MODEL-BASED DERIVATIVE-FREE METHODS FOR CONVEX-CONSTRAINED OPTIMIZATION

MATTHEW HOUGH[*] AND LINDON ROBERTS[†]

Abstract. We present a model-based derivative-free method for optimization subject to general convex constraints, which we assume are unrelaxable and accessed only through a projection operator that is cheap to evaluate. We prove global convergence and a worst-case complexity of $\mathcal{O}(\epsilon^{-2})$ iterations and objective evaluations for nonconvex functions, matching results for the unconstrained case. We introduce new, weaker requirements on model accuracy compared to existing methods. As a result, sufficiently accurate interpolation models can be constructed only using feasible points. We develop a comprehensive theory of interpolation set management in this regime for linear and composite linear models. We implement our approach for nonlinear least-squares problems and demonstrate strong practical performance compared to general-purpose solvers.

Key words. derivative-free optimization, trust-region methods, convex constraints, worst-case complexity

AMS subject classifications. 65K05, 90C30, 90C56

1. Introduction.

The minimizing a function when derivative information is not available, derivative-free optimization (DFO), is an area of growing research attention with many applications [18, 3, 26]. The two most common approaches to DFO for finding local minima are direct search, where the objective is sampled in several directions to find descent, and model-based, which uses sampled function values to construct a local model for the objective within (typically) a trust-region framework. The survey [26] provides a detailed overview of these approaches.

Because of their relative simplicity, direct search methods have been extended to a wide variety of problem classes, including general nonsmooth [2] or discontinuous [35] objectives, and those with discrete variables [4]. By contrast, the analysis of model-based DFO beyond general unconstrained optimization is less developed, largely due to the difficulty of forming appropriate notions of model accuracy. Most commonly the local models are formed by interpolation, and model accuracy is guaranteed by requiring the interpolation set to satisfy particular geometric conditions.

In this work we consider model-based DFO for nonlinear optimization in the presence of convex constraints. That is, we aim to solve

$$\min_{\boldsymbol{x}\in C} f(\boldsymbol{x}), \tag{1.1}$$

where $C \subseteq \mathbb{R}^n$ is closed and convex with nonempty interior. We assume that the objective $f : \mathbb{R}^n \to \mathbb{R}$ is smooth and nonconvex, although access to its derivatives is not possible. In our approach, the constraint set is made available to the algorithm only through a projection operator which we assume is cheap to compute. To do this, we follow the approach from derivative-based trust-region methods [15, Chapter 12] in which the trust-region subproblem is solved using projected gradient descent. We provide a first-order convergence analysis and worst-case complexity bounds for our approach, motivated by the analysis for derivative-based cubic regularization methods from [10].

[*]Department of Combinatorics and Optimization, University of Waterloo, 200 University Avenue W., Waterloo, ON N2L 3G1, Canada (mhough@uwaterloo.ca)

[†]Mathematical Sciences Institute, Building 145, Science Road, Australian National University, Canberra ACT 2601, Australia (lindon.roberts@anu.edu.au).

Importantly, our algorithm is strictly feasible, and so f is only ever evaluated at points in C. This requires the development of new notions of model accuracy suitable when the interpolation set is restricted to lie in C. To that end, we introduce a novel, comprehensive theory in which we generalize the notions of fully linear models and Λ-poised interpolation sets [16, 17] to arbitrary convex sets. This generalization allows us to overcome a known limitation of model-based DFO, namely that for some problems "*it may be impossible to obtain a fully linear model using only feasible points*" [26, p. 362]. Our theory is suitable for models based on linear interpolation, and so are particularly useful for composite minimization. To that end, we provide numerical experiments on constrained nonlinear least-squares problems, where we show strong practical performance.

1.1. Existing Works. Trust-region methods for solving (1.1) in a derivative-based setting are well-studied, with [15, Chapter 12] providing a comprehensive development of measures of stationarity, the projected gradient method for solving the trust-region problem, and global convergence analysis. A worst-case complexity analysis of this technique, where global convergence is guaranteed by adaptive cubic regularization was given in [10]. A related earlier work is [14], which considered the derivative-based trust-region method in the case of inexact gradients $\boldsymbol{g}_k$ satisfying $\|\boldsymbol{g}_k - \nabla f(\boldsymbol{x}_k)\| \leq \kappa_{\mathrm{eg}}\Delta_k$ at all iterations k (c.f. (4.7)). More recently, [6] considers high-order adaptive regularization methods for general constrained optimization problems subject to inexact objective and derivative information.

For model-based DFO, the works [16, 17] introduced the relevant concepts of model accuracy (full linearity) and interpolation set geometry (Λ-poisedness) used to analyze unconstrained problems, and worse-case complexity analysis was developed (including for composite minimization) in [22, 21]. A variety of model-based DFO approaches have been proposed for constrained optimization problems. These vary by whether the constraints are known exactly or are black-box, and whether they are relaxable or not. A comprehensive survey of these methods, categorized using the taxonomy of [28], can be found in [26, Section 7]. However, we particularly note model-based DFO methods for problems with unrelaxable constraints, including [34, 24] and [36, Section 6.3] for bound constraints, and [25] for linear inequality constraints. All of these works focus on the development of efficient algorithms and implementations, and do not have convergence analysis. We extend these approaches to a significantly broader class of constraints by assuming a convex constraint set accessed only via projections, while providing theoretical guarantees of convergence.

The most similar existing work to ours is [13], which introduces a model-based DFO method for convex constrained optimization. Similar to our setting, access to the constraints is assumed to be through a projection operator and global convergence is analyzed through the framework of [15, Chapter 12] provided the local model is always fully linear in the sense of [17]. Our work extends this by generalizing full linearity to general convex constraint sets, providing interpolation set management routines for ensuring these conditions, and having an algorithmic framework where models are (occasionally) allowed to not be fully linear. We also provide a worst-case complexity analysis of our method (as well as global convergence) and numerical results for nonlinear least-squares problems. We also note that the same first-order criticality measure was also used to study probabilistic direct search methods subject to linear constraints [23].

We will specialize our approach to composite minimization, such as nonlinear least-squares problems. In the derivative-based setting, [32] analyzes a Gauss-Newton method for nonlinear least-squares problems with bound constraints and inexact trust-region subproblem solutions. In the DFO context, [37, 11] analyze unconstrained model-based DFO methods for nonlinear least-squares problems, and [31] studies a derivative-free linesearch method for nonlinear systems with convex constraints.

1.2. Contributions. The contributions of this paper are threefold.

First, we introduce a model-based DFO algorithm for (1.1), naturally extending the standard framework of unconstrained model-based DFO [18] to convex constraints, based on the approach in [15]. Our method only accesses constraint information through a projection operator, which is assumed to be cheap to evaluate. Compared to [13], our approach uses the same broad trust-region framework but with a different criticality measure (2.1). We also introduce a new notion of fully linear models suitable for the constrained setting, while also permitting iterations where the local model is not accurate. Our new fully linear definition (Definition 2.3) is similar to that of [17], but is specifically adapted to the setting of convex constraints. As in [13], we prove global convergence to first-order critical points, but we augment this by showing a worst-case complexity bound of $\mathcal{O}(\epsilon^{-2})$ iterations to drive the criticality measure below ϵ. In the unconstrained case, our bound recovers the result of [21], including the same dependency on problem dimension in the case of linear interpolation models. Our approach allows a wider range of constraint sets than [34, 36, 24, 25], which also do not provide convergence analysis.

Second, in order to construct interpolation models satisfying our new notion of full linearity, we extend the theory of Λ-poisedness from [16] to the constrained setting. Our weaker notion of full linearity means that we can construct Λ-poised interpolation sets by only sampling from feasible points (overcoming the aforementioned limitation described in [26, p. 362]). We show that linear interpolation based on Λ-poised sets produce fully linear models (in our new sense of both terms). We also give concrete procedures to generate Λ-poised sets and validate whether or not a given set is Λ-poised. Although linear interpolation models are sufficient for producing fully linear models, they have gained particular popularity in composite minimization (e.g. [22, 21, 11]), and so we show that Λ-poised sets with linear interpolation produces fully linear models for composite minimization.

Lastly, and motivated by the case of composite minimization, we use our approach to extend the state-of-the-art solver DFO-LS [9] for nonlinear least-squares problems to handle general convex constraints via projection operators.[1] Testing on low-dimensional test problems with a variety of linear and Euclidean ball inequality constraints, we show that the new DFO-LS can outperform the general-purpose solvers COBYLA [33], which requires the functional form for the constraints, and NOMAD [27], which handles constraints via an extreme barrier. These solvers differ based on how they access the constraints, and are not designed to exploit the least-squares structure of the objective.

Code Availability. The latest version of DFO-LS with these improvements is available on Github.[2]

Notation. Throughout, we let proj_C denote the projection operator for the convex set $C \subseteq \mathbb{R}^n$ in the Euclidean norm.[3] We use $\|\cdot\|$ to be the 2-norm for vectors and matrices, and define $B(\boldsymbol{x}, \Delta) := \{\boldsymbol{y} \in \mathbb{R}^n : \|\boldsymbol{y}-\boldsymbol{x}\| \le \Delta\}$ to be the closed ball centered

[1]We note that DFO-LS could previously already handle box constraints using techniques from [34], but did not have any convergence guarantees.

[2]Version 1.3, https://github.com/numericalalgorithmsgroup/dfols.

[3]This is a well-defined operator from $\mathbb{R}^n$ to C, as per [5, Theorem 6.25], for instance.

at $\boldsymbol{x} \in \mathbb{R}^n$ of radius $\Delta \geq 0$.

2. Algorithm Statement.

In this section, we outline the general model-based DFO algorithm for (1.1). Our framework closely mimics the unconstrained setting given in [18, Chapter 10], with modifications for the constraints similar to the ones in the derivative-based trust-region setting [15, Chapter 12]. We first outline our notion of first-order criticality for (1.1), and then give our algorithm.

Throughout this work, we assume the following about our feasible set C.

Assumption 2.1. *The feasible set $C \subseteq \mathbb{R}^n$ is a closed, convex set with nonempty interior,* $\operatorname{int} C \neq \emptyset$.

2.1. Criticality Measure.

We will seek a first-order optimal solution to (1.1). To measure this, we use the following first-order criticality measure from [15, Section 12.1.4], defined for all $\boldsymbol{x} \in C$:

$$\pi^f(\boldsymbol{x}) := \left| \min_{\substack{\boldsymbol{x}+\boldsymbol{d}\in C \\ \|\boldsymbol{d}\|\leq 1}} \nabla f(\boldsymbol{x})^T \boldsymbol{d} \right|. \tag{2.1}$$

We will say that $\boldsymbol{x}^*$ is a first-order critical point for (1.1) if $\pi^f(\boldsymbol{x}^*) = 0$ (see [15, Theorem 12.1.6]). If $C = \mathbb{R}^n$, then we recover the first-order criticality measure $\pi^f(\boldsymbol{x}) = \|\nabla f(\boldsymbol{x})\|$. As shown in [15, Theorem 12.1.4], the minimizer of (2.1) is of the form $\boldsymbol{d}^* = \boldsymbol{x}(t) - \boldsymbol{x}$ for some $t \geq 0$ sufficiently large, where $\boldsymbol{x}(t)$ is the projected gradient path, $\boldsymbol{x}(t) := \operatorname{proj}_C[\boldsymbol{x} - t\nabla f(\boldsymbol{x})]$.

However, in a DFO setting, we cannot measure $\pi^f(\boldsymbol{x})$ as it depends on $\nabla f(\boldsymbol{x})$. Instead, we will only have access to some approximate gradient $\boldsymbol{g} \approx \nabla f(\boldsymbol{x})$. As a consequence, we will need the following result, which quantifies the sensitivity of the criticality measure (2.1) to errors in the gradient $\nabla f(\boldsymbol{x})$. This is a modification of [10, Theorem 3.4], which proves that $\pi^f(\boldsymbol{x})$ is Lipschitz continuous provided $\nabla f(\boldsymbol{x})$ is also.

Lemma 2.2. *Fix $\boldsymbol{x} \in C$ and $\boldsymbol{g}_1, \boldsymbol{g}_2 \in \mathbb{R}^n$, and define*

$$\pi_1(\boldsymbol{x}) := \left| \min_{\substack{\boldsymbol{x}+\boldsymbol{d}\in C \\ \|\boldsymbol{d}\|\leq 1}} \boldsymbol{g}_1^T \boldsymbol{d} \right|, \qquad \text{and} \qquad \pi_2(\boldsymbol{x}) := \left| \min_{\substack{\boldsymbol{x}+\boldsymbol{d}\in C \\ \|\boldsymbol{d}\|\leq 1}} \boldsymbol{g}_2^T \boldsymbol{d} \right|. \tag{2.2}$$

Then we have

$$|\pi_1(\boldsymbol{x}) - \pi_2(\boldsymbol{x})| \leq \max_{\substack{\boldsymbol{x}+\boldsymbol{d}\in C \\ \|\boldsymbol{d}\|\leq 1}} \left|(\boldsymbol{g}_1 - \boldsymbol{g}_2)^T \boldsymbol{d}\right|. \tag{2.3}$$

Proof. Since $\boldsymbol{d} = \boldsymbol{0}$ is feasible for both $\pi_1(\boldsymbol{x})$ and $\pi_2(\boldsymbol{x})$, the quantity inside the absolute values is always non-positive. Hence, defining $\boldsymbol{d}_i$ to be the minimizer corresponding to $\pi_i(\boldsymbol{x})$, we have $\pi_i(\boldsymbol{x}) = -\boldsymbol{g}_i^T \boldsymbol{d}_i$.

First, suppose that $\pi_1(\boldsymbol{x}) \geq \pi_2(\boldsymbol{x})$. Then,

$$|\pi_1(\boldsymbol{x}) - \pi_2(\boldsymbol{x})| = \pi_1(\boldsymbol{x}) - \pi_2(\boldsymbol{x}) = -\boldsymbol{g}_1^T \boldsymbol{d}_1 + \boldsymbol{g}_2^T \boldsymbol{d}_2, \tag{2.4}$$

$$= -(\boldsymbol{g}_1 - \boldsymbol{g}_2)^T \boldsymbol{d}_1 + (\boldsymbol{g}_2^T \boldsymbol{d}_2 - \boldsymbol{g}_2^T \boldsymbol{d}_1), \tag{2.5}$$

$$\leq \left|(\boldsymbol{g}_1 - \boldsymbol{g}_2)^T \boldsymbol{d}_1\right|, \tag{2.6}$$

where to get the last inequality we use $\boldsymbol{g}_2^T \boldsymbol{d}_2 \leq \boldsymbol{g}_2^T \boldsymbol{d}_1$ (since $\boldsymbol{d}_2$ is the minimizer for $\pi_2(\boldsymbol{x})$ and $\boldsymbol{d}_1$ is a feasible point for the same problem).

Instead, if $\pi_1(\boldsymbol{x}) < \pi_2(\boldsymbol{x})$, then by similar reasoning we have

$$|\pi_1(\boldsymbol{x}) - \pi_2(\boldsymbol{x})| \leq \left|(\boldsymbol{g}_2 - \boldsymbol{g}_1)^T \boldsymbol{d}_2\right|. \tag{2.7}$$

Combining (2.6) and (2.7) proves (2.3). ◻

We also note that applying Cauchy-Schwarz and $\|\boldsymbol{d}\| \leq 1$ to (2.3) gives us the Lipschitz-type property

$$|\pi_1(\boldsymbol{x}) - \pi_2(\boldsymbol{x})| \leq \|\boldsymbol{g}_1 - \boldsymbol{g}_2\|. \tag{2.8}$$

2.2. Main Algorithm. Our main algorithm follows a trust-region framework. At each iteration k, we construct a local quadratic model which approximates the objective near the current iterate $\boldsymbol{x}_k$:

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) := c_k + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{y} - \boldsymbol{x}_k)^T H_k (\boldsymbol{y} - \boldsymbol{x}_k), \tag{2.9}$$

where we hope this approximation is accurate when $\boldsymbol{y} \approx \boldsymbol{x}_k$. Following [15, Chapter 12], we calculate a tentative new iterate $\boldsymbol{x}_k + \boldsymbol{s}_k$, where the step $\boldsymbol{s}_k$ is an approximate minimizer of the trust-region subproblem

$$\min_{\substack{\boldsymbol{x}+\boldsymbol{s}\in C \\ \|\boldsymbol{s}\|\leq\Delta_k}} m_k(\boldsymbol{x}_k + \boldsymbol{s}), \tag{2.10}$$

where the trust-region radius $\Delta_k > 0$ is updated at each iteration. We accept this step (i.e. set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$) and increase Δ_k if this point sufficiently decreases f, otherwise we set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and (possibly) decrease Δ_k.

