ORBIT: OPTIMIZATION BY RADIAL BASIS FUNCTION INTERPOLATION IN TRUST-REGIONS*

STEFAN M. WILD†, ROMMEL G. REGIS‡, AND CHRISTINE A. SHOEMAKER§

Abstract. We present a new derivative-free algorithm, ORBIT, for unconstrained local optimization of computationally expensive functions. A trust-region framework using interpolating Radial Basis Function (RBF) models is employed. The RBF models considered often allow ORBIT to interpolate nonlinear functions using fewer function evaluations than the polynomial models considered by present techniques. Approximation guarantees are obtained by ensuring that a subset of the interpolation points are sufficiently poised for linear interpolation. The RBF property of conditional positive definiteness yields a natural method for adding additional points. We present numerical results on test problems to motivate the use of ORBIT when only a relatively small number of expensive function evaluations are available. Results on two very different application problems, calibration of a watershed model and optimization of a PDE-based bioremediation plan, are also very encouraging and support ORBIT's effectiveness on blackbox functions for which no special mathematical structure is known or available.

Key words. Derivative-Free Optimization, Radial Basis Functions, Trust-Region Methods, Nonlinear Optimization.

AMS subject classifications. 65D05, 65K05, 90C30, 90C56.

1. Introduction.

In this paper we address unconstrained local minimization

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1.1}$$

of a computationally expensive, real-valued deterministic function f assumed to be continuous and bounded from below. While we require additional smoothness properties to guarantee convergence of the algorithm presented, we assume that all derivatives of f are either unavailable or intractable to compute or approximate directly.

The principal motivation for the current work is optimization of complex deterministic computer simulations which usually entail numerically solving systems of partial differential equations governing underlying physical phenomena. These simulators often take the form of proprietary or legacy codes which must be treated as a *blackbox*, permitting neither insight into special structure or straightforward application of automatic differentiation techniques. For the purposes of this paper, we assume that any available parallel computing resources are devoted to parallelization within the computationally expensive objective function, and are not utilized by the optimization algorithm.

Unconstrained local optimization has been studied extensively in the nonlinear programming literature but much less frequently for the case when the computation or estimation of even ∇f is computationally intractable. Traditionally, if analytic derivatives are unavailable, practitioners rely on classical first-order techniques employing

*This work was supported by a Department of Energy Computational Science Graduate Fellowship, grant number DE-FG02-97ER25308 and NSF grants BES-0229176 and CCF-0305583. This research was conducted using the resources of the Cornell Theory Center, which receives funding from Cornell University, New York State, federal agencies, foundations, and corporate partners.

†School of Operations Research and Information Engineering, Cornell University, Rhodes Hall, Ithaca, NY 14853 (smw58@cornell.edu).

‡Cornell Theory Center, Cornell University, Rhodes Hall, Ithaca, NY, 14853 (rgr6@cornell.edu).

§School of Civil and Environmental Engineering and School of Operations Research and Information Engineering, Cornell University, Hollister Hall, Ithaca, NY, 14853. (cas12@cornell.edu).

finite difference-based estimates of ∇f to solve (1.1). However, as both the dimension and computational expense of the function grows, the n function evaluations required for such a gradient estimate are often better spent sampling the function elsewhere. For their ease of implementation and ability to find global solutions, heuristics such as genetic algorithms and simulated annealing are favored by engineers. However, these algorithms are often inefficient in achieving decreases in the objective function given only a limited number of function evaluations.

The approach followed by our ORBIT algorithm is based on forming a surrogate model which is computationally simple to evaluate and possesses well-behaved derivatives. This surrogate model approximates the true function locally by interpolating it at a set of sufficiently scattered data points. The surrogate model is optimized over compact regions to generate new points which can be evaluated by the computationally expensive function. By using this new function value to update the model, an iterative process develops. Over the last ten years, such derivative-free trust-region algorithms have become increasingly popular (see for example [5, 13, 14, 15]). However, they are often tailored to minimize the underlying computational complexity, as in [15], or to yield global convergence, as in [5]. In our setting we assume that the computational expense of function evaluation both dominates any possible internal optimization expense and limits the number of evaluations which can be performed.

In ORBIT, we have isolated the components which we believe to be responsible for the success of the algorithm in preliminary numerical experiments. As in the work of Powell [16], we form a nonlinear interpolation model using fewer than a quadratic (in the dimension) number of points. A so-called "fully linear tail" is employed to guarantee that the model approximates both the function and its gradient reasonably well, similar to the class of models considered by Conn, Scheinberg, and Vicente in [6]. Using a technique in the global optimization literature [2], additional interpolation points then generate a nonlinear model in a computationally stable manner.

In developing this new way of managing the set of interpolation points, we have simultaneously generalized the results of Oeuvray and Bierlaire [13] to include a richer set of RBF models and created an algorithm which is particularly efficient in the computationally expensive setting. While we have recently established a global convergence result in [22], the focus of this paper is on implementation details and the success of ORBIT in practice.

To the best of the authors' knowledge, besides [12], there has been no attempt in the literature to measure the relative performance of optimization algorithms when a limited number of function evaluations are available. In our case, this is due to the computational expense of the underlying function. Thus our numerical tests are also novel and we hope the present work will promote a discussion to better understand the goals of practitioners constrained by computational budgets.

1.1. Outline. We begin by providing the necessary background on trust-region methods and outlining the work done to date on derivative-free trust-region methods in Section 2. In Section 3 we introduce interpolating models based on RBFs. The computational details of the ORBIT algorithm are outlined in Section 4. In Section 5 we introduce techniques for benchmarking optimization algorithms in the computationally expensive setting and provide numerical results on standard test problems. Results on two applications from Environmental Engineering are presented in Section 6.

2. Trust-Region Methods. We begin with a review of the trust-region framework upon which our algorithm relies. Trust-region methods employ a surrogate

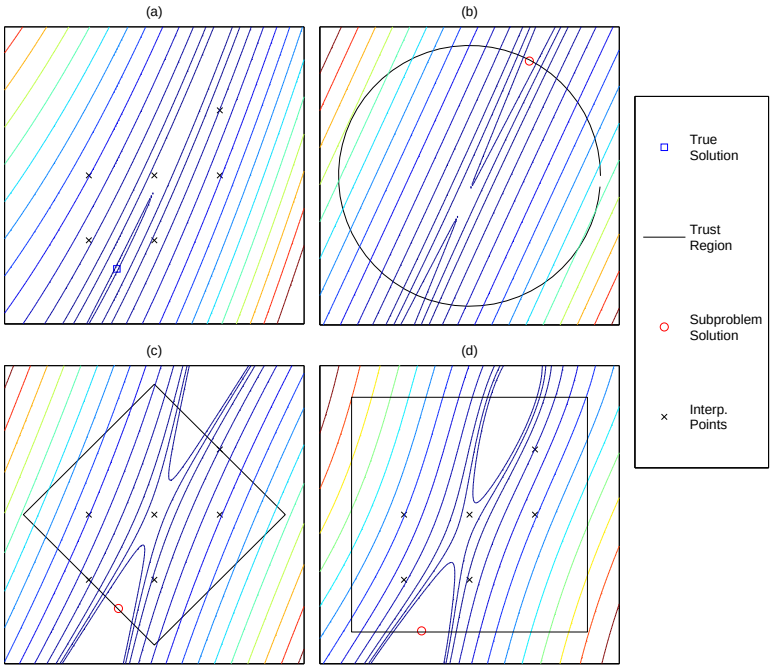

FIG. 2.1. *Trust-Region Subproblem Solutions: (a) True Function, (b) Quadratic Taylor Model in 2-norm Trust-Region, (c) Quadratic Interpolation Model in 1-norm Trust-Region, (d) RBF Interpolation Model in ∞-norm Trust-Region.*

model m_k which is assumed to approximate f within a neighborhood of the current iterate x_k. We define this so-called *trust-region* for an implied (center, radius) pair $(x_k, \Delta_k > 0)$ as:

$$\mathcal{B}_k = \{x \in \mathbb{R}^n : \|x - x_k\|_k \leq \Delta_k\}, \tag{2.1}$$

where we are careful to distinguish the trust-region norm (at iteration k), $\|\cdot\|_k$, from the standard 2-norm $\|\cdot\|$ and other norms used in the sequel. We assume here only that there exists a constant c_k (depending only on the dimension n) such that $\|\cdot\| \leq c_k \|\cdot\|_k$ for all k.

Trust-region methods obtain new points by solving a "subproblem" of the form:

$$\min \{m_k(x_k + s) : x_k + s \in \mathcal{B}_k\}. \tag{2.2}$$

As an example, in the upper left of Figure 2.1 we show the contours and optimal solution of the well-studied Rosenbrock function, $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$. The remaining plots show three different models: a derivative-based quadratic, an interpolation-based quadratic and a radial basis function model, approximating f within three different trust-regions, defined by the 2-norm, 1-norm and infinity-norm, respectively. The corresponding subproblem solution is also shown in each plot.

