

# A derivative-free trust-region approach for Low Order-Value Optimization problems\*

 A. E. Schwertner<sup>†</sup>       F. N. C. Sobral<sup>‡</sup>

25th November 2025

## Abstract

The Low Order-Value Optimization (LOVO) problem involves minimizing the minimum among a finite number of function values within a feasible set. LOVO has several practical applications such as robust parameter estimation, protein alignment, portfolio optimization, among others. In this work, we are interested in the constrained nonlinear optimization LOVO problem of minimizing the minimum between a finite number of function values subject to a nonempty closed convex set where each function is a black-box and continuously differentiable, but the derivatives are not available. We develop the first derivative-free trust-region algorithm for constrained LOVO problems with convergence to weakly critical points. Under suitable conditions, we establish the global convergence of the algorithm and also its worst-case iteration complexity analysis. An initial open-source implementation using only linear interpolation models is developed. Extensive numerical experiments and comparison with existing alternatives show the properties and the efficiency of the proposed approach when solving LOVO problems.

**Keywords:** Derivative-Free Optimization, Trust-Region Methods, Low Order-Value Optimization, Worst-case Iteration Complexity

## 1 Introduction

Let us consider the following constrained nonlinear optimization problem


$$\min_{x \in \Omega} f_{\min}(x) = \min_{x \in \Omega} \min\{f_1(x), \dots, f_r(x)\}, \quad (1)$$

where  $\Omega \subset \mathbb{R}^n$  is a nonempty closed convex set and  $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $i = 1, \dots, r$ , are continuously differentiable black-box functions. Although we consider that each function that composes the objective function is smooth, we assume that there is no information available about its first and second-order derivatives.

---

\*This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001

<sup>†</sup>Graduate Program in Mathematics, State University of Maringá, Paraná, Brazil. [schwertner.ae@gmail.com](mailto:schwertner.ae@gmail.com)

<sup>‡</sup> Corresponding author, Department of Mathematics, State University of Maringá, Paraná, Brazil. [fncsobral@uem.br](mailto:fncsobral@uem.br)

Problem (1) is known as the Low Order-Value Optimization problem (LOVO), for which extensive theory and application exist [4, 34]. Applications include protein alignment [3, 4, 33, 35], robust parameter estimation [4, 10], portfolio optimization [7, 33], robust response surface methodology [39], among others. In recent years, several authors have developed LOVO versions of classical continuous optimization methods, such as the Levenberg-Marquardt [10], Augmented Lagrangian [4], Trust-Region [3] and Coordinate-Search [33] algorithms.

In this article, we address derivative-free optimization, which has several scientific, medical, social, and engineering applications [5, 16, 19, 31], specially in situations where the objective function is black-box, derived from a simulation, or without a known analytical expression. Detailed surveys of derivative-free optimization methods can be found in [31, 44], and theoretical aspects can be revisited in [5, 19].

Among the various algorithms developed for derivative-free optimization, variations of the trust-region method stand out. The study of such methods began with Winfield [54, 55] and became popular due to the good numerical performance of the algorithms proposed by Powell [41, 42, 43]. There are several derivative-free trust-region algorithms described in the literature, which can be applied to unconstrained problems [17, 18, 20, 42, 53], bound-constrained [26, 41, 51], linearly-constrained [27, 40], convex-constrained [12, 29, 50], and general-constrained [8, 11, 21, 49] problems, as well as composite nonsmooth optimization [22, 24, 32], and multi-objective optimization [47] problems, among others.

The goal of this work is to develop and analyse a derivative-free optimization algorithm designed for convex-constrained LOVO problems. Note that the LOVO problem can be understood as a particular case of nonsmooth composite optimization. In this sense, Garmanjani, Júdeice and Vicente [22] and Grapiglia, Yuan and Yuan [24], proposed model-based derivative-free algorithms for objective functions of the form  $f + h \circ g$ , where  $f$  and  $g$  are smooth, and  $h$  is convex. However, such methods are not compatible with the specificities of the LOVO problem since the minimum function is not convex. Furthermore, the construction of the models involves the evaluation of  $h \circ g$  at each one of the points in the sample set, which translates into higher computational costs when compared to the strategy adopted in our approach. Larson, Menickelly and Zhou [32] proposed a manifold sampling algorithm for minimizing nonsmooth compositions  $g \circ f$ , where  $g$  is nonsmooth,  $f$  is smooth with no information about its derivatives, and  $g \circ f$  is a continuous selection. From a theoretical point of view, the algorithm generates a sequence of iterates that converge to a Clarke stationary point. However, in [33] it is shown that the concept of Clarke stationarity is not sufficiently satisfactory for LOVO problems, which makes it necessary to adopt methods that are probably to converge to points that satisfy stronger optimality conditions, such as [33, Theorem 6.1].

Hough and Roberts [29] and Roberts [45] define a suitable notion of  $\Lambda$ -poised interpolation models and show convergence and worst-case iteration complexity results for general convex-constrained derivative-free optimization problems. The algorithm is strongly based in the method proposed in [12]. Recently, convergence and worst-case iteration complexity was also discussed in [1] in the context of convex-constrained LOVO problems based on derivatives.

The main contributions of the present work can be stated as

- We develop a trust-region derivative-free algorithm for convex-constrained LOVO problems, which is suitable when the projection onto the convex set is available. This in turn, is an extension of [3] to the derivative-free constrained case;
- Inexact interpolation models are allowed, as well as the use of two different radii [42], improving the results of [50];
- Converge theory and worst-case iteration complexity are provided, showing that the behaviour of the algorithm follows the usual trust-region framework;
- An efficient and open-source numerical implementation is provided and is competitive against algorithms for nonsmooth derivative-free problems.

This work is organized as follows. In Section 2, we introduce our derivative-free trust-region algorithm for LOVO problems. In Section 3, we discuss its convergence analysis and global convergence results. In Section 4, we study the worst-case complexity of our algorithm. Implementation details and numerical experiments are discussed in Section 5, and conclusions are given in Section 6.

**Notation**  $\|\cdot\|$  denotes the Euclidean norm of vectors and matrices;  $\mathcal{P}_2^n$  is the set of all polynomials of degree less than or equal to 2 in  $\mathbb{R}^n$ ; and  $\overline{B}(x^*, \delta) = \{x \in \mathbb{R}^n \mid \|x - x^*\| \leq \delta\}$ .

## 2 A derivative-free trust-region LOVO algorithm

In order to help us with the theoretical results exposed in this text, consider the following definition.

**Definition 2.1.** [10, Definition 1] Given a feasible point  $x \in \Omega$  we define  $\mathcal{I} = \{1, \dots, r\}$  and  $\mathcal{I}_{min}(x) = \{i \in \{1, \dots, q\} \mid f_i(x) = f_{min}(x)\}$ .

We consider the following assumptions on the objective function.

**Assumption 2.2.** *The function  $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ , for  $i \in \mathcal{I}$ , is continuously differentiable and its gradient  $\nabla f_i : \mathbb{R}^n \rightarrow \mathbb{R}^n$  is Lipschitz continuous, with constant  $L_i > 0$ , in a sufficiently large open bounded domain  $\mathcal{X} \subset \mathbb{R}^n$ .*

**Assumption 2.3.** *The function  $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ , for  $i \in \mathcal{I}$ , is bounded below in  $\Omega \subset \mathbb{R}^n$ , i.e., there is a constant  $M_i \in \mathbb{R}$  such that  $f_i(x) \geq M_i$ , for all  $x \in \Omega$ .*

Note that if Assumption 2.3 holds and  $M := \min_{i \in \mathcal{I}} \{M_i\}$ , thus the function  $f_{min}$  is bounded below in  $\Omega$  by the constant  $M$ .

### 2.1 Trust-region framework

Our algorithm is based on the trust-region frameworks of [3, 50]. At each iteration  $k \in \mathbb{N}$ , we consider the current iterate  $x_k \in \Omega$ , the associated index  $i_k \in \mathcal{I}_{min}(x_k)$  and the quadratic model for  $f_{i_k}$

$$m_k(x) = m_k(x_k + d) = b_k + \mathbf{g}_k^T d + \frac{1}{2} d^T \mathbf{H}_k d, \quad (2)$$

where  $d = x - x_k \in \mathbb{R}^n$ ,  $b_k \in \mathbb{R}$ ,  $\mathbf{g}_k \in \mathbb{R}^n$  and  $\mathbf{H}_k \in \mathbb{R}^{n \times n}$  is a symmetric matrix. In [3], a similar approach is taken with  $\mathbf{g}_k = \nabla f_{i_k}(x_k)$  and  $\mathbf{H}_k = \nabla^2 f_{i_k}(x_k)$ . Note that only one index  $i_k \in \mathcal{I}_{min}(x_k)$  is chosen at iteration  $k \in \mathbb{N}$ . Thus, we will omit the indication of such index in the model definition and other expressions that depend on the choice of index.

We also follow the practical ideas of Powell [41, 42], further developed in [50], of using two radii:  $\delta_k$  is related to the quality of the model, and  $\Delta_k$  is associated with the trust-region. Assumption 2.4 defines what “quality of the model” means and is a weaker assumption than “fully linear models” of [19], since it does not specify how to build such models.

**Assumption 2.4.** *For all  $k \in \mathbb{N}$ ,  $\|\nabla f_{i_k}(x) - \nabla m_k(x)\| \leq \kappa_g \delta_k$ , for all  $x \in \mathcal{X} \cap \overline{B}(x_k, \delta_k)$  and some constant  $\kappa_g > 0$  independent of  $x$  and  $k$ .*

We define the stationarity measure at  $x_k$  for the problem of minimizing  $m_k$  over the set  $\Omega$  by

$$\pi_k = \|P_\Omega(x_k - \mathbf{g}_k) - x_k\|, \quad (3)$$

where  $P_\Omega$  denotes the orthogonal projection onto  $\Omega$ , which exists because it is a closed convex set [15]. In our context, given  $i \in \mathcal{I}$ , we say that a point  $x_* \in \Omega$  is stationary for the problem  $\min_{x \in \Omega} f_i(x)$  when  $\|P_\Omega(x_* - \nabla f_i(x_*)) - x_*\| = 0$  [15, Section 12.1.4].

At iteration  $k \in \mathbb{N}$ , the candidate point  $x_k + d_k \in \Omega$  is computed, where  $d_k$  is the approximate solution of the following trust-region subproblem

$$\begin{aligned} & \text{minimize} && m_k(x_k + d) \\ & \text{subject to} && x_k + d \in \Omega \\ & && \|d\| \leq \Delta_k. \end{aligned} \quad (4)$$

If the model is accurate for  $f_{i_k}$ , we expect that  $x_k + d_k$  may also decrease  $f_{min}$ , since  $x_k + d_k \in \Omega$  and  $f_{min}(x_k + d_k) \leq f_{i_k}(x_k + d_k) \leq f_{i_k}(x_k)$ . By “approximate solution” we mean that  $x_k + d_k$  must satisfy a sufficient decrease condition of  $m_k$ , given by Assumption 2.5.

**Assumption 2.5.** *The approximate solution of (4) satisfies the sufficient decrease condition*

$$m_k(x_k) - m_k(x_k + d_k) \geq \theta \pi_k \min \left\{ \frac{\pi_k}{1 + \|\mathbf{H}_k\|}, \Delta_k, 1 \right\},$$

for some constant  $\theta > 0$  independent of  $k$ .

Conditions of this type are well-known in trust-region strategies, and were used in different contexts by several authors, as we can see in [12, 50], [5, Lemma 11.3], [15, Assumption AA.1], [19, Theorem 10.1], [29, Assumption 3.3], [38, Lemma 4.5], and [48, Section 3.1.4].

We accept step  $x_k + d_k$  when the ratio between actual and predicted reductions

$$\rho_k = \frac{f_{min}(x_k) - f_{min}(x_k + d_k)}{m_k(x_k) - m_k(x_k + d_k)} \quad (5)$$

is greater than or equal to a fixed constant  $\eta > 0$ . In this case, the new iterate becomes  $x_{k+1} = x_k + d_k$ , the model is updated and the trust-region radius  $\Delta_k$  is possibly increased. Otherwise, we reject the step and  $\Delta_k$  is decreased.

## 2.2 Algorithm

In this section, we present the derivative-free trust-region algorithm for solving the LOVO problem (1), without specification about the model update or the solution of subproblem (4). Our algorithm is based on the ideas discussed by Andreani, Martínez and Martínez [3] and on the algorithms proposed by Conejo, Karas, Pedroso, Ribeiro and Sachine [12] and Verdério, Karas, Pedroso and Scheinberg [50].

