



ISE

Industrial and
Systems Engineering

Exploiting Prior Function Evaluations in Derivative-Free Optimization

FRANK E. CURTIS¹, SHIMA DEZFULIAN², AND ANDREAS WÄCHTER²

¹Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA

²Department of IEMS, Northwestern U., Evanston, IL, USA

ISE Technical Report 22T-003



LEHIGH
UNIVERSITY.

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.

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

45 **1.1. Contributions.** We propose and analyze a model-based DFO algorithm
 46 that exploits function evaluations that have been performed when solving related op-
 47 timization problems. This allows the algorithm to avoid new evaluations of black-box
 48 functions at interpolation points when the function values at these points can be
 49 approximated sufficiently accurately using prior evaluations, e.g., by an application-
 50 specific surrogate model. Our algorithm also differs from other model-based DFO
 51 approaches in that it exploits prior information when determining the set of interpo-
 52 lation points itself. This feature is incorporated through the use of a specially designed
 53 utility function that predicts the usefulness of a potential interpolation point. Our
 54 numerical experiments show that our algorithm often outperforms a state-of-the-art
 55 model-based DFO method in our setting of interest, especially when the budget for
 56 function evaluations is small, which is typically the case in real-world applications.

57 **1.2. Notation.** The sets of real numbers, n -dimensional real vectors, and n -by-
 58 m -dimensional real matrices are denoted by \mathbb{R} , \mathbb{R}^n , and $\mathbb{R}^{n \times m}$, respectively. The sets
 59 of nonnegative and positive real numbers are denoted by $\mathbb{R}_{\geq 0}$ and $\mathbb{R}_{> 0}$, respectively.
 60 The vector of all zeros is denoted as $\mathbf{0}$, the identity matrix is denoted as I , the vector
 61 of all ones is denoted as $\mathbf{1}$, and the i th unit vector is denoted as \mathbf{e}_i , where in each case
 62 the size of the object is determined by the context. The set of nonnegative integers
 63 is denoted as \mathbb{N} and we define $[n] := \{1, \dots, n\}$ for any $n \in \mathbb{N} \setminus \{0\}$.

64 For sets $\mathcal{S}_1 \subseteq \mathbb{R}^n$ and $\mathcal{S}_2 \subseteq \mathbb{R}^n$, we define Minkowski addition and subtraction
 65 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
 66 the particular case when one of the sets is a singleton, say $\{c\}$ with $c \in \mathbb{R}^n$, then we
 67 simply write $c + \mathcal{S} = \{c + s : s \in \mathcal{S}\}$. The cardinality of a set \mathcal{S} is denoted by $|\mathcal{S}|$.

68 Given any real number $q \geq 1$, the ℓ_q -norm of a vector $v \in \mathbb{R}^n$ is written as $\|v\|_q$.
 69 The closed ℓ_q -norm ball with center $x \in \mathbb{R}^n$ and radius $\Delta \in \mathbb{R}_{\geq 0}$ is denoted as
 70 $B_q(x, \Delta) := \{\bar{x} : \|\bar{x} - x\|_q \leq \Delta\}$. The dual norm of $\|\cdot\|_q$ is denoted and defined by
 71 $\|z\|_{q^*} := \max\{z^T x : x \in B_q(0, 1)\}$. The Frobenius-norm of a matrix M is denoted by
 72 $\|M\|_F$. 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
 73 M is written as $\|M\|_{p,q} = \left(\sum_{j=1}^n \left(\sum_{i=1}^m |M_{ij}|^p\right)^{\frac{q}{p}}\right)^{\frac{1}{q}}$ and the (p,q) -operator norm of M
 74 is written as $\|M\|_{(p,q)} = \sup_{\|x\|_p \leq 1} \|Mx\|_q$, although, for shorthand, $\|\cdot\|_p = \|\cdot\|_{(p,p)}$.

75 For the sake of generality, two norms used throughout the paper are those defining
 76 the *trust region radius* and the *approximation radius* in our algorithm, which we
 77 respectively denote as $\|\cdot\|_{\text{tr}}$ and $\|\cdot\|_{\text{app}}$ for some real numbers $\text{tr} \geq 1$ and $\text{app} \geq 1$.
 78 In our analysis, we make use of norm equivalence constants for finite-dimensional real
 79 vector spaces; in particular, due to the equivalence of all such norms, for any positive
 80 integers n_x and p , there exist constants $(\kappa_{\text{tr}0}, \kappa_{\text{tr}1}, \kappa_{\text{tr}0}^*, \kappa_{\text{tr}1}^*, \kappa_{\text{tr}2}^*, \kappa_{\text{tr}3}^*, \kappa_{\text{app}0}^*) \in \mathbb{R}_{\geq 0}^7$
 81 such that

$$\begin{aligned}
 82 \quad (1.1a) \quad & \|v\|_2 \leq \kappa_{\text{tr}0} \|v\|_{\text{tr}} && \text{for all } v \in \mathbb{R}^{n_x}, \\
 83 \quad (1.1b) \quad & \|v\|_{\text{tr}} \leq \kappa_{\text{tr}1} \|v\|_2 && \text{for all } v \in \mathbb{R}^{n_x}, \\
 84 \quad (1.1c) \quad & \|v\|_{\text{tr}^*} \leq \kappa_{\text{tr}0}^* \|V^{-1}\|_2 \|Vv\|_{\infty} && \text{for all } (v, V) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x \times n_x}, \\
 85 \quad (1.1d) \quad & \|Vv\|_{\text{tr}^*} \leq \kappa_{\text{tr}1}^* \|V\|_{\text{tr}^*, 1} \|v\|_2 && \text{for all } (v, V) \in \mathbb{R}^p \times \mathbb{R}^{n_x \times p}, \\
 86 \quad (1.1e) \quad & \|VUV^T\|_{(\text{tr}, \text{tr}^*)} \leq \kappa_{\text{tr}2}^* \|V\|_{\text{tr}^*, 1}^2 \|U\|_2 && \text{for all } (V, U) \in \mathbb{R}^{n_x \times p} \times \mathbb{R}^{p \times p}, \\
 87 \quad (1.1f) \quad & \|v\|_{\text{tr}} \|v\|_{\text{tr}^*} \leq \kappa_{\text{tr}3}^* \|v\|_2^2 && \text{for all } v \in \mathbb{R}^{n_x}, \\
 88 \quad (1.1g) \quad & \text{and } \|Vv\|_{\text{app}^*} \leq \kappa_{\text{app}0}^* \|v\|_{\text{app}} \|V\|_{(2,2)} && \text{for all } (v, V) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x \times n_x}.
 \end{aligned}$$

90 For differentiable $g : \mathbb{R}^n \rightarrow \mathbb{R}$ and $y \in \mathbb{R}^n$, we use $\partial_i g(y)$ and $\partial_i \partial_j g(y)$ to denote
 91 $\frac{\partial g(y)}{\partial y_i}$ 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
 92 g is $\nabla g : \mathbb{R}^n \rightarrow \mathbb{R}^n$, where $[\nabla g(y)]_i = \partial_i g(y)$ for any $i \in [n]$ and $y \in \mathbb{R}^n$. If g
 93 is twice differentiable, then the Hessian function of g is $\nabla^2 g : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$, where
 94 $[\nabla^2 g(y)]_{i,j} = \partial_i \partial_j g(y)$ for any $(i, j) \in [n] \times [n]$ and $y \in \mathbb{R}^n$. For differentiable vector-
 95 valued $G : \mathbb{R}^n \rightarrow \mathbb{R}^m$, the transpose of the Jacobian function of G is denoted by
 96 $\nabla G : \mathbb{R}^n \rightarrow \mathbb{R}^{m \times n}$, where $[\nabla G(y)]_{i,j} = \partial_i G_j(y)$ for any $(i, j) \in [n] \times [m]$ and $y \in \mathbb{R}^n$.

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

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

105 **1.3. Problem Formulation.** Consider a sequence of optimization problems,
 106 namely, for all $t \in \mathbb{N}$, consider the minimization problem

$$107 \quad (1.2) \quad \min_{x \in \Omega_t} f_t(x) \quad \text{with} \quad f_t(x) := h_t(F(x, \theta_t)),$$

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

112 Here, constraints are said to be relaxable (respectively, unrelaxable) if they do
 113 not (respectively, do) need to be satisfied for one to be able to evaluate the objective
 114 function. As an example of a set of unrelaxable constraints, consider the case when
 115 $F(x, \theta_t)$ is the output of a simulation that fails to produce a meaningful output unless
 116 $x \in \Omega_t$ [16]. Along with our proposed DFO framework, we provide techniques for
 117 handling unrelaxable constraints. We also conjecture that our proposed techniques
 118 can be extended to the setting when Ω_t is any closed convex set, such as by following
 119 the ideas in [20]. However, since bound constraints are sufficient for our numerical
 120 experimentation, we consider bound constraints only.

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

$$123 \quad (1.3) \quad F(x, \theta_t) = [F_1(x, \theta_t) \quad \cdots \quad F_p(x, \theta_t)]^T,$$

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

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

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

$$139 \quad (1.4) \quad |F_i(x, \theta_t) - \tilde{F}_i(x, \theta_t, \mathcal{H}_{i,t}, \delta)| \leq \kappa_{\text{app}} \delta.$$

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

144 *Least-Squares Optimization Problems.* Consider a sequence of sets of observations,
 145 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 $(w_{i,t}, y_{i,t}) \in \mathcal{W} \times \mathbb{R}$
 146 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
 147 the black-box function $\phi : \Omega_t \times \mathcal{W} \rightarrow \mathbb{R}$ best describes the observations by solving

$$148 \quad (1.5) \quad \min_{x \in \Omega_t} \frac{1}{2} \sum_{i=1}^p (\phi(x, w_{i,t}) - y_{i,t})^2, \quad \text{where } \Omega_t \text{ represents element-wise bounds.}$$

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

$$150 \quad \theta_t := [w_{1,t}^T \quad w_{2,t}^T \quad \dots \quad w_{p,t}^T]^T, \quad F_i(x, \theta_t) := \phi(x, w_{i,t}) \text{ for all } i \in [p],$$

$$151 \quad y_t := [y_{1,t} \quad \dots \quad y_{p,t}]^T, \quad \text{and } h_t(v) := \frac{1}{2} \|v - y_t\|^2 \text{ for all } v \in \mathbb{R}^p.$$

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

$$154 \quad (1.6) \quad \mathcal{H}_{i,t} = \{(x, w, \phi(x, w)) : \phi(x, w) \text{ has been evaluated}\}.$$

155 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
 156 approximate the value of *any* component of F by exploiting *all* prior evaluations of ϕ ,
 157 even those obtained for different components. In other words, the history is the same
 158 for all components of F in the sense that $\mathcal{H}_{i,t} = \mathcal{H}_{j,t}$ for all $(i, j) \in [p] \times [p]$.

