Avoiding Geometry Improvement in Derivative-Free Model-Based Methods via Randomization

Matt Menickelly

May 30, 2023

Abstract

We present a technique for model-based derivative-free optimization called *basis sketching*. Basis sketching consists of taking random sketches of the Vandermonde matrix employed in constructing an interpolation model. This randomization enables weakening the general requirement in model-based derivative-free methods that interpolation sets contain a full-dimensional set of affinely independent points in every iteration. Practically, this weakening provides a theoretically justified means of avoiding potentially expensive geometry improvement steps in many model-based derivative-free methods. We demonstrate this practicality by extending the nonlinear least squares solver, POUNDers to a variant that employs basis sketching and we observe encouraging results on higher dimensional problems.

1 Introduction

Derivative-free optimization (DFO) is an important and practical class of nonlinear optimization characterized by an assumption that derivatives of an objective function (and/or constraint functions) cannot be directly computed. Instead, it is assumed that one has access only to a black box oracle for computing the objective (and/or constraint) value(s). There exists an abundance of methods designed for derivative-free optimization, see e.g., the survey [39].

In this paper, we focus on a particular subclass of these methods, in particular, model-based DFO trust-region methods for unconstrained optimization:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{1}$$

for some $f(x) : \mathbb{R}^n \to \mathbb{R}$. Such methods employ interpolation models of $f(x)$ as a proxy for a first- (or possibly second-) order Taylor model within a trust-region framework. While the convergence analysis of these methods and a formal analysis of interpolation model quality was largely pioneered by [20–23] and is most of the subject of the textbook [24], the implementation and practical use of this subclass of methods was very much driven by Michael Powell, and forms the backbone of many of his well-known DFO solvers [46–54].

It is well-known that methods within this subclass do not theoretically (or practically) scale well with the dimension n. Intuitively, in order to approximate a gradient $\nabla f(x)$, one must compute n directional derivatives centered at x, which requires $\mathcal{O}(n)$ function evaluations. This linear dependence on n, barring some very restrictive assumptions, is inevitable; see [39][Table 8.1] for a recent summary of worst case complexity results for methods in this subclass exhibiting this dependence on dimension.

The intention of this paper is to suggest a framework for *practically* alleviating this $\mathcal{O}(n)$ dependence. We do this by taking a recognizable framework (model-based trust-region methods) and *randomizing* the construction of interpolation models by only updating an average model gradient and average model Hessian within a random (but realized from a judiciously selected distribution) subspace in each iteration. The inspiration for such a procedure draws heavily from SEGA [37], which employs *gradient sketches* to update an average gradient estimator. The name SEGA is intentionally close to SAGA [26], which can be viewed as a particular sketch of the summands of a finite sum, as opposed to a sketch of an n-dimensional gradient (see [31]). Analogously, the contribution of this paper is an analogue of our recent work in SAM-POUNDers [41], which randomly sketched a finite sum in a nonlinear least-squares problem by building on a model-based trust-region method for nonlinear least-squares problems, POUNDers [61].

1

1.1 Related Work

Owing to practical relevance, there exists a body of literature concerning the scalability of DFO methods. Most notably, [14] analyzed a trust-region framework that iteratively selects a low-dimensional subspace, constructs an interpolation model *only intended to be a reasonable approximation of an objective function on that subspace*, and then minimizes the low-dimensional model within a trust region. By choosing the low-dimensional subspace according to constructions based on Johnson-Lindenstrauss transforms (see, e.g., [25]), [14] demonstrates a high probability worst case complexity result that is *independent of n*. Such a result is encouraging in that it lends credence to methods based on iterative subspace selection, and motivates further exploration of methods such as the one in this paper. Prior work in subspace model-based methods (but without such rigorous convergence guarantees) includes VXQR [44] and work with moving ridge functions [35].

Other classes of DFO methods besides model-based methods have also considered issues of scalability. In particular, the class of direct search methods (i.e., methods based on making direct comparisons between pairs of function evaluations as opposed to constructing models, see [4] for a textbook treatment) can intuitively always choose to evaluate fewer than n poll points per iteration. Probabilistic convergence results for such randomized direct search methods are given in [8, 32, 34].

We comment on the existence of additional DFO methods for large-scale problems outside of general model-based or direct search methods. There exist methods that make particular assumptions (and then exploit) some known structure of the problem, including partial separability [19] or sparse Hessians [5]. Others methods, based on a derivative-free method popularized by the analysis of Nesterov [43], attempt to only approximate one directional derivative per iteration; however, it is common knowledge among practitioners that these methods are practically very inefficient in terms of function evaluation complexity, see [7] for a critical view.

Finally, there exists a large and growing body of related work in sketching in *derivative-based* methods. In the context of sketching for dimensionality reduction to yield subspace methods akin to those in [14], sketching has been applied to Newton's method [6, 29, 45], quasi-Newton methods [28], SAGA [31], trust-region methods [11], quadratic regularization methods [12], and cubic regularization methods [36]. For a distinct class of methods, (randomized) coordinate descent methods and their block variants employ directional derivative information in (randomized) coordinate directions. While not subspace methods, or sketching methods, in exactly the terms we have used so far, coordinate descent methods effectively update a subset of variables parameterizing a coordinate-aligned subspace in each iteration. The structure of many problems in machine learning naturally lends itself to these methods. As a result, the literature on coordinate descent methods is vast, and so we only provide references to two good surveys of the topic, [58, 63].

1.2 Motivation

We are especially motivated by computationally expensive problems. In particular, and without any further quantification, we will assume that the cost (in time, energy consumption, dollar cost, or whatever relevant metric) of computing a single function evaluation $f(x)$ far exceeds the cost in the same metric of relevant linear algebraic procedures which are polynomial in n (e.g., QR decompositions or solving trust region subproblems).

We are additionally motivated by our recent work in SAM-POUNDers [41]. SAM-POUNDers is motivated by problems in nuclear model calibration [10]; these problems are naturally formulated as derivative-free (and potentially computationally expensive) nonlinear least squares problems of the form

$$\min_{x \in \mathcal{D}} \sum_{i=1}^{p} f_i(x)^2, \tag{2}$$

where $\mathcal{D}$ is defined by bound constraints. Unlike most assumptions in the literature, however, it is often the case in nuclear model calibration that most, if not all, of the individual expensive function evaluations $f_i(x)$ in (2) can be performed in parallel. SAM-POUNDers exploits this by maintaining *separate* local interpolation models of each $f_i(x)$, with each generally employing different interpolation sets. Most importantly, and like in a stochastic average gradient method (e.g. SAG [57] or SAGA [26]), not all of the p local models of the function $f_i(x)$ in (2) will be updated in every iteration of SAM-POUNDers. Thus, in such a method, we must always have access to a (potentially stale) local model of each component function $f_i(x)$ that models $f_i(x)$

on the full n-dimensional space. As described in Section 1.1, a subspace method does not aim to provide such a model, electing instead to build a model of a function on a random subspace and then immediately discard the model for the next iteration. With this aim in mind, this paper will maintain a model – in particular, a quadratic model defined by the *average gradient* and *average Hessian*) – between iterations without discarding the model. Moreover, having such an estimator of full-space gradient information can also aid in defining more practical stopping criteria, which is an identified shortcoming in subspace methods.

1.3 Contributions and Organization

In this paper, we will present a method called *basis sketching*. Basis sketching maintains a running *average estimator* of both the gradient and Hessian of $f(x)$, which is employed to compute a set of specifically and stochastically weighted estimators of the gradient and Hessian, called the *ameliorated estimators*. While we will present a general framework for this method in Algorithm 1, we will specifically implement this algorithm using the model-building routines of `POUNDers` as a foundation. We will demonstrate through numerical results a clear advantage of the randomized variant presented in this paper over the original implementation of `POUNDers` on problems of sufficiently high dimension.

We begin in Section 2 with preliminaries on the interpolation models that we will use in this work. Then, in Section 3, we will discuss the particular estimators we will employ that are derived from our interpolation models. In Section 4, we will present the basis sketching algorithm, and will immediately afterwards discuss practical modifications in Section 5. We conclude with a presentation of numerical results in Section 6, which are intended to show that basis sketching offers improvement over the standard (deterministic) geometry improvement and model improvement steps made in `POUNDers`.

2 Preliminaries on Interpolation Models

We begin with a discussion of general nonlinear interpolation models in Section 2.1, and we will then restrict ourselves to a discussion of underdetermined interpolation models in Section 2.2.

2.1 General Nonlinear Interpolation

Denote by $\mathcal{P}_n^d$ the space of polynomials of degree less than or equal to d in $\mathbb{R}^n$. We say that a polynomial $m(x) \in \mathcal{P}_n^d$ *interpolates* the function $f(x)$ at a point y provided $m(y) = f(y)$. Suppose we have a set of points $Y = \{y^0, y^1, \ldots, y^p\} \subset \mathbb{R}^n$. To find a polynomial $m(x) \in \mathcal{P}_n^d$ that interpolates $f(x)$ at each point in Y, one can choose a basis for $\mathcal{P}_n^d$, which we denote by $\Phi = \{\phi_0, \phi_1, \ldots, \phi_q\}$, and solve for the coefficients α in the system

$$M(\Phi, Y)\alpha_\Phi = f(Y), \tag{3}$$

where we have written

$$M(\Phi, Y) = \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \cdots & \phi_q(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \cdots & \phi_q(y^1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \cdots & \phi_q(y^p) \end{bmatrix}, \alpha_\Phi = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_p \end{bmatrix} \text{ and } f(Y) = \begin{bmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^p) \end{bmatrix}.$$

We refer to $M(\Phi, Y)$ as the *Vandermonde matrix*. If the Vandermonde matrix $M(\Phi, Y)$ is square (i.e., $p = q$, where $q + 1$ is the dimension of $\mathcal{P}_n^d$) and additionally, $M(\Phi, Y)$ is full rank, then (3) has a unique solution, that is, there exists a unique polynomial $m(x) \in \mathcal{P}_n^d$ given by

$$m(x) = \sum_{j=0}^q \alpha_j^* \phi_j(x)$$

that interpolates $f(x)$ at each point in Y; this unique polynomial is defined by the solution α^* to $M(\Phi, Y)\alpha = f(Y)$. We record this in a definition:

Definition 1. *Given a basis Φ of dimension $p+1$ for a polynomial space $\mathcal{P}$, and a set of points $Y = \{y^0, y^1, \ldots, y^p\}$, we say that Y is* poised for interpolation with respect to $\mathcal{P}$ *provided $M(\Phi, Y)$ is invertible.*

A common and practical choice of basis for $\mathcal{P}_n^d$ is given by

$$\Phi_L(y) = \{1, y_1, y_2, \ldots, y_n\}$$

when $d = 1$ and

$$\Phi_Q(y) = \Phi_L(y) \cup \left\{ \frac{y_1^2}{2}, \ldots, \frac{y_n^2}{2}, \frac{y_1 y_2}{\sqrt{2}}, \ldots, \frac{y_1 y_n}{\sqrt{2}}, \frac{y_2 y_3}{\sqrt{2}}, \frac{y_2 y_n}{\sqrt{2}}, \ldots, \frac{y_{n-1} y_n}{\sqrt{2}} \right\}$$

when $d = 2$. In the setting of expensive derivative-free optimization, it is often the case when we employ quadratic models (i.e., $d = 2$) that $q = (n+1)(n+2)/2$ function evaluations at points Y – which would guarantee $M(\Phi_Q, Y)$ is invertible – are simply not available. Moreover, the conditioning of the matrix $M(\Phi, Y)$ must be bounded, as model errors essentially scale linearly with this condition number; this will be seen, for instance, in Theorem 1. In model-based DFO, this notion of bounded condition numbers is often formalized through the language of Lagrange polynomials and well-poisedness. For simplicity in this mansucript, and as we will see in Theorem 1, we will only seek to ensure poisedness with respect to a transformation of the basis Φ_L, which effectively amounts to bounding $\|S\tilde{Y}_\Delta\|^{-1}$ where $S \in \mathbb{R}^{p \times n}$ with $p \leq n$, and where the ith column of the $n \times p$ matrix $\tilde{Y}_\Delta$ is given by $(y^i - y^0)/\Delta$ for some $\Delta > 0$. The fact that $p \leq n$ implies that S is a sketching matrix, hence giving us the term *basis sketching* for the mechanism we introduce in this manuscript.

