DERIVATIVE-FREE OPTIMIZATION OF EXPENSIVE FUNCTIONS WITH COMPUTATIONAL ERROR USING WEIGHTED REGRESSION

STEPHEN C. BILLUPS*, JEFFREY LARSON†, AND PETER GRAF‡

Abstract. We propose a derivative-free algorithm for optimizing computationally expensive functions with computational error. The algorithm is based on the trust region regression method by Conn, Scheinberg, and Vicente [4], but uses weighted regression to obtain more accurate model functions at each trust region iteration. A heuristic weighting scheme is proposed which simultaneously handles i) differing levels of uncertainty in function evaluations, and ii) errors induced by poor model fidelity. We also extend the theory of Λ-poisedness and strong Λ-poisedness to weighted regression. We report computational results comparing interpolation, regression, and weighted regression methods on a collection of benchmark problems. Weighted regression appears to outperfrom interpolation and regression models on nondifferentiable functions and functions with deterministic noise.

Key words. Derivative-Free Optimization, Weighted Regression Models, Noisy Function Evaluations.

AMS subject classifications. 90C56, 49J52, 65K05, 49M30

1. Introduction.

The algorithm described in this paper is designed to optimize functions evaluated by large computational codes, taking minutes, hours or even days for a single function call, for which derivative information is unavailable, and for which function evaluations are subject to computational error. Such error may be deterministic (arising, for example, from discretization error), or stochastic (for example, from Monte-Carlo simulation). Because function evaluations are extremely expensive, it is sensible to perform substantial work at each iteration to reduce the number of function evaluations required to obtain an optimum.

In some cases, the magnitude of the uncertainty for each function evaluation can be controlled by the user. For example, accuracy can be improved by using a finer discretization or by increasing the sample size of a Monte-Carlo simulation. But this greater accuracy comes at the cost of increased computational time; so it makes sense to vary the accuracy requirements over the course of the optimization. Such an approach was proposed, for example, in [1]. Our algorithm is designed to take advantage of varying levels of uncertainty.

The algorithm fits into a general framework for derivative-free trust region methods using quadratic models, which was described by Conn, Scheinberg, and Vicente in [6, 5]. We shall refer to this framework as the *CSV2-framework*. This framework constructs a quadratic model function $m_k(\cdot)$, which approximates the objective function f on a set of sample points $Y^k \subset \mathbb{R}^n$ at each iteration k. The next iterate is then determined by minimizing m_k over a trust region. In order to guarantee global convergence, the CSV2-framework monitors the distribution of points in the sample set, and occasionally invokes a model-improvement algorithm that modifies the sample set to ensure m_k accurately approximates f. The CSV2-framework is the basis

*University of Colorado Denver, Department of Mathematical and Statistical Sciences (Stephen.Billups@ucdenver.edu).

†University of Colorado Denver, Department of Mathematical and Statistical Sciences (Jeffrey.Larson@ucdenver.edu), research partially supported by National Science Foundation Grant GK-12-0742434.

‡National Renewable Energy Laboratory, Computational Science Center (Peter.Graf@nrel.gov).

of the well-known DFO algorithm which is freely available on the COIN-OR website [17].

There have been previous attempts at optimizing noisy functions without derivatives. For functions with stochastic noise, replications of function evaluations can be a simple way to modify existing algorithms. For example, [8] modifies Powell's UOBYQA [21], [24] modifies Nelder-Mead [20], and [9] modifies DIRECT [15]. For deterministic noise, Kelley [16] proposes a technique to detect and restart Nelder-Mead methods. Neumaier's SNOBFIT [14] algorithm accounts for noise by not requiring the surrogate functions to interpolate function values, but rather fit a stochastic model. A least-squares regression method for handling noise was considered in [4].

In our paper, we propose using *weighted* regression, which can handle differing levels of uncertainty for each function evaluation. We propose a weighting scheme that can use knowledge of the relative levels of uncertainty in each function evaluation. The weighting scheme also handles errors resulting from poor model fidelity (i.e. fitting a non-quadratic function by a quadratic model) by taking into account the distances of each sample point to the center of the trust region.

To fit our algorithm into the CSV2-framework we extend the theory of poisedness, as described in [6], to weighted regression. We show (Proposition 4.11) that a sample set that is strongly Λ-poised in the regression sense is also strongly $c\Lambda$-poised in the *weighted* regression sense for some constant c, provided that no weight is too small relative to the other weights. Thus, any model improvement scheme that ensures strong Λ-poisedness in the regression sense can be used in the weighted regression framework.

The paper is organized as follows. We begin by describing the CSV2 framework in §2. To put our algorithm into this framework, we describe 1) how model functions are constructed (§3), and 2) a model improvement algorithm (§5). Before describing the model improvement algorithm, we first extend the theory of Λ-poisedness to the weighted regression framework (§4). Computational results are presented in §6 using a heuristic weighting scheme, which is described in that section. §7 concludes the paper.

Notation. The following notation will be used: $\mathbb{R}^n$ denotes the real Euclidean space of vectors of length n. $\|\cdot\|_p$ denotes the standard ℓ_p vector norm, and $\|\cdot\|$ (without the subscript) denotes the Euclidean norm. $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. C^k denotes the set of functions on $\mathbb{R}^n$ with k continuous derivatives. $D^j f$ denotes the jth derivative of a function $f \in C^k$, $j \leq k$. Given an open set $\Omega \in \mathbb{R}^n$, $LC^k(\Omega)$ denotes the set of C^k functions with Lipschitz continuous kth derivatives. That is, for $f \in LC^k(\Omega)$, there exists a Lipschitz constant L such that

$$\left\|D^k f(y) - D^k f(x)\right\| \leq L \|y - x\| \quad \text{for all } x, y \in \Omega.$$

$\mathcal{P}_n^d$ denotes the space of polynomials of degree less than or equal to d in $\mathbb{R}^n$; q_1 denotes the dimension of $\mathcal{P}_n^2$ (specifically $q_1 = (n+1)(n+2)/2$). We use standard "big-Oh" notation (written $\mathcal{O}(\cdot)$) to state, for example, that for two functions on the same domain, $f(x) = \mathcal{O}(g(x))$ if there exists a constant M such that $|f(x)| \leq M|g(x)|$ for all x with sufficiently small norm. Given a set Y, $|Y|$ denotes the cardinality and $\text{conv}(Y)$ denotes the convex hull. For a real number α, $\lfloor \alpha \rfloor$ denotes the greatest integer less than or equal to α. For a matrix A, A^+ denotes the Moore-Penrose generalized inverse [11]. e^j denotes the jth column of the identity matrix. The ball of radius Δ centered at $x \in \mathbb{R}^n$ is denoted $B(x; \Delta)$. Given a vector w, $\text{diag}(w)$ denotes the diagonal matrix W with diagonal entries $W_{ii} = w_i$. For a square matrix A, $\text{cond}(A)$ denotes the condition number, and $\lambda_{min}(A)$ denotes the smallest eigenvalue.

2. Background. The algorithm proposed in this paper fits into a general framework for derivative-free trust region methods using quadratic models described by Conn, Scheinberg, and Vicente [6, Algorithm 10.3]. We shall refer to this framework as the *CSV2-framework*. Algorithms in the framework construct a model function $m_k(\cdot)$ at iteration k, which approximates the objective function f on a set of sample points $Y^k = \{y^0, \ldots, y^{p_k}\} \subset \mathbb{R}^n$. The next iterate is then determined by minimizing m_k. Specifically, given the iterate x^k, a putative next iterate is given by $x^{k+1} = x^k + s^k$, where the step s^k solves the trust region subproblem

$$\begin{array}{ll} \min\limits_{s} & m_k(x^k + s) \\ \text{subject to} & \|s\| \leq \Delta_k \end{array},$$

where the scalar $\Delta_k > 0$ denotes the trust region radius, which may vary from iteration to iteration. If x^{k+1} produces sufficient descent in the model function, then $f(x^{k+1})$ is evaluated, and the iterate is accepted if $f(x^{k+1}) < f(x^k)$; otherwise, the step is not accepted. In either case, the trust region radius may be adjusted, and a model-improvement algorithm may be called to obtain a more accurate model.

To establish convergence properties, the following smoothness assumption is made on f:

ASSUMPTION 2.1. *Suppose that a set of points $S \subset \mathbb{R}^n$ and a radius $\Delta_{\max}$ are given. Let Ω be an open domain containing the $\Delta_{\max}$ neighborhood $\bigcup_{x \in S} B(x; \Delta_{\max})$ of the set S. Assume $f \in LC^2(\Omega)$ with Lipschitz constant L.*

The CSV2-framework does not specify how the quadratic model functions are constructed. However, it does require that the model functions be selected from a *fully quadratic* class of model functions $\mathcal{M}$:

DEFINITION 2.2. *Let f satisfy Assumption 2.1. Let $\kappa = (\kappa_{ef}, \kappa_{eg}, \kappa_{eh}, \nu_2^m)$ be a given vector of constants, and let $\Delta > 0$. A model function $m \in C^2$* κ-fully quadratic *on $B(x; \Delta)$ if m has a Lipschitz continuous Hessian with corresponding Lipschitz constant bounded by ν_2^m and*

- *the error between the Hessian of the model and the Hessian of the function satisfies*

$$\left\|\nabla^2 f(y) - \nabla^2 m(y)\right\| \leq \kappa_{eh}\Delta \quad \text{for all } y \in B(x; \Delta),$$

- *the error between the gradient of the model and the gradient of the function satisfies*

$$\|\nabla f(y) - \nabla m(y)\| \leq \kappa_{eg}\Delta^2 \quad \text{for all } y \in B(x; \Delta),$$

- *the error between the model and the function satisfies*

$$|f(y) - m(y)| \leq \kappa_{ef}\Delta^3 \quad \text{for all } y \in B(x; \Delta).$$

DEFINITION 2.3. *Let f satisfy Assumption 2.1. A set of model functions $\mathcal{M} = \{m : \mathbb{R}^n \to \mathbb{R}, m \in C^2\}$ is called a* fully quadratic *class of models if there exist positive constants $\kappa = (\kappa_{ef}, \kappa_{eg}, \kappa_{eh}, \nu_2^m)$, such that the following hold:*

1. *for any $x \in S$ and $\Delta \in (0, \Delta_{\max}]$, there exists a model function m in $\mathcal{M}$ which is κ-fully quadratic on $B(x; \Delta)$.*
2. *For this class $\mathcal{M}$, there exists an algorithm, called a "model-improvement" algorithm, that in a finite, uniformly bounded (with respect to x and Δ) number of steps can*

- *either certify that a given model $m \in \mathcal{M}$ is κ-fully quadratic on $B(x;\Delta)$,*
- *or, find a model $\bar{m} \in \mathcal{M}$ that is κ-fully quadratic on $B(x;\Delta)$.*

Note that this definition of a fully quadratic class of models is equivalent to [6, Definition 6.2]; but we have given a separate definition of a κ-fully quadratic model (Definition 2.2) that includes the use of κ to stress the fixed nature of the bounding constants. This change simplifies some analysis by allowing us to discuss κ-fully quadratic models independent of the class of models they belong to. It is important to note that κ does not need to be known explicitly. Instead, it can be defined implicitly by the model improvement algorithm. All that is required is for κ to be fixed (that is, independent of x and Δ). We also note that the set $\mathcal{M}$ may include non-quadratic functions, but when the model functions are quadratic, the Hessian is fixed, so ν_2^m can be chosen to be zero.

2.1. CSV2-framework. The CSV2-framework can now be specified. A critical distinction between this framework and classical trust region methods lies in the optimality criteria. In classical trust region methods, m_k is the 2nd order Taylor approximation of f at x^k; so if x^k is optimal for m_k, it satisfies the first and second order necessary conditions for an optimum of f. In the CSV2-framework, x^k must be optimal for m_k, but m_k must also be an accurate approximation of f near x^k. This requires that the trust region radius is small and that m_k is κ-fully quadratic on the trust region for some fixed κ.

In the algorithm below, g_k^{icb} and H_k^{icb} denote the gradient and Hessian of the incumbent model m_k^{icb}. (We use the superscript icb to stress that incumbent parameters from the previous iterates may be changed before they are used in the trust region step.) The optimality of x^k with respect to m_k is tested by calculating $\varsigma_k^{icb} = \max\left\{\|g_k^{icb}\|, -\lambda_{min}(H_k^{icb})\right\}$. This quantity is zero if and only if the first and second order optimality conditions for m_k are satisfied. The algorithm enters the criticality step when ς_k^{icb} is close to zero. This routine builds a (possibly) new κ-fully quadratic model for the current Δ_k^{icb}, and tests if ς_k^{icb} for this model is sufficiently large. If so, a descent direction has been determined, and the algorithm can proceed. If not, the criticality step reduces Δ_k^{icb} and updates the sample set to improve the accuracy of the model function near x^k. The criticality step ends when ς_k^{icb} is large enough (and the algorithm proceeds) or when both ς_k^{icb} and Δ_k^{icb} are smaller than given threshold values ϵ_c and Δ_{min} (in which case the algorithm has identified a second order stationary point). We refer the reader to [6] for a more detailed discussion of the algorithm, including explanations of the parameters $\eta_0, \eta_1, \gamma, \gamma_{inc}, \beta, \mu$ and ω.

