Exploiting Prior Function Evaluations in Derivative-Free Optimization

FRANK E. CURTIS*, SHIMA DEZFULIAN†, AND ANDREAS WÄCHTER.†

Abstract. A derivative-free optimization (DFO) algorithm is presented. The distinguishing feature of the algorithm is that it allows for the use of function values that have been made available through prior runs of a DFO algorithm for solving prior related optimization problems. Applications in which sequences of related optimization problems are solved such that the proposed algorithm is applicable include certain least-squares and simulation-based optimization problems. A convergence guarantee of a generic algorithmic framework that exploits prior function evaluations is presented, then a particular instance of the framework is proposed and analyzed. The results of numerical experiments when solving engineering test problems show that the algorithm gains advantage over a state-of-the-art DFO algorithm when prior function values are available.

Key words. derivative-free optimization, trust-region methods, least squares

AMS subject classifications. 49M15, 65K05, 65K10, 90C30, 90C56

1. Introduction.

We propose an algorithmic framework for solving optimization problems that have two distinct characteristics. First, we are interested in problems that arise in contexts when sequences of optimization problems are to be solved when the data defining the objective functions are similar from one problem to the next. Second, we assume that these problems involve objective functions that are smooth (i.e., continuously differentiable), yet derivatives are intractable to compute. Problems with one or the other of these characteristics have been considered previously in the continuous optimization literature, but, to the best of our knowledge, no algorithms have been proposed specifically for the combined setting. The key feature of our proposed framework is that it exploits, in a derivative-free optimization (DFO) context, function evaluations that have already been performed during the solution of prior optimization problems when new problems in the sequence are considered.

Sequences of related optimization problems need to be solved in various applications throughout science, engineering, and economics. Here, we mention two settings that have motivated this work. First, nonlinear least-square problems that commonly appear in data-fitting applications fall into our setting of interest. For such problems [3, 15], one aims to find the parameters in a function—which may be a *black-box* function, as in our setting—that accurately maps inputs to outputs in a set of observed data. The second setting considers multi-output simulation optimization problems, where one aims to minimize a cost associated with the outputs of a simulation; see, e.g., [3, 17]. The black-box functions that arise in these contexts can, for example, be those defined by numerical simulations, say that solve complex partial differential equations (PDEs) or perform Monte Carlo simulations of complex systems [37].

Our proposed algorithm builds upon state-of-the-art model-based DFO methods for solving problems that possess a composite structure of the objective. In particular, inspired by [5, 34, 37], our algorithm computes search directions by minimizing local models of the objective that have been constructed using previously computed function values, where the composite structure of the objective is exploited by combining local models that have been constructed for black-box functions separately. It has been shown, e.g., in the context of nonlinear least squares, that algorithms that

*Department of ISE, Lehigh U., Bethlehem, PA 18015 (frank.e.curtis@lehigh.edu).

†Department of IEMS, Northwestern U., Evanston, IL 60208 (dezfulian@u.northwestern.edu, andreas.waechter@northwestern.edu). Supported in part by NSF grant DMS-2012410.

exploit such structure often perform better than those that do not; see, e.g., [5, 34, 37].

1.1. Contributions. We propose and analyze a model-based DFO algorithm that exploits function evaluations that have been performed when solving related optimization problems. This allows the algorithm to avoid new evaluations of black-box functions at interpolation points when the function values at these points can be approximated sufficiently accurately using prior evaluations, e.g., by an application-specific surrogate model. Our algorithm also differs from other model-based DFO approaches in that it exploits prior information when determining the set of interpolation points itself. This feature is incorporated through the use of a specially designed utility function that predicts the usefulness of a potential interpolation point. Our numerical experiments show that our algorithm often outperforms a state-of-the-art model-based DFO method in our setting of interest, especially when the budget for function evaluations is small, which is typically the case in real-world applications.

1.2. Notation. The sets of real numbers, n-dimensional real vectors, and n-by-m-dimensional real matrices are denoted by $\mathbb{R}$, $\mathbb{R}^n$, and $\mathbb{R}^{n\times m}$, respectively. The sets of nonnegative and positive real numbers are denoted by $\mathbb{R}_{\geq 0}$ and $\mathbb{R}_{>0}$, respectively. The vector of all zeros is denoted as $\mathbf{0}$, the identity matrix is denoted as I, the vector of all ones is denoted as $\mathbf{1}$, and the ith unit vector is denoted as $\mathbf{e}_i$, where in each case the size of the object is determined by the context. The set of nonnegative integers is denoted as $\mathbb{N}$ and we define $[n] := \{1, \ldots, n\}$ for any $n \in \mathbb{N} \setminus \{0\}$.

For sets $\mathcal{S}_1 \subseteq \mathbb{R}^n$ and $\mathcal{S}_2 \subseteq \mathbb{R}^n$, we define Minkowski addition and subtraction in the usual manner, e.g., $\mathcal{S}_1 + \mathcal{S}_2 = \{s_1 + s_2 : s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2\}$. That said, in the particular case when one of the sets is a singleton, say $\{c\}$ with $c \in \mathbb{R}^n$, then we simply write $c + \mathcal{S} = \{c + s : s \in \mathcal{S}\}$. The cardinality of a set $\mathcal{S}$ is denoted by $|\mathcal{S}|$.

Given any real number $q \geq 1$, the ℓ_q-norm of a vector $v \in \mathbb{R}^n$ is written as $\|v\|_q$. The closed ℓ_q-norm ball with center $x \in \mathbb{R}^n$ and radius $\Delta \in \mathbb{R}_{\geq 0}$ is denoted as $B_q(x, \Delta) := \{\bar{x} : \|\bar{x} - x\|_q \leq \Delta\}$. The dual norm of $\|\cdot\|_q$ is denoted and defined by $\|z\|_{q^*} := \max\{z^T x : x \in B_q(0, 1)\}$. The Frobenius-norm of a matrix M is denoted by $\|M\|_{\mathrm{F}}$. For a matrix $M \in \mathbb{R}^{m\times n}$ and real numbers $p \geq 1$ and $q \geq 1$, the $L_{p,q}$-norm of M is written as $\|M\|_{p,q} = \big(\sum_{j=1}^n (\sum_{i=1}^m |M_{ij}|^p)^{\frac{q}{p}}\big)^{\frac{1}{q}}$ and the (p, q)-operator norm of M is written as $\|M\|_{(p,q)} = \sup_{\|x\|_p \leq 1} \|Mx\|_q$, although, for shorthand, $\|\cdot\|_p = \|\cdot\|_{(p,p)}$.

For the sake of generality, two norms used throughout the paper are those defining the *trust region radius* and the *approximation radius* in our algorithm, which we respectively denote as $\|\cdot\|_{\mathrm{tr}}$ and $\|\cdot\|_{\mathrm{app}}$ for some real numbers $\mathrm{tr} \geq 1$ and $\mathrm{app} \geq 1$. In our analysis, we make use of norm equivalence constants for finite-dimensional real vector spaces; in particular, due to the equivalence of all such norms, for any positive integers n_x and p, there exist constants $(\kappa_{\mathrm{tr}_0}, \kappa_{\mathrm{tr}_1}, \kappa_{\mathrm{tr}_0^*}, \kappa_{\mathrm{tr}_1^*}, \kappa_{\mathrm{tr}_2^*}, \kappa_{\mathrm{tr}_3^*}, \kappa_{\mathrm{app}_0^*}) \in \mathbb{R}^7_{\geq 0}$ such that

$$\|v\|_2 \leq \kappa_{\mathrm{tr}_0}\|v\|_{\mathrm{tr}} \quad \text{for all } v \in \mathbb{R}^{n_x}, \tag{1.1a}$$

$$\|v\|_{\mathrm{tr}} \leq \kappa_{\mathrm{tr}_1}\|v\|_2 \quad \text{for all } v \in \mathbb{R}^{n_x}, \tag{1.1b}$$

$$\|v\|_{\mathrm{tr}^*} \leq \kappa_{\mathrm{tr}_0^*}\|V^{-1}\|_2\|Vv\|_\infty \quad \text{for all } (v, V) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x\times n_x}, \tag{1.1c}$$

$$\|Vv\|_{\mathrm{tr}^*} \leq \kappa_{\mathrm{tr}_1^*}\|V\|_{\mathrm{tr}^*,1}\|v\|_2 \quad \text{for all } (v, V) \in \mathbb{R}^p \times \mathbb{R}^{n_x\times p}, \tag{1.1d}$$

$$\|VUV^T\|_{(\mathrm{tr},\mathrm{tr}^*)} \leq \kappa_{\mathrm{tr}_2^*}\|V\|^2_{\mathrm{tr}^*,1}\|U\|_2 \quad \text{for all } (V, U) \in \mathbb{R}^{n_x\times p} \times \mathbb{R}^{p\times p}, \tag{1.1e}$$

$$\|v\|_{\mathrm{tr}}\|v\|_{\mathrm{tr}^*} \leq \kappa_{\mathrm{tr}_3^*}\|v\|_2^2 \quad \text{for all } v \in \mathbb{R}^{n_x}, \tag{1.1f}$$

$$\text{and } \|Vv\|_{\mathrm{app}^*} \leq \kappa_{\mathrm{app}_0^*}\|v\|_{\mathrm{app}}\|V\|_{(2,2)} \quad \text{for all } (v, V) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x\times n_x}. \tag{1.1g}$$

For differentiable $g : \mathbb{R}^n \to \mathbb{R}$ and $y \in \mathbb{R}^n$, we use $\partial_i g(y)$ and $\partial_i \partial_j g(y)$ to denote $\frac{\partial g(y)}{\partial y_i}$ and $\frac{\partial^2 g(y)}{\partial y_i \partial y_j}$ for any $(i,j) \in [n] \times [n]$, respectively. The gradient function of g is $\nabla g : \mathbb{R}^n \to \mathbb{R}^n$, where $[\nabla g(y)]_i = \partial_i g(y)$ for any $i \in [n]$ and $y \in \mathbb{R}^n$. If g is twice differentiable, then the Hessian function of g is $\nabla^2 g : \mathbb{R}^n \to \mathbb{R}^{n \times n}$, where $[\nabla^2 g(y)]_{i,j} = \partial_i \partial_j g(y)$ for any $(i,j) \in [n] \times [n]$ and $y \in \mathbb{R}^n$. For differentiable vector-valued $G : \mathbb{R}^n \to \mathbb{R}^m$, the transpose of the Jacobian function of G is denoted by $\nabla G : \mathbb{R}^n \to \mathbb{R}^{n \times m}$, where $[\nabla G(y)]_{i,j} = \partial_i G_j(y)$ for any $(i,j) \in [n] \times [m]$ and $y \in \mathbb{R}^n$.

Given any $M \in \mathbb{R}^{m \times n}$, its column space is $\text{span}(M)$, its null space is $\text{null}(M)$, and given any $v \in \mathbb{R}^m$, the projection of v onto $\text{span}(M)$ is $\text{proj}_M(v)$.

One of the main quantities in the paper is a function F that takes two input arguments. For the purposes of designing our optimization algorithm, it is convenient to write the domain of the function as $\mathbb{R}^{n_x + n_\theta}$, although for ease of exposition we write the function and its gradient as taking inputs in $\mathbb{R}^{n_x} \times \mathbb{R}^{n_\theta}$; in particular, these are written in the form $F(x;\theta)$ and $\nabla F(x;\theta)$. This should not lead to confusion due to the natural one-to-one mapping between elements of $\mathbb{R}^{n_x} \times \mathbb{R}^{n_\theta}$ and $\mathbb{R}^{n_x + n_\theta}$.

1.3. Problem Formulation. Consider a sequence of optimization problems, namely, for all $t \in \mathbb{N}$, consider the minimization problem

$$\min_{x \in \Omega_t} f_t(x) \quad \text{with} \quad f_t(x) \coloneqq h_t(F(x, \theta_t)), \tag{1.2}$$

where the objective parameter satisfies $\theta_t \in \Theta$ for some $\Theta \subseteq \mathbb{R}^{n_\theta}$ (independent of t), the function $h_t : \mathbb{R}^p \to \mathbb{R}$ is *glass-box* in the sense that its analytical form is known, and $x \in \Omega_t$ represents a set of (finite or infinite) bound constraints on the elements of x, which are allowed to be relaxable or unrelaxable.

Here, constraints are said to be relaxable (respectively, unrelaxable) if they do not (respectively, do) need to be satisfied for one to be able to evaluate the objective function. As an example of a set of unrelaxable constraints, consider the case when $F(x, \theta_t)$ is the output of a simulation that fails to produce a meaningful output unless $x \in \Omega_t$ [16]. Along with our proposed DFO framework, we provide techniques for handling unrelaxable constraints. We also conjecture that our proposed techniques can be extended to the setting when Ω_t is any closed convex set, such as by following the ideas in [20]. However, since bound constraints are sufficient for our numerical experimentation, we consider bound constraints only.

As for the function $F : \mathbb{R}^{n_x} \times \Theta \to \mathbb{R}^p$, it is *black-box* in the sense that its derivatives cannot be evaluated. We write

$$F(x, \theta_t) = \begin{bmatrix} F_1(x, \theta_t) & \cdots & F_p(x, \theta_t) \end{bmatrix}^T, \tag{1.3}$$

where F_i for each $i \in [p]$ is the ith *element function* of the vector-valued function F.

Our presumption is that evaluations of F are expensive and, since one is solving a sequence of problems over $t \in \mathbb{N}$, it may be beneficial to store and make use of function values obtained from previous runs of an algorithm when solving subsequent problems. That is, for all $t > 0$, we presume that a history of prior black-box function values is available *a priori*. We denote the set of prior information for element function i when solving problem t (obtained when solving problems 0 through $t-1$ and during prior iterations when solving problem t) by $\mathcal{H}_{i,t}$, which is a finite set and is defined below. In addition, we use $\tilde{F}_i(x, \theta_t, \mathcal{H}_{i,t}, \delta)$ to denote an approximation of $F_i(x, \theta_t)$ given $\mathcal{H}_{i,t}$ and an approximation radius $\delta \in \mathbb{R}_{>0}$, which in turn depends on a trust region radius $\Delta \in \mathbb{R}_{>0}$. The approximation radius specifies which elements in $\mathcal{H}_{i,t}$

are *near* (x, θ_t) and hence might be useful for approximating $F_i(x, \theta_t)$. (We formalize the notion of a point being sufficiently near (x, θ_t) in Section 3.) In particular, we shall ensure that the approximation error is bounded by a value proportional to the approximation radius δ in the sense that, for some fixed $\kappa_{\text{app}} \in \mathbb{R}_{\geq 0}$,

$$|F_i(x, \theta_t) - \tilde{F}_i(x, \theta_t, \mathcal{H}_{i,t}, \delta)| \leq \kappa_{\text{app}} \delta. \tag{1.4}$$

Two particular instances of problem (1.2) are sequences of least-squares optimization problems and multi-output simulation problems. For concreteness, let us now present how the information sets $\{\mathcal{H}_{i,t}\}$ can be defined for these particular cases.

Least-Squares Optimization Problems. Consider a sequence of sets of observations, say, $\{\{(w_{i,t}, y_{i,t})\}_{i \in [p]}\}_{t \in \mathbb{N}}$, where for all $(i, t) \in [p] \times \mathbb{N}$ one has $(w_{i,t}, y_{i,t}) \in \mathcal{W} \times \mathbb{R}$ for some $\mathcal{W} \subseteq \mathbb{R}^{n_w}$. The goal of problem $t \in \mathbb{N}$ is to find a vector $x \in \mathbb{R}^{n_x}$ such that the black-box function $\phi : \Omega_t \times \mathcal{W} \to \mathbb{R}$ best describes the observations by solving

$$\min_{x \in \Omega_t} \ \frac{1}{2} \sum_{i=1}^{p} \left(\phi(x, w_{i,t}) - y_{i,t}\right)^2, \quad \text{where } \Omega_t \text{ represents element-wise bounds.} \tag{1.5}$$

One can formulate this problem as an instance of problem (1.2) by defining

$$\theta_t \coloneqq \begin{bmatrix} w_{1,t}^T & w_{2,t}^T & \dots & w_{p,t}^T \end{bmatrix}^T, \qquad F_i(x, \theta_t) \coloneqq \phi(x, w_{i,t}) \text{ for all } i \in [p],$$
$$y_t \coloneqq \begin{bmatrix} y_{1,t} & \dots & y_{p,t} \end{bmatrix}^T, \qquad \text{and} \quad h_t(v) \coloneqq \tfrac{1}{2}\|v - y_t\|^2 \text{ for all } v \in \mathbb{R}^p.$$

The prior information that may be relevant for the ith component of F contains all tuples of the form $(x, w, \phi(x, w))$ where $\phi(x, w)$ has previously been evaluated, i.e.,

$$\mathcal{H}_{i,t} = \{(x, w, \phi(x, w)) : \phi(x, w) \text{ has been evaluated}\}. \tag{1.6}$$

Notice that since $F_i(\cdot, \theta_t) \equiv \phi(\cdot, w_{i,t})$ for all $(i, t) \in [p] \times \mathbb{N}$, this means that one may approximate the value of *any* component of F by exploiting *all* prior evaluations of ϕ, even those obtained for different components. In other words, the history is the same for all components of F in the sense that $\mathcal{H}_{i,t} = \mathcal{H}_{j,t}$ for all $(i, j) \in [p] \times [p]$.

