A. S. Berahas a, O. Soháb b and L. N. Vicente b

aDepartment of Industrial & Operations Engineering, University of Michigan, Ann Arbor, MI, USA; bDepartment of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA

ABSTRACT
We propose a new class of rigorous methods for derivative-free optimization with the aim of delivering efficient and robust numerical performance for functions of all types, from smooth to non-smooth, and under different noise regimes. To this end, we have developed a class of methods, called Full-Low Evaluation methods, organized around two main types of iterations. The first iteration type (called Full-Eval) is expensive in function evaluations, but exhibits good performance in the smooth and non-noisy cases. For the theory, we consider a line search based on an approximate gradient, backtracking until a sufficient decrease condition is satisfied. In practice, the gradient was approximated via finite differences, and the direction was calculated by a quasi-Newton step (BFGS). The second iteration type (called Low-Eval) is cheap in function evaluations, yet more robust in the presence of noise or non-smoothness. For the theory, we consider direct search, and in practice we use probabilistic direct search with one random direction and its negative. A switch condition from Full-Eval to Low-Eval iterations was developed based on the values of the line-search and direct-search stepsizes. If enough Full-Eval steps are taken, we derive a complexity result of gradient-descent type. Under failure of Full-Eval, the Low-Eval iterations become the drivers of convergence yielding non-smooth convergence. Full-Low Evaluation methods are shown to be efficient and robust in practice across problems with different levels of smoothness and noise.

1. Introduction
Derivative-Free Optimization (DFO) methods [4,22,24,40,49] are developed for the minimization of functions whose corresponding derivatives are unavailable for use or expensive to compute or approximate. The value of the functions is often computed via numerical simulations and may be subject to statistical noise or other forms of inaccuracy. Constraints may be part of the problem formulation and their derivatives may also be unavailable. DFO methods have applications in all fields of engineering and applied science (see the recent survey [1]), in particular whenever data is fitted with the purpose of finding optimal values for design or control variables of complex models or systems, or when there is the need...
to determine optimal parameters of computational solvers or to tune hyperparameters in artificial intelligence.

In this paper, we consider the following unconstrained problem

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

(1)

where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is assumed to have different properties throughout the paper. We will consider both the cases where \( f \) is smooth (assuming that it has a Lipschitz continuous gradient) and is possibly non-smooth (assuming only local Lipschitz continuity of \( f \)). The methods discussed and developed in this paper find application on problems where derivative information of whatever form is not available or is too expensive to compute [1]. Although we do not consider the noisy case explicitly in our theory, we do take it into account when developing and numerically testing the methods.

1.1. Advantages and disadvantages of the current DFO methods

When designing numerical algorithms for DFO [22], two main algorithmic paradigms are often used, in part depending on how \( f \) is sampled, leading to algorithms which are either directional or model-based.

Directional algorithms are based on the concept of a displacement along a direction. Among those, we find directional direct-search methods [3,13,34,39,53], which rely only on function evaluations without any implicit or explicit approximation of the gradient, or construction of a model. At each iteration, a finite set of directions is first generated, from which a set of polling points is considered by adding to the current iterate these directions multiplied by a certain stepsize. The objective function is then evaluated at all, or some, of these points depending on the polling type (opportunistic or complete), in order to search for a new point satisfying a (possibly sufficient) decrease condition. The iterate and stepsize are updated according to the outcome of the polling step. A search step [13] can be taken to improve numerical performance, with no influence on the convergence analysis. A relevant feature of these methods is their ability to converge when the function is non-smooth. In fact, if the normalized directions are asymptotically dense in the unit sphere, the iterates converge to a Clarke stationary point [3,54].

The directional direct-search framework has two main variants, a deterministic one [39,53] and a probabilistic one [34]. In the deterministic variant, the directions belong to positive spanning sets (PSS), which are sets of vectors that span the whole space with non-negative coefficients. When the objective function is smooth, at least one of these directions is a descent one [39]. However, the cardinal number of a PSS is at least \( n + 1 \), where \( n \) is the dimension of the problem, and one iteration may cost \( O(n) \) function evaluations. The probabilistic variant consists of using randomly generated polling directions that are probabilistically descent. Such property relies on the existence of a direction that makes an acute angle with the negative gradient with a sufficiently large probability, conditioned on the history of iterations [34]. An example of such a set is a uniformly drawn direction on the unit ball and its negative. In this case, the cost of an iteration is at most 2 function evaluations, leading to significant gains in efficiency [34]. In summary, directional direct-search methods are slow when using deterministic directions (although faster when randomized), but robust and parallelizable. Moreover, these methods work for non-smooth and possibly noisy functions.
Among directional methods for DFO, one also finds those based on line-search schemes. In [37,52], the authors considered fixed predetermined stepsizestrategies coupled with directions computed via (possibly randomized) finite differences (FD) approximations to the gradient of the objective function. Methods with adaptive stepsize choices have also been proposed [9,11,14]. These methods utilize either a simplex gradient approximation or a FD approximation to the gradient of the objective function (leading to $O(n)$ evaluations per iteration), and enhance the search direction by applying a quasi-Newton scheme. Some form of sufficient decrease condition is imposed to restrict the size of the step, and possibly a curvature condition to avoid short steps. In the noisy setting, approximating the gradient can be problematic. To mitigate this issue, regression techniques can be used to compute the gradient approximation [14]. Alternatively, optimal (in terms of minimizing the approximation error) FD approximations can be computed if the noise level is known, or can be estimated [43], and incorporated in a FD scheme [44]. The authors in [9] propose a FD quasi-Newton method with explicit noise estimation [43] and a relaxed sufficient decrease condition that is robust in the noisy setting, and that avoids re-estimating the noise at every iteration. These methods are moderately efficient and potentially scalable in the smooth case. However, when the objective function under consideration is non-smooth, such line-search methods are no longer suitable.

The other most commonly followed paradigm in the development of DFO algorithms relies on building models using objective function samples for use in trust-region methods [18,21,30,48,55]. Models can be built via interpolation or regression techniques, using basis functions such as quadratic polynomials or radial basis functions. Sample points can result from trust-region steps, be computed from well-designed deterministic model-improving techniques, or obtained from random sampling. At each iteration of a trust-region method [19], one typically considers the minimization of a quadratic model in a region around the current iterate, from which a trial point is obtained. Accepting the trial point and updating the trust-region radius are based on the ratio between the decrease achieved in the function and the one attained at the model. The model serves as a local approximation of the function curvature, and in the DFO case the quality of the model depends on the geometry of the sample points [20]. DFO trust-region methods are efficient and robust when the dimension of the problems is small. However, these methods do not scale well, not only because the overall computing time becomes an issue due to the dense linear algebra of the interpolation, but also due to the ill conditioning (poor geometry) of the sample sets. Moreover, these methods are mainly designed for smooth functions and are not easily parallelizable. Trust-region methods for DFO have been extended to handle noise in the objective function [12,26] (see [40] for a review of recent developments on stochastic functions).

There have been approaches in the literature hybridizing model-based and direct-search methods. While numerical results have been promising, no comprehensive convergence theory has been presented (other than the use of the optional search step [13] in direct search). Pioneer examples include [23,25,27] and more recent ones are [5,17], and the reader is referred to [40, Section 2.3] for a summary of this topic.

1.2. A rigorous DFO method that is robust for all function types

As we have seen in Section 1.1, most of the existing DFO methods have been designed and tailored for a specific type of problems, and one has to carefully choose the class to use
for the best results. In this paper, we introduce a new class of derivative-free optimization methods called Full-Low Evaluation, taking advantage of two types of iterations, with the goal of achieving sustainedly good performance across all possible function types. A first iteration type (Full-Eval) is expensive in function evaluations, but exhibits good performance in the smooth, non-noisy case. A second iteration type (Low-Eval) is cheap in terms of function evaluations and more appropriate in the presence of non-smoothness or/and noise.

In its general form, the Full-Eval iteration consists of a line-search step based on an approximate gradient, and the Low-Eval iteration consists of a direct-search step. The integration of the two iterations is done by switching from one to the other when it is deemed beneficial or necessary. The main switch is a form of detection of non-smoothness or noise in $f$ during Full-Eval iterations. The new class of methods is shown to be globally convergent with appropriate rates in the smooth case, whenever Full-Eval generates enough iterates. It is also shown to be globally convergent in the non-smooth case (and this is assured by Low-Eval when Full-Eval fails to bring the approximate gradient close to zero). Failure of Full-Eval is detected by the activation of the switch condition which compares the line-search and direct-search stepsizes. To our knowledge such type of rigorous results is novel in DFO.

Our practical implementation of Full-Low Evaluation considers a BFGS step based on a FD gradient for Full-Eval, and a probabilistic direct-search step based on a random vector and its negative for Low-Eval (see Section 4.1 for more details). The numerical results show that this practical version is robust and relatively efficient for all function types of varied smoothness, and noise origin and level.

Some authors often refer to a surrogate as a low-precision model, and to the true problem as a full precision one. In the context of our paper, Low and Full refer to the deployed algorithmic effort, not the quality of the evaluations and thus not related to the surrogate framework from [13].