Algorithm CSV2. ([6, Algorithm 10.3])

Step 0 (initialization): Choose a fully quadratic class of models $\mathcal{M}$ and a corresponding model-improvement algorithm, with associated κ defined by Definition 2.3. Choose an initial point x^0 and maximum trust region radius $\Delta_{max} > 0$. We assume that the following are given: an initial model $m_0^{icb}(x^0 + s)$ (with gradient and Hessian at $s = 0$ given by g_0^{icb} and H_0^{icb}, respectively), $\varsigma_0^{icb} = \max\left\{\|g_0^{icb}\|, -\lambda_{min}(H_0^{icb})\right\}$, and a trust region radius $\Delta_0^{icb} \in (0, \Delta_{max}]$.

The constants $\eta_0, \eta_1, \gamma, \gamma_{inc}, \epsilon_c, \beta, \mu, \omega$ are given and satisfy the conditions $0 \leq \eta_0 \leq \eta_1 < 1$ (with $\eta_1 \neq 0$), $0 < \gamma < 1 < \gamma_{inc}, \epsilon_c > 0, \mu > \beta > 0, \omega \in (0,1)$. Set $k = 0$.

Step 1 (criticality step): If $\varsigma_k^{icb} > \epsilon_c$, then $m_k = m_k^{icb}$ and $\Delta_k = \Delta_k^{icb}$.

If $\varsigma_k^{icb} \leq \epsilon_c$, then proceed as follows. Call the model-improvement algorithm to attempt to certify if the model m_k^{icb} is κ-fully quadratic on $B(x^k; \Delta_k^{icb})$. If at least one of the following conditions hold,

- the model m_k^{icb} is not certifiably κ-fully quadratic on $B(x^k; \Delta_k^{icb})$, or

- $\Delta_k^{icb} > \mu\varsigma_k^{icb}$,

then apply Algorithm CriticalityStep (described below) to construct a model $\tilde{m}_k(x^k + s)$ (with gradient and Hessian at $s = 0$ given by $\tilde{g}_k$, and $\tilde{H}_k$, respectively), with $\tilde{\varsigma}_k^m = \max\left\{\|\tilde{g}_k\|, -\lambda_{min}(\tilde{H}_k)\right\}$, which is κ-fully quadratic on the ball $B(x^k; \tilde{\Delta}_k)$ for some $\tilde{\Delta}_k \in (0, \mu\tilde{\varsigma}_k^m]$ given by [6, Algorithm 10.4]. In such a case set

$$m_k = \tilde{m}_k \text{ and } \Delta_k = \min\left\{\max\left\{\tilde{\Delta}_k, \beta\tilde{\varsigma}_k^m\right\}, \Delta_k^{icb}\right\}.$$

Otherwise, set $m_k = m_k^{icb}$ and $\Delta_k = \Delta_k^{icb}$.

Step 2 (step calculation): Compute a step s_k that sufficiently reduces the model m_k (in the sense of [6, (10.13)]) such that $x^k + s_k \in B(x^k; \Delta_k)$.

Step 3 (acceptance of the trial point): Compute $f(x^k + s_k)$ and define

$$\rho_k = \frac{f(x^k) - f(x^k + s_k)}{m_k(x^k) - m_k(x^k + s_k)}$$

If $\rho_k \geq \eta_1$ or if both $\rho_k \geq \eta_0$ and the model is κ-fully quadratic on $B(x^k; \Delta_k)$, then $x^{k+1} = x^k + s_k$ and the model is updated to include the new iterate into the sample set resulting in a new model $m_{k+1}^{icb}(x^{k+1} + s)$ (with gradient and Hessian at $s = 0$ given by g_{k+1}^{icb} and H_{k+1}^{icb}, respectively), with $\varsigma_{k+1}^{icb} = \max\left\{\|g_{k+1}^{icb}\|, -\lambda_{min}(H_{k+1}^{icb})\right\}$; otherwise, the model and the iterate remain unchanged ($m_{k+1}^{icb} = m_k$ and $x^{k+1} = x^k$).

Step 4 (model improvement): If $\rho_k < \eta_1$, use the model-improvement algorithm to

- attempt to certify that m_k is κ-fully quadratic on $B(x^k; \Delta_k)$,
- if such a certificate is not obtained, we say that m_k is not certifiably κ-fully quadratic and make one or more suitable improvement steps.

Define m_{k+1}^{icb} to be the (possibly improved) model.

Step 5 (trust region update): Set

$$\Delta_{k+1}^{icb} \in \begin{cases} \{\min\{\gamma_{inc}\Delta_k, \Delta_{max}\}\} & \text{if } \rho_k \geq \eta_1 \text{ and } \Delta_k < \beta\varsigma_k^m, \\ [\Delta_k, \min\{\gamma_{inc}\Delta_k, \Delta_{max}\}] & \text{if } \rho_k \geq \eta_1 \text{ and } \Delta_k \geq \beta\varsigma_k^m, \\ \{\gamma\Delta_k\} & \text{if } \rho_k < \eta_1 \text{ and } m_k \text{ is } \kappa\text{-fully quadratic}, \\ \{\Delta_k\} & \text{if } \rho_k < \eta_1 \text{ and } m_k \text{ is not certifiably} \\ & \quad \kappa\text{-fully quadratic}. \end{cases}$$

Increment k by 1 and go to Step 1.

Algorithm CriticalityStep. ([6, Algorithm 10.4]) This algorithm is applied only if $\varsigma_k^{icb} \leq \epsilon_c$ and at least one of the following holds: the model m_k^{icb} is not certifiably κ-fully quadratic on $B(x^k; \Delta_k^{icb})$ or $\Delta_k^{icb} > \mu\varsigma_k^{icb}$.

Initialization: Set $i = 0$. Set $m_k^{(0)} = m_k^{icb}$.

Repeat Increment i by one. Use the model improvement algorithm to improve the previous model $m_k^{(i-1)}$ until it is κ-fully quadratic on $B(x^k; \omega^{i-1}\Delta_k^{icb})$. Denote the new model by $m_k^{(i)}$. Set $\tilde{\Delta}_k = \omega^{i-1}\Delta_k^{icb}$ and $\tilde{m}_k = m_k^{(i)}$.

Until $\tilde{\Delta}_k \leq \mu(\varsigma_k^m)^{(i)}$.

2.1.1. Global convergence. Define the set $L(x^0) = \left\{x \in \mathbb{R}^n : f(x) \leq f(x^0)\right\}$.

Assumption 2.4. *Assume that f is bounded from below on $L(x^0)$.*

Assumption 2.5. *There exists a constant $\kappa_{bhm} > 0$ such that, for all x^k generated by the algorithm,*

$$\|H_k\| \leq \kappa_{bhm}.$$

THEOREM 2.6. *([6, Theorem 10.23]) Let Assumptions 2.1, 2.4 and 2.5 hold with* $S = L(x^0)$. *Then*

$$\lim_{k\to+\infty} \max\{\|\nabla f(x^k)\|, -\lambda_{\min}(\nabla^2 f(x^k))\} = 0.$$

It follows from this theorem that any accumulation point of $\{x^k\}$ satisfies the first and second order necessary conditions for a minimum of f.

To specify an algorithm within this framework, three things are required:

1. Define the class of model functions $\mathcal{M}$. This is determined by the method for constructing models from the sample set. In [6], models were constructed using interpolation, least squares regression, and minimum Frobenius norm methods. We describe the general form of our weighted regression models in §3 and propose a specific weighting scheme in §6.
2. Define a model-improvement algorithm. §5 describes our model improvement algorithm which tests the geometry of the sample set, and if necessary, adds and/or deletes points to ensure that the model function constructed from the sample set satisfies the error bounds in Definition 2.2 (i.e. it is κ-fully quadratic).
3. Demonstrate that the model-improvement algorithm satisfies the requirements for the definition of a class of fully quadratic models. For our algorithm, this is discussed in §5.

3. Model Construction. This section describes how we construct the model function m_k at the kth iteration. For simplicity, we drop the subscript k for the remainder of this section. Let $\bar{f} = (\bar{f}_0, \ldots, \bar{f}_p)^T$ where $\bar{f}_i$ denotes the computed function value at y^i, and let E_i denote the associated computational error. That is

$$\bar{f}_i = f(y^i) + E_i. \tag{3.1}$$

Let $w = (w_0, \ldots, w_p)^T$ be a vector of positive weights. A quadratic polynomial m is said to be a *weighted least squares approximation* of f (with respect to w) if it minimizes

$$\sum_{i=0}^{p} w_i^2 \left(m(y^i) - \bar{f}_i\right)^2 = \left\|W\left(m(Y) - \bar{f}\right)\right\|^2.$$

where $m(Y)$ denotes the vector $(m(y^0), m(y^1), \ldots, m(y^p))^T$ and $W = \mathrm{diag}(w)$. In this case, we write

$$Wm(Y) \overset{\ell.s.}{=} W\bar{f}. \tag{3.2}$$

Let $\phi = \{\phi_0, \phi_1, \ldots, \phi_q\}$ be a basis for the quadratic polynomials in $\mathbb{R}^n$. For example, ϕ might be the monomial basis

$$\bar{\phi} = \{1, x_1, x_2, \ldots, x_n, x_1^2/2, x_1x_2, \ldots, x_{n-1}x_n, x_n^2/2\}. \tag{3.3}$$

Define

$$M(\phi, Y) = \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \cdots & \phi_q(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \cdots & \phi_q(y^1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \cdots & \phi_q(y^p) \end{bmatrix}.$$

Since ϕ is a basis for the quadratic polynomials, the model function can be written $m(x) = \sum_{i=0}^{q} \alpha_i \phi_i(x)$. The coefficients $\alpha = (\alpha_0, \ldots, \alpha_q)^T$ solve the weighted least squares regression problem

$$WM(\phi, Y)\alpha \overset{\ell.s.}{=} W\bar{f}. \tag{3.4}$$

If $M(\phi, Y)$ has full column rank, the sample set Y is said to be *poised for quadratic regression*. The following lemma is a straightforward generalization of [6, Lemma 4.3]:

LEMMA 3.1. *If Y is poised for quadratic regression, then the weighted least squares regression polynomial (with respect to positive weights $w = (w_0, \ldots, w_p)$) exists, is unique and is given by $m(x) = \phi(x)^T\alpha$, where*

$$\alpha = (WM)^+ W\bar{f} = (M^T W^2 M)^{-1} M^T W^2 \bar{f}, \tag{3.5}$$

where $W = \mathrm{diag}(w)$ and $M = M(\phi, Y)$.

Proof. Since W and M both have full column rank, so does WM. Thus, the least squares problem (3.4) has a unique solution given by $(WM)^+ W\bar{f}$. Moreover, since WM has full column rank, $(WM)^+ = \left((WM)^T(WM)\right)^{-1} M^T W$. ∎

4. Error analysis and the geometry of the sample set. Throughout this section, $Y = \left\{y^0, \cdots, y^p\right\}$ denotes the sample set, $p_1 = p + 1$, $w \in \mathbb{R}^{p_1}_+$ is a vector of positive weights, $W = \mathrm{diag}(w)$, and $M = M(\phi, Y)$. $\bar{f}$ denotes the vector of computed function values at the points in Y as defined by (3.1).

The accuracy of the model function m_k relies critically on the geometry of the sample set. In this section, we generalize the theory of poisedness from [6] to the weighted regression framework. This section also includes error analysis which extends results from [6] to weighted regression, as well as considering the impact of computational error on the error bounds. We start by defining *weighted regression Lagrange polynomials*.

4.1. Weighted regression Lagrange polynomials.

DEFINITION 4.1. *A set of polynomials $\ell_j(x), j = 0, \ldots, p$ in $\mathcal{P}_n^d$ are called* weighted regression Lagrange polynomials *with respect to the weights w and sample set Y if for each j,*

$$W\ell_j^Y \overset{\ell.s.}{=} We^j,$$

where $\ell_j^Y = \left[\ell_j(y^0), \cdots, \ell_j(y^p)\right]^T$.

The following lemma is a direct application of Lemma 3.1.

LEMMA 4.2. *Let $\phi(x) = (\phi_0(x), \ldots, \phi_q(x))^T$. If Y is poised, then the set of weighted regression Lagrange polynomials exists and is unique, and is given by $\ell_j(x) = \phi(x)^T a^j$, $j = 0, \cdots, p$, where a^j is the j^{th} column of the matrix*

$$A = (WM)^+ W. \tag{4.1}$$

Proof. Note that $m(\cdot) = \ell_j(\cdot)$ satisfies (3.2) with $\bar{f} = e^j$. By Lemma 3.1, $\ell_j(x) = \phi(x)^T a^j$ where $a^j = (WM)^+ We^j$ which is the jth column of $(WM)^+ W$. ∎

The following lemma is based on [6, Lemma 4.6].

LEMMA 4.3. *If Y is poised, then the model function defined by* (3.2) *satisfies*

$$m(x) = \sum_{i=0}^{p} \bar{f}_i \ell_i(x),$$

where $\ell_j(x),\ j = 0, \cdots, p$ denote the weighted regression Lagrange polynomials corresponding to Y and W.