Multi-Output Simulation Optimization Problems. Consider now a setting where, for each $t \in \mathbb{N}$, one aims to solve (1.2) when the evaluation of $F(x, \theta_t)$ corresponds to the run of a simulation that produces $p \geq 1$ output values, i.e., $F(x, \theta_t)$ is as defined in (1.3). In such cases, the prior information for the ith component of F is

$$\mathcal{H}_{i,t} = \{(x, \theta, F_i(x, \theta)) : F_i(x, \theta) \text{ has been evaluated}\}. \tag{1.7}$$

1.4. Literature Review. Generally speaking, DFO algorithms can be categorized as direct-search, finite-difference, or model-based methods. Direct-search methods, such as pattern search methods [1, 18] and the Nelder-Mead method [23], are often outperformed by model-based methods [22] when one is minimizing a smooth objective. That said, one of the strengths of direct-search methods is that they are more readily applicable when an objective function is nonsmooth [2, 14] or even discontinuous [31]. Finite-difference approaches approximate derivatives using finite difference schemes, which are then embedded within a gradient-based optimization approach, such as a steepest descent or quasi-Newton method; see, e.g., [30]. Empirical evidence has shown that finite-difference methods can be competitive with model-based methods, at least when one presumes no noise in the function values. The method that we

propose in this paper falls into the model-based method category; hence, we provide a more comprehensive overview of them in the remainder of this section.

Model-based trust-region methods for unconstrained optimization have received a lot of attention in the literature; see, e.g., [10, 11, 13, 21, 25, 26, 27, 33]. Such methods operate by constructing in each iteration a local multivariate model of the objective function. Typically, linear or quadratic interpolation models are used. The resulting model is minimized within a trust region to compute a search direction, then common trust-region strategies for updating the iterate and trust region radius are employed. Example theoretical results on the convergence of model-based methods to first- or second-order stationary points can be found in [9, 13]. In order to guarantee convergence, the interpolation points that are used for building the models need to satisfy a geometry condition referred to as *well-poisedness*. See [10, 11, 13, 21, 33] for further discussion. Other types of models can be employed as well while leading to convergence guarantees. For example, interpolating radial basis function models have been used in ORBIT [35, 36], a model-based trust-region algorithm with theoretical convergence guarantees. Other types of local models such as linear or quadratic *regression* models can be used in place of interpolation models; see [12].

Model-based algorithmic ideas have been extended to solve constrained optimization problems as well. In [7], a high-level discussion on how to handle various types of constraints is provided. BOBYQA [28] is designed to solve bound-constrained problems that are unrelaxable. COBYLA [24] can solve inequality constrained problems by constructing linear interpolation models for the objective and constraint functions using points that lie on a simplex. In [6] and [20], DFO methods for solving problems with closed convex constraints are studied, where it is assumed that projections onto the feasible region are tractable. The proposed algorithms in [6] and [20] have global convergence guarantees. CONORBIT [29] is an extension of ORBIT [35] that handles relaxable inequality constraints and unrelaxable bound constraints.

One of the motivating settings for our proposed algorithm is least-squares optimization. Hence, we mention that for the special case of such problems, tailored algorithms that take into account the structure of the problem have been proposed. For example, DFO-LS [4] and DFO-GN [5] suggest constructing separate linear models for the residuals, then exploiting a Gauss-Newton approach in order to construct a quadratic model for the least-squares objective function. DFO-LS [4] can handle bound constraints as well. Constructing separate quadratic interpolation models for the residuals and using Taylor approximation to construct a quadratic model for the least-squares objective has been considered in POUNDERS [34] and DFLS [37].

Most relevant for this paper is the fact that all of the aforementioned methods do not consider how to exploit the use of function values obtained when previously solving related optimization problems. This is the distinguishing feature of our algorithm.

1.5. Organization. A general DFO algorithmic framework that exploits prior function evaluations for solving (1.2) is presented in Section 2. In Section 3, we provide an approximation scheme that uses prior function value information such that one obtains an implementable instance of the framework. In Section 4, ideas are presented for obtaining well-poised interpolation sets. The results of numerical experiments are presented in Section 5 and concluding remarks are provided in Section 6.

2. A Derivative-Free Algorithmic Framework. A general model-based algorithmic framework for solving problem (1.2) that exploits prior information in order to build the models is presented in this section. The framework that we propose is designed to solve (1.2) for any $t \in \mathbb{N}$. Hence, hereafter, we simplify notation by dropping

the index t. In particular, presuming that $t \geq 0$ is fixed, we consider the minimization of $f(\cdot) := h(F(\cdot, \theta))$ over Ω, write $\mathcal{H}_i$ in place of $\mathcal{H}_{i,t}$, and write $\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta)$ in place of $\tilde{F}_i(x, \theta_t, \mathcal{H}_{i,t}, \delta)$. In this manner, (1.4) can be rewritten as

$$|F_i(x, \theta) - \tilde{F}_i(x, \theta, \mathcal{H}_i, \delta)| \leq \kappa_{\text{app}} \delta. \tag{2.1}$$

In Section 2.1, we describe a generic procedure for the construction of so-called *fully linear* element function models and fully linear master models. In Section 2.2, a description of the algorithm is provided with pseudocode presented in Algorithm 2.1. We sketch a convergence proof for Algorithm 2.1 in Section 2.3 and Appendix A.

Inputs to our algorithm include an initial iterate $x_0 \in \mathbb{R}^n$, a maximal trust region radius $\Delta_{\max} \in \mathbb{R}_{>0}$ such that $\Delta_k \leq \Delta_{\max}$ for all $k \in \mathbb{N}$, and a maximal approximation radius $\delta_{\max} \in \mathbb{R}_{>0}$ such that $\delta_k \leq \delta_{\max}$ for all $k \in \mathbb{N}$. Denoting the $f(x_0)$-sublevel set for the objective function f as $\mathcal{L}_0 = \{x \in \mathbb{R}^{n_x} : f(x) \leq f(x_0)\}$, it follows by construction that the iterate sequence $\{x_k\}$ generated by the algorithm and the points that are used to approximate function values are contained in the enlarged set

$$\mathcal{L}_{\text{enl}} := \mathcal{L}_0 + B_{\text{tr}}(0, \Delta_{\max}) + B_{\text{app}}(0, \delta_{\max}).$$

Throughout this section, we assume the following about the functions defining f.

ASSUMPTION 2.1. *There exists an open convex set $\mathcal{X}$ containing $\mathcal{L}_{\text{enl}} \cap \Omega$ over which, for each $i \in [p]$, one has that $F_i(\cdot, \theta)$ is continuously differentiable, $F_i(\cdot, \theta)$ is Lipschitz continuous with constant $L_{F_i,x} \in \mathbb{R}_{>0}$ such that*

$$|F_i(x, \theta) - F_i(\bar{x}, \theta)| \leq L_{F_i,x} \|x - \bar{x}\|_{\text{tr}} \quad \text{for all} \quad (x, \bar{x}) \in \mathcal{X} \times \mathcal{X},$$

and $\nabla F_i(\cdot, \theta)$ is Lipschitz continuous with constant $L_{\nabla F_i,x} \in \mathbb{R}_{>0}$ such that

$$\|\nabla F_i(x, \theta) - \nabla F_i(\bar{x}, \theta)\|_{\text{tr}^*} \leq L_{\nabla F_i,x} \|x - \bar{x}\|_{\text{tr}} \quad \text{for all} \quad (x, \bar{x}) \in \mathcal{X} \times \mathcal{X}.$$

In addition, there exists an open convex set $\mathcal{Y}$ containing $\{F(x) : x \in \mathcal{L}_{\text{enl}} \cap \Omega\}$ over which one has that h is twice-continuously differentiable, h is Lipschitz continuous with constant $L_h \in \mathbb{R}_{>0}$ such that

$$|h(y) - h(\bar{y})| \leq L_h \|y - \bar{y}\|_2 \quad \text{for all} \quad (y, \bar{y}) \in \mathcal{Y} \times \mathcal{Y}, \tag{2.2}$$

∇h is Lipschitz continuous with constant $L_{\nabla h} \in \mathbb{R}_{>0}$ such that

$$\|\nabla h(y) - \nabla h(\bar{y})\|_2 \leq L_{\nabla h} \|y - \bar{y}\|_2 \quad \text{for all} \quad (y, \bar{y}) \in \mathcal{Y} \times \mathcal{Y}, \tag{2.3}$$

there exists $\kappa_{\nabla h} \in \mathbb{R}_{>0}$ such that $\|\nabla h(y)\|_2 \leq \kappa_{\nabla h}$ for all $y \in \mathcal{Y}$, and for each $i \in [p]$ there exists $\kappa_{\partial_i h} \in \mathbb{R}_{>0}$ such that $|\partial_i h(y)| \leq \kappa_{\partial_i h}$ for all $y \in \mathcal{Y}$.

We augment this assumption in the next section following our introduction of models for approximating the element functions of F. In addition, defining

$$L_{F_x} = \sum_{i=1}^{p} L_{F_i,x}, \quad \bar{L}_{F_x} = \left(\sum_{i=1}^{p} L_{F_i,x}^2 \right)^{\frac{1}{2}}, \quad \text{and} \quad L_{\nabla F_x} = \sum_{i=1}^{p} L_{\nabla F_i,x}, \tag{2.4}$$

let us also mention that, under Assumption 2.1, it follows from (2.4) that f is continuously differentiable and with (1.1) one finds that

$$\|\nabla f(x) - \nabla f(\bar{x})\|_{\text{tr}^*} = \|\nabla F(x, \theta) \nabla h(F(x, \theta)) - \nabla F(\bar{x}, \theta) \nabla h(F(\bar{x}, \theta))\|_{\text{tr}^*}$$

$$\begin{aligned} &\leq \kappa_{\mathrm{tr}_1^*} \|\nabla F(x,\theta)\|_{\mathrm{tr}^*,1} \|\nabla h(F(x,\theta)) - \nabla h(F(\bar{x},\theta))\|_2 \\ &\quad + \kappa_{\mathrm{tr}_1^*} \|\nabla F(x,\theta) - \nabla F(\bar{x},\theta)\|_{\mathrm{tr}^*,1} \|\nabla h(F(\bar{x},\theta))\|_2 \\ &\leq \kappa_{\mathrm{tr}_1^*} \left(\kappa_{\mathrm{tr}_3^*} L_{F_x} L_{\nabla h} \bar{L}_{F_{i,x}} + \kappa_{\nabla h} L_{\nabla F_x}\right) \|x - \bar{x}\|_{\mathrm{tr}} \end{aligned} \tag{2.5}$$

for all $(x, \bar{x}) \in \mathcal{X} \times \mathcal{X}$, where $\mathcal{X}$ is defined as in the assumption, which is to say that the gradient function ∇f is Lipschitz continuous over $\mathcal{X}$ with constant as shown.

2.1. Fully Linear Models. Model-based derivative-free optimization methods often use local linear or quadratic interpolation models [13, 35]. More generally, in iteration $k \in \mathbb{N}$, let $\mathcal{D}_k$ be a set of interpolation points in a neighborhood of the current iterate x_k, e.g., each member of the set $\mathcal{D}_k$ has the form $x - x_k$ for some $x \in B_{\mathrm{tr}}(x_k, \Delta_k) \cap \Omega$. Considering an element function F_i for some $i \in [p]$, one commonly lets $q_{k,i}$ denote a local interpolation model of F_i around x_k satisfying $q_{k,i}(x_k + d) = F_i(x_k + d, \theta)$ for all $d \in \mathcal{D}_k$. However, our algorithm merely requires

$$q_{k,i}(x_k + d) = \tilde{F}_i(x_k + d, \theta, \mathcal{H}_{k,i}, \delta_k) \quad \text{for all} \quad d \in \mathcal{D}_k, \tag{2.6}$$

where the approximate function value $\tilde{F}_i(x_k + d, \theta, \mathcal{H}_{k,i}, \delta_k)$ is required to satisfy a condition similar to (2.1) for all $d \in \mathcal{D}_k$.

We make the following assumption about $q_{k,i}$ for all $k \in \mathbb{N}$ and $i \in [p]$. Let us define $q_k(x) := \begin{bmatrix} q_{k,1}(x) & \dots & q_{k,p}(x) \end{bmatrix}^T$. With respect to the functions h and ∇h, this assumption should be seen to augment Assumption 2.1, e.g., with respect to the definitions of the Lipschitz constants.

ASSUMPTION 2.2. *There exists an open convex set $\mathcal{X}$ containing $\mathcal{L}_{\mathrm{enl}} \cap \Omega$ over which, for all $k \in \mathbb{N}$ and $i \in [p]$, one has that $q_{k,i}$ is twice-continuously differentiable, $\nabla q_{k,i}$ is Lipschitz continuous with constant $L_{\nabla q_i} \in \mathbb{R}_{>0}$ such that*

$$\|\nabla q_{k,i}(x) - \nabla q_{k,i}(\bar{x})\|_{\mathrm{tr}^*} \leq L_{\nabla q_i} \|x - \bar{x}\|_{\mathrm{tr}} \quad \textit{for all} \quad (x, \bar{x}) \in \mathcal{X} \times \mathcal{X},$$

and there exists $\kappa_{\nabla q_i} \in \mathbb{R}_{>0}$ such that $\|\nabla q_{k,i}(x)\|_2 \leq \kappa_{\nabla q_i}$ for all $x \in \mathcal{X}$. In addition, there exists an open convex set $\mathcal{Y}$ containing $\bigcup_{k=1}^{\infty} \{q_k(x) : x \in \mathcal{L}_{\mathrm{enl}} \cap \Omega\}$ over which h is twice-continuously differentiable, h is Lipschitz continuous with constant $L_h \in \mathbb{R}_{>0}$ such that (2.2) holds, ∇h is Lipschitz continuous with constant $L_{\nabla h} \in \mathbb{R}_{>0}$ such that (2.3) holds, there exists $\kappa_{\nabla h} \in \mathbb{R}_{>0}$ such that $\|\nabla h(y)\|_2 \leq \kappa_{\nabla h}$ for all $y \in \mathcal{Y}$, and for each $i \in [p]$ there exists $\kappa_{\partial_i h} \in \mathbb{R}_{>0}$ such that $|\partial_i h(y)| \leq \kappa_{\partial_i h}$ for all $y \in \mathcal{Y}$.

A practical method for constructing a model $q_{k,i}$ that satisfies Assumption 2.2 is provided in [32, 33]. For our purposes later on, let us define

$$L_{\nabla q} = \sum_{i=1}^{p} L_{\nabla q_i} \quad \text{and} \quad \kappa_{\nabla q} = \sum_{i=1}^{p} \kappa_{\nabla q_i}. \tag{2.7}$$

In model-based derivative-free optimization, since the derivative of a black-box function cannot be evaluated analytically and Taylor models are replaced with interpolation or regression models, one needs to ensure that the model is accurate enough in a neighborhood of the current iterate, commonly defined by the trust region. This is achieved by providing bounds on the error in the model and its gradient. The concept of a *fully linear* model defines such a situation common in modern DFO methods.

