \left(2^{p_k+1}\delta^{p_k+1}\mu_k^{-1} + \omega\left(\frac{2^{p_k+1}\delta^{p_k+1}-1}{2\delta-1}\right)\right)\|d\|^2. \end{aligned}$$

This implies via the Rayleigh–Ritz theorem [19] that after all updates have been performed, we have

$$d^{\mathrm{T}} H_k d \geq \left(2^{p_k+1}\delta^{p_k+1}\mu_k^{-1} + \omega\left(\frac{2^{p_k+1}\delta^{p_k+1}-1}{2\delta-1}\right)\right)^{-1}\|d\|^2.$$

As before, since $p \geq p_k$, this implies the lower bound in (14a). ■

We note that the following corollary follows by applying the Rayleigh–Ritz theorem to the result of Theorem 3.3.

COROLLARY 3.4 *For any k, after all updates have been performed via (10b) for sample points zero through p_k, the following holds for any $d \in \mathbb{R}^n$:*

$$d^{\mathrm{T}} W_k d \leq \left(2^{p+1}\left(1+\frac{\sigma}{\gamma^2}\right)^{p+1} \mu_k^{-1} + \frac{1}{\gamma}\left(\frac{2^{p+1}(1+\sigma/\gamma^2)^{p+1}-1}{2(1+\sigma/\gamma^2)-1}\right)\right)\|d\|^2, \tag{15a}$$

$$d^{\mathrm{T}} W_k d \geq \left(\mu_k + \frac{(p+1)\sigma}{\gamma}\right)^{-1}\|d\|^2. \tag{15b}$$

3.2 *Updates to promote model overestimation*

During iteration k, the primal subproblem (3) is equivalent to the following:

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

If $m_k(d) \geq f(x_k + d)$ for all $d \in \mathbb{R}^n$, then a reduction in f is obtained after a step along $d_k \neq 0$ computed from (3)–(4). Thus, it is desirable to choose H_k so that m_k overestimates f to guarantee that such reductions occur in AGS.

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

$$H_k \leftarrow M_{k,p_k}^{\mathrm{T}} \cdots M_{k,0}^{\mathrm{T}} (\mu_k I) M_{k,0} \cdots M_{k,p_k}, \tag{16}$$

where $M_{k,i}$ is chosen based on information obtained along $d_{k,i}$. (Note that such an H_k can be obtained by initializing $H_k \leftarrow \mu_k I$ and updating $H_k \leftarrow M_{k,i}^{\mathrm{T}} H_k M_{k,i}$ for $i = 0, \ldots, p_k$.) We choose $M_{k,i}$ in such a way that H_k remains well conditioned and obtain bounds of the type required in Assumption 3.1 where $\overline{\xi} \geq \underline{\xi} > 0$ depend only on p and a user-defined constant $\rho \geq \frac{1}{2}$.

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

$$M_{k,i} = \begin{cases} I + \dfrac{\rho_{k,i}}{d_{k,i}^{\mathrm{T}} d_{k,i}} d_{k,i} d_{k,i}^{\mathrm{T}} \succ 0 & \text{if } d_{k,i} \neq 0, \\ I & \text{if } d_{k,i} = 0, \end{cases} \tag{17}$$

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

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

Specifically, if $m_k(d_{k,i}) \geq f(x_{k,i})$, then we choose $\rho_{k,i} \leftarrow 0$, which by (17) means that $M_{k,i} \leftarrow I$. Otherwise, we set

$$\rho_{k,i} = -1 + \sqrt{\frac{2\Delta_{k,i}}{d_{k,i}^{\mathrm{T}} H_k d_{k,i}}}, \tag{18}$$

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

$$\Delta_{k,i} = \min\{f(x_{k,i}) - m_k(d_{k,i}) + \tfrac{1}{2} d_{k,i}^{\mathrm{T}} H_k d_{k,i}, \rho d_{k,i}^{\mathrm{T}} H_k d_{k,i}\}. \tag{19}$$

In this latter case, when $m_k(d_{k,i}) < f(x_{k,i})$, we have $\Delta_{k,i} \geq \frac{1}{2} d_{k,i}^{\mathrm{T}} H_k d_{k,i}$, implying that $\rho_{k,i} \geq 0$. Moreover, as (19) also yields $\Delta_{k,i} \leq \rho d_{k,i}^{\mathrm{T}} H_k d_{k,i}$, it follows that $\rho_{k,i} \leq \sqrt{2\rho} - 1$. Thus, $\rho_{k,i} \in [0, \sqrt{2\rho} - 1]$. Notice that in the process of performing the update with $d_{k,i} \neq 0$ and $\rho_{k,i}$ set by

(18), we have from (17) that

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

Thus, by (19), if $\Delta_{k,i} = f(x_{k,i}) - m_k(d_{k,i}) + \frac{1}{2}d_{k,i}^{\mathrm{T}}H_kd_{k,i}$, then the model value $m_k(d_{k,i})$ has been increased to the function value $f(x_{k,i})$. Otherwise, if $\Delta_{k,i} = \rho d_{k,i}^{\mathrm{T}}H_kd_{k,i}$, then the model value is still increased since $\rho \geq \frac{1}{2}$.

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

Lemma 3.5 *Let $M_{k,i}$ be defined by (17). Then, for any $d \in \mathbb{R}^n$, we have*

$$\|d\|^2 \leq d^{\mathrm{T}}M_{k,i}^{\mathrm{T}}M_{k,i}d \leq (1+\rho_{k,i})^2\|d\|^2. \tag{20}$$

Proof The result is trivial for $d = 0$. Moreover, for d with $\|d\| \neq 0$, we find

$$d^{\mathrm{T}}M_{k,i}^{\mathrm{T}}M_{k,i}d = d^{\mathrm{T}}d + \frac{2\rho_{k,i} + \rho_{k,i}^2}{d_{k,i}^{\mathrm{T}}d_{k,i}}(d_{k,i}^{\mathrm{T}}d)^2.$$

Thus, (20) follows by the extreme cases $d_{k,i}^{\mathrm{T}}d = 0$ and $d_{k,i}^{\mathrm{T}}d = \|d_{k,i}\|\|d\|$. ■

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

Theorem 3.6 *For any k, with H_k defined by (16), $M_{k,i}$ defined by (17), and $\rho_{k,i} \in [0, \sqrt{2\rho} - 1]$ for $i = 0, \ldots, p_k$, the following holds for any $d \in \mathbb{R}^n$:*

$$\mu_k\|d\|^2 \leq d^{\mathrm{T}}H_kd \leq \mu_k(2\rho)^{p+1}\|d\|^2. \tag{21}$$

