Line search in noisy unconstrained black box optimization

Morteza Kimiaei

Abstract In this paper, a new randomized solver for noisy unconstrained black box optimization, called VRBBON, is discussed. Complexity bounds in the presence of noise for nonconvex, convex, and strongly convex functions are studied. Two effective ingredients of VRBBON are an improved derivative-free line search algorithm with many heuristic enhancements and quadratic models in adaptively determined subspaces. A recommendation is made as to which solvers are robust and efficient based on dimension and noise level. It turns out that VRBBON is more robust and efficient than the state-of-the-art solvers, especially for medium and high dimensions.

Keywords Noisy black box optimization · heuristic optimization · randomized line search method · complexity bounds · sufficient decrease

2000 AMS Subject Classification: primary primary 90C56.

October 25, 2021

1. 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 and two unknown referees for useful comments.

M. Kimiaei
Fakultät für Mathematik, Universität Wien, Oskar-Morgenstern-Platz 1, A-1090 Wien, Austria
E-mail: kimiaeim83@univie.ac.at
WWW: http://www.mat.univie.ac.at/~kimiae/
1 Introduction

We consider the problem of finding a minimizer of the smooth real-valued function $f : \mathbb{R}^n \to \mathbb{R}$

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

where $f$ 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 noise. This problem is called the noisy black box optimization problem, NBBOP. We denote by $g(x)$ the unknown exact gradient vector of $f$ at $x$ and by $\tilde{g}(x)$ its approximation, and by $B(x)$ the Hessian matrix of $f$ at $x$ and by $\tilde{B}(x)$ its approximation. The algorithm uses no knowledge of $g$, the Lipschitz constants of $f$, the structure of $f$, and the statistical properties of the noise. The noise may be deterministic (caused by modelling, truncation, and/or discretization errors) or stochastic (caused by inaccurate measurements or rounding errors).

A complexity bound of an algorithm for NBBOP is an upper bound on the number of function evaluations to find an approximate point near the local optimizer whose unknown exact gradient norm is below a given fixed threshold $\omega > 0$ (which is unknown to us but appears in our complexity bound) and whose function value is as small as possible compared to the initial function value. In practice, such a point is unknown because the Lipschitz constants and gradients are unknown. However, the function value of this point is equal to or better than a point whose gradient is small only near a global optimizer. To obtain such a complexity bound, one assumes that the noise is bounded by

$$|\tilde{f}(x) - f(x)| \leq \omega.$$  

(2)

To determine which solvers are competitive for NBBOP, two main tools are used: robustness (highest number of problems solved) and efficiency (lowest relative cost of function evaluations). The data profile of Moré & Wild [32] and the performance profile of Dolan & Moré [14] are used to compare these solvers in terms of robustness and efficiency, respectively. A solver with the highest number of solved problems is called robust and with the lowest relative cost of function evaluations is called efficient.

1.1 Related work

There are many efficient and robust methods for NBBOP. Larson et al. [28] have discussed these methods and their complexity bounds (if any). These methods are based on line search, direct search, model-based, etc. and are either deterministic or randomized or both. We here focus on

- line search methods, e.g., see [28, Section 2.3.4] (Grippo & Rinaldi [20], Grippo & Sciandrone [21], Lucidi & Sciandrone [30], and Neumaier et al. [33]),
- model-based methods, e.g., see [28, Section 2.2] (Bandeira et al. [3], Buhmann [8], Conn & Toint [10], Gratton et al. [17,18], Gratton et al. [19], Powell
Huyer & Neumaier [24], and Wild et al. [40]),
• randomized methods, e.g., see [28, Section 3.2] (Audet & Dennis [1], Bandeira et al. [3], Diniz-Ehrhardt et al. [13], Gratton et al. [17,18], and Van Dyke & Asaki [39]).

Two good references for studying the efficiency and robustness of these solvers are Rios & Sahinidis [37] and Kimiaei & Neumaier [27]. As a result of their numerical results, some efficient and robust solvers are:
• Line search: VRBBO (randomized) by Kimiaei & Neumaier [27], SDBOX (deterministic) by Lucidi & Sciandrone [30], FMINUNC (Wolfe conditions along standard BFGS directions) by Matlab Optimization Toolbox.
• Nelder–Mead: NMSMAX by Higham [22].
• Direct search: DSPFD (randomized) by Gratton et al. [17], BFO by Porcelli & Toint [34], and MCS (multi coordinate search) by Huyer & Neumaier [23].
• Model-based: BCDFO by Gratton et al. [19], UOBYQA and NEWUOA by Powell [35,36], and SnobFit by Huyer & Neumaier [24].
• Covariance matrix adaptation evolution: CMAES (stochastic) by Auger & Hansen [2].

Other variant matrix adaptation evolution solvers are LMMAES (limited memory) by Loshchilov et al. [29], fMAES (fast) by Beyer [6], and BiPopMAES (Bi-Population) by Beyer & Sendhoff [7]. GRID by Elster & Neumaier [15] is another model-based solver for bound constrained noisy black box optimization problems.

We discuss the advantages and disadvantages of the above solvers. The model-based solvers are only effective for problems in low dimensions in the presence of noise, but they cannot handle problems in medium to high dimensions because $n(n+3)/2$ sample points are needed to construct fully quadratic models. By using extrapolations, the line search solvers (VRBBO and SDBOX) are more efficient and robust than the direct search solvers. In fact, extrapolations are the accelerated component of these line search methods. They expand step sizes along with fixed directions until the function values are improved. Both direct search and line search solvers can handle problems in small to large dimensions. Although another line search solver, FMINUNC, is effective in the noiseless case, it is numerically inefficient in the presence of noise because the finite difference technique for estimating the gradient leads to misleading information and poor quasi-Newton directions. The Nelder–Mead solvers are effective for small scale problems; however, they can also handle problems in medium dimensions, although their efficiency and robustness decrease with increasing dimension, e.g., Torczon [38] and Wright [41]. Loshchilov et al. [29] discussed and showed that solvers based on full covariance matrix adaptation evolution strategies are costly for large problems because the covariance matrices are stored. Therefore, they proposed a limited memory covariance matrix adaptation evolution method (LMMAES) for large scale problems. However, this method is not effective for large scale problems because it ignores some parts of the covariance matrix, see Section 7.5.3.
1.2 Known limit accuracy and complexity bounds

In this section, we discuss the achievable limit accuracy and complexity bounds of several well-known black box optimization methods under standard assumptions:

(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 \( L(x^0) := \{ x \in \mathbb{R}^n | f(x) \leq f(x^0) \} \) of \( f \) at \( x^0 \) is compact.

(A3) The uncertainty of the function value \( \tilde{f}(x) \) obtained by a noisy oracle is globally bounded by a small threshold \( \omega > 0 \), i.e., the condition (2) holds. In the noiseless case \( \omega = 0 \) (A3) implies \( \tilde{f} = f \).

(A2) implies that \( \hat{f} := \inf \{ f(x) | x \in \mathbb{R}^n \} = f(\hat{x}) > -\infty \) for any global minimizer \( \hat{x} \) of (1).

Given a positive scaling vector \( s \in \mathbb{R}^n \) (fixed in throughout the paper), we define the scaled 2-norm \( \| p \| \) of \( p \in \mathbb{R}^n \) and the dual norm \( \| g \|_* \) of \( g \in \mathbb{R}^n \) by

\[
\| p \| := \sqrt{\sum_i p_i^2 / s_i^2} \quad \text{and} \quad \| g \|_* := \sqrt{\sum_i s_i^2 g_i^2}.
\]

For the noiseless case, see Larson et al. [28, Table 8.1] for a summary of known results on worst-case complexity and corresponding references. To obtain \( \| g(x) \|_* \leq \varepsilon \) (under the assumptions (A1) and (A2)), one needs

- \( O(\varepsilon^{-3}) \) function evaluations for general case,
- \( O(\varepsilon^{-3}) \) function evaluations for convex case,
- \( O(\log \varepsilon^{-1}) \) function evaluations for 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 [3].

In the presence of noise, the limit accuracy of some algorithms was investigated:

- For the unconstrained case, Berahas et al. [4] proved convergence results for the problem (1) when \( f \) is strongly convex. Assuming strong convexity of \( f \) and boundedness of the noise in the approximation gradient, they proved that a quasi-Newton method with a fixed step size has linear convergence to a neighborhood of the solution; the gradient is estimated by the forward or central finite differences. Under the additional assumption (A3), they showed that a quasi-Newton method with step sizes found by a relaxed Armijo line search, called \textbf{FDLM}, has asymptotic accuracy

\[
f - \hat{f} = O(L\omega).
\]

\(\text{i.i.d.}\) stands for the independent and identically distributed. Chen [9] proposed a randomized algorithm with Gaussian directions and estimated step sizes, called \textbf{STRRS} for various types of noise, one of which is discussed here. Under the assumptions

(i) \( \tilde{f}(x) - f(x) = \omega(x; \zeta) \) is a stochastic noise component, where \( \zeta \) is a random vector with probability distribution \( Pr(\zeta) \),
(ii) for all \( x \in \mathbb{R}^n \), \( \omega \) is i.i.d with bounded variance \( \text{var}(\omega) > 0 \),

(iii) for all \( x \in \mathbb{R}^n \), the noise is unbiased, i.e., \( \mathbb{E}_x(\omega) = 0 \),

(iv) \( f \) is convex and (A1) holds,

(v) \( \text{var}(\omega) \leq \mathcal{O}(\varepsilon/n) \),

\( \text{STRRS} \) needs at most \( \mathcal{O}(nL\varepsilon^{-1}) \) to ensure that

\[ x_N := \arg\min_x \{ f(x) \mid x \in \{x^0, \ldots, x_N\} \} \]

satisfies \( \mathbb{E}[f(x_N)] - \hat{f} \leq \varepsilon \).

- For the bound constrained case, Elster & Neumaier [15] introduced a grid algorithm, called \( \text{GRID} \). Here we restrict their results to the unconstrained case. Under the assumptions (A1)–(A3), [15, Theorem 2] ensures that there exists a constant \( C_n \) such that

\[ \|g(x^k)\|_* \leq C_n(2\omega/h^k + Lh^k) \text{ for } x^k \in \mathbb{R}^n \]

at the end of the \( k \)th refinement step, where \( h^k := \Theta(\sqrt{\omega}) \), then the best order of magnitude can be obtained at least for a point with the gradient

\[ \|g\|_* = \mathcal{O}(C_n\sqrt{\omega}). \] (5)

The dependence of \( C_n \) on \( n \) is not specified. Under the same assumptions, Lucidi & Sciandrone [30] constructed a derivative-free line search algorithm, called \( \text{SD-BOX} \), using only the coordinate directions. They proved that, for any \( k \),

\[ \|g(x^k)\|_* = \mathcal{O}\left(n^{3/2}L a_{\text{max}}^k + \frac{n\omega}{a_{\text{min}}^k}\right), \text{ for } x^k \in \mathbb{R}^n. \]

Here \( a_{\text{min}}^k \) and \( a_{\text{max}}^k \) are minimum and maximum values, respectively, for the \( n \) step sizes used along the coordinate directions in the iteration \( k \). If \( a_{\text{max}}^k := \Theta(\sqrt{\omega}) \), the best order of magnitude can be obtained at least for a point with the gradient

\[ \|g\|_* = \mathcal{O}\left(n^{3/2}\sqrt{\omega}\right). \] (6)

The order in this bound is the same as in (5).

As stated in the introduction, \( \hat{g}(x) \) stands for the estimated gradient at \( x \). Berahas et al. [5] used the line search discussed in [4]

\[ \hat{f}(x + \alpha p) \leq \hat{f}(x) + c_1 \alpha p^T \hat{g}(x) + 2\omega \text{ with } 0 < c_1 < 1 \] (7)

but step sizes are updated in a different way. Here \( \hat{f}(x) - f(x) = \omega(x; \zeta) \) is stochastic where \( \zeta \) may be either the dependent, independent or identical. Under the assumptions (A1)–(A3) (neither stochastic noise nor reduction or control of noise is assumed) and

norm condition: \( \|\hat{g}(x) - g(x)\| \leq \theta\|g(x)\| \), for some \( 0 < \theta < 1 \), (8)

they found that the expected complexity results for a given accuracy \( \varepsilon > 0 \), which exceed a near optimal neighborhood to the noise for all cases, are sufficiently larger than \( \omega \). In the general case, \( \mathcal{O}(\varepsilon^{-2}) \) function evaluations with \( \mathbb{E}(\|g\|_*) \leq \varepsilon \) are used. In the convex and strongly convex cases, \( \mathcal{O}(\varepsilon^{-1}) \) and \( \mathcal{O}(\log\varepsilon^{-1}) \) function evaluations with \( \mathbb{E}(\|g\|_*) \leq \varepsilon \) and \( \mathbb{E}(f - \hat{f}) \leq \varepsilon \), respectively, are used.
1.3 An overview of our method

We propose a new solver for noisy unconstrained black box optimization – called Vienna noisy randomized black box optimization (VRBBON). Its basic version is called VRBBON-basic. Following the classifications of Larson et al. [28] and Rios & Sahinidis [37], our new solver VRBBON is a local model-based randomized solver.

Similarity of VRBBON with other solvers:
(i) (Algorithmic) VRBBON is an adaptation of our recent solver VRBBO (Kimez and Neumaier [27]) to the noisy case, while retaining the main structure of VRBBO, namely a multi-line search algorithm.
(ii) (Practical enhancement) VRBBON uses only one of the practical enhancements of VRBBO, namely random subspace directions.
(iii) (Complexity results) In all cases, the order of our bounds is the same as in Berahas et al. [5].