DEFINITION 2.3. *A sequence of differentiable models $\{m_k\}$, with $m_k : \mathbb{R}^n \to \mathbb{R}$ for all $k \in \mathbb{N}$, is fully linear with respect to a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ over*

$B_{\rm tr}(x_k,\Delta_k)\cap\Omega$ *with constants* $(\kappa_{\rm ef},\kappa_{\rm eg})\in\mathbb{R}_{\geq 0}\times\mathbb{R}_{\geq 0}$ *if, for all* $x\in B_{\rm tr}(x_k,\Delta_k)\cap\Omega$,

$$|m_k(x)-f(x)|\leq\kappa_{\rm ef}\Delta_k^2 \quad \text{and} \quad \|\nabla m_k(x)-\nabla f(x)\|_{\rm tr^*}\leq\kappa_{\rm eg}\Delta_k. \tag{2.8}$$

The following lemma provides sufficient conditions for the models $\{q_{k,i}\}_{k\in\mathbb{N}}$ to be a fully linear with respect to $F_i(\cdot,\theta)$ over a trust region for all $i\in[p]$.

LEMMA 2.4. *Suppose that Assumptions 2.1 and 2.2 hold and consider arbitrary* $k\in\mathbb{N}$ *and* $\mathcal{D}_k:=\{d_1,\ldots,d_{n_x}\}\subset B_{\rm tr}(0,\Delta_k)\cap(\Omega-x_k)$ *such that* (*a*) *for all* $i\in[p]$, *the model* $q_{k,i}$ *satisfies* (2.6) *on* $\mathcal{D}_k\cup\{0\}$, *and* (*b*) *the matrix* $\begin{bmatrix}d_1 & \ldots & d_{n_x}\end{bmatrix}$ *is invertible and, for some* $\Lambda\in\mathbb{R}_{>0}$ *and* $\bar\kappa_{\rm app}\in\mathbb{R}_{>0}$, *one has that*

$$\left\|\begin{bmatrix}d_1 & \ldots & d_{n_x}\end{bmatrix}^{-1}\right\|_2\leq\frac{\Lambda}{\Delta_k} \quad \text{and} \tag{2.9}$$

$$|F_i(x_k+d,\theta)-\tilde F_i(x_k+d,\theta,\mathcal{H}_{k,i},\delta_k)|\leq\bar\kappa_{\rm app}\Delta_k^2 \quad \text{for all} \quad d\in\mathcal{D}_k\cup\{0\}. \tag{2.10}$$

Then, for any $x\in B_{\rm tr}(x_k;\Delta_k)\cap\Omega$ *and any* $i\in[p]$, *the model* $q_{k,i}$ *satisfies*

$$|q_{k,i}(x)-F_i(x,\theta)|\leq\hat\kappa_{\rm ef}\Delta_k^2 \tag{2.11}$$

$$\text{and} \quad \|\nabla q_{k,i}(x)-\nabla F_i(x,\theta)\|_{\rm tr^*}\leq\hat\kappa_{\rm eg}\Delta_k, \tag{2.12}$$

for some $\hat\kappa_{\rm ef}\in\mathbb{R}_{>0}$ *and* $\hat\kappa_{\rm eg}\in\mathbb{R}_{>0}$ *independent of* k.

Proof. Consider an arbitrary index $k\in\mathbb{N}$, set $\mathcal{D}_k:=\{d_1,\ldots,d_{n_x}\}\subset B_{\rm tr}(0,\Delta_k)\cap(\Omega-x_k)$ satisfying the conditions of the lemma, point $x=x_k+s$ for some $\|s\|_{\rm tr}\leq\Delta_k$ such that $x\in\Omega$, and $i\in[p]$. For brevity, let us drop the dependence of F_i on θ; i.e., let us use $F_i(x)$ to represent $F_i(x,\theta)$. Following the proof of Theorem 4.1 in [32], let

$$e_q(s):=q_{k,i}(x_k+s)-F_i(x_k+s) \quad \text{and} \quad e_{\nabla q}(s):=\nabla q_{k,i}(x_k+s)-\nabla F_i(x_k+s).$$

In addition, let $d_0:=0$, and, for all $j\in\{0,1,\ldots,n_x\}$, let

$$I_{1,j}:=\int_0^1\langle\nabla F_i\big(x_k+s+t(d_j-s)\big)-\nabla F_i(x_k+s),d_j-s\rangle dt,$$

$$I_{2,j}:=\int_0^1\langle\nabla q_{k,i}\big(x_k+s+t(d_j-s)\big)-\nabla q_{k,i}(x_k+s),d_j-s\rangle dt,$$

$$I_{3,j}:=\int_0^1\langle\nabla F_i\big(x_k+s-ts\big)-\nabla F_i(x_k+s),-s\rangle dt, \quad \text{and}$$

$$I_{4,j}:=\int_0^1\langle\nabla q_{k,i}\big(x_k+s-ts\big)-\nabla q_{k,i}(x_k+s),-s\rangle dt.$$

By [15, Lemma 4.1.2], it follows for all $j\in\{0,1,\ldots,n_x\}$ that

$$\langle e_{\nabla q}(s),d_j-s\rangle=I_{1,j}-I_{2,j}-e_q(s)-F_i(x_k+d_j)+q_{k,i}(x_k+d_j), \tag{2.13}$$

while one also finds for all $j\in\{0,1,\ldots,n_x\}$ that

$$\begin{aligned}\langle e_{\nabla q}(s),d_j\rangle=&\,I_{1,j}-I_{2,j}-I_{3,j}+I_{4,j}\\&-F_i(x_k+d_j)+q_{k,i}(x_k+d_j)+F_i(x_k)-q_{k,i}(x_k).\end{aligned} \tag{2.14}$$

Let us bound the integrals on the right-hand sides of (2.13) and (2.14). First, under Assumption 2.1, it follows for all $j\in\{0,1,\ldots,n_x\}$ that

$$|I_{1,j}|\leq\int_0^1\|\nabla F_i\big(x_k+s+t(d_j-s)\big)-\nabla F_i(x_k+s)\|_{\rm tr^*}\|d_j-s\|_{\rm tr}dt$$

$$\leq L_{\nabla F_{i,x}} \|d_j - s\|_{\mathrm{tr}}^2 \int_0^1 t dt \leq 2L_{\nabla F_{i,x}} \Delta_k^2. \tag{2.15}$$

Similarly, under Assumptions 2.1 and 2.2 it follows that

$$|I_{2,j}| \leq 2L_{\nabla q_i}\Delta_k^2, \quad |I_{3,j}| \leq \tfrac{1}{2}L_{\nabla F_{i,x}}\Delta_k^2, \quad |I_{4,j}| \leq \tfrac{1}{2}L_{\nabla q_i}\Delta_k^2. \tag{2.16}$$

Moreover, by (2.6) and (2.10), it follows for all $j \in \{0, 1, \ldots, n_x\}$ that

$$|F_i(x_k + d_j) - q_{k,i}(x_k + d_j)| \leq \bar{\kappa}_{\mathrm{app}}\Delta_k^2. \tag{2.17}$$

To show (2.12), let $\bar{D} \coloneqq \begin{bmatrix} d_1 & \ldots & d_{n_x} \end{bmatrix}$, note (2.14)–(2.17) shows $\|\bar{D}^T e_{\nabla q}(s)\|_\infty \leq \kappa_1 \Delta_k^2$, where $\kappa_1 = (\frac{5}{2}(L_{\nabla F_{i,x}} + L_{\nabla q_i}) + 2\bar{\kappa}_{\mathrm{app}})$, then, with (1.1c) and (2.9), see that

$$\|e_{\nabla q}(s)\|_{\mathrm{tr}^*} \leq \kappa_{\mathrm{tr}_0^*}\|\bar{D}^{-T}\|_2\|\bar{D}^T e_{\nabla q}(s)\|_\infty \leq \kappa_{\mathrm{tr}_0^*}\kappa_1\Lambda\Delta_k =: \hat{\kappa}_{\mathrm{eg}}\Delta_k; \tag{2.18}$$

(2.12) follows since s was chosen arbitrarily in $B_{\mathrm{tr}}(0;\Delta_k)$ such that $x = x_k + s \in \Omega$. To show (2.11), it follows from (2.18), (2.13) for $d_1 = 0$, the fact that $I_{1,j}$ (respectively, $I_{2,j}$) reduces to $I_{3,j}$ (respectively, $I_{4,j}$) for $j = 0$ (since $d_0 = 0$), and (2.15)–(2.17) that

$$\begin{aligned} |e_q(s)| &\leq \|e_{\nabla q}(s)\|_{\mathrm{tr}^*}\|s\|_{\mathrm{tr}} + \tfrac{1}{2}(L_{\nabla F_{i,x}} + L_{\nabla q_i})\Delta_k^2 + \bar{\kappa}_{\mathrm{app}}\Delta_k^2 \\ &\leq \left(\hat{\kappa}_{\mathrm{eg}} + \frac{1}{2}(L_{\nabla F_{i,x}} + L_{\nabla q_i}) + \bar{\kappa}_{\mathrm{app}}\right)\Delta_k^2 =: \hat{\kappa}_{\mathrm{ef}}\Delta_k^2; \end{aligned}$$

(2.11) follows since s was chosen arbitrarily in $B_{\mathrm{tr}}(0;\Delta_k)$ with $x = x_k + s \in \Omega$. ∎

With local models for all of the element functions, a local quadratic model for f around x_k can be obtained using a Taylor expansion. For example, let us consider the model m_k for each $k \in \mathbb{N}$ to be a second-order Taylor series approximation and can be expressed as

$$\begin{aligned} m_k(x) &= h(q_k(x_k)) + \nabla h(q_k(x_k))^T \nabla q_k(x_k)^T (x - x_k) \\ &+ \tfrac{1}{2}(x - x_k)^T \left(\sum_{i=1}^{p} \partial_i h(q_k(x_k)) \nabla^2 q_{k,i}(x_k) + \nabla q_k(x_k) \nabla^2 h(q_k(x_k)) \nabla q_k(x_k)^T\right)(x - x_k). \end{aligned} \tag{2.19}$$

Under our stated assumptions, the second-order derivatives of this model are bounded and the models $\{m_k\}$ are fully linear with respect to f within a trust region. These facts are shown in the next two lemmas.

LEMMA 2.5. *Suppose that Assumption 2.2 holds. Then, there exists $\kappa_{\mathrm{bhm}} \in \mathbb{R}_{>0}$ such that $\|\nabla^2 m_k(x_k)\|_{(\mathrm{tr},\mathrm{tr}^*)} \leq \kappa_{\mathrm{bhm}}$ for all $k \in \mathbb{N}$.*

Proof. Under Assumption 2.2, it follows that $\|\nabla^2 q_{k,i}(x_k)\|_{(\mathrm{tr},\mathrm{tr}^*)} \leq L_{\nabla q_i}$ for all $k \in \mathbb{N}$ and $i \in [p]$ and $\|\nabla^2 h(q_k(x_k))\|_2 \leq L_{\nabla h}$ for all $k \in \mathbb{N}$. Consequently, by (1.1e), (2.7), and the definition of m_k in (2.19), one finds that

$$\begin{aligned} &\|\nabla^2 m_k(x_k)\|_{(\mathrm{tr},\mathrm{tr}^*)} \\ &\leq \sum_{i=1}^{p} |\partial_i h(q_k(x_k))| \|\nabla^2 q_{k,i}(x_k)\|_{(\mathrm{tr},\mathrm{tr}^*)} + \kappa_{\mathrm{tr}_2^*}\|\nabla q_k(x_k)\|_{\mathrm{tr}^*,1}^2 \|\nabla^2 h(q_k(x_k))\|_2 \\ &\leq \sum_{i=1}^{p} (\kappa_{\partial_i h} L_{\nabla q_i}) + \kappa_{\mathrm{tr}_2^*}\kappa_{\nabla q}^2 L_{\nabla h} =: \kappa_{\mathrm{bhm}}, \end{aligned}$$

as desired. ∎

LEMMA 2.6. *Suppose that Assumptions 2.1 and 2.2 hold and consider arbitrary* $k \in \mathbb{N}$ *and* $\mathcal{D}_k := \{d_1, \ldots, d_{n_x}\} \subset B_{\text{tr}}(0, \Delta_k) \cap (\Omega - x_k)$ *such that* (*a*) *for all* $i \in [p]$, *the model* $q_{k,i}$ *satisfies* (2.6) *on* $\mathcal{D}_k \cup \{0\}$, *and* (*b*) *the matrix* $\begin{bmatrix} d_1 & \cdots & d_{n_x} \end{bmatrix}$ *is invertible and, for some* $\Lambda \in \mathbb{R}_{>0}$ *and* $\bar{\kappa}_{\text{app}} \in \mathbb{R}_{>0}$, (2.9) *and* (2.10) *hold. Then,* m_k *defined by* (2.19) *satisfies* (2.8) *in* $B(x_k, \Delta_k) \cap \Omega$ *for some* $(\kappa_{\text{ef}}, \kappa_{\text{eg}}) \in \mathbb{R}^2_{>0}$ *independent of* k.

Proof. Consider an arbitrary index $k \in \mathbb{N}$, set $\mathcal{D}_k := \{d_1, \ldots, d_{n_x}\} \subset B_{\text{tr}}(0, \Delta_k) \cap (\Omega - x_k)$ satisfying the conditions of the lemma, point $x = x_k + s$ for some $\|s\|_{\text{tr}} \leq \Delta_k$ such that $x \in \Omega$, and $i \in [p]$. For brevity, let us drop the dependence of F and F_i on θ; i.e., let us use $F(x)$ and $F_i(x)$ to represent $F(x, \theta)$ and $F_i(x, \theta)$, respectively. To show the first bound in (2.8), observe by the Mean Value Theorem that

$$\begin{aligned} f(x) &= f(x_k) + \nabla f(x_k + \tau_1 s)^T s \\ &= h(F(x_k)) + \nabla h(F(x_k + \tau_1 s))^T \nabla F(x_k + \tau_1 s)^T s \quad \text{for some } \tau_1 \in [0, 1]. \end{aligned} \tag{2.20}$$

Similarly, for the master model m_k in (2.19), it follows for some $\tau_2 \in [0, 1]$ that

$$m_k(x) = h(q_k(x_k)) + \nabla h(q_k(x_k + \tau_2 s))^T \nabla q_k(x_k + \tau_2 s)^T s. \tag{2.21}$$

Therefore, by (2.20) and (2.21), one finds that

$$\begin{aligned} |f(x) - m_k(x)| &\leq \big|h(F(x_k)) - h(q_k(x_k))\big| \\ &\quad + \|\nabla F(x_k + \tau_1 s) \nabla h(F(x_k + \tau_1 s)) - \nabla q_k(x_k + \tau_2 s) \nabla h(q_k(x_k + \tau_2 s))\|_{\text{tr}^*} \|s\|_{\text{tr}}. \end{aligned} \tag{2.22}$$