Finally, we remind the reader of the definition of *full linearity*, a common (see, e.g., [24][Definition 6.1]) measure of model quality in model-based DFO.

Definition 2. *Given $\kappa_{ef}, \kappa_{eg} \geq 0$ and $\Delta > 0$, a model $m(x)$ is a $(\kappa_{ef}, \kappa_{eg})$-fully linear model of $f(x)$ on a ball of radius Δ centered at x_c (i.e., the ball $\mathcal{B}(x_c; \Delta) := \{y : \|y - x_c\| \leq \Delta\}$) provided*

$$|m(x) - f(x)| \leq \kappa_{ef} \Delta^2 \quad \text{and} \quad \|\nabla m(x) - \nabla f(x)\| \leq \kappa_{eg} \Delta$$

for all $y \in \mathcal{B}(x_c; \Delta)$.

Intuitively, Definition 2 captures the idea that, up to constants, the error of a general model $m(x)$ of $f(x)$ is no worse than the error made by employing a first-order Taylor model of $f(x)$.

2.2 Underdetermined Interpolation Systems

The idea of basis sketching is largely motivated by the idea of minimal norm Hessian (MNH) interpolation introduced in [60]. In the notation established here, MNH assumes the basis Φ_Q is employed, and partitions the matrix $M(\Phi_Q, Y)$ into two submatrices

$$M(\Phi_Q, Y) = [M(\Phi_L, Y) \quad M(\Phi_Q \setminus \Phi_L, Y)],$$

where "\" denotes the set difference operation. The MNH approach replaces (3) with an optimization problem

$$\min_{\alpha \in \mathbb{R}^{n+1}, \beta \in \mathbb{R}^{n(n+1)/2}} \left\{ \frac{1}{2} \|\beta\|^2 : M(\Phi_L, Y)\alpha + M(\Phi_Q \setminus \Phi_L, Y)\beta = f(Y) \right\}. \tag{4}$$

In words, (4) removes the degrees of freedom from the underdetermined linear system in the constraint of (4) by choosing the basis coefficients corresponding to degree two polynomials in the basis such that these coefficients are of minimal norm. The work in [60] uses the KKT conditions of (4) to develop a method based on an iteratively updated QR factorization of $M(\Phi_L, Y)$ to choose a set Y such that $M(\Phi_L, Y)$ is well-conditioned. The selection of Y in the QR-based method of [60] greedily selects points from a bank of points for which f has previously been evaluated, and then adds points to Y (determined from the nullspace suggested by the QR factorization) for which function evaluations must be performed only after the greedy procedure has been run.

Basis sketching will mimic this greedy QR-based procedure of [60], but will first modify the subproblem (4) with a sketch. In particular, given a sketching matrix $S \in \mathbb{R}^{p \times n}$ with mutually orthonormal rows $s_1, \ldots, s_p$, we define a basis

$$\Phi_S = \{1, s_1 y, \ldots, s_p y\}$$

for a subspace of $\mathcal{P}_n^1$. Because we assume S has mutually orthornormal rows, we can choose a perpendicular matrix $S^\perp \in \mathbb{R}^{(n-p) \times n}$ satisfying $s_j^\perp s_i^\top = 0$ for $i = 1, \ldots, p$ and $j = 1, \ldots, n - p$, where j indexes the rows of $S^\perp$. This gives us a basis for an orthogonal subspace of $\mathcal{P}_n^1$,

$$\Phi_{S^\perp} = \left\{ s_1^\perp y, \ldots, s_{n-p}^\perp y \right\}.$$

With this notation, our subproblem of interest is

$$\min_{\alpha \in \mathbb{R}^{p+1}, \beta \in \mathbb{R}^{n(n+2)/2}, \gamma \in \mathbb{R}^{n-p}} \left\{ \frac{1}{2} \|\beta\|^2 + \frac{1}{2} \|\gamma\|^2 : M(\Phi_S, Y)\alpha + M(\Phi_{S^\perp}, Y)\gamma + M(\Phi_Q \setminus \Phi_L, Y)\beta = f(Y) \right\}. \quad (5)$$

In words, (5) fits an underdetermined quadratic model by limiting the degrees of freedom not only by minimizing the contribution from the degree two polynomials in the basis, but also from the subspace of $\mathcal{P}_n^1$ spanned by $\Phi_{S^\perp}$.

To motivate future development in this manuscript, we additionally mention that the subproblem (4) is trivially extended to a problem of identifying a minimal *change* Hessian, that is, replacing the objective with $\frac{1}{2}\|\beta - \bar\beta\|^2$, where $\bar\beta$ denotes the coefficients on $\Phi_Q \setminus \Phi_L$ taken from, for instance, the model Hessian employed in a previous iteration of an optimization algorithm. In fact, this trivial extension of (4) is what is employed in POUNDers [61] for the purpose of model-building. Any method for solving (4) can be extended to solving the minimal change problem simply by replacing each entry of the right hand side of (4) with a residual term

$$f(y^i) - \sum_{j=1}^{\frac{n(n+1)}{2}} \bar\beta_j \phi_j(y^i),$$

where the $n(n+1)/2$ basis functions ϕ_j are from $\Phi_Q \setminus \Phi_L$. Similarly, if we replace the objective of (5) with $\frac{1}{2}\|\beta - \bar\beta\|^2 + \frac{1}{2}\|\gamma - \bar\gamma\|^2$, then any method for solving (5) can be extended to this modified subproblem by replacing each entry in the right hand side of (5) with the appropriate residual term.

Our proposed basis sketching method is essentially a *sketch-and-project* process [30]. Basis sketching will maintain an *average estimate* of the gradient $\bar g \in \mathbb{R}^n$. On a given iteration of our basis sketching method, we will, in general, not construct a fully linear model of f, but will instead construct a model that is only fully linear when restricted to a particular subspace of $\mathbb{R}^n$ defined by a matrix S. We formalize this with an extension of the full linearity definition Definition 2 which we call S-full linearity. While our definition of S-full linearity is similar to the definition of Q-full linearity suggested in recent works in subspace methods for DFO [14, 27], note that they are not quite the same.

Definition 3. *Given $S \in \mathbb{R}^{p \times n}$ with $p \leq n$, constants $\kappa_{ef}, \kappa_{eg} \geq 0$ and $\Delta > 0$, a model $m(x)$ is a $(S, \kappa_{ef}, \kappa_{eg})$-fully linear model of $f(x)$ on $\mathcal{B}(x_c; \Delta)$ provided*

$$|m(x_c + S^\top d) - f(x_c + S^\top d)| \leq \kappa_{ef}\Delta^2 \quad and \quad \|\nabla m(x_c + S^\top d) - \nabla f(x_c + S^\top d)\| \leq \kappa_{eg}\Delta$$

for all $d \in \mathbb{R}^p$ satisfying $\|S^\top d\| \leq \Delta$.

As is often seen in the literature with full linearity, we will often drop the constants κ_{ef}, κ_{eg} when discussing S-full linearity. We now prove an extension of a result in [59, 62] which provides sufficient conditions to guarantee S-full linearity given some general conditions on the interpolation set Y. We first state an assumption made throughout this paper.

Assumption 1. *The function $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuously differentiable on $\mathcal{D}$. In particular, there exist $0 \leq L_f, L_g < \infty$ such that*

- $|f(x) - f(y)| \leq L_f \|x - y\|$ *and*
- $\|\nabla f(x) - \nabla f(y)\| \leq L_g \|x - y\|$

for all $x, y \in \mathcal{D}$.

Theorem 1. *Let $\Delta > 0$, let $\Lambda > 0$, and let $S \in \mathbb{R}^{p \times n}$ with $p \leq n$ have mutually orthonormal rows. Let $\{y^0, y^1, \ldots, y^p\} \subset \mathcal{B}(y^0; c\Delta)$ for some $c > 0$ and suppose that there exists $\delta^i \in \mathbb{R}^p$ so that $S\delta^i = y^i - y^0$ for each $i = 1, \ldots, p$. Let $\tilde{Y}$ denote an $p \times p$ matrix where the ith column is given by δ^i. Suppose f satisfies Assumption 1 and additionally suppose m is twice continuously differentiable on $\mathcal{B}(y^0; c\Delta)$ with gradient Lipschitz constant L_{mg}[1]. If both*

1. *$\|\tilde{Y}\|^{-1} \leq \dfrac{\Lambda}{c\Delta}$ and*

2. *$f(y^i) = m(y^i)$ for $i = 0, 1, \ldots, p$,*

then $m(x)$ is an $(S, \kappa_{ef}, \kappa_{eg})$-fully linear model of $f(x)$ with constants

$$\kappa_{ef} = \frac{4 + 5\Lambda\sqrt{p}}{2}(L_g + L_{mg})c^2 \quad and \quad \kappa_{eg} = \frac{5\Lambda\sqrt{p}}{2}(L_g + L_{mg})c.$$

We defer the proof to Appendix A for readability.

3 Average and Ameliorated Estimators

We now present the estimators that will be used in the basis sketching algorithm. Section 3.1 will discuss the motivation behind *average estimators*, a generally biased estimator of a gradient $\nabla f(x)$. In turn, Section 3.2 will then discuss a modification we will employ to yield *ameliorated estimators*, which are, by construction, an unbiased estimator for the gradient. Finally, Section 3.3 will discuss how (ideally) one would yield a minimum variance unbiased estimator for the gradient.

3.1 Average Estimators: Sketch and Project

The basis sketching trust region method is an iterative method (iterations indexed by k) that maintains an *average estimate* $\bar{g}^k \in \mathbb{R}^n$ of the gradient $\nabla f(x^k)$, where x^k is an incumbent point held by the algorithm. On each iteration k, we will identify (via a greedy procedure similar to the one employed in the MNH method) an orthonormal matrix $Q_k \in \mathbb{R}^{n \times n}$. We will then (randomly) select a subset $J_k \subset \{1, 2, \ldots, n\}$ of size p_k. Note that the value of $p_k \leq n$ may change on each iteration. We then choose the sketching matrix $S_k = ([Q_k]_{J_k})^\top$, that is, S_k is the transpose of the submatrix consisting of the columns of Q_k indexed by J_k. Then, motivated by Theorem 1 and using the notation of that theorem, we select a set Y_k of p_k many points so that $\|\tilde{Y}_k\|^{-1}$ is sufficiently bounded from above and so that each point in Y_k is sufficiently close to x^k. We then perform any necessary function evaluations at the points in Y_k, noting that the selection of Q_k is designed to encourage that the function evaluations being performed are relatively few. With this data, we obtain a model from the solution to (5) where the two bases for the linear polynomials are defined as Φ_{S_k} and $\Phi_{S_k^\perp}$; we note that by the construction of the orthonormal Q_k, obtaining the matrix $S_k^\perp$ is immediate. Because the constraint of (5) enforces that $m(y^i) = f(y^i)$ for all $y^i \in Y_k$, we have from Theorem 1 that the quadratic model $m(x)$ derived from the solution to (5) is a S_k-fully linear model of $f(x)$ on the ball $\mathcal{B}(x^k, \Delta_k)$. Let $\hat{g}^k \in \mathbb{R}^n$ denote the gradient term of the model $m(x)$. We then consider a subproblem to *update* the average gradient,

$$\bar{g}^k := \begin{array}{c} \arg\min_{\alpha \in \mathbb{R}^n} \quad \frac{1}{2}\|\alpha - \bar{g}^{k-1}\|^2 \\ \text{s. to} \quad S_k\alpha = S_k\hat{g}^k \end{array} \tag{6}$$

In words, (6) selects $\bar{g}^k$ as the closest (in Euclidean norm) vector to $\bar{g}^{k-1}$ such that the S_k-sketch of $\bar{g}^k$ agrees with the S_k-sketch of the (S_k-fully linear) model gradient $\hat{g}^k$. We quickly prove that (6) has a closed-form solution.

---

[1] In the case of quadratic models considered in this paper, L_{mg} is trivially derived from the spectral norm of the model Hessian.