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

$$163 \quad (1.7) \quad \mathcal{H}_{i,t} = \{(x, \theta, F_i(x, \theta)) : F_i(x, \theta) \text{ has been evaluated}\}.$$

164 **1.4. Literature Review.** Generally speaking, DFO algorithms can be catego-
 165 rized as direct-search, finite-difference, or model-based methods. Direct-search meth-
 166 ods, such as pattern search methods [1, 18] and the Nelder-Mead method [23], are
 167 often outperformed by model-based methods [22] when one is minimizing a smooth ob-
 168 jective. That said, one of the strengths of direct-search methods is that they are more
 169 readily applicable when an objective function is nonsmooth [2, 14] or even discontin-
 170 uous [31]. Finite-difference approaches approximate derivatives using finite difference
 171 schemes, which are then embedded within a gradient-based optimization approach,
 172 such as a steepest descent or quasi-Newton method; see, e.g., [30]. Empirical evidence
 173 has shown that finite-difference methods can be competitive with model-based meth-
 174 ods, at least when one presumes no noise in the function values. The method that we

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

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

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

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

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

218 **1.5. Organization.** A general DFO algorithmic framework that exploits prior
 219 function evaluations for solving (1.2) is presented in Section 2. In Section 3, we provide
 220 an approximation scheme that uses prior function value information such that one
 221 obtains an implementable instance of the framework. In Section 4, ideas are presented
 222 for obtaining well-poised interpolation sets. The results of numerical experiments are
 223 presented in Section 5 and concluding remarks are provided in Section 6.

224 **2. A Derivative-Free Algorithmic Framework.** A general model-based al-
 225 gorithmic framework for solving problem (1.2) that exploits prior information in order
 226 to build the models is presented in this section. The framework that we propose is de-
 227 signed 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

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

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 $\Delta_{\text{max}} \in \mathbb{R}_{>0}$ such that $\Delta_k \leq \Delta_{\text{max}}$ for all $k \in \mathbb{N}$, and a maximal approximation radius $\delta_{\text{max}} \in \mathbb{R}_{>0}$ such that $\delta_k \leq \delta_{\text{max}}$ 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

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

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*

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

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

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

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

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

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

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

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

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

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

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

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

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 $x \in B_{\text{tr}}(x_k, \Delta_k) \cap \Omega$. 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

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

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 $q_k(x) := [q_{k,1}(x) \ \dots \ q_{k,p}(x)]^T$. 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 $\mathcal{L}_{\text{enl}} \cap \Omega$ 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*

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

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 $\bigcup_{k=1}^{\infty} \{q_k(x) : x \in \mathcal{L}_{\text{enl}} \cap \Omega\}$ 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

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

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 $m_k : \mathbb{R}^n \rightarrow \mathbb{R}$ for all $k \in \mathbb{N}$, is fully linear with respect to a differentiable function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ over*

309 $B_{\text{tr}}(x_k, \Delta_k) \cap \Omega$ with constants $(\kappa_{\text{ef}}, \kappa_{\text{eg}}) \in \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$ if, for all $x \in B_{\text{tr}}(x_k, \Delta_k) \cap \Omega$,

$$310 \quad (2.8) \quad |m_k(x) - f(x)| \leq \kappa_{\text{ef}} \Delta_k^2 \quad \text{and} \quad \|\nabla m_k(x) - \nabla f(x)\|_{\text{tr}^*} \leq \kappa_{\text{eg}} \Delta_k.$$

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

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

$$318 \quad (2.9) \quad \left\| [d_1 \ \dots \ d_{n_x}]^{-1} \right\|_2 \leq \frac{\Lambda}{\Delta_k} \quad \text{and}$$

$$319 \quad (2.10) \quad |F_i(x_k + d, \theta) - \tilde{F}_i(x_k + d, \theta, \mathcal{H}_{k,i}, \delta_k)| \leq \bar{\kappa}_{\text{app}} \Delta_k^2 \quad \text{for all } d \in \mathcal{D}_k \cup \{0\}.$$

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

$$322 \quad (2.11) \quad |q_{k,i}(x) - F_i(x, \theta)| \leq \hat{\kappa}_{\text{ef}} \Delta_k^2$$

$$323 \quad (2.12) \quad \text{and} \quad \|\nabla q_{k,i}(x) - \nabla F_i(x, \theta)\|_{\text{tr}^*} \leq \hat{\kappa}_{\text{eg}} \Delta_k,$$

325 for some $\hat{\kappa}_{\text{ef}} \in \mathbb{R}_{> 0}$ and $\hat{\kappa}_{\text{eg}} \in \mathbb{R}_{> 0}$ independent of k .

326 *Proof.* Consider an arbitrary index $k \in \mathbb{N}$, set $\mathcal{D}_k := \{d_1, \dots, d_{n_x}\} \subset B_{\text{tr}}(0, \Delta_k) \cap$
327 $(\Omega - x_k)$ satisfying the conditions of the lemma, point $x = x_k + s$ for some $\|s\|_{\text{tr}} \leq \Delta_k$
328 such that $x \in \Omega$, and $i \in [p]$. For brevity, let us drop the dependence of F_i on θ ; i.e.,
329 let us use $F_i(x)$ to represent $F_i(x, \theta)$. Following the proof of Theorem 4.1 in [32], let

$$330 \quad e_q(s) := q_{k,i}(x_k + s) - F_i(x_k + s) \quad \text{and} \quad e_{\nabla q}(s) := \nabla q_{k,i}(x_k + s) - \nabla F_i(x_k + s).$$

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

$$333 \quad I_{1,j} := \int_0^1 \langle \nabla F_i(x_k + s + t(d_j - s)) - \nabla F_i(x_k + s), d_j - s \rangle dt,$$

$$334 \quad I_{2,j} := \int_0^1 \langle \nabla q_{k,i}(x_k + s + t(d_j - s)) - \nabla q_{k,i}(x_k + s), d_j - s \rangle dt,$$

$$335 \quad I_{3,j} := \int_0^1 \langle \nabla F_i(x_k + s - ts) - \nabla F_i(x_k + s), -s \rangle dt, \quad \text{and}$$

$$336 \quad I_{4,j} := \int_0^1 \langle \nabla q_{k,i}(x_k + s - ts) - \nabla q_{k,i}(x_k + s), -s \rangle dt.$$

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

$$339 \quad (2.13) \quad \langle e_{\nabla q}(s), d_j - s \rangle = I_{1,j} - I_{2,j} - e_q(s) - F_i(x_k + d_j) + q_{k,i}(x_k + d_j),$$

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

$$341 \quad \langle e_{\nabla q}(s), d_j \rangle = I_{1,j} - I_{2,j} - I_{3,j} + I_{4,j}$$

$$342 \quad (2.14) \quad - F_i(x_k + d_j) + q_{k,i}(x_k + d_j) + F_i(x_k) - q_{k,i}(x_k).$$

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

$$346 \quad |I_{1,j}| \leq \int_0^1 \|\nabla F_i(x_k + s + t(d_j - s)) - \nabla F_i(x_k + s)\|_{\text{tr}^*} \|d_j - s\|_{\text{tr}} dt$$

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

Similarly, under Assumptions 2.1 and 2.2 it follows that

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

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

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

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

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

(2.12) follows since s was chosen arbitrarily in $B_{\text{tr}}(0; \Delta_k)$ 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

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

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

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

$$(2.19) \quad m_k(x) = h(q_k(x_k)) + \nabla h(q_k(x_k))^T \nabla q_k(x_k)^T (x - x_k) + \frac{1}{2} (x - x_k)^T \left(\sum_{i=1}^p \partial_i h(q_k(x_k)) \nabla^2 q_{k,i}(x_k) + \nabla q_k(x_k) \nabla^2 h(q_k(x_k)) \nabla q_k(x_k)^T \right) (x - x_k).$$

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 $\kappa_{\text{bhm}} \in \mathbb{R}_{>0}$ such that $\|\nabla^2 m_k(x_k)\|_{(\text{tr}, \text{tr}^*)} \leq \kappa_{\text{bhm}}$ for all $k \in \mathbb{N}$.*

Proof. Under Assumption 2.2, it follows that $\|\nabla^2 q_{k,i}(x_k)\|_{(\text{tr}, \text{tr}^*)} \leq L_{\nabla q_i}$ 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

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

as desired. \square

387 LEMMA 2.6. *Suppose that Assumptions 2.1 and 2.2 hold and consider arbitrary*
 388 *$k \in \mathbb{N}$ and $\mathcal{D}_k := \{d_1, \dots, d_{n_x}\} \subset B_{\text{tr}}(0, \Delta_k) \cap (\Omega - x_k)$ such that (a) for all $i \in [p]$, the*
 389 *model $q_{k,i}$ satisfies (2.6) on $\mathcal{D}_k \cup \{0\}$, and (b) the matrix $[d_1 \ \dots \ d_{n_x}]$ is invertible*
 390 *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*
 391 *(2.19) satisfies (2.8) in $B(x_k, \Delta_k) \cap \Omega$ for some $(\kappa_{\text{ef}}, \kappa_{\text{eg}}) \in \mathbb{R}_{>0}^2$ independent of k .*