---

### Algorithm 1: DERIVATIVE-FREE TRUST-REGION LOVO ALGORITHM

---

**Input:**  $x_0 \in \Omega$ ,  $\beta > 0$ ,  $0 < \delta_0 \leq \Delta_0$ ,  $0 < \tau_1 \leq \tau_2 < 1 \leq \tau_3 \leq \tau_4$ ,  $\eta_1 \in (0, 1)$ ,  
 $0 \leq \eta < \eta_1 \leq \eta_2$ ,  $\Gamma = 0$ ,  $1 \leq \Gamma_{\max} \in \mathbb{N}$ ,  $i_0 \in \mathcal{I}_{\min}(x_0)$ .

```

1 for  $k = 0, 1, \dots$  do
2   Construct the model  $m_k$ .
3   if  $\delta_k > \beta\pi_k$  then /* Criticality Phase */
4     Let  $x_{k+1} = x_k$ ,  $i_{k+1} = i_k$ ,  $\rho_k = 0$ ,  $d_k = 0$ ,  $\delta_{k+1} = \tau_1\delta_k$ , and choose
        $\Delta_{k+1} \in [\tau_1\Delta_k, \tau_2\Delta_k]$ .
5   else
6     Find an approximate solution  $d_k$  for (4) that satisfies Assumption 2.5.
7     Compute  $\rho_k$  by (5).
8     if  $\rho_k \geq \eta$  then /* Step Acceptance Phase */
9       Let  $x_{k+1} = x_k + d_k$ , and choose  $i_{k+1} \in \mathcal{I}_{\min}(x_{k+1})$ .
10    else
11      Let  $x_{k+1} = x_k$ , and  $i_{k+1} = i_k$ .
12    end
13    if  $\rho_k \geq \eta_1$  then  $\Gamma = 0$  end
14    if  $\rho_k \geq \eta$  and  $i_{k+1} \neq i_k$  and  $\Gamma \leq \Gamma_{\max}$  then /* Radii Adjustment Phase */
15      Let  $\delta_{k+1} = \tau_4\delta_k$ ,  $\Delta_{k+1} = \tau_4\Delta_k$ , and  $\Gamma = \Gamma + 1$ .
16    else /* Radii Update Phase */
17      Let
          
$$\delta_{k+1} = \begin{cases} \tau_1\delta_k, & \text{if } \rho_k < \eta_1; \\ \tau_3\delta_k, & \text{if } \rho_k > \eta_2 \text{ and } \|d_k\| = \Delta_k; \\ \delta_k, & \text{otherwise;} \end{cases}$$

          and
          
$$\Delta_{k+1} = \begin{cases} \tau_1\Delta_k, & \text{if } \rho_k < \eta_1; \\ \tau_3\Delta_k, & \text{if } \rho_k > \eta_2 \text{ and } \|d_k\| = \Delta_k; \\ \Delta_k, & \text{otherwise.} \end{cases}$$

18    end
19  end
20 end
```

---

The general idea of the algorithm is given below:

- The algorithm allows freedom in the choice of  $m_k$  as long as it satisfies Assumption 2.4. However, its practical efficiency greatly relies in the way it is implemented, as discussed in Section 5.
- When  $\delta_k$  is large with respect to  $\pi_k$ , we cannot guarantee that the model accurately represents  $f_{i_k}$ . Hence, if  $\delta_k > \beta\pi_k$ , the radius  $\delta_k$  is reduced in an attempt to find a more accurate model.
- In the case where there was a successful iteration ( $\rho_k \geq \eta$ ) and another function is taken as the representative of  $f_{\min}$  ( $i_{k+1} \neq i_k$ ), we increase  $\delta_{k+1}$  and  $\Delta_{k+1}$  by a factor of  $\tau_4 \geq 1$ . After a successful iteration with  $\rho_k \geq \eta_1$  and index swap, this procedure is executed at most  $\Gamma_{\max} \in \mathbb{N}$  iterations with index swap satisfying  $\eta \leq \rho_k < \eta_1$ . Thus, the radii of

the other iterations satisfying  $\rho_k < \eta_1$  continue to be reduced by  $\tau_1$ . By allowing the radius of the trust-region to increase, we expect the algorithm to try larger steps when a new  $f_{i_{k+1}}$  is considered. In [3],  $\Delta_{k+1}$  is reset to  $\Delta_0$  whenever index swap occurs, but we observed that such choice results in worse complexity results.

- In the case where  $|\mathcal{I}| = 1$  (usual convex constrained optimization problem), Algorithm 1 is reduced to the derivative-free trust-region algorithm proposed by [50].

### 3 Convergence analysis

In this section, we will provide global convergence results, in the LOVO sense, for sequences generated by Algorithm 1. We will show that every point of accumulation of a sequence generated by Algorithm 1 is stationary for some index  $i \in \mathcal{I}$ .

In the following lemma, we prove that the Hessian of the model  $m_k$  is bounded.

**Lemma 3.1.** *Suppose that Assumptions 2.2 and 2.4 hold. Thus, for every iteration  $k \in \mathbb{N}$ ,*

$$\|\mathbf{H}_k\| \leq \kappa_H - 1,$$

where  $\kappa_H := 2\kappa_g + L + 1$  and  $L := \max_{i \in \mathcal{I}} \{L_i\}$ .

*Proof.* Let  $k \in \mathbb{N}$ ,  $i_k \in \mathcal{I}_{min}(x_k)$  be the chosen index associated with  $m_k$  in this iteration and  $d \in \mathbb{R}^n$  an arbitrary direction satisfying  $\|d\| = \delta_k$ . Initially, note that,  $\nabla m_k(x_k + d) = \mathbf{g}_k + \mathbf{H}_k d$ , and  $\nabla m_k(x_k) = \mathbf{g}_k$ . The results follow by [50, Lemma 1] by observing that  $L_{i_k} \leq L$ .  $\square$

Analogously to [12, 50], let us consider the following sets of indexes

$$\mathcal{S} = \{k \in \mathbb{N} \mid \rho_k \geq \eta\} \text{ and } \bar{\mathcal{S}} = \{k \in \mathbb{N} \mid \rho_k \geq \eta_1\}, \quad (6)$$

and note that  $\bar{\mathcal{S}} \subset \mathcal{S}$ . The next lemma establishes that if the trust-region radius is sufficiently small, then the iteration will be successful.

**Lemma 3.2.** *Suppose that Assumptions 2.2, 2.4 and 2.5 hold. Consider the set*

$$\mathcal{K} = \left\{ k \in \mathbb{N} : \Delta_k \leq \min \left\{ \frac{\pi_k}{\kappa_H}, \frac{(1 - \eta_1)\pi_k}{c_1}, \beta\pi_k, 1 \right\} \right\}, \quad (7)$$

where  $c_1 := (L + \kappa_g + \kappa_H/2)/\theta$ . If  $k \in \mathcal{K}$ , then  $k \in \bar{\mathcal{S}}$ .

*Proof.* Let  $k \in \mathcal{K}$  and consider  $i_k \in \mathcal{I}_{min}(x_k)$  the index associated with the model  $m_k$  in this iteration. By the Mean Value Theorem, there exists  $\xi_k \in (0, 1)$  such that

$$f_{i_k}(x_k + d_k) = f_{i_k}(x_k) + \nabla f_{i_k}(x_k + \xi_k d_k)^T d_k.$$

Therefore, by Assumptions 2.2, 2.4 and Lemma 3.1, by following the same arguments as [12, Lemma 3.1], we obtain

$$|m_k(x_k) - m_k(x_k + d_k) + f_{i_k}(x_k + d_k) - f_{i_k}(x_k)| \leq \left( L + \kappa_g + \frac{1}{2}\kappa_H \right) \Delta_k^2 \quad (8)$$

By (7) we have that  $\pi_k > 0$ . Thus, the sufficient decrease condition of Assumption 2.5 implies  $m_k(x_k) - m_k(x_k + d_k) > 0$ . The key point in this proof is to observe that, since  $i_k \in \mathcal{I}_{min}(x_k)$ , we know that  $f_{i_k}(x_k) = f_{min}(x_k)$  and  $f_{min}(x_k + d_k) \leq f_{i_k}(x_k + d_k)$ . By (5), (8), Assumption 2.5 and Lemma 3.1,

$$\begin{aligned}
1 - \rho_k &= 1 - \frac{f_{min}(x_k) - f_{min}(x_k + d_k)}{m_k(x_k) - m_k(x_k + d_k)} \\
&\leq 1 - \frac{f_{i_k}(x_k) - f_{i_k}(x_k + d_k)}{m_k(x_k) - m_k(x_k + d_k)} \\
&\leq \frac{|m_k(x_k) - m_k(x_k + d_k) - f_{i_k}(x_k) + f_{i_k}(x_k + d_k)|}{m_k(x_k) - m_k(x_k + d_k)} \\
&\leq \frac{(L + \kappa_g + \frac{1}{2}\kappa_H)\Delta_k^2}{\theta\pi_k \min\left\{\frac{\pi_k}{\kappa_H}, \Delta_k, 1\right\}} \\
&= \frac{c_1\Delta_k^2}{\pi_k \min\left\{\frac{\pi_k}{\kappa_H}, \Delta_k, 1\right\}},
\end{aligned}$$

where  $c_1 = (L + \kappa_g + \kappa_H/2)/\theta$ . The remaining of the proof follows exactly as [12, Lemma 3.1], adapting our constants.  $\square$

The following result is a direct consequence of Lemma 3.2. Assuming that  $\pi_k \geq \varepsilon > 0$ , we can prove that the trust-region radius is bounded below by  $\Delta_{min}$ , which depends on the parameters of the algorithm, the characteristics of problem (1) and the type of model  $m_k$  used.

**Corollary 3.3.** *Suppose that Assumptions 2.2, 2.4 and 2.5 hold, and let  $\varepsilon > 0$ . If  $\pi_k \geq \varepsilon$ , then*

$$\Delta_k \geq \Delta_{min} := \min\left\{\Delta_0, \tau_1 \min\left\{\frac{\varepsilon}{\kappa_H}, \frac{(1 - \eta_1)\varepsilon}{c_1}, \beta\varepsilon, 1\right\}\right\}.$$

As mentioned earlier, the radius of the sample region  $\delta_k$  controls the quality of the model. More specifically, Assumption 2.4 tells us that the smaller the sample radius, the better the models represent  $f_{i_k}$ . Therefore, it is reasonable to expect  $\delta_k$  to go to zero. The following two lemmas show us that this happens. In the remainder of this section, unless otherwise stated, we will assume that Assumptions 2.2, 2.3, 2.4 and 2.5 hold.

**Lemma 3.4.** *If  $|\overline{\mathcal{S}}| = \infty$ , then the sequence  $\{\delta_k\}_{k \in \overline{\mathcal{S}}}$  converges to zero.*

*Proof.* The proof follows the same arguments as [50, Lemma 3]. Note that  $\{f_{min}(x_k)\}_{k \in \mathbb{N}}$  is a monotone nonincreasing sequence and bounded below, and the functions  $f_i$ ,  $i = 1, \dots, r$ , are bounded below in  $\Omega$ .  $\square$

**Lemma 3.5.** *The sequence  $\{\delta_k\}_{k \in \mathbb{N}}$  converges to zero.*

*Proof.* The proof is based on [50, Lemma 3], with the main difference that we are using  $\Gamma_{max}$ . We will split the demonstration into two cases.

- i)  $\overline{\mathcal{S}}$  is finite.

Initially consider that the set  $\bar{\mathcal{S}}$  is finite. Hence, there is a finite number of iterations for which the radii adjustment phase (line 14) is called. In fact, such phase is called for at most  $\Gamma_{\max} |\bar{\mathcal{S}}|$  iterations. Thus, there is  $k_0 \in \mathbb{N}$  such that for every  $k \geq k_0$ , the iteration  $k \notin \bar{\mathcal{S}}$  and the radii adjustment phase (line 14) is not called. Let  $\mathcal{M}_1 := \{k \in \mathbb{N} \mid k \geq k_0\}$ . Note that, by the radii update phase (line 16) present in the algorithm, we have that for every iteration  $k \in \mathcal{M}_1$   $\delta_{k+1} = \tau_1 \delta_k$ , where  $\tau_1 < 1$ . Therefore, given that  $\delta_{k_0}$  is a constant, it follows that

$$\lim_{k \rightarrow \infty} \delta_k = \lim_{k \in \mathcal{M}_1} \delta_k = \lim_{k \rightarrow \infty} \tau_1^k \delta_{k_0} = 0.$$

ii)  $\bar{\mathcal{S}}$  is infinite.

Now, assume that the set  $\bar{\mathcal{S}}$  is infinite and let  $\mathcal{M}_2 := \{k \in \mathbb{N} \mid k \notin \bar{\mathcal{S}}\}$ . If the set  $\mathcal{M}_2$  is finite, then by Lemma 3.4 it follows that

$$\lim_{k \rightarrow \infty} \delta_k = \lim_{k \in \bar{\mathcal{S}}} \delta_k = 0.$$

This leaves only the case where  $\mathcal{M}_2$  is infinite. Note that, by Lemma 3.4, we obtain the desired limit for indices in  $\bar{\mathcal{S}}$ . We will extend this result to the entire sequence. For this purpose, let  $k \in \mathcal{M}_2$  and  $l_k$  be the last iteration in  $\bar{\mathcal{S}}$  before  $k$ . Since between the iterations  $l_k$  and  $k$  the radii adjustment phase (line 14) is called at most  $\Gamma_{\max}$  times and  $\tau_4 \geq \tau_3 \geq 1$ , we have that the value of  $\delta_k$  is at most  $\tau_4^{\Gamma_{\max}} \delta_{l_k}$ . Therefore,

$$\lim_{k \in \mathcal{M}_2} \delta_k \leq \lim_{k \in \mathcal{M}_2} \tau_4^{\Gamma_{\max}} \delta_{l_k} = \tau_4^{\Gamma_{\max}} \lim_{l_k \in \bar{\mathcal{S}}} \delta_{l_k} = 0,$$

and we conclude the proof.  $\square$

Next, we show a weak convergence result for the problem of minimizing the model  $m_k$  in the feasible region  $\Omega$ .

**Lemma 3.6.** *The sequence  $\{\pi_k\}_{k \in \mathbb{N}}$  admits a subsequence that converges to zero.*

*Proof.* We will show that

$$\liminf_{k \rightarrow \infty} \pi_k = 0.$$

In fact, suppose by contradiction that there are  $\varepsilon > 0$  and an integer  $k_0 > 0$  such that  $\pi_k \geq \varepsilon$  for all  $k \geq k_0$ . Let  $k \in \bar{\mathcal{S}}$ , with  $k \geq k_0$  arbitrary. By the definition of  $\rho_k$  given in (5), Assumption 2.5, the contradiction hypothesis and Corollary 3.3, it follows that

$$\begin{aligned} f_{\min}(x_k) - f_{\min}(x_k + d_k) &\geq \rho_k \theta \pi_k \min \left\{ \frac{\pi_k}{\kappa_H}, \Delta_k, 1 \right\} \\ &\geq \eta_1 \theta \varepsilon \min \left\{ \frac{\varepsilon}{\kappa_H}, \Delta_{\min}, 1 \right\}. \end{aligned}$$

By Assumption 2.3,  $\{f_{\min}(x_k)\}_{k \in \mathbb{N}}$  is bounded below, and given that it is a monotone nonincreasing sequence, it follows that  $f_{\min}(x_k) - f_{\min}(x_k + d_k) \rightarrow 0$ . Since the right-hand side of the above inequality is a positive constant, the set  $\{k \in \bar{\mathcal{S}} : k \geq k_0\}$  is finite, and thus the radii adjustment phase (line 14) is also called only a finite number of iterations. However, by the Lemma 3.5 we have



that  $\delta_k \rightarrow 0$ , and given that  $\pi_k \geq \varepsilon$  for all  $k \geq k_1$ , it follows that the criticality phase (line 3) is called only for a finite number of iterations. Thus, for every sufficiently large  $k \in \mathbb{N}$ , we only have iterations with  $\rho_k < \eta_1$  without the radii adjustment phase (line 14) being called, which implies that  $\Delta_{k+1} = \tau_1 \Delta_k$ . Consequently,  $\Delta_k \rightarrow 0$ , contradicting Corollary 3.3.  $\square$

The following result is a direct consequence of Lemma 3.6.