Proposition 1. *A closed form solution to* (6) *exists and is given as*

$$\bar{g}^k = \bar{g}^{k-1} - S_k^\top S_k \bar{g}^{k-1} + S_k^\top S_k \hat{g}. \tag{7}$$

Proof. The KKT conditions associated with (6) can be expressed as

$$\begin{aligned}
\alpha &= \bar{g}_{k-1} - S_k^\top \mu \quad (stationarity) \\
S_k \alpha &= S_k \hat{g}^k \qquad\qquad (primal feasibility),
\end{aligned}$$

where $\mu \in \mathbb{R}^{p_k}$ is a vector of Lagrange multipliers. Plugging the stationarity condition into the primal feasibility condition, $S_k \bar{g}^{k-1} - S_k S_k^\top \mu = S_k \hat{g}^k$. Solving for μ and using the orthonormality of Q_k, we obtain $\mu = S_k(\bar{g}^{k-1} - \hat{g}^k)$. Plugging these Lagrange multipliers μ back into the stationarity condition, we obtain (7). Because the constraints of the problem determining $\bar{g}^k$ in (6) are affine, and because the objective is convex in α, the KKT conditions are also sufficient for optimality. $\square$

3.2 Ameliorated Estimators

At this point in our development, one could imagine and fully describe an iterative randomized method that essentially uses the model determined by the subproblem (5) in each iteration, but replaces the model gradient $\hat{g}^k$ with the average estimate $\bar{g}^k$ updated by (7). However, in the same sense that SAG [57] estimators are biased estimators of the true gradient, $\nabla f(x^k)$, $\bar{g}^k$ is a biased estimator of $\nabla f(x^k)$. We now describe how to slightly modify the update (7) via control variates so as to attain an unbiased estimator.

In the kth iteration of the basis sketching method, we associate with each $i \in \{1, \ldots, n\}$ an independent Bernoulli variable with success probability $\pi_i > 0$. Each of the n Bernoulli trials are realized, and for each successful trial, we include $i \in J_k$. Letting q_k^i denote the ith column of Q_k, we note that the update in (7) can be equivalently written

$$\bar{g}^k = \bar{g}^{k-1} - \sum_{i \in J_k} \left[q_k^i q_k^{i\top} \right] \bar{g}^k + \sum_{i \in J_k} \left[q_k^i q_k^{i\top} \right] \hat{g}^k.$$

Thus, one can see how

$$\tilde{g}^k := \bar{g}^{k-1} - \sum_{i \in J_k} \frac{1}{\pi_i} \left[q_k^i q_k^{i\top} \right] \bar{g}^{k-1} + \sum_{i \in J_k} \frac{1}{\pi_i} \left[q_k^i q_k^{i\top} \right] \hat{g}^k. \tag{8}$$

is a particular reweighting of the closed-form update (7). To simplify notation, given a vector of nonzero probabilities $\pi^k \in \mathbb{R}^n$ and the realization J_k, we define a diagonal matrix $D(\pi^k, J_k) \in \mathbb{R}^{p_k \times p_k}$ (recall that $|J_k| = p_k$) via

$$D(\pi^k, J_k) = diag([1/\pi_i^k : i \in J_k]).$$

We can then rewrite (8) as

$$\tilde{g}^k = \bar{g}^{k-1} - S_k^\top D(\pi^k, J_k) S_k \bar{g}^{k-1} + S_k^\top D(\pi^k, J_k) S_k \hat{g}^k. \tag{9}$$

We remark that there are no further restrictions on π^k other than all entries be nonzero, and the following theorems in this subsection hold under no further assumptions. The structure of the update (9) is inspired by *arbitrary sampling*; for papers on the subject of arbitrary sampling within the related context of (block) coordinate descent, see [3, 38, 55, 56].

We now record the following important observation, namely, that $\tilde{g}^k$ is an *unbiased estimator* of a $\mathcal{O}(\Delta_k)$-accurate approximation of $\nabla f(x^k)$.

Theorem 2. *Let $\kappa_{ef}, \kappa_{eg} \geq 0$. Suppose, for each J_k, that we obtain a model m_{J_k} such that m_{J_k} is a $(S_{J_k}, \kappa_{ef}, \kappa_{eg})$ fully linear model of f on $\mathcal{B}(x^k, \Delta_k)$. Let $\hat{g}_{J_k}$ denote the model gradient $\nabla m_{J_k}(x^k)$. Then, the expectation of $\tilde{g}^k$ with respect to the probability distribution on the selection of J_k satisfies*

$$\|\mathbb{E}_{J_k}[\tilde{g}^k] - \nabla f(x)\| \leq \sqrt{n}\kappa_{eg}\Delta_k$$

for all $x \in \mathcal{B}(x^k, \Delta_k)$.

Proof. Starting from the definition (8),

$$
\begin{aligned}
\mathbb{E}_{J_k}\left[\tilde{g}^k\right] &= \bar{g}^{k-1} - \mathbb{E}_{J_k}\left[\sum_{i\in J_k}\frac{1}{\pi_i}\left[q_k^i q_k^{i\top}\right]\bar{g}^{k-1}\right] + \mathbb{E}_{J_k}\left[\sum_{i\in J_k}\frac{1}{\pi_i}\left[q_k^i q_k^{i\top}\right]\hat{g}_{J_k}\right] \\
&= \bar{g}^{k-1} - \mathbb{E}_{J_k}\left[\sum_{i=1}^{n}\mathbb{1}\left[i\in J_k\right]\frac{1}{\pi_i}\left[q_k^i q_k^{i\top}\right]\right]\bar{g}^{k-1} + \mathbb{E}_{J_k}\left[\sum_{i\in J_k}\frac{1}{\pi_i}\left[q_k^i q_k^{i\top}\right]\hat{g}_{J_k}\right] \\
&= \bar{g}^{k-1} - \sum_{i=1}^{n}\left[\pi_i\frac{1}{\pi_i}\left[q_k^i q_k^{i\top}\right]\right]\bar{g}^{k-1} + \mathbb{E}_{J_k}\left[\sum_{i\in J_k}\frac{1}{\pi_i}\left[q_k^i q_k^{i\top}\right]\hat{g}_{J_k}\right] \\
&= \bar{g}^{k-1} - \bar{g}^{k-1} + \mathbb{E}_{J_k}\left[\sum_{i\in J_k}\frac{1}{\pi_i}\left[q_k^i q_k^{i\top}\right]\hat{g}_{J_k}\right] \\
&= \mathbb{E}_{J_k}\left[\sum_{i\in J_k}\frac{1}{\pi_i}\left[q_k^i q_k^{i\top}\right]\hat{g}_{J_k}\right] \\
&= \sum_{J_k} p(J_k)\sum_{i\in J_k}\frac{1}{\pi_i}\left[q_k^i q_k^{i\top}\right]\hat{g}_{J_k}.
\end{aligned}
$$

Now consider $Q_k^\top\left(\mathbb{E}_{J_k}[\tilde{g}^k] - \nabla f(x)\right)$ for any $x\in\mathcal{B}(x^k,\Delta_k)$. By orthonormality, the jth coordinate of $Q_k^\top\mathbb{E}_{J_k}[\tilde{g}^k]$ is

$$
\left|Q_k^\top[\mathbb{E}_{J_k}[\tilde{g}^k]]_j\right| = \sum_{J_k:j\in J_k} p(J_k)\frac{1}{\pi_j}q_k^{j\top}\hat{g}_{J_k}.
$$

Thus,

$$
\left|Q_k^\top[\mathbb{E}_{J_k}[\tilde{g}^k] - \nabla f(x)]_j\right| = q_k^{j\top}\left(\left[\frac{1}{\pi_j}\sum_{J_k:j\in J_k} p(J_k)\hat{g}_{J_k}\right] - \nabla f(x)\right)
$$

Because $\displaystyle\sum_{J_k:j\in J_k} p(J_k) = \pi_j$, we can equivalently write

$$
\left|Q_k^\top[\mathbb{E}_{J_k}[\tilde{g}^k] - \nabla f(x)]_j\right| = q_k^{j\top}\left[\frac{1}{\pi_j}\sum_{J_k:j\in J_k} p(J_k)(\hat{g}_{J_k} - \nabla f(x))\right]
$$

By our supposition that each $\hat{g}_{J_k}$ is the gradient of a S_{J_k}-fully-linear model,

$$
\left|Q_k^\top[\mathbb{E}_{J_k}[\tilde{g}^k] - \nabla f(x)]_j\right| \leq \|q_k^j\|\frac{1}{\pi_j}\left|\sum_{J_k:j\in J_k} p(J_k)(\hat{g}_{J_k} - \nabla f(x))\right| \leq \frac{1}{\pi_j}\sum_{J_k:j\in J_k} p(J_k)\|\hat{g}_{J_k} - \nabla f(x^k)\| \leq \kappa_{eg}\Delta_k.
$$

Thus,

$$
\|\mathbb{E}_{J_k}[\tilde{g}^k] - \nabla f(x)\| = \|Q_k^\top\left(\mathbb{E}_{J_k}[\tilde{g}^k] - \nabla f(x)\right)\| \leq \sqrt{n}\kappa_{eg}\Delta_k,
$$

as we meant to show. $\qquad\square$

3.3 Towards A Minimum Variance Estimator

Having established the sense in which $\tilde{g}^k$ is an unbiased estimator of a particular approximation of $\nabla f(x)$, we now state a result concerning the variance of this estimator.

Theorem 3. *The variance of $\tilde{g}^k$, with respect to the distribution governing J_k, is*

$$
\mathbb{E}_{J_k}\left[\|\tilde{g}^k - \mathbb{E}_{I_k}[\tilde{g}^k]\|^2\right] = \|\bar{g}^{k-1} - \mathbb{E}_{I_k}[\tilde{g}^k]\|^2_{Q_k D_k Q_k^\top - I_n}, \tag{10}
$$

where we denote $D_k = D(\pi^k,\{1,\ldots,n\})$ and I_n denotes the n-dimensional identity matrix.

Proof. Denote

$$P(J_k) = \sum_{i=1}^{n} \mathbb{1}\left[i \in J_k\right] \frac{1}{\pi_i} q_i q_i^\top \quad \text{and} \quad v(J_k) = P(J_k)\hat{g}_{J_k}$$

We first record that

$$
\begin{aligned}
\mathbb{E}_{J_k}\left[\|\tilde{g}^k - \mathbb{E}_{I_k}[\tilde{g}^k]\|^2\right] &= \mathbb{E}_{J_k}\left[\|[I - P(J_k)]\bar{g}^{k-1} + v(J_k)\|^2\right] - \|\mathbb{E}_{I_k}[\tilde{g}^k]\|^2 \\
&= \mathbb{E}_{J_k}\left[(\bar{g}^{k-1})^\top P(J_k)^\top P(J_k)\bar{g}^{k-1} - 2(\bar{g}^{k-1})^\top P(J_k)\bar{g}^{k-1}\right. \\
&\quad \left. + 2(\bar{g}^{k-1})^\top v(J_k) - 2(\bar{g}^{k-1})^\top P(J_k)v(J_k) + \|v(J_k)\|^2\right] + \|\bar{g}^{k-1}\|^2 - \|\mathbb{E}_{I_k}[\tilde{g}^k]\|^2
\end{aligned}
\tag{11}
$$

We now compute the expectations, with respect to J_k, of the matrices $P(J_k)$ and $P(J_k)^\top P(J_k)$, of the vectors $P(J_k)v(J_k)$ and $v(J_k)$, and of the scalar $\|v(J_k)\|^2$.