392 *Proof.* Consider an arbitrary index $k \in \mathbb{N}$, set $\mathcal{D}_k := \{d_1, \dots, d_{n_x}\} \subset B_{\text{tr}}(0, \Delta_k) \cap$
 393 $(\Omega - x_k)$ satisfying the conditions of the lemma, point $x = x_k + s$ for some $\|s\|_{\text{tr}} \leq \Delta_k$
 394 such that $x \in \Omega$, and $i \in [p]$. For brevity, let us drop the dependence of F and F_i on
 395 θ ; i.e., let us use $F(x)$ and $F_i(x)$ to represent $F(x, \theta)$ and $F_i(x, \theta)$, respectively. To
 396 show the first bound in (2.8), observe by the Mean Value Theorem that

$$397 \quad (2.20) \quad f(x) = f(x_k) + \nabla f(x_k + \tau_1 s)^T s$$

$$398 \quad = h(F(x_k)) + \nabla h(F(x_k + \tau_1 s))^T \nabla F(x_k + \tau_1 s)^T s \text{ for some } \tau_1 \in [0, 1].$$

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

$$401 \quad (2.21) \quad m_k(x) = h(q_k(x_k)) + \nabla h(q_k(x_k + \tau_2 s))^T \nabla q_k(x_k + \tau_2 s)^T s.$$

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

$$404 \quad (2.22) \quad |f(x) - m_k(x)| \leq |h(F(x_k)) - h(q_k(x_k))|$$

$$405 \quad + \|\nabla F(x_k + \tau_1 s) \nabla h(F(x_k + \tau_1 s)) - \nabla q_k(x_k + \tau_2 s) \nabla h(q_k(x_k + \tau_2 s))\|_{\text{tr}^*} \|s\|_{\text{tr}}.$$

408 To bound the first term on the right-hand side of (2.22), one finds from (2.6) and
 409 (2.10) and Assumptions 2.1 and 2.2 that

$$410 \quad (2.23) \quad |h(F(x_k)) - h(q_k(x_k))| \leq L_h \|F(x_k) - q_k(x_k)\|_2 \leq \kappa_2 \Delta_k^2$$

412 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
 413 $\Delta_k \leq \Delta_{\text{max}}$ for all $k \in \mathbb{N}$; in addition, by (1.1d), (2.4), (2.7), the fact that $|\tau_1 - \tau_2| \leq 1$,
 414 Lemma 2.4, and Assumptions 2.1 and 2.2 it follows that

$$415 \quad (2.24) \quad \|\nabla F(x_k + \tau_1 s) \nabla h(F(x_k + \tau_1 s)) - \nabla q_k(x_k + \tau_2 s) \nabla h(q_k(x_k + \tau_2 s))\|_{\text{tr}^*}$$

$$416 \quad \leq \kappa_{\text{tr}_1^*} \|\nabla F(x_k + \tau_1 s) - \nabla q_k(x_k + \tau_1 s)\|_{\text{tr}^*, 1} \|\nabla h(F(x_k + \tau_1 s))\|_2$$

$$417 \quad + \kappa_{\text{tr}_1^*} \|\nabla q_k(x_k + \tau_1 s) - \nabla q_k(x_k + \tau_2 s)\|_{\text{tr}^*, 1} \|\nabla h(q_k(x_k + \tau_2 s))\|_2$$

$$418 \quad + \kappa_{\text{tr}_1^*} \|\nabla q_k(x_k + \tau_1 s)\|_{\text{tr}^*, 1} \|\nabla h(F(x_k + \tau_1 s)) - \nabla h(q_k(x_k + \tau_2 s))\|_2$$

$$419 \quad \leq p \kappa_{\text{tr}_1^*} \hat{\kappa}_{\text{eg}} \|\nabla h(F(x_k + \tau_1 s))\|_2 \Delta_k + \kappa_{\text{tr}_1^*} L_{\nabla q} \|\nabla h(q_k(x_k + \tau_2 s))\|_2 \Delta_k$$

$$420 \quad + \kappa_{\text{tr}_1^*} L_{\nabla h} \|\nabla q_k(x_k + \tau_1 s)\|_{\text{tr}^*, 1} \|F(x_k + \tau_1 s) - F(x_k + \tau_2 s)\|_2$$

$$421 \quad + \kappa_{\text{tr}_1^*} L_{\nabla h} \|\nabla q_k(x_k + \tau_1 s)\|_{\text{tr}^*, 1} \|F(x_k + \tau_2 s) - q_k(x_k + \tau_2 s)\|_2 \leq \kappa_3 \Delta_k;$$

423 where $\kappa_3 := \kappa_{\text{tr}_1^*} (\kappa_{\nabla h} (p \hat{\kappa}_{\text{eg}} + L_{\nabla q}) + L_{\nabla h} \kappa_{\nabla q} (\bar{L}_{F_x} + \sqrt{p} \hat{\kappa}_{\text{ef}} \Delta_{\text{max}}))$. From (2.22)–
 424 (2.24) it follows that the first bound in (2.8) holds for $\kappa_{\text{ef}} = \kappa_2 + \kappa_3$, as desired. Next,
 425 to show the second bound in (2.8), first observe by Taylor's Theorem that

$$426 \quad (2.25) \quad \nabla f(x) = \nabla F(x_k) \nabla h(F(x_k)) + \int_0^1 \nabla^2 f(x_k + \tau s) s d\tau,$$

428 where

429

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

Also, from (2.19), it follows that

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

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

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

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

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

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

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

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

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

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

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

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

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

as desired. \square

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

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

$$470 \quad (2.32) \quad |F_i(x_k + d, \theta) - \tilde{F}_i(x_k + d, \theta, \mathcal{H}_{k,i}, \delta_k)| \leq \kappa_{\text{app}} \delta_k \quad \text{for all } d \in \mathcal{D}_k \cup \{0\}.$$

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

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

491 Upon construction of m_k , the algorithm considers the stationary measure

$$492 \quad (2.33) \quad \pi_k^m = \left| \min_{\substack{x_k + d \in \Omega \\ \|d\|_{\text{tr}} \leq 1}} \nabla m_k(x_k)^T d \right|.$$

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

$$501 \quad (2.34) \quad m_k(x_k) - m_k(x_k + s_k) \geq \kappa_{\text{fcd}} \pi_k^m \min \left\{ \frac{\pi_k^m}{\kappa_{\text{bhm}} + 1}, \Delta_k, 1 \right\}$$

503 is achieved for some user-prescribed $\kappa_{\text{fcd}} \in (0, 1]$, where $\kappa_{\text{bhm}} \in \mathbb{R}_{>0}$ is defined as
 504 in Lemma 2.5. Sufficient conditions on s_k to achieve (2.34) and an algorithm for
 505 obtaining such a step can be found in [8]. After calculating the step s_k , the trial point
 506 $x_k + s_k$ is considered. According to the actual-to-predicted reduction ratio, namely,

$$507 \quad (2.35) \quad \rho_k := \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)},$$

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

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

513 The main difference between our algorithm and other modern model-based DFO
 514 frameworks is that function value approximations may be used at interpolation points
 515 in place of exact function values. These approximations are obtained using function
 516 values computed at nearby points, either in earlier runs of the algorithm (when solving
 517 a related optimization problem) or in earlier iterations of the current run of the
 518 algorithm. Specifically, in each iteration $k \in \mathbb{N}$, given the prior information $\mathcal{H}_{k,i}$,
 519 the algorithm attempts to approximate $F_i(x_k + d, \theta)$ for all $d \in \mathcal{D}_k$ such that the
 520 approximation error is sufficiently small. If there is not enough information available
 521 for approximating $F_i(x_k + d, \theta)$ for some $d \in \mathcal{D}_k$, then one may need to evaluate the
 522 black-box function F_i at $x_k + d$, which is information that can be used to augment
 523 $\mathcal{H}_{k,i}$. See Section 3 for further details on our function approximation strategy.

524 Another important feature of our framework is that one can also exploit prior
 525 function value information when determining the set of points to include in the in-
 526 terpolation set \mathcal{D}_k . Contemporary model-based DFO methods that do not exploit
 527 prior function evaluations often choose the interpolation set so that it is well-posed
 528 [6, 13], for which it is sufficient to ensure that (2.9) holds. For example, [7, 33] provide
 529 techniques for obtaining well-posed sets of interpolation points. For our purposes,
 530 however, beyond satisfying (2.9), our aim is to determine an interpolation set that
 531 exploits prior function evaluations when possible. Toward this end, we define a utility
 532 function to assess the potential usefulness of points in $\{\mathcal{H}_{k,i}\}$ that are contained in
 533 $B_{\text{tr}}(x_k, \Delta_k) \cap \Omega$. In Section 4, we present our utility function and a complete algorithm
 534 (Algorithm 4.1) for constructing the interpolation set such that (2.9) is guaranteed in
 535 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); $\epsilon_c \in (0, \infty)$; $\kappa_{\text{app}} \in (0, \infty)$;
 $\bar{\kappa}_{\text{app}} \in (0, \infty)$; $c_{\text{app}} \in [0, \bar{\kappa}_{\text{app}}/\kappa_{\text{app}}]$; $\kappa_{\text{fed}} \in (0, 1]$; $u_{\text{thr}} \in [0, \infty)$.

