Comput Optim Appl (2013) 56:1–38
DOI 10.1007/s10589-013-9547-6

A derivative-free approximate gradient sampling algorithm for finite minimax problems

W. Hare · J. Nutini

Received: 6 February 2012 / Published online: 1 March 2013

Abstract In this paper we present a derivative-free optimization algorithm for finite minimax problems. The algorithm calculates an approximate gradient for each of the active functions of the finite max function and uses these to generate an approximate subdifferential. The negative projection of 0 onto this set is used as a descent direction in an Armijo-like line search. We also present a robust version of the algorithm, which uses the 'almost active' functions of the finite max function in the calculation of the approximate subdifferential. Convergence results are presented for both algorithms, showing that either $f(x^k) \to -\infty$ or every cluster point is a Clarke stationary point. Theoretical and numerical results are presented for three specific approximate gradients: the simplex gradient, the centered simplex gradient and the Gupal estimate of the gradient of the Steklov averaged function. A performance comparison is made between the regular and robust algorithms, the three approximate gradients, and a regular and robust stopping condition.

Keywords Derivative-free optimization · Minimax problems · Generalized gradient · Subgradient approximation

1 Introduction

In this paper we consider the finite minimax problem:

$$\min_x f(x) \quad \text{where } f(x) = \max\{f_i(x) : i = 1, \ldots, N\},$$

where each individual f_i is continuously differentiable. We further restrict ourselves to the field of derivative-free optimization (DFO), where we are only permitted to

W. Hare (✉) · J. Nutini
University of British Columbia, Kelowna, BC, Canada
e-mail: warren.hare@ubc.ca

compute function values, i.e., we cannot compute gradient values ∇f_i directly. We present a derivative-free algorithm that exploits the smooth substructure of the finite max problem, thereby creating a robust algorithm with an elegant convergence theory.

Finite minimax problems occur in numerous applications, such as portfolio optimization [8], control system design [21], engineering design [32], and determining the cosine measure of a positive spanning set [10, Def. 2.7]. In a finite max function, although each individual f_i may be smooth, taking the maximum forms a nonsmooth function with 'nondifferentiable ridges'. For this reason, most algorithms designed to solve finite minimax problems employ some form of smoothing technique; [31, 33, 34] and [39] (among many others). In general, these smoothing techniques require gradient calculations.

However, in many situations gradient information is not available or can be difficult to compute accurately (see [4, 15, 20, 29] and [10, Chap. 1] for some examples of such situations). Such situations are considered by research in the area of derivative-free optimization. For a thorough introduction to several basic DFO frameworks and convergence results for each, see [10].

Research on optimizing finite max functions without calculating derivatives can be seen as early as 1975 [28], while more recently we have seen a resurface in this area [26] and [19].

In 2006, Liuzzi, Lucidi and Sciandrone used a smoothing technique based on an exponential penalty function in a directional direct-search framework to form a derivative-free optimization method for finite minimax problems [26]. This method is shown to globally converge towards a standard stationary point of the original finite minimax problem.

Also specific to the finite minimax problem, a derivative-free method is presented in [19] that exploits the smooth substructure of the problem. It combines the frameworks of a directional direct search method [10, Chap. 7] and the gradient sampling algorithm (GS algorithm) presented in [6] and [7]. Loosely speaking, the GS algorithm uses a collection of local gradients to build a 'robust subdifferential' of the objective function and uses this to determine a 'robust descent direction'. In [19], these ideas are used to develop several methods to find an approximate descent direction that moves close to parallel to an 'active manifold'. During each iteration, points are sampled from around the current iterate and the simplex gradient is calculated for each of the active functions of the objective function. The calculated simplex gradients are then used to form an approximate subdifferential, which is then used to determine a likely descent direction.

Ideas from the GS algorithm have appeared in two other recent DFO methodologies [2] and [24].

In 2008, Bagirov, Karasözen and Sezer presented a discrete gradient derivative-free method for unconstrained nonsmooth optimization problems [2]. Described as a derivative-free version of the bundle method presented in [37], the method uses discrete gradients to approximate subgradients of the function and build an approximate subdifferential. The analysis of this method provides proof of convergence to a Clarke stationary point for an extensive class of nonsmooth problems. In this paper, we focus on the finite minimax problem. This allows us to require few (other) assumptions on our function while maintaining strong convergence analysis. It is worth noting that we use the same set of test problems as in [2]. Specifically, we use the [27] test set and exclude one problem as its sub-functions are complex-valued. (The numerics in [2] exclude the same problem, and several others, without explanation.)

Using approximate gradient calculations instead of gradient calculations, the GS algorithm is made derivative free by Kiwiel in [24]. Specifically, Kiwiel employs the *Gupal estimate* of the gradient of the *Steklov averaged function* (see [18] or Sect. 4.3 herein) as an approximate gradient. It is shown that, with probability 1, this derivative-free algorithm satisfies the same convergence results as the GS algorithm—it either drives the f-values to $-\infty$ or each cluster point is found to be Clarke stationary [24, Theorem 3.8]. No numerical results are presented for Kiwiel's derivative-free algorithm.

In this paper, we use the GS algorithm framework with approximate gradients to form a derivative-free approximate gradient sampling algorithm. As we are dealing with finite max functions, instead of calculating an approximate gradient at each of the sampled points, we calculate an approximate gradient for each of the active functions. Expanding the active set to include 'almost' active functions, we also present a robust version of our algorithm, which is more akin to the GS algorithm. In this robust version, when our iterate is close to a point of nondifferentiability, the size and shape of our approximate subdifferential will reflect the presence of 'almost active' functions. Hence, when we project 0 onto our approximate subdifferential, the descent direction will direct minimization parallel to a 'nondifferentiable ridge', rather than straight at this ridge. It can be seen in our numerical results that these robust changes greatly influence the performance of our algorithm.

Our algorithm differs from the above in a few key manners. Unlike in [26] we do not employ a smoothing technique. Unlike in [19], which uses the directional direct-search framework to imply convergence, we employ an approximate steepest descent framework. Using this framework, we are able to analyze convergence directly and develop stopping conditions for the algorithm. Unlike in [2] and [24], where convergence is proven for a specific approximate gradient, we prove convergence for *any* approximate gradient that satisfies a simple error bound dependent on the sampling radius. As examples, we present the simplex gradient, the centered simplex gradient and the Gupal estimate of the gradient of the Steklov averaged function. (As a side-note, Sect. 4.3 also provides, to the best of the authors' knowledge, novel error analysis of the Gupal estimate of the gradient of the Steklov averaged function.)

Focusing on the finite minimax problem provides us with an advantage over the methods of [2] and [24]. In particular, we only require order n function calls per iteration (where n is the dimension of the problem), while both [2] and [24] require order mn function calls per iteration (where m is the number of gradients they approximate to build their approximate subdifferential). (The original GS algorithm suggests that $m \approx 2n$ provides a good value for m.)

The remainder of this paper is organized as follows. In Sect. 2, we present the approximate gradient sampling algorithm (AGS algorithm) and our convergence analysis. In Sect. 3, we present a robust version of the AGS algorithm (RAGS algorithm), which uses 'almost active' functions in the calculation of the approximate subdifferential. In Sect. 4, we show that the AGS and RAGS algorithms converge using three specific approximate gradients: simplex gradient, centered simplex gradient and the

Gupal estimate of the gradient of the Steklov averaged function. Finally, in Sect. 5, we present our numerical results and analysis.

2 Approximate gradient sampling algorithm

Throughout this paper, we assume that our objective function is of the form

$$\min_x f(x) \quad \text{where } f(x) = \max\{f_i(x) : i = 1, \dots, N\}, \tag{1}$$

where each $f_i \in \mathcal{C}^1$, but we cannot compute ∇f_i. We use $\mathcal{C}^1$ to denote the class of differentiable functions whose gradient mapping ∇f is continuous. We denote by $\mathcal{C}^{1+}$ the class of continuously differentiable functions whose gradient mapping ∇f is locally Lipschitz and we denote by $\mathcal{C}^{2+}$ the class of twice continuously differentiable functions whose Hessian mapping $\nabla^2 f$ is locally Lipschitz. Additionally, throughout this paper, $|\cdot|$ denotes the Euclidean norm and $\|\cdot\|$ denotes the corresponding matrix norm.

For the finite max function in (1), we define the **active set** of f at a point $\bar{x}$ to be the set of indices

$$A(\bar{x}) = \{i : f(\bar{x}) = f_i(\bar{x})\}.$$

The set of **active gradients** of f at $\bar{x}$ is denoted by

$$\{\nabla f_i(\bar{x})\}_{i \in A(\bar{x})}.$$

Let f be locally Lipschitz at a point $\bar{x}$. As f is Lipschitz, there exists an open dense set $D \subset \mathbb{R}^n$ such that f is continuously differentiable on D. The Clarke subdifferential [9] is constructed via

$$\partial f(x) = \bigcap_{\varepsilon > 0} G_\varepsilon(x) \quad \text{where } G_\varepsilon(x) = \operatorname{cl}\operatorname{conv}\{\nabla f(y) : y \in B_\varepsilon(x) \cap D\}.$$

For a finite max function, assuming $f_i \in \mathcal{C}^1$ for each $i \in A(\bar{x})$, the Clarke subdifferential (as proven in [9, Prop. 2.3.12]) is equivalent to

$$\partial f(\bar{x}) = \operatorname{conv}\{\nabla f_i(\bar{x})\}_{i \in A(\bar{x})}. \tag{2}$$

By (2), it is clear that for finite max functions the subdifferential is a compact set. This will be important in the convergence analysis in Sect. 2.2.

We are now ready to state the general form of the AGS algorithm, an approximate subgradient descent method.

2.1 Algorithm—AGS

We first provide a partial glossary of notation used in the definition of the AGS algorithm (see Table 1).

Table 1 Glossary of notation used in the AGS algorithm

Glossary of notation	
k: Iteration counter	x^k: Current iterate
μ^k: Accuracy measure	Δ^k: Sampling radius
m: Sample size	θ: Sampling radius reduction factor
y^j: Sampling points	Y: Sampled set of points
η: Armijo-like parameter	d^k: Search direction
t^k: Step length	t^{min}: Minimum step length
$\nabla_A f_i$: Approximate gradient of f_i	$A(x^k)$: Active set at x^k
G^k: Approximate subdifferential	ε_{tol}: Stopping tolerance

Conceptual Algorithm: [Approximate Gradient Sampling Algorithm]

0. INITIALIZE: Set $k = 0$ and input

   x^0—starting point
   $\mu^0 > 0$—accuracy measure
   $\Delta^0 > 0$—initial sampling radius
   $\theta \in (0, 1)$—sampling radius reduction factor
   $0 < \eta < 1$—Armijo-like parameter
   t^{min}—minimum step length
   $\varepsilon_{tol} > 0$—stopping tolerance

1. GENERATE APPROXIMATE SUBDIFFERENTIAL G^k:
   Generate a set $Y = [x^k, y^1, \ldots, y^m]$ around the current iterate x^k such that

$$\max_{j=1,\ldots,m} \left|y^j - x^k\right| \le \Delta^k.$$

   Use Y to calculate the approximate gradient of f_i, denoted $\nabla_A f_i$, at x^k for each $i \in A(x^k)$. Set

$$G^k = \operatorname{conv}\left\{\nabla_A f_i\left(x^k\right)\right\}_{i\in A(x^k)}.$$

2. GENERATE SEARCH DIRECTION:
   Let

$$d^k = -\operatorname{Proj}\left(0|G^k\right).$$

   Check if

$$\Delta^k \le \mu^k \left|d^k\right|. \tag{3}$$

   If (3) does not hold, then set $x^{k+1} = x^k$,

$$\Delta^{k+1} = \begin{cases} \theta\mu^k|d^k| & \text{if } |d^k| \ne 0,\\ \theta\Delta^k & \text{if } |d^k| = 0,\end{cases} \tag{4}$$

$k = k + 1$ and return to Step 1. If (3) holds and $|d^k| < \varepsilon_{tol}$, then STOP. Else, continue to the line search.

3. LINE SEARCH:
   Attempt to find $t^k > 0$ such that
   $$f(x^k + t^k d^k) < f(x^k) - \eta t^k |d^k|^2.$$
   LINE SEARCH FAILURE:
   Set $\mu^{k+1} = \frac{\mu^k}{2}$, $x^{k+1} = x^k$ and go to Step 4.
   LINE SEARCH SUCCESS:
   Let x^{k+1} be any point such that
   $$f(x^{k+1}) \le f(x^k + t^k d^k).$$
4. UPDATE AND LOOP:
   Set $\Delta^{k+1} = \max_{j=1,\dots,m} |y^j - x^k|$, $k = k + 1$ and return to Step 1.

In Step 0 of the AGS algorithm, we set the iterate counter to 0, provide an initial starting point x^0, and initialize the parameter values.

In Step 1, we create the approximate subdifferential. First, we select a set of points around x^k within a sampling radius of Δ^k. In implementation, the points are randomly and uniformly sampled from a ball of radius Δ^k (using the MATLAB `randsphere.m` function [36]). Using this set Y, we then calculate an approximate gradient for each of the active functions at x^k and set the approximate subdifferential G^k equal to the convex hull of these active approximate gradients, $\nabla_A f_i(x^k)$. Details on various approximate gradients appear in Sect. 4.

In Step 2, we generate a search direction by solving the projection of 0 onto the approximate subdifferential: $\text{Proj}(0|G^k) \in \arg\min_{g \in G^k} |g|^2$. The search direction d^k is set equal to the negative of the solution, i.e., $d^k = -\text{Proj}(0|G^k)$.

After finding a search direction, we check the inequality $\Delta^k \le \mu^k |d^k|$. This inequality determines if the current sampling radius is sufficiently small relative to the distance from 0 to the approximate subdifferential. If this inequality holds and $|d^k| < \varepsilon_{tol}$, then we terminate the algorithm, as 0 is within ε_{tol} of the approximate subdifferential and the sampling radius is small enough to reason that the approximate subdifferential is accurate. If the above inequality does not hold, then the approximate subdifferential is not sufficiently accurate to warrant a line search, so we decrease the sampling radius, set $x^{k+1} = x^k$, update the iterate counter and loop (Step 1). If the above inequality holds, but $|d^k| \ge \varepsilon_{tol}$, then we proceed to a line search.