In the DFO setting, there is the possibility that the model (2.9) is not a good approximation for the objective. The usual notion of m_k being a 'good approximation' is the notion of a 'fully linear' model (e.g. [18, Definition 6.1]). For the constrained case, we introduce a similar notion of full linearity, in part motivated by (2.3).

Definition 2.3. *The model m_k (2.9) is* fully linear *in $B(\boldsymbol{x}_k, \Delta_k)$ if there exist $\kappa_{\text{ef}}, \kappa_{\text{eg}} > 0$, independent of k, such that*

$$\max_{\substack{\boldsymbol{x}+\boldsymbol{d}\in C \\ \|\boldsymbol{d}\|\leq\Delta_k}} |f(\boldsymbol{x}_k + \boldsymbol{d}) - m_k(\boldsymbol{x}_k + \boldsymbol{d})| \leq \kappa_{\text{ef}} \Delta_k^2, \tag{2.11a}$$

$$\max_{\substack{\boldsymbol{x}+\boldsymbol{d}\in C \\ \|\boldsymbol{d}\|\leq 1}} \left|(\nabla f(\boldsymbol{x}_k) - \boldsymbol{g}_k)^T \boldsymbol{d}\right| \leq \kappa_{\text{eg}} \Delta_k. \tag{2.11b}$$

In particular, we note that we require $\|\boldsymbol{d}\| \leq \Delta_k$ in (2.11a) but $\|\boldsymbol{d}\| \leq 1$ in (2.11b). In the unconstrained case $C = \mathbb{R}^n$, Definition 2.3 is slightly weaker than the usual version of full linearity because it requires that $\nabla m(\boldsymbol{y}) \approx \nabla f(\boldsymbol{y})$ only at $\boldsymbol{y} = \boldsymbol{x}_k$, rather than all $\boldsymbol{y} \in B(\boldsymbol{x}_k, \Delta_k)$. We assume the existence of procedures which can verify whether m_k is fully linear, and (if not) to make m_k fully linear. We describe such procedures in Section 4.

The local model m_k (2.9) induces an approximate first-order criticality measure, namely

$$\pi^m(\boldsymbol{x}) := \left| \min_{\substack{\boldsymbol{x}+\boldsymbol{d}\in C \\ \|\boldsymbol{d}\|\leq 1}} \boldsymbol{g}_k^T \boldsymbol{d} \right|. \tag{2.12}$$

Our full linearity definition allows us to compare our approximate criticality measure $\pi_k^m := \pi^m(\boldsymbol{x}_k)$ with the true value $\pi_k^f := \pi^f(\boldsymbol{x}_k)$.

Algorithm 2.1 CDFO-TR: model-based DFO method for (1.1).

```
Input: Starting point $\boldsymbol{x}_0 \in \mathbb{R}^n$ and trust-region radius $\Delta_0 > 0$.
    Parameters: maximum trust-region radius $\Delta_{\max} \geq \Delta_0$, scaling factors $0 < \gamma_{\text{dec}} < 1 < \gamma_{\text{inc}}$, criticality constants $\epsilon_C, \mu > 0$, and acceptance threshold $\eta \in (0,1)$.
1: Build an initial model $m_0$ (2.9).
2: for $k = 0, 1, 2, \ldots$ do
3:     if $\pi_k^m < \epsilon_C$ and ($\pi_k^m < \mu^{-1}\Delta_k$ or $m_k$ is not fully linear in $B(\boldsymbol{x}_k, \Delta_k)$) then
4:         Criticality step: Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$. If $m_k$ is fully linear in $B(\boldsymbol{x}_k, \Delta_k)$, set $\Delta_{k+1} = \gamma_{\text{dec}}\Delta_k$, otherwise set $\Delta_{k+1} = \Delta_k$. Construct $m_{k+1}$ to be fully linear in $B(\boldsymbol{x}_{k+1}, \Delta_{k+1})$.
5:     else  $\leftarrow \pi_k^m \geq \epsilon_C$ or ($\pi_k^m \geq \mu^{-1}\Delta_k$ and $m_k$ is fully linear in $B(\boldsymbol{x}_k, \Delta_k)$)
6:         Approximately solve (2.10) to get a step $\boldsymbol{s}_k$.
7:         Evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and calculate ratio

           (2.14)   $$\rho_k := \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{s}_k)}{m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)}.$$

8:         if $\rho_k \geq \eta$ then
9:             Successful step: Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = \min(\gamma_{\text{inc}}\Delta_k, \Delta_{\max})$. Form $m_{k+1}$ in any manner.
10:        else if $m_k$ is not fully linear in $B(\boldsymbol{x}_k, \Delta_k)$ then
11:            Model-improving step: Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and $\Delta_{k+1} = \Delta_k$, and construct $m_{k+1}$ to be fully linear in $B(\boldsymbol{x}_{k+1}, \Delta_{k+1})$.
12:        else  $\leftarrow \rho_k < \eta$ and $m_k$ is fully linear in $B(\boldsymbol{x}_k, \Delta_k)$
13:            Unsuccessful step: Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and $\Delta_{k+1} = \gamma_{\text{dec}}\Delta_k$. Form $m_{k+1}$ in any manner.
14:        end if
15:    end if
16: end for
```

LEMMA 2.4. *If m_k is fully linear in $B(\boldsymbol{x}_k, \Delta_k)$, then the approximate criticality measure π_k^m (2.12) satisfies*

$$\left|\pi_k^f - \pi_k^m\right| \leq \kappa_{\text{eg}}\Delta_k. \tag{2.13}$$

Proof. This immediately follows from (2.11b) and Lemma 2.2. ∎

In the unconstrained case, (2.13) is exactly the fully linear condition (2.11b), and so there is nothing to show.

Our main algorithm, CDFO-TR, for solving (1.1) is given in Algorithm 2.1. It is essentially the same as [18, Algorithm 10.1], but we modify the criticality step to avoid having an inner loop, similar to [21, Algorithm 3.1] and [12, Algorithm 1], and replace references to the criticality measure $\|\boldsymbol{g}_k\|$ with π_k^m.

3. Convergence & Worst-Case Complexity.

We now prove convergence and provide a worst-case complexity analysis for Algorithm 2.1. We will require the following (standard) assumptions.

ASSUMPTION 3.1. *The objective function f is bounded below by f_{low} continuously differentiable. Furthermore, the gradient ∇f is Lipschitz continuous with constant $L_{\nabla f}$ in $\cup_k B(\boldsymbol{x}_k, \Delta_{\max})$.*

ASSUMPTION 3.2. *There exists $\kappa_H \geq 1$ such that $\|H_k\| \leq \kappa_H - 1$ for all k.*

Lastly, we need an assumption about the accuracy with which the trust-region subproblem (2.10) is solved.

ASSUMPTION 3.3. *There exists a constant $c_1 \in (0,1)$ such that the computed step $\boldsymbol{s}_k$ satisfies $\boldsymbol{x}_k + \boldsymbol{s}_k \in C$, $\|\boldsymbol{s}_k\| \leq \Delta_k$ and the generalized Cauchy decrease condition:*

$$m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k) \geq c_1 \pi_k^m \min\left(\frac{\pi_k^m}{1 + \|H_k\|}, \Delta_k, 1\right). \tag{3.1}$$

Assumption 3.3 is a slight modification to the unconstrained Cauchy decrease assumption [15, Assumption AA.1]. We note that Assumption 3.3 is achievable using a Goldstein-type linesearch method [15, Algorithm 12.2.2 & Theorem 12.2.2].

3.1. Convergence of Algorithm 2.1.

LEMMA 3.4. *Suppose Assumption 3.1 holds and the criticality step is not called in iteration k. Then $\pi_k^m \geq \min(\epsilon_C, \Delta_k/\mu)$. Also, if $\pi_k^f \geq \epsilon > 0$ then*

$$\pi_k^m \geq \epsilon_{\text{mc}} := \min\left(\epsilon_C, \frac{\epsilon}{1 + \kappa_{\text{eg}}\mu}\right) > 0. \tag{3.2}$$

Proof. This proof is based on [12, Lemma 2.10]. We first note that $\pi_k^m \geq \min(\epsilon_C, \Delta_k/\mu)$ is simply a necessary condition for not entering the criticality step.

For the second part, suppose that the criticality step is not called in iteration k, and $\pi_k^f \geq \epsilon$ and $\pi_k^m < \epsilon_C$. Then $\Delta_k \leq \mu\pi_k^m$ and m_k is fully linear in $B(\boldsymbol{x}_k, \Delta_k)$, and so Lemma 2.4 gives

$$\epsilon \leq \pi_k^f \leq \left|\pi_k^f - \pi_k^m\right| + \pi_k^m \leq \kappa_{\text{eg}}\Delta_k + \pi_k^m \leq (\kappa_{\text{eg}}\mu + 1)\pi_k^m, \tag{3.3}$$ ∎

and so $\pi_k^m \geq \epsilon/(1 + \kappa_{\text{eg}}\mu)$.

LEMMA 3.5. *Suppose Assumptions 3.1, 3.2 and 3.3 hold, m_k is fully linear on $B(\boldsymbol{x}_k, \Delta_k)$ and that*

$$\Delta_k \leq \min(c_0\pi_k^m, 1), \qquad \text{where} \qquad c_0 := \min\left(\mu, \frac{1}{\kappa_H}, \frac{c_1(1-\eta)}{2\kappa_{\text{ef}}}\right). \tag{3.4}$$

Then the criticality step is not called in iteration k, and iteration k is successful (i.e. $\rho_k \geq \eta$).

Proof. This proof is similar to [12, Lemma 2.6]. Since m_k is fully linear and $\Delta_k \leq c_0\pi_k^m \leq \mu\pi_k^m$, the criticality step is not called.

Then, Assumptions 3.2 and 3.3 with $\Delta_k \leq \min(c_0\pi_k^m, 1) \leq \min(\kappa_H^{-1}\pi_k^m, 1)$ imply

$$m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k) \geq c_1\pi_k^m \min\left(\frac{\pi_k^m}{\kappa_H}, \Delta_k, 1\right) = c_1\pi_k^m\Delta_k. \tag{3.5}$$

Now, since m_k is fully linear, we apply (2.11a) to get

$$|\rho_k - 1| \leq \left|\frac{f(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k)}{m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)}\right| + \left|\frac{f(\boldsymbol{x}_k + \boldsymbol{s}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)}{m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)}\right|, \tag{3.6}$$

$$\leq \frac{2\kappa_{\text{ef}}\Delta_k^2}{c_1\pi_k^m\Delta_k}, \tag{3.7}$$

$$\leq 1 - \eta, \tag{3.8}$$ ∎

where the last inequality follows from $\Delta_k \le c_0\pi_k^m \le c_1(1-\eta)\pi_k^m/(2\kappa_{\text{ef}})$. That $\rho_k \ge \eta$ then follows from (3.8).

LEMMA 3.6. *Suppose that Assumptions 3.1, 3.2 and 3.3 hold, and $\pi_k^f \ge \epsilon > 0$ for all $k \le K$. Then $\Delta_k \ge \Delta_{\min}$ for all $k \le K$, where*

$$\Delta_{\min} := \min\left(\Delta_0, \frac{\gamma_{\text{dec}}\epsilon}{\kappa_{\text{eg}}+\mu^{-1}}, \gamma_{\text{dec}}\mu\epsilon_{\text{mc}}, \frac{\gamma_{\text{dec}}\epsilon_{\text{mc}}}{\kappa_H}, \gamma_{\text{dec}}\left(\kappa_{\text{eg}}+\frac{2\kappa_{\text{ef}}}{c_1(1-\eta)}\right)^{-1}\epsilon, \gamma_{\text{dec}}\right). \tag{3.9}$$

Proof. This argument is similar to [11, Lemma 3.8]. To find a contradiction, suppose that $k' \le K$ is the first iteration with $\Delta_{k'} < \Delta_{\min}$. Since $\Delta_{\min} \le \Delta_0$, we must have $k' > 0$. Given that k' is the first such iteration, we have $\Delta_{k'} < \Delta_{\min} \le \Delta_{k'-1}$, and so iteration $(k'-1)$ must have either been a criticality step (with $m_{k'-1}$ fully linear) or an unsuccessful step; these are the only situations in which $\Delta_{k+1} < \Delta_k$.

First, suppose that iteration $(k'-1)$ had $m_{k'-1}$ fully linear in $B(\boldsymbol{x}_{k'-1}, \Delta_{k'-1})$ and the criticality step was called. By the entry condition for the criticality step, this means that $\pi_{k'-1}^m < \mu^{-1}\Delta_{k'-1}$. Combining these properties, we have

$$\epsilon \le \pi_{k'-1}^f \le \left|\pi_{k'-1}^f - \pi_{k'-1}^m\right| + \pi_{k'-1}^m \le \kappa_{\text{eg}}\Delta_{k'-1} + \mu^{-1}\Delta_{k'-1}, \tag{3.10}$$

and so $\Delta_{k'-1} \ge \epsilon/(\kappa_{\text{eg}}+\mu^{-1})$. Thus

$$\Delta_{k'} = \gamma_{\text{dec}}\Delta_{k'-1} \ge \frac{\gamma_{\text{dec}}\epsilon}{\kappa_{\text{eg}}+\mu^{-1}} \ge \Delta_{\min}, \tag{3.11}$$

which is a contradiction. Thus iteration $(k'-1)$ must have been an unsuccessful step.

For iteration $(k'-1)$ to have been an unsuccessful step, we must have had $\rho_{k'-1} < \eta$ and $m_{k'-1}$ fully linear in $B(\boldsymbol{x}_{k'-1}, \Delta_{k'-1})$. Also, since the criticality step is not called, we must have $\pi_{k'-1}^m \ge \epsilon_{\text{mc}}$ by Lemma 3.4.

Now, suppose that

$$\Delta_{k'-1} > \frac{c_1(1-\eta)\pi_{k'-1}^m}{2\kappa_{\text{ef}}}. \tag{3.12}$$

Then by full linearity we have

$$\epsilon \le \kappa_{\text{eg}}\Delta_{k'-1} + \pi_{k'-1}^m < \left(\kappa_{\text{eg}} + \frac{2\kappa_{\text{ef}}}{c_1(1-\eta)}\right)\Delta_{k'-1}. \tag{3.13}$$

However, since iteration $(k'-1)$ was unsuccessful, we have

$$\Delta_{k'-1} = \gamma_{\text{dec}}^{-1}\Delta_{k'} < \gamma_{\text{dec}}^{-1}\Delta_{\min} \le \left(\kappa_{\text{eg}} + \frac{2\kappa_{\text{ef}}}{c_1(1-\eta)}\right)^{-1}\epsilon, \tag{3.14}$$

contradicting (3.13), and hence (3.12) must be false.

Separately, we also have

$$\Delta_{k'-1} < \gamma_{\text{dec}}^{-1}\Delta_{\min} \le \min\left(\mu\epsilon_{\text{mc}}, \frac{\epsilon_{\text{mc}}}{\kappa_H}, 1\right), \tag{3.15}$$

and so altogether we have $\Delta_{k'-1} \le \min(c_0\pi_{k'-1}^m, 1)$. Hence by Lemma 3.5 iteration $(k'-1)$ must have $\rho_{k'-1} \ge \eta$, a contradiction. ∎

We now proceed to show convergence of Algorithm 2.1. The following results follow broadly the approach of [18, Chapter 10] and [11].

LEMMA 3.7. *Suppose Assumption 3.1 holds and there are finitely many successful iterations. Then*

$$\lim_{k\to\infty} \pi_k^f = 0. \tag{3.16}$$

Proof. Let k_0 be the final successful iteration. Then all iterations $k > k_0$ must either be criticality, model-improving or unsuccessful steps. If a model in some iteration $k > k_0$ is not fully linear, then Algorithm 2.1 guarantees that the model in iteration $k+1$ is fully linear. Hence there are infinitely many iterations $k > k_0$ for which the model is fully linear. Such iterations (criticality or unsuccessful) always cause Δ_k to be reduced by a factor of γ_{dec}, and so $\lim_{k\to\infty}\Delta_k = 0$.