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

- 1: **for** $k = 0, 1, \dots$ **do**
 - 2: Set $\delta_k \leftarrow c_{\text{app}} \Delta_k^2$.
 - 3: 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 $\{\bar{F}_i(x_k + d, \theta, \mathcal{H}_{k,i}, \delta_k)\}_{i \in [p], d \in \mathcal{D}_k \cup \{0\}}$ yield
 $\{q_{k,i}\}_{i=1}^p$ and m_k satisfying (2.6), (2.32), Assumption 2.2, and (2.19).
 - 4: **if** $\pi_k^m \leq \epsilon_c$ and $\Delta_k > \mu \pi_k^m$ **then**
 - 5: Criticality step: set $s_k = 0$, $\rho_k = 0$, $\Delta_{k+1} \leftarrow \gamma_{\text{dec}} \Delta_k$, and $x_{k+1} \leftarrow x_k$.
 - 6: **else**
 - 7: Compute s_k satisfying (2.34).
 - 8: Evaluate $F_i(x_k + s_k, \theta)$ and augment $\mathcal{H}_{k,i}$ for all $i \in [p]$.
 - 9: Compute ρ_k by (2.35).
 - 10: **if** $\rho_k \geq \eta$ **then**
 - 11: Successful step: set $\Delta_{k+1} \leftarrow \min\{\gamma_{\text{inc}} \Delta_k, \Delta_{\max}\}$ and $x_{k+1} \leftarrow x_k + s_k$.
 - 12: **else**
 - 13: Unsuccessful step: set $\Delta_{k+1} \leftarrow \gamma_{\text{dec}} \Delta_k$ and $x_{k+1} \leftarrow x_k$.
 - 14: Set $\mathcal{H}_{k+1,i} \leftarrow \mathcal{H}_{k,i}$ for all $i \in [p]$.
-

536 **2.3. Convergence Result.** A convergence result for Algorithm 2.1 follows in a
 537 similar manner as for [13, Algorithm 10.1] and [6, Algorithm 1]. We state the result
 538 in terms of a stationarity measure for the minimization of f over Ω that is similar to
 539 the previously defined measure for m_k (recall (2.33)), namely,

$$540 \quad (2.36) \quad \pi_k^f = \left| \min_{\substack{x_k+d \in \Omega \\ \|d\|_{\text{tr}} \leq 1}} \nabla f(x_k)^T d \right|;$$

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

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

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

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

548 **3. Function Approximation.** Our general DFO framework presented in Algo-
 549 rithm 2.1 makes use of approximate function values for building interpolation models;
 550 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

$$551 \quad \tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) \approx F_i(x, \theta) \quad \text{for all } i \in [p].$$

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

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

567 **3.1. Least-Squares Optimization Problems.** Recall problem (1.5) and con-
 568 sider arbitrary $i \in [p]$. Let us also assume the following about the function ϕ .

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

$$571 \quad |\phi(x, w) - \phi(\bar{x}, \bar{w})| \leq L_\phi \left\| \begin{bmatrix} x - \bar{x} \\ w - \bar{w} \end{bmatrix} \right\|_{\text{app}}$$

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

573 Under Assumption 3.1, computing $\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) \approx F_i(x, \theta) = \phi(x, w_i)$ satisfying
 574 the requirements of Algorithm 2.1 requires a sufficient amount of function value in-
 575 formation in \mathcal{H}_i (defined in (1.6)) with respect to F_i . We capture such information
 576

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

$$578 \quad (3.1) \quad \mathcal{A}_i(x, w_i, \delta) = \left\{ (\bar{x}, \bar{w}, \phi(\bar{x}, \bar{w})) \in \mathcal{H}_i : \left\| \begin{bmatrix} \bar{x} - x \\ \bar{w} - w_i \end{bmatrix} \right\|_{\text{app}} \leq \delta \right\}.$$

580 For notational convenience, let us explicitly express the finite set $\mathcal{A}_i(x, w_i, \delta) =$
 581 $\{(u_j, \phi_j)\}_{j=1}^N$, 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
 582 $\mathcal{A}_i(x, w_i, \delta)$ for some $i \in [p]$, let us now describe a procedure for computing an ap-
 583 proximation $\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) \approx F_i(x, \theta) = \phi(x, w_i)$ satisfying the requirements in Algo-
 584 rithm 2.1. In particular, with parameters $\alpha^T = [\alpha_0 \quad \alpha_1^T \quad \alpha_2^T]$, let us express

$$585 \quad \tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) = \alpha_0 + \alpha_1^T x + \alpha_2^T w_i.$$

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

$$589 \quad (3.2) \quad \min_{\alpha \in \mathbb{R}^{n_u+1}} \frac{1}{2} \sum_{j=1}^N (\alpha_0 + \alpha_1^T x_j + \alpha_2^T w_j - \phi_j)^2 + \frac{\lambda}{2} (\|\alpha_1\|_2^2 + \|\alpha_2\|_2^2).$$

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

$$595 \quad (3.3) \quad M := \begin{bmatrix} 1 & \cdots & 1 \\ u_1 & \cdots & u_N \end{bmatrix}^T \in \mathbb{R}^{N \times (n_u+1)} \quad \text{and} \quad \phi := [\phi_1 \quad \cdots \quad \phi_N]^T \in \mathbb{R}^N.$$

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

$$598 \quad (3.4) \quad \min_{\alpha \in \mathbb{R}^{n_u+1}} \frac{1}{2} \|M\alpha - \phi\|_2^2 + \frac{\lambda}{2} (\|\alpha_1\|_2^2 + \|\alpha_2\|_2^2).$$

599 Defining $\bar{I} := I - \mathbf{e}_1 \mathbf{e}_1^T$, one finds that the unique solution of this problem is given by
 600 $\alpha = (M^T M + \lambda \bar{I})^{-1} M^T \phi$, from which it follows that the desired approximation is
 601 obtained by solving (3.4), then computing

$$602 \quad (3.5) \quad \tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) := [1 \quad u^T] \alpha = \beta^T \phi \approx F_i(x, \theta),$$

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

$$605 \quad (3.6) \quad \beta^T = [1 \quad u^T] (M^T M + \lambda \bar{I})^{-1} M^T.$$

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

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

$$612 \quad [1 \quad (u - u_c)^T] (M_c^T M_c + \lambda \bar{I})^{-1} M_c^T = [1 \quad u^T] (M^T M + \lambda \bar{I})^{-1} M^T,$$

614 where

$$615 \quad M_c := \begin{bmatrix} 1 & \cdots & 1 \\ u_1 - u_c & \cdots & u_N - u_c \end{bmatrix}^T.$$

617 *Proof.* Defining $C := \bar{I} + \mathbf{e}_1 [1 \quad -u_c^T]$, one finds that $M_c = MC$; thus,

$$\begin{aligned} 618 & \quad [1 \quad (u - u_c)^T] (M_c^T M_c + \lambda \bar{I})^{-1} M_c^T \\ 619 & = [1 \quad u^T] C (C^T (M^T M + \lambda \bar{I}) C)^{-1} C^T M^T = [1 \quad u^T] (M^T M + \lambda \bar{I})^{-1} M^T, \end{aligned}$$

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

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

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

625 *Proof.* Let $u_c = u$, and $M_c = M - \mathbf{1} [0 \quad u^T]$. By Lemma 3.2, one finds

$$626 \quad \beta^T = [1 \quad u^T] (M^T M + \lambda \bar{I})^{-1} M^T = \mathbf{e}_1^T (M_c^T M_c + \lambda \bar{I})^{-1} M_c^T.$$

628 Hence, the result follows as long as $\mathbf{e}_1^T (M_c^T M_c + \lambda \bar{I})^{-1} M_c^T \mathbf{1} = 1$. To see this, ob-
629 serve that since the first column of M_c is $\mathbf{1}$, one finds $(M_c^T M_c + \lambda \bar{I}) \mathbf{e}_1 = M_c^T \mathbf{1}$.
630 Multiplying both sides on the left by $(M_c^T M_c + \lambda \bar{I})^{-1}$, one concludes that $\mathbf{e}_1 =$
631 $(M_c^T M_c + \lambda \bar{I})^{-1} M_c^T \mathbf{1}$. Thus, $\mathbf{e}_1^T (M_c^T M_c + \lambda \bar{I})^{-1} M_c^T \mathbf{1} = \mathbf{e}_1^T \mathbf{e}_1 = 1$, as desired. \square

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

633 **LEMMA 3.4.** *Letting $\sigma_{\min} \in \mathbb{R}_{\geq 0}$ be the minimum singular value of $M_d^T M_d$, where
634 $M_d := [u_1 - u \quad \dots \quad u_N - u]^T$, it follows that $\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta)$ defined in (3.5) satisfies*

$$\begin{aligned} 635 \quad (3.7) \quad & \left| \tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) - F_i(x, \theta) \right| \leq \kappa_{\text{app}} \delta \\ & \text{where } \kappa_{\text{app}} := L_\phi \left(1 + \left(\frac{2N\kappa_{\text{app}_0^*}}{\sigma_{\min} + \lambda} \right) \delta \right). \end{aligned}$$

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

$$\begin{aligned} 639 \quad & \left| \tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) - F_i(x, \theta) \right| = \left| \beta^T \phi - F_i(x, \theta) \beta^T \mathbf{1} \right| \leq \sum_{j=1}^N |\beta_j| |\phi_j - F_i(x, \theta)| \\ 640 & = \sum_{j=1}^N |\beta_j| |\phi(x_j, w_j) - \phi(x, w_i)| \leq L_\phi \|\beta\|_1 \left\| \begin{bmatrix} x_j - x \\ w_j - w_i \end{bmatrix} \right\|_{\text{app}} \\ 641 \quad (3.8) & \leq L_\phi \|\beta\|_1 \delta, \end{aligned}$$

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

$$\begin{aligned} 645 & \quad (M_c^T M_c + \lambda \bar{I})^{-1} = \begin{bmatrix} \frac{1}{N} & \mathbf{0}^T \\ \mathbf{0} & (M_d^T M_d + \lambda I)^{-1} \end{bmatrix}. \\ 646 & \end{aligned}$$

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

$$648 \quad \beta^T = [1 \quad (u - u_c)^T] (M_c^T M_c + \lambda \bar{I})^{-1} M_c^T = \frac{1}{N} \mathbf{1}^T + (u - u_c)^T (M_d^T M_d + \lambda I)^{-1} M_d^T;$$

