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

Line search in noisy unconstrained black box optimization

Morteza Kimiaei[1]

the date of receipt and acceptance should be inserted later

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/~kimiaei/

1 Introduction

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

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1}$$

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

$$|\tilde{f}(x) - f(x)| \leq \omega. \tag{2}$$

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

[35,36], 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 $\mathcal{L}(x^0) := \{x \in \mathbb{R}^n \mid f(x) \leq f(x^0)\}$ 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

$$\widehat{f} := \inf\{f(x) \mid x \in \mathbb{R}^n\} = f(\widehat{x}) > -\infty \tag{3}$$

for any global minimizer $\widehat{x}$ 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
• $\mathcal{O}(\varepsilon^{-2})$ function evaluations for general case,
• $\mathcal{O}(\varepsilon^{-1})$ function evaluations for convex case,
• $\mathcal{O}(\log \varepsilon^{-1})$ 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 **FDLM**, has asymptotic accuracy

$$f - \widehat{f} = \mathcal{O}(L\omega). \tag{4}$$

i.i.d. stands for the independent and identically distributed. Chen [9] proposed a randomized algorithm with Gaussian directions and estimated step sizes, called **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 ζ is a random vector with probability distribution $\Pr(\zeta)$,

(ii) for all $x \in \mathbb{R}^n$, ω is i.i.d with bounded variance $\mathrm{var}(\omega) > 0$,
(iii) for all $x \in \mathbb{R}^n$, the noise is unbiased, i.e., $\mathbf{E}_\zeta(\omega) = 0$,
(iv) f is **convex** and (A1) holds,
(v) $\mathrm{var}(\omega) \leq \mathcal{O}(\varepsilon/n)$,
STRRS needs at most $\mathcal{O}(nL\varepsilon^{-1})$ to ensure that

$$x_N := \underset{x}{\operatorname{argmin}}\{f(x) \mid x \in \{x^0, \cdots, x_N\}\}$$

satisfies $\mathbf{E}[f(x_N)] - \widehat{f} \leq \varepsilon$.
• For the **bound constrained case**, Elster & Neumaier [15] introduced a grid algorithm, called **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 kth refinement step, where h^k denotes the kth grid size. If $h^k := \Theta(\sqrt{\omega})$, then the best order of magnitude can be obtained at least for a point with the gradient

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

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 **SDBOX**, using only the coordinate directions. They proved that, for any k,

$$\|g(x^k)\|_* = \mathcal{O}\Big(n^{3/2}L\mathbf{a}^k_{\max} + \frac{n\omega}{\mathbf{a}^k_{\min}}\Big), \ \text{ for } x^k \in \mathbb{R}^n.$$

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

$$\|g\|_* = \mathcal{O}\Big(n^{3/2}\sqrt{\omega}\Big). \tag{6}$$

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

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

$$\tilde{f}(x + \alpha p) \leq \tilde{f}(x) + c_1\alpha p^T\tilde{g}(x) + 2\omega \ \text{ with } 0 < c_1 < 1 \tag{7}$$

but step sizes are updated in a different way. Here $\tilde{f}(x) - f(x) = \omega(x;\zeta)$ is stochastic where ζ 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

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

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 ω. In the general case, $\mathcal{O}(\varepsilon^{-2})$ function evaluations with $\mathbf{E}(\|g\|_*) \leq \varepsilon$ are used. In the convex and strongly convex cases, $\mathcal{O}(\varepsilon^{-1})$ and $\mathcal{O}(\log\varepsilon^{-1})$ function evaluations with $\mathbf{E}(\|g\|_*) \leq \varepsilon$ and $\mathbf{E}(f - \widehat{f}) \leq \varepsilon$, 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** (Kimiaei & 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 $c_1\alpha p^T\tilde{g}(x)$ 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/~kimiaei/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 $\gamma_{\rm rd}/\|p\|$, where $0<\gamma_{\rm rd}<1$ is a tiny tuning parameter, resulting in $\|p\|=\gamma_{\rm rd}$.

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*

$$\Pr(\|g(x)\|_*\|p\|\le 2\sqrt{cn}|g(x)^Tp|)\ge\frac{1}{2}$$

with a positive constant $c\le 12.5$. 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

Given the tuning parameters $Q > 1$, $0 < \gamma < 1$, $\gamma_e > 1$, $\delta_{\max} > 0$, $0 \le \delta_{\min} \le 1$, $R \ge 2$ (integer), $T_0 \ge 1$ (integer).

Compute $\tilde{f}_0 := \tilde{f}(x^0)$ and set $n_{\rm func} := 1$, $\delta_0 := \delta_{\max}$, $z_{\rm best} := x^0$, and $\tilde{f}(z_{\rm best}) := \tilde{f}(x^0)$.
for $k = 1, 2, 3, \cdots$ **do**
  initialize the number of successful iterations of **DS-basic** by $n^{\tt DS}_{\tt succ} := 0$;
  for $t = 1, \cdots, T_0$ **do** ▷ starting **DS-basic**
    initialize the step size $\alpha_1 := \delta_k$; ▷ starting **MLS-basic**
    initialize the number of successful iterations of **MLS-basic** by $n^{\tt MLS}_{\tt succ} := 0$;
    for $r = 1, \cdots, R$ **do**
      compute the scaled random direction p^r and set `good` $:= 0$;
      while true **do** ▷ starting extrapolation along one of $\pm p^r$; if possible
        compute $z^r := z_{\rm best} + \alpha_r p^r$ and $\tilde{f}(z^r)$;
        set $n_{\rm func} := n_{\rm func} + 1$; ▷ counting the number `nfmax` of function evaluations
        if $n_{\rm func}$ reaches `nfmax` **then**, **VRBBON-basic** ends;
        end if
        if $\tilde{f}(z_{\rm best}) - \tilde{f}(z^r) > \gamma\alpha_r^2$ **then** ▷ $\gamma\alpha_r^2$-sufficient gain along p^r is found
          set `good` $:= 1$, $\tilde{f}^r_e := \tilde{f}(z^r)$, and expand the step size to $\alpha_r := \gamma_e \alpha_r$;
        else if $-p^r$ has not been tried already **then**
          set $p^r := -p^r$; ▷ opposite direction is tried
        else, **break**; ▷ no $\gamma\alpha_r^2$-sufficient gain along $\pm p^r$
        end if
      end while
      if `good` **then** ▷ the rth iteration of **MLS-basic** is **successful**
        set $\alpha_{r+1} := \dfrac{\alpha_r}{\gamma_e}$ and $z_{\rm best} := z_{\rm best} + \alpha_{r+1} p^r$; ▷ ending extrapolation
        update $\tilde{f}(z_{\rm best}) := f^r_e$ and $n^{\tt MLS}_{\tt succ} := n^{\tt MLS}_{\tt succ} + 1$;
      else ▷ the rth iteration of **MLS-basic** is **unsuccessful**
        reduce the step size to $\alpha_{r+1} := \dfrac{\alpha_r}{\gamma_e}$;
      end if
    end for ▷ ending **MLS-basic**
    if $n^{\tt MLS}_{\tt succ} > 0$ **then** ▷ the tth iteration of **DS-basic** is **successful**
      set $n^{\tt DS}_{\tt succ} := n^{\tt DS}_{\tt succ} + 1$, $y^t := z_{\rm best}$, and $\tilde{f}(y^t) := \tilde{f}(z_{\rm best})$;
    else ▷ the tth iteration of **DS-basic** is **unsuccessful**
      set $y^t := x^k$ and $\tilde{f}(y^t) := \tilde{f}(x^k)$; ▷ **MLS-basic** is **inefficient**
    end if
  end for
  if $n^{\tt DS}_{\tt succ} > 0$ **then** set $t' := \underset{t=1:T_0}{\rm argmin}\{\tilde{f}(y^t)\}$, $x^k := y^{t'}$, and $\tilde{f}(x^k) := \tilde{f}(y^{t'})$;
  else, set $x^k := x^{k-1}$ and $\tilde{f}(x^k) := \tilde{f}(x^{k-1})$; ▷ **DS-basic** is **inefficient**
  end if ▷ ending **DS-basic**
  if $\delta_{k-1} \le \delta_{\min}$ **then** set $x_{\rm best} := x^k$ and $\tilde{f}_{\rm best} := \tilde{f}(x^k)$; **VRBBON-basic** ends.
  end if
  if $n^{\tt DS}_{\tt succ}$ is zero **then** ▷ the kth iteration of **VRBBON-basic** is **unsuccessful**
    set $\delta_k := \delta_{k-1}/Q$; ▷ δ_k is reduced
  else ▷ the kth iteration of **VRBBON-basic** is **successful**
    set $\delta_k := \delta_{k-1}$; ▷ δ_k remains fixed
  end if
end for