Now suppose, for each $k > k_0$, that i_k is the index of the first iteration after k_0 for which m_k is fully linear. Then by the reasoning above we must have $i_k \in \{k, k+1\}$. It follows that as $k\to\infty$,

$$\|\boldsymbol{x}_k - \boldsymbol{x}_{i_k}\| \le \Delta_k \to 0. \tag{3.17}$$

Note that we may write

$$\pi_k^f \le \left|\pi_k^f - \pi_{i_k}^f\right| + \left|\pi_{i_k}^f - \pi_{i_k}^m\right| + \pi_{i_k}^m. \tag{3.18}$$

Since ∇f is Lipschitz continuous, applying [10, Theorem 3.4] gives that $\left|\pi_k^f - \pi_{i_k}^f\right| \le M\|\boldsymbol{x}_k - \boldsymbol{x}_{i_k}\|$ for some $M > 0$. So from (3.17), the first term converges to zero. The second term converges to zero by using Lemma 2.4 to write $\left|\pi_{i_k}^f - \pi_{i_k}^m\right| \le \kappa_{\text{eg}}\Delta_{i_k} \to 0$. Finally, the third term converges to zero, since if that were not the case, Lemma 3.5 would contradict the fact that $k > k_0$ are not successful iterations. It must then hold that $\pi_k^f \to 0$. ∎

LEMMA 3.8. *Suppose that Assumptions 3.1, 3.2 and 3.3 hold. Then*

$$\lim_{k\to\infty}\Delta_k = 0. \tag{3.19}$$

Proof. Let $\mathcal{S}$ be the set of successful iterations. Then the result when $\mathcal{S}$ is finite is shown in the proof of Lemma 3.7. We will now consider the case when $\mathcal{S}$ is infinite.

For any $k \in \mathcal{S}$ we have from Assumptions 3.2 and 3.3,

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \ge \eta\left[m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)\right] \ge \eta c_1 \pi_k^m \min\left(\frac{\pi_k^m}{\kappa_H}, \Delta_k, 1\right). \tag{3.20}$$

Since the criticality step is not called for $k\in\mathcal{S}$, from Lemma 3.4 we must have that $\pi_k^m \ge \min(\epsilon_C, \Delta_k/\mu)$; hence

$$\sum_{k\in\mathcal{S}} f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \ge \sum_{k\in\mathcal{S}} \eta c_1 \min(\epsilon_C, \Delta_k/\mu)\min\left(\frac{\min(\epsilon_C, \Delta_k/\mu)}{\kappa_H}, \Delta_k, 1\right) \ge 0. \tag{3.21}$$

By Assumption 3.1, the left-hand side of this inequality is bounded above by $f(\boldsymbol{x}_0) - f_{\text{low}}$. Hence the right-hand side must be summable. Since $\mathcal{S}$ is infinite, this then implies that $\lim_{k\in\mathcal{S}}\Delta_k \to 0$.

Now recall that the trust-region radius can be increased only during a successful iteration, and only by a ratio of at most γ_{inc}. Let $k \notin S$ be the index of any non-successful iteration after at least one successful iteration has occurred, and let $s_k \in \mathcal{S}$ be the last successful iteration before k. Then $\Delta_k \leq \gamma_{\text{inc}}\Delta_{s_k}$. Since from above we have that $\Delta_{s_k} \to 0$, then $\Delta_k \to 0$ for $k \notin S$ as $k \to \infty$, which proves the result in general. ∎

LEMMA 3.9. *Suppose that Assumptions 3.1, 3.2 and 3.3 hold. Then*

$$\liminf_{k\to\infty} \pi_k^f = 0. \tag{3.22}$$

Proof. Let $\mathcal{S}$ be the set of successful iterations. Then the result when $\mathcal{S}$ is finite follows from Lemma 3.7, so suppose $\mathcal{S}$ is infinite.

Assume for contradiction that for all k sufficiently large (e.g. $k \geq k_0$),

$$\pi_k^f \geq \epsilon, \tag{3.23}$$

for some $\epsilon > 0$. From Lemma 3.4, $\pi_k^m \geq \epsilon_{\text{mc}} > 0$ must also hold. Now, consider some $k \in \mathcal{S}$. By Assumptions 3.2 and 3.3, Lemma 3.6 and conditions (3.23),

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \geq \eta[m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)] \geq \eta c_1 \epsilon_1 \min\left(\frac{\epsilon_{\text{mc}}}{\kappa_H}, \Delta_{\min}, 1\right). \tag{3.24}$$

We can take the sum over all successful iterations from k_0 up to k:

$$f(\boldsymbol{x}_{k_0}) - f(\boldsymbol{x}_{k+1}) = \sum_{\substack{j=k_0\\ j\in\mathcal{S}}}^{k} [f(\boldsymbol{x}_j) - f(\boldsymbol{x}_{j+1})] \geq \sigma_k \eta c_1 \epsilon_{\text{mc}} \left(\frac{\epsilon_{\text{mc}}}{\kappa_H}, \Delta_{\min}, 1\right), \tag{3.25}$$

where σ_k is the number of successful iterations between iterations k_0 and k. However, since there are infinitely many successful iterations, we have that $\lim_{k\to\infty} \sigma_k = \infty$, and thus the difference between $f(\boldsymbol{x}_{k_0})$ and $f(\boldsymbol{x}_{k+1})$ is unbounded; so f is unbounded below. This contradicts Assumption 3.1, so we are done. ∎

THEOREM 3.10. *Suppose that Assumptions 3.1, 3.2 and 3.3 hold. Then*

$$\lim_{k\to\infty} \pi_k^f = 0. \tag{3.26}$$

Moreover, $\lim_{k\to\infty} \pi_k^m = 0$*, cannot occur before* (3.26) *holds.*

Proof. Assume for contradiction that there exists a subsequence of successful iterates indexed by $\{t_i\} \subseteq \mathcal{S}$ such that for some $\epsilon > 0$

$$\pi_{t_i}^f \geq 2\epsilon > 0, \tag{3.27}$$

for all i. For each t_i, we know from Lemma 3.9 that there exists a first successful iteration $\ell_i > t_i$ such that $\pi_{\ell(t_i)}^f < \epsilon$. Hence we have subsequences $\{t_i\}$ and $\{\ell_i\}$ of $\mathcal{S}$ such that such that

$$\pi_{t_i}^f \geq 2\epsilon, \qquad \pi_k^f \geq \epsilon \quad \text{for } t_i < k < \ell_i, \qquad \text{and} \qquad \pi_{\ell_i}^f < \epsilon. \tag{3.28}$$

From Lemma 3.4, we must also have that $\pi_k^m \geq \epsilon_{\text{mc}} > 0$ for $t_i \leq k < \ell_i$. Let us restrict our attention to the subsequence of successful iterations whose indices are in the set

$$\mathcal{K} := \{k \in \mathcal{S} : t_i \leq k < \ell_i\}, \tag{3.29}$$

where t_i and ℓ_i belong to the two subsequences defined above. Since $\mathcal{K} \subseteq \mathcal{S}$, using $\pi_k^m \geq \epsilon_{\text{mc}}$ and Assumption 3.2, for $k \in \mathcal{K}$ we have

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \geq \eta[m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)] \geq \eta c_1 \epsilon_{\text{mc}} \min\left(\frac{\epsilon_{\text{mc}}}{\kappa_H}, \Delta_k, 1\right) \geq 0. \tag{3.30}$$

The sequence $\{f(\boldsymbol{x}_k)\}$ is bounded below by assumption and is monotone decreasing. We must then have that $\{f(\boldsymbol{x}_k)\}$ converges. Since it converges, $\{f(\boldsymbol{x}_k)\}$ must be Cauchy, so the difference on the left hand side in (3.30) goes to zero as k goes to infinity. This implies that

$$\lim_{\substack{k\to\infty \\ k\in\mathcal{K}}} \Delta_k = 0. \tag{3.31}$$

Applying (3.31) to (3.30), we get that $\min(\epsilon_{\text{mc}}/\kappa_H, \Delta_k, 1) = \Delta_k$ for k large enough. So,

$$\Delta_k \leq \frac{1}{\eta c_1 \epsilon_{\text{mc}}} \left[f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})\right], \tag{3.32}$$

for k large enough. We can write

$$\|\boldsymbol{x}_{t_i} - \boldsymbol{x}_{\ell_i}\| \leq \sum_{\substack{j=t_i \\ j\in\mathcal{K}}}^{\ell_i - 1} \|\boldsymbol{x}_j - \boldsymbol{x}_{j+1}\| \leq \sum_{\substack{j=t_i \\ j\in\mathcal{K}}}^{\ell_i - 1} \Delta_j \leq \frac{1}{\eta c_1 \epsilon_{\text{mc}}} \sum_{\substack{j=t_i \\ j\in\mathcal{K}}}^{\ell_i - 1} \left[f(\boldsymbol{x}_j) - f(\boldsymbol{x}_{j+1})\right], \tag{3.33}$$

$$= \frac{1}{\eta c_1 \epsilon_{\text{mc}}} \left[f(\boldsymbol{x}_{t_i}) - f(\boldsymbol{x}_{\ell_i})\right], \tag{3.34}$$

observing that the first inequality comes from applying the triangle inequality to the telescoping sum $\left\|\boldsymbol{x}_{t_i} - \boldsymbol{x}_{t_{i+1}} + \boldsymbol{x}_{t_{i+1}} - \boldsymbol{x}_{t_{i+2}} + \cdots\right\|$, and the final inequality comes from (3.32). We can again use our assumption that f is bounded below with the fact that the sequence $\{f(\boldsymbol{x}_k)\}$ is monotone decreasing, to say that $\{f(\boldsymbol{x}_k)\}$ converges and is thus Cauchy. So the right hand side of (3.34) goes to zero as $i \to \infty$, and hence $\|\boldsymbol{x}_{t_i} - \boldsymbol{x}_{\ell_i}\|$ also goes to zero as $i \to \infty$.

Using Lemma 2.2, we have that $\left|\pi_{t_i}^f - \pi_{\ell_i}^f\right| \leq M \|\boldsymbol{x}_{t_i} - \boldsymbol{x}_{\ell_i}\| \to 0$. But from the definitions of $\{t_i\}$ and $\{\ell_i\}$, this is a contradiction to $\left|\pi_{t_i}^f - \pi_{\ell_i}^f\right| \geq \epsilon > 0$, a consequence of (3.28), and we are done. ∎

3.2. Worst-Case Complexity. We now analyze the worst-case complexity of Algorithm 2.1. That is, we bound the number of iterations and objective evaluations until $\pi_k^f < \epsilon$ is first achieved, in terms of the desired optimality level ϵ. To that end, we define i_ϵ to be the last iteration before $\pi_k^f < \epsilon$ for the first time (which exists for all $\epsilon > 0$ by Lemma 3.9). That is, $\pi_k^f \geq \epsilon$ for all $k \leq i_\epsilon$ and $\pi_{i_\epsilon+1}^f < \epsilon$.

LEMMA 3.11. *Suppose that Assumptions 3.1, 3.2 and 3.3 hold, and let S_{i_ϵ} be the set of iterations up to i_ϵ that are successful. Then*

$$|S_{i_\epsilon}| \leq \frac{f(\boldsymbol{x}_0) - f_{\text{low}}}{\eta c_1} \max\left(\epsilon_{\text{mc}}^{-1}\Delta_{\min}^{-1}, \epsilon_{\text{mc}}^{-1}\right), \tag{3.35}$$

where ϵ_{mc} is defined in (3.2) and $\Delta_{\min}$ in (3.9).

Proof. For all $k \in S_{i_\epsilon}$, we have the sufficient decrease condition:

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \geq \eta[m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)] \geq \eta c_1 \pi_k^m \min\left(\frac{\pi_k^m}{\kappa_H}, \Delta_k, 1\right). \tag{3.36}$$

From Lemmas 3.4 and 3.6, we have that $\pi_k^m \geq \epsilon_{\text{mc}}$ and $\Delta_k \geq \Delta_{\min}$ for all $k \leq i_\epsilon$, and hence (3.36) becomes

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \geq \eta c_1 \epsilon_{\text{mc}} \min\left(\frac{\epsilon_{\text{mc}}}{\kappa_H}, \Delta_{\min}, 1\right), \qquad \forall k \in S_{i_\epsilon}. \tag{3.37}$$

Summing (3.37) over all $k \in S_{i_\epsilon}$, and noting that $f_{\text{low}} \leq f(\boldsymbol{x}_k) \leq f(\boldsymbol{x}_0)$, we obtain

$$f(\boldsymbol{x}_0) - f_{\text{low}} \geq \sum_{k \in S_{i_\epsilon}} f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \geq |S_{i_\epsilon}| \, \eta c_1 \epsilon_{\text{mc}} \min\left(\frac{\epsilon_{\text{mc}}}{\kappa_H}, \Delta_{\min}, 1\right). \tag{3.38}$$

Finally, (3.35) follows from $\Delta_{\min} \leq \gamma_{\text{dec}}\epsilon_{\text{mc}}/\kappa_H \leq \epsilon_{\text{mc}}/\kappa_H$. ∎

It remains to count the number of iterations of Algorithm 2.1 that are not successful. Counting until iteration i_ϵ (inclusive), we let

1. $C_{i_\epsilon}^M$ be the set of criticality step iterations $k \leq i_\epsilon$ for which Δ_k is not reduced. This iteration occurs at most once per sequence of criticality steps;
2. $C_{i_\epsilon}^U$ be the set of criticality step iterations $k \leq i_\epsilon$ for which Δ_k is reduced. This amounts to every criticality iteration except perhaps the single iteration where Δ_k is not reduced;
3. M_{i_ϵ} be the set of iterations where the model-improvement step is called; and
4. U_{i_ϵ} be the set of unsuccessful iterations.

LEMMA 3.12. *Suppose that Assumptions 3.1, 3.2 and 3.3 hold. We have the bounds*

$$|U_{i_\epsilon}| + \left|C_{i_\epsilon}^U\right| \leq \frac{\log \gamma_{\text{inc}}}{|\log \gamma_{\text{dec}}|} |S_{i_\epsilon}| + \frac{\log(\Delta_0/\Delta_{\min})}{|\log \gamma_{\text{dec}}|}, \tag{3.39}$$

$$\left|C_{i_\epsilon}^M\right| + |M_{i_\epsilon}| \leq \left|C_{i_\epsilon}^U\right| + |S_{i_\epsilon}| + |U_{i_\epsilon}| + 1. \tag{3.40}$$

Proof. Note that if $k \in S_{i_\epsilon}$, we set $\Delta_{k+1} \leq \gamma_{\text{inc}}\Delta_k$; and if $k \in U_{i_\epsilon}$, we set $\Delta_{k+1} = \gamma_{\text{dec}}\Delta_k$. For iterations in $C_{i_\epsilon}^U$, we also set $\Delta_{k+1} = \gamma_{\text{dec}}\Delta_k$. Thus,

$$\gamma_{\text{inc}}^{|S_{i_\epsilon}|} \gamma_{\text{dec}}^{|U_{i_\epsilon}| + \left|C_{i_\epsilon}^U\right|} \geq \frac{\Delta_{i_\epsilon}}{\Delta_0} \geq \frac{\Delta_{\min}}{\Delta_0}, \tag{3.41}$$

where the second inequality follows from Lemma 3.6. The first result (3.39) follows by taking the log of both sides.

Now, if $k \in C_{i_\epsilon}^M \cup M_{i_\epsilon}$, then the model m_k was not fully linear, but the mechanisms of Algorithm 2.1 ensure that m_{k+1} is fully linear. Hence iteration $k+1$ must be a criticality step in which Δ_{k+1} is reduced, successful or unsuccessful. Thus $k+1 \in C_{i_\epsilon}^U \cup S_{i_\epsilon} \cup U_{i_\epsilon}$, or $k+1 > i_\epsilon$ (i.e. $k = i_\epsilon$), and (3.40) follows. ∎

To get an overall complexity bound, we will lastly need to assume that the algorithm parameter ϵ_C is not much smaller than the desired tolerance ϵ.

ASSUMPTION 3.13. *The algorithm parameter $\epsilon_C \geq c_2\epsilon$ for some constant $c_2 > 0$.*

As noted in [11], Assumption 3.13 can be easily satisfied by choosing ϵ_C appropriately in Algorithm 2.1.

