Full-low evaluation methods for derivative-free optimization

A. S. Berahas* O. Sohab† L. N. Vicente‡

July 25, 2021

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 `Full-Low Evaluation` methods, organized around two main types of iterations.

The first iteration type 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 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 [1, 17, 18, 30, 38] 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, in particular whenever data is fitted with the purpose of finding optimal values for design or control variables of complex models or

*Department of Industrial & Operations Engineering, University of Michigan, 1205 Beal Avenue, Ann Arbor, MI 48109-2102, USA (`albertberahas@gmail.com`).

†Department of Industrial and Systems Engineering, Lehigh University, 200 West Packer Avenue, Bethlehem, PA 18015-1582, USA (`ous219@lehigh.edu`).

‡Department of Industrial and Systems Engineering, Lehigh University, 200 West Packer Avenue, Bethlehem, PA 18015-1582, USA. 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.

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

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

where $f : \mathbb{R}^n \to \mathbb{R}$ may be non-smooth and its evaluation subject to noise. In the smooth and non-noisy setting, the methods discussed and developed in this paper find application on problems where derivative information is not available or is too expensive to compute.

1.1 Advantages and disadvantages of the current DFO methods

When designing numerical algorithms for DFO [17], 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, 9, 26, 29, 42], 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 [9] 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, 43].

The directional direct-search framework has two main variants, a deterministic one [29, 42] and a probabilistic one [26]. 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 [29]. 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 $\mathcal{O}(n)$ 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 [26]. 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 [26]. 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 [28, 41], the authors considered fixed predetermined stepsize strategies 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 [5, 7, 10]. These methods utilize either a simplex gradient approximation or a FD approximation to the gradient of the objective function (leading to $\mathcal{O}(n)$ 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 [10]. Alternatively, optimal (in terms of minimizing the approximation error) FD approximations can be computed if the noise level is known, or can be estimated [32], and incorporated in a FD scheme [33]. The authors in [5] propose a FD quasi-Newton method with explicit noise estimation [32] 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 [14, 16, 22, 37, 44]. 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 [13], 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 [15]. 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 [8, 19] (see [30] for a review of recent developments on stochastic functions).

1.2 Can one design a rigorous DFO method that is robust for all function types?

As we have seen, 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.

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 $\mathcal{I}_{SF}$ and $\mathcal{I}_{SL}$, respectively. Similarly, the indices of unsuccessful iterations are denoted as $\mathcal{I}_{UF}$ and $\mathcal{I}_{UL}$, corresponding to the `Full-Eval` and `Low-Eval` iterations, respectively. The set of all iterations is denoted by $\mathcal{I}_{SF} \cup \mathcal{I}_{SL} \cup \mathcal{I}_{UF} \cup \mathcal{I}_{UL}$. 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. As said before, 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. 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-smoothnes 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).

In the general setting, the instance we consider for the `Full-Eval` step is of line-search nature, where the search direction p_k calculated based on an approximate gradient g_k. The conditions imposed on both p_k and g_k will be 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 β_k is a positive stepsize. As is typical in nonlinear optimization, the stepsize β_k is required to satisfy a sufficient decrease condition of the form

$$f(x_k + \beta p_k) \;\leq\; f(x_k) + c\,\beta g_k^\top p_k, \tag{2.1}$$

Algorithm 1 `Full-Low Evaluation` Algorithm

Initialization: Choose an initial iterate x_0. Set iteration i-type(0) = `Full-Eval`.

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

where $c \in (0, 1)$ is independent of k [35]. Under appropriate assumptions, condition (2.1) 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 β_k to become too small compared to a certain function of α_k, the direct-search stepsize used in `Low-Eval` iterations (described below). In fact we only consider a `Full-Eval` iteration successful if β_k satisfies

$$\beta \ \geq \ \gamma \rho(\alpha_k), \tag{2.2}$$

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 (2.2), 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. Backtracking parameters $\bar{\beta} > 0$ and $\tau \in (0, 1)$.
Output: i-type($k+1$), x_{k+1}, and α_{k+1}.

```
1: Compute an approximate gradient $g_k$.
2: Compute a direction $p_k$.
3: Backtracking line-search: Set $\beta = \bar{\beta}$. If (2.2) is false, stop.
4: While (2.1) is false
5:   Set $\beta = \tau\beta$.
6:   If (2.2) is false, set $x_{k+1} = x_k$ and i-type($k+1$) = Low-Eval, and stop the While cycle.
7: If (2.2) is true, set $\beta_k = \beta$, $x_{k+1} = x_k + \beta_k p_k$, and i-type($k+1$) = Full-Eval.
   (Note, the Low-Eval parameter $\alpha_k$ remains unchanged, $\alpha_{k+1} = \alpha_k$; see Algorithm 3.)
```

