Noname manuscript No.
(will be inserted by the editor)

Fully Adaptive Zeroth-Order Method for Minimizing Functions with Compressible Gradients

Geovani N. Grapiglia · Daniel McKenzie

Received: date / Accepted: date

Abstract We propose an adaptive zeroth-order method for minimizing differentiable functions with L-Lipschitz continuous gradients. The method is designed to take advantage of the eventual compressibility of the gradient of the objective function, but it does not require knowledge of the approximate sparsity level s or the Lipschitz constant L of the gradient. We show that the new method performs no more than $\mathcal{O}\left(n^2\epsilon^{-2}\right)$ function evaluations to find an ϵ-approximate stationary point of an objective function with n variables. Assuming additionally that the gradients of the objective function are compressible, we obtain an improved complexity bound of $\mathcal{O}\left(s\log\left(n\right)\epsilon^{-2}\right)$ function evaluations, which holds with high probability. Preliminary numerical results illustrate the efficiency of the proposed method and demonstrate that it can significantly outperform its non-adaptive counterpart.

Keywords derivative-free optimization · black-box optimization · zeroth-order optimization · worst-case complexity compressible gradients

1 Introduction

In this work we consider the unconstrained minimization of a possibly nonconvex differentiable function $f:\mathbb{R}^n\to\mathbb{R}$. Most well-known iterative algorithms for this type of problem are *gradient-based*, meaning at each iteration they require at least one evaluation of the gradient of $f(\,\cdot\,)$. However, in many problems, it is impossible to compute the gradient of the underlying objective function. Examples include simulation-based optimization [25], reinforcement learning [32,22,13] and hyperparameter tuning [5,35]. These settings create a demand for *zeroth-order optimization algorithms*, which are algorithms that rely only on evaluations of the objective function $f(\,\cdot\,)$. Recently, problems requiring zeroth-order methods have

G. N. Grapiglia was partially supported by FRS-FNRS, Belgium (Grant CDR J.0081.23).

D. McKenzie
Colorado School of Mines, Applied Mathematics and Statistics Department, Colorado, USA
dmckenzie@mines.edu.

emerged in which n, the dimension of the problem, is of order 10^5 or greater. In particular, we highlight many recent works applying zeroth-order methods to fine-tune Large Language Models (LLMs) [21,40,20,12]. For problems of this scale, classical zeroth-order algorithms can struggle to make progress as they generically require $\mathcal{O}(n)$ function evaluations ("queries") per iteration. For problems of moderate to large scale, deterministic zeroth-order algorithms often struggle to make progress, as they typically require $\mathcal{O}(n)$ function evaluations (or "queries") per iteration. To mitigate this issue, many randomized zeroth-order methods have been proposed in recent years, with cheaper per-iteration cost in terms of function evaluations. Some of these methods mimic classical zeroth-order methods, but applied in a low-dimensional subspace [28,4,18,9,31]. Others aim to exploit 'low-dimensional' structure within f [2,10,11,36,38].

Here, we focus on methods that exploit a certain kind of low dimensional structure, namely (approximate) gradient sparsity [37,1,6,7,8,41], particularly the `ZORO`-type algorithms [6,8] which robustly approximate $\nabla f(x)$ using tools from compressed sensing[1]. One adverse consequence of this fusion of optimization and signal processing techniques is that `ZORO`-type algorithms have an unusually large number of hyperparameters. In this paper, we address this issue by proposing a *fully adaptive* version of `ZORO`, which we call `ZORO-FA`. The proposed method is designed to take advantage of the eventual compressibility of the gradient of the objective function, in which case a suitable gradient approximation can be obtained with high probability using $\mathcal{O}\left(s\log(n)\right)$ function evaluations. Specifically, at each iteration, the new method first attempts to use these cheap gradient approximations, and only when they fail a forward-finite difference gradient is computed using $\mathcal{O}\left(n\right)$ function evaluations. Moreover, `ZORO-FA` is endowed with a novel line-search procedure for selecting multiple hyperparameters at once, adapted from that proposed in [16]. We show that the new method takes at most $\mathcal{O}\left(n^2\epsilon^{-2}\right)$ function evaluations to find ϵ-approximate stationary points. Assuming that the gradients of the objective function are compressible with sparsity level s, we obtain an improved complexity bound of $\mathcal{O}\left(s\log\left(n\right)\epsilon^{-2}\right)$ function evaluations, which holds with high probability. Our preliminary numerical results indicate that `ZORO-FA` requires significantly fewer function evaluations than `ZORO` to find points with similar function values, and that the improvement factor increases as the dimensionality of the problems grows.

Contents. The rest of the paper is laid out as follows. In Section 2 we present the key auxiliary results about our gradient approximation and the corresponding variant of the descent lemma. In Section 3, we present the new zeroth-order method and establish a worst-case oracle complexity bound. Finally, in Section 4 we report some numerical results.

2 Auxiliary Results

In what follows we will consider the following assumptions:

A1 $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is L-Lipschitz continuous.

[1] In [2], compressed sensing techniques have been used to exploit (approximate) sparsity of $\nabla^2 f(x)$ in the context of derivative-free trust-region methods.

A2 f is bounded below, that is there exists $f_{\text{low}} \in \mathbb{R}$ such that

$$f(x) \geq f_{\text{low}} \quad \text{for all } x \in \mathbb{R}^n.$$

A3 There exists $p \in (0,1)$ such that, for all $x \in \mathbb{R}^n$, $\nabla f(x)$ is p-compressible with magnitude $\|\nabla f(x)\|$, i.e.,

$$|\nabla f(x)|_{(j)} \leq \frac{\|\nabla f(x)\|}{j^{1/p}}, \quad j = 1, \ldots, n,$$

where $|u|_{(j)}$ denotes the j-th largest-in-magnitude component of $u \in \mathbb{R}^n$.

Given $x \in \mathbb{R}^n$ and $s \in \{1, \ldots, n\}$, let

$$[\nabla f(x)]_{(s)} = \arg\min \{\|v - \nabla f(x)\|_2 \,:\, \|v\|_0 \leq s\},$$

where $\|v\|_0$ denotes the cardinality of the support of v. Note that $[\nabla f(x)]_{(s)}$ is the s-sparse vector formed by the s largest (in modulus) entries of $\nabla f(x)$.

Lemma 1 *Suppose that A3 holds and let $s \in \{1, \ldots, n\}$. Then, for all $x \in \mathbb{R}^n$ we have*

$$\|\nabla f(x) - [\nabla f(x)]_{(s)}\|_1 \leq \frac{\|\nabla f(x)\|_2}{\left(\frac{1}{p} - 1\right)} s^{1-1/p},$$

and

$$\|\nabla f(x) - [\nabla f(x)]_{(s)}\|_2 \leq \frac{\|\nabla f(x)\|_2}{\left(\frac{2}{p} - 1\right)^{\frac{1}{2}}} s^{1/2-1/p}.$$

Proof See Section 2.5 in [26]. □

.

We say that a matrix $Z \in \mathbb{R}^{m \times n}$ has the $4s$-Restricted Isometry Property ($4s$-RIP) when, for every $v \in \mathbb{R}^n$ with $\|v\|_0 \leq 4s$ we have

$$(1 - \delta_{4s}(Z)) \|v\|_2^2 \leq \|Zv\|_2^2 \leq (1 + \delta_{4s}(Z)) \|v\|_2^2$$

for some constant $\delta_{4s}(Z) \in (0,1)$.

Lemma 2 *(Proposition 3.5 in [26]) If $Z \in \mathbb{R}^{m \times n}$ satisfies the $4s$-RIP for some $s \in \{1, \ldots, n\}$, then for any $v \in \mathbb{R}^n$ we have*

$$\|Zv\|_2 \leq \sqrt{1 + \delta_{4s}(Z)} \left(\|v\|_2 + \frac{1}{\sqrt{s}} \|v\|_1\right).$$

Consider the function $c_0 : \mathbb{R} \to \mathbb{R}$ given by

$$c_0(a) = \frac{a^2}{4} - \frac{a^3}{6}. \tag{1}$$

Lemma 3 *(Theorem 5.2 in [3]) Given $s \in \mathbb{N}$ such that $m := \lceil bs\ln(n) \rceil < n$, let $Z \in \mathbb{R}^{m\times n}$ be defined as*

$$Z = \frac{1}{\sqrt{m}} [z_1 \dots z_m]^T ,$$

where $z_i \in \mathbb{R}^n$ are Rademacher random vectors[2]. *Given $\delta \in (0,1)$, if*

$$b > c_1(\delta) \equiv \frac{4\left(1 + \frac{1+\ln(\frac{12}{\delta})}{\ln(\frac{n}{4s})}\right)}{c_0(\frac{\delta}{2})}, \tag{2}$$

for $c_0(\,\cdot\,)$ defined in (1), then Z satisfies the $4s$-RIP for $\delta_{4s}(Z) = \delta$, with probability $\geq 1 - 2e^{-\gamma(\delta)m}$, where

$$\gamma(\delta) \equiv \left[c_0\left(\frac{\delta}{2}\right) - \frac{4}{b}\left(1 + \frac{1+\ln(\frac{12}{\delta})}{\ln(\frac{n}{4s})}\right)\right]. \tag{3}$$

2.1 Gradient Approximation

`ZORO` [8] uses finite differences along random directions to approximate directional derivatives, thought of as linear measurements of $\nabla f(x)$. More precisely, let $\{z_i\}_{i=1}^m \subset \mathbb{R}^n$ be Rademacher random vectors. For fixed $h > 0$ define the vector $y \in \mathbb{R}^m$ with

$$y_i = \frac{f(x + hz_i) - f(x)}{h\sqrt{m}}, \quad j = i, \dots, m. \tag{4}$$

and the matrix

$$Z = \frac{1}{\sqrt{m}} [z_1 \dots z_m]^T \in \mathbb{R}^{m\times n}. \tag{5}$$

Lemma 4 *Suppose that A1 holds and let y and Z be defined by (4) and (5), respectively. Then*

$$y = Z\nabla f(x) + hw, \tag{6}$$

where $w \in \mathbb{R}^m$ with $\|w\| \leq \frac{Ln}{2}$.