Difference between VRBBON and other solvers:
(i) (Algorithmic) VRBBON repeatedly calls a decrease search called DS that uses an improved multi-line search algorithm called MLS that is likely to reduce function values. MLS uses heuristics to find and update step sizes that are significantly different from the way step sizes are updated in other solvers, e.g., SDBOX, VRBBO, and FMINUNC. After a few calls to MLS by DS, without decreasing the function value, the step sizes may become too small if the noise is high. All derivative free line search methods have this drawback when the noise is high. To remedy this, DS reconstructs the step sizes heuristically.
(ii) (New practical enhancements) Unlike VRBBO, SDBOX, and FMINUNC, VRBBON uses many new practical enhancements. This solver 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. MLS is performed along the new directions, either random approximate coordinate, perturbed random or improved trust region directions:
• It is well known that the complexity of randomized black box optimization methods is better than that of deterministic methods by a factor of \( n \) in the worst case (cf. [3]); therefore, using random directions seems preferable to using deterministic ones.
• Even better 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.
(iii) (Complexity results for VRBBON-basic) For VRBBON-basic that uses only scaled random directions and no practical enhancements, we prove the complexity results with probability arbitrarily close to one for nonconvex, convex, and strongly convex functions in the presence of noise. In contrast to the method of Berahas et al. [5], which uses the norm condition (8), our line search does not use the term \( c_1 \alpha \gamma^2 \tilde{g}(x) \) of the condition (7), 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, e.g., see behaviour of FMINUNC in Section 8. However, we estimate the gradient to generate different heuristic directions in Section 5. Therefore, we obtain our complexity bound regardless of the norm condition (8) since the nature of the line search algorithms is different. On the other hand, our bounds are obtained with high probability. Therefore, they are more suitable than those of Berahas et al. [5], which are valid only in expectation.

(iv) (Complexity results for VRBBON) To obtain the complexity results for VRBBON (implemented version), random scaled directions should be used; this is not a problem when other directions are used. In fact, the order of the complexity bounds of VRBBON does not change for all cases after applying the heuristic improvements, only their factors become larger. However, the robustness and efficiency of VRBBON are numerically increased.

In Section 2 we discuss how to generate random directions. Then, in Section 3, we describe a basic version of VRBBON using random directions. Complexity results of a basic version of VRBBON for all cases with a given probability arbitrarily close to one in the presence of noise are proved in Section 4. An implemented version of the VRBBON solver with many new heuristic techniques is discussed in Section 5. VRBBON (implemented version) is discussed in Section 6 and its complexity results are discussed in Section 7. Section 8 provides a comparison between VRBBON and the solvers discussed in Section 1.1 on the 549 unconstrained CUTEst test problems from the collection of Gould et al. [16] and makes a recommendation as to which solvers are robust and efficient based on dimension and noise level. It turns out that VRBBON is more robust and efficient, especially for medium and high dimensions. The VRBBON package is publicly available at

https://www.mat.univie.ac.at/~kimiae/software/VRBBON.

2 Search direction

In this section it is described how to generate random directions. Then it is shown that these directions satisfy the two-sided angle condition with probability at least half.

The standard random direction is the random direction $p$ uniformly i.i.d. in $[-\frac{1}{2}, \frac{1}{2}]^n$. As explained in the introduction, i.i.d. stands for the independent and identically distributed. The scaled random direction is a standard random direction $p$ scaled by $\gamma_{rd}/\|p\|$, where $0 < \gamma_{rd} < 1$ is a tiny tuning parameter, resulting in $\|p\| = \gamma_{rd}$.

Essential for our complexity bounds is the following result (Proposition 2 in [27]) for the unknown gradient $g(x)$ of $f(x)$ at $x \in \mathbb{R}^n$.

**Proposition 1** Scaled random directions $p$ satisfy the inequality

$$\Pr(\|g(x)\|_{\star} \|p\| \leq 2\sqrt{cn}\|g(x)^T p\|) \geq 1/2$$

with a positive constant $c \leq 12.5$. The approximate value for the constant $c$ has been discussed in [27, Section 9.1].
3 VRBBON-basic, a basic version of VRBBON

This section discusses the following algorithm, a basic randomized method for NBBOP, called VRBBON-basic.

Algorithm 1 VRBBON-basic, a basic randomized method for NBBOP

```plaintext
Given the tuning parameters \( Q > 1, 0 < \gamma < 1, \gamma_e > 1, \delta_{\text{max}} > 0, 0 \leq \delta_{\text{min}} \leq 1, R \geq 2 \) (integer), \( T_0 \geq 1 \) (integer).

1: Compute \( f_0 := f(x^0) \) and set \( n_{\text{succ}} := n_{\text{func}} := 1, \delta_0 := \delta_{\text{max}}, f_{\text{best}} := x^0, \) and \( \hat{f}(z_{\text{best}}) := \hat{f}(x^0) \).
2: for \( k = 1, 2, 3, \ldots \) do
3:     initialize the number of successful iterations of DS-basic by \( n_{\text{succ}}^{DS} := 0 \);
4:     for \( t = 1, \ldots, T_0 \) do
5:         initialize the step size \( \alpha_1 := \delta_k \); \( \triangleright \) starting DS-basic
6:     end for
7:     initialize the step size \( \alpha_1 := \delta_k \); \( \triangleright \) starting MLS-basic
8:     compute the scaled random direction \( p^r \) and set \( \text{good} := 0 \);
9:     while true do \( \triangleright \) starting extrapolation along one of \( \pm p^r \); if possible:
10:        compute \( z^r := z_{\text{best}} + \alpha_p p^r \) and \( \hat{f}(z^r) \);
11:        set \( n_{\text{func}} := n_{\text{func}} + 1 \); \( \triangleright \) counting the number of function evaluations
12:        if \( n_{\text{func}} \) reaches \( n_{\text{max}} \) then, VRBBON-basic ends;
13:     end if
14:     if \( \hat{f}(z_{\text{best}}) - \hat{f}(z^r) > \gamma 0_2^2 \) then \( \triangleright \) \( \gamma 0^2 \)-sufficient gain along \( p^r \) is found
15:         set \( \text{good} := 1, f^r := \hat{f}(z^r), \) and expand the step size to \( \alpha_r := \gamma_e \alpha_r \);
16:     else if \( -p^r \) has not been tried already then
17:         set \( p^r := -p^r \); \( \triangleright \) opposite direction is tried
18:     else, break; \( \triangleright \) no \( \gamma 0^2 \)-sufficient gain along \( \pm p^r \)
19:     end if
20:     end while
21:     if \( \text{good} \) then \( \triangleright \) the \( r \)th iteration of MLS-basic is successful
22:         set \( \alpha_{r+1} := \frac{\alpha_r}{\gamma_e} \) and \( z_{\text{best}} := z_{\text{best}} + \alpha_{r+1} p^r \); \( \triangleright \) ending extrapolation
23:     update \( \hat{f}(z_{\text{best}}) := f^r \) and \( n_{\text{succ}}^{MLS} := n_{\text{succ}}^{MLS} + 1 \);
24:     else \( \triangleright \) the \( r \)th iteration of MLS-basic is unsuccessful
25:         reduce the step size to \( \alpha_{r+1} := \frac{\alpha_r}{\gamma_e} \);
26:     end if
27:     end for
28:     if \( n_{\text{succ}}^{MLS} > 0 \) then \( \triangleright \) the \( t \)th iteration of DS-basic is successful
29:         set \( n_{\text{succ}}^{DS} := n_{\text{succ}}^{DS} + 1 \), \( y^t := z_{\text{best}}, \) and \( \hat{f}(y^t) := \hat{f}(z_{\text{best}}) \);
30:     else \( \triangleright \) the \( t \)th iteration of DS-basic is unsuccessful
31:     set \( y^t := x^k \) and \( \hat{f}(y^t) := \hat{f}(x^k) \); \( \triangleright \) MLS-basic is inefficient
32:     end if
33:     end for
34:     if \( n_{\text{succ}}^{DS} > 0 \) then set \( t' := \arg\min \{ \hat{f}(y^t) \}, x^k := y^{t'}, \) and \( \hat{f}(x^{k}) := \hat{f}(y^{t'}) \);
35:     else, set \( x^{k} := x^{k-1} \) and \( \hat{f}(x^{k}) := \hat{f}(x^{k-1}) \); \( \triangleright \) DS-basic is inefficient
36:     end if
37:     if \( \delta_{k-1} \leq \delta_{\text{min}} \) then set \( z_{\text{best}} := x^k \) and \( f_{\text{best}} := \hat{f}(x^k) \); VRBBON-basic ends.
38:     end if
39:     if \( n_{\text{succ}} \) is zero then \( \triangleright \) the \( r \)th iteration of VRBBON-basic is unsuccessful
40:         set \( \delta_k := \delta_{k-1} / Q_k \); \( \triangleright \) \( \delta_k \) is reduced
41:     else
42:         set \( \delta_k := \delta_{k-1} \);
43:     end if
44:     end for
```

8
VRBBON-basic counts with $k \in \mathbb{N}_0 := \mathbb{N} \cup \{0\}$ and $t \in \{1, 2, \cdots, T_0\}$ how many times the DS-basic and MLS-basic procedures are performed, respectively. Here $1 \leq T_0 < \infty$ is a tuning parameter. Moreover, the MLS-basic procedure counts the number of scaled random directions by $r \in \{1, 2, \cdots, R\}$, whose limit is the tuning parameter $2 \leq R < \infty$.

Let $z^r$ ($r = 1, \cdots, R$) be the sequence generated by the MLS-basic procedure. Defining $r' := \arg\min_{r=1:R} \{ \hat{f}(z^r) \}$, we denote by $z_{\text{best}} := z^{r'}$ the best point found by the MLS-basic procedure and by $\hat{f}(z_{\text{best}}) := \hat{f}(z^{r'})$ its inexact function value. The extrapolation is the accelerating component of the MLS-basic procedure. It increases the convergence speed to achieve an $\varepsilon = \sqrt{\omega}$-approximate stationary point. The goal is to expand the extrapolation step sizes $\delta_r$ by the tuning parameter $\gamma \geq 1$ along the fixed direction $p^r$ and compute the corresponding trial point $z' = z_{\text{best}} + \alpha_r p^r$ and its inexact function value $\hat{f}(z')$, as long as the condition

$$\hat{f}(z_{\text{best}}) < \hat{f}(z') - \gamma \alpha_r^2$$

is satisfied, where $0 < \gamma < 1$ is a tuning parameter. The MLS-basic procedure accepts the trial point $z' = z_{\text{best}} + \alpha_r p^r$ as the new best point $z_{\text{best}} = z'$. We denote $\hat{f}(x) - \hat{f}(z')$ as the gain. If the gain along $p^r$ is at least $\gamma \alpha_r^2$, we say that $\gamma \alpha_r^2$-sufficient gain has been found along $p^r$, which means that the Boolean variable good is equivalent to true and false otherwise. The $r$th iteration of the MLS-basic procedure is called successful if good is true, and unsuccessful otherwise. Successful iterations are the results of extrapolations. The MLS-basic procedure uses $n_{\text{succ}}^{\text{MLS}}$ to count the number of successful iterations and is called inefficient when it terminates with $n_{\text{succ}}^{\text{MLS}} = 0$. After the $r$th extrapolation (if possible), since the last point generated by the extrapolation does not provide $\gamma \alpha_r^2$-sufficient gain along $p^r$, we set $\alpha_{r+1} = \alpha_r / \gamma$ to obtain the corresponding step size of the penultimate point $z_{\text{best}} := z_{\text{best}} + \alpha_r p^r$, while its inexact function value $\hat{f}(z_{\text{best}}) := \hat{f}(z')$ has already been computed. During the $r$th iteration of the MLS-basic procedure, if good is false, the corresponding step size is reduced to $\alpha_{r+1} = \alpha_r / \gamma$; otherwise, if it is true, extrapolation is attempted and the step size is extended to $\alpha_r = \gamma \alpha_r$. The question is what is the initial value for $\alpha_r$. We denote by $\delta_k$ the $k$th step size generated by VRBBON-basic, and discuss below how to update it. Let $\delta_{\text{max}} > 0$ be the initial step size, which is a tuning parameter, i.e., $\delta_0 := \delta_{\text{max}}$. The initial step size of the MLS-basic procedure is $\alpha_r = \delta_0$ in the $k$th iteration of VRBBON-basic.

We denote by $y^r$ ($r = 1, \cdots, T_0$) the sequence generated by the DS-basic procedure and by $x^k$ ($k = 1, 2, \cdots$) the sequence generated by VRBBON-basic. The $t$th iteration of the DS-basic procedure is unsuccessful if the MLS-basic procedure is inefficient and is successful otherwise. The DS-basic procedure counts by $n_{\text{succ}}^{\text{DS}}$ the number of successful iterations and is called inefficient when it terminates with $n_{\text{succ}}^{\text{DS}} = 0$. We denote by $x_{\text{best}}$ the overall best point and by $f(x_{\text{best}})$ the overall best inexact function value of VRBBON-basic, i.e., the final best point and its inexact function value found by the DS-basic procedure. Indeed, the overall best point is an $\varepsilon$-approximate stationary point of the sequence $x^k$ ($k = 1, 2, \cdots$). If $n_{\text{succ}}^{\text{DS}} = 0$ is zero, we say that the $k$th iteration of VRBBON-basic is unsuccessful and otherwise successful.
VRBBON-basic first chooses the best starting point $z_{\text{best}} = z^1 = x^0$ and its inexact function value $f(z_{\text{best}}) = f(z^1) = f(x^0)$ before the first call to the MLS-basic procedure. Next, it chooses $z_{\text{best}} = z^1 = x^{k-1}$ and $f(z_{\text{best}}) = f(z^1) = f(x^{k-1})$ for $k \geq 2$, where $x^{k-1}$ is the $(k-1)$th point generated by VRBBON-basic, and $f(x^{k-1})$ is its inexact function value. In each iteration, the DS-basic procedure hopefully improves the function value by repeatedly ($T_0$ times) calling the MLS-basic procedure with $R$ scaled random search directions and taking $y^i = z_{\text{best}}$ and $f(y^i) = f(z_{\text{best}})$ when $n_{\text{succ}}^D$ is positive. Otherwise, $y^i = y^{i-1}$ and $f(y^i) = f(y^{i-1})$. When the DS-basic procedure terminates after $T_0$ executions, VRBBON-basic selects the $k$th point $x^k := y^{t'}$ and its inexact function value $f(x^k) := f(y^{t'})$ of the sequence $x^k$ ($k = 1, 2, 3, \ldots$), where $t' := \text{argmin}\{f(y^i)\}$. Once the step size $\delta_k$ is below the minimum threshold $0 \leq \delta_{\text{min}} < 1$, which is a tuning parameter, VRBBON-basic terminates with the overall best point $x_{\text{best}} := x^k$ and its inexact function value $f(x_{\text{best}}) := f(x^k)$ in the iteration $k$. Otherwise, if $n_{\text{succ}}^D = 0$ the step size $\delta_k$ is reduced by a factor of $Q > 1$, which is a tuning parameter; otherwise, it remains fixed $\delta_k = \delta_{k-1}$. VRBBON-basic tries to find an $\varepsilon$-approximate stationary point $x_{\text{best}}$ that satisfies $\|g(x_{\text{best}})\|_* \leq \varepsilon$ for a threshold $\varepsilon = \sqrt{\omega}$ before the condition $\delta_k \leq \delta_{\text{min}}$ is satisfied. When $\delta_{\text{min}} = 0$ and the gradient $g(x_{\text{best}})$ is unknown, $n_{\text{fmax}}$ is required as an upper bound on the number of function evaluations for a finite termination.

