SIAM J. SCI. COMPUT.
Vol. 30, No. 6, pp. 3197–3219

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 is 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 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

DOI. 10.1137/070691814

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 available parallel computing resources are devoted to parallelization within the computationally expensive function, and are not utilized by the optimization algorithm. We note that pattern search methods are well-suited for parallelization [11, 15].

*Received by the editors May 15, 2007; accepted for publication (in revised form) May 23, 2008; published electronically October 13, 2008. 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.

http://www.siam.org/journals/sisc/30-6/69181.html

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

‡Mathematics and Computer Science Department, Saint Joseph's University, Philadelphia, PA 19131 (rregis@sju.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).

When analytic derivatives are unavailable, one approach is to rely on a classical first-order technique employing finite difference-based estimates of ∇f to solve (1.1). However, in addition to the potential presence of computational noise, as both the dimension and computational expense of the function grows, the $n+1$ function evaluations required for such estimates 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 often 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, 21, 23, 24]). However, they are often tailored to minimize the underlying computational complexity, as in [24], 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 [25] and Oeuvray and Bierlaire [21], 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 [7].

To minimize computational overhead, in both [25] and [21], the number of interpolation points employed is fixed, and hence each time a new point is evaluated, a previous point must be dropped from the interpolation set. In ORBIT, the number of points interpolated depends on the number of nearby points available, and the set of points used can vary more freely from one iteration to the next. Our method for adding additional points in a computationally stable manner relies on a property of radial basis functions and is based on a technique from the global optimization literature [2]. Our algorithmic framework works for a wide variety of radial basis functions. While we have recently established a global convergence result in [29], the focus of this paper is on implementation details and the success of ORBIT in practice.

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 ORBIT 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 model m_k which is assumed to approximate f within a neighborhood of the current

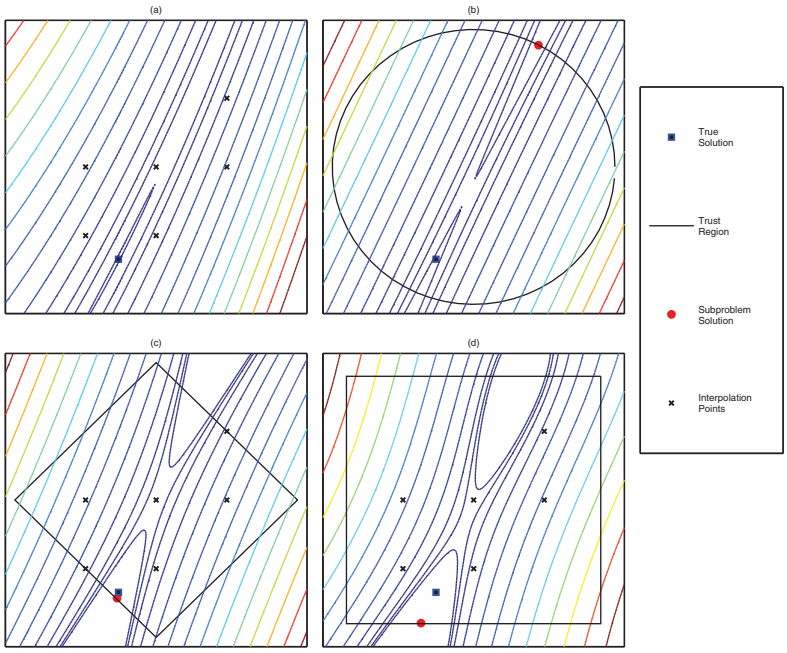


FIG. 2.1. *Trust-region subproblem solutions (all have same axes):* (a) *true function,* (b) *quadratic Taylor model in* 2*-norm trust-region,* (c) *quadratic interpolation model in* 1*-norm trust-region, and* (d) *Gaussian RBF interpolation model in* ∞*-norm trust-region.*

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 Gaussian radial basis function model, approximating f within 2-norm, 1-norm, and ∞-norm trust regions, 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,

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

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 basic trust-region method proceeds iteratively as shown in Figure 2.2. The design of

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.*

the trust-region algorithm ensures that f is sampled only 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 critical points is possible given only minor modifications to (2.6) and the update of Δ_{k+1} in the basic algorithm in Figure 2.2. 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 may be impractical when the function f is computationally expensive. Further, computational noise found in practice means that these finite difference estimates may be unreliable.

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 $\mathcal{Y} = \{y_1 = 0, y_2, \ldots, y_{|\mathcal{Y}|}\} \subset \mathbb{R}^n$:

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

TABLE 2.1
Number of interpolation points needed to uniquely define a full quadratic model.

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

The idea of forming quadratic models by interpolation for optimization without derivatives was proposed by Winfield in the late 1960's [30] and revived in the mid-1990's independently by Powell (UOBYQA [23]) and Conn, Scheinberg, and Toint (DFO [5]).

These methods rely heavily on results from multivariate interpolation, a problem much more difficult than its univariate counterpart [28]. 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 [7], 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 strive for 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 fully quadratic surrogate model can be constructed and the subproblem optimization can begin. Of course, fully linear models can also be obtained in $n + 1$ function evaluations.