**Corollary 3.7.** *There is an index  $i \in \mathcal{I}$ , that is selected an infinite number of iterations, such that*

$$\liminf_{k \in \mathcal{J}(i)} \pi_k = 0,$$

where  $\mathcal{J}(i) = \{k \in \mathbb{N} \mid i_k = i\}$ .

*Proof.* From Lemma 3.6, we know that  $\{\pi_k\}_{k \in \mathbb{N}}$  admits a subsequence that converges to zero. Given that such a subsequence is infinite and that  $|\mathcal{I}| = r < \infty$ , there must be an index  $i \in \mathcal{I}$  that is chosen in an infinite number of iterations of that subsequence. Thus, collecting such iterations into the set  $\mathcal{J}(i) = \{k \in \mathbb{N} \mid i_k = i\}$ , we have that  $\liminf_{k \in \mathcal{J}(i)} \pi_k = 0$ , and we conclude the proof.  $\square$

If we, in addition, ask for a sufficient decrease condition  $\eta > 0$  in Algorithm 1, then we can show that  $\{\pi_k\}_{k \in \mathbb{N}}$  converges to zero for any subsequence when an index  $i \in \mathcal{I}$  is selected an infinite number of iterations.

**Lemma 3.8.** *Suppose that  $\eta > 0$  and let  $i \in \mathcal{I}$  be an index chosen an infinite number of iterations. Then,*

$$\lim_{k \in \mathcal{J}(i)} \pi_k = 0,$$

where  $\mathcal{J}(i) = \{k \in \mathbb{N} \mid i_k = i\}$ .

*Proof.* Initially, we know that  $\mathcal{J}(i)$  exists by the same arguments of Corollary 3.7. Suppose by contradiction that for some  $\varepsilon > 0$  the set  $\mathcal{M}_1 = \mathcal{J}(i) \cap \{k \in \mathbb{N} : \pi_k \geq \varepsilon\}$  is infinite. Given  $k \in \mathcal{M}_1$ , consider  $u_k \in \mathbb{N}$  the first iteration such that  $u_k > k$  and  $\pi_{u_k} \leq \varepsilon/2$ , and so  $\pi_k - \pi_{u_k} \geq \varepsilon/2$ . Note that the existence of  $u_k$  is guaranteed by Lemma 3.6. By Lemma 3.5, there is  $k_0 \in \mathbb{N}$  such that, for  $k \geq k_0$ , we have  $\delta_k \leq \varepsilon/(8\kappa_g)$ , where  $\kappa_g$  is the constant given in Assumption 2.4. Also, let us define  $\mathcal{C}_k = \{j \in \mathcal{S} : k \leq j < u_k\}$ . We will show that  $\mathcal{C}_k$  is nonempty by considering two cases:  $u_k \in \mathcal{J}(i)$  and  $u_k \notin \mathcal{J}(i)$ .

i)  $u_k \in \mathcal{J}(i)$ .

Given that  $u_k \in \mathcal{J}$ , we have that  $i_k = i_{u_k}$ . In other words, the functions representing  $f_{min}$  at  $k$  and  $u_k$  are the same. In this case, the conclusion that  $\mathcal{C}_k$  is nonempty follows from the same arguments of [50, Lemma 5].

ii)  $u_k \notin \mathcal{J}(i)$ .

In this case, there is no guarantee that  $\nabla f_{i_k}$  and  $\nabla f_{i_{u_k}}$  are the same functions and, therefore, the argument used in item i) is not valid. However, given that  $u_k \notin \mathcal{J}(i)$ , by the index choice condition in the step acceptance phase (line 8), we know that there was at least one index swap between iterations  $k$  and  $u_k$ . Thus, the set  $\mathcal{C}_k$  is nonempty.

Therefore, in both cases we obtain that  $\mathcal{C}_k$  is a nonempty set. Thus, by Assumption 2.5, Lemma 3.1, Corollary 3.3, and the fact that  $\pi_j \geq \varepsilon/2$ , for all  $j \in \mathcal{C}_k$ , we have that, for all  $k \in \mathcal{M}_1$ ,  $k \geq k_0$ ,

$$\begin{aligned}
f_{\min}(x_k) - f_{\min}(x_{u_k}) &\geq \sum_{j \in \mathcal{C}_k} (f_{\min}(x_j) - f_{\min}(x_{j+1})) \\
&\geq \sum_{j \in \mathcal{C}_k} \rho_j \theta \pi_j \min \left\{ \frac{\pi_j}{\kappa_H}, \Delta_j, 1 \right\} \\
&\geq \frac{\eta \theta \varepsilon}{2} \min \left\{ \frac{\varepsilon}{2\kappa_H}, \sum_{j \in \mathcal{C}_k} \Delta_j, 1 \right\} \\
&\geq \frac{\eta \theta \varepsilon}{2} \min \left\{ \frac{\varepsilon}{2\kappa_H}, \Delta_{\min}, 1 \right\}.
\end{aligned} \tag{9}$$

Thus, since  $\eta > 0$ , it follows that the right-hand side of (9) is a positive constant. On the other hand, by Assumption 2.3,  $\{f_{\min}(x_k)\}_{k \in \mathbb{N}}$  is bounded below and, by the construction of the algorithm, is a monotone nonincreasing sequence. Hence,  $f_{\min}(x_k) - f_{\min}(x_{u_k}) \rightarrow 0$ , which is a contradiction with (9), and completes the proof.  $\square$

We are now ready to prove the global convergence results for Algorithm 1. In a way similar to the stationarity measure  $\pi_k$  when solving subproblems (4), we adopt the criticality measure  $\|\mathcal{P}_\Omega(x - \nabla f(x)) - x\|$ , proposed by Conn, Gould and Toint [14] for optimization problems involving a continuous function  $f$  on a convex feasible set  $\Omega$ . This measure was used in [12, 13, 50], for example, to establish global convergence results. The following theorem is the main result of this section. It allows us to further describe what kind of stationarity can be achieved by Algorithm 1.

**Theorem 3.9.** *Let us define sets  $\mathcal{J}(i) = \{k \in \mathbb{N} \mid i_k = i\}$ , for  $i \in \mathcal{I}$ , and let  $\{x_k\}_{k \in \mathbb{N}}$  be a sequence generated by Algorithm 1. The following statements hold.*

i) *If  $\eta = 0$ , then*

$$\liminf_{k \rightarrow \infty} \|\mathcal{P}_\Omega(x_k - \nabla f_{i_k}(x_k)) - x_k\| = 0.$$

*Furthermore, there is an index  $i \in \mathcal{I}$  such that*

$$\liminf_{k \in \mathcal{J}(i)} \|\mathcal{P}_\Omega(x_k - \nabla f_{i_k}(x_k)) - x_k\| = 0.$$

ii) *If  $\eta > 0$  and  $i \in \mathcal{I}$  is any index chosen for an infinite number of iterations, then*

$$\lim_{k \in \mathcal{J}(i)} \|\mathcal{P}_\Omega(x_k - \nabla f_{i_k}(x_k)) - x_k\| = 0.$$

iii) *If  $i \in \mathcal{I}$  is an index chosen for an infinite number of iterations and  $x_* \in \mathbb{R}^n$  is an accumulation point for the subsequence  $\{x_k\}_{k \in \mathcal{J}(i)} \subseteq \{x_k\}_{k \in \mathbb{N}}$  then  $i \in \mathcal{I}_{\min}(x_*)$ .*

*Proof.* Initially, note that by the triangular inequality, the properties of the projection operator, and Assumption 2.4, we have

$$\begin{aligned}
& \|P_\Omega(x_k - \nabla f_{i_k}(x_k)) - x_k\| \\
&= \|P_\Omega(x_k - \nabla f_{i_k}(x_k)) - P_\Omega(x_k - \mathbf{g}_k) + P_\Omega(x_k - \mathbf{g}_k) - x_k\| \\
&\leq \|P_\Omega(x_k - \nabla f_{i_k}(x_k)) - P_\Omega(x_k - \mathbf{g}_k)\| + \|P_\Omega(x_k - \mathbf{g}_k) - x_k\| \\
&= \|P_\Omega(x_k - \nabla f_{i_k}(x_k) - x_k + \mathbf{g}_k)\| + \|P_\Omega(x_k - \mathbf{g}_k) - x_k\| \quad (10) \\
&= \|P_\Omega(\mathbf{g}_k - \nabla f_{i_k}(x_k))\| + \|P_\Omega(x_k - \mathbf{g}_k) - x_k\| \\
&\leq \|\mathbf{g}_k - \nabla f_{i_k}(x_k)\| + \|P_\Omega(x_k - \mathbf{g}_k) - x_k\| \\
&\leq \kappa_g \delta_k + \pi_k.
\end{aligned}$$

Thus, using Lemmas 3.5 and 3.6, Corollary 3.7, and (10), we obtain i). On the other hand, by Lemmas 3.5 and 3.8, and expression (10), we conclude the proof of ii).

It still remains for us to prove iii). Note that by the statement of the theorem we have that  $\lim_{k \in \mathcal{J}(i)} x_k = x_*$ , and by Assumption 2.2, the functions  $f_j$ ,  $j \in \mathcal{I}$ , are continuously differentiable. So the continuity of  $f_j$  gives us

$$\lim_{k \in \mathcal{J}(i)} f_j(x_k) = f_j(x_*), \quad (11)$$

for any index  $j \in \mathcal{I}$ . Given that  $i \in \mathcal{I}_{min}(x_k)$ , for every  $k \in \mathcal{J}(i)$ , we have by the definition of this set that  $f_i(x_k) \leq f_j(x_k)$  for any index  $j \in \mathcal{I}$  and  $k \in \mathcal{J}(i)$ . Thus, taking the limit in  $\mathcal{J}(i)$ , by (11) it follows that  $f_i(x_*) \leq f_j(x_*)$ , and therefore  $i \in \mathcal{I}_{min}(x_*)$ .  $\square$

Theorem 3.9 states that if  $\eta > 0$  and  $x_* \in \mathbb{R}^n$  is an accumulation point of a sequence  $\{x_k\}_{k \in \mathbb{N}}$  generated by Algorithm 1, it is possible to construct a subsequence  $\{x_k\}_{k \in \mathcal{J}(i)}$  which converges to  $x_*$ , where  $\mathcal{J}(i) = \{k \in \mathbb{N} \mid i_k = i\}$ , and  $x_*$  satisfies a necessary optimality condition of gradient projected type for the problem

$$\begin{aligned}
& \text{minimize} && f_i(x) \\
& \text{subject to} && x \in \Omega.
\end{aligned} \quad (12)$$

In other words,  $x_*$  is a first-order stationary point for the problem (12) (see [14] and [15, p. 450]).

The following definition and corollaries help us to understand Theorem 3.9 from the perspective of the LOVO theory [4].

**Definition 3.10.** [4, p. 05] Given  $x_* \in \Omega$  and established a necessary optimality condition for the problem (12), we say that

- i)  $x_*$  is strongly critical if it satisfies a necessary optimality condition for (12), for all  $i \in \mathcal{I}_{min}(x_*)$ .
- ii)  $x_*$  is weakly critical if it satisfies a necessary optimality condition for (12), for some index  $i \in \mathcal{I}_{min}(x_*)$ .

**Corollary 3.11.** Suppose that  $\eta > 0$ . If  $x_* \in \mathbb{R}^n$  is an accumulation point of a sequence  $\{x_k\}_{k \in \mathbb{N}}$  generated by Algorithm 1, then  $x_*$  is weakly critical.

*Proof.* Let  $x_*$  be an accumulation point of the sequence  $\{x_k\}_{k \in \mathbb{N}}$  generated by Algorithm 1. Thus, there is a subsequence  $\{x_k\}_{k \in \mathcal{N}} \subseteq \{x_k\}_{k \in \mathbb{N}}$  that converges to  $x_*$ . If  $i \in \mathcal{I}$  is an index that repeats for an infinite number of iterations in this subsequence then, as subsequence  $\{x_k\}_{k \in \mathcal{J}(i)} \subseteq \{x_k\}_{k \in \mathcal{N}} \subseteq \{x_k\}_{k \in \mathbb{N}}$  converges to  $x_*$ , by iii) of Theorem 3.9 it follows that  $i \in \mathcal{I}_{\min}(x_*)$ . Furthermore, from ii) of the same theorem, it follows that

$$\lim_{k \in \mathcal{J}(i)} \|\mathcal{P}_\Omega(x_k - \nabla f_i(x_k)) - x_k\| = 0,$$

and thus  $x_*$  satisfies a necessary optimality condition for problem (12). Therefore,  $x_*$  is a weakly critical point.  $\square$

**Corollary 3.12.** *If Algorithm 1 generates a sequence  $\{x_k\}_{k \in \mathbb{N}}$  that converges to  $x_* \in \mathbb{R}^n$ , then  $x_*$  is weakly critical.*

*Proof.* Let  $\{x_k\}_{k \in \mathbb{N}}$  be a sequence generated by Algorithm 1 and suppose that it converges to a point  $x_* \in \mathbb{R}^n$ . If

- $\eta = 0$ , let  $i \in \mathcal{I}$  be the index from statement i) of Theorem 3.9;
- $\eta > 0$ , let  $i \in \mathcal{I}$  be any index chosen for an infinite number of iterations;

Note that  $\{x_k\}_{k \in \mathcal{J}(i)} \subseteq \{x_k\}_{k \in \mathbb{N}}$  converges to  $x_*$  and so, by iii) of Theorem 3.9, it follows that  $i \in \mathcal{I}_{\min}(x_*)$ . Furthermore, by i) or ii) of the same theorem, it follows that  $\lim_{k \in \mathcal{J}(i)} \|\mathcal{P}_\Omega(x_k - \nabla f_i(x_k)) - x_k\| = 0$  and thus  $x_*$  satisfies a necessary optimality condition for the problem (12). Therefore,  $x_*$  is a weakly critical point.  $\square$

Theorem 3.9 highlights another very desirable feature of Algorithm 1. It can be used as the inner solver for general-constrained derivative-free algorithms, especially of the Inexact Restoration type [8, 21], a case where  $\Omega$  is composed by linear constraints. In such a case, Algorithm 1 is able to generate sequences to *Approximate Gradient Projection* (AGP) points. The AGP condition is a strong practical necessary optimality condition [2], which is satisfied by every local minimizer of a nonlinear optimization problem without any constraint qualification.

