A developed randomized algorithm with noise level tuning for large-scale noisy unconstrained DFO problems and its real-life applications

Morteza Kimiaei

Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, A-1090 Wien, Austria

Abstract

In this paper, a new randomized solver (called `VRDFON`) for noisy unconstrained derivative-free optimization problems is discussed. Complexity results in the presence of noise for nonconvex, convex, and strongly convex functions are studied. Two effective ingredients of `VRDFON` are an improved derivative-free line search algorithm with many heuristic enhancements and quadratic models in adaptively determined subspaces. Numerical results show that, on the large scale unconstrained `CUTEst` test problems contaminated by the absolute uniform noise, `VRDFON` is more efficient than several state-of-the-art solvers.

Keywords: Noisy derivative-free optimization, heuristic optimization, randomized line search method, complexity bounds, sufficient decrease
2000 AMS Subject Classification: 90C15, 90C30, 90C56.

1. Introduction

We consider the problem of finding a minimizer of the unconstrained derivative-free optimization (DFO) problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1.1}$$

thoroughly discussed in the books by Audet & Hare [1] and Conn et al. [2]. Here the smooth real-valued function $f : \mathbb{R}^n \to \mathbb{R}$ is known only by a noisy oracle which, for a given $x \in \mathbb{R}^n$, gives an approximation $\tilde{f}(x)$ to the exact function value $f(x)$, contaminated by the noise $\tilde{f}(x) - f(x)$. This problem is called the **noisy DFO problem**. We denote by $g(x)$ the unknown exact gradient vector of f at x and by $\tilde{g}(x)$ its approximation. The algorithm does not use knowledge of g, the Lipschitz constants of f, the structure of f, or the statistical properties of noise. Noise may be deterministic (caused by modelling, truncation, and/or discretization errors) or stochastic (caused by inaccurate measurements or rounding errors).

There are many DFO methods (model-based, line search, direct search, and evolution strategy) for solving noisy DFO problems of the form (1.1), see the survey paper by Larson et al. [3]:

• Model-based methods approximate $\tilde{f}$ at each trial point by an approximate quadratic model through fitting or interpolation, find an approximate solution of this model restricted to a region around the trial point to avoid large steps, and only accept trial points with low inexact function values.

• Line search methods perform extrapolation steps along random or coordinate directions or their opposite directions with the goal of accepting trial points with low inexact function values by using a line search condition.

• Direct search methods search along coordinate directions, directions from a fixed poll set, or random directions, using a decrease condition, like the line search condition, only to accept points with low inexact function values.

• Matrix adaptation evolution strategy methods repeatedly generate a finite number of individuals, select some individuals to generate parents, and choose a new mean for the distribution.

We propose a new randomized algorithm for noisy unconstrained DFO problems – called **Vienna noisy randomized derivative-free optimization** (`VRDFON`). Following the classifications of Larson et al. [3] and Rios & Sahinidis [4], `VRDFON` is a local randomized model-based line search-based solver. This solver is a development of the noiseless `VRBBO` solver (Kimiaei & Neumaier [5]) to handle the variability of noise intensity.

`VRDFON` repeatedly performs `DS`, a decrease search, using `MLS`, a multi-line search algorithm. `MLS` is likely to reduce inexact function values, reaching regions close to an approximate stationary point, and finally finding an approximate stationary point. This can be done by performing `extrapolate`, an extrapolation step, along a finite number of random directions or their opposite directions, using a line search condition to accept points with low inexact function values.

An implemented version of VRDFON uses the following new features:

• New heuristics are used to find and update step sizes in an implemented version of MLS, so that step sizes are neither too small nor too large to avoid line search failure.

• Step sizes of MLS are heuristically changed at the end of an implemented version of DS if these step sizes are too small. The goal is to avoid getting stuck before an approximate stationary point is found.

• Surrogate quadratic models are constructed in adaptively determined subspaces that can handle medium and large scale problems.

• Several new directions (random approximate coordinate, perturbed random, and improved trust region) are generated so that an implemented version of MLS can be performed not only along random scaled directions required to achieve complexity results, but also along these new directions to avoid large steps, which is a source of line search failure.

Section 2 discusses some concepts and assumptions required to obtain complexity results for DFO methods. Then Section 3 explains the VRDFON algorithm. Complexity results of VRDFON for all cases with a given probability arbitrarily close to one in the presence of noise are proved in Section 4. Section 5 provides a comparison between VRDFON and several state-of-the-art DFO solvers on the large and very large scale noisy problems obtained from the noiseless unconstrained CUTEst test problems from the collection of Gould et al. [6]. In Section 6, it is shown that how VRDFON finds an approximate stationary point of a real-life problem Our findings on the features VRDFON from numerical results are summarized in Section 7.

2. Preliminaries

In this section, we present a list of known DFO solvers, state the assumptions required to obtain the complexity results of DFO methods and summarize some known limit accuracy and complexity results.

Table 1 includes a list of deterministic or randomized DFO solvers which use at least one of model-based, line search-based, direct search, and evolution strategy algorithms. By comparing these solvers, we can see the quality of a new composite algorithm compared to any single algorithm that forms it, or to other solvers that use a single or composite algorithm. This helps us know how to construct a new composite algorithm efficiently or robustly; for example, `NOMAD` and `MCS` use model-based direct search algorithms that are more robust and efficient than their model-free versions.

solver	model-based	line search	direct search	evolution strategy	deterministic	randomized	Reference
`BCDFO`	+	−	−	−	+	−	Gratton et al. [7]
`UOBYQA`	+	−	−	−	+	−	Powell [8]
`NEWUOA`	+	−	−	−	+	−	Powell [9]
`SnobFit`	+	−	−	−	+	−	Huyer & Neumaier [10]
`GRID`	+	−	−	−	+	−	Elster & Neumaier [11]
`MCS`	+	−	+	−	+	−	Huyer & Neumaier[12]
`NOMAD`	+	−	+	−	+	−	[13, 14, 15, 16]
`VRDFON`	+	+	−	−	+	+	present paper
`subUOBYQA`	+	−	−	−	+	−	present paper
`subNEWUOA`	+	−	−	−	+	−	present paper
`VRBBO`	−	+	−	−	+	+	Kimiaei & Neumaier [5]
`SDBOX`	−	+	−	−	+	−	Lucidi & Sciandrone [17]
`FMINUNC`	−	+	−	−	+	−	Matlab Optimization Toolbox
`DSPFD`	−	−	+	−	−	+	Gratton et al. [18]
`BFO`	−	−	+	−	+	+	Porcelli & Toint [19]
`NMSMAX`	−	−	+	−	+	−	Higham [20]
`subNMSMAX`	−	−	+	−	+	−	present paper
`CMAES`	−	−	−	+	−	+	Auger & Hansen [21]
`LMMAES`	−	−	−	+	−	+	Loshchilov et al. [22]
`fMAES`	−	−	−	+	−	+	Beyer [23]
`BiPopMAES`	−	−	−	+	−	+	Beyer & Sendhoff [24]

Table 1: A list of DFO solvers needed in this paper. `subUOBYQA`, `subNEWUOA`, and `subNMSMAX` are, respectively, `UOBYQA`, `NEWUOA`, and `NMSMAX` in random subspaces to handle problems in medium and high dimensions.

A **complexity bound** of an algorithm for noisy DFO problems in the form (1.1) is an upper bound on the number of function evaluations to find an approximate point x (unknown to us since L and $g(x)$ are unknown) near a local optimizer whose unknown exact gradient norm is below a given fixed threshold $\omega > 0$ (unknown to us but appearing in our complexity bound) and whose function value $f(x)$ is sufficiently small compared to the initial function value $f(x^0)$, i.e.,

$$f(x) \le \sup\Big\{f(y) \mid y \in \mathbb{R}^n, \quad f(y) \le f(x^0), \quad \|g(y)\| = \mathcal{O}(\sqrt{n\omega L})\Big\}. \tag{2.1}$$

Here $\|.\|$ is the Euclidean norm.

To analyze the limit accuracy and the complexity of our algorithm for solving noisy DFO problems in the form (1.1), we assume, like other DFO methods, see, e.g., Bergou et al. [25], Gratton et al. [18], and Kimiaei & Neumaier [5], that
(A1) the function f is continuously differentiable on $\mathbb{R}^n$, and its gradient is Lipschitz continuous with Lipschitz constant L,
(A2) the level set $\mathcal{L}(x^0) := \{x \in \mathbb{R}^n \mid f(x) \le f(x^0)\}$ of f at x^0 is compact, and
(A3) the approximation $\tilde{f}(x)$ of f at $x \in \mathbb{R}^n$ satisfies

$$|\tilde{f}(x) - f(x)| \le \omega. \tag{2.2}$$

In the noiseless case $\omega = 0$, (A3) implies $\tilde{f} = f$. (A2) implies that

$$\widehat{f} := \inf\{f(x) \mid x \in \mathbb{R}^n\} = f(\widehat{x}) > -\infty \tag{2.3}$$

for any global minimizer $\widehat{x}$ of (1.1).

For the noiseless case, see Larson et al. [3, Table 8.1] and Kimiaei & Neumaier [5, Tables 1–3] for a summary of known results on worst case complexity and corresponding references. To obtain $\|g(x)\| \le \varepsilon$ (under the assumptions (A1) and (A2)), one needs
• $\mathcal{O}(\varepsilon^{-2})$ function evaluations for the general case,
• $\mathcal{O}(\varepsilon^{-1})$ function evaluations for the convex case,
• $\mathcal{O}(\log \varepsilon^{-1})$ function evaluations for the strongly convex case. In all cases, the factors are ignored. Randomized algorithms typically have complexity bounds that are a factor n better than those of deterministic algorithms, see [26].

In the presence of noise, the limit accuracy and complexity results of some algorithms have been investigated. We summarized only the results of line search based in Table 2. Other useful references for complexity results of stochastic DFO methods are Chen [27], Dzahini [28], and Blanchet et al. [29].

Table 2: Known limit accuracy and complexity noisy DFO methods regardless of L and n. As stated in the introduction, $\tilde{g}(x)$ stands for the estimated gradient at x and $\widehat{f}$ is the function value at any global minimizer $\widehat{x}$.

type of noise	theoretical result
deterministic	nonconvex: $\|g\| = \mathcal{O}(\sqrt{\omega})$
assumptions:	(A1)–(A3)
reference	Lucidi & Sciandrone [17] and Elster & Neumaier [11]
deterministic	strongly convex: $f - \widehat{f} = \mathcal{O}(\omega)$
assumptions:	(A1)–(A3)
reference:	Berahas et al. [30]
stochastic	nonconvex: $\mathcal{O}(\varepsilon^{-2})$ with $\mathbf{E}(\|g\|) \le \varepsilon$
	convex: $\mathcal{O}(\varepsilon^{-1})$ with $\mathbf{E}(\|g\|) \le \varepsilon$, $\mathbf{E}(f - \widehat{f}) \le \varepsilon$
	strongly convex: $\mathcal{O}(\log \varepsilon^{-1})$ with $\mathbf{E}(\|g\|) \le \varepsilon$, $\mathbf{E}(f - \widehat{f}) \le \varepsilon$
assumptions:	(A1)–(A3) and norm condition:
	$\|\tilde{g}(x) - g(x)\| \le \theta \|g(x)\|$ for some $0 < \theta < 1$
reference:	Berahas et al. [31]

3. Proposed VRDFON algorithm

In this section we describe the VRDFON algorithm and how it works. Until an approximate stationary point is found, VRDFON repeatedly calls DS, which has a finite number of calls to MLS, using extrapolate to leave regions close to saddle point or maximizer. extrapolate uses a line search condition to accept points with low inexact function values in regions close to an approximate stationary point.

An iteration of MLS is called **successful** if at least one reduction of the inexact function value is found and **unsuccessful** otherwise. An iteration of DS is called **successful** if MLS has at least one successful iteration. An iteration of VRDFON is called **successful** if DS has at least one successful iteration.

We denote by x_{best} the **overall best point** and by $\tilde{f}_{\text{best}} := \tilde{f}(x_{\text{best}})$ the **overall best inexact function value** of `VRDFON`, i.e., the final best point and its inexact function value found by `DS`. Indeed, the overall best point is an ε-approximate stationary point of the sequence x^k ($k = 1, 2, 3, \ldots$) after `VRDFON` terminates at a finite number of iterations. To simplify our algorithms, all tuning parameters are given once in line 1 of `VRDFON` and are not mentioned as input for the other algorithms.