Before proceeding, we note that Powell has addressed this difficulty by proposing to satisfy (2.7) uniquely by certain underdetermined quadratics [24]. He developed NEWUOA, a complex but computationally efficient Fortran code using updates of the model such that the change in the model's Hessian is of minimum Frobenius norm [25]. A similar approach is used in a later version of the DFO package [6].

2.2. Fully linear models. In order to avoid geometric conditions on $\mathcal{O}(n^2)$ points, 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 [7] 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 [7], fully linear interpolation models are defined by geometric conditions on the

TABLE 3.1
Popular twice continuously differentiable RBFs and order of conditional positive definiteness.

$\phi(r)$	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, $e^{-\frac{r^2}{\gamma^2}}$

interpolation set. In section 4 we will explore the conditions (2.8) and (2.9) for the radial basis function models introduced next.

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 $d-1$. In addition to guaranteeing uniqueness of the model m_k, the polynomial tail p ensures that m_k belongs to a linear space that also contains the polynomial space $\mathcal{P}_{d-1}^n$.

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, provided in (4.15), of the model.

In addition to being sufficiently smooth, these radial functions in Table 3.1 all share the property of *conditional positive definiteness* [28].

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 $\mathcal{P}_{d-1}^n$. A set of $\hat{p}$ points is said to

be *poised for interpolation in* $\mathcal{P}^n_{d-1}$ if the zero polynomial is the only polynomial in $\mathcal{P}^n_{d-1}$ which vanishes at all $\hat{p}$ points. Conditional positive definiteness of a function is usually proved by Fourier transforms [3, 28] 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 $\mathcal{P}^n_{d-1}$, the interpolation set $\mathcal{Y}$ being poised for interpolation in $\mathcal{P}^n_d$ is equivalent to $\text{rank}(\Pi) = \dim\mathcal{P}^n_{d-1} = \hat{p}$. 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 symmetric system 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 $\mathcal{N}(\Pi)$ (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, 12, 26]. 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. A Gaussian 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 in particular that if more than $\frac{(n+1)(n+2)}{2}$ points are available, RBF models which are CPD of order 0 are able to (uniquely) interpolate f at as many of the points as desired.

As part of his 2005 dissertation, Oeuvray developed a derivative-free trust-region algorithm employing a cubic RBF model with a linear tail [22]. His algorithm, BOOSTERS, was motivated by problems in the area of medical image registration and was subsequently modified to include gradient information when available [21]. Convergence theory was based on the literature available at the time [4].

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 knowledge of all points previously evaluated by the algorithm. This is a fundamental difference between ORBIT and previous algorithms in [21, 23, 25], where, in order to reduce linear algebraic costs, the interpolation set was allowed to change by at most one point. For these algorithms, once a point is dropped from the interpolation

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.*

set it may never return. In our view, these evaluated points contain information which could allow the model to gain additional insight into the function, which is especially valuable when only a limited number of evaluations is possible. This is particularly important when larger trust-regions, especially seen in the early phases of the optimization, result in large steps.

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 coincides with the derivative-based procedure in Figure 2.2 only 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 [7] and is proved in [29].

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 $\left\|Y^{-1}\right\| \leq \frac{\Lambda_Y}{c_k \Delta}$, where c_k (introduced in section 2) is related only to the trust-region norm. 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 $\mathcal{B}$. The assumption that the model's gradient is Lipschitz continuous is milder than assuming that the model is twice differentiable. However, in practice, we expect that this assumption would, in fact, be guaranteed by enforcing a bound on the norm of the model's Hessian.

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 $\|Y^{-1}\| \leq \frac{n^{\frac{n-1}{2}} \theta_1^{-n}}{c_k \Delta}$.*

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 $\|\hat{Y}_j\| \leq 1$. 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

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

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

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

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 beyond the θ_1 thresholding. Further 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 there are very few points within $\mathcal{B}$ at which f has been evaluated. For this reason, we will look to make m_k fully linear within an enlarged 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 **Steps 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

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 no such d_j, continue to **3b**):
  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 $|\mathcal{Y}| = n + 1$, set `linear=true`.
3b. If $|\mathcal{Y}| < n + 1$, set `linear=false`,
  Save first column z_1 of Z as a model-improving direction,
  For $d_j \in \mathcal{D}$ such that $\|d_j\|_k \leq 2\Delta_{\max}$ (if no such d_j, continue below):
    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 columns z_i of 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.*

$r_{ii} \geq \theta_1$. 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 defined in (4.3) must be well-conditioned. In particular we note that

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

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, 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 $\|Y^{-1}\|$. If we still are unable to find $n + 1$ points, $\mathcal{Y}$ is not poised for linear interpolation and hence the RBF model would not be uniquely defined. Thus we must evaluate additional points (given by the directions in Z) to ensure 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 [22] the RBF interpolation set generally changes by at most one point from one iteration to the next and is based on an updating technique applied to the larger system in (3.3). 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:

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

As suggested by Lemmas 4.1 and 4.2, we note that in practice we work with a scaled polynomial matrix Π_y^T. For example, for our linear polynomial matrix, we scale the displacements in Y by Δ_k, e.g., $\pi(\frac{y}{\Delta_k})$. This scaling does not affect the analysis of the algorithm using linear tails in [29].