THEOREM 3.14. *Suppose that Assumptions 3.1, 3.2, 3.3 and 3.13 hold. The number of iterations i_ϵ until $\pi^f_{i_\epsilon+1} < \epsilon$ is at most*

$$\begin{aligned}&\frac{f(\boldsymbol{x}_0)-f_{\text{low}}}{\eta c_1}\left(2+\frac{2\log\gamma_{\text{inc}}}{|\log\gamma_{\text{dec}}|}\right)\max\left(\frac{1}{c_3\epsilon\Delta_0},\frac{1}{c_3c_4\epsilon^2},\frac{1}{c_3\gamma_{\text{dec}}\epsilon},\frac{1}{c_3\epsilon}\right)\\ &\qquad+\frac{2}{|\log\gamma_{\text{dec}}|}\max\left(0,\log\left(\frac{\Delta_0}{c_4\epsilon}\right),\log\left(\gamma_{\text{dec}}^{-1}\Delta_0\right)\right)+1,\end{aligned}\tag{3.42}$$

where $c_3 := \min(c_2,(1+\kappa_{\text{eg}}\mu)^{-1})$ and

$$c_4 := \gamma_{\text{dec}}\min\left(\frac{1}{\kappa_{\text{eg}}+\mu^{-1}},\mu c_3,\frac{c_3}{\kappa_H},\left(\kappa_{\text{eg}}+\frac{2\kappa_{\text{ef}}}{c_1(1-\eta)}\right)^{-1}\right).\tag{3.43}$$

Proof. We first observe that, using Lemma 3.12, the total number of iterations can be written as

$$i_\epsilon = |S_{i_\epsilon}|+|M_{i_\epsilon}|+|U_{i_\epsilon}|+\left|C^M_{i_\epsilon}\right|+\left|C^U_{i_\epsilon}\right|,\tag{3.44}$$

$$\leq 2\left(|S_{i_\epsilon}|+|U_{i_\epsilon}|+\left|C^U_{i_\epsilon}\right|\right)+1,\tag{3.45}$$

$$\leq\left(2+\frac{2\log\gamma_{\text{inc}}}{|\log\gamma_{\text{dec}}|}\right)|S_{i_\epsilon}|+\frac{2\log(\Delta_0/\Delta_{\min})}{|\log\gamma_{\text{dec}}|}+1.\tag{3.46}$$

Hence, we apply Lemma 3.11 to obtain

$$i_\epsilon\leq\left(2+\frac{2\log\gamma_{\text{inc}}}{|\log\gamma_{\text{dec}}|}\right)\frac{f(\boldsymbol{x}_0)-f_{\text{low}}}{\eta c_1}\max\left(\epsilon_{\text{mc}}^{-1}\Delta_{\min}^{-1},\epsilon_{\text{mc}}^{-1}\right)+\frac{2\log(\Delta_0/\Delta_{\min})}{|\log\gamma_{\text{dec}}|}+1.\tag{3.47}$$

From Lemma 3.4 and Assumption 3.13, we have $\epsilon_{\text{mc}} = \min(\epsilon_C,\epsilon/(1+\kappa_{\text{eg}}\mu))\geq c_3\epsilon$. Similarly, (3.9) gives

$$\Delta_{\min}\geq\min\left(\Delta_0,\frac{\gamma_{\text{dec}}\epsilon}{\kappa_{\text{eg}}+\mu^{-1}},\gamma_{\text{dec}}\mu c_3\epsilon,\frac{\gamma_{\text{dec}}c_3\epsilon}{\kappa_H},\gamma_{\text{dec}}\left(\kappa_{\text{eg}}+\frac{2\kappa_{\text{ef}}}{c_1(1-\eta)}\right)^{-1}\epsilon,\gamma_{\text{dec}}\right),\tag{3.48}$$

$$=\min\left(\Delta_0,c_4\epsilon,\gamma_{\text{dec}}\right),\tag{3.49}$$

and so

$$\log\left(\frac{\Delta_0}{\Delta_{\min}}\right)\leq\max\left(0,\log\left(\frac{\Delta_0}{c_4\epsilon}\right),\log\left(\gamma_{\text{dec}}^{-1}\Delta_0\right)\right),\tag{3.50}$$

and

$$\max\left(\epsilon_{\text{mc}}^{-1}\Delta_{\min}^{-1},\epsilon_{\text{mc}}^{-1}\right)\leq\max\left(c_3^{-1}\epsilon^{-1}\Delta_{\min}^{-1},c_3^{-1}\epsilon^{-1}\right),\tag{3.51}$$

$$\leq\max\left(\frac{1}{c_3\epsilon\Delta_0},\frac{1}{c_3c_4\epsilon^2},\frac{1}{c_3\gamma_{\text{dec}}\epsilon},\frac{1}{c_3\epsilon}\right).\tag{3.52}$$

As desired, (3.42) follows from (3.47), (3.50) and (3.52). ☐

COROLLARY 3.15. *Suppose the assumptions of Theorem 3.14 hold. For $\epsilon\in(0,1]$, the number of iterations of Algorithm 2.1 until $\pi^f_k<\epsilon$ for the first time is at most $\mathcal{O}(\kappa_H\kappa_d^2\epsilon^{-2})$, where $\kappa_d := \max(\kappa_{\text{ef}},\kappa_{\text{eg}})$.*

Proof. We note that $c_0^{-1} = \mathcal{O}(\max(\kappa_H, \kappa_d))$ from Lemma 3.5, and $c_3^{-1} = \mathcal{O}(\kappa_d)$ and $c_4^{-1} = \mathcal{O}(\kappa_H \kappa_d)$ from Theorem 3.14. The result then follows immediately from Theorem 3.14. ∎

Remark 3.16. We will see in Section 4 that if we are using linear or composite linear interpolation models, then we require at most $\mathcal{O}(n)$ objective evaluations per iteration. Further, we show in Theorems 4.4 and 4.9 that under standard smoothness conditions we can get $\kappa_d = \mathcal{O}(n)$ for linear interpolation and $\kappa_d = \mathcal{O}(n^2)$ for composite linear interpolation. We also have Hessian bound $\kappa_H = 1$ for linear models (since there is no model Hessian) and the realistic estimate $\kappa_H = \mathcal{O}(\kappa_d) = \mathcal{O}(n^2)$ for composite linear interpolation (see (4.73)).

Combining Corollary 3.15 with Remark 3.16 gives a worst-case complexity bound of $\mathcal{O}(n^2\epsilon^{-2})$ iterations and $\mathcal{O}(n^3\epsilon^{-3})$ evaluations for linear interpolation models, and $\mathcal{O}(n^6\epsilon^{-2})$ iterations and $\mathcal{O}(n^7\epsilon^{-2})$ evaluations for composite linear interpolation. These results match the first-order complexity bounds for unconstrained model-based DFO both for fully linear models [21] and nonlinear least-squares with composite linear models [11].[4]

4. Construction of Fully Linear Models.

In this section we consider the requirement for full linearity in the constrained case (Definition 2.3). As above, we generalize the standard approach for constructing fully linear models, namely the concept of Λ-poisedness. Specifically, in this section, we look at approaches for constructing linear interpolation models which satisfy the fully linear requirement. We show that an approach based on maximizing Lagrange polynomials, similar to the unconstrained approach, allows us to construct fully linear models and/or verify if an existing model is fully linear. We also extend this approach to linear interpolation for composite function minimization, for which linear models are sufficient to yield high-quality model-based DFO algorithms [11].

4.1. Unconstrained Linear Interpolation.

Suppose we have a set of $n+1$ points in $\mathbb{R}^n$, denoted $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_n\}$. If we have evaluated our objective f at all these points, we can construct a linear model (c.f. (2.9))

$$f(\boldsymbol{y}) \approx m(\boldsymbol{y}) := c + \boldsymbol{g}^T(\boldsymbol{y} - \boldsymbol{y}_0), \tag{4.1}$$

by enforcing the interpolation conditions

$$f(\boldsymbol{y}_t) = m(\boldsymbol{y}_t), \qquad \forall t = 1, \ldots, n. \tag{4.2}$$

This reduces to the linear system

$$M \begin{bmatrix} c \\ \boldsymbol{g} \end{bmatrix} := \begin{bmatrix} 1 & (\boldsymbol{y}_0 - \boldsymbol{y}_0)^T \\ \vdots & \vdots \\ 1 & (\boldsymbol{y}_n - \boldsymbol{y}_0)^T \end{bmatrix} \begin{bmatrix} c \\ \boldsymbol{g} \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{y}_0) \\ \vdots \\ f(\boldsymbol{y}_n) \end{bmatrix}. \tag{4.3}$$

This system is invertible provided the directions $\{\boldsymbol{y}_t - \boldsymbol{y}_0 : t = 1, \ldots, n\}$ are linearly independent [18, Section 2.3].

Given our interpolation set, the Lagrange polynomials associated with with this set are the linear functions

$$\ell_t(\boldsymbol{y}) := c_t + \boldsymbol{g}_t^T(\boldsymbol{y} - \boldsymbol{y}_0), \qquad \forall t = 0, \ldots, n, \tag{4.4}$$

[4]Note that different authors use different conventions for estimating κ_d in terms of n; see [11, Remark 3.20] for details.

defined by $\ell_t(\boldsymbol{y}_s) = \delta_{s,t}$. These conditions give

$$M \begin{bmatrix} c_t \\ \boldsymbol{g}_t \end{bmatrix} = \boldsymbol{e}_t, \qquad \forall t = 0, \ldots, n, \tag{4.5}$$

where $\boldsymbol{e}_t \in \mathbb{R}^{n+1}$ is the t-th coordinate vector. More details of this construction are given in [18, Section 3.2].

In the case of linear interpolation, the key result regarding full linearity is the following.

THEOREM 4.1. *Suppose f satisfies Assumption 3.1. Then if $\|\boldsymbol{y}_t - \boldsymbol{y}_0\| \le \Delta$ for all $t = 1, \ldots, n$ and there exists $\Lambda \ge 1$ such that*

$$\max_{t=0,\ldots,n} |\ell_t(\boldsymbol{y})| \le \Lambda, \qquad \forall \boldsymbol{y} \in B(\boldsymbol{y}_0, \Delta), \tag{4.6}$$

then the linear interpolation model (4.1) satisfies

$$|f(\boldsymbol{y}) - m(\boldsymbol{y})| \le \left(\frac{(3 + (n+1)\Lambda)L_{\nabla f}}{2}\right)\Delta^2, \tag{4.7a}$$

$$\|\nabla f(\boldsymbol{y}) - \nabla m(\boldsymbol{y})\| \le \left(\frac{(2 + (n+1)\Lambda)L_{\nabla f}}{2}\right)\Delta, \tag{4.7b}$$

for all $\boldsymbol{y} \in B(\boldsymbol{y}_0, \Delta)$.

Proof. This comes from [18, Theorems 2.11, 2.12 & 3.14]. ∎

The condition (4.6) is called Λ-poisedness (e.g. [18, Definition 3.6]) and (4.7) gives the standard notion of full linearity (which immediately implies our fully linear notion—Definition 2.3—for any constraint set C) with specific values for κ_{ef} and κ_{eg}. We also note that (4.6) is closely related to the condition number of M [18, eq. (3.11) & Theorem 3.14], and that $\ell_0(\boldsymbol{y}_0) = 1$ implies $\Lambda \ge 1$.

Constrained Example. The standard notion of Λ-poisedness (4.6) allows us to construct fully linear models (Definition 2.3) via linear interpolation. However, to achieve this with a small value of Λ, it may be necessary to include interpolation points outside the feasible set C. For example, consider the case $C = \{(x_1, x_2) : |x_2| \le \epsilon\} \subset \mathbb{R}^2$ (for some $\epsilon \ll 1$) and the interpolation set $\{\boldsymbol{0}, (1, 0), (0, \epsilon)\}$ in $B(\boldsymbol{0}, 1)$. This is depicted in Figure 1. In the ball $B(\boldsymbol{0}, 1)$, these points are Λ-poised with $\Lambda \sim \epsilon^{-1}$, and so the corresponding fully linear constants (4.7) will be similarly large. However, if we instead only consider the size of the Lagrange polynomials only within the feasible set,

$$\max_{t=0,\ldots,n} |\ell_t(\boldsymbol{y})| \le \Lambda, \qquad \forall \boldsymbol{y} \in C \cap B(\boldsymbol{y}_0, \Delta), \tag{4.8}$$

then it is not hard[5] to show that our interpolation set satisfies (4.8) with $\Lambda \le 3$ for any choice of ϵ.

This example demonstrates that, in the constrained case, we are likely to get improved error bounds if we consider the size of the Lagrange polynomials not over the whole ball $B(\boldsymbol{y}_0, \Delta)$, but only the feasible subset of this ball.

[5]This can be shown by relaxing $B(\boldsymbol{y}_0, \Delta) \cap C$ in (4.8) to $\{(y_1, y_2) : |y_1| \le 1, |y_2| \le \epsilon\}$.

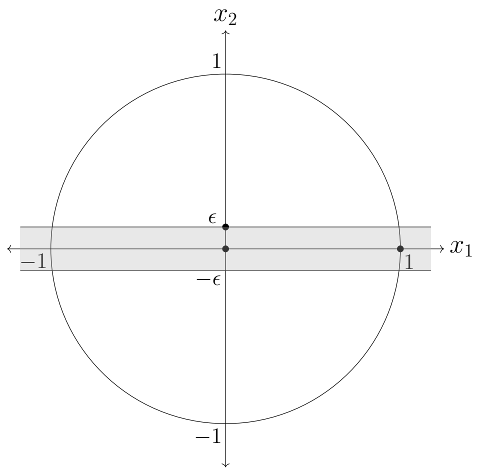


Fig. 1. Example feasible region and interpolation points with bad Λ-poisedness using (4.6) but good Λ-poisedness using (4.8).

4.2. Λ-poisedness in the constrained case. We now prove a generalization of Theorem 4.1 suitable for problems with convex constraints. To do this, we first introduce a notion of Λ-poisedness suitable for this setting.

DEFINITION 4.2. *Given $\Lambda \geq 1$, an interpolation set $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_n\} \subset C$ is Λ-poised for linear interpolation in $B(\boldsymbol{y}_0, \Delta) \cap C$ if $\{\boldsymbol{y}_t - \boldsymbol{y}_0 : t = 1, \ldots, n\}$ is linearly independent and*

$$\max_{t=0,\ldots,n} |\ell_t(\boldsymbol{y})| \leq \Lambda, \qquad \forall \boldsymbol{y} \in C \cap B(\boldsymbol{y}_0, \min(\Delta, 1)). \tag{4.9}$$

Note that this is slightly weaker than (4.8) as we only have $\boldsymbol{y} \in B(\boldsymbol{y}_0, \min(\Delta, 1))$ rather than $B(\boldsymbol{y}_0, \Delta)$ (c.f. [18, Lemma 10.25]).

LEMMA 4.3. *Suppose f satisfies Assumption 3.1 and C satisfies Assumption 2.1. Then if $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_n\}$ is Λ-poised for linear interpolation in $B(\boldsymbol{y}_0, \Delta) \cap C$ and $\|\boldsymbol{y}_t - \boldsymbol{y}_0\| \leq \beta \min(\Delta, 1)$ for all $t = 1, \ldots, n$ and some $\beta > 0$, we have*

$$\left|(\boldsymbol{y} - \boldsymbol{y}_0)^T (\boldsymbol{g} - \nabla f(\boldsymbol{y}_0))\right| \leq \frac{n\Lambda L_{\nabla f} \beta^2}{2} \min(\Delta, 1)^2, \tag{4.10}$$

for all $\boldsymbol{y} \in B(\boldsymbol{y}_0, \min(\Delta, 1)) \cap C$.

Proof. The first row of (4.3) gives $c = f(\boldsymbol{y}_0)$. Thus the remaining interpolation conditions give

$$(\boldsymbol{y}_t - \boldsymbol{y}_0)^T \boldsymbol{g} = f(\boldsymbol{y}_t) - f(\boldsymbol{y}_0), \qquad \forall t = 1, \ldots, n. \tag{4.11}$$

Therefore, for all $t = 1, \ldots, n$ we have

$$\left|(\boldsymbol{y}_t - \boldsymbol{y}_0)^T (\boldsymbol{g} - \nabla f(\boldsymbol{y}_0))\right| = \left|f(\boldsymbol{y}_t) - f(\boldsymbol{y}_0) - \nabla f(\boldsymbol{y}_0)^T (\boldsymbol{y}_t - \boldsymbol{y}_0)\right|, \tag{4.12}$$

$$\leq \frac{L}{2} \|\boldsymbol{y}_t - \boldsymbol{y}_0\|^2 \leq \frac{L_{\nabla f} \beta^2}{2} \min(\Delta, 1)^2, \tag{4.13}$$

which follows from Assumption 3.1 and $\|\boldsymbol{y}_t - \boldsymbol{y}_0\| \leq \beta \min(\Delta, 1)$.