Algorithm 1 `VRDFON`**, a randomized method for noisy DFO problems**

```
1: Tuning parameters: $Q > 1$ (factor for reducing $\delta_k$), $0 < \gamma_{\text{rd}} < 1$ (parameter for scaling random directions), $0 < \gamma < 1$ (parameter for line search), $\gamma_e > 1$ (factor for updating step size inside MLS), $\delta_{\max} > 0$ (initial value for $\delta_k$), $0 \le \delta_{\min} \le 1$ (minimum threshold for $\delta_k$), $0 < \eta < \frac{1}{2}$ (parameter for $R_m$), $R_m := \lceil \log_2 \eta^{-1} \rceil \ge 2$ (number of random direction in each MLS), $T_0 \ge 1$ (number of calls to MLS by DS).
2: Set $\delta_0 := \delta_{\max}$ and compute $\tilde{f}(x^0)$.
3: for $k = 0, 1, 2, \ldots$ do
4:     $[x^{k+1},\ \tilde{f}(x^{k+1})] =$DS$(\delta_k,\ x^k,\ \tilde{f}(x^k))$.
5:     if $\delta_k \le \delta_{\min}$, $x_{\text{best}} := x^{k+1}$ and $\tilde{f}_{\text{best}} := \tilde{f}(x^{k+1})$; stop; end if
6:     if $\tilde{f}(x^{k+1}) \ge \tilde{f}(x^k)$ then, $\delta_{k+1} := \delta_k / Q$.
7:     else, $\delta_{k+1} := \delta_k$.
8:     end if
9: end for
```

`VRDFON` initializes the initial step size $\delta_0 := \delta_{\max} > 0$, which is a tuning parameter, and computes the inexact function value $\tilde{f}(x^0)$ at the initial point x^0. In each iteration, `VRDFON` calls `DS` to hopefully find at least one reduction of the inexact function value. Once the step size δ_k is below a minimum threshold $0 \le \delta_{\min} < 1$, which is a tuning parameter, `VRDFON` terminates with the overall best point $x_{\text{best}} := x^{k+1}$ and its inexact function value $\tilde{f}(x_{\text{best}}) := \tilde{f}(x^{k+1})$ in the $(k+1)$th iteration. Otherwise, if `DS` cannot find any reduction of the inexact function value the step size δ_{k+1} is reduced by a factor of $Q > 1$, which is a tuning parameter; otherwise, δ_{k+1} is δ_k. `VRDFON` tries to find an ε-approximate stationary point x_{best} that satisfies $\|g(x_{\text{best}})\| \le \varepsilon$ for a threshold $\varepsilon = \sqrt{\omega}$ before the condition $\delta_k \le \delta_{\min}$ is satisfied. If $\delta_{\min} = 0$ is chosen, for a finite termination of `VRDFON`, three other stopping tests are needed, see Section 5.3.

Since `VRDFON` has $1 \le K < \infty$ calls to `DS` and `DS` has $1 \le T_0 < \infty$ calls to `MLS`, `VRDFON` has KT_0 calls to `MLS`. As defined in line 1 of `VRDFON`, $R_m = \lceil \log_2 \eta^{-1} \rceil$ for a given $0 < \eta < \frac{1}{2}$ is the number of random directions used by `MLS`. For given $1 \leq T_0 < \infty$ and $1 \leq K < \infty$, defined by

$$R_d := T_0 R_m = T_0 \lceil \log_2 \eta^{-1} \rceil \geq \log_2 \eta^{-T_0} \tag{3.1}$$

is the number of random directions used by `DS` and defined by

$$R_v := K R_d = K T_0 R_m = K T_0 \lceil \log_2 \eta^{-1} \rceil \geq \log_2 \eta^{-KT_0} \tag{3.2}$$

is the number of random directions used by `VRDFON`.

3.1. `DS`*, a decrease search*

This subsection discusses `DS` and how it works. The goal of `DS` is to find a significant decreases $\tilde{f}$ by performing `MLS`.

Algorithm 2 `DS`**, a decrease search**

```
    function   $[x^{k+1},\ \tilde{f}(x^{k+1})] = \mathtt{DS}(\delta_k,\ x^k,\ \tilde{f}(x^k))$
1: $y^0 := x^k$ and $\tilde{f}(y^0) := \tilde{f}(x^k)$.
2: for $t = 1, \ldots, T_0$ do, $[y^t,\ \tilde{f}(y^t)] = \mathtt{MLS}(\delta_k,\ y^{t-1},\ \tilde{f}(y^{t-1}))$; end for
3: $x^{k+1} := y^{T_0}$ and $\tilde{f}(x^{k+1}) := \tilde{f}(y^{T_0})$.
```

`DS` generates the sequences y^t and $\tilde{f}(y^t)$ $(t = 1, \ldots, T_0)$. First, it initializes two sequences y^t and $\tilde{f}(y^t)$ $(t = 1, \ldots, T_0)$. Subsequently, `DS` has T_0 calls to `MLS` to find reductions of the inexact function values. If no reduction is found, then

$$x^{k+1} = y^{T_0} = x^k \quad \text{and} \quad \tilde{f}(x^{k+1}) = \tilde{f}(y^{T_0}) = \tilde{f}(x^k).$$

Otherwise, at least one reduction of the inexact function value is found, so that

$$x^{k+1} = y^{T_0} \neq x^k \quad \text{and} \quad \tilde{f}(x^{k+1}) = \tilde{f}(y^{T_0}) < \tilde{f}(x^k).$$

3.2. `MLS`*, a multi line search*

This subsection discusses the main component `MLS` of `DS`, whose goal is to perform extrapolation along scaled random directions (discussed in 3.3) or their opposite directions and enter or move along a valley to achieve an approximate stationary point.

Algorithm 3 `MLS`**, a multi line search**

```
function    $[y^t,\ \tilde f(y^t)] = \texttt{MLS}(\delta_k,\ y^{t-1},\ \tilde f(y^{t-1}))$
1: Set $\alpha_1 := \delta_k$, $z_{\rm best} := y^{t-1}$, and $\tilde f_{\rm best} := \tilde f(y^{t-1})$.
2: for  $r = 1, \dots, R_m$ do
3:     Compute the scaled random direction $p^r$.
4:     Compute $z^r := z_{\rm best} + \alpha_r p^r$ and $\tilde f_r = \tilde f(z^r)$.
5:     if $\tilde f_{\rm best} - \tilde f_r > \gamma\alpha_r^2$, then                ▷ extrapolation along $p^r$
6:         $[z_{\rm best}, \tilde f_{\rm best}] = \texttt{extrapolate}(z_{\rm best}, \tilde f_{\rm best}, \tilde f_r, \alpha_r, p^r)$.
7:     else
8:         Compute $p^r := -p^r$, $z^r := z_{\rm best} + \alpha_r p^r$, and $\tilde f_r = \tilde f(z^r)$.
9:         if $\tilde f_{\rm best} - \tilde f_r > \gamma\alpha_r^2$ then            ▷ extrapolation along $-p^r$
10:            $[z_{\rm best}, \tilde f_{\rm best}] = \texttt{extrapolate}(z_{\rm best}, \tilde f_{\rm best}, \tilde f_r, \alpha_r, p^r)$.
11:        else
12:            $\alpha_{r+1} := \alpha_r/\gamma_e$.                             ▷ no decrease in $\tilde f^r$
13:        end if
14:    end if
15: end for
16: Set $y^t := z_{\rm best}$ and $\tilde f(y^t) := \tilde f_{\rm best}$.
```

The condition $\tilde f_{\rm best} - \tilde f_r > \gamma\alpha_r^2$ (used in lines 5 and 9 of Algorithm 3) is called *line search condition* whose goal is to identify whether the inexact function value at the rth point $z^r = z_{\rm best} + \alpha_r p^r$ is decreased or not. If this condition holds, we say that $\gamma\alpha_r^2$-sufficient gain ($r \in \{1, 2, \dots, R_m\}$) is found along the search directions p^r.

`MLS` generates the sequences z^r and $\tilde f(z^r)$ ($r = 1, \dots, R_m$). It initializes the extrapolation step size α_r and the point $z_{\rm best}$ (found in the $(t-1)$th iteration of `DS`) and its inexact function value $\tilde f(z_{\rm best})$. `MLS` includes a for loop. In the rth iteration, the scaled random direction p^r is computed. Then, if the line search condition holds, extrapolation along one of $\pm p^r$ can be performed. If extrapolation cannot be performed, α_r is reduced.

3.3. Scaled random directions

In this subsection, we describe how scaled random directions are generated and show that these directions satisfy a two-sided angle condition with probability at least half.

We define a standard random direction as a random direction p drawn uniformly **i.i.d.** (independent and identically distributed) in $[-\frac{1}{2}, \frac{1}{2}]^n$. A **scaled random direction** is a standard random direction p scaled by $\gamma_{\rm rd}/\|p\|$, where $0 < \gamma_{\rm rd} < 1$ is a tiny tuning parameter, resulting in $\|p\| = \gamma_{\rm rd}$. The scaling of the direction p by $\gamma_{\rm rd}$ is the same as the scaling of the direction p by δ in [5, (17)]. Therefore, our scaled random direction is the scaled random direction of [5, (17)].

Essential for our complexity bounds is the following result (Proposition 2 in [5]) for the unknown gradient $g(x)$ of $f(x)$ at $x \in \mathbb{R}^n$. It holds for any norm; hence for any scaling vector $s \in \mathbb{R}^n$.

Proposition 3.1. *Any scaled random direction p satisfies the inequality*

$$\Pr\Big(\|g(x)\|\|p\| \leq 2\sqrt{cn}|g(x)^T p|\Big) \geq \frac{1}{2} \tag{3.3}$$

with a positive constant $c \leq 12.5$.

The condition (3.3) is called **two-sided angle condition** because we cannot check whether any scaled random direction is a descent direction or not. Hence, instead of searching along one ray $\alpha > 0$ only, our line search allows to search the line $x + \alpha p$ in both directions ($\alpha \in \mathbb{R}$).

3.4. `extrapolate`*, an extrapolation step*

This subsection discusses the main component `extrapolate` of `MLS`, whose goal is to discard points located in regions near a saddle point or maximizer and accept points located in regions near an approximate minimizer. This can be done by a line search condition.

`extrapolate` finds at least one reasonable reduction of the inexact function value. It speeds up reaching an approximate minimizer. Indeed, as long as reductions of the inexact function values are found, extrapolation step sizes are expanded and new trial points and their inexact function values

Algorithm 4 `extrapolate`**, an extrapolation step**

```
function    $[z_{\text{best}}, \tilde{f}_{\text{best}}] = \texttt{extrapolate}(z_{\text{best}}, \tilde{f}_{\text{best}}, \tilde{f}_z, \alpha, p)$
1: while true do
2:     Set $\tilde{f}_e = \tilde{f}_z$ and expand $\alpha = \gamma_e \alpha$.
3:     Compute $z := z_{\text{best}} + \alpha p$ and $\tilde{f}_z = \tilde{f}(z)$.
4:     if $\tilde{f}_{\text{best}} - \tilde{f}_z \leq \gamma \alpha^2$
5:         $\alpha := \alpha / \gamma_e$, $z_{\text{best}} := z_{\text{best}} + \alpha p$, and $\tilde{f}_{\text{best}} := \tilde{f}_e$; stop.
6:     end if
7: end while
```

are computed. Once, no decrease in the inexact function value is found, `extrapolate` ends.

Since the inexact function value at the last point evaluated by `extrapolate` does not decrease, the penultimate point is accepted as the new best point in line 5, whose inexact function value was already stored in $\tilde{f}_e$ in line 2.

4. Convergence and complexity analysis of `VRDFON`

In this section, we prove for `VRDFON` complexity results with probability arbitrarily close to one for nonconvex, convex, and strongly convex functions in the presence of noise. In all cases, the order of ω in our bounds is the same as in Berahas et al. [31]. In contrast to the method of Berahas et al. [31], which uses the norm condition defined in Table 2, our line search does not use the approximate directional derivative $\tilde{g}^T p$ in the line search condition, but $\gamma\alpha^2$ with $0 < \gamma < 1$ because the estimation of the gradient may be inaccurate in the presence of high noise, leading to failure of the line search algorithm. However, we estimate the gradient to generate different heuristic directions in Section 5.2. Therefore, we obtain our complexity bound regardless of the norm condition since the nature of the line search algorithms is different. Our bounds are obtained with high probability, while the results of Berahas et al. [31] are valid in expectation.

In addition to (A1)–(A3) to obtain our complexity results we assume the following assumption:
(A4) The tuning parameter $\gamma_e > 1$ and the step size δ_k satisfy the condition

$$\gamma_e^{1-R_m} \delta_k \geq \alpha_{\min}$$

for a fixed minimum threshold $0 < \alpha_{\min} < \infty$ and a fixed small positive integer R_m.

Under the assumptions (A1)–(A4), this section proves that, with a given probability arbitrarily close to 1, `VRDFON` terminates after at most
• $\mathcal{O}(R_m T_0 L\omega^{-1})$ function evaluations in the general case,
• $\mathcal{O}(\sqrt{n} R_m T_0 L^{3/2}\omega^{-1/2})$ function evaluations in the convex case,
• $\mathcal{O}(nR_m T_0 L \log(\omega^{-1}))$ function evaluations in the strongly convex case
with a point x (unknown to us because gradients and Lipschitz constants are unknown), satisfying (2.1).

For the choice $R_m = 2$ and $T_0 = 1$, our bounds are better by a factor n than in the deterministic case (see Bandeira et al. [26] and Kimiaei & Neumaier [5]), but numerically these factors should be large to increase the efficiency and robustness of our algorithm. In practice, our algorithm has better numerical performance if R_m and T_0 are chosen such that $R_m T_0 \sim n$, with the same complexity bound as in the deterministic case.

4.1. An auxiliary result

The following result generalizes shows that if the rth iteration of `MLS` is unsuccessful, a useful bound for the directional derivative can be found. In this case, no $\gamma\alpha_r^2$-sufficient gain ($r \in \{1, 2, \ldots, R_m\}$) is found along the search directions $\pm p^r$.