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 $r' := \underset{r=1:R}{\operatorname{argmin}}\{\tilde{f}(z^r)\}$, we denote by $z_{\mathrm{best}} := z^{r'}$ the **best point** found by the **MLS-basic** procedure and by $\tilde{f}(z_{\mathrm{best}}) := \tilde{f}(z^{r'})$ 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** α_r by the tuning parameter $\gamma_e > 1$ along the fixed direction p^r and compute the corresponding trial point $z^r = z_{\mathrm{best}} + \alpha_r p^r$ and its inexact function value $\tilde{f}(z^r)$, as long as the condition

$$\tilde{f}(z_{\mathrm{best}}) < \tilde{f}(z^r) - \gamma\alpha_r^2$$

is satisfied, where $0 < \gamma < 1$ is a tuning parameter. The **MLS-basic** procedure accepts the trial point $z^r = z_{\mathrm{best}} + \alpha_r p^r$ as the new best point $z_{\mathrm{best}} = z^r$. We denote $\tilde{f}(x) - \tilde{f}(z^r)$ 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 rth 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 $n_{\mathtt{succ}}^{\mathtt{MLS}}$ to count the number of successful iterations and is called **inefficient** when it terminates with $n_{\mathtt{succ}}^{\mathtt{MLS}} = 0$. After the rth 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 $\alpha_{r+1} = \alpha_r/\gamma_e$ to obtain the corresponding step size of the penultimate point $z_{\mathrm{best}} := z_{\mathrm{best}} + \alpha_r p^r$, while its inexact function value $\tilde{f}(z_{\mathrm{best}}) := \tilde{f}_e^r$ has already been computed. During the rth iteration of the **MLS-basic** procedure, if `good` is false, the corresponding step size is reduced to $\alpha_{r+1} = \alpha_r/\gamma_e$; otherwise, if it is true, extrapolation is attempted and the step size is extended to $\alpha_r = \gamma_e\alpha_r$. The question is what is the initial value for α_r. We denote by δ_k the kth step size generated by **VRBBON-basic**, and discuss below how to update it. Let $\delta_{\max} > 0$ be the initial step size, which is a tuning parameter, i.e., $\delta_0 := \delta_{\max}$. The initial step size of the **MLS-basic** procedure is $\alpha_r = \delta_k$ in the kth iteration of **VRBBON-basic**.

We denote by y^t $(r = 1, \cdots, T_0)$ the sequence generated by the **DS-basic** procedure and by x^k $(k = 1, 2, 3, \cdots)$ the sequence generated by **VRBBON-basic**. The tth 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 $n_{\mathtt{succ}}^{DS}$ the number of successful iterations and is called **inefficient** when it terminates with $n_{\mathtt{succ}}^{DS} = 0$. We denote by x_{best} the **overall best point** and by $f(x_{\mathrm{best}})$ 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 ε-approximate stationary point of the sequence x^k $(k = 1, 2, 3, \cdots)$. If $n_{\mathtt{succ}}^{\mathtt{DS}} = 0$ is zero, we say that the kth iteration of **VRBBON-basic** is **unsuccessful** and otherwise **successful**.

VRBBON-basic first chooses the best starting point $z_{\rm best} = z^1 = x^0$ and its inexact function value $\tilde{f}(z_{\rm best}) = \tilde{f}(z^1) = \tilde{f}(x^0)$ before the first call to the **MLS-basic** procedure. Next, it chooses $z_{\rm best} = z^1 = x^{k-1}$ and $\tilde{f}(z_{\rm best}) = \tilde{f}(z^1) = \tilde{f}(x^{k-1})$ 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 $y^t = z_{\rm best}$ and $\tilde{f}(y^t) = \tilde{f}(z_{\rm best})$ when $n^{\texttt{MLS}}_{\texttt{succ}}$ is positive. Otherwise, $y^t = y^{t-1}$ and $\tilde{f}(y^t) = \tilde{f}(y^{t-1})$. When the **DS-basic** procedure terminates after T_0 executions, **VRBBON-basic** selects the kth point $x^k := y^{t'}$ and its inexact function value $\tilde{f}(x^k) := \tilde{f}(y^{t'})$ of the sequence x^k $(k = 1, 2, 3, \cdots)$, where $t' := \underset{t=1:T_0}{\rm argmin}\{\tilde{f}(y^t)\}$. Once the step size δ_k is below the minimum threshold $0 \leq \delta_{\rm min} < 1$, which is a tuning parameter, **VRBBON-basic** terminates with the overall best point $x_{\rm best} := x^k$ and its inexact function value $\tilde{f}(x_{\rm best}) := \tilde{f}(x^k)$ in the iteration k. Otherwise, if $n^{DS}_{\texttt{succ}} = 0$ the step size δ_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 ε-approximate stationary point $x_{\rm best}$ that satisfies $\|g(x_{\rm best})\|_* \leq \varepsilon$ for a threshold $\varepsilon = \sqrt{\omega}$ before the condition $\delta_k \leq \delta_{\rm min}$ is satisfied. When $\delta_{\rm min} = 0$ and the gradient $g(x_{\rm best})$ is unknown, `nfmax` 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,
- $\mathcal{O}(\sqrt{n}\omega^{-1/2})$ function evaluations in the convex case,
- $\mathcal{O}(n \log(\omega^{-1}))$ function evaluations in the strongly convex case

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

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

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 $c_1 \alpha p^T \tilde{g}(x)$ 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 ($r \in \{1, 2, \cdots, R\}$) is found along the search directions $\pm p^r$ (`good` is false when these directions are tried).

Proposition 2 *Let $\{z^r\}$ $(r = 1, 2, \cdots, R)$ 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) $\tilde{f}(z^r + \alpha_r p^r) < \tilde{f}(z^r) - \gamma\alpha_r^2$,
(ii) $\tilde{f}(z^r + \alpha_r p^r) > \tilde{f}(z^r) + \gamma\alpha_r^2$ *and* $\tilde{f}(z^r - \alpha_r p^r) < \tilde{f}(z^r) - \gamma\alpha_r^2$,
(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

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

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

$$\begin{aligned}\tilde{f}(z^r + \alpha_r p^r) - \tilde{f}(z^r) &\leq f(z^r + \alpha_r p^r) - f(z^r) + 2\omega \\ &\leq \alpha_r g(z^r)^T p^r + \frac{1}{2}L\alpha_r^2\|p^r\|^2 + 2\omega \\ &= -\alpha_r|g(z^r)^T p^r| + \frac{1}{2}L\alpha_r^2\|p^r\|^2 + 2\omega < -\gamma\alpha_r^2, \end{aligned} \quad (11)$$

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

$$\begin{aligned}\tilde{f}(z^r - \alpha_r p^r) - \tilde{f}(z^r) &\leq f(z^r - \alpha_r p^r) - f(z^r) + 2\omega \\ &\leq g(z^r)^T(-\alpha_r p^r) + \frac{1}{2}L\alpha_r^2\|p^r\|^2 + 2\omega \\ &= -\alpha_r|g(z^r)^T p^r| + \frac{1}{2}L\alpha_r^2\|p^r\|^2 + 2\omega < -\gamma\alpha_r^2, \end{aligned} \quad (12)$$

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

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

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 $\alpha_{\min} > 0$ 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 Rth 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* `nfmax` *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 $\gamma\alpha_R^2$.*
(ii) Assume that

$$\gamma_e > 1,\ 0 < \eta \le \tfrac{1}{2},\ R := \lceil \log_2 \eta^{-1} \rceil \ge 2,\ \text{and}\ \gamma_e^{1-R}\delta_k \ge \alpha_{\min}. \tag{13}$$

If the **MLS-basic** *procedure has no successful iteration, then at least one $z^{r'}$ $(1 \le r' \le R)$ 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*

$$\|g(z^{r'})\|_* \le \sqrt{cn}\Gamma(\delta_k), \tag{14}$$

where c comes from Proposition 1 and $\Gamma(\delta_k) := (2\gamma + L)\delta_k + 4\gamma_e^{R-1}\omega/\delta_k$. Here δ_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 rth scaled random search direction, by z^r the rth point, and by $\alpha_r = \gamma_e^{1-r}\delta_k \ge \alpha_{\min}$ (by (13)) the rth step size.

(i) The worst case requires $2R + 1$ function evaluations and assumes that the Rth 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 Rth 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 $\gamma\alpha_R^2$-sufficient gain. Consequently, (i) is verified.

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

$$\begin{aligned}\Gamma_0(\alpha_r) &\le \max\{\Gamma_0(\alpha_1), \Gamma_0(\alpha_R)\} < (2\gamma + L)\alpha_1 + 4\omega/\alpha_R \\ &= \Gamma(\delta_k) := (2\gamma + L)\delta_k + 4\gamma_e^{R-1}\omega/\delta_k,\end{aligned} \tag{15}$$

where $\alpha_1 := \max_{r\in\mathcal{R}}\{\alpha_r\} = \delta_k > \alpha_R := \min_{r\in\mathcal{R}}\{\alpha_r\} = \gamma_e^{1-R}\delta_k$ 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

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

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

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

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

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

□

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,* `nfmax` *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 $\{y^t\}$ $(t = 1, 2, \cdots, T_0)$ 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*

$$\overline{\gamma}^{-1}\delta_k^{-2}\Big(f(x^0) - \widehat{f} + 2\omega\Big), \tag{16}$$

where $\overline{\gamma} := \gamma_e^{2(2-R)}\gamma > 0$, $\widehat{f}$ is finite by (A1) and (A2) discussed in Section 1.2, and the step size δ_k is fixed, independent of t, and updated outside the **DS-basic** *procedure. Moreover, the number of function evaluations of the* **DS-basic** *procedure is bounded by*

$$2RT_0 + (2R+1)T_0\overline{\gamma}^{-1}\delta_k^{-2}\Big(f(x^0) - \widehat{f} + 2\omega\Big). \tag{17}$$

(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*

$$\|g(y^{t'})\|_* \leq \sqrt{cn}\Gamma(\delta_k) \tag{18}$$

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 Rth 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 α_R to γ_e along the fixed direction $-p^R$, but at least once α_R is expanded by γ_e in an extrapolation and therefore at most $R-1$ times $\alpha_1 = \delta_k$ is reduced by γ_e if we cannot extrapolate along the other scaled random directions and their opposite directions. Therefore, for each $t \in S$, according to the role of updating α_r in lines 5, 15, and 25 of Algorithm 1,

$$\alpha_R \geq \gamma_e \delta_k / \gamma_e^{R-1} = \gamma_e^{2-R}\delta_k$$

in the $(k+1)$th iteration of **VRBBON-basic**. Put $\overline{\gamma} := \gamma_e^{2(2-R)}\gamma > 0$. 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

$$\tilde{f}(y^{t+1}) - \tilde{f}(y^t) = \tilde{f}(z^R) - \tilde{f}(z_{\text{best}}) \leq -\gamma\alpha_R^2 \leq -\overline{\gamma}\delta_k^2,$$

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

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

Therefore, (16) is valid. The step size δ_k is fixed, independent of t, and updated outside the **DS-basic** procedure. As mentioned earlier, the **MLS-baisc** 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 Rth 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

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

function evaluations.