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 $\mathcal{N}(\Pi_y)$ 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 $\mathcal{N}(\Pi)$. 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

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

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

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:

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

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. Hence, our procedure is to iteratively add available points to $\mathcal{Y}$ provided that $\sqrt{\sigma - \|L^{-1}v\|^2} \geq \theta_2$ until $|\mathcal{Y}| = p_{\max}$.

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 and hence guarantee the Lipschitz continuity needed in Lemma 4.1.

Recall that we have confined ourselves to consider only 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 $|\frac{\phi'(r)}{r}|$ and $|\phi''(r)|$. Of particular interest is the quantity

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

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

The following Lemma is a consequence of the preceding remarks and is proved formally in [29].

LEMMA 4.3. *Let $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_k\|_k \leq 2\Delta_{\max}\}$. Let $\mathcal{Y} \subset \mathcal{B} - x_k$ be a set of distinct interpolation points, $n+1$ of which are affinely independent, and $|f(x_k + y_i)| \leq f_{\max}$ 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 $x \in \mathcal{B}$:*

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

Note that if $\sup_{x \in \mathcal{L}(x_0)} |f(x)| \leq f_{\max}$, $\|\nabla^2 m_k(x)\|$ is bounded on $\mathbb{R}^n$ for all k. Since $m_k \in C^2$, it follows that ∇m is Lipschitz continuous and κ_H is a possible Lipschitz constant on $\mathcal{L}(x_0)$. This justifies 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).

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 $\kappa_d \in (0, 1]$:

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

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

Initialize $s = -\frac{\nabla m_k(x_k)}{\|\nabla m_k(x_k)\|_k}\Delta_k$.

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

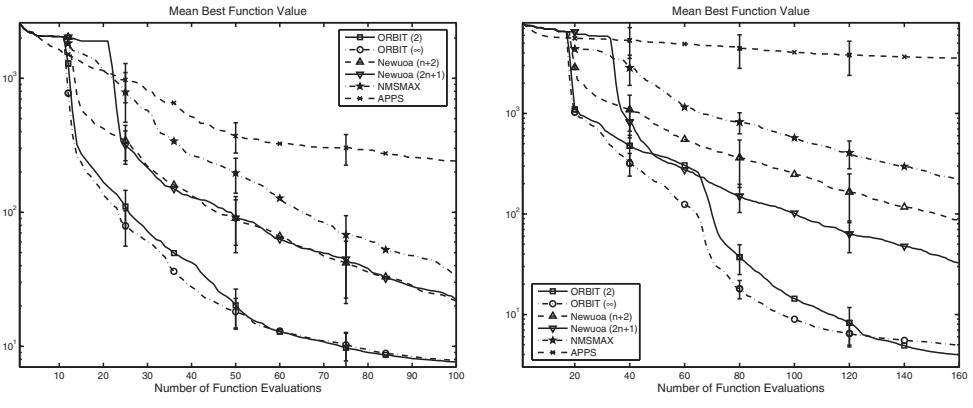
FIG. 4.3. *Backtracking algorithm for obtaining a sufficient decrease in* ***Step*** 5 *of Figure* 4.1 ($\alpha \in (0,1), \kappa_d \in (0,1]$).

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 $m_k(x_k + \tilde{s}) \leq m_k(x_k + \hat{s})$.

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 [29] of ORBIT to first-order critical points satisfying $\nabla f(x_*) = 0$.

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 limited 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 [22] and [21] the number of function evaluations needed to reach some convergence goal is reported, while in [9] and [26] means plots similar to those shown in Figure 5.1 are given.

Using 30 different starting points (see section 5.3), 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



(a) (b)

FIG. 5.1. *Average of* 30 *starting points on two functions from* $\mathcal{P}$ *(*$\log_{10}$ *scale, lowest line is best):* (a) *Extended Rosenbrock* ($n = 10$)*;* (b) *Extended Powell singular* ($n = 16$).

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 using the same starting point. We now discuss one way to complement the means plots in Figure 5.1.

5.1. Performance and data profiles. In [8], Dolan and Moré develop a procedure for visualizing the relative success of solvers on a set of benchmark problems. Their *performance profiles* are gaining currency in the optimization community and 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 α is:

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

The performance profile $\rho_s(\alpha)$ is a probability distribution function capturing the probability that the performance ratio for s is within a factor α of the best possible ratio. Conclusions based on $\rho_s(\alpha)$ 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 and derivative-free settings have recently been examined in [19]. The convergence test used there is

$$f(x_0) - f(x) \geq (1 - \tau)(f(x_0) - f_L), \tag{5.3}$$

which is equivalent to requiring that x achieve a reduction which is at least $1-\tau$ of the best possible reduction $f(x_0) - f_L$. The parameter f_L is chosen to be the minimum function value found by any of the solvers in $\mathcal{S}$ within the maximum number of function evaluations, μ_f. This convergence test allows a user to choose an accuracy level τ appropriate for the resolution of their simulator and goals of their application.

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 of interest. The performance metric $t_{p,s}$ used here is then the number of function evaluations needed to satisfy the convergence test (5.3) for a given $\tau > 0$.