Proposition 4.1. *Let $\{z^r\}$ ($r = 1, 2, \ldots, R_m$) be the sequence generated by* `MLS` *in the $(k+1)$th iteration of* `VRDFON`*. Moreover, suppose that (A1)–(A3) hold and $0 < \gamma < 1$. Then, for all $z^r, p^r \in \mathbb{R}^n$, at least one of the following holds:*
(i) $\tilde f(z_{\text{best}} + \alpha_r p^r) < \tilde f(z^r) - \gamma\alpha_r^2$,
(ii) $\tilde f(z_{\text{best}} + \alpha_r p^r) > \tilde f(z^r) + \gamma\alpha_r^2$ *and* $\tilde f(z_{\text{best}} - \alpha_r p^r) < \tilde f(z^r) - \gamma\alpha_r^2$,
(iii) $|g(z^r)^T p^r| \le \gamma\alpha_r + 2\omega/\alpha_r + \frac{1}{2}L\alpha_r\|p^r\|^2$.
Here, in the kth iteration of `VRDFON`*, z_{best} is the best point found by* `MLS`*.*

Proof. (A1) results in

$$\alpha_r g(z^r)^T p^r - \frac{1}{2}L\alpha_r^2\|p^r\|^2 \le f(z_{\text{best}}+\alpha_r p^r) - f(z^r) \le \alpha_r g(z^r)^T p^r + \frac{1}{2}L\alpha_r^2\|p^r\|^2. \tag{4.1}$$

We assume that (iii) is violated, so that

$$|g(z^r)^T p^r| > \gamma\alpha_r + 2\omega/\alpha_r + \frac{1}{2}L\alpha_r\|p^r\|^2. \tag{4.2}$$

We consider the proof in the two cases:
CASE 1. If $g(z^r)^T p^r \le 0$, then from (2.2) and (4.2) we get

$$\begin{aligned}
\tilde{f}(z_{\text{best}} + \alpha_r p^r) - \tilde{f}(z^r) &\le f(z_{\text{best}} + \alpha_r p^r) - f(z^r) + 2\omega \\
&\le \alpha_r g(z^r)^T p^r + \frac{1}{2}L\alpha_r^2\|p^r\|^2 + 2\omega \\
&= -\alpha_r|g(z^r)^T p^r| + \frac{1}{2}L\alpha_r^2\|p^r\|^2 + 2\omega \\
&< -\gamma\alpha_r^2, \qquad (4.3)
\end{aligned}$$

meaning that (i) must hold.
CASE 2. If $g(z^r)^T p^r \ge 0$, then from (2.2) and (4.2) we get

$$\begin{aligned}
\tilde{f}(z_{\text{best}} - \alpha_r p^r) - \tilde{f}(z^r) &\le f(z_{\text{best}} - \alpha_r p^r) - f(z^r) + 2\omega \\
&\le g(z^r)^T(-\alpha_r p^r) + \frac{1}{2}L\alpha_r^2\|p^r\|^2 + 2\omega \\
&= -\alpha_r|g(z^r)^T p^r| + \frac{1}{2}L\alpha_r^2\|p^r\|^2 + 2\omega \\
&< -\gamma\alpha_r^2, \qquad (4.4)
\end{aligned}$$

Therefore, the second inequality in (ii) holds. By (2.2), (4.1), and (4.2) the first half

$$\begin{aligned}
\tilde{f}(z_{\text{best}} + \alpha_r p^r) - \tilde{f}(z^r) &\ge f(z_{\text{best}} + \alpha_r p^r) - f(z^r) - 2\omega \\
&\ge \alpha_r g(z^r)^T p^r - \frac{1}{2}L\alpha_r^2\|p^r\|^2 - 2\omega > \gamma\alpha_r^2
\end{aligned}$$

must be satisfied. Hence the first inequality in (ii) holds. □

This proposition is used to get a worst-case complexity for `MLS` in Subsection 4.2.

4.2. Complexity for `MLS`

This subsection discusses a worst-case complexity for `MLS`. Assuming (A1)–(A4), we prove that one of the following holds:

(i) If at least on iteration ($r' \in \{1, 2, \ldots, R_m\}$) of `MLS` is successful, a minimum reduction in the inexact function value is found when $-p^{r'}$ is tried.
(ii) An upper bound on the unknown gradient norm of at least one $z^{r''}$ ($r'' \in \{1, 2, \ldots, R_m\}$) of the points generated by the unsuccessful iterations of `MLS` is found with a given probability arbitrarily close to one. In fact, it is not clear to us which point, since the gradients and Lipschitz constants are not available.

Theorem 4.1. *Assume that (A1)–(A4) hold, $0 < \eta < \frac{1}{2}$, $0 < \gamma_{\rm rd} < 1$, $\gamma_e > 1$, and $0 < \gamma < 1$. Moreover, define*

$$\overline{L} := 2\gamma/\gamma_{\rm rd} + L\gamma_{\rm rd} \tag{4.5}$$

and let $\{z^r\}$ ($r = 1, 2, \ldots, R_m$) be the sequence generated by `MLS` *in the $(k+1)$th iteration of* `VRDFON`*. Then one of the following happens:*
(i) If at least `MLS` *has a successful iteration, $r' \in \{1, 2, \ldots, R_m\}$, then it decreases the inexact function value by at least $\gamma\alpha_{r'}^2$.*
(ii) If `MLS` *has no successful iteration, then at least one $z^{r''}$ ($1 \le r'' \le R_m$) of the points evaluated by the unsuccessful iterations of* `MLS`*, with the probability at least $1 - \eta$, has an unknown gradient $g(z^{r''})$ satisfying*

$$\|g(z^{r''})\| \le \sqrt{cn}\,\Gamma(\delta_k) \quad \textit{with } \Gamma(\delta_k) := \overline{L}\delta_k + \gamma_e^{R_m-1}\frac{4\omega}{\gamma_{\rm rd}\delta_k} \tag{4.6}$$

for a given $0 < \eta < \frac{1}{2}$. Here c comes from Proposition 3.1, δ_k is fixed in `MLS`*, independent of r, and is updated outside* `MLS`*.*

Proof. Let $\mathcal{R} := \{1, \ldots, R_m\}$. We denote by p^r the rth scaled random search direction, by z^r the rth point, and by $\alpha_r = \gamma_e^{1-r}\delta_k \ge \alpha_{\min}$ the rth step size from (A4).

(i) Let $r' \in \{1, 2, \ldots, R_m\}$. The worst case requires $2R_m + 1$ function evaluations and assumes that the r'th iteration of `MLS` is successful and the other iterations are unsuccessful. In the unsuccessful iterations, two function values are computed along the directions $\pm p^r$ ($r \in \mathcal{R} \setminus \{r'\}$), but in the r'th iteration which is successful, no reduction of the inexact function value is found when $p^{r'}$ is attempted, while reduction of the inexact function value is found when $-p^{r'}$ is attempted. Therefore, an extrapolation step along $-p^{r'}$ is performed with at most two additional function evaluations and the $\gamma\alpha_{r'}^2$-sufficient gain. Consequently, (i) is verified.

(ii) Suppose that $\tilde{f}(z^r)$ does not decrease by more than $\gamma\alpha_r^2$ for all $r \in \mathcal{R}$; all iterations are unsuccessful. Then we define $\Gamma_0(\alpha_r) := \overline{L}\alpha_r + \dfrac{4\omega}{\gamma_{\rm rd}\alpha_r}$. Since $\Gamma_0(\alpha_r)$ for $\alpha_r > 0$ is a convex function, we obtain for $r \in \mathcal{R}_m$

$$\begin{aligned} \Gamma_0(\alpha_r) &\leq \max\{\Gamma_0(\alpha_1), \Gamma_0(\alpha_R)\} < \overline{L}\alpha_1 + \frac{4\omega}{\gamma_{\rm rd}\alpha_R} \\ &= \Gamma(\delta_k) = \overline{L}\delta_k + \gamma_e^{R_m-1}\frac{4\omega}{\gamma_{\rm rd}\delta_k}, \end{aligned} \tag{4.7}$$

where $\alpha_1 := \max\limits_{r\in\mathcal{R}}\{\alpha_r\} = \delta_k > \alpha_R := \min\limits_{r\in\mathcal{R}}\{\alpha_r\} = \gamma_e^{1-R_m}\delta_k$ since $\gamma_e > 1$ and $R_m \geq 2$. Then we obtain from Proposition 4.1 and since $\|p^r\| = \gamma_{\rm rd}$ (due to the definition of the scaled random direction in Section 2), for all $r \in \mathcal{R}$,

$$|g(z^r)^T p^r| \leq \gamma\alpha_r + 2\omega/\alpha_r + \frac{L}{2}\alpha_r\|p^r\|^2 = \gamma\alpha_r + 2\omega/\alpha_r + \frac{L}{2}\gamma_{\rm rd}^2\alpha_r,$$

so that for all $r \in \mathcal{R}$ and from (4.7), the inequality

$$\begin{aligned} \|g(z^r)\| &= \|g(z^r)\|\|p^r\|/\gamma_{\rm rd} \leq 2\sqrt{cn}|g(z^r)^T p^r|/\gamma_{\rm rd} \\ &\leq \sqrt{cn}\left(\Big(\frac{2\gamma}{\gamma_{\rm rd}} + L\gamma_{\rm rd}\Big)\alpha_r + \frac{4\omega}{\gamma_{\rm rd}\alpha_r}\right) = \sqrt{cn}\Gamma_0(\alpha_r) < \sqrt{cn}\Gamma(\delta_k) \end{aligned}$$

holds with probability $\frac{1}{2}$ or more according to Proposition 3.1. In other words,

$$\Pr\Big(\|g(z^r)\| > \sqrt{cn}\Gamma(\delta_k)\Big) < \frac{1}{2}, \quad \text{for any fixed } r \in \mathcal{R}.$$

Therefore, we find at least one of the gradients $g = g(z^{r''})$ ($r'' \in \mathcal{R}$) such that (4.6) holds, that is, for a given $0 < \eta < \frac{1}{2}$

$$\begin{aligned} \Pr\Big(\|g\| \leq \sqrt{cn}\Gamma(\delta_k)\Big) &= 1 - \prod_{r=1}^{R_m}\Pr\Big(\|g(z^r)\| > \sqrt{cn}\Gamma(\delta_k)\Big) \\ &\geq 1 - 2^{-R_m} \geq 1 - \eta. \end{aligned}$$

□

4.3. Complexity for `DS`

This subsection discusses a worst-case complexity for `DS`. Assuming (A1)–(A4), we prove that either an upper bound on the number of function evaluations is found or an upper bound on the unknown gradient norm of at least one of the points generated by the unsuccessful iterations of DS is found with a given probability arbitrarily close to one in the presence of noise; in fact, it is not clear to us which point, since the gradients and Lipschitz constants are not available.

Theorem 4.2. *Suppose that (A1)–(A4) hold and let $f(x^0)$ be the initial value of f. Moreover, let $\{y^t\}$ $(t = 1, 2, \ldots, T_0)$ be the sequence generated by* DS *in the $(k+1)$th iteration of* VRDFON *and let $0 < \eta < \frac{1}{2}$, $0 < \gamma_{\rm rd} < 1$, $\gamma_e > 1$ and $0 < \gamma < 1$. Then:*
(i) The number of successful iterations of DS *is bounded by*

$$\overline{\gamma}^{-1}\delta_k^{-2}\Big(f(x^0) - \widehat{f} + 2\omega\Big), \tag{4.8}$$

where $\overline{\gamma} := \gamma_e^{2(2-R_m)}\gamma > 0$, $\widehat{f}$ is finite by (A1) and (A2), and the step size δ_k is fixed, independent of t, and updated outside DS*. Moreover, the number of function evaluations of* DS *is bounded by*

$$2R_mT_0 + (2R_m + 1)T_0\overline{\gamma}^{-1}\delta_k^{-2}\Big(f(x^0) - \widehat{f} + 2\omega\Big). \tag{4.9}$$

(ii) Unsuccessful iterations of DS *have at least one point $y^{t'}$ $(1 \le t' \le T_0)$, with probability at least $\ge 1 - 2^{-R_d} \ge 1 - \eta > \frac{1}{2}$ for a given $0 < \eta < \frac{1}{2}$, satisfying*

$$\|g(y^{t'})\| \le \sqrt{cn}\,\Gamma(\delta_k), \tag{4.10}$$

where c and $\Gamma(\delta_k)$ come from Proposition 3.1 and Theorem 4.1. Here R_d comes from (3.1)*.*

Proof. (i) S denotes the index set of successful iterations of DS, where each successful iteration is a result of at least one successful iteration of MLS. Let $r' \in \{1, 2, \ldots, R_m\}$. As discussed in the proof of Theorem 4.1(i), in the worst case at least the r'th iteration of MLS is successful that a result of an extrapolation along $-p^{r'}$. We do not know how many times we can extrapolate $\alpha_{r'}$ to γ_e along the fixed direction $-p^{r'}$, but at least once $\alpha_{r'}$ is expanded by γ_e in an extrapolation and therefore at most $R_m - 1$ times $\alpha_1 = \delta_k$ is reduced by γ_e if we cannot extrapolate along the other scaled random directions and their opposite directions. Therefore, for each $t \in S$, according to the role of updating α_r in lines 1 and 12 of Algorithm 3 and line 5 of Algorithm 4,

$$\alpha_{r'} \ge \gamma_e\delta_k/\gamma_e^{R_m-1} = \gamma_e^{2-R_m}\delta_k$$

in the $(k+1)$th iteration of VRDFON. Put $\overline{\gamma} := \gamma_e^{2(2-R_m)}\gamma > 0$. We now find an upper bound on the number of successful iterations and the corresponding function evaluations of DS. For all $t \in S$ in the $(k+1)$th iteration of VRDFON, we have

$$\tilde{f}(y^{t+1}) - \tilde{f}(y^t) = \tilde{f}(z^{\hat{r}}) - \tilde{f}(z_{\text{best}}) \le -\gamma\alpha_{r'}^2 \le -\overline{\gamma}\delta_k^2,$$

recursively resulting in $\tilde{f}(y^{t+1}) \le \tilde{f}(x^0) - \overline{\gamma}\delta_k^2 \sum_{t\in S} 1 = \tilde{f}(x^0) - \overline{\gamma}\delta_k^2|S|$. From (2.2) we conclude that

$$|S| \le \overline{\gamma}^{-1}\delta_k^{-2}\Big(\tilde{f}(x^0) - \tilde{f}(y^{t+1})\Big) \le \overline{\gamma}^{-1}\delta_k^{-2}\Big(f(x^0) - \widehat{f} + 2\omega\Big).$$

Therefore, (4.8) is valid. The step size δ_k is fixed, independent of t, and updated outside DS. As mentioned earlier, MLS requires at most $2R_m + 1$ function evaluations in the worst case (using R_m scaled random directions and R_m corresponding opposite directions, all iterations of MLS are unsuccessful; however, if the r'th iteration is successful, then no reduction of the inexact function value is found when $p^{r'}$ is attempted, while reduction of the inexact function value is found when $-p^{r'}$ is attempted; a sufficient gain along the last opposite direction $-p^{r'}$ is found. Therefore, an extrapolation with at most two function evaluations is attempted. Therefore, the successful iterations of DS use at most

$$(2R_m + 1)\overline{\gamma}^{-1}\delta_k^{-2}\Big(f(x^0) - \widehat{f} + 2\omega\Big)$$

function evaluations.