Proof First, we repeatedly apply the result of Lemma 3.5 to obtain

$$\begin{aligned}
d^{\mathrm{T}}H_kd &\leftarrow d^{\mathrm{T}}M_{k,p_k}^{\mathrm{T}}\cdots M_{k,0}^{\mathrm{T}}(\mu_kI)M_{k,0}\cdots M_{k,p_k}d \\
&= \mu_k(M_{k,1}\cdots M_{k,p_k}d)^{\mathrm{T}}M_{k,0}^{\mathrm{T}}M_{k,0}(M_{k,1}\cdots M_{k,p_k}d) \\
&\leq \mu_k(1+\rho_{k,0})^2(M_{k,1}\cdots M_{k,p_k}d)^{\mathrm{T}}(M_{k,1}\cdots M_{k,p_k}d) \\
\cdots &\leq \mu_k\left(\prod_{i=0}^{p_k}(1+\rho_{k,i})^2\right)\|d\|^2.
\end{aligned}$$

Since $\rho_{k,i} \in [0, \sqrt{2\rho} - 1]$ for $0 = 1, \ldots, p_k$ and since AGS maintains $p_k \geq p$ for all k, we then find

$$\mu_k\left(\prod_{i=0}^{p_k}(1+\rho_{k,i})^2\right)\|d\|^2 \leq \mu_k(2\rho)^{p_k+1}\|d\|^2 \leq \mu_k(2\rho)^{p+1}\|d\|^2.$$

The lower bound on $d^{\mathrm{T}}H_kd$ in (21) can be found in a similar manner. ■

The approximation $W_k = H_k^{-1}$ for the inverse Hessian corresponding to (16) is

$$W_k \leftarrow M_{k,p_k}^{-T} \cdots M_{k,1}^{-T}(\mu_k^{-1} I) M_{k,1}^{-1} \cdots M_{k,p_k}^{-1} \tag{22}$$

where the Sherman–Morrison–Woodbury formula [13] reveals that for each $i = 0, \ldots, p_k$, we have

$$M_{k,i}^{-1} = \begin{cases} I - \dfrac{\rho_{k,i}}{(1+\rho_{k,i}) d_{k,i}^{\mathrm{T}} d_{k,i}} d_{k,i} d_{k,i}^{\mathrm{T}} \succ 0 & \text{if } d_{k,i} \neq 0, \\ I & \text{if } d_{k,i} = 0. \end{cases} \tag{23}$$

The following corollary follows by applying the Rayleigh–Ritz theorem to the result of Theorem 3.6.

Corollary 3.7 *For any k, with W_k defined by (22), $M_{k,i}^{-1}$ defined by (23), and $\rho_{k,i} \in [0, \sqrt{2\rho} - 1]$ for $0 = 1, \ldots, p_k$, the following holds for any $d \in \mathbb{R}^n$:*

$$\mu_k^{-1}(2\rho)^{-p-1}\|d\|^2 \leq d^{\mathrm{T}} W_k d \leq \mu_k^{-1}\|d\|^2. \tag{24}$$

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

Theorem 3.8 *Suppose f is convex. Then, for any k, the matrices H_k and W_k described in Theorems 3.6 and 3.7, respectively, satisfy $H_k = \mu_k I$ and $W_k = \mu_k^{-1} I$.*

Proof It suffices to show that for any k and i, we have $m_k(d_{k,i}) \geq f(x_{k,i})$, meaning that the method will always choose $\rho_{k,i} \leftarrow 0$ and $M_{k,i} \leftarrow I$ and so H_k and W_k will never be altered from their initial values. Since f is convex and $x_{k,i} = x_k + d_{k,i}$,

$$\begin{aligned} m_k(d_{k,i}) &= f(x_k) + \max_{x \in X_k}\{\nabla f(x)^{\mathrm{T}} d_{k,i}\} + \tfrac{1}{2} d_{k,i}^{\mathrm{T}} H_k d_{k,i} \\ &\geq f(x_k) + \nabla f(x_{k,i})^{\mathrm{T}} d_{k,i} + \tfrac{1}{2} d_{k,i}^{\mathrm{T}} H_k d_{k,i} \\ &\geq f(x_{k,i}) + \tfrac{1}{2} d_{k,i}^{\mathrm{T}} H_k d_{k,i}. \end{aligned}$$

Thus, since $H_k = \mu_k I \succ 0$ initially, we have $m_k(d_{k,i}) \geq f(x_{k,i})$ for all k and i. ■

4. Global convergence analysis

We make the following assumption about the objective function f of (1) throughout our global convergence analysis.

Assumption 4.1 *The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz and continuously differentiable in an open dense subset $\mathcal{D} \subset \mathbb{R}^n$.*

We also make Assumption 3.1 stated previously at the beginning of Section 3.

The result we prove is the following.

Theorem 4.2 *AGS produces an infinite sequence of iterates $\{x_k\}$ and, with probability one, either $f(x_k) \to -\infty$ or $\{\epsilon_k\} \to 0$ and every cluster point of $\{x_k\}$ is stationary for f.*

Our analysis follows closely that of Kiwiel in [22]. However, there are subtle differences due to the adaptive sampling procedure and the variable-metric Hessian approximations. Thus, we analyse the global convergence behaviour of AGS for the sake of completeness.

We begin our analysis for proving Theorem 4.2 by showing that AGS is well posed in the sense that each iteration terminates finitely. It is clear that this will be true as long as the backtracking line search in step 6 terminates finitely.