U denotes the index set of unsuccessful iterations of the **DS-basic** procedure. Since $T_0 = |U| + |S|$ and the **MLS-baisc** 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 $(t = 1, \cdots, T_0)$, resulting in, that for at least one $y^{t'}$ $(1 \leq t' \leq T_0)$ of the evaluated points, with probability $\geq 1 - 2^{-R} \geq 1 - \eta$, $\|g(y^{t'})\|_* \leq \sqrt{cn}\Gamma(\delta_k)$ holds. □

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

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

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 $\{x^k\}$ $(k = 1, 2, \cdots)$ be the sequence generated by* **VRBBON-basic**. *Assume that* (13) *holds, $\delta_{\max} > 0$, $Q > 1$,* `nfmax` *is sufficiently large,*

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

Then

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

and **VRBBON-basic** *terminates after at most*

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

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*

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

where r_0 is given by

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

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

$$f(x^{\ell'}) - \widehat{f} = \frac{\mathcal{O}(n\omega)}{2\sigma}, \quad \textit{and} \quad \|x^{\ell'} - \widehat{x}\| = \frac{\mathcal{O}(\sqrt{n\omega})}{\sigma^2}. \tag{26}$$

Proof (i) Since **VRBBON-basic** has K unsuccessful iterations, from calls to the **DS-basic** procedure, the condition

$$\|g(x^{\ell'})\|_* \leq \sqrt{cn} \min_{\ell=0:K} \Gamma(\delta_\ell), \tag{27}$$

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

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

whose application in (27) leads to (23).
(ii) The convexity of f leads to $\widehat{f} \geq f_\ell + g(x^\ell)^T(\widehat{x} - x^\ell)$ 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 $f_{\ell'} - \widehat{f} \leq g(x^{\ell'})^T(x^{\ell'} - \widehat{x}) \leq \|g(x^{\ell'})\|_* \|x^{\ell'} - \widehat{x}\| = \mathcal{O}(r_0\sqrt{n\omega})$ 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 $y_i = x_i - s_i\sigma^{-1}g_i(x)$ for $i = 1, \cdots, n$, leading to $f(y) \geq f(x) - \dfrac{1}{2\sigma}\|g(x)\|_*^2$. 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 $\geq 1 - \eta$

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

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

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

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) $\mathcal{O}(\omega^{-1})$ function evaluations in the nonconvex case,
(ii) $\mathcal{O}\left(\sqrt{n}\omega^{-1/2}\right)$ function evaluations in the convex case,
(iii) $\mathcal{O}(n \log \omega^{-1})$ 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-basic** 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 $\delta_\ell = Q^{1-\ell}\delta_{\max} \le \delta_{\min}$ for $\ell \ge K$ and

$$\delta_\ell \ge \delta_{\min} \quad \text{for } \ell \in \mathcal{B} := \{1, \cdots, K\}. \tag{28}$$

The condition (28) is used in the proof of (ii) and (iii).
(i) We conclude from (17) and (21)–(22) that

$$\begin{aligned} N_K &\le 1 + \sum_{\ell=1}^{K} \Big(2RT_0 + (2R+1)T_0\overline{\gamma}^{-1}\delta_\ell^{-2}(f(x^0) - \widehat{f} + 2\omega)\Big) \\ &= 1 + 2RT_0K + (2R+1)T_0\overline{\gamma}^{-1}\Big(f(x^0) - \widehat{f} + 2\omega\Big)\sum_{\ell=1}^{K}\delta_\ell^{-2} \\ &= 1 + 2RT_0K + (2R+1)T_0\overline{\gamma}^{-1}\delta_{\max}^{-2}\Big(f(x^0) - \widehat{f} + 2\omega\Big)\sum_{\ell=1}^{K}Q^{2\ell-2} \\ &= 1 + 2RT_0K + (2R+1)T_0\overline{\gamma}^{-1}\delta_{\max}^{-2}\Big(f(x^0) - \widehat{f} + 2\omega\Big)\frac{Q^{2K}-1}{Q^2-1}. \end{aligned}$$

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

$$N_K = \mathcal{O}(R\omega^{-1}) = \mathcal{O}(\omega^{-1}).$$

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

$$f_\ell - f_{\ell+1} \le f_\ell - \widehat{f} \le g(x^\ell)^T(x^\ell - \widehat{x}) \le \|g(x^\ell)\|_*\|x^\ell - \widehat{x}\| \tag{29}$$

$$\le r_0\sqrt{cn}\Big((2\gamma + L)\delta_\ell + 4\gamma_e^{R-1}\omega/\delta_\ell\Big) \quad \text{for } \ell \in \mathcal{B}. \tag{30}$$

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

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

Then we define $\mathcal{B}_1 := \{\ell \in \mathcal{B} \mid (31) \text{ holds}\}$.
Case 2. The second term $4\gamma_e^{R-1}\omega/\delta_\ell$ in (30) dominates the first term. Then we conclude from (28) that

$$f_\ell - f_{\ell+1} = \mathcal{O}(\sqrt{n}(\omega/\delta_\ell)) = \mathcal{O}(\sqrt{n}(\omega/\delta_{\min})) = \mathcal{O}(\sqrt{n\omega}) \quad \text{for } \ell \in \mathcal{B}. \tag{32}$$

Then we define $\mathcal{B}_2 = \{\ell \in \mathcal{B} \mid (32) \text{ holds}\}$.
Then we conclude from (21), (20), and (22) that with probability $\geq 1-\eta$

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

so that by (i) and (13)

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

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 $i = 1, \cdots, n$, resulting in $f(y) \geq f(x) - \frac{1}{2\sigma}\|g(x)\|_*^2$. Here as mentioned earlier $s \in \mathbb{R}^n$ is a scaling vector. By applying (23) in this inequality, we get with probability $\geq 1-\eta$

$$f_\ell - f_{\ell+1} \leq f_\ell - \widehat{f} \leq \frac{1}{2\sigma}\|g(x^\ell)\|_*^2 \leq \frac{cn}{2\sigma}\Big((2\gamma+L)\delta_\ell + 4\gamma_e^{R-1}\omega/\delta_\ell\Big)^2 \quad \text{for } \ell \in \mathcal{B}. \tag{33}$$

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

$$f_\ell - f_{\ell+1} = \mathcal{O}(n\delta_\ell^2) \quad \text{for } \ell \in \mathcal{B} \tag{34}$$

and denote $\mathcal{B}_1 := \{\ell \in \mathcal{B} \mid (34) \text{ holds}\}$.
Case 2. The second term $4\gamma_e^{R-1}\omega/\delta_\ell$ in (33) dominates the first term. Then we conclude from (28) that

$$f_\ell - f_{\ell+1} = \mathcal{O}\Big(n(\omega/\delta_\ell)^2\Big) = \mathcal{O}\Big(n(\omega/\delta_{\min})^2\Big) = \mathcal{O}(n\omega) \quad \text{for } \ell \in \mathcal{B} \tag{35}$$

and denote $\mathcal{B}_2 = \{\ell \in \mathcal{B} \mid (35) \text{ holds}\}$.
Then we conclude from (20)–(22) that with probability $\geq 1 - \eta$

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

so that by (i) and (13)

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

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

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 $\gamma_{\mathrm{rd}}/\|p\|$ with $0 < \gamma_{\mathrm{rd}} < 1$, 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 α_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 $m_{\max} := \min\left\{\overline{m}, \frac{1}{2}n(n+3)\right\}$, where $\overline{m}$ is a tuning parameter. The **number of sample points** is denoted by $m \in [2, m_{\max}]$ and the **subspace size** m^{o} is defined as the largest integer satisfying $\frac{1}{2}m^{\mathrm{o}}(m^{\mathrm{o}}+3) \leq m$.

If m is greater than $m_{\max}$, 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 $A_{II} := (A_{ij})_{i\in I, j\in I}$ for the submatrix of A with row and column indices of I by A_{II}, $A_{:k}$ for the kth 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 $\alpha^{\rm rs}$ and then scale it with $\alpha^{\rm rs} := \alpha^{\rm rs}/\|\alpha^{\rm rs}\|$. Then we compute the **random subspace direction** by $p := \sum_{i=1, i\neq b}^{m} \alpha_i^{\rm rs}(Z_{:i} - Z_{:b})$.