Proof. By Lemma 3.1 $m(x) = \phi(x)^T \alpha$ where

$$\alpha = (WM)^+ W\bar{f} = A\bar{f}$$

for A defined by (4.1). Let $\ell(x) = [\ell_0(x), \cdots, \ell_p(x)]^T$. Then

$$m(x) = \phi^T(x)A\bar{f} = \bar{f}^T\ell(x) = \sum_{i=0}^{p} \bar{f}_i \ell_i(x).$$

□

4.2. Error Analysis. For the remainder of this paper, let $\hat{Y} = \left\{\hat{y}^0, \cdots, \hat{y}^p\right\}$ denote the shifted and scaled sample set where $\hat{y}^i = (y^i - y^0)/R$ and $R = \max_i \left\|y^i - y^0\right\|$. Note that $\hat{y}^0 = 0$ and $\max_i \left\|\hat{y}^i\right\| = 1$. Any analysis of Y can be directly related to $\hat{Y}$ by the following lemma:

LEMMA 4.4. *Define the basis $\hat{\phi} = \left\{\hat{\phi}_0(x), \cdots, \hat{\phi}_q(x)\right\}$, where $\hat{\phi}_i(x) = \bar{\phi}_i(Rx + y^0)$, $i = 0, \ldots, q$ and $\bar{\phi}$ is the monomial basis. Let $\{\ell_0(x), \cdots, \ell_p(x)\}$ be weighted regression Lagrange polynomials for Y and $\left\{\hat{\ell}_0(x), \cdots, \hat{\ell}_p(x)\right\}$ be weighted regression Lagrange polynomials for $\hat{Y}$. Then $M(\bar{\phi}, Y) = M(\hat{\phi}, \hat{Y})$. If Y is poised, then*

$$\ell(x) = \hat{\ell}\left(\frac{x - y^0}{R}\right).$$

Proof. Observe that

$$M(\bar{\phi}, Y) = \begin{bmatrix} \bar{\phi}_0(y^0) & \cdots & \bar{\phi}_q(y^0) \\ \bar{\phi}_0(y^1) & \cdots & \bar{\phi}_q(y^1) \\ \vdots & \vdots & \vdots \\ \bar{\phi}_0(y^p) & \cdots & \bar{\phi}_q(y^p) \end{bmatrix} = \begin{bmatrix} \hat{\phi}_0(\hat{y}^0) & \cdots & \hat{\phi}_q(\hat{y}^0) \\ \hat{\phi}_0(\hat{y}^1) & \cdots & \hat{\phi}_q(\hat{y}^1) \\ \vdots & \vdots & \vdots \\ \hat{\phi}_0(\hat{y}^p) & \cdots & \hat{\phi}_q(\hat{y}^p) \end{bmatrix} = M(\hat{\phi}, \hat{Y}).$$

By the definition of poisedness, Y is poised if and only if $\hat{Y}$ is poised. Let $\bar{\phi}(x) = \left(\bar{\phi}_0(x), \ldots, \bar{\phi}_q(x)\right)^T$ and $\hat{\phi}(x) = \left(\hat{\phi}_0(x), \ldots, \hat{\phi}_q(x)\right)^T$. Then

$$\hat{\phi}(\hat{x}) = \begin{bmatrix} \hat{\phi}_0(\frac{x-y^0}{R}) \\ \vdots \\ \hat{\phi}_q(\frac{x-y^0}{R}) \end{bmatrix} = \begin{bmatrix} \bar{\phi}_0(x) \\ \vdots \\ \bar{\phi}_q(x) \end{bmatrix} = \bar{\phi}(x).$$

By Lemma 4.2, if Y is poised, then

$$\ell(x) = \bar{\phi}(x)^T(WM(\bar{\phi}, Y))^+ W = \hat{\phi}(\hat{x})^T(WM(\hat{\phi}, \hat{Y}))^+ W = \hat{\ell}(\hat{x}).$$

□

Let $\bar{f}_i$ be defined by (3.1) and let Ω be an open convex set containing Y. If f is C^2 on Ω, then by Taylor's theorem, for each sample point $y^i \in Y$, and a fixed $x \in \text{conv}(Y)$, there exists a point $\eta_i(x)$ on the line segment connecting x to y^i such that

$$\begin{aligned}\bar{f}_i &= f(y^i) + E_i \\ &= f(x) + \nabla f(x)^T(y^i - x) + \frac{1}{2}(y^i - x)^T\nabla^2 f(\eta_i(x))(y^i - x) + E_i \\ &= f(x) + \nabla f(x)^T(y^i - x) + \frac{1}{2}(y^i - x)^T\nabla^2 f(x)(y^i - x) \\ &\qquad + \frac{1}{2}(y^i - x)^T H_i(x)(y^i - x) + E_i,\end{aligned} \tag{4.2}$$

where $H_i(x) = \nabla^2 f(\eta_i(x)) - \nabla^2 f(x)$.

Let $\{\ell_i(x)\}$ denote the weighted-regression Lagrange polynomials associated with Y. The following lemma and proof are inspired by [2, Theorem 1]:

LEMMA 4.5. *Let f be twice continuously differentiable on Ω and let $m(x)$ denote the quadratic function determined by weighted regression. Then, for any $x \in \Omega$ the following identities hold:*

- $m(x) = f(x) + \frac{1}{2}\sum_{i=0}^p (y_i - x)^T H_i(x)(y_i - x)\ell_i(x) + \sum_{i=0}^p E_i\ell_i(x)$,
- $\nabla m(x) = \nabla f(x) + \frac{1}{2}\sum_{i=0}^p (y_i - x)^T H_i(x)(y_i - x)\nabla\ell_i(x) + \sum_{i=0}^p E_i\nabla\ell_i(x)$,
- $\nabla^2 m(x) = \nabla^2 f(x) + \frac{1}{2}\sum_{i=0}^p (y_i - x)^T H_i(x)(y_i - x)\nabla^2\ell_i(x) + \sum_{i=0}^p E_i\nabla^2\ell_i(x)$,

where $H_i(x) = \nabla^2 f(\eta_i(x)) - \nabla^2 f(x)$ for some point $\eta_i(x) = \theta x + (1-\theta)y^i, 0 \le \theta \le 1$ on the line segment connecting x to y^i.

Proof. Let D denote the differential operator as defined in [2]. In particular, $D^0 f(x) = f(x)$, $D^1 f(x)z = \nabla f(x)^T z$, and $D^2 f(x)z^2 = z^T\nabla^2 f(x)z$. By Lemma 4.3, $m(x) = \sum_{i=0}^p \bar{f}_i\ell_i(x)$; so for $h = 0, 1$ or 2,

$$D^h m(x) = \sum_{i=0}^p \bar{f}_i D^h\ell_i(x).$$

Substituting (4.2) for $\bar{f}_i$ in the above equation yields

$$\begin{aligned}D^h m(x) \quad &= \sum_{j=0}^2 \frac{1}{j!}\sum_{i=0}^p D^j f(x)(y^i - x)^j D^h\ell_i(x) \\ &\quad + \frac{1}{2}\sum_{i=0}^p (y^i - x)^T H_i(x)(y^i - x)D^h\ell_i(x) + \sum_{i=0}^p E_i D^h\ell_i(x)\end{aligned} \tag{4.3}$$

where $H_i(x) = \nabla^2 f(\eta_i(x)) - \nabla^2 f(x)$ for some point $\eta_i(x)$ on the line segment connecting x to y^i. Consider the first term on the right hand side above. We shall show that

$$\frac{1}{j!}\sum_{i=0}^p D^j f(x)(y^i - x)^j D^h\ell_i(x) = \begin{cases} D^h f(x) & \text{for } j = h \\ 0 & \text{for } j \ne h. \end{cases} \tag{4.4}$$

Let $B_j = D^j f(x)$, and let $g_j : \mathbb{R}^n \to \mathbb{R}$ be the polynomial defined by $g_j(z) = \frac{1}{j!}B_j(z - x)^j$. Observe that $D^j g_j(x) = B_j$ and $D^h g_j(x) = 0$ for $h \ne j$. Since g_j

has degree $j \leq 2$, the weighted least squares approximation of g_j by a quadratic polynomial is g_j itself. Thus, by Lemma 4.3 and the definition of g_j,

$$g_j(z) = \sum_{i=0}^{p} g_j(y^i)\ell_i(z) = \frac{1}{j!}\sum_{i=0}^{p} B_j(y^i - x)^j \ell_i(z). \tag{4.5}$$

Applying the differential operator D^h with respect to z yields

$$\begin{aligned} D^h g_j(z) &= \frac{1}{j!}\sum_{i=0}^{p} B_j(y^i - x)^j D^h \ell_i(z) \\ &= \frac{1}{j!}\sum_{i=0}^{p} D^j f(x)(y^i - x)^j D^h \ell_i(z) \end{aligned}$$

Letting $z = x$, the expression on the right is identical to the left side of (4.4). This proves (4.4) since $D^h g_j(x) = 0$ for $j \neq h$ and $D^j g_j(x) = B_j$ for $j = h$. By (4.4), (4.3) reduces to

$$D^h m(x) = D^h f(x) + \frac{1}{2}\sum_{i=0}^{p} H_i(x)(y^i - x)^2 D^h \ell_i(x) + \sum_{i=0}^{p} E_i D^h \ell_i(x).$$

Applying this with $h = 0, 1, 2$ proves the lemma. □

Since $\|H_i(x)\| \leq L\left\|y^i - x\right\|$ by the Lipschitz continuity of $\nabla^2 f$, the following is a direct consequence of Lemma 4.5.

COROLLARY 4.6. *Let f satisfy Assumption 2.1 for some convex set Ω, and let $m(x)$ denote the quadratic function determined by weighted regression. Then, for any $x \in \Omega$ the following error bounds hold:*

- $|f(x) - m(x)| \leq \sum_{i=0}^{p}\left(\frac{L}{2}\left\|y^i - x\right\|^3 + |E_i|\right)|\ell_i(x)|$
- $\|\nabla f(x) - \nabla m(x)\| \leq \sum_{i=0}^{p}\left(\frac{L}{2}\left\|y^i - x\right\|^3 + |E_i|\right)\|\nabla \ell_i(x)\|$
- $\left\|\nabla^2 f(x) - \nabla^2 m(x)\right\| \leq \sum_{i=0}^{p}\left(\frac{L}{2}\left\|y^i - x\right\|^3 + |E_i|\right)\left\|\nabla^2 \ell_i(x)\right\|$.

Using this corollary, the following result provides error bounds between the function and the model in terms of the sample set radius.

COROLLARY 4.7. *Let Y be poised, and let $R = \max_i \left\|y^i - y^0\right\|$. Suppose $|E_i| \leq \epsilon$ for $i = 0, \ldots, p$. If f satisfies Assumption 2.1 with Lipschitz constant L, then there exist constants Λ_1, Λ_2, and Λ_3, independent of R, such that for all $x \in B(y^0; R)$,*

- $|f(x) - m(x)| \leq \Lambda_1\sqrt{p_1}\left(4LR^3 + \epsilon\right)$.
- $\|\nabla f(x) - \nabla m(x)\| \leq \Lambda_2\sqrt{p_1}\left(4LR^2 + \epsilon/R\right)$.
- $\left\|\nabla^2 f(x) - \nabla^2 m(x)\right\| \leq \Lambda_3\sqrt{p_1}\left(4LR + \epsilon/R^2\right)$.

Proof. Let $\{\hat{\ell}_0(x), \ldots, \hat{\ell}_p(x)\}$ be the Lagrange polynomials generated by the shifted and scaled set $\hat{Y}$, and let $\{\ell_0(x), \ldots, \ell_p(x)\}$ be the Lagrange polynomials generated by the set Y. By Lemma 4.4, for each $x \in B(y^0; R)$, $\ell_i(x) = \hat{\ell}_i(\hat{x})\ \forall\ i$, where $\hat{x} = (x - y^0)/R$. Thus, $\nabla \ell_i(x) = \nabla \hat{\ell}_i(\hat{x})/R$, and $\nabla^2 \ell_i(x) = \nabla^2 \hat{\ell}_i(\hat{x})/R^2$. Let $\hat{\ell}(x) = \left[\hat{\ell}_0(x), \ldots, \hat{\ell}_p(x)\right]^T$, $\hat{g}(x) = \left[\left\|\nabla \hat{\ell}_0(x)\right\|, \ldots, \left\|\nabla \hat{\ell}_p(x)\right\|\right]^T$ and $\hat{h}(x) = \left[\left\|\nabla^2 \hat{\ell}_0(x)\right\|, \ldots, \left\|\nabla^2 \hat{\ell}_p(x)\right\|\right]^T$.

By Corollary 4.6,

$$\begin{aligned}
|f(x) - m(x)| &\leq \sum_{i=0}^{p} \left(\frac{L}{2} \left\|y^i - x\right\|^3 + |E_i| \right) |\ell_i(x)| \\
&\leq \sum_{i=0}^{p} \left(4LR^3 + \epsilon\right) |\ell_i(x)| \ \text{(since } \|y^i - x\| \leq 2R, \text{ and } |E_i| \leq \epsilon) \\
&= \left(4LR^3 + \epsilon\right) \|\hat{\ell}_i(\hat{x})\|_1 \\
&\leq \sqrt{p_1} \left(4LR^3 + \epsilon\right) \left\|\hat{\ell}(\hat{x})\right\|, \ \text{(since for } x \in \mathbb{R}^n, \ \|x\|_1 \leq \sqrt{n}\|x\|_2).
\end{aligned}$$

Similarly, $\|\nabla f(x) - \nabla m(x)\| \leq \sqrt{p_1} \left(4LR^2 + \frac{\epsilon}{R}\right) \|\hat{g}(\hat{x})\|$

and $\left\|\nabla^2 f(x) - \nabla^2 m(x)\right\| \leq \sqrt{p_1} \left(4LR + \frac{\epsilon}{R^2}\right) \left\|\hat{h}(\hat{x})\right\|$.