1.3. Structure of the paper and notation

The paper is organized in the following way. In Section 2, we start by giving the general algorithmic description of the Full-Low Evaluation framework. We then present, in Section 3, the global convergence rate analysis of the method when applied to a smooth objective function. Specifically, we show that there exists a subsequence of iterates for which the gradients decay to 0 at a sublinear rate in the non-convex case, and at a linear one in the strongly convex case. We also derive in Section 3 a global convergence result for the non-smooth case. In Section 4, we introduce our practical choices for the Full-Eval and Low-Eval iterations, as well as the other solvers used in our testing environment. Finally, the performance of Full-Low Evaluation is reported in Section 5, based on tests conducted on different classes of functions (smooth, non-smooth, and noisy). The paper is concluded with some remarks in Section 6.

Let the set of indices corresponding to successful Full-Eval and Low-Eval iterations be denoted by $I_{SF}$ and $I_{SL}$, respectively. Similarly, the indices of unsuccessful iterations are denoted as $I_{UF}$ and $I_{UL}$, corresponding to the Full-Eval and Low-Eval iterations, respectively. The set of all iterations is denoted by $I_{SF} \cup I_{SL} \cup I_{UF} \cup I_{UL}$. All norms in this paper are Euclidian.
2. Full-low evaluation methods

The main idea of the Full-Low Evaluation methods is based on the combination of two types of steps. The first type is expensive in function evaluations (Full-Eval), but exhibits good performance in the smooth, non-noisy case. The second type is cheaper in function evaluations (Low-Eval) and at the same time more appropriate in the presence of noise and/or non-smoothness. For such two goals, we have in mind some form of model-based or enhanced gradient-descent step for Full-Eval, capable of exploring curvature, and a randomized directional procedure for Low-Eval, less sensitive to noise and qualified for delivering convergence in the non-smooth regime. The general mechanism of the Full-Low Evaluation approach is described in Algorithm 1. Full-Eval steps are consecutively taken until a certain condition, designed to sense the presence of non-smoothness and/or noise, is activated, after which one switches to Low-Eval iterations. The number of consecutive Low-Eval iterations is a user defined parameter, and can be selected as a function of the last successful Full-Eval iteration (see Section 4.1 for details of our practical implementation of the Full-Low Evaluation method).

Algorithm 1 Full-Low Evaluation Algorithm

Initialization: Choose an initial iterate $x_0$. Set iteration $t_0 =$ Full-Eval.

1. For $k = 0, 1, 2, \ldots$
2. If $t_k =$ Full-Eval, attempt to compute a Full-Eval step.
3. If success, update $x_{k+1}$ and set $t_{k+1} =$ Full-Eval.
4. Else, $x_{k+1} = x_k$ and $t_{k+1} =$ Low-Eval.
5. If $t_k =$ Low-Eval, compute a Low-Eval step.
6. Update $x_{k+1}$. Decide on $t_{k+1} \in \{\text{Low-Eval, Full-Eval}\}$.

In the general setting, the instance we consider for the Full-Eval step is of line-search nature, where the search direction $p_k$ is calculated based on an approximate gradient $g_k$. The conditions imposed on both $p_k$ and $g_k$ are stated in Section 3. If the Full-Eval step is successful, the next iterate is necessarily of the form $x_k + \beta_k p_k$, where $\beta_k$ is a positive stepsize. As is typical in nonlinear optimization, the stepsize $\beta_k$ is required to satisfy a sufficient decrease condition of the form

$$f(x_k + \beta_k p_k) \leq f(x_k) + c \beta_k g_k^T p_k,$$

(2)

where $c \in (0, 1)$ is independent of $k$ [46]. Under appropriate assumptions, condition (2) can be ensured by backtracking from a fixed stepsize until it is satisfied. By doing so, one ensures steps that are simultaneously of restricted size and not too small.

A key modification we introduce in the Full-Eval iteration is that we do not allow the stepsize $\beta_k$ to become too small compared to a certain function of $\alpha_k$, the direct-search stepsize used in Low-Eval iterations (described below). In fact we only consider a Full-Eval iteration successful if $\beta_k$ satisfies

$$\beta \geq \gamma \rho(\alpha_k),$$

(3)

where $\gamma > 0$ is independent of $k$ and $\rho(\cdot)$ is the forcing function used in the direct-search scheme of the Low-Eval iteration. If we backtrack too much so that we violate (3), the Full-Eval step is skipped. The Full-Eval iteration is described in Algorithm 2.
Algorithm 2 Full-Eval Iteration: Line Search

**Input:** Iterate \( x_k \) and \( \text{Low-Eval} \) stepsize \( \alpha_k \). Backtracking parameters \( \beta > 0 \) and \( \tau \in (0,1) \). **Output:** \( t_{k+1}, x_{k+1}, \) and \( \alpha_{k+1} \).

1. Compute an approximate gradient \( g_k \).
2. Compute a direction \( p_k \).
3. Backtracking line-search: Set \( \beta = \bar{\beta} \). If (3) is false, stop.
4. While (2) is false
5. Set \( \beta = \tau \beta \).
6. If (3) is false, set \( x_{k+1} = x_k \) and \( t_{k+1} = \text{Low-Eval} \), and stop the While cycle.
7. If (3) is true, set \( \beta_k = \beta, x_{k+1} = x_k + \beta_k p_k \), and \( t_{k+1} = \text{Full-Eval} \).
   (Note, the \( \text{Low-Eval} \) parameter \( \alpha_k \) remains unchanged, \( \alpha_{k+1} = \alpha_k \); see Algorithm 3.)

Our proposed \( \text{Low-Eval} \) iteration is described in Algorithm 3, and consists of applying one step of direct search. The set of polling directions \( D_k \) is for the moment left unspecified. Similar to a line-search, at each iteration, the stepsize \( \alpha_k \) is required to satisfy a sufficient decrease condition of the form

\[
 f(x_k + \alpha_k d_k) \leq f(x_k) - \rho(\alpha_k),
\]

where \( \rho(\cdot) \) is a (positive) forcing function satisfying the conditions given in Section 3. If the iteration is successful the stepsize is kept constant or increased, otherwise it is decreased. The factors for stepsize increase/decrease must obey the theory, which is met for instance if they are independent of \( k \). An initial value of \( \alpha_0 > 0 \) must be supplied to the first \( \text{Low-Eval} \) iteration.

Algorithm 3 Low-Eval Iteration: Direct Search

**Input:** Iterate \( x_k \) and stepsize \( \alpha_k \).

Direct-search parameters \( \lambda \geq 1 \) and \( \theta \in (0,1) \). **Output:** \( t_{k+1}, x_{k+1}, \) and \( \alpha_{k+1} \).

1. Generate a finite set \( D_k \) of non-zero polling directions.
2. If (4) is true for some \( d_k \in D_k \), set \( x_{k+1} = x_k + \alpha_k d_k \) and \( \alpha_{k+1} = \lambda \alpha_k \).
3. Else, set \( x_{k+1} = x_k \) and \( \alpha_{k+1} = \theta \alpha_k \).
4. Decide if \( t_{k+1} = \text{Low-Eval} \) or if \( t_{k+1} = \text{Full-Eval} \).

In order to switch from \( \text{Low-Eval} \) to \( \text{Full-Eval} \), one can compare the number of unsuccessful consecutive \( \text{Low-Eval} \) iterations \( (nu_k) \) to the number of line-search backtracks \( (nb_{jk}) \) done in the previous \( \text{Full-Eval} \) iteration \( (J_k) \). When \( nu_k \) becomes greater than \( nb_{jk} \), that is perhaps a sign that too much effort was put in the current \( \text{Low-Eval} \) iteration, a switch is desirable. We will return to this in Section 4. For the purpose of the convergence theory, we will assume an infinity of \( \text{Full-Eval} \) or \( \text{Low-Eval} \) iterations whenever necessary.

### 3. Convergence and rates of convergence of full-low evaluation methods

Convergence analysis for the Full-Low Evaluation methods is presented for both the smooth and non-smooth cases. The noisy case will be addressed within the non-smooth analysis; see Section 3.2.
3.1. Convergence rates in the smooth case

In this section, we analyze the behavior of the class of Full-Low Evaluation methods in the smooth case. We show that if the Full-Eval step generates an infinity of iterates, the convergence and rates of convergence guaranteed match those of deterministic gradient descent. We now introduce the assumptions needed for the analysis, starting by the smoothness of $f$.

**Assumption 3.1:** The function $f$ is continuously differentiable and its gradient $\nabla f$ is Lipschitz continuous with constant $L > 0$.

The approximate gradient used in Full-Eval iterations is required to satisfy the following assumption.

**Assumption 3.2:** The approximate gradient $g_k$ computed at $x_k$ satisfies

$$\|\nabla f(x_k) - g_k\| \leq u_g \beta_k \|g_k\|,$$

where $u_g > 0$ is independent of $k$.