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

$$651 \quad |\beta_j| = \left| \frac{1}{N} + (u - u_c)^T (M_d^T M_d + \lambda I)^{-1} (u_j - u) \right|$$

$$\begin{aligned}
652 \quad & \leq \frac{1}{N} + \|u - u_c\|_{\text{app}} \|(M_d^T M_d + \lambda I)^{-1} (u_j - u)\|_{\text{app}^*} \\
653 \quad & \leq \frac{1}{N} + \|(M_d^T M_d + \lambda I)^{-1} (u_j - u)\|_{\text{app}^*} \delta \\
654 \quad & \leq \frac{1}{N} + \kappa_{\text{app}^*} \|u_j - u\|_{\text{app}} \|(M_d^T M_d + \lambda I)^{-1}\|_{(2,2)} \delta = \frac{1}{N} + \left(\frac{2\kappa_{\text{app}^*}}{\sigma_{\min} + \lambda} \right) \delta^2. \\
655
\end{aligned}$$

656 Therefore, it follows that

$$657 \quad \|\beta\|_1 = \sum_{j=1}^N |\beta_j| \leq 1 + \left(\frac{2N\kappa_{\text{app}^*}}{\sigma_{\min} + \lambda} \right) \delta^2, \\
658$$

659 which combined with (3.8) yields the desired conclusion. \square

660 **3.2. Multi-Output Simulation.** Recall the multi-output simulation optimiza-
661 tion setting with \mathcal{H}_i defined in (1.7) and consider arbitrary $i \in [p]$. The set of available
662 information for approximating $F_i(x, \theta)$ for some $\delta \in \mathbb{R}_{>0}$ is given by

$$663 \quad (3.9) \quad \mathcal{A}_i(x, \theta, \delta) = \left\{ (\bar{x}, \bar{\theta}, F_i(\bar{x}, \bar{\theta})) \in \mathcal{H}_i : \left\| \begin{bmatrix} \bar{x} - x \\ \bar{\theta} - \theta \end{bmatrix} \right\|_{\text{app}} \leq \delta \right\}. \\
664$$

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

667 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 $\phi_j =$
668 $F_i(x_j, \theta_j)$. Similar to the least-squares case, we propose setting $\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) = \alpha_0 +$
669 $\alpha_1^T x + \alpha_2^T \theta$ for some $\alpha = [\alpha_0 \quad \alpha_1^T \quad \alpha_2^T]$. Let us define M and ϕ as in (3.3), but with
670 u_j and ϕ_j defined as above for all $j \in [N]$. One can obtain α by solving a problem of
671 the form (3.4), and as a result, for $u = (x, \theta)$, an approximation for $F_i(x, \theta)$ is given
672 by (3.5). Likewise, Lemma 3.2 and 3.3 hold. Finally, a result similar to Lemma 3.4
673 can be obtained if the following assumption is made.

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

$$676 \quad |F_i(x, \theta) - F_i(\bar{x}, \bar{\theta})| \leq L_{F_i} \left\| \begin{bmatrix} x - \bar{x} \\ \theta - \bar{\theta} \end{bmatrix} \right\|_{\text{app}} \\
677$$

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

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

681 **LEMMA 3.6.** *Letting $\sigma_{\min} \in \mathbb{R}_{>0}$ be the minimum singular value of $M_d^T M_d$, where*
682 $M_d := [u_1 - u \quad \dots \quad u_N - u]^T$, *it follows that $\tilde{F}_i(x, \theta, \mathcal{H}_i, \delta)$ defined in (3.5) satisfies*

$$683 \quad \left| \tilde{F}_i(x, \theta, \mathcal{H}_i, \delta) - F_i(x, \theta) \right| \leq \kappa_{\text{app}} \delta \text{ where } \kappa_{\text{app}} := L_{F_i} \left(1 + \left(\frac{2N\kappa_{\text{app}^*}}{\sigma_{\min} + \lambda} \right) \delta \right). \\
684$$

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

689 $x_U \in (\mathbb{R} \cup \{\infty\})^{n_x}$ with $x_{L,j} < x_{U,j}$ for all $j \in [n_x]$. Our strategy involves first run-
 690 ning a modified version of [33, Algorithm 4.1], which is presented as Algorithm 4.1
 691 below. If the output of this procedure is a set of $n_x + 1$ points, then the set is com-
 692 plete and we show that (2.9) is guaranteed to hold. Otherwise, the algorithm that we
 693 present as Algorithm 4.2 in this section can be called iteratively until a total of $n_x + 1$
 694 points have been obtained, where again we show that (2.9) is guaranteed to hold. In
 695 either case, after obtaining $n_x + 1$ points, we call [33, Algorithm 4.2] to potentially
 696 add additional points to the interpolation set and construct a quadratic model.

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

$$702 \quad (4.1) \quad \mathcal{E} \leftarrow \{x \in B_{\text{tr}}(x_k, \Delta_k) \cap \Omega : (x, w, \phi(x, w)) \in \{\mathcal{H}_{k,i}\}, u(x, \theta, \{\mathcal{H}_{k,i}\}) \geq u_{\text{thr}}\},$$

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

$$706 \quad (4.2) \quad \mathcal{E} \leftarrow \{x \in B_{\text{tr}}(x_k, \Delta_k) \cap \Omega : (x, \theta, F_i(x, \theta)) \in \{\mathcal{H}_{k,i}\}, u(x, \theta, \{\mathcal{H}_{k,i}\}) \geq u_{\text{thr}}\}.$$

708 The set \mathcal{E} is then ordered according to the utility function u , the definition of which
 709 may be problem formulation dependent. For example, in the context of least-squares
 710 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
 711 define the utility as the fraction of element functions $\{\phi(\cdot, w_i)\}$ that can be approxi-
 712 mated with the desired accuracy δ_k (see line 2 of Algorithm 2.1) at x . On the other
 713 hand, for multi-output simulation optimization, since performing one simulation gener-
 714 ates p outputs simultaneously, at any such point x , either all element functions can
 715 be well approximated or there is not enough information available for approximating
 716 any of them. A good choice for the utility function in this case may be a function
 717 that estimates the accuracy of the approximation at x . As discussed in Section 3, the
 718 accuracy of the approximation at x depends on the number of nearby points as well
 719 as the geometry of the nearby points (see Lemma 3.6).

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

$$722 \quad (4.3) \quad [\mathcal{D}_k]_{1:\text{end}} = \text{matrix composed of vectors in } \mathcal{D}_k \text{ except } 0.$$

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

725 If Algorithm 4.1 returns an interpolation set with $|\mathcal{D}_k| = n_x + 1$, then (2.9)
 726 holds (as we show in Theorem 4.8). Otherwise, additional points need to be added.
 727 Under the assumption that the (potentially infinite) bound constraints defined by
 728 $\Omega = [x_L, x_U]$ might not be relaxable, we propose Algorithm 4.2, which employs the
 729 subroutine in Algorithm 4.3, for augmenting the set \mathcal{D}_k in a manner that ensures that
 730 (2.9) continues to hold. This algorithm can be called iteratively until $|\mathcal{D}_k| = n_x + 1$.

731 Consider \mathcal{D}_k with $|\mathcal{D}_k| < n_x + 1$ obtained after a call to Algorithm 4.1 and
 732 (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
 733 $n_z \geq 1$. Following the spirit of Algorithm 4.1, one can augment \mathcal{D}_k with $d \in \mathbb{R}^{n_x}$ in
 734 a manner that guarantees that (2.9) continues to hold by ensuring that

$$735 \quad (4.4) \quad \left\| \text{proj}_Z \left(\frac{d}{\Delta_k} \right) \right\|_2 \geq \xi, \quad \|d\|_{\text{tr}} \leq \Delta_k, \quad \text{and} \quad x_k + d \in \Omega.$$

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

Require: Values and parameters from Algorithm 2.1.

- 1: Set $\mathcal{D}_k \leftarrow \{0\}$ and $Z \leftarrow I_{n_x}$.
 - 2: 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)).
 - 3: Sort \mathcal{E} as $\{x^{(1)}, \dots, x^{(|\mathcal{E}|)}\}$ so $u(x^{(j)}, \theta, \{\mathcal{H}_{k,i}\}) \geq u(x^{(j+1)}, \theta, \{\mathcal{H}_{k,i}\})$ for $j \in [|\mathcal{E}|]$.
 - 4: **for** $j = 1, \dots, |\mathcal{E}|$ **do**
 - 5: **if** $\left\| \text{proj}_Z \left(\frac{x^{(j)} - x_k}{\Delta_k} \right) \right\|_2 \geq \xi$ **then**
 - 6: Set $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{x^{(j)} - x_k\}$.
 - 7: Compute orthonormal Z with $\text{span}(Z) = \text{null}([\mathcal{D}_k]_{1:\text{end}})$ (see (4.3)).
 - 8: **return** (\mathcal{D}_k, Z)
-

736

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

$$739 \quad (4.5) \quad \max_{d \in \mathbb{R}^n} \Upsilon(Z, d) \quad \text{s.t.} \quad \|d\|_{\text{tr}} \leq \Delta_k \quad \text{and} \quad x_L - x_k \leq d \leq x_U - x_k,$$

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

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

$$746 \quad (4.6) \quad \max_{(v,i,\tau) \in \mathbb{R}^{n_x} \times [n_z] \times \mathbb{R}_{\geq 0}} \Upsilon(Z, d(v, \tau)) \quad \text{s.t.} \quad v \in \{-z_i, z_i\} \quad \text{and} \quad \|d(v, \tau)\|_{\text{tr}} \leq \Delta_k,$$

748 where z_i is the i th column of Z and where for any $(i, \tau) \in [n_z] \times \mathbb{R}_{\geq 0}$ and $v \in \{-z_i, z_i\}$
 749 the vector $x_k + d(v, \tau)$ is the projection of $x_k + \tau v$ onto $\Omega = [x_L, x_U]$. A formula for
 750 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

$$751 \quad (4.7) \quad \bar{\tau}_j(v) := \begin{cases} \frac{x_{U,j} - x_{k,j}}{v_j} & \text{if } v_j > 0 \text{ and } x_{U,j} < \infty, \\ \frac{x_{L,j} - x_{k,j}}{v_j} & \text{if } v_j < 0 \text{ and } x_{L,j} > -\infty, \\ \infty & \text{otherwise} \end{cases}$$

753 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

$$754 \quad (4.8) \quad d(v, \tau) = [\min\{\tau, \bar{\tau}_1(v)\}v_1 \quad \dots \quad \min\{\tau, \bar{\tau}_{n_x}(v)\}v_{n_x}]^T.$$

756 *Remark 4.1.* In (4.8) and subsequent calculations below, we interpret equations
 757 and operations involving infinite quantities in the following natural ways: $\infty = \infty$;
 758 $\infty \times \infty = \infty$; $\infty - (-\infty) = \infty$; $\min\{\infty, \infty\} = \infty$; $b - \infty = -\infty$ and $b + \infty = \infty$ for
 759 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
 760 norm $\|\cdot\|$ and any $v \in (\mathbb{R} \cup \{-\infty, \infty\})^{n_x}$ that has an element equal to $-\infty$ or ∞ .
 761 One additional rule that we intend, which is not natural in all contexts, is $0 \times \infty = 0$.