U denotes the index set of unsuccessful iterations of DS. Since $T_0 = |U| + |S|$ and MLS uses $2R_m$ function evaluations for each unsuccessful iteration, we conclude that the unsuccessful iterations of DS use at most $2R_m|U| \le 2R_mT_0$ function evaluations. Consequently, the number of function evaluations of DS is bounded by (4.9).
(ii) In this case, the unsuccessful iterations of DS generate the sequence y^t $(t = 1, \dots, T_0)$, resulting in, that for at least one $y^{t'}$ $(1 \le t' \le T_0)$ of the evaluated points, with probability $\ge 1 - 2^{-R_d} = 1 - 2^{-T_0R_m} \ge 1 - \eta^{T_0} \ge 1 - \eta$, $\|g(y^{t'})\| \le \sqrt{cn}\Gamma(\delta_k)$ holds for a given $0 < \eta < \frac{1}{2}$. As mentioned in (i), the step size δ_k is fixed, independent of t; hence the bound $\sqrt{cn}\Gamma(\delta_k)$ is fixed for DS. □

4.4. Complexity for `VRDFON`

This subsection discusses a worst-case complexity for `VRDFON`. The objective function f is convex ($\sigma = 0$) if the condition

$$f(y) \geq f(x) + g(x)^T(y-x) + \frac{1}{2}\sigma\|y-x\| \quad \text{for } x, y \in \mathbb{R}^n \tag{4.11}$$

holds and is strongly convex ($\sigma > 0$) if (4.11) holds.

Assuming (A1)–(A4) and using (4.11) for the convex case ($\sigma = 0$) and the strongly convex case ($\sigma > 0$), we prove that an upper bound for the unknown gradient norm of at least one of the points generated by the unsuccessful iterations of `VRDFON` is found for all cases with a given probability arbitrarily close to one in the presence of noise.

Theorem 4.3. *Assume that (A1)–(A4) hold and* $\delta_{\max} > 0$*,* $Q > 1$*,* $0 < \gamma_{\rm rd} < 1$*,* $\gamma_e > 1$*,* $0 < \gamma < 1$*,* $\overline{\tau} > \underline{\tau} > 0$*, and*

$$\underline{\tau}\sqrt{\omega/L} \leq \delta_{\min} \leq \overline{\tau}\sqrt{\omega/L}. \tag{4.12}$$

Let $\{x^k\}$ *(*$k = 1, 2, \ldots$*) be the sequence generated by* `VRDFON`*. Then*

$$\delta_\ell = Q^{1-\ell}\delta_{\max} \quad \textit{for } \ell \geq 1 \tag{4.13}$$

and `VRDFON` *terminates after at most*

$$K := 1 + \left\lfloor \frac{\log(\delta_{\max}/\delta_{\min})}{\log Q} \right\rfloor \tag{4.14}$$

unsuccessful iterations. Then, for a given $0 < \eta < \frac{1}{2}$*,* `VRDFON` *finds at least one point* $x^{\ell'}$ *with probability at least* $1 - 2^{-R_v} \geq 1 - \eta$ *satisfying*
(i) in the nonconvex case the condition

$$\|g(x^{\ell'})\| = \mathcal{O}(\sqrt{nL\omega}); \tag{4.15}$$

(ii) in the convex case the condition (4.15) *and*

$$f(x^{\ell'}) - \widehat{f} = \mathcal{O}(r_0\sqrt{nL\omega}), \tag{4.16}$$

where r_0 *is given by*

$$r_0 := \sup\Big\{\|x - \widehat{x}\| \mid x \in \mathbb{R}^n, \quad f(x) \leq f(x^0)\Big\} < \infty; \tag{4.17}$$

(iii) in the strongly convex case the condition (4.15),

$$f(x^{\ell'}) - \widehat{f} = \frac{\mathcal{O}(nL\omega)}{2\sigma}, \quad \textit{and} \quad \|x^{\ell'} - \widehat{x}\| = \frac{\mathcal{O}(\sqrt{nL\omega})}{\sigma}. \tag{4.18}$$

Here $\widehat{f}$ is finite by (A1) and (A2) and R_v comes from (3.2).

Proof. (i) Since `VRDFON` has K unsuccessful iterations, from calls to `DS` and Theorem 4.2(ii), the condition

$$\|g(x^{\ell'})\| \le \sqrt{cn} \min_{\ell=0:K} \Gamma(\delta_\ell), \tag{4.19}$$

holds for at least one $x^{\ell'}$ of the evaluated points with probability

$$\ge 1 - 2^{-R_v} = 1 - 2^{-T_0 K R_m} \ge 1 - \eta^{T_0 K} \ge 1 - \eta > \frac{1}{2}$$

for a given $0 < \eta < \frac{1}{2}$. By (4.14), we have $\delta_K = Q^{1-K}\delta_{\max} \le \delta_{\min}$, resulting in

$$\begin{aligned}
\Gamma(\delta_K) \quad &= \quad \overline{L}\delta_K + \gamma_e^{R_m-1}\frac{4\omega}{\gamma_{\rm rd}\delta_K} = \overline{L}Q^{1-K}\delta_{\max} + \gamma_e^{R_m-1}Q^{K-1}\frac{4\omega}{\gamma_{\rm rd}\delta_{\max}} \\
&\overset{(4.14)}{=} \quad \overline{L}\frac{\delta_{\min}}{\delta_{\max}}\delta_{\max} + \gamma_e^{R_m-1}\frac{\delta_{\max}}{\delta_{\min}}\frac{4\omega}{\gamma_{\rm rd}\delta_{\max}} \\
&\overset{(4.12)}{\le} \quad (2\gamma/\gamma_{\rm rd} + L\gamma_{\rm rd})\overline{\tau}\Big(\sqrt{\omega/L}\Big) + \gamma_e^{R_m-1}\frac{4\omega}{\gamma_{\rm rd}\underline{\tau}\sqrt{\omega/L}} = \mathcal{O}\Big(\sqrt{L\omega}\Big),
\end{aligned}$$

whose application in (4.19) leads to (4.15). Here $\overline{L}$ comes from (4.5).
(ii) The convexity of f leads to

$$\widehat{f} \ge f_\ell + g(x^\ell)^T(\widehat{x} - x^\ell) \ \text{ for all } \ell \ge 0.$$

From (i), we conclude that for at least one $x^{\ell'}$ of the evaluated points with probability $\ge 1 - 2^{-R_v} > \frac{1}{2}$ the condition

$$f_{\ell'} - \widehat{f} \le g(x^{\ell'})^T(x^{\ell'} - \widehat{x}) \le \|g(x^{\ell'})\|\|x^{\ell'} - \widehat{x}\| = \mathcal{O}\Big(r_0\sqrt{nL\omega}\Big)$$

holds for a given $0 < \eta < \frac{1}{2}$.
(iii) If x is assumed to be fixed, the right-hand side of (4.11) is a convex

quadratic function with respect to y whose gradient in the components vanishes at $y_i = x_i - s_i\sigma^{-1}g_i(x)$ for $i = 1, \ldots, n$, leading to

$$f(y) \geq f(x) - \frac{1}{2\sigma}\|g(x)\|^2.$$

As mentioned earlier, $s \in \mathbb{R}^n$ is a scaling vector here. By applying (4.15) in this inequality, we obtain at least for one $x^{\ell'}$ of the evaluated points with probability $\geq 1 - 2^{-R_v} > \frac{1}{2}$

$$f_{\ell'} - \widehat{f} \leq \frac{1}{2\sigma}\|g(x^{\ell'})\|^2 = \frac{\mathcal{O}(nL\omega)}{2\sigma} \quad \text{for } \ell' \geq 0 \text{ and a given } 0 < \eta < \tfrac{1}{2}.$$

Substituting x for $\widehat{x}$ and y for $x^{\ell'}$ into (4.11), we get $f_{\ell'} \geq f(\widehat{x}) + \dfrac{\sigma}{2}\|x^{\ell'} - \widehat{x}\|^2$ such that (i) leads to the fact that for at least one $x^{\ell'}$ of the evaluated points with probability $\geq 1 - 2^{-R_v} > \frac{1}{2}$

$$\|x^{\ell'} - \widehat{x}\|^2 \leq \frac{2}{\sigma}(f_{\ell'} - \widehat{f}) \leq \frac{1}{\sigma^2}\|g(x^{\ell'})\|^2 = \frac{\mathcal{O}(nL\omega)}{\sigma^2}$$

holds for a given $0 < \eta < \frac{1}{2}$. □

Compared to the results discussed in Section 2, the order of ω in the bound (4.15) is the same as that in the conditions defined in Table 2. The conditions (4.16) and (4.18) are the same as those of Berahas et al. [31], except that they are satisfied with high probability.

For ill conditioned problems the Lipschitz constant L is too large and therefore $\delta_{\min} = \sqrt{\omega/L}$ becomes zero. Hence, Algorithm 2 can choose $\delta_{\min} = 0$.

The following result discusses a worst-case complexity for `VRDFON` for all cases. We prove that an upper bound on the number of function evaluations used by `VRDFON` is found with a given probability arbitrarily close to one in the presence of noise.

Theorem 4.4. *Let $\{x^k\}$ ($k = 1, 2, \ldots$) be the sequence generated by* `VRDFON`*. Under the assumptions of Theorem 4.3,* `VRDFON` *terminates after at most*
(i) $\mathcal{O}(R_mT_0L\omega^{-1})$ function evaluations in the nonconvex case,
(ii) $\mathcal{O}(\sqrt{n}R_mT_0L^{3/2}\omega^{-1/2})$ function evaluations in the convex case,
(iii) $\mathcal{O}(nR_mT_0L\log\omega^{-1})$ function evaluations in the strongly convex case.

Proof. Let $N_0 := 1$, $f_\ell = f(x^\ell)$ and denote by N_K the number of function evaluations for the termination of `VRDFON`. Here K comes from (4.14). In worst case, we terminate `VRDFON` after at most K unsuccessful iterations from calls to `DS`, with K points satisfying (4.10), and at least one point satisfying (4.15). Since the gradient and Lipschitz constants are unknown, these points are unknown. As a consequence of this termination, we have $\delta_\ell = Q^{1-\ell}\delta_{\max} \le \delta_{\min}$ for $\ell \ge K$ and

$$\delta_\ell \ge \delta_{\min} \quad \text{for } \ell \in \mathcal{B} := \{1, \ldots, K\}. \tag{4.20}$$

The condition (4.20) is used in the proof of (ii) and (iii).
(i) We conclude from (4.9) and (4.13)–(4.14) that

$$\begin{aligned} N_K &\le 1 + \sum_{\ell=1}^{K} \Big(2T_0 + (2R_m+1)T_0\overline{\gamma}^{-1}\delta_\ell^{-2}(f(x^0) - \widehat{f} + 2\omega)\Big) \\ &= 1 + 2R_mT_0K + (2R_m+1)T_0\overline{\gamma}^{-1}\Big(f(x^0) - \widehat{f} + 2\omega\Big)\sum_{\ell=1}^{K}\delta_\ell^{-2} \\ &= 1 + 2R_mT_0K + (2R_m+1)T_0\overline{\gamma}^{-1}\delta_{\max}^{-2}\Big(f(x^0) - \widehat{f} + 2\omega\Big)\sum_{\ell=1}^{K}Q^{2\ell-2} \\ &= 1 + 2R_mT_0K + (2R_m+1)T_0\overline{\gamma}^{-1}\delta_{\max}^{-2}\Big(f(x^0) - \widehat{f} + 2\omega\Big)\frac{Q^{2K}-1}{Q^2-1}. \end{aligned}$$

Here $\overline{\gamma}$ comes from Theorem 4.2 and for a given $0 < \eta < \frac{1}{2}$

$$R_m = \lceil \log_2 \eta^{-1} \rceil \ge 2$$

comes from Algorithm 2. In this case, using (4.12) and (4.14), $R_mT_0Q^{2K}$ dominates the other terms, resulting in

$$N_K = \mathcal{O}(R_mT_0Q^{2K}) = \mathcal{O}(R_mT_0\delta_{\min}^{-2}) = \mathcal{O}(R_mT_0L\omega^{-1}).$$