Given an approximate solution s_k to (2.2), the pair (x_k, Δ_k) is updated according to the ratio of actual to predicted improvement,

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}. \tag{2.3}$$

Build model m_k approximating f in the trust-region $\mathcal{B}_k$.
Solve Subproblem (2.2).
Evaluate $f(x_k + s_k)$ and compute ρ_k using (2.3).
Adjust trust-region according to:

$$\Delta_{k+1} = \begin{cases} \min\{\gamma_1 \Delta_k, \Delta_{\max}\} & \text{if } \rho_k \geq \eta_1 \\ \Delta_k & \text{if } \eta_0 \leq \rho_k < \eta_1 \\ \gamma_0 \Delta_k & \text{if } \rho_k < \eta_0, \end{cases}$$

$$x_{k+1} = \begin{cases} x_k + s_k & \text{if } \rho_k \geq \eta_0 \\ x_k & \text{if } \rho_k < \eta_0. \end{cases}$$

FIG. 2.2. *Iteration k of a Basic Trust-Region Algorithm.*

Given inputs $0 \leq \eta_0 \leq \eta_1 < 1$, $0 < \gamma_0 < 1 < \gamma_1$, $0 < \Delta_0 \leq \Delta_{\max}$, and $x_0 \in \mathbb{R}^n$, a general trust-region method proceeds iteratively as described in Figure 2.2. The design of the trust-region algorithm ensures that f is only sampled within the relaxed level set:

$$\mathcal{L}(x_0) = \{y \in \mathbb{R}^n : \|x - y\|_k \leq \Delta_{\max} \text{ for some } x \text{ with } f(x) \leq f(x_0)\}. \tag{2.4}$$

Usually a quadratic model,

$$m_k(x_k + s) = f(x_k) + g_k^T s + \frac{1}{2} s^T H_k s, \tag{2.5}$$

is employed and the approximate solution, s_k, to the subproblem (2.2) is required to satisfy a sufficient decrease condition of the form:

$$m_k(x_k) - m_k(x_k + s_k) \geq \frac{\kappa_d}{2} \|g_k\|_k \min\left\{ \frac{\|g_k\|_k}{\|H_k\|_k}, \Delta_k \right\}, \tag{2.6}$$

for some constant $\kappa_d \in (0, 1]$.

When the model is built with exact derivative information (e.g.- $g_k = \nabla f(x_k)$ and $H_k = \nabla^2 f(x_k)$), global convergence to second-order points is possible. It is also possible to use estimates of the function's Hessian and still guarantee convergence. Useful results in this area are given comprehensive treatment in [4]. In the derivative-free setting, other models must be constructed.

2.1. Derivative-Free Trust-Region Models. The quadratic model in (2.5) is attractive because with it, the subproblem in (2.2) is one of the only nonlinear programs for which *global* solutions can be efficiently computed. One extension to the derivative-free setting is to estimate the gradient $\nabla f(x_k)$ by finite difference methods using n additional function evaluations and apply classical derivative-based techniques. However, since finite difference evaluations are only useful for estimating derivatives at the current center, x_k, this approach is often impractical when the function f is computationally expensive.

An alternative approach is to obtain the model parameters g_k and H_k by requiring that the model interpolate the function at a set of distinct data points $\mathcal{Y} = \{y_1 = 0, y_2, \ldots, y_{|\mathcal{Y}|}\} \subset \mathbb{R}^n$:

$$m_k(x_k + y_i) = f(x_k + y_i) \qquad \text{for all } y_i \in \mathcal{Y}. \tag{2.7}$$

n	10	20	30	40	50	60	70	80	90	100
$\frac{(n+1)(n+2)}{2}$	66	231	496	861	1326	1891	2556	3321	4186	5151

TABLE 2.1
Number of Interpolation Points Needed to Uniquely Define a Full Quadratic Model.

The idea of forming quadratic models by interpolation for optimization without derivatives was proposed by Winfield in the late 1960's [23] and revived in the mid 1990's independently by Powell [14] and Conn, Scheinberg, and Toint [5].

These methods rely heavily on results from multivariate interpolation, a problem much more difficult than its univariate counterpart [21]. In particular, since the dimension of quadratics in $\mathbb{R}^n$ is $\hat{p} = \frac{1}{2}(n+1)(n+2)$, at least $\hat{p}$ function evaluations must be done to provide enough interpolation points to ensure uniqueness of the quadratic model. Further, these points must satisfy strict geometric conditions for the interpolation problem in (2.7) to be well-posed. These geometric conditions have received recent treatment in [6], where Taylor-like error bounds between the polynomial models and the true function were proposed. A quadratic model interpolating 6 points in $\mathbb{R}^2$ is shown in the lower left corner of Figure 2.1.

A significant drawback of these full quadratic methods is that the number of interpolation points they require is quadratic in the dimension of the problem. For example, we see in Table 2.1 that when $n = 30$, nearly 500 function evaluations are required before the first surrogate model can be constructed and the subproblem optimization can begin. In contrast, finite difference estimates of the gradient can be obtained in n function evaluations and hence $\frac{n}{2}$ iterations of a classical first-order methods could be run in the same time required to form the first quadratic model.

Before proceeding, we note that Powell has addressed this difficulty by proposing to satisfy (2.7) uniquely by certain underdetermined quadratics [15]. He developed NEWOA, a complex but computationally efficient Fortran code using underdetermined quadratic updates [16].

2.2. Fully Linear Models. For the reasons mentioned, we will rely on a class of so-called *fully linear* interpolation models, which can be formed using as few as $n+1$ function evaluations. To establish Taylor-like error bounds, the function f must be reasonably smooth. Throughout the sequel we will make the following assumptions on the function f:

(Assumption on Function) $f \in C^1[\Omega]$ for some open $\Omega \supset \mathcal{L}(x_0)$, ∇f is Lipschitz continuous on $\mathcal{L}(x_0)$, and f is bounded on $\mathcal{L}(x_0)$.

We borrow the following definition from [6] and note that three similar conditions define *fully quadratic* models.

DEFINITION 2.1. *For fixed $\kappa_f > 0, \kappa_g > 0$, x_k such that $f(x_k) \leq f(x_0)$, and $\Delta \in (0, \Delta_{\max}]$ defining $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_k\|_k \leq \Delta\}$, a model $m \in C^1[\Omega]$ is said to be fully linear on $\mathcal{B}$ if for all $x \in \mathcal{B}$:*

$$|f(x) - m(x)| \leq \kappa_f \Delta^2, \tag{2.8}$$

$$\|\nabla f(x) - \nabla m(x)\| \leq \kappa_g \Delta. \tag{2.9}$$

If a fully linear model can be obtained for any $\Delta \in (0, \Delta_{\max}]$, these conditions ensure that an approximation to even the true function's gradient can achieve any desired degree of precision within a small enough neighborhood of x_k. As exemplified in [6], fully linear interpolation models are defined by geometry conditions on the interpolation set. We will explore these conditions for the radial basis function models of the next section in Section 4.

3. Radial Basis Functions. Quadratic surrogates of the form (2.5) have the benefit of being easy to implement while still being able to model curvature of the underlying function f. Another way to model curvature is to consider interpolating surrogates, which are linear combinations of nonlinear basis functions and satisfy (2.7) for the interpolation points $\{y_j\}_{j=1}^{|\mathcal{Y}|}$. One possible model is of the form:

$$m_k(x_k+s) = \sum_{j=1}^{|\mathcal{Y}|} \lambda_j \phi(\|s-y_j\|) + p(s), \tag{3.1}$$

where $\phi : \mathbb{R}_+ \to \mathbb{R}$ is a univariate function and $p \in \mathcal{P}_{d-1}^n$, where $\mathcal{P}_{d-1}^n$ is the (trivial if $d=0$) space of polynomials in n variables of total degree no more than $d-1$.

Such models are called *radial basis functions* (RBFs) because $m_k(x_k+s)-p(s)$ is a linear combination of shifts of the function $\phi(\|x\|)$, which is constant on spheres in $\mathbb{R}^n$. For concreteness, we represent the polynomial tail by $p(s) = \sum_{i=1}^{\hat{p}} \nu_i \pi_i(s)$, for $\hat{p} = \dim\mathcal{P}_{d-1}^n$ and $\{\pi_1(s), \ldots, \pi_{\hat{p}}(s)\}$, a basis for $\mathcal{P}_{d-1}^n$. Some examples of popular radial functions are given in Table 3.1.

For fixed coefficients λ, these radial functions are all twice continuously differentiable. We briefly note that for an RBF model to be twice continuously differentiable, the radial function ϕ must be both twice continuously differentiable and have a derivative that vanishes at the origin. We then have relatively simple analytic expressions for both the gradient,

$$\nabla m_k(x_k+s) = \sum_{i=1}^{|\mathcal{Y}|} \lambda_i \phi'(\|s-y_i\|)\frac{s-y_i}{\|s-y_i\|} + \nabla p(s), \tag{3.2}$$

and Hessian of the model.

In addition to being sufficiently smooth, these radial functions in Table 3.1 all share the property of *conditional positive definiteness* [21].

DEFINITION 3.1. *Let π be a basis for $\mathcal{P}_{d-1}^n$, with the convention that $\pi = \emptyset$ if $d=0$. A function ϕ is said to be conditionally positive definite (cpd) of order d if for all sets of distinct points $\mathcal{Y} \subset \mathbb{R}^n$ and all $\lambda \neq 0$ satisfying $\sum_{j=1}^{|\mathcal{Y}|} \lambda_j \pi(y_j) = 0$, the quadratic form $\sum_{i,j=1}^{|\mathcal{Y}|} \lambda_j \phi(\|y_i - y_j\|)\lambda_j$ is positive.*

This property ensures that there exists a unique model of the form (3.1) provided that $\hat{p}$ points in $\mathcal{Y}$ are poised for interpolation in $\mathcal{P}_{d-1}^n$. Conditional positive definiteness is usually proved by Fourier transforms [3, 21] and is beyond the scope of the present work. Before addressing solution techniques, we note that if ϕ is cpd of order d, then it is also cpd of order $\hat{d} \geq d$.