Setting $\Lambda_1 = \max_{x \in B(0;1)} \left\|\hat{\ell}(x)\right\|$, $\Lambda_2 = \max_{x \in B(0;1)} \|\hat{g}(x)\|$, and $\Lambda_3 = \max_{x \in B(0;1)} \left\|\hat{h}(x)\right\|$ yields the desired result. □

Note the similarity between these error bounds and those in the definition of κ-fully quadratic models. If there is no computational error (or if the error is $\mathcal{O}(\Delta^3)$), κ-fully quadratic models (for some fixed κ) can be obtained by controlling the geometry of the sample set so that $\Lambda_i \sqrt{p_1}$, $i = 1, 2, 3$ are bounded by fixed constants and by controlling the trust region radius Δ so that $\frac{\Delta}{R}$ is bounded. This motivates the definitions of Λ-poised and strongly Λ-poised in the weighted regression sense in the next section.

4.3. Λ-poisedness (in the weighted regression sense). In this section, we restrict our attention to the monomial basis $\bar{\phi}$ defined in (3.3). In order to produce accurate model functions, the points in the sample set need to be distributed in such a way that the matrix $M = M(\bar{\phi}, Y)$ is sufficiently well-conditioned. This is the motivation behind the following definitions of Λ-poised and strongly Λ-poised sets. These definitions are identical to [6, Definitions 4.7, 4.10] except that the Lagrange polynomials in the definitions are *weighted regression* Lagrange polynomials.

DEFINITION 4.8. *Let $\Lambda > 0$ and let B be a set in $\mathbb{R}^n$. Let $w = (w_0, \ldots, w_p)$ be a vector of positive weights, $Y = \{y^0, \ldots, y^p\}$ be a poised set, and let $\{\ell_0, \ldots, \ell_p\}$ be the associated weighted regression Lagrange polynomials. Let $\ell(x) = (\ell_0(x), \cdots, \ell_p(x))^T$.*

- *Y is said to be Λ-poised in B (in the weighted regression sense) if and only if*

$$\Lambda \geq \max_{x \in B} \max_{0 \leq i \leq p} |\ell_i(x)| .$$

- *Y is said to be strongly Λ-poised in B (in the weighted regression sense) if and only if*

$$\frac{q_1}{\sqrt{p_1}} \Lambda \geq \max_{x \in B} \|\ell(x)\| .$$

Note that if the weights are all equal, the above definitions are equivalent to those for Λ-poised and strongly Λ-poised given in [6].

We are naturally interested in using these weighted regression Lagrange polynomials to form models that are guaranteed to sufficiently approximate f. Let Y^k, Δ_k, and R_k denote the sample set, trust region radius, and sample set radius at iteration

k as defined at the beginning of §4.2. Assume that $\frac{\Delta_k}{R_k}$ is bounded. If the number of sample points is bounded, it can be shown, using Corollary 4.7, that if Y^k is Λ-poised for all k, then the corresponding model functions are κ-fully quadratic, (assuming no computational error, or that the computational error is $\mathcal{O}(\Delta^3)$). When the number of sample points is not bounded, Λ-poisedness is not enough. In the following, we show that if Y^k is *strongly* Λ-poised for all k, then the corresponding models are κ-fully quadratic, regardless of the number of points in Y^k.

LEMMA 4.9. *Let $\hat{M} = M(\bar{\phi}, \hat{Y})$. If $\left\|W(\hat{M}^T W)^+\right\| \leq \sqrt{q_1/p_1}\Lambda$, then $\hat{Y}$ is strongly Λ-poised in $B(0;1)$ in the weighted regression sense, with respect to the weights w. Conversely, if $\hat{Y}$ is strongly Λ-poised in $B(0;1)$ in the weighted regression sense, then*

$$\left\|W(\hat{M}^T W)^+\right\| \leq \frac{\theta q_1}{\sqrt{p_1}}\Lambda,$$

where $\theta > 0$ is a fixed constant dependent only on n (but independent of Y and Λ).

Proof. Let $A = (W\hat{M})^+ W$ and $\ell(x) = (\ell_0(x), \ldots, \ell_p(x))^T$. By Lemma 4.2, $\ell(x) = A^T\phi(x)$. It follows that for any $x \in B(0;1)$,

$$\|\ell(x)\| = \left\|A^T\bar{\phi}(x)\right\| \leq \|A\| \left\|\bar{\phi}(x)\right\| \leq \left(\sqrt{q_1/p_1}\Lambda\right)\left(\sqrt{q_1}\left\|\bar{\phi}(x)\right\|_\infty\right) \leq \left(q_1/\sqrt{p_1}\right)\Lambda.$$

(For the last inequality, we used the fact that $\max_{x\in B(0;1)} \left\|\bar{\phi}(x)\right\|_\infty \leq 1$).

To prove the converse, let $U\Sigma V^T = A^T$ be the reduced singular value decomposition of A^T. Note that U and V are $p_1 \times q_1$ and $q_1 \times q_1$ matrices, respectively, with orthonormal columns; Σ is a $q_1 \times q_1$ diagonal matrix, whose diagonal entries are the singular values of A^T. Let σ_1 be the largest singular value with v^1 the corresponding column of V. As shown in the proof of [4, Theorem 2.9], there exists a constant $\gamma > 0$ such that for any unit vector v, there exists an $\bar{x} \in B(0;1)$ such that $\left|v^T\bar{\phi}(\bar{x})\right| \geq \gamma$. Therefore, since $\left\|v^1\right\| = 1$, there is an $\bar{x} \in B(0;1)$ such that $\left|(v^1)^T\bar{\phi}(\bar{x})\right| \geq \gamma$. Let $v^\perp$ be the orthogonal projection of $\bar{\phi}(\bar{x})$ onto the subspace orthogonal to v^1; so $\bar{\phi}(\bar{x}) = \left((v^1)^T\bar{\phi}(\bar{x})\right)v^1 + v^\perp$. Note that $\Sigma V^T v^1$ and $\Sigma V^T v^\perp$ are orthogonal vectors. Note also that for any vector z, $\left\|U\Sigma V^T z\right\| = \left\|\Sigma V^T z\right\|$ (since U has orthonormal columns). It follows that

$$\begin{aligned}\|\ell(\bar{x})\| = \left\|A^T\phi(\bar{x})\right\| = \left\|\Sigma V^T\bar{\phi}(\bar{x})\right\| &= \left\|\Sigma V^T v^\perp\right\| + \left\|\Sigma V^T\left((v^1)^T\bar{\phi}(\bar{x})\right)\right\| \\ &\geq \left|(v^1)^T\bar{\phi}\right| \left\|\Sigma V^T v^1\right\| \geq \gamma\left\|\Sigma V^T v^1\right\| = \gamma\left\|\Sigma e^1\right\| = \gamma\|A\|.\end{aligned}$$

Thus, $\|A\| \leq \max_{x\in B(0;1)} \|\ell(x)\|/\gamma \leq \frac{q_1}{\gamma\sqrt{p_1}}\Lambda$, which proves the result with $\theta = 1/\gamma$. □

We can now prove that models generated by weighted regression Lagrange polynomials are κ-fully quadratic.

PROPOSITION 4.10. *Let f satisfy Assumption 2.1 and let $\Lambda > 0$ be fixed. There exists a vector $\kappa = (\kappa_{ef}, \kappa_{eg}, \kappa_{eh}, 0)$ such that for any $y^0 \in S$ and $\Delta \leq \Delta_{max}$, if*

1. *$Y = \{y^0, \ldots, y^p\} \subset B(y^0; \Delta)$ is strongly Λ-poised in $B(y^0; \Delta)$ in the weighted regression sense with respect to positive weights $w = \{w_0, \ldots, w_p\}$, and*
2. *the computational error $|E_i|$ is bounded by $\mathcal{C}\Delta^3$, where $\mathcal{C}$ is a fixed constant,*

then the corresponding model function m is κ-fully quadratic.

Proof. Let $\hat{x}$, $\hat{\ell}(\cdot), \hat{g}(\cdot)$, $\hat{h}(\cdot)$, Λ_1, Λ_2, and Λ_3 be as defined in the proof of Corollary 4.7. Let $\hat{M} = M(\bar{\phi}, \hat{Y})$ and $W = \mathrm{diag}(w^k)$. By Lemma 4.2, $\hat{\ell}(x) = A^T\bar{\phi}(x)$,

where $A = (W\hat{M})^+ W$. By Lemma 4.9, $\|A\| \leq \dfrac{\theta q_1}{\sqrt{p_1}}\Lambda$, where θ is a fixed constant. It follows that

$$\Lambda_1 = \max_{x \in B(0;1)} \left\|\hat{\ell}(x)\right\| \leq \max_{x \in B(0;1)} \|A\| \left\|\bar{\phi}(x)\right\| \leq c_1 \frac{\theta q_1}{\sqrt{p_1}}\Lambda,$$

where $c_1 = \max\limits_{x \in B(0;1)} \left\|\bar{\phi}(x)\right\|$ is a constant independent of Y. Similarly,

$$\begin{aligned}
\Lambda_2 &= \max_{x \in B(0;1)} \|\hat{g}(x)\| = \max_{x \in B(0;1)} \left\|\left(\|\nabla \hat{\ell}_0(x)\|, \cdots, \|\nabla \hat{\ell}_p(x)\|\right)\right\| \\
&= \max_{x \in B(0;1)} \left\|A^T \nabla \bar{\phi}(x)\right\|_F \leq \sqrt{q_1} \max_{x \in B(0;1)} \left\|A^T \nabla \bar{\phi}(x)\right\| \\
&\leq \sqrt{q_1} \max_{x \in B(0;1)} \|A\| \left\|\nabla \bar{\phi}(x)\right\| \leq c_2 \frac{\theta q_1^{\frac{3}{2}}}{\sqrt{p_1}}\Lambda,
\end{aligned}$$

where $c_2 = \max\limits_{x \in B(0;1)} \left\|\nabla \bar{\phi}(x)\right\|$ is independent of Y.

To bound Λ_3, let $J_{s,t}$ denote the unique index j such that x_s and x_t both appear in the quadratic monomial $\bar{\phi}_j(x)$. For example $J_{1,1} = n+2$, $J_{1,2} = J_{2,1} = n+3$, etc. Observe that

$$\left[\nabla^2 \bar{\phi}_j(x)\right]_{s,t} = \begin{cases} 1 & \text{if } j = J_{s,t}, \\ 0 & \text{otherwise.} \end{cases}$$

It follows that

$$\nabla^2 \hat{\ell}_i(x) = \sum_{j=0}^{q} A_{i,j}^T \nabla^2 \bar{\phi}_j(x) = \begin{bmatrix} A_{i,J_{1,1}}^T & A_{i,J_{1,2}}^T & \cdots & A_{i,J_{1,n}}^T \\ A_{i,J_{2,1}}^T & A_{i,J_{2,2}}^T & \cdots & A_{i,J_{2,n}}^T \\ & \vdots & \ddots & \\ A_{i,J_{n,1}}^T & A_{i,J_{n,2}}^T & \cdots & A_{i,J_{n,n}}^T \end{bmatrix}.$$

We conclude that $\left\|\nabla^2 \hat{\ell}_i(x)\right\| \leq \left\|\nabla^2 \hat{\ell}_i(x)\right\|_F \leq \sqrt{2}\left\|A_{i,\cdot}^T\right\|$. Thus,

$$\begin{aligned}
\Lambda_3 &= \max_{x \in B(0;1)} \left\|\hat{h}(x)\right\| = \max_{x \in B(0;1)} \left\|\left(\|\nabla^2 \hat{\ell}_0(x)\|, \cdots, \|\nabla^2 \hat{\ell}_p(x)\|\right)\right\| \\
&\leq \sqrt{2\sum_{i=0}^{p} \left\|A_{i,\cdot}^T\right\|^2} = \sqrt{2}\|A\|_F \leq \sqrt{2q_1}\|A\| \leq \frac{\sqrt{2}\theta q_1^{\frac{3}{2}}}{\sqrt{p_1}}\Lambda.
\end{aligned}$$

By assumption, the computational error $|E_i|$ is bounded by $\epsilon = C\Delta^3$. So, by Corollary 4.7, for all $x \in B(y^0; \Delta)$,

- $|f(x) - m(x)| \leq \sqrt{p_1}\,(4L + C)\,\Delta^3 \Lambda_1 \leq c_1 \theta q_1 \Lambda\,(4L + C)\,\Delta^3 = \kappa_{ef}\Delta^3$.
- $\|\nabla f(x) - \nabla m(x)\| \leq \sqrt{p_1}\,(4L + C)\,\Delta^2 \Lambda_2 \leq c_2 \theta q_1^{\frac{3}{2}} \Lambda\,(4L + C)\,\Delta^2 = \kappa_{eg}\Delta^2$.
- $\left\|\nabla^2 f(x) - \nabla^2 m(x)\right\| \leq \sqrt{p_1}\,(4L + C)\,\Delta \Lambda_3 \leq \sqrt{2}\theta q_1^{\frac{3}{2}} \Lambda\,(4L + C)\,\Delta = \kappa_{eh}\Delta$.

where $\kappa_{ef} = c_1 \theta q_1 \Lambda\,(4L + C)$, $\kappa_{eg} = c_2 \theta q_1^{\frac{3}{2}} \Lambda\,(4L + C)$, $\kappa_{eh} = \sqrt{2}\theta q_1^{\frac{3}{2}} \Lambda\,(4L + C)$. Thus, $m(x)$ is $(\kappa_{ef}, \kappa_{eg}, \kappa_{eh}, 0)$-fully quadratic, and since these constants are independent of y^0 and Δ, the result is proven. □

The final step in establishing that we have a fully quadratic class of models is to define an algorithm that produces a strongly Λ-poised sample set in a finite number of steps.