Now, fix any $\boldsymbol{y} \in B(\boldsymbol{y}_0, \min(\Delta, 1)) \cap C$. Since $\{\boldsymbol{y}_t - \boldsymbol{y}_0 : t = 1, \ldots, n\}$ forms a basis of $\mathbb{R}^n$ (as it is linearly independent), there exist $\alpha_1(\boldsymbol{y}), \ldots, \alpha_n(\boldsymbol{y}) \in \mathbb{R}$ such that

$$\boldsymbol{y} - \boldsymbol{y}_0 = \sum_{t=1}^{n} \alpha_t(\boldsymbol{y})(\boldsymbol{y}_t - \boldsymbol{y}_0). \tag{4.14}$$

Hence from (4.13) we get

$$\left|(\boldsymbol{y} - \boldsymbol{y}_0)^T(\boldsymbol{g} - \nabla f(\boldsymbol{y}_0))\right| \leq \sum_{t=1}^{n} \left|\alpha_t(\boldsymbol{y})(\boldsymbol{y}_t - \boldsymbol{y}_0)^T(\boldsymbol{g} - \nabla f(\boldsymbol{y}_0))\right|, \tag{4.15}$$

$$\leq \frac{L_{\nabla f}\beta^2}{2}\left(\sum_{t=1}^{n} |\alpha_t(\boldsymbol{y})|\right)\min(\Delta, 1)^2. \tag{4.16}$$

We now construct a bound on each $|\alpha_t(\boldsymbol{y})|$ in terms of Λ. For $t = 1, \ldots, n$, the condition $\ell_t(\boldsymbol{y}_0) = 0$ gives $c_t = 0$, and so the remaining interpolation conditions $\ell_t(\boldsymbol{y}_s) = \delta_{s,t}$ give

$$L\boldsymbol{g}_t := \begin{bmatrix} (\boldsymbol{y}_1 - \boldsymbol{y}_0)^T \\ \vdots \\ (\boldsymbol{y}_n - \boldsymbol{y}_0)^T \end{bmatrix} \boldsymbol{g}_t = \boldsymbol{e}_t, \tag{4.17}$$

where here $\boldsymbol{e}_t \in \mathbb{R}^n$ is the t-th coordinate vector. Hence the Λ-poisedness condition gives

$$|\ell_t(\boldsymbol{y})| = \left|\boldsymbol{g}_t^T(\boldsymbol{y} - \boldsymbol{y}_0)\right| = \left|\boldsymbol{e}_t^T L^{-T}(\boldsymbol{y} - \boldsymbol{y}_0)\right| \leq \Lambda. \tag{4.18}$$

Separately, we can find the values $\alpha_t(\boldsymbol{y})$ by solving the linear system associated with (4.14), which is

$$\underbrace{\begin{bmatrix} (\boldsymbol{y}_1 - \boldsymbol{y}_0) & \cdots & (\boldsymbol{y}_n - \boldsymbol{y}_0) \end{bmatrix}}_{=L^T} \begin{bmatrix} \alpha_1(\boldsymbol{y}) \\ \vdots \\ \alpha_n(\boldsymbol{y}) \end{bmatrix} = \boldsymbol{y} - \boldsymbol{y}_0. \tag{4.19}$$

Combining this with (4.18) we get

$$|\alpha_t(\boldsymbol{y})| = \left|[L^{-T}(\boldsymbol{y} - \boldsymbol{y}_0)]_t\right| = \left|\boldsymbol{e}_t^T L^{-T}(\boldsymbol{y} - \boldsymbol{y}_0)\right| \leq \Lambda. \tag{4.20}$$

This and (4.16) gives us the result. □

As we might hope, this notion of Λ-poisedness guarantees that our interpolation models are fully linear.

THEOREM 4.4. *Suppose f and C satisfy Assumptions 3.1 and 2.1 respectively. Then if $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_n\}$ is Λ-poised for linear interpolation in $B(\boldsymbol{y}_0, \Delta) \cap C$ and $\|\boldsymbol{y}_t - \boldsymbol{y}_0\| \leq \beta \min(\Delta, 1)$ for all $t = 1, \ldots, n$ and some $\beta > 0$, then the interpolation model (4.1) is fully linear in $B(\boldsymbol{y}_0, \Delta)$ (in the sense of Definition 2.3) with constants*

$$\kappa_{\mathrm{ef}} = \frac{L_{\nabla f}(n\Lambda\beta^2 + 1)}{2}, \qquad \text{and} \qquad \kappa_{\mathrm{eg}} = \frac{n\Lambda L_{\nabla f}\beta^2}{2}, \tag{4.21}$$

in (2.11).

Proof. Fix $\boldsymbol{y} \in B(\boldsymbol{y}_0, \Delta) \cap C$. We first derive κ_{ef} by considering the cases $\Delta \geq 1$ and $\Delta < 1$ separately. If $\Delta \geq 1$ then $\hat{\boldsymbol{y}} := \boldsymbol{y}_0 + \Delta^{-1}(\boldsymbol{y} - \boldsymbol{y}_0) \in B(\boldsymbol{y}_0, 1) = B(\boldsymbol{y}_0, \min(\Delta, 1))$ and $\hat{\boldsymbol{y}} \in C$ since C is convex. We can then apply Lemma 4.3 to get

$$\left|(\boldsymbol{y} - \boldsymbol{y}_0)^T(\boldsymbol{g} - \nabla f(\boldsymbol{y}_0))\right| = \Delta \left|(\hat{\boldsymbol{y}} - \boldsymbol{y}_0)^T(\boldsymbol{g} - \nabla f(\boldsymbol{y}_0))\right|, \tag{4.22}$$

$$\leq \frac{n\Lambda L_{\nabla f}\beta^2}{2} \Delta \min(\Delta, 1)^2, \tag{4.23}$$

$$\leq \frac{n\Lambda L_{\nabla f}\beta^2}{2} \Delta^2, \tag{4.24}$$

where the last inequality follows from $\min(\Delta, 1)^2 = 1 \leq \Delta$. Instead, if $\Delta < 1$ then $\boldsymbol{y} \in B(\boldsymbol{y}_0, \min(\Delta, 1)) \cap C$ and so we apply Lemma 4.3 directly, giving

$$\left|(\boldsymbol{y} - \boldsymbol{y}_0)^T(\boldsymbol{g} - \nabla f(\boldsymbol{y}_0))\right| \leq \frac{n\Lambda L_{\nabla f}\beta^2}{2} \min(\Delta, 1)^2 = \frac{n\Lambda L_{\nabla f}\beta^2}{2} \Delta^2. \tag{4.25}$$

Finally, regardless of the size of Δ, we again use $c = f(\boldsymbol{y}_0)$ (as in the proof of Lemma 4.3) with either (4.24) or (4.25) to get

$$|f(\boldsymbol{y}) - m(\boldsymbol{y})| = \left|f(\boldsymbol{y}) - f(\boldsymbol{y}_0) - \boldsymbol{g}^T(\boldsymbol{y} - \boldsymbol{y}_0)\right|, \tag{4.26}$$

$$\leq \left|f(\boldsymbol{y}) - f(\boldsymbol{y}_0) - \nabla f(\boldsymbol{y}_0)^T(\boldsymbol{y} - \boldsymbol{y}_0)\right| + \left|(\boldsymbol{y} - \boldsymbol{y}_0)^T(\boldsymbol{g} - \nabla f(\boldsymbol{y}_0))\right|, \tag{4.27}$$

$$\leq \frac{L_{\nabla f}}{2}\Delta^2 + \frac{n\Lambda L_{\nabla f}\beta^2}{2}\Delta^2, \tag{4.28}$$

and we get the desired value of κ_{ef}.

To get κ_{eg} we now fix an arbitrary $\widetilde{\boldsymbol{y}} \in B(\boldsymbol{y}_0, 1) \cap C$ and again consider the cases $\Delta \geq 1$ and $\Delta < 1$ separately. First, if $\Delta \geq 1$, then $\widetilde{\boldsymbol{y}} \in B(\boldsymbol{y}_0, \min(\Delta, 1)) \cap C$, and applying Lemma 4.3 we get

$$\left|(\widetilde{\boldsymbol{y}} - \boldsymbol{y}_0)^T(\boldsymbol{g} - \nabla f(\boldsymbol{y}_0))\right| \leq \frac{n\Lambda L_{\nabla f}\beta^2}{2} \min(\Delta, 1)^2 \leq \frac{n\Lambda L_{\nabla f}\beta^2}{2} \Delta, \tag{4.29}$$

since $\min(\Delta, 1)^2 = 1 \leq \Delta$. Alternatively, if $\Delta < 1$ then the convexity of C implies that $\hat{\boldsymbol{y}} := \boldsymbol{y}_0 + \Delta(\widetilde{\boldsymbol{y}} - \boldsymbol{y}_0) \in B(\boldsymbol{y}_0, \Delta) \cap C = B(\boldsymbol{y}_0, \min(\Delta, 1)) \cap C$. Again we apply Lemma 4.3 and get

$$\left|(\widetilde{\boldsymbol{y}} - \boldsymbol{y}_0)^T(\boldsymbol{g} - \nabla f(\boldsymbol{y}_0))\right| = \Delta^{-1} \left|(\hat{\boldsymbol{y}} - \boldsymbol{y}_0)^T(\boldsymbol{g} - \nabla f(\boldsymbol{y}_0))\right|, \tag{4.30}$$

$$\leq \frac{n\Lambda L_{\nabla f}\beta^2}{2} \Delta^{-1} \min(\Delta, 1)^2 = \frac{n\Lambda L_{\nabla f}\beta^2}{2} \Delta. \tag{4.31}$$

The value for κ_{eg} then follows from (4.29) and (4.31). ∎

Remark 4.5. The proof of Theorem 4.1 strongly relies on the fact that $\|M\| = \|M^T\|$ (in the first line of [18, Theorem 3.14]), where M arises from the interpolation model and M^T arises from the construction of Lagrange polynomials. However, a similar statement is not necessarily true when we consider the matrix operator norm restricted to inputs in C, and so to prove Theorem 4.4 we introduce a novel approach to achieve very similar results.

Theorem 4.4 gives exactly the type of result we want: that in order to achieve accurate models inside the feasible region, we only need to control the size of $\ell_t(\boldsymbol{y})$

inside the feasible region. As our example in Figure 1 illustrates, this can be a substantially easier requirement to satisfy, particularly with the restriction that all interpolation points must be feasible.

With regards to our worst-case complexity bound Corollary 3.15, we note that Theorem 4.4 gives $\kappa_{\mathrm{ef}}, \kappa_{\mathrm{eg}} = \mathcal{O}(nL_{\nabla f}\Lambda)$, which is the same size as in Theorem 4.1 (and so in terms of n, L and Λ our results match the complexity bounds from [21]).

4.3. Constructing Λ-Poised Sets. We now turn our attention to procedures which check whether or not a set is Λ-poised (in the sense of Definition 4.2), and to modify a non-poised set to ensure that it is Λ-poised. For simplicity of notation, in this section we define $r := \beta\min(\Delta, 1)$.

4.3.1. Verifying Λ-poisedness. Verifying whether or not a set $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_n\} \subset C$ is Λ-poised per Definition 4.2 is straightforward, based on the procedure:

1. Verify if the set $\{\boldsymbol{y}_t - \boldsymbol{y}_0 : t = 1, \ldots, n\}$ is linearly independent (e.g. via QR factorization, or checking if the matrix M in (4.5) is invertible);
2. Construct each Lagrange polynomial $\ell_t(\boldsymbol{y})$ for $t = 0, \ldots, n$ by solving the corresponding system (4.5);
3. For each $t = 0, \ldots, n$, calculate
$$\max_{\boldsymbol{y}\in C\cap B(\boldsymbol{y}_0, \min(\Delta,1))} |\ell_t(\boldsymbol{y})|, \tag{4.32}$$
and verify if this quantity is at most Λ. This reduces to solving two smooth convex optimization problems—minimizing $\pm\ell_t(\boldsymbol{y})$—for each t (e.g. using the projected gradient method [15, Theorem 12.1.4]), at least to sufficient accuracy that it is known whether the optimal value is at most Λ.

Further, to guarantee fully linear models (using Theorem 4.4) we also need to ensure each interpolation point $\boldsymbol{y}_t \in C \cap B(\boldsymbol{y}_0, r)$: clearly $\|\boldsymbol{y}_t - \boldsymbol{y}_0\| \leq r$ is easily checked, and $\boldsymbol{y}_t \in C$ can be checked by verifying $\mathrm{proj}_C(\boldsymbol{y}_t) = \boldsymbol{y}_t$.

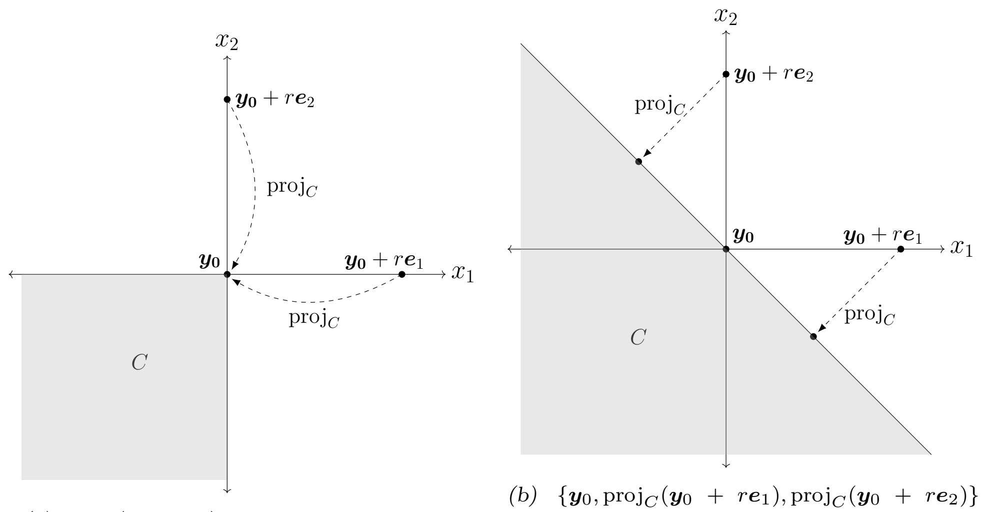
4.3.2. Constructing Affinely Independent Points. If the process to verify Λ-poisedness outlined in Section 4.3.1 fails, the first reason for this would be that the interpolation set $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_n\}$ does not produce n linearly independent directions $\{\boldsymbol{y}_t - \boldsymbol{y}_0 : t = 1, \ldots, n\}$ or is not contained in $C \cap B(\boldsymbol{y}_0, r)$. In this case, we need to replace our interpolation set by one satisfying these properties. At this stage we are not concerned with verifying the Λ-poisedness property (4.9); this is addressed in Section 4.3.3.

In the unconstrained case, it is easy to create a suitable replacement set: simply taking $\boldsymbol{y}_0$ and $\boldsymbol{y}_t = \boldsymbol{y}_0 + r\boldsymbol{e}_t$ for $t = 1, \ldots, n$ is sufficient. However, this is more difficult when $C \neq \mathbb{R}^n$, as two undesirable situations may arise, even from sampling directions orthogonal to $\boldsymbol{y}_0$:

- $\mathrm{proj}_C(\boldsymbol{y}_0 + r\boldsymbol{e}_t)$ might be equal to $\boldsymbol{y}_0$ for some or all $t = 1, \ldots, n$. This arises, for example if $C = \{\boldsymbol{x} \in \mathbb{R}^2 : x_1, x_2 \leq 0\}$ and $\boldsymbol{y}_0 = (0, 0)$.
- $\mathrm{proj}_C(\boldsymbol{y}_0 + r\boldsymbol{e}_{t_1}) - \boldsymbol{y}_0$ and $\mathrm{proj}_C(\boldsymbol{y}_0 + r\boldsymbol{e}_{t_2}) - \boldsymbol{y}_0$ may not be linearly independent even if $t_1 \neq t_2$. For example, if $C = \{\boldsymbol{x} \in \mathbb{R}^2 : x_1 + x_2 \leq 0\}$ and $\boldsymbol{y}_0 = (0, 0)$, then $\mathrm{proj}_C(\boldsymbol{y}_0 + r\boldsymbol{e}_1) - \boldsymbol{y}_0 = (r/2, -r/2)$ is parallel to $\mathrm{proj}_C(\boldsymbol{y}_0 + r\boldsymbol{e}_2) - \boldsymbol{y}_0 = (-r/2, r/2)$.

These situations are depicted in Figure 2.

To address these difficulties and guarantee that we get a linearly independent set of directions after projecting onto C, we replace the perturbations $\{r\boldsymbol{e}_t : t = 1, \ldots, n\}$ with a countably infinite set of directions which are dense on the sphere $\{\boldsymbol{d} \in \mathbb{R}^n : \|\boldsymbol{d}\| = r\}$, and continue adding points of the form $\mathrm{proj}_C(\boldsymbol{y}_0 + \boldsymbol{d})$ to the interpolation


x_2
$\boldsymbol{y_0} + r\boldsymbol{e}_2$
proj_C
$\boldsymbol{y_0}$
$\boldsymbol{y_0} + r\boldsymbol{e}_1$
x_1
proj_C
C
x_2
$\boldsymbol{y_0} + r\boldsymbol{e}_2$
proj_C
$\boldsymbol{y_0}$
$\boldsymbol{y_0} + r\boldsymbol{e}_1$
x_1
proj_C
C

(a) $\mathrm{proj}_C(\boldsymbol{y}_0 + r\boldsymbol{e}_t) = \boldsymbol{y}_0$ *for all* $t = 1, \ldots, n$.

(b) $\{\boldsymbol{y}_0, \mathrm{proj}_C(\boldsymbol{y}_0 + r\boldsymbol{e}_1), \mathrm{proj}_C(\boldsymbol{y}_0 + r\boldsymbol{e}_2)\}$ *colinear.*

Fig. 2. Examples of potential difficulties constructing affinely independent points using $\mathrm{proj}_C(\boldsymbol{y}_0 + r\boldsymbol{e}_t)$ *for* $t = 1, \ldots, n$.