3.1. Obtaining Model Parameters. We now illustrate one method for obtaining the parameters defining an RBF model that interpolates data as in (2.7) at knots in $\mathcal{Y}$. Defining the matrices $\Pi \in \mathbb{R}^{\hat{p}\times|\mathcal{Y}|}$ and $\Phi \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|}$, as $\Pi_{i,j} = \pi_i(y_j)$ and $\Phi_{i,j} = \phi(\|y_i - y_j\|)$, respectively, we consider the symmetric linear system:

$$\begin{bmatrix} \Phi & \Pi^T \\ \Pi & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \nu \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}. \tag{3.3}$$

Since $\{\pi_j(s)\}_{j=1}^{\hat{p}}$ forms a basis for $\mathcal{P}_{d-1}^n$, the interpolation set $\mathcal{Y}$ being poised for interpolation in $\mathcal{P}_d^n$ is equivalent to $\text{rank}(\Pi) = \dim\mathcal{P}_{d-1}^n = \hat{p}$. It is then easy to see that for cpd functions of order d, a sufficient condition for the nonsingularity of (3.3) is

that the points in $\mathcal{Y}$ are distinct and yield a Π^T of full column rank. It is instructive to note that, as in polynomial interpolation, these are *geometric* conditions on the interpolation nodes and are independent of the data values in f.

We will exploit this property of RBFs by using a null-space method (see for example [1]) for solving the saddle point problem in (3.3). Suppose that Π^T is of full column rank and admits the truncated QR factorization $\Pi^T = QR$ and hence $R \in \mathbb{R}^{(n+1)\times(n+1)}$ is nonsingular. By the lower set of equations in (3.3) we must have $\lambda = Z\omega$ for $\omega \in \mathbb{R}^{|\mathcal{Y}|-n-1}$ and any orthogonal basis Z for $\mathcal{N}(\Pi^T)$ (e.g.- from the orthogonal columns of a full QR decomposition) . Hence (3.3) reduces to:

$$Z^T \Phi Z \omega = Z^T f \tag{3.4}$$

$$R\nu = Q^T(f - \Phi Z\omega). \tag{3.5}$$

By the rank condition on Π^T and the distinctness of the points in $\mathcal{Y}$, $Z^T\Phi Z$ is positive definite for any ϕ that is cpd of at most order d. Hence the matrix that determines the RBF coefficients λ admits the Cholesky factorization:

$$Z^T \Phi Z = LL^T \tag{3.6}$$

for a nonsingular lower triangular L. Since Z is orthogonal we immediately note the bound:

$$\|\lambda\| = \left\|ZL^{-T}L^{-1}Z^T f\right\| \leq \left\|L^{-1}\right\|^2 \|f\|, \tag{3.7}$$

which will prove useful for the analysis in Section 4.2.

3.2. RBFs for Optimization. Although the idea of interpolation by RBFs has been around for more than 20 years, such methods have only recently gained popularity in practice [3]. Their use to date has been mainly confined to global optimization [2, 8, 17]. The success of RBFs in global optimization can be attributed to the ability of RBFs to model multimodal behavior while still exhibiting favorable numerical properties. An RBF model interpolating 6 points within an ∞-norm region in $\mathbb{R}^2$ is shown in the lower right of Figure 2.1. We note a particular benefit of RBF models in lower dimensions is that they can (uniquely) interpolate more than the $\frac{(n+1)(n+2)}{2}$ limit of quadratic models.

As part of his 2005 dissertation, Oeuvray developed a derivative-free trust-region algorithm employing a cubic RBF model with a linear tail. His algorithm, BOOSTERS, was motivated by problems in the area of medical image registration and was subsequently modified to include gradient information when available [13]. Convergence theory was borrowed from the literature available at the time [4].

$\phi(r)$	Order	Parameters	Example
r^β	2	$\beta \in (2,4)$	Cubic, r^3
$(\gamma^2 + r^2)^\beta$	2	$\gamma > 0, \beta \in (1,2)$	Multiquadric I, $(\gamma^2 + r^2)^{\frac{3}{2}}$
$-(\gamma^2 + r^2)^\beta$	1	$\gamma > 0, \beta \in (0,1)$	Multiquadric II, $-\sqrt{\gamma^2 + r^2}$
$(\gamma^2 + r^2)^{-\beta}$	0	$\gamma > 0, \beta > 0$	Inv. Multiquadric, $\frac{1}{\sqrt{\gamma^2+r^2}}$
$e^{-\frac{r^2}{\gamma^2}}$	0	$\gamma > 0$	Gaussian, $e^{-\frac{r^2}{\gamma^2}}$

TABLE 3.1
Popular Twice Continuously Differentiable RBFs & Order of Conditional Positive Definiteness.

Step 1: Find $n+1$ affinely independent points:
AffPoints($\mathcal{D}_k, \theta_0, \theta_1, \Delta_k$) (detailed in Figure 4.2)
Step 2: Add up to $p_{\max} - n - 1$ additional points to $\mathcal{Y}$:
AddPoints($\mathcal{D}_k$,θ_2,$p_{\max}$)
Step 3: Obtain RBF model parameters from (3.4) and (3.5).
Step 4: While $\|\nabla m_k(x_k)\| \leq \frac{\epsilon_g}{2}$:
If m_k is fully linear in $\mathcal{B}_k^g = \{x \in \mathbb{R}^n : \|x_k - x\|_k \leq (2\kappa_g)^{-1}\epsilon_g\}$,
Return.
Else,
Obtain a model m_k that is fully linear in $\mathcal{B}_k^g$,
Set $\Delta_k = \frac{\epsilon_g}{2\kappa_g}$.
Step 5: Approximately solve subproblem (2.2) to obtain a step s_k satisfying (4.18),
Evaluate $f(x_k + s_k)$.
Step 6: Update trust-region parameters:

$$\Delta_{k+1} = \begin{cases} \min\{\gamma_1 \Delta_k, \Delta_{\max}\} & \text{if } \rho_k \geq \eta_1 \\ \Delta_k & \text{if } \rho_k < \eta_1 \text{ and } m_k \text{ is not fully linear on } \mathcal{B}_k \\ \gamma_0 \Delta_k & \text{if } \rho_k < \eta_1 \text{ and } m_k \text{ is fully linear on } \mathcal{B}_k \end{cases}$$

$$x_{k+1} = \begin{cases} x_k + s_k & \text{if } \rho_k \geq \eta_1 \\ x_k + s_k & \text{if } \rho_k > \eta_0 \text{ and } m_k \text{ is fully linear on } \mathcal{B}_k \\ x_k & \text{else} \end{cases}$$

Step 7: Evaluate a model-improving point if $\rho_k \leq \eta_0$ and m_k is not fully linear on $\mathcal{B}_k$.

FIG. 4.1. *Iteration k of the ORBIT Algorithm.*

4. The ORBIT Algorithm. In this section we detail our algorithm, "ORBIT", and establish several of the computational techniques employed. Given trust-region inputs $0 \leq \eta_0 \leq \eta_1 < 1$, $0 < \gamma_0 < 1 < \gamma_1$, $0 < \Delta_0 \leq \Delta_{\max}$, and $x_0 \in \mathbb{R}^n$, and additional inputs $0 < \theta_1 \leq \theta_0^{-1} \leq 1$, $\theta_2 > 0$, $\kappa_f, \kappa_g, \epsilon_g > 0$, and $p_{\max} > n+1$, an outline of the kth iteration of the algorithm is provided in Figure 4.1.

Besides the current trust-region center and radius, the algorithm works with a set of displacements, $\mathcal{D}_k$, from the current center x_k. This set consists of all points at which the true function value is known:

$$d_i \in \mathcal{D}_k \iff f(x_k + d_i) \text{ is known.} \tag{4.1}$$

Since evaluation of f is computationally expensive, we stress the importance of having complete memory of all points previously evaluated by the algorithm. This is a fundamental difference between ORBIT and previous algorithms in [5], [13], [14] and [16], where, in order to reduce linear algebraic costs, the interpolation set was allowed to change by at most one point and hence a very limited memory was required.

The model m_k at iteration k will employ an interpolation subset $\mathcal{Y} \subseteq \mathcal{D}_k$ of the available points. In **Step 1** (Figure 4.1), points are selected for inclusion in $\mathcal{Y}$ in order to establish (if possible) a model which is fully linear within a neighborhood of the current trust-region as discussed in Section 4.1. Additional points are added to $\mathcal{Y}$ in **Step 2** (discussed in Section 4.2) in a manner which ensures that the model

parameters, and hence the first two derivatives of the model, remain bounded. A well-conditioned RBF model, interpolating at most $p_{\max}$ points, is then fit in **Step 3** using the previously discussed solution techniques.