Consider our initial problem (1), and note that the feasible set  $\Omega \subset \mathbb{R}^n$  is convex, closed, and nonempty. Assume that  $\Omega$  can be expressed by a set of expressions of the form  $g_j \leq 0$ ,  $j = 1, 2, \dots, k_1$ , and  $h_l = 0$ ,  $l = 1, 2, \dots, k_2$ , where the functions  $g_j$  are convex, and  $h_l$  are affine. In this context, our algorithm is able to generate sequences that satisfy a *Convex Approximate Gradient Projection* (C-AGP) condition [2, p. 635]. C-AGP can be more appropriate in some contexts and given a feasible point  $x_*$  that satisfies C-AGP and the *Mangasarian-Fromovitz* (MFCQ) constraint qualification then  $x_*$  also satisfies the KKT conditions for problem (12) [2, Theorem 3.2]. However, convergence to C-AGP points does not imply AGP points and vice versa. Now consider the situation where the functions  $g_j$ ,  $j = 1, 2, \dots, k_1$ , are linear. This case is particularly interesting when we have bound (or linear in general) constraints on the feasible set. Under these circumstances, we can employ a *Linear Approximate Gradient Projection* (L-AGP) criterion [2, p. 638], which is a first-order optimality condition stronger than the original AGP condition, in the sense that L-AGP implies AGP.

The last discussion in this section is related to Assumption 2.4. It is well known that such assumption is satisfied if, for example, linear or quadratic interpolation models are constructed using the so-called  $\Lambda$ -poised sample points [18],  $\Lambda > 0$ . Models with a relaxed interpolation condition were also considered in [46, 50]. Unfortunately, usually the techniques used to ensure well poisedness deal only with the unconstrained case. Powell [41] uses a technique to build good underdetermined quadratic interpolation models for box constraints. Hough and Roberts [29] present a weaker definition of “fully linear models” and extend the concept of  $\Lambda$ -poisedness. Unfortunately, this definition depends on a different stationarity measure, which requires a convex-constrained optimization problem to be solved in order to be verified. On the other hand, the projected gradient measure used by Algorithm 1 and by Theorem 3.9 has a closed form in many special cases, when  $\Omega$  is a box or a hyper-sphere, for example. Another possibility to build well poised models using unconstrained strategies is to allow  $f_{min}$  to be evaluated in points outside  $\Omega$ . That was the case in [11, 21]. When  $f_{min}$  is not defined outside  $\Omega$  (also known as *hard constraints*), then the criticality measure and strategy from [29] can be adapted to Algorithm 1 and benefit from using two different radii.

## 4 Worst-case complexity

In this section we will perform the worst-case iteration and function evaluation complexity analysis for Algorithm 1. For this purpose, let us first define the stationarity measure  $\pi_k^f = \|\mathcal{P}_\Omega(x_k - \nabla f_{i_k}(x_k)) - x_k\|$ , which is related to the stationarity of the convex-constrained problem (12) for  $f_{i_k}$ . Given  $\epsilon > 0$ , we will bound the number of iterations necessary to verify  $\epsilon$ -weakly stationarity, that is  $\pi_k^f < \epsilon$ , in terms of the initial point, problem’s and algorithm’s constants, and  $\epsilon$ . Note that, by Theorem 3.9, this condition can be satisfied at least for a subsequence generated by Algorithm 1.

Let  $k_\epsilon \in \mathbb{N}$  be the first iteration that  $\pi_{k_\epsilon}^f < \epsilon$  is verified. In order to help us count the number of iterations up to  $k_\epsilon$ , we define the sets

- $\mathcal{C}_\epsilon$ , iterations  $k \leq k_\epsilon$  that entered the criticality phase (line 3);
- $\mathcal{U}_\epsilon$ , all unsuccessful iterations ( $\rho_k < \eta$ );
- $\mathcal{A}_\epsilon^{\mathcal{R}}$  and  $\mathcal{A}_\epsilon^{\mathcal{NR}}$ , all acceptable iterations ( $\eta \leq \rho_k < \eta_1$ ) in which the radii adjustment phase (line 14) is and is not called, respectively;
- $\overline{\mathcal{S}}_\epsilon^{\mathcal{R}}$  and  $\overline{\mathcal{S}}_\epsilon^{\mathcal{NR}}$ , all successful iterations ( $\rho_k \geq \eta_1$ ) in which the radii adjustment phase (line 14) is and is not called, respectively;

For the purposes of notation, we also define

- $\overline{\mathcal{S}}_\epsilon = \overline{\mathcal{S}}_\epsilon^{\mathcal{R}} \cup \overline{\mathcal{S}}_\epsilon^{\mathcal{NR}}$ , the set of all successful iterations;
- $\mathcal{R}_\epsilon = \mathcal{U}_\epsilon \cup \mathcal{A}_\epsilon^{\mathcal{NR}}$ , the set of all iterations in which there was necessarily a reduction in the trust-region radius;
- $\mathcal{N}_\epsilon = \mathcal{C}_\epsilon \cup \mathcal{U}_\epsilon \cup \mathcal{A}_\epsilon^{\mathcal{R}} \cup \mathcal{A}_\epsilon^{\mathcal{NR}} = \mathcal{C}_\epsilon \cup \mathcal{R}_\epsilon \cup \mathcal{A}_\epsilon^{\mathcal{R}}$ , the set of all iterations that are not successful (not in  $\overline{\mathcal{S}}_\epsilon$ ).

The following two results help us to establish a relation between the stationarity measures  $\pi_k$  and  $\pi_k^f$ .

**Corollary 4.1.** *Suppose that Assumption 2.4 holds. Then the stationarity measures  $\pi_k$  and  $\pi_k^f$  satisfy  $|\pi_k - \pi_k^f| \leq \kappa_g \delta_k$ .*

*Proof.* The result follows from the proof of Theorem 3.9.  $\square$

**Lemma 4.2.** *Suppose that Assumption 2.4 holds and that Algorithm 1 has not entered in the criticality phase (line 3) at iteration  $k \in \mathbb{N}$ . If  $\pi_k^f \geq \epsilon$ , then  $\pi_k \geq c_2 \epsilon$ , where  $c_2 := 1/(1 + \kappa_g \beta)$ .*

*Proof.* Since  $k \in \mathbb{N}$  is not a criticality iteration, we have that  $\delta_k \leq \beta \pi_k$ . Thus, by Corollary 4.1, it follows that

$$\epsilon \leq \pi_k^f \leq |\pi_k^f - \pi_k| + \pi_k \leq \kappa_g \delta_k + \pi_k \leq \kappa_g \beta \pi_k + \pi_k \leq (\kappa_g \beta + 1) \pi_k.$$

Therefore, by letting  $c_2 = 1/(1 + \kappa_g \beta)$ , we obtain  $\pi_k \geq c_2 \epsilon$ .  $\square$

We are now able to count the number of successful iterations. We recall Corollary 3.3, which defines the  $\epsilon$ -dependent constant  $\Delta_{\min}$ , and observe that  $\epsilon$  is a lower bound of  $\pi_k$ . The value of  $\epsilon$  is not the same of  $\epsilon$  and will be set to suitable choices, in order to obtain the desired results.

**Lemma 4.3.** *Suppose that Assumptions 2.2, 2.3, 2.4, and 2.5 hold. Then Algorithm 1 needs a maximum of*

$$|\overline{\mathcal{S}}_\epsilon| \leq \frac{f_{\min}(x_0) - M}{\theta \eta_1 c_2} \Delta_{\min}^{-1} \epsilon^{-1},$$

*successful iterations to reach  $\pi_{k_\epsilon}^f < \epsilon$ , where  $M := \min_{i \in \mathcal{I}} \{M_i\}$ ,  $M_i$  is defined in Assumption 2.3 and  $\Delta_{\min}$  is defined in Corollary 3.3.*

*Proof.* Let  $k \in \overline{\mathcal{S}}_\epsilon$ . Thus, by the definition of  $\rho_k$ , Lemma 4.2, and letting  $\epsilon = c_2 \epsilon$  in Corollary 3.3, we have

$$\begin{aligned} f_{\min}(x_k) - f_{\min}(x_{k+1}) &= \rho_k (\mathbf{m}_k(x_k) - \mathbf{m}_k(x_{k+1})) \\ &\geq \rho_k \theta \pi_k \min \left\{ \frac{\pi_k}{\kappa_H}, \Delta_k, 1 \right\} \\ &\geq \eta_1 \theta \pi_k \min \left\{ \frac{\pi_k}{\kappa_H}, \Delta_k, 1 \right\} \\ &\geq \eta_1 \theta c_2 \epsilon \min \left\{ \frac{c_2 \epsilon}{\kappa_H}, \Delta_k, 1 \right\} \\ &\geq \eta_1 \theta c_2 \epsilon \min \left\{ \frac{c_2 \epsilon}{\kappa_H}, \Delta_{\min}, 1 \right\} \\ &= \theta \eta_1 c_2 \Delta_{\min} \epsilon, \end{aligned} \tag{13}$$

where the last inequality comes from the definition of  $\Delta_{\min}$ . Thus, by adding

up (13) to every  $k \in \overline{\mathcal{S}}_\epsilon$ , we get

$$\begin{aligned} f_{\min}(x_0) - f_{\min}(x_{k_\epsilon}) &\geq \sum_{k \in \overline{\mathcal{S}}_\epsilon} (f_{\min}(x_k) - f_{\min}(x_{k+1})) \\ &\geq \sum_{k \in \overline{\mathcal{S}}_\epsilon} \theta \eta_1 c_2 \Delta_{\min} \epsilon \\ &= \theta \eta_1 c_2 \Delta_{\min} \epsilon |\overline{\mathcal{S}}_\epsilon|. \end{aligned}$$

Given that  $f_{\min}(x_{k_\epsilon}) \geq M$ , then

$$f_{\min}(x_0) - M \geq f_{\min}(x_0) - f_{\min}(x_{k_\epsilon}) \geq \theta \eta_1 c_2 \Delta_{\min} \epsilon |\overline{\mathcal{S}}_\epsilon|,$$

and it follows that,

$$|\overline{\mathcal{S}}_\epsilon| \leq \frac{f_{\min}(x_0) - M}{\theta \eta_1 c_2} \Delta_{\min}^{-1} \epsilon^{-1}.$$

□

Next, we set an upper bound on the number of iterations that are not successful. Recall that the set of all iterations up to  $k_\epsilon$  is given by  $\overline{\mathcal{S}}_\epsilon \cup \mathcal{N}_\epsilon$ .

**Lemma 4.4.** *Under the conditions established in Lemma 4.3, Algorithm 1 needs at most*

$$|\mathcal{N}_\epsilon| \leq \frac{\log(\Delta_0) - \log(\Delta_{\min})}{|\log(\tau_2)|} + c_3 |\overline{\mathcal{S}}_\epsilon| \quad (14)$$

not successful iterations to reach  $\pi_{k_\epsilon}^f < \epsilon$ , where  $c_3 = \Gamma_{\max} + (\Gamma_{\max} + 1) \frac{\log(\tau_4)}{|\log(\tau_2)|}$ .

*Proof.* Initially, note that, by the description of Algorithm 1

- $\Delta_{k+1} \leq \tau_2 \Delta_k$ , for all  $k \in \mathcal{C}_\epsilon$ ;
- $\Delta_{k+1} = \tau_1 \Delta_k$ , for all  $k \in \mathcal{R}_\epsilon$ ;
- $\Delta_{k+1} \leq \tau_3 \Delta_k$ , for all  $k \in \overline{\mathcal{S}}_\epsilon^{\mathcal{NR}}$ ;
- $\Delta_{k+1} = \tau_4 \Delta_k$ , for all  $k \in \mathcal{A}_\epsilon^{\mathcal{R}} \cup \overline{\mathcal{S}}_\epsilon^{\mathcal{R}}$ .