LEMMA 4.3 *If $p_k < p$ in step 6, then $\alpha_k > 0$ is computed satisfying (5) or $\alpha_k \leftarrow 0$. If $p_k \geq p$ in step 6, then $\alpha_k > 0$ is computed satisfying (5).*

Proof If $p_k < p$ in step 6, then the statement is obviously true since only a finite number of values of α_k are considered. Next, we consider the case when $p_k \geq p$. The Karush–Kuhn–Tucker conditions of (3) are

$$ze - f(x_k)e - G_k^{\mathrm{T}} d \geq 0, \tag{25a}$$

$$\pi \geq 0, \tag{25b}$$

$$1 - \pi^{\mathrm{T}} e = 0, \tag{25c}$$

$$H_k d + G_k \pi = 0, \tag{25d}$$

$$\pi^{\mathrm{T}}(ze - f(x_k)e - G_k^{\mathrm{T}} d) = 0. \tag{25e}$$

Let (z_k, d_k, π_k) be the unique solution of (25). Then, (25c)–(25e) and the fact that H_k is symmetric yield

$$z_k - f(x_k) = \pi_k^{\mathrm{T}} G_k^{\mathrm{T}} d_k = -d_k^{\mathrm{T}} H_k d_k. \tag{26}$$

Plugging the above equality into (25a), we have

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

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

$$\nabla f(x_k)^{\mathrm{T}} d_k \leq -d_k^{\mathrm{T}} H_k d_k. \tag{27}$$

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

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \eta \alpha_k \nabla f(x_k)^{\mathrm{T}} d_k \leq f(x_k) - \eta \alpha_k d_k^{\mathrm{T}} H_k d_k. \qquad \blacksquare$$

Lemma 4.3 reveals that the line search will yield $\alpha_k \leftarrow 0$ or $\alpha_k > 0$ satisfying (5). Our next lemma builds on this result and shows that there will be an infinite number of iterations during which the latter situation occurs.

LEMMA 4.4 *There exists an infinite subsequence of iterations in which $\alpha_k > 0$.*

Proof By step (5), if $\min\{\|d_k\|^2, d_k^{\mathrm{T}} H_k d_k\} \leq \nu \epsilon_k^2$ an infinite number of times, then the result follows as the algorithm sets $\alpha_k \leftarrow 1$ for such iterations. Otherwise, to derive a contradiction, suppose there exists $k' \geq 0$ such that for $k \geq k'$, step 6 is reached and sets $\alpha_k \leftarrow 0$. By Lemma 4.3, this means that for $k \geq k'$, we have $p_k \leq p - 1$. However, by steps 7, 8, and then 2, it is clear that if $\alpha_k \leftarrow 0$, then $p_{k+1} = \min\{p, p_k + \bar{p}\}$, contradicting the conclusion that $\{p_k\}$ is bounded above by $p - 1$ for all $k \geq k'$. ■

We now show a critical result about the sequence of decreases produced in f. A similar result was proved in [22].

Lemma 4.5 *The following inequality holds for all k*:

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

Proof By the triangle inequality, condition (6b) ensures that

$$\|x_{k+1} - x_k\| \le \min\{\alpha_k, \epsilon_k\}\|d_k\| + \alpha_k\|d_k\| \le 2\alpha_k\|d_k\|. \tag{28}$$

Indeed, this inequality holds trivially if the algorithm sets $x_{k+1} \leftarrow x_k$ in step 5 or sets $\alpha_k \leftarrow 0$ in step 6, and holds by the triangle inequality if step 6 yields $x_{k+1} \leftarrow x_k + \alpha_k d_k$. Thus, by (5), (6), and (28), we find that for all k,

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

as desired. ■

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

$$\inf_d \; q(d; x', \mathbb{B}_{\epsilon_k}(x'), H_k), \tag{29}$$

where

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

Given a solution d' of (29), we have the following reduction in its objective:

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

Similarly, writing (3) in the form [21]

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

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

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

We now show a result about the above reduction.

Lemma 4.6 *The following equality holds*:

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

Proof By the definition of q, we have $q(0; x_k, X_k, H_k) = f(x_k)$. Moreover, by (26), we have $q(d_k; x_k, X_k, H_k) = z_k + \tfrac{1}{2} d_k^{\mathrm{T}} H_k d_k = f(x_k) - \tfrac{1}{2} d_k^{\mathrm{T}} H_k d_k$. Therefore, $\Delta q(d_k; x_k, X_k, H_k) = \tfrac{1}{2} d_k^{\mathrm{T}} H_k d_k$. ■

The purpose of our next lemma is to show that for any desired level of accuracy (though not necessarily perfect accuracy), as long as x_k is sufficiently close to x', there exists a sample set X_k such that the reduction $\Delta q(d_k; x_k, X_k, H_k)$ produced by the solution d_k of (3) will be sufficiently close to the reduction $\Delta q(d'; x', \mathbb{B}_{\epsilon_k}(x'), H_k)$ produced by the solution d' of (29). For a given x' and tolerance ω, we define

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

This set plays a critical role in the following lemma. A similar result was proved in [22], and in the context of constrained optimization in [8].

LEMMA 4.7 *If $p_k \ge n+1$, then for any $\omega > 0$, there exists $\zeta > 0$ and a non-empty set $\mathcal{T}$ such that for all $x_k \in \mathbb{B}_\zeta(x')$, we have $\mathcal{T} \subset \mathcal{T}_k(x', \omega)$.*

Proof Under Assumption 4.1, there exists a vector d satisfying

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

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

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

Then, since $p_k \ge n+1$, Carathéodory's theorem [29] implies that there exists $\{y_0, \dots, y_{p_k}\} \subset \mathbb{B}_{\epsilon_k}(x') \cap \mathcal{D}$ and a set of non-negative scalars $\{\lambda_0, \dots, \lambda_{p_k}\}$ such that

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

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

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

lies in $\mathbb{B}_{\epsilon_k - \zeta}(x')$ and the solution d_k to (3) with $X_k \in \mathcal{T}$ satisfies

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

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

We are now prepared to prove Theorem 4.2. Our proof follows closely that of [22, Theorem 3.3]. We provide a proof for the sake of completeness and since subtle changes to the proof are required due to our adaptive sampling strategy.

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

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

Then, we have from (5), (6), and Lemma 4.5 that

$$\sum_{k=0}^{\infty} \alpha_k d_k^{\mathrm{T}} H_k d_k < \infty \tag{30a}$$

and

$$\sum_{k=0}^{\infty} \|x_{k+1} - x_k\| \|d_k\| < \infty. \tag{30b}$$

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

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

$$\min\{\|d_k\|^2, d_k^{\mathrm{T}} H_k d_k\} > \nu\epsilon'^2 \quad \text{for all } k \geq k'. \tag{31}$$

In conjunction with (30), this implies $\alpha_k \to 0$ and $x_k \to x'$ for some x'. Moreover, the fact that $\alpha_k \to 0$ implies that there exists an infinite subsequence of iterations in which $p_k = p$. Indeed, if $p_k < p$ for all large k, then since $\alpha_k \to 0$, step 6 implies that $\alpha_k \leftarrow 0$ for all large k. However, as in the proof of Lemma 4.4, this leads to a contradiction as we eventually find $p_k = p$ for some k. Therefore, we can define $\mathcal{K}$ as the subsequence of iterations in which $p_k = p$ and know that $\mathcal{K}$ is infinite.