When approximating the gradient by finite differences, Assumption 3.2 can be rigorously ensured by Algorithm 4 (see Proposition 3.1). Note that if $u_g \beta_k < 1$, Assumption 3.2 implies that the negative gradient approximation $-g_k$ is a descent direction:

$$-\nabla f(x_k)^T g_k = -g_k^T g_k + (g_k - \nabla f(x_k))^T g_k \geq \|g_k\|^2 - u_g \beta_k \|g_k\|^2 > 0.$$

Finally, we state the assumptions on the Full-Eval directions.

**Assumption 3.3:** There exist constants $\kappa, u_p > 0$ such that

$$\cos(-g_k, p_k) = \frac{(-g_k)^T p_k}{\|g_k\| \|p_k\|} > \kappa$$

and

$$\|g_k\| \leq u_p \|p_k\|.$$  

Condition (6) is classical in line-search methods [46, Section 3.2], and imposes an acute angle between the direction and the approximate negative gradient, bounded away from ninety degrees. Note that if $u_g \beta_k < \kappa$, conditions (5) and (6) together imply that $p_k$ is a descent direction:

$$-\nabla f(x_k)^T p_k = -g_k^T p_k + (g_k - \nabla f(x_k))^T p_k \geq \kappa \|g_k\| \|p_k\| - u_g \beta_k \|g_k\| \|p_k\| > 0.$$  

Condition (7) is also mild. In a Newton or quasi-Newton context $p_k = -B_k^{-1} g_k$, it is essentially saying that the Hessian or secant matrix $B_k$ is bounded (which is what is imposed in trust-region methods). One can relax (7) to $\|g_k\| \leq k^a \|p_k\|$, with $a > 0$, although the sublinear rate $k^{-1/2}$ would then increase to $k^{-1/(2+a)}$. We start the analysis by establishing a lower bound on the stepsize $\beta_k$. 
Lemma 3.1: Suppose that Assumptions 3.1–3.3 hold. If $k$ is a successful Full-Eval iteration, then

$$\beta_k \geq \min \left\{ \bar{\beta}, \frac{2\tau(1-c)}{2u_g u_p + L} \frac{(-g_k)^\top p_k}{\|p_k\|^2} \right\}.$$ 

Proof: Using a Taylor expansion and by Assumption 3.1, one has

$$f(x_k + \beta p_k) \leq f(x_k) + \beta p_k^\top \nabla f(x_k) + \frac{\beta^2}{2} L \|p_k\|^2$$

$$\leq f(x_k) + \beta g_k^\top p_k + \left( \frac{1}{2} L + u_g u_p \right) \beta^2 \|p_k\|^2,$$

where the last line follows from applying (5) and (7).

If $\beta_k \neq \bar{\beta}$, then $\beta = \beta_k/\tau$ does not satisfy the sufficient decrease condition, which means

$$f(x_k + (\beta_k/\tau)p_k) - f(x_k) > c (\beta_k/\tau) g_k^\top p_k.$$ (9)

We can now plug $\beta = \beta_k/\tau$ into (8), and combine it with (9), to obtain

$$g_k^\top p_k + \left( \frac{1}{2} L + u_g u_p \right) \frac{\beta_k}{\tau} \|p_k\|^2 > c g_k^\top p_k.$$ 

We conclude that

$$\beta_k > \frac{2\tau(1-c)}{2u_g u_p + L} \frac{(-g_k)^\top p_k}{\|p_k\|^2},$$

where $1-c > 0$. The proof is completed by combining the case where $\beta_k = \bar{\beta}$ and the one presented above. ■

Next, we prove that the minimum norm of the true gradient decays at the appropriate sublinear rate, for iterations $k$ in the set $\mathcal{I}_{SF}$ of successful Full-Eval iterations.

Theorem 3.1: Let Assumptions 3.1–3.3 hold. Assume that $f$ is bounded from below (and let $f_{\text{low}}$ be a lower bound). Then,

$$\min_{i=0\ldots k-1} \|\nabla f(x_i)\| \leq (\bar{\beta} u_g + 1) \sqrt{\frac{f(x_0) - f_{\text{low}}}{M}} \frac{1}{\sqrt{\text{ns}}},$$

where $\text{ns} = |\mathcal{I}_{SF} \cap \{0, \ldots, k-1\}|$ is the number of successful Full-Eval iterations up to iteration $k$, and

$$M = c \min \left\{ \frac{\bar{\beta} \kappa}{u_p}, \frac{2\tau \kappa^2(1-c)}{2u_g u_p + L} \right\}.$$ (10)
Proof: Let \( k \in \mathcal{I}_{SF} \). First, using (6) and (7), we write

\[
\left( (-g_k) \top p_k \right)^2 / \| p_k \|^2 \geq \kappa^2 \| g_k \|^2 \quad \text{and} \quad (-g_k) \top p_k \geq \frac{\kappa}{u_p} \| g_k \|^2.
\]

Then, from the sufficient decrease condition (2) and the lower bound for the stepsize established in Lemma 3.1, one obtains

\[
f(x_k) - f(x_{k+1}) \geq M \| g_k \|^2,
\]

where \( M \) is given in (10).

By Assumption 3.2, the triangle inequality, and \( \beta_k \leq \bar{\beta} \), we have

\[
\frac{1}{\beta u_g + 1} \| \nabla f(x_k) \| \leq \| g_k \|.
\]

Hence, plugging this bound into inequality (12) yields

\[
f(x_k) - f(x_{k+1}) \geq \frac{M}{(\beta u_g + 1)^2} \| \nabla f(x_k) \|^2.
\]

In both unsuccessful Full-Eval and Low-Eval iterations, the function value does not decrease, and thus from (13)

\[
f(x_0) - f(x_k) \geq \frac{M}{(\beta u_g + 1)^2} \sum_{i \in \mathcal{I}_{SF} \cap \{0, \ldots, k-1\}} \| \nabla f(x_i) \|^2.
\]

The proof is completed by taking the minimum from 0 to \( k-1 \) on the successful Full-Eval iterations and using the lower bound \( f(x_k) \geq f_{\text{low}} \).

Remark 3.1: One can see from Theorem 3.1 that the number of successful Full-Eval iterations needed to drive the gradient below a positive threshold \( \epsilon \) is of the order of \( \epsilon^{-2} \).

The number of Full-Eval backtracking steps can also be bounded under an additional assumption. In fact, one has \( \beta_k = \beta \tau^{nb_k} \), where \( nb_k \) is the number of backtracks at Full-Eval iteration \( k \). By applying logs to this equation, one obtains \( nb_k = \log(\beta_k) / \log(\tau) - \log(\bar{\beta}) / \log(\tau) \), where \( \log(\tau) < 0 \).

When \( p_k = -g_k \), one can see from Assumption 3.3 that

\[
\frac{(-g_k) \top p_k}{\| p_k \|^2} \geq \frac{\kappa}{\| g_k \|},
\]

and thus, from Lemma 3.1, \( \beta_k \) is bounded from below. Hence, \( nb_k \) is bounded by a constant.

When \( p_k \neq -g_k \), one needs the additional assumption \( \| g_k \| \geq \ell_p \| p_k \| \), where \( \ell_p \) is a positive constant, to also state that \( nb_k \) is of the order of 1. Such an assumption is guaranteed in a Newton or quasi-Newton approach \( p_k = -B_k^{-1} g_k \) when \( B_k \) is positive definite and bounded away from singularity.

Suppose now that the sequence \( \mathcal{I}_{SF} \) of Full-Eval successful iterations is infinite. From the rate established, one directly concludes that the gradient of \( f \) goes to zero for
a subsequence of $\mathcal{I}_{SF}$. However, since the series $\sum_{i \in \mathcal{I}_{SF}} \| \nabla f(x_i) \|^2$ is summable, we can also state $\lim_{k \in \mathcal{I}_{SF}} \| \nabla f(x_k) \| = 0$.

The convergence rate becomes linear when $f$ is strongly convex, which is the case when there exists a positive constant $\mu > 0$ such that for all $(x, y) \in \mathbb{R}^n$

$$f(y) \geq f(x) + \nabla f(x) \top (y - x) + \frac{\mu}{2} \| y - x \|^2.$$ 

The argument uses the Polyak–Łojasiewicz (PL) inequality

$$\| \nabla f(x_k) \|^2 \geq 2 \mu(f(x_k) - f(x*)) - \frac{\mu}{2} \| x_* - x_k \|^2,$$

obtained using strong convexity with $x = x_k$ and $y = x_*$ (the unique minimizer of $f$), with

$$2 \mu \nabla f(x_k) \top (x_* - x_k) + \mu^2 \| x_* - x_k \|^2 \geq -\| \nabla f(x_k) \|^2.$$ 

It remains to combine the PL inequality with (13), and recursively conclude that (when $k - 1 \in \mathcal{I}_{SF}$)

$$f(x_k) - f(x_*) \leq (1 - \eta \mu)^n s_k (f(x_0) - f(x_*))$$

with $\eta \mu = \frac{2 \mu c}{(\bar{\beta} + 1)^2} \min(\bar{\beta} \frac{\kappa}{\mu p}, \frac{2 \tau k^2 (1-c)}{2 c m + L})$, and again $s_k = |\mathcal{I}_{SF} \cap \{0, \ldots, k - 1\}|$. Note that if is chosen such that $c \in (0, 1/2)$, then $\eta \mu \in (0, 1)$.