4 Bounds for VRBBON-basic

Under the assumptions (A1)–(A3), this section discusses how VRBBON-basic is terminated after at most

- $\mathcal{O}(\omega^{-1})$ function evaluations in the general case,
- $\mathcal{O}((\sqrt{\omega})^{1/2})$ function evaluations in the convex case,
- $\mathcal{O}(n \log(\omega^{-1}))$ function evaluations in the strongly convex case

with an approximate point $x_{\text{best}}$ (overall best point), with a given probability arbitrarily close to 1, satisfying

$$f(x_{\text{best}}) \leq \sup\{f(x) \mid x \in \mathbb{R}^n\}, \quad f(x) \leq f(x^0), \quad \|g(x)\|_* = \mathcal{O}((\sqrt{\omega})).$$

As explained in the introduction, it is not clear to us which point, since the gradients and Lipschitz constants are unknown. In contrast to the method of Berahas et al. [5], which uses the norm condition, our line search does not use the term $\eta I\alpha P^T \hat{g}(x)$ of the condition (7), since the estimate of the gradient may be inaccurate in the presence of high noise, leading to the failure of the line search algorithm, but $\gamma \alpha^2$. Therefore, we are not interested in obtaining our complexity bound under the norm condition (8). We will only use the estimated gradient to generate some heuristic directions in Section 5. In all cases, the order of our bounds is the same as in [5], although the nature of the line search algorithms is different. On the other hand, our bounds are obtained with high probability. Therefore, they are more suitable than those of Berahas et al. [5], which are valid only in expectation.
The following result generalizes Proposition 1 in [27]. It is shown that if the rth iteration of the MLS-basic procedure is unsuccessful, a useful bound for the directional derivative can be found. In this case, no $\gamma \alpha_r^2$-sufficient gain ($r \in \{1, 2, \ldots, R\}$) is found along the search directions $\pm p^r$ (good is false when these directions are tried).

**Proposition 2** Let $\{z^r\}$ ($r = 1, 2, \ldots, R$) be the sequence generated by the MLS-basic procedure in the $(k + 1)$th iteration of VRBBON-basic. 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) $\hat{f}(z^r + \alpha_r p^r) < \hat{f}(z^r) - \gamma \alpha_r^2$;

(ii) $\hat{f}(z^r + \alpha_r p^r) > \hat{f}(z^r) + \gamma \alpha_r^2$ and $\hat{f}(z^r - \alpha_r p^r) < \hat{f}(z^r) - \gamma \alpha_r^2$;

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

**Proof** (A1) results in

$$\alpha_r g(z^r)^T p^r - \frac{1}{2} L \alpha_r \|p^r\|^2 \leq f(z^r + \alpha_r p^r) - f(z^r) \leq \alpha_r g(z^r)^T p^r + \frac{1}{2} L \alpha_r \|p^r\|^2. \quad (9)$$

We assume that (iii) is violated, so that

$$|g(z^r)^T p^r| > \gamma \alpha_r + 2\omega/\alpha_r + \frac{1}{2} L \alpha_r \|p^r\|^2. \quad (10)$$

We consider the proof in the two cases:

**Case 1.** If $g(z^r)^T p^r \leq 0$, then from (2) and (10) we get

$$\hat{f}(z^r + \alpha_r p^r) - \hat{f}(z^r) \leq f(z^r + \alpha_r p^r) - f(z^r) + 2\omega$$

$$\leq \alpha_r g(z^r)^T p^r + \frac{1}{2} L \alpha_r \|p^r\|^2 + 2\omega$$

$$= -\alpha_r |g(z^r)^T p^r| + \frac{1}{2} L \alpha_r \|p^r\|^2 + 2\omega < -\gamma \alpha_r^2, \quad (11)$$

meaning that good is true if $p^r$ was tried.

**Case 2.** If $g(z^r)^T p^r \geq 0$, then from (2) and (10) we get

$$\hat{f}(z^r - \alpha_r p^r) - \hat{f}(z^r) \leq f(z^r - \alpha_r p^r) - f(z^r) + 2\omega$$

$$\leq g(z^r)^T (-\alpha_r p^r) + \frac{1}{2} L \alpha_r \|p^r\|^2 + 2\omega$$

$$= -\alpha_r |g(z^r)^T p^r| + \frac{1}{2} L \alpha_r \|p^r\|^2 + 2\omega < -\gamma \alpha_r^2, \quad (12)$$

meaning that good is true if $-p^r$ was tried. We now show that good is false if $p^r$ is tried. In fact, (11) results in that (i) holds. Beside, (12) results in that the second inequality in (ii) holds, and by (2), (9), and (10) the first half

$$\hat{f}(z^r + \alpha_r p^r) - \hat{f}(z^r) \geq f(z^r + \alpha_r p^r) - f(z^r) - 2\omega$$

$$\geq \alpha_r g(z^r)^T p^r - \frac{1}{2} L \alpha_r \|p^r\|^2 - 2\omega > \gamma \alpha_r^2$$

is obtained, meaning that good is false if $p^r$ was tried. □
The following result is a generalization of Theorem 1 in [27] for the MLS-basic procedure to the noisy case in the \((k+1)\)th iteration of VRBBON-basic. As discussed earlier, the MLS-basic procedure uses \(R\) scaled random directions \((p^r)\) for \(r = 1, 2, \cdots, R\). If \(\alpha_{\text{min}} > 0\) is a minimum threshold for the step sizes generated by the MLS-basic procedure, it is proved that one of the following holds:

(i) If at least the \(R\)th iteration of the MLS-basic procedure is successful, meaning that good is true when \(-p^R\) is tried, a minimum reduction in the inexact function value is found.

(ii) An upper bound on the unknown gradient norm of at least one \(z^r\) of the points generated by the unsuccessful iterations of the MLS-basic procedure 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 1** Let \(\{z^r\} (r = 1, 2, \cdots, R)\) be the sequence generated by the MLS-basic procedure in the \((k+1)\)th iteration of VRBBON-basic. Assume that (A1)–(A3) hold and \(n_{\text{max}}\) is sufficiently large. Then one of the following happens:

(i) If at least the MLS-basic procedure has a successful iteration, then it decreases the inexact function value by at least \(\gamma_{e} \alpha_{R}^2\).

(ii) Assume that

\[
\gamma_{e} > 1, \quad 0 < \eta \leq \frac{1}{2}, \quad R := \lceil \log_{\frac{2}{\eta}} \eta^{-1} \rceil \geq 2, \quad \text{and} \quad \gamma_{e}^{1-R} \delta_{k} \geq \alpha_{\text{min}}. \quad (13)
\]

If the MLS-basic procedure has no successful iteration, then at least one \(z^{r'} (1 \leq r' \leq R)\) of the points evaluated by the unsuccessful iterations of the MLS-basic procedure, with the probability at least \(1-\eta\), has an unknown gradient \(g(z^{r'})\) satisfying

\[
\|g(z^{r'})\|_{\ast} \leq \sqrt{c} \Gamma(\delta_{k}), \quad (14)
\]

where \(c\) comes from Proposition 1 and \(\Gamma(\delta_{k}) := (2\gamma + L)\delta_{k} + 4\gamma_{e}^{R-1}\omega/\delta_{k}\). Here \(\delta_{k}\) is fixed in the MLS-basic procedure, independent of \(r\), and is updated outside this procedure.

**Proof** Let \(\mathcal{R} := \{1, \ldots, R\}\). We denote by \(p^r\) the \(r\)th scaled random search direction, by \(z^r\) the \(r\)th point, and by \(\alpha_r = \gamma_{e}^{1-r} \delta_{k} \geq \alpha_{\text{min}}\) (by (13)) the \(r\)th step size.

(i) The worst case requires \(2R + 1\) function evaluations and assumes that the \(R\)th iteration of the MLS-basic procedure 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, good is false when \(p^R\) is attempted and true when \(-p^R\) is attempted. Therefore, an extrapolation step along \(-p^R\) is performed with at most two additional function evaluations and the \(\gamma \alpha_{k}^2\)-sufficient gain. Consequently, (i) is verified.

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

\[
\Gamma_0(\alpha_r) \leq \max\{\Gamma_0(\alpha_1), \Gamma_0(\alpha_R)\} < (2\gamma + L)\alpha_1 + 4\omega/\alpha_R
\]

\[=
\Gamma(\delta_{k}) := (2\gamma + L)\delta_{k} + 4\gamma_{e}^{R-1}\omega/\delta_{k}, \quad (15)\]
where \( \alpha_1 := \max_r \{ \alpha_r \} = \delta_k > \alpha_R := \min_r \{ \alpha_r \} = \gamma e^{-R} \delta_k \) since \( \gamma e > 1 \) and \( R \geq 2 \).

Then we obtain from Proposition 2, for all \( r \in \mathcal{R} \),

\[
|g(z^r)^T p^r| \leq \gamma \alpha_r^2 + 2 \omega + \frac{L}{2} \| p^r \|^2 < \gamma \alpha_r^2 + \frac{L}{2} \alpha_r^2 + 2 \omega.
\]

so that for all \( r \in \mathcal{R} \) and from (15), the inequality

\[
\|g(z^r)\|_* = \|g(z^r)\|_* \| p^r \| / \alpha_r \leq 2 \sqrt{cn} \| g(z^r)^T p^r \| / \alpha_r \\
\leq \sqrt{cn} \left( (2 \gamma + L) \alpha_r + \frac{4 \omega}{\alpha_r} \right) = \sqrt{cn} \Gamma_0(\alpha_r) < \sqrt{cn} \Gamma(\delta_k)
\]

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

\[
\Pr \left( \|g(z^r)\|_* > \sqrt{cn} \Gamma(\delta_k) \right) < \frac{1}{2}, \quad \text{for any fixed } r \in \mathcal{R}.
\]

Therefore, we find at least one of the gradients \( g = g(z^{r'}) (r' \in \mathcal{R}) \) such that (14) holds, that is,

\[
\Pr \left( \|g\|_* \leq \sqrt{cn} \Gamma(\delta_k) \right) = 1 - \prod_{r=1}^{R-1} \Pr \left( \|g(z^r)\|_* > \sqrt{cn} \Gamma(\delta_k) \right) \geq 1 - 2^{-R} \geq 1 - \eta.
\]

\( \square \)

The following result discusses the complexity bound for the DS-basic procedure in the \((k+1)\)th iteration of VRBBON-basic. It is proved 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 the DS-basic procedure 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 2** Suppose that (A1)–(A3) hold, \( n_{f \max} \) is sufficiently large, the integer \( T_0 \geq 1 \), \( R \geq 2 \), \( 0 < \gamma < 1 \), \( \gamma e > 1 \), 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 the DS-basic procedure in the \((k + 1)\)th iteration of VRBBON-basic. Then:

(i) The number of successful iterations of the DS-basic procedure is bounded by

\[
\gamma^{-1} \delta_k^{-2} \left( f(x^0) - \hat{f} + 2 \omega \right), \quad (16)
\]

where \( \gamma := \gamma e^{(2-R)} \geq 0 \), \( \hat{f} \) is finite by (A1) and (A2) discussed in Section 1.2, and the step size \( \delta_k \) is fixed, independent of \( r \), and updated outside the DS-basic procedure. Moreover, the number of function evaluations of the DS-basic procedure is bounded by

\[
2R T_0 + (2R + 1) T_0 \gamma^{-1} \delta_k^{-2} \left( f(x^0) - \hat{f} + 2 \omega \right), \quad (17)
\]

\( \sqrt{cn} \Gamma(\delta_k) \)
(ii) If (13), then the unsuccessful iterations of the **DS-basic** procedure have at least one point $y^t$ \((1 \leq t' \leq T_0)\), with probability at least \(1 - \eta\), satisfying
\[
\|g(y^t')\|_* \leq \sqrt{c\Gamma(\delta_k)} \quad (18)
\]
where \(c\) and \(\Gamma(\delta_k)\) come from Proposition 1 and Theorem 1.