Thus, by applying Corollary 3.3 with  $\varepsilon = c_2 \epsilon$  and since  $\tau_1 \leq \tau_2$  and  $\tau_3 \leq \tau_4$ , we must have  $\Delta_{\min} \leq \Delta_{k_\epsilon} \leq \Delta_0 \cdot \tau_2^{|\mathcal{C}_\epsilon|} \cdot \tau_1^{|\mathcal{R}_\epsilon|} \cdot \tau_3^{|\overline{\mathcal{S}}_\epsilon^{\mathcal{NR}}|} \cdot \tau_4^{|\mathcal{A}_\epsilon^{\mathcal{R}}| + |\overline{\mathcal{S}}_\epsilon^{\mathcal{R}}|} \leq \Delta_0 \cdot \tau_2^{|\mathcal{C}_\epsilon| + |\mathcal{R}_\epsilon|} \cdot \tau_4^{|\mathcal{A}_\epsilon^{\mathcal{R}}| + |\overline{\mathcal{S}}_\epsilon^{\mathcal{R}}|} = \Delta_0 \cdot \tau_2^{|\mathcal{C}_\epsilon| + |\mathcal{R}_\epsilon|} \cdot \tau_4^{|\mathcal{A}_\epsilon^{\mathcal{R}}| + |\overline{\mathcal{S}}_\epsilon|}$ . Thereby,

$$\begin{aligned} \log(\Delta_{\min}) &\leq \log\left(\Delta_0 \cdot \tau_2^{|\mathcal{C}_\epsilon| + |\mathcal{R}_\epsilon|} \cdot \tau_4^{|\mathcal{A}_\epsilon^{\mathcal{R}}| + |\overline{\mathcal{S}}_\epsilon|}\right) \\ &= (|\mathcal{C}_\epsilon| + |\mathcal{R}_\epsilon|) \log(\tau_2) + (|\mathcal{A}_\epsilon^{\mathcal{R}}| + |\overline{\mathcal{S}}_\epsilon|) \log(\tau_4) + \log(\Delta_0), \end{aligned}$$

what provides us

$$(|\mathcal{C}_\epsilon| + |\mathcal{R}_\epsilon|) \log(\tau_2) \geq \log(\Delta_{\min}) - \log(\Delta_0) - (|\mathcal{A}_\epsilon^{\mathcal{R}}| + |\overline{\mathcal{S}}_\epsilon|) \log(\tau_4).$$

Given that  $\tau_2 < 1$ , we have  $\log(\tau_2) < 0$ , it follows that

$$\begin{aligned} |\mathcal{C}_\epsilon| + |\mathcal{R}_\epsilon| &\leq \frac{\log(\Delta_{\min}) - \log(\Delta_0) - (|\mathcal{A}_\epsilon^{\mathcal{R}}| + |\overline{\mathcal{S}}_\epsilon|) \log(\tau_4)}{\log(\tau_2)} \\ &= \frac{\log(\Delta_0) - \log(\Delta_{\min})}{|\log(\tau_2)|} + (|\mathcal{A}_\epsilon^{\mathcal{R}}| + |\overline{\mathcal{S}}_\epsilon|) \frac{\log(\tau_4)}{|\log(\tau_2)|}. \end{aligned}$$

By the radii adjustment phase (line 14), we know that after an iteration in  $\overline{\mathcal{S}}_\epsilon$ , at most  $\Gamma_{\max}$  iterations of the type  $\mathcal{A}_\epsilon^{\mathcal{R}}$  can occur, and so,  $|\mathcal{A}_\epsilon^{\mathcal{R}}| \leq \Gamma_{\max} |\overline{\mathcal{S}}_\epsilon|$ . Therefore, it follows that

$$\begin{aligned} |\mathcal{N}_\epsilon| &= |\mathcal{C}_\epsilon| + |\mathcal{R}_\epsilon| + |\mathcal{A}_\epsilon^{\mathcal{R}}| \\ &\leq \frac{\log(\Delta_0) - \log(\Delta_{\min})}{|\log(\tau_2)|} + (\Gamma_{\max} + 1) |\overline{\mathcal{S}}_\epsilon| \frac{\log(\tau_4)}{|\log(\tau_2)|} + \Gamma_{\max} |\overline{\mathcal{S}}_\epsilon| \\ &= \left( \Gamma_{\max} + (\Gamma_{\max} + 1) \frac{\log(\tau_4)}{|\log(\tau_2)|} \right) |\overline{\mathcal{S}}_\epsilon| + \frac{\log(\Delta_0) - \log(\Delta_{\min})}{|\log(\tau_2)|}, \end{aligned}$$

what, together with the definition of  $c_3$ , finishes the proof.  $\square$

**Theorem 4.5.** *Under the conditions of Lemma 4.3, Algorithm 1 needs at most  $\mathcal{O}(\kappa_g^3 \epsilon^{-2})$  iterations to reach  $\pi_{k_\epsilon}^f < \epsilon$ .*

*Proof.* By Lemmas 4.3 and 4.4, we have

$$\begin{aligned} |\overline{\mathcal{S}}_\epsilon| + |\mathcal{N}_\epsilon| &\leq |\overline{\mathcal{S}}_\epsilon| + c_3 |\overline{\mathcal{S}}_\epsilon| + \frac{\log(\Delta_0) - \log(\Delta_{\min})}{|\log(\tau_2)|} \\ &= (1 + c_3) |\overline{\mathcal{S}}_\epsilon| + \frac{\log(\Delta_0) - \log(\Delta_{\min})}{|\log(\tau_2)|} \\ &\leq \frac{(f_{\min}(x_0) - M)}{\theta \eta_1 c_2} c_3 \Delta_{\min}^{-1} \epsilon^{-1} + \frac{\log(\Delta_0) + \log(\Delta_{\min}^{-1})}{|\log(\tau_2)|}. \end{aligned} \tag{15}$$

From Lemma 3.2, we have that  $c_1 = (L + \kappa_g + \kappa_H/2) \theta^{-1} = \mathcal{O}(\max\{\kappa_g, \kappa_H\})$ . Similarly, by Lemma 4.2,  $c_2^{-1} = 1 + \kappa_g \beta = \mathcal{O}(\kappa_g)$ . Hence, by letting  $\varepsilon = c_2 \epsilon$  in Corollary 3.3,

$$\begin{aligned} \Delta_{\min}^{-1} &= \max \left\{ \Delta_0^{-1}, \frac{\kappa_H}{\tau_1 c_2 \epsilon}, \frac{c_1}{\tau_1 (1 - \eta_1) c_2 \epsilon}, \frac{1}{\tau_1 \beta c_2 \epsilon}, \tau_1^{-1} \right\} \\ &= \mathcal{O}(\max\{\kappa_g, \kappa_H\} \kappa_g \epsilon^{-1}). \end{aligned} \tag{16}$$

Therefore, we get from (15) and (16) that

$$|\overline{\mathcal{S}}_\epsilon| + |\mathcal{N}_\epsilon| = \mathcal{O}(\max\{\kappa_g, \kappa_H\} \kappa_g^2 \epsilon^{-2}) = \mathcal{O}(\kappa_g^3 \epsilon^{-2}),$$

since  $\kappa_H = 2\kappa_g + L + 1$ .  $\square$

**Remark 4.6.** *Similarly to [3, Algorithm 3.2], we can allow resets in the trust-region radius of type  $\Delta_{k+1} = \max(\tau_3 \Delta_k, \Delta_0)$  at line 15 in the radii adjustment phase. However, this choice worsens the upper bound presented in Theorem 4.5 to  $\mathcal{O}(\epsilon^{-3})$ .*



**Remark 4.7.** Similarly to Garmanjani, Jádice and Vicente [22], we chose not to include assumptions about the order of magnitude of the  $\beta$  parameter, which acts in accessing the criticality phase (line 3). Although it is desirable that  $\beta$  is taken in an inversely proportional way to the choice of  $\epsilon$ , in the context of our algorithm that is not necessary to establish complexity results. Cartis and Roberts [9], when studying the worst-case complexity of algorithm DFO-GN, need to assume a hypothesis about the magnitude of the criticality phase parameter.

In the following result, we explicitly expand constant  $\kappa_g$  to show the worst-case complexity estimates for function evaluations. This is possible by further specifying how models  $m$  are constructed. We consider determined and underdetermined linear and quadratic polynomial models satisfying the inexact interpolation condition  $|m(y^j) - f(y^j)| \leq \kappa\delta^2$ , for each point  $y^j$  in a  $\Lambda$ -poised sample set  $\mathcal{Y} \subset \overline{B}(x_k, \delta_k)$ , where  $\kappa \geq 0$  is the inexact constant. Such calculations were given in [46]. Note that this interpolation condition naturally includes the classical interpolation models when  $\kappa = 0$ . Comments on how to build and maintain  $\Lambda$ -poised sets were made in the end of Section 3 and practical considerations are subject to the next section.

**Theorem 4.8.** Under the conditions of Lemma 4.3, let us assume that the models  $m_k$  are constructed by inexact linear or quadratic interpolation using a  $\Lambda$ -poised set  $\mathcal{Y} \subset \mathbb{R}^n$  of sample points,  $\Lambda > 0$ . Then, the number of function evaluations that Algorithm 1 needs in order to reach  $\pi_{k_\epsilon}^f \leq \epsilon$  is

- i)  $\mathcal{O}((n+r)n^3\epsilon^{-2})$  if the model is linear;
- ii)  $\mathcal{O}((r+n^2)n^{12}\epsilon^{-2})$  if the model is quadratic determined;
- iii)  $\mathcal{O}((r+p)n^{\frac{9}{2}}p^{\frac{15}{2}}\epsilon^{-2})$  if the model is quadratic underdetermined, for  $n < p < (n^2 + 3n)/2$  is the number of points used.

*Proof.* Initially, note that at each iteration, at most  $|\mathcal{Y}|$  evaluations of the function  $f_i$  are performed, for some index  $i \in \mathcal{I}$ , in order to build a  $\Lambda$ -poised set from scratch. Moreover, a single evaluation of the  $f_{min}$  function is necessary to evaluate  $\rho_k$ , which in turn depends on  $|\mathcal{I}| = r$  function evaluations. Thus, by Theorem 4.5, we will need at most  $\mathcal{O}((|\mathcal{Y}| + r)\kappa_g^3\epsilon^{-2})$  function evaluations during algorithm execution. We will separate the proof into three cases.

i) Linear case.

In this case  $|\mathcal{Y}| = n + 1$  and, by [46, Theorem 2.5] and [46, Lemma 2.6], we have that

$$\kappa_g = L + \left(\frac{L}{2} + 2\kappa\right)\Lambda n = \mathcal{O}(n).$$

Thus, we conclude that at most  $\mathcal{O}((n+r)n^3\epsilon^{-2})$  function evaluations are necessary.

ii) Quadratic case.

In this case  $|\mathcal{Y}| = (n^2 + 3n)/2 + 1 = q + 1$ . Again, using [46, Theorem 2.5]

and [46, Lemma 2.6], we obtain

$$\begin{aligned}
\kappa_g &= 2 \left( 4\Lambda \sqrt{(q+1)^3} \right) \sqrt{q}(1+\sqrt{2})(\kappa+L) \\
&= 8\Lambda(1+\sqrt{2})\sqrt{q}\sqrt{(q+1)^3}(\kappa+L) \\
&< 8\Lambda(1+\sqrt{2})(q+1)^2(\kappa+L) \\
&= 8\Lambda(1+\sqrt{2}) \left( \frac{n^2+3n+2}{2} \right)^2 (\kappa+L) \\
&= \mathcal{O}(n^4)
\end{aligned}$$

what gives to us at most  $\mathcal{O}((r+n^2)n^{12}\epsilon^{-2})$  function evaluations.

iii) Quadratic underdetermined case.

Finally, suppose that the models are quadratic underdetermined, with  $|\mathcal{Y}| = p+1$  points, where  $n < p < q$ ,  $q$  defined in case ii). Thus, by [46, Theorems 3.3, 3.10, 3.11], and by letting  $c_4 = \frac{3\Lambda}{c(\delta_{\max})^2} \left( \kappa + \frac{L}{2} \right)$ , it follows that

$$\begin{aligned}
\kappa_g &= 2\sqrt{p} \left( \Lambda \sqrt{2(n+1)(p+1)} \right) \left( L + \kappa + c_4(p+1) \sqrt{2(q+1)} \right) \\
&< 2\sqrt{2}\Lambda \sqrt{(n+1)} \sqrt{(p+1)^3} \left( L + \kappa + c_4 \sqrt{2}(p+1) \sqrt{(q+1)} \right) \\
&= \mathcal{O}(n^{\frac{3}{2}} p^{\frac{5}{2}}).
\end{aligned}$$

Hence, we will need at most  $\mathcal{O}((r+p)n^{\frac{9}{2}}p^{\frac{15}{2}}\epsilon^{-2})$  function evaluations.

□

It is also worth mentioning that in the context of derivative-free optimization, most of the worst-case complexity results presented in the literature are for direct-search methods of directional type based on a condition of sufficient decrease [22, p. 1988]. Among the works that study derivative-free trust-region methods with convergence to first-order stationary points, as Algorithm 1, we can highlight the bound  $\mathcal{O}(\epsilon^{-2})$  obtained by Garmanjani, Jádice and Vicente [22] for unconstrained composite optimization problems, which is also obtained in expectation by Gratton, Royer, Vicente and Zhang [25], but based on probabilistic models. Grapiglia, Yuan and Yuan [24], on the other hand, presents the bound  $\mathcal{O}(|\log(\epsilon)|\epsilon^{-2})$  for unconstrained composite nonsmooth problems and problems with equality constraints. Unfortunately, we are not aware of worst-case complexity results for derivative-free trust-region methods for problems with general convex constraints.

## 5 Numerical implementation and experiments

In this section, we will discuss some implementation details of Algorithm 1, which we will call **LOWDER**, an acronym for Low order-value Optimization Without DERivatives. We will also present the test problem sets adopted and the performance of our algorithm.

## 5.1 Implementation details

The **LOWDER** algorithm employs linear models to solve LOVO problems and has several practical improvements when compared to the theoretical algorithm presented in Section 2. One of the most significant changes occurs in definition of the relative reduction coefficient  $\rho_k$ . Note that we need to calculate  $\rho_k$  at each iteration in order to define the step acceptance, update, and radius correction phases. Since this involves evaluating the function  $f_{min}$ , which can be costly in computational terms, in practice, we allow **LOWDER** to employ the coefficient proposed by Castelani, Lopes, Shirabayashi and Sobral [10]:

$$\hat{\rho}_k = \frac{f_{i_k}(x_k) - f_{i_k}(x_k + d_k)}{m_k(x_k) - m_k(x_k + d_k)},$$

where  $i_k \in \mathcal{I}_{min}(x_k)$ , for up to  $\mathbf{nrhomax} \in \mathbb{N}$  consecutive iterations. After  $\mathbf{nrhomax}$  iterations  $\rho_k$  has to be calculated as presented in (5). Note that  $\hat{\rho}_k \leq \rho_k$ , and so we are being more demanding about the quality of the models and the decrease obtained. During the preliminary tests,  $\mathbf{nrhomax} = 3$  proved to be efficient.