Case 1a. If x' is ϵ'-stationary for f, then for any $H_k \succ 0$, the solution d' to (29) satisfies $\Delta q(d'; x', \mathbb{B}_{\epsilon'}(x'), H_k) = 0$. Thus, with $\omega = \nu\epsilon'^2/2$ and $(\zeta, \mathcal{T})$ chosen as in Lemma 4.7, there exists $k'' \geq k'$ such that $x_k \in \mathbb{B}_{\zeta}(x')$ for all $k \geq k''$ and

$$\tfrac{1}{2} d_k^{\mathrm{T}} H_k d_k = \Delta q(d_k; x_k, X_k, H_k) \leq \tfrac{1}{2}\nu\epsilon'^2 \tag{32}$$

whenever $k \geq k''$, $k \in \mathcal{K}$, and $X_k \in \mathcal{T}$. Together, (31) and (32) imply that $X_k \notin \mathcal{T}$ for all $k \geq k''$ with $k \in \mathcal{K}$. However, this is a probability zero event since for all such k the set X_k continually collects points generated uniformly from B_k, meaning that it will eventually include an element of the set $\mathcal{T}$ yielding (32).

Case 1b. If x' is not ϵ'-stationary, then for all $k \geq k'$, any α not satisfying the sufficient decrease condition (5) yields

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

and along with (27) yields

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

Here, L_k is a finite upper bound for $(f'(x_k + \alpha d_k) - f'(x_k))/(\alpha\|d_k\|)$ on the interval $[x_k, x_k + \alpha d_k]$ whose existence follows from Assumption 4.1. Combining the above inequalities yields a lower bound on any α not satisfying (5), which, since step 6 invokes the backtracking factor κ, yields the bound

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

However, with $\omega = \Delta q(d'; x', \mathbb{B}_{\epsilon'}(x'), H_k)$ (which is strictly positive since x' is not ϵ'-stationary) and $(\zeta, \mathcal{T})$ again chosen as in Lemma 4.7, there exists $k'' \geq k'$ such that $x_k \in \mathbb{B}_{\zeta}(x')$ for all $k \geq k''$ and

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

whenever $k \geq k''$, $k \in \mathcal{K}$, and $X_k \in \mathcal{T}$. Under Assumptions 4.1 and 3.1 and since $x_k \to x'$, we have that for all k sufficiently large, $L_k \|d_k\|^2 \leq L$ for some constant $L > 0$, implying that for all

$k \geq k''$ with $k \in \mathcal{K}$ such that $X_k \in \mathcal{T}$, α_k is bounded away from zero. Together, this and the fact that $\alpha_k \to 0$ imply that $X_k \notin \mathcal{T}$ for all $k \geq k''$ with $k \in \mathcal{K}$. Again, this is a probability zero event.

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

$$\liminf_{k\to\infty} \max\{\|x_k - x'\|, \|d_k\|\} = 0. \tag{33}$$

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

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

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

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

yielding (33). On the other hand, if $x_k \nrightarrow x'$, then we proceed by contradiction and suppose that (33) does not hold. Since x' is a cluster point of $\{x_k\}$, there is an $\epsilon' > 0$ and an index $k' \geq 0$ such that the set $K' := \{k : k \geq k', \|x_k - x'\| \leq \epsilon', \|d_k\| > \epsilon'\}$ is infinite. By (30b), this means

$$\sum_{k\in K'} \|x_{k+1} - x_k\| < \infty. \tag{34}$$

Since $x_k \nrightarrow x'$, there exists an $\epsilon > 0$ such that for all $k_1 \in K'$ with $\|x_{k_1} - x'\| \leq \epsilon'/2$, there is $k_2 > k_1$ satisfying $\|x_{k_1} - x_{k_2}\| > \epsilon$ and $\|x_k - x'\| \leq \epsilon'$ for all $k_1 \leq k \leq k_2$. Thus, by the triangle inequality, we have $\epsilon < \|x_{k_1} - x_{k_2}\| \leq \sum_{k=k_1}^{k_2-1} \|x_{k+1} - x_k\|$. However, for $k_1 \in K'$ sufficiently large, (34) implies that the right-hand side of this inequality must be strictly less than ϵ, a contradiction.

Finally, since for all k the elements of X_k lie in B_k, Equation (33) and $\{\epsilon_k\} \to 0$ imply that the cluster point x' is stationary for f. ■

5. An implementation

We have implemented Algorithm 2.1 in Matlab along with the QO subproblem solver described in Appendix. In this section, we describe the algorithm variations that we have tested, the test problems that we have solved, and the results of our numerical experiments. All tests were performed on a machine running Debian 2.6.32 with two 8-Core AMD Opteron 6128 2.0 GHz processors and 32 GB RAM.

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

5.1 *Algorithm variations*

Given varying values for the input parameters, our implementation of Algorithm 2.1 yields the algorithm variations described below.

- `GS`. This is a basic GS algorithm with non-normalized search directions [22, Section 4.1], obtained by choosing $\bar{p} = p \geq n + 1$ with $W_k = I$ for all k. We consider this variant of GS for comparison purposes as it is most similar with the AGS variations described below. The global convergence analysis in Section 4 applies for this algorithm as long as Assumption 4.1 holds.

Table 1. Summary of six algorithm variations used to test the adaptive sampling procedure in Algorithm 2.1 along with the Hessian approximation updating strategies described in Sections 3.1 and 3.2.

Name	Samples per iteration	Hessian updates	γ	σ	ρ
GS	$\bar{p} = p \geq n+1$	None	–	–	–
AGS	$\bar{p} < p \geq n+1$	None	–	–	–
AGS-LBFGS	$\bar{p} < p \geq n+1$	Strategy in Section 3.1	0.1	100	–
AGS-LBFGS-ill	$\bar{p} < p \geq n+1$	Strategy in Section 3.1	0	∞	–
AGS-over	$\bar{p} < p \geq n+1$	Strategy in Section 3.2	–	–	100
AGS-over-ill	$\bar{p} < p \geq n+1$	Strategy in Section 3.2	–	–	∞