In Step 3, we carry out a line search. We attempt to find a step length $t^k > 0$ such that the Armijo-like condition holds
$$f(x^k + t^k d^k) < f(x^k) - \eta t^k |d^k|^2. \tag{5}$$
This condition ensures sufficient decrease is found in the function value. In implementation, we use a back-tracking line search (described in [30]) with an initial step-length of $t^{ini} = 1$, terminating when the step length t^k is less than a threshold t^{min}.

If we find a t^k such that (5) holds, then we declare a line search success. If not, then we declare a line search failure.

If a line search success occurs, then we let x^{k+1} be any point such that

$$f\left(x^{k+1}\right) \leq f\left(x^k + t^k d^k\right). \tag{6}$$

In implementation, we do this by searching through the function values used in the calculation of our approximate gradients ($\{f(y^i)\}_{y^i \in Y}$). As this set of function values corresponds to points distributed around our current iterate, there is a good possibility of finding further function value decrease without having to carry out additional function evaluations. We find the minimum function value in our set of evaluations and if (6) holds for this minimum value, then we set x^{k+1} equal to the corresponding input point. Otherwise, we set $x^{k+1} = x^k + t^k d^k$.

If a line search failure occurs, then we reduce the accuracy measure μ^k by a factor of $\frac{1}{2}$ and set $x^{k+1} = x^k$.

Finally, in Step 4, we update the iterate counter and the sampling radius, and then loop to Step 1 to resample.

2.2 Convergence

For the following results, we denote the approximate subdifferential of f at $\bar{x}$ as

$$G(\bar{x}) = \operatorname{conv}\left\{\nabla_A f_i(\bar{x})\right\}_{i \in A(\bar{x})},$$

where $\nabla_A f_i(\bar{x})$ is the approximate gradient of f_i at $\bar{x}$. Our first result establishes an error bound relation between the elements of the approximate subdifferential and the exact subdifferential.

Lemma 2.1 *Let $f = \max\{f_i : i = 1, \ldots, N\}$ where each $f_i \in \mathcal{C}^1$. Suppose there exists an $\varepsilon > 0$ such that $|\nabla_A f_i(\bar{x}) - \nabla f_i(\bar{x})| \leq \varepsilon$ for all $i = 1, \ldots, N$. Then*

1. *for all $w \in G(\bar{x})$, there exists a $v \in \partial f(\bar{x})$ such that $|w - v| \leq \varepsilon$, and*
2. *for all $v \in \partial f(\bar{x})$, there exists a $w \in G(\bar{x})$ such that $|w - v| \leq \varepsilon$.*

Proof 1. By definition, for all $w \in G(\bar{x})$ there exists a set of α_i such that

$$w = \sum_{i \in A(\bar{x})} \alpha_i \nabla_A f_i(\bar{x}), \quad \text{where } \alpha_i \geq 0, \sum_{i \in A(\bar{x})} \alpha_i = 1.$$

By our assumption that each $f_i \in \mathcal{C}^1$, we have $\partial f(\bar{x}) = \operatorname{conv}\{\nabla f_i(\bar{x})\}_{i \in A(\bar{x})}$. Using the same α_i as above, we see that

$$v = \sum_{i \in A(\bar{x})} \alpha_i \nabla f_i(\bar{x}) \in \partial f(\bar{x})$$

Then

$$\begin{aligned}|w-v| &= \left|\sum_{i\in A(\bar{x})} \alpha_i \nabla_A f_i(\bar{x}) - \sum_{i\in A(\bar{x})} \alpha_i \nabla f_i(\bar{x})\right| \\ &\leq \sum_{i\in A(\bar{x})} \alpha_i \left|\nabla_A f_i(\bar{x}) - \nabla f_i(\bar{x})\right| \\ &\leq \sum_{i\in A(\bar{x})} \alpha_i \varepsilon \\ &= \varepsilon\end{aligned}$$

Hence, for all $w \in G(\bar{x})$, there exists a $v \in \partial f(\bar{x})$ such that

$$|w - v| \leq \varepsilon. \tag{7}$$

2. Analogous arguments can be applied to $v \in \partial f(\bar{x})$. □

Lemma 2.1 states the quality of the approximate subdifferential as an approximation to the exact subdifferential once the approximate gradients of the component functions are quality approximations to the real gradients. Our next goal (in Theorem 2.4) is to show that eventually a line search success will occur in the AGS algorithm. To achieve this we make use of the following lemma.

Lemma 2.2 *Let* $f = \max\{f_i : i = 1, \ldots, N\}$ *where each* $f_i \in \mathcal{C}^1$. *Suppose there exists an* $\varepsilon > 0$ *such that* $|\nabla_A f_i(\bar{x}) - \nabla f_i(\bar{x})| \leq \varepsilon$ *for all* $i = 1, \ldots, N$. *Define* $d = -\operatorname{Proj}(0|G(\bar{x}))$ *and suppose* $|d| \neq 0$. *Let* $\beta \in [0, 1)$. *If* $\varepsilon < (1-\beta)|d|$, *then for all* $v \in \partial f(\bar{x})$ *we have*

$$\langle d, v\rangle < -\beta|d|^2.$$

Proof Notice that, by the Projection Theorem [3, Theorem 3.14], $d = -\operatorname{Proj}(0|G(\bar{x}))$ implies that

$$\big\langle 0 - (-d), w - (-d)\big\rangle \leq 0 \quad \text{for all } w \in G(\bar{x}).$$

Hence,

$$\langle d, w + d\rangle \leq 0 \quad \text{for all } w \in G(\bar{x}). \tag{8}$$

So we have for all $v \in \partial f(\bar{x})$,

$$\begin{aligned}\langle d, v\rangle &= \langle d, v - w + w - d + d\rangle && \text{for all } w \in G(\bar{x}) \\ &= \langle d, v - w\rangle + \langle d, w + d\rangle + \langle d, -d\rangle && \text{for all } w \in G(\bar{x}) \\ &\leq \langle d, v - w\rangle - |d|^2 && \text{for all } w \in G(\bar{x}) \\ &\leq |d||v - w| - |d|^2 && \text{for all } w \in G(\bar{x}).\end{aligned}$$

For any $v \in \partial f(\bar{x})$, using w as constructed in Lemma 2.1, we see that

$$\begin{aligned}\langle d, v\rangle &\le |d|\varepsilon - |d|^2\\ &< |d|^2(1-\beta) - |d|^2 \quad \big(\text{as } \varepsilon < (1-\beta)|d|\big)\\ &= -\beta|d|^2. \qquad \square\end{aligned}$$

Remark 2.3 In Lemma 2.2, for the case when $\beta = 0$, the condition $\varepsilon < (1-\beta)|d|$ simplifies to $\varepsilon < |d|$. Thus, if ε is bounded above by $|d|$, then Lemma 2.2 proves that for all $v \in \partial f(\bar{x})$ we have $\langle d, v\rangle < 0$, showing that d is a descent direction for f at $\bar{x}$.

To guarantee convergence, we must show that, except in the case of $0 \in \partial f(x^k)$, the algorithm will always be able to find a sampling radius that satisfies the requirements in Step 2. In Sect. 4 we show that (for three different approximate gradients) the value ε (in Lemma 2.2) is linked to Δ. As unsuccessful line searches will drive Δ to zero, this implies that eventually the requirements of Lemma 2.2 will be satisfied. We formalize this in the next two theorems.

Theorem 2.4 *Let $f = \max\{f_i : i = 1, \dots, N\}$ where each $f_i \in \mathcal{C}^1$. Suppose $0 \notin \partial f(x^k)$ for each iteration k. Suppose there exists $\bar{K} > 0$ such that given any set of points generated in Step 1 of the AGS algorithm, the approximate gradient satisfies $|\nabla_A f_i(x^k) - \nabla f_i(x^k)| \le \bar{K}\Delta^k$ for all $i = 1, \dots, N$. Let $d^k = -\operatorname{Proj}(0|G(x^k))$. Then for any $\mu > 0$, there exists $\bar{\Delta} = \bar{\Delta}(x^k) > 0$ such that,*

$$\Delta \le \mu\left|d^k\right| + \bar{K}\mu\big(\Delta^k - \Delta\big) \quad \textit{for all } 0 < \Delta < \bar{\Delta}.$$

Moreover, if $\Delta^k < \bar{\Delta}$, then the following inequality holds

$$\Delta^k \le \mu\left|d^k\right|.$$

Proof Let $\bar{v} = \operatorname{Proj}(0|\partial f(x^k))$ (by assumption, $\bar{v} \ne 0$).

Given $\mu > 0$, let

$$\bar{\Delta} = \frac{1}{\bar{K} + \frac{1}{\mu}}|\bar{v}|, \tag{9}$$

and consider $0 < \Delta < \bar{\Delta}$. Now create $G(x^k)$ and $d^k = -\operatorname{Proj}(0|G(x^k))$. As $-d^k \in G(x^k)$, by Lemma 2.1(1), there exists a $v^k \in \partial f(x^k)$ such that

$$\left|-d^k - v^k\right| \le \bar{K}\Delta^k.$$

Then

$$\begin{aligned}&\bar{K}\Delta^k \ge \left|-d^k - v^k\right|\\ &\quad\Rightarrow \quad \bar{K}\Delta^k \ge \left|v^k\right| - \left|d^k\right|\\ &\quad\Rightarrow \quad \bar{K}\Delta^k \ge |\bar{v}| - \left|d^k\right| \quad \big(\text{as } |v| \ge |\bar{v}| \text{ for all } v \in \partial f\big(x^k\big)\big).\end{aligned}$$

Thus, for $0 < \Delta < \bar{\Delta}$, we apply (9) to $|\bar{v}|$ in the above inequality to get

$$\bar{K}\Delta^k \geq \left(\bar{K} + \frac{1}{\mu}\right)\Delta - \left|d^k\right|,$$

which rearranges to

$$\Delta \leq \mu\left|d^k\right| + \bar{K}\mu\left(\Delta^k - \Delta\right).$$

Hence, $\Delta \leq \mu|d^k| + \bar{K}\mu(\Delta^k - \Delta)$ for all $0 < \Delta < \bar{\Delta}$. Finally, if $\Delta^k < \bar{\Delta}$, then

$$\Delta^k \leq \mu\left|d^k\right|. \qquad \square$$

Remark 2.5 In Theorem 2.4, it is important to note that eventually the condition $\Delta^k < \bar{\Delta}$ will hold. Examine $\bar{\Delta}$ as constructed above: $\bar{K}$ is a constant and $\bar{v}$ is associated with the current iterate. However, the current iterate is only updated when a line search success occurs, which will not occur unless the condition $\Delta^k \leq \mu^k|d^k|$ is satisfied. As a result, if $\Delta^k \geq \bar{\Delta}$, the AGS algorithm will reduce Δ^k, with $\bar{\Delta}$ remaining constant, until $\Delta^k < \bar{\Delta}$.

Recall in Step 3 of the AGS algorithm, for a given $\eta \in (0, 1)$, we attempt to find a step length $t^k > 0$ such that

$$f\left(x^k + t^k d^k\right) < f\left(x^k\right) - \eta t^k\left|d^k\right|^2.$$

The following result shows that eventually the above inequality will hold in the AGS algorithm. Recall that the exact subdifferential for a finite max function, as defined in (2), is a compact set. Thus, we know that in the following theorem $\tilde{v}$ is well-defined.

Theorem 2.6 *Fix $0 < \eta < 1$. Let $f = \max\{f_i : i = 1, \ldots, N\}$ where each $f_i \in \mathcal{C}^1$. Suppose there exists an $\varepsilon > 0$ such that $|\nabla_A f_i(\bar{x}) - \nabla f_i(\bar{x})| \leq \varepsilon$ for all $i = 1, \ldots, N$. Define $d = -\operatorname{Proj}(0|G(\bar{x}))$ and suppose $|d| \neq 0$. Let $\tilde{v} \in \arg\max\{\langle d, v\rangle : v \in \partial f(\bar{x})\}$. Let $\beta = \frac{2\eta}{1+\eta}$. If $\varepsilon < (1-\beta)|d|$, then there exists $\bar{t} > 0$ such that*

$$f(\bar{x} + td) - f(\bar{x}) < -\eta t|d|^2 \quad \textit{for all } 0 < t < \bar{t}.$$

Proof Note that $\beta \in (0, 1)$. Recall, from Lemma 2.2, we have for all $v \in \partial f(\bar{x})$

$$\langle d, v\rangle < -\beta|d|^2. \tag{10}$$

Using $\beta = \frac{2\eta}{1+\eta}$, (10) becomes

$$\langle d, v\rangle < -\frac{2\eta}{1+\eta}|d|^2 \quad \text{for all } v \in \partial f(\bar{x}). \tag{11}$$

From (11) we can conclude that for all $v \in \partial f(\bar{x})$

$$\langle d, v\rangle < 0.$$

Notice that

$$\lim_{\tau \searrow 0} \frac{f(\bar{x}+\tau d)-f(\bar{x})}{\tau} = \max\big\{\langle d, v\rangle : v \in \partial f(\bar{x})\big\} = \langle d, \tilde{v}\rangle < 0.$$

Therefore, there exists $\bar{t} > 0$ such that

$$\frac{f(\bar{x}+td)-f(\bar{x})}{t} < \frac{\eta+1}{2}\langle d, \tilde{v}\rangle \quad \text{for all } 0 < t < \bar{t}.$$

For such a t, we have

$$\begin{aligned} f(\bar{x}+td)-f(\bar{x}) &< \frac{\eta+1}{2} t\langle d, \tilde{v}\rangle \\ &< -\frac{\eta+1}{2}\frac{2\eta}{\eta+1} t|d|^2 \\ &< -\eta t|d|^2. \end{aligned}$$

Hence,

$$f(\bar{x}+td)-f(\bar{x}) < -\eta t|d|^2 \quad \text{for all } 0 < t < \bar{t}. \qquad \square$$

Combining the previous results, we can show that the AGS algorithm is guaranteed to find function value decrease (provided $0 \notin \partial f(x^k)$). We summarize with the following corollary.