Our proposed `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 α_k is required to satisfy a sufficient decrease condition of the form

$$f(x_k + \alpha_k d_k) \ \leq \ f(x_k) - \rho(\alpha_k), \tag{2.3}$$

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 `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: i-type$(k+1)$, $x_{k+1}$, and $\alpha_{k+1}$.
1: Generate a finite set $D_k$ of non-zero polling directions.
2: If (2.3) 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 i-type$(k+1)$ = Low-Eval or if i-type$(k+1)$ = Full-Eval.
```

In order to switch from `Low-Eval` to `Full-Eval`, one can compare the number of unsuccessful consecutive `Low-Eval` iterations (nu_k) to the number of line-search backtracks (nb_{j_k}) done in the previous `Full-Eval` iteration (j_k). When nu_k becomes greater than nb_{j_k}, that is perhaps a sign that too much effort was put in the current `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 `Full-Eval` or `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 ∇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*

$$\|\nabla f(x_k) - g_k\| \;\leq\; u_g \beta_k \|g_k\|, \tag{3.1}$$

where $u_g > 0$ is independent of k.

Note that if $u_g\beta_k < 1$, Assumption 3.2 implies that the negative gradient approximation $-g_k$ is a descent direction. Finally, we state the assumptions on the `Full-Eval` directions.

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

$$\cos(-g_k, p_k) \;=\; \frac{(-g_k)^\top p_k}{\|g_k\|\|p_k\|} \;>\; \kappa \tag{3.2}$$

and

$$\|g_k\| \;\leq\; u_p \|p_k\| \,. \tag{3.3}$$

Condition (3.2) is classical in line-search methods, 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 (3.1) and (3.2) together imply that p_k is a descent direction. Condition (3.3) is also mild. In a Newton or quasi-Newton context, it is essentially saying that the Hessian or secant matrix is bounded (which is what is imposed in trust-region methods). One can relax (3.3) 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 β_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

$$\begin{aligned} f(x_k + \beta p_k) &\leq f(x_k) + \beta p_k^\top \nabla f(x_k) + \frac{\beta^2}{2} L \|p_k\|^2 \\ &\leq f(x_k) + \beta p_k^\top ((\nabla f(x_k) - g_k) + g_k) + \frac{\beta^2}{2} L \|p_k\|^2 \\ &\leq f(x_k) + \beta g_k^\top p_k + \left(\frac{1}{2}L + u_g u_p\right) \beta^2 \|p_k\|^2 , \end{aligned} \tag{3.4}$$

where the last line follows from applying (3.1) and (3.3).

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

$$f(x_k + (\beta_k/\tau)p_k) - f(x_k) \;>\; c\,(\beta_k/\tau) g_k^\top p_k. \tag{3.5}$$

We can now plug $\beta = \beta_k/\tau$ into (3.4), and combine it with (3.5), 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_{low} be a lower bound). Then,*

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

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

$$M \;=\; c\min\left\{\bar{\beta}\frac{\kappa}{u_p}, \frac{2\tau\kappa^2(1-c)}{2u_g u_p + L}\right\}. \tag{3.6}$$

Proof. Let $k \in \mathcal{I}_{SF}$. First, using (3.2) and (3.3), we write

$$\frac{((-g_k)^\top p_k)^2}{\|p_k\|^2} \;\geq\; \kappa^2\|g_k\|^2 \quad \text{and} \qquad (-g_k)^\top p_k \;\geq\; \frac{\kappa}{u_p}\|g_k\|^2. \tag{3.7}$$

Then, from the sufficient decrease condition (2.1) and the lower bound for the stepsize established in Lemma 3.1, one obtains

$$f(x_k) - f(x_{k+1}) \geq M\,\|g_k\|^2\,, \tag{3.8}$$

where M is given in (3.6).

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

$$\frac{1}{\bar{\beta}u_g + 1}\,\|\nabla f(x_k)\| \;\leq\; \|g_k\|\,.$$

Hence, plugging this bound into inequality (3.8) yields