**Proof** (i) \(S\) denotes the index set of successful iterations of the **DS-basic** procedure whose each successful iteration is a result of at least one successful iteration of the **MLS-basic** procedure. As discussed in the proof of Theorem 1(i), at least the \(R\)th iteration of the **MLS-basic** procedure is successful that a result of an extrapolation along \(-p^R\). We do not know how many times we can extrapolate the step sizes \(\alpha_R\) to \(\gamma_e\) along the fixed direction \(-p^R\), but at least once \(\alpha_R\) is expanded by \(\gamma_e\) in an extrapolation and therefore at most \(R - 1\) times \(\alpha_1 = \delta_k\) is reduced by \(\gamma_e\) 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 \(\alpha_\ast\) in lines 5, 15, and 25 of Algorithm 1,
\[
\alpha_R \geq \frac{\gamma_e \delta_k}{\gamma_e \delta_k} = \gamma_e^{2-R} \delta_k
\]
in the \((k + 1)\)th iteration of **VRBBON-basic**. Put \(\bar{\gamma} := \gamma_e^{2-R} \gamma > 0\). We now find an upper bound on the number of successful iterations and their function evaluations of the **DS-basic** procedure. For all \(t \in S\) in the \((k + 1)\)th iteration of **VRBBON-basic**, we have
\[
\tilde{f}(y^{t+1}) - \tilde{f}(y^t) = \tilde{f}(z^R) - \tilde{f}(z_{\text{best}}) \leq -\gamma \alpha_R^2 \leq -\bar{\gamma} \delta_k^2,
\]
recursively resulting in \(\tilde{f}(y^{t+1}) \leq \tilde{f}(x^0) - \bar{\gamma} \delta_k^2 \sum_{t \in S} 1 = \tilde{f}(x^0) - \bar{\gamma} \delta_k^2 |S|\). From (2) we conclude that
\[
|S| \leq \bar{\gamma}^{-1} \delta_k^{-2} \left( \tilde{f}(x^0) - \tilde{f}(y^{t+1}) \right) \leq \bar{\gamma}^{-1} \delta_k^{-2} \left( f(x^0) - \tilde{f} + 2 \omega \right).
\]
Therefore, (16) is valid. The step size \(\delta_k\) is fixed, independent of \(t\), and updated outside the **DS-basic** procedure. As mentioned earlier, the **MLS-basic** procedure requires at most \(2R + 1\) function evaluations in the worst case (using \(R\) scaled random directions and \(R\) corresponding opposite directions, all iterations of **MLS-basic** procedure are unsuccessful; however, if the \(R\)th iteration is successful, then good is false when \(p^R\) is attempted and true 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 the **DS-basic** procedure use at most
\[
(2R + 1)\bar{\gamma}^{-1} \delta_k^{-2} \left( f(x^0) - \tilde{f} + 2 \omega \right)
\]
function evaluations.

\(U\) denotes the index set of unsuccessful iterations of the **DS-basic** procedure. Since \(T_0 = |U| + |S|\) and the **MLS-basic** procedure uses \(2R\) function evaluations for each
unsuccessful iteration, we conclude that the unsuccessful iterations of the DS-basic procedure use at most $2R|U| \leq 2RT_0$ function evaluations. Consequently, the number of function evaluations of the DS-basic procedure is bounded by (17).

(ii) In this case, the unsuccessful iterations of the DS-basic procedure generate the sequence $y^t$ ($t = 1, \ldots, T_0$), resulting in, that for at least one $y^{t'}$ ($1 \leq t' \leq T_0$) of the evaluated points, with probability $\geq 1 - 2^{-R} \geq 1 - \eta$, $\|g(y^{t'})\|_* \leq \sqrt{c_{\eta}n\Gamma_\eta}$. □

The objective function $f$ is convex ($\sigma = 0$) if the condition

$$f(y) \geq f(x) + g(x)^T(y-x) + \frac{1}{2}\sigma\|y-x\|$$  \hspace{1cm} \text{for } x, y \in \mathcal{L}(x^0) \tag{19}$$

holds and is strongly convex ($\sigma > 0$) if (19) holds.

It is proved that an upper bound for the unknown gradient norm of at least one of points generated by the unsuccessful iterations of VRBBON-basic is found for all cases with a given probability arbitrarily close to one in the presence of noise.

**Theorem 3** Let $\{x^k\}$ ($k = 1, 2, \ldots$) be the sequence generated by VRBBON-basic. Assume that (13) holds, $\delta_{\max} > 0$, $Q > 1$, $n_{\text{max}}$ is sufficiently large,

$$\delta_{\min} := \Theta(\sqrt{\omega}). \tag{20}$$

Then

$$\delta_\ell = Q^{1-\ell}\delta_{\max} \text{ for } \ell \geq 1 \tag{21}$$

and VRBBON-basic terminates after at most

$$K := 1 + \left\lfloor \frac{\log(\delta_{\max}/\delta_{\min})}{\log Q} \right\rfloor = \mathcal{O}(\log^{-1/2}) \tag{22}$$

unsuccessful iterations. Moreover, VRBBON-basic finds at least one point $x^{\ell'}$ with probability at least $1 - \eta$ satisfying

(i) in the nonconvex case the condition

$$\|g(x^{\ell'})\|_* = \mathcal{O}(\sqrt{n}\omega); \tag{23}$$

(ii) in the convex case the condition (23) and

$$f(x^{\ell'}) - \hat{f} = \mathcal{O}(r_0\sqrt{n}\omega), \tag{24}$$

where $r_0$ is given by

$$r_0 := \sup\left\{\|x - \hat{x}\| \mid x \in \mathbb{R}^n, \ f(x) \leq f(x^0) \right\} < \infty; \tag{25}$$

(iii) in the strongly convex case the condition (23),

$$f(x^{\ell'}) - \hat{f} = \mathcal{O}(n\omega) \frac{1}{2\sigma}, \text{ and } \|x^{\ell'} - \hat{x}\| = \mathcal{O}(\sqrt{n}\omega) \frac{1}{\sigma^2}. \tag{26}$$
Proof (i) Since VRBBON-basic has $K$ unsuccessful iterations, from calls to the DS-basic procedure, the condition

$$\|g(x^\ell)\|_* \leq \sqrt{cn}\min_{\ell=0,K} \Gamma(\delta_\ell),$$

holds for at least one $x^\ell$ of the evaluated points with probability $\geq 1 - 2^{-R} \geq 1 - \eta$ by Theorem 2(ii). By (22), we have $\delta_K = Q^{1-K}\delta_{\max} \leq \delta_{\min}$. Then (20)–(22) yield

$$\Gamma(\delta_K) = (2\gamma + L)\delta_K + 4\gamma e^{R-1}\omega/\delta_K$$
$$= (2\gamma + L)Q^{1-K}\delta_{\max} + 4\gamma e^{R-1}Q^{K-1}\omega/\delta_{\max} = O(\sqrt{n\omega}),$$

whose application in (27) leads to (23).

(ii) The convexity of $f$ leads to $\hat{f} \geq f_{\ell} + g(x^\ell)^T(x^\ell - \hat{x})$ for all $\ell \geq 0$. (i) leads to the fact that for at least one $x^\ell$ of the evaluated points with probability $\geq 1 - \eta$ the condition $f_{\ell} - \hat{f} \leq g(x^\ell)^T(x^\ell - \hat{x}) \leq \|g(x^\ell)\|_2 \|x^\ell - \hat{x}\| = O(\eta\sqrt{n\omega})$ holds.

(iii) If $x$ is assumed to be fixed, the right-hand side of (19) is a convex quadratic function with respect to $y$ whose gradient in the components vanishes at $y_i = x_i - si,\sigma^{-1}g_i(x)$ for $i = 1, \cdots, n$, 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 (23) in this inequality, we obtain at least for one $x^\ell$ of the evaluated points with probability $\geq 1 - \eta$

$$f_{\ell} - \hat{f} \leq \frac{1}{2\sigma}\|g(x^\ell)\|_2^2 = \frac{O(n\omega)}{2\sigma} \quad \text{for } \ell \geq 0.$$

Substituting $x$ for $\hat{x}$ and $y$ for $x^\ell$ into (19), we get $f_{\ell} \geq f(\hat{x}) + \frac{2}{\sigma}\|x^\ell - \hat{x}\|^2$ such that (i) leads to the fact that for at least one $x^\ell$ of the evaluated points with probability $\geq 1 - \eta$

$$\|x^\ell - \hat{x}\|^2 \leq \frac{2}{\sigma}(f_{\ell} - \hat{f}) \leq \frac{1}{\sigma^2}\|g(x^\ell)\|^2_2 = \frac{O(\sqrt{n\omega})}{\sigma^2}$$

holds. □

Compared to the results discussed in Section 1.2, the order in the bound (23) is the same as that in (5) and (6). The conditions (24) and (26) are the same as those of BERAHAS et al. [5], except that they are satisfied with high probability.

The following result discusses the complexity bound for VRBBON-basic for all cases. It is proved that an upper bound on the number of function evaluations used by VRBBON-basic is found with a given probability arbitrarily close to one in the presence of noise.

**Theorem 4** Let $\{x^k\}$ $(k = 1, 2, \cdots)$ be the sequence generated by VRBBON-basic. Under the assumptions of Theorem 3, VRBBON-basic terminates after at most

(i) $O(\omega^{-1})$ function evaluations in the nonconvex case,
(ii) $O\left(\sqrt{n\omega^{-1/2}}\right)$ function evaluations in the convex case,
(iii) $O(n\log \omega^{-1})$ function evaluations in the strongly convex case.
Proof Denote by $N_K$ the number of function evaluations for the termination of VRBBON-baisc, put $N_0 := 1$, and denote $f_\ell = f(x^\ell)$. In worst case, we terminate VRBBON-baisc after at most $K$ unsuccessful iterations from calls to the DS-basic procedure, with $K$ points satisfying (18), and at least one point satisfying (23). Since the gradient and Lipschitz constants are unknown, these points are unknown. As a consequence of this termination, we have $\delta_\ell = Q^{1-\ell}\delta_{\text{max}} \leq \delta_{\text{min}}$ for $\ell \geq K$ and

$$\delta_\ell \geq \delta_{\text{min}} \quad \text{for } \ell \in B := \{1, \ldots, K\}.$$ (28)

The condition (28) is used in the proof of (ii) and (iii).

(i) We conclude from (17) and (21)–(22) that

$$N_K \leq 1 + \sum_{\ell=1}^{K} \left( 2RT_0 + (2R + 1)T_0 \tau^{-1}\delta_\ell^{-2}(f(x^0) - \hat{f} + 2\omega) \right)$$

$$= 1 + 2RT_0K + (2R + 1)T_0 \tau^{-1}\left( f(x^0) - \hat{f} + 2\omega \right) \sum_{\ell=1}^{K} \delta_\ell^{-2}$$

$$= 1 + 2RT_0K + (2R + 1)T_0 \tau^{-1}\delta_{\text{max}}^{-2}\left( f(x^0) - \hat{f} + 2\omega \right) \sum_{\ell=1}^{K} Q^{2\ell-2}$$

$$= 1 + 2RT_0K + (2R + 1)T_0 \tau^{-1}\delta_{\text{max}}^{-2}\left( f(x^0) - \hat{f} + 2\omega \right) Q^{2K} - \frac{1}{Q^2 - 1}.$$ (30)

Here $\tau$ comes from Theorem 2. In this case, $RQ^{2K}$ dominates the other terms ($RK$, $R\omega Q^{2K}$, $Q^{2K}$, $\omega Q^{2K}$), resulting in

$$N_K = O(R\omega^{-1}) = O(\omega^{-1}).$$ (ii) From (A1) and (A2), $r_0$ is finite. The convexity of $f$ results in $\hat{f} \geq f_\ell + g(x^\ell)^T(x^\ell - x^\ell)$ for all $\ell \geq 0$. By Theorem 3, with probability $\geq 1 - \eta$, we get

$$f_\ell - f_{\ell+1} \leq f_\ell - \hat{f} \leq g(x^\ell)^T(x^\ell - \hat{x}) \leq \|g(x^\ell)\|_\infty \|x^\ell - \hat{x}\|$$

$$\leq r_0 \sqrt{m} \frac{1}{(2\gamma + L)\delta_\ell + 4\gamma R^{-1}\omega/\delta_\ell} \quad \text{for } \ell \in B.$$ (30)

We consider the following two cases:

**Case 1.** The first term $(2\gamma + L)\delta_\ell$ in (30) dominates the second term. Then we have

$$f_\ell - f_{\ell+1} = O\left(\sqrt{n}\delta_\ell \right) \quad \text{for } \ell \in B.$$ (31)

Then we define $B_1 := \{\ell \in B \mid (31) \text{ holds}\}$.

**Case 2.** The second term $4\gamma R^{-1}\omega/\delta_\ell$ in (30) dominates the first term. Then we conclude from (28) that

$$f_\ell - f_{\ell+1} = O\left(\sqrt{n}(\omega/\delta_\ell) \right) = O\left(\sqrt{n}(\omega/\delta_{\text{min}}) \right) = O\left(\sqrt{n}\omega \right) \quad \text{for } \ell \in B.$$ (32)
Then we define $B_2 = \{ \ell \in B \mid (32) \text{ holds} \}$.
Then we conclude from (21), (20), and (22) that with probability $\geq 1 - \eta$

$$\sum_{\ell \in \mathcal{B}} \frac{\hat{f}_\ell - \hat{f}_{\ell+1}}{\delta_\ell^2} = \sum_{\ell \in \mathcal{B}_1} \frac{\hat{f}_\ell - \hat{f}_{\ell+1}}{\delta_\ell^2} + \sum_{\ell \in \mathcal{B}_2} \frac{\hat{f}_\ell - \hat{f}_{\ell+1}}{\gamma \delta_\ell^2}$$

$$\leq \sum_{\ell \in \mathcal{B}_1} \frac{f_\ell - f_{\ell+1} + 2\omega}{\delta_\ell^2} + \sum_{\ell \in \mathcal{B}_2} \frac{f_\ell - f_{\ell+1} + 2\omega}{\delta_\ell^2}$$

$$\leq \sum_{\ell \in \mathcal{B}_1} \frac{O(\sqrt{n} \delta_\ell)}{\delta_\ell^2} + \sum_{\ell \in \mathcal{B}_2} \frac{O(\sqrt{n} \omega) + 2\omega}{\delta_\ell^2}$$

$$\leq \sum_{\ell \in \mathcal{B}} \frac{O(\sqrt{n} \delta_\ell) + 2\omega}{\delta_\ell^2} + \sum_{\ell \in \mathcal{B}} \frac{O(\sqrt{n} \omega) + 2\omega}{\delta_\ell^2}$$

$$= O(\sqrt{n}) \sum_{\ell \in \mathcal{B}} \delta_\ell^{-1} + O(n\omega) \sum_{\ell \in \mathcal{B}} \delta_\ell^{-2} + 4\omega \sum_{\ell \in \mathcal{B}} \delta_\ell^{-2}$$

$$= O(\sqrt{n} Q^K) + O(\sqrt{n} \omega Q^{2K}) + \omega O(n^{1/2})$$

$$= O(\sqrt{n} \omega^{-1/2}) + O(\sqrt{n} \omega^{-1}) + O(n)$$

so that by (i) and (13)

$$N_K \leq 1 + (2R + 1) R_0 \sum_{\ell \in \mathcal{B}} \frac{f_\ell - f_{\ell+1}}{\delta_\ell^2} = O\left(\sqrt{n} R \omega^{-1/2}\right) = O\left(\sqrt{n} \omega^{-1/2}\right)$$

holds with probability $\geq 1 - \eta$.