In **Step 4** a termination criteria is checked. If the model gradient is small enough, the method detailed in Section 4.1 is used to evaluate at additional points until the model is valid within a small neighborhood, $\mathcal{B}_k^g = \{x \in \mathbb{R}^n : \|x_k - x\|_k \leq (2\kappa_g)^{-1}\epsilon_g\}$, of the current iterate. The size of this neighborhood is chosen such that if m_k is fully linear on $\mathcal{B}_k^g$ and the gradient is sufficiently small then by (2.9):

$$\|\nabla f(x_k)\| \leq \|\nabla m_k(x_k)\| + \|\nabla f(x_k) - \nabla m_k(x_k)\| \leq \frac{\epsilon_g}{2} + \kappa_g \left(\frac{\epsilon_g}{2\kappa_g}\right) = \epsilon_g, \tag{4.2}$$

gives a bound for the true gradient at x_k when the algorithm is exited. While setting ambitious values for κ_g and ϵ_g ensure that the computational budget is exhausted, it may be advantageous (e.g.- in noisy or global optimization problems) to use the remaining budget by restarting this local procedure elsewhere (possibly reusing some previously obtained function evaluations).

Given that the model gradient is not too small, an approximate solution to the trust-region subproblem is computed in **Step 5** as discussed in Section 4.3. In **Step 6**, the trust-region parameters are updated. The given procedure only coincides with the derivative-based procedure in Figure 2.2 when the subproblem solution makes significant progress ($\rho_k \geq \eta_1$). In all other cases, the trust-region parameters will remain unchanged if the model is not fully linear on $\mathcal{B}_k$. If the model is not fully linear, the function is evaluated at an additional, so-called *model-improving point* in **Step 7** to ensure that the model is at least one step closer to being fully linear on $\mathcal{B}_{k+1} = \mathcal{B}_k$.

We now provide additional computational details where necessary.

4.1. Fully Linear RBF Models. As previously emphasized, the number of function evaluations required to obtain a set of points poised for quadratic interpolation is computationally unattractive for a wide range of problems. For this reason, we limit ourselves to twice continuously differentiable RBF models of the form (3.1) where $p \in \mathcal{P}_1^n$ is linear and hence ϕ must be cpd of order 2 or less. Further, we will always enforce interpolation at the current iterate x_k so that $y_1 = 0 \in \mathcal{Y}$. We will employ the standard linear basis and permute the points so that:

$$\Pi = \begin{bmatrix} y_2 & \cdots & y_{|\mathcal{Y}|} & 0 \\ 1 & \cdots & 1 & 1 \end{bmatrix} = \begin{bmatrix} Y & 0 \\ e^T & 1 \end{bmatrix}, \tag{4.3}$$

where e is the vector of ones and Y denotes the matrix of nonzero points in $\mathcal{Y}$.

The following Lemma is a generalization of similar Taylor-like error bounds found in [6] and is proved in [22].

LEMMA 4.1. *Suppose that f and m are continuously differentiable in $\mathcal{B} = \{x : \|x - x_k\|_k \leq \Delta\}$ and that ∇f and ∇m are Lipschitz continuous in $\mathcal{B}$ with Lipschitz constants γ_f and γ_m, respectively. Further suppose that m satisfies the interpolation conditions in (2.7) at a set of points $\mathcal{Y} = \{y_1 = 0, y_2, \ldots, y_{n+1}\} \subseteq \mathcal{B} - x_k$ such that $\left\|Y^{-1}\right\| \leq \frac{\Lambda_Y}{c_k \Delta}$. Then for any $x \in \mathcal{B}$:*

- $|m(x) - f(x)| \leq \sqrt{n} c_k^2 (\gamma_f + \gamma_m) \left(\frac{5}{2}\Lambda_Y + \frac{1}{2}\right) \Delta^2$, *and*
- $\|\nabla m(x) - \nabla f(x)\| \leq \frac{5}{2}\sqrt{n}\Lambda_Y c_k (\gamma_f + \gamma_m) \Delta$.

We note that Lemma 4.1 applies to many models in addition to the RBFs considered here. In particular, it says that if a model with a Lipschitz continuous gradient

interpolates a function on a sufficiently affinely independent set of points, there exist constants $\kappa_f, \kappa_g > 0$ independent of Δ such that conditions (2.8) and (2.9) are satisfied and hence m is fully linear on $\mathcal{B}$.

It remains to show that $n+1$ points in $\mathcal{B} - x_k$ can be efficiently obtained such that the norm of Y^{-1} can be bounded by a quantity of the form $\frac{\Lambda_Y}{c_k \Delta}$. In ORBIT, we ensure this by working with a QR factorization of the normalized points as justified in the following lemma.

LEMMA 4.2. *If all QR pivots of $\frac{1}{c_k\Delta}Y$ satisfy $|r_{ii}| \geq \theta_1 > 0$, then $\left\|Y^{-1}\right\| \leq \frac{n^{\frac{n-1}{2}}\theta_1^{-n}}{c_k\Delta}$.*

Proof. If $\{y_2, \ldots, y_{n+1}\} \subseteq \mathcal{B} - x_k$, all columns of the normalized matrix $\hat{Y} = \frac{1}{c_k\Delta}Y$ satisfy $\left\|\hat{Y}_j\right\| \leq 1$. Letting $QR = \hat{Y}$ denote a QR factorization of the matrix $\hat{Y}$, and $0 \leq \sigma_n \leq \cdots \leq \sigma_1 \leq \sqrt{n}$ denote the ordered singular values of $\hat{Y}$, we have

$$\sigma_n \sigma_1^{n-1} \geq \prod_{i=1}^{n} \sigma_i = |\det(\hat{Y})| = |\det(R)| = \prod_{i=1}^{n} |r_{ii}|. \tag{4.4}$$

If each of the QR pivots satisfy $|r_{ii}| \geq \theta_1 > 0$, we have the admittedly crude bound:

$$\left\|Y^{-1}\right\| = \frac{1}{c_k\Delta}\left\|\hat{Y}^{-1}\right\| = \frac{1}{c_k\Delta}\frac{1}{\sigma_n} \leq \frac{1}{c_k\Delta}\frac{n^{\frac{n-1}{2}}}{\theta_1^n}. \tag{4.5}$$

∎

While other bounds based on the size of the QR pivots are possible, we note that the one above does not rely on pivoting strategies. Pivoting may limit the number of recently sampled points that can be included in the interpolation set, particularly since choosing points in $\mathcal{B}$ that are farther away from the current iterate may prevent subsequent pivots from being sufficiently large.

We note that if Δ in Lemmas 4.1 and 4.2 is chosen to be the current trust-region radius Δ_k, the design of the algorithm may mean that the current center, x_k, is the only point within $\mathcal{B}$ at which f has been evaluated. For this reason, we will look to make m_k fully linear within a slightly enlarged the region defined by $\{x : \|x - x_k\|_k \leq \theta_0 \Delta_k\}$ for a constant $\theta_0 \geq 1$. We note that this constant still ensures that the model is fully linear within the trust-region $\mathcal{B}_k$, provided that the constants κ_f and κ_g are suitably altered in Lemma 4.1.

The subroutine AffPoints given in Figure 4.2 details our method of constructing a model which is fully linear on $\mathcal{B}_k$. We note that the projections in **2.** and **3b.** are exactly the magnitude of the pivot that results from adding point d_j to $\mathcal{Y}$.

Because of the form of Y, it is straightforward to see that for any $\theta_1 \in (0, \theta_0^{-1}]$, an interpolation set $\mathcal{Y} \subseteq \mathcal{B} - x_k$ can be constructed such that all QR pivots satisfy $r_{ii} \geq \theta_1$. In particular, we may iteratively add points to $\mathcal{Y}$ corresponding to (scaled by Δ) points in the null space of the current $\mathcal{Y}$ matrix. Such points yield pivots of magnitude exactly θ_0^{-1}. We may further immediately deduce that for any x_k with $f(x_k) \leq f(x_0)$ and any $\Delta \in (0, \Delta_{\max}]$, the model in Lemma 4.1 can be made fully linear on $\mathcal{B}$ (for appropriately chosen $\kappa_f, \kappa_g > 0$) in at most $n+1$ function evaluations.

Recall from Section 3 that a unique RBF model may only be obtained provided that $\mathcal{Y}$ contains $n+1$ affinely independent points. For our solution for the RBF polynomial parameters in (3.5) to be numerically stable, the matrix Π^T must be well-conditioned. In particular we note that

$$\Pi^{-T} = \begin{bmatrix} Y^{-T} & -Y^{-T}e \\ 0 & 1 \end{bmatrix} \tag{4.6}$$

0. Input $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\} \subset \mathbb{R}^n$, constants $\theta_0 \geq 1$, $\theta_1 \in (0, \theta_0^{-1}]$, $\Delta \in (0, \Delta_{\max}]$.
1. Initialize $\mathcal{Y} = \{0\}$, $Z = I_n$.
2. For all $d_j \in \mathcal{D}$ such that $\|d_j\|_k \leq \theta_0 \Delta$:
  If $\left|\text{proj}_Z\left(\frac{1}{\theta_0 \Delta} d_j\right)\right| \geq \theta_1$,
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{d_j\}$,
    Update Z to be an orthonormal basis for $\mathcal{N}\left([y_1 \cdots y_{|\mathcal{Y}|}]\right)$.
3a. If $|\mathcal{Y}| = n + 1$, `linear=true`.
3b. If $|\mathcal{Y}| < n + 1$, `linear=false`,
  Save $z_i \in Z$ as a model-improving direction,
  For $d_j \in \mathcal{D}$ such that $\|d_j\|_k \leq 2\Delta_{\max}$:
    If $\left|\text{proj}_Z\left(\frac{1}{\theta_0 \Delta} d_j\right)\right| \geq \theta_1$,
      $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{d_j\}$,
      Update Z to be an orthonormal basis for $\mathcal{N}\left([y_1 \cdots y_{|\mathcal{Y}|}]\right)$.
  If $|\mathcal{Y}| < n + 1$, $\mathcal{Y}$ is not poised for linear interpolation,
    Evaluate $f(x_k + \Delta z_i)$ for all $z_i \in Z$,
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup Z$.