- `AGS`. This algorithm samples gradients adaptively as we choose $\bar{p} < p$, but it does not use either Hessian updating strategy as we choose $W_k = I$ for all k. The global convergence analysis in Section 4 applies for this algorithm as long as Assumption 4.1 holds.
- `AGS-LBFGS`. This algorithm is an enhanced version of `AGS` where the updating strategy in Section 3.1 is used to set W_k for all k. We choose $\gamma = 0.1$ and $\sigma = 100$ in the updates, which by Corollary 3.4 means that the global convergence analysis in Section 4 applies as long as Assumption 4.1 holds.
- `AGS-LBFGS-ill`. This algorithm is similar to `AGS-LBFGS`, except that we choose $\gamma = 0$ and $\sigma = \infty$ so that W_k may become ill-conditioned. The global convergence analysis in Section 4 does *not* apply for this method.
- `AGS-over`. This algorithm is an enhanced version of `AGS` where the updating strategy in Section 3.2 is used to set W_k for all k. We choose $\rho = 100$ in the updates, which by Corollary 3.7 means that the global convergence analysis in Section 4 applies as long as Assumption 4.1 holds.
- `AGS-over-ill`. This algorithm is similar to `AGS-over`, except that we choose $\rho = \infty$ so that W_k may become ill-conditioned. The global convergence analysis in Section 4 does *not* apply for this method.

We summarize the differing inputs for these six algorithm variations in Table 1.

Specific values for the input parameters mentioned above, as well as for the remaining parameters that were set consistently for all algorithm variations, were chosen as those that yielded the best overall results in our experiments. As recommended in [4,22], we choose $p = 2n$ as the number of sample points required for a complete line search. (Note that this is also the number of sample points and sample gradients computed per iteration for `GS`.) For `AGS` and the remaining variants, we experimented with various values for $\bar{p}$, eventually finding that $\bar{p} = n/10$ yielded nice results. Our convergence analysis in Section 4 requires only $O(1)$ gradients per iteration, but we suggest that setting $\bar{p}$ as a fraction of n may generally yield a good balance between overall gradient evaluations and search direction quality. We set the line search backtracking constant to be $\kappa = 0.5$, sufficient decrease constant to be $\eta = 10^{-8}$, and the number of backtracks in an incomplete line search to be $u = 7$. The initial sampling radius is chosen to be $\epsilon_0 = 0.1$ and $\psi = 0.1$ is set as the sampling radius reduction factor. We choose the stationarity tolerance parameter to be $\nu = 10$ and limit the number of gradient evaluations to $100n$ before terminating the algorithm.

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

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

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

We implemented Algorithm 2.1 along with the QO subproblem solver described in Appendix. We set the subproblem optimality tolerance to 10^{-10} and maximum number of iterations to $\min\{1000, 2^{\max\{n,p_k\}}\}$.

5.2 *Test problems*

We tested the algorithm variations with 26 non-smooth minimization problems, some convex and some non-convex. The first 20 of these problems were considered in [16] and the last 6 were considered in [31]. All problems are scalable in the sense that they can be defined to have different numbers of variables n. The first 10 problems, introduced in [17], are all non-smooth at their respective minimizers: MAXQ, MXHILB, CHAINED_LQ, CHAINED_CB3_I, CHAINED_CB3_II, ACTIVE_FACES, BROWN_FUNCTION_2, CHAINED_MIFFLIN_2, CHAINED_CRESCENT_I, and CHAINED_CRESCENT_II. The first five of these problems are convex and the second five are non-convex. The second 10 problems in our set, some of which are nonconvex, were introduced in the test library TEST29 [26]: TEST29_2, TEST29_5, TEST29_6, TEST29_11, TEST29_13, TEST29_17, TEST29_19, TEST29_20, TEST29_22, and TEST29_24. Of the six remaining problems, the first four were introduced in [25], the fifth was introduced in [14], and the sixth is a problem to minimize the Schatten norm [31]: TILTED_NORM_COND, CPSF, NCPSF, EIG_PROD, GREIF_FUN, and NUC_NORM.

5.3 *Numerical results*

We chose $n = 50$ for all problems. The only exception was EIG_PROD, for which we choose $n = 64$, as the variables for this problem need to compose a square matrix. We ran each problem 10 times, the first time using a fixed initial point x'_0 and the remaining nine times using a starting point generated randomly from a ball about x'_0 with radius $\|x'_0\|$. (We choose $x'_0 \neq 0$ for all problems, so the initial points for each run were unique.) For the first 20 problems, we choose x'_0 as the initial point defined in [16]. For the remaining six problems, we choose $x'_0 = e$. The input parameters we use for TILTED_NORM_COND, CPSF, NCPSF, and NUC_NORM are those used in [31]. The only remaining problem inputs that require specification are the matrices involved in EIG_PROD and GREIF_FUN. For the former, we used the leading 8×8 submatrix of A from [1] and for the latter we used a randomly generated 10×10 symmetric positive-definite matrix A (with the $n = 50$ variables composing a 10×5 matrix X so that the product $X^{\mathrm{T}}AX$ is well defined).

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

First, we compare the results obtained by applying the algorithms GS and AGS to the $26 \times 10 = 260$ test problems. Performance profiles [9] for the final sampling radius and average QO iterations are given in Figure 1. The profiles clearly illustrate the benefits of AGS over GS. Given the same limit on the number of gradient evaluations, AGS is able to perform many more nonlinear iterations than GS due to the fact that AGS requires many fewer gradient evaluations per nonlinear iteration. This allows AGS to make much more progress towards the solution, as evidenced by the final sampling radius consistently being much smaller.

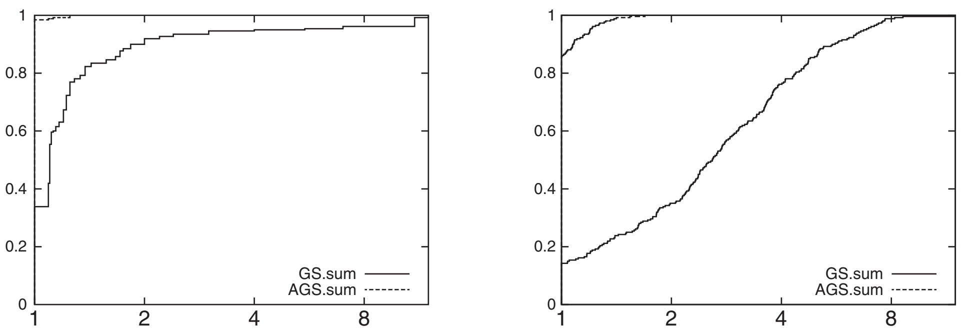


Figure 1. Performance profiles for the final sampling radius (left) and average QO iterations per nonlinear iteration (right) comparing algorithms GS and AGS.