(ii) From (A1) and (A2), r_0 is finite. The convexity of f results in

$$\widehat{f} \ge f_\ell + g(x^\ell)^T(\widehat{x} - x^\ell) \quad \text{for all } \ell \ge 0.$$

By Theorem 4.3, for a given $0 < \eta < \frac{1}{2}$, we get with probability $\ge 1 - 2^{-R_v} \ge 1 - \eta > \frac{1}{2}$

$$\begin{aligned} f_\ell - f_{\ell+1} \le f_\ell - \widehat{f} &\le g(x^\ell)^T(x^\ell - \widehat{x}) \le \|g(x^\ell)\|\|x^\ell - \widehat{x}\| && (4.21) \\ &\le r_0\sqrt{cn}\Big(\overline{L}\delta_\ell + \gamma_e^{R_m-1}\frac{4\omega}{\gamma_{\text{rd}}\delta_\ell}\Big) \quad \text{for } \ell \in \mathcal{B}. && (4.22) \end{aligned}$$

Here $\overline{L}$ comes from (4.5). We consider the following two cases:
CASE 1. The first term $\overline{L}\delta_\ell$ in (4.22) dominates the second term. Then we have

$$f_\ell - f_{\ell+1} = \mathcal{O}(\overline{L}\sqrt{n}\delta_\ell) = \mathcal{O}(\sqrt{n}L\delta_\ell) \text{ for } \ell \in \mathcal{B}. \tag{4.23}$$

Then we define $\mathcal{B}_1 := \{\ell \in \mathcal{B} \mid (4.23) \text{ holds}\}$.
CASE 2. The second term $4\gamma_e^{R_m-1}\omega/(\gamma_{\text{rd}}\delta_\ell)$ in (4.22) dominates the first term. Then we conclude from (4.20) that

$$f_\ell - f_{\ell+1} = \mathcal{O}(\sqrt{n}(\omega/\delta_\ell)) = \mathcal{O}(\sqrt{n}(\omega/\delta_{\min})) = \mathcal{O}(\sqrt{n\omega L}) \text{ for } \ell \in \mathcal{B}. \tag{4.24}$$

Then we define $\mathcal{B}_2 = \{\ell \in \mathcal{B} \mid (4.24) \text{ holds}\}$.
Then we conclude from (4.12)–(4.14) that with probability $\geq 1 - 2^{-R_v} > \frac{1}{2}$

$$\begin{aligned}
\sum_{\ell\in\mathcal{B}} \frac{\tilde{f}_\ell - \tilde{f}_{\ell+1}}{\delta_\ell^2} &= \sum_{\ell\in\mathcal{B}_1} \frac{\tilde{f}_\ell - \tilde{f}_{\ell+1}}{\delta_\ell^2} + \sum_{\ell\in\mathcal{B}_2} \frac{\tilde{f}_\ell - \tilde{f}_{\ell+1}}{\gamma\delta_\ell^2} \\
&\leq \sum_{\ell\in\mathcal{B}_1} \frac{f_\ell - f_{\ell+1} + 2\omega}{\delta_\ell^2} + \sum_{\ell\in\mathcal{B}_2} \frac{f_\ell - f_{\ell+1} + 2\omega}{\delta_\ell^2} \\
&\leq \sum_{\ell\in\mathcal{B}_1} \frac{\mathcal{O}(\sqrt{n}L\delta_\ell) + 2\omega}{\delta_\ell^2} + \sum_{\ell\in\mathcal{B}_2} \frac{\mathcal{O}(\sqrt{n\omega L}) + 2\omega}{\delta_\ell^2} \\
&\leq \sum_{\ell\in\mathcal{B}} \frac{\mathcal{O}(\sqrt{n}L\delta_\ell) + 2\omega}{\delta_\ell^2} + \sum_{\ell\in\mathcal{B}} \frac{\mathcal{O}(\sqrt{n\omega L}) + 2\omega}{\delta_\ell^2} \\
&= \mathcal{O}(\sqrt{n}L) \sum_{\ell\in\mathcal{B}} \delta_\ell^{-1} + \mathcal{O}(\sqrt{n\omega L}) \sum_{\ell\in\mathcal{B}} \delta_\ell^{-2} + 4\omega \sum_{\ell\in\mathcal{B}} \delta_\ell^{-2} \\
&= \mathcal{O}(\sqrt{n}L) \sum_{\ell\in\mathcal{B}} Q^{\ell-1} + \mathcal{O}(\sqrt{n\omega L}) \sum_{\ell\in\mathcal{B}} Q^{2\ell-2} + 4\omega \sum_{\ell\in\mathcal{B}} Q^{2\ell-2} \\
&= \mathcal{O}(\sqrt{n}LQ^K) + \mathcal{O}(\sqrt{n\omega L}Q^{2K}) + \omega\mathcal{O}(Q^{2K}) \\
&= \mathcal{O}(\sqrt{n}L(\omega/L)^{-1/2}) + \mathcal{O}(\sqrt{n\omega L}(\omega/L)^{-1}) + \omega\mathcal{O}((\omega/L)^{-1}) \\
&= \mathcal{O}(\sqrt{n}L^{3/2}\omega^{-1/2}) + \mathcal{O}(\sqrt{n}L^{3/2}\omega^{-1/2}) + \mathcal{O}(L) \\
&= \mathcal{O}(\sqrt{n}L^{3/2}\omega^{-1/2}),
\end{aligned}$$

holds for a given $0 < \eta < \frac{1}{2}$. Hence, by (i) and $R_m = \lceil \log_2 \eta^{-1} \rceil \geq 2$,

$$N_K \leq 1 + (2R_m + 1)T_0 \sum_{\ell\in\mathcal{B}} \frac{\tilde{f}_\ell - \tilde{f}_{\ell+1}}{\gamma\delta_\ell^2} = \mathcal{O}\left(\sqrt{n}L^{3/2}R_mT_0\omega^{-1/2}\right)$$

holds with probability $\geq 1 - 2^{-R_v} \geq 1 - \eta > \frac{1}{2}$ for a given $0 < \eta < \frac{1}{2}$.
(iii) When x is assumed to be fixed, the right hand side of (4.11) is a convex quadratic function in terms of y whose gradient in the components vanishes at $y_i = x_i - s_i\sigma^{-1}g_i(x)$ for $i = 1, \ldots, n$, resulting in $f(y) \geq f(x) - \frac{1}{2\sigma}\|g(x)\|^2$. Here as mentioned earlier $s \in \mathbb{R}^n$ is a scaling vector. By applying (4.15) in this inequality, for a given $0 < \eta < \frac{1}{2}$, we get with probability $\geq 1 - 2^{-R_v} \geq 1 - \eta > \frac{1}{2}$

$$f_\ell - f_{\ell+1} \leq f_\ell - \widehat{f} \leq \frac{1}{2\sigma}\|g(x^\ell)\|^2 \leq \frac{cn}{2\sigma}\Big(\overline{L}\delta_\ell + \gamma_e^{R_m-1}\frac{4\omega}{\gamma_{\rm rd}\delta_\ell}\Big)^2 \quad \text{for } \ell \in \mathcal{B}.$$

Here $\overline{L}$ comes from (4.5). We consider the following two cases:
Case 1. The first term $\overline{L}\delta_\ell$ in (4.25) dominates the second term. Then we have

$$f_\ell - f_{\ell+1} = \mathcal{O}(n\overline{L}\delta_\ell^2) = \mathcal{O}(nL\delta_\ell^2) \quad \text{for } \ell \in \mathcal{B} \tag{4.25}$$

and denote $\mathcal{B}_1 := \{\ell \in \mathcal{B} \mid (4.25) \text{ holds}\}$.
Case 2. The second term $4\gamma_e^{R_m-1}\omega/(\gamma_{\rm rd}\delta_\ell)$ in (4.25) dominates the first term. Then we conclude from (4.20) that

$$f_\ell - f_{\ell+1} = \mathcal{O}\Big(n(\omega/\delta_\ell)^2\Big) = \mathcal{O}\Big(n(\omega/\delta_{\min})^2\Big) = \mathcal{O}(nL\omega) \quad \text{for } \ell \in \mathcal{B} \tag{4.26}$$

and denote $\mathcal{B}_2 = \{\ell \in \mathcal{B} \mid (4.26) \text{ holds}\}$.
Then for a given $0 < \eta < \frac{1}{2}$ we conclude from (4.12)–(4.14) that with

probability $\geq 1 - 2^{-R_v} \geq 1 - \eta > \frac{1}{2}$

$$\begin{aligned}
\sum_{\ell \in \mathcal{B}} \frac{\tilde{f}_\ell - \tilde{f}_{\ell+1}}{\delta_\ell^2} &= \sum_{\ell \in \mathcal{B}_1} \frac{\tilde{f}_\ell - \tilde{f}_{\ell+1}}{\delta_\ell^2} + \sum_{\ell \in \mathcal{B}_2} \frac{\tilde{f}_\ell - \tilde{f}_{\ell+1}}{\delta_\ell^2} \\
&\leq \sum_{\ell \in \mathcal{B}_1} \frac{f_\ell - f_{\ell+1} + 2\omega}{\delta_\ell^2} + \sum_{\ell \in \mathcal{B}_2} \frac{f_\ell - f_{\ell+1} + 2\omega}{\delta_\ell^2} \\
&\leq \sum_{\ell \in \mathcal{B}_1} \frac{\mathcal{O}(nL\delta_\ell^2) + 2\omega}{\delta_\ell^2} + \sum_{\ell \in \mathcal{B}_2} \frac{\mathcal{O}(nL\omega) + 2\omega}{\delta_\ell^2} \\
&\leq \sum_{\ell \in \mathcal{B}} \frac{\mathcal{O}(nL\delta_\ell^2) + 2\omega}{\delta_\ell^2} + \sum_{\ell \in \mathcal{B}} \frac{\mathcal{O}(nL\omega) + 2\omega}{\delta_\ell^2} \\
&= \mathcal{O}(nL)K + \mathcal{O}(nL\omega) \sum_{\ell \in \mathcal{B}} \delta_\ell^{-2} + 4\omega \sum_{\ell \in \mathcal{B}} \delta_\ell^{-2} \\
&= \mathcal{O}(nL)K + \mathcal{O}(nL\omega) \sum_{\ell \in \mathcal{B}} Q^{2\ell-2} + 4\omega \sum_{\ell \in \mathcal{B}} Q^{2\ell-2} \\
&= \mathcal{O}(nL)K + \mathcal{O}(nL\omega)\mathcal{O}(Q^{2K}) + \omega\mathcal{O}(Q^{2K}) \\
&= \mathcal{O}(nL\log(\omega/L)^{-1}) + \mathcal{O}(nL\omega(\omega/L)^{-1}) + \omega\mathcal{O}(\omega^{-1}) \\
&= \mathcal{O}(nL\log\omega^{-1}),
\end{aligned}$$

so that by (i) and since $R_m = \lceil \log_2 \eta^{-1} \rceil \geq 2$,

$$N_K \leq 1 + (2R_m + 1)T_0 \sum_{\ell \in \mathcal{B}} \frac{\tilde{f}_\ell - \tilde{f}_{\ell+1}}{\gamma\delta_\ell^2} = \mathcal{O}(nR_mT_0L\log\omega^{-1})$$

holds with probability $\geq 1 - 2^{-R_v} \geq 1 - \eta > \frac{1}{2}$ for a given $0 < \eta < \frac{1}{2}$. □

Compared to the results discussed in Section 2, the order ω of our complexity bounds is the same as that of Berahas et al. [31] which is valid in expectation.

5. Numerical performance of VRDFON

In this section, we provide numerical experiments on a number of test problems of dimensions $300 < n \leq 5000$. We used all 96 noiseless problems of large dimensions $300 < n \leq 1000$ and all 90 noiseless problems of very large dimensions $1000 < n \leq 5000$. With the different noise levels

$\omega = 10^{-5}, 10^{-4}, 10^{-3}$, this produced 3×96 large noisy problems and 3×90 very large noisy problems. The test environment of Kimiaei & Neumaier [32] was used to produce these results. The results were averaged over five runs. We compare VRDFON with the three state-of-the-art DFO solvers, VRBBO, SDBOX, and LMMAES on noisy large and very large test problems.

5.1. Choice of solvers

Table 1 lists 21 DFO solvers. One is our new solver VRDFON and 17 others are state-of-the-art solvers from literature. The 3 remaing solvers subUOBYQA, subNEWUOA, and subNMSMAX were obtained by modifying the two model-based solvers UOBYQA and NEWUOA, and the Nelder–Mead solver NMSMAX to proceed in random subspaces.

To select the DFO solvers likely to be competitive, we made some preliminary tests by comparing all 21 DFO solvers listed in Table 1 on the 192 noiseless unconstrained CUTEst test problems of small dimensions ≤ 30. With the different noise levels $\omega = 10^{-3}, 10^{-2}, 0.1, 0.9$, this produced 4×192 noisy problems.

The nine solvers (NOMAD, UOBYQA, NEWUOA, BCDFO, GRID, SNOBFIT, FMINUNC, DSPFD, MCS) were ignored for solving noisy medium scale problems since they were not best or were model-based. Indeed, since model-based solvers need a large number of sample points to construct full quadratic models, they are too slow and are not recommended to solve noisy medium to very large scale problems.