To bound the first term on the right-hand side of (2.22), one finds from (2.6) and (2.10) and Assumptions 2.1 and 2.2 that

$$|h(F(x_k)) - h(q_k(x_k))| \leq L_h \|F(x_k) - q_k(x_k)\|_2 \leq \kappa_2 \Delta_k^2 \tag{2.23}$$

where $\kappa_2 = \sqrt{p} L_h \bar{\kappa}_{\text{app}}$. To bound the second term, recall that $\|s\|_{\text{tr}} \leq \Delta_k$ and $\Delta_k \leq \Delta_{\max}$ for all $k \in \mathbb{N}$; in addition, by (1.1d), (2.4), (2.7), the fact that $|\tau_1 - \tau_2| \leq 1$, Lemma 2.4, and Assumptions 2.1 and 2.2 it follows that

$$\begin{aligned} &\|\nabla F(x_k + \tau_1 s) \nabla h(F(x_k + \tau_1 s)) - \nabla q_k(x_k + \tau_2 s) \nabla h(q_k(x_k + \tau_2 s))\|_{\text{tr}^*} \\ &\leq \kappa_{\text{tr}_1^*} \|\nabla F(x_k + \tau_1 s) - \nabla q_k(x_k + \tau_1 s)\|_{\text{tr}^*,1} \|\nabla h(F(x_k + \tau_1 s))\|_2 \\ &\quad + \kappa_{\text{tr}_1^*} \|\nabla q_k(x_k + \tau_1 s) - \nabla q_k(x_k + \tau_2 s)\|_{\text{tr}^*,1} \|\nabla h(q_k(x_k + \tau_2 s))\|_2 \\ &\quad + \kappa_{\text{tr}_1^*} \|\nabla q_k(x_k + \tau_1 s)\|_{\text{tr}^*,1} \|\nabla h(F(x_k + \tau_1 s)) - \nabla h(q_k(x_k + \tau_2 s))\|_2 \\ &\leq p \kappa_{\text{tr}_1^*} \hat{\kappa}_{\text{eg}} \|\nabla h(F(x_k + \tau_1 s))\|_2 \Delta_k + \kappa_{\text{tr}_1^*} L_{\nabla q} \|\nabla h(q_k(x_k + \tau_2 s))\|_2 \Delta_k \\ &\quad + \kappa_{\text{tr}_1^*} L_{\nabla h} \|\nabla q_k(x_k + \tau_1 s)\|_{\text{tr}^*,1} \|F(x_k + \tau_1 s) - F(x_k + \tau_2 s)\|_2 \\ &\quad + \kappa_{\text{tr}_1^*} L_{\nabla h} \|\nabla q_k(x_k + \tau_1 s)\|_{\text{tr}^*,1} \|F(x_k + \tau_2 s) - q_k(x_k + \tau_2 s)\|_2 \leq \kappa_3 \Delta_k; \end{aligned} \tag{2.24}$$

where $\kappa_3 := \kappa_{\text{tr}_1^*} \big(\kappa_{\nabla h}(p \hat{\kappa}_{\text{eg}} + L_{\nabla q}) + L_{\nabla h} \kappa_{\nabla q} \big(\bar{L}_{F_x} + \sqrt{p} \hat{\kappa}_{\text{ef}} \Delta_{\max}\big)\big)$. From (2.22)–(2.24) it follows that the first bound in (2.8) holds for $\kappa_{\text{ef}} = \kappa_2 + \kappa_3$, as desired. Next, to show the second bound in (2.8), first observe by Taylor's Theorem that

$$\nabla f(x) = \nabla F(x_k) \nabla h(F(x_k)) + \int_0^1 \nabla^2 f(x_k + \tau s) s d\tau, \tag{2.25}$$

where

$$\nabla^2 f(x_k+\tau s) = \sum_{i=1}^{p} \partial_i h(F(x_k+\tau s))\nabla^2 F_i(x_k+\tau s) + \nabla F(x_k+\tau s)\nabla^2 h(F(x_k+\tau s))\nabla F(x_k+\tau s)^T. \tag{2.26}$$

Also, from (2.19), it follows that

$$\nabla m_k(x) = \nabla q_k(x_k)\nabla h(q_k(x_k)) + \nabla^2 m_k(x_k)s. \tag{2.27}$$

As a result, from (2.25) and (2.27), one obtains

$$\|\nabla m_k(x) - \nabla f(x)\|_{\mathrm{tr}^*} \leq \|\nabla F(x_k)\nabla h(F(x_k)) - \nabla q_k(x_k)\nabla h(q_k(x_k))\|_{\mathrm{tr}^*} + \left\|\int_0^1 \nabla^2 f(x_k+\tau s)s\,d\tau\right\|_{\mathrm{tr}^*} + \|s\|_{\mathrm{tr}}\|\nabla^2 m_k(x_k)\|_{(\mathrm{tr},\mathrm{tr}^*)}. \tag{2.28}$$

Let us now bound each of the terms on the right-hand side of (2.28). For the first term, by Assumptions 2.1 and 2.2 and Lemma 2.4, and (2.4), one has that

$$\begin{aligned} &\|\nabla F(x_k)\nabla h(F(x_k)) - \nabla q_k(x_k)\nabla h(q_k(x_k))\|_{\mathrm{tr}^*} \\ &\leq \|\nabla F(x_k)\nabla h(F(x_k)) - \nabla F(x_k)\nabla h(q_k(x_k))\|_{\mathrm{tr}^*} \\ &\quad + \|\nabla F(x_k)\nabla h(q_k(x_k)) - \nabla q_k(x_k)\nabla h(q_k(x_k))\|_{\mathrm{tr}^*} \\ &\leq \kappa_{\mathrm{tr}_1^*}\|\nabla F(x_k)\|_{\mathrm{tr}^*,1}\|\nabla h(F(x_k)) - \nabla h(q_k(x_k))\|_2 \\ &\quad + \kappa_{\mathrm{tr}_1^*}\|\nabla F(x_k) - \nabla q_k(x_k)\|_{\mathrm{tr}^*,1}\|\nabla h(q_k(x_k))\|_2 \\ &\leq \kappa_{\mathrm{tr}_1^*}\sqrt{p}L_{F_x}L_{\nabla h}\bar{\kappa}_{\mathrm{app}}\Delta_k^2 + \kappa_{\mathrm{tr}_1^*}p\kappa_{\nabla h}\hat{\kappa}_{\mathrm{eg}}\Delta_k \leq \kappa_4\Delta_k \end{aligned} \tag{2.29}$$

where $\kappa_4 := \kappa_{\mathrm{tr}_1^*}\sqrt{p}\big(L_{F_x}L_{\nabla h}\bar{\kappa}_{\mathrm{app}}\Delta_{\max} + \sqrt{p}\kappa_{\nabla h}\hat{\kappa}_{\mathrm{eg}}\big)$. For the second term, one finds

$$\left\|\int_0^1 \nabla^2 f(x_k+\tau s)s\,d\tau\right\|_{\mathrm{tr}^*} \leq \|s\|_{\mathrm{tr}}\int_0^1 \left\|\nabla^2 f(x_k+\tau s)\right\|_{(\mathrm{tr},\mathrm{tr}^*)} d\tau,$$

where, by (2.26), (1.1e), (2.4), and Assumptions 2.1 and 2.2, it follows that

$$\|\nabla^2 f(x_k+\tau s)\|_{(\mathrm{tr},\mathrm{tr}^*)} \leq \sum_{i=1}^{p} |\partial_i h(F(x_k+\tau s))|\left\|\nabla^2 F_i(x_k+\tau s)\right\|_{(\mathrm{tr},\mathrm{tr}^*)} + \kappa_{\mathrm{tr}_2^*}\|\nabla F(x_k+\tau s)\|_{\mathrm{tr}^*,1}^2\|\nabla^2 h(F(x_k+\tau s))\|_2 \leq \kappa_5. \tag{2.30}$$

where $\kappa_5 := \sum_{i=1}^{p}(\kappa_{\partial_i h}L_{\nabla F_{i,x}}) + \kappa_{\mathrm{tr}_2^*}L_{F_x}^2 L_{\nabla h}$. For the third term, by Lemma 2.5,

$$\|\nabla^2 m_k(x_k)\|_{(\mathrm{tr},\mathrm{tr}^*)} \leq \kappa_{\mathrm{bhm}}. \tag{2.31}$$

Substituting (2.29)–(2.31) into (2.28), one obtains

$$\|\nabla m_k(x) - \nabla f(x)\|_{\mathrm{tr}^*} \leq (\kappa_4 + \kappa_5 + \kappa_{\mathrm{bhm}})\Delta_k =: \kappa_{\mathrm{eg}}\Delta_k$$

as desired. ∎

2.2. Algorithm Description. Our algorithmic framework is stated as Algorithm 2.1 on page 13. The main structure of the algorithm is similar to the general trust-region DFO framework considered in [20]. In each iteration $k \in \mathbb{N}$, a set of

points $\mathcal{D}_k$ are determined and the information contained in $\{H_{k,i}\}_{i\in[p]}$ is augmented, if necessary, such that one obtains approximate function values satisfying

$$|F_i(x_k+d,\theta) - \tilde{F}_i(x_k+d,\theta,\mathcal{H}_{k,i},\delta_k)| \leq \kappa_{\text{app}}\delta_k \quad \text{for all} \quad d \in \mathcal{D}_k \cup \{0\}. \tag{2.32}$$

Since $c_{\text{app}} \leq \bar{\kappa}_{\text{app}}/\kappa_{\text{app}}$ (see Algorithm 2.1), (2.32) and the fact that $\delta_k \leftarrow c_{\text{app}}\Delta_k^2$ in line 2 ensures that (2.10) holds, which along with (2.6) and (2.19) means that the models $\{q_{k,i}\}_{i\in[p]}$ and m_k satisfy the requirements of the lemmas in the previous section. An implementable strategy for approximating function values using prior function value information (to ensure (2.32)) is presented and analyzed in Section 3.

Remark 2.7. *We emphasize that our presentation of Algorithm 2.1—in particular, its requirement that $\{q_{k,i}\}_{i\in[p]}$ and m_k are fully linear for all $k \in \mathbb{N}$—has been simplified for our discussion and analysis, even though our analysis could be extended to situations in which fully linear models are not always required. This is consistent with other modern DFO methods, such as those in [13, 20], which do not require a fully linear model in every iteration. For example, as long as $\|\nabla^2 m_k(x_k)\| \leq \kappa_{\text{bhm}}$ and the computed search direction satisfies a Cauchy decrease condition (see (2.34) below) for all $k \in \mathbb{N}$, one can relax the requirements on $\mathcal{D}_k$ in line 3 and accept the trial point as long as $\rho_k \geq \eta$, even if $\| \begin{bmatrix} d_2 & \dots & d_{n_x+1} \end{bmatrix}^{-1} \|_2 > \frac{\Lambda}{\Delta_k}$ (i.e., (2.9) is violated) or $\delta_k > c_{\text{app}}\Delta_k^2$ (i.e., (2.10) may be violated). In such a setting, a couple of modifications of the algorithm are needed. First, one needs to modify the step acceptance conditions to stop the algorithm from decreasing the trust region radius if a step has been computed with a model that is not fully linear. Second, in the criticality step, if the model is not fully linear, then again the trust region radius should not be updated and instead a fully linear model should be constructed.*

Upon construction of m_k, the algorithm considers the stationary measure

$$\pi_k^m = \Big| \min_{\substack{x_k+d\in\Omega \\ \|d\|_{\text{tr}}\leq 1}} \nabla m_k(x_k)^T d \Big|. \tag{2.33}$$

Specifically, if the algorithm finds that π_k^m is smaller than a threshold $\epsilon_c \in \mathbb{R}_{>0}$ and the trust region radius is greater than $\mu\pi_k^m$ for a constant $\mu \in \mathbb{R}_{>0}$, then a criticality step (see line 5) is performed, meaning the trust region radius is decreased and the iterate is unchanged. The purpose of the criticality step is to ensure that the sufficiently small value for π_k^m is due to a stationarity measure for f also being sufficiently small (see (2.36)), not merely due to model inaccuracy. In any case, if line 7 is reached, then a step s_k is computed in the trust-region that guarantees Cauchy decrease, i.e.,

$$m_k(x_k) - m_k(x_k+s_k) \geq \kappa_{\text{fcd}}\pi_k^m \min\left\{\frac{\pi_k^m}{\kappa_{\text{bhm}}+1}, \Delta_k, 1\right\} \tag{2.34}$$

is achieved for some user-prescribed $\kappa_{\text{fcd}} \in (0,1]$, where $\kappa_{\text{bhm}} \in \mathbb{R}_{>0}$ is defined as in Lemma 2.5. Sufficient conditions on s_k to achieve (2.34) and an algorithm for obtaining such a step can be found in [8]. After calculating the step s_k, the trial point $x_k + s_k$ is considered. According to the actual-to-predicted reduction ratio, namely,

$$\rho_k \coloneqq \frac{f(x_k) - f(x_k+s_k)}{m_k(x_k) - m_k(x_k+s_k)}, \tag{2.35}$$

it is decided whether the trial point should be accepted as the new iterate and how the trust region radius should be updated. If ρ_k is greater than a threshold $\eta \in \mathbb{R}_{>0}$

then the trial point is accepted; otherwise, it is rejected. In addition, if $\rho \geq \eta$, then the trust region radius is increased; otherwise, it is decreased.

The main difference between our algorithm and other modern model-based DFO frameworks is that function value approximations may be used at interpolation points in place of exact function values. These approximations are obtained using function values computed at nearby points, either in earlier runs of the algorithm (when solving a related optimization problem) or in earlier iterations of the current run of the algorithm. Specifically, in each iteration $k \in \mathbb{N}$, given the prior information $\mathcal{H}_{k,i}$, the algorithm attempts to approximate $F_i(x_k + d, \theta)$ for all $d \in \mathcal{D}_k$ such that the approximation error is sufficiently small. If there is not enough information available for approximating $F_i(x_k + d, \theta)$ for some $d \in \mathcal{D}_k$, then one may need to evaluate the black-box function F_i at $x_k + d$, which is information that can be used to augment $\mathcal{H}_{k,i}$. See Section 3 for further details on our function approximation strategy.

Another important feature of our framework is that one can also exploit prior function value information when determining the set of points to include in the interpolation set $\mathcal{D}_k$. Contemporary model-based DFO methods that do not exploit prior function evaluations often choose the interpolation set so that it is well-poised [6, 13], for which it is sufficient to ensure that (2.9) holds. For example, [7, 33] provide techniques for obtaining well-poised sets of interpolation points. For our purposes, however, beyond satisfying (2.9), our aim is to determine an interpolation set that exploits prior function evaluations when possible. Toward this end, we define a utility function to assess the potential usefulness of points in $\{\mathcal{H}_{k,i}\}$ that are contained in $B_{\text{tr}}(x_k, \Delta_k) \cap \Omega$. In Section 4, we present our utility function and a complete algorithm (Algorithm 4.1) for constructing the interpolation set such that (2.9) is guaranteed in a manner that prioritizes the use of prior function evaluations.

Algorithm 2.1 : General DFO Framework

Require: $\Delta_0 \in (0, \infty)$; $\Delta_{\max} \in (0, \infty)$; $\delta_{\max} \in (0, \infty)$; $\gamma_{\text{dec}} \in (0, 1)$; $\gamma_{\text{inc}} \in (1, \infty)$; $\eta \in (0, \infty)$; $\mu \in (0, \infty)$; $\xi \in (0, 1]$ (see Theorem 4.6); $\epsilon_c \in (0, \infty)$; $\kappa_{\text{app}} \in (0, \infty)$; $\bar{\kappa}_{\text{app}} \in (0, \infty)$; $c_{\text{app}} \in [0, \bar{\kappa}_{\text{app}}/\kappa_{\text{app}}]$; $\kappa_{\text{fcd}} \in (0, 1]$; $u_{\text{thr}} \in [0, \infty)$.

Require: Initial iterate $x_0 \in \mathbb{R}^{n_x}$; prior information $\{\mathcal{H}_{0,i}\}$.

for $k = 0, 1, \ldots$ **do**
  Set $\delta_k \leftarrow c_{\text{app}} \Delta_k^2$.
  Find $\mathcal{D}_k \subset \Omega - x_k$ and augment $\{\mathcal{H}_{k,i}\}_{i=1}^p$ (if necessary) such that (2.9) holds and the approximate values $\{\tilde{F}_i(x_k + d, \theta, \mathcal{H}_{k,i}, \delta_k)\}_{i \in [p], d \in \mathcal{D}_k \cup \{0\}}$ yield $\{q_{k,i}\}_{i=1}^p$ and m_k satisfying (2.6), (2.32), Assumption 2.2, and (2.19).
  if $\pi_k^m \leq \epsilon_c$ and $\Delta_k > \mu \pi_k^m$ **then**
    Criticality step: set $s_k = 0$, $\rho_k = 0$, $\Delta_{k+1} \leftarrow \gamma_{\text{dec}} \Delta_k$, and $x_{k+1} \leftarrow x_k$.
  else
    Compute s_k satisfying (2.34).
    Evaluate $F_i(x_k + s_k, \theta)$ and augment $\mathcal{H}_{k,i}$ for all $i \in [p]$.
    Compute ρ_k by (2.35).
    if $\rho_k \geq \eta$ **then**
      Successful step: set $\Delta_{k+1} \leftarrow \min\{\gamma_{\text{inc}} \Delta_k, \Delta_{\max}\}$ and $x_{k+1} \leftarrow x_k + s_k$.
    else
      Unsuccessful step: set $\Delta_{k+1} \leftarrow \gamma_{\text{dec}} \Delta_k$ and $x_{k+1} \leftarrow x_k$.
  Set $\mathcal{H}_{k+1,i} \leftarrow \mathcal{H}_{k,i}$ for all $i \in [p]$.