$$f(x_k) - f(x_{k+1}) \;\geq\; \frac{M}{(\bar{\beta}u_g+1)^2}\,\|\nabla f(x_k)\|^2\,.$$

In both unsuccessful `Full-Eval` and `Low-Eval` iterations, the function value does not decrease, and thus

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

The proof is completed by using the lower bound $f(x_k) \geq f_{low}$. □

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 proceeds as follows. First, using $x = x_k$ and $y = x_*$ (the unique minimizer of f), one has

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

Giving that

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

one arrives at

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

At this point it remains to combine this last inequality with (3.8), and recursively conclude that (when $k-1 \in \mathcal{I}_{SF}$)

$$f(x_k) - f(x_*) \;\leq\; (1-\eta_\mu)^{ns_k}(f(x_0) - f(x_*))$$

with $\eta_\mu = \frac{2\mu c}{(\bar{\beta} u_g + 1)^2} \min\left(\bar{\beta}\frac{\kappa}{u_p}, \frac{2\tau\kappa^2(1-c)}{2u_g u_p + L}\right)$, and again $ns_k = |\mathcal{I}_{SF} \cap \{0,\ldots,k-1\}|$. 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 `Full-Eval` iterations.

Assumption 3.4 (`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 (2.2) allows us to prove under this assumption that the `Low-Eval` iterations generate an infinite subsequence of iterates driving the direct-search parameter α_k to zero. This result requires the forcing function to satisfy Assumption 3.5.

Assumption 3.5 *The function ρ 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 `Full-Eval` steps are essentially considered as *search steps* from the direct-search perspective of the `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 `Low-Eval` iterates for which*

$$\lim_{k\in\mathcal{K}} x_k \;=\; x_* \quad \textit{and} \quad \lim_{k\in\mathcal{K}} \alpha_k = 0.$$

Proof. First, consider that there is an infinity of iterations in $\mathcal{I}_{SF} \cup \mathcal{I}_{UF} \cup \mathcal{I}_{SL}$. These are all iterations k for which α_k does not decrease. By Assumption 3.3, all successful `Full-Eval` iterations $k \in \mathcal{I}_{SF}$ yield

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

From the fact that β_k satisfies $\beta_k \geq \gamma\rho(\alpha_k)$,

$$f(x_{k+1}) - f(x_k) \;\leq\; -c\,\kappa\, u_p \gamma \epsilon_g \rho(\alpha_k). \tag{3.9}$$

Successful `Low-Eval` iterations $k \in \mathcal{I}_{SL}$ achieve sufficient decrease,

$$f(x_{k+1}) - f(x_k) \;\leq\; -\rho(\alpha_k). \tag{3.10}$$

Note that in `Full-Eval` unsuccessful iterations $k \in \mathcal{I}_{UF}$ neither x_k nor α_k changes.

Hence, given that for unsuccessful `Low-Eval` iterations ($\mathcal{I}_{UL}$) the function does not decrease, we can sum from 0 to $k \in \mathcal{I}_{SF} \cup \mathcal{I}_{UF} \cup \mathcal{I}_{SL}$ the inequalities (3.9) and (3.10) to obtain

$$f(x_0) - f(x_{k+1}) \;\geq\; c\,\gamma\,\kappa\, u_p \epsilon_g \sum_{i\in\mathcal{I}_{SF}} \rho(\alpha_i) + \sum_{i\in\mathcal{I}_{SL}} \rho(\alpha_i).$$

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

It remains to consider the iterations in $\mathcal{I}_{UL}$. For each $k \in \mathcal{I}_{UL}$ corresponding to an unsuccessful `Low-Eval` iteration, consider the previous iteration $k' = k'(k) \in \mathcal{I}_{SL} \cup \mathcal{I}_{UF} \cup \mathcal{I}_{SL}$ 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 $k \in \mathcal{I}_{UL}$.

We have thus proved that α_k goes to zero for all k. Since α_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 $\mathcal{K}$ of that subsequnce satisfying the statement of the lemma. $\square$