Another relevant change is the way the solution  $d_k \in \mathbb{R}^n$  of the trust region subproblem (4) is incorporated into the sample set  $\mathcal{Y}$ . To do so, we implemented simplified versions of the **TRSBOX** and **ALTMov** functions, developed by Powell [41] for the **BOBYQA** algorithm. Initially, we compute a solution  $d_k^{\text{TRS}}$  using the **TRSBOX** algorithm for the linear case. If  $\|d_k^{\text{TRS}}\| \geq \frac{\Delta_k}{2}$  and  $\rho_k > 0$  (or  $\hat{\rho}_k > 0$ ), the point  $x_{new} = x_k + d_k^{\text{TRS}}$  is inserted into  $\mathcal{Y}$ . Otherwise, we run **ALTMov** to look for an alternative direction  $d_k^{\text{ALT}}$ , and add the point  $x_{new} = x_k + d_k^{\text{ALT}}$  to  $\mathcal{Y}$ . At each iteration of **LOWDER**, only one point is added to  $\mathcal{Y}$ , while another point is removed, following procedures similar to **BOBYQA**.

In this first version, **LOWDER** employs only linear models to solve problem (1). In this sense, **LOWDER** uses the sample set construction mechanism of **BOBYQA**. As with this solver, we avoided fully rebuilding the models due to the computational cost involved. In general, the model and sample set are only rebuilt if one of the following conditions is satisfied:

- i) An index swap occurs, that is  $i_{k+1} \neq i_k$ , and the direction calculated by **TRSBOX** satisfies  $\rho_k \geq \eta$  (or  $\hat{\rho}_k \geq \eta$ ); or yet,
- ii) An index swap occurs,  $\rho_k \geq 0$  (or  $\hat{\rho}_k \geq 0$ ) and  $f_{min}(x_{new})$  is available.

In all other cases, the model is updated using the current sample set and the information obtained about the point  $x_{new}$ . Recall that only evaluations of  $f_{i_{k+1}}$  are needed to rebuild the model. Each time the model is rebuilt or updated, we also calculate the QR factorization of the matrix  $\mathbf{L}_L$  defined in [46, Section 2]. This information is needed to calculate  $d_k^{\text{ALT}}$  given by **ALTMov** and the point that should leave set  $\mathcal{Y}$  in case direction  $d_k^{\text{TRS}}$  is accepted.

Given that the radius of the sample region  $\delta_k$  controls the quality of the model, a solution  $\tilde{x} = x_k \in \Omega$  is declared successful if it satisfies  $\delta_k \leq \delta_{\min}$  and  $\beta\pi_k \leq \delta_{\min}$  (or  $\beta\bar{\pi}_k \leq \delta_{\min}$ ), where  $\delta_{\min} > 0$  is a parameter defined by the user. The default values for  $\beta$  and  $\delta_{\min}$  are 1 and  $10^{-8}$ , respectively. We say that **LOWDER** is stalled at iteration  $k$  if there is no more room for improvement in the sample set and the model. This situation translates into the case where  $\delta_k \leq \delta_{\min}$ ,  $\Delta_k \leq \delta_{\min}$  and it is not possible to perform any more **ALTMov**

iterations. In this case, the maximum number of consecutive iterations of the **ALTM** type is controlled by the `maxalt`  $\in \mathbb{N}$  parameter, whose default value is  $|\mathcal{Y}| - 1$ . Note that if the sample and trust-region radii have values greater than  $\delta_{\min}$ , we allow more than `maxalt` consecutive iterations with **ALTM** directions. However, as soon as the stalled criterion is satisfied the algorithm exits. For safety, the algorithm also exits if more than `maxcrit`  $\in \mathbb{N}$  successive iterations access the criticality phase (line 3), whose default value is  $|\mathcal{Y}| - 1$ .

The **LOWDER** solver was implemented in the **Julia** language, version 1.6.1, and is available in the repository:

<https://github.com/aschwertner/LOWDER>

## 5.2 Numerical experiments

To benchmark the **LOWDER**'s performance in solving LOVO black-box problems, we compared our solver with other algorithms able of solving derivative-free optimization problems. These algorithms are Manifold Sampling Primal (**MS-P**) and Nonlinear Optimization with the MADS algorithm (**NOMAD**).

The **MS-P** solver is a **MatLab** implementation of the Manifold Sampling method proposed in [30, 31], and is designed to solve bound-constrained nonsmooth composite minimization problems

$$\min_{x \in \Omega} f(x) = \min_{x \in \Omega} h(F(x)),$$

where  $F : \mathbb{R}^n \rightarrow \mathbb{R}^p$ ,  $h : \mathbb{R}^p \rightarrow \mathbb{R}$  is a continuous selection (see [30, Definition 1]), and the feasible set is a subset of the  $n$ -fold Cartesian product of the extended reals  $\mathbb{R} \cup \{-\infty, \infty\}$  defined by bound constraints of the form  $\Omega = \{x : l \leq x \leq u\}$ . The manifold sampling algorithm was initially proposed by Larson, Menickelly and Wild [31] as a variant of gradient sampling, and can be classified as model-based derivative-free method. In this method, models of  $F$  are combined with sampled information about the function  $h$  to construct local models called smooth master models, for use within a trust-region framework. **MS-P** builds fully linear quadratic models using the interpolation and regression mechanisms of **POUNDERS** [52]. We are grateful to Jeffrey Larson, co-author of the method, for providing the **MS-P** and **POUNDERS** codes and being available to help us.

The **NOMAD** software is a **C++** implementation of the Mesh Adaptive Direct Search (MADS) algorithm, solves black-box optimization problems in general, and uses the progressive barriers method to deal with problems with constraints. In this work, we use version 4.2.0 of **NOMAD** [6] through the interface for the **Julia** language called **NOMAD.jl**, version 2.2.1 [36]. This version of **NOMAD** presents a series of improvements compared to previous versions, especially in comparison to version 3.9, such as a new software architecture and support for new pool mechanisms and search algorithms, among others. All the algorithms were run with their default parameters on an AMD Ryzen 7 1700X 3.40GHz with 8 cores (16 threads) and 16GB of RAM and Linux Ubuntu Budgie 22.04.1 LTS operating system.

Our benchmark suite includes three sets of test problems denominated by MW, HS, and QD. The MW set contains problems proposed by Moré and Wild in [37] for benchmarking derivative-free optimization algorithms and comprises 53 unconstrained problems. These problems are variations of 22 nonlinear least

squares functions taken from the **CUTEr** collection [23]. Each function is defined by  $r$  components in  $n$  variables. By combining different values for  $n$  and  $r$ , and distinct starting points, we obtain all the problems in MW test set.

The HS test set includes 87 bound-constrained problems selected and modified from the original collection published by Hock and Schittkowski [28] for testing nonlinear programming algorithms. Initially, we selected 8 problems from [28] with bound constraints and different objective functions. By combining two, three, or four of these problems, a new objective function  $f_{min}$  is defined. The problem dimension is the largest dimension among the combined problems, and the bound constraints are taken as the intersection of the original boxes. If this procedure generates constraints with fixed values, that is  $l_i = u_i$  for some index  $i \in \{1, \dots, n\}$ , then the problem is discarded.

Lastly, the QD test set has 5 subsets with 50 problems each, all with bound constraints. The purpose of this test set is to demonstrate the impact of the number of functions  $f_i$  of the objective function  $f_{min}$  in the performance of the algorithms. In this sense, the subsets of QD consist of problems with an increasing number of component functions based on the general formula:

$$f_i(x) = 5^i + \frac{1}{2} \sum_{j=1}^n a_j^i \cdot (x_j - b_j^i)^2,$$

where  $a^i \in [0, 1000]^n$ , and  $b^i \in [0, 10]^n$  are vectors built by the pseudorandom number generator **MarsenneTwister** from the **Random** package of the **Julia** language, with the same seed. We generate the problems in the test subsets with  $r \in \{10, 25, 50, 75, 100\}$  component functions, respectively. We employ the same seed to generate vectors  $a^i$  and  $b^i$  to ensure that the generated component functions come into problems belonging to test sets with larger  $r$  values. In this way, the component functions generated for the QD10 test subset are present in the other subsets, and they are complemented by new functions  $f_i$  as the value of  $r$  increases. In our case, all problems were generated with dimension  $n = 10$ , have the same bound constraints  $l = [0, \dots, 0]$  and  $u = [10, \dots, 10]$ , and the starting point  $x_0 \in \mathbb{R}^n$  is set to the center of the box. Figure 1 illustrates a two dimensional example of objective function generated by this procedure, with 10 component functions.

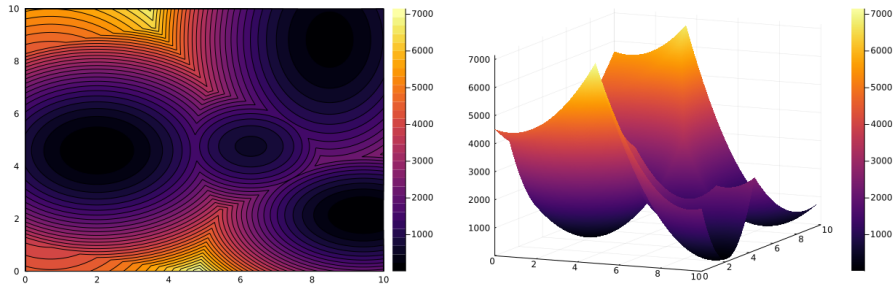


Figure 1: Contour plot (left) and surface plot (right) for an example of objective function  $f_{min}$  defined by the generating function of QD test set.

We also present in Table 1 the maximum and minimum values for the dimension  $n \in \mathbb{N}$  and the number of component functions  $r \in \mathbb{N}$  of the problems

of each test set.

Test set	$n_{\min}$	$n_{\max}$	$r_{\min}$	$r_{\max}$
MW	2	12	2	65
HS	2	10	2	4
QD	10	10	10	100

Table 1: Dimension and number of component functions of MW, HS and QD test sets.

We use data profiles [37] to compare the performance of **LOWDER**, **MS-P**, and **NOMAD**, over our benchmark test suite. We are interested in comparing the function values obtained by each of the algorithms. Thus, we consider that a method has solved a problem with tolerance level  $\tau > 0$  after  $t$  function evaluations if the iterate  $x^t$  satisfies

$$f(x^t) \leq f_L + \tau (f(x_0) - f_L), \quad (17)$$

where  $x_0$  is the starting point of the problem common to all algorithms, and  $f_L$  is the smallest function value obtained by the solvers for a given budget of function evaluations. Following Moré and Wild's suggestion [37, Section 5], we decided to investigate the behavior of algorithms with a limit of 100 simplex gradients of the objective function, where one simplex gradient is defined by  $n + 1$  function evaluations. Since **MS-P** and **NOMAD** always evaluate function  $f_{\min}$  completely and **LOWDER** has the ability to run each component function  $f_i$ ,  $i \in \mathcal{I}$ , independently, for each set of tests considered, we define the budget as  $100(n_{\max} + 1)$  evaluations of  $f_{\min}$  for **MS-P** and **NOMAD**, and  $100r_p(n_{\max} + 1)$  evaluations of  $f_i$  for **LOWDER**, where  $r_p$  is the number of component functions of the problem  $p$ . Therefore, we allow the algorithms to run at least 100 simplex gradients of the objective function  $f_{\min}$  for each problem in the test set.

To build the data profiles, we recorded the function values of  $f_{\min}$  accessed by the **MS-P** and **NOMAD** solvers and the function values of  $f_i$  computed by **LOWDER** for each one of the selected problems. Once the solver satisfies the criterion (17) for a problem  $p$  for the first time after  $t$  function evaluations, its data profile row is incremented on the vertical axis by  $\frac{1}{|\mathcal{P}|}$  at the point  $\frac{t}{(n_p+1)}$ , in the case of **MS-P** and **NOMAD**, or at the point  $\frac{t}{r_p(n_p+1)}$ , for **LOWDER**. Therefore, our metric of simplex gradient evaluations for  $f_{\min}$  is maintained.

The codes used to generate the numerical tests and data profiles presented in this chapter are available at:

[https://github.com/aschwertner/LOWDER\\_Numerical\\_Tests](https://github.com/aschwertner/LOWDER_Numerical_Tests)

### 5.2.1 MW test set

The MW set is our main problem test set since it was designed by Moré and Wild [37], especially to benchmark unconstrained derivative-free algorithms. In Figure 2, we present four data profiles for the MW test set. Each plot shows the percentage of problems solved for a specified tolerance  $\tau$  as a function of a computational budget of simplex gradients of  $f_{\min}$ . As we can see, **LOWDER** and **MS-P** solve around 90% of the problems for all tolerance levels  $\tau$ , up to the

fixed budget. The difference in robustness between this two solvers is less than 5%. We also note that the behavior of **NOMAD** is very different for distinct levels of tolerance, ranging from around 60% to less than 40%, with  $\tau = 10^{-1}$  and  $\tau = 10^{-7}$ , respectively.

**LOWDER** is very efficient at solving problems in the MW test suite, solving about 90% of the problems with less than 20 simplex gradients. **MS-P** has a behavior very close to **LOWDER**, outperforming the latter by about 2% for  $\tau = 10^{-1}$ , but loses performance as the tolerance decreases, solving only 80% of problems for  $\tau = 10^{-7}$ . Although this behavior is expected, since **LOWDER** uses mainly information of the  $f_i$ 's, we must recall that **MS-P** uses quadratic models, while **LOWDER** uses linear ones. For this computational budget and  $\tau = 10^{-1}$ , **NOMAD** can solve 50% of the problems in the test set. However its performance drops and stabilizes at around 40% for other values of tolerance. Inspecting the **NOMAD** execution data, we can see three distinct behaviors that can explain its low performance in the MW test set: difficulty in reducing the objective function, low convergence rate, and convergence to weak stationary points that are local minima, while **LOWDER** and **MS-P** can converge to global minimum points.