2.3. Convergence Result. A convergence result for Algorithm 2.1 follows in a similar manner as for [13, Algorithm 10.1] and [6, Algorithm 1]. We state the result in terms of a stationarity measure for the minimization of f over Ω that is similar to the previously defined measure for m_k (recall (2.33)), namely,

$$\pi_k^f = \Big| \min_{\substack{x_k+d\in\Omega \\ \|d\|_{\mathrm{tr}}\leq 1}} \nabla f(x_k)^T d \Big|; \tag{2.36}$$

see [8, 20]. Specifically, under the following assumption, the following theorem holds.

ASSUMPTION 2.8. *The objective $f : \mathbb{R}^{n_x} \to \mathbb{R}$ is bounded below on $\mathcal{L}_{\mathrm{enl}} \cap \Omega$.*

THEOREM 2.9. *If Assumptions 2.1, 2.2, and 2.8 hold, then* $\lim_{k\to\infty} \pi_k^f = 0$.

Given the properties of Algorithm 2.1 provided in this section, most notably the fact that it generates fully linear models, a complete proof of Theorem 2.9 follows that of other such convergence analyses of model-based trust-region DFO methods. As a brief overview of the proof, we provide a sketch in Appendix A.

3. Function Approximation. Our general DFO framework presented in Algorithm 2.1 makes use of approximate function values for building interpolation models; specifically, given $(x, \theta, \delta) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_\theta} \times \mathbb{R}_{>0}$ and $\{\mathcal{H}_i\}_{i\in[p]}$, it makes use of

$$\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) \approx F_i(x, \theta) \quad \text{for all} \quad i \in [p].$$

In this section, we describe a simple regression procedure that yields the accuracy required to obtain convergence. In a practical setting, however, an application-specific function approximation scheme, such a surrogate model with guaranteed error bounds, is likely to provide useful function estimates with fewer prior function evaluations.

In the following, we show that if a sufficient amount of function value information with respect to F_i for some $i \in [p]$ at points *near* (x, θ) (in a sense defined in this section) is available, then a simple procedure can compute $\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta)$ satisfying the requirements stated in Algorithm 2.1, namely, a bound of the form (2.32), without further function evaluations. If such information is not available, then the algorithm may instead compute $F_i(x, \theta)$ explicitly and set $\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) \leftarrow F_i(x, \theta)$, for which the requirements in the algorithm are satisfied trivially. Ultimately, we claim that there exist situations in practice when such explicit function evaluations can be avoided, as our numerical experiments in Section 5 demonstrate. For ease of exposition, we explain our approximation scheme for two particular cases introduced in Section 1, i.e., least-squares and multi-output simulation optimization problems.

3.1. Least-Squares Optimization Problems. Recall problem (1.5) and consider arbitrary $i \in [p]$. Let us also assume the following about the function ϕ.

ASSUMPTION 3.1. *There exists an open convex set Ξ containing $\mathcal{L}_{\mathrm{enl}} \times \mathcal{W}$ over which ϕ is Lipschitz continuous with constant $L_\phi \in \mathbb{R}_{>0}$ such that*

$$|\phi(x, w) - \phi(\bar{x}, \bar{w})| \leq L_\phi \left\| \begin{bmatrix} x - \bar{x} \\ w - \bar{w} \end{bmatrix} \right\|_{\mathrm{app}}$$

for all $((x, w), (\bar{x}, \bar{w})) \in \Xi \times \Xi$.

Under Assumption 3.1, computing $\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) \approx F_i(x, \theta) = \phi(x, w_i)$ satisfying the requirements of Algorithm 2.1 requires a sufficient amount of function value information in $\mathcal{H}_i$ (defined in (1.6)) with respect to F_i. We capture such information

in a set that we define for some $\delta \in \mathbb{R}_{>0}$ as

$$\mathcal{A}_i(x, w_i, \delta) = \left\{ (\bar{x}, \bar{w}, \phi(\bar{x}, \bar{w})) \in \mathcal{H}_i : \left\| \begin{bmatrix} \bar{x} - x \\ \bar{w} - w_i \end{bmatrix} \right\|_{\text{app}} \leq \delta \right\}. \tag{3.1}$$

For notational convenience, let us explicitly express the finite set $\mathcal{A}_i(x, w_i, \delta) = \{(u_j, \phi_j)\}_{j=1}^N$, where $u_j := (x_j, w_j) \in \mathbb{R}^{n_u}$ and $\phi_j = \phi(x_j, w_j)$ for all $j \in [N]$. Given $\mathcal{A}_i(x, w_i, \delta)$ for some $i \in [p]$, let us now describe a procedure for computing an approximation $\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) \approx F_i(x, \theta) = \phi(x, w_i)$ satisfying the requirements in Algorithm 2.1. In particular, with parameters $\alpha^T = \begin{bmatrix} \alpha_0 & \alpha_1^T & \alpha_2^T \end{bmatrix}$, let us express

$$\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) = \alpha_0 + \alpha_1^T x + \alpha_2^T w_i.$$

We propose computing α by solving a regularized linear regression problem with regularization parameter $\lambda \in \mathbb{R}_{>0}$, namely,

$$\min_{\alpha \in \mathbb{R}^{n_u+1}} \frac{1}{2} \sum_{j=1}^{N} \left(\alpha_0 + \alpha_1^T x_j + \alpha_2^T w_j - \phi_j \right)^2 + \frac{\lambda}{2} \left(\|\alpha_1\|_2^2 + \|\alpha_2\|_2^2 \right). \tag{3.2}$$

Observe that the intercept α_0 is not regularized in this problem, which has the result that the approximation is invariant to translations (see Lemma 3.2 below). For further discussion about regularizing in this manner, see [19, Section 3.4.1]. The objects that are used to compute the approximation are then $u := (x, w_i)$ along with

$$M \coloneqq \begin{bmatrix} 1 & \dots & 1 \\ u_1 & \dots & u_N \end{bmatrix}^T \in \mathbb{R}^{N \times (n_u+1)} \quad \text{and} \quad \phi \coloneqq \begin{bmatrix} \phi_1 & \dots & \phi_N \end{bmatrix}^T \in \mathbb{R}^N. \tag{3.3}$$

In particular, expressing (3.2) in terms of (M, ϕ) yields

$$\min_{\alpha \in \mathbb{R}^{n_u+1}} \frac{1}{2} \|M\alpha - \phi\|_2^2 + \frac{\lambda}{2} \left(\|\alpha_1\|_2^2 + \|\alpha_2\|_2^2 \right). \tag{3.4}$$

Defining $\bar{I} := I - \mathbf{e}_1 \mathbf{e}_1^T$, one finds that the unique solution of this problem is given by $\alpha = \left(M^T M + \lambda \bar{I} \right)^{-1} M^T \phi$, from which it follows that the desired approximation is obtained by solving (3.4), then computing

$$\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) := \begin{bmatrix} 1 & u^T \end{bmatrix} \alpha = \beta^T \phi \approx F_i(x, \theta), \tag{3.5}$$

where for the given regularization parameter $\lambda \in \mathbb{R}_{>0}$ we define

$$\beta^T = \begin{bmatrix} 1 & u^T \end{bmatrix} (M^T M + \lambda \bar{I})^{-1} M^T. \tag{3.6}$$

Our goal in the remainder of this section is to show that (3.5) satisfies the requirements of Algorithm 2.1, namely, a bound of the form in (2.32). Our first lemma shows that the value of the vector β defined in (3.6) has invariance properties.

LEMMA 3.2. *The vector β defined in (3.6) is invariant to translations in the sense that for any vector $u_c \in \mathbb{R}^{n_u}$ one finds that*

$$\begin{bmatrix} 1 & (u - u_c)^T \end{bmatrix} \left(M_c^T M_c + \lambda \bar{I} \right)^{-1} M_c^T = \begin{bmatrix} 1 & u^T \end{bmatrix} (M^T M + \lambda \bar{I})^{-1} M^T,$$

where

$$M_c \coloneqq \begin{bmatrix} 1 & \dots & 1 \\ u_1 - u_c & \dots & u_N - u_c \end{bmatrix}^T.$$

Proof. Defining $C := \bar{I} + \mathbf{e}_1 \begin{bmatrix} 1 & -u_c^T \end{bmatrix}$, one finds that $M_c = MC$; thus,

$$\begin{aligned} &\begin{bmatrix} 1 & (u-u_c)^T \end{bmatrix} \left(M_c^T M_c + \lambda \bar{I}\right)^{-1} M_c^T \\ &= \begin{bmatrix} 1 & u^T \end{bmatrix} C(C^T(M^TM + \lambda\bar{I})C)^{-1}C^TM^T = \begin{bmatrix} 1 & u^T \end{bmatrix} (M^TM + \lambda\bar{I})^{-1}M^T, \end{aligned}$$

where the first equation follows since $\bar{I} = C^T\bar{I}C$. ∎

Our next lemma, which is similar to [13, Lemma 4.9], shows that (3.5) yields an affine combination of the elements of ϕ since the elements of β in (3.6) sum to 1.

LEMMA 3.3. *The vector β defined in (3.6) satisfies $\beta^T\mathbf{1} = 1$.*

Proof. Let $u_c = u$, and $M_c = M - \mathbf{1}\begin{bmatrix} 0 & u^T \end{bmatrix}$. By Lemma 3.2, one finds

$$\beta^T = \begin{bmatrix} 1 & u^T \end{bmatrix} (M^TM + \lambda\bar{I})^{-1}M^T = \mathbf{e}_1^T(M_c^TM_c + \lambda\bar{I})^{-1}M_c^T.$$

Hence, the result follows as long as $\mathbf{e}_1^T(M_c^TM_c + \lambda\bar{I})^{-1}M_c^T\mathbf{1} = 1$. To see this, observe that since the first column of M_c is $\mathbf{1}$, one finds $(M_c^TM_c + \lambda\bar{I})\mathbf{e}_1 = M_c^T\mathbf{1}$. Multiplying both sides on the left by $(M_c^TM_c + \lambda\bar{I})^{-1}$, one concludes that $\mathbf{e}_1 = (M_c^TM_c + \lambda\bar{I})^{-1}M_c^T\mathbf{1}$. Thus, $\mathbf{e}_1^T(M_c^TM_c + \lambda\bar{I})^{-1}M_c^T\mathbf{1} = \mathbf{e}_1^T\mathbf{e}_1 = 1$, as desired. ∎

We are now prepared to prove our main result of this section.

LEMMA 3.4. *Letting $\sigma_{\min} \in \mathbb{R}_{\geq 0}$ be the minimum singular value of $M_d^TM_d$, where $M_d := \begin{bmatrix} u_1 - u & \dots & u_N - u \end{bmatrix}^T$, it follows that $\tilde{F}_i(x,\theta,\mathcal{H}_i,\delta)$ defined in (3.5) satisfies*

$$\left|\tilde{F}_i(x,\theta,\mathcal{H}_i,\delta) - F_i(x,\theta)\right| \leq \kappa_{\text{app}}\delta \tag{3.7}$$

$$\text{where } \kappa_{\text{app}} := L_\phi\left(1 + \left(\frac{2N\kappa_{\text{app}_0^*}}{\sigma_{\min} + \lambda}\right)\delta\right).$$

Proof. By Lemma 3.3, it follows that $\beta^T\mathbf{1} = 1$, meaning

$$\begin{aligned} |\tilde{F}_i(x,\theta,\mathcal{H}_i,\delta) - F_i(x,\theta)| &= |\beta^T\phi - F_i(x,\theta)\beta^T\mathbf{1}| \leq \sum_{j=1}^N |\beta_j||\phi_j - F_i(x,\theta)| \\ &= \sum_{j=1}^N |\beta_j||\phi(x_j,w_j) - \phi(x,w_i)| \leq L_\phi\|\beta\|_1 \left\| \begin{bmatrix} x_j - x \\ w_j - w_i \end{bmatrix} \right\|_{\text{app}} \\ &\leq L_\phi\|\beta\|_1\delta, \end{aligned} \tag{3.8}$$

where the last inequality follows by the definition of $\mathcal{A}_i(x,w_i,\delta)$. Our aim now is to bound $\|\beta\|_1$. Letting $u_c = \frac{1}{N}\sum_{j=1}^N u_j$ and $M_c = M - \mathbf{1}\begin{bmatrix} 0 & u_c^T \end{bmatrix}$, one finds that

$$(M_c^TM_c + \lambda\bar{I})^{-1} = \begin{bmatrix} \frac{1}{N} & \mathbf{0}^T \\ \mathbf{0} & (M_d^TM_d + \lambda I)^{-1} \end{bmatrix}.$$

Therefore, by applying the result of Lemma 3.2, one finds that

$$\beta^T = \begin{bmatrix} 1 & (u-u_c)^T \end{bmatrix} \left(M_c^TM_c + \lambda\bar{I}\right)^{-1} M_c^T = \tfrac{1}{N}\mathbf{1}^T + (u-u_c)^T(M_d^TM_d + \lambda I)^{-1}M_d^T;$$

thus, for any $j \in [N]$, the jth element of β satisfies

$$|\beta_j| = \left|\frac{1}{N} + (u-u_c)^T(M_d^TM_d + \lambda I)^{-1}(u_j - u)\right|$$

$$\leq \frac{1}{N} + \|u - u_c\|_{\text{app}} \|(M_d^T M_d + \lambda I)^{-1}(u_j - u)\|_{\text{app}^*}$$
$$\leq \frac{1}{N} + \|(M_d^T M_d + \lambda I)^{-1}(u_j - u)\|_{\text{app}^*} \delta$$
$$\leq \frac{1}{N} + \kappa_{\text{app}_0^*} \|u_j - u\|_{\text{app}} \|(M_d^T M_d + \lambda I)^{-1}\|_{(2,2)} \delta = \frac{1}{N} + \left(\frac{2\kappa_{\text{app}_0^*}}{\sigma_{\min} + \lambda}\right) \delta^2.$$

Therefore, it follows that

$$\|\beta\|_1 = \sum_{j=1}^{N} |\beta_j| \leq 1 + \left(\frac{2N\kappa_{\text{app}_0^*}}{\sigma_{\min} + \lambda}\right) \delta^2,$$

which combined with (3.8) yields the desired conclusion. ∎

3.2. Multi-Output Simulation. Recall the multi-output simulation optimization setting with $\mathcal{H}_i$ defined in (1.7) and consider arbitrary $i \in [p]$. The set of available information for approximating $F_i(x, \theta)$ for some $\delta \in \mathbb{R}_{>0}$ is given by

$$\mathcal{A}_i(x, \theta, \delta) = \left\{ (\bar{x}, \bar{\theta}, F_i(\bar{x}, \bar{\theta})) \in \mathcal{H}_i : \left\| \begin{bmatrix} \bar{x} - x \\ \bar{\theta} - \theta \end{bmatrix} \right\|_{\text{app}} \leq \delta \right\}. \tag{3.9}$$

The general procedure for approximating $F_i(x, \theta)$ is similar to the least-squares case; hence in order to avoid repetition, we only highlight the differences.

Let us express $\mathcal{A}_i(x, \theta, \delta) = \{(u_j, \phi_j)\}_{j=1}^N$, where $u_j = (x_j, \theta_j) \in \mathbb{R}^{n_u}$ and $\phi_j = F_i(x_j, \theta_j)$. Similar to the least-squares case, we propose setting $\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) = \alpha_0 + \alpha_1^T x + \alpha_2^T \theta$ for some $\alpha = \begin{bmatrix} \alpha_0 & \alpha_1^T & \alpha_2^T \end{bmatrix}$. Let us define M and ϕ as in (3.3), but with u_j and ϕ_j defined as above for all $j \in [N]$. One can obtain α by solving a problem of the form (3.4), and as a result, for $u = (x, \theta)$, an approximation for $F_i(x, \theta)$ is given by (3.5). Likewise, Lemma 3.2 and 3.3 hold. Finally, a result similar to Lemma 3.4 can be obtained if the following assumption is made.