Corollary 2.7 *Let* $f = \max\{f_i : i = 1, \dots, N\}$ *where each* $f_i \in \mathcal{C}^1$. *Suppose* $0 \notin \partial f(x^k)$ *for each iteration* k. *Suppose there exists a* $\bar{K} > 0$ *such that given any set of points generated in Step* 1 *of the AGS algorithm, the approximate gradient satisfies* $|\nabla_A f_i(x^k) - \nabla f_i(x^k)| \le \bar{K}\Delta^k$ *for all* $i = 1, \dots, N$. *Then after a finite number of iterations, the algorithm will find a new iterate with a lower function value.*

Proof Consider x^k, where $0 \notin \partial f(x^k)$.

To find function value decrease with the AGS algorithm, we must declare a line search success in Step 3. The AGS algorithm will only carry out a line search if the condition below is satisfied

$$\Delta^k \le \mu^k |d^k|, \tag{12}$$

where $d^k = -\operatorname{Proj}(0|G(x^k))$, as usual. In Theorem 2.4, we showed that for any $\mu^k > 0$, there exists a $\bar{\Delta} = \bar{\Delta}(x^k) > 0$ such that if $\Delta^k < \bar{\Delta}(x^k)$, then (12) is satisfied. If (12) is not satisfied, then Δ^k is updated according to (4) and $x^{k+1} = x^k$, which further implies $\bar{\Delta} = \bar{\Delta}(x^{k+1}) = \bar{\Delta}(x^k)$ is unchanged. In this case, whether $|d^k| \neq 0$ or $|d^k| = 0$, we can see that $\Delta^{k+1} \le \theta\Delta^k$. Hence an infinite sequence of (12) being unsatisfied is impossible (as eventually we would have $\Delta^k < \bar{\Delta}$). So eventually (12) will be satisfied and the AGS algorithm will carry out a line search.

Now, in order to have a line search success, we must be able to find a step length t^k such that the Armijo-like condition holds,

$$f\big(x^k + t^k d^k\big) < f\big(x^k\big) - \eta t^k |d^k|^2.$$

In Theorem 2.6, we showed that there exists $\bar{t} > 0$ such that

$$f\left(x^k + t^k d^k\right) - f\left(x^k\right) < -\eta t^k \left|d^k\right|^2 \quad \text{for all } 0 < t^k < \bar{t},$$

provided that for $\beta \in (0, 1)$,

$$\varepsilon < (1-\beta)\left|d^k\right|. \tag{13}$$

Set $\varepsilon = \bar{K}\Delta^k$. If (13) does not hold, then a line search failure will occur, resulting in $\mu^{k+1} = 0.5\mu^k$. Thus, eventually we will have $\mu^k < \frac{(1-\beta)}{\bar{K}}$ and

$$\Delta^k \leq \mu^k \left|d^k\right| < \frac{(1-\beta)}{\bar{K}}\left|d^k\right|,$$

which means (13) will hold. Thus, after a finite number of iterations, the AGS algorithm will declare a line search success and find a new iterate with a lower function value. □

We are now ready to prove convergence. In particular, we study the limiting case of the algorithm generating an infinite sequence (i.e., the situation with $\varepsilon_{tol} = 0$). In the following, assuming that the step length t^k is bounded away from 0 means that there exists a $\bar{t} > 0$ such that $t^k > \bar{t}$.

Theorem 2.8 *Let $f = \max\{f_i : i = 1, \ldots, N\}$ where each $f_i \in \mathcal{C}^1$. Set $\varepsilon_{tol} = 0$ and suppose that $\{x^k\}_{k=0}^{\infty}$ is an infinite sequence generated by the AGS algorithm. Suppose there exists a $\bar{K} > 0$ such that given any set of points generated in Step 1 of the AGS algorithm, the approximate gradient satisfies the error bound $|\nabla_A f_i(x^k) - \nabla f_i(x^k)| \leq \bar{K}\Delta^k$ for all $i = 1, \ldots, N$. Suppose t^k is bounded away from 0. Then either*

1. $f(x^k) \downarrow -\infty$, *or*
2. $|d^k| \to 0$, $\Delta^k \downarrow 0$ *and every cluster point $\bar{x}$ of the sequence $\{x^k\}_{k=0}^{\infty}$ satisfies $0 \in \partial f(\bar{x})$.*

Proof If $f(x^k) \downarrow -\infty$, then we are done.

Conversely, if $f(x^k)$ is bounded below, then $f(x^k)$ is non-increasing and bounded below, therefore $f(x^k)$ converges. We consider two cases.

Case 1: An infinite number of line search successes occur.

Let $\bar{x}$ be a cluster point of $\{x^k\}_{k=0}^{\infty}$. Notice that x^k only changes for line search successes, so there exists a subsequence $\{x^{k_j}\}_{j=0}^{\infty}$ of line search successes such that $x^{k_j} \to \bar{x}$. Then for each corresponding step length t^{k_j} and direction d^{k_j}, the following condition holds

$$f\left(x^{k_j+1}\right) \leq f\left(x^{k_j} + t^{k_j} d^{k_j}\right) < f\left(x^{k_j}\right) - \eta t^{k_j} \left|d^{k_j}\right|^2.$$

Note that

$$0 \leq \eta t^{k_j} \left|d^{k_j}\right|^2 < f\left(x^{k_j}\right) - f\left(x^{k_j+1}\right).$$

Since $f(x^k)$ converges we know that $f(x^{k_j}) - f(x^{k_j+1}) \to 0$. Since t^{k_j} is bounded away from 0, we see that

$$\lim_{j\to\infty} \left|d^{k_j}\right| = 0.$$

Recall from the AGS algorithm, we check the condition

$$\Delta^{k_j} \leq \mu^{k_j} \left|d^{k_j}\right|.$$

As $\Delta^{k_j} > 0$, $\mu^{k_j} \leq \mu^0$, and $|d^{k_j}| \to 0$, we conclude that $\Delta^{k_j} \downarrow 0$.

Finally, from Lemma 2.1(1), as $-d^{k_j} \in G(x^{k_j})$, there exists a $v^{k_j} \in \partial f(x^{k_j})$ such that

$$\begin{aligned} &\left|-v^{k_j} - d^{k_j}\right| \leq \bar{K}\Delta^{k_j} \\ &\quad\Rightarrow \quad \left|-v^{k_j}\right| - \left|d^{k_j}\right| \leq \bar{K}\Delta^{k_j} \\ &\quad\Rightarrow \quad \left|v^{k_j}\right| \leq \bar{K}\Delta^{k_j} + \left|d^{k_j}\right|, \end{aligned}$$

which implies that

$$0 \leq \left|v^{k_j}\right| \leq \bar{K}\Delta^{k_j} + \left|d^{k_j}\right| \to 0.$$

So,

$$\lim_{j\to\infty} \left|v^{k_j}\right| = 0,$$

where $|v^{k_j}| \geq \operatorname{dist}(0|\partial f(x^{k_j})) \geq 0$, which implies $\operatorname{dist}(0|\partial f(x^{k_j})) \to 0$. We have $x^{k_j} \to \bar{x}$. As f is a finite max function, ∂f is outer semicontinuous (see [35, Definition 5.4 & Proposition 8.7]). Hence, every cluster point $\bar{x}$ of a convergent subsequence of $\{x^k\}_{k=0}^{\infty}$ satisfies $0 \in \partial f(\bar{x})$.

Case 2: A finite number of line search successes occur.

This means there exists a $\bar{k}$ such that $x^k = x^{\bar{k}} = \bar{x}$ for all $k \geq \bar{k}$. However, by Corollary 2.7, if $0 \notin \partial f(\bar{x})$, then after a finite number of iterations, the algorithm will find function value decrease (line search success). Hence, we have $0 \in \partial f(\bar{x})$.

To see $\Delta^k \downarrow 0$ and $|d^k| \to 0$, note that by Lemma 2.1(1) and $0 \in \partial f(\bar{x})$, we have that for all $k > \bar{k}$ there exists $d \in G(x^k)$ such that $|d - 0| \leq \bar{K}\Delta^k$. In particular, $|d^k| = |\operatorname{Proj}(0|G(x^k))| \leq \bar{K}\Delta^k \leq \bar{K}\Delta^0$ for all $k > \bar{k}$. Now note that one of two situations must occur: either (3) is unsatisfied an infinite number of times or after a finite number of steps (3) is always satisfied and a line search failure occurs in Step 4. In the first case, we directly have $\Delta^k \downarrow 0$ (by Step 3). In the second case, we have $\mu^k \downarrow 0$ (by Step 4), so $\Delta^k \leq \mu^k |d^k| \leq \mu^k \bar{K}\Delta^0$ (by (3)) implies $\Delta^k \downarrow 0$. Finally, $|d^k| \leq \bar{K}\Delta^k$ completes the proof. □

Our last result shows that if the algorithm terminates in Step 2, then the distance from 0 to the exact subdifferential is controlled by ε_{tol}.

Theorem 2.9 *Let $f = \max\{f_i : i = 1, \ldots, N\}$ where each $f_i \in \mathcal{C}^1$. Suppose there exists a $\bar{K} > 0$ such that for each iteration k, the approximate gradient satisfies*

$|\nabla_A f_i(x^k) - \nabla f_i(x^k)| \le \bar{K}\Delta^k$ for all $i = 1, \dots, N$. Suppose the AGS algorithm terminates at some iteration $\bar{k}$ in Step 2 for $\varepsilon_{tol} > 0$. Then

$$\operatorname{dist}\big(0|\partial f\big(x^{\bar{k}}\big)\big) < \big(1 + \bar{K}\mu^0\big)\varepsilon_{tol}.$$

Proof Let $\bar{w} = \operatorname{Proj}(0|G(x^k))$. We use $\bar{v} \in \partial f(x^k)$ as constructed in Lemma 2.1(1) to see that

$$\begin{aligned}\operatorname{dist}\big(0|\partial f\big(x^k\big)\big) &\le \operatorname{dist}(0|\bar{w}) + \operatorname{dist}(\bar{w}|\bar{v})\\ &= \big|d^k\big| + |\bar{w} - \bar{v}|\\ &\le \big|d^k\big| + \bar{K}\Delta^k\\ &< \varepsilon_{tol} + \bar{K}\Delta^k\end{aligned}$$

The final statement now follows by the test $\Delta^k \le \mu^k|d^k| \le \mu^0\varepsilon_{tol}$ in Step 2. $\square$

3 Robust approximate gradient sampling algorithm

The AGS algorithm depends on the active set of functions at each iterate, $A(x^k)$. Of course, it is possible at various times in the algorithm for there to be functions that are inactive at the current iterate, but active within a small radius of the current iterate. Typically such behaviour means that the current iterate is close to a 'nondifferentiable ridge' formed by the function. In [6] and [7], it is suggested that allowing an algorithm to take into account these 'almost active' functions will provide a better idea of what is happening at and around the current iterate, thus, making the algorithm more robust.

In this section we present the robust gradient sampling algorithm (RAGS algorithm). Specifically, we adapt the AGS algorithm by expanding our active set to include all functions that are active at any of the points in the set Y. Recall from the AGS algorithm that the set Y is sampled from within a ball of radius Δ^k. Thus, the points in Y are not far from the current iterate. We define the robust active set next.

Definition 3.1 Let $f = \max\{f_i : i = 1, \dots, N\}$ where $f_i \in \mathcal{C}^1$. Let $y^0 = x^k$ be the current iterate and $Y = [y^0, y^1, y^2, \dots, y^m]$ be a set of randomly sampled points from a ball centered at y^0 with radius Δ^k. The **robust active set** of f on Y is

$$A(Y) = \bigcup_{y^j \in Y} A\big(y^j\big) \quad .$$

3.1 Algorithm—RAGS

Using the idea of the robust active set, we alter the AGS algorithm to accommodate the robust active set by replacing Steps 1 and 2 with the following.

1. Generate Approximate Subdifferential G_Y^k (Robust):
   Generate a set $Y = [x^k, y^1, \ldots, y^m]$ around the current iterate x^k such that

   $$\max_{j=1,\ldots,m} \left|y^j - x^k\right| \leq \Delta^k.$$

   Use Y to calculate the approximate gradient of f_i, denoted $\nabla_A f_i$, at x^k for each $i \in A(Y)$. Then set $G^k = \text{conv}\{\nabla_A f_i(x^k)\}_{i\in A(x^k)}$ and $G_Y^k = \text{conv}\{\nabla_A f_i(x^k)\}_{i\in A(Y)}$.
2. Generate Search Direction:
   Let

   $$d^k = -\operatorname{Proj}\bigl(0|G^k\bigr).$$

   Let

   $$d_Y^k = -\operatorname{Proj}\bigl(0|G_Y^k\bigr).$$

   Check if

   $$\Delta^k \leq \mu^k \left|d^k\right|. \tag{14}$$

   If (14) does not hold, then set $x^{k+1} = x^k$,

   $$\Delta^{k+1} = \begin{cases} \theta\mu^k|d^k| & \text{if } |d^k| \neq 0, \\ \theta\Delta^k & \text{if } |d^k| = 0, \end{cases}$$

   $k = k+1$ and return to Step 1. If (14) holds and $|d^k| < \varepsilon_{tol}$, then STOP. Else, continue to the line search, using d_Y^k as a search direction.

Notice that in Step 2 we still use the stopping conditions from Sect. 2. Although this modification requires the calculation of two projections, it should be noted that neither of these projections are particularly difficult to calculate. In Sect. 3.3, we use the Goldstein approximate subdifferential to adapt Theorem 2.9 to work for stopping conditions based on d_Y^k, but we still do not have theoretical results for the exact subdifferential. It is important to note that no additional function evaluations are required for this modification.