The main theorem uses the notion of generalized Clarke derivative [12] at x along a direction $d \in \mathbb{R}^n$

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

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, 43].

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_* defined in Lemma 3.2. Let the set of limit points of*

$$\left\{ \frac{d_k}{\|d_k\|},\; d_k \in D_k, k \in \mathcal{K} \right\} \tag{3.11}$$

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

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

Proof. Let $\bar{d}$ be a limit point of (3.11), identified for a certain subsequence $\mathcal{L} \subseteq \mathcal{K}$. Then, from basic properties of the generalized Clarke derivative, and $k \in \mathcal{L}$,

$$\begin{aligned} f^\circ(x_*; \bar{d}) &= \lim_{x_k \to x_*} \sup_{\alpha_k \downarrow 0} \frac{f(x_k + \alpha_k d_k) - f(x_k)}{\alpha_k} \\ &\geq \lim_{x_k \to x_*} \sup_{\alpha_k \downarrow 0} \left\{ \frac{f(x_k + \alpha_k d_k) - f(x_k)}{\alpha_k} + \frac{\rho(\alpha_k)}{\alpha_k} \right\}. \end{aligned}$$

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 $f^\circ(x_*; \bar{d}) \geq 0$. Given the continuity of $f^\circ(x_*; \cdot)$, one has for any $d \in \mathbb{R}^n$ such that $\|d\| = 1$, $f^\circ(x_*; d) = \lim_{\bar{d} \to d} f^\circ(x_*; \bar{d}) \geq 0$. $\square$

3.3 More on the non-smooth case (noise)

Let us now consider $f = \phi + \varepsilon$, where $\phi : \mathbb{R}^n \to \mathbb{R}$ is a smooth function and $\varepsilon : \mathbb{R}^n \to \mathbb{R}$ is some additive noisy function. We assume that both ϕ and ε are Lipschitz continuous around x_*. The Clarke stationarity of ϕ at x_* is guaranteed if ε satisfies

$$\varepsilon^\circ(x_*; d) \;\leq\; -\sigma\phi^\circ(x_*; d), \tag{3.12}$$

for some $\sigma \in [0, 1)$. In fact, applying Theorem 3.2,

$$f^\circ(x_*; d) \;=\; \phi^\circ(x_*; d) + \varepsilon^\circ(x_*; d) \;\geq\; 0,$$

and then using (3.12), it follows that $\phi^\circ(x_*; d) \geq 0$. In particular the result is true if $\varepsilon^\circ(x_*; d) \leq 0$.

In the multiplicative noise case, where $f(x) = \phi(x)(1 + \varepsilon(x))$, one can still establish similar results. In this case we can write f as a sum, $f(x) = \phi(x) + \phi(x)\varepsilon(x)$. If ϕ and ε are Lipschitz continuous around x_*, then it follows that $\phi\varepsilon$ is also Lipschitz continuous around x_*. Applying Theorem 3.2,

$$\phi^\circ(x_*; d)(1 + \varepsilon(x_*)) + \phi(x_*)\varepsilon^\circ(x_*; d) \;\geq\; 0.$$

If $\phi(x_*) > 0$, then the Clarke stationarity of ϕ at x_*, would result from

$$\varepsilon^\circ(x_*; d) \;\leq\; -\sigma\,\frac{\phi^\circ(x_*; d)(1 + \varepsilon(x_*))}{\phi(x_*)},$$

for some $\sigma \in [0, 1)$, and as long as $1 + \varepsilon(x_*) > 0$. The same happens in the case $\phi(x_*) < 0$, if

$$\varepsilon^\circ(x_*; d) \;\geq\; -\sigma\,\frac{\phi^\circ(x_*; d)(1 + \varepsilon(x_*))}{\phi(x_*)}.$$

3.4 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 [16, 17] 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

$$[\nabla_{h_k} f(x_k)]_i \;=\; \frac{f(x_k + h_k e_i) - f(x_k)}{h_k}, \quad i = 1, \ldots, n, \tag{3.13}$$

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. By using a Taylor expansion, the error in the FD gradient (in the smooth and noiseless setting) can be shown to satisfy

$$\|\nabla f(x_k) - \nabla_{h_k} f(x_k)\| \;\leq\; \frac{1}{2}\sqrt{n}\,L\,h_k. \tag{3.14}$$

It becomes then clear that one way to ensure Assumption 3.2 in practice is to enforce $h_k \leq u_g'\beta\|\nabla_{h_k} f(x_k)\|$, 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 $h_k > u'_g\beta \left\|\nabla_{h_k} f(x_k)\right\|$