FIG. 4.2. *AffPoints($\mathcal{D}, \theta_0, \theta_1, \Delta$): Algorithm for Obtaining Fully Linear Models.*

and hence

$$\left\|\Pi^{-T}\right\| \leq \left\|Y^{-1}\right\| \sqrt{n+1} + 1 \tag{4.7}$$

provides an easily obtainable bound based on $\left\|Y^{-1}\right\|$. If desired, the vector e in the matrix Π can be scaled such that this bound is independent of the dimension.

In either case, if not enough points within the enlarged trust-region have been previously evaluated at, the model is not fully linear and additional points must be considered. By ensuring that these remaining points are within $2\Delta_{max}$ of the current center, we are again providing a bound on $\left\|Y^{-1}\right\|$. If we still are unable to find $n + 1$ points, we generate additional points to ensure that the RBF model is uniquely defined, confirming that at termination the procedure in Figure 4.2 yields an interpolation set of $n + 1$ points suitably poised for linear interpolation.

4.2. Adding Additional Points. We now assume that $\mathcal{Y}$ consists of $n + 1$ points that are sufficiently poised for linear interpolation. Given only these $n + 1$ points, $\lambda = 0$ is the unique solution to (3.3) and hence the RBF model in (3.1) is linear. In order to take advantage of the nonlinear modeling benefits of RBFs it is thus clear that additional points should be added to $\mathcal{Y}$. Note that by Lemma 4.1, adding these points will not affect the property of a model being fully linear.

We now detail ORBIT's method of adding additional model points to $\mathcal{Y}$ while maintaining bounds on the conditioning of the system (3.4). In [13] Oeuvray utilizes a different technique applied to the larger system in (3.3) with a similar goal. ORBIT's method largely follows the development in [2] and directly addresses the conditioning of the system used by our solution techniques.

Employing the notation of Section 3.1, we now consider what happens when $y \in \mathbb{R}^n$ is added to the interpolation set $\mathcal{Y}$. We denote the basis function and polynomial matrices obtained when this new point is added as Φ_y and Π_y^T, respectively:

$$\Phi_y = \begin{bmatrix} \Phi & \phi_y \\ \phi_y^T & \phi(0) \end{bmatrix}, \qquad \Pi_y^T = \begin{bmatrix} \Pi^T \\ \pi(y) \end{bmatrix}. \tag{4.8}$$

We begin by noting that by applying $n+1$ Givens rotations to the full QR factorization of Π^T, we obtain an orthogonal basis for $\mathcal{N}(\Pi_y^T)$ of the form:

$$Z_y = \begin{bmatrix} Z & Q\tilde{g} \\ 0 & \hat{g} \end{bmatrix}, \tag{4.9}$$

where, as in Section 3.1, Z is any orthogonal basis for $\mathcal{N}(\Pi^T)$. Hence, $Z_y^T \Phi Z_y$ is of the form:

$$Z_y^T \Phi Z_y = \begin{bmatrix} Z^T \Phi Z & v \\ v^T & \sigma \end{bmatrix}, \tag{4.10}$$

and it can easily be shown that:

$$L_y^T = \begin{bmatrix} L^T & L^{-1}v \\ 0 & \sqrt{\sigma - \|L^{-1}v\|^2} \end{bmatrix}, \qquad L_y^{-T} = \begin{bmatrix} L^{-T} & \frac{-L^{-T}L^{-1}v}{\sqrt{\sigma - \|L^{-1}v\|^2}} \\ 0 & \frac{1}{\sqrt{\sigma - \|L^{-1}v\|^2}} \end{bmatrix} \tag{4.11}$$

yields $L_y L_y^T = Z_y^T \Phi Z_y$. Careful algebra shows that:

$$v = Z^T (\Phi Q \tilde{g} + \phi_y \hat{g}) \tag{4.12}$$

$$\sigma = \tilde{g}^T Q^T \Phi Q \tilde{g} + 2\tilde{g}^T Q^T \phi_y \hat{g} + \phi(0)\hat{g}^2. \tag{4.13}$$

Assuming that both $\mathcal{Y}$ and the new point y belong to $\{x \in \mathbb{R}^n : \|x\|_k \leq 2\Delta_{\max}\}$, the quantities $\{\|x - z\| : x, y \in \mathcal{Y} \cup \{y\}\}$ are all of magnitude no more than $4c_k\Delta_{\max}$. Using the isometry of Z_y and $(Q\tilde{g}, \hat{g})$ we hence have the bound:

$$\|v\| \leq \sqrt{|\mathcal{Y}|(|\mathcal{Y}|+1)}\max\{|\phi(r)| : r \in [0, 4c_k\Delta_{\max}]\}. \tag{4.14}$$

Hence, provided that L^{-1} was previously well-conditioned, the resulting factors L_y^{-1} remain bounded provided that $\sqrt{\sigma - \|L^{-1}v\|^2}$ is bounded away from 0. Assuming that no more than $p_{\max} - n - 1$ points are considered for addition, induction gives a bound on the norm of the final L_Y^{-1}. Assuming that $\|f\|$ is bounded, this would immediately give the bound for λ in (3.7). This bound will be necessary in order to ensure that the RBF model Hessians remain bounded.

Recall that we have confined ourselves to only consider RBF models that are both twice continuously differentiable and have $\nabla^2 p \equiv 0$ for the polynomial tail $p(\cdot)$. For such models we have:

$$\nabla^2 m_k(x_k + s) = \sum_{i=1}^{|\mathcal{Y}|} \lambda_i \left[\frac{\phi'(\|z_i\|)}{\|z_i\|} I_n + \left(\phi''(\|z_i\|) - \frac{\phi'(\|z_i\|)}{\|z_i\|} \right) \frac{z_i}{\|z_i\|} \frac{z_i^T}{\|z_i\|} \right], \tag{4.15}$$

for $z_i = s - y_i$. When written in this way, we see that the magnitude of the model Hessian depends on the quantities $\left|\frac{\phi'(r)}{r}\right|$ and $|\phi''(r)|$. Of particular interest is the quantity:

$$b_2(\overline{\Delta}) = \max\left\{ 2\left|\frac{\phi'(r)}{r}\right| + |\phi''(r)| : r \in [0, \overline{\Delta}] \right\}, \tag{4.16}$$

which is again bounded whenever $\overline{\Delta}$ is for all of the radial functions considered in Table 3.1.

Initialize $s = -\frac{\nabla m_k(x_k)}{\|\nabla m_k(x_k)\|_k}\Delta_k$.
While $m_k(x_k) - m_k(x_k+s) < \frac{\kappa_d}{2}\|\nabla m_k(x_k)\| \min\left\{\frac{\|\nabla m_k(x_k)\|}{\kappa_H}, \frac{\|\nabla m_k(x_k)\|}{\|\nabla m_k(x_k)\|_k}\Delta_k\right\}$:
$s \leftarrow s\alpha$.

FIG. 4.3. *Backtracking Algorithm for Obtaining a Sufficient Decrease in* ***Step 5*** *of Figure 4.1* ($\alpha \in (0,1)$, $\kappa_d \in (0,1]$).

The following Lemma is a consequence of the preceding remarks and is proved formally in [22].

LEMMA 4.3. *Let* $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_k\|_k \le 2\Delta_{\max}\}$. *Let* $\mathcal{Y} \subset \mathcal{B} - x_k$ *be a set of distinct interpolation points,* $n+1$ *of which are affinely independent, and* $|f(x_k + y_i)| \le f_{\max}$ *for all* $y_i \in \mathcal{Y}$. *Then for a model of the form (3.1) interpolating* f *on* $x_k + \mathcal{Y}$, *we have that for all* $x \in \mathcal{B}$:

$$\left\|\nabla^2 m_k(x)\right\| \le |\mathcal{Y}| \left\|L^{-1}\right\|^2 b_2(4c_k\Delta_{\max}) f_{\max} =: \kappa_H. \tag{4.17}$$

Note that if $\sup_{x\in\mathcal{L}(x_0)} |f(x)| \le f_{\max}$, $\left\|\nabla^2 m_k(x)\right\|$ is bounded on $\mathbb{R}^n$ for all k. Since $m_k \in C^2$, it follows that ∇m is Lipschitz continuos and κ_H is a possible Lipschitz constant on $\mathcal{L}(x_0)$. Thus we have justifying the use of Lemma 4.1 for our RBF models.

4.3. Solving the Subproblem. The trust-region subproblem (2.2) is made considerably more difficult using the RBF model in (3.1). Given that the radial function ϕ is chosen from Table 3.1, the model will be twice continuously differentiable, and hence local optimization methods can employ the first- and second- order derivatives ∇m and $\nabla^2 m$ to solve (2.2). However, unlike the quadratic model, the global solution to this problem is no longer attainable in polynomial time.