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 $\mathcal{O}(N^3)$ 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 $\overline{s}_i := z_i - Z_{:b}$. 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

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

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 $m_{\max} \times m^{\rm o}$ matrix $Z^{\rm o}$ and the $m^{\rm o} \times 1$ vector $z^{\rm o}$, respectively. In other words, the restriction of the entries of Z and z^m is done by choosing a random subset of size $m^{\rm o}$.

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 $Z^{\rm o}$ ($z^{\rm o}$) from a random subset of the entries of Z ($z_{\rm best}$).

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

$$\varepsilon_i := \frac{\tilde{f}_i - F_b - (\tilde{g}^{\rm o})^T\overline{s}_i - \frac{1}{2}\overline{s}_i^T\tilde{B}^{\rm o}\overline{s}_i}{\mathtt{sc}_i} \quad \text{for all } i = 1, \cdots, K, \tag{37}$$

where sc 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 $\mathcal{O}(\|\overline{s}_i\|^3)$ if $m = N$; otherwise it is $\mathcal{O}(\|\overline{s}_i\|^2)$. To obtain ε_i of a uniform magnitude, we choose sc as follows:
(1) We form the matrix $S := (\,\overline{s}_1, \cdots, \overline{s}_K\,)^T$, where $\overline{s}_i := z_i^{\rm o} - Z_{:b}^{\rm o}$ for all $i = 1, \cdots, K$.
(2) We compute the matrix

$$H^{\rm o} := \Big(\sum_l \overline{s}_l \overline{s}_l^T \Big)^{-1} \tag{38}$$

by constructing a reduced QR factorization $S = QR$, where $Q \in \mathbb{R}^{K \times m^{\rm o}}$ is an orthogonal matrix and $R \in \mathbb{R}^{m^{\rm o} \times m^{\rm o}}$ is a square upper triangular matrix.
(3) We compute the scaling vector sc

$$\mathtt{sc}_i := (\overline{s}_i^T H^{\rm o} \overline{s}_i)^{e/2} \quad \text{for } i = 1, \cdots, K, \tag{39}$$

with $\overline{s}_i^T H^{\rm o} \overline{s}_i = \|R^{-T} \overline{s}_i\|^2$ for $i = 1, \cdots, K$. In (39), $e = 3$ if $m = N$ holds (full quadratic model) and $e = 2$ otherwise (reduced quadratic model).

We calculate $H^{\rm o}$ and sc in the same way as **SNOBFIT** [24] by performing a reduced QR factorization, except that they are performed in adaptively determined subspaces.

We have $\varepsilon = Ay - a$, where, for $i = 1, \cdots, K$,

$$a_i := \frac{F_b - \tilde{f}_i}{\mathtt{sc}_i} \quad \text{and} \quad A_{ij} := \begin{cases} \dfrac{\overline{s}_i^j}{\mathtt{sc}_i} & \text{if } j \in \{1, \cdots, m^{\rm o}\}, \\ \dfrac{(\overline{s}_i^{j-m^{\rm o}})^2}{2\mathtt{sc}_i} & \text{if } j \in \{m^{\rm o}+1, \cdots, 2m^{\rm o}\}, \\ \dfrac{\overline{s}_i^{j'} \overline{s}_i^{j''}}{\mathtt{sc}_i} & \text{if } j \in \{2m^{\rm o}+1, \cdots, M\}. \end{cases} \tag{40}$$

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

$$\min_{y \in \mathbb{R}^M} \|Ay - a\|_2^2. \tag{41}$$

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

We construct the vector a and the matrix A by (40), adjust a, and find all multipliers by solving (41). Then we adjust y and define $\tilde{g}^{\rm o}$ by the first $m^{\rm o}$ components of y, the

diagonal entries of $\tilde{B}^{\mathrm{o}}$ by the next m^{o} components of y, and the off-diagonal entries of $\tilde{B}^{\mathrm{o}}$ 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 $m_{\max}$. 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 `sc` and adjust it. Third, we calculate y and adjust it. Finally, we estimate $\tilde{g}^{\mathrm{o}}$ and $\tilde{B}^{\mathrm{o}}$.

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 $\tilde{B}^{\mathrm{o}}$ is not computable, which implies that at least one entry of $\tilde{B}^{\mathrm{o}}$ 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 $\tilde{p} := p^{\mathrm{o}} - \alpha^{\mathrm{o}}\tilde{g}^{\mathrm{o}}$ is a perturbation of a standard random direction p^{o} by the steepest descent direction $-\tilde{g}^{\mathrm{o}}$ with the approximate gradient. Both $\tilde{g}^{\mathrm{o}}$ and p^{o} are restricted to a subspace which is a random subset $\mathcal{J}$ of $\{1, \cdots, n\}$ 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 $\alpha^{\mathrm{o}} := (1 + \kappa(\tilde{g}^{\mathrm{o}})^T p^{\mathrm{o}})/\|\tilde{g}^{\mathrm{o}}\|^2$ and the decreasing sequence $\kappa := 1/(1+n_{\mathrm{func}})^{\gamma_\kappa}$ with the tuning parameter $0 < \gamma_\kappa < 1$ and the number of function evaluations n_{func}. It is easy to show that $\tilde{p}^T\tilde{g} = (\kappa p^{\mathrm{o}} - \alpha^{\mathrm{o}}\tilde{g}^{\mathrm{o}})^T\tilde{g}^{\mathrm{o}} < 0$; 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

$$\begin{aligned} &\min\ \zeta^T\tilde{g}^{\mathrm{o}} + \frac{1}{2}\zeta^T\tilde{B}^{\mathrm{o}}\zeta \\ &\text{s.t.}\quad \|\zeta - z^{\mathrm{o}}\|_\infty \le d, \end{aligned}$$

whose solution is denoted by ζ_{best}. Here d is denoted as the **trust region radius**. The trust region direction is $p^{\mathrm{o}}_{\mathrm{tr}} := \zeta_{\mathrm{best}} - z^{\mathrm{o}}$ in a subspace which is a random subset of $\{1, \cdots, n\}$ whose size is m^{o}. The idea is to construct the **improved trust region direction** by scaling $p^{\mathrm{o}}_{\mathrm{tr}}$ with the positive tuning parameter γ_p and perturbing it by

$p_{\rm mean} := z_{\rm mean} - Z_{:b}$, where $z_{\rm mean}$ is the mean of Z. This perturbation transforms $p^{\rm o}_{\rm tr}$ into a full subspace direction enriched by $p_{\rm mean}$.

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 $0 \leq \underline{\alpha}_{\rm init} < \overline{\alpha}_{\rm init} \leq \infty$ be the lower and upper bounds of an initial interval as the tuning parameters. We start with the initial interval $[\underline{\alpha}, \overline{\alpha}] = [\underline{\alpha}_{\rm init}, \overline{\alpha}_{\rm init}]$. Then $\underline{\alpha}$ or $\overline{\alpha}$ is updated. We describe this below.

• **The initial interval.** When the trial point $z^r = z_{\rm best} \pm \alpha_r p^r$ and its inexact function value $\tilde{f}(z^r)$ are computed in line 10 of **VRBBON-basic**, the gain $\tilde{f}(z_{\rm best}) - \tilde{f}(z^r)$ is stored in the vector $\mathtt{dF}$ and its step size α_r in the vector $\mathbf{a}$. Namely, when an extrapolation with many sufficient gains along p^r ($r \in \{1, \cdots, R\}$) is found, much information is stored in the vectors $\mathtt{dF}$ and $\mathbf{a}$. Then, an index set of the stored points with the decreasing inexact function values is found by $\mathtt{ind} := \{i \mid \mathtt{dF}_i < 0 \ \text{ for } i = 1, \cdots, n\}$. If $\mathtt{ind}$ is non-empty, the lower bound of the interval $[\underline{\alpha}, \overline{\alpha}]$ can be found by $\underline{\alpha} := \max(\mathbf{a}_{\mathtt{ind}})$; if $\underline{\alpha} = \underline{\alpha}_{\rm init}$ is given as a positive tuning parameter, then $\underline{\alpha}$ is updated by $\underline{\alpha} := \min(\underline{\alpha}, \max(\mathbf{a}_{\mathtt{ind}}))$. Next, an index set of positive gains or the corresponding step sizes strictly larger than $\overline{\alpha}$ is found by

$$\overline{\mathtt{ind}} := \{i \mid \mathtt{dF}_i \geq 0 \text{ or } \mathbf{a}_i > \overline{\alpha} \ \text{ for } i = 1, \cdots, n\}.$$

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

• **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 $\alpha_1 := \max\{\sqrt{\underline{\alpha}\overline{\alpha}}, \delta_{k-1}\}$ whenever the interval $[\underline{\alpha}, \overline{\alpha}] \subseteq (0, \infty)$ 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

$$\alpha_{r+1} := \max\left\{\overline{\alpha}_{\min}, \min\left\{\sqrt{\underline{\alpha}\overline{\alpha}}, \frac{\alpha_r}{\gamma_e}\right\}\right\}$$

with the goal of slowly reducing the step sizes. Here $\overline{\alpha}_{\min} \in (0,1)$ is a minimum threshold for the step size which is a tuning parameter.