### 3.2. Convergence in the non-smooth case

The analysis in the non-smooth case is based on the failure of the $\text{Full-Eval}$ iterations.

**Assumption 3.4 ($\text{Full-Eval}$ Failure):** The sequence of gradient norms $\{\| g_k \| \}_{k \in \mathcal{I}_{SF}}$ is bounded away from 0, i.e. there exists $\epsilon_g > 0$ such that for all iterations $k \in \mathcal{I}_{SF}$, one has $\| g_k \| > \epsilon_g$.

The switching condition (3) allows us to prove under this assumption that the $\text{Low-Eval}$ iterations generate an infinite subsequence of iterates driving the direct-search parameter $\alpha_k$ to zero. This result requires the forcing function to satisfy Assumption 3.5.

**Assumption 3.5:** The function $\rho$ is positive, non-decreasing, and satisfies $\lim_{\alpha \to 0^+} \rho(\alpha)/\alpha = 0$.

One can see from the proof of Lemma 3.2 (below) that $\text{Full-Eval}$ steps are essentially considered as search steps from the direct-search perspective of the $\text{Low-Eval}$ iterations.

**Lemma 3.2:** Let Assumptions 3.3–3.5 hold. Assume that the sequence of iterates $\{x_k\}$ is bounded. Then, there exists a point $x_*$ and a subsequence $\mathcal{K} \subset \mathcal{I}_{UL}$ of unsuccessful $\text{Low-Eval}$ iterates for which

$$\lim_{k \in \mathcal{K}} x_k = x_* \quad \text{and} \quad \lim_{k \in \mathcal{K}} \alpha_k = 0.$$
Proof: First, consider that there is an infinity of iterations in $I_{SF} \cup I_{UF} \cup I_{SL}$. These are all iterations $k$ for which $\alpha_k$ does not decrease. By the second inequality in (11), all successful Full-Eval iterations $k \in I_{SF}$ yield

$$f(x_{k+1}) \leq f(x_k) - c(\kappa/u_p)\beta_k \|g_k\|^2.$$  

From Assumption 3.4 and the fact that $\beta_k$ satisfies $\beta_k \geq \gamma \rho(\alpha_k)$,

$$f(x_{k+1}) - f(x_k) \leq -c(\kappa/u_p)\gamma \epsilon_g \rho(\alpha_k). \quad (14)$$

Successful Low-Eval iterations $k \in I_{SL}$ achieve sufficient decrease,

$$f(x_{k+1}) - f(x_k) \leq -\rho(\alpha_k). \quad (15)$$

Note that in Full-Eval unsuccessful iterations $k \in I_{UF}$ neither $x_k$ nor $\alpha_k$ changes.

Hence, given that for unsuccessful Low-Eval iterations ($I_{UL}$) the function does not decrease, we can sum from $0$ to $k \in I_{SF} \cup I_{UF} \cup I_{SL}$ the inequalities (14) and (15) to obtain

$$f(x_0) - f(x_{k+1}) \geq c \gamma (\kappa/u_p)\epsilon_g \sum_{i \in I_{SF}} \rho(\alpha_i) + \sum_{i \in I_{SL}} \rho(\alpha_i).$$

By the boundedness (from below) of $f$, we conclude that the series is summable, which implies $\lim_{k \in I_{SF} \cup I_{UF} \cup I_{SL}} \rho(\alpha_k) = 0$. In conjunction with Assumption 3.5, this in turn implies $\lim_{k \in I_{SF} \cup I_{UF} \cup I_{SL}} \alpha_k = 0$.

It remains to consider the iterations in $I_{UL}$. For each $k \in I_{UL}$ corresponding to an unsuccessful Low-Eval iteration, consider the previous iteration $k' = k'(k) \in I_{SL} \cup I_{UF} \cup I_{SL}$ with $k' < k$ ($k'$ could be zero). The direct-search stepsize can then be written as $\alpha_k = \theta^{k-k'} \alpha_{k'}$. Since $k' \to \infty$ and $\tau \in (0, 1)$, one obtains $\alpha_k \to 0$ for all $k \in I_{UL}$.

We have thus proved that $\alpha_k$ goes to zero for all $k$. Since $\alpha_k$ is only decreased in unsuccessful Low-Eval iterations, there must be an infinite subsequence of those. From the boundedness of the sequence of iterates, one can extract a subsequence $K$ of that subsequence satisfying the statement of the lemma.

The main theorem uses the notion of generalized Clarke derivative [16] at $x$ along a direction $d \in \mathbb{R}^n$

$$f^\diamondsuit(x; d) = \limsup_{y \to x, t\downarrow 0} \frac{f(y + td) - f(y)}{t},$$

which is well defined if $f$ is Lipschitz continuous around the point $x$. Establishing that there is a limit point which is Clarke stationary requires the density of the so-called refining directions to be dense in the unit sphere. The proof follows the arguments in [2,3,54].

Theorem 3.2: Let Assumptions 3.3–3.5 hold. Assume that the sequence of iterates $\{x_k\}$ is bounded. Let the function $f$ be Lipschitz continuous around the point $x_\ast$ defined in Lemma 3.2. Let the set of limit points of

$$\left\{ \frac{d_k}{\|d_k\|}, \; d_k \in D_k, \; k \in K \right\}$$

be dense in the unit sphere, where $K \subset I_{UL}$ is given in Lemma 3.2.

Then, $x_\ast$ is a Clarke stationary point, i.e. $f^\diamondsuit(x_\ast; d) \geq 0$ for all $d \in \mathbb{R}^n$. 

**Proof:** Let \( \widetilde{d} \) be a limit point of (16), identified for a certain subsequence \( \mathcal{L} \subseteq \mathcal{K} \). Then, from basic properties of the generalized Clarke derivative, and \( k \in \mathcal{L} \),

\[
f^{\circ}(x^*; \tilde{d}) = \lim_{x_k \to x^*} \sup_{\alpha_k \downarrow 0} \frac{f(x_k + \alpha_k \tilde{d}) - f(x_k)}{\alpha_k} \\
\geq \lim_{x_k \to x^*} \sup_{\alpha_k \downarrow 0} \left\{ \frac{f(x_k + \alpha_k d_k) - f(x_k)}{\alpha_k} - L_f^* \|d_k - \tilde{d}\| \right\} \\
= \lim_{x_k \to x^*} \sup_{\alpha_k \downarrow 0} \left\{ \frac{f(x_k + \alpha_k d_k) - f(x_k)}{\alpha_k} + \frac{\rho(\alpha_k)}{\alpha_k} \right\},
\]

where \( L_f^* \) is the Lipschitz constant of \( f \) around \( x^* \). Since \( k \in \mathcal{L} \) are unsuccessful Low-Eval iterations, it follows that \( f(x_k + \alpha_k d_k) > f(x_k) - \rho(\alpha_k) \) which implies that

\[
\lim_{x_k \to x^*} \sup_{\alpha_k \downarrow 0} \frac{f(x_k + \alpha_k d_k) - f(x_k) + \rho(\alpha_k)}{\alpha_k} \geq 0.
\]

From this and Assumption 3.5, we obtain \( f^{\circ}(x^*; \tilde{d}) \geq 0 \). Given the continuity of \( f^{\circ}(x^*; \cdot) \), one has for any \( d \in \mathbb{R}^n \) such that \( \|d\| = 1 \), \( f^{\circ}(x^*; d) = \lim_{\tilde{d} \to d} f^{\circ}(x^*; \tilde{d}) \geq 0 \). \( \blacksquare \)

### 3.3. More on the smooth case (use of finite difference gradients)

Let us return to the smooth case to clarify the imposition of Assumption 3.2. Such an assumption is related to the satisfaction of the so-called criticality step in DFO trust-region methods [21,22] when using fully linear models. In the context of the line-search method used in the **Full-Eval** iterations, those models correspond to using a finite difference (FD) scheme to compute the approximate gradient \( g_k \).

The \( i \)-th component of the forward FD approximation of the gradient at \( x_k \) is defined as

\[
[\nabla_h f(x_k)]_i = \frac{f(x_k + h_k e_i) - f(x_k)}{h_k}, \quad i = 1, \ldots, n, \tag{17}
\]

where \( h_k \) is the finite difference parameter and \( e_i \in \mathbb{R}^n \) is the \( i \)-th canonical vector. Computing such a gradient approximation costs \( n \) function evaluations per iteration, and it is implicitly assumed that such evaluations can be made. By using a Taylor expansion, the error in the FD gradient (in the smooth and noiseless setting) can be shown to satisfy

\[
\|\nabla f(x_k) - \nabla_h f(x_k)\| \leq \frac{1}{2} \sqrt{n} L h_k. \tag{18}
\]

It becomes then clear that one way to ensure Assumption 3.2 in practice, when \( g_k = \nabla_h f(x_k) \), is to enforce \( h_k \leq u_g' \beta \|\nabla_h f(x_k)\| \), for some \( u_g' > 0 \), in which case \( u_g = \frac{1}{2} \sqrt{n} L u_g' \). Enforcing such a condition is expensive but can be rigorously done through a criticality-step type argument (see Algorithm 4).
Algorithm 4 Criticality step: Performed if $h_k > u'_g \beta \| \nabla h_k f(x_k) \|

Input: $h_k$, $\nabla h_k f(x_k)^{(0)} = \nabla h_k f(x_k)$, $\beta$, and $\omega \in (0, 1)$. Let $j = 0$. Output: $\nabla h_k f(x_k) = \nabla h_k f(x_k)^{(j)}$ and $h_k$.