Proof From A1,

$$\begin{aligned}\left| f(x + hz_i) - f(x) - \nabla f(x)^T(hz_i)\right| &\leq \frac{h^2}{2} L\|z_i\|_2^2 \\ &= \frac{Lh^2}{2} n \qquad \text{as } [z_i]_j = \pm 1 \text{ for all } j\end{aligned}$$

Then

$$\left| \frac{1}{\sqrt{m}} \frac{f(x + hz_i) - f(x)}{h} - \frac{1}{\sqrt{m}} \nabla f(x)^T z_i \right| \leq \frac{Ln}{2\sqrt{m}} h.$$

[2] That is, $[z_i]_j = \pm 1$ with equal probability.

Now, writing y as in (6), it follows from (5) that

$$h|\,[w]_i\,| = |\,[y]_i - [Z\nabla f(x)]_i\,| \leq \frac{Ln}{2\sqrt{m}}h.$$

Therefore,

$$|\,[w]_i\,| \leq \frac{Ln}{2\sqrt{m}}, \quad i = 1, \ldots, m,$$

which implies that $\|w\|_2 \leq \frac{Ln}{2}$. □

We view (6) as a *linear inverse problem* to be solved for $\nabla f(x)$. As in [8], given $s \in \mathbb{N}\setminus\{0\}$, we will compute an approximation to $\nabla f(x)$ by approximately solving the problem

$$\min_{v\in\mathbb{R}^n} \|Zv - y\| \quad \text{s.t.} \quad \|v\|_0 \leq s, \tag{7}$$

using the algorithm CoSaMP proposed in [26].

Lemma 5 *Suppose that A1 and A3 hold. Given $x \in \mathbb{R}^n$, a set $\{z_i\}_{i=1}^m \subset \mathbb{R}^n$ of Rademacher random vectors, $s \in \{1, \ldots, n\}$ and $h > 0$, let $y \in \mathbb{R}^m$ and $Z \in \mathbb{R}^{m\times n}$ be defined by (4) and (5), respectively. Denote by $\left\{v^\ell\right\}_{\ell\geq 0}$ the sequence generated by applying CoSaMP to the corresponding problem (7), with starting point $v^0 = 0$. If Z satisfies the $4s$-RIP for a constant $\delta_{4s}(Z) < 0.22665$, then for every $\ell \geq 0$,*

$$\|v^\ell - \nabla f(x)\|_2 \leq \left[\left(\frac{13.2}{\left(\frac{2}{p}-1\right)^{\frac{1}{2}}} + \frac{11}{\left(\frac{1}{p}-1\right)}\right)s^{-\frac{2-p}{2p}} + \left(\frac{1}{2}\right)^\ell\right]\|\nabla f(x)\|_2 + \frac{11Ln}{2}h \tag{8}$$

Proof By Lemma 4,

$$y = Z\,[\nabla f(x)]_{(s)} + e,$$

where

$$e = Z\left(\nabla f(x) - [\nabla f(x)]_{(s)}\right) + hw,$$

with $\|w\|_2 \leq Ln/2$. Thus, by Theorem 5 and Remark 3 in [14] we have

$$\begin{aligned}
\|v^\ell - [\nabla f(x)]_{(s)}\,\|_2 &\leq \left(\frac{1}{2}\right)^\ell \|\,[\nabla f(x)]_{(s)}\,\|_2 + 11\|e\|_2 \\
&\leq \left(\frac{1}{2}\right)^\ell \|\nabla f(x)\|_2 + 11\left\|Z\left(\nabla f(x) - [\nabla f(x)]_{(s)}\right)\right\|_2 + 11h\|w\| \\
&\leq \left(\frac{1}{2}\right)^\ell \|\nabla f(x)\|_2 + 11\left\|Z\left(\nabla f(x) - [\nabla f(x)]_{(s)}\right)\right\|_2 + \frac{11Ln}{2}h.
\end{aligned}$$

Then, using Lemma 1, we obtain

$$\begin{aligned}
\|v^\ell - \nabla f(x)\|_2 &\leq \|v^\ell - [\nabla f(x)]_{(s)}\,\|_2 + \|\,[\nabla f(x)]_{(s)} - \nabla f(x)\|_2 \\
&\leq 11\left\|Z\left(\nabla f(x) - [\nabla f(x)]_{(s)}\right)\right\|_2 + \frac{11Ln}{2}h \\
&\quad + \left[\left(\frac{1}{2}\right)^\ell + \frac{s^{\frac{1}{2}-\frac{1}{p}}}{\left(\frac{2}{p}-1\right)^{\frac{1}{2}}}\right]\|\nabla f(x)\|_2
\end{aligned} \tag{9}$$

From Lemmas 1 and 2, and the assumption $\delta_{4s}(Z) < 0.22665$, we also have

$$\left\|Z\left(\nabla f(x) - [\nabla f(x)]_{(s)}\right)\right\|_2 \leq \sqrt{1+\delta_{4s}(Z)}\left(\left\|\nabla f(x) - [\nabla f(x)]_{(s)}\right\| + \frac{1}{\sqrt{s}}\left\|\nabla f(x) - [\nabla f(x)]_{(s)}\right\|_1\right)$$
$$\leq \sqrt{1.22665}\left(\frac{1}{\left(\frac{2}{p}-1\right)^{\frac{1}{2}}} + \frac{1}{\left(\frac{1}{p}-1\right)}\right)s^{\frac{1}{2}-\frac{1}{p}}\|\nabla f(x)\|_2 \quad (10)$$

Finally, combining (9) and (10), we conclude that

$$\|v^\ell - \nabla f(x)\|_2 \leq \left[\left(\frac{11\sqrt{1.22665}+1}{\left(\frac{2}{p}-1\right)^{\frac{1}{2}}} + \frac{11}{\left(\frac{1}{p}-1\right)}\right)s^{-\frac{2-p}{2p}} + \left(\frac{1}{2}\right)^\ell\right]\|\nabla f(x)\|_2 + \frac{11Ln}{2}h,$$

which implies that (8) is true. □

The following corollary may be deduced from Lemma 5 by direct computation.

Corollary 1 *Let $\theta, \epsilon \in (0,1)$. Under the assumptions of Lemma 5, if*

$$\ell \geq \left\lceil \frac{\log(\theta/4)}{\log(0.5)} \right\rceil, \quad (11)$$

$$s \geq s(\theta, p) := \left[\frac{4}{\theta}\left(\frac{13.2}{\left(\frac{2}{p}-1\right)^{\frac{1}{2}}} + \frac{11}{\left(\frac{1}{p}-1\right)}\right)\right]^{\frac{2p}{2-p}}, \quad (12)$$

and

$$0 < h \leq \frac{\theta}{11Ln}\epsilon, \quad (13)$$

then, for $g = v^\ell$ we have

$$\|g - \nabla f(x)\| \leq \frac{\theta}{2}\|\nabla f(x)\| + \frac{\theta}{2}\epsilon. \quad (14)$$

We call the quantity $s(\theta, p)$ defined in (12) the effective sparsity level. We emphasize that $s(\theta, p)$ is independent of n.

Note that, from Lemma 5, the number of iterations of CoSaMP only affects the factor $(1/2)^\ell$ in (8), leaving the other factors unchanged. The per-iteration cost of CoSaMP is $\mathcal{O}(mn+ms^2)$ operations. Thus, Corollary 1 suggests that it is sufficient to perform

$$\ell = \left\lceil \frac{\log(\theta/4)}{\log(0.5)} \right\rceil$$

iterations to guarantee (14), provided that (12) and (13) hold, thereby saving computational effort. For example, if $\theta = 0.25$, it follows that $\ell = 4$ iterations of CoSaMP would be enough.