(iii) When $x$ is assumed to be fixed, the right hand side of (19) 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 $f(y) \geq f(x) - \frac{1}{2} \| g(x) \|^2$. Here as mentioned earlier $s \in \mathbb{R}^n$ is a scaling vector. By applying (23) in this inequality, we get with probability $\geq 1 - \eta$

$$f_\ell - f_{\ell+1} \leq \frac{1}{2\sigma} \| g(x) \|^2 \leq \frac{cn}{2\sigma} \left( (2\gamma + L) \delta_\ell + 4\gamma R^{-1} \omega / \delta_\ell \right)^2$$

for $\ell \in \mathcal{B}$. (33)

We consider the following two cases:

CASE 1. The first term $(2\gamma + L) \delta_\ell$ in (33) dominates the second term. Then we have

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

(34)

and denote $\mathcal{B}_1 := \{ \ell \in \mathcal{B} \mid (34) \text{ holds} \}$.

CASE 2. The second term $4\gamma R^{-1} \omega / \delta_\ell$ in (33) dominates the first term. Then we conclude from (28) that

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

(35)
and denote $B_2 = \{ \ell \in B \mid (35) \text{ holds} \}$.

Then we conclude from (20)–(22) that with probability $\geq 1 - \eta$

$$\sum_{\ell \in B} \frac{\hat{f}_\ell - \hat{f}_{\ell+1}}{\delta_\ell^2} = \sum_{\ell \in B_1} \frac{\hat{f}_\ell - \hat{f}_{\ell+1}}{\delta_\ell^2} + \sum_{\ell \in B_2} \frac{\hat{f}_\ell - \hat{f}_{\ell+1}}{\delta_\ell^2} \leq \sum_{\ell \in B_1} \frac{f_\ell - f_{\ell+1} + 2\omega}{\delta_\ell^2} + \sum_{\ell \in B_2} \frac{f_\ell - f_{\ell+1} + 2\omega}{\delta_\ell^2} \leq \sum_{\ell \in B_1} \frac{\mathcal{O}(n\delta_\ell^2) + 2\omega}{\delta_\ell^2} + \sum_{\ell \in B_2} \frac{\mathcal{O}(n\omega) + 2\omega}{\delta_\ell^2} \leq \sum_{\ell \in B} \frac{\mathcal{O}(n\delta_\ell^2) + 2\omega}{\delta_\ell^2} + \sum_{\ell \in B_2} \frac{\mathcal{O}(n\omega) + 2\omega}{\delta_\ell^2} = \mathcal{O}(n) + \mathcal{O}(n\omega) \sum_{\ell \in B} \delta_\ell^{-2} + 4\omega \sum_{\ell \in B} \delta_\ell^{-2} = \mathcal{O}(n)K + \mathcal{O}(n\omega) \sum_{\ell \in B} Q^{2\ell-2} + 4\omega \sum_{\ell \in B} Q^{2\ell-2} = \mathcal{O}(n)K + \mathcal{O}(n\omega) \mathcal{O}(Q^{2K}) + \omega \mathcal{O}(Q^{2K}) = \mathcal{O}(n \log \omega^{-1}) + \mathcal{O}(n\omega \omega^{-1}) + \omega \mathcal{O}(\omega^{-1}) = \mathcal{O}(n \log \omega^{-1}) + \mathcal{O}(n) + \mathcal{O}(1) = \mathcal{O}(n \log \omega^{-1})$$

so that by (i) and (13)

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

holds with probability $\geq 1 - \eta$.  $\square$

Compared to the results discussed in Section 1.2, the order of our complexity bounds is the same as that of Berahas et al. [5] which is valid in expectation.

5 Heuristic enhancements

In this section we propose many practical improvements that make VRBBON a very competitive solver. Two of the most important of these are surrogate quadratic models in adaptively determined subspaces and finding and updating step sizes in an improved version of the MLS-basic procedure, called MLS. Improved versions of the DS-basic procedure, called DS, use the MLS procedure with different search directions. The DS procedure heuristically reconstructs the step sizes in each unsuccessful iteration, resulting in small extrapolation step sizes. VRBBON calls the DS procedure repeatedly until an $\varepsilon = \sqrt{\omega}$-approximate stationary point is not found.
5.1 Random approximate coordinate directions

As mentioned in the introduction, the use of random directions is preferable to the use of deterministic directions (see [3]). On the other hand, it is known that coordinate directions are useful to estimate the gradient, since at least one of these directions has a good angle with the gradient (see [11]). Inspired by these points, we construct an effective version of scaled random directions, which we call **random approximate coordinate direction**, that combines the advantages of both scaled random and coordinate directions. The random coordinate direction $p$ multiplies a standard random direction by a scaling vector. Only its one component is $1/\|p\|$ and its other components are randomly generated and are $\gamma_{rd}/\|p\|$ with $0 < \gamma_{rd} < 1$, where is a small tuning parameter.

5.2 Subspace information

The points $Z_i$ with the best inexact function values are stored as columns of a matrix $Z$, their inexact function values $\tilde{f}(Z_i)$ in the vector $F$, and their step sizes $\alpha_i$ in the vector $S$. We adjust the matrix $Z$ if its entries are contaminated by NaN or $\pm\infty$ by inserting a large positive tuning parameter $\gamma_Z > 0$.

We denote the points stored in $Z$ as **sample points**, whose maximum number is defined by $m_{\text{max}} := \min \left\{ m, \frac{1}{2}n(n + 3) \right\}$, where $m$ is a tuning parameter. The **number of sample points** is denoted by $m \in [2, m_{\text{max}}]$ and the **subspace size** $m^o$ is defined as the largest integer satisfying $\frac{1}{2}m^o(m^o + 3) \leq m$.

If $m$ is greater than $m_{\text{max}}$, we update $Z$, $F$, $S$ by replacing the worst point, its inexact function value, and its step size with the current best point, its inexact function value, and its step size, respectively. Otherwise, we append the current best point to $Z$, its inexact function value to $F$, and its step size to $S$.

5.3 Random subspace directions

In [27] it was shown by extensive numerical results that after using a derivative-free line search algorithm with coordinate directions, the use of random subspace directions by such an algorithm is very useful. Inspired by this, after using random coordinate directions by **VRBBON**, random subspace directions are used by such an algorithm in the hope of reducing the influence of noise.

We write $A_{Ij} := (A_{ij})_{i \in I, j \in I}$ for the submatrix of $A$ with row and column indices of $I$ by $A_{Ij}$, $A_k$ for the $k$th column of a matrix $A$, and $b$ for the index of the best point.

Random subspace directions are constructed based on the information of sample points with good function values. As in [27], we generate a $(m - 1) \times 1$ standard
random vector $\alpha^s$ and then scale it with $\alpha^s := \alpha^s / \|\alpha^s\|$. Then we compute the **random subspace direction** by $p := \sum_{i=1, i \neq b}^{m} \alpha^s_i (Z_i - Z_b)$.

5.4 Reduced quadratic models

It is well-known [24] that model-based algorithms need at least

$$N := n + \frac{1}{2} n(n + 1) = \frac{1}{2} n(n + 3)$$

sample points and $O(N^3)$ operations for estimating the gradient vector and Hessian matrix of an objective quadratic model. For medium or large scale problems, this is prohibitively expensive. To overcome this problem, we construct quadratic models in adaptively determined subspaces called **reduced quadratic models**, one of which is a fully quadratic model when $m = N$.

For all $i = 1, \cdots, m$, let $z_i$ and $\tilde{f}_i := \tilde{f}(z_i)$ be the sample points and their inexact function values stored in $Z$ and $F$, respectively, and define $\pi_i := z_i - Z_b$. Before defining the model errors, we need to know whether the number of sample points $m$ is allowable to construct a full or reduced quadratic model or not. Therefore, we compute all the subspace sizes that are admissible to construct quadratic models.

The subspace size defined in Subsection 5.2 is calculated as follows

$$m^o := \left\lfloor \frac{1}{2} \left( -3 + \sqrt{9 + 8m} \right) \right\rfloor.$$  \hspace{1cm} (36)

It is clear that for $m < N$ no fully quadratic model can be constructed; reduced quadratic models are constructed instead. When the dimension is larger than $m_{\max}$, the $n \times m_{\max}$ matrix $Z$ and the $n \times 1$ vector $z^m$ are reduced to the $m_{\max} \times m^o$ matrix $Z^o$ and the $m^o \times 1$ vector $z^o$, respectively. In other words, the restriction of the entries of $Z$ and $z^m$ is done by choosing a random subset of size $m^o$.

Some components of each best point stored in $Z$ may be ignored. To overcome this shortcoming, reduced quadratic models are constructed several times before $m$ exceeds $m_{\max}$, each time choosing $Z^o$ ($z^o$) from a random subset of the entries of $Z$ ($z_{\best}$).

We write $\tilde{g}^o$ and $\tilde{B}^o$ for the estimated gradient vector and the symmetric Hessian matrix in a subspace, respectively. Let $M := \frac{1}{2} m^o (m^o + 3)$ and $K := \min(2M, m - 1)$.

To evaluate the inexact $\tilde{g}^o$ and $\tilde{B}^o$, we define the model errors by

$$\varepsilon_i := \frac{\tilde{f}_i - F_b - (\tilde{g}^o)^T \pi_i - \frac{1}{2} \pi_i^T \tilde{B}^o \pi_i}{\|\pi_i\|} \quad \text{for all } i = 1, \cdots, K.$$  \hspace{1cm} (37)
where \( \mathbf{a} \) is an appropriate scaling vector. It is shown in Deuflhard & Heindl [12] that numerical methods can generally perform much better when they preserve affine invariance. The following choice ensures the affine invariance of the fitting method.

The numerator of (37) is \( O(\|\mathbf{s}_i\|^3) \) if \( m = N \); otherwise it is \( O(\|\mathbf{s}_i\|^2) \). To obtain \( \varepsilon_i \) of a uniform magnitude, we choose \( \mathbf{a} \) as follows:

1. We form the matrix \( \mathbf{S} := (\mathbf{s}_1, \ldots, \mathbf{s}_K)^T \), where \( \mathbf{s}_i := \mathbf{z}_o - \mathbf{Z}_o^b \) for all \( i = 1, \ldots, K \).
2. We compute the matrix
   \[
   H^o := \left( \sum_i \mathbf{s}_i \mathbf{s}_i^T \right)^{-1}
   \] (38)
   by constructing a reduced QR factorization \( \mathbf{S} = \mathbf{Q} \mathbf{R} \), where \( \mathbf{Q} \in \mathbb{R}^{K \times m_o} \) is an orthogonal matrix and \( \mathbf{R} \in \mathbb{R}^{m_o \times m_o} \) is a square upper triangular matrix.
3. We compute the scaling vector \( \mathbf{a} \)
   \[
   \mathbf{a}_i := (\mathbf{s}_i^T H^o \mathbf{s}_i)^{\epsilon/2}
   \] for \( i = 1, \ldots, K \), (39)
   with \( \mathbf{a}_i := \|\mathbf{R}^{-T} \mathbf{s}_i\|^2 \) for \( i = 1, \ldots, K \). In (39), \( \epsilon = 3 \) if \( m = N \) holds (full quadratic model) and \( \epsilon = 2 \) otherwise (reduced quadratic model).

We calculate \( H^o \) and \( \mathbf{a} \) in the same way as SnoBfit [24] by performing a reduced QR factorization, except that they are performed in adaptively determined subspaces.

We have \( \varepsilon = A \mathbf{y} - \mathbf{a} \), where, for \( i = 1, \ldots, K \),

\[
\mathbf{a}_i := \frac{F_b - \tilde{f}_i}{\mathbf{a}_i} \quad \text{and} \quad A_{ij} := \begin{cases} 
\frac{\mathbf{s}_j}{\mathbf{a}_i} & \text{if } j \in \{1, \ldots, m^o\}, \\
\frac{(\mathbf{s}_j^{m^o - m^o})^2}{2\mathbf{a}_i} & \text{if } j \in \{m^o + 1, \ldots, 2m^o\}, \\
\frac{\mathbf{s}_j^{'m^o}}{\mathbf{a}_i} & \text{if } j \in \{2m^o + 1, \ldots, M\}.
\end{cases}
\] (40)

Here \( j' \) and \( j'' \) are the remainders of the division of \( j - 2m^o \) and \( j - 2m^o + 1 \) by \( m^o \), respectively, and \( \mathbf{s}_j' \) is the \( j \)th component of the vector \( \mathbf{s}_j \). To find the entries of the inexact \( \tilde{\mathbf{g}}^o \) and \( \tilde{\mathbf{B}}^o \) we solve the linear least squares problem

\[
\min_{\mathbf{y} \in \mathbb{R}^M} \|A \mathbf{y} - \mathbf{a}\|_2^2.
\] (41)

In finite precision arithmetic, each of the vectors \( \mathbf{a}, \mathbf{a}_i, \mathbf{y}, \) and hence \( \tilde{\mathbf{g}}^o, \tilde{\mathbf{B}}^o \) can have entries with value \( \text{NaN} \) or \( \pm \infty \). We replace the components of the vectors \( \mathbf{a}, \mathbf{a}_i \) and \( \mathbf{y} \) with value \( \text{NaN} \) or \( \pm \infty \) by a large positive tuning parameter \( \gamma_v > 0 \).