```
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 $h_k > u'_g\beta \left\|\nabla_{h_k} f(x_k)^{(j)}\right\|$ Do
2:     Set $j = j + 1$ and $h_k = \omega^j u'_g \beta \left\|\nabla_{h_k} f(x_k)^{(0)}\right\|$.
3:     Compute the FD approximation $\nabla_{h_k} f(x_k)^{(j)}$ using (3.13).
```

It can be proven that Algorithm 4 terminates in a finite number of steps (see Proposition 3.1).

Proposition 3.1 *Condition $h_k \le u'_g\beta\|\nabla_{h_k} f(x_k)\|$ 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 $j \ge 1$, using Step 3 and the satisfaction of the while–condition in Step 1,

$$\left\|\nabla_{h_k} f(x_k)^{(j)}\right\| \;\le\; \omega^j \left\|\nabla_{h_k} f(x_k)^{(0)}\right\|. \tag{3.15}$$

On the other hand, for all $j \ge 1$, the FD bound (3.14), followed by Step 3, gives us

$$\left\|\nabla f(x_k) - \nabla_{h_k} f(x_k)^{(j)}\right\| \;\le\; \frac{1}{2}\sqrt{n}L\,\omega^j u'_g\beta_k \left\|\nabla_{h_k} f(x_k)^{(0)}\right\|. \tag{3.16}$$

Hence, using (3.15)–(3.16), we have

$$\|\nabla f(x_k)\| \;\le\; \left\|\nabla f(x_k) - \nabla_{h_k} f(x_k)^{(j)}\right\| + \left\|\nabla_{h_k} f(x_k)^{(j)}\right\| \;\le\; \left(\frac{\sqrt{n}Lu'_g\beta_k}{2} + 1\right)\omega^j \left\|\nabla_{h_k} f(x_k)^{(0)}\right\|.$$

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 16GB of RAM and an Intel Core i7-8565U processor running at 1.80GHz.

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 (3.13) 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 [11, 23, 24, 39]. The choice of BFGS is justified by its strong performance in the presence of derivatives [35]. 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 [5, 6, 40]. Our `Full-Eval` line-search iteration is described in Algorithm 5.

BFGS updates the inverse Hessian approximation H_k using (4.3), where j_k is the previous `Full-Eval` iteration, and s_k and y_k are given in (4.2). 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 $\tau = 0.5$. As explained and rigorously proved in our paper, a key feature of our `Full-Low Evaluation` methodology is to stop the line-search once condition (2.2) is violated. In our implementation, we used

$$\gamma \ = \ 1, \quad \rho(\alpha_k) \ = \ \min(\gamma_1, \gamma_2 \alpha_k^2), \quad \text{with} \quad \gamma_1 = 10^{-5} \quad \text{and} \quad \gamma_2 = 10^{-3}. \tag{4.1}$$

When $k = 0$, we perform a backtracking line-search using $p_0 = -g_0$ (and update i-type(1) and x_1) as in Algorithm 2 (with constants as in Algorithm 5). The initialization of H_0 is done as follows. If i-type(1) = `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 [35]). However, if i-type(1) = `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 $(x_{j_k}, g_{j_k}, H_{j_k})$ 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**: i-type($k+1$) 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 (3.13).
2: Set
       $$s_k = x_k - x_{j_k} \quad \text{and} \quad y_k = g_k - g_{j_k}. \tag{4.2}$$
3: **If** $s_k^\top y_k \geq \epsilon_c \|s_k\| \|y_k\|$, set
       $$H_k \ = \ \left(I - \frac{s_k y_k^\top}{y_k^\top s_k}\right) H_{j_k} \left(I - \frac{y_k s_k^\top}{y_k^\top s_k}\right) + \frac{s_k s_k^\top}{y_k^\top s_k}. \tag{4.3}$$
4: **Else**, set $H_k = H_{j_k}$.
5: Compute the direction $-H_k g_k$.
6: Perform a backtracking line-search, and update i-type($k+1$) 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 [26]). 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 (4.1). 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_{j_k} 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 i-type$(k-1)$ = Full-Eval.) Number $nb_{j_k}$ of backtracks done at Full-Eval iteration $j_k$.