2.2 Sufficient Decrease Condition

From Corollary 1, we see that if s is sufficiently large and h is sufficiently small, then CoSaMP applied to (7) is able to find a vector g that satisfies

$$\|g - \nabla f(x)\| \leq \frac{\theta}{2}\|\nabla f(x)\| + \frac{\theta}{2}\epsilon \tag{15}$$

whenever the matrix Z satisfies the $4s$-RIP with parameter $\delta_{4s}(Z) < 0.22665$. The next lemma justifies our interest in this type of gradient approximation.

Lemma 6 (Descent Lemma) *Suppose that A1 holds. Given $\epsilon \in (0,1)$ and $\theta \in [0, 1/2)$, let $x \in \mathbb{R}^n$ and $g \in \mathbb{R}^n$ be such that $\|\nabla f(x)\| > \epsilon$ and (15) hold. Moreover, let*

$$x^+ = x - \frac{1}{\sigma}g$$

for some $\sigma > 0$. If

$$\sigma \geq \frac{(\theta+1)^2}{(1-2\theta)}L \tag{16}$$

then

$$f(x) - f(x^+) \geq \frac{1}{2\sigma}\epsilon^2. \tag{17}$$

Proof Since $\|\nabla f(x)\| > \epsilon$, it follows from (15) that

$$\|\nabla f(x) - g\| \leq \theta\|\nabla f(x)\|. \tag{18}$$

Using assumption A1, the Cauchy-Schwarz inequality and (18) we get

$$\begin{aligned}
f(x^+) &\leq f(x) + \nabla f(x)^T\left(-\frac{1}{\sigma}g\right) + \frac{L}{2}\|\frac{1}{\sigma}g\|^2 \\
&= f(x) - \frac{1}{\sigma}\nabla f(x)^T g + \frac{L}{2\sigma^2}\|g\|^2 \\
&= f(x) - \frac{1}{\sigma}\nabla f(x)^T\left[\nabla f(x) - \nabla f(x) + g\right] + \frac{L}{2\sigma^2}\|g\|^2 \\
&\leq f(x) - \frac{1}{\sigma}\|\nabla f(x)\|^2 + \frac{1}{\sigma}\|\nabla f(x)\|\|\nabla f(x) - g\| + \frac{L}{2\sigma^2}\|g\|^2 \\
&\leq f(x) - \frac{1}{\sigma}\|\nabla f(x)\|^2 + \frac{\theta}{\sigma}\|\nabla f(x)\|^2 + \frac{L}{2\sigma^2}\|g\|^2.
\end{aligned} \tag{19}$$

On the other hand, by (15) we also have

$$\|g\| \leq \|g - \nabla f(x)\| + \|\nabla f(x)\| \leq \theta\|\nabla f(x)\| + \|\nabla f(x)\| = (\theta+1)\|\nabla f(x)\|,$$

which gives

$$\|g\|^2 \leq (\theta+1)^2\|\nabla f(x)\|^2. \tag{20}$$

Combining (19) and (20) it follows that

$$\begin{aligned}
f(x^+) &\leq f(x) - \left(\frac{1}{\sigma} - \frac{\theta}{\sigma} - \frac{L}{2\sigma^2}(\theta+1)^2\right)\|\nabla f(x)\|^2 \\
&= f(x) - \left(1 - \theta - \frac{L}{2\sigma}(\theta+1)^2\right)\frac{1}{\sigma}\|\nabla f(x)\|^2,
\end{aligned}$$

and so

$$f(x) - f(x^+) \geq \left(1 - \theta - \frac{L}{2\sigma}(\theta+1)^2\right)\frac{1}{\sigma}\|\nabla f(x)\|^2. \tag{21}$$

By (16) we have

$$1 - \theta - \frac{L}{2\sigma}(\theta+1)^2 \geq \frac{1}{2}. \tag{22}$$

Finally, combining (21), (22) and using the inequality $\|\nabla f(x)\| > \epsilon$, we obtain (17). □

The lemma below suggests a way to construct a gradient approximation g that satisfies (15) with probability one.

Lemma 7 *Suppose that A1 holds. Given $x \in \mathbb{R}^n$, $\epsilon, \theta > 0$, let $g \in \mathbb{R}^n$ be given by*

$$g_i = \frac{f(x+he_i) - f(x)}{h}, \quad i = 1, \dots, n, \tag{23}$$

with

$$0 < h \leq \frac{2\theta}{L\sqrt{n}}\epsilon. \tag{24}$$

If $\|\nabla f(x)\| > \epsilon$, then g satisfies (15).

Proof By (23), A1, (24) and $\epsilon < \|\nabla f(x)\|$, we have

$$\|\nabla f(x) - g\| \leq \frac{L\sqrt{n}}{2}h \leq \theta\epsilon < \frac{\theta}{2}\|\nabla f(x)\| + \frac{\theta}{2}\epsilon.$$

□

3 New Method and its Complexity

The proposed method is presented below as Algorithm 1. We call it Fully Adaptive `ZORO`, or `ZORO-FA`, as the target sparsity (s), sampling radius (h), step-size ($1/\sigma$), and number of queries (m) are all adaptively selected. `ZORO-FA` includes a safeguard mechanism, ensuring that it will make progress even when $\nabla f(x)$ *does not* satisfy the compressibility assumption A3.

Algorithm 1. Fully Adaptive `ZORO`, or `ZORO-FA`.

Step 0. Given $x_0 \in \mathbb{R}^n$, $\epsilon \in (0,1)$, $\theta \in (0, 1/2)$, $b \geq 1$, and $\sigma_0 > 0$. Choose $s_0 \in \mathbb{N}$ such that $\lceil bs_0 \ln(n) \rceil \leq n/4$, sample i.i.d. Rademacher random vectors $\{z_i\}_{i=1}^n$, and set $k := 0$.

Step 1. Set $j := 0$.

Step 2.1 Set $s_j = 2^j s_0$, $\sigma_j = 2^j \sigma_0$, and $m_j = \lceil bs_j \ln(n) \rceil$. If $m_j \geq n$ go to Step 2.4.

Step 2.2 Set

$$h_j = \frac{\theta}{11n\sigma_j}\epsilon \quad \text{and} \quad Z^{(j)} = \frac{1}{\sqrt{m_j}}\left[z_1 \; \dots \; z_{m_j}\right].$$

Compute $y_k^{(j)} \in \mathbb{R}^{m_j}$ by

$$\left[y_k^{(j)}\right]_i = \frac{f(x_k + h_j z_i) - f(x_k)}{\sqrt{m_j} h_j}, \quad i = 1, \ldots, m_j.$$

Step 2.3 Compute

$$g_k^{(j)} \approx \arg \min_{\|g\|_0 \leq s_j} \|Z^{(j)} g - y_k^{(j)}\|_2$$

by applying $\left\lceil \frac{\log(\theta/4)}{\log(0.5)} \right\rceil$ iterations of CoSaMP. Go to Step 2.5.
Step 2.4 For

$$h_j = \frac{2\theta}{\sigma_j \sqrt{n}} \epsilon,$$

compute $g_k^{(j)} \in \mathbb{R}^n$ by

$$[g_k^{(j)}]_\ell = \frac{f(x_k + h_j e_\ell) - f(x_k)}{h_j}, \ \ell = 1, \ldots, n.$$

Step 2.5 Let $x_{k,j}^+ = x_k - \left(\frac{1}{\sigma_j}\right) g_k^{(j)}$. If

$$f(x_k) - f(x_{k,j}^+) \geq \frac{1}{2\sigma_j} \epsilon^2 \tag{25}$$

holds, set $j_k = j$, $g_k = g_k^{(j)}$, $s_k = s_{j_k}$, $\sigma_k = \sigma_{j_k}$ and go to Step 3. Otherwise, set $j := j + 1$ and go to Step 2.1.

Step 3. Define $x_{k+1} = x_{k,j_k}^+$ set $k := k + 1$ and go to Step 1.

Remark 1 Note that vectors $\{z_i\}_{i=1}^n$ are sampled only once, at Step 0. Moreover, in step 2.2 it is important to evaluate the finite difference using the *unscaled* z_i (i.e. compute $f(x_k + hz_i)$), not the i-th column of $Z^{(j)}$.

Remark 2 Note that `ZORO-FA` bears some resemblence to the adaptive `ZORO`, or `adaZORO` algorithm presented in [8]. However, that algorithm only selects the target sparsity (s) adaptively. Moreover, we are able to provide theoretical complexity guarantees for `ZORO-FA`, whereas no such guarantees are presented for `adaZORO`.

3.1 Oracle Complexity Without Compressible Gradients

In this subsection, we analyze Algorithm 1 under Assumptions A1 and A2 only; that is, we do not assume that the gradients of $f(\cdot)$ are compressible. We start by establishing an upper bound on the number of inner loops j_k required to find a point that yields (25).