In particular, since the RBF model may be multimodal, an optimal solution to (2.2), guaranteed to exist by continuity and compactness, would require the use of global optimization techniques. However, our solution is only required to satisfy a sufficient decrease condition similar to (2.6) for some fixed $\kappa_d \in (0,1]$:

$$m_k(x_k) - m_k(x_k+s) \ge \frac{\kappa_d}{2}\|\nabla m_k(x_k)\| \min\left\{\frac{\|\nabla m_k(x_k)\|}{\kappa_H}, \frac{\|\nabla m_k(x_k)\|}{\|\nabla m_k(x_k)\|_k}\Delta_k\right\}. \tag{4.18}$$

Figure 4.3 gives a simple algorithm for backtracking line search in the direction of steepest descent. Since subproblem solutions are only calculated in Figure 4.1 if $\|\nabla m_k(x_k)\| \ge \frac{\epsilon_g}{2} > 0$, an easy consequence of the differentiability of m_k guarantees that there are at most $\max\left\{\log_\alpha \frac{2\Delta_k\kappa_H}{\epsilon_g}, 0\right\}$ iterations of the backtracking line search.

Further, since the objective function is expensive to evaluate, additional more-sophisticated methods can be employed in the optimization between function evaluations. In particular, derivative-based constrained local optimization methods can be initiated from the solution, $\hat{s}$, of the backtracking line search as well as other points in $\mathcal{B}_k$. Any resulting point, $\tilde{s}$, can then be chosen as the approximate solution to the subproblem provided that $m_k(x_k + \tilde{s}) \le m_k(x_k + \hat{s})$.

The sufficient decrease condition in (4.18) guarantees that we can efficiently obtain an approximate solution to the trust-region subproblem. Further, it allows us to establish the global convergence in [22] of ORBIT to first-order critical points satisfying $\nabla f(x_*) = 0$.

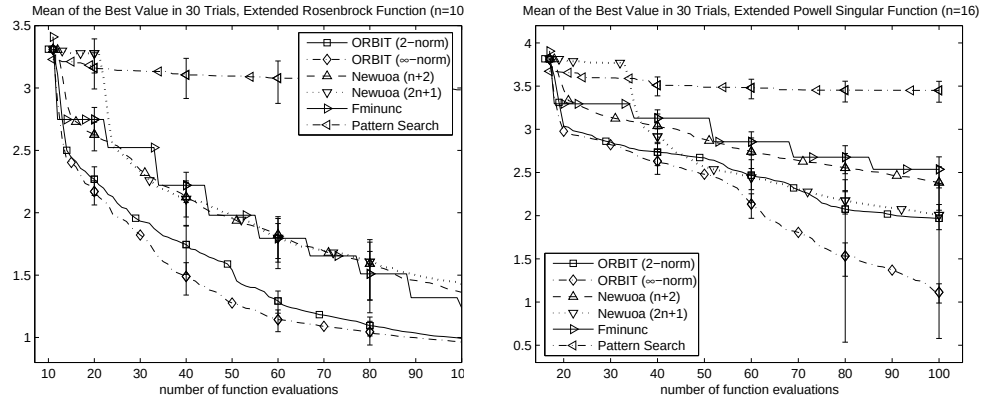

FIG. 5.1. *Average of 30 Starting Points on Two Test Problems ($\log_{10}$ scale, lowest line is best).*

5. Testing Algorithms for Optimization of Computationally Expensive Functions. A user ideally seeks an algorithm whose best function value is smaller than alternative algorithms, regardless of the number of function evaluations available. Since this will not be possible for all functions, we seek an algorithm that performs better than alternatives (given a fixed number of evaluations) on as large a class of problems as possible. Examples in the literature of systematic testing of algorithms for computationally expensive optimization are infrequent. In [17], Regis and Shoemaker develop plots like those shown in Figure 5.1, while in [13], Oeuvray and Bierlaire track the number of function evaluations needed to reach some convergence goal.

Using 30 different starting points, Figure 5.1 shows the mean and 95% pointwise confidence intervals for the minimum function value obtained as a function of the number of evaluations performed. Such plots are useful for determining the number of evaluations needed to obtain some desired function value and for providing insight into an algorithm's average progress. However, by grouping all starting points together, we are unable to determine the relative success of algorithms within the same starting point. In Section 5.2 we discuss performance plots in effort to complement the means plots in Figure 5.1. We first summarize the alternative algorithms considered here.

5.1. Alternative Algorithms. We compare ORBIT to a number of competitive algorithms for derivative-free optimization.

Pattern search [10] is a direct search optimization method widely used in practice. The function is systematically sampled at points on a space-filling *pattern*, scaled much the same way as a trust-region. We use the implementation of pattern search available in the MATLAB Genetic Algorithm and Direct Search Toolbox [18].

To compare with a derivative-based approach we use a quasi-Newton method where the derivatives are approximated by finite differences. We use the FMINUNC routine from the MATLAB Optimization Toolbox [19], where iterates are generated by running Newton's method using an approximate Hessian obtained with BFGS updates.

NEWUOA [15, 16] is a derivative-free trust-region method employing an underdetermined quadratic model that interpolates f at $p \in \{n+2, \ldots, \frac{1}{2}(n+1)(n+2)\}$ points, the value $p = 2n+1$ recommended by Powell for computational efficiency. In each iteration, one interpolation point is altered and the model m is updated so that the norm of the resulting change in $\nabla^2 m$ is minimized. We use Powell's Fortran NEWUOA code with $p = 2n+1$ interpolation points as well as the minimum $p = n+2$, a strategy which may work well in the initial stages of the optimization.

5.2. Performance Profiles. In [7], Dolan and Moré develop a procedure for visualizing the relative success of solvers on a set of benchmark problems. Their *performance profiles*, now widely used by the optimization community, are defined by three characteristics: a set of benchmark problems $\mathcal{P}$, a convergence test $\mathcal{T}$, and a set of algorithms/solvers $\mathcal{S}$. Based on the convergence test, a performance metric $t_{p,s}$ to be minimized (e.g.- the amount of computing time required to meet some termination criteria) is obtained for each $(p, s) \in \mathcal{P} \times \mathcal{S}$. For a pair (p, s), the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}} \tag{5.1}$$

defines the success of an algorithm relative to the other algorithms in $\mathcal{S}$. The best algorithm for a particular problem attains the lower bound $r_{p,s} = 1$, while $r_{p,s} = \infty$ if an algorithm fails to meet the convergence test. For algorithm s, the fraction of problems where the performance ratio is at most τ is:

$$\rho_s(\tau) = \frac{1}{|\mathcal{P}|}\text{size}\Big\{p \in \mathcal{P} : r_{p,s} \leq \tau\Big\}. \tag{5.2}$$

The performance profile $\rho_s(\tau)$ is a probability distribution function capturing the probability that the performance ratio for s is within a factor τ of the best possible ratio. Conclusions based on $\rho_s(\tau)$ should only be extended to other problems, convergence tests, and algorithms similar to those in $\mathcal{P}, \mathcal{T}$, and $\mathcal{S}$.

Extensions to the computationally expensive setting have recently been examined in [12]. Since classical forms of convergence cannot be expected, the best measure of an algorithm's performance is the minimum function value obtained within a given computational budget. We let $\{x_{i,p,s}\}_{i=0}^{\hat{k}}$ denote the sequence of points at which function f_p is evaluated by algorithm s, so that $f_p(x_{i,p,s})$ is the ith function evaluation performed by s. Thus

$$F_{p,s}(k) = \min\{f_p(x_{i,p,s}) : i < k\} \tag{5.3}$$

denotes the minimum function value obtained by s in the first k evaluations of f_p.

We assume that any computations done by an algorithm except evaluation of the function are negligible and that the time required to evaluate a function is the same at any point in the domain of interest. While other options are addressed in [12], here we seek the algorithm which obtains the largest decrease in the first k_p evaluations. Hence we employ the performance metric:

$$t_{p,s} = \frac{1}{F_{p,s}(1) - F_{p,s}(k_p)}, \tag{5.4}$$

with the implicit assumption that for fixed $p \in \mathcal{P}$, $x_{0,p,s_1} = x_{0,p,s_1}$ for all $(s_1, s_2) \in \mathcal{S}^2$ and hence $F_{p,s}(1)$ is the same for all $s \in \mathcal{S}$. The performance ratio is of the form:

$$r_{p,s} = \frac{\max_s(F_{p,s}(1) - F_{p,s}(k_p))}{F_{p,s}(1) - F_{p,s}(k_p)}. \tag{5.5}$$

5.3. Test Problems. We employ a subset of eight functions of varying dimensions from the Moré-Garbow-Hillstrom (MGH) set of test functions for unconstrained optimization [11]: Powell Singular ($n = 4$), Wood ($n = 4$), Trigonometric ($n = 5$), Discrete Boundary Value ($n = 8$), Extended Rosenbrock ($n = 10$), Variably Dimensioned

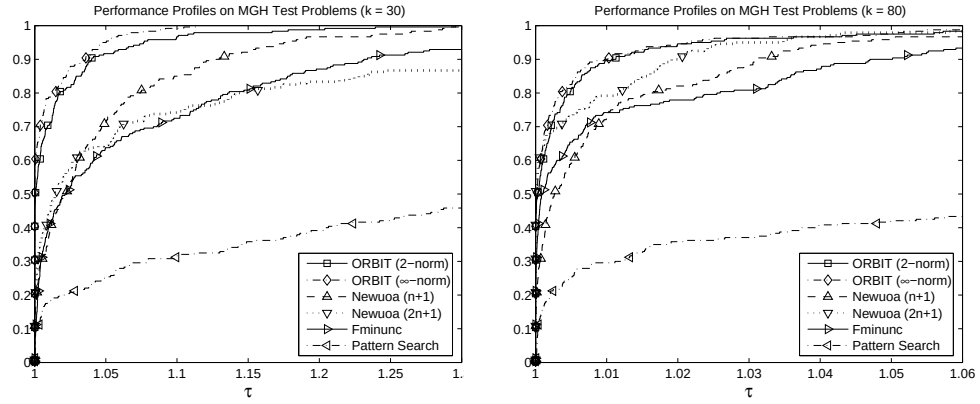

FIG. 5.2. *Performance Profile $\rho_s(\tau)$ on Set of 240 Test Problems After: (a) 30 Evaluations, (b) 80 Evaluations (highest line is best).*

($n = 10$), Broyden Tridiagonal ($n = 11$), and Extended Powell Singular ($n = 16$). For each function, we generate 30 random starting points and hence have 30 different "problems" for each function, yielding a total of 240 test problems. Throughout the sequel, we examine performance profiles after a fixed number of evaluations for each problem so that $k_p = k$ for all $p \in \mathcal{P}$ denotes the number of evaluations used.