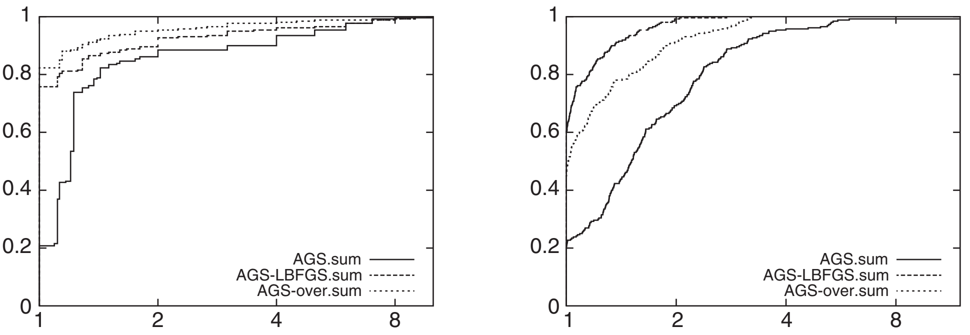
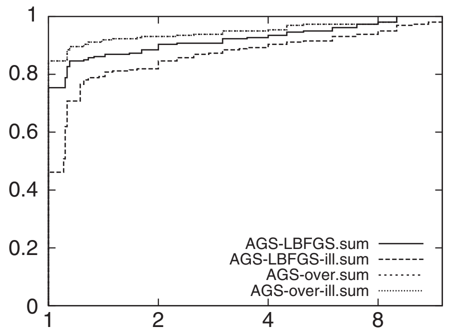


Figure 2. Performance profiles for the final sampling radius (left) and average QO iterations per nonlinear iteration (right) comparing algorithms AGS, AGS-LBFGS, and AGS-over.

One additional remark to make about the performance profiles in Figure 1 is that the number of average QO iterations is significantly fewer for AGS when compared with GS. This can be attributed to the fact that the subproblems in AGS are often smaller than those in GS, and when they are the same size as in GS, warm-starting the solver reduces the number of QO iterations required.

Our second set of performance profiles illustrate the benefits of the Hessian updating strategies in Section 3 by comparing the results for AGS-LBFGS and AGS-over with those for AGS. AGS-over performs better than AGS-LBFGS in terms of the final sampling radius, while it is dominated by AGS-LBFGS in terms of average QO iterations. Both AGS-over and AGS-LBFGS perform better than AGS no matter which performance measure is considered. We did not expect to see a reduction in average QO iterations when the inverse Hessian updates are employed, but in any case our experiments reveal that the updates bring benefits in terms of progress towards a minimizer.

We close this section with performance profiles comparing AGS-over and AGS-LBFGS with AGS-over-ill and AGS-LBFGS-ill, the latter two being variants for which our global convergence analysis in Section 4 does not apply. Despite the fact that in some situations it is believed that allowing Hessian approximations to tend to singularity can be beneficial [25], we do not see much of an impact in our numerical results. (In fact, there appears to be a disadvantage in terms of the final sampling radius when allowing ill-conditioning of AGS-LBFGS-ill.) There are at least a couple possible explanations for this phenomenon. First, due to the fact that we reinitialize W_k during each iteration and perform only a finite number of updates based on sample point information, our Hessian approximations may naturally remain better conditioned than



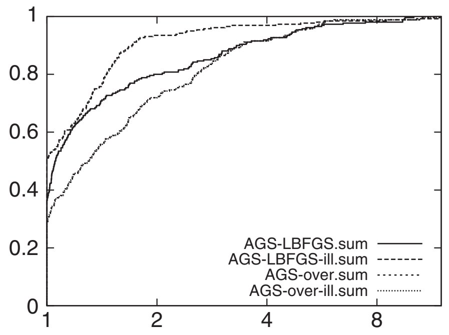


Figure 3. Performance profiles for the final sampling radius (left) and average QO iterations per nonlinear iteration (right) comparing algorithms `AGS-LBFGS`, `AGS-LBFGS-ill`, `AGS-over`, and `AGS-over-ill`.

those obtained by standard quasi-Newton updating techniques that continually build these matrices based on gradient information obtained at all previous algorithm iterates. Second, our input parameter choices may be generous enough that, for example, `AGS-over` and `AGS-over-ill` produce similar Hessian approximations during most iterations.

6. Conclusion

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

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

Finally, we remark that there are interesting similarities between AGS and a forerunner of bundle methods, namely Wolfe's conjugate subgradient method [32]. Wolfe's method, a reasonably effective method for non-differentiable convex optimization, is an extension of the conjugate gradient algorithm for minimizing differentiable functions. It is similar to AGS in that the central idea behind both algorithms is to approximate the subdifferential at (or near) a non-differentiable point via (sub)gradients at nearby points. In particular, both algorithms compute search directions by finding the minimum norm vector in the convex hull of (sub)gradients evaluated at these nearby points. A major difference, however, is that the (sub)gradients in Wolfe's method are the (sub)gradients and search directions obtained at previous iterates, whereas AGS employs random sampling and does not utilize previous search directions in place of gradients. Another important difference between AGS and Wolfe's method is that the latter has guarantees only for convex objective functions, whereas the former can also solve non-convex problems. Still, despite these differences, we plan to investigate whether borrowing ideas from Wolfe's method, namely that of including previous search directions in the collection of sampled gradients, can help enhance the practical performance of AGS methods.

Acknowledgements

Frank E. Curtis was supported by National Science Foundation grant DMS-1016291. X. Que was supported by National Science Foundation grant DMS-1016291.