PROPOSITION 4.11. *Let $\hat{Y} = \{y^0, \ldots, y^p\} \subset \mathbb{R}^n$ be a set of points in the unit ball $B(0;1)$ such that $\|y^j\| = 1$ for at least one j. Let $w = (w_0, \ldots, w_p)^T$ be a vector of positive weights. If $\hat{Y}$ is strongly Λ-poised in $B(0;1)$ in the sense of* unweighted *regression, then there exists a constant $\bar{\theta} > 0$, independent of $\hat{Y}$, Λ and w, such that $\hat{Y}$ is strongly $\big(\mathrm{cond}(W)\bar{\theta}\Lambda\big)$-poised in the* weighted *regression sense.*

Proof. Let $\hat{M} = M(\bar{\phi}, \hat{Y})$, where $\bar{\phi}$ is the monomial basis. By Lemma 4.9 (applied with unit weights), $\mathrm{cond}\,\hat{M}^+ \leq \theta q_1 \Lambda/\sqrt{p_1}$, where θ is a constant independent of $\hat{Y}$ and Λ. Thus,

$$\left\|W(\hat{M}^T W)^+\right\| \leq \mathrm{cond}(W)\left\|\hat{M}^+\right\| \leq \frac{\mathrm{cond}(W)\theta q_1}{\sqrt{p_1}}\Lambda.$$

The result follows with $\bar{\theta} = \theta\sqrt{q_1}$.
□

The significance of this proposition is that any model improvement algorithm for unweighted regression can be used for weighted regression to ensure the same global convergence properties, provided $\mathrm{cond}(W)$ is bounded. For the model improvement algorithm described in the following section, this requirement is satisfied by bounding the weights away from zero while keeping the largest weight equal to 1.

In practice, we need not ensure Λ-poisedness of Y^k at every iterate to guarantee the algorithm convergences to a second-order minimum. Rather, Λ-poisedness only needs to be enforced as the algorithm stagnates.

5. Model Improvement Algorithm. This section describes a model improvement algorithm (MIA) for regression which, by the preceding section, can also be used for weighted regression to ensure that the sample sets are strongly Λ-poised for some fixed Λ (which is not necessarily known). The algorithm is based on the following observation, which is a straightforward extension of [6, Theorem 4.11].

The MIA presented in [6] makes assumptions (such as all points must lie within $B(y^0;\Delta)$) to simplify the theory. We resist such assumptions to account for practical concerns (points which lie outside of $B(y^0,\Delta)$) that arise in the algorithm.

PROPOSITION 5.1. *If the shifted and scaled sample set $\hat{Y}$ of p_1 points contains $l = \lfloor \frac{p_1}{q_1} \rfloor$ disjoint subsets of q_1 points, each of which are Λ-poised in $B(0;1)$ (in the interpolation sense), then $\hat{Y}$ is strongly $\sqrt{\frac{l+1}{l}}\Lambda$-poised in $B(0;1)$ (in the regression sense).*

Proof. Let $Y_j = \{y_j^0, y_j^1, \ldots, y_j^q\}$, $j = 1, \ldots, l$ be the disjoint Λ-poised subsets of $\hat{Y}$, and let Y_r be the remaining points in $\hat{Y}$. Let λ_i^j, $i = 0, \ldots, q$ be the (interpolation) Lagrange polynomials for the set Y_j. As noted in [6], for any $x \in \mathbb{R}^n$,

$$\sum_{i=0}^{q} \lambda_i^j(x)\bar{\phi}(y_j^i) = \bar{\phi}(x),\ j = 1, \ldots, l.$$

Dividing each of these equations by l and summing yields

$$\sum_{j=1}^{l}\sum_{i=0}^{q} \frac{1}{l}\lambda_i^j(x)\bar{\phi}(y_j^i) = \bar{\phi}(x). \tag{5.1}$$

Let $\lambda^j(x) = \left(\lambda_1^j(x), \cdots, \lambda_{q_1}^j(x)\right)^T$, and let $\bar{\lambda} \in \mathbb{R}^{p_1}$ be formed by concatenating the $\lambda^j(x)$, $j = 1, \cdots, l$ and a zero vector of length $|Y_r|$ and then dividing every entry by

l. By (5.1), $\bar{\lambda}$ is a solution to the equation

$$\sum_{i=0}^{p} \bar{\lambda}_i \bar{\phi}(y^i) = \bar{\phi}(x). \tag{5.2}$$

Since Y_j is Λ-poised in $B(0;1)$, for any $x \in B(0;1)$,

$$\left\|\lambda^j(x)\right\| \leq \sqrt{q_1} \left\|\lambda^j(x)\right\|_\infty \leq \sqrt{q_1}\Lambda.$$

Thus,

$$\left\|\bar{\lambda}\right\| \leq \sqrt{l} \max_j \frac{\left\|\lambda^j(x)\right\|}{l} \leq \sqrt{\frac{q_1}{l}}\Lambda \leq \sqrt{\frac{l+1}{l}}\sqrt{\frac{q_1}{p_1/q_1}}\Lambda = \sqrt{\frac{l+1}{l}}\frac{q_1}{\sqrt{p_1}}\Lambda.$$

Let $\ell_i(x)$, $i = 0, \cdots, p_1$ be the regression Lagrange polynomials for the complete set $\hat{Y}$. As observed in [6], $\ell(x) = (\ell_0(x), \cdots, \ell_{p_1}(x))^T$ is the minimum norm solution to (5.2). Thus,

$$\|\ell(x)\| \leq \left\|\bar{\lambda}\right\| \leq \sqrt{\frac{l+1}{l}}\frac{q_1}{\sqrt{p_1}}\Lambda.$$

Since this holds for all $x \in B(0;1)$, $\hat{Y}$ is strongly $\sqrt{\dfrac{l+1}{l}}\Lambda$-poised in $B(0;1)$. ∎

Based on this observation, and noting that $\frac{l+1}{l} \leq 2$ for $l \geq 1$, we adopt the following strategy for improving a shifted and scaled regression sample set $\hat{Y} \subset B(0;1)$:

1. If $\hat{Y}$ contains $l \geq 1$ Λ-poised subsets with at most q_1 points left over, $\hat{Y}$ is strongly $\sqrt{2}\Lambda$-poised.
2. Otherwise, if $\hat{Y}$ contains at least one Λ-poised subset, save as many Λ-poised subsets as possible, plus at most q_1 additional points from $\hat{Y}$, discarding the rest.
3. Otherwise, add additional points to $\hat{Y}$ in order to create a Λ-poised subset. Keep this subset, plus at most q_1 additional points from $\hat{Y}$.

To implement this strategy, we first describe an algorithm that attempts to find a Λ-poised subset of $\hat{Y}$. To discuss the algorithm we introduce the following definition:

DEFINITION 5.2. *A set $Y \subset B$ is said to be* Λ-subpoised *in a set B if there exists a superset $Z \supseteq Y$ that is Λ-poised in B with $|Z| = q_1$.*

Given a sample set $Y \subset B(0;1)$ (not necessarily shifted and scaled) and a radius $\tilde{\Delta}$, the algorithm below selects a Λ-subpoised subset $Y_{new} \subset Y$ containing as many points as possible. If $|Y_{new}| = q_1$, then Y_{new} is Λ-poised in $B(0;\tilde{\Delta})$ for some fixed Λ. Otherwise, the algorithm determines a new point $y_{new} \in B(0;\tilde{\Delta})$ such that $Y_{new} \bigcup \{y_{new}\}$ is Λ-subpoised in $B(0;\tilde{\Delta})$.

Algorithm FindSet. (Finds a Λ-subpoised set)

Input: A sample set $Y \subset B(0;1)$ and a trust region radius $\tilde{\Delta} \in \left[\sqrt{\xi_{acc}}, 1\right]$, for fixed parameter $\xi_{acc} > 0$.

Output: A set $Y_{new} \subset Y$ that is Λ-poised in $B(0;\tilde{\Delta})$; or a Λ-subpoised set $Y_{new} \subset B(0;\tilde{\Delta})$ and a new point $y_{new} \subset B(0;\tilde{\Delta})$ such that $Y_{new} \bigcup \{y_{new}\}$ is Λ-subpoised in $B(0;\tilde{\Delta})$.

Step 0 (Initialization:) Initialize the pivot polynomial basis to the monomial basis:

$u_i(x) = \bar{\phi}_i(x), i = 0, \ldots, q$. Set $Y_{new} = \emptyset$. Set $i = 0$.

Step 1 (Point Selection:) If possible, choose $j_i \in \{i, \ldots, |Y| - 1\}$ such that $|u_i(y^{j_i})| \geq \xi_{acc}$ (threshold test).

If such an index is found, add the corresponding point to Y_{new} and swap the positions of points y^i and y^{j_i} in Y.

Otherwise, compute $y_{new} = \arg\max_{x \in B(0;\tilde{\Delta})} |u_i(x)|$, and **exit**, returning Y_{new} and y_{new}.

Step 2 (Gaussian Elimination:) For $j = i + 1, \ldots, |Y| - 1$

$$u_j(x) = u_j(x) - \frac{u_j(y^i)}{u_i(y^i)} u_i(x).$$

If $i < q$, set $i = i + 1$ and go to step 1.

Exit, returning Y_{new}.

The algorithm, which is modeled after Algorithms 6.5 and 6.6 in [6], applies Gaussian elimination with a threshold test to form a basis of pivot polynomials $\{u_i(x)\}$. As discussed in [6], at the completion of the algorithm, the values $u_i(y^i), y^i \in Y_{new}$ are exactly the diagonal entries of the diagonal matrix D in the LDU factorization of $M = M(\bar{\phi}, Y_{new})$. If $|Y_{new}| = q_1$, M is a square matrix. In this case, since $\left|u_i(y^i)\right| \geq \xi_{acc}$,

$$\left\|M^{-1}\right\| \leq \frac{\sqrt{q_1}\xi_{growth}}{\xi_{acc}}, \tag{5.3}$$

where ξ_{growth} is the growth factor for the factorization (see [13]).

Point Selection. The point selection rule allows flexibility in how an acceptable point is chosen. For example, to keep the growth factor down, one could choose the index j_i that maximizes $|u_i(y^j)|$ (which corresponds to Gaussian elimination with partial pivoting). But in practice, it is often better to select points according to their proximity to the current iterate. In our implementation, we balance these two criteria by choosing the index that maximizes $|u_i(y^j)|/d_j^3$, over $j \geq i$, where $d_j = \max\{1, \left\|y^j\right\| / \tilde{\Delta}\}$. If all sample points are contained in $B(0; \tilde{\Delta})$, then $d_j = 1$ for all j. In this case, the point selection rule is identical to the one used in Algorithm 6.6 of [6] (with the addition of the threshold test). When Y contains points outside $B(0; \tilde{\Delta})$, the corresponding values of d_j are greater than 1, so the point selection rule gives preference to points that are within $B(0; \tilde{\Delta})$.

The theoretical justification for our revised point selection rule comes from examining the error bounds in Corollary 4.6. For a given point x in $B(0; \tilde{\Delta})$, each sample point y^i makes a contribution to the error bound that is proportional to $\left\|y^i - x\right\|^3$ (assuming the computational error is relatively small). Since x can be anywhere in the trust region, this suggests modifying the point selection rule to maximize $\dfrac{|u_i(y^{j_i})|}{\hat{d}_{j_i}^3}$, where $\hat{d}_j = \max_{x \in B(0;\tilde{\Delta})} \left\|y^j - x\right\| / \tilde{\Delta} = \left\|y^j\right\| / \tilde{\Delta} + 1$. To simplify analysis, we modify this formula so that all points inside the trust region are treated equally, resulting in the formula $d_j = \max(1, \left\|y^j\right\| / \tilde{\Delta})$.

LEMMA 5.3. *Suppose Algorithm FindSet returns a set Y_{new} with $|Y_{new}| = q_1$. Then Y_{new} is Λ-poised in $B(0; \tilde{\Delta})$ for some Λ, which is proportional to $\dfrac{\xi_{growth}}{\xi_{acc}} \max\{1, \tilde{\Delta}^2/2, \tilde{\Delta}\}$, where ξ_{growth} is the growth factor for the Gaussian elimination.*

Proof. Let $\tilde{M} = M(\bar{\phi}, Y_{new})$. By (5.3), $\left\|\tilde{M}^{-1}\right\| \leq \sqrt{q_1}\xi_{growth}/\xi_{acc}$. Let $\ell(x) = (\ell_0(x), \ldots, \ell_q(x))^T$ be the vector of interpolation Lagrange polynomials for the sample set Y_{new}. For any $x \in B(0; \tilde{\Delta})$,

$$\begin{aligned}\|\ell(x)\|_\infty &= \left\|\tilde{M}^{-T}\bar{\phi}(x)\right\|_\infty \leq \left\|\tilde{M}^{-1}\right\|_\infty \|\bar{\phi}(x)\|_\infty \leq \sqrt{q_1}\left\|\tilde{M}^{-1}\right\| \|\bar{\phi}(x)\|_\infty \\ &\leq \frac{\sqrt{q_1}\xi_{growth}}{\xi_{acc}} \|\bar{\phi}(x)\|_\infty \leq \frac{\sqrt{q_1}\xi_{growth}}{\xi_{acc}} \max\{1, \tilde{\Delta}^2/2, \tilde{\Delta}\}.\end{aligned}$$

Since this inequality holds for all $x \in B(0; \tilde{\Delta})$, Y_{new} is Λ-poised for $\Lambda = \left(\sqrt{q_1}\xi_{growth}/\xi_{acc}\right) \max\{1, \tilde{\Delta}^2/2, \tilde{\Delta}\}$, which establishes the result. ∎

In general, the growth factor in the above lemma depends on the matrix M and the threshold ξ_{acc}. Because of the threshold test, it is possible to establish a bound on the growth factor that is independent of $\tilde{M}$. So we can claim that the algorithm selects a Λ-poised set for a fixed Λ that is independent of Y. However, the bound is extremely large, so is not very useful. Nevertheless, in practice ξ_{growth} is quite reasonable, so Λ tends to be proportional to $\max\{1, \tilde{\Delta}^2/2, \tilde{\Delta}\}/\xi_{acc}$.