1: While $h_k > u'_g \beta \| \nabla h_k f(x_k)^{(j)} \|$ Do
2: Set $j = j + 1$ and $h_k = \omega^j u'_g \beta \| \nabla h_k f(x_k)^{(0)} \|$.
3: Compute the FD approximation $\nabla h_k f(x_k)^{(j)}$ using (17).

Proposition 3.1 shows that Algorithm 4 terminates in a finite number of steps.

**Proposition 3.1:** Let Assumption 3.1 hold. Condition $h_k \leq u'_g \beta \| \nabla h_k f(x_k) \|$ can be attained in a finite number of steps using Algorithm 4 if $\| \nabla f(x_k) \| \neq 0$.

**Proof:** Let us suppose that the algorithm loops infinitely. Then, for all $j \geq 1$, using Step 3 and the satisfaction of the while–condition in Step 1,

$$\| \nabla h_k f(x_k)^{(j)} \| \leq \omega^j \| \nabla h_k f(x_k)^{(0)} \|. \quad (19)$$

On the other hand, for all $j \geq 1$, the FD bound (18), followed by Step 3, gives us

$$\| \nabla f(x_k) - \nabla h_k f(x_k)^{(j)} \| \leq \frac{1}{2} \sqrt{n} L \omega^j u'_g \beta \| \nabla h_k f(x_k)^{(0)} \|. \quad (20)$$

Hence, using (19)–(20), we have

$$\| \nabla f(x_k) \| \leq \| \nabla f(x_k) - \nabla h_k f(x_k)^{(j)} \| + \| \nabla h_k f(x_k)^{(j)} \| \leq \left( \frac{\sqrt{n} L u'_g \beta}{2} + 1 \right) \omega^j \| \nabla h_k f(x_k)^{(0)} \|. $$

By taking limits (and noting that $\omega \in (0, 1)$), we conclude that $\| \nabla f(x_k) \| = 0$, which yields a contradiction. 

4. Numerical setup

In this section, we will first present our implementation choices for the Full-Low Evaluation method. The numerical environment of our experiments is also introduced (other methods/solvers tested, test problems chosen, and performance profiles). The tests were run using MATLAB R2019b on an Asus Zenbook with 16 GB of RAM and an Intel Core i7-8565U processor running at 1.80 GHz.

4.1. Our practical full-low evaluation implementation

Our proposed Full-Eval line-search iteration consists of using a direction $p_k$ of the form $p_k = -H_k g_k$. Our choice for the approximate gradient $g_k$ is forward FD (17) with $h_k$
set to the square root of Matlab’s machine precision. Our choice for \( H_k \) is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton update \([15,31,32,50]\). The choice of BFGS is justified by its strong performance in the presence of derivatives \([46]\). The combination of BFGS with FD is regarded as a useful resource for derivative-free optimization in many NLP solvers such as for instance the `fminunc` Matlab routine, and as shown in \([9,10,51]\). Our Full-Eval line-search iteration is described in Algorithm 5.

BFGS updates the inverse Hessian approximation \( H_k \) using \([23]\), where \( j_k \) is the previous Full-Eval iteration, and \( s_k \) and \( y_k \) are given in \([22]\). In our implementation, the first Full-Low Evaluation iteration is always Full-Eval. In the non-convex case, the products \( s_k^\top y_k \) cannot be ensured positive. In order to maintain the positive definiteness of the matrix \( H_k \), we skip the BFGS update if \( s_k^\top y_k \geq \epsilon_c s_k \|y_k\| \) is not satisfied, for \( \epsilon_c \in (0,1) \) independent of \( k \). In the implementation, we used \( \epsilon_c = 10^{-10} \). The line-search follows the backtracking scheme described in Algorithm 2, using \( \bar{\beta} = 1 \) and \( \tau = 0.5 \). A key feature of our Full-Low Evaluation methodology that led to rigorous results (see the proof of Lemma 3.2) is to stop the line-search once condition (3) is violated. In our implementation, we used \( \gamma = 1 \), \( \rho(\alpha_k) = \min(\gamma_1, \gamma_2 \alpha_k^2) \), with \( \gamma_1 = 10^{-5} \) and \( \gamma_2 = 10^{-3} \). (21)

When \( k = 0 \), we perform a backtracking line-search using \( p_0 = -g_0 \) (and update \( t_1 \) and \( x_1 \)) as in Algorithm 2 (with constants as in Algorithm 5). The initialization of \( H_0 \) is done as follows. If \( t_1 = \text{Full-Eval} \), then we set \( H_0 = (y_0^\top s_0)/(y_0^\top y_0)I \). This formula attempts to make the size of \( H_0 \) similar to the one of \( \nabla^2 f(x_0)^{-1} \) (see \([46]\)). However, if \( t_1 = \text{Low-Eval} \), we set \( H_0 = I \).

**Algorithm 5 Full-Eval Iteration: BFGS with FD Gradients**

**Input:** Iterate \( x_k \) with \( k \geq 1 \). Information \((x_{jk}, g_{jk}, H_{jk})\) from the previous Full-Eval iteration \( j_k \) (if \( k > 0 \)). Backtracking parameters \( \bar{\beta} > 0 \) and \( \tau \in (0,1) \). Other parameters \( \epsilon_c, \gamma_1, \gamma_2 > 0, \gamma = 1 \). **Output:** \( t_{k+1} \) and \((x_{k+1}, H_k, g_k)\). Return the number \( nb_k \) of backtrack attempts.

1. Compute the FD gradient \( g_k = \nabla h_k f(x_k) \) using \((17)\).
2. Set \( s_k = x_k - x_{jk} \) and \( y_k = g_k - g_{jk} \).
3. **If** \( s_k^\top y_k \geq \epsilon_c s_k \|y_k\| \), **set**
   \[
   H_k = \left( I - \frac{s_k y_k^\top}{y_k^\top s_k} \right) H_{jk} \left( I - \frac{y_k s_k^\top}{y_k^\top s_k} \right) + \frac{s_k s_k^\top}{y_k^\top s_k}.
   \]
4. **Else**, set \( H_k = H_{jk} \).
5. Compute the direction \(-H_k g_k\).
6. Perform a backtracking line-search, and update \( t_{k+1} \) and \( x_{k+1} \) as in Algorithm 2.

For the Low-Eval iterations, we considered the set of polling directions \( D_k \) to be formed by one random direction and its negative, a variant which has shown superior
performance compared with deterministic direct search based on PSS (see [34]). The random direction is drawn uniformly on the unit sphere. Note that the cost in function evaluations (at most 2 per iteration) is considerably lower than in our Full-Eval iteration, especially when \( n \) is large. The forcing function \( \rho(\alpha_k) \) used to accept polling points is set as in (21). As suggested after Algorithm 3, the switch from Low-Eval to Full-Eval will occur when the number \( nu_k \) of consecutive unsuccessful Low-Eval iterations reaches the number \( nb_{jk} \) of backtracks done in the last Full-Eval line-search. In the implementation, we set the stepsize updating factors to \( \lambda = 1/\theta = 2 \). See Algorithm 6.

Algorithm 6 Low-Eval Iteration: Probabilistic Direct Search

**Input:** Iterate \( x_k \) and stepsize \( \alpha_k \). Direct-search parameters \( \lambda \geq 1 \) and \( \theta \in (0,1) \). Number \( nu_k \) of unsuccessful Low-Eval iterations since last Full-Eval iteration \( j_k \). (Set \( nu_k = 0 \) if \( t_{k-1} = \text{Full-Eval} \).) Number \( nb_{jk} \) of backtracks done at Full-Eval iteration \( j_k \).

**Output:** \( t_{k+1}, x_{k+1}, \alpha_{k+1}, \) and \( nu_{k+1} \).

1: Generate \( d \in \mathbb{R}^n \) uniformly on the unit sphere of \( \mathbb{R}^n \), and set \( D_k = [d, -d] \).
2: Poll as in Algorithm 3.
3: Set \( nu_{k+1} = nu_k \) in the successful case, \( nu_{k+1} = nu_k + 1 \) otherwise.
4: If \( nu_k < nb_{jk} \), \( t_{k+1} = \text{Low-Eval} \).
5: Else, \( t_{k+1} = \text{Full-Eval} \).

4.2. Other solvers tested

We compared the numerical performance of our implementation of Full-Low Evaluation to four other approaches: (i) a line-search BFGS method based on FD gradients (as if there were only Full-Eval iterations), referred to as BFGS-FD; (ii) probabilistic direct search (as if there were only Low-Eval iterations), referred to as pDS; (iii) an interpolation-based trust-region solver, DFO-TR [8]; (iv) a line-search BFGS method based on FD gradients and noise estimation, FDLM [9]; (v) a mesh adaptive direct search solver, NOMAD.