Lemma 8 *Suppose that A1 holds. If $\|\nabla f(x_k)\| > \epsilon$, then*

$$0 \leq j_k < \max\left\{1, \log_2\left(\frac{2n}{bs_0\ln(n)}\right), \log_2\left(\frac{2(\theta+1)^2}{(1-2\theta)}\frac{L}{\sigma_0}\right)\right\}. \tag{26}$$

Proof From Steps 1 and 2 of Algorithm 1, we see that j_k is the smallest non-negative integer for which (25) holds. Suppose by contradiction that (26) is not true, i.e.,

$$j_k \geq \max\left\{1, \log_2\left(\frac{2n}{bs_0\ln(n)}\right), \log_2\left(\frac{2(\theta+1)^2}{(1-2\theta)}\frac{L}{\sigma_0}\right)\right\}. \tag{27}$$

Then

$$m_{j_k-1} = \lceil bs_{j_k-1}\ln(n)\rceil \geq b\left(2^{j_k-1}s_0\right)\ln(n) \geq b\left(\frac{n}{b\ln(n)}\right)\ln(n) = n.$$

Consequently, by Steps 2.1 and 2.4 of Algorithm 1, it follows that $g_k^{(j_k-1)}$ was defined by

$$\left[g_k^{(j_k-1)}\right]_i = \frac{f(x_k+h_{j_k-1}e_i)-f(x_k)}{h_{j_k-1}}, \quad i=1,\ldots,n, \tag{28}$$

with

$$h_{j_k-1} = \frac{2\theta}{\sigma_{j_k-1}\sqrt{n}}\epsilon. \tag{29}$$

In addition, (27) also implies that

$$\sigma_{j_k-1} = 2^{j_k-1}\sigma_0 \geq \frac{(\theta+1)^2}{(1-2\theta)}L \tag{30}$$

Since $\theta \in (0,1)$, (29) and (30) imply that

$$0 < h_{j_k-1} < \frac{2\theta}{L\sqrt{n}}\epsilon. \tag{31}$$

In view of (28), (31) and assumption $\|\nabla f(x_k)\| > \epsilon$, it follows from Lemma 7 that (15) holds for $x = x_k$ and $g = g_k^{(j_k-1)}$. Thus, by (30) and Lemma 6 we would have

$$f(x_k) - f(x_{k,j_k-1}^{+}) \geq \frac{1}{2\sigma_{j_k-1}}\epsilon^2,$$

contradicting the definition of j_k. □

As a consequence of Lemma 8, we have the following result.

Lemma 9 *Suppose that A1 holds. If $\|\nabla f(x_k)\| > \epsilon$ then*

$$s_k \leq \max\left\{2s_0, \left[\frac{2n}{b\ln(n)}\right], \left[\frac{2(\theta+1)^2L}{(1-2\theta)\sigma_0}\right]s_0\right\}.$$

and

$$\sigma_k \leq \max\left\{2\sigma_0, \left[\frac{2n}{bs_0\ln(n)}\right]\sigma_0, \left[\frac{2(\theta+1)^2}{(1-2\theta)}\right]L\right\}.$$

Denote

$$T(\epsilon) = \inf \{k \in \mathbb{N} \,:\, \|\nabla f(x_k)\| \leq \epsilon\}. \tag{32}$$

Theorem 1 *Suppose that A1-A2 hold and let $\{x_k\}_{k=0}^{T(\epsilon)}$ be generated by Algorithm 1. Then*

$$T(\epsilon) \leq 2 \max \left\{ 2\sigma_0, \left[\frac{2n}{bs_0 \ln(n)}\right] \sigma_0, \left[\frac{2(\theta+1)^2}{(1-2\theta)}\right] L \right\} (f(x_0) - f_{low})\epsilon^{-2}. \tag{33}$$

Proof From Step 2.5 of Algorithm 1, we have

$$f(x_k) - f(x_{k+1}) \geq \frac{1}{2\sigma_k}\epsilon^2, \quad \text{for } k = 0, \ldots, T(\epsilon) - 1.$$

Summing up these inequalities and using A2 and Lemma 9, we obtain

$$\begin{aligned} f(x_0) - f_{low} &\geq \sum_{k=0}^{T(\epsilon)-1} f(x_k) - f(x_{k+1}) \geq \sum_{k=0}^{T(\epsilon)-1} \frac{1}{2\sigma_k}\epsilon^2 \\ &\geq \frac{T(\epsilon)}{2 \max \left\{ 2\sigma_0, \left[\frac{2n}{bs_0 \ln(n)}\right] \sigma_0, \left[\frac{2(\theta+1)^2}{(1-2\theta)}\right] L \right\}}\epsilon^2, \end{aligned}$$

which implies that (33) is true. □

Corollary 2 *Suppose that A1-A2 hold and denote by $FE_{T(\epsilon)}$ the total number of function evaluations performed by Algorithm 1 up to the $(T(\epsilon) - 1)$-st iteration. Then*

$$\begin{aligned} FE_{T(\epsilon)} \leq\ & 2(n+1) \max \left\{ 1, \log_2 \left(\frac{2n}{bs_0 \ln(n)} \right), \log_2 \left(\frac{2(\theta+1)^2}{1-2\theta} \frac{L}{\sigma_0} \right) \right\} \times \\ & \left[2 \max \left\{ 2\sigma_0, \left[\frac{2n}{bs_0 \ln(n)}\right] \sigma_0, \left[\frac{2(\theta+1)^2}{(1-2\theta)}\right] L \right\} (f(x_0) - f_{low})\epsilon^{-2} \right]. \end{aligned} \tag{34}$$

Proof Let us consider the kth iteration of Algorithm 1. For any $j \in \{0, \ldots, j_k\}$, the computation of $g_k^{(j)}$ requires at most $n+1$ function evaluations. Moreover, checking the acceptance condition (25) at Step 2.5 requires one additional function evaluation. Thus, the total number of function evaluations at the kth iteration is bounded from above by $(n+1)(j_k+1)$. Therefore, by Lemma 8,

$$\begin{aligned} FE_{T(\epsilon)} &\leq (n+1) \sum_{k=0}^{T(\epsilon)-1} (j_k + 1) \leq 2(n+1) \sum_{k=0}^{T(\epsilon)-1} j_k \\ &\leq 2(n+1) \max \left\{ 1, \log_2 \left(\frac{2n}{bs_0 \ln(n)} \right), \log_2 \left(\frac{2(\theta+1)^2}{(1-2\theta)} \frac{L}{\sigma_0} \right) \right\} T(\epsilon). \end{aligned} \tag{35}$$

Finally, combining (35) and Theorem 1 we conclude that (34) is true. □

Remark 3 Recall that $m_0 = \lceil bs_0 \log(n) \rceil$. Thus, it follows from Corollary 2 that

$$FE_{T(\epsilon)} \leq \mathcal{O}\left(n \left[\left(\frac{2n}{m_0} \right) \log_2 \left(\frac{2n}{m_0} \right) \right] \epsilon^{-2} \right).$$

Therefore, our complexity bound ranges from $\mathcal{O}(n\epsilon^{-2})$ to $\mathcal{O}(n^2\epsilon^{-2})$, depending on the choice of s_0. When s_0 is chosen such that $m_0 = \lceil bs_0 \ln(n) \rceil = n/4$, Algorithm 1 requires no more than $\mathcal{O}\left(n\epsilon^{-2}\right)$ function evaluations to find an ϵ-approximate stationary point of $f(\,\cdot\,)$. Regarding the dependence on n and ϵ, this bound is consistent, in order, with the complexity bounds established for the method DFQRM in [16]. However, to take advantage of potential compressibility of gradients, one should take s_0 sufficiently small such that $m_0 \ll n$. In this case, the worst-case complexity bound becomes $\mathcal{O}\left(n^2\epsilon^{-2}\right)$.

3.2 Oracle Complexity Assuming Compressible Gradients

In this subsection, we analyze Algorithm 1 under the additional assumption that the gradients of $f(\,\cdot\,)$ are compressible (Assumption A3). Note that

$$J \equiv \left\lceil \log_2\left(\frac{n}{bs_0\ln(n)}\right)\right\rceil$$

is the smallest positive integer such that $m_J = \lceil bs_J \ln(n)\rceil \geq n$. Therefore,

$$Z^{(j)} = \frac{1}{\sqrt{m_j}}\left[z_1 \ldots z_{m_j}\right], \quad j = 0, \ldots, J-1$$

are the only possible sensing matrices used in Algorithm 1. Moreover, since s_0 is chosen such that $\lceil bs_0 \ln(n)\rceil \leq n/4$, we have $J \geq 2$. In this subsection we will consider the following additional assumptions:

A4. Under A1 and A3, the index

$$j^* := \left\lceil \max\left\{1, \log_2\left(\frac{s(\theta,p)}{s_0}\right), \log_2\left(\frac{(\theta+1)^2}{(1-2\theta)}\frac{L}{\sigma_0}\right)\right\}\right\rceil, \tag{36}$$

with $s(\theta,p)$ defined in (12), belongs to the interval $[1, J-1)$.