Algorithm 4.1 Method for constructing affinely independent points.

```
Input: Constraint set $C \subset \mathbb{R}^n$ with projection operator $\mathrm{proj}_C : \mathbb{R}^n \to C$, starting
    point $\boldsymbol{y}_0 \in C$, and sampling radius $r > 0$.
 1: Construct a sequence $(\boldsymbol{d}_k)_{k\in\mathbb{N}}$ of unit vectors $\boldsymbol{d}_k \in \mathbb{R}^n$ which are dense on the
    unit sphere, and initialize $\mathcal{Y} = \{\boldsymbol{y}_0\}$ and $\mathcal{D} = \emptyset$.
 2: for $k = 1, 2, \ldots$ do
 3:     Set $\boldsymbol{y}_k = \mathrm{proj}_C(\boldsymbol{y}_0 + r\boldsymbol{d}_k)$.
 4:     if $\boldsymbol{y}_k - \boldsymbol{y}_0 \notin \mathrm{span}\,\mathcal{D}$ then
 5:         Set $\mathcal{Y} = \mathcal{Y} \cup \{\boldsymbol{y}_k\}$ and $\mathcal{D} = \mathcal{D} \cup \{\boldsymbol{y}_k - \boldsymbol{y}_0\}$.
 6:         if $\mathcal{Y}$ has $n+1$ elements then
 7:             return $\mathcal{Y}$.
 8:         end if
 9:     end if
10: end for
```

set until we reach $n+1$ points with n linearly independent directions. The precise algorithm is given in Algorithm 4.1. We note that there are many constructions for sampling dense directions on the unit sphere, such as the methods used to generate poll directions for Mesh Adaptive Direct Search (MADS) [2, 1].

THEOREM 4.6. *If C satisfies Assumption 2.1 and $\boldsymbol{y}_0 \in C$, then Algorithm 4.1 terminates in finite time, returning a set $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_n\} \subset C \cap B(\boldsymbol{y}_0, r)$ for which $\{\boldsymbol{y}_t - \boldsymbol{y}_0 : t = 1, \ldots, n\}$ is linearly independent.*

Proof. Provided that Algorithm 4.1 terminates, then by construction we know that it will return a set $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_n\} \subset C$ for which $\{\boldsymbol{y}_t - \boldsymbol{y}_0 : t = 1, \ldots, n\}$ is linearly independent. Since $\boldsymbol{y}_0 \in C$ we also have

$$\|\boldsymbol{y}_k - \boldsymbol{y}_0\| = \|\operatorname{proj}_C(\boldsymbol{y}_0 + r\boldsymbol{d}_k) - \operatorname{proj}_C(\boldsymbol{y}_0)\| \le r\|\boldsymbol{d}_k\| = r, \tag{4.33}$$

where the inequality follows from the non-expansiveness of the projection operator (e.g. [5, Theorem 6.42]). Hence each $\boldsymbol{y}_t \in B(\boldsymbol{y}_0, r)$ and the result follows.

It remains to prove that Algorithm 4.1 terminates in finite time. We do this by showing that, for any $\mathcal{D}$ with $|\mathcal{D}| < n$, there will be infinitely many iterations for which $\boldsymbol{y}_k - \boldsymbol{y}_0 \notin \operatorname{span}\mathcal{D}$. On the first such iteration, $\mathcal{Y}$ and $\mathcal{D}$ gain one more element, and then either $|\mathcal{D}| = n$ and we are done, or we are again in the case $|\mathcal{D}| < n$ (and so again there are infinitely many future iterations in which $\mathcal{D}$ can be expanded).

To this end, suppose at any iteration that $|\mathcal{D}| < n$. Since C has nonempty interior and $\boldsymbol{y}_0 \in C$, there exists $\widetilde{\boldsymbol{y}} \in \operatorname{int}(C) \cap B(\boldsymbol{y}_0, r/2)$ with $\widetilde{\boldsymbol{y}} \notin \boldsymbol{y}_0 + \operatorname{span}\mathcal{D}$ (since $\boldsymbol{y}_0 + \operatorname{span}\mathcal{D}$ is a set of measure zero). Let $\widetilde{\boldsymbol{y}}_{\mathcal{D}}$ be the closest point to $\widetilde{\boldsymbol{y}}$ in $\boldsymbol{y}_0 + \operatorname{span}\mathcal{D}$, and so $\widetilde{\boldsymbol{y}}_{\mathcal{D}} \neq \widetilde{\boldsymbol{y}}$ and $\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}} \in (\operatorname{span}\mathcal{D})^{\perp}$. The non-expansiveness of projections implies $\|\widetilde{\boldsymbol{y}}_{\mathcal{D}} - \boldsymbol{y}_0\| \le \|\widetilde{\boldsymbol{y}} - \boldsymbol{y}_0\| \le r/2$

We will show that, since $\widetilde{\boldsymbol{y}} \in C$, points near to the line between $\widetilde{\boldsymbol{y}}_{\mathcal{D}}$ and $\widetilde{\boldsymbol{y}}$, but which are closer to $\widetilde{\boldsymbol{y}}$, give directions for which $\operatorname{proj}_C(\boldsymbol{y}_0 + r\boldsymbol{d}) \notin \boldsymbol{y}_0 + \operatorname{span}\mathcal{D}$. The line between $\widetilde{\boldsymbol{y}}_{\mathcal{D}}$ and $\widetilde{\boldsymbol{y}}$ can be parametrized by $\alpha \mapsto \widetilde{\boldsymbol{y}}_{\mathcal{D}} + \alpha(\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}})$. We can compute

$$\|\widetilde{\boldsymbol{y}}_{\mathcal{D}} + \alpha(\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}) - \boldsymbol{y}_0\|^2 = \alpha^2\|\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}\|^2 + 2\alpha(\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}})^T(\widetilde{\boldsymbol{y}}_{\mathcal{D}} - \boldsymbol{y}_0) + \|\widetilde{\boldsymbol{y}}_{\mathcal{D}} - \boldsymbol{y}_0\|^2. \tag{4.34}$$

Since $\|\widetilde{\boldsymbol{y}}_{\mathcal{D}} - \boldsymbol{y}_0\|^2 \le r^2/4 < r^2$ and $\|\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}\|^2 > 0$, there exists a unique $\alpha^* > 0$ such that $\|\widetilde{\boldsymbol{y}}_{\mathcal{D}} + \alpha(\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}) - \boldsymbol{y}_0\|^2 = r^2$. Let $\widetilde{\boldsymbol{y}}^* := \widetilde{\boldsymbol{y}}_{\mathcal{D}} + \alpha^*(\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}})$ and $\boldsymbol{d}^* := \widetilde{\boldsymbol{y}}^* - \boldsymbol{y}_0$. For this point, we have $\widetilde{\boldsymbol{y}}^* - \widetilde{\boldsymbol{y}}_{\mathcal{D}} = \alpha^*(\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}) \in (\operatorname{span}\mathcal{D})^{\perp}$, and so $\widetilde{\boldsymbol{y}}_{\mathcal{D}} = \operatorname{proj}_{\boldsymbol{y}_0 + \operatorname{span}\mathcal{D}}(\widetilde{\boldsymbol{y}}^*)$.

In fact, we must have $\alpha^* > 1$, since if $\alpha \in [0, 1]$ we have

$$\|\widetilde{\boldsymbol{y}}_{\mathcal{D}} + \alpha(\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}) - \boldsymbol{y}_0\| \le (1-\alpha)\|\widetilde{\boldsymbol{y}}_{\mathcal{D}} - \boldsymbol{y}_0\| + \alpha\|\widetilde{\boldsymbol{y}} - \boldsymbol{y}_0\| \le r/2. \tag{4.35}$$

Thus

$$\|\widetilde{\boldsymbol{y}}^* - \widetilde{\boldsymbol{y}}\| = |1-\alpha^*|\,\|\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}\| = \alpha^*\|\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}\| - \|\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}\|, \tag{4.36}$$

$$< \alpha^*\|\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}\| = \|\widetilde{\boldsymbol{y}}^* - \widetilde{\boldsymbol{y}}_{\mathcal{D}}\|. \tag{4.37}$$

Hence $\widetilde{\boldsymbol{y}}^*$ is strictly closer to $\widetilde{\boldsymbol{y}} \in C$ than $\widetilde{\boldsymbol{y}}_{\mathcal{D}}$, and so we conclude that $\operatorname{proj}_C(\widetilde{\boldsymbol{y}}^*) \notin \boldsymbol{y}_0 + \operatorname{span}\mathcal{D}$.

Lastly, consider any $\boldsymbol{y} \in B(\widetilde{\boldsymbol{y}}^*, \|\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}\|/4)$. Then we have

$$\|\boldsymbol{y} - \operatorname{proj}_{\boldsymbol{y}_0 + \operatorname{span}\mathcal{D}}(\boldsymbol{y})\| \ge \|\widetilde{\boldsymbol{y}}^* - \operatorname{proj}_{\boldsymbol{y}_0 + \operatorname{span}\mathcal{D}}(\boldsymbol{y})\| - \|\boldsymbol{y} - \widetilde{\boldsymbol{y}}^*\|, \tag{4.38}$$

$$\ge \|\widetilde{\boldsymbol{y}}^* - \widetilde{\boldsymbol{y}}_{\mathcal{D}}\| - \|\boldsymbol{y} - \widetilde{\boldsymbol{y}}^*\|, \tag{4.39}$$

$$\ge \left(\alpha^* - \frac{1}{4}\right)\|\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}\|, \tag{4.40}$$

$$> \left(\alpha^* - \frac{3}{4}\right)\|\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}\|, \tag{4.41}$$

$$\ge \|\widetilde{\boldsymbol{y}}^* - \widetilde{\boldsymbol{y}}\| + \|\boldsymbol{y} - \widetilde{\boldsymbol{y}}^*\|, \tag{4.42}$$

$$\ge \|\boldsymbol{y} - \widetilde{\boldsymbol{y}}\|, \tag{4.43}$$

where (4.39) follows as $\widetilde{\boldsymbol{y}}^*$ is closer to $\widetilde{\boldsymbol{y}}_{\mathcal{D}}$, its projection onto $\boldsymbol{y}_0 + \operatorname{span}\mathcal{D}$, than $\operatorname{proj}_{\boldsymbol{y}_0 + \operatorname{span}\mathcal{D}}(\boldsymbol{y})$, and (4.42) follows from (4.36). Thus, just like $\widetilde{\boldsymbol{y}}^*$, the point $\boldsymbol{y}$ also satisfies $\operatorname{proj}_C(\boldsymbol{y}) \notin \boldsymbol{y}_0 + \operatorname{span}\mathcal{D}$. Since the directions $\boldsymbol{d}_k$ in Algorithm 4.1 are dense in the unit sphere, there must be infinitely many k for which $\boldsymbol{y}_0 + r\boldsymbol{d}_k \in B(\widetilde{\boldsymbol{y}}^*, \|\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}_{\mathcal{D}}\|/4)$, and we are done. ∎

4.3.3. Geometry Improving Algorithm. The last component we need is a geometry improving algorithm: that is, given an interpolation set $\{\boldsymbol{y}_0, \dots, \boldsymbol{y}_n\}$ contained in $C \cap B(\boldsymbol{y}_0, r)$ for which $\{\boldsymbol{y}_t - \boldsymbol{y}_0 : t = 1, \dots, n\}$ is linearly independent, but where (4.9) does not hold, modify the set so that (4.9) also holds. Together with Algorithm 4.1, this gives us a method for constructing a Λ-poised interpolation set.

Fortunately, the poisedness-improving mechanism for the unconstrained case [18, Algorithm 6.3] remains suitable in the constrained case. First, suppose our initial interpolation set is $\mathcal{Y} := \{\boldsymbol{y}_0, \dots, \boldsymbol{y}_n\} \subset C$. Then

$$\begin{bmatrix} 1 & 0 \\ -\boldsymbol{e} & I \end{bmatrix} \underbrace{\begin{bmatrix} 1 & \boldsymbol{y}_0^T \\ \vdots & \vdots \\ 1 & \boldsymbol{y}_n^T \end{bmatrix}}_{=:M} = \begin{bmatrix} 1 & \boldsymbol{y}_0^T \\ 0 & (\boldsymbol{y}_1 - \boldsymbol{y}_0)^T \\ \vdots & \vdots \\ 0 & (\boldsymbol{y}_n - \boldsymbol{y}_0)^T \end{bmatrix}, \tag{4.44}$$

where $\boldsymbol{e} := (1, \dots, 1)^T \in \mathbb{R}^n$. Hence $\det M = \det L \neq 0$ since L has linearly independent rows by assumption.

Now, if $h : \mathbb{R}^n \to \mathbb{R}$ is any linear function, it must be equal to its linear interpolant (in Lagrange form) from [18, Lemma 3.5]; that is,

$$h(\boldsymbol{y}) = \sum_{t=0}^{n} h(\boldsymbol{y}_t)\ell_t(\boldsymbol{y}). \tag{4.45}$$

By taking the family of functions $h_0(\boldsymbol{y}) = 1$ and $h_j(\boldsymbol{y}) = y_j$ for $j = 1, \dots, n$ we get

$$\underbrace{\begin{bmatrix} 1 & \cdots & 1 \\ \boldsymbol{y}_0 & \cdots & \boldsymbol{y}_n \end{bmatrix}}_{=M^T} \begin{bmatrix} \ell_0(\boldsymbol{y}) \\ \vdots \\ \ell_n(\boldsymbol{y}) \end{bmatrix} = \begin{bmatrix} 1 \\ \boldsymbol{y} \end{bmatrix}, \tag{4.46}$$

which is [18, Eq. (3.4)] specialized to linear interpolation. Hence by Cramer's rule, $|\ell_t(\boldsymbol{y})|$ is the relative change in volume of the simplex given by $\mathcal{Y}$ when $\boldsymbol{y}_t$ is replaced by $\boldsymbol{y}$ (see [18, p. 41] for details). The full poisedness improving algorithm is given in Algorithm 4.2, a simple extension of [18, Algorithm 6.3].

THEOREM 4.7. *If C satisfies Assumption 2.1, $\boldsymbol{y}_0 \in C$, and we run Algorithm 4.2 with $\Lambda > 1$, $\Delta > 0$ and $\beta \geq 1$, then it terminates in finite time, returning a set $\mathcal{Y} \subset C \cap B(\boldsymbol{y}_0, \beta \min(\Delta, 1))$ which is Λ-poised for linear interpolation in $B(\boldsymbol{y}_0, \Delta) \cap C$.*

Proof. The initial $\mathcal{Y}$ is contained in $C \cap B(\boldsymbol{y}_0, \beta \min(\Delta, 1))$ by assumption or by Theorem 4.6. Any points added during the main loop of Algorithm 4.2 must be in $C \cap B(\boldsymbol{y}_0, \min(\Delta, 1))$, and so since $\beta \geq 1$ the output $\mathcal{Y}$ is also contained in $C \cap B(\boldsymbol{y}_0, \beta \min(\Delta, 1))$.

By construction, the output of Algorithm 4.2 (if it terminates) must be Λ-poised for linear interpolation in $B(\boldsymbol{y}_0, \Delta) \cap C$. It remains to show that Algorithm 4.2 terminates in finite time.

Let V_k be the volume of the simplex $\text{vol}(\mathcal{Y}) > 0$ at the start of iteration k of Algorithm 4.2. In particular, we have $V_1 > 0$ by assumption or by Theorem 4.6 (from $\det L \neq 0$). At each iteration of Algorithm 4.2, we modify $\mathcal{Y}$ in such a way that $V_{k+1} \geq \Lambda V_k$ by the reasoning above. So, if $V_{\max}$ is the maximum volume of all simplices contained in $C \cap B(\boldsymbol{y}_0, \min(\Delta, 1))$, then Algorithm 4.2 must terminate after at most $\lceil \log_\Lambda(V_{\max}/V_1) \rceil$ iterations. ∎

Algorithm 4.2 Method for constructing Λ-poised sets.

```
Input: Constraint set $C \subset \mathbb{R}^n$ with projection operator $\text{proj}_C : \mathbb{R}^n \to C$, starting
    point $\boldsymbol{y}_0 \in C$, poisedness constant $\Lambda > 1$, trust-region radius $\Delta$ and scaling factor
    $\beta \geq 1$.
1: If a suitable initial interpolation set is not available, using Algorithm 4.1, construct
    a set $\mathcal{Y} \subset C \cap B(\boldsymbol{y}_0, \beta \min(\Delta, 1))$ with $|\det M| > 0$.
2: for $k = 1, 2, \ldots$ do
3:     Construct the Lagrange polynomials $\ell_t$ for $t = 0, \ldots, n$ for $\mathcal{Y}$ by solving the
       system (4.5) and calculate
       (4.47)   $\Lambda_t := \max_{\boldsymbol{y} \in C \cap B(\boldsymbol{y}_0, \min(\Delta,1))} |\ell_t(\boldsymbol{y})|.$
4:     if $\Lambda_t > \Lambda$ for some $t$ then
5:         For some $t_k$ and $\boldsymbol{y}_k$ with $|\ell_{t_k}(\boldsymbol{y}_k)| > \Lambda$, set $\mathcal{Y} = \mathcal{Y} \setminus \{\boldsymbol{y}_{t_k}\} \cup \{\boldsymbol{y}_k\}$.
6:     else
7:         return $\mathcal{Y}$.
8:     end if
9: end for
```

Of course, Theorem 4.7 with Theorem 4.4 gives a way of constructing fully linear models in Algorithm 2.1. We note that, compared to the unconstrained result [18, Theorem 6.3], Theorem 4.7 is a weaker result in that the number of iterations required for Algorithm 4.2 to terminate is not uniformly bounded.