In the case where the threshold test is not satisfied, Algorithm FindSet determines a new point y_{new} by maximizing $|u_i(x)|$ over $B(0; \tilde{\Delta})$. In this case, we need to show that the new point would satisfy the threshold test. The following lemma shows that this is possible, provided ξ_{acc} is small enough. The proof is modeled after the proof of [6, Lemma 6.7].

LEMMA 5.4. *Let $v^T\bar{\phi}(x)$ be a quadratic polynomial of degree 2, where $\|v\|_\infty = 1$. Then*

$$\max_{x \in B(0;\tilde{\Delta})} |v^T\bar{\phi}(x)| \geq \min\{1, \frac{\tilde{\Delta}^2}{4}\}.$$

Proof. Since $\|v\|_\infty = 1$, at least one of the coefficients of $q(x) = v^T\bar{\phi}(x)$ is 1, -1, 1/2, or -1/2.

Looking at the case where the largest coefficient is 1 or 1/2 (-1 and -1/2 are similarly proven), either this coefficient corresponds to the constant term, a linear term x_i, or a quadratic term $x_i^2/2$ or x_ix_j. Restrict all variables not appearing in the term corresponding to the largest coefficient to zero.

- If $q(x) = 1$ then the lemma trivially holds.
- If $q(x) = x_i^2/2 + ax_i + b$, let $\tilde{x} = \tilde{\Delta}e^i \in B(0; \tilde{\Delta})$

  $$q(\tilde{x}) = \tilde{\Delta}^2/2 + \tilde{\Delta}a + b,\ q(-\tilde{x}) = \tilde{\Delta}^2/2 - \tilde{\Delta}a + b,\ \text{and } q(0) = b.$$

  If $|q(-\tilde{x})| \geq \frac{\tilde{\Delta}^2}{4}$ or $|q(\tilde{x})| \geq \frac{\tilde{\Delta}^2}{4}$ the result is shown. Otherwise, $\frac{-\tilde{\Delta}}{4} < q(\tilde{\Delta}) < \frac{\tilde{\Delta}^2}{4}$ and $\frac{-\tilde{\Delta}}{4} < q(-\tilde{\Delta}) < \frac{\tilde{\Delta}^2}{4}$. Adding these together yields $\frac{\tilde{\Delta}^2}{2} < \tilde{\Delta}^2 + 2b < \frac{\tilde{\Delta}^2}{2}$. Therefore $b < \frac{\tilde{\Delta}^2}{4} - \frac{\tilde{\Delta}^2}{2} = -\frac{\tilde{\Delta}^2}{4}$ and therefore $|q(0)| > \frac{\tilde{\Delta}^2}{4}$.
- If $q(x) = ax_i^2/2 + x_i + b$, then let $\tilde{x} = \tilde{\Delta}e^i$, yielding $q(\tilde{x}) = \tilde{\Delta} + a\tilde{\Delta}^2/2 + b$ and $q(-\tilde{x}) = -\tilde{\Delta} + a\tilde{\Delta}^2/2 + b$ then

  $$\max\{|q(-\tilde{x})|, |q(\tilde{x})|\} = \max\left\{|-\tilde{\Delta} + \alpha|, |\tilde{\Delta} + \alpha|\right\} \geq \tilde{\Delta} \geq \min\{1, \frac{\tilde{\Delta}^2}{4}\}$$

  where $\alpha = a\tilde{\Delta}/2 + b = 0$.

- If $q(x) = ax_i^2/2 + bx_j^2/2 + x_i x_j + cx_i + dx_j + e$, we consider 4 points on $B(0;\tilde{\Delta})$

$$y_1 = \begin{bmatrix} \sqrt{\frac{\tilde{\Delta}}{2}} \\ \sqrt{\frac{\tilde{\Delta}}{2}} \end{bmatrix}, \quad y_2 = \begin{bmatrix} \sqrt{\frac{\tilde{\Delta}}{2}} \\ -\sqrt{\frac{\tilde{\Delta}}{2}} \end{bmatrix}, \quad y_3 = \begin{bmatrix} -\sqrt{\frac{\tilde{\Delta}}{2}} \\ \sqrt{\frac{\tilde{\Delta}}{2}} \end{bmatrix}, \quad y_4 = \begin{bmatrix} -\sqrt{\frac{\tilde{\Delta}}{2}} \\ -\sqrt{\frac{\tilde{\Delta}}{2}} \end{bmatrix}.$$

$$q(y_1) = \frac{a\tilde{\Delta}}{4} + \frac{b\tilde{\Delta}}{4} + \frac{\tilde{\Delta}}{2} + c\sqrt{\frac{\tilde{\Delta}}{2}} + d\sqrt{\frac{\tilde{\Delta}}{2}} + e$$

$$q(y_2) = \frac{a\tilde{\Delta}}{4} + \frac{b\tilde{\Delta}}{4} - \frac{\tilde{\Delta}}{2} + c\sqrt{\frac{\tilde{\Delta}}{2}} - d\sqrt{\frac{\tilde{\Delta}}{2}} + e$$

$$q(y_3) = \frac{a\tilde{\Delta}}{4} + \frac{b\tilde{\Delta}}{4} - \frac{\tilde{\Delta}}{2} - c\sqrt{\frac{\tilde{\Delta}}{2}} + d\sqrt{\frac{\tilde{\Delta}}{2}} + e$$

$$q(y_4) = \frac{a\tilde{\Delta}}{4} + \frac{b\tilde{\Delta}}{4} + \frac{\tilde{\Delta}}{2} - c\sqrt{\frac{\tilde{\Delta}}{2}} - d\sqrt{\frac{\tilde{\Delta}}{2}} + e$$

Note that $q(y_1) - q(y_2) = \tilde{\Delta} + 2d\sqrt{\frac{\tilde{\Delta}}{2}}$ and $q(y_3) - q(y_4) = -\tilde{\Delta} + 2d\sqrt{\frac{\tilde{\Delta}}{2}}$. There are two cases:

1. If $d \geq 0$, then $q(y_1) - q(y_2) \geq \tilde{\Delta}$, so either $|q(y_1)| \geq \frac{\tilde{\Delta}}{2}$ or $|q(y_2)| \geq \min\{1, \frac{\tilde{\Delta}}{2}\}$.
2. If $d < 0$, then a similar study of $q(y_3) - q(y_4)$ proves the result.

□

LEMMA 5.5. *Suppose $\xi_{acc} \leq \min\{1, \tilde{\Delta}^2/4\}$. If Algorithm FindSet exits during the point selection step, then $Y_{new} \bigcup \{y_{new}\}$ is Λ-subpoised in $B(0;\tilde{\Delta})$ for some fixed Λ, which is proportional to $\frac{\xi_{growth}}{\xi_{acc}} \max\{1, \tilde{\Delta}^2/2, \tilde{\Delta}\}$, where ξ_{growth} is the growth parameter for the Gaussian elimination.*

Proof. Consider a modified version of Algorithm FindSet that does not exit in the point selection step. Instead, y^i is replaced by y_{new} and y_{new} is added to Y_{new}. This modified algorithm will always return a set consisting of q_1 points. Call this set Z. Let Y_{new} and y_{new} be the output of the unmodified algorithm, and observe that $Y_{new} \bigcup \{y_{new}\} \subset Z$.

To show that $Y_{new} \bigcup \{y_{new}\}$ is Λ-subpoised, we show that Z is Λ-poised in $B(0;\tilde{\Delta})$. From the Gaussian elimination, after k iterations of the algorithm, the $(k+1)$st pivot polynomial $u_k(x)$ can be expressed as $(v^k)^T \bar{\phi}(x)$, where $v^k = (v_0, \ldots, v_{k-1}, 1, 0, \ldots, 0)$. Observe that $\|v^k\|_\infty \geq 1$, and let $\tilde{v} = \frac{v^k}{\|v^k\|_\infty}$. By Lemma 5.4,

$$\max_{x \in B(0;\tilde{\Delta})} |u_k(x)| = \max_{x \in B(0;\tilde{\Delta})} \left|(v^k)^T \bar{\phi}(x)\right| = \|v^k\|_\infty \left(\max_{x \in B(0;\tilde{\Delta})} |\tilde{v}^T \bar{\phi}(x)| \right)$$
$$\geq \min\{1, \frac{\tilde{\Delta}^2}{4}\} \|v^k\|_\infty \geq \min\{1, \frac{\tilde{\Delta}^2}{4}\} \geq \xi_{acc}.$$

It follows that each time a new point is chosen in the point selection step, that point will satisfy the threshold test. Thus, the set Z returned by the modified algorithm will include q_1 points, all of which satisfy the threshold test. By Lemma 5.3,

Z is Λ-poised, with Λ proportional to $\frac{\xi_{growth}}{\xi_{acc}} \max\{1, \tilde{\Delta}^2/2, \tilde{\Delta}\}$. It follows that $Y_{new} \bigcup \{y_{new}\}$ is Λ-subpoised. □

We are now ready to state our model improvement algorithm for regression. Prior to calling this algorithm, we discard all points in Y with distance greater than $\Delta/\sqrt{\xi_{acc}}$ from y^0. We then form the shifted and scaled set Y by the transformation $y^j = (y^j - y^0)/d$, where $d = \max_{y^j \in Y} \left\|y^j - y^0\right\|$, and scale the trust region radius accordingly (i.e., $\tilde{\Delta} = \Delta/d$). This ensures that $\tilde{\Delta} = \frac{\Delta}{d} \geq \frac{\Delta}{\Delta/\sqrt{\xi_{acc}}} = \sqrt{\xi_{acc}}$. After calling the algorithm, we reverse the shift and scale transformation.

Algorithm MIA. (Model Improvement for Regression)

Input: A shifted and scaled sample set $\hat{Y} \subset B(0;1)$, a trust region radius $\tilde{\Delta} \geq \sqrt{\xi_{acc}}$ for fixed $\xi_{acc} \in (0, \frac{1}{4r^2})$, where $r \geq 1$ is a fixed parameter.

Output: A modified set Y' with improved poisedness on $B(0;\tilde{\Delta})$.

Step 0 (Initialization:) Remove the point in $\hat{Y}$ farthest from $y^0 = 0$ if it is outside $B(0; r\tilde{\Delta})$. Set $Y' = \emptyset$.

Step 1 (Find Poised Subset:) Use Algorithm **FindSet** either to identify a Λ-poised subset $Y_{new} \subset \hat{Y}$, or to identify a Λ-subpoised subset $Y_{new} \subset \hat{Y}$ and one additional point $y_{new} \in B(0;\tilde{\Delta})$ such that $Y_{new} \bigcup \{y_{new}\}$ is Λ-subpoised on $B(0;\tilde{\Delta})$.

Step 2 (Update Set:)

If Y_{new} is Λ-poised, add it to Y' and remove Y_{new} from $\hat{Y}$. Remove all points from $\hat{Y}$ that are outside of $B(0; r\tilde{\Delta})$.

Otherwise, if $|Y'| = \emptyset$, set $Y' = Y_{new} \bigcup \{y_{new}\}$ plus $q_1 - |Y_{new}| - 1$ other points from $\hat{Y}$. Otherwise, set $Y' = Y' \bigcup Y_{new}$ plus $q_1 - |Y_{new}|$ additional points from $\hat{Y}$. Set $\hat{Y} = \emptyset$.

Step 3 If $|\hat{Y}| \geq q_1$, go to **Step 1**.

Step 4 Set $Y' = Y' \bigcup Y_{new}$

In Algorithm MIA, if every call to Algorithm FindSet yields a Λ-poised set Y_{new}, then eventually all points in $\hat{Y}$ will be included in Y'. In this case, the algorithm has verified that $\hat{Y}$ contains $\ell = \lfloor \frac{p_1}{q_1} \rfloor$ Λ-poised sets. By Proposition 5.1, $\hat{Y}$ is strongly $\frac{l+1}{l}\Lambda$-poised in $B(0;1)$.

If the first call to FindSet fails to identify a Λ-poised subset, the algorithm improves the sample set by adding a new point y_{new} and by removing points so that the output set Y' contains at most q_1 points. In this case the output set contains the Λ-subpoised set $Y_{new} \bigcup \{y_{new}\}$. Thus, if the algorithm is called repeatedly, with $\hat{Y}$ replaced by Y' after each call, eventually Y' will contain a Λ-poised subset and will be strongly 2Λ-poised, by Proposition 5.1.