$$\mathbb{E}_{J_k}\left[P(J_k)\right] = \mathbb{E}_{J_k}\left[\sum_{i=1}^{n} \mathbb{1}[i \in J_k]\frac{1}{\pi_i^k}q_i q_i^\top\right] = \sum_{i=1}^{n} q_i q_i^\top = Q_k Q_k^\top = I_n$$

$$
\begin{aligned}
\mathbb{E}_{J_k}\left[P(J_k)^\top P(J_k)\right] &= \mathbb{E}_{J_k}\left[\left[\sum_{i=1}^{n} \mathbb{1}[i \in J_k]\frac{1}{\pi_i^k}q_i q_i^\top\right]^\top \left[\sum_{j=1}^{n} \mathbb{1}[j \in J_k]\frac{1}{\pi_j^k}q_j q_j^\top\right]\right] \\
&= \mathbb{E}_{J_k}\left[\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{1}[i,j \in J_k]\frac{1}{\pi_i^k \pi_j^k}\left[q_i q_i^\top\right]q_j q_j^\top\right] \\
&= \mathbb{E}_{J_k}\left[\sum_{i=1}^{n} \mathbb{1}[i \in J_k]\frac{1}{(\pi_i^k)^2}q_i q_i^\top\right] \\
&= \sum_{i=1}^{n} \frac{1}{\pi_i^k}q_i q_i^\top = Q_k D_k Q_k^\top
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}_{J_k}\left[P(J_k)v(J_k)\right] &= \mathbb{E}_{J_k}\left[\left[\sum_{i=1}^{n} \mathbb{1}\left[i \in J_k\right]\frac{1}{\pi_i^k}q_i q_i^\top\right]\left[\sum_{i=1}^{n} \mathbb{1}\left[i \in J_k\right]\frac{1}{\pi_i^k}q_i q_i^\top \hat{g}^{J_k}\right]\right] \\
&= \mathbb{E}_{J_k}\left[\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{1}\left[i,j \in J_k\right]\frac{1}{\pi_i^k \pi_j^k}q_i q_i^\top q_j q_j^\top \hat{g}^{J_k}\right] \\
&= \mathbb{E}_{J_k}\left[\sum_{i=1}^{n} \mathbb{1}\left[i \in J_k\right]\frac{1}{(\pi_i^k)^2}q_i q_i^\top \hat{g}^{J_k}\right] \\
&= Q_k D_k Q_k^\top \mathbb{E}_{I_k}[\tilde{g}^k]
\end{aligned}
$$

$$\mathbb{E}_{J_k}\left[v(J_k)\right] = \mathbb{E}_{J_k}\left[\sum_{i=1}^{n} \mathbb{1}\left[i \in J_k\right]\frac{q_i^\top \hat{g}^{J_k}}{\pi_i^k}q_i\right] = \mathbb{E}_{I_k}[\tilde{g}^k]$$

$$\mathbb{E}_{J_k}\left[\|v(J_k)\|^2\right] = \mathbb{E}_{J_k}\left[\sum_{i \in J_k} \frac{(q_i^\top \hat{g}^{J_k})^2}{(\pi_i^k)^2}q_i^\top q_i\right] = \sum_{J_k} p(J_k)\frac{(q_i^\top \hat{g}^{J_k})^2}{(\pi_i^k)^2} = \mathbb{E}_{I_k}[\tilde{g}^k]^\top Q_k D_k Q_k^\top \mathbb{E}_{I_k}[\tilde{g}^k]$$

We can now continue the equalities in (11):

$$
\begin{aligned}
\mathbb{E}_{J_k}\left[\|\tilde{g}^k - \mathbb{E}_{I_k}[\tilde{g}^k]\|^2\right] &= (\bar{g}^{k-1})^\top Q_k D_k Q_k^\top \bar{g}^{k-1} - 2(\bar{g}^{k-1})^\top \bar{g}^{k-1} + 2(\bar{g}^{k-1})^\top \mathbb{E}_{I_k}[\tilde{g}^k] \\
&\quad - 2(\bar{g}^{k-1})^\top D_k \mathbb{E}_{I_k}[\tilde{g}^k] + \mathbb{E}_{I_k}[\tilde{g}^k]^\top Q_k D_k Q_k^\top \mathbb{E}_{I_k}[\tilde{g}^k] + \|\bar{g}^{k-1}\|^2 - \|\mathbb{E}_{I_k}[\tilde{g}^k]\|^2 \\
&= \|\bar{g}^{k-1} - \mathbb{E}_{I_k}[\tilde{g}^k]\|^2_{Q_k D_k Q_k^\top - I_n},
\end{aligned}
\tag{12}
$$

as we intended to show. $\qquad\square$

We make three observations concerning Theorem 3. First, as a sanity check, notice that because $D_k \succ I_n$ by the definition of D_k, we have that $Q_k D_k Q_k^\top \succ I_n$, and so the quantity (10) is always nonnegative. Second, if $D_k = I_n$, then the variance is zero; this makes sense because this means each $\pi_i^k = 1$, i.e., deterministically, $J_k = \{1, 2, \ldots, p\}$, and so the variance is trivial. Third, and of practical importance, the variance of a (nontrivial) estimator will be generally unknown, since the quantity $\mathbb{E}_{J_k}[\tilde{g}^k]$ (and its expectation, a $\mathcal{O}(\Delta_k)$-accurate approximation of $\nabla f(x^k)$) appearing in the right hand side of (10) is unknown. This is of course a practical concern, and we will provide a proxy for this unknown quantity in Section 5, but we will first discuss how to derive an estimator $\tilde{g}^k$ of minimum variance, assuming this quantity were known.

Observe first that, by the assumed independence of the Bernoulli variables,

$$\mathbb{E}\left[|J_k|\right] = \sum_{i=1}^{n} \pi_i^k.$$

We consider an optimization problem to minimize the variance of $\tilde{g}^k$ under the constraint that the *expected* size of J_k is a given value, $p_k \leq n$. That is, assuming we had access to $\delta^k := \bar{g}^{k-1} - \mathbb{E}_{J_k}[\tilde{g}^k]$, we would solve

$$
\begin{array}{llll}
\min_{\pi^k} & \|\delta^k\|^2_{Q_k D_k Q_k^\top - I_n} & \min_{\pi^k} & \|Q_k^\top \delta^k\|^2_{D_k - I_n} & \min_{\pi^k} & \sum_{i=1}^{n}\left(\frac{1}{\pi_i^k} - 1\right)\left[Q_k^\top \delta^k\right]_i^2 & \min_{\pi^k} & \sum_{i=1}^{n}\frac{\left[Q_k^\top \delta^k\right]_i^2}{\pi_i^k} \\
\text{s. to} & \sum_{i=1}^{n}\pi_i^k = p_k & \equiv \ \text{s. to} & \sum_{i=1}^{n}\pi_i^k = p_k & \text{s. to} & \sum_{i=1}^{n}\pi_i^k = p_k & \equiv \ \text{s. to} & \sum_{i=1}^{n}\pi_i^k = p_k \\
& 0 \leq \pi_i^k \leq 1 \quad \forall i & & 0 \leq \pi_i^k \leq 1 \quad \forall i & & 0 \leq \pi_i^k \leq 1 \quad \forall i & & 0 \leq \pi_i^k \leq 1 \quad \forall i
\end{array}
\tag{13}
$$

The following theorem is immediate by deriving KKT conditions.

Theorem 4. *The optimal solution of* (13) *is expressible in closed form and is given, for each i, as*

$$
\pi_{(i)}^k = \begin{cases} (p_k + c - n)\dfrac{\left|\left[Q_k^\top \delta^k\right]_{(i)}\right|}{\displaystyle\sum_{j=1}^{c}\left|\left[Q_k^\top \delta^k\right]_{(j)}\right|} & \text{if } i \leq c \\[3ex] 1 & \text{if } i > c, \end{cases}
\tag{14}
$$

where c is the largest integer satisfying

$$
0 < p_k + c - n \leq \sum_{j=1}^{n}\frac{\left|\left[Q_k^\top \delta^k\right]_{(j)}\right|}{\left|\left[Q_k^\top \delta^k\right]_{(c)}\right|},
$$

and we have use the order statistics notation $|Q_k^\top \delta^k|_{(1)} \leq |Q_k^\top \delta^k|_{(2)} \leq \ldots \leq |Q_k^\top \delta^k|_{(n)}$.

4 Basis Sketching

Having discussed all the components, we are now in a position to fully state a basis sketching model-based trust-region algorithm. Pseudocode is provided in Algorithm 1.

Summarily, the basis sketching method begins each iteration employing a method very much resembling Algorithm 4.1 in [60]. This method chooses an orthogonal matrix Q_k such that a set of previously evaluated points in Y_k satisfies the first condition of Theorem 1 for a sketching matrix defined by the transpose of the first few columns of Q_k. Pseudocode for this initial subspace-determining algorithm is stated in Algorithm 2.

Algorithm 2 is a greedy procedure for selecting a subspace based on the bank of previously evaluated points $\mathcal{Y}$. Algorithm 2 maintains two orthogonal subspaces, S and $S^\perp$, which are effectively initialized as $\{0_n\}$ and $\mathbb{R}^n$, respectively. If a point in the bank is both within distance $c\Delta$ from x, and has a sufficiently large projection onto the current subspace $S^\perp$, then its displacement from x is added to a set of vectors whose span is S. We then update a(n orthonormal) basis for $S^\perp$; although not explicit in the statement of Algorithm 2, this is achieved in practice via an insertion into a maintained QR decomposition.

1 **(Initialization)** Choose algorithmic constants $\eta_1, \eta_2, \Delta_{\max} > 0$ and $0 < \nu_1 < 1 < \nu_2$.

2 Choose initial point $x^0 \in \mathbb{R}^n$ and initial trust-region radius $\Delta_0 \in (0, \Delta_{\max})$.

3 Initialize $\bar{g}^0 \in \mathbb{R}^n$.

4 Initialize a bank of points $\mathcal{Y}$ with pairs $(x, f(x))$ for which $f(x)$ is known.

5 **for** $k = 1, 2, \ldots$ **do**

6   **(Get initial subspace)** Use Algorithm 2 to obtain S_k, $S_k^\perp$ and Q_k.

7   **(Choose sketch size and error estimate)** Choose p_k and δ^k.

8   **(Determine probabilities)** Compute π^k according to (14).

9   **(Realize a random subset)** Generate J_k using Bernoulli parameters π^k.

10   **(Perform additional function evaluations)** Evaluate $\{f(x^k + \Delta_k q_i^k) : i \in J_k\}$ and update $\mathcal{Y}, S_k$ and $S_k^\perp$.

11   **(Choose interpolation set)** Use Algorithm 4 to obtain Y_k.

12   **(Get model parameters)** Compute model gradient $\hat{g}^k$ and model Hessian H^k from (5).

13   **(Compute ameliorated estimator)** Compute $\tilde{g}^k$ via (9).

14   **(Update average estimator)** Update $\bar{g}^k$ via (7)

15   **(Solve TRSP)** (Approximately) solve $\min\limits_{y \in \mathbb{R}^n} m_k(y) \triangleq \tilde{g}^{k\top} y + \frac{1}{2} y^\top H^k y$ to obtain d^k.

16   **(Evaluate new point)** Evaluate $f(x^k + d^k)$ and update $\mathcal{Y}$.

17   **(Determine acceptance)** Compute $\rho_k \leftarrow \dfrac{f(x^k) - f(x^k + d^k)}{m_k(0) - m_k(d^k)}$.

18   **if** $\rho_k \geq \eta_1$ **then**

19 $x^{k+1} \leftarrow x^k + d^k$.

20   **else**

21 $x^{k+1} \leftarrow x^k$.

22   **end**

23   **(Trust-region adjustment) if** $\rho_k \geq \eta_1$ **then**

24   **if** $\|\tilde{g}^k\| \geq \eta_2 \delta_k$ **then**

25 $\Delta_{k+1} \leftarrow \min\{\nu_2 \Delta_k, \Delta_{\max}\}$.

26   **else**

27 $\Delta_{k+1} \leftarrow \nu_1 \Delta_k$

28   **end**

29   **else**

30 $\Delta_{k+1} \leftarrow \nu_1 \Delta_k$.

31   **end**

32 **end**

Algorithm 1: Basis Sketching Trust-Region Method

Returning to Algorithm 1, given an expected sketch size p_k and some estimate δ_k of $\bar{g}^{k-1} - \mathbb{E}_{I_k}[\tilde{g}^k]$ (we will discuss practical means of choosing these quantities in Section 5, but state the algorithm in full generality allowing for any choice of p_k and δ_k), we compute a probability distribution π^k on the columns of Q_k. We then realize a random subset J_k according to n independent Bernoulli variables with respective probability parameters π_i^k, and evaluate f at each of $\{x^k + \Delta q_i^k : i \in J_k\}$. Appropriately splitting Q_k into S_k and $S_k^\perp$ based on the output of Algorithm 2 and the subsequent realization of J_k, we then choose an interpolation set for use in the subproblem (5). We will make the choice of interpolation set via Algorithm 4.

Algorithm 4 is a greedy procedure for selecting an interpolation set from the bank of points $\mathcal{Y}$. Its explanation is a bit more involved and is moved to the appendix, so as not to distract from the explanation of the basis sketching method, but the procedure is derived in such a way to ensure good properties of the solution to (5), illustrated in Theorem 8.