• **Updating the interval.** After finding the new step size α_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 $\pm p^r$ (`good` is true).
(ii) After no $\gamma\alpha_r^2$-sufficient gain along $\pm p^r$ has been found (`good` is false).
To do this, we insert the following statement between lines 26 and 27 of **VRBBON-basic**:

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

• **Decrease of f in flat regions.** If the rth iteration of the **MLS** procedure is unsuccessful (no $\gamma\alpha_r^2$-sufficient gain along $\pm p^r$ is found), the best point, its inexact function value, and its step size are updated when $\tilde{f}(z_{\text{best}} + \alpha_r p^r) < \tilde{f}(z_{\text{best}})$. 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** $\tilde{f}(z^r) < \tilde{f}(z_{\text{best}})$ **then**, set $z_{\text{best}} := z^r$ and $\tilde{f}(z_{\text{best}}) := \tilde{f}_e^r$; $n_{\texttt{succ}}^{\texttt{MLS}} := n_{\texttt{succ}}^{\texttt{MLS}} + 1$;
2: **end if**

• **Updating the best point.** If the rth iteration of the **MLS** procedure is successful ($\gamma\alpha_r^2$-sufficient gains are found along one of $\pm p^r$). 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 $i' := \min\limits_{i \geq 0} \left\{ \tilde{f}(z_{\text{best}} + \alpha_r \gamma_e^i p^r) \;\middle|\; \tilde{f}(z_{\text{best}} + \alpha_r \gamma_e^i p^r) < \tilde{f}(z_{\text{best}}) \right\}$;
2: compute $\alpha_{r+1} = \gamma_e^{i'} \alpha_r$, $z_{\text{best}} := z_{\text{best}} + \alpha_{r+1} p^r$, and $\tilde{f}(z_{\text{best}}) := \tilde{f}(z_{\text{best}} + \alpha_{r+1} p^r)$;
3: update $n_{\texttt{succ}}^{\texttt{MLS}} := n_{\texttt{succ}}^{\texttt{MLS}} + 1$;

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 $\gamma_{d_1} > 1$ and $\gamma_{d_2} \in (0,1)$ 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 ($n_{\mathtt{succ}}^{DS} = 0$), the lower bound $\underline{\alpha}$ of the current interval $[\underline{\alpha}, \overline{\alpha}]$ 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**

for $t = 1, \cdots, T_0$ **do**
  perform **MLS** with either R **scaled random directions**, C **random**
  approximate coordinate directions, or both;
  if $m \geq 3$ **then**
    while true **do** ▷ until `good` is true
      perform **MLS** with **random subspace directions**;
      if `good` is false, **break**; **end** ▷ **MLS** is inefficient
    end while
  end if
  if $m \geq 2$ **then** ▷ either a reduced quadratic or a linear model is constructed
    if both estimations are computable **then**
      construct a reduced quadratic model;
      generate the trust region radius by $d_t := \|z_{\text{mean}} - Z_{:b}\|$;
      restrict the trust region radius by $d_t := \max(d_{\min}, \min(d_{\max}, \gamma_{d_1} d_t))$;
      minimize the model to get an improved trust region direction;
      while true **do** ▷ until `good` is true
        perform **MLS** with **improved trust region directions**;
        if `good` is false, **break**; **end** ▷ **MLS** is inefficient
        update $d_t := (\gamma_{d_2} + \text{rand}) d_t$; ▷ rand $\in (0, 1]$ is a random value
      end while
    else
      while true **do** ▷ until `good` is true
        perform **MLS** with **perturbed random directions**;
        if `good` is false, **break**; **end** ▷ **MLS** is inefficient
      end while
    end if
  end if
  if $n_{\mathtt{succ}}^{\mathtt{DS}}$ is zero **then** ▷ **MLS** is inefficient in all T_0 calls
    for $i = 1, \cdots, n$ **do**
      compute $\mathtt{dz}_i := Z_{:i} - Z_{:b}$; ▷ the difference of the old good points
      find $\mathcal{I}_i := \{j \mid \mathtt{dz}_j \neq 0 \text{ and } (Z_{:b})_j \neq 0\}$;
      compute $\beta_i^t := \min\{|(Z_{:b})_j / \mathtt{dz}_j| \mid j \in \mathcal{I}_i\}$;
    end for
    generate two random values μ_1 and μ_2 satisfying $0 < \mu_1 < \mu_2 < 1$;
    reconstruct $[\underline{\alpha}, \overline{\alpha}] := [\gamma_a \mu_1 \beta_{\min}^t, \gamma_a \mu_2 \beta_{\min}^t]$ with $\beta_{\min}^t := \min_{i=1,\cdots,n} \beta_i^t$;
  end if
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 $\varepsilon = \sqrt{\omega}$-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 $\delta_k := \max(\delta_{k-1}, \sqrt{\underline{\alpha}\overline{\alpha}})$ if the interval $[\underline{\alpha}, \overline{\alpha}] \subseteq (0, \infty)$ 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 `minq8` [25], available at

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

`minq8` requires the tuning parameters `minqmax` $= 10000$ and `minqeps` $= 10^{-8}$.

The values of tuning parameters of **VRBBON** are chosen in Table 1, except the other five tuning parameters `comBound` (are the complexity results valid?), `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**). `model` is a Boolean variable taking 0 (model-free) and 1 (model-based).

Table 1: The values of some tuning parameters of **VRBBON**, except `comBound`, `model`, T_0, and R

$\overline{m} = 230$	$m_{\max} = \min(0.5n(n+3), \overline{m})$	$\delta_{\min} = 0$	$\delta_{\max} = 1$	$\gamma_Z = 100$	$\gamma_r = 10^{-30}$
$\gamma_{d_1} = 2$	$Q = 1.5$	$d_{\min} = 10^{-4}$	$\gamma_v = 100$	$\gamma = 10^{-6}$	$\gamma_e = 3$
$d_{\max} = 10^3$	$\underline{\alpha}_{\text{init}} = 0.01$	$\overline{\alpha}_{\text{init}} = 0.99$	$\gamma_\kappa = 0.85$	$\gamma_{d_2} = 0.5$	$\gamma_p = 0.25$
$\gamma_a := 10^{-5}$	$\overline{\alpha}_{\min} = 10^{-3} * \text{rand}$				

In the following three cases we discuss the conditions under which complexity results can be found for **VRBBON**:
Case 1. `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. `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. `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

$$(2R+4)T_0 + (2R+5)T_0\overline{\gamma}^{-1}\delta_k^{-2}\Big(f(x^0) - \widehat{f} + 2\omega\Big)$$

in Case 1 and

$$(2R+2C+4)T_0 + (2R+2C+5)T_0\overline{\gamma}^{-1}\delta_k^{-2}\Big(f(x^0) - \widehat{f} + 2\omega\Big)$$

in Case 2. As mentioned earlier, the step size δ_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 $\tilde{f} = f + (2 * \text{rand} - 1)\omega$, 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 Kimiaei & 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/software/GRID`.

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:

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

• **SDBOX** – a derivative-free algorithm for bound constrained optimization problems discussed in [30], downloaded from

`http://www.iasi.cnr.it/~liuzzi/DFL/index.php/list3`.

• **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

`oCMAES.MaxFunEvals` = `nfmax`, `oCMAES.DispFinal` = `0`, `oCMAES.DispModulo` = `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 $S := 2 * \mathrm{rand}(n, m) - 1$ (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 $\tilde{f}(x + \Delta Sz)$ with respect to $z \in \mathbb{R}^n$. Let Δ be the trust region radius, which is initially one, and let $0 < \overline{\gamma} < 1$ be a parameter for the decrease of the function value. If $\tilde{f}(x) > \tilde{f}(x + \Delta Sz) - \overline{\gamma}\Delta^2$ holds, the trial point $x + \Delta Sz$ can be accepted as a new point and Δ is expanded to $\Delta = \max(\Delta, \|Sz\|)$; otherwise Δ is reduced to $\Delta = \frac{1}{2}\min(\Delta, \|Sz\|)$. For each call to these solvers, the maximum number of function evaluations is $\max(\lceil n/m \rceil, 10m)$. According to our findings, the best value for the subspace dimesion 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 Moré & Wild [32] and the performance profile of Dolan & Moré [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 κ groups of n_p+1 function evaluations is the data profile of the solver s, i.e.,

$$\delta_s(\kappa) := \frac{1}{|\mathcal{P}|}\left|\left\{p \in \mathcal{P} \;\middle|\; cr_{p,s} := \frac{c_{p,s}}{n_p+1} \leq \kappa\right\}\right|. \tag{42}$$

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 τ is the performance profile of the solver s, i.e.,

$$\rho_s(\tau) := \frac{1}{|\mathcal{P}|}\left|\left\{p \in \mathcal{P} \;\middle|\; pr_{p,s} := \frac{c_{p,s}}{\min(c_{p,\bar{s}} \mid \bar{s} \in S)} \leq \tau\right\}\right|. \tag{43}$$

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 τ (κ) that the solver s can solve. **The data and performance profiles are based on the problem scales, but not on the noise levels. 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

$$q_s := (f_s - f_{\text{opt}})/(f_0 - f_{\text{opt}}) \quad \text{for } s \in \mathcal{S} \tag{44}$$

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. Here

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

• f_{opt} is the function value at the best known point (in most cases a global minimizer or at least a better local minimizer) found by running a sequence of gradient-based and local/global gradient free solvers; see Appendix B in [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 `nfmax` of function evaluations nor the maximum allowed time `secmax` in seconds is satisfied, and **unsolved** otherwise. ε, `secmax` and `nfmax` 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, ε is chosen slightly larger for problems in medium and high dimensions than for problems in low dimensions. The following choices were found valuable:

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

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

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**

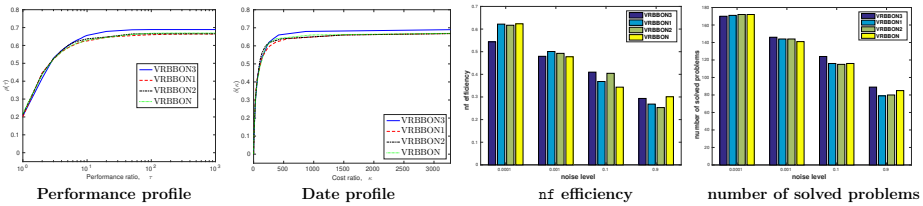
Five important tuning parameters are `comBound` (are the complexity results valid?), `model` (model-based?), R (number of scaled random directions in **MLS**), C (number of random approximate coordinate directions), and T_0 (number of times **MLS** is called by **DS**). Other tuning parameters were chosen 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].