ASSUMPTION 3.5. *There exists an open convex set Ξ containing $\mathcal{L}_{\text{enl}} \times \Theta$ over which F_i is Lipschitz continuous with constant $L_{F_i} \in \mathbb{R}_{>0}$ such that*

$$\left| F_i(x, \theta) - F_i(\bar{x}, \bar{\theta}) \right| \leq L_{F_i} \left\| \begin{bmatrix} x - \bar{x} \\ \theta - \bar{\theta} \end{bmatrix} \right\|_{\text{app}}$$

for all $\big((x, \theta), (\bar{x}, \bar{\theta})\big) \in \Xi \times \Xi$.

The following lemma is the revised version of Lemma 3.4 for this setting, the proof of which is similar to the proof of Lemma 3.4; hence, it is omitted.

LEMMA 3.6. *Letting $\sigma_{\min} \in \mathbb{R}_{\geq 0}$ be the minimum singular value of $M_d^T M_d$, where $M_d \coloneqq \begin{bmatrix} u_1 - u & \dots & u_N - u \end{bmatrix}^T$, it follows that $\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta)$ defined in (3.5) satisfies*

$$\left| \tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) - F_i(x, \theta) \right| \leq \kappa_{\text{app}} \delta \text{ where } \kappa_{\text{app}} \coloneqq L_{F_i} \left(1 + \left(\frac{2N\kappa_{\text{app}_0^*}}{\sigma_{\min} + \lambda}\right) \delta\right).$$

4. Interpolation Set Construction. In this section, we provide an implementable method for determining an interpolation set that contains $n_x + 1$ sufficiently affinely independent points in the sense that (2.9) is guaranteed to hold. For our purposes here, let us write $\Omega = [x_L, x_U]$ for some $x_L \in (\mathbb{R} \cup \{-\infty\})^{n_x}$ and

$x_U \in (\mathbb{R} \cup \{\infty\})^{n_x}$ with $x_{L,j} < x_{U,j}$ for all $j \in [n_x]$. Our strategy involves first running a modified version of [33, Algorithm 4.1], which is presented as Algorithm 4.1 below. If the output of this procedure is a set of $n_x + 1$ points, then the set is complete and we show that (2.9) is guaranteed to hold. Otherwise, the algorithm that we present as Algorithm 4.2 in this section can be called iteratively until a total of $n_x + 1$ points have been obtained, where again we show that (2.9) is guaranteed to hold. In either case, after obtaining $n_x + 1$ points, we call [33, Algorithm 4.2] to potentially add additional points to the interpolation set and construct a quadratic model.

Algorithm 4.1 determines a set of points to include in $\mathcal{D}_k$ from those in $\{\mathcal{H}_{k,i}\}$. First, in line 2, the subset of points from this history that are in $B_{\text{tr}}(x_k, \Delta_k) \cap \Omega$ and have a utility as measured by a function u that exceeds a threshold u_{thr} are identified (more on this shortly). For example, in the case of least-squares optimization (when the information history $\{\mathcal{H}_{k,i}\}$ is defined as in (1.6)), this set may have the form

$$\mathcal{E} \leftarrow \{x \in B_{\text{tr}}(x_k, \Delta_k) \cap \Omega : (x, w, \phi(x, w)) \in \{\mathcal{H}_{k,i}\}, u(x, \theta, \{\mathcal{H}_{k,i}\}) \geq u_{\text{thr}}\}, \tag{4.1}$$

while for the case of multi-output simulation optimization (when the information history $\{\mathcal{H}_{k,i}\}$ is defined as in (1.7)), the set may have the form

$$\mathcal{E} \leftarrow \{x \in B_{\text{tr}}(x_k, \Delta_k) \cap \Omega : (x, \theta, F_i(x, \theta)) \in \{\mathcal{H}_{k,i}\}, u(x, \theta, \{\mathcal{H}_{k,i}\}) \geq u_{\text{thr}}\}. \tag{4.2}$$

The set $\mathcal{E}$ is then ordered according to the utility function u, the definition of which may be problem formulation dependent. For example, in the context of least-squares optimization, for any point $x \in B_{\text{tr}}(x_k, \Delta_k) \cap \Omega$ such that $(x, w, \phi(x, w)) \in \{\mathcal{H}_{k,i}\}$, we define the utility as the fraction of element functions $\{\phi(\cdot, w_i)\}$ that can be approximated with the desired accuracy δ_k (see line 2 of Algorithm 2.1) at x. On the other hand, for multi-output simulation optimization, since performing one simulation generates p outputs simultaneously, at any such point x, either all element functions can be well approximated or there is not enough information available for approximating any of them. A good choice for the utility function in this case may be a function that estimates the accuracy of the approximation at x. As discussed in Section 3, the accuracy of the approximation at x depends on the number of nearby points as well as the geometry of the nearby points (see Lemma 3.6).

Algorithm 4.1 adds elements to $\mathcal{D}_k$ iteratively while also maintaining $Z \in \mathbb{R}^{n_x \times n_z}$ as a corresponding orthonormal matrix such that $\text{span}(Z) = \text{null}([\mathcal{D}_k]_{1:\text{end}})$, where

$$[\mathcal{D}_k]_{1:\text{end}} = \text{matrix composed of vectors in } \mathcal{D}_k \text{ except } 0. \tag{4.3}$$

Observe that the norm of the projection in line 5 of the algorithm can be computed cheaply since it is equal to $\|Z^T(x^{(i)} - x_k)\|_2 / \Delta_k$.

If Algorithm 4.1 returns an interpolation set with $|\mathcal{D}_k| = n_x + 1$, then (2.9) holds (as we show in Theorem 4.8). Otherwise, additional points need to be added. Under the assumption that the (potentially infinite) bound constraints defined by $\Omega = [x_L, x_U]$ might not be relaxable, we propose Algorithm 4.2, which employs the subroutine in Algorithm 4.3, for augmenting the set $\mathcal{D}_k$ in a manner that ensures that (2.9) continues to hold. This algorithm can be called iteratively until $|\mathcal{D}_k| = n_x + 1$.

Consider $\mathcal{D}_k$ with $|\mathcal{D}_k| < n_x + 1$ obtained after a call to Algorithm 4.1 and (potentially) call(s) to Algorithm 4.2. Since $|\mathcal{D}_k| < n_x + 1$, $\text{null}([\mathcal{D}_k]_{1:\text{end}}) \neq \{0\}$, so $n_z \geq 1$. Following the spirit of Algorithm 4.1, one can augment $\mathcal{D}_k$ with $d \in \mathbb{R}^{n_x}$ in a manner that guarantees that (2.9) continues to hold by ensuring that

$$\left\| \text{proj}_Z \left(\frac{d}{\Delta_k} \right) \right\|_2 \geq \xi, \quad \|d\|_{\text{tr}} \leq \Delta_k, \quad \text{and} \quad x_k + d \in \Omega. \tag{4.4}$$

Algorithm 4.1 : Construction of elements for $\mathcal{D}_k$ from $\{\mathcal{H}_{k,i}\}$

Require: Values and parameters from Algorithm 2.1.

Set $\mathcal{D}_k \leftarrow \{0\}$ and $Z \leftarrow I_{n_x}$.
Choose $\mathcal{E} \subset B_{\text{tr}}(x_k, \Delta_k) \cap \Omega$ from elements of $\{H_{k,i}\}$ (see, e.g., (4.1) or (4.2)).
Sort $\mathcal{E}$ as $\{x^{(1)}, \ldots, x^{(|\mathcal{E}|)}\}$ so $u(x^{(j)}, \theta, \{\mathcal{H}_{k,i}\}) \geq u(x^{(j+1)}, \theta, \{\mathcal{H}_{k,i}\})$ for $j \in [|\mathcal{E}|]$.
for $j = 1, \ldots, |\mathcal{E}|$ **do**
    if $\left\| \text{proj}_Z \left(\frac{x^{(j)} - x_k}{\Delta_k} \right) \right\|_2 \geq \xi$ **then**
        Set $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{x^{(j)} - x_k\}$.
        Compute orthonormal Z with $\text{span}(Z) = \text{null}([\mathcal{D}_k]_{1:\text{end}})$ (see (4.3)).
return $(\mathcal{D}_k, Z)$

for ξ sufficiently small. Recalling that $\|\text{proj}_Z (d/\Delta_k)\| = \|Z^T d\|/\Delta_k$, one way in which one might ensure this property is by solving

$$\max_{d \in \mathbb{R}^n} \ \Upsilon(Z, d) \ \text{ s.t. } \ \|d\|_{\text{tr}} \leq \Delta_k \ \text{ and } \ x_L - x_k \leq d \leq x_U - x_k, \tag{4.5}$$

where $\Upsilon(Z, d) := \|Z^T d\|_2^2$. However, a globally optimal solution of (4.5) may be computationally expensive to obtain, and such a solution is not actually necessary.

The motivation for our proposed Algorithm 4.2 is to solve an approximation of problem (4.5), a solution of which turns out to be sufficient for ensuring that (2.9) continues to hold as long as ξ is sufficiently small. In particular, the algorithm solves

$$\max_{(v,i,\tau) \in \mathbb{R}^{n_x} \times [n_z] \times \mathbb{R}_{\geq 0}} \ \Upsilon(Z, d(v, \tau)) \ \text{ s.t. } \ v \in \{-z_i, z_i\} \ \text{ and } \ \|d(v, \tau)\|_{\text{tr}} \leq \Delta_k, \tag{4.6}$$

where z_i is the ith column of Z and where for any $(i, \tau) \in [n_z] \times \mathbb{R}_{\geq 0}$ and $v \in \{-z_i, z_i\}$ the vector $x_k + d(v, \tau)$ is the projection of $x_k + \tau v$ onto $\Omega = [x_L, x_U]$. A formula for this vector is easily derived. In particular, for $v \in \mathbb{R}^{n_x}$, let $\bar{\tau}(v) \in (\mathbb{R} \cup \{\infty\})^{n_x}$ have

$$\bar{\tau}_j(v) := \begin{cases} \frac{x_{U,j} - x_{k,j}}{v_j} & \text{if } v_j > 0 \text{ and } x_{U,j} < \infty, \\ \frac{x_{L,j} - x_{k,j}}{v_j} & \text{if } v_j < 0 \text{ and } x_{L,j} > -\infty, \\ \infty & \text{otherwise} \end{cases} \tag{4.7}$$

for $j \in [n_x]$. The projection of $x_k + \tau v$ onto $[x_L, x_U]$ is given by $x_k + d(v, \tau)$, where

$$d(v, \tau) = \begin{bmatrix} \min\{\tau, \bar{\tau}_1(v)\} v_1 & \ldots & \min\{\tau, \bar{\tau}_{n_x}(v)\} v_{n_x} \end{bmatrix}^T. \tag{4.8}$$

Remark 4.1. In (4.8) and subsequent calculations below, we interpret equations and operations involving infinite quantities in the following natural ways: $\infty = \infty$; $\infty \times \infty = \infty$; $\infty - (-\infty) = \infty$; $\min\{\infty, \infty\} = \infty$; $b - \infty = -\infty$ and $b + \infty = \infty$ for any $b \in \mathbb{R}$; $a \times \infty = \infty$ and $-a \times \infty = -\infty$ for any $a \in \mathbb{R}_{>0}$; and $\|v\| = \infty$ for any norm $\|\cdot\|$ and any $v \in (\mathbb{R} \cup \{-\infty, \infty\})^{n_x}$ that has an element equal to $-\infty$ or ∞. One additional rule that we intend, which is not natural in all contexts, is $0 \times \infty = 0$.

Algorithm 4.2 operates by iterating through the columns of Z, where for each $i \in [n_z]$ the optimal values of $\tau \in \mathbb{R}_{\geq 0}$ (with respect to maximizing Υ) are determined along z_i and $-z_i$. If the search along either direction yields a larger value of Υ than

has been observed so far, the solution estimate (with respect to (4.6)) is updated. The subroutine in Algorithm 4.3 is responsible for computing the optimal step sizes.

Algorithm 4.2 : Construction of additional element for $\mathcal{D}_k$

Require: $\mathcal{D}_k$ and corresponding $Z \in \mathbb{R}^{n_x \times n_z}$.

Set $d \leftarrow \mathbf{0}$.
for $i = 1, \ldots, n_z$ **do**
  $\tau^+ \leftarrow \texttt{OptStep}(Z, z_i, \bar{\tau}(z_i))$.
  if $\Upsilon(Z, d(z_i, \tau^+)) > \Upsilon(Z, d)$ **then** set $d \leftarrow d(z_i, \tau^+)$.
  $\tau^- \leftarrow \texttt{OptStep}(Z, -z_i, \bar{\tau}(-z_i))$.
  if $\Upsilon(Z, d(-z_i, \tau^-)) > \Upsilon(Z, d)$ **then** set $d \leftarrow d(-z_i, \tau^-)$.
Set $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{d\}$.
Compute orthonormal Z with $\text{span}(Z) = \text{null}([\mathcal{D}_k]_{1:\text{end}})$ (see (4.3)).
return $(\mathcal{D}_k, Z)$.

Algorithm 4.3 : $\texttt{OptStep}(Z, v, \bar{\tau})$ (see Algorithm 4.2)

Let $\bar{\tau}_{(0)} \leftarrow 0$ and sort distinct values of $\{\bar{\tau}\}_{j=1}^{n_x}$ as $0 < \bar{\tau}_{(1)} \leq \cdots \leq \bar{\tau}_{(\hat{n}_x)}$.
if $\|d(v, \bar{\tau}_{(\hat{n}_x)})\|_{\text{tr}} \leq \Delta_k$ **then**
  return $\arg\max_{\tau \in \{\bar{\tau}_{(j)}\}_{j=1}^{\hat{n}_x}} \Upsilon(Z, d(v, \tau))$.
else
  Find $\hat{j} \in [n_x]$ such that $\|d(v, \bar{\tau}_{(\hat{j}-1)})\|_{\text{tr}} \leq \Delta_k \leq \|d(v, \bar{\tau}_{(\hat{j})})\|_{\text{tr}}$.
  Find $\hat{\tau} \in [\bar{\tau}_{(\hat{j}-1)}, \bar{\tau}_{(\hat{j})}]$ such that $\|d(v, \hat{\tau})\|_{\text{tr}} = \Delta_k$.
  return $\arg\max_{\tau \in \{\bar{\tau}_{(j)}\}_{j=1}^{\hat{j}-1} \cup \{\hat{\tau}\}} \Upsilon(Z, d(v, \tau))$.

Our goal in the remainder of this section is to prove that the output $\mathcal{D}_k$ from Algorithm 4.1 satisfies (2.9) and, after any subsequent call to Algorithm 4.2, the elements of $\mathcal{D}_k$ continue to satisfy (2.9) as long as ξ is sufficiently small. This notion of sufficiently small is determined by a threshold revealed in Theorem 4.6.

Our first lemma bounds $\tau \in \mathbb{R}_{\geq 0}$ below if $d(\cdot, \tau)$ lies on the trust-region boundary.

LEMMA 4.2. *If* $(v, \tau) \in \mathbb{R}^{n_x} \times \mathbb{R}_{\geq 0}$, $\|v\|_2 = 1$, *and* $\|d(v, \tau)\|_{\text{tr}} = \Delta_k$, *then* $\tau \geq \frac{\Delta_k}{\kappa_{\text{tr}_1}}$.

Proof. By (1.1b), (4.8), and the conditions of the lemma, $\text{tr} < \infty$ implies

$$\Delta_k^{\text{tr}} = \|d(v, \tau)\|_{\text{tr}}^{\text{tr}} = \sum_{j=1}^{n_x} (\min\{\tau, \bar{\tau}_j(v)\})^{\text{tr}} |v_j|^{\text{tr}} \leq \tau^{\text{tr}} \|v\|_{\text{tr}}^{\text{tr}} \leq \tau^{\text{tr}} \kappa_{\text{tr}_1}^{\text{tr}} \|v\|_2^{\text{tr}} = \tau^{\text{tr}} \kappa_{\text{tr}_1}^{\text{tr}},$$

which yields the desired conclusion. (Here, the superscript "tr" denotes the tr-th power of a number.) The result for $\text{tr} = \infty$ can be shown in a similar manner. ∎

Going forward, corresponding to $v \in \mathbb{R}^{n_x}$, we choose an index $j^*(v)$ such that

$$j^*(v) \in \arg\max_{j \in [n_x]} |v_j|. \tag{4.9}$$

The following lemma is trivial (recall (4.7)), so we state it without proof.