In the numerics section, we test each version of our algorithm using the robust descent direction to check the stopping conditions. This alteration shows convincing results that the robust stopping conditions not only guarantee convergence, but significantly decrease the number of function evaluations required for the algorithm to converge.

3.2 Convergence

To show that the RAGS algorithm is well-defined we will require the fact that when Δ^k is small enough, the robust active set is in fact equal to the original active set.

Lemma 3.2 *Let $f = \max\{f_i : i = 1, \ldots, N\}$ where each $f_i \in \mathcal{C}^1$. Let $Y = [\bar{x}, y^1, \ldots, y^m]$ be a randomly sampled set from a ball centered at $\bar{x}$ with radius Δ. Then there exists an $\tilde{\varepsilon} > 0$ such that if $Y \subseteq B_{\tilde{\varepsilon}}(\bar{x})$, then $A(\bar{x}) = A(Y)$.*

Proof Clearly, if $i \in A(\bar{x})$, then $i \in A(Y)$ as $\bar{x} \in Y$.

Consider $i \notin A(\bar{x})$. Then by the definition of f, we have that

$$f_i(\bar{x}) < f(\bar{x}).$$

By definition, f is continuous, thus, there exists an $\tilde{\varepsilon}_i > 0$ such that for all $z \in B_{\tilde{\varepsilon}_i}(\bar{x})$,

$$f_i(z) < f(z).$$

If $\Delta < \tilde{\varepsilon}_i$, then we have $|y^j - \bar{x}| < \tilde{\varepsilon}_i$ for all $j = 1, \dots, m$. Therefore,

$$f_i\big(y^j\big) < f\big(y^j\big) \quad \text{for all } j = 1, \dots, m, \tag{15}$$

so $i \notin A(Y)$. Setting $\tilde{\varepsilon} = \min_i \tilde{\varepsilon}_i$ completes the proof. □

Using Lemma 3.2, we can easily conclude that the AGS algorithm is still well-defined when using the robust active set.

Corollary 3.3 *Let $f = \max\{f_i : i = 1, \dots, N\}$ where each $f_i \in \mathcal{C}^1$. Suppose $0 \notin \partial f(x^k)$ for each iteration k. Suppose there exists a $\bar{K} > 0$ such that given any set of points generated in Step 1 of the RAGS algorithm, the approximate gradient satisfies $|\nabla_A f_i(x^k) - \nabla f_i(x^k)| \le \bar{K}\Delta^k$ for all $i = 1, \dots, N$. Then after a finite number of iterations, the RAGS algorithm will find function value decrease.*

Proof Consider x^k, where $0 \notin \partial f(x^k)$.

For eventual contradiction, suppose we do not find function value decrease. In the RAGS algorithm, this corresponds to an infinite number of line search failures. If we have an infinite number of line search failures, then $\Delta^k \to 0$, as $|d^k|$ is bounded, and $x^{\bar{k}} = x^k$ for all $\bar{k} \ge k$. In Lemma 3.2, $\tilde{\varepsilon}$ depends only on x^k. Hence, we can conclude that eventually $\Delta^k \le \tilde{\varepsilon}$ and therefore $Y \subseteq B_{\tilde{\varepsilon}}(x^k)$. Thus, eventually $A(x^k) = A(Y^k)$. Once the two active sets are equal, the results of Sect. 2 will hold. □

To examine convergence of the RAGS algorithm we use the result that eventually the robust active set at the current iterate will be a subset of the regular active set at any cluster point of the algorithm.

Lemma 3.4 *Let $f = \max\{f_i : i = 1, \dots, N\}$ where each $f_i \in \mathcal{C}^1$. Let $Y^k = [x^k, y^1, \dots, y^m]$ be a randomly sampled set from a ball centered at x^k with radius Δ^k. Let $x^k \to \bar{x}$. Then there exists an $\tilde{\varepsilon} > 0$ such that if $Y^k \subseteq B_{\tilde{\varepsilon}}(\bar{x})$, then $A(Y^k) \subseteq A(\bar{x})$.*

Proof Let $i \notin A(\bar{x})$. We must show that for k sufficiently large $i \notin A(Y^k)$.

By definition of f, we have that

$$f_i(\bar{x}) < f(\bar{x}).$$

Since f is continuous, there exists an $\tilde{\varepsilon}_i > 0$ such that for all $z \in B_{\tilde{\varepsilon}_i}(\bar{x})$

$$f_i(z) < f(z).$$

If $Y^k \subseteq B_{\tilde{\varepsilon}_i}(\bar{x})$, then we have $|x^k - \bar{x}| < \tilde{\varepsilon}_i$ and $|y^j - \bar{x}| < \tilde{\varepsilon}_i$ for all $j = 1, \ldots, m$. Therefore

$$f_i\left(x^k\right) < f\left(x^k\right)$$

and

$$f_i\left(y^j\right) < f\left(y^j\right) \quad \text{for all } j = 1, \ldots, m.$$

Thus, if $Y^k \subseteq B_{\tilde{\varepsilon}_i}(\bar{x})$, then $i \notin A(Y^k)$. Letting $\tilde{\varepsilon} = \min_i \tilde{\varepsilon}_i$ completes the proof. □

Now we examine the convergence of the RAGS algorithm.

Theorem 3.5 *Let $f = \max\{f_i : i = 1, \ldots, N\}$ where each $f_i \in \mathcal{C}^1$. Set $\varepsilon_{tol} = 0$ and suppose that $\{x^k\}_{k=0}^{\infty}$ is an infinite sequence generated by the RAGS algorithm. Suppose there exists a $\bar{K} > 0$ such that given any set of points generated in Step 1 of the RAGS algorithm, the approximate gradient satisfies the error bound $|\nabla_A f_i(x^k) - \nabla f_i(x^k)| \leq \bar{K}\Delta^k$ for all $i = 1, \ldots, N$. Suppose t^k is bounded away from 0. Then either*

1. $f(x^k) \downarrow -\infty$, *or*
2. $|d^k| \to 0$, $\Delta^k \downarrow 0$ *and every cluster point $\bar{x}$ of the sequence $\{x^k\}_{k=0}^{\infty}$ satisfies* $0 \in \partial f(\bar{x})$.

Proof If $f(x^k) \downarrow -\infty$, then we are done.

Conversely, if $f(x^k)$ is bounded below, then $f(x^k)$ is non-increasing and bounded below, therefore $f(x^k)$ converges. We consider two cases.

Case 1: An infinite number of line search successes occur.

Let $\bar{x}$ be a cluster point of $\{x^k\}_{k=0}^{\infty}$. Notice that x^k only changes for line search successes, so there exists a subsequence $\{x^{k_j}\}_{k=0}^{\infty}$ of line search successes such that $x^{k_j} \to \bar{x}$. Following the arguments of Theorem 2.8, we have $|d^{k_j}| \to 0$ and $\Delta^{k_j} \downarrow 0$. Notice that if $\Delta^{k_j} \downarrow 0$, then eventually $Y^{k_j} \subseteq B_{\tilde{\varepsilon}}(\bar{x})$, where $x^{k_j} \to \bar{x}$ and $\tilde{\varepsilon}$ is defined as in Lemma 3.4. Thus, by Lemma 3.4, we have that $A(Y^{k_j}) \subseteq A(\bar{x})$. This means that $G_{Y^{k_j}}(x^{k_j})$ is formed from a subset of the approximated active gradients, related to $\partial f(\bar{x})$. Thus, by Lemma 2.1(1), as $-d^{k_j} \in G_{Y^{k_j}}(x^{k_j})$, we can construct a $v^{k_j} \in \partial f(\bar{x})$ from the same set of approximated active gradients, related to $G_{Y^{k_j}}(x^{k_j})$, such that

$$\begin{aligned}
\left|-d^{k_j} - v^{k_j}\right| &= \left|\sum_{i \in A(Y^{k_j})} \alpha_i \nabla_A f_i\left(x^{k_j}\right) - \sum_{i \in A(Y^{k_j})} \alpha_i \nabla f_i(\bar{x})\right| \\
&\leq \sum_{i \in A(Y^{k_j})} \alpha_i \left|\nabla_A f_i\left(x^{k_j}\right) - \nabla f_i(\bar{x})\right| \\
&\leq \sum_{i \in A(Y^{k_j})} \alpha_i \left|\nabla_A f_i\left(x^{k_j}\right) - \nabla f_i\left(x^{k_j}\right)\right| + \alpha_i \left|\nabla f_i\left(x^{k_j}\right) - \nabla f_i(\bar{x})\right|
\end{aligned}$$

$$\leq \sum_{i\in A(Y^{k_j})} \alpha_i \bar{K}\Delta^{k_j} + \alpha_i\left|\nabla f_i\left(x^{k_j}\right) - \nabla f_i(\bar{x})\right|$$

$$\leq \bar{K}\Delta^{k_j} + \sum_{i\in A(Y^{k_j})} \alpha_i\left|\nabla f_i\left(x^{k_j}\right) - \nabla f_i(\bar{x})\right|.$$

Using the Triangle Inequality, we have that

$$\left|v^{k_j}\right| - \left|d^{k_j}\right| \leq \bar{K}\Delta^{k_j} + \sum_{i\in A(Y^{k_j})} \alpha_i\left|\nabla f_i\left(x^{k_j}\right) - \nabla f_i(\bar{x})\right|.$$

We already showed that $|d^{k_j}| \to 0$ and $\Delta^{k_j} \downarrow 0$. Furthermore, since ∇f_i is continuous and $x^{k_j} \to \bar{x}$, we have $|\nabla f_i(x^{k_j}) - \nabla f_i(\bar{x})| \to 0$. So,

$$\lim_{j\to\infty}\left|v^{k_j}\right| = 0.$$

Using the same arguments as in Theorem 2.8, the result follows.
Case 2: A finite number of line search successes occur.

This means there exists a $\bar{k}$ such that $x^k = x^{\bar{k}} = \bar{x}$ for all $k \geq \bar{k}$. However, by Corollary 3.3, if $0 \notin \partial f(\bar{x})$, then after a finite number of iterations, the algorithm will find function value decrease (line search success). Hence, we have $0 \in \partial f(\bar{x})$. This implies $\Delta^k \downarrow 0$ and $|d^k| \to 0$, as in the proof of Theorem 2.8. □

Remark 3.6 Using d^k to check our stopping conditions allows the result of Theorem 2.9 to still hold.

3.3 Robust stopping with Goldstein approximate subdifferential

We want to provide some insight as to how Theorem 2.9 can work for stopping conditions based on d^k_Y, that is, replacing the stopping conditions $\Delta^k \leq \mu^k|d^k|$ and $|d^k| < \varepsilon_{tol}$ in Step 2 with the robust stopping conditions

$$\Delta^k \leq \mu^k\left|d^k_Y\right| \quad \text{and} \quad \left|d^k_Y\right| < \varepsilon_{tol}. \tag{16}$$

In the situation when the algorithm terminates, the following proposition does not theoretically justify why the stopping conditions are sufficient, but does help explain their logic. Theoretically, since we do not know what $\bar{x}$ is, we cannot tell when $A(Y) \subseteq A(\bar{x})$. However, as seen above, we do know that if $x^k \to \bar{x}$, then eventually $A(Y^k) \subseteq A(\bar{x})$.

Proposition 3.7 *Let $f = \max\{f_i : i = 1, \dots, N\}$ where each $f_i \in \mathcal{C}^1$. Suppose there exists a $\bar{K} > 0$ such that $|\nabla_A f_i(x) - \nabla f_i(x)| \leq \bar{K}\Delta^k$ for all $i = 1, \dots, N$ and for all $x \in B_{\Delta^k}(x^k)$. Suppose the RAGS algorithm terminates at some iteration $\bar{k}$ in Step 2 using the robust stopping conditions given in (16). Furthermore, suppose there exists $\bar{x} \in B_{\Delta^{\bar{k}}}(x^{\bar{k}})$ such that $A(Y^{\bar{k}}) \subseteq A(\bar{x})$. Then*

$$\operatorname{dist}\big(0|\partial f(\bar{x})\big) < \big(1 + \bar{K}\mu^0\big)\varepsilon_{tol}.$$

Proof If $A(Y^{\bar{k}}) \subseteq A(\bar{x})$, then the proofs of Lemma 2.1(1) and Theorem 2.9 still hold. □

Additionally, in the following results, we approach the theory for robust stopping conditions using the Goldstein approximate subdifferential. If the RAGS algorithm terminates in Step 2, then it is shown that the distance between 0 and the Goldstein approximate subdifferential is controlled by ε_{tol}. Again, this does not prove the robust stopping conditions are sufficient for the exact subdifferential.

First, the *Goldstein approximate subdifferential*, as defined in [17], is given by the set

$$\partial^G_\Delta f(\bar{x}) = \operatorname{conv}\{\partial f(z) : z \in B_\Delta(\bar{x})\}. \tag{17}$$

We now show that the Goldstein approximate subdifferential contains all of the gradients of the active functions in the robust active set.

Lemma 3.8 *Let $f = \max\{f_i : i = 1, \dots, N\}$. Let $Y = [y^0, y^1, y^2, \dots, y^m]$ be a randomly sampled set from a ball centered at $y^0 = \bar{x}$ with radius Δ. If $f_i \in \mathcal{C}^1$ for each i, then*

$$\partial^G_\Delta f(\bar{x}) \supseteq \operatorname{conv}\{\nabla f_i(y^j) : y^j \in Y, i \in A(y^j)\}.$$

Proof If $f_i \in \mathcal{C}^1$ for each $i \in A(Y)$, then by (2), for each $y^j \in Y$ we have

$$\partial f(y^j) = \operatorname{conv}\{\nabla f_i(y^j)\}_{i \in A(y^j)} = \operatorname{conv}\{\nabla f_i(y^j) : i \in A(y^j)\}.$$

Using this in our definition of the Goldstein approximate subdifferential in (17) and knowing $B_\Delta(\bar{x}) \supseteq Y$, we have

$$\partial^G_\Delta f(\bar{x}) \supseteq \operatorname{conv}\{\operatorname{conv}\{\nabla f_i(y^j) : i \in A(y^j)\} : y^j \in Y\},$$

which simplifies to

$$\partial^G_\Delta f(\bar{x}) \supseteq \operatorname{conv}\{\nabla f_i(y^j) : y^j \in Y, i \in A(y^j)\}. \tag{18}$$

□

Now we have a result similar to Lemma 2.1(1) for d^k_Y with respect to the Goldstein approximate subdifferential.