C	T_0	R	`comBound`	`model`	complexity	chosen as solver
n	n	–	2	0	no	no
n	n	–	2	1	no	no
n	10	–	2	0	no	no
n	10	–	2	1	no	no
n	5	–	2	0	no	no
n	5	–	2	1	no	no
n	2	–	2	0	no	no
n	2	–	2	1	no	**VRBBON1**
10	n	–	2	0	no	no
10	n	–	2	1	no	no
5	n	–	2	0	no	**VRBBON2**
5	n	–	2	1	no	no
2	n	–	2	0	no	no
2	n	–	2	1	no	no
–	n	n	0	1	yes	no
$\lceil n/2 \rceil$	n	$\lceil n/2 \rceil$	1	1	yes	**VRBBON3**
–	n	n	2	0	yes	no
–	n	n	2	1	yes	no
–	10	n	2	0	yes	no
–	10	n	2	1	yes	no
–	5	n	2	0	yes	no
–	5	n	2	1	yes	**VRBBON**
–	2	n	2	0	yes	no
–	2	n	2	1	yes	no
–	n	10	2	0	yes	no
–	n	10	2	1	yes	no
–	n	5	2	0	yes	no
–	n	5	2	1	yes	no
–	n	2	2	0	yes	no
–	n	2	2	1	yes	no

Fig. 1: 30 variants of tuning of **VRBBON**

For the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-1}, 0.9\}$ 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 `nf` efficiency versus the noise level ω and the number of solved problems versus the noise level ω). 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.

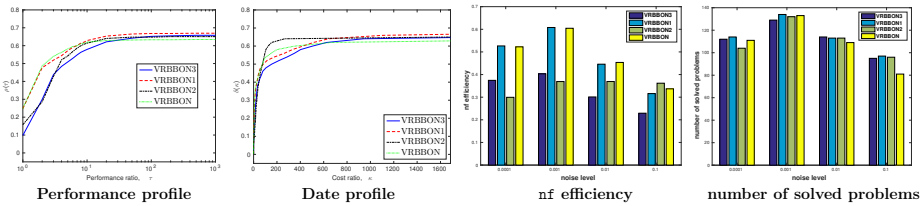
For the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and medium scale problems ($30 < n \leq 300$), Figure 3 shows in its subfigures the cumulative (over all noise levels used) performance and data profiles in terms of the number of function evaluations and the other two plots (the `nf` efficiency versus the noise level ω and the number of solved problems versus the noise level ω). We conclude from these subfigures that **VRB-**



Performance profile | Date profile | nf efficiency | number of solved problems

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 κ on the cost ratio, see (42) while performance profile $\rho(\tau)$ in dependence of a bound τ 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**.

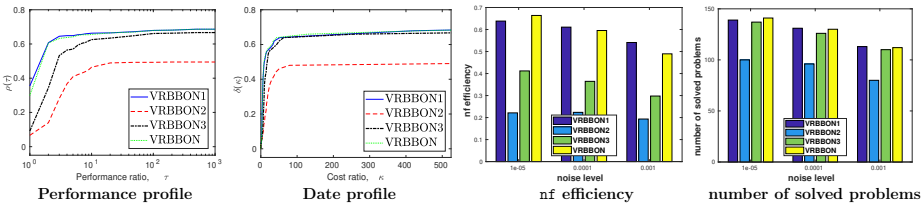


Performance profile | Date profile | nf efficiency | number of solved problems

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 ω and the number of solved problems versus the noise level ω). 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.



Performance profile | Date profile | nf efficiency | number of solved problems

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 ω and the number of solved problems versus the noise level ω). 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**.

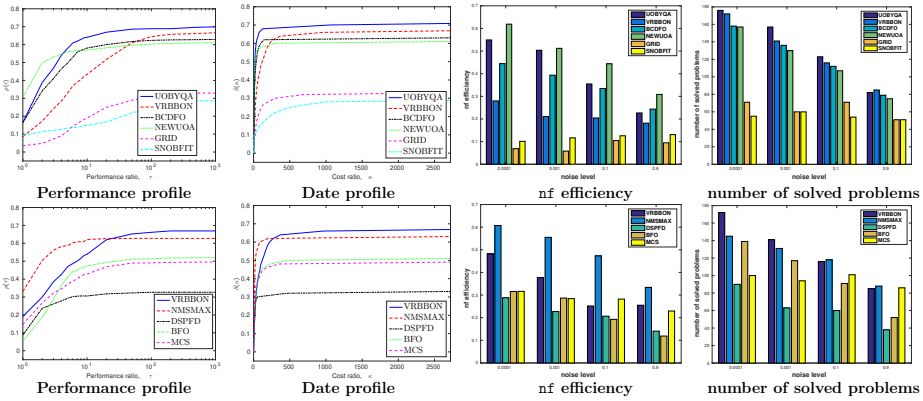


Fig. 5: Comparison between **VRBBON** and model-based solvers (first row) and direct search solvers (second row) for the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-1}, 0.9\}$ and small dimensions $1 < n \leq 30$. Other details as in Figure 2.

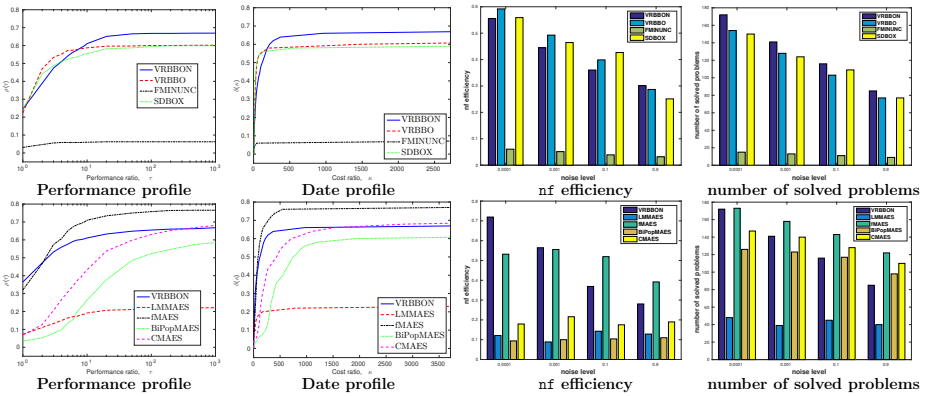


Fig. 6: Comparison between **VRBBON** and line search solvers (first row) and matrix adaptation evolution solvers solvers (second row). Details as in Figure 2.

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 ω and the number of solved problems versus the noise level ω). 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.

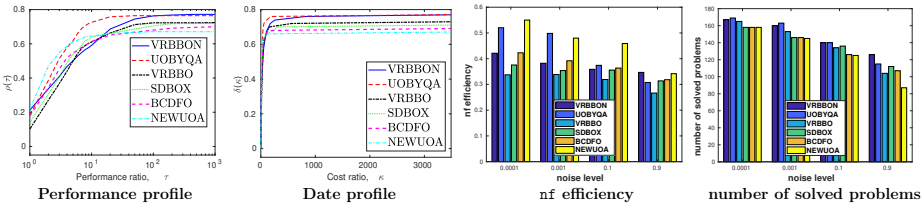


Fig. 7: For the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-1}, 0.9\}$ and small dimensions $1 < n \leq 30$. The standard initial points are used. Details as in Figure 2.

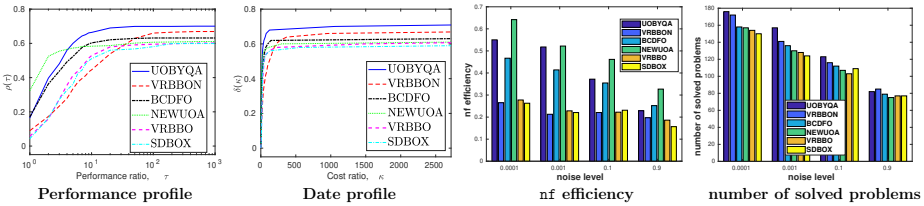


Fig. 8: For the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-1}, 0.9\}$ and small dimensions $1 < n \leq 30$. The shifted initial points are used. Details as in Figure 2.

Figures 9 and 10 show in their 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 ω and the number of solved problems versus the noise level ω). 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.

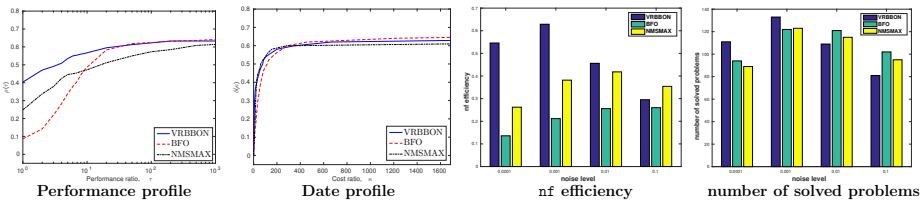


Fig. 9: Comparison between **VRBBON** and direct search solvers for the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and medium dimensions $30 < n \leq 300$. Other details as in Figure 2.