Output: i-type$(k+1)$, $x_{k+1}$, $\alpha_{k+1}$, 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 $nu_k < nb_{j_k}$, i-type$(k+1)$ = Low-Eval.
5: Else, i-type$(k+1)$ = 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` [4]; (iv) a line-search BFGS method based on FD gradients and noise estimation, `FDLM` [5].

`DFO-TR` [4] 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 $x_0 \pm \Delta_0 e_i$, with e_i the i-th canonical vector. Until the cardinal number of the sample set reaches $p_{\max} = (n+1)(n+2)/2$, all trust-region trial points $x_k + s_k$ are added to the sample set, and models are computed by minimum Frobenius norm interpolation [17, 36]. Once there are p_{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 Δ_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 Δ_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` [5] is a linesearch 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 [32], and such estimate is used to select the FD parameter as suggested in [33]. Second, in [5], 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 [5] developed a recovery procedure that estimates the noise only as needed (as the optimization progresses). The `FDLM` solver 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.

4.3 Classes of problems tested and profiles used

We considered 62 problems[1] from the CUTEst library [25] for the smooth case, with dimensions between 2 and 51; Table 1 summarizes the distribution of the problems in terms of their dimensions.

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 1: 62 problems from [25].

In order to test the scalability of the methods, we selected 12 problems[2] from the CUTEst library [25] 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 [31]. The dimensions in this set vary from 2 to 12, and its distribution is summarized in Table 2. The problems in [31] come in three different perturbed forms: piecewise smooth, deterministic noise, and stochastic noise. In [31], the authors considered the multiplicative noise case, however, we also tested additive noise.

Additive noise. The function has the form $f(x) = \phi(x) + \varepsilon(x)$, where ϕ is a smooth function. In the case of deterministic noise, $\varepsilon(x) = \epsilon_f \psi(x)$, where $\psi : x \in \mathbb{R}^n \to [-1, 1]^n$ is a deterministic noisy function (see [31] for a full description of the function ψ). 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 ϕ 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.

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

Table 2: 53 problems from [31].

Performance profiles are a metric of comparison introduced in [20] to assess the performance of a given set of solvers $\mathcal{S}$ for a given set of problems $\mathcal{P}$. 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 $s \in \mathcal{S}$ on the problem $p \in \mathcal{P}$, which in our case was

[1] 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, JENSMP, KOWOSB, LOGHAIRY, MARATOSB, METHANB8, MEXHAT, MEYER3, NONMSQRT, PALMER1C, PALMER1D, PALMER2C, PALMER3C, PALMER4C, PALMER4E, PALMER5C, PALMER5D, PALMER6C, PALMER7C, PALMER8C, ROSENBR, SINEVAL, SISSER, TOINTQOR, WATSON, YFITU, and ZANGWIL2.

[2] ARGLINA, ARWHEAD, BROYDN3D, DQRTIC, ENGVAL1, FREUROTH, PENALTY2, NONDQUAR, ROSENBR, SINQUAD, TRIDIA, and WOODS.

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 α,

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

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

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

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

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