References

[1] K.M. Anstreicher and J. Lee, *A masked spectral bound for maximum-entropy sampling*, in *MODA 7 – Advances in Model-Oriented Design and Analysis*, Contributions to Statistics, Springer, Berlin, 2004, pp. 1–10.
[2] C.G. Broyden, *The convergence of a class of double-rank minimization algorithms*, J. Inst. Math. Appl. 6 (1970), pp. 76–90.
[3] J.V. Burke, A.S. Lewis, and M.L. Overton, *Approximating subdifferentials by random sampling of gradients*, Math. Oper. Res. 27 (2002), pp. 567–584.
[4] J.V. Burke, A.S. Lewis, and M.L. Overton, *A robust gradient sampling algorithm for nonsmooth, nonconvex optimization*, SIAM J. Optim. 15 (2005), pp. 751–779.
[5] J.V. Burke, A.S. Lewis, and M.L. Overton, *Pseudospectral components and the distance to uncontrollability*, SIAM J. Matrix Anal. Appl. 26 (2005), pp. 350–361.
[6] J.V. Burke, D. Henrion, A.S. Lewis, and M.L. Overton, *HIFOO – A MATLAB package for fixed-order controller design and H-infinity optimization*, Proceedings of the IFAC Symposium on Robust Control Design, Haifa, Israel, 2006.
[7] F.H. Clarke, *Optimization and Nonsmooth Analysis*, Canadian Mathematical Society Series of Monographs and Advanced Texts, Wiley, New York, 1983.
[8] F.E. Curtis and M.L. Overton, *A sequential quadratic programming method for nonconvex, nonsmooth constrained optimization*, SIAM J. Optim. 22(2) (2012), pp. 474–500.
[9] E. Dolan and J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program. 91 (2002), pp. 201–213.
[10] R. Fletcher, *A new approach to variable metric algorithms*, Comput. J. 13 (1970), pp. 317–322.
[11] D. Goldfarb, *A family of variable metric updates derived by variational means*, Math. Comput. 24 (1970), pp. 23–26.
[12] A.A. Goldstein, *Optimization of Lipschitz continuous functions*, Math. Program. 13 (1977), pp. 14–22.
[13] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[14] C. Greif and J. Varah, *Minimizing the condition number for small rank modifications*, SIAM J. Matrix Anal. Appl. 29 (2006), pp. 82–97.
[15] A.M. Gupal, *On a minimization method for almost-differentiable functions*, Cybernetics 13 (1977), pp. 115–117.
[16] M. Haarala, *Large-scale nonsmooth optimization: Variable metric bundle method with limited memory*, Ph.D. thesis, University of Jyväskylä, 2004.
[17] M. Haarala, K. Miettinen, and M.M. Mäkelä, *New limited memory bundle method for large-scale nonsmooth optimization*, Optim. Methods Softw. 19 (2004), pp. 673–692.
[18] J.B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms II*, A Series of Comprehensive Studies in Mathematics, Springer, New York, 1993.
[19] R.A. Horn and C.R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, MA, 1990.
[20] K.C. Kiwiel, *Methods of Descent for Nondifferentiable Optimization*, Lecture Notes in Mathematics, Springer, New York, 1985.
[21] K.C. Kiwiel, *A method for solving certain quadratic programming problems arising in nonsmooth optimization*, IMA J. Numer. Anal. 6 (1986), pp. 137–152.
[22] K.C. Kiwiel, *Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization*, SIAM J. Optim. 18 (2007), pp. 379–388.
[23] K.C. Kiwiel, *A nonderivative version of the gradient sampling algorithm for nonsmooth nonconvex optimization*, SIAM J. Optim. 20 (2010), pp. 1983–1994.
[24] A.S. Lewis, *Local structure and algorithms in nonsmooth optimization*, in *Optimization and Applications*, F. Jarre, C. Lemaréchal, and J. Zowe, eds., Mathematisches Forschungsinstitut Oberwolfach, Oberwolfach, Germany, 2005, pp. 104–106.
[25] A.S. Lewis and M.L. Overton, *Nonsmooth optimization via quasi-Newton methods*, Math. Program. Series A (in press). Doi: 10.1007/s10107-012-0514-2.
[26] L. Lukšan, M. Tůma, M. Šiška, J. Vlček, and N. Ramešová, *UFO 2002: Interactive system for universal functional optimization*, Tech. Rep. 883, Institute of Computer Science, Academy of Sciences of the Czech Republic, 2002.
[27] J. Nocedal, *Updating quasi-newton matrices with limited storage*, Math. Comput. 35 (1980), pp. 773–782.
[28] J. Nocedal and S.J. Wright, *Numerical Optimization*, 2nd ed., Springer, Berlin, 2006.
[29] R.T. Rockafellar, *Convex Analysis*, Princeton Landmarks in Mathematics and Physics, Princeton University Press, Princeton, NJ, 1970.

[30] D.F. Shanno, *Conditioning of quasi-newton methods for function minimization*, Math. Comput. 24 (1970), pp. 647–656.
[31] A. Skajaa, *Limited memory BFGS for nonsmooth optimization*, Master's thesis, New York University, 2010.
[32] P. Wolfe, *A method of conjugate subgradients for minimizing nondifferentiable functions*, Math. Program. Study 3 (1975), pp. 145–173.

Appendix: QO subproblem solver

In this appendix, we discuss a specialized technique for solving the primal–dual pair (3) and (4) during step 4 of AGS. Specifically, we present an approach for solving (4) that follows the technique described in [21] for solving a similar subproblem. The differences are that, in our subproblem, there is no linear term in the objective, and we allow for the use of general positive-definite Hessian approximations. We drop iteration number subscripts in this section. Now, subscripts are used to indicate column number of a matrix or element number(s) in a vector.

The benefits of our QO solver are that it can produce more accurate solutions than, say, an interior-point method, and we can easily warm-start the approach to take advantage of the fact that the columns of G often do not change drastically between iterations of AGS. The algorithm also carefully handles ill-conditioning in G, which is extremely important in our context as the columns of G come from the calculation of gradients of f at points that may be very close to one another.

By (25), we can write necessary and sufficient conditions for (4) as

$$g_j^{\mathrm{T}} WG\pi - (\pi^{\mathrm{T}} G^{\mathrm{T}} WG\pi) \geq 0, \quad j = 1, \dots, q, \tag{A.1a}$$

$$e^{\mathrm{T}}\pi = 1, \quad \pi \geq 0. \tag{A.1b}$$

A vector π satisfying these conditions is the unique optimal solution to (4), with which the unique optimal solution d to (3) can be computed as $d \leftarrow -WG\pi$.