Since performance profiles do not show the number of function evaluations needed to solve a problem, *data profiles* are also introduced in [19]. The data profile,

$$d_s(\kappa) = \frac{1}{|\mathcal{P}|}\text{size}\left\{p \in \mathcal{P} : \frac{t_{p,s}}{n_p + 1} \leq \kappa\right\}, \tag{5.4}$$

where n_p is the dimension of problem $p \in \mathcal{P}$, represents the percentage of problems

that can be solved by solver s with the equivalent of κ (simplex) gradient estimates, corresponding to $\kappa(n+1)$ function evaluations. A data profile is again paired with an accuracy level $\tau > 0$ associated with the convergence test (5.3).

5.2. Algorithms tested. We compared ORBIT to a number of competitive serial algorithms for derivative-free optimization. Here we report the results of this testing for three freely available algorithms, APPSPACK, NMSMAX, and NEWUOA. No implementation of the BOOSTERS code in [22, 21] is available.

All algorithms considered required an initial starting point, x_0, a maximum number of function evaluations, μ_f, and a starting parameter, Δ_0. The values of these inputs change from problem to problem but are kept consistent across all of the algorithms considered. We note that appropriate scaling of variables is a key determinant of performance of derivative-free algorithms and any knowledge of the specific application by the user should be provided to these algorithms in order to obtain a scaled trust-region norm or pattern. For our tests, we assumed no such knowledge was available and gave all algorithms standard (unit) scaling in each variable.

We used the APPSPACK (version 4.0.2) pattern search method because APPSPACK performed well in recent testing on a groundwater problem [9]. APPSPACK is an asynchronous parallel pattern search method [11, 14], which systematically samples the function along search directions defining a pattern (which is by default the set of plus and minus coordinate directions $\{\pm e_1, \ldots, \pm e_n\}$), scaled much the same way as a trust-region. APPSPACK can be run in serial mode (used here) and can handle constraints. We note that since APPSPACK is designed to be run in parallel, its full power is not demonstrated in serial mode. This code requires a choice of scaling, an initial step size, and a final pattern size. We set the scaling to 1, the initial step size to Δ_0 to conform with the other algorithms tested, and the final pattern size to $10^{-19}\Delta_0$ to ensure that the algorithm will not terminate until it reaches the maximum number of function evaluations.

An implementation of the Nelder–Mead method was used because this method is popular among application scientists. Many implementations of this method exist, and we used the NMSMAX code, available from the Matrix Computation Toolbox [13], because it came with default inputs which performed well. The NMSMAX code requires a choice of starting simplex and final step size. We used a right-angled simplex with side length Δ_0 to conform with the other algorithms tested and a final step size of 0 to ensure that the algorithm will not terminate until it reaches the maximum number of function evaluations.

We used the NEWUOA code because it performed best among quadratic model-based methods in comparisons in [22, 21]. Powell's Fortran NEWUOA code [25] requires an initial radius which we set to Δ_0, a final radius which we set to $10^{-15}\Delta_0$ to again ensure that the algorithm will not terminate until it reaches the maximum number of function evaluations, and a number of interpolation points p. We tested two variants, one with Powell's recommended $p = 2n+1$ and one with the minimum $p = n+2$, a strategy which may work well in the initial stages of the optimization.

We implement ORBIT using a cubic RBF model with both 2-norm and ∞-norm trust-regions. For all experiments we used the ORBIT (Figure 4.1) parameters: $\eta_0 = 0$, $\eta_1 = .2$, $\gamma_0 = \frac{1}{2}$, $\gamma_1 = 2$, $\Delta_{\max} = 10^3\Delta_0$, $\theta_0 = 10$ $\theta_1 = 10^{-3}$, $\theta_2 = 10^{-7}$, $\epsilon_g = 10^{-10}$, and $p_{\max} = 3n$. For the backtracking line search algorithm in Figure 4.3, we set $\kappa_d = 10^{-4}$ and $\alpha = .9$. In the ORBIT implementation tested here, we also relied on the FMINCON routine from MATLAB [17], which is based on a sequential quadratic programming (SQP) method.

TABLE 5.1
Hypercube bounds for generating the starting points and initial step size Δ_0 for the test functions.

Function	n	Hypercube bounds	Δ_0
Wood	4	$[-3,2]^4$	.5
Trigonometric	5	$[-1,3]^5$	.4
Discrete Boundary Value	8	$[-3,3]^8$	.6
Extended Rosenbrock	10	$[-2,2]^{10}$	.4
Variably Dimensioned	10	$[-2,2]^{10}$	.4
Broyden Tridiagonal	11	$[-1,1]^{11}$	.2
Extended Powell Singular	16	$[-1,3]^{16}$	.4
Town Brook Problem	14	$[0,1]^{14}$	.1
GWB18 Problem	18	$[0,1]^{18}$	.1

On the largest problem tested here ($n = 18$), ORBIT required nearly .5 seconds per average iteration on a 2.4 GHz Pentium 4 desktop, with the overwhelming majority of the time being spent inside FMINCON. This expense is magnitudes more than the other algorithms tested, making our present implementation viable only for sufficiently expensive objective functions.