5.4. Test Problem Results. The mean $F_{p,s}(k)$ trajectory over the 30 starting points on the Extended Rosenbrock Function is shown in Figure 5.1 (a). Note that when using performance plots, we avoid grouping problems across different starting points. To ensure fair comparison among the different algorithms, we use the same starting point and same initial trust-region radius/pattern size in all algorithms.

We implement ORBIT using a cubic RBF model with both 2-norm and ∞-norm trust-regions, and compare them with the four alternative algorithms on each of the 240 resulting test problems. For all experiments we used the ORBIT (Figure 4.1) parameters: $\eta_0 = 0$, $\eta_1 = .2$, $\gamma_0 = \frac{1}{2}$, $\gamma_1 = 2$, $\Delta_{\max} = 10^3\Delta_0$, $\theta_0 = 2$ $\theta_1 = 10^{-3}$, $\theta_2 = 10^{-7}$, $\epsilon_g = 10^{-4}$, and $p_{\max} = 2n + 1$, with the starting parameters (x_0, Δ_0) varying from problem to problem. The parameters κ_f and κ_g for checking whether a model is fully linear are inferred from $\theta_0 = 2$ and $\theta_1 = 10^{-3}$ as discussed in Section 4.1. For the backtracking line search algorithm in Figure 4.3, we set $\kappa_d = 10^{-4}$ and $\alpha = .9$.

In Figure 5.2 (a) we show the performance profile in (5.2) after 30 function evaluations. We note that for any problem of dimension $n > 6$, 30 evaluations is insufficient for forming even a single full quadratic model. Based on this plot, we see that the ∞-norm and 2-norm variants of ORBIT were the best algorithm for 42.1% and 31.7% of the 240 problems, respectively. Further, these variants acheived a decrease within a factor 1.1 of the best decrease in 99.2% and 96.7% of the 240 problems, respectively. These profiles illustrate the success of ORBIT particularly when very few function evaluations are available.

In Figure 5.2 (b) we see that after 80 function evaluations the ∞-norm and 2-norm variants of ORBIT were the best algorithm 30.4% and 28.3% of the time, respectively, while the $2n + 1$ and $n + 2$ variants of NEWUOA were the best algorithm 27.9% and 6.7% of the time, respectively. We believe this is because after 80 evaluations, some of the algorithms are focusing on smaller neighborhoods near the optimal solution, where a quadratic approximation is particularly appropriate.

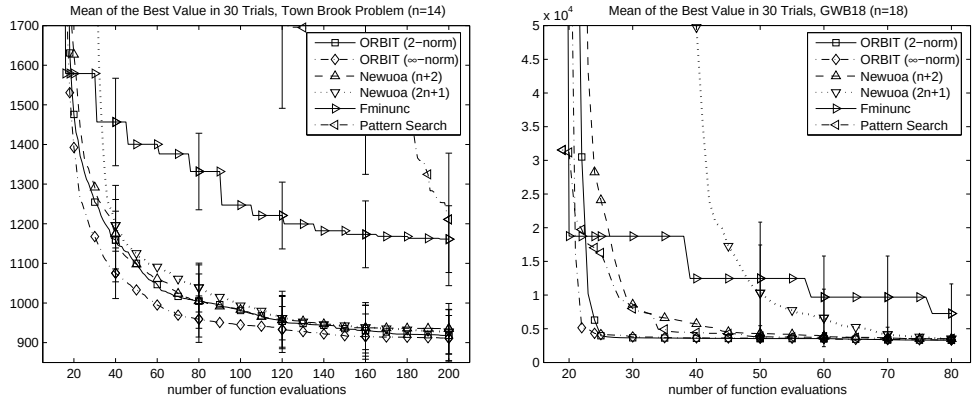

FIG. 6.1. *Mean Best Function Value (30 trials): (a) Town Brook Problem, (b) GWB18 Problem.*

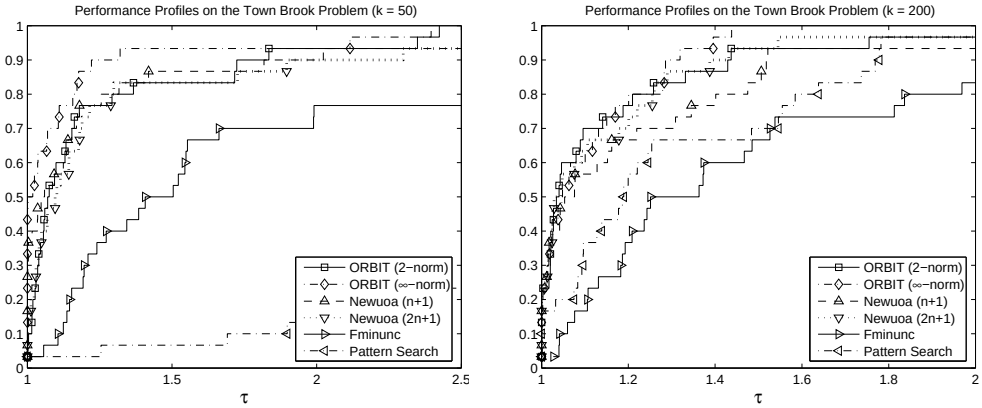

FIG. 6.2. *$\rho_s(\tau)$ on Town Brook Problem After: (a) 50 evaluations, (b) 200 evaluations.*

6. Environmental Applications. Our motivation for developing ORBIT is optimization of problems in Environmental Engineering relying on complex numerical simulations of physical phenomena. In this section we consider two such applications. As is often the case in practice, both simulators are constrained blackbox functions. In the first problem, the constraints can only be checked after the simulation has been carried out, while in the second, simple box constraints are present. We will treat both of these problems as unconstrained by adding a smooth penalty term.

The problems presented here are computationally less expensive (a smaller watershed is employed in the first problem while a coarse grid of a groundwater problem is used in the second) of actual problems. As a result, both simulations require less than 6 seconds on a 2.4 GHz Pentium 4 desktop. This practical simplification allows us to test a variety of optimization algorithms at 30 different starting points while keeping both examples representative of the type of functions used in more complex watershed calibration and groundwater bioremediation problems.

6.1. Calibration of a Watershed Simulation Model. The Cannonsville Reservoir in upstate New York provides drinking water to New York City (NYC). Phosphorous loads from the watershed into the reservoir are monitored carefully because of concerns about *eutrophication*, a form of pollution that can cause severe water quality problems. In particular, phosphorous promotes the growth of algae, which then clogs the water supply. Currently, NYC has no filtration plant for the drinking water from its reservoirs in upstate New York. If phosphorous levels become

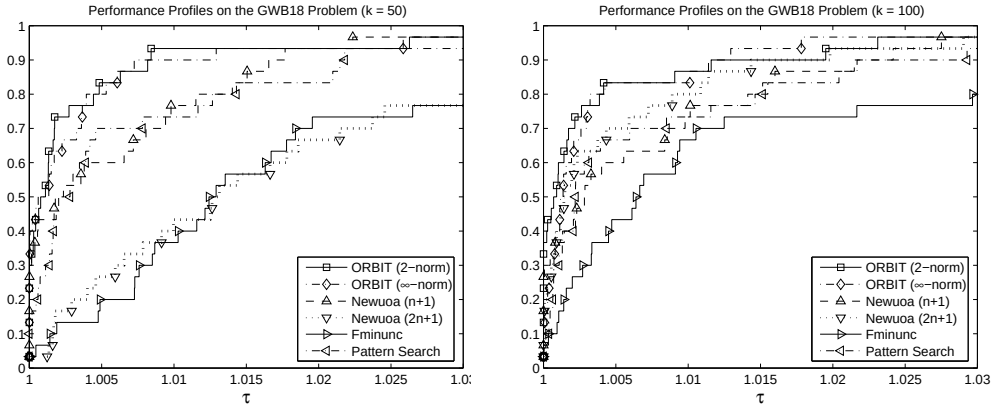

FIG. 6.3. *$\rho_s(\tau)$ on GWB18 After: (a) 50 evaluations, (b) 100 evaluations.*

too high, NYC would either have to abandon the water supply or build a filtration plant costing around $8 billion. It is thus more effective to control the phosphorous at the watershed level than to build a plant. Hence an accurate model is required to assess the impact of changes in management practices on phosphorous loads.