Remark 3.9 For the following two results, we assume each of the $f_i \in \mathcal{C}^{1+}$ with Lipschitz constant L. Note that this implies the Lipschitz constant L is independent of i. If each $f_i \in \mathcal{C}^{1+}$ with Lipschitz constant L_i, then L is easily obtained by $L = \max\{L_i : i = 1, \dots, N\}$.

Lemma 3.10 *Let $f = \max\{f_i : i = 1, \dots, N\}$ where $f_i \in \mathcal{C}^{1+}$ with Lipschitz constant L. Let $Y = [y^0, y^1, y^2, \dots, y^m]$ be a randomly sampled set from a ball centered at $y^0 = \bar{x}$ with radius Δ. Suppose there exists a $\bar{K} > 0$ such that $|\nabla_A f_i(\bar{x}) - \nabla f_i(\bar{x})| \le \bar{K}\Delta$. Then for all $w \in G_Y(\bar{x})$, there exists a $g \in \partial^G_\Delta f(\bar{x})$ such that*

$$|w - g| \le (\bar{K} + L)\Delta.$$

Proof By definition, for all $w \in G_Y(\bar{x})$ there exists a set of α_i such that

$$w = \sum_{i \in A(Y)} \alpha_i \nabla_A f_i(\bar{x}), \quad \text{where } \alpha_i \geq 0, \sum_{i \in A(Y)} \alpha_i = 1.$$

By our assumption that each $f_i \in \mathcal{C}^{1+}$, Lemma 3.8 holds. It is clear that for each $i \in A(Y)$, $i \in A(y^j)$ for some $y^j \in Y$. Let j_i be the index corresponding to this active index; i.e., $i \in A(y^{j_i})$. Thus, for each $i \in A(Y)$, there is a corresponding active gradient

$$\nabla f_i(y^{j_i}) \in \operatorname{conv}\{\nabla f_i(y^{j_i}) : y^{j_i} \in Y, i \in A(y^{j_i})\} \subseteq \partial_\Delta^G f(\bar{x}).$$

Using the same α_i as above, we can construct

$$g = \sum_{i \in A(Y)} \alpha_i \nabla f_i(y^{j_i}) \in \operatorname{conv}\{\nabla f_i(y^{j_i}) : y^{j_i} \in Y, i \in A(y^{j_i})\} \subseteq \partial_\Delta^G f(\bar{x}).$$

Then

$$\begin{aligned}
|w - g| &= \left| \sum_{i \in A(Y)} \alpha_i \nabla_A f_i(\bar{x}) - \sum_{i \in A(Y)} \alpha_i \nabla f_i(y^{j_i}) \right| \\
&\leq \sum_{i \in A(Y)} \alpha_i \left| \nabla_A f_i(\bar{x}) - \nabla f_i(y^{j_i}) \right| \\
&\leq \sum_{i \in A(Y)} \alpha_i \left(\left| \nabla_A f_i(\bar{x}) - \nabla f_i(\bar{x}) \right| + \left| \nabla f_i(\bar{x}) - \nabla f_i(y^{j_i}) \right| \right) \\
&\leq \sum_{i \in A(Y)} \alpha_i \left(\bar{K}\Delta + L \max_{j_i} |\bar{x} - y^{j_i}| \right) \\
&\leq (\bar{K} + L)\Delta. \qquad \square
\end{aligned}$$

Thus, using Lemma 3.10, we can show that if the algorithm stops due to the robust stopping conditions, then the distance from 0 to the Goldstein approximate subdifferential is controlled by ε_{tol}.

Proposition 3.11 *Let $f = \max\{f_i : i = 1, \ldots, N\}$ where each $f_i \in \mathcal{C}^{1+}$ with Lipschitz constant L. Suppose there exists a $\bar{K} > 0$ such that for each iteration k, the approximate gradient satisfies $|\nabla_A f_i(x^k) - \nabla f_i(x^k)| \leq \bar{K}\Delta^k$ for all $i = 1, \ldots, N$. Suppose the RAGS algorithm terminates at some iteration $\bar{k}$ in Step 2 using the robust stopping conditions given in (16). Then*

$$\operatorname{dist}(0 | \partial_{\Delta^{\bar{k}}}^G f(x^{\bar{k}})) < [1 + \mu^0(\bar{K} + L)]\varepsilon_{tol}.$$

Proof Let $\bar{w} = \operatorname{Proj}(0 | G_{Y^{\bar{k}}}(x^{\bar{k}}))$. We use $\bar{g} \in \partial_{\Delta^{\bar{k}}}^G f(x^{\bar{k}})$ as constructed in Lemma 3.10 to see that

$$\begin{aligned}
\operatorname{dist}\big(0|\partial^{G}_{\Delta^{\bar{k}}} f\big(x^{\bar{k}}\big)\big) &\le \operatorname{dist}(0|\bar{g}) \\
&\le \operatorname{dist}(0|\bar{w}) + \operatorname{dist}(\bar{w}|\bar{g}) \\
&= \big|d_Y^{\bar{k}}\big| + |\bar{w} - \bar{g}| \\
&\le \big|d_Y^{\bar{k}}\big| + (\bar{K} + L)\Delta^{\bar{k}} \\
&< \varepsilon_{tol} + (\bar{K} + L)\Delta^{\bar{k}}
\end{aligned}$$

The statement now follows by the test $\Delta^{\bar{k}} \le \mu^{\bar{k}}|d_Y^{\bar{k}}|$ in Step 2 and the fact that $\mu^{\bar{k}} \le \mu^0$ as $\{\mu^k\}_{k=0}$ is a non-increasing sequence. □

4 Approximate gradients

As seen in the previous two sections, in order for convergence to be guaranteed in the AGS or RAGS algorithm, the approximate gradient used must satisfy an **error bound** for each of the active f_i. Specifically, there must exist a $\bar{K} > 0$ such that

$$\big|\nabla_A f_i\big(x^k\big) - \nabla f_i\big(x^k\big)\big| \le \bar{K}\Delta^k,$$

where $\Delta^k = \max_j |y^j - x^k|$. In this section, we present three specific approximate gradients that satisfy the above requirement: the simplex gradient, the centered simplex gradient and the Gupal estimate of the gradient of the Steklov averaged function.

4.1 Simplex gradient

The simplex gradient is a commonly used approximate gradient. In recent years, several derivative-free algorithms have been proposed that use the simplex gradient ([5, 11, 12, 22] and [19] among others). It is geometrically defined as the gradient of the linear interpolation of f over a set of $n + 1$ points in $\mathbb{R}^n$. Mathematically, we define it as follows.

Let $Y = [y^0, y^1, \dots, y^n]$ be a set of affinely independent points in $\mathbb{R}^n$. We say that Y forms the **simplex** $S = \operatorname{conv}\{Y\}$. Thus, S is a simplex if it can be written as the convex hull of an affinely independent set of $n + 1$ points in $\mathbb{R}^n$.

The **simplex gradient** of a function f over the set Y is given by

$$\nabla_s f(Y) = L^{-1}\delta f(Y),$$

where

$$L = \begin{bmatrix} y^1 - y^0 & \cdots & y^n - y^0 \end{bmatrix}^\top$$

and

$$\delta f(Y) = \begin{bmatrix} f(y^1) - f(y^0) \\ \vdots \\ f(y^n) - f(y^0) \end{bmatrix}.$$

The **condition number** of the simplex formed by Y is given by $\|\hat{L}^{-1}\|$, where

$$\hat{L} = \frac{1}{\Delta}\begin{bmatrix} y^1 - y^0 & y^2 - y^0 & \cdots & y^n - y^0 \end{bmatrix}^\top \quad \text{and where } \Delta = \max_{j=1,\ldots,n} |y^j - y^0|.$$

4.1.1 Convergence

The following result (by Kelley [23]) shows that there exists an appropriate error bound between the simplex gradient and the exact gradient of our objective function. We note that the Lipschitz constant used in the following theorem corresponds to ∇f_i.

Theorem 4.1 *Consider $f_i \in \mathcal{C}^{1+}$ with Lipschitz constant K_i for ∇f_i. Let $Y = [y^0, y^1, \ldots, y^n]$ form a simplex. Let*

$$\hat{L} = \frac{1}{\Delta}\begin{bmatrix} y^1 - y^0 & y^2 - y^0 & \cdots & y^n - y^0 \end{bmatrix}^\top, \quad \textit{where } \Delta = \max_{j=1,\ldots,n} |y^j - y^0|.$$

Then the simplex gradient satisfies the error bound

$$|\nabla_s f_i(Y) - \nabla f_i(y^0)| \leq \bar{K}\Delta,$$

where $\bar{K} = \frac{1}{2}K_i\sqrt{n}\|\hat{L}^{-1}\|$.

Proof See [23, Lemma 6.2.1]. □

With the above error bound result, we conclude that convergence holds when using the simplex gradient as an approximate gradient in both the AGS and RAGS algorithms.

Corollary 4.2 *Let $f = \max\{f_i : i = 1, \ldots, N\}$ where each $f_i \in \mathcal{C}^{1+}$ with Lipschitz constant K_i for ∇f_i. If the approximate gradient used in the AGS or RAGS algorithm is the simplex gradient and $\|\hat{L}^{-1}\|$ is bounded above for each simplex gradient computed, then the results of Theorems 2.4, 2.6, 2.8, 2.9 and 3.5 hold.*

4.1.2 Algorithm—simplex gradient

In order to calculate a simplex gradient in Step 1, we generate a set $Y = [x^k, y^1, \ldots, y^n]$ of points in $\mathbb{R}^n$ and then check to see if Y forms a well-poised simplex by calculating its condition number, $\|\hat{L}^{-1}\|$. A bounded condition number ($\|\hat{L}^{-1}\| < n$) ensures a 'good' error bound between the approximate gradient and the exact gradient.

If the set Y does not form a well-poised simplex ($\|\hat{L}^{-1}\| \geq n$), then we resample. If Y forms a well-poised simplex, then we calculate the simplex gradient of f_i over Y for each $i \in A(x^k)$ and then set the approximate subdifferential equal to the convex hull of the active simplex gradients. We note that the probability of generating a random matrix with a condition number greater than n is asymptotically constant [38]. Thus, randomly generating simplices is a quick and practical option. Furthermore, notice that calculating the condition number does not require function evaluations; thus, resampling does not affect the number of function evaluations required by the algorithm.

4.2 Centered simplex gradient

The centered simplex gradient is the average of two simplex gradients. Although it requires more function evaluations, it contains an advantage that the error bound satisfied by the centered simplex gradient is in terms of Δ^2, rather than Δ.

Let $Y = [y^0, y^1, \dots, y^n]$ form a simplex. We define the sets

$$Y^+ = \left[x, x + \tilde{y}^1, \dots, x + \tilde{y}^n\right]$$

and

$$Y^- = \left[x, x - \tilde{y}^1, \dots, x - \tilde{y}^n\right],$$

where $x = y^0$ and $\tilde{y}^i = y^i - y^0$ for $i = 1, \dots, n$. The **centered simplex gradient** is the average of the two simplex gradients over the sets Y^+ and Y^-, i.e.,

$$\nabla_{CS} f(Y) = \frac{1}{2}\left(\nabla_S f\left(Y^+\right) + \nabla_S f\left(Y^-\right)\right).$$

4.2.1 Convergence

To show that the AGS and RAGS algorithms are both well-defined when using the centered simplex gradient as an approximate gradient, we provide an error bound between the centered simplex gradient and the exact gradient (again by Kelley [23]).

Theorem 4.3 *Consider $f_i \in \mathcal{C}^{2+}$ with Lipschitz constant K_i for $\nabla^2 f_i$. Let $Y = [y^0, y^1, \dots, y^n]$ form a simplex. Let*

$$\hat{L} = \frac{1}{\Delta}\left[y^1 - y^0, \dots, y^n - y^0\right] \quad \text{where } \Delta = \max_{j=1,\dots,n} \left|y^j - y^0\right|.$$

Then the centered simplex gradient satisfies the error bound

$$\left|\nabla_{CS} f_i(Y) - \nabla f_i\left(y^0\right)\right| \le \bar{K}\Delta^2,$$

where $\bar{K} = K_i\sqrt{n}\|\hat{L}^{-1}\|$.

Proof See [23, Lemma 6.2.5]. □

Notice that Theorem 4.3 requires $f_i \in \mathcal{C}^{2+}$. If $f_i \in \mathcal{C}^{1+}$, then the error bound is in terms of Δ, not Δ^2. With the above error bound result, we conclude that convergence holds when using the centered simplex gradient as an approximate gradient in both the AGS and RAGS algorithms.

Corollary 4.4 *Let $f = \max\{f_i : i = 1, \dots, N\}$ where each $f_i \in \mathcal{C}^{2+}$ with Lipschitz constant K_i for $\nabla^2 f_i$. If the approximate gradient used in the AGS or RAGS algorithm is the centered simplex gradient, $\|\hat{L}^{-1}\|$ is bounded above for each centered simplex gradient computed and $\Delta^0 \le 1$, then the results of Theorems 2.4, 2.6, 2.8, 2.9 and 3.5 hold.*

Proof Since $\Delta^0 \leq 1$ and non-increasing, $(\Delta^k)^2 \leq \Delta^k$ and ergo, Theorems 2.4, 2.6, 2.8, 2.9 and 3.5 hold. □

4.2.2 Algorithm

In Step 1, to adapt the AGS or RAGS algorithm to use the centered simplex gradient, we sample our set Y in the same manner as for the simplex gradient (resampling until a well-poised set is achieved). We then form the sets Y^+ and Y^- and proceed as expected.