4.4. Application to Composite Minimization. In the previous sections, we outline how to construct linear interpolation models which satisfy the fully linear condition Definition 2.3. Model-based DFO methods based on linear interpolation is particularly useful (e.g. [11]) for composite minimization, where the objective has the form

$$f(\boldsymbol{x}) = F(\boldsymbol{r}(\boldsymbol{x})), \tag{4.48}$$

where $\boldsymbol{r} : \mathbb{R}^n \to \mathbb{R}^N$ is a black-box function for which derivatives are unavailable, and $F : \mathbb{R}^N \to \mathbb{R}$ is some known function (e.g. $F(\boldsymbol{r}) = \frac{1}{2}\|\boldsymbol{r}\|^2$ gives nonlinear least-squares minimization).

ASSUMPTION 4.8. *The composite objective* (4.48) *satisfies:*

(a) On the set $\cup_k B(\boldsymbol{x}_k, \Delta_{\max})$, $\boldsymbol{r}$ is continuously differentiable and its Jacobian $J : \mathbb{R}^n \to \mathbb{R}^{N\times n}$ is Lipschitz continuous with constant L_J and uniformly bounded by $J_{\max}$; and

(b) On the image under $\boldsymbol{r}$ of the set $\cup_k B(\boldsymbol{x}_k, \Delta_{\max})$, F is twice continuously differentiable with its gradient bounded by $G_{\max}$ and its Hessian bounded by $H_{\max}$, and is bounded below by F_{low}.

We note that under Assumption 4.8 that each component r_i of $\boldsymbol{r}$ has L_J-Lipschitz continuous gradients, since

$$\|\nabla r_i(\boldsymbol{x}_1) - \nabla r_i(\boldsymbol{x}_2)\| = \|(J(\boldsymbol{x}_1) - J(\boldsymbol{x}_2))^T \boldsymbol{e}_i\| \leq L_J \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \, \|\boldsymbol{e}_i\|. \tag{4.49}$$

Also, we have that f is differentiable with $\nabla f(\boldsymbol{x}) = J(\boldsymbol{x})^T \nabla F(\boldsymbol{r}(\boldsymbol{x}))$, and ∇f is

Lipschitz continuous, from

$$\begin{aligned} \|\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2)\| &\le \|J(\boldsymbol{x}_1)\| \, \|\nabla F(\boldsymbol{r}(\boldsymbol{x}_1)) - \nabla F(\boldsymbol{r}(\boldsymbol{x}_2))\| \\ &\quad + \|J(\boldsymbol{x}_1) - J(\boldsymbol{x}_2)\| \, \|\nabla F(\boldsymbol{r}(\boldsymbol{x}_2))\|, \quad (4.50) \\ &\le J_{\max} H_{\max} \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| + L_J G_{\max} \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|. \quad (4.51) \end{aligned}$$

All together, this means that our composite $f(\boldsymbol{x})$ (4.48) satisfies Assumption 3.1 with $L_{\nabla f} := J_{\max} H_{\max} + L_J G_{\max}$ and so the convergence theory from Section 3 applies.

Fully Linear Model Construction. In this situation, we assume that we have the ability to evaluate $\boldsymbol{r}(\boldsymbol{x})$, and so can construct a linear interpolation model

$$\boldsymbol{r}(\boldsymbol{y}) \approx \boldsymbol{m}(\boldsymbol{y}) := \boldsymbol{c} + J(\boldsymbol{y} - \boldsymbol{y}_0), \tag{4.52}$$

by requiring that $\boldsymbol{r}(\boldsymbol{y}_t) = \boldsymbol{m}(\boldsymbol{y}_t)$ for all $t = 0, \ldots, n$ for some interpolation set $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_n\}$. This gives us the linear system (c.f. (4.3))

$$M \begin{bmatrix} \boldsymbol{c}^T \\ J^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}(\boldsymbol{y}_0) \\ \vdots \\ \boldsymbol{r}(\boldsymbol{y}_n) \end{bmatrix}. \tag{4.53}$$

We then build a local quadratic model for $f(\boldsymbol{x})$ by taking a Taylor series for F, evaluated at $\boldsymbol{m}$:

$$f(\boldsymbol{y}) \approx m(\boldsymbol{y}) := c + \boldsymbol{g}^T(\boldsymbol{y} - \boldsymbol{y}_0) + \frac{1}{2}(\boldsymbol{y} - \boldsymbol{y}_0)^T H (\boldsymbol{y} - \boldsymbol{y}_0), \tag{4.54}$$

where the model gradient and Hessian are $\boldsymbol{g} := J^T \nabla F(\boldsymbol{c})$ and $H := J^T \nabla^2 F(\boldsymbol{c}) J$. The main result of this section is the following, which says that $m(\boldsymbol{y})$ is a quadratic fully linear model for f, provided we can build linear models for $\boldsymbol{r}$ with interpolation over Λ-poised sets.

THEOREM 4.9. *Suppose Assumption 4.8 holds and $\Delta \le \Delta_{\max}$. If the model $\boldsymbol{m}$ is constructed from an interpolation set $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_n\}$ which is Λ-poised for linear interpolation in $B(\boldsymbol{y}_0, \Delta) \cap C$ and $\|\boldsymbol{y}_t - \boldsymbol{y}_0\| \le \beta \min(\Delta, 1)$, then $m(\boldsymbol{y})$ (4.54) is a fully linear model for f in $B(\boldsymbol{y}_0, \Delta)$, with*

$$\kappa_{\mathrm{ef}} = \frac{L_{\nabla f}}{2} + G_{\max}\sqrt{N}\frac{n\Lambda L_r \beta^2}{2} + H_{\max}\left(\sqrt{N}\Delta_{\max}\frac{n\Lambda L_r \beta^2}{2} + J_{\max}\right)^2, \quad \textit{and} \tag{4.55}$$

$$\kappa_{\mathrm{eg}} = G_{\max}\sqrt{N}\frac{n\Lambda L_r \beta^2}{2}. \tag{4.56}$$

Proof. The interpolation condition $\boldsymbol{r}(\boldsymbol{y}_0) = \boldsymbol{m}(\boldsymbol{y}_0)$ gives $\boldsymbol{c} = \boldsymbol{r}(\boldsymbol{y}_0)$ and $c = f(\boldsymbol{y}_0)$. From Theorem 4.4 we have that each component model $m_t(\boldsymbol{y})$ is fully linear for $r_t(\boldsymbol{y})$ in $B(\boldsymbol{y}_0, \Delta)$.

Now, fix $\boldsymbol{d} \in C \cap B(\boldsymbol{y}_0, 1)$. Then

$$\begin{aligned} |(\nabla f(\boldsymbol{y}_0) - \boldsymbol{g})^T \boldsymbol{d}| &= \left|(J(\boldsymbol{y}_0)^T \nabla F(\boldsymbol{r}(\boldsymbol{y}_0)) - J^T \nabla F(\boldsymbol{c}))^T \boldsymbol{d}\right|, \quad (4.57) \\ &\le \|\nabla F(\boldsymbol{r}(\boldsymbol{y}_0))\| \, \|(J(\boldsymbol{y}_0) - J)\boldsymbol{d}\|, \quad (4.58) \\ &\le G_{\max}\sqrt{N} \|(J(\boldsymbol{y}_0) - J)\boldsymbol{d}\|_\infty, \quad (4.59) \\ &\le G_{\max}\sqrt{N}\frac{n\Lambda L_r \beta^2}{2}\Delta, \quad (4.60) \end{aligned}$$

where the last line follows from (2.11b) applied to the component models m_t. Next, we instead fix $\boldsymbol{s} \in C \cap B(\boldsymbol{y}_0, \Delta)$ and compute

$$
\begin{aligned}
&|f(\boldsymbol{y}_0+\boldsymbol{s}) - m(\boldsymbol{y}_0+\boldsymbol{s})| = \left| f(\boldsymbol{y}_0+\boldsymbol{s}) - f(\boldsymbol{y}_0) - \boldsymbol{g}^T\boldsymbol{s} - \frac{1}{2}\boldsymbol{s}^T H \boldsymbol{s}\right|, && (4.61)\\
&\qquad \le \left|f(\boldsymbol{y}_0+\boldsymbol{s}) - f(\boldsymbol{y}_0) - \nabla f(\boldsymbol{y}_0)^T\boldsymbol{s}\right| + \left|(\nabla f(\boldsymbol{y}_0) - \boldsymbol{g})^T\boldsymbol{s}\right| \\
&\qquad\qquad + \frac{1}{2}|\boldsymbol{s}^T J^T \nabla F^2(\boldsymbol{r}(\boldsymbol{y}_0)) J\boldsymbol{s}|, && (4.62)\\
&\qquad \le \frac{L_{\nabla f}}{2}\Delta^2 + \left|(\nabla f(\boldsymbol{y}_0) - \boldsymbol{g})^T\boldsymbol{s}\right| + H_{\max}\|J\boldsymbol{s}\|^2. && (4.63)
\end{aligned}
$$

Now, if $\Delta \ge 1$, we let $\boldsymbol{d} = \Delta^{-1}\boldsymbol{s}$ and so $\|\boldsymbol{d}\| \le 1$. Also, since $\|\boldsymbol{d}\| \le \|\boldsymbol{s}\|$ and $\boldsymbol{y}_0, \boldsymbol{y}_0+\boldsymbol{s} \in C$ we have $\boldsymbol{y}_0 + \boldsymbol{d} \in C$, which gives us

$$
\left|(\nabla f(\boldsymbol{y}_0) - \boldsymbol{g})^T\boldsymbol{s}\right| = \Delta\left|(\nabla f(\boldsymbol{y}_0) - \boldsymbol{g})^T\boldsymbol{d}\right| \le G_{\max}\sqrt{N}\frac{n\Lambda L_r \beta^2}{2}\Delta^2, \tag{4.64}
$$

where the last inequality follows from (4.60), and also

$$
\begin{aligned}
\|J\boldsymbol{s}\| &\le \|(J - J(\boldsymbol{y}_0))\boldsymbol{s}\| + \|J(\boldsymbol{y}_0)\boldsymbol{s}\|, && (4.65)\\
&\le \sqrt{N}\Delta\|(J - J(\boldsymbol{y}_0))\boldsymbol{d}\|_\infty + J_{\max}\Delta, && (4.66)\\
&\le \sqrt{N}\Delta^2\frac{n\Lambda L_r\beta^2}{2} + J_{\max}\Delta, && (4.67)\\
&\le \left(\sqrt{N}\Delta_{\max}\frac{n\Lambda L_r\beta^2}{2} + J_{\max}\right)\Delta, && (4.68)
\end{aligned}
$$

where (4.67) follows from (2.11b) applied to the component models m_t Instead, if $\Delta < 1$, from Lemma 4.3 we have

$$
\left|(\nabla f(\boldsymbol{y}_0) - \boldsymbol{g})^T\boldsymbol{s}\right| \le G_{\max}\sqrt{N}\|(J(\boldsymbol{y}_0) - J)\boldsymbol{s}\|_\infty \le G_{\max}\sqrt{N}\frac{n\Lambda L_r\beta^2}{2}\Delta^2, \tag{4.69}
$$

and

$$
\begin{aligned}
\|J\boldsymbol{s}\| &\le \|(J - J(\boldsymbol{y}_0))\boldsymbol{s}\| + \|J(\boldsymbol{y}_0)\boldsymbol{s}\|, && (4.70)\\
&\le \sqrt{N}\frac{n\Lambda L_r\beta^2}{2}\Delta^2 + J_{\max}\Delta, && (4.71)\\
&\le \left(\sqrt{N}\Delta_{\max}\frac{n\Lambda L_r\beta^2}{2} + J_{\max}\right)\Delta. && (4.72)
\end{aligned}
$$

The result then follows from (4.63) combined with either (4.64) and (4.68), or (4.69) and (4.72). ∎

We also note that the above proof also gives us an effective bound on the model Hessian, in the sense that

$$
\|\boldsymbol{s}^T H \boldsymbol{s}\| \le H_{\max}\left(\sqrt{N}\Delta_{\max}\frac{n\Lambda L_r\beta^2}{2} + J_{\max}\right)^2\Delta^2, \tag{4.73}
$$

for all $\boldsymbol{s} \in C \cap B(\boldsymbol{y}_0, \Delta)$, whenever the interpolation set is Λ-poised.

Algorithm 5.1 Modified FISTA for solving (2.10)

```
Input: Constraint set $C \subset \mathbb{R}^n$, starting point $\boldsymbol{x}_0 \in C$, model gradient $\boldsymbol{g}_0$ and Hessian $H$, and trust-region radius $\Delta$.
 1: Set $\boldsymbol{y}_0 = \boldsymbol{x}_0$, $L = \|H\|$ if $H \neq 0$ (or $L = 1$ otherwise), and $t_0 = 1$.
 2: for $j = 0, 1, 2, \ldots$ do
 3:     Project the gradient step onto $D := C \cap B(\boldsymbol{x}_0, \Delta)$ and set $\boldsymbol{x}_{j+1} = \mathrm{proj}_D\left(\boldsymbol{y}_j - \frac{1}{L}\boldsymbol{g}_j\right)$
 4:     if the stopping conditions are satisfied then
 5:         return the step $(\boldsymbol{x}_{j+1} - \boldsymbol{x}_j)$
 6:     else
 7:         Set $t_{j+1} = (1 + \sqrt{1 + 4t_j^2})/2$ and $\boldsymbol{y}_{j+1} = \boldsymbol{x}_{j+1} + \left(\frac{t_j - 1}{t_{j+1}}\right)(\boldsymbol{x}_{j+1} - \boldsymbol{x}_j)$.
 8:         Calculate the gradient at our new iterate $\boldsymbol{g}_{j+1} = \boldsymbol{g}_j + H(\boldsymbol{y}_{j+1} - \boldsymbol{y}_j)$.
 9:     end if
10: end for
```

5. Numerical Results.

5.1. Implementation. To study the numerical performance of CDFO-TR with composite linear models (specifically for nonlinear least-squares problems), we extend the software package DFO-LS [9] to a new implementation we will call CDFO-LS[6]. CDFO-LS differs from DFO-LS primarily in its calculation of the trust-region step (2.10) in the presence of the constraint set C. It also requires alternative methods for calculating geometry-improving steps (4.47), and the criticality measure (2.12), however these may be viewed as special cases of (2.10) with a linear objective. In all cases, this problem is convex (as the model Hessian in (4.54) has the form $J^T J$).[7] To efficiently solve these problems, we use an implementation of FISTA [5, Chapter 10.7] as given in Algorithm 5.1. We terminate FISTA when either $\|\tilde{\boldsymbol{x}}_{j+1} - \tilde{\boldsymbol{x}}_j\| \leq 10^{-12}$ or after $100n^2$ iterations, whichever is achieved first. Projection onto the intersection of C and a closed ball was performed using Dykstra's algorithm [20, 8] using the stopping criterion proposed in [7] with tolerance 10^{-10}.

The other key difference between CDFO-LS and DFO-LS is the construction of an initial interpolation set. Our implementation uses Algorithm 4.1, but where the sequence $\boldsymbol{d}_k$ is constructed by first trying $\pm\boldsymbol{e}_t$ for $t = 1, \ldots, n$, and then generating the vectors $\boldsymbol{d}_k$ i.i.d. from a uniform distribution on the unit sphere. DFO-LS only uses the choices $\pm\boldsymbol{e}_t$, since it only allows the use of bound constraints.

Implementation for box constraints. The original implementation of DFO-LS can handle box constraints via a tailored trust-region subproblem solver [34]. For such problems we tested CDFO-LS (using projections) against the original DFO-LS, and found that the original implementation was slightly better. Hence when only box constraints are present, CDFO-LS uses the original DFO-LS implementation.