762 Algorithm 4.2 operates by iterating through the columns of Z , where for each
 763 $i \in [n_z]$ the optimal values of $\tau \in \mathbb{R}_{\geq 0}$ (with respect to maximizing Υ) are determined
 764 along z_i and $-z_i$. If the search along either direction yields a larger value of Υ than

765 has been observed so far, the solution estimate (with respect to (4.6)) is updated.
 766 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 $Z \in \mathbb{R}^{n_x \times n_z}$.

- 1: Set $d \leftarrow \mathbf{0}$.
 - 2: **for** $i = 1, \dots, n_z$ **do**
 - 3: $\tau^+ \leftarrow \text{OPTSTEP}(Z, z_i, \bar{\tau}(z_i))$.
 - 4: **if** $\Upsilon(Z, d(z_i, \tau^+)) > \Upsilon(Z, d)$ **then** set $d \leftarrow d(z_i, \tau^+)$.
 - 5: $\tau^- \leftarrow \text{OPTSTEP}(Z, -z_i, \bar{\tau}(-z_i))$.
 - 6: **if** $\Upsilon(Z, d(-z_i, \tau^-)) > \Upsilon(Z, d)$ **then** set $d \leftarrow d(-z_i, \tau^-)$.
 - 7: Set $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{d\}$.
 - 8: Compute orthonormal Z with $\text{span}(Z) = \text{null}([\mathcal{D}_k]_{1:\text{end}})$ (see (4.3)).
 - 9: **return** (\mathcal{D}_k, Z) .
-

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

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

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

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

772 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 $\tau \geq \frac{\Delta_k}{\kappa_{\text{tr}1}^{\text{tr}}}$.*

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

$$774 \quad \Delta_k^{\text{tr}} = \|d(v, \tau)\|_{\text{tr}}^{\text{tr}} = \sum_{j=1}^{n_x} (\min\{\tau, \bar{\tau}_j(v)\})^{\text{tr}} |v_j|^{\text{tr}} \leq \tau^{\text{tr}} \|v\|_{\text{tr}}^{\text{tr}} \leq \tau^{\text{tr}} \kappa_{\text{tr}1}^{\text{tr}} \|v\|_2^{\text{tr}} = \tau^{\text{tr}} \kappa_{\text{tr}1}^{\text{tr}},$$

775

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

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

$$779 \quad (4.9) \quad j^*(v) \in \arg \max_{j \in [n_x]} |v_j|.$$

780

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

782 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*

$$783 \quad \max\{\bar{\tau}_{j^*(v)}(-v), \bar{\tau}_{j^*(v)}(v)\} \geq \frac{x_{U, j^*(v)} - x_{L, j^*(v)}}{2|v_{j^*(v)}|}.$$

784

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

787 LEMMA 4.4. Consider line 3 in Algorithm 4.2 for any $i \in [n_z]$. If it is found
788 within the call to `OptStep`($Z, z_i, \bar{\tau}(z_i)$) that $\|d(z_i, \bar{\tau}_{(\hat{n}_x)}(z_i))\|_{\text{tr}} \leq \Delta_k$, then $\tau^+ \in$
789 $\mathbb{R}_{\geq 0}$ 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
790 `OptStep`($Z, -z_i, \bar{\tau}(-z_i)$) with respect to $\tau^- \in \mathbb{R}_{\geq 0}$.

791 *Proof.* Without loss of generality, consider the call to `OptStep`($Z, z_i, \bar{\tau}(z_i)$). The
792 proof for the call to `OptStep`($Z, -z_i, \bar{\tau}(-z_i)$) is nearly identical. By the conditions of
793 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

$$\begin{aligned} 794 \quad \Upsilon(Z, d(z_i, \tau^+)) &\geq \Upsilon(Z, d(z_i, \bar{\tau}_{(\hat{n}_x)})) = \sum_{l=1}^{n_z} (z_l^T d(z_i, \bar{\tau}_{(\hat{n}_x)}))^2 \\ 795 &\geq \left(\sum_{j=1}^{n_x} [z_i]_j d_j(z_i, \bar{\tau}_{(n_x)}) \right)^2 = \left(\sum_{j=1}^{n_x} [z_i]_j^2 \min\{\bar{\tau}_{(n_x)}, \bar{\tau}_j(z_i)\} \right)^2 \\ 796 &\geq [z_i]_{j^*(z_i)}^4 \min\{\bar{\tau}_{(n_x)}, \bar{\tau}_{j^*(z_i)}(z_i)\}^2 \geq \frac{\bar{\tau}_{j^*(z_i)}(z_i)^2}{n_x^2}, \\ 797 \end{aligned}$$

798 as desired. \square

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

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

$$803 \quad \Upsilon(Z, d(z_i, \tau^+)) \geq \left(\min \left\{ \frac{\Delta_k}{\kappa_{\text{tr}_1}}, \bar{\tau}_{j^*(z_i)}(z_i) \right\} \right)^2 \frac{1}{n_x^2}.$$

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

805 *Proof.* The proof is nearly identical to that of Lemma 4.4, except with the com-
806 puted value $\hat{\tau}$ in place of $\bar{\tau}_{(n_x)}$, where by Lemma 4.2 one has that $\hat{\tau} \geq \frac{\Delta_k}{\kappa_{\text{tr}_1}}$. \square

807 We now prove our main result of this section.

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

$$809 \quad \xi \in \left(0, \frac{1}{n_x} \min \left\{ \frac{1}{\kappa_{\text{tr}_1}}, \frac{\min_{j \in [n_x]} (x_{U,j} - x_{L,j})}{2\Delta_{\text{max}}} \right\} \right],$$

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

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

$$815 \quad \frac{1}{\Delta_k} \Upsilon_{i, \text{max}} := \frac{1}{\Delta_k} \max \left\{ \sqrt{\Upsilon(Z, d(z_i, \tau^+))}, \sqrt{\Upsilon(Z, d(-z_i, \tau^-))} \right\} \geq \xi.$$

817 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
818 step sizes yield points within the trust region, then Lemmas 4.3 and 4.4 imply

$$819 \quad \frac{1}{\Delta_k} \Upsilon_{i, \text{max}} \geq \frac{1}{n_x \Delta_k} \max\{\bar{\tau}_{j^*(z_i)}(z_i), \bar{\tau}_{j^*(-z_i)}(-z_i)\}$$

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

Otherwise, if the call to `OptStep`($Z, z_i, \bar{\tau}(z_i)$) and/or `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

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

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

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

Combining the results of these cases, the desired conclusion follows. \square

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*

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

where r is the last diagonal of R in a QR factorization of $\frac{1}{\Delta_k} [D_k]_{1:\text{end}} \ d$.

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

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

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

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

as desired. \square

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 $\Lambda = \frac{n_x \frac{n_x - 1}{2} \kappa_{\text{tr}_0}^{n_x - 1}}{\xi^{n_x}}$.*

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

$$(4.10) \quad \|[D_k]_{1:\text{end}}^{-1}\|_2 = \frac{1}{\kappa_{\text{tr}_0} \Delta_k} \left\| \left(\frac{[D_k]_{1:\text{end}}}{\kappa_{\text{tr}_0} \Delta_k} \right)^{-1} \right\|_2 = \frac{1}{\sigma_1 \kappa_{\text{tr}_0} \Delta_k}.$$

855

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

$$859 \quad \sigma_{n_x} = \left\| \frac{[\mathcal{D}_k]_{1:\text{end}}}{\kappa_{\text{tr}_0} \Delta_k} \right\|_2 \leq \left\| \frac{[\mathcal{D}_k]_{1:\text{end}}}{\kappa_{\text{tr}_0} \Delta_k} \right\|_F \leq \sqrt{n_x}.$$

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

$$861 \quad \sigma_1 n_x^{\frac{n_x-1}{2}} \geq \sigma_1 \sigma_{n_x}^{n_x-1} \geq \prod_{i=1}^{n_x} \sigma_i = \left| \det \left(\frac{[\mathcal{D}_k]_{1:\text{end}}}{\kappa_{\text{tr}_0} \Delta_k} \right) \right| = \left| \prod_{i=1}^{n_x} \frac{r_i}{\kappa_{\text{tr}_0}} \right| \geq \left(\frac{\xi}{\kappa_{\text{tr}_0}} \right)^{n_x}.$$

862 These inequalities along with (4.10) yields

$$864 \quad \left\| [\mathcal{D}_k]_{1:\text{end}}^{-1} \right\|_2 \leq \frac{n_x^{\frac{n_x-1}{2}} \kappa_{\text{tr}_0}^{n_x-1}}{\xi^{n_x} \Delta_k} = \frac{\Lambda}{\Delta_k},$$

865 which is the desired conclusion. \square

867 **5. Numerical Experiments.** The purpose of our experiments is to demon-
 868 strate the reduction in function evaluations that can be achieved by our method
 869 through its exploitation of prior function evaluations when solving a sequence of re-
 870 lated problems. We compare Algorithm 2.1 as it is stated, referred to in this section as
 871 $\text{Alg}_{\mathcal{H}}$, and an algorithm that has all of the same features of Algorithm 2.1 except that
 872 does not utilize prior function evaluations, referred to as Alg_{\emptyset} . We choose $T = 100$,
 873 and for all $t \in \{0, \dots, T-1\}$ we run both algorithms until a fixed budget of two
 874 simplex gradient evaluations (i.e., $2p(n_x + 1)$ function evaluations) is exhausted. We
 875 consider such a relatively small amount of function evaluations because the advantage
 876 of our method is exploited best when solution time is crucial, such as an online setting,
 877 in which the true optimum is out of reach and a good approximate solution suffices.