DFO-TR [8] builds models by quadratic interpolation. At the first iteration, the function is evaluated using a sample set of \( 2n + 1 \) points, given by \( x_0 \) and \( x_0 \pm \Delta \epsilon_i \), with \( \epsilon_i \) the \( i \)th canonical vector. Until the cardinal number of the sample set reaches \( p_{\text{max}} = (n + 1)(n + 2)/2 \), all trust-region trial points \( x_k + s_k \) are added to the sample set, and models are computed by minimum Frobenius norm interpolation [22,47]. Once there are \( p_{\text{max}} \) points in the sample set, a quadratic model is built by determined interpolation. Then, for each new trial point added, an existing sample point is discarded (the farthest away from \( x_k + s_k \)). When \( \Delta_k \) falls below a certain threshold, points that are too far from the current iterate are discarded, expecting that the next iterations will refill the sample set, providing an effect similar to a criticality step. The trust-region radius \( \Delta_k \) is updated as it is common in trust-region methods, the major difference being that it is never reduced when the sample set has less than \( n + 1 \) points.

FDLM [9] is a linesear BFGS method for noisy functions where the gradients are approximated using FD. This approach differs from our Full-Eval iterations in two main ways. First, in FDLM the noise level is estimated using [43], and such estimate is
used to select the FD parameter as suggested in [44]. Second, in [9], the sufficient decrease condition is relaxed using the estimated noise level to prevent valuable points from being discarded. Moreover, to avoid estimating the noise at every iteration, the authors [9] developed a recovery procedure that estimates the noise only as needed (as the optimization progresses). The FDL method was chosen to compare our method to a version of FD-based BFGS where the noise is directly taken into account. We should emphasize again that in our proposed method, Full-Low Evaluation, the noise is directly handled by the Low-Eval iterations.

NOMAD [6] is a solver that implements Mesh Adaptive Direct Search (MADS) [3] under general nonlinear constraints. MADS, like pDS, belongs to the class of directional direct-search methods described in Section 1.1, and the main differences lie in the choice of polling directions and the decrease condition. The polling directions belong to positive spanning sets that asymptotically cover the unit sphere densely. Similarly to probabilistic direct search, the iterates and the mesh/step size are updated according to the outcome of the iteration, but new points are accepted based on a simple decrease of the function. Since its release in 2001, NOMAD has been improved to account for various features and extensions such as polling strategies, surrogate functions, and parallelism. In particular, the current default version includes a search step minimizing quadratic interpolation models based on previously evaluated poll points.

4.3. Classes of problems tested and profiles used

We considered 62 problems\(^2\) from the CUTEst library [33] for the smooth case, with dimensions between 2 and 51; Table 1 summarizes the distribution of the problems in terms of their dimensions.

In order to test the scalability of the methods, we selected 12 problems\(^3\) from the CUTEst library [33] that represent problems with different features (non-linearity, non-convexity, partial separability) and for which one can vary the dimension \(n\). For these problems, we considered dimensions \(n = 40\) and \(n = 80\).

For the non-smooth and noisy cases, we chose 53 problems from the test set described in [42]. The dimensions in this set vary from 2 to 12, and its distribution is summarized in Table 2. The problems in [42] come in three different perturbed forms: piecewise smooth, deterministic noise, and stochastic noise. In [42], the authors considered the multiplicative noise case, however, we also tested additive noise.

Additive noise. The function has the form \(f(x) = \phi(x) + \epsilon(x)\), where \(\phi\) is a smooth function. In the case of deterministic noise, \(\epsilon(x) = \epsilon(x)\), where \(\psi : x \in \mathbb{R}^n \rightarrow [-1, 1]^n\)

| Table 1. 62 problems from [33]. |
|---------------------------------|
| Dimension of the problem    | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 31 | 50 | 51 |
| Number of problems           | 21| 9 | 5 | 1 | 2 | 1 | 10| 1 | 5  | 2  | 4  | 1  |

| Table 2. 53 problems from [42]. |
|---------------------------------|
| Dimension of the problem    | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Number of problems           | 5 | 6 | 5 | 4 | 4 | 5 | 6 | 5 | 4  | 4  | 5  |
is a deterministic noisy function (see [42] for a full description of the function $\psi$). When considering stochastic noise, $\varepsilon(x)$ is a realization of a uniform random variable $U(-\epsilon_f, \epsilon_f)$.

**Multiplicative noise.** The function takes the form $f(x) = \phi(x)(1 + \xi(x))$ where $\phi$ is a smooth function. As in the additive noise case, when the noise is deterministic $\xi(x) = \epsilon_f \psi(x)$. In the case of stochastic noise, the values $\xi(x)$ are drawn from a uniform random variable as described earlier. The multiplicative noise case can be regarded as a special case of the additive noise case as $f(x) = \phi(x) + \varepsilon(x)$, where $\varepsilon(x) = \phi(x)\xi(x)$. When the true optimal objective function value is equal to 0, the noise is also decaying to 0 when converging to the optimum.

Performance profiles are a metric of comparison introduced in [28] to assess the performance of a given set of solvers $S$ for a given set of problems $P$. They are a visual tool where the highest curve (top and left) corresponds to the solver with the best overall performance. Let $t_{p,s} > 0$ be a performance measure of the solver $s \in S$ on the problem $p \in P$, which in our case was set to the number of function evaluations. The curve for a solver $s$ is defined as the fraction of problems where the performance ratio is at most $\alpha$,

$$\rho_s(\alpha) = \frac{1}{|P|} \text{size} \left\{ p \in P : r_{p,s} \leq \alpha \right\},$$

where the performance ratio $r_{p,s}$ is defined as

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}.$$

The convention $r_{p,s} = +\infty$ is used when a solver $s$ fails to satisfy the convergence test for problem $p$. The convergence test used is

$$f(x_0) - f(x) \geq (1 - \tau)(f(x_0) - f_L),$$

(24)

where $\tau > 0$ is a tolerance, $x_0$ is the starting point for the problem, and $f_L$ is computed for each problem $p \in P$ as the smallest value of $f$ obtained by any solver within a given number of function evaluations. Solvers with the highest values of $\rho_s(1)$ are the most efficient, and those with the highest values of $\rho_s(\alpha)$, for large $\alpha$, are the most robust.

Users with expensive optimization problems are often interested in the performance of solvers as the number of function evaluations increases. Specifically, they are interested in the percentage of problems that can be solved for a given tolerance and a given (small) budget of evaluations. Data profiles [42] are a tool to visualize such a behavior. A data profile of a solver $s \in S$ is defined as

$$d_s(\alpha) = \frac{1}{|P|} \text{size} \left\{ p \in P : \frac{t_{p,s}}{n_p + 1} \leq \alpha \right\},$$

(25)

where $n_p$ is the dimension of $p \in P$, and again, $t_{p,s}$ is the number of function evaluations taken to satisfy (24). Similarly to performance profiles, $t_{p,s} = +\infty$ when the solver $s$ fails to satisfy (24) for problem $p$ (using a given number of function evaluations). The budget of function evaluations in data profiles (25) is scaled by $n_p + 1$, which is the number of evaluations required to compute a simplex gradient. While performance profiles curves reflect relative solvers performance, data profiles display each curve independently of the others.
5. Numerical results

For performance profiles, the solvers were given a budget of $2000n$ function evaluations. A budget of $100(n + 1)$ was used for the data profiles in order to test the efficiency of the methods under small budgets of evaluations. The optimality tolerance parameters were chosen smaller for the non-noisy cases, $\tau \in \{10^{-2}, 10^{-5}\}$, and larger for the noisy ones, $\tau \in \{10^{-1}, 10^{-3}\}$, to account for the relative noise level of the problems.

5.1. Smooth problems

Figure 1 shows that DFO-TR (magenta downward triangles) is the most efficient method for the small smooth problems.

The Full-Low Evaluation method (blue squares) and BFGS-FD (black upward triangles) exhibit essentially the same performance, and this is because condition (3) is rarely violated for smooth problems. For low accuracy, NOMAD (green leftward triangles) has a comparable performance to Full-Low Evaluation and pDS (red circles) does perform quite well as expected. For the $n = 40$, 80 smooth problems (see Figure 2), the Full-Low Evaluation method appears to be both the most efficient and the most robust solver (performance similar to BFGS-FD as no Low-Eval iterations seem to occur). The performance of DFO-TR (magenta downward triangles) deteriorates with the dimension, perhaps due to ill-conditioning of the sample set. In all the tests for smooth

![Figure 1](image-url). Performance profiles (first row) and data profiles (second row) with $\tau = 10^{-2}, 10^{-5}$ for the 6 solvers. Results for the 62 small smooth problems from CUTEst [33].
problems, FDLM (light blue stars) delivers poor efficiency but strong robustness. NOMAD delivers the worst performance looking both at performance and data profiles.