A5. For j^* defined in (36), the corresponding matrix $Z^{(j^*)}$ satisfies the $4s_{j^*}$-RIP with parameter $\delta_{4s_{j^*}}(Z^{(j^*)}) < 0.22665$.

Remark 4 Assumption 4 implicitly requires $s(\theta,p)$ to be sufficiently small. In view of (12), this means that the gradients should be p-compressible (Assumption A3), with p sufficiently close to zero. Moreover, by Lemma 3, Assumption A5 holds with high probability when b in Step 0 is chosen sufficiently large. This fact is clarified in Corollary 4.

Lemma 10 *Suppose that A1, A4 and A5 hold. If $\|\nabla f(x_k)\| > \epsilon$, then*

$$0 \leq j_k \leq \left\lceil \max\left\{1, \log_2\left(\frac{s(\theta,p)}{s_0}\right), \log_2\left(\frac{(\theta+1)^2}{(1-2\theta)}\frac{L}{\sigma_0}\right)\right\}\right\rceil. \tag{37}$$

Proof Let us show that, for j^* defined in (36), the corresponding point x^+_{k,j^*} satisfies (25). Indeed, by A4 and the definition of J, it follows that $g_k^{(j^*)}$ was obtained at Step 2.3 of Algorithm 1, using the sensing matrix $Z^{(j^*)}$, and the vector $y_k^{(j^*)}$ obtained with

$$h_{j^*} = \frac{\theta}{11n\sigma_{j^*}}. \tag{38}$$

In view of (36) and Assumption A5, the matrix $Z^{(j^*)}$ satisfies the $4s_{j^*}$-RIP with

$$s_{j^*} = 2^{j^*} s_0 \geq s(\theta, p). \tag{39}$$

Moreover, (36) also implies that

$$\sigma_{j^*} = 2^{j^*} \sigma_0 \geq \frac{(\theta+1)^2}{(1-2\theta)} L. \tag{40}$$

Combining (38) and (40), we have that

$$h_{j^*} < \frac{\theta}{11nL}\epsilon. \tag{41}$$

Then, it follows from (39), (41) and Corollary 1 that (15) holds for $x = x_k$ and $g = g_k^{(j^*)}$. Thus, by (40) and Lemma 6 we would have

$$f(x_k) - f(x_{k,j^*}^{+}) \geq \frac{1}{2\sigma_{j^*}}\epsilon^2.$$

Since j_k is the smallest non-negative integer j for which (25) holds, we conclude that $j_k \leq j^*$, and so (37) is true. □

Lemma 10 implies the following upper bounds on s_k and σ_k.

Lemma 11 *Suppose that A1, A4 and A5 hold. If* $\|\nabla f(x_k)\| > \epsilon$ *then*

$$s_k \leq \max\left\{2s_0, 2s(\theta, p), \left[\frac{2(\theta+1)^2 L}{(1-2\theta)\sigma_0}\right] s_0\right\}.$$

and

$$\sigma_k \leq \max\left\{2\sigma_0, \left[\frac{2s(\theta,p)}{s_0}\right]\sigma_0, \left[\frac{2(\theta+1)^2}{(1-2\theta)}\right] L\right\}.$$

Theorem 2 *Suppose that A1-A5 hold and let* $\{x_k\}_{k=0}^{T(\epsilon)}$ *be generated by Algorithm 1, where* $T(\epsilon)$ *is defined by (32). Then*

$$T(\epsilon) \leq 2\max\left\{2\sigma_0, \left[\frac{2s(\theta,p)}{s_0}\right]\sigma_0, \left[\frac{2(\theta+1)^2}{(1-2\theta)}\right] L\right\}(f(x_0) - f_{low})\epsilon^{-2}. \tag{42}$$

Proof As in the proof of Theorem 1, we have

$$f(x_0) - f_{low} \geq \frac{T(\epsilon)}{2\max_{k=0,\ldots,T(\epsilon)-1}\{\sigma_k\}} \tag{43}$$

Thus, combining (43) and the second inequality in Lemma 11 we see that (42) is true. □

Corollary 3 *Suppose that A1-A5 hold and denote by $FE_{T(\epsilon)}$ the total number of function evaluations performed by Algorithm 1 up to the $(T(\epsilon)-1)$-st iteration. Then*

$$\begin{aligned} FE_{T(\epsilon)} \leq 4b &\left[1+\max\left\{1, \log_2\left(\frac{s(\theta,p)}{s_0}\right), \log_2\left(\frac{(\theta+1)^2}{(1-2\theta)}\frac{L}{\sigma_0}\right)\right\}\right] \times \\ &\max\left\{2s_0, 2s(\theta,p), \left\lceil\frac{2(\theta+1)^2 L}{(1-2\theta)\sigma_0}\right\rceil s_0\right\}\ln(n) \times \\ &\max\left\{2\sigma_0, \left\lceil\frac{2s(\theta,p)}{s_0}\right\rceil\sigma_0, \left\lceil\frac{2(\theta+1)^2}{(1-2\theta)}\right\rceil L\right\}(f(x_0)-f_{low})\epsilon^{-2}. \end{aligned} \tag{44}$$

Proof Let us consider the kth iteration of Algorithm 1. By Lemma 10 and Assumption A4, we have $m_j = bs_j\ln(n) < n$ for $j=0,\ldots,j_k$. Then, for $j\in\{0,\ldots,j_k\}$, the computation of $g_k^{(j)}$ requires at most (m_j+1) function evaluations. Moreover, checking the acceptance condition (25) requires one additional function evaluation. Thus, the total number of function evaluations at the kth iteration is bounded from above by

$$\begin{aligned} \sum_{j=0}^{j_k}(m_j+2) \leq 2\sum_{j=0}^{j_k} bs_j\ln(n) &\leq 2b(1+j_k)s_{j_k}\ln(n) \\ &\leq 2b\left[1+\left\lceil\max\left\{1, \log_2\left(\frac{s(\theta,p)}{s_0}\right), \log_2\left(\frac{(\theta+1)^2}{(1-2\theta)}\frac{L}{\sigma_0}\right)\right\}\right\rceil\right] \times \\ &\quad\max\left\{2s_0, 2s(\theta,p), \left\lceil\frac{2(\theta+1)^2 L}{(1-2\theta)\sigma_0}\right\rceil s_0\right\}\ln(n). \end{aligned}$$

Consequently

$$\begin{aligned} FE_{T(\epsilon)} \leq 2b &\left[1+\left\lceil\max\left\{1, \log_2\left(\frac{s(\theta,p)}{s_0}\right), \log_2\left(\frac{(\theta+1)^2}{(1-2\theta)}\frac{L}{\sigma_0}\right)\right\}\right\rceil\right] \times \\ &\max\left\{2s_0, 2s(\theta,p), \left\lceil\frac{2(\theta+1)^2 L}{(1-2\theta)\sigma_0}\right\rceil s_0\right\}\ln(n)T(\epsilon). \end{aligned} \tag{45}$$

Finally, combining (45) and Theorem 2 we conclude that (44) is true. □

When $s_0 \leq s(\theta,p)$, it follows from Corollary 3 that

$$FE_{T(\epsilon)} \leq \mathcal{O}\left(\left\lceil\frac{s(\theta,p)}{s_0}\right\rceil \log_2\left(\frac{s(\theta,p)}{s_0}\right) s(\theta,p)\ln(n)\epsilon^{-2}\right).$$

Assuming further that $s_0 = \mathcal{O}(s(\theta,p))$, i.e., the initial sparsity s_0 is not too far from $s(\theta,p)$, this bound reduces to

$$FE_{T(\epsilon)} \leq \mathcal{O}\left(s(\theta,p)\ln(n)\epsilon^{-2}\right). \tag{46}$$

Corollary 4 *Suppose Assumptions A1–A4 hold, and consider the functions $c_1(\cdot)$ and $\gamma(\cdot)$ defined in (2) and (3), respectively. If $b \geq c_1(0.22664)$, then, with probability at least*

$$1-2e^{-\gamma(0.22664)m_{j^*}}, \tag{47}$$

Algorithm 1 requires no more than $\mathcal{O}\big(s(\theta,p)\ln(n)\epsilon^{-2}\big)$ function evaluations to find an ϵ-approximate stationary point of $f(\cdot)$.