878 We implemented our algorithm in Python and ran our experiments on a Linux
 879 workstation. We tested our algorithm on a variety of problems and the results were
 880 similar in all cases. For our purposes here, we present the results obtained from
 881 a single representative least-squares problem involving an ODE that describes the
 882 conversion of methanol into various hydrocarbons [17], which for parameters $x =$
 883 $[x_1 \ \dots \ x_5]^T \in \mathbb{R}_{\geq 0}^5$ and state $v(\tau) = [v_1(\tau) \ v_2(\tau) \ v_3(\tau)]^T \in \mathbb{R}^3$ is described
 884 (with τ denoting time) by:

$$885 \quad \frac{dv_1}{d\tau} = -\left(2x_2 - \frac{x_1 v_2}{(x_2 + x_5)v_1 + v_2} + x_3 + x_4\right)v_1;$$

$$886 \quad \frac{dv_2}{d\tau} = \frac{x_1 v_1 (x_2 v_1 - v_2)}{(x_2 + x_5)v_1 + v_2} + x_3 v_1;$$

$$887 \quad \frac{dv_3}{d\tau} = \frac{x_1 v_1 (v_2 + x_5 v_1)}{(x_2 + x_5)v_1 + v_2} + x_4 v_1.$$

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

890 We fix a vector $\bar{x} = [1.78 \ 2.17 \ 1.86 \ 1.80 \ 0]^T \in \mathbb{R}_{\geq 0}^5$, then, iteratively for
 891 each $t \in \{0, \dots, T-1\}$, we generate the data for the t th least-squares objective in the

892 following manner. First, we establish seven initial conditions by taking each of the
 893 following vectors, perturbing it by a realization of a random vector having a uniform
 894 distribution over a 2-norm ball with radius 0.1, then projecting the result onto the
 895 3-dimensional standard simplex so that the elements are nonnegative and sum to one:

$$896 \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{3}{4} \\ \frac{1}{4} \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{3}{4} \\ 0 \\ \frac{1}{4} \end{bmatrix}, \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix}, \begin{bmatrix} \frac{1}{4} \\ \frac{3}{4} \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{1}{4} \\ 0 \\ \frac{3}{4} \end{bmatrix} \right\}.$$

897 (This projection of the initial condition is meaningful for the application since the state
 898 elements correspond to proportions.) We denote the resulting l th initial condition by
 899 $v_{(l)}^0 \in \mathbb{R}_{\geq 0}^3$ for all $l \in [7]$. Second, we establish the time points $\tau_{(1)} = 0.1$, $\tau_{(2)} = 0.4$,
 900 and $\tau_{(3)} = 0.8$, which are fixed for all $t \in \{0, \dots, T-1\}$. At this point in the data
 901 generation for problem t , we have established $\{w_{i,t}\}_{i \in [21]}$, with each one corresponding
 902 to a given initial condition and time point; specifically, each $i \in [21]$ corresponds to a
 903 unique pair $(j, l) \in [3] \times [7]$, corresponding to which we define $w_{i,t} := \begin{bmatrix} \tau_{(j)} & (v_{(l)}^0)^T \end{bmatrix}^T$.
 904 All that remains to generate the data for problem t is to establish the values $\{y_{i,t}\}_{i \in [21]}$.
 905 For this, we first generate $x_{(t)} \in \mathbb{R}_{\geq 0}^5$ by adding to \bar{x} a realization from a uniform
 906 $[0, 1]^5$ distribution. Then, for all $i \in [21]$, we let $\phi(x_{(t)}, w_{i,t})$ denote the value of v_3
 907 from the ODE at time $[w_{i,j}]_0$ when using the initial condition $[w_{i,t}]_{1:3}$ and set

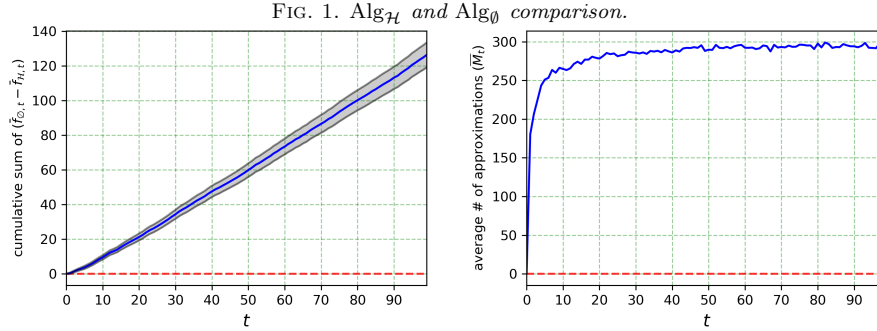
$$908 y_{i,t} \leftarrow \phi(x_{(t)}, w_{i,t}) + |\phi(x_{(t)}, w_{i,t})| u_{i,t},$$

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

914 To understand the typical behavior of our algorithm, we generated a total of
 915 $N = 100$ replications with $T = 100$ problems as described in the previous paragraph.
 916 Each repetition starts with problem $t = 0$, which involves no history of function
 917 evaluations, meaning that $\text{Alg}_{\mathcal{H}}$ and Alg_{\emptyset} always perform equivalently for $t = 0$.
 918 However, for all $t \in \{1, \dots, T-1\}$, $\text{Alg}_{\mathcal{H}}$ makes use of prior function evaluations
 919 when possible while Alg_{\emptyset} does not. Let $f_{\mathcal{H},t}^k$ and $f_{\emptyset,t}^k$, respectively, denote the final
 920 objective function value obtained by $\text{Alg}_{\mathcal{H}}$ and Alg_{\emptyset} when solving repetition k of
 921 problem t . Averaging over the repetitions, we obtain the values $\bar{f}_{\mathcal{H},t} = \frac{1}{N} \sum_{k \in [N]} f_{\mathcal{H},t}^k$
 922 and $\bar{f}_{\emptyset,t} = \frac{1}{N} \sum_{k \in [N]} f_{\emptyset,t}^k$ for all $t \in \{0, \dots, T-1\}$. In addition, to get a sense
 923 of $\text{Alg}_{\mathcal{H}}$'s ability to take more steps within the function evaluation limit by using
 924 approximated function values in place of true function values, we record M_t^k as the
 925 number of approximated function values used in repetition k of problem t . These are
 926 averaged over the repetitions to obtain $\bar{M}_t = \frac{1}{N} \sum_{k \in [N]} M_t^k$ for all $t \in \{0, \dots, T-1\}$.

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

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



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

950

REFERENCES

- 951 [1] C. AUDET AND J. E. DENNIS JR, *A pattern search filter method for nonlinear programming*
 952 *without derivatives*, SIAM Journal on Optimization, 14 (2004), pp. 980–1010.
 953 [2] C. AUDET AND J. E. DENNIS JR, *Mesh adaptive direct search algorithms for constrained opti-*
 954 *mization*, SIAM Journal on optimization, 17 (2006), pp. 188–217.
 955 [3] D. M. BATES AND D. G. WATTS, *Nonlinear regression analysis and its applications*, John Wiley
 956 & Sons, New York, 1988.
 957 [4] C. CARTIS, J. FIALA, B. MARTEAU, AND L. ROBERTS, *Improving the flexibility and robustness*
 958 *of model-based derivative-free optimization solvers*, ACM Transactions on Mathematical
 959 Software (TOMS), 45 (2019), pp. 1–41.
 960 [5] C. CARTIS AND L. ROBERTS, *A derivative-free gauss–newton method*, Mathematical Program-
 961 ming Computation, 11 (2019), pp. 631–674.
 962 [6] P. CONEJO, E. W. KARAS, L. G. PEDROSO, A. A. RIBEIRO, AND M. SACHINE, *Global conver-*
 963 *gence of trust-region algorithms for convex constrained minimization without derivatives*,
 964 Applied Mathematics and Computation, 220 (2013), pp. 324–330.
 965 [7] A. CONN, K. SCHEINBERG, AND P. TOINT, *A derivative free optimization algorithm in prac-*
 966 *tice*, in 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and
 967 Optimization, 1998, p. 4718.
 968 [8] A. R. CONN, N. GOULD, A. SARTENAER, AND P. L. TOINT, *Global convergence of a class of*
 969 *trust region algorithms for optimization using inexact projections on convex constraints*,
 970 SIAM Journal on Optimization, 3 (1993), pp. 164–221.
 971 [9] A. R. CONN, K. SCHEINBERG, AND P. L. TOINT, *On the convergence of derivative-free methods*
 972 *for unconstrained optimization*, Approximation theory and optimization: tributes to MJD
 973 Powell, (1997), pp. 83–108.