5.2. Non-smooth problems

In Figure 3, we present the results on the piecewise smooth problems. One can see that the Full-Low Evaluation curve is above all, followed by the BFGS-FD, then NOMAD. On this set of problems, one can see that Full-Low Evaluation performs better that BFGS-FD, due to the presence of non-smoothness. In fact, the use of Low-Eval iterations has significantly improved the performance of BFGS-FD. Note that pDS and DFO-TR have the worst performance, but FDLM is not much better than these two. Note that none of these methods were designed to handle problems with non-smoothness. Looking at data profiles, Full-Low Evaluation solves the largest number of problems for both low and high accuracies for small budgets.

5.3. Noisy problems

We now compare the 6 solvers on the noisy problems described in Section 4.3. The noise level $\epsilon_f$ is chosen to be equal to $10^{-3}$ for both deterministic and stochastic cases.

From Figure 4, one can see that for additive deterministic noise the best performance is the one by DFO-TR, which is not surprising. For these problems, it is still the case...
Figure 3. Performance profiles (first row) and data profiles (second row) with $\tau = 10^{-2}, 10^{-5}$ for the 6 solvers. Results for the piecewise smooth problems in [42].

Figure 4. Performance profiles (first row) and data profiles (second row) with $\tau = 10^{-1}, 10^{-3}$ of the 6 solvers. Results for the additive deterministic noise problems (modified from [42]).
that Full-Low Evaluation has the same efficiency and robustness as BFGS-FD, due to the fact that BFGS-FD (i.e. always doing Full-Eval iterations) has a superior performance compared to pDS (i.e. always doing Low-Eval iterations). In the additive stochastic case (Figure 5), NOMAD is clearly the best in terms of efficiency and robustness. Even though BFGS-FD is the worst, Full-Low Evaluation is still robust compared to the remaining methods due to the incorporation of pDS in the Low-Eval iterations.

In the multiplicative deterministic case (see Figure 6), we can see that NOMAD is robust regardless of the accuracy. Full-Low Evaluation is a little more efficient for high accuracy. FDLM performs better than in the additive deterministic case. We can see that for high accuracy, Full-Low Evaluation performs better than just doing Full-Eval or Low-Eval iterations. In particular, by looking at data profiles, it solves a higher percentage of problems within small budgets.

The results for the stochastic multiplicative noise are given in Figure 7. The best solver is NOMAD for both high and low accuracy. This time, the curve corresponding to Full-Low Evaluation is no longer above the BFGS-FD and pDS, instead it is in between them. This is because BFGS-FD performs poorly when \( h \) is equal to the square root of machine precision (and the noise is ‘differentiated’). Note that FDLM exhibits the second best robustness for low accuracy even though it relies mostly on BFGS.

Taking into consideration all problem types, Full-Low Evaluation stands out as the best overall performer, when both efficiency and robustness are considered. For the noisy problems, we have also tried noise levels \( \epsilon_f = 10^{-2}, 10^{-4} \), but the relative positions of the curves in the profiles remain the same.
Figure 6. Performance profiles (first row) and data profiles (second row) with $\tau = 10^{-1}, 10^{-3}$ of the 6 solvers. Results for the multiplicative deterministic noise problems in [42].

Figure 7. Performance profiles (first row) and data profiles (second row) with $\tau = 10^{-1}, 10^{-3}$ for the 6 solvers. Results for the multiplicative stochastic noise problems in [42].
6. Concluding remarks

We introduced a new framework for unconstrained derivative-free optimization, consisting of the rigorous integration of two different methodologies. The goal was to combine the strengths of both methodologies in order to achieve solid numerical behavior (efficiency and robustness) regardless of the smoothness or noise regime of the objective function. The first methodology is related to the computation of fully linear models (in our case by computing finite difference gradients). The second is based on the low evaluation paradigm of probabilistic direct search. Our convergence analysis is novel in the way it connects the two methodologies, to rigorously extract their best properties.

The Full-Low Evaluation framework can be analyzed and implemented with other choices for the Full-Eval and Low-Eval iterations. For example, a trust-region step based on fully linear models [8] is a candidate for the Full-Eval iterations. Other possibilities include line search methods that use simplex gradients [36] or deterministic direct search (with complete polling on a set of points defined by a PSS) [39], instead of FD. Candidates for Low-Eval include, for instance, randomized Gaussian smoothing methods (one-point [45] or two-point approaches [29]).

There are a number of future research items to be investigated, and two stand out naturally. The extension to the constrained case, in particular to bound and linear constraints, and the consideration of DFO problems with larger dimensions. We plan to report on these topics in future manuscripts.

Notes
1. Density of search directions is not the single technique that leads to convergence results in non-smooth DFO. Other major approaches include the use of convex hulls of (possibly randomly) sampled approximate gradients [7,35,38] or trust-region models [41] with randomized linear terms.
2. ALLINITU, ARGLINB, ARGLINC, BARD, BEALE, BOX3, BRKMCC, BROWNAL, BROWNBS, BROWNDEN, CHNROSNB, CUBE, DECONVU, DENSCHNA, DENSCHNB, DENSCHNC, DENSCHND, DENSCHNF, DIXON3DQ, DJTL, ENGVAl2, ERRINROS, EXPFIT, EXTROSNB, GENHUMPS, GROWTH, GROWTHLS, HAIRY, HEART6LS, HEART8LS, HELIX, HILBERTA, HILBERTB, HIMMELBF, HIMMELBG, HUMPS, JENNSMP, KOWOSB, LOGHAIRY, MARATOSB, METHAN8, MEXHAT, MEYER3, NONMSQRT, PALMER1C, PALMER1D, PALMER2C, PALMER3C, PALMER4C, PALMER4E, PALMER5C, PALMER5D, PALMER6C, PALMER7C, PALMER8C, ROSENBR, SINEVAL, SISER, TOINTQOR, WATSON, YFITU, and ZANGWIL2.
3. ARGLINA, ARWHEAD, BROYDN3D, DQRTIC, ENGVAl1, FREUROTH, PENALTY2, NONDQUAR, ROSENBR, SINQUAD, TRIDIA, and WOODS.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

Support for this author was partially provided by the Centre for Mathematics of the University of Coimbra under grant FCT/MCTES UIDB/MAT/00324/2020.
Notes on contributors

A. S. Berahas is an Assistant Professor in the Department of Industrial and Operations Engineering at the University of Michigan. Prior to this appointment, he was a Postdoctoral Research Fellow at Lehigh University (2018-2020) and at Northwestern University (2018). He received his PhD in Engineering Sciences and Applied Mathematics in 2018 from Northwestern University. He received his undergraduate degree in Operations Research and Industrial Engineering from Cornell University in 2009, and in 2012 obtained an MSC degree in Engineering Sciences and Applied Mathematics from Northwestern University. His research broadly focuses on designing, developing and analyzing algorithms for solving large scale nonlinear optimization problems.

O. Sohab is a PhD candidate in the Department of Industrial and Systems Engineering at Lehigh University. She got her engineering degree (equivalent to a Bachelor and a Master) in Applied mathematics and Computer science from ENSEEIHT in Toulouse, France (2019) and a M.Sc in Performance in Software, Media and Scientific Computing from the same School (2019). Her research interest include nonlinear optimization and Machine learning.

L. N. Vicente is the Timothy J. Wilmott ’80 Endowed Faculty Professor and Chair of Lehigh University’s Department of Industrial and Systems Engineering (ISE), effective August 1, 2018, after a career as a faculty member in the Department of Mathematics of the University of Coimbra. He obtained his PhD from Rice University in 1996, under a Fulbright scholarship, receiving from Rice the Ralph Budd Thesis Award. He was one of the three finalists of the 94-96 A. W. Tucker Prize of the Mathematical Optimization Society (MOS). In 2015, he was awarded the Lagrange Prize of SIAM (Society for Industrial and Applied Mathematics) and MOS for the co-authorship of the book “Introduction to Derivative-Free Optimization, MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2009”.
His research interests include Continuous Optimization, Computational Science and Engineering, and Machine Learning and Data Science.

ORCID

A. S. Berahas  http://orcid.org/0000-0002-2371-9398
L. N. Vicente  https://orcid.org/0000-0003-1097-6384

References

[1] S. Alarie, C. Audet, A.E. Gheribi, M. Kokkolaras, and S. Le Digabel, Two decades of blackbox optimization applications, Euro. J. Comput. Optim. 9 (2021), p. 100011.
[2] C. Audet and J.E. Dennis Jr., Analysis of generalized pattern searches, SIAM J. Optim. 13 (2002), pp. 889–903.
[3] C. Audet and J.E. Dennis Jr., Mesh adaptive direct search algorithms for constrained optimization, SIAM J. Optim. 17 (2006), pp. 188–217.
[4] C. Audet and W. Hare, Derivative-Free and Blackbox Optimization, Springer Series in Operations Research and Financial Engineering, Springer, Cham, 2017.
[5] C. Audet, M. Kokkolaras, S. Le Digabel, and B. Talgorn, Order-based error for managing ensembles of surrogates in mesh adaptive direct search, J. Global Optim. 70 (2018), pp. 645–675.
[6] C. Audet, S.L. Digabel, V.R. Montplaisir, and C. Tribes, Nomad version 4: Nonlinear optimization with the mads algorithm, arXiv preprint arXiv:2104.11627 (2021).
[7] A.M. Bagirov, B. Karasözen, and M. Sezer, Discrete gradient method: Derivative-free method for nonsmooth optimization, J. Optim. Theory Appl. 137 (2008), pp. 317–334.
[8] A.S. Bandeira, K. Scheinberg, and L.N. Vicente, Computation of sparse low degree interpolating polynomials and their application to derivative-free optimization, Math. Program. 134 (2012), pp. 223–257.
[9] A.S. Berahas, R.H. Byrd, and J. Nocedal, Derivative-free optimization of noisy functions via quasi-Newton methods, SIAM J. Optim. 29 (2019), pp. 965–993.
[10] A.S. Berahas, L. Cao, K. Choromanski, and K. Scheinberg, A theoretical and empirical comparison of gradient approximations in derivative-free optimization, Found. Comput. Math. 22 (2021), pp. 507–560. https://doi.org/10.1007/s10208-021-09513-z

[11] A.S. Berahas, L. Cao, and K. Scheinberg, Global convergence rate analysis of a generic line search algorithm with noise, SIAM J. Optim. 31 (2021), pp. 1489–1518.