Proof If Assumption A5 also holds, then the stated complexity bound follows from Corollary 3. Since $b \geq c_1(0.22664)$, it follows from Lemma 3 that, with probability at least (47), the sensing matrix $Z^{(j^*)}$ satisfies the $4s_{j^*}$-RIP with $\delta_{4s_{j^*}}(Z^{(j^*)}) < 0.22665$. This implies that Assumption A5 holds with probability at least (47). Therefore, the stated complexity bound holds with the same probability. □

4 Illustrative Numerical Experiments

In the following experiments, we compare `ZORO-FA` to a representative sample of *deterministic* DFO algorithms, namely `Nelder-Mead` [27], `DFQRM` [16], the zeroth-order variant [19] of stochastic subspace descent [18] (henceforth: `SSD`), direct search in reduces spaces [31] (henceforth: `DS-RS`), and `ZORO` and `adaZORO` [8]. We avoid methods which incorporate curvature, or "second-order" information such as `CARS` [17] or the derivative-free variation of `L-BFGS` studied in [34], noting that such functionality could be added to `ZORO-FA` fairly easily. We also do not compare against model-based approaches which use a quadratic model, such as `DFBGN` [9] or `NEWUOA` [30,39], as these offer different trade-offs between run-time, query complexity, and solution quality.

4.1 Sparse Gradient Functions

We test `ZORO-FA` on two challenging test functions known to exhibit gradient sparsity, namely the max-s-squared function (f_{ms}) [7,33] and a variant of Nesterov's 'worst function in the world' [18,29] ($f_{N,\lambda,s}$). These functions are defined as

$$f_{ms}(x) = \sum_{i=1}^{s} x_{\pi(i)}^2 \quad \text{where } |x_{\pi(1)}| \geq |x_{\pi(2)}| \geq \ldots \geq |x_{\pi(n)}|,$$

$$f_{N,\lambda,s}(x) = \frac{\lambda}{8}\left(x_1^2 + \sum_{i=1}^{s}(x_i - x_{i+1})^2 + x_s^2\right) - \frac{\lambda}{4}x_1.$$

The function f_{ms} is challenging as, while ∇f_{ms} is p-compressible (in fact, sparse), the location of the largest-in-magnitude entries ∇f_{ms} changes frequently. Moreover, ∇f_{ms} is not Lipschitz continuous. The function $f_{N,\lambda,s}$ is a variant of a well-known pathological example used to prove tightness of convergence rate bounds in convex optimization. Note that $\nabla f_{N,\lambda,s}$ is Lipschitz continuous with Lipschitz constant $\lambda/2$. For both functions, we take n, the dimension of the function, to be 1000 and $s = 30$. For $f_{N,\lambda,s}$ we take $\lambda = 8$ as in [18].

Parameters. We used the default parameters given in the code for `DFQRM`, as these worked well. For `DS` we used the parameters given in Section 4 of [31] for the Gaussian polling direction, $r = 1$, setting.[3] That is, with notation as in [31], we take $\alpha_0 = 1.0, \alpha_{\max} = 1000, \gamma_{\text{inc}} = 2, \gamma_{\text{dec}} = 0.5$, and we terminate whenever $\alpha_k < 10^{-6}$. For `SSD` there did not appear to be a 'best' choice of subspace dimension

[3] That is, at each iteration, we query f at $f(x_k + z)$ and $f(x_k - z)$ where z is a Gaussian random vector as this setting appears to work best.

Algorithm	b	step size	s_0†	ϵ	θ‡	$\sigma_0\Diamond$	$\sigma^\star_{\min}$	samp. radius
DFQRM	-	-	-	10^{-5}	0	1	0.01	-
ZORO	1	1/8	30	-	-	-	-	10^{-4}
adaZORO	0.5	1/8	30	-	-	-	-	10^{-4}
ZORO-FA	1	-	20	10^{-5}	0.25	2.5	-	-
Nelder-Mead	-	-	-	-	-	-	0.001	-
SSD	-	$\frac{s_0}{8n}$	30	-	-	-	-	10^{-4}
DS-RS	-	-	1	-	-	1	10^{-6}	-

Table 1: Parameters used in sparse benchmarking experiments of Section 4.1. †: Here s_0 refers to the target sparsity in ZORO, the initial target sparsity of adaZORO and ZORO-FA, and the subspace dimension of SSD and DS. ‡: In DFQRM, θ has a slightly different meaning. $\Diamond$: For DS, this is the initial step size (α_0 in [31]). $\star$: In Nelder-Mead this is the minimum simplex size; in DS-RS this is threshold for α_k at which we terminate.

in the results of [18]. So, we take this to be equal to s in our experiments.[4]. Three algorithms require a step-size parameter: ZORO, adaZORO, and SSD, which should theoretically be equal to or proportional to the inverse of the Lipschitz constant of the objective function. In order to guarantee convergence for both f_{ms} and $f_{N,\lambda,s}$, we set the step size to the more pessimistic value[5] of $1/\lambda$. Finally, both ZORO and adaZORO are given the exact sparsity. To make the setting more challenging for ZORO-FA, we deliberately provide it with an underestimate of the true sparsity as s_0, namely 20. See Table 1 for the values of the major parameters used. In all trials x_0 is a random draw from the multivariate normal distribution with mean 0 and covariance $10I$. Each algorithm is given a query budget of $350(n+1)$.

Results. Figure 1 displays typical trajectories for both functions. We present just a single trajectory as we observed very little variation between runs. As is clear, ZORO-FA substantially outperforms all other algorithms. Although ZORO and adaZORO descend quickly at the beginning—likely due to the fact they are given the exact sparsity as input—their progress quickly stalls out as they are unable to decrease their step size or sampling radius. Finally, we note that all algorithms save Nelder-Mead had similar run times—around 10–50 seconds. Nelder-Mead typically took two orders of magnitude more time to run.

4.2 Moré-Garbow-Hillstrom Test Functions

Next, we benchmark ZORO-FA on a subset of the well-known Moré-Garbow-Hillstrom (MGH) suite of test problems [23]. For simplicity, we do not include SSD or DS-RS in this experiment.

Functions. We select the 18 problems with variable dimensions n and considered three values of n: $n = 100$, $n = 500$ and $n = 1000$. We synthetically increase the

[4] We experimented with other values, and did not see significantly different performance.
[5] Although f_{ms} is not everwhere differentiable, it has a Lipshitz constant of 2 wherever it is differentiable.

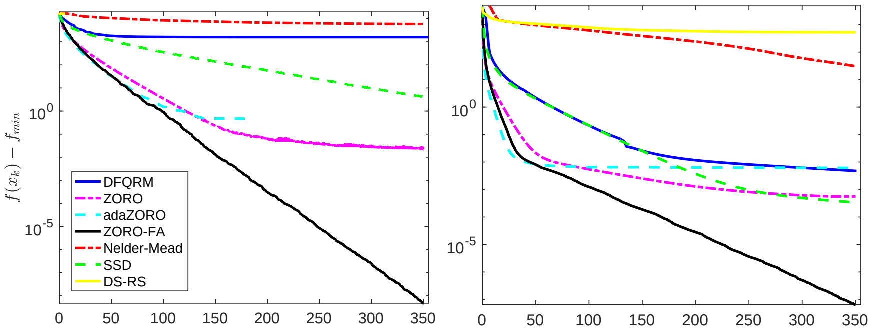

Fig. 1: **Left:** Typical trajectories on f_{ms}. `adaZORO` fails when it is unable to find a gradient approximation satisfying its acceptance criterion. `DS-RS` fails when α_k fall below 10^{-6}. **Right:** Typical trajectories on $f_{N,\lambda,s}$. Note the x-axis displays the cumulative number of queries divided by $n + 1$.

number of functions to 36 by toggling the initial point $x_0 = 10^s \tilde{x}_0$, for $s \in \{0, 1\}$ and $\tilde{x}_0$ the default initial point associated with the MGH function in question.

Parameters. We tuned the parameters of each algorithm to the best of our ability but use the same parameter values for all test functions, for all values of n, see Table 2 for precise values.

Data Profiles. We present the results as data profiles [24] for various tolerances τ. Let $\mathcal{P}$ denote the test problem set. A point (α, y) on a curve in a data profile indicates that a fraction of $y|\mathcal{P}|$ problems were solved, to tolerance τ, given $(n + 1) \times \alpha$ function evaluations. We say an algorithm solves a problem to tolerance τ if it finds an iterate x_k satisfying

$$f(x_k) \leq f_L + \tau \left(f(x_0) - f_L \right), \tag{48}$$

where f_L is the smallest objective function values attained by any algorithm on the problem.

Algorithm	b	step size	s_0†	ϵ	θ	σ_0	$\sigma^\star_{\min}$	samp. radius
`DFQRM`	-	-	-	10^{-5}	0‡	1	0.01	-
`ZORO`	1	$\frac{0.005}{s_0 \log n}$	$0.1n$	-	-	-	-	5×10^{-4}
`adaZORO`	0.5	$\frac{0.005}{s_0 \log n}$	$0.1n$	-	-	-	-	5×10^{-4}
`ZORO-FA`	1	-	$0.1n$	0.01	0.25	$\frac{1}{s_0 \log n}$	-	-
`Nelder-Mead`	-	-	-	-	-	-	0.001	-

Table 2: Parameters used in the Moré-Garbow-Hillstrom benchmarking experiments of Section 4.2. †: Here s_0 refers to the target sparsity in `ZORO`, and the initial target sparsity of `adaZORO` and `ZORO-FA` (which both dynamically adjust this parameter). n refers to the problem size. ‡: In `DFQRM`, θ has a slightly different meaning. $\star$: In `Nelder-Mead` this is the minimum simplex size.