We construct the vector \( \mathbf{a} \) and the matrix \( \mathbf{A} \) by (40), adjust \( \mathbf{a} \), and find all multipliers by solving (41). Then we adjust \( \mathbf{y} \) and define \( \tilde{\mathbf{g}}^o \) by the first \( m^o \) components of \( \mathbf{y} \), the
diagonal entries of $\tilde{B}^o$ by the next $m^o$ components of $y$, and the off-diagonal entries of $\tilde{B}^o$ symmetrically by the remaining entries of $y$.

In summary, we construct reduced quadratic models. Two advantages of such models are the use of limited sample points and the ability to construct them multiple times by increasing the size of the subspace from 2 to $m_{\text{max}}$. To obtain a robust model, we adjust the matrix $Z$ and the vector $y$ whenever they are contaminated by NaN or $\pm \infty$. Second, we compute $\mathbf{c}$ and adjust it. Third, we calculate $y$ and adjust it. Finally, we estimate $\tilde{g}^o$ and $\tilde{B}^o$.

5.5 Perturbed random directions

The use of other good directions is required when the MLS procedure with scaled random directions and random subspace directions are inefficient in some iterations when noise is present. In this case, perturbed random directions are used if $\tilde{B}^o$ is not computable, which implies that at least one entry of $\tilde{B}^o$ is contaminated by NaN or $\pm \infty$. Otherwise, reduced quadratic models are constructed in Subsection 5.4 and improved trust region directions are computed, as explained later in Subsection 5.6.

A perturbed random direction $\tilde{p} := p^o - \alpha^o \tilde{g}^o$ is a perturbation of a standard random direction $p^o$ by the steepest descent direction $-\tilde{g}^o$ with the approximate gradient. Both $\tilde{g}^o$ and $p^o$ are restricted to a subspace which is a random subset $\mathcal{J}$ of $\{1, \ldots, n\}$ whose size is $m^o$. For all $i \notin \mathcal{J}$, $\tilde{p}_i = 0$. To be numerically appropriate, both are scaled by the heuristic step size $\alpha^o := (1 + \kappa (\tilde{g}^o)^T p^o) / \|\tilde{g}^o\|^2$ and the decreasing sequence $\kappa := 1/(1 + n_{\text{func}})^{\gamma \kappa}$ with the tuning parameter $0 < \gamma \kappa < 1$ and the number of function evaluations $n_{\text{func}}$. It is easy to show that $\tilde{p}^T \tilde{g} = (\epsilon p^o - \alpha^o \tilde{g}^o)^T \tilde{g}^o < 0$; hence the perturbed random directions are descent directions.

5.6 An improved trust region direction

The goal of trust region methods is to restrict steps within a trust region to increase the accuracy of surrogate models. Therefore, trust region directions can be very useful, even in the presence of noise.

We now solve the trust region subproblem in a subspace

$$\min \zeta^T \tilde{g}^o + \frac{1}{2} \zeta^T \tilde{B}^o \zeta$$

s.t. $\|\zeta - z^o\|_\infty \leq d,$

whose solution is denoted by $\zeta_{\text{best}}$. Here $d$ is denoted as the trust region radius. The trust region direction is $p_{tr}^o := \zeta_{\text{best}} - z^o$ in a subspace which is a random subset of $\{1, \ldots, n\}$ whose size is $m^o$. The idea is to construct the improved trust region direction by scaling $p_{tr}^o$ with the positive tuning parameter $\gamma_p$ and perturbing it by
\[ p_{\text{mean}} := z_{\text{mean}} - Z_b, \text{ where } z_{\text{mean}} \text{ is the mean of } Z. \] This perturbation transforms \( p \) into a full subspace direction enriched by \( p_{\text{mean}} \).

5.7 MLS – an improved version of MLS-basic

In this section, we discuss an improved version of the MLS-basic procedure, the MLS procedure. Namely, the MLS procedure is enriched by heuristic techniques for finding and updating step sizes.

The efficiency of line search methods depends on how their step sizes are updated. The line search algorithms discussed in [27] and [30] find their step sizes in a way that does not seem to be effective, especially for large scale problems, because the step sizes are updated independently; they depend only on the corresponding step size generated by the previous executions. To address this shortcoming, we construct an improved version of the MLS-basic procedure whose step sizes are generated and updated in a new way.

Let \( 0 \leq \alpha_{\text{init}} \leq \bar{\alpha}_{\text{init}} \leq \infty \) be the lower and upper bounds of an initial interval as the tuning parameters. We start with the initial interval \([\alpha, \bar{\alpha}] = [\alpha_{\text{init}}, \bar{\alpha}_{\text{init}}]\). Then \( \alpha \) or \( \bar{\alpha} \) is updated. We describe this below.

- **The initial interval.** When the trial point \( z^* = z_{\text{best}} \pm \alpha_r p^r \) and its inexact function value \( \tilde{f}(z_{\text{best}}) - \tilde{f}(z^*) \) is stored in the vector \( \mathbf{dF} \) and its step size \( \alpha_r \) in the vector \( \mathbf{a} \). Namely, when an extrapolation with many sufficient gains along \( p^r (r \in \{1, \ldots, R\}) \) is found, much information is stored in the vectors \( \mathbf{dF} \) and \( \mathbf{a} \). Then, an index set of the stored points with the decreasing inexact function values is found by \( \text{ind} := \{i \mid \mathbf{dF}_i < 0 \text{ for } i = 1, \ldots, n\} \). If \( \text{ind} \) is non-empty, the lower bound of the interval \([\alpha, \bar{\alpha}]\) can be found by \( \alpha := \max(\mathbf{a}_{\text{ind}}) \); if \( \alpha = \alpha_{\text{init}} \) is given as a positive tuning parameter, then \( \alpha \) is updated by \( \alpha := \min(\alpha, \max(\mathbf{a}_{\text{ind}})) \). Next, an index set of positive gains or the corresponding step sizes strictly larger than \( \bar{\alpha} \) is found by

\[
\text{ind} := \{i \mid \mathbf{dF}_i \geq 0 \text{ or } \mathbf{a}_i > \bar{\alpha} \text{ for } i = 1, \ldots, n\}.
\]

If \( \text{ind} \) is nonempty, an upper bound on the initial interval is found by \( \bar{\alpha} := \min(\alpha_{\text{ind}}) \); if \( \alpha = \alpha_{\text{init}} \) is given as a positive tuning parameter, the upper bound of the initial interval is updated by \( \bar{\alpha} := \max(\bar{\alpha}, \min(\alpha_{\text{ind}})) \).

- **The initial extrapolation step size.** If extrapolation step sizes are too small, extrapolations may be performed slowly with many function evaluations. To overcome this problem, we only change line 5 in VRBBON-basic to \( \alpha_1 := \max(\sqrt{\alpha_1 \alpha_{k-1}}) \) whenever the interval \([\alpha, \bar{\alpha}] \subseteq (0, \infty)\) is found.

- **Reducing extrapolation step sizes.** Since the source of the inefficiency of the line search algorithm is generating step sizes that are too small when the interval has already been found, we change line 22 in VRBBON-basic to

\[
\alpha_{r+1} := \max \left\{ \frac{\alpha_{\text{min}}}{\gamma_{\text{e}}}, \min \left\{ \sqrt{\alpha_{\text{min}} \alpha_r}, \alpha_{\text{min}} \right\} \right\}
\]
with the goal of slowly reducing the step sizes. Here $\pi_{\min} \in (0, 1)$ is a minimum threshold for the step size which is a tuning parameter.

- **Updating the interval.** After finding the new step size $\alpha_r$, we update either the lower or the upper bound of the interval. This can be done in the following two cases:
  (i) After extrapolation has been performed with $\gamma \alpha_r^2$-sufficient gain along one of $\pm \rho_r$ (good is true).
  (ii) After no $\gamma \alpha_r^2$-sufficient gain along $\pm \rho_r$ has been found (good is false).

To do this, we insert the following statement between lines 26 and 27 of VRBBON-basic:

1: if $\alpha_{r+1} > \alpha$ then, set $\alpha = \alpha_{r+1}$; \> upper bound is updated
2: else, set $\alpha = \alpha_{r+1}$; \> lower bound is updated
3: end if

- **Decrease of $f$ in flat regions.** If the $r$th iteration of the MLS procedure is unsuccessful (no $\gamma \alpha_r^2$-sufficient gain along $\pm \rho_r$ is found), the best point, its inexact function value, and its step size are updated when $\tilde{f}(z_{r+1} + \alpha_r \rho_r) < \tilde{f}(z_r)$. In particular, this can happen in very flat regions of the feasible domain that do not contain a nearby stationary point. Therefore, we insert the following statement between lines 25 and 26 in VRBBON-basic:

1: if $\tilde{f}(z') < \tilde{f}(z_{r+1})$ then, set $z = z'$ and $\tilde{f}(z) := \tilde{f}_{r}$; $n_{\text{ML}S \text{succ}} := n_{\text{ML}S \text{succ}} + 1$
2: end if

- **Updating the best point.** If the $r$th iteration of the MLS procedure is successful ($\gamma \alpha_r^2$-sufficient gains are found along one of $\pm \rho_r$). A point with lowest inexact function value is chosen as the new best point. As discussed in Subsection 5.2, the matrices $X$, $F$, and $S$ are then updated. In fact, lines 22 and 23 of VRBBON-basic should be changes as:

1: find $i' := \min_{i \geq 0} \left\{ \tilde{f}(z_{r+1} + \alpha_i \gamma_i \rho_r) \mid \tilde{f}(z_{r+1} + \alpha_i \gamma_i \rho_r) < \tilde{f}(z_{r+1}) \right\}$
2: compute $\alpha_{r+1} = \gamma_i \alpha_r$, $z_{r+1} := z_{r+1} + \alpha_{r+1} \rho_r$, and $\tilde{f}(z_{r+1}) := \tilde{f}(z_{r+1} + \alpha_{r+1} \rho_r)$;
3: update $n_{\text{ML}S \text{succ}} := n_{\text{ML}S \text{succ}} + 1$

5.8 **DS – an improved version of DS-basic**

This section discusses the DS procedure, an improved version of the DS-basic procedure that extended to include different search directions and heuristic techniques for updating step sizes and reconstructing the lower and upper bounds of the interval. It also describes how many worst case function evaluations are used in each successful and unsuccessful iteration of the DS procedure.

Denote by $d_t$ the trust region radius in the iteration $t$ and let $C$ be the number of random approximate coordinate directions. Let $\gamma_{d1} > 1$ and $\gamma_{d2} \in (0, 1)$ be the parameters for updating $d_t$ and let $0 < d_{\min} < d_{\max} < \infty$ be the parameters for controlling $d_t$. As described in Subsection 3, the DS-basic procedure has $R$ calls to the MLS-basic procedure with scaled random directions. An improved version
of the DS-basic procedure, the DS procedure, retains this procedure and additionally has \(C\) calls to the MLS-basic procedure with random approximate coordinate directions with the goal of restoring and updating the \(m\) sample points and their inexact function values. Then, the DS procedure constructs a quadratic model or a linear model to produce an improved trust region direction or a perturbed random direction, and makes \(T_0\) calls to the MLS procedure along these directions as long as the MLS procedure is efficient (good is true). If the DS procedure is inefficient for all \(T_0\) calls to the MLS procedure (\(n_{\text{succ}}^{DS} = 0\), the lower bound \(\alpha\) of the current interval \([\alpha, \pi]\) becomes too small, so the length of this interval becomes large. Therefore, we need to restart the interval and heuristically reconstruct the lower and upper bounds of this interval using the information from the \(m\) sample points. Let \(\gamma_a > 0\) be the parameter for adjusting heuristic step size. The structure of DS is shown below.

Algorithm 2 DS – an improved version of DS-basic

1: for \(t = 1, \cdots, T_0\) do
2: perform MLS with either \(R\) scaled random directions, \(C\) random
3: approximate coordinate directions, or both;
4: if \(m \geq 3\) then
5: while true do \(\triangleright\) until good is true
6: perform MLS with random subspace directions;
7: if good is false, break; end \(\triangleright\) MLS is inefficient
8: end while
9: if \(m \geq 2\) then
10: either a reduced quadratic or a linear model is constructed
11: if both estimations are computable then
12: construct a reduced quadratic model;
13: generate the trust region radius by \(d_t := \|z_{\text{mean}} - Z_b\|\);
14: restrict the trust region radius by \(d_t := \max(d_{\text{min}}, \min(d_{\text{max}}, \gamma d_t \cdot d_t))\);
15: minimize the model to get an improved trust region direction;
16: while true do \(\triangleright\) until good is true
17: perform MLS with improved trust region directions;
18: if good is false, break; end \(\triangleright\) MLS is inefficient
19: update \(d_t := (\gamma d_2 + \text{rand})d_t\); \(\triangleright\) rand \(\in (0, 1]\) is a random value
20: end while
21: else
22: while true do \(\triangleright\) until good is true
23: perform MLS with perturbed random directions;
24: if good is false, break; end \(\triangleright\) MLS is inefficient
25: end while
26: end if
27: end if
28: if \(n_{\text{succ}}^{DS}\) is zero then \(\triangleright\) MLS is inefficient in all \(T_0\) calls
29: for \(i = 1, \cdots, n\) do
30: compute \(d_{z_i} := Z_i - Z_b\); \(\triangleright\) the difference of the old good points
31: find \(I_i := \{j \mid d_{z_j} \neq 0 \text{ and } (Z_b)_j \neq 0\}\);
32: compute \(\beta_i^r := \min\{|(Z_b)_j/d_{z_j}| \mid j \in I_i\}\);
33: end for
34: generate two random values \(\mu_1\) and \(\mu_2\) satisfying \(0 < \mu_1 < \mu_2 < 1\);
35: reconstruct \([\alpha, \pi] := [\gamma a \mu_1 \beta_{\min}^{r}, \gamma a \mu_2 \beta_{\min}^{r}]\) with \(\beta_{\min}^{r} := \min_{i = 1, \cdots, n} \beta_i^r\);
36: end if
37: end for
We discuss how many function evaluations are required for each successful and unsuccessful iteration of the DS procedure in the worst case:

- Lines 2-3 of this procedure require at most $2R$ function evaluations if only scaled random directions are used, while this procedure is inefficient. This is because the MLS procedure is inefficient with $R$ scaled random directions and $R$ their opposite directions.