LEMMA 4.3. *If* $v \in \mathbb{R}^{n_x}$ *has* $\|v\|_2 = 1$, *then* $\frac{1}{\sqrt{n_x}} \leq |v_{j^*(v)}| \leq 1$ *and*

$$\max\{\bar{\tau}_{j^*(v)}(-v), \bar{\tau}_{j^*(v)}(v)\} \geq \frac{x_{U,j^*(v)} - x_{L,j^*(v)}}{2|v_{j^*(v)}|}.$$

Our next lemma shows a lower bound for the optimal value of (4.6) if a call to the `OptStep` subroutine finds all step sizes yielding points within the trust region.

LEMMA 4.4. *Consider line 3 in Algorithm 4.2 for any $i \in [n_z]$. If it is found within the call to* `OptStep`$(Z, z_i, \bar{\tau}(z_i))$ *that* $\|d(z_i, \bar{\tau}_{(\hat{n}_x)}(z_i))\|_{\mathrm{tr}} \leq \Delta_k$, *then* $\tau^+ \in \mathbb{R}_{\geq 0}$ *yields* $\Upsilon(Z, d(z_i, \tau^+)) \geq \frac{(\bar{\tau}_{j^*(z_i)}(z_i))^2}{n_x^2}$. *The same property holds for line 5 and* `OptStep`$(Z, -z_i, \bar{\tau}(-z_i))$ *with respect to* $\tau^- \in \mathbb{R}_{\geq 0}$.

Proof. Without loss of generality, consider the call to `OptStep`$(Z, z_i, \bar{\tau}(z_i))$. The proof for the call to `OptStep`$(Z, -z_i, \bar{\tau}(-z_i))$ is nearly identical. By the conditions of the lemma, (4.8), Lemma 4.3, and the fact that $\bar{\tau}_{(n_x)} \geq \bar{\tau}_{j^*(z_i)}(z_i)$, it follows that

$$\begin{aligned}
\Upsilon(Z, d(z_i, \tau^+)) &\geq \Upsilon(Z, d(z_i, \bar{\tau}_{(\hat{n}_x)})) = \sum_{l=1}^{n_z} (z_l^T d(z_i, \bar{\tau}_{(\hat{n}_x)}))^2 \\
&\geq \left(\sum_{j=1}^{n_x} [z_i]_j d_j(z_i, \bar{\tau}_{(n_x)}) \right)^2 = \left(\sum_{j=1}^{n_x} [z_i]_j^2 \min\{\bar{\tau}_{(n_x)}, \bar{\tau}_j(z_i)\} \right)^2 \\
&\geq [z_i]_{j^*(z_i)}^4 \min\{\bar{\tau}_{(n_x)}, \bar{\tau}_{j^*(z_i)}(z_i)\}^2 \geq \frac{\bar{\tau}_{j^*(z_i)}(z_i)^2}{n_x^2},
\end{aligned}$$

as desired. ∎

Next, we present the following lemma, which considers cases when a call to the `OptStep` subroutine finds that not all step sizes yield points within the trust region.

LEMMA 4.5. *Consider line 3 in Algorithm 4.2 for any $i \in [n_z]$. If it is found in the call to* `OptStep`$(Z, z_i, \bar{\tau}(z_i))$ *that* $\|d(z_i, \bar{\tau}_{(\hat{n}_x)}(z_i))\|_{\mathrm{tr}} > \Delta_k$, *then* $\tau^+ \in \mathbb{R}_{\geq 0}$ *yields*

$$\Upsilon(Z, d(z_i, \tau^+)) \geq \left(\min\left\{ \frac{\Delta_k}{\kappa_{\mathrm{tr}_1}}, \bar{\tau}_{j^*(z_i)}(z_i) \right\} \right)^2 \frac{1}{n_x^2}.$$

The same property holds for line 5 and `OptStep`$(Z, -z_i, \bar{\tau}(-z_i))$ *w.r.t.* $\tau^- \in \mathbb{R}_{\geq 0}$.

Proof. The proof is nearly identical to that of Lemma 4.4, except with the computed value $\hat{\tau}$ in place of $\bar{\tau}_{(n_x)}$, where by Lemma 4.2 one has that $\hat{\tau} \geq \frac{\Delta_k}{\kappa_{\mathrm{tr}_1}}$. ∎

We now prove our main result of this section.

THEOREM 4.6. *If the well-poisedness parameter is chosen such that*

$$\xi \in \left(0, \frac{1}{n_x} \min\left\{ \frac{1}{\kappa_{\mathrm{tr}_1}}, \frac{\min_{j \in [n_x]}(x_{U,j} - x_{L,j})}{2\Delta_{\max}} \right\} \right],$$

then any call to Algorithm 4.2 adds $d \in \mathbb{R}^{n_x}$ to $\mathcal{D}_k$ that satisfies (4.4).

Proof. By construction, the vector $d \in \mathbb{R}^{n_x}$ that is added to the elements of $\mathcal{D}_k$ by the algorithm satisfies $\|d\|_{\mathrm{tr}} \leq \Delta_k$ and $x_k + d \in \Omega = [x_L, x_U]$. Hence, all that remains is to prove that d also satisfies the first inequality in (4.4). According to the construction of the algorithm, it is sufficient to show that for some $i \in [n_z]$ one finds

$$\frac{1}{\Delta_k} \Upsilon_{i,\max} := \frac{1}{\Delta_k} \max\left\{ \sqrt{\Upsilon(Z, d(z_i, \tau^+))}, \sqrt{\Upsilon(Z, d(-z_i, \tau^-))} \right\} \geq \xi.$$

If the calls to `OptStep`$(Z, z_i, \bar{\tau}(z_i))$ and `OptStep`$(Z, -z_i, \bar{\tau}(-z_i))$ both find that all step sizes yield points within the trust region, then Lemmas 4.3 and 4.4 imply

$$\frac{1}{\Delta_k} \Upsilon_{i,\max} \geq \frac{1}{n_x \Delta_k} \max\{\bar{\tau}_{j^*(z_i)}(z_i), \bar{\tau}_{j^*(-z_i)}(-z_i)\}$$

$$\geq \frac{1}{n_x \Delta_k} \frac{x_{U,j^*(z_i)} - x_{L,j^*(z_i)}}{2|[z_i]_{j^*(z_i)}|} \geq \frac{x_{U,j^*(z_i)} - x_{L,j^*(z_i)}}{2n_x \Delta_{\max}}.$$

Otherwise, if the call to $\texttt{OptStep}(Z, z_i, \bar{\tau}(z_i))$ and/or $\texttt{OptStep}(Z, -z_i, \bar{\tau}(-z_i))$ finds that some of the step sizes yield points that are outside of the trust region, then one may conclude from Lemma 4.5 that

$$\frac{1}{\Delta_k}\Upsilon_{i,\max} \geq \frac{1}{n_x \Delta_k} \max\left\{ \min\left\{ \frac{\Delta_k}{\kappa_{\mathrm{tr}_1}}, \bar{\tau}_{j^*(z_i)}(z_i) \right\}, \min\left\{ \frac{\Delta_k}{\kappa_{\mathrm{tr}_1}}, \bar{\tau}_{j^*(-z_i)}(-z_i) \right\} \right\}.$$

Considering all possible cases for which term obtains in the minima in the expression above, one may conclude with Lemmas 4.2 and 4.3 that

$$\frac{1}{\Delta_k}\Upsilon_{i,\max} \geq \frac{1}{n_x} \min\left\{ \frac{1}{\kappa_{\mathrm{tr}_1}}, \frac{x_{U,j^*(z_i)} - x_{L,j^*(z_i)}}{2\Delta_{\max}} \right\}.$$

Combining the results of these cases, the desired conclusion follows. ∎

The next lemma shows that the norm in line 5 of Algorithm 4.1 and the first inequality in (4.4) measures the magnitude of a pivot of a QR factorization.

LEMMA 4.7. *At line 5 of Algorithm 4.1, one finds with $d = (x^{(i)} - x_k)$ that*

$$\left\| proj_Z\left(\frac{d}{\Delta_k}\right) \right\|_2 = |r|,$$

where r is the last diagonal of R in a QR factorization of $\frac{1}{\Delta_k}\begin{bmatrix}[\mathcal{D}_k]_{1:\mathrm{end}} & d\end{bmatrix}$.

Proof. Letting QR be a QR factorization of $\frac{1}{\Delta_k}[\mathcal{D}_k]_{1:\mathrm{end}}$, one finds that a QR factorization of the augmented matrix has, for some vector q satisfying $Q^T q = 0$,

$$\frac{1}{\Delta_k}\begin{bmatrix}[\mathcal{D}_k]_{1:\mathrm{end}} & d\end{bmatrix} = \begin{bmatrix}Q & q\end{bmatrix}\begin{bmatrix}R & v \\ \mathbf{0}^T & r\end{bmatrix}.$$

Left-multiplication by Z^T yields $Z^T \frac{d}{\Delta_k} = (Z^T q)r$. Then, since $Q^T q = 0$, it follows that $q = Zu$ for some vector u with $\|u\|_2 = 1$, from which it follows that

$$\left\| \mathrm{proj}_Z\left(\frac{d}{\Delta_k}\right) \right\|_2 = \left\| Z^T\left(\frac{d}{\Delta_k}\right) \right\|_2 = |r| \left\| Z^T q \right\|_2 = |r| \left\| Z^T Z u \right\|_2 = |r|,$$

as desired. ∎

The following theorem/proof is similar to [35, Lemma 4.2] and [32, Lemma 3.2].

THEOREM 4.8. *Once Algorithm 4.1 or iterative calls to Algorithm 4.2 yields an interpolation set $\mathcal{D}_k$ with $|\mathcal{D}_k| = n_x+1$, it follows that (2.9) holds with $\Lambda = \dfrac{n_x^{\frac{n_x-1}{2}} \kappa_{\mathrm{tr}_0}^{n_x-1}}{\xi^{n_x}}$.*

Proof. Since $|\mathcal{D}_k| = n_x + 1$, let us express $[\mathcal{D}_k]_{1:\mathrm{end}} = \begin{bmatrix}d_1 & \cdots & d_{n_x}\end{bmatrix}$, where it follows from $\mathcal{E} \subset B_{\mathrm{tr}}(x_k, \Delta_k)$, the manner in which $\mathcal{D}_k$ is constructed, and (1.1a) that $\|d_i\|_2 \leq \kappa_{\mathrm{tr}_0}\|d_i\|_{\mathrm{tr}} \leq \kappa_{\mathrm{tr}_0}\Delta_k$ for all $i \in [n_x]$. In addition, let $\{\sigma_i\}_{i=1}^{n_x}$ be the singular values of $\frac{1}{\kappa_{\mathrm{tr}_0}\Delta_k}[\mathcal{D}_k]_{1:\mathrm{end}}$ such that $\sigma_1 \leq \cdots \leq \sigma_{n_x}$. One finds that

$$\|[\mathcal{D}_k]_{1:\mathrm{end}}^{-1}\|_2 = \frac{1}{\kappa_{\mathrm{tr}_0}\Delta_k} \left\| \left(\frac{[\mathcal{D}_k]_{1:\mathrm{end}}}{\kappa_{\mathrm{tr}_0}\Delta_k} \right)^{-1} \right\|_2 = \frac{1}{\sigma_1 \kappa_{\mathrm{tr}_0}\Delta_k}. \tag{4.10}$$

In addition, letting QR denote a QR factorization of $\frac{1}{\Delta_k}[\mathcal{D}_k]_{1:\text{end}}$, it follows that the determinant of $\frac{1}{\Delta_k}[\mathcal{D}_k]_{1:\text{end}}$ is equal to the product of the diagonal elements of R, call them $\{r_i\}_{i=1}^{n_x}$. Recalling $\|d_i\|_2 \leq \kappa_{\text{tr}_0}\Delta_k$, we have

$$\sigma_{n_x} = \left\| \frac{[\mathcal{D}_k]_{1:\text{end}}}{\kappa_{\text{tr}_0}\Delta_k} \right\|_2 \leq \left\| \frac{[\mathcal{D}_k]_{1:\text{end}}}{\kappa_{\text{tr}_0}\Delta_k} \right\|_F \leq \sqrt{n_x}.$$

Since Lemma 4.7 ensures that $|r_i| \geq \xi$ for all $i \in [n_x]$, it follows that

$$\sigma_1 n_x^{\frac{n_x-1}{2}} \geq \sigma_1 \sigma_{n_x}^{n_x-1} \geq \prod_{i=1}^{n_x} \sigma_i = \left| \det\left(\frac{[\mathcal{D}_k]_{1:\text{end}}}{\kappa_{\text{tr}_0}\Delta_k} \right) \right| = \left| \prod_{i=1}^{n_x} \frac{r_i}{\kappa_{\text{tr}_0}} \right| \geq \left(\frac{\xi}{\kappa_{\text{tr}_0}} \right)^{n_x}.$$

These inequalities along with (4.10) yields

$$\|[\mathcal{D}_k]_{1:\text{end}}^{-1}\|_2 \leq \frac{n_x^{\frac{n_x-1}{2}} \kappa_{\text{tr}_0}^{n_x-1}}{\xi^{n_x}\Delta_k} = \frac{\Lambda}{\Delta_k},$$

which is the desired conclusion. ∎

5. Numerical Experiments. The purpose of our experiments is to demonstrate the reduction in function evaluations that can be achieved by our method through its exploitation of prior function evaluations when solving a sequence of related problems. We compare Algorithm 2.1 as it is stated, referred to in this section as $\text{Alg}_{\mathcal{H}}$, and an algorithm that has all of the same features of Algorithm 2.1 except that does not utilize prior function evaluations, referred to as $\text{Alg}_{\emptyset}$. We choose $T = 100$, and for all $t \in \{0, \ldots, T-1\}$ we run both algorithms until a fixed budget of two simplex gradient evaluations (i.e., $2p(n_x + 1)$ function evaluations) is exhausted. We consider such a relatively small amount of function evaluations because the advantage of our method is exploited best when solution time is crucial, such as an online setting, in which the true optimum is out of reach and a good approximate solution suffices.

We implemented our algorithm in Python and ran our experiments on a Linux workstation. We tested our algorithm on a variety of problems and the results were similar in all cases. For our purposes here, we present the results obtained from a single representative least-squares problem involving an ODE that describes the conversion of methanol into various hydrocarbons [17], which for parameters $x = \begin{bmatrix} x_1 & \ldots & x_5 \end{bmatrix}^T \in \mathbb{R}^5_{\geq 0}$ and state $v(\tau) = \begin{bmatrix} v_1(\tau) & v_2(\tau) & v_3(\tau) \end{bmatrix}^T \in \mathbb{R}^3$ is described (with τ denoting time) by:

$$\begin{aligned}
\frac{dv_1}{d\tau} &= -\Big(2x_2 - \frac{x_1 v_2}{(x_2 + x_5)v_1 + v_2} + x_3 + x_4\Big)v_1; \\
\frac{dv_2}{d\tau} &= \frac{x_1 v_1 (x_2 v_1 - v_2)}{(x_2 + x_5)v_1 + v_2} + x_3 v_1; \\
\frac{dv_3}{d\tau} &= \frac{x_1 v_1 (v_2 + x_5 v_1)}{(x_2 + x_5)v_1 + v_2} + x_4 v_1.
\end{aligned}$$

Here, the constraint set is $\Omega_t = \mathbb{R}^5_{\geq 0}$ for all $t \in \{0, \ldots, T-1\}$.