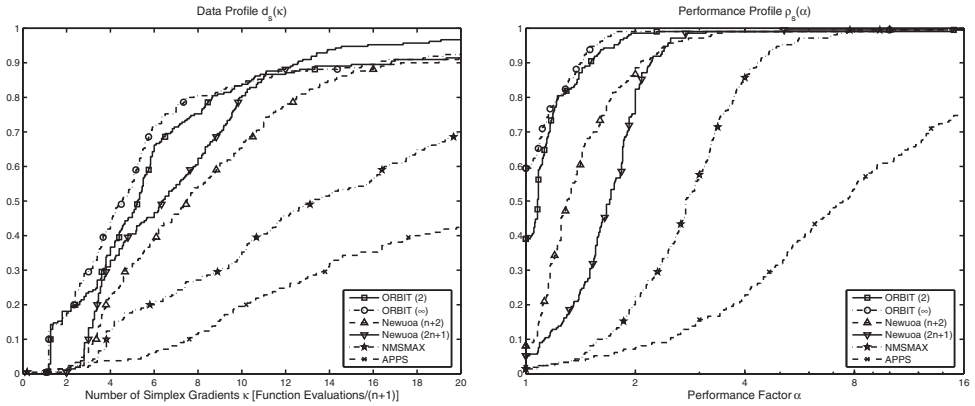
Our choice of inputs ensures that ORBIT, NMSMAX, and NEWUOA initially evaluate the function at the vertices of the right-angled simplex with sides of length Δ_0. The APPSPACK code is given this same initialization but moves off this pattern as soon as a lower function value is achieved.

5.3. Test problems. We first employ a subset of seven functions of varying dimensions from the Moré–Garbow–Hillstrom (MGH) set of test functions for unconstrained optimization [18]: Wood ($n = 4$), Trigonometric ($n = 5$), Discrete Boundary Value ($n = 8$), Extended Rosenbrock ($n = 10$), Variably Dimensioned ($n = 10$), Broyden Tridiagonal ($n = 11$), and Extended Powell Singular ($n = 16$). For each function, we use MATLAB's uniform random number generator `rand` to obtain 30 random starting points within a hypercube containing the true solution. Table 5.1 lists the hypercubes, each chosen by the authors to contain the corresponding solution (see [18]) roughly in its interior, for each function used to generate the 30 starting points. Also shown are the initial step length Δ_0 values, corresponding to 10% of the size of each hypercube. We note that this step length is an important parameter for many of the algorithms tested and was chosen to be of the same order of magnitude as the distance from the starting point to the solution in order to not put any algorithm at a disadvantage.

We collect these 30 different trials for each function, to yield a set $\mathcal{P}$ of 210 profile problems, each consisting of a (function, starting point) pair. For these profile problems we set the maximum number of function evaluations to $\mu_f = 510$, corresponding to 30 simplex gradients for the largest of the profile problems in $\mathcal{P}$ ($n = 16$).

The mean trajectories over the 30 starting points on the Extended Rosenbrock and Powell Singular functions are shown in Figures 5.1(a) and (b), respectively. We note that in the first $n + 1$ evaluations, APPSPACK obtains the least function value since it moves off the initial simplex with which the remaining algorithms start. These plots show that, after this initialization, the means and 95%-pointwise confidence intervals of the two ORBIT implementations are below the four alternatives, with the two NEWUOA variants being the next best for this range of function evaluations.

The plots shown are representative of the behavior on the other five test functions as is evidenced by the data profiles shown in Figure 5.2(a). Here we see the ORBIT



(a) (b)

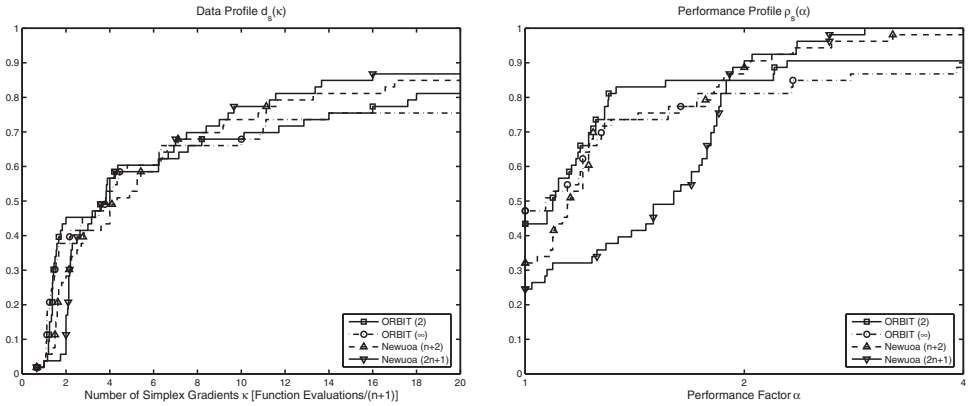
FIG. 5.2. *Set $\mathcal{P}$ of smooth test problems:* (a) *data profile $d_s(\kappa)$ shows the percentage of problems solved, $\tau = .01$;* (b) *performance profile $\rho_s(\alpha)$ shows the relative performance, $\tau = .1$.*

variants solve the largest percentage of these profile problems to an accuracy level $\tau = .01$ when fewer than 12 simplex gradients (corresponding to $12(n+1)$ function evaluations) are available. As the number of function evaluations available increases, we see that the NEWUOA $(2n+1)$ algorithm solves a larger percentage of profile problems.