If $\hat{Y}$ fails to be Λ-poised after the second or later call to FindSet, no new points are added. Instead, the sample set is improved by removing points from $\hat{Y}$ so that the output set Y' consists of all the Λ-poised subsets identified by FindSet, plus up to q_1 additional points. The resulting set is then strongly $\frac{\hat{\ell}+1}{\hat{\ell}}\Lambda$-poised, where $\hat{\ell} = \lfloor \frac{|Y_{new}|}{q_1} \rfloor$.

Trust region scale factor. The trust region scale factor r was suggested in [6, Section 11.2], although implementation details were omitted. The scale factor determines what points are allowed to remain in the sample set. Each call to Algorithm MIA removes a point from outside $B(0; r\tilde{\Delta})$ if such a point exists. Thus, if the algorithm is called repeatedly with $\hat{Y}$ replaced by Y' each time, eventually all points in the

sample set will be in the region $B(0; r\tilde{\Delta})$. Using a scale factor $r > 1$ can improve the efficiency of the algorithm. To see this, observe that if $r = 1$, a slight movement of the trust region center may result in previously "good" points lying just outside of $B(y^0; \Delta)$. These points would then be unnecessarily removed from $\hat{Y}$.

To justify this approach, suppose that $\hat{Y}$ is strongly Λ-poised in $B(0; \tilde{\Delta})$. By Proposition 4.10, the associated model function m is κ-fully quadratic for some fixed vector κ, which depends on Λ. If instead $\hat{Y}$ has points outside of $B(0; \tilde{\Delta})$, we can show (by a simple modification to the proof of Proposition 4.10) that the model function is $R^3\kappa$-fully quadratic, where $R = \max\left\{\left\|y^i - y^0\right\|\right\}$. Thus, if $\hat{Y} \subset B(0; r\tilde{\Delta})$ for some fixed $r \geq 1$, then calling the model improvement algorithm will result in a model function m that is $\hat{\kappa}$-fully quadratic with respect to a different (but still fixed) $\hat{\kappa} = r^3\kappa$. In this case, however, whenever new points are added during the model improvement algorithm, they are always chosen within the original trust region $B(0; \tilde{\Delta})$.

The discussion above demonstrates that Algorithm MIA satisfies the requirements of a model improvement algorithm specified in Definition 2.2. This algorithm is used in the CSV2 framework described in Section 2 as follows:

- In Step 1 of Algorithm CSV2, Algorithm MIA is called once. If no change is made to the sample set, the model is certified to be κ-fully quadratic.
- In Step 4 of Algorithm CSV2, Algorithm MIA is called once. If no change is made to the sample set, the model is κ-fully quadratic. Otherwise, the sample set has been modified to improve the model.
- In Algorithm CriticalityStep, Algorithm MIA is called repeatedly to improve the model until it is κ-fully quadratic.

In our implementation, we modified Algorithm CriticalityStep to improve efficiency by introducing an additional exit criterion. Specifically, after each call to the model improvement algorithm, ϑ_k^m is tested. If $\vartheta_k^m > \epsilon_c$, x^k is no longer a second order stationary point of the model function; so we exit the criticality step.

6. Computational Results. As shown in the previous section, the CSV2 framework using weighted quadratic regression converges to a second-order stationary point provided the ratio between the largest and smallest weight is bounded. This leaves much leeway in the derivation of the weights. We now describe a heuristic strategy based on the error bounds derived in §4.

6.1. Using Error Bounds to Choose Weights. Intuitively, the models used throughout our algorithm will be most effective if the weights are chosen so that $m(x)$ is as accurate as possible in the sense that it agrees with the 2nd order Taylor approximation of $f(x)$ around the current trust region center y^0. That is, we want to estimate the quadratic function

$$q(x) = f(y^0) + \nabla f(y^0)^T(x - y^0) + \frac{1}{2}(x - y^0)^T\nabla^2 f(y^0)(x - y^0).$$

If $f(x)$ happens to be a quadratic polynomial, then

$$\bar{f}_i = q(y^i) + E_i.$$

If the errors E_i are uncorrelated random variables with zero mean and finite variances $\sigma_i^2, i = 0, \ldots, p$, then the best linear unbiased estimator of the polynomial $q(x)$ is given by $m(x) = \phi(x)^T\alpha$, where α solves (3.4) with the i^{th} weight proportional to $1/\sigma_i$ [23, Theorem 4.4]. This is intuitively appealing since each sample point will have the same expected contribution to the weighted sum of square errors.

When $f(x)$ is not a quadratic function, the errors depend not just on the computational error, but also on the distances from each point to y^0. In the particular case when $x = y^0$, the first three terms of (4.2) are the quadratic function $q(y^i)$. Thus, the error between the computed function value and $q(y^i)$ is given by:

$$\bar{f}_i - q(y^i) = \frac{1}{2}(y^i - y^0)^T H_i(y^0)(y^i - y^0) + E_i, \tag{6.1}$$

where $H_i(y^0) = \nabla^2 f(\eta_i(y^0)) - \nabla^2 f(y^0)$ for some point $\eta_i(y^0)$ on the line segment connecting y^0 and y^i.

We shall refer to the first term on the right as the *Taylor error* and the second term on the right as the *computational error*. By Assumption 2.1, $\left\|H_i(y^0)\right\| \leq L \left\|y^i - y^0\right\|$. This leads us to the following heuristic argument for choosing the weights: Suppose that $H_i(y^0)$ is a random symmetric matrix such that the standard deviation of $\left\|H_i(y^0)\right\|$ is proportional to $L \left\|y^i - y^0\right\|$. Then the Taylor error will have standard deviation proportional to $L \left\|y^i - y^0\right\|^3$. Assuming the computational error is independent of the Taylor error, the *total error* $\bar{f}_i - q(y^i)$ will have standard deviation $\sqrt{\left(\zeta L \left\|y^i - y^0\right\|^3\right)^2 + \sigma_i^2}$, where ζ is a proportionality constant, and σ_i is the standard deviation of E_i. This leads to the following formula for the weights:

$$w_i \propto \frac{1}{\sqrt{\zeta^2 L^2 \left\|y^i - y^0\right\|^6 + \sigma_i^2}}.$$

Of course, this formula depends on knowing ζ, L and σ_i. If L, σ_i, and/or ζ are not known, this formula could still be used in conjunction with some strategy for estimating L, σ_i, and ζ (for example, based upon the accuracy of the model functions at known points). Alternatively, ζ and L can be combined into a single parameter C that could be chosen using some type of adaptive strategy:

$$w_i \propto \frac{1}{\sqrt{C \left\|y^i - y^0\right\|^6 + \sigma_i^2}}.$$

If the computational errors have equal variances, the formula can be further simplified as

$$w_i \propto \frac{1}{\sqrt{\bar{C} \left\|y^i - y^0\right\|^6 + 1}}, \tag{6.2}$$

where $\bar{C} = C/\sigma_i^2$.

An obvious flaw in the above development is that the errors in $|\bar{f}_i - q(y^i)|$ are not uncorrelated. Additionally, the assumption that $\left\|H_i(y^0)\right\|$ is proportional to $L \left\|y^i - y^0\right\|$ is valid only for limited classes of functions. Nevertheless, based on our computational experiments, (6.2) appears to provide a sensible strategy for balancing differing levels of computational uncertainty with the Taylor error.

6.2. Benchmark Performance. To study the impact of weighted regression, we developed MATLAB implementations of three quadratic model-based trust region algorithms using interpolation, regression, and weighted regression, respectively, to construct the quadratic model functions. To the extent possible, the differences

between these algorithms were minimized, with code shared whenever possible. Obviously, all three methods use different strategies for constructing the model from the sample set. Beyond that, the only difference is that the two regression methods use the model improvement algorithm described in Section 5, whereas the interpolation algorithm uses the model improvement algorithm described in [6, Algorithm 6.6].

We compared the three algorithms using the suite of test problems for benchmarking derivative-free optimization algorithms made available by Moré and Wild [19]. We ran our tests on 3 types of problems from this test suite: smooth problems (with no noise), piecewise smooth functions, and functions with deterministic noise. The suite also includes stochastically noisy function, but we did not consider these. Doing so would require significantly modifying the algorithm, for example by sampling points multiple times or removing excessively "old" points which appear locally optimal because of a large magnitude of negative noise. We consider such modifications non-trivial and outside the scope of the current work. The problems were run with the following parameter settings:
$\Delta_{max} = 100$, $\Delta_0^{icb} = 1$, $\eta_0 = 10^{-6}$, $\eta_1 = 0.5$, $\gamma = 0.5$, $\gamma_{inc} = 2$, $\epsilon_c = 0.01$, $\mu = 2$, $\beta = 0.5$, $\omega = .5$, $r = 3$, $\xi_{acc} = 10^{-4}$. For the interpolation algorithm, we used $\xi_{imp} = 1.01$, for the calls to [6, Algorithm 6.6].

As described in [19], the smooth problems are derived from 22 nonlinear least squares functions defined in the CUTEr [12] collection. For each problem, the objective function $f(x)$ is defined by

$$f(x) = \sum_{k=1}^{m} g_k(x)^2,$$

where $g : \mathbb{R}^n \to \mathbb{R}^m$ represents one of the CUTEr test functions. The piecewise-smooth problems are defined using the same CUTEr test functions by

$$f(x) = \sum_{k=1}^{m} |g_k(x)| .$$

The noisy problems are derived from the smooth problems by multiplying by a noise function as follows:

$$f(x) = (1 + \varepsilon_f \Gamma(x)) \sum_{k=1}^{m} g_k(x)^2,$$

where ε_f defines the relative noise level, and $\Gamma(x)$ is a function that oscillates between -1 and 1, with both high-frequency and low-frequency oscillations. Using multiple starting points for some of the test functions, a total of 53 different problems are specified in the test suite for each of these 3 types of problems.

For the weighted regression algorithm, the weights were determined by the weighting scheme (6.2), with $\alpha = 1$, and $\bar{C} = 100$.

The relative performances of the algorithms were compared using performance profiles and data profiles [10, 19]. If S is the set of solvers to be compared on a suite of problems P, let $t_{p,s}$ be the number of iterates required for solver $s \in S$ on a problem $p \in P$ to find a function value satisfying:

$$f(x) \leq f_L + \tau \left(f(x_0) - f_L \right),$$

where f_L is the best function value achieved by any $s \in S$. Then the performance profile of a solver $s \in S$ is the following fraction:

$$\rho_s(\alpha) = \frac{1}{|P|} \left| \left\{ p \in P : \frac{t_{p,s}}{\min \{t_{p,s} : s \in S\}} \leq \alpha \right\} \right|.$$

The data profile of a solver $s \in S$ is:

$$d_s(\alpha) = \frac{1}{|P|} \left| \left\{ p \in P : \frac{t_{p,s}}{n_p + 1} \leq \alpha \right\} \right|$$

where n_p is the dimension of problem $p \in P$. For more information on these profiles, including their relative merits and faults, see [19].

The stopping criterion for each problem is based on the best function value f_L achieved by any of the methods. Specifically, the test is satisfied if

$$f(x) \leq f_L + \tau \left(f(0) - f_L \right),$$

where $\tau > 0$ is a specified accuracy.

Performance profiles comparing the three algorithms are shown in Figures 6.1–6.2, for an accuracy of $\tau = 10^{-5}$. We observe that on the smooth problems, the weighted and unweighted regression methods had similar performance and both performed slightly better than interpolation. For the noisy problems, we see slightly better performance from the weighted regression method. And for the piecewise differentiable functions, the performance of the weighted regression method is significantly better. This mirrors the findings in [7] where SID-PSM using regression models shows considerable improvement over interpolation models.

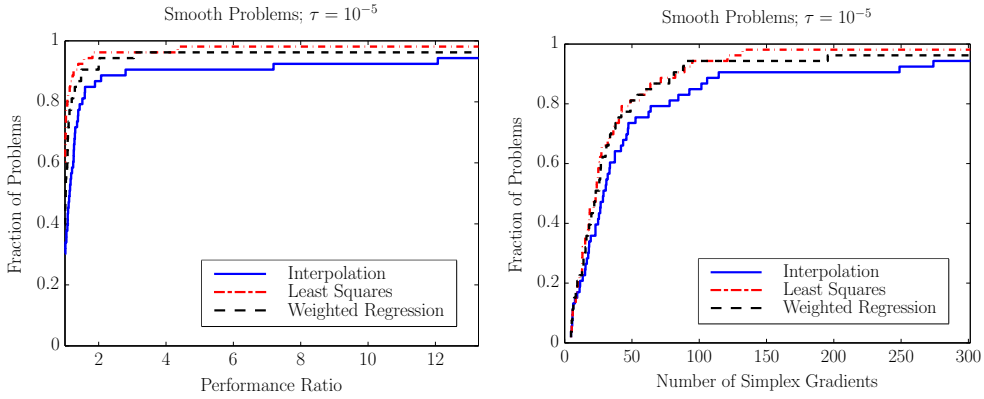

FIGURE 6.1. *Performance (left) and data (right) profiles: Interpolation vs. regression vs. weighted regression (Smooth Problems)*

We also compared our weighted regression algorithm with the DFO algorithm [3] as well as NEWUOA [22], (which had the best performance of the three solvers compared in [19]). We obtained the DFO code from the COIN-OR website [17]. This code calls IPOPT, which we also obtained from COIN-OR. We obtained NEWUOA from [18]. We ran both algorithms on the benchmark problems with a stopping criteria of $\Delta_{min} = 10^{-8}$, where Δ_{min} denotes the minimum trust region radius. For NEWUOA, the number of interpolation conditions was set to NPT=$2n + 1$.