Results. The results are shown in Figure 2. We observe that `ZORO-FA` is quickly able to solve a subset of MGH problems, as shown by the left-hand side of each data profile. We found that `adaZORO` and `ZORO` struggled in all settings. We suspect this is because (i) Only some MGH test functions exhibit gradient sparsity, and (ii) The Lipschitz constants, and consequently the ideal step sizes, vary greatly among MGH functions. Thus, algorithms which are able to adaptively select their step size have an innate advantage.

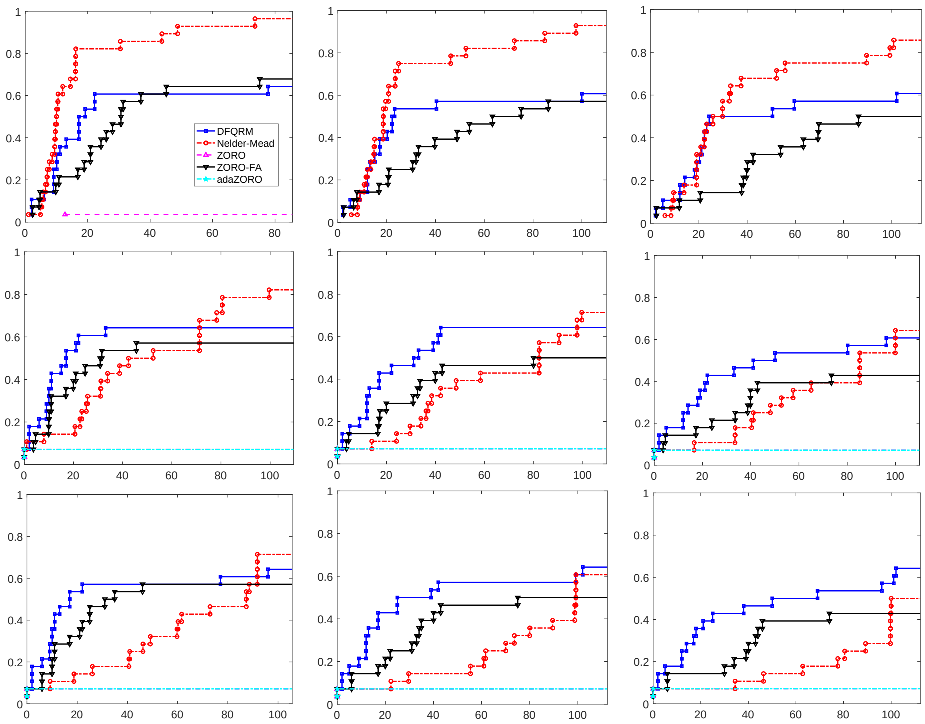

Fig. 2: Data profiles for various tolerances (τ) and problem dimensions (n). **Top Row:** $n = 100$. **Middle Row:** $n = 500$. **Bottom Row:** $n = 1000$. **Left Column:** $\tau = 10^{-1}$. **Middle Column:** $\tau = 10^{-2}$. **Right Column:** $\tau = 10^{-3}$. In the top row, plots for `ZORO` and `adaZORO` are absent as they did not solve any problems to the required tolerance.

4.3 Discussion

On functions exhibiting gradient sparsity `ZORO-FA` outperforms existing benchmarks. Even though `ZORO` and `adaZORO` are given the exact gradient sparsity parameter, the fact that they cannot decrease their step-size or sampling radius means that their progress eventually plateaus. For the MGH test functions `ZORO` and `adaZORO` struggle, while `Nelder-Mead`, `DFQRM`, and `ZORO-FA` perform well. It was surprising, at least to us, that `Nelder-Mead` performed so well. We note however that this performance came at high computational expense, with `Nelder-Mead` typically taking 10–100 times longer to run.

4.4 Hidden Gradient Compressibility

In contrast to Section 4.1, the MGH functions of Section 4.2 *are not* explicitly designed to possess sparse or compressible gradients. Nonetheless we observe that `ZORO-FA` works well, and is frequently able to make progress with $s_k \ll n$. To investigate this further, we plot the magnitudes of the entries of $\nabla f(x)$, sorted from largest to smallest, for two MGH test functions (with $n = 500$) at 20 randomly selected x. Letting $|\nabla f(x)|_{(j)}$ denote the j-th largest-in-magnitude component of $\nabla f(x)$, we plot the mean of $|\nabla f(x)|_{(j)}$ over all 20 values of x, as well as the minimum and maximum values of $|\nabla f(x)|_{(j)}$ observed.

We also plot a trajectory of `ZORO-FA` applied to these two functions, as well as the sparsity levels found at each iteration (s_k). The results are displayed in Figure 3. As is clear, some MGH test functions (e.g., `rosex`) exhibit extreme gradient compressibility, while others (e.g., `trig`) do not. As suggested above, `ZORO-FA` can indeed make progress using s_k smaller than n. It is interesting that `ZORO-FA` uses lower values of s_k for the function (`trig`) whose gradients appear less compressible. We suspect this might be due to the interplay of the sparsity and Lipschitz continuity of the gradients. As `ZORO-FA` searches for a suitable sparsity level (s_j) and Lipschitz constant (σ_j) simultaneously, it might 'miss out' on gradient sparsity when the Lipschitz constant is large. This further emphasizes the need for further research into optimization methods which can take advantage of the presence of gradient sparsity, yet are robust to its absence.

5 Conclusion

In this paper, we presented a fully adaptive variant of `ZORO`, originally proposed in [8], for derivative-free minimization of a smooth functions with n variables and L-Lipschitz continuous gradient. The new method, called `ZORO-FA`, does not require the knowledge of the Lipschitz constant L or the effective sparsity level (when the gradients are compressible). At each iteration, `ZORO-FA` attempts to exploit a possible compressibility of gradient to compute a suitable gradient approximation *using fewer than* $\mathcal{O}(n)$ function evaluations. When the corresponding trial points fail to ensure a functional decrease of $\mathcal{O}\left(\epsilon^2\right)$, the method computes gradient approximations via forward finite-differences, which guarantee that the trial points will yield the desired functional decrease. With this safeguard procedure, we proved that, with probability one, `ZORO-FA` needs no more than $\mathcal{O}\left(n^2\epsilon^{-2}\right)$ function evaluations to find an ϵ-approximate stationary point of the objective function. When the gradient vectors of the objective function are p-compressible for some $p \in (0,1)$, we proved that, with high probability, `ZORO-FA` has an improved worst-case complexity of $\mathcal{O}\left(s(\theta,p)\ln(n)\epsilon^{-2}\right)$ function evaluations, where $\theta \in (0,1/2)$ is a user-defined parameter that controls the relative error of the gradient approximations, and $s(\theta,p)$ is the effective sparsity level as defined in Corollary 1. Our preliminary numerical results indicate that `ZORO-FA` is able to

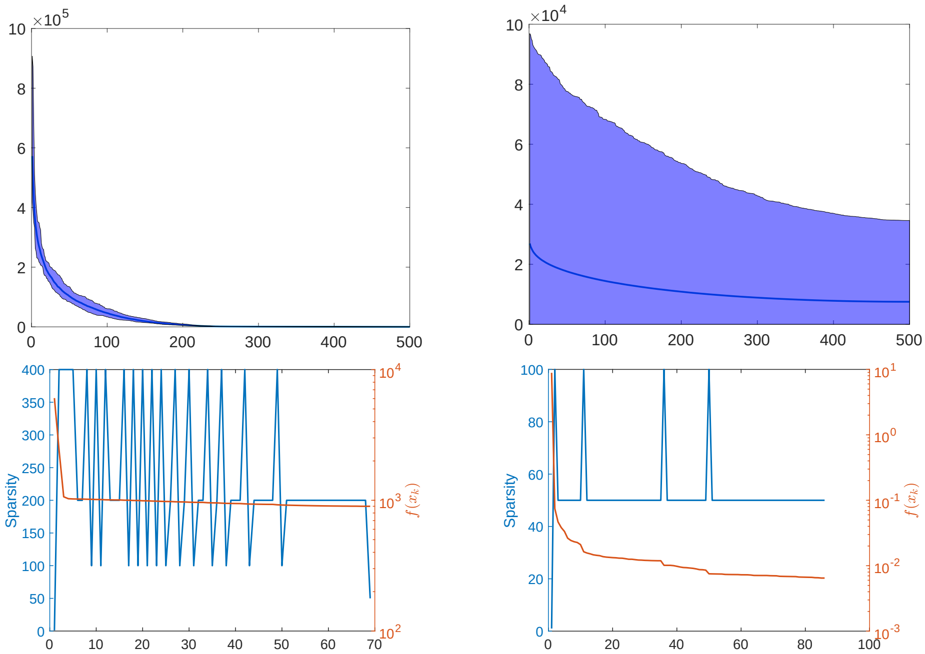

Fig. 3: **Top row:** Gradient magnitude profiles for two MGH functions: `rosex` (**Left**), and `trig` (**Right**). The solid line represents the mean (over 20 trials), while the shading represents the minimum and maximum values of gradient component magnitudes observed. **Bottom row:** The successful sparsity level for `ZORO-FA` (left axes) as well as the objective function values (right axes) for `rosex` (**Left**), and `trig` (**Right**).

exploit the presence of gradient compressibility, while being robust towards its absence.