Figure 5.2(b) shows the performance profiles for the 210 profile problems in $\mathcal{P}$ and the accuracy level $\tau = .1$. Here we see, for example, that the ∞-norm variant of ORBIT was the fastest algorithm to achieve 90% of the reduction possible in 510 evaluations on roughly half of the profile problems. Further, the ∞-norm and 2-norm variants of ORBIT achieved this reduction within a factor $\alpha = 2$ of the fewest evaluations on nearly 95% of the profile problems in $\mathcal{P}$. These data and performance profiles illustrate the success of ORBIT on smooth problems, particularly when few function evaluations are available.

The MGH problems considered are twice-continuously differentiable and have a single local minimum. We also tested the algorithms on a set of 53 profile problems $\mathcal{P}_N$ detailed in [19], which are "noisy" variants of a set of CUTEr problems [10]. The functions which these profile problems are based on are distinct from the seven considered above and vary in dimension from $n = 2$ to $n = 12$. We set the maximum number of function evaluations to $\mu_f = 390$, corresponding to 30 simplex gradients for the largest of the profile problems in $\mathcal{P}_N$. We show only the results for ORBIT and NEWUOA since these performed the best in the above MGH tests. More extensive comparisons between the $2n+1$ NEWUOA variant, APPSPACK, and NMSMAX on $\mathcal{P}_N$ can be found in [19].

In the data profiles for $\tau = .01$ shown in Figure 5.3(a) we see that ORBIT and NEWUOA have very similar performance, with ORBIT solving slightly more profile problems with smaller numbers of function evaluations, and NEWUOA solving slightly more profile problems with more evaluations. In the performance profiles shown in Figure 5.3(b), we see that the ORBIT variants each achieve the $\tau = .1$ accuracy level in the fewest number of evaluations on roughly 45% of the profile problems. These plots show that ORBIT is competitive with NEWUOA on the profile problems in $\mathcal{P}_N$, particularly for lower accuracy levels and when fewer function evaluations are available.



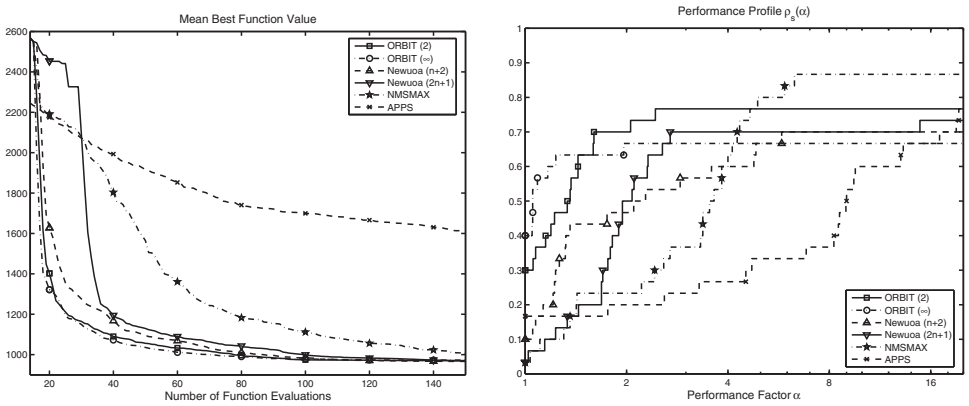
(a) (b)

FIG. 5.3. *Set $\mathcal{P}_N$ of noisy test problems:* (a) *data profile $d_s(\kappa)$ shows the percentage of problems solved, $\tau = .01$;* (b) *performance profile $\rho_s(\alpha)$ shows the relative performance, $\tau = .1$.*

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 bound constraints are present. We will treat both of these problems as unconstrained by adding a smooth penalty term. This approach is justifiable since the bound constraints are not rigid, and this penalty term is representative of the goals of these particular environmental problems in practice. We note that since APPSPACK can handle bound constraints, we will provide it with the bound constraints given by the problem and allow it to automatically compute its default scaling based on these bounds. The value of Δ_0 for each application was again chosen to be 10% of the size of the hypercube corresponding to these bounds as shown in Table 5.1.

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. A more complex groundwater bioremediation model than GWB18 is described in [20] where it takes 3 hours per simulation. The Town Brook watershed is part of the Cannonsville watershed, and simulations with flow, sediment, and phosphorous require up to 7 minutes. Larger models like the Chesapeake watershed model of the EPA [16] can take over 2 hours per simulation.

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



(a) (b)

FIG. 6.1. *Town Brook problem* ($n = 14$)*:* (a) *mean best function value (*30 *trials);* (b) *performance profile* $\rho_s(\alpha)$ *shows the relative performance,* $\tau = .2$.

its reservoirs in upstate New York. If phosphorous levels become 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.