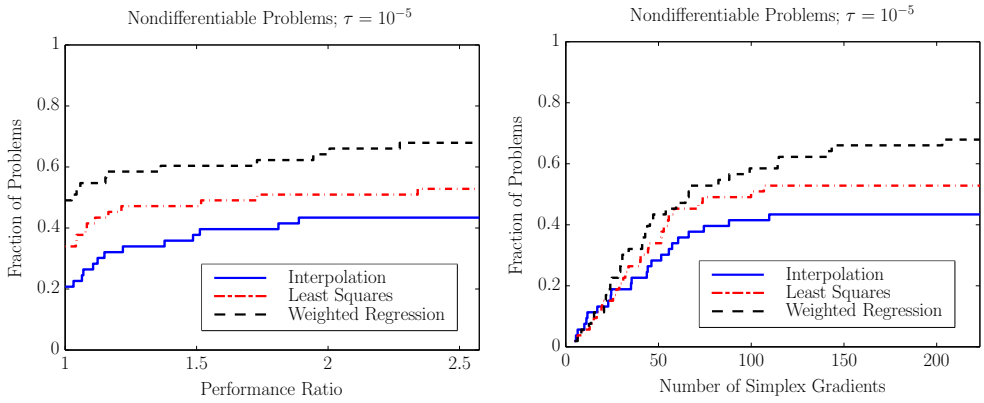

FIGURE 6.2. *Performance (left) and data (right) profiles: Interpolation vs. regression vs. weighted regression (Nondifferentiable Problems)*

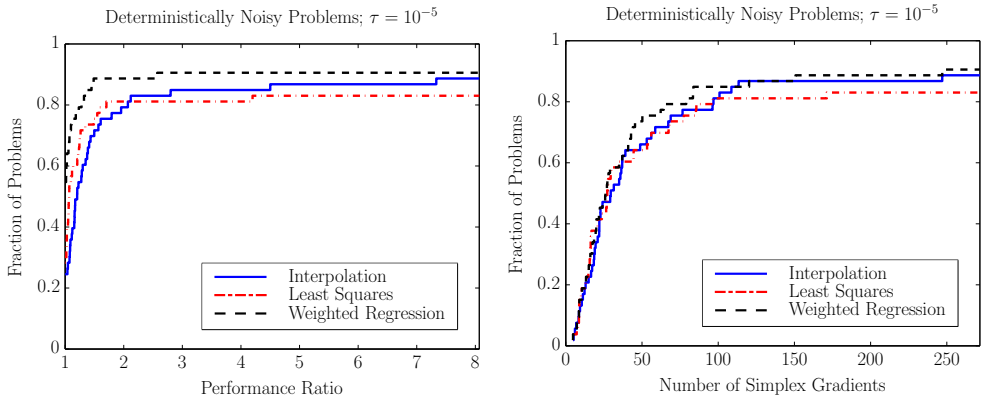

FIGURE 6.3. *Performance (left) and data (right) profiles: Interpolation vs. regression vs. weighted regression (Problems with Deterministic Noise)*

The performance profiles are shown in Figures 6.4–6.5, with an accuracy of $\tau = 10^{-5}$. NEWUOA outperforms both our algorithm and DFO on the smooth problems. This is not surprising since NEWUOA is a mature code that has been refined over several years, whereas our code is a relatively unsophisticated implementation. In contrast, on the noisy problems and the piecewise differentiable problems, our weighted regression algorithm achieves superior performance.

7. Summary and Conclusions. Our computational results indicate that using weighted regression to construct more accurate model functions can reduce the number of function evaluations required to reach a stationary point. Encouraged by these results, we believe that further study of weighted regression methods is warranted. This paper provides a theoretical foundation for such study. In particular, we have extended the concepts of Λ-poisedness and strong Λ-poisedness to the weighted regression framework, and we demonstrated that any scheme that maintains strongly Λ-poised sample sets for (unweighted) regression can also be used to maintain strongly Λ-poised sample sets for weighted regression, provided that no weight is too small relative to the other weights. Using these results, we showed that, when the computational error is sufficiently small relative to the trust region radius, the

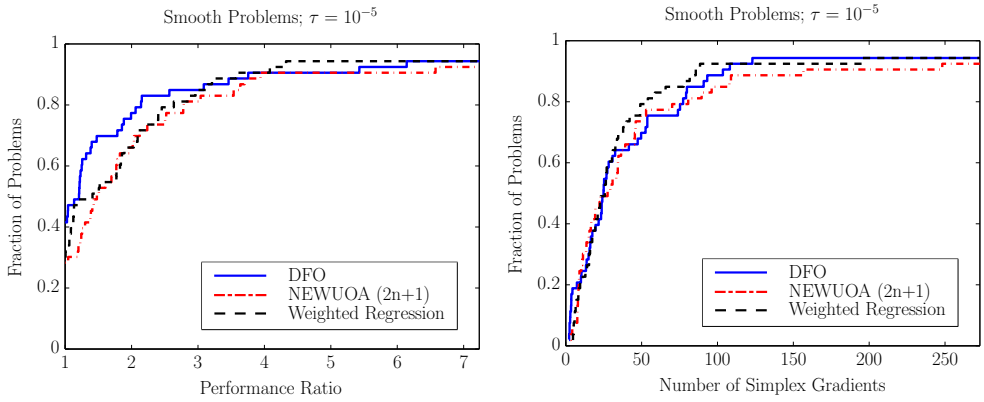

FIGURE 6.4. *Performance (left) and data (right) profiles: weighted regression vs. NEWUOA vs. DFO (Smooth Problems)*

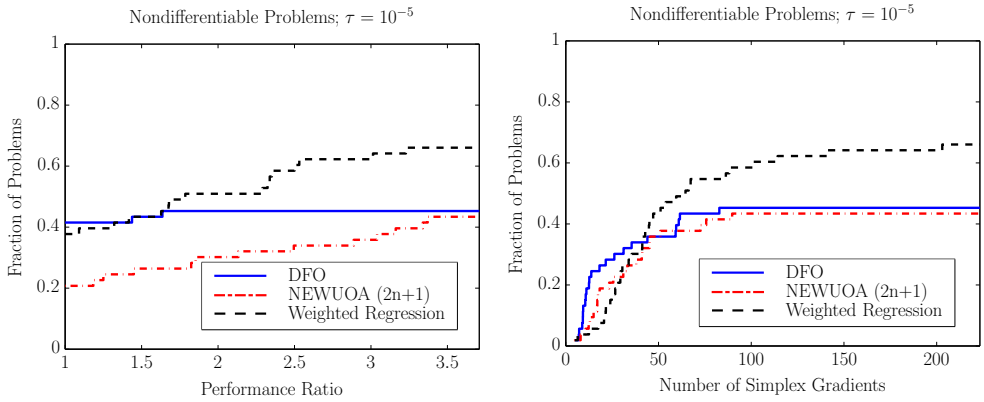

FIGURE 6.5. *Performance (left) and data (right) profiles: weighted regression vs. NEWUOA vs. DFO (Nondifferentiable Problems)*

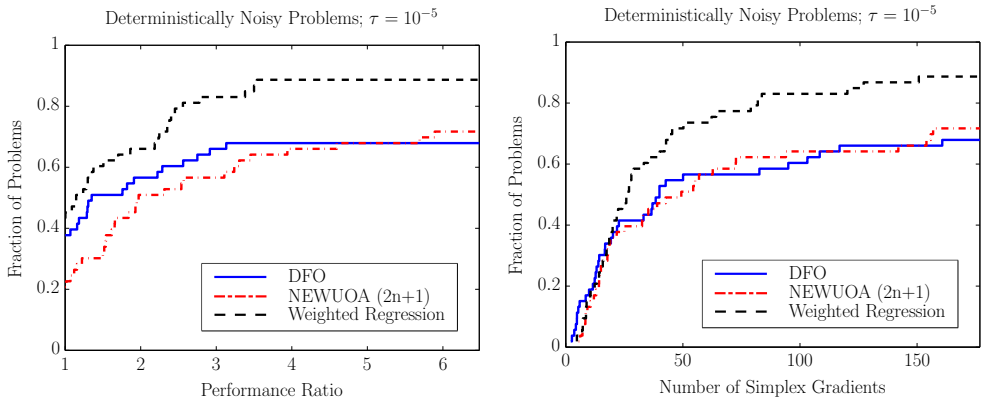

FIGURE 6.6. *Performance (left) and data (right) profiles: weighted regression vs. NEWUOA vs. DFO (Problems with Deterministic Noise)*

algorithm converges to a stationary point under standard assumptions.

This investigation began with a goal of more efficiently dealing with computational error in derivative-free optimization, particularly under varying levels of uncertainty. Surprisingly, we discovered that regression based methods can be advantageous even in the absence of computational error. Regression methods produce quadratic approximations that better fit the objective function close to the trust region center. This is due partly to the fact that interpolation methods throw out points that are close together in order to maintain a well-poised sample set. In contrast, regression models keep these points in the sample set, thereby putting greater weight on points close to the trust region center.

The question of how to choose weights needs further study. In this paper, we proposed a heuristic that balances uncertainties arising from computational error with uncertainties arising from poor model fidelity (i.e., Taylor error as described in §6.1). This weighting scheme appears to provide a benefit for noisy problems or non-differentiable problems. We believe better schemes can be devised based on more rigorous analysis.

Finally, we note that the advantage of regression-based methods is not without cost in terms of computational efficiency. In the regression methods, quadratic models are constructed from scratch every iteration, requiring $O(n^6)$ operations. In contrast, interpolation-based methods are able to use an efficient scheme developed by Powell [22] to update the quadratic models at each iteration. It is not clear whether such a scheme can be devised for regression methods. Nevertheless, when function evaluations are extremely expensive, and when the number of variables is not too large, this advantage is outweighed by the reduction in function evaluations realized by regression based methods.

8. Acknowledgements. This research was partially supported by National Science Foundation Grant GK-12-0742434. We are grateful to two anonymous referees for their comments that led to improvement of an earlier version of the paper.

REFERENCES

[1] E. J. Anderson and M. C. Ferris, *A direct search algorithm for optimization with noisy function evaluations*, SIAM Journal on Optimization, 11 (2001), pp. 837–857.

[2] P. G. Ciarlet and P. A. Raviart, *General Lagrange and Hermite interpolation in R^n with applications to finite element methods*, Archive for Rational Mechanics and Analysis, 46 (1972), pp. 177–199.

[3] A. R. Conn, K. Scheinberg, and P. L. Toint, *Recent progress in unconstrained nonlinear optimization without derivatives*, Mathematical Programming, 79 (1997), pp. 397–414.

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

[5] ———, *Global Convergence of General Derivative-Free Trust-Region Algorithms to First- and Second-Order Critical Points*, SIAM Journal on Optimization, 20 (2009), pp. 387–415.

[6] ———, *Introduction to Derivative-Free Optimization*, MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2009.

[7] A. L. Custódio, H. Rocha, and L. N. Vicente, *Incorporating minimum Frobenius norm models in direct search*, Computational Optimization and Applications, 46 (2010), pp. 265–278.

[8] G. Deng and M. C. Ferris, *Adaptation of the UOBQYA algorithm for noisy functions*, in Proceedings of the Winter Simulation Conference, 2006, pp. 312–319.

[9] ———, *Extension of the direct optimization algorithm for noisy functions*, in Proceedings of the Winter Simulation Conference, 2007, pp. 497–504.

[10] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91 (2001), pp. 201–213.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 3rd ed., 1996.
[12] N. I. M. Gould, D. Orban, and P. L. Toint, *CUTEr and SifDec: A constrained and unconstrained testing environment, revisited*, ACM Transactions on Mathematical Software, 29 (2003), pp. 373–394.
[13] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 2nd ed., 2002.
[14] W. Huyer and A. Neumaier, *SNOBFIT stable noisy optimization by branch and fit*, ACM Transactions on Mathematical Software, 35 (2008), pp. 1–25.
[15] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, *Lipschitzian optimization without the Lipschitz constant*, Journal of Optimization Theory and Applications, 79 (1993), pp. 157–181.
[16] C. T. Kelley, *Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition*, SIAM Journal on Optimization, 10 (1999), pp. 43–55.
[17] R. Lougee-Heimer, *The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community*, IBM Journal of Research and Development, 47 (2003), pp. 57–66.
[18] H. D. Mittelmann, *Decision tree for optimization software.* http://plato.asu.edu/guide.html, 2010.
[19] J. J. Moré and S. M. Wild, *Benchmarking derivative-free optimization algorithms*, SIAM Journal on Optimization, 20 (2009), pp. 172–191.
[20] J. A. Nelder and R. Mead, *A simplex method for function minimization*, Computer Journal, 7 (1965), pp. 308–313.
[21] M. J. D. Powell, *UOBYQA: Unconstrained optimization by quadratic approximation*, Mathematical Programming, 92 (2002), pp. 555–582.
[22] ———, *Developments of NEWUOA for minimization without derivatives*, IMA Journal on Numerical Analysis, 28 (2008), pp. 649–664.
[23] C. R. Rao and H. Toutenburg, *Linear Models: Least Squares and Alternatives*, Springer Series in Statistics, Springer-Verlag, 2nd ed., 1999.
[24] J. J. Tomick, S. F. Arnold, and R. R. Barton, *Sample size selection for improved Nelder-Mead performance*, in Proceedings of the Winter Simulation Conference, 1995, pp. 341–345.