Following [20], we consider the Town Brook watershed (37 km^2), which is inside the larger Cannonsville (1200 km^2) watershed. Our goal is to calibrate the watershed model for flow against real measured flow data over a period of 1096 days:

$$\min \left\{ \sum_{t=1}^{1096} (Q_t^{meas} - Q_t^{sim}(x))^2 : x_i^{\min} \leq x_i \leq x_i^{\max},\ i = 1, \ldots, n \right\}. \tag{6.1}$$

Here, x is a vector of $n = 14$ model parameters, Q_t^{meas} and Q_t^{sim} are the measured and simulated flows on day t, respectively.

Figure 6.1 (a) shows the mean of the best function value for 30 different starting points. In Figure 6.2 (a) we show $\log_2$-scaled performance plots after 50 function evaluations (a full quadratic model in $\mathbb{R}^{14}$ would require 120 evaluations). ORBIT ∞-norm is the best algorithm on 46.7% of the starting points while the NEWUOA algorithm interpolating fewer points is the best for 26.7% of the starting points. We note that both of these algorithms are within a factor of 2 of the best possible algorithm at least 90% of the time.

After 200 evaluations, the ∞-norm and 2-norm variants of ORBIT were the best algorithm 20% and 16.7% of the time, respectively, while the $2n+1$ and $n+2$ variants of NEWUOA were the best algorithms 23.3% of the time as shown in Figure 6.1 (b).

6.2. Optimization for Groundwater Bioremediation. Groundwater bioremediation is the process of cleaning up contaminated groundwater by utilizing the energy-producing and cell-synthesizing activities of microorganisms to transform contaminants into harmless substances. Injection wells pump water and electron acceptors (e.g. oxygen) or nutrients (e.g. nitrogen and phosphorus) into the groundwater in order to promote growth of microorganisms. We assume that sets of both injection wells and monitoring wells, used for measuring concentration of the contaminant, are currently in place at fixed locations. The entire planning horizon is divided into management periods and the goal is to determine the pumping rates for each injection well at the beginning of each management period so that the total pumping cost is minimized subject to constraints that the contaminant concentrations at the monitoring wells are below some threshold level at the end of the remediation period.

In this investigation, we consider a hypothetical contaminated aquifer whose characteristics are symmetric about a horizontal axis. The aquifer is discretized using a two-dimensional finite element mesh. There are 6 injection wells and 84 monitoring wells (located at the nodes of the mesh) that are also symmetric about the horizontal axis. By symmetry, we only need to make pumping decisions for 3 of the injection wells. Six management periods are employed, yielding a total of 18 decision variables. Since we are only able to detect feasibility of a pumping strategy after running the simulation, we eliminate the constraints by means of a penalty term as done by Yoon and Shoemaker [24]. We refer to this problem as GWB18.

Figure 6.1 (b) shows the mean of the best function value for 30 different starting points. Note that by the time the NEWUOA variant interpolating $2n+1 = 37$ points has formed its first underdetermined quadratic model, the two ORBIT variants have made significantly greater progress in minimizing the function. Further note that the finite difference-based Fminunc only makes progress every $n + 1 = 19$ evaluations since it must estimate a gradient using evaluations very close to the current point.

In Figure 6.3 (a) we show performance plots after 50 function evaluations. ORBIT ∞-norm and ORBIT 2-norm are the best algorithms on 33.3% and 30.0% of the starting points, respectively while the NEWUOA algorithm interpolating fewer points is the best for 23.3% of the starting points. These three algorithms are relatively robust, coming within a factor of 1.02 of the best performing algorithm 94% of the time. The NEWUOA $2n + 1$ variant interpolates at 37 points and does not achieve the greatest decrease for any of the starting points. After 100 evaluations (still fewer than the number required to fit a single full quadratic model in $\mathbb{R}^{18}$), the ∞-norm and 2-norm variants of ORBIT were the best algorithm 13.3% and 36.7% of the time, respectively, while the $2n+1$ and $n+2$ variants of NEWUOA were the best algorithm 10.0% and 26.7% of the time, respectively, as shown in Figure 6.3 (b).

7. Conclusions and Future Work. Our numerical results allow us to conclude that ORBIT is an effective algorithm for derivative-free optimization of a computationally expensive objective function when only a limited number of function evaluations are permissible. More computationally expensive functions, simulating larger physical domains or using finer discretizations, than the applications considered here would only increase the need for efficient optimization techniques in this setting.

Why do RBF models perform well in our setting? We hypothesize that even though smooth functions look like quadratics locally, our interest is mostly in short term performance. Our nonlinear RBF models can be formed (and maintained) using fewer points than quadratic models while still preserving the approximation bounds guaranteed for linear interpolation models. While other nonlinear models with linear tails could be tailored to better approximate special classes of functions, the property of positive conditional definiteness makes RBFs particularly computationally attractive. Further, the parametric radial functions in Table 3.1 can model a wide variety of functions.

In the future, we hope to better delineate the types of functions which we expect ORBIT to perform well on. We are particularly interested in determining whether ORBIT still outperforms similarly greedy algorithms based on underdetermined quadratic models, especially on problems involving calibration (nonlinear least squares) and feasibility determination based on a quadratic penalty approach. While we have run numerical tests using a variety of different radial functions, to what extent the particular radial function affects the performance of ORBIT remains an open question.

Lastly, we acknowledge that many practical blackbox problems only admit a

limited degree of parallelization. For such problems, researchers with large scale computing environments would achieve greater success with an algorithm, such as Asynchronous Parallel Pattern Search [9], which explicitly evaluates the function in parallel. We have recently begun exploring extensions of ORBIT which take advantage of multiple function evaluations occurring in parallel. Several theoretical questions also remain and are discussed in [22].

Acknowledgments. The authors are grateful to Raju Rohde, Bryan Tolson, and Jae-Heung Yoon for providing the simulation codes after simulation codes from their papers [20] and [24] used by us here as application problems.

REFERENCES

[1] M. Benzi, G.H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numerica, 14 (2005), pp. 1–137.
[2] M. Björkman and K. Holmström, *Global optimization of costly nonconvex functions using radial basis functions*, Optimization and Engineering, 1 (2000), pp. 373 – 397.
[3] M.D. Buhmann, *Radial Basis Functions: Theory and Implementations*, Cambridge University Press, Cambridge, England, 2003.
[4] A.R. Conn, N.I.M. Gould, and P.L. Toint, *Trust-region methods*, MPS-SIAM Series on Optimization, SIAM, Philadelphia, PA, USA, 2000.
[5] A.R. Conn, K. Scheinberg, and P.L. Toint, *Recent progress in unconstrained nonlinear optimization without derivatives*, Math. Programming, 79 (1997), pp. 397–414.
[6] A.R. Conn, K. Scheinberg, and L.N. Vicente, *Geometry of sample sets in derivative free optimization. Part I: Polynomial interpolation*, to appear in Math. Programming, (2005).
[7] E.D. Dolan and J.J. Moré, *Benchmarking optimization software with performance profiles*, Math. Programming, 91 (2002), pp. 201–213.
[8] H.-M. Gutmann, *A radial basis function method for global optimization*, J. of Global Optimization, 19 (2001), pp. 201–227.
[9] P.D. Hough, T.G. Kolda, and V.J. Torczon, *Asynchronous parallel pattern search for nonlinear optimization*, SIAM J. on Scientific Computing, 23 (2001), pp. 134–156.
[10] T.G. Kolda, R.M. Lewis, and V.J. Torczon, *Optimization by direct search: New perspectives on some classical and modern methods*, SIAM Review, 45 (2003), pp. 385–482.
[11] J.J. Moré, B.S. Garbow, and K.E. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Softw., 7 (1981), pp. 17–41.
[12] J.J. Moré and S.M. Wild, *Benchmarking algorithms for optimization of computationally expensive functions*, In preparation, 2007.
[13] R. Oeuvray and M. Bierlaire, *BOOSTERS: A derivative-free algorithm based on radial basis functions*, Tech. Report To appear in International J. of Modelling and Simulation, 2005.
[14] M.J.D. Powell, *UOBYQA: unconstrained optimization by quadratic approximation*, Math. Programming, 92 (2002), pp. 555–582.
[15] ———, *Least Frobenius norm updating of quadratic models that satisfy interpolation conditions*, Math. Programming, 100 (2004), pp. 183–215.
[16] ———, *The NEWUOA software for unconstrained optimization without derivatives*, in Large-Scale Nonlinear Optimization, Springer, 2006, pp. 255–297.
[17] R.G. Regis and C.A. Shoemaker, *A stochastic radial basis function method for the global optimization of expensive functions*, INFORMS J. of Computing, Forthcoming (2007).
[18] The Mathworks, Inc, *Genetic Algorithm and Direct Search Toolbox for Use with MATLAB: User's Guide, Version 1*, 2004.
[19] ———, *Optimization Toolbox for Use with MATLAB: User's Guide, Version 3*, 2004.
[20] B.A. Tolson and C.A. Shoemaker, *Cannonsville watershed swat2000 model development, calibration and validation*, In press, 2007.
[21] H. Wendland, *Scattered Data Approximation*, Cambridge University Press, Cambridge, England, 2005.
[22] S.M. Wild and C.A. Shoemaker, *Global convergence of radial basis function trust-region algorithms for computationally expensive derivative-free optimization*, In preparation.
[23] D. Winfield, *Function minimization by interpolation in a data table*, J. of the Institute of Mathematics and its Applications, 12 (1973), pp. 339–347.
[24] J.-H. Yoon and C.A. Shoemaker, *Comparison of optimization methods for ground-water bioremediation*, J. of Water Resources Planning and Management, 125 (1999), pp. 54–63.