- 974 [10] A. R. CONN, K. SCHEINBERG, AND P. L. TOINT, *Recent progress in unconstrained nonlinear*
975 *optimization without derivatives*, Mathematical programming, 79 (1997), pp. 397–414.
- 976 [11] A. R. CONN, K. SCHEINBERG, AND L. N. VICENTE, *Geometry of interpolation sets in derivative*
977 *free optimization*, Mathematical programming, 111 (2008), pp. 141–172.
- 978 [12] A. R. CONN, K. SCHEINBERG, AND L. N. VICENTE, *Geometry of sample sets in derivative-free*
979 *optimization: polynomial regression and underdetermined interpolation*, IMA journal of
980 numerical analysis, 28 (2008), pp. 721–748.
- 981 [13] A. R. CONN, K. SCHEINBERG, AND L. N. VICENTE, *Introduction to derivative-free optimization*,
982 MPS/SIAM, Philadelphia, 2009.
- 983 [14] A. CUSTÓDIO, J. DENNIS, AND L. N. VICENTE, *Using simplex gradients of nonsmooth functions*
984 *in direct search methods*, IMA journal of numerical analysis, 28 (2008), pp. 770–784.
- 985 [15] J. E. DENNIS JR AND R. B. SCHNABEL, *Numerical methods for unconstrained optimization and*
986 *nonlinear equations*, SIAM, Philadelphia, 1996.
- 987 [16] S. L. DIGABEL AND S. M. WILD, *A taxonomy of constraints in simulation-based optimization*,
988 arXiv preprint arXiv:1505.07881, (2015).
- 989 [17] E. D. DOLAN, J. J. MORÉ, AND T. S. MUNSON, *Benchmarking optimization software with cops*
990 *3.0.*, tech. report, Argonne National Lab., Argonne, IL, 2004.
- 991 [18] G. A. GRAY AND T. G. KOLDA, *Appspack 4.0: asynchronous parallel pattern search for*
992 *derivative-free optimization.*, tech. report, Sandia National Lab., Livermore, CA, 2004.
- 993 [19] T. HASTIE, R. TIBSHIRANI, AND J. H. FRIEDMAN, *The Elements of Statistical Learning*,
994 Springer, New York, 2009.
- 995 [20] M. HOUGH AND L. ROBERTS, *Model-based derivative-free methods for convex-constrained opti-*
996 *mization*, arXiv preprint arXiv:2111.05443, (2021).
- 997 [21] M. MARAZZI AND J. NOCEDAL, *Wedge trust region methods for derivative free optimization*,
998 Mathematical programming, 91 (2002), pp. 289–305.
- 999 [22] J. J. MORÉ AND S. M. WILD, *Benchmarking derivative-free optimization algorithms*, SIAM
1000 Journal on Optimization, 20 (2009), pp. 172–191.
- 1001 [23] J. A. NELDER AND R. MEAD, *A simplex method for function minimization*, The computer
1002 journal, 7 (1965), pp. 308–313.
- 1003 [24] M. J. POWELL, *A direct search optimization method that models the objective and constraint*
1004 *functions by linear interpolation*, in Advances in optimization and numerical analysis,
1005 Springer, 1994, pp. 51–67.
- 1006 [25] M. J. POWELL, *Uobyqa: unconstrained optimization by quadratic approximation*, Mathematical
1007 Programming, 92 (2002), pp. 555–582.
- 1008 [26] M. J. POWELL, *Least Frobenius norm updating of quadratic models that satisfy interpolation*
1009 *conditions*, Mathematical Programming, 100 (2004), pp. 183–215.
- 1010 [27] M. J. POWELL, *The newuoa software for unconstrained optimization without derivatives*, in
1011 Large-scale nonlinear optimization, Springer, 2006, pp. 255–297.
- 1012 [28] M. J. POWELL, *The BOBYQA algorithm for bound constrained optimization without deriva-*
1013 *tives*, Cambridge NA Report NA2009/06, University of Cambridge, (2009), pp. 26–46.
- 1014 [29] R. G. REGIS AND S. M. WILD, *Conorbit: constrained optimization by radial basis function*
1015 *interpolation in trust regions*, Opt. Meth. & Soft.imization Methods and Software, 32
1016 (2017), pp. 552–580.
- 1017 [30] H.-J. M. SHI, M. Q. XUAN, F. OZTOPRAK, AND J. NOCEDAL, *On the numerical performance*
1018 *of derivative-free optimization methods based on finite-difference approximations*, arXiv
1019 preprint arXiv:2102.09762, (2021).
- 1020 [31] L. N. VICENTE AND A. L. CUSTÓDIO, *Analysis of direct searches for discontinuous functions*,
1021 Mathematical programming, 133 (2012), pp. 299–325.
- 1022 [32] S. M. WILD, *Derivative-Free Optimization Algorithms For Computationally Expensive Func-*
1023 *tions*, PhD dissertation, Cornell University, 2008.
- 1024 [33] S. M. WILD, *Mnh: A derivative-free optimization algorithm using minimal norm Hessians*,
1025 tech. report, Cornell University Operations Research and Industrial Engineering, 2008.
- 1026 [34] S. M. WILD, *Chapter 40: Pounders in tao: Solving derivative-free nonlinear least-squares prob-*
1027 *lems with pounders*, in Advances and trends in optimization with engineering applications,
1028 SIAM, 2017, pp. 529–539.
- 1029 [35] S. M. WILD, R. G. REGIS, AND C. A. SHOEMAKER, *ORBIT: Optimization by radial basis*
1030 *function interpolation in trust-regions*, SIAM Journal on Scientific Computing, 30 (2008),
1031 pp. 3197–3219.
- 1032 [36] S. M. WILD AND C. SHOEMAKER, *Global convergence of radial basis function trust region*
1033 *derivative-free algorithms*, SIAM Journal on Optimization, 21 (2011), pp. 761–781.
- 1034 [37] H. ZHANG, A. R. CONN, AND K. SCHEINBERG, *A derivative-free algorithm for least-squares*
1035 *minimization*, SIAM Journal on Optimization, 20 (2010), pp. 3555–3576.

1036 **Appendix A. Sketch of Proof for Theorem 2.9.**

1037 For the sake of brevity and since one can prove the results stated in this section
 1038 using arguments that have already appeared for similar results for related algorithms
 1039 in the literature (e.g., see [5]), we provide in this appendix only a discussion of the
 1040 results that one needs to prove in order to prove Theorem 2.9.

1041 First, one can show for all $k \in \mathbb{N}$ that the difference between the stationarity
 1042 measures with respect to f and m_k are proportional to Δ_k ; see, e.g., [8, Lemma 7].

1043 LEMMA A.1. *For all $k \in \mathbb{N}$, it follows that for any value of Δ_k such that a
 1044 model m_k is constructed, one has that $|\pi_k^f - \pi_k^m| \leq \kappa_{\text{eg}} \Delta_k$.*

1045 Second, one can show lower bounds for the stationarity measures in certain cases.
 1046 The proof of this result relies on Lemma A.1.

1047 LEMMA A.2. *For any $k \in \mathbb{N}$, if the condition in line 4 does not hold, then $\pi_k^m \geq$
 1048 $\min\{\epsilon_c, \mu^{-1} \Delta_k\}$. If, in addition, $\pi_k^f \geq \epsilon \in \mathbb{R}_{>0}$, then*

$$1049 \pi_k^m \geq \epsilon_{\text{mc}} := \min \left\{ \epsilon_c, \frac{\epsilon}{1 + \kappa_{\text{eg}} \mu} \right\} \in \mathbb{R}_{>0}.$$

1051 Third, one can show that if Δ_k is sufficiently small, then a successful step occurs.

1052 LEMMA A.3. *For any $k \in \mathbb{N}$, if trust region radius satisfies*

$$1053 \text{(A.1)} \quad \Delta_k \leq \min\{c_0 \pi_k^m, 1\}, \quad \text{where } c_0 := \min \left\{ \mu, \frac{1}{\kappa_{\text{bhm}} + 1}, \frac{\kappa_{\text{fcd}}(1 - \eta)}{2\kappa_{\text{ef}}} \right\},$$

1054 *then the condition in line 4 does not hold and $\rho_k \geq \eta$, i.e., the step is successful.*

1056 Fourth, one can show that if the stationarity measure with respect to f is bounded
 1057 below by a positive constant, then the trust region radius is similarly bounded below.
 1058 The proof of this result relies on all of the preceding lemmas.

1059 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*

$$1060 \Delta_{\min} := \min \left\{ \Delta_0, \frac{\gamma_{\text{dec}} \epsilon}{\kappa_{\text{eg}} + \mu^{-1}}, \gamma_{\text{dec}} \left(\kappa_{\text{eg}} + \frac{2\kappa_{\text{ef}}}{\kappa_{\text{fcd}}(1 - \eta)} \right)^{-1} \epsilon, \gamma_{\text{dec}} \mu \epsilon_{\text{mc}}, \frac{\gamma_{\text{dec}} \epsilon_{\text{mc}}}{\kappa_{\text{bhm}} + 1}, \gamma_{\text{dec}} \right\} \blacksquare$$

1062 Fifth, one can show that if the number of successful steps is finite, then the trust
 1063 region radius and stationarity measure with respect to f must vanish. The proof of
 1064 this result relies on Lemma A.1.

1065 LEMMA A.5. *If $|\{k \in \mathbb{N} : \rho_k \geq \eta\}|$ is finite, then $\lim_{k \rightarrow \infty} \Delta_k = 0$ and $\lim_{k \rightarrow \infty} \pi_k^f = 0$.*

1066 Sixth, one can show the trust region radius always vanishes. The proof of this
 1067 result relies on Lemmas A.2 and A.4.

1068 LEMMA A.6. *The trust region radius vanishes, i.e., $\lim_{k \rightarrow \infty} \Delta_k = 0$.*

1069 Seventh, one can show that a subsequence of stationarity measures vanishes. The
 1070 proof of this result relies on Lemmas A.2, A.4, and A.5.

1071 LEMMA A.7. *The limit inferior of $\{\pi_k^f\}$ is zero, i.e., $\liminf_{k \rightarrow \infty} \pi_k^f = 0$.*

1072 Finally, given the ‘‘liminf’’ result in Lemma A.7, one can prove Theorem 2.9
 1073 using Lemmas A.2, A.5, and A.6 along with common techniques for the analysis of
 1074 trust-region methods that can turn ‘‘liminf’’ into ‘‘lim’’ results.