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

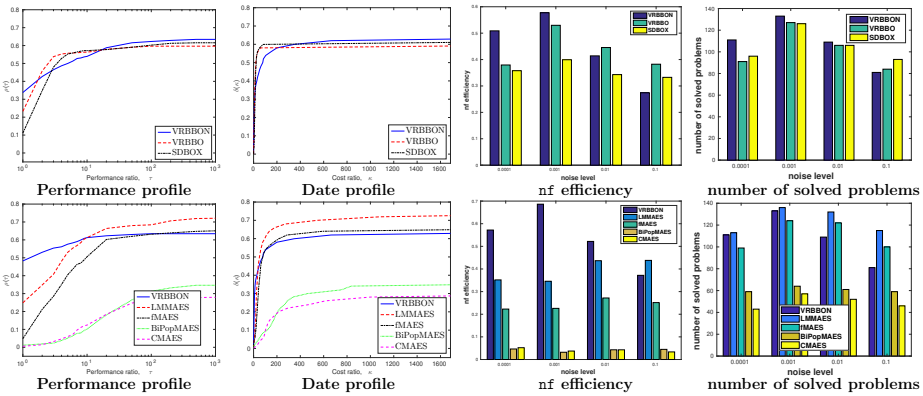


Fig. 10: Comparison between **VRBBON** and line search solvers (first row) and matrix adaptation evolution solvers solvers (second row) for the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and medium dimensions $30 < n \leq 300$. Other details as in Figure 2.

noise, while **subNEWUOA**, **subUOBYQA**, and **NMSMAX** are slightly more robust than **VRBBON** only at high noise.

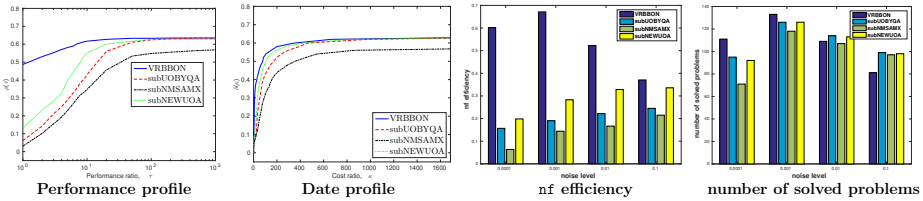


Fig. 11: Comparison between **VRBBON** and model-based solvers in a random subspace for the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and medium dimensions $30 < n \leq 300$. Other details as in Figure 2.

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 ω and the number of solved problems versus the noise level ω). We conclude from these subfigures that **VRBBON** is slightly more efficient than others, while **VRBBO** and **SDBOX** are slightly more robust than **VRBBON**.

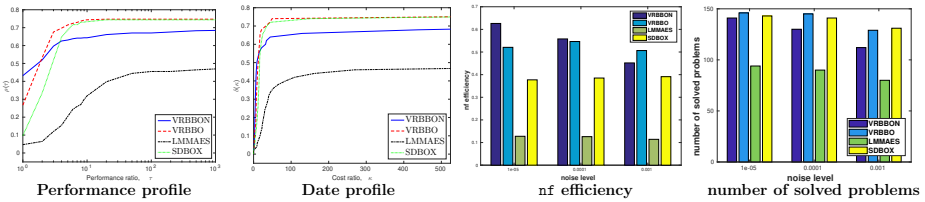


Fig. 12: Comparison between **VRBBON** and the effective solvers on the medium scale problems for the noise levels $\omega \in \{10^{-5}, 10^{-4}, 10^{-3}\}$ and large dimensions $300 < n \leq 5000$. Details as in Figure 2.

8.6 Comparison on a real-life test problem

Consider the driven cavity problem

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

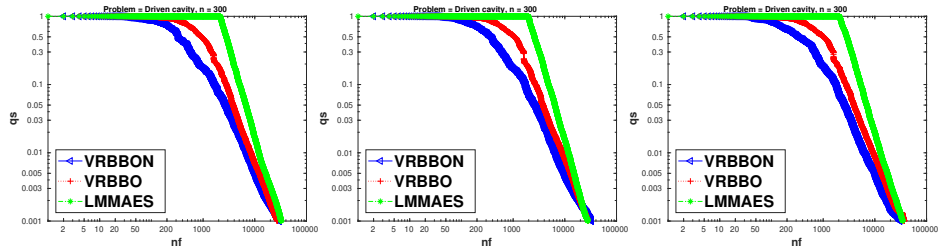
from the collection Lukšan & Vlcek [31], where the equation $f(x) = 0$ is a finite difference analogue of the nonlinear partial differential equation

$$\Delta\Delta u + \theta\Big(\frac{\partial u}{\partial y}\frac{\partial \Delta u}{\partial x} - \frac{\partial u}{\partial x}\frac{\partial \Delta u}{\partial y}\Big) = 0, \quad \theta = 500,$$

over the unit square Ω with the boundary conditions $u = 0$ on $\partial\Omega$ and

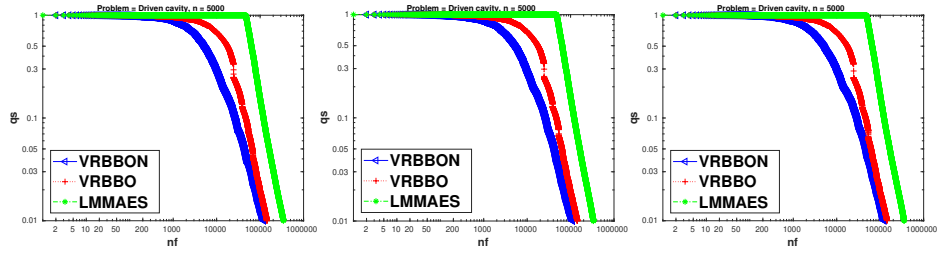
$$\frac{\partial u(0,y)}{\partial x} = 0, \ \frac{\partial u(1,y)}{\partial x} = 0, \ \frac{\partial u(x,0)}{\partial y} = 0, \ \frac{\partial u(x,1)}{\partial y} = 1.$$

We chosen 13 standard finite difference points on a shifted uniform grid with 50×50 internal nodes. The initial approximate solution is a discretization of $u_0(x, y) = 0$. We chosen the initial point as $x_0 = \mathrm{rand}(n, 1) - 0.5$. We ran **VRBBON**, **VRBBO** and **LMMMAES** on this problem with the dimension $n \in \{300, 5000\}$, the budgets `nfmax` $= 200n$ and `secmax` $= \infty$, 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 $f_{\mathrm{best}} \approx 0.01$ in (44) is known to us. In fact, it was obtained by running a sequence of solvers.



$\omega = 0.0001$ $\omega = 0.001$ $\omega = 0.01$

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



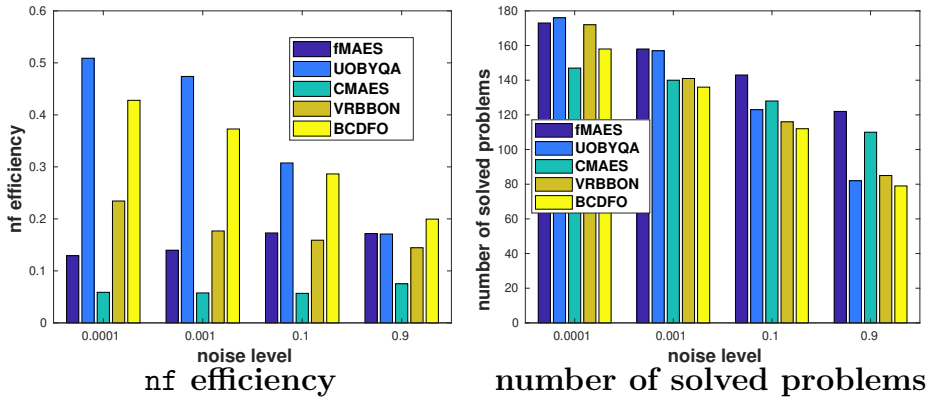
$\omega = 0.0001$ $\omega = 0.001$ $\omega = 0.01$