[12] S.C. Billups, J. Larson, and P. Graf, Derivative-free optimization of expensive functions with computational error using weighted regression, SIAM J. Optim. 23 (2013), pp. 27–53.

[13] A.J. Booker, J.E. Dennis Jr., P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset, A rigorous framework for optimization of expensive functions by surrogates, Struct. Multidiscip. Optim. 17 (1998), pp. 1–13.

[14] D.M. Bortz and C.T. Kelley, The simplex gradient and noisy optimization problems, in Computational Methods in Optimal Design and Control, Progress in Systems and Control Theory, Vol. 24, J.T. Borggaard, J. Burns, E. Cliff, and S. Schreck, eds., Birkhäuser, Boston, 1998, pp. 77–90.

[15] C.G. Broyden, The convergence of a class of double-rank minimization algorithms 1. General considerations, IMA J. Appl. Math. 6 (1970), pp. 76–90.

[16] F.H. Clarke, Optimization and Nonsmooth Analysis, John Wiley & Sons, New York, 1983. Reissued by SIAM, Philadelphia, 1990.

[17] A.R. Conn and S. Le Digabel, Use of quadratic models with mesh-adaptive direct search for constrained black box optimization, Optim. Methods Softw. 28 (2013), pp. 139–158.

[18] A.R. Conn and Ph.L. Toint, An algorithm using quadratic interpolation for unconstrained derivative free optimization, in Nonlinear Optimization and Applications, G. Di Pillo and F. Gianessi, eds., Plenum Publishing, New York, 1996, pp. 27–47.

[19] A.R. Conn, N.I.M. Gould, and Ph.L. Toint, Trust-Region Methods, MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2000.

[20] A.R. Conn, K. Scheinberg, and L.N. Vicente, Geometry of interpolation sets in derivative free optimization, Math. Program. 111 (2008), pp. 141–172.

[21] A.R. Conn, K. Scheinberg, and L.N. Vicente, Global convergence of general derivative-free trust-region algorithms to first and second order critical points, SIAM J. Optim. 20 (2009), pp. 387–415.

[22] A.R. Conn, K. Scheinberg, and L.N. Vicente, Introduction to Derivative-Free Optimization, MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2009.

[23] A.L. Custódio and L.N. Vicente, Using sampling and simplex derivatives in pattern search methods, SIAM J. Optim. 18 (2007), pp. 537–555.

[24] A.L. Custódio, K. Scheinberg, and L.N. Vicente, Methodologies and software for derivative-free optimization, in Chapter 37 of Advances and Trends in Optimization with Engineering Applications, MOS-SIAM Book Series on Optimization, T. Terlaky, M.F. Anjos, and S. Ahmed, eds., SIAM, Philadelphia, 2017.

[25] A.L. Custódio, H. Rocha, and L.N. Vicente, Incorporating minimum Frobenius norm models in direct search, Comput. Optim. Appl. 46 (2010), pp. 265–278.

[26] G. Deng and M.C. Ferris, Adaptation of the UOBYQA algorithm for noisy functions, in Proceedings of the 2006 Winter Simulation Conference, L.F. Perrone, F.P. Weiland, J. Liu, B.G. Lawson, D.M. Nicol, and R.M. Fujimoto, eds., 2006, pp. 312–319.

[27] M.A. Diniz-Ehrhardt, J.M. Martinez, and M. Raydan, A derivative-free nonmonotone line search technique for unconstrained optimization, J. Comput. Appl. Math. 219 (2008), pp. 383–397.

[28] E.D. Dolan and J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2002), pp. 201–213.

[29] J.C. Duchi, M.I. Jordan, M.J. Wainwright, and A. Wibisono, Optimal rates for zero-order convex optimization: The power of two function evaluations, IEEE Trans. Inform. Theory 61 (2015), pp. 2788–2806.

[30] G. Fasano, J.L. Morales, and J. Nocedal, On the geometry phase in model-based algorithms for derivative-free optimization, Optim. Methods Softw. 24 (2009), pp. 145–154.

[31] R. Fletcher, A new approach to variable metric algorithms, Comput. J. 13 (1970), pp. 317–322.

[32] D. Goldfarb, A family of variable-metric methods derived by variational means, Math. Comp. 24 (1970), pp. 23–26.
[33] N.I.M. Gould, D. Orban, and Ph. L. Toint, *CUTEst: A constrained and unconstrained testing environment with safe threads*, Comput. Optim. Appl. 60 (2015), pp. 545–557.

[34] S. Gratton, C.W. Royer, L.N. Vicente, and Z. Zhang, *Direct search based on probabilistic descent*, SIAM J. Optim. 25 (2015), pp. 1515–1541.

[35] W. Hare and J. Nutini, *A derivative-free approximate gradient sampling algorithm for finite minimax problems*, Comput. Optim. Appl. 56 (2013), pp. 1–38.

[36] C.T. Kelley, *Implicit Filtering*, SIAM, Philadelphia, 2011.

[37] J. Kiefer and J. Wolfowitz, *Stochastic estimation of the maximum of a regression function*, Ann. Math. Statist. 23 (1952), pp. 462–466.

[38] K.C. Kiwiel, *A nonderivative version of the gradient sampling algorithm for nonsmooth nonconvex optimization*, SIAM J. Optim. 20 (2010), pp. 1983–1994.

[39] T.G. Kolda, R.M. Lewis, and V. Torczon, *Optimization by direct search: New perspectives on some classical and modern methods*, SIAM Rev. 45 (2003), pp. 385–482.

[40] J. Larson, M. Menickelly, and S. Wild, *Derivative-free optimization methods*, Acta Numer. 28 (2019), pp. 287–404.

[41] G. Liuzzi, S. Lucidi, F. Rinaldi, and L.N. Vicente, *Trust-region methods for the derivative-free optimization of nonsmooth black-box functions*, SIAM J. Optim. 29 (2019), pp. 3012–3035.

[42] J.J. Moré and S.M. Wild, *Benchmarking derivative-free optimization algorithms*, SIAM J. Optim. 20 (2009), pp. 172–191.

[43] J.J. Moré and S.M. Wild, *Estimating computational noise*, SIAM J. Sci. Comput. 33 (2011), pp. 1292–1314.

[44] J.J. Moré and S.M. Wild, *Estimating derivatives of noisy simulations*, ACM Trans. Math. Softw. 38 (2012), pp. 1–21.

[45] Y. Nesterov, *Random gradient-free minimization of convex functions*, Tech. Rep. 2011/1, CORE, 2011.

[46] J. Nocedal and S.J. Wright, *Numerical Optimization*, 2nd ed., Springer-Verlag, Berlin, 2006.

[47] M.J.D. Powell, *Least Frobenius norm updating of quadratic models that satisfy interpolation conditions*, Math. Program. 100 (2004), pp. 183–215.

[48] M.J.D. Powell, *Developments of NEWUOA for minimization without derivatives*, IMA J. Numer. Anal. 28 (2008), pp. 649–664.

[49] L.M. Rios and N.V. Sahinidis, *Derivative-free optimization: a review of algorithms and comparison of software implementations*, J. Global Optim. 56(3) (2013), pp. 1247–1293.

[50] D.F. Shanno, *Conditioning of quasi-Newton methods for function minimization*, Math. Comp. 24 (1970), pp. 647–656.

[51] H.J.M. Shi, M.Q. Xuan, F. Oztoprak, and J. Nocedal, *On the numerical performance of derivative-free optimization methods based on finite-difference approximations*, arXiv preprint arXiv:2102.09762, 2021.

[52] J.C. Spall, *Multivariate stochastic approximation using a simultaneous perturbation gradient approximation*, IEEE Trans. Automat. Control 37 (1992), pp. 332–341.

[53] V. Torczon, *On the convergence of pattern search algorithms*, SIAM J. Optim. 7 (1997), pp. 1–25.

[54] L.N. Vicente and A.L. Custódio, *Analysis of direct searches for discontinuous functions*, Math. Program. 133 (2012), pp. 299–325.

[55] D. Winfield, *Function minimization by interpolation in a data set*, J. Inst. Math. Appl. 12 (1973), pp. 339–347.