Continuing our summary of Algorithm 1 with an interpolation set Y_k in hand, we next solve the subproblem (5) to obtain model parameters. While we use the optimal parameter vector β^* from (5) "as is" to define a model Hessian, we use $S_k^\top \alpha^*$ as $\hat{g}^k$ in order to compute $\tilde{g}^k$ according to (9), and then update $\bar{g}^k$ according to (7). The quadratic model used in the kth iteration is thus the one with its degree two monomials defined

1 Input: Center point $x \in \mathbb{R}^n$, bank of evaluated points $\mathcal{Y} = \{(y^1, f(y^1)), \ldots, (y^{|\mathcal{Y}|}, f(y^{|\mathcal{Y}|}))\}$ satisfying $(x, f(x)) \in \mathcal{Y}$, trust region radius Δ.

2 Initalize: Choose algorithmic constants $c \geq 1$, $\theta_1 \in (0, \frac{1}{c}]$.

3 Set $S = \{s^1\} = \{0_n\}$.

4 Set $S^\perp = I_n$.

5 for $i = 1, \ldots, |\mathcal{Y}|$ **do**

6 | **if** $\|y^i - x\| \leq c\Delta$ *and* $\left| proj_{S^\perp} \left(\frac{1}{c\Delta}(y^i - x) \right) \right| \geq \theta_1$ **then**

7 | | $S = S \cup \{y^i - x\}$

8 | | Update $S^\perp$ to be an orthonormal basis for $\mathcal{N}([s^2 \cdots s^{|S|}])$

9 | **end**

10 | **if** $|S| = n + 1$ **then**

11 | | **break** (the for loop)

12 | **end**

13 end

14 $S = [s^2, \cdots, s^{|S|}]$

15 $Q = [S \ \ S^\perp]$

16 Return: $S, S^\perp, Q$

Algorithm 2: Identify Initial Subspace

by β^* and degree one terms monomials by $\bar{g}^k$. This quadratic model is then minimized over a trust region to obtain a trial step, and a standard acceptance test and trust region radius update is performed.

5   Practical Considerations

In this section, we concern ourselves with two practical considerations, the first of which prevents Algorithm 1 from being directly implemented as written.

5.1   Choosing δ_k in Line 7

We recall that in Line 7 of Algorithm 1, we must compute some approximation δ_k, as defined in (14). As defined, δ_k is not particularly easy to approximate. This motivates several modifications to Algorithm 1, which we now describe and for which we provide some theoretical motivation.

We begin by making the key observation that the results concerning unbiasedness and variance of the estimator $\tilde{g}^k$, respectively in Theorem 2 and Theorem 3, hold *regardless of the value of the previous iteration's average estimator*, $\bar{g}^{k-1}$. Thus, for the sake of choosing an ameliorated estimator $\tilde{g}^k$ in each iteration, we will replace $\bar{g}^{k-1}$ in (9) with 0_n. In other words, we effectively replace (9) in Line 13 of Algorithm 1 with

$$\tilde{g}^k \leftarrow S_k^\top D(\pi^k, I_k) S_k \hat{g}^k. \tag{15}$$

Per Theorem 2, (15) is still an unbiased estimator of an approximation of $\nabla f(x)$, but it exhibits a different variance. However, when x^k approaches a stationary point, the vector 0_n ought to become an increasingly good initial approximation of the gradient near stationarity, and hence the variance of (15) as an estimator of $\nabla f(x^k)$ decreases proportionally with the stationarity $\|\nabla f(x^k)\|$.

The substitution of (15) has multiple practical effects on the overall logic of Algorithm 1. First, it is apparent from (15) that the gradient $\tilde{g}^k$ exists entirely in a (weighted) subspace determined by S_k. Thus, it is practically desirable to solve a lower dimensional (the rank of S_k) trust region subproblem in each iteration. To determine an appropriate Hessian approximation, we record several results, which are respectively extensions of Proposition 1, Theorem 2 and Theorem 3.

Proposition 2. *Denote*

$$\bar{H}^k := \mathrm{mat} \left(\begin{array}{cc} \arg\min\limits_{\alpha \in \mathbb{R}^{n \times n}} & \frac{1}{2}\|\mathrm{vec}(\alpha) - \mathrm{vec}(\bar{H}^{k-1})\|^2 \\ s.\ to & S_k \alpha S_k^\top = \hat{H}^{J_k} \end{array} \right) \tag{16}$$

The subproblem (16) *admits a closed-form solution given by*

$$\bar{H}^k = \bar{H}^{k-1} - S_k^\top S_k \bar{H}^{k-1} S_k^\top S_k + S_k^\top \hat{H}^{J_k} S_k. \tag{17}$$

Proof. The reasoning is essentially the same as in the proof of Proposition 1. The KKT conditions in (16) can be expressed as

$$\begin{aligned}
\text{vec}(\alpha) &= \text{vec}(\bar{H}^{k-1}) - (S_k^\top \otimes S_k^\top)\mu && (\mathit{stationarity}) \\
(S_k \otimes S_k)\text{vec}(\alpha) &= \text{vec}(\hat{H}^{J_k}) && (\mathit{primal\,feasibility}),
\end{aligned}$$

Plugging the stationary condition into the primal feasibility condition,

$$(S_k \otimes S_k)\text{vec}(\bar{H}^{k-1}) - (S_k \otimes S_k)(S_k^\top \otimes S_k^\top)\mu = \text{vec}(\hat{H}^{J_k}).$$

Using properties of the Kronecker product and the orthonormality of Q_k, this simplifies to

$$\mu = (S_k \otimes S_k)\text{vec}(\bar{H}^{k-1}) - \text{vec}(\hat{H}^{J_k}).$$

Plugging these Lagrange multipliers back into the stationarity condition, we obtain

$$\begin{aligned}
\text{vec}(\alpha) &= \text{vec}(\bar{H}^{k-1}) - (S_k^\top \otimes S_k^\top)(S_k \otimes S_k)\text{vec}(\bar{H}^{k-1}) - (S_k^\top \otimes S_k^\top)\text{vec}(\hat{H}^{J_k}) \\
&= \text{vec}(\bar{H}^{k-1}) - (S_k^\top S_k \otimes S_k^\top S_k)\text{vec}(\bar{H}^{k-1}) - (S_k^\top \otimes S_k^\top)\text{vec}(\hat{H}^{J_k}) \\
&= \text{vec}(\bar{H}^{k-1}) - S_k^\top S_k \bar{H}^{k-1} S_k^\top S_k - S_k^\top \hat{H}^{J_k} S_k,
\end{aligned}$$

where we have used the property that for appropriately sized matrices A, B, and C,

$$ABC = \text{mat}((C^\top \otimes A)\text{vec}(B)).$$

$\square$

We state the next two results without proof, but note that by using vec and mat operators as in the proof of Equation (17), the proofs are virtually the same as those of Theorem 2 and Theorem 3, respectively.

Theorem 5. *Suppose for each J_k, we can compute $\hat{H}_{J_k}$ satisfying $\|S_k^\top (\hat{H}_{J_k} - \nabla^2 f(x))S_k\| \leq \kappa_{eH}$ for all $x \in \mathcal{B}(x^k, \Delta_k)$ and for some $\kappa_{eH} \in [0, \infty)$. Let*

$$\tilde{H}^k = \bar{H}^{k-1} - S_k^\top D(\pi^k, J_k) S_k \bar{H}^{k-1} S_k^\top D(\pi^k, J_k) S_k + S_k^\top D(\pi^k, J_k)\hat{H}^{J_k} D(\pi^k, J_k) S_k. \tag{18}$$

Then,

$$\|\mathbb{E}_{J_k}[\tilde{H}^k] - \nabla^2 f(x)\| \leq n\kappa_{eH}$$

for all $x \in \mathcal{B}(x^k, \Delta_k)$.

Theorem 6. *The variance of $\tilde{H}^k$, with respect to the distribution governing J_k, is*

$$\mathbb{E}_{J_k}\left[\left\|\tilde{H}^k - \mathbb{E}_{J_k}\left[\tilde{H}^k\right]\right\|^2\right] = \left\|\bar{H}^{k-1} - \mathbb{E}_{J_k}\left[\tilde{H}^k\right]\right\|^2_{Q_k D_k Q_k^\top - I_n}. \tag{19}$$

We note that we could extend Definition 3 to produce a definition of $(S, \kappa_{ef}, \kappa_{eg}, \kappa_{eH})$-fully quadratic models of $f(x)$. Doing so would enable a $\mathcal{O}(\Delta_k)$ error bound (as opposed to $\mathcal{O}(1)$) in Theorem 5. However, because our model-building procedure employs the subproblem (5) and uses Algorithm 4 to determine an interpolation set that only guarantees an affinely independent subset of points, our algorithm is not intended to guarantee fully quadratic models in a subspace. Thus, such a fully quadratic extension is outside of the scope of this paper, but would be easily achieved. We also note a high-level similarity between the motivation for (16) and what is done in work on randomized Hessian estimation in [40].

Taken together, Proposition 2, Theorem 5 and Theorem 6 imply that, given the subspace gradient $D(\pi^k, J_k)S_k \tilde{g}^k \in \mathbb{R}^{rank(S_k)}$ implied by (15), and provided we maintain an average estimator $\bar{H}^{k-1}$ between iterations, a reasonable choice of corresponding model Hessian is the projection of the unbiased estimator $\tilde{H}^k$ into the subspace defined by S_k, that is, the model Hessian is $S_k \tilde{H}^k S_k^\top \in \mathbb{R}^{rank(S_k) \times rank(S_k)}$.

To summarize these changes to Algorithm 1, we

- Replace the ameliorated estimator update (9) with (15) in Line 13

- Maintain an average estimator for the Hessian $\bar{H}^k$ via the update (16) in addition to the average estimator for the Hessian $\bar{g}$ in Line 14.

- Replace the subproblem in Line 15 with the lower dimensional subproblem

$$y^* := \min_{y \in \mathbb{R}^{rank(S_k)}} \langle D(\pi^k, I_k) S_k \hat{g}^k, y \rangle + \frac{1}{2} y^\top S_k \tilde{H}^k S_k^\top y, \tag{20}$$

where $\tilde{H}^k$ is computed as in (18). The trial step is replaced with $d^k = S_k^\top y^*$.

With these changes made, a reasonable estimator for δ_k is simply $\bar{g}^k$, the (biased) estimate of a $\mathcal{O}(\Delta_k)$-accurate approximation to $\nabla f(x^k)$.

5.2 Choosing p_k in Line 7

While we do not intend to prove convergence results for Algorithm 1 in this paper, we note that Algorithm 1 can be readily analyzed as a trust region method with probabilistic models and deterministic function values, which is within the scope of [33]. Indeed, by viewing Algorithm 1 through the lens of probabilistic models, we will derive a practical method for dynamically selecting p_k. In light of Theorem 3, larger expected (and hence, realized) sketch sizes p_k will intuitively yield estimators of lower variance, suggesting more stable convergence to a local minimizer of f. In one extreme, and by our prior observations, the variance of the estimator $\tilde{g}^k$ is 0 when $p_k = n$; in this case, Algorithm 1 is just a deterministic DFO trust-region method. We now state a result about the probabilistic accuracy of our models, using conservative (Markov inequality-derived) concentration inequalities.

Theorem 7. *Let Assumption 1 hold. Suppose in the kth iteration of Algorithm 1, we denote*

$$V := \|\bar{g}^{k-1} - \mathbb{E}_{I_k}[\tilde{g}^k]\|^2_{Q_k D_k Q_k^\top - I_n},$$

the variance of $\tilde{g}^k$. Let $C > 0$ and let κ_{eg} be as in Theorem 1. If $V < n\kappa_{eg}^2 \Delta_k^2$, then,

$$\|\tilde{g}^k - \nabla f(x^k)\| \leq (C+1)\sqrt{n}\kappa_{eg}\Delta_k$$

with probability $1 - \frac{1}{C^2}$

Proof. By Chebyshev's inequality and the supposition on V,

$$\|\tilde{g} - \mathbb{E}_{I_k}[\tilde{g}^k]\| \leq C\sqrt{V} < C\sqrt{n}\kappa_{eg}\Delta_k$$

with probability $1 - \frac{1}{C^2}$. Combining this result with Theorem 2, we get the desired result. $\square$

In words, Theorem 7 shows that the models defined by $\tilde{g}^k$ nearly satisfy the definition of being probabilistically fully linear as defined by, for instance, [9, 15, 16, 33], when the variance V is sufficiently small. Thus, given the approximation of V (the quality of which is, of course, totally dependent on the quality of the estimate δ^k), Theorem 7 suggests employing the adaptive scheme in Algorithm 3 for choosing a sketch size p_k. In Algorithm 3, the constant C has effectively absorbed the unknown constant κ_{eg}. Choosing $C = 0$ will force the estimator $\tilde{g}^k$ to have zero variance (that is, we will choose each column of Q_k with probability one), while larger values of C will result in the estimator $\tilde{g}^k$ exhibiting proportionally more variance.