4.3 Gupal estimate

The nonderivative version of the gradient sampling algorithm presented by Kiwiel in [24] uses the Gupal estimate of the gradient of the Steklov averaged function as an approximate gradient. We see in Theorem 4.8 that an appropriate error bound exists for this approximate gradient. Surprisingly, unlike the error bounds for the simplex and centered simplex gradients, the error bound in Theorem 4.8 does not include a condition number term. Mathematically, we define the Gupal estimate of the gradient of the Steklov averaged function as follows.

For $\alpha > 0$, the **Steklov averaged** function f_α, as defined in [16, Def. 3.1], is given by

$$f_\alpha(x) = \int_{\mathbb{R}^n} f(x-y)\psi_\alpha(y)dy,$$

where $\psi_\alpha : \mathbb{R}^n \to \mathbb{R}_+$ is the **Steklov mollifier** defined by

$$\psi_\alpha(y) = \begin{cases} 1/\alpha^n & \text{if } y \in [-\alpha/2, \alpha/2]^n, \\ 0 & \text{otherwise.} \end{cases}$$

We can equivalently define the Steklov averaged function by

$$f_\alpha(x) = \frac{1}{\alpha^n} \int_{x_1-\alpha/2}^{x_1+\alpha/2} \cdots \int_{x_n-\alpha/2}^{x_n+\alpha/2} f(y)dy_1 \cdots dy_n. \tag{19}$$

The partial derivatives of f_α are given by ([16, Prop. 3.11] and [18])

$$\frac{\partial f_\alpha}{\partial x_i}(x) = \int_{\mathbb{B}_\infty} \gamma_i(x,\alpha,\zeta)d\zeta \tag{20}$$

for $i = 1, \ldots, n$, where $\mathbb{B}_\infty = [-1/2, 1/2]^n$ is the unit cube centred at 0 and

$$\begin{aligned} &\gamma_i(x,\alpha,\zeta) \\ &\quad = \frac{1}{\alpha}\Bigg[f\Bigg(x_1+\alpha\zeta_1,\ldots,x_{i-1}+\alpha\zeta_{i-1},x_i+\frac{1}{2}\alpha,x_{i+1}+\alpha\zeta_{i+1},\ldots,x_n+\alpha\zeta_n\Bigg) \\ &\qquad - f\Bigg(x_1+\alpha\zeta_1,\ldots,x_{i-1}+\alpha\zeta_{i-1},x_i-\frac{1}{2}\alpha,x_{i+1}+\alpha\zeta_{i+1},\ldots,x_n+\alpha\zeta_n\Bigg)\Bigg]. \end{aligned} \tag{21}$$

Given $\alpha > 0$ and $z = (\zeta^1, \ldots, \zeta^n) \in \prod_{i=1}^n \mathbb{B}_\infty$, the **Gupal estimate** of $\nabla f_\alpha(x)$ over z is given by

$$\gamma(x, \alpha, z) = \left(\gamma_1\left(x, \alpha, \zeta^1\right), \ldots, \gamma_n\left(x, \alpha, \zeta^n\right)\right). \tag{22}$$

Remark 4.5 Although we define $\gamma(x, \alpha, z)$ as the Gupal estimate of $\nabla f_\alpha(x)$, in Sect. 4.3.1, we will show that $\gamma(x, \alpha, z)$ provides a good approximation to the exact gradient, $\nabla f(x)$.

Remark 4.6 For the following results, we note that the α used in the above definitions is equivalent to our sampling radius Δ. Thus, we will be replacing α with Δ in the convergence results in Sect. 4.3.1.

4.3.1 Convergence

As before, in order to show that both the AGS and RAGS algorithms are well-defined when using the Gupal estimate as an approximate gradient, we must establish that it provides a good approximate of our exact gradient. To do this, we first need the following classic result from [13].

Lemma 4.7 [13, Lemma 4.1.12] *Let $f \in \mathcal{C}^{1+}$ with Lipschitz constant K for ∇f. Let $y^0 \in \mathbb{R}^n$. Then for any $y \in \mathbb{R}^n$*

$$\left|f(y) - f\left(y^0\right) - \nabla f\left(y^0\right)^\top \left(y - y^0\right)\right| \le \frac{1}{2} K \left|y - y^0\right|^2.$$

Using Lemma 4.7, we establish an error bound between the Gupal estimate and the exact gradient of f.

Theorem 4.8 *Consider $f_i \in \mathcal{C}^{1+}$ with Lipschitz constant K_i for ∇f_i. Let $\varepsilon > 0$. Then for $\Delta > 0$, $z = (\zeta^1, \ldots, \zeta^n) \in Z = \prod_{i=1}^n \mathbb{B}_\infty$ and any point $x \in \mathbb{R}^n$, the Gupal estimate of $\nabla f_{i,\Delta}(x)$ satisfies the error bound*

$$\left|\gamma(x, \Delta, z) - \nabla f_i(x)\right| \le \sqrt{n} \frac{1}{2} K_i \Delta(\sqrt{n} + 3).$$

Proof For $\Delta > 0$, let

$$y^{j-} = \left[x_1 + \Delta\zeta_1, \ldots, x_{j-1} + \Delta\zeta_{j-1}, x_j - \frac{1}{2}\Delta, x_{j+1} + \Delta\zeta_{j+1}, \ldots, x_n + \Delta\zeta_n\right]^\top$$

and

$$y^{j+} = \left[x_1 + \Delta\zeta_1, \ldots, x_{j-1} + \Delta\zeta_{j-1}, x_j + \frac{1}{2}\Delta, x_{j+1} + \Delta\zeta_{j+1}, \ldots, x_n + \Delta\zeta_n\right]^\top.$$

Applying Lemma 4.7, we have that

$$\left|f_i\left(y^{j+}\right) - f_i\left(y^{j-}\right) - \nabla f_i\left(y^{j-}\right)^\top \left(y^{j+} - y^{j-}\right)\right| \le \frac{1}{2} K_i \left|y^{j+} - y^{j-}\right|^2. \tag{23}$$

From (21) (with $\alpha = \Delta$), we can see that

$$f_i\big(y^{j+}\big) - f_i\big(y^{j-}\big) = \Delta\gamma_j\big(x, \Delta, \zeta^j\big).$$

Hence, (23) becomes

$$\big|\Delta\gamma_j\big(x, \Delta, \zeta^j\big) - \nabla f_i\big(y^{j-}\big)^\top\big(y^{j+} - y^{j-}\big)\big| \le \frac{1}{2}K_i\big|y^{j+} - y^{j-}\big|^2. \tag{24}$$

From our definitions of y^{j-} and y^{j+}, we can see that

$$y^{j+} - y^{j-} = [0, \dots, 0, \Delta, 0, \dots, 0]^\top.$$

The inner product in (24) simplifies to

$$\nabla f_i\big(y^{j-}\big)^\top\big(y^{j+} - y^{j-}\big) = \Delta\frac{\partial f_i}{\partial x_j}\big(y^{j-}\big).$$

Thus, we have

$$\left|\Delta\gamma_j\big(x, \Delta, \zeta^j\big) - \Delta\frac{\partial f_i}{\partial x_j}\big(y^{j-}\big)\right| \le \frac{1}{2}K_i\Delta^2,$$

implying

$$\left|\gamma_j\big(x, \Delta, \zeta^j\big) - \frac{\partial f_i}{\partial x_j}\big(y^{j-}\big)\right| \le \frac{1}{2}K_i\Delta. \tag{25}$$

Also notice that

$$\big|y^{j-} - x\big| = \Delta\left|\left(\zeta_1^j, \dots, \zeta_{j-1}^j, -\frac{1}{2}, \zeta_{j+1}^j, \dots, \zeta_n^j\right)\right|.$$

Using the standard basis vector e^j, we have

$$\big|y^{j-} - x\big| = \Delta\big|\zeta^j - \zeta_j e^j - \frac{1}{2}e^j\big| \le \Delta\left(\big|\zeta^j\big| + \big|\zeta_j e^j\big| + \frac{1}{2}\right) \le \frac{1}{2}\Delta(\sqrt{n} + 2).$$

Thus, since $f_i \in \mathcal{C}^{1+}$, we have

$$\big|\nabla f_i\big(y^{j-}\big) - \nabla f_i(x)\big| \le K_i\frac{1}{2}\Delta(\sqrt{n} + 2). \tag{26}$$

Noting that

$$\left|\frac{\partial f_i}{\partial x_j}\big(y^{j-}\big) - \frac{\partial f_i}{\partial x_j}(x)\right| \le \big|\nabla f_i\big(y^{j-}\big) - \nabla f_i(x)\big|,$$

we have

$$\left|\frac{\partial f_i}{\partial x_j}\big(y^{j-}\big) - \frac{\partial f_i}{\partial x_j}(x)\right| \le K_i\frac{1}{2}\Delta(\sqrt{n} + 2). \tag{27}$$

Using (25) and (27), we have

$$\begin{aligned}\left|\gamma_j\big(x,\Delta,\zeta^j\big)-\frac{\partial f_i}{\partial x_j}(x)\right| &\le \left|\gamma_j\big(x,\Delta,\zeta^j\big)-\frac{\partial f_i}{\partial x_j}\big(y^{j-}\big)\right|+\left|\frac{\partial f_i}{\partial x_j}\big(y^{j-}\big)-\frac{\partial f_i}{\partial x_j}(x)\right|\\ &\le \frac{1}{2}K_i\Delta+K_i\frac{1}{2}\Delta(\sqrt{n}+2)\\ &=\frac{1}{2}K_i\Delta(\sqrt{n}+3).\end{aligned}$$

Finally,

$$\begin{aligned}\left|\gamma(x,\Delta,z)-\nabla f_i(x)\right| &= \sqrt{\sum_{j=1}^{n}\left(\gamma_j\big(x,\Delta,\zeta^j\big)-\frac{\partial f_i}{\partial x_j}(x)\right)^2}\\ &\le \sqrt{\sum_{j=1}^{n}\left(\frac{1}{2}K_i\Delta(\sqrt{n}+3)\right)^2}\\ &=\sqrt{n}\frac{1}{2}K_i\Delta(\sqrt{n}+3). \qquad \square\end{aligned}$$

We conclude that convergence holds when using the Gupal estimate of the gradient of the Steklov averaged function of f as an approximate gradient in both the AGS and RAGS algorithms.

Corollary 4.9 *Let $f=\max\{f_i : i=1,\dots,N\}$ where each $f_i\in\mathcal{C}^{1+}$ with Lipschitz constant K_i for ∇f_i. If the approximate gradient used in the AGS or RAGS algorithm is the Gupal estimate of the gradient of the Steklov averaged function, then the results of Theorems 2.4, 2.6, 2.8, 2.9 and 3.5 hold.*

4.3.2 Algorithm

To use the Gupal estimate of the gradient of the Steklov averaged function in both the AGS and RAGS algorithms, in Step 1, we sample independently and uniformly $\{z^{kl}\}_{l=1}^{m}$ from the unit cube in $\mathbb{R}^{n\times n}$, respectively, where m is the number of active functions.

5 Numerical results

5.1 Versions of the AGS and RAGS algorithms

We implemented the AGS and RAGS algorithms using the simplex gradient, the centered simplex gradient and the Gupal estimate of the gradient of the Steklov averaged function as approximate gradients.

Additionally, we used the robust descent direction to create *robust stopping conditions*. That is, the algorithm terminates when

$$\Delta^k \leq \mu^k \left|d_Y^k\right| \quad \text{and} \quad \left|d_Y^k\right| < \varepsilon_{tol}, \tag{28}$$

where d_Y^k is the projection of 0 onto the approximate subdifferential generated using the *robust* active set (see Proposition 3.11 for results linking the robust stopping conditions with the Goldstein approximate subdifferential). The implementation was done in MATLAB (v. 7.11.0.584, R2010b). Software is available by contacting the corresponding author.

Let d^k denote the regular descent direction and let d_Y^k denote the robust descent direction. There are three scenarios that could occur when using the robust stopping conditions:

1. $|d^k| = |d_Y^k|$;
2. $|d^k| \geq |d_Y^k|$, but checking the stopping conditions leads to the same result (line search, radius decrease or termination); or
3. $|d^k| \geq |d_Y^k|$, but checking the stopping conditions leads to a different result.

In Scenarios 1 and 2, the robust stopping conditions have no influence on the algorithm. In Scenario 3, we have two cases:

1. $\Delta^k \leq \mu^k |d_Y^k| \leq \mu^k |d^k|$, but $|d^k| \geq \varepsilon_{tol}$ and $|d_Y^k| < \varepsilon_{tol}$ or
2. $\Delta^k \leq \mu^k |d^k|$ holds, but $\Delta^k > \mu^k |d_Y^k|$.

Thus, we hypothesize that the robust stopping conditions will cause the AGS and RAGS algorithms to do one of two things: to terminate early, providing a solution that has a smaller quality measure, but requires less function evaluations to find, or to reduce its sampling radius instead of carrying out a line search, reducing the number of function evaluations carried out during that iteration and calculating a more accurate approximate subdifferential at the next iteration.

Our goal in this testing is to determine if there are any notable numerical differences in the quality of the three approximate gradients (simplex, centered simplex, and Gupal estimate), the two search directions (robust and non-robust), and the two stopping conditions (robust and non-robust). This leads to the following 12 versions:

AGS Simplex (1. non-robust/2. robust stopping)

RAGS Simplex (3. non-robust/4. robust stopping)

AGS Centered Simplex (5. non-robust/6. robust stopping)

RAGS Centered Simplex (7. non-robust/8. robust stopping)

AGS Gupal (9. non-robust/10. robust stopping)

RAGS Gupal (11. non-robust/12. robust stopping)

5.2 Test sets and software

Testing was performed on a 2.0 GHz Intel Core i7 Macbook Pro. We used the test set from Lukšan-Vlček [27]. The first 25 problems presented are of the desired form

$$\min_x \bigl\{F(x)\bigr\} \quad \text{where } F(x) = \max_{i=1,2,\dots,N} \bigl\{f_i(x)\bigr\}.$$

Of these 25 problems, we omit problem 2.17 because the sub-functions are complex-valued. Thus, our test set presents a total of 24 finite minimax problems with dimensions from 2 to 20. There are several problems with functions f_i that have the form $f_i = |\mathbf{f_i}|$, where $\mathbf{f_i}$ is a smooth function. We rewrote these functions as $f_i = \max\{\mathbf{f_i}, -\mathbf{f_i}\}$. The resulting test problems have from 2 to 130 sub-functions. A summary of the test problems appears in Table 2 in the Appendix.