- Lines 2-3 of this procedure use at most $2C$ function evaluations if only random approximate coordinate directions are used, while this procedure is inefficient. Since the MLS procedure is inefficient with $C$ random approximate coordinate directions and $C$ their opposite directions.

- Lines 2-3 of this procedure use at most $2R + 2C$ function evaluations if both scaled random directions and random approximate coordinate directions are used, while this procedure is inefficient. Since the MLS procedure is inefficient with $R + C$ directions and $R + C$ their opposite directions.

- Lines 5-8 of this procedure use at most 2 function evaluations, while this procedure is inefficient. This is because the MLS procedure is inefficient with the first random subspace direction and its opposite direction.

- Either lines 16-20 or 22-26 of this procedure use at most 3 function evaluations. Since an extrapolation step is attempted with at most two function evaluations, the MLS procedure is efficient along the opposite direction of the last improved trust region direction (or the last random perturbed direction). Then, the best point is updated.

The result of this discussion is that the DS procedure requires at most $2R + 2C + 4$ function evaluations for each unsuccessful iteration and at most $2R + 2C + 5$ function evaluations for each successful iteration. In the next section, we use this discussion to determine how the complexity results for an implemented version of VRBBON-basic will change compared to the complexity results of VRBBON-basic.

6 VRBBON – an improved version of VRBBON-basic

As mentioned earlier, VRBBON is an implemented version of VRBBON-basic. It uses the DS and MLS procedures instead of their basic versions, and uses all practical enhancements to achieve an $\epsilon = \sqrt{\omega}$-approximate stationary point.

If the step sizes are too small, VRBBON-basic may end up in strongly non-convex regions before a minimizer is found. To overcome this drawback, we change line 42 in VRBBON-basic to $\delta_k := \max(\delta_{k-1}, \sqrt{\alpha})$ if the interval $[\alpha, \alpha]$ has already been found.

7 VRBBON

This section discusses some implementation details, tuning parameters, and under which conditions the complexity results for VRBBON can be guaranteed.

To obtain the improved trust region directions discussed in Subsection 5.6, we solve the trust region subproblem by ming2 [25], available at
minq8 requires the tuning parameters \( \text{minqmax} = 10000 \) and \( \text{minqeps} = 10^{-8} \).

The values of tuning parameters of VRBBON are chosen in Table 1, except the other five tuning parameters \text{comBound} (are the complexity results valid?), \text{model} (model-based?), \( R \geq 2 \) (number of scaled random directions in MLS), \( C \geq 2 \) (number of random approximate coordinate directions), and \( T_0 \geq 1 \) (number of times MLS is called by DS). \text{model} is a Boolean variable taking 0 (model-free) and 1 (model-based).

Table 1: The values of some tuning parameters of VRBBON, except \text{comBound}, \text{model}, \( T_0 \), and \( R \)

| \( m = 230 \) | \( q_{d1} = 2 \) | \( d_{\text{max}} = 10^2 \) | \( c_{\text{max}} = 10^{-5} \) | \( d_{\text{min}} = 1 \) | \( \overline{d}_{\text{init}} = 0.01 \) | \( \overline{d}_{\text{init}} = 0.99 \) | \( \overline{d}_{\min} = 10^{-3} + \text{rand} \) | \( \delta_{\text{min}} = 10^{-4} \) | \( \gamma_c = 100 \) | \( \gamma_c = 0.85 \) | \( \gamma_{d2} = 0.5 \) |

In the following three cases we discuss the conditions under which complexity results can be found for VRBBON:

Case 1. \text{comBound} = 0. In this case, random scaled directions are used and random approximate coordinate directions are ignored. Other proposed directions and heuristic techniques are also used. The order of complexity bounds does not change for all cases after applying the heuristic improvements, only their factors become larger.

Case 2. \text{comBound} = 1. Both random scaled directions and random approximate coordinate directions are used. Other proposed directions and heuristic techniques are also used. In this case, the order of complexity bounds does not change for all cases, such as Case 1, after heuristic improvements are used, but only their factors become larger compared to VRBBON and Case 1. In fact, random approximate coordinate directions are numerically better than random scaled directions, but Proposition 1 may not hold for random approximate coordinate directions.

Case 3. \text{comBound} = 2. In this case, random approximate coordinate directions are used and random scaled directions are ignored. Other proposed directions and heuristic techniques are also used. In this case, Proposition 1 may not be valid and no complexity result is found for VRBBON.

In Case 1 and Case 2, Theorem 1, Theorem 2, and Theorem 4 remain valid with the following modifications:

- In Theorem 1(i), \( R \) (number of scaled random steps) must be replaced by \( T \) (number of all trial steps), and Theorem 1(ii) remains valid with probability \( \geq 1 - \eta \) for a given \( 0 < \eta \leq \frac{1}{2} \).

- In Theorem 2(i), the number of function evaluations of DS is bounded by

\[
(2R + 4)T_0 + (2R + 5)T_0 \bar{T}^{-1} \delta_k^{-2} \left( f(x^0) - \hat{f} + 2\omega \right)
\]

in Case 1 and

\[
(2R + 2C + 4)T_0 + (2R + 2C + 5)T_0 \bar{T}^{-1} \delta_k^{-2} \left( f(x^0) - \hat{f} + 2\omega \right)
\]

28
in Case 2. As mentioned earlier, the step size $\delta_k$ is fixed, independent of $t$, and updated outside the DS procedure. We have described at the end of Section 5.8 how to obtain the factors of the above two bounds. Theorem 3(ii) remains valid.

- In the proof of Theorem 4, $2RT_0$ and $(2R + 1)T_0$ must be replaced by $(2R + 4)T_0$ and $(2R + 5)T_0$ respectively in Case 1, and by $(2R + 2C + 4)T_0$ and $(2R + 2C + 5)T_0$ respectively in Case 2.

8 Numerical results

In this section, we describe how the test problems are selected and give details of the solvers compared. Performance measures are then defined to determine which solvers are robust and efficient for small to large scale problems. Next, a testing and tuning are performed to increase the efficiency and robustness of our solver while preserving the complexity results. Then, we compare our solver with state-of-the-art solvers for low to high dimensional problems. Afterwards, a real-life problem for the dimension $n = 300$ and $n = 5000$ is used to observe how VRBBON approaches a minimizer of this problem compared to other solvers. Finally, we make a recommendation as to which solvers are robust and efficient based on dimension and noise level.

8.1 Test problems

For our numerical results, we used the 549 unconstrained CUTEst test problems from the collection of Gould et al. [16]. To prepare these results, the test environment of Kimiaei & Neumaier [26] was used.

The starting point. As in [27], we choose the starting point $x^0 := 0$ and shift the arguments by

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

[to avoid a solver guessing the solution of toy problems with a simple solution (e.g., all zeros or all ones) – there are quite a few of these in the CUTEst library. That is, the initial point is chosen by $x^0 := \xi$ and the initial inexact function value is $\tilde{f}_0 := \tilde{f}(x^0)$ while the other inexact function values are computed by $\tilde{f}_\ell := \tilde{f}(x^\ell + \xi)$ for all $\ell \geq 0$. In fact, this choice increases the difficulty of the problems, see Section 8.5.1.

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 $\tilde{f} = f + (2 \times \text{rand} - 1)\omega$, 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.
8.2 Codes compared

We compare VRBBON with:

• **VRBBO** – an efficient random algorithm by Kimiae & Neumaier [27]. It can be downloaded from

  https://www.mat.univie.ac.at/~neum/software/VRBBO/.

• **SNOBFIT**, obtained from

  http://www.mat.univie.ac.at/~neum/software/snobfit/snobfit_v2.1.tar.gz,

  is a combination of a branching strategy to enhance the chance of finding a global minimum with a sequential quadratic programming method based on fitted quadratic models to have good local properties by Huyer & Neumaier [24].

• **GRID** – a grid algorithm for bound constrained optimization by Elster & Neumaier [15], available at

  http://www.mat.univie.ac.at/~neum/softwareGRID.

  This solver was originally written in Fortran with auxiliary routines that are no longer available. We reimplemented GRID in Matlab. The trust region subproblem is solved with **minq8** [25].

• **UOBYQA** and **NEWUOA**, obtained from

  https://www.pdfo.net/docs.html,

  are model-based solvers by Powell [35,36].

• **BFO** – a trainable stochastic derivative free solver for mixed integer bound-constrained optimization by Porcelli & Toint [34], available at

  https://github.com/m01marpor/BFO.

• **DSPFD** – a direct search MATLAB code for derivative-free optimization by Gratton et al. [17], available at

  pages.discovery.wisc.edu/%7Ecroyer/codes/dspfd_sources.zip.

• **MCS** – a deterministic global optimization by a multilevel coordinate search by Huyer & Neumaier [23], downloaded from

  https://www.mat.univie.ac.at/~neum/software/mcs/.

  It used the following parameters:

  ```
  iinit = 1; nfMCS = nfmax; smax = 5n + 10; stop = 3n; local = 50;
  gamma = eps; hess = ones(n,n);prt = 0.
  ```

• **BCDFO** – a deterministic model-based trust region algorithm for derivative-free bound constrained minimization by Gratton et al. [19], obtained from Anke
Troeltzsch (personal communication).

- **FMINUNC** – a deterministic quasi Newton or trust region algorithm, available at the Matlab Optimization Toolbox at https://ch.mathworks.com/help/optim/ug/fminunc.html. The following options set was used:

```matlab
opts = optimoptions(@fminunc,'Algorithm','quasi-newton','Display','Iter','MaxIter',Inf,'MaxFunEvals',limits.nfmax,'TolX',0,'TolFun',0,'ObjectiveLimit','-1e-50');
```


- **CMAES**, obtained from http://cma.gforge.inria.fr/count-cmaes-m.php?Down=cmaes.m, is the stochastic covariance matrix adaptation evolution strategy by Auger & Hansen [2]. We used CMAES with the tuning parameters

```matlab
ocMAES.MaxFunEvals = nfmax, ocMAESDispFinal = 0, ocMAESDispModulo = 0,
ocMAES.LogModulo = 0, ocMAES.SaveVariables = 0, ocMAES.MaxIter = nfmax,
ocMAES.Restarts = 7.
```

- **LMMAES** by Loshchilov et al. [29], fMAES by Beyer [6], BiPopMAES by Beyer & Sendhoff [7], obtained from https://homepages.fhv.at/hgb/downloads.html, are three various covariance matrix adaptation evolution strategies.

- **subUOBYQA**, subNEWUOA and subNMSMAX are UOBYQA, NEWUOA and NMSMAX, respectively, in the random subspace generated by the columns of $S := 2 \ast \text{rand}(n, m) - 1$ (in Matlab) to handle problems in medium and high dimensions. Here $n$ is the dimension of the problem and $m$ is the subspace dimension. Using these solvers, we recursively minimize the problem $\tilde{f}(x + \Delta z)$ with respect to $z \in \mathbb{R}^n$. Let $\Delta$ be the trust region radius, which is initially one, and let $0 < \gamma < 1$ be a parameter for the decrease of the function value. If $f(x) > f(x + \Delta z) - \gamma \Delta^2$ holds, the trial point $x + \Delta z$ can be accepted as a new point and $\Delta$ is expanded to $\Delta = \max(\Delta, ||S||)$; otherwise $\Delta$ is reduced to $\Delta = \frac{1}{2} \min(\Delta, ||S||)$. For each call to these solvers, the maximum number of function evaluations is $\max([n/m], 10m)$. According to our findings, the best value for the subspace dimension $m$ is 10. For $m < 10$ the efficiency and robustness of these subspace solvers decrease when they are used to solve problems in medium and high dimensions, while for $10 < m \leq 30$ the number of sample points to construct the quadratic models increases, making each call to the original solver (UOBYQA, NEWUOA and NMSMAX) costly.
Unfortunately, software for **FDLM** by Berahas et al. [4] and **STRRS** by Chen [9] was not available to us.

### 8.3 Performance measures

Two important tools for figuring out which solver is **robust** and **efficient** are the data profile of More & Wild [32] and the performance profile of Dolan & More [14], 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 $\kappa$ groups of $n_p + 1$ function evaluations is the data profile of the solver $s$, i.e.,

$$
\delta_s(\kappa) := \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} \mid cr_{p,s} := \frac{c_{p,s}}{n_p + 1} \leq \kappa \right\} \right|.
$$

(42)

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 $\tau$ is the performance profile of the solver $s$, i.e.,

$$
\rho_s(\tau) := \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} \mid pr_{p,s} := \frac{c_{p,s}}{\min(c_{p,s} \mid s \in \mathcal{S})} \leq \tau \right\} \right|.
$$

(43)

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 $\tau$ ($\kappa$) that the solver $s$ can solve. **The data and performance profiles are based on the problem scales, but not on the noise levels.** The other two plots are based on the noise levels. These four plots are used to identify the behaviour of the compared solvers with respect to problem scales and noise levels.

**Efficiency.** The **efficiency** $e_{p,s}$ of the solver $s$ to solve the problem $p$ is the inverse of the performance ratio $pr_{p,s}$. Efficiency measures the ability of a solver $s \in \mathcal{S}$ relative to an ideal solver. The number 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.

**Other plots based on the noise level.** To see the behavior of the compared solvers in the presence of low to high noise, we plot the number of problems solved and the efficiency versus the noise level.