We fix a vector $\bar{x} = \begin{bmatrix} 1.78 & 2.17 & 1.86 & 1.80 & 0 \end{bmatrix}^T \in \mathbb{R}^5_{\geq 0}$, then, iteratively for each $t \in \{0, \ldots, T-1\}$, we generate the data for the tth least-squares objective in the

following manner. First, we establish seven initial conditions by taking each of the following vectors, perturbing it by a realization of a random vector having a uniform distribution over a 2-norm ball with radius 0.1, then projecting the result onto the 3-dimensional standard simplex so that the elements are nonnegative and sum to one:

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{3}{4} \\ \frac{1}{4} \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{3}{4} \\ 0 \\ \frac{1}{4} \end{bmatrix}, \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix}, \begin{bmatrix} \frac{1}{4} \\ \frac{3}{4} \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{1}{4} \\ 0 \\ \frac{3}{4} \end{bmatrix} \right\}.$$

(This projection of the initial condition is meaningful for the application since the state elements correspond to proportions.) We denote the resulting lth initial condition by $v^0_{(l)} \in \mathbb{R}^3_{\geq 0}$ for all $l \in [7]$. Second, we establish the time points $\tau_{(1)} = 0.1$, $\tau_{(2)} = 0.4$, and $\tau_{(3)} = 0.8$, which are fixed for all $t \in \{0, \ldots, T-1\}$. At this point in the data generation for problem t, we have established $\{w_{i,t}\}_{i \in [21]}$, with each one corresponding to a given initial condition and time point; specifically, each $i \in [21]$ corresponds to a unique pair $(j, l) \in [3] \times [7]$, corresponding to which we define $w_{i,t} := \begin{bmatrix} \tau_{(j)} & (v^0_{(l)})^T \end{bmatrix}^T$. All that remains to generate the data for problem t is to establish the values $\{y_{i,t}\}_{i \in [21]}$. For this, we first generate $x_{(t)} \in \mathbb{R}^5_{\geq 0}$ by adding to $\bar{x}$ a realization from a uniform $[0, 1]^5$ distribution. Then, for all $i \in [21]$, we let $\phi(x_{(t)}, w_{i,t})$ denote the value of v_3 from the ODE at time $[w_{i,j}]_0$ when using the initial condition $[w_{i,t}]_{1:3}$ and set

$$y_{i,t} \leftarrow \phi(x_{(t)}, w_{i,t}) + |\phi(x_{(t)}, w_{i,t})| u_{i,t},$$

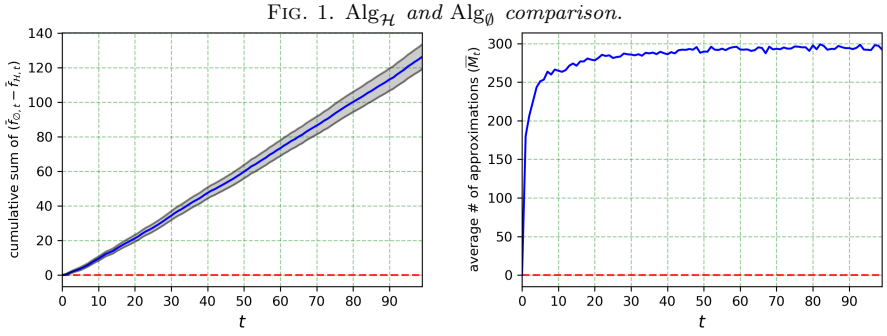
where $u_{i,t}$ is a realization from a uniform $[-0.1, 0.1]$ distribution. Overall, we have established the data $\{(w_{i,t}, y_{i,t})\}_{i \in [21]}$ for problem t, which is defined as in (1.5) with ϕ defined as above. By generating the problem data in this manner for $t \in \{0, \ldots, T-1\}$, each optimization problem is similar, but different due to the randomization of the initial conditions and the noise in the measurement data.

To understand the typical behavior of our algorithm, we generated a total of $N = 100$ replications with $T = 100$ problems as described in the previous paragraph. Each repetition starts with problem $t = 0$, which involves no history of function evaluations, meaning that $\text{Alg}_{\mathcal{H}}$ and $\text{Alg}_{\emptyset}$ always perform equivalently for $t = 0$. However, for all $t \in \{1, \ldots, T-1\}$, $\text{Alg}_{\mathcal{H}}$ makes use of prior function evaluations when possible while $\text{Alg}_{\emptyset}$ does not. Let $f^k_{\mathcal{H},t}$ and $f^k_{\emptyset,t}$, respectively, denote the final objective function value obtained by $\text{Alg}_{\mathcal{H}}$ and $\text{Alg}_{\emptyset}$ when solving repetition k of problem t. Averaging over the repetitions, we obtain the values $\bar{f}_{\mathcal{H},t} = \frac{1}{N} \sum_{k \in [N]} f^k_{\mathcal{H},t}$ and $\bar{f}_{\emptyset,t} = \frac{1}{N} \sum_{k \in [N]} f^k_{\emptyset,t}$ for all $t \in \{0, \ldots, T-1\}$. In addition, to get a sense of $\text{Alg}_{\mathcal{H}}$'s ability to take more steps within the function evaluation limit by using approximated function values in place of true function values, we record M^k_t as the number of approximated function values used in repetition k of problem t. These are averaged over the repetitions to obtain $\overline{M}_t = \frac{1}{N} \sum_{k \in [N]} M^k_t$ for all $t \in \{0, \ldots, T-1\}$.

Figure 1 presents the results of our experiments. For all $t \in \{0, \ldots, T-1\}$, the plot on the left shows $\sum_{\bar{t} \in [t]} (\bar{f}_{\emptyset,\bar{t}} - \bar{f}_{\mathcal{H},\bar{t}})$, the accumulated improvement of $\text{Alg}_{\mathcal{H}}$ over $\text{Alg}_{\emptyset}$, as well as the surrounding interval of width $\pm \frac{1.96}{\sqrt{N}} \sigma_t$, where σ_t is the standard deviation of $\{\sum_{\bar{t} \in [t]} (f_{\emptyset,\bar{t}} - f_{\mathcal{H},\bar{t}})\}_{k \in [N]}$. The increasing trend shows that as the function evaluation history increases in size, $\text{Alg}_{\mathcal{H}}$ is continually able to obtain improved final objective values over $\text{Alg}_{\emptyset}$. The plot on the right shows, for all $t \in \{0, \ldots, T-1\}$, the average number of function values that $\text{Alg}_{\mathcal{H}}$ is able to approximate in a run of the algorithm. Recalling that the budget in each run is $2p(n_x + 1) = 2 \times 21 \times (5+1) = 252$,

the plot shows that by problem $t \approx 10$ over half of the function values used by $\text{Alg}_{\mathcal{H}}$ come from approximations rather than (expensive) actual evaluations, which allows the algorithm to take more iterations to improve the objective compared to $\text{Alg}_{\emptyset}$.

FIG. 1. $\text{Alg}_{\mathcal{H}}$ *and* $\text{Alg}_{\emptyset}$ *comparison.*



6. Conclusion. We proposed and analysed a model-based DFO algorithm for solving a sequence of related optimization problems under the assumption that the black-box objective function is smooth and black-box function evaluations are the computational bottleneck of the algorithm. We provided a regression-based method that approximates the objective function at interpolation points by using black-box function evaluations that are obtained from solving prior problems. In a practical setting, this could be replaced by a more efficient scheme that is tailored to the specific application. In addition, we proposed an algorithm for choosing the interpolation points that are more likely to be approximated accurately given the history of prior black-box function evaluations. Our numerical results showed that our algorithm outperforms a state-of-the-art DFO algorithm for solving an engineering problem when a history of black-box function evaluations is available.

REFERENCES

[1] C. Audet and J. E. Dennis Jr, *A pattern search filter method for nonlinear programming without derivatives*, SIAM Journal on Optimization, 14 (2004), pp. 980–1010.
[2] C. Audet and J. E. Dennis Jr, *Mesh adaptive direct search algorithms for constrained optimization*, SIAM Journal on optimization, 17 (2006), pp. 188–217.
[3] D. M. Bates and D. G. Watts, *Nonlinear regression analysis and its applications*, John Wiley & Sons, New York, 1988.
[4] 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 (TOMS), 45 (2019), pp. 1–41.
[5] C. Cartis and L. Roberts, *A derivative-free gauss–newton method*, Mathematical Programming Computation, 11 (2019), pp. 631–674.
[6] P. Conejo, E. W. Karas, L. G. Pedroso, A. A. Ribeiro, and M. Sachine, *Global convergence of trust-region algorithms for convex constrained minimization without derivatives*, Applied Mathematics and Computation, 220 (2013), pp. 324–330.
[7] A. Conn, K. Scheinberg, and P. Toint, *A derivative free optimization algorithm in practice*, in 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 1998, p. 4718.
[8] 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.
[9] A. R. Conn, K. Scheinberg, and P. L. Toint, *On the convergence of derivative-free methods for unconstrained optimization*, Approximation theory and optimization: tributes to MJD Powell, (1997), pp. 83–108.

[10] A. R. Conn, K. Scheinberg, and P. L. Toint, *Recent progress in unconstrained nonlinear optimization without derivatives*, Mathematical programming, 79 (1997), pp. 397–414.
[11] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Geometry of interpolation sets in derivative free optimization*, Mathematical programming, 111 (2008), pp. 141–172.
[12] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Geometry of sample sets in derivative-free optimization: polynomial regression and underdetermined interpolation*, IMA journal of numerical analysis, 28 (2008), pp. 721–748.
[13] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*, MPS/SIAM, Philadelphia, 2009.
[14] A. Custódio, J. Dennis, and L. N. Vicente, *Using simplex gradients of nonsmooth functions in direct search methods*, IMA journal of numerical analysis, 28 (2008), pp. 770–784.
[15] J. E. Dennis Jr and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, SIAM, Philadelphia, 1996.
[16] S. L. Digabel and S. M. Wild, *A taxonomy of constraints in simulation-based optimization*, arXiv preprint arXiv:1505.07881, (2015).
[17] E. D. Dolan, J. J. Moré, and T. S. Munson, *Benchmarking optimization software with cops 3.0.*, tech. report, Argonne National Lab., Argonne, IL, 2004.
[18] G. A. Gray and T. G. Kolda, *Appspack 4.0: asynchronous parallel pattern search for derivative-free optimization.*, tech. report, Sandia National Lab., Livermore, CA, 2004.
[19] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*, Springer, New York, 2009.
[20] M. Hough and L. Roberts, *Model-based derivative-free methods for convex-constrained optimization*, arXiv preprint arXiv:2111.05443, (2021).
[21] M. Marazzi and J. Nocedal, *Wedge trust region methods for derivative free optimization*, Mathematical programming, 91 (2002), pp. 289–305.
[22] J. J. Moré and S. M. Wild, *Benchmarking derivative-free optimization algorithms*, SIAM Journal on Optimization, 20 (2009), pp. 172–191.
[23] J. A. Nelder and R. Mead, *A simplex method for function minimization*, The computer journal, 7 (1965), pp. 308–313.
[24] M. J. Powell, *A direct search optimization method that models the objective and constraint functions by linear interpolation*, in Advances in optimization and numerical analysis, Springer, 1994, pp. 51–67.
[25] M. J. Powell, *Uobyqa: unconstrained optimization by quadratic approximation*, Mathematical Programming, 92 (2002), pp. 555–582.
[26] M. J. Powell, *Least Frobenius norm updating of quadratic models that satisfy interpolation conditions*, Mathematical Programming, 100 (2004), pp. 183–215.
[27] M. J. Powell, *The newuoa software for unconstrained optimization without derivatives*, in Large-scale nonlinear optimization, Springer, 2006, pp. 255–297.
[28] M. J. Powell, *The BOBYQA algorithm for bound constrained optimization without derivatives*, Cambridge NA Report NA2009/06, University of Cambridge, (2009), pp. 26–46.
[29] R. G. Regis and S. M. Wild, *Conorbit: constrained optimization by radial basis function interpolation in trust regions*, Opt. Meth. & Soft.imization Methods and Software, 32 (2017), pp. 552–580.
[30] H.-J. M. Shi, M. Q. Xuan, F. Oztoprak, and J. Nocedal, *On the numerical performance of derivative-free optimization methods based on finite-difference approximations*, arXiv preprint arXiv:2102.09762, (2021).
[31] L. N. Vicente and A. L. Custódio, *Analysis of direct searches for discontinuous functions*, Mathematical programming, 133 (2012), pp. 299–325.
[32] S. M. Wild, *Derivative-Free Optimization Algorithms For Computationally Expensive Functions*, PhD dissertation, Cornell University, 2008.
[33] S. M. Wild, *Mnh: A derivative-free optimization algorithm using minimal norm hessians*, tech. report, Cornell University Operations Research and Industrial Engineering, 2008.
[34] S. M. Wild, *Chapter 40: Pounders in tao: Solving derivative-free nonlinear least-squares problems with pounders*, in Advances and trends in optimization with engineering applications, SIAM, 2017, pp. 529–539.
[35] S. M. Wild, R. G. Regis, and C. A. Shoemaker, *ORBIT: Optimization by radial basis function interpolation in trust-regions*, SIAM Journal on Scientific Computing, 30 (2008), pp. 3197–3219.
[36] S. M. Wild and C. Shoemaker, *Global convergence of radial basis function trust region derivative-free algorithms*, SIAM Journal on Optimization, 21 (2011), pp. 761–781.
[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.

Appendix A. Sketch of Proof for Theorem 2.9.

For the sake of brevity and since one can prove the results stated in this section using arguments that have already appeared for similar results for related algorithms in the literature (e.g., see [5]), we provide in this appendix only a discussion of the results that one needs to prove in order to prove Theorem 2.9.

First, one can show for all $k \in \mathbb{N}$ that the difference between the stationarity measures with respect to f and m_k are proportional to Δ_k; see, e.g., [8, Lemma 7].

LEMMA A.1. *For all $k \in \mathbb{N}$, it follows that for any value of Δ_k such that a model m_k is constructed, one has that $|\pi_k^f - \pi_k^m| \leq \kappa_{\text{eg}} \Delta_k$.*

Second, one can show lower bounds for the stationarity measures in certain cases. The proof of this result relies on Lemma A.1.

LEMMA A.2. *For any $k \in \mathbb{N}$, if the condition in line 4 does not hold, then $\pi_k^m \geq \min\{\epsilon_c, \mu^{-1}\Delta_k\}$. If, in addition, $\pi_k^f \geq \epsilon \in \mathbb{R}_{>0}$, then*

$$\pi_k^m \geq \epsilon_{\text{mc}} := \min\left\{\epsilon_c, \frac{\epsilon}{1+\kappa_{\text{eg}}\mu}\right\} \in \mathbb{R}_{>0}.$$

Third, one can show that if Δ_k is sufficiently small, then a successful step occurs.

LEMMA A.3. *For any $k \in \mathbb{N}$, if trust region radius satisfies*

$$\Delta_k \leq \min\{c_0 \pi_k^m, 1\}, \quad \textit{where} \quad c_0 := \min\left\{\mu, \frac{1}{\kappa_{\text{bhm}}+1}, \frac{\kappa_{\text{fcd}}(1-\eta)}{2\kappa_{\text{ef}}}\right\}, \tag{A.1}$$

then the condition in line 4 does not hold and $\rho_k \geq \eta$, i.e., the step is successful.

Fourth, one can show that if the stationarity measure with respect to f is bounded below by a positive constant, then the trust region radius is similarly bounded below. The proof of this result relies on all of the preceding lemmas.

LEMMA A.4. *If $\pi_k^f \geq \epsilon \in \mathbb{R}_{>0}$ for all $k \in \mathbb{N}$, then $\Delta_k \geq \Delta_{\min}$ for all $k \in \mathbb{N}$, where*

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

Fifth, one can show that if the number of successful steps is finite, then the trust region radius and stationarity measure with respect to f must vanish. The proof of this result relies on Lemma A.1.

LEMMA A.5. *If $|\{k \in \mathbb{N} : \rho_k \geq \eta\}|$ is finite, then $\lim_{k\to\infty} \Delta_k = 0$ and $\lim_{k\to\infty} \pi_k^f = 0$.*

Sixth, one can show the trust region radius always vanishes. The proof of this result relies on Lemmas A.2 and A.4.

LEMMA A.6. *The trust region radius vanishes, i.e., $\lim_{k\to\infty} \Delta_k = 0$.*

Seventh, one can show that a subsequence of stationary measures vanishes. The proof of this result relies on Lemmas A.2, A.4, and A.5.

LEMMA A.7. *The limit inferior of $\{\pi_k^f\}$ is zero, i.e., $\liminf_{k\to\infty} \pi_k^f = 0$.*

Finally, given the "liminf" result in Lemma A.7, one can prove Theorem 2.9 using Lemmas A.2, A.5, and A.6 along with common techniques for the analysis of trust-region methods that can turn "liminf" into "lim" results.