5.3 A note on the realization of J_k

In this paper, for simplicity, our subproblem for determining (14) only constrains the *expected* size of a realized sample $|J_k|$. In settings where function evaluations are particularly expensive, it may be very undesirable to under-utilize (or worse, over-utilize) available computational resources by not being able to control the actual, realized number of function evaluations performed in each iteration of Algorithm 1. In [41], we proposed a means to handle this via conditional Poisson sampling (see, e.g., [17, 18]). However, in this paper, we do not concern ourselves with this issue. Incorporating conditional Poisson sampling in the present work would be straightforward.

1 Input: Trust-region radius Δ_k, orthonormal matrix Q_k, accuracy parameter $C \geq 0$, error estimate δ^k, and minimal expected size of sample $b_0 \in (0, n)$.

2 $b \leftarrow b_0$.

3 while $b \leq n$ **do**

4 $\quad$ Compute π^k according to (13) with $p_k = b$ and δ^k.

5 $\quad$ **if** $\|\tilde{\delta}^k\|_{Q_k D_k Q_k^\top - I_n} \leq nC^2\Delta_k^2$ **then**

6 $\quad\quad$ **return** $p_k = b$.

7 $\quad$ **else**

8 $\quad\quad$ $b \leftarrow b + 1$.

9 $\quad$ **end**

10 end

<div align="center">Algorithm 3: Adaptive scheme to choose p_k</div>

6 Numerical Results

We implemented Algorithm 1 using the existing `Matlab` software for `POUNDers` [2] as a starting point. In particular, the same set of parameters dictating trust-region dynamics are used in both methods, the trust-region subproblems are solved in the same way, and as described previously, our implementations of Algorithm 2 and Algorithm 4 are very minor modifications of a model-building routine (`formquad.m`) that already existed in `POUNDers`.

While `POUNDers` was originally developed for nonlinear least squares problems (2), the most recent implementation allows for more general problems of the form (1). As an extension of this implementation of `POUNDers`, our implementation of Algorithm 1, which we call `SS-POUNDers`, or "subspace `POUNDers`", can solve problems of both the general form (1) as well as the specialized nonlinear least squares form (2). `POUNDers` (and hence, `SS-POUNDers`) achieves this by maintaining separate quadratic models of each component function $f_i(x)$, each parameterized by a component model gradient term $g_i \in \mathbb{R}^n$ and a component model Hessian term $H_i \in \mathbb{R}^{n \times n}$. In turn, these component models are combined to yield a full model gradient $g(x)$ and full model Hessian $H(x)$ via

$$g(x) = \sum_{i=1}^{p} f_i(x) g_i, \quad H(x) = \sum_{i=1}^{p} \left[g_i g_i^\top + H_i \right]. \tag{21}$$

By applying the average estimator update (7) to each component model gradient g_i and the update (17) to each component model Hessian H_i, it is clear by linearity that we are effectively maintaining average estimators of $g(x)$ and $H(x)$ in (21).

6.1 Parameter Settings

We use all the same default parameters as in `POUNDers`. In particular, in Algorithm 1, we use $\eta_1 = 0.05, \Delta_{\max} = 1000\Delta_0, \nu_1 = 0.5$, and $\nu_2 = 2.0$. In Algorithm 2, which is again derived from the model-building routine of `POUNDers`, we use the `POUNDers` default settings of $c = \sqrt{n}$ and $\theta_1 = 10^{-5}$. Similarly, in Algorithm 4, we use the `POUNDers` defaults of $c = \sqrt{n}$ and $\theta_2 = 10^{-3}$. In terms of parameters unique to `SS-POUNDers`, in Algorithm 1, we chose $\eta_2 = 10^{-3}$ and in Algorithm 3, we chose $C = 0.01\sqrt{n}$ and $b_0 = 1$.

Finally, we make the observation that Algorithm 1 intentionally leaves the choice of $\bar{g}^0$ open. Naturally, if one had knowledge of a reasonable estimate of $\nabla f(x^0)$ (from previously obtained function evaluations, for instance), then one should employ that. In the assumed absence of that information, and in our tests, we tried two natural, but very different, choices of $\bar{g}^0$. Our first choice was to proceed like `POUNDers` in the first iteration and simply spend $n + 1$ function evaluations to compute an initial ($\mathcal{O}(\Delta_0)$-accurate) simplex gradient. The obvious disadvantage to this approach is that it spends n function evaluations immediately. The advantage to this approach is that despite our amendments to Algorithm 1 discussed in Section 5.1, it is still important to maintain a good approximation $\bar{g}^k$ for the sake of obtaining reasonable variance estimates,

---

and hence sample sizes p_k, from Algorithm 3. Beginning the algorithm with a reasonably accurate $\bar{g}^0$ promises to lower the variance of estimators throughout the course of the algorithm. Our second choice naively set $\bar{g}^0 = 0_n$. This second choice has completely opposite advantages and disadvantages to the first choice. We pay no upfront cost to obtaining an "approximation" to $\nabla f(x^0)$, but the approximation is completely arbitrary and our variance estimates will accordingly be arbitrary, likely for many iterations. We found in preliminary testing that the former choice was practically superior, and so we implement that in `SS-POUNDers`, and demonstrate only the results stemming from that choice in what follows.

6.2 Test Problems

We employ two separate test sets, the Moré-Wild testset as implemented in the "Benchmarking DFO" (BenDFO) repoistory [1], and "Yet Another Test Set for Optimization" (YATSOP) repository [2]. Both test sets consist of unconstrained nonlinear least squares problems of the form (2). The former test set, which we simply refer to as BenDFO in the remainder, is better-known and was first compiled from (mostly) extant problems in [42]. For our purposes, we note that these 53 problems are all fairly low-dimensional, with n ranging between 2 and 12. The latter test set, which we simply refer to as YATSOp, consists of 38 problems and are significantly larger in dimension, with n ranging between 98 and 125. We chose all of the problems labelled "midscale" in YATSOp.

The decision to employ these two separate test sets was based on two considerations. First is the fact that they have both been previously employed in DFO literature; BenDFO has been used very frequently, and YATSOp most recently in [27], with a superset of problems similar to YATSOp employed in [13, 14]. Secondly, the difference in average problem sizes between the two test sets is important for this study. Our intention is to demonstrate that for small problems (as represented in BenDFO), the performance of `SS-POUNDers` is *comparable* to the performance of `POUNDers`. In low dimensions n, one intuitively expects that finding $n + 1$ affinely independent interpolation points is "more likely". That is, potentially expensive geometry-improvement steps in `POUNDers` are less likely to occur. However, as n increases, one expects (and often sees in practice) that finding $n + 1$ affinely independent points is "less likely", and `POUNDers` will therefore require geometry improvement steps more often. `SS-POUNDers`, on the other hand, does not have a geometry improvement step like `POUNDers`, because it only aims to ensure S_k-full linearity in each iteration, where S_k was initially chosen by Algorithm 2 and then randomly augmented with points that are well-poised in the nullspace of the initial subspace chosen by Algorithm 2. Based on this intuition, we expect that it is more likely to see `SS-POUNDers` outperform `POUNDers` on YATSOp than BenDFO, but we would hope that `SS-POUNDers` at least recovers the performance of `SS-POUNDers`.

6.3 Performance Profiles

We will use performance profiles (see, e.g., [42]) to illustrate the relative performance of `POUNDers` and `SS-POUNDers`. Given a tolerance $\tau > 0$, a set of solvers $\mathcal{S}$ and a set of problems $\mathcal{P}$, we count, for each combination of solver and problem, the number (normalized by $n + 1$, where n is the the problem dimension) $N_{solv,prob,\tau}$ of evaluations of $f(x)$ a solver $solv$ must make on problem $prob$ before it evaluates an x such that

$$f(x) \leq f(x^*) + \tau(f(x^0) - f(x^*)).$$

Remarkably, despite the nonconvexity of most of the problems in BenDFO and YATSOp, but owing to the original curation of these test sets, every run converges to a neighborhood of the known optimal solution x^*, and so the metric $N_{solv,prob,\tau}$ is reasonable for these tests.

A performance profile computes, for each problem $prob$,

$$N_{prob,\tau} = \min_{solv \in \mathcal{S}} N_{solv,prob,\tau}$$

and then plots $\alpha \geq 1$ on the x-axis and

$$\frac{1}{|\mathcal{P}|} |\{prob \in \mathcal{P} : N_{solv,prob,\tau} \leq \alpha N_{prob,\tau}\}|$$

Figure 1: Performance profiles comparing the performance of `POUNDers` with the *median* performance of `SS-POUNDers` on the (low-dimensional) BenDFO test set with convergence tolerances $\tau = 10^{-1}$ (left), $\tau = 10^{-3}$ (center), and $\tau = 10^{-5}$ (right).
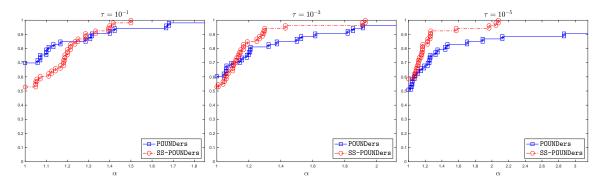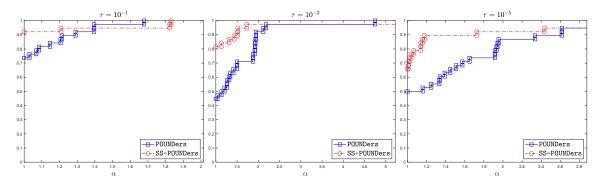

Figure 2: Performance profiles comparing the performance of `POUNDers` with the *median* performance of `SS-POUNDers` on the (higher-dimensional) YATSOp test set with convergence tolerances $\tau = 10^{-1}$ (left), $\tau = 10^{-3}$ (center), and $\tau = 10^{-5}$ (right).

on the y-axis for each solver $solv \in \mathcal{S}$. In words, a performance profile shows, as a function of α, the percentage of problems solved to tolerance τ by each solver within a number of budget units no more than α times larger than the least number of budget units required by any solver on the same problem.

We run `SS-POUNDers` a total of 30 times on each problem, each with a different random seed. In the profiles in Figure 1, we illustrate the *median* performance of `SS-POUNDers` on BenDFO.

In the profiles in Figure 2, we illustrate the median performance of of `SS-POUNDers` on YATSOp. In both BenDFO and YATSOp, we see that as the convergence tolerance τ becomes tighter, there is an argument for an increasing preference for `SS-POUNDers`. In fact, when we demand a tight convergence tolerance ($\tau = 10^{-5}$), these results demonstrate that the median performance of `SS-POUNDers` completely dominates the performance of `POUNDers`. Although not a perfect explanation, one intuition for this phenomenon may be that the longer an algorithm runs (and hence becomes closer to convergence), the more opportunities `SS-POUNDers` is presented to randomize, and hence avoid the geometry improvement steps that `POUNDers` must take.

Figure 1 and Figure 2 are certainly encouraging, but because `SS-POUNDers` is a *randomized* method, we must demonstrate that performance is reasonable even in the worst case, in order to demonstrate the robustness of a randomized approach. Towards that end, we use the same data as in the previous figures, but show the *worst-case* performance of `SS-POUNDers` across all 30 trials performed. These results are shown in Figure 3 and Figure 4.