It is clear from (A.1b) that any solution π to (A.1) must have at least one positive entry. Thus, the method commences with a non-empty estimate $\mathcal{A} \subseteq \{1, \dots, q\}$ of the optimal *positive set*, that is, the indices corresponding to positive values of π in the solution to (A.1). Denoting $\hat{G}$ and $\hat{\pi}$, respectively, as the ordered submatrix of G and the ordered subvector of π corresponding to the indices in the positive-set estimate $\mathcal{A}$, we begin with $\hat{\pi} \geq 0$ solving

$$\min_{\pi} \ \tfrac{1}{2}\pi^{\mathrm{T}} \hat{G}^{\mathrm{T}} W \hat{G}\pi \quad \text{s.t. } e^{\mathrm{T}}\pi = 1, \tag{A.2}$$

that is, $\hat{\pi} \geq 0$ where $(\hat{\pi}, \hat{v})$ is the unique solution to

$$\begin{bmatrix} \hat{G}^{\mathrm{T}} W \hat{G} & e \\ e^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \pi \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{A.3}$$

(If for a given $\mathcal{A}$ the solution $\hat{\pi}$ of (A.2) does not satisfy $\hat{\pi} \geq 0$, then $\mathcal{A}$ is replaced by $\{i\}$ for some $i \in \{1, \dots, q\}$ and (A.2) is re-solved to obtain $\hat{\pi}$ with $\hat{\pi}_i = 1$ and $\hat{\pi}_{\{1,\dots,q\}\backslash\{i\}} = 0$.) Assuming always that the elements of the solution corresponding to $\{1, \dots, q\}\backslash\mathcal{A}$ are set to zero, this solution is either optimal or, for some $j \notin \mathcal{A}$,

$$g_j^{\mathrm{T}} W\hat{G}\hat{\pi} - (\hat{\pi}^{\mathrm{T}} \hat{G}^{\mathrm{T}} W \hat{G}\hat{\pi}) < 0. \tag{A.4}$$

An improvement in the objective of (4) can then be obtained by including j in $\mathcal{A}$. If the direct inclusion of j in $\mathcal{A}$ yields a new $\hat{G}$ such that $[e \ \hat{G}^{\mathrm{T}}]$ has full row rank, then $\mathcal{A}$ is simply augmented to include j. (Determining whether or not this matrix has full row rank can be done by solving the least-squares system

$$(\hat{G}^{\mathrm{T}} W \hat{G} + ee^{\mathrm{T}})\tilde{\pi} = e + \hat{G}^{\mathrm{T}} W g_j \tag{A.5}$$

and then determining whether $e^{\mathrm{T}}\tilde{\pi} = 1$ and $\hat{G}\tilde{\pi} = g_j$.) Otherwise, j is swapped with an appropriate element in $\mathcal{A}$ to avoid rank-deficiency. In either case, a new trial solution $(\bar{\pi}, \bar{v})$ is obtained by solving (A.3). If $\bar{\pi} > 0$, then $\hat{\pi} \leftarrow \bar{\pi}$ becomes the new solution estimate and the above procedures are repeated. Otherwise, a step from $\hat{\pi}$ in the direction of $\bar{\pi}$ is made until some element hits zero (say, corresponding to the $\bar{j}$th column of G), in which case $\bar{j}$ is removed from $\mathcal{A}$ and (A.3) is reformulated and re-solved for the new positive-set estimate.

A complete description of our subproblem solver, ASQO, is presented as Algorithm A.1 on page 1324. The algorithm returns a vector $\hat{\pi}$ corresponding to $\mathcal{A}$. This vector is to be permuted and augmented with zeros in the appropriate entries to construct the optimal π from which the optimal primal solution d is obtained.

Algorithm A.1 Active-Set Quadratic Optimization Subproblem Solver (ASQO)

```
1: (Initialization) Choose $\mathcal{A}$ such that, with $\hat{G}$ as the submatrix of $G$ corresponding to the indices
   in $\mathcal{A}$, the solution $(\hat{\pi}, \hat{v})$ of (A.3) has $\hat{\pi} \geq 0$.
2: (Termination check) If (A.1a) holds, then terminate. Otherwise, choose an index $j \in
   \{1, \dots, q\} \backslash \mathcal{A}$ such that (A.4) holds.
3: (Rank-deficiency check) Solve (A.5) for $\tilde{\pi}$. If $\tilde{\pi}^{\mathrm{T}}[e\ \hat{G}^{\mathrm{T}}] = [1\ g_j^{\mathrm{T}}]$, then go to step 5; otherwise,
   continue.
4: (Column augmentation) Append $j$ to $\mathcal{A}$, $g_j$ to $\hat{G}$, and 0 to $\hat{\pi}$. Go to step 6.
5: (Column exchange) Replace $\hat{\pi}$ by $\hat{\pi} - t\tilde{\pi}$ where
       $$t = \min_i \{\hat{\pi}_i / \tilde{\pi}_i : \tilde{\pi}_i > 0\}.$$
   Find some $i$ such that $\hat{\pi}_i = 0$. Delete the $i$th index from $\mathcal{A}$, the $i$th column from $\hat{G}$, and the
   $i$th component from $\hat{\pi}$. Append $j$ to $\mathcal{A}$, $g_j$ to $\hat{G}$, and $t$ to $\hat{\pi}$.
6: (Subproblem solution) Solve (A.3) for $(\bar{\pi}, \bar{v})$. If $\bar{\pi} > 0$, then set $\hat{\pi} = \bar{\pi}$ and go to step 2;
   otherwise, continue.
7: (Column deletion) Replace $\hat{\pi}$ by $t\bar{\pi} + (1-t)\hat{\pi}$ where
       $$t = \min\left\{1, \min_i \{\hat{\pi}_i / (\hat{\pi}_i - \bar{\pi}_i) : \bar{\pi}_i < 0\}\right\}.$$
   Find $i$ such that $\hat{\pi}_i = 0$. Delete the $i$th index from $\mathcal{A}$, the $i$th column from $\hat{G}$, and the $i$th
   component from $\hat{\pi}$. Go to step 6.
```

Our implementation of ASQO actually maintains a Cholesky factorization of $(\hat{G}^{\mathrm{T}} W \hat{G} + ee^{\mathrm{T}})$ that is updated during each iteration. Specifically, we maintain an upper triangular matrix R satisfying

$$R^{\mathrm{T}} R = \hat{G}^{\mathrm{T}} W \hat{G} + ee^{\mathrm{T}},$$

with which it can easily be verified that the solutions to (A.3) and (A.5) can be obtained, respectively, by solving

$$\left\{\begin{array}{c} R^{\mathrm{T}} r_1 = e + \hat{G}^{\mathrm{T}} W g_j \\ R\bar{\pi} = r_1 \end{array}\right\} \quad \text{and} \quad \left\{\begin{array}{c} R^{\mathrm{T}} r_2 = e \\ R\tilde{\pi} = \dfrac{r_2}{\|r_2\|^2} \end{array}\right\}.$$

The maintenance of R and calculation of the intermediate vectors above allows for sophisticated extensions of the rank-deficiency check in step (3) of ASQO so that it is less susceptible to numerical errors. We have implemented these extensions in our code, but suppress the details here as they are out of the scope of this paper; see [21] for details.