where $\tau > 0$ is a tolerance, x_0 is the starting point for the problem, and f_L is computed for each problem $p \in \mathcal{P}$ 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 α, are the most robust.

5 Numerical results

In the tests performed, all solvers were given a budget of $2000n$ function evaluations. Comparisons are based on two values of the performance profile optimality tolerance parameter $\tau \in \{10^{-2}, 10^{-5}\}$.

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 (2.2) is rarely violated for smooth problems. For low accuracy, `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`). 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 problems, `FDLM` (light blue stars) delivers poor efficiency but strong robustness.

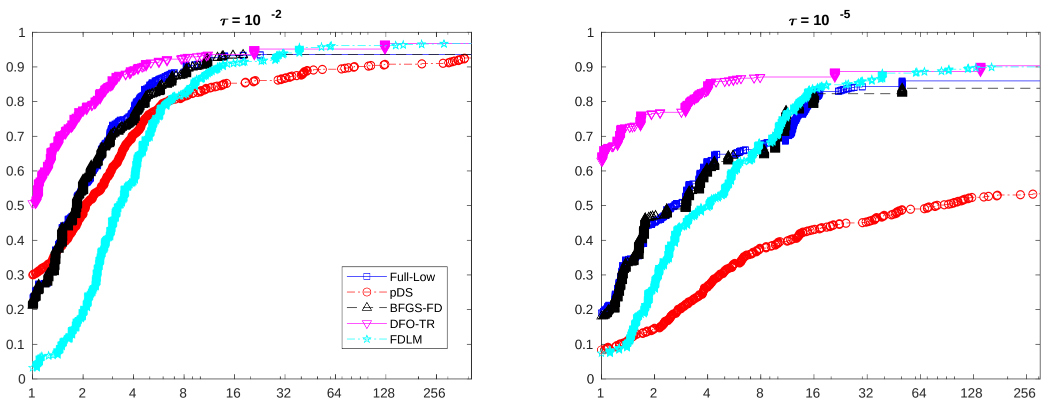

Figure 1: Performance profiles with $\tau = 10^{-2}, 10^{-5}$ for the 5 solvers. Results for the 62 small smooth problems from CUTEst *[25].*

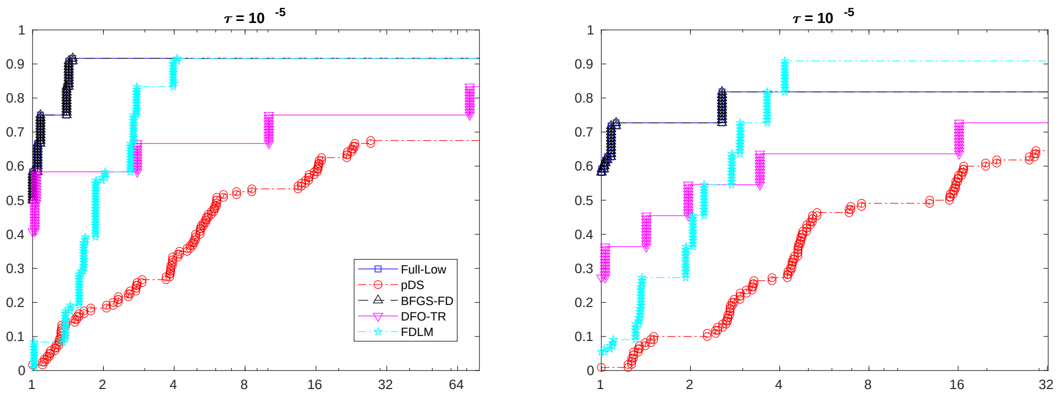

Figure 2: Performance profiles with $\tau = 10^{-5}$ for the 5 solvers. Results for the 12 not-so-small smooth problems from CUTEst *[25] ($n = 40$ on left and $n = 80$ on the right).*

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. 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.

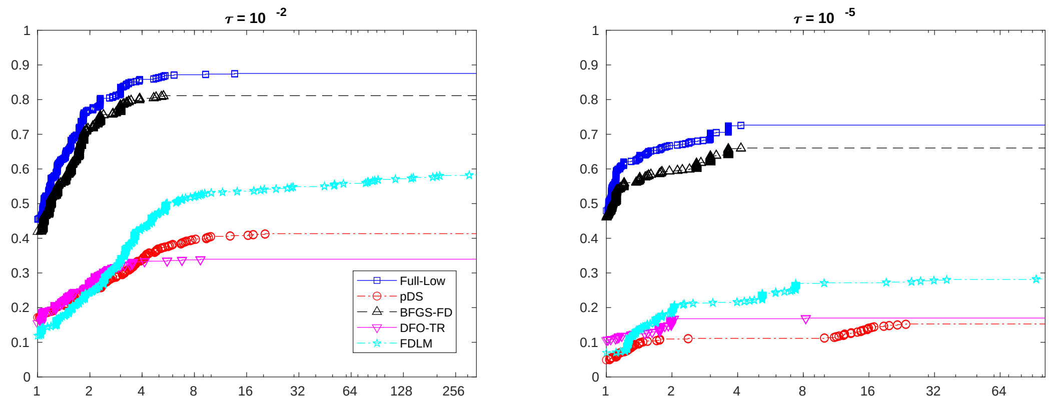

Figure 3: Performance profiles with $\tau = 10^{-2}, 10^{-5}$ for the 5 solvers. Results for the piecewise smooth problems in [31].

5.3 Noisy problems

We now compare the 5 solvers on the noisy problems described in Section 4.3. The noise level ϵ_f is chosen to be equal to 10^{-3} for both deterministic and stochastic cases.

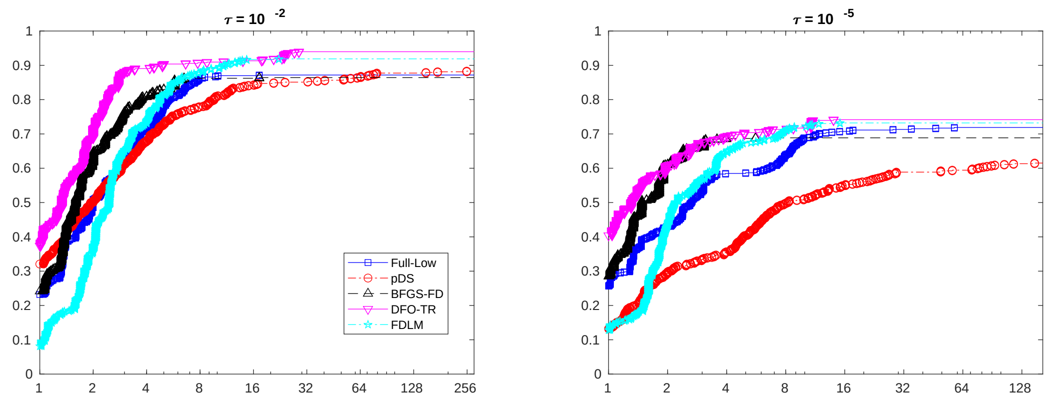

Figure 4: Performance profiles with $\tau = 10^{-2}, 10^{-5}$ of the 5 solvers. Results for the additive deterministic noise problems (modified from [31]).

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 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), `FDLM` is clearly the best in terms of 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.

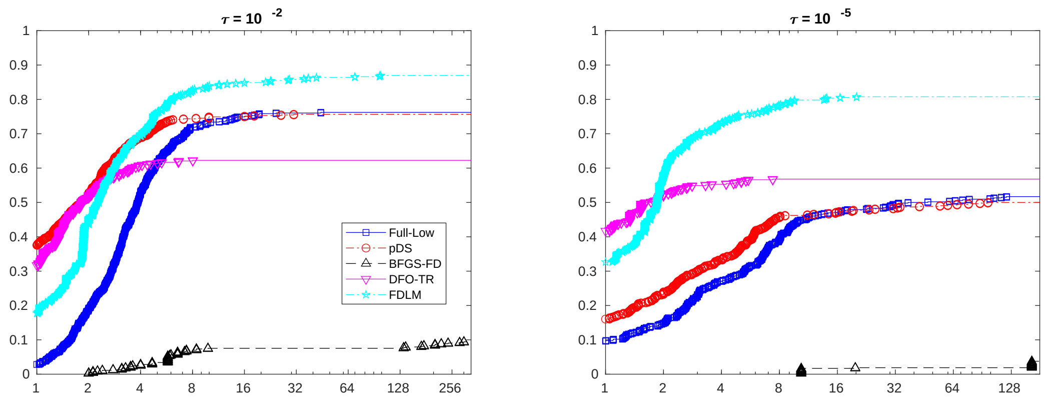

Figure 5: Performance profiles with $\tau = 10^{-2}, 10^{-5}$ of the 5 solvers. Results for the additive stochastic noise problems (modified from [31]).

In the multiplicative deterministic case (see Figure 6), we can see that `Full-Low Evaluation` is both efficient and robust regardless of the accuracy. `DFO-TR` is a little more efficient for low accuracy. `FDLM` performs better than in the additive deterministic case. We can see that in this case, `Full-Low Evaluation` performs better than just doing `Full-Eval` or `Low-Eval` iterations.

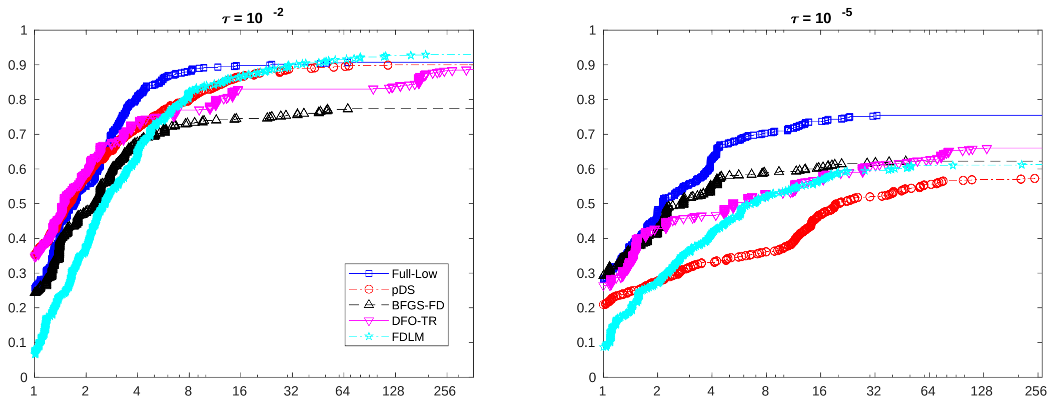

Figure 6: Performance profiles with $\tau = 10^{-2}, 10^{-5}$ of the 5 solvers. Results for the multiplicative deterministic noise problems in [31].

The results for the stochastic multiplicative noise are given in Figure 7. The best solver in terms of efficiency is `pDS` 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 best robustness for low accuracy even though it relies mostly on BFGS.

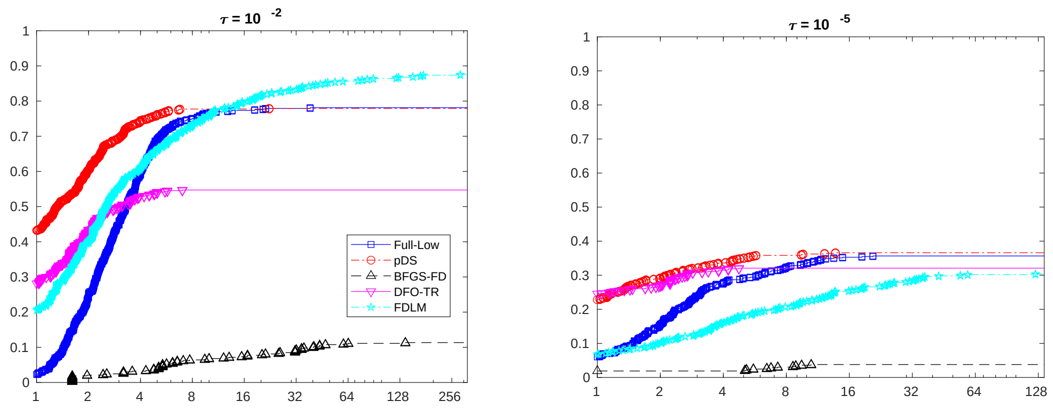

Figure 7: Performance profiles with $\tau = 10^{-2}, 10^{-5}$ of the 5 solvers. Results for the multiplicative stochastic noise problems in [31].

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.

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 [4] is a candidate for the `Full-Eval` iterations. Other possibilities include line search methods that use simplex gradients [27] or deterministic direct search (with complete polling on a set of points defined by a PSS) [29], instead of FD. Candidates for `Low-Eval` include, for instance, randomized Gaussian smoothing methods (one-point [34] or two-point approaches [21]).