5.3 Initialization and stopping conditions

We first describe our choices for the initialization parameters used in the AGS and RAGS algorithms.

The initial starting points are given for each problem in [27]. We set the initial accuracy measure to 0.5 with a reduction factor of 0.5. We set the initial sampling radius to 0.1 with a reduction factor of 0.5. The Armijo-like parameter η was chosen to be 0.1 to ensure that a line search success resulted in a significant function value decrease. We set the minimum step length to 10^{-10}.

Next, we discuss the stopping tolerances used to ensure finite termination of the AGS and RAGS algorithms. We encoded four possible reasons for termination in our algorithm. The first is our theoretical stopping condition, while the remaining three are to ensure numerical stability of the algorithm.

1. Stopping conditions met—As stated in the theoretical algorithm, the algorithm terminates for this reason when $\Delta^k \leq \mu^k |d^k|$ and $|d^k| < \varepsilon_{tol}$, where d^k is either the regular or the robust descent direction.
2. Hessian matrix has *NaN*/*Inf* entries—For the solution of the quadratic program in Step 2, we use the *quadprog* command in MATLAB, which has certain numerical limitations. When these limitations result in *NaN* or *Inf* entries in the Hessian, the algorithm terminates.
3. Δ^k, μ^k, and $|d^k|$ are small—This stopping criterion bipasses the test $\Delta^k \leq \mu^k |d^k|$ (in Step 2) and stops if $\Delta^k < \Delta^{tol}$, $|\mu^k| < \mu^{tol}$ and $|d^k| < \varepsilon_{tol}$. Examining Theorem 2.9 along with Theorems 4.1, 4.3 and 4.8, it is clear that this is also a valid stopping criterion. We used a bound of 10^{-6} in our implementation for both Δ^k and μ^k.
4. Max number of function evaluations reached—As a final failsafe, we added an upper bound of 10^6 on the number of function evaluations allowed. (This stopping condition only occurs once in our results.)

5.4 Results

Due to the randomness in the AGS and RAGS algorithms, we carry out 25 trials for each version. For each of the 25 trials, we record the number of function evaluations, the number of iterations, the solution, the quality of the solution and the reason for termination. The quality was measured by the improvement in the number of digits of accuracy, which is calculated using the formula

$$-\log\left(\frac{|F_{\min} - F^*|}{|F_0 - F^*|}\right),$$

where $F_{\min}$ is the function value at the final (best) iterate, F^* is the true minimum value (optimal value) of the problem (as given in [27]) and F_0 is the function value at the initial iterate. Results on function evaluations and solution quality appear in Tables 3, 4 and 5 of the Appendix.

To visually compare algorithmic versions, we use performance profiles. A performance profile is the (cumulative) distribution function for a performance metric [14]. For the AGS and RAGS algorithms, the performance metric is the ratio of the number of function evaluations taken by the current version to successfully solve each test problem versus the least number of function evaluations taken by any of the versions to successfully solve each test problem. Performance profiles eliminate the need to discard failures in numerical results and provide a visual representation of the performance difference between several solvers. For full details on the construction of performance profiles, see [14].

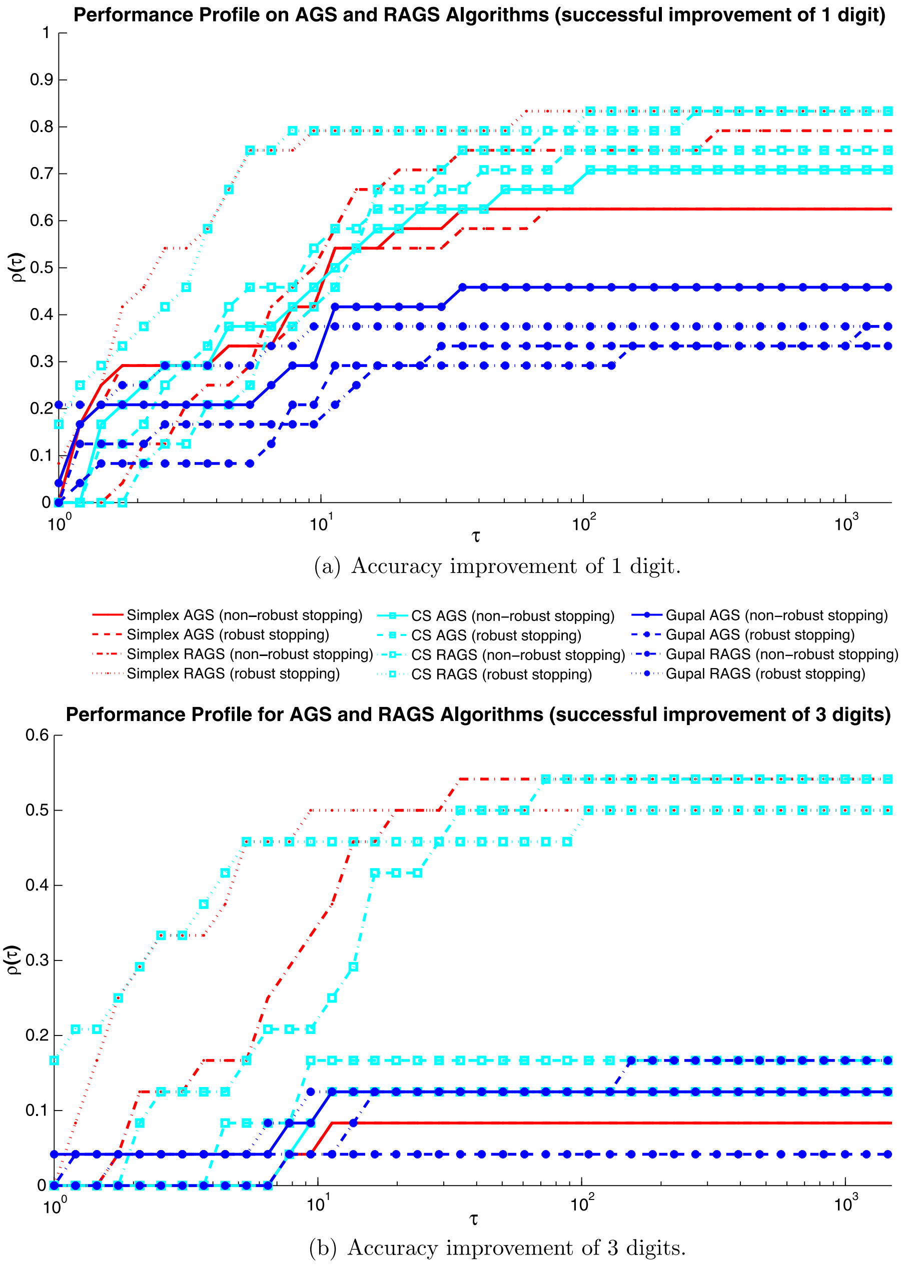
In Figs. 1(a) and 1(b) we include a performance profile showing all 12 versions of the AGS and RAGS algorithms tested, declaring a success for 1 digit and 3 digits of improvement, respectively.

In general, we can see that using the Gupal estimate of the gradient of the Steklov averaged function as an approximate gradient does not produce the best results. It only produces 1 or more digits of accuracy for problems 2.1, 2.2, 2.4, 2.10, 2.18 and 2.23 (robust version). There is no significant difference between the performance of the AGS and RAGS algorithms using the simplex and centered simplex gradients as approximate gradients.

Looking at the results in Tables 3, 4 and 5, and our performance profiles, we can make the following two observations:

(i) the versions of the RAGS algorithm generally outperform (converge faster than) the versions of the AGS algorithm, and
(ii) the RAGS algorithm using the robust stopping conditions terminates faster and with lower (but still significant) accuracy.

Robust active set: From our results, it is clear that expanding the active set to include 'almost active' functions in the RAGS algorithm greatly improves performance for the simplex and centered simplex algorithm. This robust active set brings more local information into the approximate subdifferential and thereby allows for descent directions that are more parallel to any nondifferentiable ridges formed by the function.
Robust stopping conditions: We notice from the performance profiles that in terms of function evaluations, the robust stopping conditions improve the overall performance of the RAGS algorithm, although they decrease the average accuracy on some problems. These results correspond with our previously discussed hypothesis. Furthermore, upon studying the reasons for termination, it appears that the non-robust stopping conditions cause the AGS and RAGS algorithms to terminate mainly due to Δ^k and μ^k becoming too small. For the robust stopping conditions, the RAGS algorithm terminated often because the stopping conditions were satisfied. As our theory in Sect. 3.3 is not complete, we cannot make any theoretical statements about how the robust stopping conditions would perform in general (like those in Theorem 2.9). However, from our results, we conjecture that the alteration is beneficial for decreasing function evaluations.



(a) Accuracy improvement of 1 digit.

(b) Accuracy improvement of 3 digits.

Fig. 1 Performance profiles for 12 versions of AGS/RAGS algorithm

In 23 of the 24 problems tested, for both robust and non-robust stopping conditions, the RAGS algorithm either matches or outperforms the AGS algorithm in average accuracy obtained over 25 trials using the simplex and centered simplex gra-

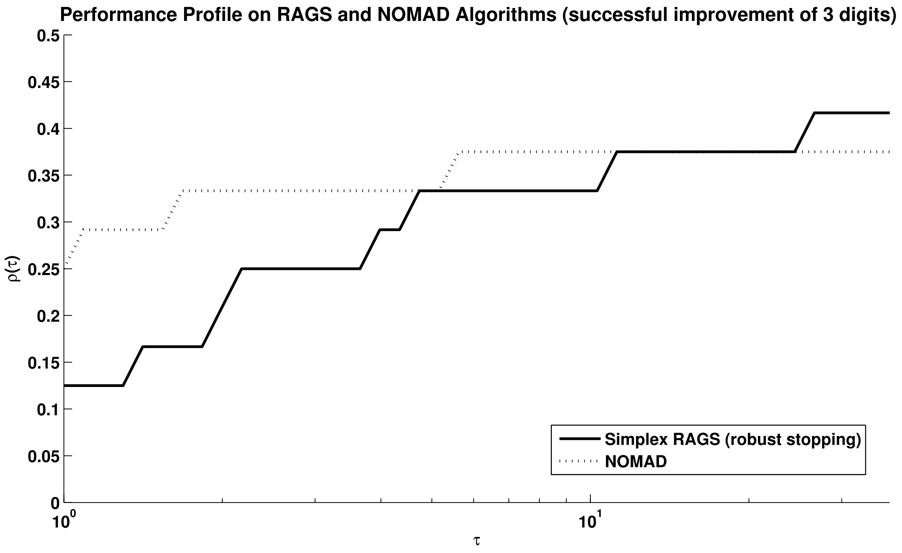


Fig. 2 Performance profile comparing RAGS with NOMAD for an improvement of 3 digits of accuracy

dients. Knowing this, we conclude that the improvement of the accuracy is due to the choice of a robust search direction.

5.5 Comparison to NOMAD

NOMAD is a well established general DFO solver, not specialized for the class of minimax problems [1, 25]. Using the same set of 24 test problems as we did with our previous tests, we compare the RAGS algorithm to the NOMAD algorithm. Based on the previous tests we use the simplex gradient with robust stopping conditions. All parameters for the NOMAD algorithm we left to default settings. The performance profile generated by these two algorithms for the improvement of 3 digits of accuracy appears in Fig. 2.

As seen in Fig. 2, NOMAD appears slightly faster than RAGS, but slightly less robust. Overall, the two algorithms are fairly comparable. Further research into RAGS may lead to algorithmic improvement. For example, a more informed selection of the sample set Y in Step 1, an improved line search, or better selection of default parameters for RAGS, could all lead to improved convergence. We leave such research for future study.

6 Conclusion

We have presented a new derivative-free algorithm for finite minimax problems that exploits the smooth substructure of the problem. Convergence results are given for any arbitrary approximate gradient that satisfies an error bound dependent on the sampling radius. Three examples of such approximate gradients are given. Additionally, a robust version of the algorithm is presented and shown to have the same convergence results as the regular version.

Through numerical testing, we find that the robust version of the algorithm outperforms the regular version with respect to both the number of function evaluations required and the accuracy of the solutions obtained. Additionally, we tested robust stopping conditions and found that they generally required less function evaluations before termination. Overall, the robust stopping conditions paired with the robust version of the algorithm performed best (as seen in the performance profiles).

Considerable future work is available in this research direction. Most obvious is an exploration of the theory behind the performance of the robust stopping conditions. Another direction lies in the theoretical requirement bounding the step length away from 0 (see Theorems 2.8 and 3.5). In gradient based methods, one common way to avoid this requirement is with the use of Wolfe-like conditions. We are unaware of any derivative-free variant on the Wolfe conditions.

Acknowledgements The authors would like to express their gratitude to Dr. C. Sagastizábal for inspirational conversations regarding the Goldstein subdifferential. This research was partially funded by the NSERC DG program and by UBC IRF.