We then compared the 11 remaining solvers (VRBBO, SDBOX, BiPopMAES, LMMAES, fMAES, CMAES, BFO, NMSMAX, subUOBYQA, subNEWUOA, subNMSMAX) with VRDFON on all 171 noiseless unconstrained CUTEst test problems of medium dimensions $30 < n \leq 300$. With the different noise levels $\omega = 10^{-4}, 10^{-3}, 10^{-2}, 0.1$, this produced 4×171 noisy problems. The only solvers that remained competitive were VRDFON, VRBBO, SDBOX, and LMMAES.

VRDFON was found more efficient and robust than VRBBO, SDBOX, and LMMAES on the noisy small and medium scale problems. The other solvers were not competitive on medium scale problems. Detailed numerical results on 1452 noisy small and mediums scale problems of dimensions 2 to 300 can be found in impVRDFON.pdf from the publicly available VRDFON package [33]. This file describes details of enhancements, the solvers compared, and testing and tuning for VRDFON.

5.2. New practical enhancements

As explained in detail in [33], VRDFON uses several different directions to enrich MLS in the presence of noise. It is well known that the complexity of randomized DFO methods is better than that of deterministic methods by a factor of n in the worst case (cf. [26]); therefore, using random directions seems preferable to using deterministic ones. Even better directions than random directions are random approximate coordinate directions. Improved trust region directions are found by minimizing surrogate quadratic models in adaptively determined subspaces within a trust region. Perturbed random directions are perturbations of random directions by scaled approximate descent directions in adaptively determined subspaces. VRDFON constructs surrogate quadratic models in adaptively determined subspaces that can handle medium and large scale problems. Although these models have lower accuracy in higher dimensions, their usefulness has been confirmed in extensive numerical experiments in the presence of strong noise. VRDFON changes extrapolation step sizes heuristically when they become too small in the presence of substantial noise to prevent generating zero steps.

5.3. Starting and stopping

The starting point: As in [5], the starting point $x^0 := \xi$

$$\xi_i := (-1)^{i-1}\frac{2}{2+i}, \text{ for all } i = 1, \ldots, n$$

is chosen and the initial inexact function value $\tilde{f}_0 := \tilde{f}(x^0)$, while we compute the other inexact function values by $\tilde{f}_\ell := \tilde{f}(x^\ell + \xi)$ for all $\ell \geq 0$. The reason for this choice is that there are some toy problems in the CUTEst library with a simple solution whose solution can be easily guessed by the solver.

Stopping tests: The convergence speed of the solver s to reach a minimum of the smooth true function f is identified by measuring the quotients

$$q_s := (f_s - f_{\text{opt}})/(f_0 - f_{\text{opt}}) \quad \text{for } s \in \mathcal{S}; \tag{5.1}$$

not available in real applications, where $\mathcal{S}$ denotes the list of compared solvers, f_s denotes the best function value found by the solver *so*, f_0 denotes the function value at the starting point (common for all solvers), and f_{opt} denotes the function value at the best known point (in most cases a global minimizer or at least a better local minimizer) found by running a sequence of gradient-based and local/global gradient free solvers; see Appendix B in [5].

Type of noise: In the numerical results reported here, uniform random noise is used, which is consistent with the assumption (A3). The function values are calculated by $\tilde{f} = f + (2 * \text{rand} - 1)\omega$, where f is the true function value and $\omega \geq 0$ is a noise level whose size identifies the difficulty of the noisy problems. Here rand stands for the uniformly distributed random number.

Measure for the convergence speed: The quotients

$$q_s := (f_s - f_{\text{opt}})/(f_0 - f_{\text{opt}}) \quad \text{for } s \in \mathcal{S} \tag{5.2}$$

are measures to identify the convergence speed of the solver s to reach a minimum of the smooth true function f. These quotients are not available in real applications. Here
• f_s is the best function value found by the solver s,
• f_0 is the function value at the starting point (common for all solvers),
• f_{opt} is the function value at the best known point (in most cases a global minimizer or at least a better local minimizer) found by running a sequence of gradient-based and local/global gradient free solvers; see Appendix B in [5].

Stopping: We consider a problem **solved** by the solver s if $q_s \leq \varepsilon$ and neither the maximum number `nfmax` of function evaluations nor the maximum allowed time `secmax` in seconds was reached, and **unsolved** otherwise. The following choices were found valuable for large and very large scale noisy problems:

$$\texttt{secmax} := 420, \quad \texttt{nfmax} := 500n, \quad \varepsilon := 0.05.$$

ε, `secmax` and `nfmax` were chosen so that the best solver can solve more than half of the problems.

5.4. Profiles

Efficiency and robustness: Efficiency measures the ability of a solver $s \in \mathcal{S}$ relative to an ideal solver. The number `nf` of function evaluations is taken as a suitable cost measure, and the efficiency relative to this measure is called the `nf` efficiency. The **robustness** of a solver counts the number of problems it solves. A solver with a high number of solved problems is called **robust** and a solver with a low relative cost of function evaluations is called **efficient**.

Performance and data profile: Two important tools for figuring out which solver is **robust** and **efficient** are the data profile of Moré & Wild [34] and the performance profile of Dolan & Moré [35], respectively. $\mathcal{S}$ denotes the list of compared solvers and $\mathcal{P}$ denotes the list of problems. The fraction of problems that the solver s can solve with κ groups of $n_p + 1$ function evaluations is the data profile of the solver s, i.e.,

$$\delta_s(\kappa) := \frac{1}{|\mathcal{P}|}\Big|\Big\{p \in \mathcal{P} \;\Big|\; cr_{p,s} := \frac{c_{p,s}}{n_p + 1} \le \kappa\Big\}\Big|. \tag{5.3}$$

Here n_p is the dimension of the problem p, $c_{p,s}$ is the **cost measure** of the solver s to solve the problem p and $cr_{p,s}$ is the **cost ratio** of the solver s to solve the problem p. The fraction of problems that the performance ratio $pr_{p,s}$ is at most τ is the performance profile of the solver s, i.e.,

$$\rho_s(\tau) := \frac{1}{|\mathcal{P}|}\Big|\Big\{p \in \mathcal{P} \;\Big|\; pr_{p,s} := \frac{c_{p,s}}{\min(c_{p,\overline{s}} \mid \overline{s} \in S)} \le \tau\Big\}\Big|. \tag{5.4}$$

Note that $\rho_s(1)$ is the fraction of problems that the solver s wins compared to the other solvers, while $\rho_s(\tau)$ ($\delta_s(\kappa)$) is the fraction of problems for sufficiently large τ (κ) that the solver s can solve. The data and performance profiles are based on the problem scales, but not on the noise levels.

Noise profiles: To identify the behaviour of the compared solvers in the presence of low to high noise, we plot the number of problems solved and the efficiency of the compared solvers against the noise level, yielding two noise profiles for efficiency and robustness with respect to the noise levels.

5.5. Large scale: $300 < n \leq 1000$

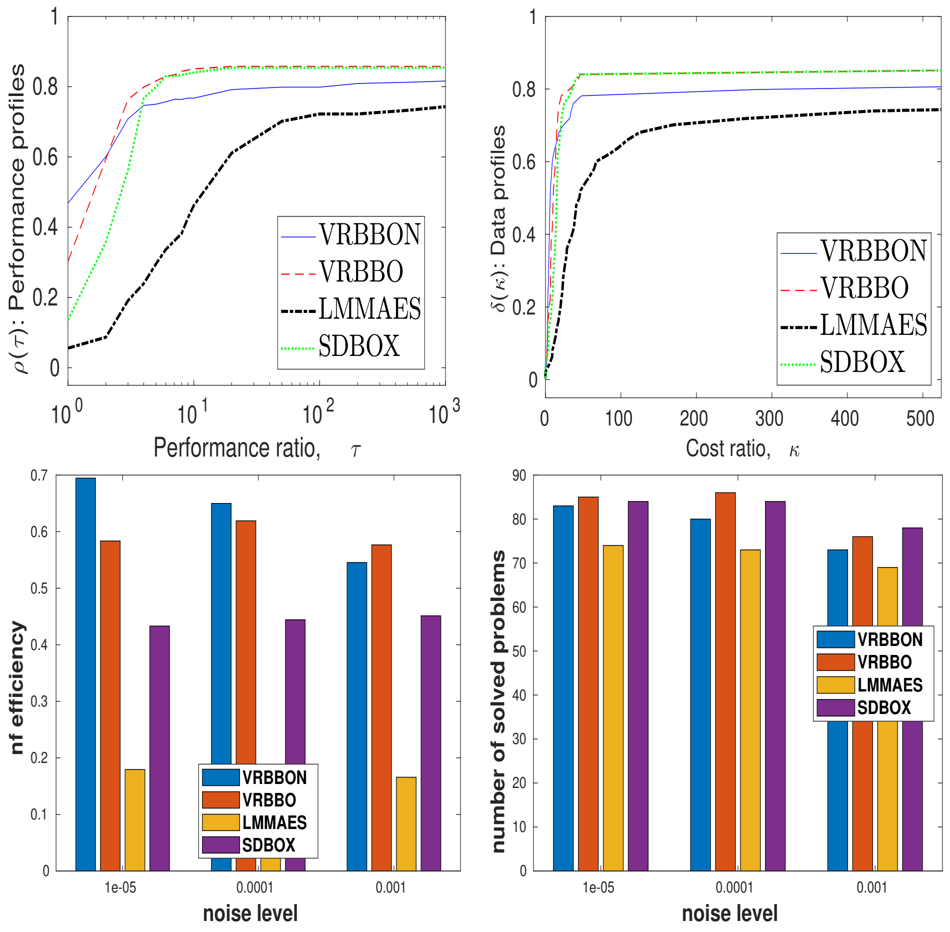


Figure 1: First row: Comparison between VRDFON and the effective solvers on the large scale problems $300 < n \leq 1000$ for the noise levels $\omega \in \{10^{-5}, 10^{-4}, 10^{-3}\}$. Data profile $\delta(\kappa)$ in dependence of a bound κ on the cost ratio while performance profile $\rho(\tau)$ in dependence of a bound τ on the performance ratio. Problems solved by no solver are ignored. Second row: Noisy profiles for more robust and efficient DFO solvers listed in Table 1 on large scale problems $300 < n \leq 1000$. Here '# solved problems' counts the number of solved problems

This subsection contains a comparison between `VRDFON` and the three more robust and efficient solvers (`VRBBO`, `LMMAES`, and `SDBOX`) on the noisy problems in medium dimensions $30 < n \leq 300$ (see `impVRDFON.pdf`).

In terms of the number of function evaluations and for the noise levels $\omega \in \{10^{-5}, 10^{-4}, 10^{-3}\}$, The first row of Figure 1 shows the cumulative (over all noise levels used) performance and data profiles, while its second row shows noise profiles with respect to the noise levels.

For all noise levels, `VRDFON` solved 236 noisy problems out of 288 noisy problems, while `VRBBO`, `SDBOX`, and `LMMAES` solved 247, 246, and 216 noisy problems out of 288 noisy problems, respectively. In terms of relative cost for `nf`, `VRDFON` is the winner with 44% noisy problems compared to the others. Thus, we conclude that `VRDFON` is more efficient than others, while `VRBBO` and `SDBOX` are more robust than `VRDFON`.

5.6. Very large scale: $1000 < n \leq 5000$

In terms of the number of function evaluations and for the noise levels $\omega \in \{10^{-5}, 10^{-4}, 10^{-3}\}$, the first row of Figure 2 shows the cumulative (over all noise levels used) performance and data profiles, while its second row shows noise profiles with respect to the noise levels.

For all noise levels, `VRDFON` solved 147 noisy problems out of 270 noisy problems, while `VRBBO`, `SDBOX`, and `LMMAES` solved 173, 169, and 48 noisy problems out of 270 noisy problems, respectively. In terms of relative cost for `nf`, `VRDFON` is the winner at 43% noisy problems compared to the others. Hence, we conclude from these subfigures that `VRDFON` is more efficient than others, while `VRBBO` and `SDBOX` are more robust than `VRDFON`.