The results in Figure 3 show that, in this lower-dimensional setting of BenDFO problems, randomization can, *in the worst case*, harm the method fairly considerably. with the majority of problems being solved up to 5 times slower by `SS-POUNDers` than by `POUNDers`. However, in line with our previous intuition, on the higher dimensional problems, we see that for tighter convergence tolerances ($\tau \in \{10^{-3}, 10^{-5}\}$) there remains a compelling argument for employing `SS-POUNDers` over `POUNDers` across most of the test set, *even*

Figure 3: Performance profiles comparing the performance of POUNDers with the *worst-case* performance of SS-POUNDers on the (low-dimensional) BenDFO test set with convergence tolerances $\tau = 10^{-1}$ (left), $\tau = 10^{-3}$ (center), and $\tau = 10^{-5}$ (right).
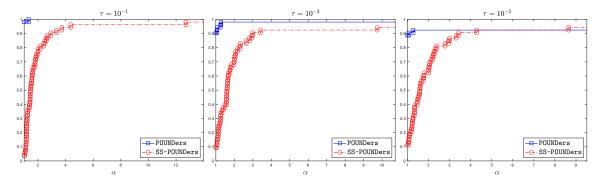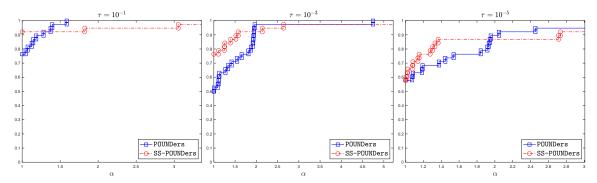

Figure 4: Performance profiles comparing the performance of POUNDers with the *worst-case* performance of SS-POUNDers on the (higher-dimensional) YATSOp test set with convergence tolerances $\tau = 10^{-1}$ (left), $\tau = 10^{-3}$ (center), and $\tau = 10^{-5}$ (right).

in the worst case. These are encouraging results, but suggest further empirical studies to better a priori quantify the dimension n at which `SS-POUNDers` is, with high probability, more efficient in its use of function evaluations than `POUNDers`.

7   Conclusion

In this paper, we presented a novel randomized algorithm for DFO, which we named *basis sketching*. The algorithm employs a running average estimator of an interpolation model gradient and Hessian, as well as an unbiased counterpart, the ameliorated estimator. By using these estimators, basis sketching does *not* require the maintenance of a fully linear model in every iteration of a model-based trust region method. Using `POUNDers` as a starting point, we implemented a basis sketching method called `SS-POUNDers`, which never employs geometry improvement to yield fully linear models, but instead only ensures fully linearity restricted to randomized subspaces realized from judiciously chosen probability distributions. Our numerical results demonstrated, on problems with roughly $n = 100$ variables, a fairly clear preference for using `SS-POUNDers` over `POUNDers`.

A   Proof of Theorem 1

Proof. For simplicity of notation, let

$$e^m(d) = m(y^0 + S^\top d) - f(y^0 + S^\top d) \quad \text{and} \quad e^g(d) = S\nabla m(y^0 + S^\top d) - S\nabla f(y^0 + S^\top d)$$

for $d \in \mathbb{R}^p$. Without loss of generality, shift all points in Y by y^0 so that $y^0 = 0$ (and hence, $\{y^0, y^1, \ldots, y^p\} \subset \mathcal{B}(0_p; c\Delta)$, and each column of $\tilde{Y}$ is Sy^i). Towards proving a result about S-full linearity, suppose that d satisfies $\|S^\top d\| \leq c\Delta$. By Assumption 1 and our assumption on m, we may take a first-order Taylor expansion of f about y^0, and so for $i = 0, 1, \ldots, p$ we have

$$
\begin{aligned}
\langle e^g(d), Sy^i - d \rangle &= \int_0^1 \langle S\nabla f(y^0 + S^\top d + t(y^i - S^\top d)) - S\nabla f(y^0 + S^\top d), Sy^i - d \rangle \mathrm{d}t \\
&\quad - \int_0^1 \langle S\nabla m(y^0 + S^\top d + t(y^i - S^\top d)) - S\nabla m(y^0 + S^\top d), Sy^i - d \rangle \mathrm{d}t \\
&\quad - e^m(d).
\end{aligned}
\tag{22}
$$

Subtracting (22) with $i = 0$ from each of (22) with $i = 1, \ldots, p$, we have

$$
\begin{aligned}
\langle e^g(d), Sy^i \rangle = \ &\int_0^1 \langle S\nabla f(y^0 + S^\top d + t(y^i - S^\top d)) - S\nabla f(y^0 + S^\top d), Sy^i - d \rangle \mathrm{d}t \\
&- \int_0^1 \langle S\nabla m(y^0 + S^\top d + t(y^i - S^\top d)) - S\nabla m(y^0 + S^\top d), Sy^i - d \rangle \mathrm{d}t \\
&- \int_0^1 \langle S\nabla f(y^0 + (1 - t)S^\top d) - S\nabla f(y^0 + S^\top d), -d \rangle \mathrm{d}t \\
&+ \int_0^1 \langle S\nabla m(y^0 + (1 - t)S^\top d) - S\nabla m(y^0 + S^\top d), -d \rangle \mathrm{d}t
\end{aligned}
\tag{23}
$$

for $i = 1, \ldots, p$. We now bound each row of the right hand side of (23) and will combine the bounds by Cauchy-Schwarz inequality. The first row can be bounded as

$$
\begin{aligned}
& \int_0^1 \langle S\nabla f(y^0 + S^\top d + t(y^i - S^\top d)) - S\nabla f(y^0 + S^\top d), Sy^i - d\rangle \mathrm{d}t \\
={} & \int_0^1 \langle \nabla f(y^0 + S^\top d + t(y^i - S^\top d)) - \nabla f(y^0 + S^\top d), S^\top Sy^i - S^\top d\rangle \mathrm{d}t \\
={} & \int_0^1 \langle \nabla f(y^0 + S^\top d + t(y^i - S^\top d)) - \nabla f(y^0 + S^\top d), y^i - S^\top d\rangle \mathrm{d}t \\
\leq{} & \int_0^1 \|\nabla f(y^0 + S^\top d + t(y^i - S^\top d)) - \nabla f(y^0 + S^\top d)\| \|y^i - S^\top d\| \mathrm{d}t \\
\leq{} & \int_0^1 L_g \|y^i - S^\top d\|^2 t \mathrm{d}t = \frac{1}{2} L_g \|y^i - S^\top d\|^2 \leq 2L_g c^2 \Delta^2,
\end{aligned}
$$

where the second equality comes from the mutual orthonormality of the rows of S and the fact that $y^i = S^\top \delta^i$ for some δ^i, and the last line comes from the Lipschitz continuity of Assumption 1 and the fact that $\|y^i - S^\top d\| \leq \|y^i\| + \|S^\top d\| \leq 2c\Delta$. Bounding the remaining three lines of the right hand side of (23) is similar; we arrive at a final bound of

$$
\langle e^g(d), Sy^i\rangle \leq \frac{5}{2} c^2 (L_g + L_{mg})\Delta^2.
$$

Using our matrix notation, the left hand side of these p equations can be written $\tilde{Y}^\top e^g(d)$, and so we have the trivial bound

$$
\|\tilde{Y}^\top e^g(d)\| \leq \sqrt{p} \frac{5}{2} c^2 (L_g + L_{mg})\Delta^2.
$$

By our supposition on $\|\tilde{Y}^{-1}\|$,

$$
\|e^g(d)\| = \|\tilde{Y}^{-1}\tilde{Y} e^g(d)\| \leq \|\tilde{Y}^{-1}\| \|\tilde{Y} e^g(d)\| \leq \sqrt{p} \frac{5\Lambda}{2} c(L_g + L_{mg})\Delta,
$$

which is the value of κ_{eg} that we intended to show.

To derive the value of κ_{ef}, we return to (22) for $i = 0$ and note that

$$
e^m(d) = \int_0^1 \langle S\nabla f((1-t)S^\top d) - S\nabla f(S^\top d), -d\rangle \mathrm{d}t - \int_0^1 \langle S\nabla m((1-t)S^\top d) - S\nabla m(S^\top d), -d\rangle \mathrm{d}t + \langle e^g(d), d\rangle,
$$

and so by similar reasoning to our previous derivations,

$$
\begin{aligned}
|e^m(d)| \leq 2(L_g + L_{mg})c^2\Delta^2 + \|e^g(d)\| \|d\| & \leq 2(L_g + L_{mg})c^2\Delta^2 + \sqrt{p}\frac{5\Lambda}{2}c^2(L_g + L_{mg})\Delta^2. \\
& = \frac{4 + 5\Lambda\sqrt{p}}{2}(L_g + L_{mg})c^2\Delta^2,
\end{aligned}
$$

as we meant to show. $\qquad\square$

B Algorithm 4

Algorithm 4 effectively operates by considering the effect of adding points to a partitioning of the Vandermonde matrix employed in the constraint of (5). We note that up to notation and subspace considerations, the statement and subsequent development of Algorithm 4 is a very close analogue of [60][Algorithm 4.2].

To ease our notation, for a given pair of orthogonal subspaces $S, S^\perp$, and for a given interpolation set Y of size p, we will slightly abuse previous notation and abbreviate

$$
M_S(Y) = M(\Phi_S, Y) \quad \text{and} \quad M_{S^\perp}(Y) = M(\Phi_{S^\perp} \cup \{\Phi_Q \setminus \Phi_L\}, Y).
$$

We can easily see that the stationarity and primal feasibility conditions from the KKT conditions of (5) can be arranged as the saddle-point system

$$
\begin{bmatrix} -M_{S^\perp}(Y)M_{S^\perp}(Y)^\top & M_S(Y) \\ M_S(Y)^\top & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \alpha \end{bmatrix} = \begin{bmatrix} f(Y) \\ 0 \end{bmatrix}, \tag{24}
$$

1 Input: Subspaces S and $S^\perp$, initial set of points $Z = \{z^1 = x, z^2, \ldots, z^{rank(S)+1}\} \subset \mathbb{R}^n$, bank of
   evaluated points $\mathcal{Y} = \{(y^1, f(y^1)), \ldots, (y^{|\mathcal{Y}|}, f(y^{|\mathcal{Y}|}))\}$ satisfying $(z^i, f(z^i)) \in \mathcal{Y}$ for all $i = 1, \ldots, |Z|$,
   trust region radius Δ.
2 Initialize: Choose algorithmic constants $c \geq 1$, $\theta_2 > 0$.
3 Compute QR decomposition of $M(\Phi_S, Z)$, initialize $N = \emptyset$.
4 for $i = 1, \ldots, |\mathcal{Y}|$ **do**
5   **if** $y^i \notin Z$ *and* $\|y^i - x\| \leq c\Delta$ **then**
6   Compute N_+ as in (26).
7   **if** $\sigma_{\min}(N_+) \geq \theta_2$ **then**
8 $Z \leftarrow Z \cup \{y^i\}$
9 $N \leftarrow N_+$
10   **end**
11   **end**
12 end

Algorithm 4: Determine Interpolation Set Y

Let N denote an orthogonal basis for the null space $\mathcal{N}(M_S(Y)^\top)$ and let $QR = M_S(Y)$ be a QR factorization. Because the second row of (24) indicates that $\lambda \in \mathcal{N}(M_S(Y)^\top)$, we can write $\lambda = N\omega$ for $\omega \in \mathbb{R}^{p-rank(S)-1}$ and so (24) reduces to p equations

$$\begin{aligned} N^\top M_{S^\perp}(Y) M_{S^\perp}(Y)^\top N\omega &= N^\top f(Y) \\ R\alpha &= Q^\top (f(Y) - M_{S^\perp}(Y) M_{S^\perp}(Y)^\top N\omega). \end{aligned} \tag{25}$$

We can now state and prove a very close analogue of [60][Theorem 3.2].

Theorem 8. *Suppose* $s := rank(S) \geq 2$. *If both*

- $rank(M_S(Y)^\top) = s + 1$, *and*

- $N^\top M_{S^\perp}(Y) M_{S^\perp}(Y)^\top N$ *is positive definite,*

then, for any $f(Y) \in \mathbb{R}^p$, *there exists a unique solution* $(\alpha^*, \beta^*, \gamma^*)$ *to* (5).