5.2. Solvers tested. The numerical performance of CDFO-LS was measured against the following solvers:

- COBYLA [33], a DFO solver for handling constrained general objectives based on a linear interpolation model, incorporated in the SciPy package;

[6] Version 1.3, available from https://github.com/numericalalgorithmsgroup/dfols
[7] For geometry-improving steps (4.47), we separately maximize $\pm\ell_t(\boldsymbol{y})$.

and

- pyNOMAD, the Python interface to NOMAD [27], an instance of Mesh Adaptive Direct Search (MADS) for black-box optimization of constrained general objectives.

Our tests on CDFO-LS used the default parameters. For all solvers, we used an initial trust-region radius of $\rho_0 = \Delta_0 = 0.1 \max(\|\boldsymbol{x}_0\|_\infty, 1)$. In performing tests with NOMAD, constraints were introduced as *extreme barrier* constraints to prevent constraint violations. As NOMAD requires a feasible initial point, in this case we used $\mathrm{proj}_C(\boldsymbol{x}_0)$.

5.3. Test problems and methodology. The three solvers were tested on 5 problems[8] chosen out of the test suite from Moré, Garbow, and Hillstrom [29], and the entire test suite of 53 problems from Moré and Wild [30]. All objectives were nonlinear least-squares functions with dimension $2 \le n \le 12$ and $2 \le m \le 65$. For each problem, there were four types of constraints that were individually tested:

- Unconstrained: $\boldsymbol{x} \in \mathbb{R}^n$;
- Box: $\boldsymbol{x}$ must satisfy (element-wise) $\frac{1}{10} \le \boldsymbol{x} \le 20$;
- Ball: $\boldsymbol{x} \in B(\boldsymbol{x}_c, 6.9)$, $\boldsymbol{x}_c = (5, 5, \ldots, 5) \in \mathbb{R}^n$; and
- Halfspace: we require $\boldsymbol{x} \in \{\boldsymbol{y} : \mathbf{1}^T y \le 1\}$;

where $\boldsymbol{x} \in \mathbb{R}^n$ is some point at which we evaluate the objective, and $\mathbf{1}$ is the vector of all ones. This means that in total there were $4 \times (53 + 5) = 232$ problems tested.

We additionally performed tests where stochastic noise was introduced when evaluating the residuals r_i. The two noise models implemented were as follows:

- Multiplicative Gaussian noise: we evaluate residual $\tilde{r}_i(\boldsymbol{x}) = r_i(\boldsymbol{x})(1 + \epsilon)$; and
- Additive Gaussian noise: we evaluate residual $\tilde{r}_i(\boldsymbol{x}) = r_i(\boldsymbol{x}) + \epsilon$;

where $\epsilon \sim N(0, \sigma^2)$ i.i.d. for each i and $\boldsymbol{x}$.

Prior to making comparisons, for each solver $\mathcal{S}$, every problem p, and an accuracy level $\tau \in (0, 1)$, we determined the number of function evaluations $N_p(\mathcal{S}; \tau)$ required for a problem to be considered solved:

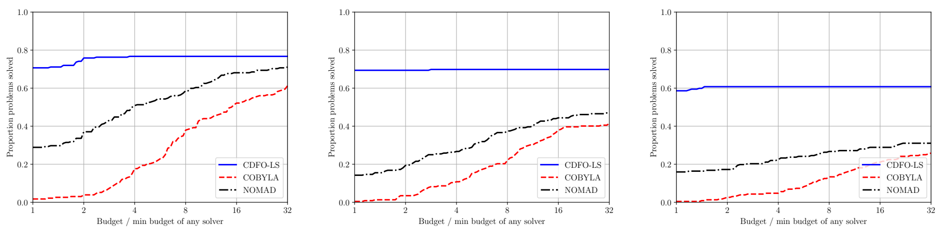
$$N_p(\mathcal{S}; \tau) := \#\text{ objective evals before } \; f(\boldsymbol{x}_k) \le \mathbb{E}\left[f^* + \tau\left(f(\boldsymbol{x}_0) - f^*\right)\right], \tag{5.1}$$

where f^* is the approximate true minimum $f(\boldsymbol{x}^*)$ of the corresponding problem. If the problem is unconstrained, we source the true minimum from [29] and [30]. For all other cases, f^* is taken to be the smallest objective value obtained by any of the three solvers. An important consideration in our comparisons is that the COBYLA algorithm allows constraints to be violated in search of a local minimizer. To keep our comparisons as fair as possible, any iteration for which COBYLA evaluated an infeasible point $\boldsymbol{x}_k$ we recorded as $f(\boldsymbol{x}_k) = \infty$. Moreover, if any solver is not able to obtain the true minimum for a corresponding p and τ in the maximum allowed computational budget, then we set $N_p(\mathcal{S}, \tau) = \infty$.

To compare solvers, we use *performance profiles* [19], where we plot the number of objective evaluations $N_p(\mathcal{S}; \tau)$ normalized by the minimum objective evaluations needed by any solver $N_p^*(\tau)$:

$$\pi_{\mathcal{S},\tau}(\alpha) := \frac{\left|\{p \in \mathcal{P} : N_p(\mathcal{S}; \tau) \le \alpha N_p^*(\tau)\}\right|}{|\mathcal{P}|}, \quad \alpha \ge 1. \tag{5.2}$$

[8]The names of the five problems chosen from [29] are *Biggs Exp6*, *Dixon*, *Gulf research and development*, *Powell badly scaled*, and *Wood*.



(a) $\tau = 10^{-1}$ (b) $\tau = 10^{-3}$ (c) $\tau = 10^{-5}$

Fig. 3. Performance profiles comparing CDFO-LS against NOMAD and COBYLA on a collection of noiseless, constrained nonlinear least-squares problems.

5.4. Test results. In all tests, we generated performance profiles for accuracy levels $\tau = 10^{-1}, \tau = 10^{-3}$ and $\tau = 10^{-5}$; in our noise models we used noise level $\sigma = 10^{-2}$. When testing CDFO-LS in the noisy case, we specified through a flag that the objective was noisy. With the flag set, CDFO-LS uses a different value of γ_{dec} and changes the values of other parameters to better suit a noisy objective (see [9] for details).

Figure 3 shows the performance profiles in the case that the objective is noiseless. We see in all accuracy settings, that CDFO-LS is able to solve a much greater proportion of problems in comparison to both COBYLA and NOMAD in a limited number of evaluations. This is to be expected, as it is able to explicitly exploit the least-squares problem structure, and has more information about the constraint set than NOMAD (a projection operator rather than a simple feasible/infeasible flag). At the low accuracy level, as the number of evaluations increase, NOMAD manages to solve a similar proportion of problems to CDFO-LS; COBYLA performs worse, but is not far behind NOMAD.

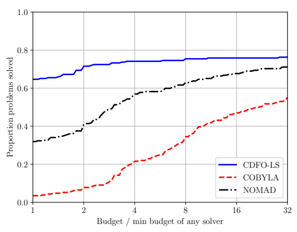
As the accuracy level increases, the proportion of problems solved decreases across all solvers. However, the gap between the proportion of problems solved by CDFO-LS and NOMAD increases in favour of CDFO-LS. NOMAD performs more similarly to COBYLA at higher accuracy. Neither NOMAD nor COBYLA are able to solve many problems in comparison to CDFO-LS when $\tau = 10^{-5}$.

In the case of noisy objectives, shown in Figures 4 and 5, the results are similar. However, in comparison to CDFO-LS, the performance of NOMAD and COBYLA is not as affected. Still, CDFO-LS significantly outperforms all other solvers, particularly at lower performance ratios α.

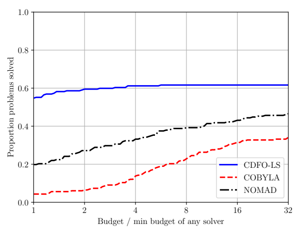
6. Conclusions and Future Work.

Current model-based DFO approaches for handling unrelaxable constraints consider only simple bound ([34, 24] and [36, Section 6.3]) or linear inequality constraints [25], or requires strong model accuracy assumptions that are difficult to satisfy using only feasible points [13].

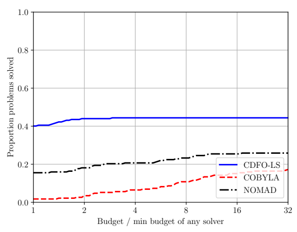
We extend these approaches to handle a significantly broader class of constraints by assuming a convex constraint set accessed only via projections, while providing theoretical guarantees of convergence. A worst-case complexity analysis gives a bound of $\mathcal{O}(\epsilon^{-2})$ iterations to reach ϵ first-order optimality in the case of linear and composite linear interpolation models; the same result as in the unconstrained setting [21]. To achieve this, we introduce a weaker definition of a fully linear model (Definition 2.3) adapted from [17] for convex constraints. We also generalize the concept of Λ-



(a) $\tau = 10^{-1}$

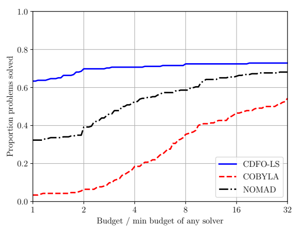


(b) $\tau = 10^{-3}$

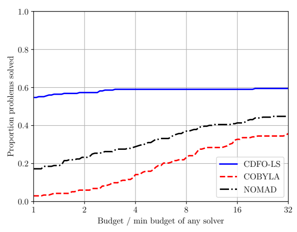


(c) $\tau = 10^{-5}$

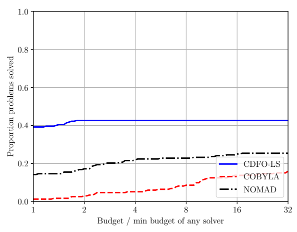
Fig. 4. Performance profiles comparing CDFO-LS against NOMAD and COBYLA on a collection of constrained nonlinear least-squares problems in the presence of multiplicative noise.



(a) $\tau = 10^{-1}$



(b) $\tau = 10^{-3}$



(c) $\tau = 10^{-5}$

Fig. 5. Performance profiles comparing CDFO-LS against NOMAD and COBYLA on a collection of constrained nonlinear least-squares problems in the presence of additive noise.

poisedness from [16] to the constrained setting, which importantly allows Λ-poised interpolation sets to be constructed using only feasible points. We demonstrate how to verify and construct Λ-poised sets for linear and composite linear interpolation.

Numerically, we implement our method by extending DFO-LS [9] for nonlinear least-squares problems, a setting where linear interpolation models are sufficient to achieve have strong performance. Tests on low-dimensional test problems with a variety of convex constraints show that our implementation outperforms the general-purpose solvers COBYLA [33] and NOMAD [27]. We do note that neither solver is able to exploit the least-squares problem structure, and that COBYLA requires the functional form of the constraint sets, and NOMAD handles constraints via an extreme barrier approach.

A natural extension to our work here is to develop convergence theory and complexity analysis for quadratic interpolation models, which would enable a practical implementation of our method for general nonconvex minimization.

REFERENCES

[1] M. A. Abramson, C. Audet, J. E. Dennis, and S. L. Digabel, *OrthoMADS: A deterministic MADS instance with orthogonal directions*, SIAM Journal on Optimization, 20 (2009), pp. 948–966.

[2] C. Audet and J. E. Dennis, *Mesh adaptive direct search algorithms for constrained*

optimization, SIAM Journal on Optimization, 17 (2006), pp. 188–217.

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

[4] C. Audet, S. Le Digabel, and C. Tribes, *The mesh adaptive direct search algorithm for granular and discrete variables*, SIAM Journal on Optimization, 29 (2019), pp. 1164–1189.

[5] A. Beck, *First-Order Methods in Optimization*, MOS-SIAM Series on Optimization, MOS/SIAM, Philadelphia, 2017.

[6] S. Bellavia, G. Gurioli, B. Morini, and P. L. Toint, *Adaptive regularization algorithms with inexact evaluations for nonconvex optimization*, SIAM Journal on Optimization, 29 (2019), pp. 2881–2915.

[7] E. G. Birgin and M. Raydan, *Robust stopping criteria for dykstra's algorithm*, SIAM Journal on Scientific Computing, 26 (2005), pp. 1405–1414.

[8] J. P. Boyle and R. L. Dykstra, *A method for finding projections onto the intersection of convex sets in hilbert spaces*, in Advances in Order Restricted Statistical Inference, R. Dykstra, T. Robertson, and F. T. Wright, eds., New York, NY, 1986, Springer New York, pp. 28–47.

[9] C. Cartis, J. Fiala, B. Marteau, and L. Roberts, *Improving the flexibility and robustness of model-based derivative-free optimization solvers*, ACM Transactions on Mathematical Software, 45 (2019), pp. 1–41.

[10] C. Cartis, N. I. M. Gould, and P. L. Toint, *An adaptive cubic regularization algorithm for nonconvex optimization with convex constraints and its function-evaluation complexity*, IMA Journal of Numerical Analysis, 32 (2012), pp. 1662–1695.

[11] C. Cartis and L. Roberts, *A derivative-free Gauss-Newton method*, Mathematical Programming Computation, 11 (2019), pp. 631–674.

[12] C. Cartis and L. Roberts, *Scalable subspace methods for derivative-free nonlinear least-squares optimization*, arXiv preprint arXiv:2102.12016, (2021).

[13] P. Conejo, E. Karas, L. Pedroso, A. Ribeiro, and M. Sachine, *Global convergence of trust-region algorithms for convex constrained minimization without derivatives*, Applied Mathematics and Computation, 220 (2013), pp. 324–330.

[14] A. R. Conn, N. Gould, A. Sartenaer, and P. L. Toint, *Global convergence of a class of trust region algorithms for optimization using inexact projections on convex constraints*, SIAM Journal on Optimization, 3 (1993), pp. 164–221.

[15] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust-Region Methods*, vol. 1 of MPS-SIAM Series on Optimization, MPS/SIAM, Philadelphia, 2000.

[16] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Geometry of interpolation sets in derivative free optimization*, Mathematical Programming, 111 (2007), pp. 141–172.

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

[18] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*, vol. 8 of MPS-SIAM Series on Optimization, MPS/SIAM, Philadelphia, 2009.

[19] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91 (2002), pp. 201–213.

[20] R. L. Dykstra, *An algorithm for restricted least squares regression*, Journal of the American Statistical Association, 78 (1983), pp. 837–842.

[21] R. Garmanjani, D. Júdice, and L. N. Vicente, *Trust-region methods without using derivatives: Worst case complexity and the nonsmooth case*, SIAM Journal on Optimization, 26 (2016), pp. 1987–2011.

[22] G. N. Grapiglia, J. Yuan, and Y.-x. Yuan, *A derivative-free trust-region algorithm for composite nonsmooth optimization*, Computational and Applied Mathematics, 35 (2016), pp. 475–499.

[23] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang, *Direct search based on probabilistic feasible descent for bound and linearly constrained problems*, Computational Optimization and Applications, 72 (2019), pp. 525–559.

[24] S. Gratton, P. L. Toint, and A. Tröltzsch, *An active-set trust-region method for derivative-free nonlinear bound-constrained optimization*, Optimization Methods and Software, 26 (2011), pp. 873–894.

[25] E. A. E. Gumma, M. H. A. Hashim, and M. M. Ali, *A derivative-free algorithm for linearly constrained optimization problems*, Computational Optimization and Applications, 57 (2014), pp. 599–621.

[26] J. Larson, M. Menickelly, and S. M. Wild, *Derivative-free optimization methods*, Acta

Numerica, 28 (2019), pp. 287–404.

[27] S. Le Digabel, *Algorithm 909: NOMAD: Nonlinear optimization with the mads algorithm*, ACM Trans. Math. Softw., 37 (2011).

[28] S. Le Digabel and S. M. Wild, *A taxonomy of constraints in simulation-based optimization*, arXiv preprint arXiv:1505.07881, (2015).

[29] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *Testing unconstrained optimization software*, ACM Transactions on Mathematical Software, 7 (1981), pp. 17–41.

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

[31] B. Morini, M. Porcelli, and P. L. Toint, *Approximate norm descent methods for constrained nonlinear systems*, Mathematics of Computation, 87 (2017), pp. 1327–1351.

[32] M. Porcelli, *On the convergence of an inexact Gauss-Newton trust-region method for nonlinear least-squares problems with simple bounds*, Optimization Letters, 7 (2013), pp. 447–465.

[33] M. J. D. Powell, *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, Springer Netherlands, Dordrecht, 1994, pp. 51–67.

[34] M. J. D. Powell, *The BOBYQA algorithm for bound constrained optimization without derivatives*, Tech. Report DAMTP 2009/NA06, University of Cambridge, 2009.

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

[36] S. M. Wild, *Derivative-Free Optimization Algorithms for Computationally Expensive Functions*, PhD thesis, Cornell University, 2009.

[37] H. Zhang, A. R. Conn, and K. Scheinberg, *A derivative-free algorithm for least-squares minimization*, SIAM Journal on Optimization, 20 (2010), pp. 3555–3576.