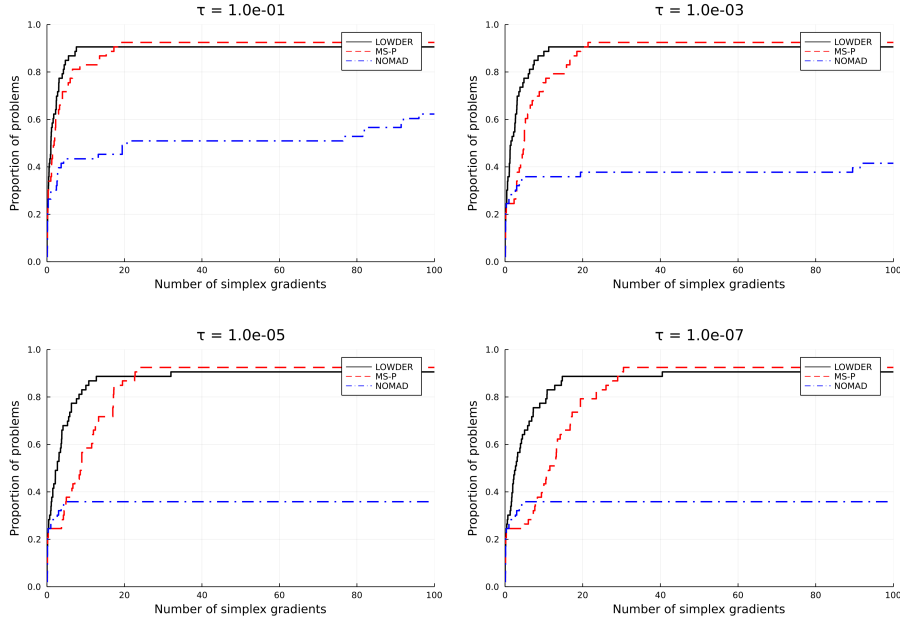


Figure 2: Data profiles for the problems in MW test set with tolerance  $\tau \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$ .

### 5.2.2 HS test set

Analyzing Figure 3, we can see that **NOMAD** can solve all problems for  $\tau = 10^{-1}$  and can solve 95% or more problems for other values of tolerance. **LOWDER** and **MS-P** have similar performances, alternating in the second position. When considering the computational budget of 100 simplex gradients, **LOWDER** is overcome by **MS-P** by 1% to 3%. For budget values between 20 and 40 simplex gradients, **LOWDER** performs slightly better than **MS-P**.

The great performance of **NOMAD** can be explained by the small size of the problems and by the recognized ability of the algorithm to solve complex problems, especially for non-linear problems with constraints, as is the case of this test set. Note that **LOWDER** employs only linear models to solve the trust-region subproblem (4). This can harm its performance in the case of strongly non-linear problems since the solutions generated by **TRSBOX** tend not to satisfy the conditions of the step acceptance phase (line 8), favoring set improvement iterations with directions calculated by **ALTMOV**, which may not be directions in which the objective function decrease. Another factor that can impact the quality of the solutions obtained by **LOWDER** is the existence of several problems in HS test set that have an infinity of local minima. **NOMAD** has two distinct mechanisms that help it escape from local minima. The first one is a global search procedure called **SEARCH** step, which can return any point on the underlying mesh, always trying to identify points that improve the best solution found so far. The second is a local search called the **POOL** step, and your purpose is to generate trial mesh points in the vicinity of the best solution [6]. Another behavior of the **NOMAD** solver that we noticed during the execution of these problems was its tendency to find local and global solutions on the boundary of the feasible set. Among the 87 problems that constitute the HS test set, **NOMAD** found boundary solutions for 55 of them. Considering its good performance in the data profile, this suggests the solver is very efficient when exploring and evaluating the limits of the feasible set.

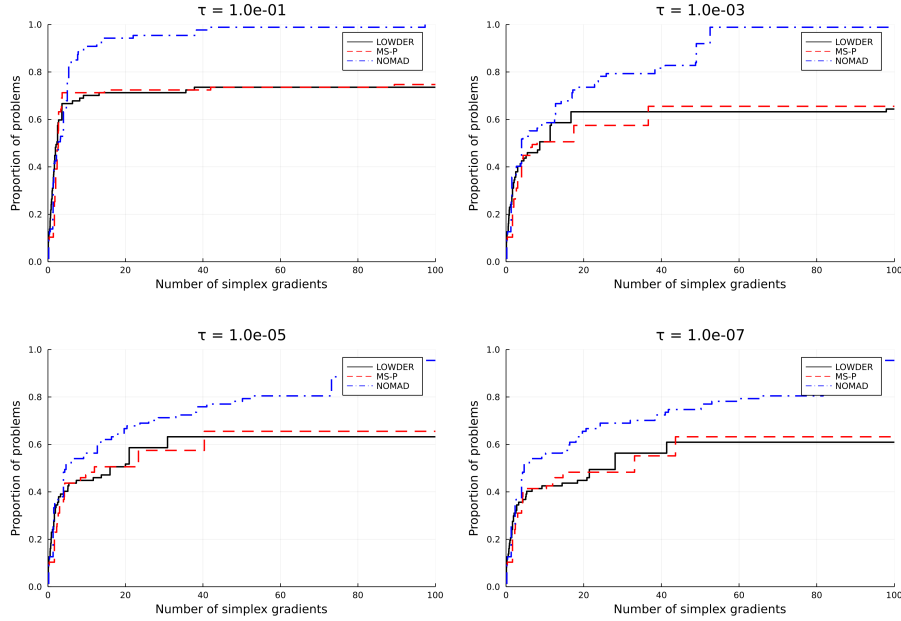


Figure 3: Data profiles for the problems in HS test set with tolerance  $\tau \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$ .

When looking at the proportion of problems solved by the three solvers, we can see that they can solve about 40% of problems with 5 simplex gradients or less. From that point on, the performance differences are quite significant,



especially when comparing NOMAD with the other solvers. By setting  $\tau = 10^{-3}$ , NOMAD can solve 60% of problems with just 13 simplex gradients, while LOWDER needs 17, and MS-P needs about 37 simplex gradients. Another situation worth mentioning occurs with  $\tau = 10^{-7}$ . In this case, considering the 50% mark of problems solved, NOMAD can reach this value with 6 simplex gradients, while LOWDER and MS-P need approximately 28 and 33 simplex gradients, respectively.

### 5.2.3 QD test set

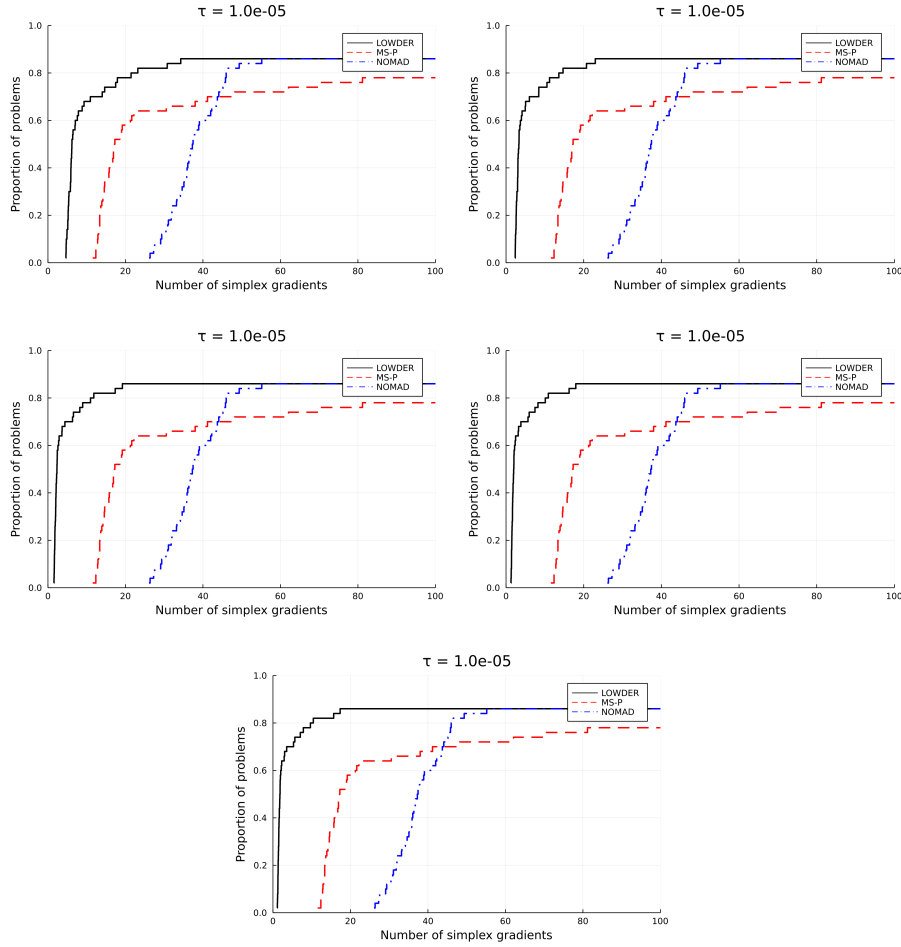


Figure 4: Data profiles for the problems in QD test set, with  $\tau = 10^{-5}$ , and  $r \in \{10, 25, 50, 75, 100\}$ .

Our goal in this test set is to show the benefits that LOWDER enjoys due to exploring the structure of LOVO problems. Figure 4 presents the data profiles generated with tolerance  $\tau = 10^{-5}$  for the problems in the QD test set, taking into account the subsets generated with 10, 25, 50, 75, and 100 component functions. Analyzing the plots presented, if we consider a computational budget of 100 simplex gradients, the robustness of the algorithms is not affected by the number of functions  $f_i$ . LOWDER and NOMAD manage to solve approximately

85% of the problems while **MS-P** solves 77% of the problems, in the 5 scenarios presented. Note that **LOWDER** outperforms **MS-P** and **NOMAD** by a large amount, especially for computational budgets of less than 40 simplex gradients. For this fixed budget, **MS-P** also outperforms **NOMAD**, but the situation reverses for higher budget values. More specifically, the data profile shows us that for robustness rates above 70%, **NOMAD** is more efficient than **MS-P** and is able to match **LOWDER** using less than 60 simplex gradients.

Another relevant fact is that the curves presented by **MS-P** and **NOMAD** do not appear to be affected by the variation in the number of component functions, while the **LOWDER** data profile curve has a clear tendency to approach the vertical axis according to the number of component functions increases. That is, the performance of **LOWDER** tends to improve, especially when considering small computational budgets, with less than 20 simplex gradients. One way to visualize this influence is to verify the number of simplex gradients needed to solve a certain percentage of problems, as shown in Table 2.

Test subset	20%	40%	60%	80%	85%
QD10	5	6	7	21	34
QD25	3	3	4	14	23
QD50	2	2	3	11	19
QD75	2	2	2	10	18
QD100	1	2	2	9	17

Table 2: Approximate number of function simplex gradients that **LOWDER** needs to solve given ranges of problems on the QD test subsets.

Note that as the number of component functions increases, represented by the number associated with the test subset, there is a tendency for the number of simplex gradients to decrease, and this can be verified for all ranges of problems considered.

## 6 Conclusions

In this work, we introduced a new class of low order-value optimization methods, considering an approach reasoned on model-based derivative-free optimization. We presented a derivative-free trust-region algorithm for constrained black-box LOVO problems, which is based on the algorithms proposed by Conejo, Karas, Pedroso, Ribeiro and Sachine [12] and Verdério, Karas, Pedroso and Scheinberg [50], and the ideas discussed by Andreani, Martínez and Martínez [3]. Our algorithm can deal with general closed convex constraints and is specially designed for problems whose objective function values are provided through an oracle (black-box function). Note that we assume that we know how to project an arbitrary point onto the feasible set. Algorithm 1 has a structure very similar to the traditional trust-region framework and considers two different radii, one for the sample region and another for the trust-region. We allowed some freedom in the choice of models, as long as the gradient of the model is a good approximation of the gradient of the selected component function, in the sense of well poisedness (Assumption 2.4).

We discussed global convergence results of the algorithm, adopting common assumptions for this class of problems, as well as an interpretation from the perspective of the classical theory of LOVO problems. Inspired by the works of Cartis and Roberts [9] and Garmanjani, Júdice and Vicente [22], we also studied the worst-case complexity analysis of Algorithm 1, which showed us that the number of iterations and function evaluations performed by the algorithm is in line with what is expected for model-based methods, and that the adoption of minimum Frobenius-norm quadratic models is competitive in terms of function evaluations when compared to complete determined models.

We also presented **LOWDER**, our implementation of Algorithm 1 in **Julia** language, discussed implementation details and performed numerical tests. In its current version, **LOWDER** solves bound-constrained black-box LOVO problems and builds only determined linear models. **LOWDER** has several practical improvements, many of them derived from **BOBYQA**, a general-purpose derivative-free optimization solver for bound-constrained problems. In particular, **LOWDER** inherits the initial sampling mechanisms and has simplified versions of the **TRSBX** and **ALTMOV** routines for solving the trust region subproblem (4) and improving the geometry of the sample set, respectively. Like **BOBYQA**, we also avoided completely rebuilding models through an update mechanism.

Since **LOWDER** is designed for LOVO problems, we compared it with algorithms that can handle, in some way, this type of problem. In this sense, we selected **MS-P** [30] and **NOMAD** [6]. **MS-P** is a manifold sampling algorithm for composition minimization problems. **NOMAD** is a well-established algorithm for black-box optimization problems based on the direct search.