**Measure for the convergence speed.** The quotients

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

(44)

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.

- $f_s$ is the best function value found by the solver $s$,
- $f_0$ is the function value at the starting point (common for all solvers),

32
• $f_{\text{opt}}$ 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 [27].

Maximum budgets and stopping tests.

We consider a problem solved by the solver $s$ if $q_s \leq \varepsilon$ and neither the maximum number $\text{nmax}$ of function evaluations nor the maximum allowed time $\text{semax}$ in seconds is satisfied, and unsolved otherwise. $\varepsilon$, $\text{semax}$ and $\text{nmax}$ are chosen so that the best solver can solve at least half of the problems. They depend on the dimension and the noise level because increasing the noise level and dimension extremely increases the difficulty of the problems. Therefore, $\varepsilon$ is chosen slightly larger for problems in medium and high dimensions than for problems in low dimensions. The following choices were found valuable:

$$\text{semax} = \begin{cases} 180 & \text{if } 1 \leq n \leq 300, \\ 420 & \text{if } 301 \leq n \leq 5000, \end{cases}$$

$$\text{nmax} = \begin{cases} 2n^2 + 1000n + 5000 & \text{if } 1 \leq n \leq 300, \\ 500n & \text{if } 301 \leq n \leq 5000, \end{cases}$$

and

$$\varepsilon := \begin{cases} 10^{-3} & \text{if } \omega \in \{10^{-4}, 10^{-3}\} \text{ and } n \in [1, 30], \\ 10^{-2} & \text{if } \omega \in \{0.1, 0.9\} \text{ and } n \in [1, 30], \\ 10^{-3} & \text{if } \omega = 10^{-4} \text{ and } n \in [31, 300], \\ 0.05 & \text{if } \omega \in \{0.1, 0.01, 0.001\} \text{ and } n \in [31, 300], \\ 0.05 & \text{if } \omega \in \{10^{-5}, 10^{-4}, 10^{-3}\} \text{ and } n \in [301, 5000]. \end{cases}$$

8.4 Tuning of VRBBON

Five important tuning parameters are $\text{comBound}$ (are the complexity results valid?), $\text{model}$ (model-based?), $R$ (number of scaled random directions in MLS), $C$ (number of random approximate coordinate directions), and $T_0$ (number of times MLS is called by DS). Other tuning parameters were chosen fixed as they did not change the efficiency and robustness of our solver. Accordingly, for a testing and tuning for small scale problems ($1 < n \leq 30$), we consider the 30 versions of VRBBON given in Table 1. Then we chose the four best versions of VRBBON and ran them for medium scale problems ($31 < n \leq 300$) and large scale problems ($300 < n \leq 5000$) to select the best version of VRBBON to compare with the other solvers. Finding the optimal tuning parameters for VRBBON and the other goal is still a work in progress [26].
For the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.9\}$ and small scale problems ($1 < n \leq 30$), Figure 2 shows in its subfigures the cumulative (over all noise levels used) performance and data profiles in terms of the number of function evaluations and the other two plots (the $\text{nf eff}$iciency versus the noise level $\omega$ and the number of solved problems versus the noise level $\omega$). The result of this comparison is that VRBBON3 is slightly more robust than the others, since all the proposed directions are used. For this version, the complexity results are valid since random scaled directions were tried. VRBBON1 and VRBBON2 are slightly more efficient than VRBBON3 and VRBBON, except for high noise. In fact, using scaled random directions guarantees the existence of complexity results and increases the robustness of our solver under low to high noise and the efficiency of our solver only under high noise. Moreover, since VRBBON1, VRBBON3, and VRBBON are model-based, it is confirmed that VRBBON is effective when it is model-based.

For the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and medium scale problems ($30 < n \leq 300$), Figure 3 shows in its subfigures the cumulative (over all noise levels used) performance and data profiles in terms of the number of function evaluations and the other two plots (the $\text{nf eff}$iciency versus the noise level $\omega$ and the number of solved problems versus the noise level $\omega$). We conclude from these subfigures that VRBB-
Fig. 2: For the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-1}, 0.9\}$ and small dimensions $1 < n \leq 30$. Data profile $\delta(\kappa)$ in dependence of a bound $\kappa$ on the cost ratio, see (42) while performance profile $\rho(\tau)$ in dependence of a bound $\tau$ on the performance ratio, see (43). Problems solved by no solver are ignored.

**BON1** and **VRBBON** are slightly more efficient and robust than **VRBBON2** and **VRBBON3**.

Fig. 3: For the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and medium dimensions $30 < n \leq 300$. Other details as in Figure 2.

For the noise levels $\omega \in \{10^{-5}, 10^{-4}, 10^{-3}\}$ and large scale problems ($300 < n \leq 5000$), Figure 4 shows in its subfigures, the cumulative (over all noise levels used) performance and data profiles in terms of the number of function evaluations and the other two plots (the $nf$ efficiency versus the noise level $\omega$ and the number of solved problems versus the noise level $\omega$). We conclude from these subfigures that **VRBBON1** and **VRBBON** are more efficient and robust than **VRBBON2** and **VRBBON3**.

As a result, we choose **VRBBON** as default version to compare with the other solvers for problems in low to high dimensions. Note that our complexity results hold for this version.

Fig. 4: For the noise levels $\omega \in \{10^{-5}, 10^{-4}, 10^{-3}\}$ and large dimensions $300 < n \leq 5000$. Other details as in Figure 2.
8.5 Comparison with the other solvers

This section compares the default version of VRBBON with the other solvers for problems in low to high dimensions. In each figure, we display only the best five solvers, but the best four solvers in high dimensions.

8.5.1 Small scale: $1 < n \leq 30$

For the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-1}, 0.9\}$ and small scale problems ($1 < n \leq 30$), this section contains a comparison between VRBBON and well-known model-based solvers (UOBYQA, NEWUOA, BCDFO, GRID, and SNOBFIT), direct search solvers (NMSMAX, BFO, MCS, DSPFD), line search solvers (VRBBO, SDBOX, FMINUNC), and matrix adaptation evolution solvers (CMAES, FMAES, BiPopMAES, LMMAES). Moreover, a comparison is given between standard and shifted initial points.

To compare our algorithm with the well-known model-based and direct search solvers, Figure 5 shows the cumulative (over all noise levels used) performance and data profiles in its subfigures in terms of the number of function evaluations and the other two plots (the nf efficiency versus the noise level $\omega$ and the number of solved problems versus the noise level $\omega$). From the subfigures of Figure 5, NEWUOA is more efficient than the well-known model-based solvers and VRBBON, while UOBYQA is more robust than others. On the other hand, VRBBON, the second robust solver, is more efficient than SNOBFIT and GRID and less efficient than others. Moreover, VRBBON and NMSMAX are more robust and efficient than the well-known direct search solvers. In fact, VRBBON is more robust than NMSMAX, while NMSMAX is more efficient than VRBBON.

![Fig. 5: Comparison between VRBBON and model-based solvers (first row) and direct search solvers (second row) for the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-1}, 0.9\}$ and small dimensions $1 < n \leq 30$. Other details as in Figure 2.](image-url)
To compare our algorithm with the well-known line search and matrix adaptation evolution solvers, Figure 6 shows in its subfigures, the cumulative (over all noise levels used) performance and data profiles in terms of the number of function evaluations and the other two plots in terms of the noise levels (the nf efficiency versus the noise level $\omega$ and the number of solved problems versus the noise level $\omega$). From the subfigures of Figure 6, we conclude that VRBBON is more robust and efficient than the known line search solvers, while VRBBON is more robust and efficient than the known matrix adaptation evolution solvers at low noise and fMAES is more robust and efficient than others at high noise. At low to high noise, VRBBON is slightly more efficient than the known matrix adaptation evolution solvers, while fMAES is more robust than others.

Since our solver is model-based and line search-based, we now provides two comparisons between VRBBON and the best model-based (UOBYQA, NEWUOA, and BCDFO) and line search solvers (SDBOX and VRBBO) with the shifted and initial starting points for small scale problems.

If the default starting points are used, we conclude from Figure 7 that VRBBON is comparable to NEWUOA in terms of efficiency and is more efficient than others. It is also more robust than others. In fact, the efficiency and robustness of VRBBON are higher when the standard initial points are used than when the shifted initial points are used, see Figure 8. Therefore, we used the shifted points for all test problems.

8.5.2 Medium scale: $30 < n \leq 300$

For the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and medium scale problems ($30 < n \leq 300$), this section contains a comparison between VRBBON and the well-known direct search solvers, line search solvers, and matrix adaptation evolution solvers.
VRBBON is more robust and efficient than the known direct search solvers. From the subfigures of Figure 9, we conclude that VRBBON is more robust and efficient than the known direct search solvers. From the subfigures of Figure 10, we can also conclude that VRBBON is more robust and efficient than the known line search solvers and matrix adaptation evolution solvers at low and medium noise, while LMMAES is more robust than VRBBON and the other matrix adaptation evolution solvers at high noise.

Since the model-based solvers cannot handle problems in high dimensions, as described earlier, we denote UOBYQA in the random subspace by subUOBYQA and NEWUOA in the random subspace by subNEWUOA. Moreover, we denote NMSMAX in the random subspace by subNMSMAX.

The subfigures of Figure 11 show that VRBBON are much more efficient than others at low to high noise, but VRBBON are more robust than others at low
noise, while subNEWUOA, subUOBYQA, and NMSMAX are slightly more robust than VRBBON only at high noise.

Fig. 11: Comparison between VRBBON and model-based solvers in a random subspace for the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and medium dimensions $30 < n \leq 300$. Other details as in Figure 2.

8.5.3 Large scale: $300 < n \leq 5000$

For the noise levels $\omega \in \{10^{-5}, 10^{-4}, 10^{-3}\}$ and large dimensions $300 < n \leq 5000$, this section contains a comparison between VRBBON and the four effective solvers (VRBBO, LMMAES, and SDBOX) for problems in medium dimensions.

Figure 12 shows in its subfigures, the cumulative (over all noise levels used) performance and data profiles in terms of the number of function evaluations for the noise levels $\omega \in \{10^{-5}, 10^{-4}, 10^{-3}\}$ and the other two plots (the nf efficiency versus the noise level $\omega$ and the number of solved problems versus the noise level $\omega$). We conclude from these subfigures that VRBBON is slightly more efficient than others, while VRBBO and SDBOX are slightly more robust than VRBBON.
8.6 Comparison on a real-life test problem

Consider the driven cavity problem

\[ F(x) = \frac{1}{2} f(x)^T f(x) \]

from the collection Lukšan & Vlcek [31], where the equation \( f(x) = 0 \) is a finite difference analogue of the nonlinear partial differential equation

\[
\Delta \Delta u + \theta \left( \frac{\partial u}{\partial y} \frac{\partial \Delta u}{\partial x} - \frac{\partial u}{\partial x} \frac{\partial \Delta u}{\partial y} \right) = 0, \quad \theta = 500,
\]

over the unit square \( \Omega \) with the boundary conditions \( u = 0 \) on \( \partial \Omega \) and

\[
\frac{\partial u(0,y)}{\partial x} = 0, \quad \frac{\partial u(1,y)}{\partial x} = 0, \quad \frac{\partial u(x,0)}{\partial y} = 0, \quad \frac{\partial u(x,1)}{\partial y} = 1.
\]

We chosen 13 standard finite difference points on a shifted uniform grid with 50 \( \times \) 50 internal nodes. The initial approximate solution is a discretization of \( u_0(x,y) = 0. \)

We chosen the initial point as \( x_0 = \text{rand}(n,1) - 0.5. \) We ran VRBBON, VRBBO and LMMMAES on this problem with the dimension \( n \in \{300, 5000\} \), the budgets \( n\text{fmax} = 200n \) and \( n\text{ecmax} = \infty \), the accuracy \( \varepsilon \in \{0.01, 0.001\} \), and \( \omega \in \{0.001, 0.0001, 0.00001\} \). Then the relative function values \( q_s \), defined by (44), are plotted against the number of function evaluations in Figures 13 and 14. It can be seen that VRBBON approaches a minimizer slightly faster than others. Note that \( f_{\text{best}} \approx 0.01 \) in (44) is known to us. In fact, it was obtained by running a sequence of solvers.
Problem = Driven cavity, $n = 300$

Fig. 13: Convergence (relative function values $q_s$ against number of function evaluations) of **VRBBON**, **VRBBO** and **LMMAES** on the real-life test problem (Driven cavity) with the dimension $n = 300$, the accuracy $\varepsilon = 0.001$, $\text{secmax} = \infty$ sec, $\text{nfmax} = 200n$, and $f_{\text{best}} = 0.01$.

Fig. 14: Convergence (relative function values $q_s$ against number of function evaluations) of **VRBBON**, **VRBBO** and **LMMAES** on the real-life test problem (Driven cavity) with the dimension $n = 5000$, the accuracy $\varepsilon = 0.01$, $\text{secmax} = \infty$ sec, $\text{nfmax} = 200n$, and $f_{\text{best}} = 0.01$.

### 8.7 Recommendation

Using Figures 12, 15, and 16, we have recommended which solvers are robust and which are efficient, depending on the noise level and dimension. Indeed, for small scale problems, Figure 15 is a comparison between five more robust and efficient solvers, and for medium scale problems, Figure 16 is a comparison between five more robust and efficient solvers. As shown in Subsection 8.5.1, Figure 12 is a comparison between four more robust and more efficient solvers.

Figure 17 is a Flow chart, classified by the problem dimension and the noise level, with the result that **VRBBON** is one of the three more robust and efficient solvers in most cases. Therefore, this solver is highly recommended for NBBOP.
Fig. 15: Comparison between five more robust and efficient solvers for the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-2}, 0.9\}$ and small dimensions $1 < n \leq 30$. Other details as in Figure 2.

Fig. 16: Comparison between five more robust and efficient solvers for the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and medium dimensions $30 < n \leq 300$. Other details as in Figure 2.
Fig. 17: Flow chart classified by the problem dimension and the noise level.
References