Appendix

Table 2 Test set summary: problem name and number, problem dimension (N), and number of sub-functions (M)

Prob. #	Name	N	M	Prob. #	Name	N	M
2.1	CB2	2	3	2.13	GAMMA	4	122[a]
2.2	WF	2	3	2.14	EXP	5	21
2.3	SPIRAL	2	2	2.15	PBC1	5	60[a]
2.4	EVD52	3	6	2.16	EVD61	6	102[a]
2.5	Rosen-Suzuki	4	4	2.18	Filter	9	82[a]
2.6	Polak 6	4	4	2.19	Wong 1	7	5
2.7	PCB3	3	42[a]	2.20	Wong 2	10	9
2.8	Bard	3	30[a]	2.21	Wong 3	20	18
2.9	Kow.-Osborne	4	22[a]	2.22	Polak 2	10	2
2.10	Davidon 2	4	40[a]	2.23	Polak 3	11	10
2.11	OET 5	4	42[a]	2.24	Watson	20	62[a]
2.12	OET 6	4	42[a]	2.25	Osborne 2	11	130[a]

[a]denotes an absolute value operation (doubled number of sub-functions).

Table 3 Average accuracy for 25 trials obtained by the AGS and RAGS algorithms for the simplex gradient

Prob.	AGS				RAGS			
	Regular stop		Robust stop		Regular stop		Robust stop	
	f-evals	Acc.	f-evals	Acc.	f-evals	Acc.	f-evals	Acc.
2.1	3018	2.082	2855	2.120	2580	9.470	202	6.759
2.2	3136	4.565	3112	4.987	4179	13.211	418	6.343
2.3	3085	0.002	3087	0.002	3090	0.002	3096	0.002
2.4	3254	2.189	3265	2.238	2986	11.559	367	7.570
2.5	3391	1.379	3138	1.351	3576	1.471	539	1.471
2.6	3260	1.236	3341	1.228	4258	1.338	859	1.338
2.7	2949	1.408	2757	1.367	4155	9.939	4190	7.230
2.8	4959	0.879	4492	0.913	3634	9.941	3435	7.655
2.9	2806	0.732	3303	0.581	16000	8.049	13681	3.975
2.10	2978	3.343	2993	3.342	3567	3.459	1924	3.459
2.11	3303	2.554	3453	2.559	35367	6.099	11725	5.063
2.12	2721	1.866	3117	1.871	15052	2.882	8818	2.660
2.13	2580	1.073	2706	0.874	43618	1.952	141	1.679
2.14	3254	1.585	3289	1.086	7713	2.696	4221	1.476
2.15	3917	0.262	5554	0.259	31030	0.286	12796	0.277
2.16	3711	2.182	4500	2.077	20331	3.242	11254	2.178
2.18	10468	0.000	10338	0.000	76355	17.717	30972	17.138
2.19	3397	0.376	3327	0.351	5403	7.105	1767	7.169
2.20	4535	1.624	4271	1.624	8757	8.435	7160	6.073
2.21	8624	2.031	8380	2.157	15225	1.334	11752	1.393
2.22	1563	0.958	1408	1.042	64116	3.049	1256	2.978
2.23	7054	2.557	10392	2.744	6092	6.117	970	6.178
2.24	4570	0.301	7857	0.298	93032	0.447	21204	0.328
2.25	3427	0.339	4197	0.340	98505	0.342	343	0.342

Table 4 Average accuracy for 25 trials obtained by the AGS and RAGS algorithms for the centered simplex gradient

	AGS				RAGS			
	Regular stop		Robust stop		Regular stop		Robust stop	
Prob.	f-evals	Acc.	f-evals	Acc.	f-evals	Acc.	f-evals	Acc.
2.1	3769	2.054	3573	2.051	2351	9.469	221	7.125
2.2	3705	6.888	1284	5.154	4151	9.589	330	5.594
2.3	5410	0.003	5352	0.003	5332	0.003	5353	0.003
2.4	4059	2.520	4154	2.456	4347	11.578	296	6.834
2.5	3949	1.422	3813	1.437	4112	1.471	452	1.471
2.6	3756	1.302	3880	1.309	4815	1.338	879	1.338
2.7	4227	1.435	4187	1.373	5285	9.950	7164	6.372
2.8	6928	0.988	6933	1.003	4116	9.939	3754	7.775
2.9	3301	0.933	3743	0.949	17944	8.072	13014	2.436
2.10	3447	3.343	3424	3.342	4744	3.459	427	3.459
2.11	3593	2.768	4082	2.785	47362	6.344	11886	5.115
2.12	3321	1.892	3406	1.876	15550	2.843	10726	2.651
2.13	3067	1.355	3508	1.216	36969	1.873	519	1.643
2.14	3967	1.771	6110	1.152	9757	2.692	7284	1.510
2.15	4646	0.272	6014	0.273	23947	0.280	15692	0.277
2.16	4518	2.223	6911	2.074	22225	2.628	17001	2.215
2.18	30492	16.931	14671	16.634	125859	17.804	20815	17.293
2.19	4473	0.551	4484	0.591	8561	7.113	1697	5.851
2.20	5462	1.615	5503	1.599	8908	9.011	7846	6.042
2.21	11629	1.887	11724	1.661	18957	1.304	17067	1.339
2.22	1877	1.166	1604	1.160	1453	3.139	2066	3.644
2.23	3807	2.150	7850	3.586	15625	6.117	1020	6.230
2.24	7198	0.302	12745	0.301	115787	0.436	61652	0.329
2.25	4749	0.339	4896	0.341	256508	0.342	568	0.342

Table 5 Average accuracy for 25 trials obtained by the AGS and RAGS algorithm for the Gupal estimate of the gradient of the Steklov averaged function

	AGS				RAGS			
	Regular stop		Robust stop		Regular stop		Robust stop	
Prob.	f-evals	Acc.	f-evals	Acc.	f-evals	Acc.	f-evals	Acc.
2.1	2775	2.448	2542	2.124	13126	3.896	89	2.708
2.2	3729	3.267	2221	2.813	5029	15.904	1776	7.228
2.3	2243	0.000	2262	0.000	2276	0.000	2255	0.000
2.4	2985	2.771	2841	2.892	3475	3.449	2362	3.738
2.5	3493	1.213	3529	1.196	3447	1.211	338	1.200
2.6	3144	0.187	3245	0.188	3018	0.162	3059	0.162
2.7	2631	1.368	3129	1.248	2476	1.048	2208	1.047
2.8	2711	1.125	3898	0.893	2231	0.514	5846	0.515
2.9	3102	0.727	3011	0.600	2955	0.937	3248	0.863
2.10	3075	3.241	2927	3.272	3100	0.000	3050	0.000
2.11	2947	1.527	3307	1.528	3003	1.560	2905	1.560
2.12	3095	1.099	7179	0.000	2670	0.788	7803	0.000
2.13	2755	0.710	1485	0.715	2517	0.231	6871	0.227
2.14	2965	0.574	3070	0.427	2860	0.708	4571	0.668
2.15	2658	0.010	2386	0.017	3355	0.050	3210	0.031
2.16	3431	0.457	3256	0.459	2861	0.199	2620	0.119
2.18	3936	16.345	5814	0.000	3950	16.451	6598	4.542
2.19	3337	0.014	3270	0.011	3488	0.970	3376	0.957
2.20	4604	0.835	4434	0.808	9459	1.360	10560	1.359
2.21	5468	0.000	5418	0.000	6632	0.641	6159	0.635
2.22	21	0.000	21	0.000	21	0.000	21	0.000
2.23	5436	1.814	5176	1.877	1.00E + 06	2.354	954	2.415
2.24	7426	0.280	171	0.017	7927	0.043	6389	0.283
2.25	4519	0.286	4814	0.300	3760	0.017	3209	0.023

References

1. Abramson, A.M., Audet, C., Couture, G., Dennis, J.E. Jr., Le Digabel, S., Tribes, C.: The NOMAD project. Software available at http://www.gerad.ca/nomad
2. Bagirov, A.M., Karasözen, B., Sezer, M.: Discrete gradient method: derivative-free method for nonsmooth optimization. J. Optim. Theory Appl. **137**(2), 317–334 (2008)
3. Bauschke, H.H., Combettes, P.L.: Convex Analysis and Monotone Operator Theory in Hilbert Spaces. CMS Books in Mathematics. Springer, New York (2011)
4. Booker, A.J., Dennis, J.E. Jr., Frank, P.D., Serafini, D.B., Torczon, V.: Optimization using surrogate objectives on a helicopter test example. In: Computational Methods for Optimal Design and Control, Arlington, VA, 1997. Progr. Systems Control Theory, vol. 24, pp. 49–58. Birkhäuser, Boston (1998)

5. Bortz, D.M., Kelley, C.T.: The simplex gradient and noisy optimization problems. In: Computational Methods for Optimal Design and Control. Progr. Systems Control Theory, vol. 24, pp. 77–90. Birkhäuser, Boston (1998)
6. Burke, J.V., Lewis, A.S., Overton, M.L.: Approximating subdifferentials by random sampling of gradients. Math. Oper. Res. **27**(3), 567–584 (2002)
7. Burke, J.V., Lewis, A.S., Overton, M.L.: A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM J. Optim. **15**(3), 751–779 (2005)
8. Cai, X., Teo, K., Yang, X., Zhou, X.: Portfolio optimization under a minimax rule. Manag. Sci. **46**(7), 957–972 (2000)
9. Clarke, F.H.: Optimization and Nonsmooth Analysis, 2nd edn. Classics Appl. Math., vol. 5. SIAM, Philadelphia (1990)
10. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization. MPS/SIAM Series on Optimization, vol. 8. SIAM, Philadelphia (2009)
11. Custódio, A.L., Dennis, J.E. Jr., Vicente, L.N.: Using simplex gradients of nonsmooth functions in direct search methods. IMA J. Numer. Anal. **28**(4), 770–784 (2008)
12. Custódio, A.L., Vicente, L.N.: Using sampling and simplex derivatives in pattern search methods. SIAM J. Optim. **18**(2), 537–555 (2007)
13. Dennis, J.E. Jr., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Classics in Applied Mathematics. SIAM, Philadelphia (1996)
14. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2, Ser. A), 201–213 (2002)
15. Duvigneau, R., Visonneau, M.: Hydrodynamic design using a derivative-free method. Struct. Multidiscip. Optim. **28**, 195–205 (2004)
16. Ermoliev, Y.M., Norkin, V.I., Wets, R.J.-B.: The minimization of semicontinuous functions: mollifier subgradients. SIAM J. Control Optim. **33**, 149–167 (1995)
17. Goldstein, A.A.: Optimization of Lipschitz continuous functions. Math. Program. **13**(1), 14–22 (1977)
18. Gupal, A.M.: A method for the minimization of almost differentiable functions. Kibernetika **1**, 114–116 (1977) (in Russian); English translation in: Cybernetics, **13**(2), 220–222 (1977)
19. Hare, W., Macklem, M.: Derivative-free optimization methods for finite minimax problems. Optim. Methods Softw. **28**(2), 300–312 (2013)
20. Hare, W.L.: Using derivative free optimization for constrained parameter selection in a home and community care forecasting model. In: International Perspectives on Operations Research and Health Care, Proceedings of the 34th Meeting of the EURO Working Group on Operational Research Applied to Health Sciences, pp. 61–73 (2010)
21. Imae, J., Ohtsuki, N., Kikuchi, Y., Kobayashi, T.: A minimax control design for nonlinear systems based on genetic programming: Jung's collective unconscious approach. Int. J. Syst. Sci. **35**, 775–785 (2004)
22. Kelley, C.T.: Detection and remediation of stagnation in the Nelder–Mead algorithm using a sufficient decrease condition. SIAM J. Optim. **10**(1), 43–55 (1999)
23. Kelley, C.T.: Iterative Methods for Optimization. Frontiers in Applied Mathematics, vol. 18. SIAM, Philadelphia (1999)
24. Kiwiel, K.C.: A nonderivative version of the gradient sampling algorithm for nonsmooth nonconvex optimization. SIAM J. Optim. **20**(4), 1983–1994 (2010)
25. Le Digabel, S.: Algorithm 909: NOMAD: nonlinear optimization with the MADS algorithm. ACM Trans. Math. Softw. **37**(4), 1–15 (2011)
26. Liuzzi, G., Lucidi, S., Sciandrone, M.: A derivative-free algorithm for linearly constrained finite minimax problems. SIAM J. Optim. **16**(4), 1054–1075 (2006)
27. Lukšan, L., Vlček, J.: Test Problems for Nonsmooth Unconstrained and Linearly Constrained Optimization. Technical report (February 2000)
28. Madsen, K.: Minimax solution of non-linear equations without calculating derivatives. Math. Program. Stud. **3**, 110–126 (1975)
29. Marsden, A.L., Feinstein, J.A., Taylor, C.A.: A computational framework for derivative-free optimization of cardiovascular geometries. Comput. Methods Appl. Mech. Eng. **197**(21–24), 1890–1905 (2008)
30. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer Series in Operations Research. Springer, New York (1999)
31. Di Pillo, G., Grippo, L., Lucidi, S.: A smooth method for the finite minimax problem. Math. Program., Ser. A **60**(2), 187–214 (1993)

32. Polak, E.: On the mathematical foundations of nondifferentiable optimization in engineering design. SIAM Rev. **29**(1), 21–89 (1987)
33. Polak, E., Royset, J.O., Womersley, R.S.: Algorithms with adaptive smoothing for finite minimax problems. J. Optim. Theory Appl. **119**(3), 459–484 (2003)
34. Polyak, R.A.: Smooth optimization methods for minimax problems. SIAM J. Control Optim. **26**(6), 1274–1286 (1988)
35. Rockafellar, R.T., Wets, R.J.-B.: Variational Analysis. Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 317. Springer, Berlin (1998)
36. Stafford, R.: Random Points in an n-Dimensional Hypersphere. MATLAB File Exchange (2005). http://www.mathworks.com/matlabcentral/fileexchange/9443-random-points-in-an-n-dimensional-hypersphere
37. Wolfe, P.: A method of conjugate subgradients for minimizing nondifferentiable functions. Math. Program. Stud. **3**, 145–173 (1975)
38. Wschebor, M.: Smoothed analysis of $\kappa(A)$. J. Complex. **20**(1), 97–107 (2004)
39. Xu, S.: Smoothing method for minimax problems. Comput. Optim. Appl. **20**(3), 267–279 (2001)