We proposed a test suite with three sets: **MW**, **HS**, and **QD**. **MW** is our main test set and contains the problems of Moré and Wild [37] for benchmarking derivative-free optimization algorithms. **HS** is a test set created with a combination of problems from the Hock and Schittkowski [28] collection for testing nonlinear optimization algorithms. Finally, **QD** is a test set created by us to measure the impact of the number of component functions on the performance of **LOWDER**. Despite not being the most robust algorithm, **LOWDER** can solve about 90% of the problems from **MW** with less than 20 simplex gradients of the objective function, being the most efficient algorithm for this range of computational budget. In the **QD** test set, **LOWDER** is the most efficient and robust algorithm. Although **NOMAD** is the least efficient algorithm for budgets smaller than 40 simplex gradients, it outperforms **MS-P** and matches **LOWDER** for larger budgets. Despite having similar performances for budgets of up to 5 simplex gradients in the **HS** test set, in general, **NOMAD** had the best performance and robustness, managing to solve practically all problems with a budget of 100 simplex gradients. **MS-P** and **LOWDER** obtain similar performances, solving about 70% of the problems with tolerance  $\tau = 10^{-1}$  and little more than 60% of the problems considering smaller values of  $\tau$ . The worse performance of **LOWDER** can be explained by the fact that we employ only linear models, and since the problems in **HS** are strongly nonlinear, this fact can impair the acceptance phases of the step and the general progress of the algorithm.

In order to increase the performance of **LOWDER**, we can implement the construction of determined and underdetermined quadratic models, based on the mechanisms proposed by Powell [41] for the **BOBYQA** solver, and also modified versions of **RESCUE**, to avoid the full reconstruction of the models. In addition, we can improve the implementation of the linear models using more efficient

ways to calculate and update the QR factorization. Another possible advance is the usage of the sampling strategies presented by Hough and Roberts [29] and thus avoiding situations for which  $\Lambda$ -poised sets are impossible to be constructed in constrained problems.

Furthermore, we can implement a mechanism of long-term memory and store relevant information about old sample points, such as objective function value, component function index, and stationarity measure. This information can be useful in constructing new sample sets and saving objective function evaluations. When interpreting LOVO as a nonsmooth composite minimization problem, such a memory mechanism can also add information about the minimum function when we calculate the function  $f_{min}$  completely, similar to what happens in the manifold sampling algorithms proposed in [30, 32].

Finally, it is well known that the Low Order-Value Optimization can generalize the nonlinear least-squares problem, as it allows us to discard observations considered outliers, as we can see in [4, 10]. Therefore, we can enhance **LOWDER** to take advantage of the structure of the least-squares problem, such as well-established algorithms like **DF0-GN** [9], **POUNDERS** [52], and **DFBOLS** [56], making it a competitive solver in this segment.

## References

- [1] G. Q. Álvarez, E. G. Birgin and J. M. Martínez. ‘A first-order regularized algorithm with complexity properties for the unconstrained and the convexly constrained low order-value optimization problem’. In: *Journal of Global Optimization* (2025). DOI: 10.1007/s10898-025-01521-5.
- [2] R. Andreani, G. Haeser and J. M. Martínez. ‘On sequential optimality conditions for smooth constrained optimization’. In: *Optimization* 60.5 (2011), pp. 627–641.
- [3] R. Andreani, J. M. Martínez and L. Martínez. ‘Trust-region superposition methods for protein alignment’. In: *IMA journal of numerical analysis* 28.4 (2008), pp. 690–710.
- [4] R. Andreani, J. M. Martínez, L. Martínez and F. S. Yano. ‘Low order-value optimization and applications’. In: *Journal of Global Optimization* 43.1 (2009), pp. 1–22.
- [5] C. Audet and W. Hare. *Derivative-free and blackbox optimization*. Springer, 2017.
- [6] C. Audet, S. Le Digabel, V. Rochon Montplaisir and C. Tribes. ‘Algorithm 1027: NOMAD version 4: Nonlinear optimization with the MADS algorithm’. In: *Transactions on Mathematical Software* 48.3 (2022), 35:1–35:22.
- [7] E. G. Birgin, L. F. Bueno, N. Krejić and J. M. Martínez. ‘Low order-value approach for solving VaR-constrained optimization problems’. In: *Journal of Global Optimization* 51.4 (2011), pp. 715–742.
- [8] L. F. Bueno, A. Friedlander, J. M. Martínez and F. N. C. Sobral. ‘Inexact restoration method for derivative-free optimization with smooth constraints’. In: *SIAM Journal on Optimization* 23.2 (2013), pp. 1189–1213.

- [9] C. Cartis and L. Roberts. ‘A derivative-free Gauss-Newton method’. In: *Mathematical Programming Computation* 11.4 (2019), pp. 631–674.
- [10] E. V. Castelani, R. Lopes, W. V. I. Shirabayashi and F. N. C. Sobral. ‘A robust method based on LOVO functions for solving least squares problems’. In: *Journal of Global Optimization* 80.2 (2021), pp. 387–414.
- [11] P. D. Conejo, E. W. Karas and L. G. Pedroso. ‘A trust-region derivative-free algorithm for constrained optimization’. In: *Optimization Methods and Software* 30.6 (2015), pp. 1126–1145.
- [12] P. D. Conejo, E. W. Karas, L. G. Pedroso, A. A. Ribeiro and M. Sachine. ‘Global convergence of trust-region algorithms for convex constrained minimization without derivatives’. In: *Applied Mathematics and Computation* 220 (2013), pp. 324–330.
- [13] A. R. Conn, N. I. M. Gould, A. Sartenaer and P. L. Toint. ‘Convergence properties of minimization algorithms for convex constraints using a structured trust region’. In: *SIAM Journal on Optimization* 6.4 (1996), pp. 1059–1086.
- [14] A. R. Conn, N. I. M. Gould and P. L. Toint. ‘Global convergence of a class of trust region algorithms for optimization with simple bounds’. In: *SIAM Journal on Numerical Analysis* 25.2 (1988), pp. 433–460.
- [15] A. R. Conn, N. I. M. Gould and P. L. Toint. *Trust-region methods*. SIAM, 2000.
- [16] A. R. Conn, K. Scheinberg and P. L. Toint. ‘A derivative free optimization algorithm in practice’. In: *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. 1998, p. 4718.
- [17] A. R. Conn, K. Scheinberg and P. L. Toint. ‘On the convergence of derivative-free methods for unconstrained optimization’. In: *Approximation theory and optimization: tributes to MJD Powell* (1997), pp. 83–108.
- [18] 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’. In: *SIAM Journal on Optimization* 20.1 (2009), pp. 387–415.
- [19] A. R. Conn, K. Scheinberg and L. N. Vicente. *Introduction to derivative-free optimization*. Philadelphia: SIAM, 2009.
- [20] G. Fasano, J. L. Morales and J. Nocedal. ‘On the geometry phase in model-based algorithms for derivative-free optimization’. In: *Optimization Methods & Software* 24.1 (2009), pp. 145–154.
- [21] P. S. Ferreira, E. W. Karas, M. Sachine and F. N. C. Sobral. ‘Global convergence of a derivative-free inexact restoration filter algorithm for nonlinear programming’. In: *Optimization* 66.2 (2017), pp. 271–292.
- [22] R. Garmanjani, D. Júdice and L. N. Vicente. ‘Trust-region methods without using derivatives: worst case complexity and the nonsmooth case’. In: *SIAM Journal on Optimization* 26.4 (2016), pp. 1987–2011.
- [23] N. I. M. Gould, D. Orban and P. L. Toint. ‘CUTEr and SifDec: A constrained and unconstrained testing environment, revisited’. In: *ACM Transactions on Mathematical Software (TOMS)* 29.4 (2003), pp. 373–394.

- [24] G. N. Grapiglia, J. Yuan and Y.-x. Yuan. ‘A derivative-free trust-region algorithm for composite nonsmooth optimization’. In: *Computational and Applied Mathematics* 35.2 (2016), pp. 475–499.
- [25] S. Gratton, C. W. Royer, L. N. Vicente and Z. Zhang. ‘Complexity and global rates of trust-region methods based on probabilistic models’. In: *IMA Journal of Numerical Analysis* 38.3 (2018), pp. 1579–1597.
- [26] S. Gratton, P. L. Toint and A. Tröltzsch. ‘An active-set trust-region method for derivative-free nonlinear bound-constrained optimization’. In: *Optimization Methods and Software* 26.4-5 (2011), pp. 873–894.
- [27] E. A. E. Gumma, M. H. A. Hashim and M. M. Ali. ‘A derivative-free algorithm for linearly constrained optimization problems’. In: *Computational Optimization and Applications* 57.3 (2014), pp. 599–621.
- [28] W. Hock and K. Schittkowski. ‘Test examples for nonlinear programming codes’. In: *Journal of optimization theory and applications* 30.1 (1980), pp. 127–129.
- [29] M. Hough and L. Roberts. ‘Model-Based Derivative-Free Methods for Convex-Constrained Optimization’. In: *SIAM Journal on Optimization* 32.4 (2022), pp. 2552–2579.
- [30] J. Larson and M. Menickelly. ‘Structure-aware methods for expensive derivative-free nonsmooth composite optimization’. In: *Mathematical Programming Computation* (2023). DOI: 10.1007/s12532-023-00245-5.
- [31] J. Larson, M. Menickelly and S. M. Wild. ‘Derivative-free optimization methods’. In: *Acta Numerica* 28 (2019), pp. 287–404.
- [32] J. Larson, M. Menickelly and B. Zhou. ‘Manifold sampling for optimizing nonsmooth nonconvex compositions’. In: *SIAM Journal on Optimization* 31.4 (2021), pp. 2638–2664.
- [33] J. M. Martínez. ‘Generalized order-value optimization’. In: *Top* 20.1 (2012), pp. 75–98.
- [34] J. M. Martínez. ‘Order-value optimization and new applications’. In: *ICIAM 07: 6th International Conference on Industrial and Applied Mathematics, Zürich, Switzerland, 16-20 July 2007: Invited Lectures*. European Mathematical Society. 2009, pp. 279–296.
- [35] L. Martínez, R. Andreani and J. M. Martínez. ‘Convergent algorithms for protein structural alignment’. In: *BMC bioinformatics* 8.1 (2007), pp. 1–15.
- [36] A. Montoison, P. Pascal and L. Salomon. *NOMAD.jl: A Julia interface for the constrained blackbox solver NOMAD*. <https://github.com/bbopt/NOMAD.jl>. 2020. DOI: 10.5281/zenodo.3700167.
- [37] J. J. Moré and S. M. Wild. ‘Benchmarking Derivative-Free Optimization Algorithms’. In: *SIAM Journal on Optimization* 20.1 (2009), pp. 172–191.
- [38] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
- [39] B. L. Pelegriani, F. M. B. Fernandes, T. Fernandes et al. ‘Novel green strategy to improve the hydrophobicity of cellulose nanocrystals and the interfacial elasticity of Pickering emulsions’. In: *Cellulose* 28.10 (2021), pp. 6201–6238.

- [40] M. J. D. Powell. ‘On fast trust region methods for quadratic models with linear constraints’. In: *Mathematical Programming Computation* 7.3 (2015), pp. 237–267.
- [41] M. J. D. Powell. ‘The BOBYQA algorithm for bound constrained optimization without derivatives’. In: *Cambridge NA Report NA2009/06*, University of Cambridge, Cambridge (2009), pp. 26–46.
- [42] M. J. D. Powell. ‘The NEWUOA software for unconstrained optimization without derivatives’. In: *Large-scale nonlinear optimization*. Boston: Springer, 2006, pp. 255–297.
- [43] M. J. D. Powell. ‘UOBYQA: unconstrained optimization by quadratic approximation’. In: *Mathematical Programming* 92.3 (2002), pp. 555–582.
- [44] L. M. Rios and N. V. Sahinidis. ‘Derivative-free optimization: a review of algorithms and comparison of software implementations’. In: *Journal of Global Optimization* 56.3 (2013), pp. 1247–1293.
- [45] L. Roberts. ‘Model Construction for Convex-Constrained Derivative-Free Optimization’. In: *SIAM Journal on Optimization* 35.2 (2025), pp. 622–650. DOI: 10.1137/24m1649113.
- [46] A. E. Schwertner and F. N. C. Sobral. ‘On complexity constants of linear and quadratic models for derivative-free trust-region algorithms’. In: *Optimization Letters* 19.5 (2025), pp. 919–930. DOI: 10.1007/s11590-024-02147-4.
- [47] J. Thomann and G. Eichfelder. ‘A trust-region algorithm for heterogeneous multiobjective optimization’. In: *SIAM Journal on Optimization* 29.2 (2019), pp. 1017–1047.
- [48] A. Tröltzsch. ‘An active-set trust-region method for bound-constrained nonlinear optimization without derivatives applied to noisy aerodynamic design problems’. PhD thesis. Institut National Polytechnique de Toulouse-INPT, 2011.
- [49] A. Tröltzsch, C. Ilic and M. Siggel. ‘SQPDFO-a Trust-Region Based Algorithm for Generally-Constrained Derivative-Free Optimization’. In: *Proceedings of the 13th AMiTaNS* (2021).
- [50] A. Verdério, E. W. Karas, L. G. Pedroso and K. Scheinberg. ‘On the construction of quadratic models for derivative-free trust-region algorithms’. In: *EURO J. Comput. Optim.* 5.4 (2017), pp. 501–527.
- [51] S. M. Wild. ‘Derivative-free optimization algorithms for computationally expensive functions’. PhD thesis. Ithaca: Cornell University, 2008.
- [52] S. M. Wild. ‘POUNDERS in TAO: Solving Derivative-Free Nonlinear Least-Squares Problems with POUNDERS’. In: *Advances and Trends in Optimization with Engineering Applications*. MOS-SIAM Series on Optimization. SIAM, 2017. Chap. 40, pp. 529–539.
- [53] S. M. Wild, R. G. Regis and C. A. Shoemaker. ‘ORBIT: Optimization by radial basis function interpolation in trust-regions’. In: *SIAM Journal on Scientific Computing* 30.6 (2008), pp. 3197–3219.
- [54] D. H. Winfield. ‘Function and Functional Minimization by Interpolation in Data Tables’. PhD thesis. Cambridge MA USA: Harvard University, 1969.

- [55] D. H. Winfield. ‘Function minimization by interpolation in a data table’. In: *IMA Journal of Applied Mathematics* 12.3 (1973), pp. 339–347.
- [56] H. Zhang, A. R. Conn and K. Scheinberg. ‘A Derivative-Free Algorithm for Least-Squares Minimization’. In: *SIAM Journal on Optimization* 20.6 (2010), pp. 3555–3576.