Acknowledgements

We are very grateful to the associate editor and the two anonymous reviewers for their valuable comments and suggestions, which helped us improve this work.

References

1. Balasubramanian, K., Ghadimi, S.: Zeroth-order (non)-convex stochastic optimization via conditional gradient and gradient updates. Advances in Neural Information Processing Systems 31 (2018)
2. Bandeira, A.S., Scheinberg, K., Vicente, L.N.: Computation of sparse low degree interpolating polynomials and their application to derivative-free optimization. Mathematical Programming 134, 223–257 (2012)
3. Baraniuk, R., Davenport, M., DeVore, R., Wakin, M.: A simple proof of the restricted isometry property for random matrices. Constructive approximation 28, 253–263 (2008)

4. Bergou, E.H., Gorbunov, E., Richtárik, P.: Stochastic three points method for unconstrained smooth optimization. SIAM Journal on Optimization 30 (2020)
5. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 1, 281–305 (2012)
6. Cai, H., Lou, Y., McKenzie, D., Yin, W.: A zeroth-order block coordinate descent algorithm for huge-scale black-box optimization. PMLR, 1193–1203 (2021)
7. Cai, H., McKenzie, D., Yin, W., Zhang, Z.: A one-bit, comparison-based gradient estimator. Applied and Computational Harmonic Analysis 60, 242–266 (2022)
8. Cai, H., Mckenzie, D., Yin, W., Zhang, Z.: Zeroth-order regularized optimization (zoro): Approximately sparse gradients and adaptive sampling. SIAM Journal on Optimization 32, 687–714 (2022)
9. Cartis, C., Roberts, L.: Scalable subspace methods for derivative-free nonlinear least-squares optimization. Mathematical Programming 199, 461–524 (2023)
10. Cartis, C., Otemissov, A.: A dimensionality reduction technique for unconstrained global optimization of functions with low effective dimensionality. Information and Inference: A Journal of the IMA, 11(1), 167-201.
11. Cartis, C., Massart, E., Otemissov, A.: Bound-constrained global optimization of functions with low effective dimensionality using multiple random embeddings. Mathematical Programming, 198(1), 997-1058.
12. Chen, A., Zhang, Y., Jia, J., Diffenderfer, J., Liu, J., Parasyris, K., Zhang, Y., Zhang, Z., Kailkhura, B., Liu, S.: Deepzero: Scaling up zeroth-order optimization for deep model training. arXiv preprint arXiv:2310.02025 (2023)
13. Choramanski, K, Pacchiano, A., Parker-Harbor, J., Tang, Y., Jain, D., Yang, Y., Iscen, A., Hsu, J., Sindhwani, V.: Provably robust blackbox optimization for reinforcement learning. Conference on Robot Learning, 683–696 (2020)
14. Foucart, S.: Sparse recovery algorithms: sufficient conditions in terms of restricted isometry constants. Approximation Theory XIII: San Antonio 2010. Springer, 2012.
15. Grapiglia, G.N.: Quadratic regularization methods with finite-difference gradient approximations. Computational Optimization and Applications 85, 683–703 (2023)
16. Grapiglia, G.N.: Worst-case evaluation complexity of a derivative-free quadratic regularization method. Optimization Letters 18, 195–213 (2024)
17. Kim, B., Cai, H., McKenzie, D., Yin, W.: Curvature-aware derivative-free optimization. arXiv preprint arXiv:2109.13391 (2021)
18. Kozak, D., Becker, S., Doostan, A., Tenorio, L.: A stochastic subspace approach to gradient-free optimization in high dimensions. Computational Optimization and Applications 79, 339–368 (2021)
19. Kozak, D., Molinari, C., Rosasco, L., Tenorio, L., Villa, S.: Zeroth-order optimization with orthogonal random directions. Mathematical Programming, 199(1), 1179-1219 (2023)
20. Liu, Y., Zhu, Z., Gong, C., Cheng, M., Hsieh, C-J., You, Y.: Sparse MeZO: Less Parameters for Better Performance in Zeroth-Order LLM Fine-Tuning. arXiv preprint arXiv:2402.15751 (2024)
21. Malladi, S., Gao, T., Nichani, E., Damian, A., Lee, J.D., Chen, D., Arora, S.: Fine-tuning language models with just forward passes. Advances in Neural Information Processing Systems 36 (2024)
22. Mania, H., Guy, A., Recht, B.: Simple random search of static linear policies is competitive for reinforcement learning. Proceedings of the 32nd International Conference on Neural Information Processing Systems, 1805–1814 (2018)
23. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Transactions on Mathematical Software 7, 17–41 (1981)
24. Moré, J.J., Wild, S.M.: Benchmarking derivative-free optimization algorithms. SIAM Journal on Optimization 20, 172–191 (2009)
25. Nakamura, N., Seepaul, J., Kadane, J.B., Reeja-Jayan, B.: Design for low-temperature microwave-assisted crystallization of ceramic thin films. Applied Stochastic Models in Business and Industry 33, 314–321 (2017)
26. Needell, D., Tropp, J.A.: CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. Applied and computational harmonic analysis 26, 301–321 (2009)
27. Nelder, J.A., Mead, R.: A simplex method for function minimization. The Computer Journal 7, 308-313 (1965)
28. Nesterov, Yu., Spokoiny, V.: Random Gradient-Free Minimization of Convex Functions. Foundations of Computational Mathematics 17, 527–566 (2017)

29. Nesterov, Yu.: Introductory Lectures on Convex Optimization: A Basic Course. Vol 87. Spring, Berlin (2013)
30. Powell, M.J.D.: The NEWUOA software for unconstrained optimization without derivatives. Large-scale nonlinear optimization, 255–297 (2006)
31. Roberts, L., Royer, C.W.: Direct search based on probabilistic descent in reduced spaces. SIAM Journal on Optimization 33 (2023)
32. Salimans, T., Ho, J., Chen, X., Sidor, S., Sutskever, I.: Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint arXiv:1703.03864 (2017)
33. Slavin, I., McKenzie, D.: Adapting zeroth order algorithms for comparison-based optimization. arXiv preprint arXiv:2210.05824 (2022)
34. Shi, H.M., Xuan, M.Q., Oztoprak, F., Nocedal, J.: On the numerical performance of finite-difference-based methods for derivative-free optimization. Optimization Methods and Software 38, 289–311 (2023)
35. Tett, S., Gregory, J., Freychet, N., Cartis, C., Mineter, M., Roberts, L.: Calibrating Climate Models-what observations matter? EGU General Assembly Conference Abstracts (2022)
36. Wang, Z., Hutter, F., Zoghi, M., Matheson, D., De Feitas, N.: Bayesian optimization in a billion dimensions via random embeddings. Journal of Artificial Intelligence Research, 55, 361-387.
37. Wang, Y., Du, S., Balakrishnan, S., Singh, A.: Stochastic zeroth-order optimization in high dimensions. PMLR, 1356–1365 (2018)
38. Yue, P., Yang, L., Fang, C., Lin, Z.: Zeroth-order optimization with weak dimension dependency. In The Thirty Sixth Annual Conference on Learning Theory. PMLR, 4429–4472 (2023).
39. Zhang, Z.: PRIMA: Reference Implementation for Powell's Methods with Modernization and Amelioration. Available at `http://www.libprima.net,DOI:10.5281/zenodo.8052654` (2023)
40. Zhang, Y., Li, P., Hong, J., Li, J., Zhang, Y., Zheng, W., Chen, P-Y., Lee, J.D., Yin, W., Hong, M., Wang, Z., Liu, S., Chen, T.: Revisiting Zeroth-Order Optimization for Memory-Efficient LLM Fine-Tuning: A Benchmark. arXiv preprint arXiv:2402.11592 (2024)
41. R. Qiu, H. Tong: Gradient Compressed Sensing: A Query-Efficient Gradient Estimator for High-Dimensional Zeroth-Order Optimization. Proceedings of the Forty-first International Conference on Machine Learning (2024)

Declarations

– Conflict of interest: The authors have no conflicts of interest to declare that are relevant to the content of this article.
– Data availability: The Moré-Garbow-Hillstrom benchmarking functions used in the experiments of Section 4.2 can be downloaded at `https://arnold-neumaier.at/glopt/test.html`