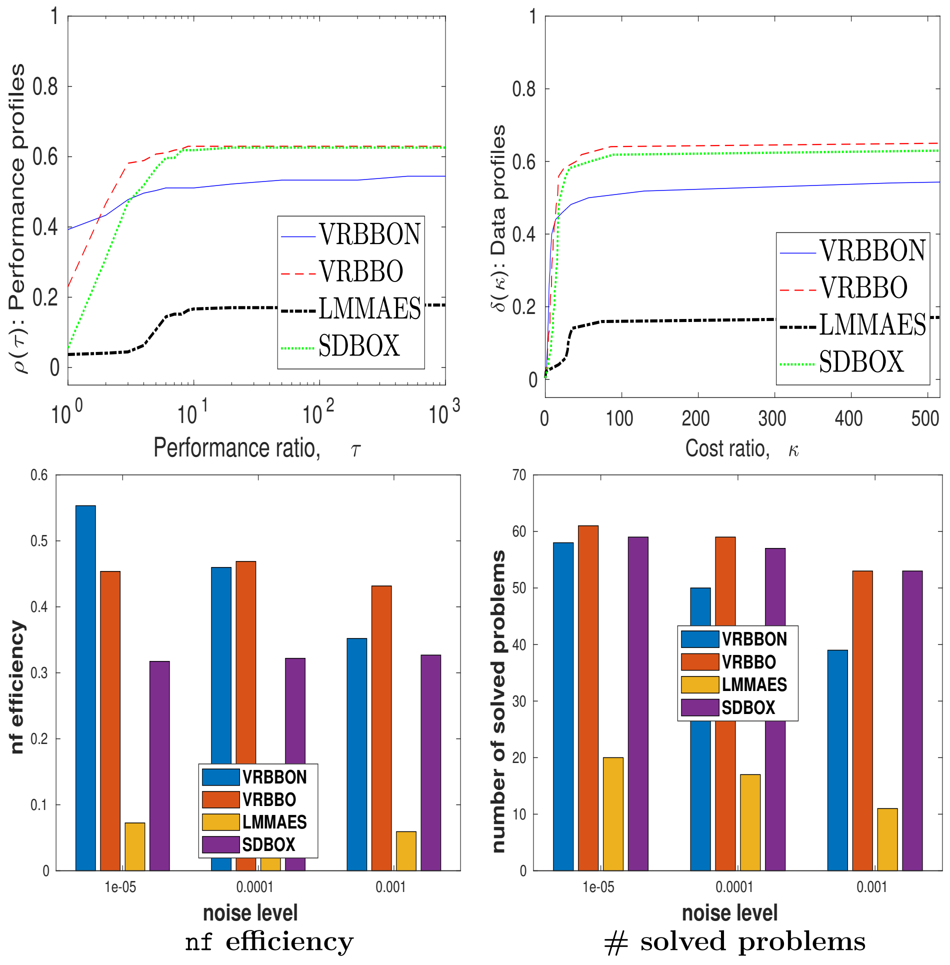


Figure 2: First row: Comparison between `VRDFON` and the effective solvers on large dimensions $1000 < n \leq 5000$ for the noise levels $\omega \in \{10^{-5}, 10^{-4}, 10^{-3}\}$. Data profile $\delta(\kappa)$ in dependence of a bound κ on the cost ratio while performance profile $\rho(\tau)$ in dependence of a bound τ on the performance ratio. Problems solved by no solver are ignored. Second row: Noisy profiles for more robust and efficient DFO solvers listed in Table 1 on the very large scale problems $1000 < n \leq 5000$. Here '# solved problems' counts the number of solved problems.

6. Real-life applications

This section gives a comparison between VRDFON and several stochastic DFO solvers to solve the real-life problem MM1 from SimOpt by Dong et al. [36], available at

https://github.com/simopt-admin/simopt/wiki,

originally from Cheng & Kleijnen [37].

SimOpt is a test environment for simulation optimization problems and solvers with the goal of promoting the development and constructive comparison of simulation optimization solvers. In practice, SimOpt tests the performance of solvers in finite time rather than the asymptotic results often found in the related literature. It provides a solvers library with ten solvers, 8 of which are listed in Table 3 (except the SPSA and GASSO solvers that were not used for our comparison because they have the lowest performance compared to others), and a problem library with 9 test problems, one MM1 of which is the only unconstrained problem. We here compare VRDFON with VRBBO and 8 solvers listed in Table 3 to solve the MM1 problem. This problem has 3 variables and describes the parameter estimation in a queueing problem.

As in [36], we choose a simulation budget for finite termination and evaluation between the compared solvers so that each compared solver can find the estimated best solution before the budget is reached. The simulation budget is in terms of the number of objective function evaluations (i.e., **replications**). We denote by $f(x_n)$ the true objective function value of the estimated best solution x_n visited in the first n objective function evaluations on a given **macroreplication** (i.e., a single execution of an algorithm on a given problem instance that reaches the budget). Since x_n is random, $f(x_n)$ is a random variable. Conditional on x_n, the objective function value $f(x_n)$ is not random, but it is unlikely that we can compute it accurately, since we evaluate the objective function by simulations.

In our experiments, we follow the testing procedure of [36]. Thus, we perform additional replications in a post-processing step to obtain fairly precise estimates of x_n conditional on x_n. These replications are not counted in the budget of the algorithm. Since the plot of the $f(x_n)$ curve for one macroreplication is of limited value and the location of the curve is random, it is more informative to run several macroreplications and average them to obtain a mean performance curve.

To solve the MM1 problem, we performed 15 macroreplications of each algorithm. For each macroreplication, we used a post-processing step to generate a sequence of estimated best solutions x_n whose objective function values $f(x_n)$ are the average of two different replications $r = 1, 30$. These post-processing replications are independent of the replications used to determine the sequence of solutions, and they use common random numbers for all algorithms. We then averaged the 15 estimates of $f(x_n)$ to produce the $f_{\mathrm{mean}}(x_n)$ curve in the following figures. In these figures, we computed 95% normal confidence intervals around $f_{\mathrm{mean}}(x)$ by plotting $f_{\mathrm{mean}}(x) + \lambda\sigma$ and $f_{\mathrm{mean}}(x) - \lambda\sigma$ of the 15 (macroreplication) samples of $f(x_n)$, where σ is the sample standard deviation and λ is chosen such that 95% of $N(0, 1)$ distributed random number are in $[-\lambda, \lambda]$.

Figure 3 shows a comparison between `VRDFON` with $r = 1$ and `VRDFON` with $r = 30$ (note that for $r = 30$ each function value average uses a budget of 30 ran evaluation; hence curves with $R = 30$ start later). From this figure, we conclude that, for a sufficiently large budget, `VRDFON` with $r = 30$ reaches a better accuracy than `VRDFON` with $r = 1$. Consequently, `VRDFON` with large replications has better performance than with small replications. This is the result of the variance reduction effect due to the 30-fold averaging in the oracle for the objective function.

With $r = 1, 30$, Figure 4 shows a comparison between `VRDFON` and `VRBBO` and all 8 solvers from Table 3, and Figure 5 shows a comparison between three best solvers. From these figures, to reach a better accuracy, we conclude that:

• With $r = 1$, `VRDFON` is the third best solver, while `STRONG` and `SASGD` are the best and second best solvers, respectively.

• With $r = 30$, `VRDFON` is the third best solver, while `VRBBO` and `STRONG` are the best and second best solvers, respectively.

solver	algorithm
`ANDFER`	direct search algorithm for noisy DFO [38]
`ASTRDF`	adaptive sampling trust-region algorithm for stochastic DFO [39]
`SASGD`	adaptive sampling stochastic Gradient Descent [36]
`SSSGD`	stopping sampling stochastic Gradient Descent [36]
`KWCDLS`	Kiefer-Wolfowitz SA with central differences and line search [36]
`NELDMD`	Nelder-Mead for simulation optimization [40]
`RANDSH`	random search [36]
`STRONG`	stochastic trust-region response-surface method [41]

Table 3: The 8 solvers from the solver library of `SimOpt`.

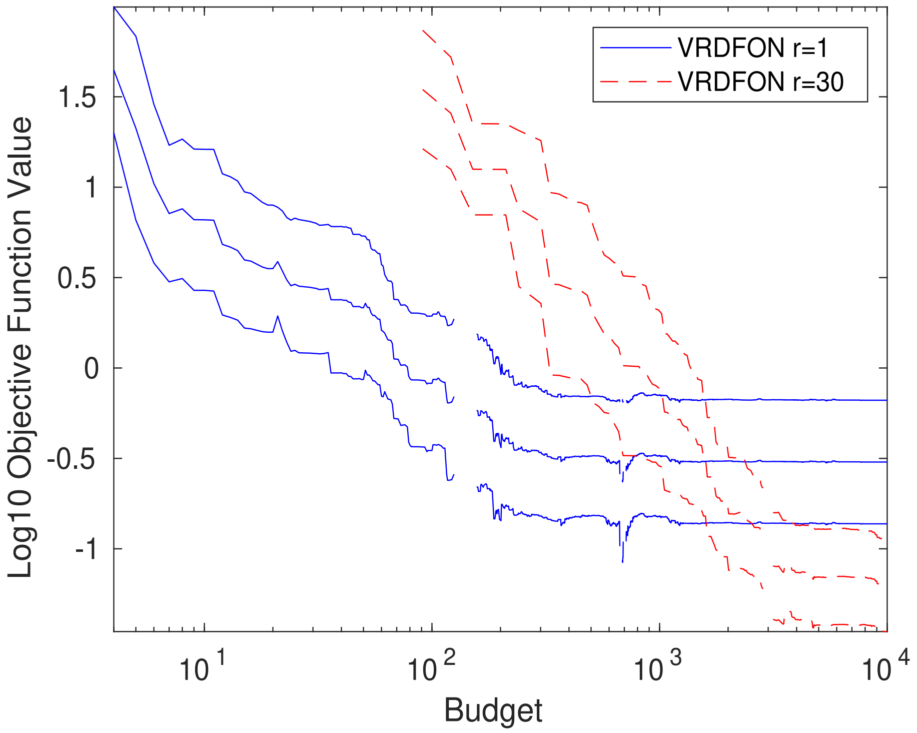
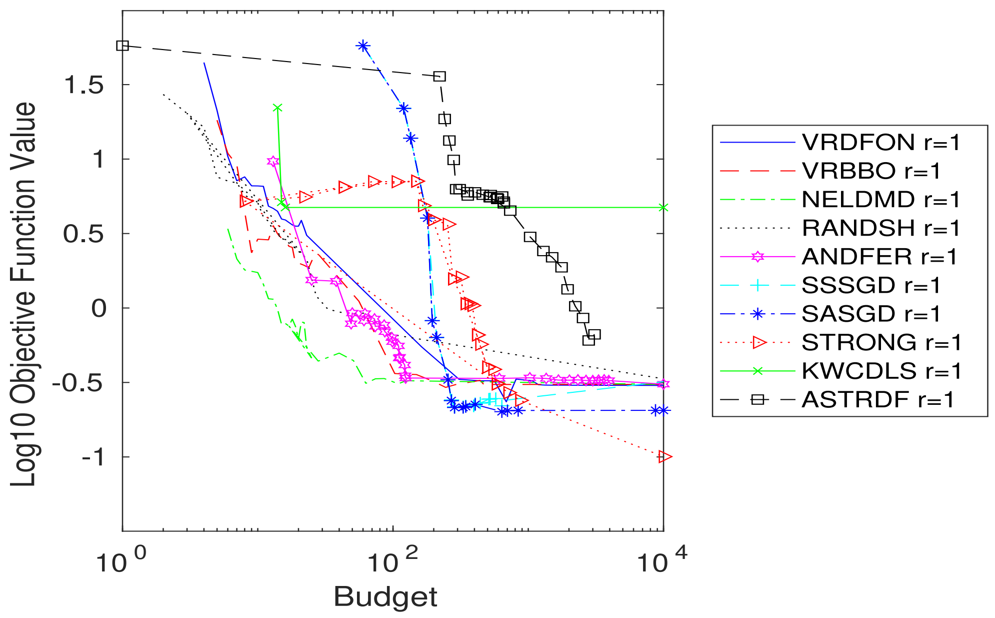


Figure 3: Average performance (mean-confidence interval) for the `MM1` problem with the dimension $n = 3$ and $\texttt{nfmax} = 10000$.



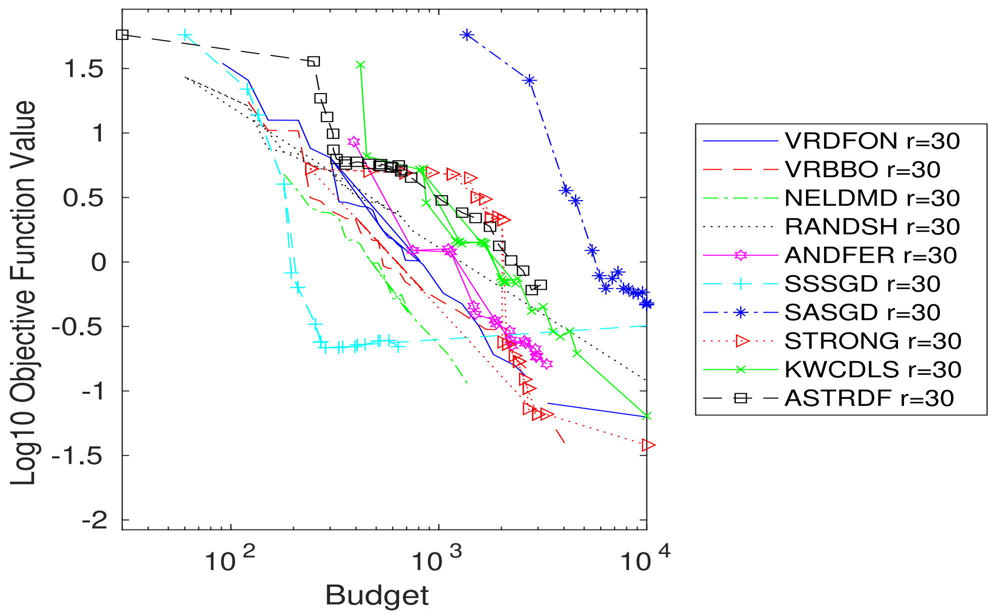
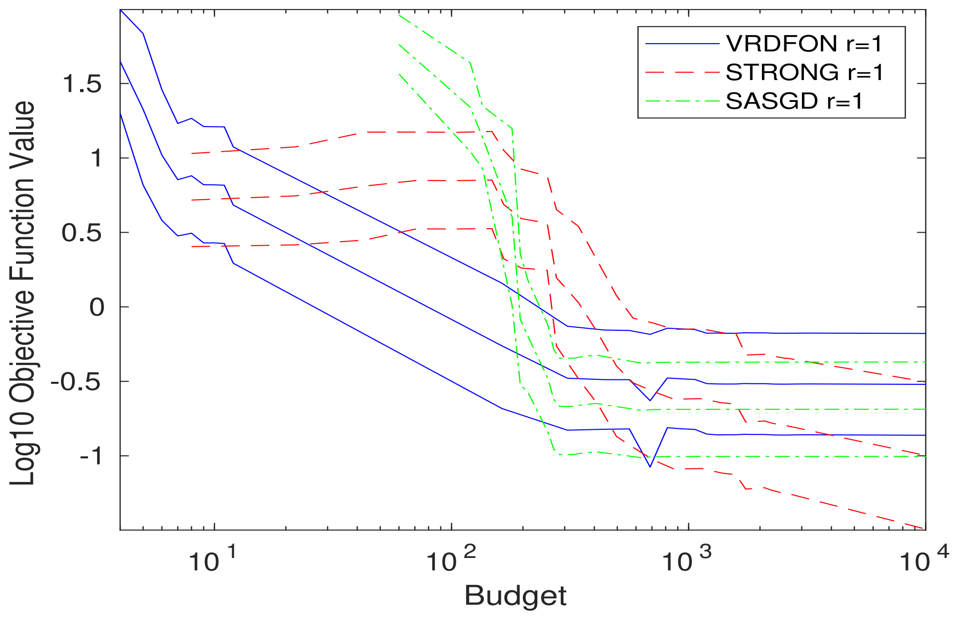


Figure 4: For the `MM1` problem with the dimension $n = 3$ and $\texttt{nfmax} = 10000$, average performances of all 10 solvers for $r = 1$ (left) and $r = 30$ (right).