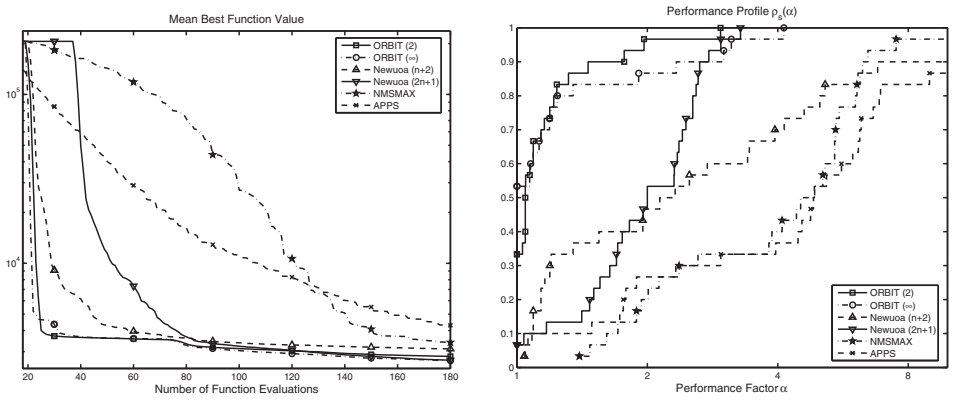
Following [27], we consider the Town Brook watershed (37 km^2), which is inside the larger Cannonsville (1200 km^2) watershed. Our goal is to calibrate the watershed model for flow against real measured flow data over a period of 1096 days:

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

Here, x is a vector of $n = 14$ model parameters, and 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 generated uniformly within the bound-constrained region by MATLAB's random number generator `rand`. Here we see that, on average, ORBIT obtains the best function value when between 20 and 140 function evaluations are available (a single full quadratic model in $\mathbb{R}^{14}$ would require 120 evaluations). We note that while APPSPACK was given the bound constraints of the problem, there was very little difference (on few of the starting points) between this trajectory and the trajectory APPSPACK produced without the constraints. This is because, for the most part, the optimization remained in the bound-constrained region.

When a maximum number of function evaluations of $\mu_f = 450$ are available, we obtain the performance profiles (for $\tau = .2$) shown in Figure 6.1(b). Here we see that the ∞- and 2-norm variants of ORBIT require the fewest function evaluations to attain this accuracy on 40% and 30% of the problems, respectively. While rarely the fastest algorithm, NMSMAX is the most *reliable* algorithm for this test. It is the only algorithm which is able to attain this accuracy on over 80% of the starting points (for f_L computed with $\mu_f = 450$ evaluations). We note that this is consistent with the findings in [19] where APPSPACK and NMSMAX were successful at finding higher accuracy solutions for particularly messy functions if given many function evaluations.



(a) (b)

FIG. 6.2. *GWB*18 *problem* ($n = 18$)*:* (a) *(*$\log_{10}$ *scale) mean best function value (*30 *trials);* (b) *performance profile* $\rho_s(\alpha)$ *shows the relative performance,* $\tau = .01$.

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 [31]. We refer to this problem as GWB18.

Figure 6.2(a) shows the mean of the best function value for 30 different starting points again generated uniformly within the bound-constrained region by MATLAB's random number generator. 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 significant progress in minimizing the function. Also note that since ORBIT is interpolating at up to $3n$ points, it is able to make greater progress than the $n + 2$ variant of NEWUOA.

In Figure 6.2(b) we show performance plots for $\tau = .01$ with a maximum number of function evaluations of $\mu_f = 570$. Here we see that the ORBIT ∞-norm and ORBIT 2-norm are the best algorithms on 53% and 33% of the starting points, respectively.

7. Conclusions and future work. Our numerical results allow us to conclude that ORBIT is an effective algorithm for derivative-free optimization of a computation-

ally 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 full quadratic models while still preserving the approximation bounds guaranteed for linear interpolation models. Other nonlinear models with linear tails could be tailored to better approximate special classes of functions, but the property of conditional positive definiteness makes RBFs particularly computationally attractive since we can include virtually as many evaluated points as desired. Our method of adding points relies on precisely this property. Further, because we are not bound by linear algebraic expenses, the number of points interpolated can vary from iteration to iteration, and we can keep a complete history of the points available for interpolation. Lastly, 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 on which we expect ORBIT to perform well. 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, as does the performance of ORBIT on problems containing areas of nondifferentiability. We also hope to better understand the computational effects of different pivoting strategies in our method of verifying that a model is fully linear. We also intend to explore alternative methods for solving the subproblem since the current use of FMINCON is both a large part of the algorithm's overhead and currently limits ORBIT from obtaining high accuracy solutions.

Lastly, we acknowledge that many practical blackbox problems admit only 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 [11, 14], 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 and the presence of bound constraints. Several theoretical questions also remain and are discussed in [29].

Acknowledgments. The authors are grateful to Raju Rohde, Bryan Tolson, and Jae-Heung Yoon for providing the simulation codes based on simulation codes from their papers [27] and [31] used by us here as application problems. We are also grateful to Andy Conn, Jorge Moré, and two anonymous referees for their helpful suggestions.