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.

References

[1] C. Audet and W. Hare. *Derivative-Free and Blackbox Optimization.* Springer Series in Operations Research and Financial Engineering. Springer, Cham, Switzerland, 2017.

[2] C. Audet and J. E. Dennis Jr. Analysis of generalized pattern searches. *SIAM J. Optim.*, 13:889–903, 2002.

[3] C. Audet and J. E. Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM J. Optim.*, 17:188–217, 2006.

[4] 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.*, 134:223–257, 2012.

[5] A. S. Berahas, R. H. Byrd, and J. Nocedal. Derivative-free optimization of noisy functions via quasi-Newton methods. *SIAM J. Optim.*, 29:965–993, 2019.

[6] A. S. Berahas, L. Cao, K. Choromanski, and K. Scheinberg. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Foundations of Computational Mathematics*, DOI: 10.1007/s10208-021-09513-z, 2021.

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

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

[9] 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. *Structural and Multidisciplinary Optimization*, 17:1–13, 1998.

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

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

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

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

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

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

[16] 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.*, 20:387–415, 2009.

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

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

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

[20] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91:201–213, 2002.

[21] 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 Transactions on Information Theory*, 61:2788–2806, 2015.

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

[23] R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13:317–322, 1970.

[24] D. Goldfarb. A family of variable-metric methods derived by variational means. *Math. Comp.*, 24:23–26, 1970.

[25] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEst: a Constrained and Unconstrained Testing Environment with safe threads. *Comput. Optim. Appl.*, 60:545–557, 2015.

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

[27] C. T. Kelley. *Implicit filtering.* SIAM, 2011.

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

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

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

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

[32] J. J. Moré and S. M. Wild. Estimating computational noise. *SIAM J. Sci. Comput.*, 33:12921314, 2011.

[33] J. J. Moré and S. M. Wild. Estimating derivatives of noisy simulations. *ACM Trans. Math. Software*, 38:1–21, 2012.

[34] Y. Nesterov. Random gradient-free minimization of convex functions. Technical Report 2011/1, CORE, 2011.

[35] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, Berlin, second edition, 2006.

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

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

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

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

[40] 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.

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

[42] V. Torczon. On the convergence of pattern search algorithms. *SIAM J. Optim.*, 7:1–25, 1997.

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

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