Proof. Let both suppositions hold. Because $N^\top M_{S^\perp}(Y) M_{S^\perp}(Y)^\top N$ is positive definite, we have that $M_{S^\perp}(Y)^\top N$ is full rank. Because $s \geq 2$, $M_{S^\perp}(Y)^\top N$ being full rank in turn implies that its nullspace satisfies $\mathcal{N}(M_{S^\perp}(Y)^\top N) = \{0_{p-s-1}\}$. Since the columns of N form a basis for $\mathcal{N}(M_S(Y)^\top)$, this implies that $\mathcal{N}(M_{S^\perp}(Y)^\top) \cap \mathcal{N}(M_S(Y)^\top) = \{0_{p-s-1}\}$, which means that the full Vandermonde matrix $M(\Phi, Y) = [M_S(Y) \;\; M_{S^\perp}(Y)]$ is full rank. Thus, the feasible region of (5) is nonempty.

Because the objective of (5) is convex in β and γ, both he optimal solution $(\alpha^*, \beta^*, \gamma^*)$ to (5) and the Lagrange multipliers λ in (24) are unique, as we meant to show. $\qquad\square$

With Theorem 8 in hand, we see that the intention of Algorithm 4 is two-fold; it is designed to ensure that the second condition of Theorem 8 holds, while maintaining reasonable conditioning of the linear system in (25). Algorithm 4 computes an initial QR decomposition of the square matrix $M_S(Z)$ and maintains in the matrix N an orthogonal basis for $\mathcal{N}(M_S(Z)^\top)$. In a greedy fashion, if a point y^i in the bank $\mathcal{Y}$ is not already in the interpolation set Z and is sufficiently close to x, we consider the effect of adding it to Z. By performing $s+1$ many Givens rotations to the initial QR factorization of $M(\Phi_S, Y)$, we obtain an orthogonal basis for $\mathcal{N}(M_S(Y \cup \{y^i\})^\top)$ of the form

$$N_+ = \begin{bmatrix} N & Q\phi_1 \\ 0 & \phi_2 \end{bmatrix}.$$

Thus, we can easily update

$$M_{S^\perp}(Y \cup \{y^i\})^\top N_+ = \begin{bmatrix} M_{S^\perp}(Y)^\top N, & M_{S^\perp}(Y)^\top Q\phi_1 + \phi_2 M_{S^\perp}(\{y^i\})^\top \end{bmatrix} \tag{26}$$

Thus, by ensuring the least eigenvalue of N_+ is bounded away from 0 via the algorithmic parameter θ_2, the output of Algorithm 4 will guarantee that the second condition of Theorem 8 is met. We note that the first condition is automatically met within the scope of Algorithm 1, since the initial set Z will already contain the affinely independent points selected by Algorithm 2, and adding to Z cannot change the subspace that they span.

References

[1] BenDFO: Code for benchmarking derivative-fre optimization algorithms, 2022.

[2] YATSOp: Yet another test set for optimization, 2022.

[3] Z. Allen-Zhu, Z. Qu, P. Richtárik, and Y. Yuan. Even faster accelerated coordinate descent using non-uniform sampling. In *International Conference on Machine Learning*, pages 1110–1119. PMLR, 2016.

[4] C. Audet and W. L. Hare. *Derivative-Free and Blackbox Optimization*. Springer, 2017.

[5] A. S. Bandeira, K. Scheinberg, and L. N. Vicente. Computation of sparse low degree interpolating polynomials and their application to derivative-free optimization. *Mathematical Programming*, 134(1):223–257, 2012.

[6] A. S. Berahas, R. Bollapragada, and J. Nocedal. An investigation of newton-sketch and subsampled newton methods. *Optimization Methods and Software*, 35(4):661–680, 2020.

[7] A. S. Berahas, L. Cao, K. Choromanski, and K. Scheinberg. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Foundations of Computational Mathematics*, 22:507–560, 2022.

[8] E. H. Bergou, E. Gorbunov, and P. Richtárik. Stochastic three points method for unconstrained smooth minimization. *SIAM Journal on Optimization*, 30(4):2726–2749, 2020.

[9] J. Blanchet, C. Cartis, M. Menickelly, and K. Scheinberg. Convergence rate analysis of a stochastic trust-region method via supermartingales. *INFORMS Journal on Optimization*, 1(2):92–119, 2019.

[10] R. Bollapragada, M. Menickelly, W. Nazarewicz, J. O'Neal, P.-G. Reinhard, and S. M. Wild. Optimization and supervised machine learning methods for fitting numerical physics models without derivatives. *Journal of Physics G: Nuclear and Particle Physics*, 48(2):024001, 2021.

[11] C. Cartis, J. Fowkes, and Z. Shao. A randomised subspace gauss-newton method for nonlinear least-squares. *arXiv preprint arXiv:2211.05727*, 2022.

[12] C. Cartis, J. Fowkes, and Z. Shao. Randomised subspace methods for non-convex optimization, with applications to nonlinear least-squares. *arXiv preprint arXiv:2211.09873*, 2022.

[13] C. Cartis and L. Roberts. A derivative-free gauss–newton method. *Mathematical Programming Computation*, 11:631–674, 2019.

[14] C. Cartis and L. Roberts. Scalable subspace methods for derivative-free nonlinear least-squares optimization. *Mathematical Programming*, 2022. To appear.

[15] C. Cartis and K. Scheinberg. Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *Mathematical Programming*, 169(2):337–375, 2018.

[16] R. Chen, M. Menickelly, and K. Scheinberg. Stochastic optimization using a trust-region method and random models. *Mathematical Programming*, 169(2):447–487, 2018.

[17] S. X. Chen. General properties and estimation of conditional Bernoulli models. *Journal of Multivariate Analysis*, 74(1):69–87, 2000.

[18] X.-H. Chen, A. P. Dempster, and J. S. Liu. Weighted finite population sampling to maximize entropy. *Biometrika*, 81(3):457–469, 1994.

[19] B. Colson and P. L. Toint. Optimizing partially separable functions without derivatives. *Optimization Methods and Software*, 20(4-5):493–508, 2005.

[20] A. R. Conn, K. Scheinberg, and P. L. Toint. On the convergence of derivative-free methods for unconstrained optimization. In A. Iserles and M. Buhmann, editors, *Approximation Theory and Optimization: Tributes to M. J. D. Powell*, pages 83–108. Cambridge University Press, 1997.

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

[22] A. R. Conn, K. Scheinberg, and L. N. Vicente. Geometry of sample sets in derivative free optimization: Polynomial regression and underdetermined interpolation. *IMA Journal of Numerical Analysis*, 28(4):721–748, 2008.

[23] A. R. Conn, K. Scheinberg, and L. N. Vicente. Global convergence of general derivative-free trust-region algorithms to first and second order critical points. *SIAM Journal on Optimization*, 20(1):387–415, 2009.

[24] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization*. SIAM, 2009.

[25] S. Dasgupta and A. Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

[26] A. Defazio, F. R. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1646–1654, 2014.

[27] K. J. Dzahini and S. M. Wild. Stochastic trust-region algorithm in random subspaces with convergence and expected complexity analyses. Technical Report 2207.06452, ArXiv, 2022.

[28] R. Gower, D. Goldfarb, and P. Richtárik. Stochastic block bfgs: Squeezing more curvature out of data. In *International Conference on Machine Learning*, pages 1869–1878. PMLR, 2016.

[29] R. Gower, D. Kovalev, F. Lieder, and P. Richtárik. Rsn: Randomized subspace newton. *Advances in Neural Information Processing Systems*, 32, 2019.

[30] R. M. Gower and P. Richtárik. Randomized iterative methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1660–1690, 2015.

[31] R. M. Gower, P. Richtárik, and F. Bach. Stochastic quasi-gradient methods: Variance reduction via jacobian sketching. *Mathematical Programming*, 188:135–192, 2021.

[32] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang. Direct search based on probabilistic descent. *SIAM Journal on Optimization*, 25(3):1515–1541, 2015.

[33] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang. Complexity and global rates of trust-region methods based on probabilistic models. *IMA Journal Of Numerical Analysis*, 38(3):1579–1597, 2018.

[34] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang. Direct search based on probabilistic feasible descent for bound and linearly constrained problems. *Computational Optimization and Applications*, 72(3):525–559, 2019.

[35] J. C. Gross and G. T. Parks. Optimization by moving ridge functions: derivative-free optimization for computationally intensive functions. *Engineering Optimization*, 54(4):553–575, 2022.

[36] F. Hanzely, N. Doikov, Y. Nesterov, and P. Richtarik. Stochastic subspace cubic newton method. In *International Conference on Machine Learning*, pages 4027–4038. PMLR, 2020.

[37] F. Hanzely, K. Mishchenko, and P. Richtárik. Sega: Variance reduction via gradient sketching. *Advances in Neural Information Processing Systems*, 31, 2018.

[38] F. Hanzely and P. Richtarik. Accelerated coordinate descent with arbitrary sampling and best rates for minibatches. In K. Chaudhuri and M. Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89, pages 304–312. PMLR, 2019.

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

[40] D. Leventhal and A. Lewis. Randomized hessian estimation and directional search. *Optimization*, 60(3):329–345, 2011.

[41] M. Menickelly and S. M. Wild. Stochastic average model methods. Technical Report 2207.06305, ArXiv, 2022.

[42] J. J. Moré and S. M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009.

[43] Y. Nesterov and V. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[44] A. Neumaier, H. Fendl, H. Schilly, and T. Leitner. VXQR: derivative-free unconstrained optimization based on QR factorizations. *Soft Computing*, 15(11):2287–2298, 2011.

[45] M. Pilanci and M. J. Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017.

[46] M. J. D. Powell. UOBYQA: unconstrained optimization by quadratic approximation. *Mathematical Programming*, 92:555–582, 2002.

[47] M. J. D. Powell. On trust region methods for unconstrained minimization without derivatives. *Mathematical Programming*, 97:605–623, 2003.

[48] M. J. D. Powell. Least Frobenius norm updating of quadratic models that satisfy interpolation conditions. *Mathematical Programming*, 100(1):183–215, 2004.

[49] M. J. D. Powell. On the use of quadratic models in unconstrained minimization without derivatives. *Optimization Methods and Software*, 19(3–4):399–411, 2004.

[50] M. J. D. Powell. The NEWUOA software for unconstrained optimization without derivatives. In G. Di Pillo and M. Roma, editors, *Large-Scale Nonlinear Optimization*, volume 83 of *Nonconvex Optimization and its Applications*, pages 255–297. Springer, 2006.

[51] M. J. D. Powell. Developments of NEWUOA for minimization without derivatives. *IMA Journal of Numerical Analysis*, 28(4):649–664, 2008.

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

[53] M. J. D. Powell. On the convergence of trust region algorithms for unconstrained minimization without derivatives. *Computational Optimization and Applications*, 53(2):527–555, 2012.

[54] M. J. D. Powell. Beyond symmetric Broyden for updating quadratic models in minimization without derivatives. *Mathematical Programming*, 138(1-2):475–500, 2013.

[55] Z. Qu and P. Richtárik. Coordinate descent with arbitrary sampling i: Algorithms and complexity. *Optimization Methods and Software*, 31(5):829–857, 2016.

[56] P. Richtárik and M. Takáč. On optimal probabilities in stochastic coordinate descent methods. *Optimization Letters*, 10(6):1233–1243, 2016.

[57] M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[58] H.-J. M. Shi, S. Tu, Y. Xu, and W. Yin. A primer on coordinate descent algorithms. *arXiv preprint arXiv:1610.00040*, 2016.

[59] S. M. Wild. *Derivative-Free Optimization Algorithms for Computationally Expensive Functions*. PhD thesis, Cornell University, 2008.

[60] S. M. Wild. MNH: A derivative-free optimization algorithm using minimal norm Hessians. In *Tenth Copper Mountain Conference on Iterative Methods*, 2008.

[61] S. M. Wild. Solving derivative-free nonlinear least squares problems with POUNDERS. In T. Terlaky, M. F. Anjos, and S. Ahmed, editors, *Advances and Trends in Optimization with Engineering Applications*, pages 529–540. SIAM, 2017.

[62] S. M. Wild, R. G. Regis, and C. A. Shoemaker. ORBIT: Optimization by radial basis function interpolation in trust-regions. *SIAM Journal on Scientific Computing*, 30(6):3197–3219, 2008.

[63] S. J. Wright. Coordinate descent algorithms. *Mathematical programming*, 151(1):3–34, 2015.