REFERENCES

[1] M. Benzi, G.H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[2] M. Björkman and K. Holmström, *Global optimization of costly nonconvex functions using radial basis functions*, Optim. Eng., 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, 2000.
[5] A.R. Conn, K. Scheinberg, and P.L. Toint, *Recent progress in unconstrained nonlinear optimization without derivatives*, Math. Program., 79 (1997), pp. 397–414.
[6] A.R. Conn, K. Scheinberg, and P.L. Toint, *A derivative free optimization algorithm in practice*, in Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 1998.
[7] A.R. Conn, K. Scheinberg, and L.N. Vicente, *Geometry of interpolation sets in derivative free optimization*, Math. Program., 111 (2008), pp. 141–172.
[8] E.D. Dolan and J.J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.
[9] K.R. Fowler, J.P. Reese, C.E. Kees, Jr., J.E. Dennis, C.T. Kelley, C.T. Miller, C. Audet, A.J. Booker, G. Couture, R.W. Darwin, M.W. Farthing, D.E. Finkel, J.M. Gablonsky, G. Gray, and T.G. Kolda, *A comparison of derivative-free optimization methods for water supply and hydraulic capture community problems*, Adv. Water Resources, 31 (2008), pp. 743–757.
[10] N.I.M. Gould, D. Orban, and P.L. Toint, *CUTEr and SifDec: A constrained and unconstrained testing environment, revisited*, ACM Trans. Math. Software, 29 (2003), pp. 373–394.
[11] G.A. Gray and T.G. Kolda, *Algorithm* 856*: APPSPACK* 4.0*: Asynchronous parallel pattern search for derivative-free optimization*, ACM Trans. Math. Software, 32 (2006), pp. 485–507.
[12] H.-M. Gutmann, *A radial basis function method for global optimization*, J. Global Optim., 19 (2001), pp. 201–227.
[13] N. Higham, *The Matrix Computation Toolbox*, www.ma.man.ac.uk/~higham/mctoolbox.
[14] T.G. Kolda, *Revisiting asynchronous parallel pattern search for nonlinear optimization*, SIAM J. Optim., 16 (2005), pp. 563–586.
[15] T.G. Kolda, R.M. Lewis, and V.J. Torczon, *Optimization by direct search: New perspectives on some classical and modern methods*, SIAM Rev., 45 (2003), pp. 385–482.
[16] L.C. Linker, G.W. Shenk, R.L. Dennis, and J.S. Sweeney, *Cross-media models of the Chesapeake Bay watershed and airshed*, Water Quality Eco. Model., 1 (2000), pp. 91–122.
[17] Mathworks, Inc, *Optimization Toolbox for Use with MATLAB: User's Guide, V.* 3, 2004.
[18] J.J. Moré, B.S. Garbow, and K.E. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Software, 7 (1981), pp. 17–41.
[19] J.J. Moré and S.M. Wild, *Benchmarking derivative-free optimization algorithms*, Technical report ANL/MCS-P1471-1207, Argonne National Laboratory, MCS Division, 2007.
[20] P. Mugunthan, C.A. Shoemaker, and R.G. Regis, *Comparison of function approximation, heuristic and derivative-based methods for automatic calibration of computationally expensive groundwater bioremediation models*, Water Resour. Res., 41 (2005).
[21] R. Oeuvray and M. Bierlaire, *A new derivative-free algorithm for the medical image registration problem*, Int. J. Modelling and Simulation, 27 (2007), pp. 115–124.
[22] R. Oeuvray, *Trust-region methods based on radial basis functions with application to biomedical imaging*, Ph.D. thesis, EPFL, Lausanne, Switzerland, 2005.
[23] M.J.D. Powell, *UOBYQA: Unconstrained optimization by quadratic approximation*, Math. Program., 92 (2002), pp. 555–582.
[24] M.J.D. Powell, *Least Frobenius norm updating of quadratic models that satisfy interpolation conditions*, Math. Program., 100 (2004), pp. 183–215.
[25] M.J.D. Powell, *The NEWUOA software for unconstrained optimization without derivatives*, in Large-Scale Nonlinear Optimization, Springer, New York, 2006, pp. 255–297.
[26] R.G. Regis and C.A. Shoemaker, *A stochastic radial basis function method for the global optimization of expensive functions*, INFORMS J. Comput., 19 (2007), pp. 497–509.
[27] B.A. Tolson and C.A. Shoemaker, *Cannonsville reservoir watershed swat*2000 *model development, calibration and validation*, J. Hydrology, 337 (2007), pp. 68–86.
[28] H. Wendland, *Scattered Data Approximation*, Cambridge University Press, Cambridge, England, 2005.
[29] S.M. Wild and C.A. Shoemaker, *Global convergence of radial basis function trust-region algorithms*, Technical report ORIE-1470, Cornell University, School of Operations Research and Information Engineering, 2008.
[30] D. Winfield, *Function minimization by interpolation in a data table*, J. Inst. Math. Appl., 12 (1973), pp. 339–347.
[31] J.-H. Yoon and C.A. Shoemaker, *Comparison of optimization methods for ground-water bioremediation*, J. Water Res. Planning and Management, 125 (1999), pp. 54–63.