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

8.7 Recommendation

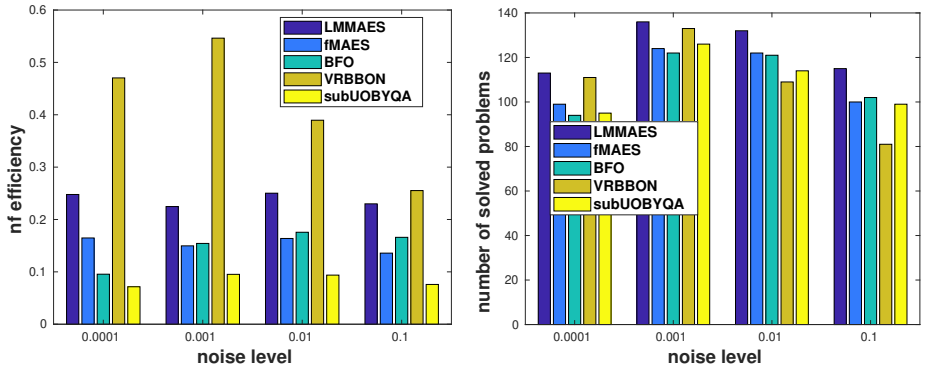
Using Figures 12, 15, and 16, we have recommended which solvers are robust and which are efficient, depending on the noise level and dimension. Indeed, for small scale problems, Figure 15 is a comparison between five more robust and efficient solvers, and for medium scale problems, Figure 16 is a comparison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



nf efficiency number of solved problems

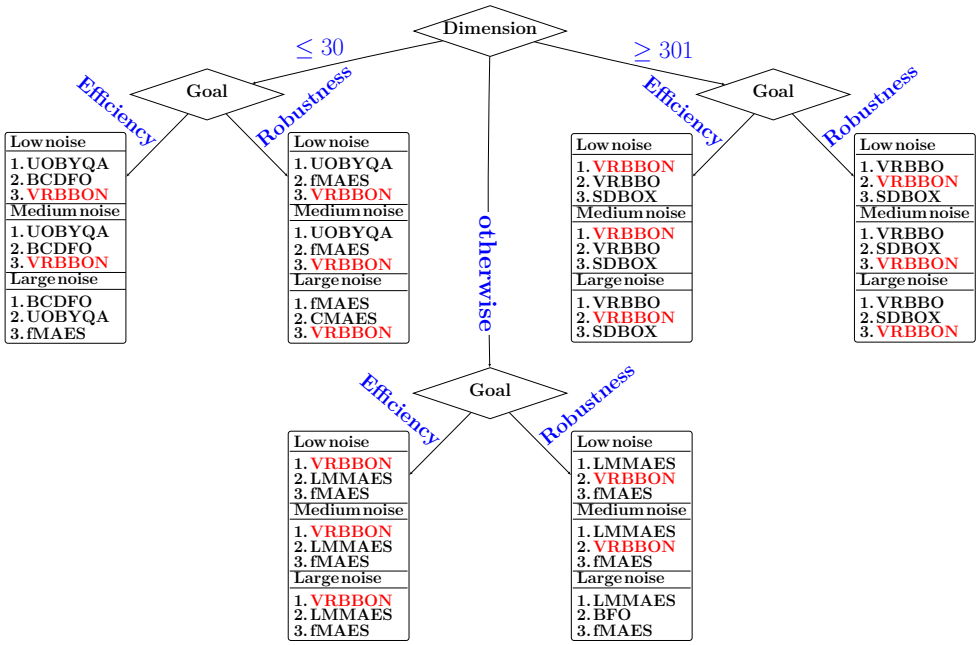
Fig. 15: Comparison between five more robust and efficient solvers for the noise levels $\omega \in \{10^{-4}, 10^{-3}, 10^{-1}, 0.9\}$ and small dimensions $1 < n \leq 30$. Other details as in Figure 2.



nf efficiency number of solved problems

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

1. C. Audet and J. E. Dennis. Mesh adaptive direct search algorithms for constrained optimization. *SIAM J. Optim.* **17** (January 2006), 188–217.
2. A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In *2005 IEEE Congress on Evolutionary Computation*. IEEE (2005).
3. A. S. Bandeira, K. Scheinberg, and L. N. Vicente. Convergence of trust-region methods based on probabilistic models. *SIAM J. Optim* **24** (January 2014), 1238–1264.
4. A. S. Berahas, R. H. Byrd, and J. Nocedal. Derivative-free optimization of noisy functions via quasi-newton methods. *SIAM J. Optim.* **29** (January 2019), 965–993.
5. A. S. Berahas, L. Cao, and K. Scheinberg. Global convergence rate analysis of a generic line search algorithm with noise (2021).
6. H. G. Beyer. Design principles for matrix adaptation evolution strategies (2020).
7. H. G. Beyer and B. Sendhoff. Simplify your covariance matrix adaptation evolution strategy. *IEEE Trans. Evol. Comput.* **21** (October 2017), 746–759.
8. M. D. Buhmann. Radial basis functions. *Acta Numer.* **9** (January 2000), 1–38.
9. R. Chen. *Stochastic Derivative-Free Optimization of Noisy Functions*. PhD thesis, Lehigh University (2015). Theses and Dissertations. 2548.
10. A. R. Conn and Ph. L. Toint. An algorithm using quadratic interpolation for unconstrained derivative free optimization. In *Nonlinear Optimization and Applications*, pp. 27–47. Springer US (1996).
11. C. Davis. Theory of positive linear dependence. *Amer. J. Math.* **76** (October 1954), 733.
12. P. Deuflhard and G. Heindl. Affine invariant convergence theorems for newton's method and extensions to related methods. *SIAM J. Numer. Anal.* **16** (February 1979), 1–10.
13. M. A. Diniz-Ehrhardt, J.M. Martínez, and M. Raydan. A derivative-free nonmonotone line-search technique for unconstrained optimization. *J. Comput. Appl. Math.* **219** (October 2008), 383–397.
14. E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.* **91** (January 2002), 201–213.
15. C. Elster and A. Neumaier. A grid algorithm for bound constrained optimization of noisy functions. *IMA J. Numer. Anal.* **15** (1995), 585–608.
16. N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Comput. Optim. Appl.* **60** (2015), 545–557.
17. S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang. Direct search based on probabilistic descent. *SIAM J. Optim* **25** (January 2015), 1515–1541.
18. S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang. Complexity and global rates of trust-region methods based on probabilistic models. *IMA J. Numer. Anal.* **38** (August 2017), 1579–1597.
19. S. Gratton, Ph. L. Toint, and A. Tröltzsch. An active-set trust-region method for derivative-free nonlinear bound-constrained optimization. *Optim. Methods Softw.* **26** (October 2011), 873–894.
20. L. Grippo and F. Rinaldi. A class of derivative-free nonmonotone optimization algorithms employing coordinate rotations and gradient approximations. *Comput. Optim. Appl.* **60** (June 2014), 1–33.
21. L. Grippo and M. Sciandrone. Nonmonotone derivative-free methods for nonlinear equations. *Comput. Optim. Appl.* **37** (March 2007), 297–328.
22. N. J. Higham. Optimization by direct search in matrix computations. *SIAM J. Matrix Anal. Appl.* **14** (April 1993), 317–333.
23. W. Huyer and A. Neumaier. Global optimization by multilevel coordinate search. *J. Glob. Optim.* **14** (1999), 331–355.
24. W. Huyer and A. Neumaier. SNOBFIT – stable noisy optimization by branch and fit. *ACM. Trans. Math. Softw.* **35** (July 2008), 1–25.

25. W. Huyer and A. Neumaier. MINQ8: general definite and bound constrained indefinite quadratic programming. *Comput. Optim. Appl.* **69** (October 2017), 351–381.
26. M. Kimiaei and A. Neumaier. Testing and tuning optimization algorithm. Preprint, Vienna University, Fakultät für Mathematik, Universität Wien, Oskar-Morgenstern-Platz 1, A-1090 Wien, Austria (2019).
27. M. Kimiaei and A. Neumaier. Efficient global unconstrained black box optimization. `http://www.optimization-online.org/DB_HTML/2018/08/6783.html` (Jul 2020).
28. J. Larson, M. Menickelly, and S. M. Wild. Derivative-free optimization methods. *Acta Numer.* **28** (May 2019), 287–404.
29. I. Loshchilov, T. Glasmachers, and H. G. Beyer. Large scale black-box optimization by limited-memory matrix adaptation. *IEEE Trans. Evol. Comput.* **23** (April 2019), 353–358.
30. S. Lucidi and M. Sciandrone. A derivative-free algorithm for bound constrained optimization. *Comput. Optim. Appl.* **21** (2002), 119–142.
31. L. Lukšan, L and J. Vlcek. Sparse and partially separable test problems for unconstrained and equality constrained optimization. ICS AS CR, 1999. 30 s. Technical Report, V-767.
32. J. J. Moré and S. M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM J. Optim.* **20** (January 2009), 172–191.
33. A. Neumaier, H. Fendl, H. Schilly, and Thomas Leitner. VXQR: derivative-free unconstrained optimization based on QR factorizations. *Soft Comput.* **15** (September 2010), 2287–2298.
34. M. Porcelli and P. Toint. Global and local information in structured derivative free optimization with BFO. *arXiv: Optimization and Control* (2020).
35. M. J. D. Powell. UOBYQA: unconstrained optimization by quadratic approximation. *Math. Program.* **92** (May 2002), 555–582.
36. M. J. D. Powell. Developments of NEWUOA for minimization without derivatives. *IMA. J. Numer. Anal.* **28** (February 2008), 649–664.
37. L. M. Rios and N. V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *J. Global. Optim.* **56** (July 2012), 1247–1293.
38. V. J. Torczon. *Multidirectional search: A direct search algorithm for parallel machines*. PhD thesis, Diss., Rice University (1989).
39. B. Van Dyke and T. J. Asaki. Using QR decomposition to obtain a new instance of mesh adaptive direct search with uniformly distributed polling directions. *J. Optim. Theory Appl* **159** (June 2013), 805–821.
40. S. M. Wild, R. G. Regis, and C. A. Shoemaker. ORBIT: Optimization by radial basis function interpolation in trust-regions. *SIAM J. Sci. Comput.* **30** (January 2008), 3197–3219.
41. M. H Wright. Direct search methods: Once scorned, now respectable. *Pitman Research Notes in Math. Series* (1996), 191–208.