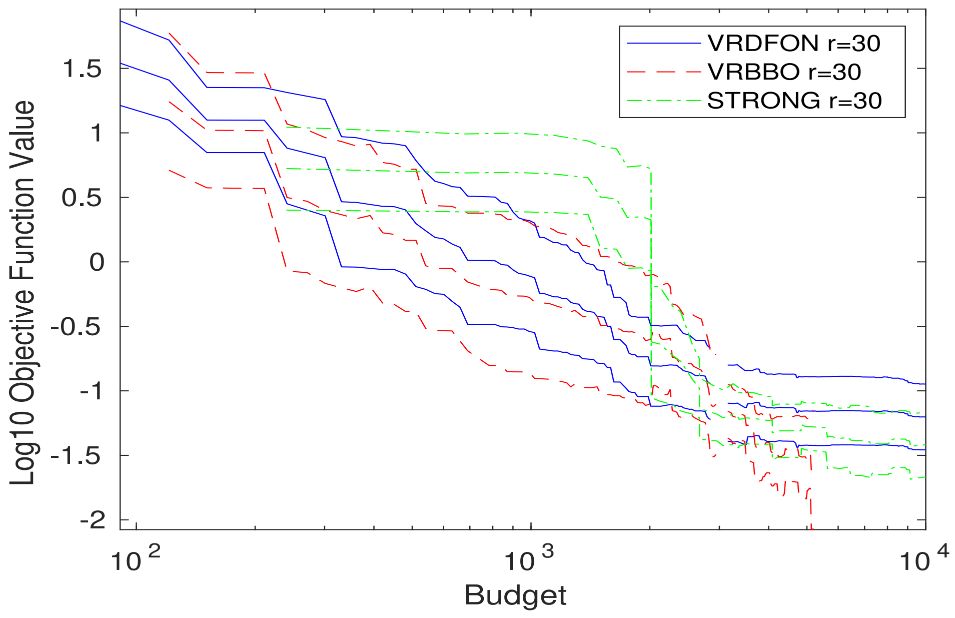


Figure 5: For the `MM1` problem with the dimension $n = 3$ and `nfmax` $= 10000$, average performances (mean confidence interval) of the 3 best solvers for $r = 1$ (left) and $r = 30$ (right).

7. Conclusion

This paper discusses `VRDFON`, a generalized randomized line search algorithm for noisy unconstrained large scale DFO problems. Complexity results for `VRDFON` in the nonconvex, convex, and strongly convex cases with a given probability arbitrarily close to one are proved.

Due to the use of quadratic models in adaptively determined subspaces and other new heuristic techniques (discussed in `impVRDFON.pdf`), `VRDFON` is much more efficient and robust than `VRBBO` for small and medium scale problems. As the quality of these quadratic models decreases with increasing dimension, `VRBBO` is more robust than `VRDFON`, but still more efficient than `VRBBO` for large problems due to the use of other heuristic techniques.

As a consequence of our results, `VRDFON` is highly recommended for solving noisy unconstrained large scale problems when the computation of the function value is expensive and efficiency is more important than robustness. It also has good performance in solving the real-life problem `MM1` [37] with large replications. This is the result of the variance reduction effect due to the 30-fold averaging in the oracle for the objective function.

Future work could be to increase the quality of quadratic models in the subspace, so that a new version of `VRDFON` can be not only the most efficient, but also the most robust for large scale DFO problems.

Acknowledgment. The author acknowledges the financial support of the Doctoral Program *Vienna Graduate School on Computational Optimization* (*VGSCO*) funded by the Austrian Science Foundation under Project No W1260-N35. Thanks also to Giampaolo Liuzzi for preparing a MEX file for calling `UOBYQA` from Matlab and to Arnold Neumaier for useful discussions.

References

[1] C. Audet, W. Hare, Derivative-Free and Blackbox Optimization, Springer International Publishing, Gewerbestrasse, Switzerland, 2017. doi:10.1007/978-3-319-68913-5.

[2] A. R. Conn, K. Scheinberg, L. N. Vicente, Introduction to Derivative-Free Optimization, Society for Industrial and Applied Mathematics, Philadelphia, USA, 2009. doi:10.1137/1.9780898718768.

[3] J. Larson, M. Menickelly, S. M. Wild, Derivative-free optimization methods, Acta Numer. 28 (2019) 287–404. doi:10.1017/s0962492919000060.

[4] L. M. Rios, N. V. Sahinidis, Derivative-free optimization: a review of algorithms and comparison of software implementations, J. Global. Optim. 56 (3) (2012) 1247–1293. doi:10.1007/s10898-012-9951-y.

[5] M. Kimiaei, A. Neumaier, Efficient unconstrained black box optimization, Math. Program. Comput. 14 (2022) 365–414. doi:10.1007/s12532-021-00215-9.

[6] N. I. M. Gould, D. Orban, P. L. Toint, CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization, Comput. Optim. Appl. 60 (2015) 545–557. doi:10.1007/s10589-014-9687-3.

[7] S. Gratton, P. L. Toint, A. Tröltzsch, An active-set trust-region method for derivative-free nonlinear bound-constrained optimization, Optim. Methods Softw. 26 (4-5) (2011) 873–894. doi:10.1080/10556788.2010.549231.

[8] M. J. D. Powell, UOBYQA: unconstrained optimization by quadratic approximation, Math. Program. 92 (3) (2002) 555–582. doi:10.1007/s101070100290.

[9] M. J. D. Powell, Developments of NEWUOA for minimization without derivatives, IMA. J. Numer. Anal. 28 (4) (2008) 649–664. doi:10.1093/imanum/drm047.

[10] W. Huyer, A. Neumaier, SNOBFIT – stable noisy optimization by branch and fit, ACM. Trans. Math. Softw. 35 (2) (2008) 1–25. doi:10.1145/1377612.1377613.

[11] C. Elster, A. Neumaier, A grid algorithm for bound constrained optimization of noisy functions, IMA J. Numer. Anal. 15 (4) (1995) 585–608. doi:10.1093/imanum/15.4.585.

[12] W. Huyer, A. Neumaier, Global optimization by multilevel coordinate search, J. Glob. Optim. 14 (4) (1999) 331–355. doi:10.1023/a:1008382309369.

[13] C. Audet, J. Dennis, Jr., Mesh adaptive direct search algorithms for constrained optimization, SIAM J. Optim. 17 (1) (2006) 188–217. doi:doi:10.1137/040603371.

[14] C. Audet, S. Le Digabel, V. Rochon Montplaisir, C. Tribes, The NOMAD project, Software available at `https://www.gerad.ca/nomad/`.

[15] C. Audet, S. Le Digabel, C. Tribes, The Mesh Adaptive Direct Search Algorithm for Granular and Discrete Variables, SIAM J. Optim. 29 (2) (2019) 1164–1189. doi:10.1137/18M1175872.

[16] S. Le Digabel, Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm, ACM. Trans. Math. Softw. 37 (4) (2011) 1–15. doi:10.1145/1916461.1916468.

[17] S. Lucidi, M. Sciandrone, A derivative-free algorithm for bound constrained optimization, Comput. Optim. Appl. 21 (2) (2002) 119–142. doi:10.1023/a:1013735414984.

[18] S. Gratton, C. W. Royer, L. N. Vicente, Z. Zhang, Direct search based on probabilistic descent, SIAM J. Optim 25 (3) (2015) 1515–1541. doi:10.1137/140961602.

[19] M. Porcelli, P. L. Toint, Exploiting problem structure in derivative free optimization, ACM. Trans. Math. Softw. 48 (1) (2022) 1–25. doi:10.1145/3474054.

[20] N. J. Higham, Optimization by direct search in matrix computations, SIAM J. Matrix Anal. Appl. 14 (2) (1993) 317–333. doi:10.1137/0614023.

[21] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, in: 2005 IEEE Congress on Evolutionary Computation, IEEE, Edinburgh, UK, 2005. doi:10.1109/cec.2005.1554902.

[22] I. Loshchilov, T. Glasmachers, H. G. Beyer, Large scale black-box optimization by limited-memory matrix adaptation, IEEE Trans. Evol. Comput. 23 (2) (2019) 353–358. doi:10.1109/tevc.2018.2855049.

[23] H. G. Beyer, Design principles for matrix adaptation evolution strategies, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, 2020, pp. 682–700. doi:10.1145/3377929.3389870.

[24] H. G. Beyer, B. Sendhoff, Simplify your covariance matrix adaptation evolution strategy, IEEE Trans. Evol. Comput. 21 (5) (2017) 746–759. doi:10.1109/tevc.2017.2680320.

[25] E. H. Bergou, E. Gorbunov, P. Richtárik, Stochastic three points method for unconstrained smooth minimization, SIAM J. Optim. 30 (4) (2020) 2726–2749. doi:10.1137/19m1244378.

[26] A. S. Bandeira, K. Scheinberg, L. N. Vicente, Convergence of trust-region methods based on probabilistic models, SIAM J. Optim 24 (3) (2014) 1238–1264. doi:10.1137/130915984.

[27] R. Chen, Stochastic derivative-free optimization of noisy functions, Ph.D. thesis, Lehigh University, theses and Dissertations. 2548 (2015). `https://preserve.lehigh.edu/etd/2548`

[28] K. J. Dzahini, Expected complexity analysis of stochastic direct-search, Comput. Optim. Appl. 81 (1) (2021) 179–200. doi:10.1007/s10589-021-00329-9.

[29] J. Blanchet, C. Cartis, M. Menickelly, K. Scheinberg, Convergence rate analysis of a stochastic trust-region method via supermartingales, INFORMS J. Comput. 1 (2) (2019) 92–119. doi:10.1287/ijoo.2019.0016.

[30] A. S. Berahas, R. H. Byrd, J. Nocedal, Derivative-free optimization of noisy functions via quasi-newton methods, SIAM J. Optim. 29 (2) (2019) 965–993. doi:10.1137/18m1177718.

[31] A. S. Berahas, L. Cao, K. Scheinberg, Global convergence rate analysis of a generic line search algorithm with noise, SIAM Journal on Optimization 31 (2) (2021) 1489–1518. doi:10.1137/19m1291832.

[32] M. Kimiaei, A. Neumaier, Testing and tuning optimization algorithm, in preparation (2023).

[33] M. Kimiaei, The VRDFON solver, software available at `https://github.com/GS1400/VRDFON` (2022).

[34] J. J. Moré, S. M. Wild, Benchmarking derivative-free optimization algorithms, SIAM J. Optim. 20 (1) (2009) 172–191. doi:10.1137/080724083.

[35] E. D. Dolan, J. J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2) (2002) 201–213. doi:10.1007/s101070100263.

[36] N. A. Dong, D. J. Eckman, X. Zhao, S. G. Henderson, M. Poloczek, Empirically comparing the finite-time performance of simulation-optimization algorithms, in: 2017 Winter Simulation Conference (WSC), IEEE, 2017. doi:10.1109/wsc.2017.8247952.

[37] R. C. H. Cheng, J. P. C. Kleijnen, Improved design of queueing simulation experiments with highly heteroscedastic responses, Operations Research 47 (5) (1999) 762–777. doi:10.1287/opre.47.5.762.

[38] E. J. Anderson, M. C. Ferris, A direct search algorithm for optimization with noisy function evaluations, SIAM J. Optim. 11 (3) (2001) 837–857. doi:10.1137/s1052623496312848.

[39] S. Shashaani, F. S. Hashemi, R. Pasupathy, ASTRO-DF: A class of adaptive sampling trust-region algorithms for derivative-free stochastic optimization, SIAM J. Optim. 28 (4) (2018) 3145–3176. doi:10.1137/15m1042425.

[40] R. R. Barton, J. S. Ivey Jr, Nelder-mead simplex modifications for simulation optimization, Manag. Sci. 42 (7) (1996) 954–973. doi:https://doi.org/10.1287/mnsc.42.7.954.

[41] K. H. Chang, L. J. Hong, H. Wan, Stochastic trust-region response-surface method (STRONG)—a new response-surface framework for simulation optimization, INFORMS J. Comput. 25 (2) (2013) 230–243. doi:10.1287/ijoc.1120.0498.