DERIVATIVE-FREE OPTIMIZATION ALGORITHMS FOR COMPUTATIONALLY EXPENSIVE FUNCTIONS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Stefan Martin Wild

January 2009

DERIVATIVE-FREE OPTIMIZATION ALGORITHMS FOR COMPUTATIONALLY EXPENSIVE FUNCTIONS

Stefan Martin Wild, Ph.D.

Cornell University 2009

This thesis concerns the development and analysis of derivative-free optimization algorithms for simulation-based functions that are computationally expensive to evaluate.

The first contribution is the introduction of *data profiles* as a tool for analyzing the performance of derivative-free optimization solvers when constrained by a computational budget. Using these profiles, together with a convergence test that measures the decrease in function value, we find that on three different sets of test problems, a model-based solver performs better than the two direct search solvers tested.

The next contribution is a new model-based derivative-free algorithm, ORBIT, for unconstrained local optimization. A trust-region framework using interpolating Radial Basis Function (RBF) models is employed. RBF models allow ORBIT to interpolate nonlinear functions using fewer function evaluations than many of the polynomial models considered by present techniques. We provide an analysis of the approximation guarantees obtained by interpolating the function at a set of sufficiently affinely independent points. We detail necessary and sufficient conditions that an RBF model must obey to fit within our framework and prove that this framework allows for convergence to first-order critical points. We present numerical results on test problems as well as three application problems from environmental engineering to support ORBIT's effectiveness when relatively few function evaluations are available. The framework used by ORBIT is also extended to include other models, in particular undetermined interpolating quadratics. These quadratics are flexible in their ability to interpolate at dynamic numbers of previously evaluated points.

The third contribution is a new multistart global optimization algorithm, GORBIT, that takes advantage of the expensive function evaluations done in the course of both the global exploration and local refinement phases. We modify ORBIT to handle both bound constraints and external functional evaluations and use it as the local solver. For the global exploration phase, a new procedure for making maximum use of the information from previous evaluations, MIPE, is introduced. Numerical tests motivating our approach are presented and we illustrate using GORBIT on the problem of finding error-prone systems for Gaussian elimination.

BIOGRAPHICAL SKETCH

Stefan Wild was born in March 1980 in Memphis, Tennessee to Swiss parents. He worked a diverse set of jobs (office assistant, trail laborer, bus boy, telephone book deliverer, dishwasher, farm hand, ice skating guard, boat renter, ice cream scooper, record store clerk, study monitor, environmental consulting intern, resident adviser) until he discovered that research in applied mathematics interested him. He received concurrent Bachelor of Science and Master of Science degrees from the University of Colorado (Boulder) in Applied Mathematics in May 2003.

He began his studies at the School of Operations Research and Information Engineering in the Fall of 2003. He was awarded a Department of Energy Computational Science Graduate Fellowship in the Spring of 2005, which funded most of his work at Cornell. In December of 2005 he married Hilary Petrock on the big island of Hawaii. He spent the Summer of 2006 at the Laboratory of Advanced Numerical Simulations at Argonne National Laboratory. He received the Ph.D. degree in August 2008 and will be joining the Mathematics and Computer Science Division at Argonne National Laboratory in September 2008 as a Director's Postdoctoral Fellow.

To my family.

ACKNOWLEDGEMENTS

I am grateful to Christine Shoemaker for her guidance and support as my advisor. Her experience and passion for saving the world with mathematics is an inspiration. I am indebted to Jorge Moré who has served as a great mentor in all things research and look forward to our future collaborations. I also thank Peter Diamessis, Philip Liu, David Ruppert, and Charles Van Loan for their help over the years and for serving on my committee. I had the fortune of being taught continuous optimization by the unparalleled Cornell faculty: Bob Bland, Tom Coleman, Adrian Lewis, Jim Renegar, Mike Todd, and Les Trotter.

My wife Hilary has been a never-ending source of motivation and this dissertation is as much a credit to her patience and support as anything academic I picked up along the way. Her tolerance of math parties has resulted in her knowing more optimization than any attorney should. I would like to thank my parents, brother, sister, and bulldog Elsie for their constant support and encouragement.

I am also grateful for the other students in my advisor's group. I especially mention Rommel Regis, whose thorough reading has aided much of this work, and Amandeep Singh for his help running computationally expensive models.

My stay at Cornell was not without valuable diversions and the Computer Science Megahurtz hockey team, Fanclub Collective, and Swiss Club of Cornell deserve special mention. My colleagues in ORIE have tolerated my quirks for too long and I'll fondly remember more students than could be listed here. I should especially thank my perpetually sunny officemates Anke, Emmanuel, and Jie. Lastly, I had many fine student mentors while at Cornell, and mention Bharath, Chandra, Davina, Deniz, Dhruv, Dmitriy, Frans, Gavin, Ghosh, Joerg, Nikolai, Parthanil, Pascal, Peter, Retsef, the Sams, Souvik, Tuncay, and Yuriy.

I am exceptionally grateful for support from a Department of Energy Computational Science Graduate Fellowship, grant number DE-FG02-97ER25308, which has opened doors and forced me to earn the title of *applied* mathematician. The research in this thesis was additionally supported by NSF grants BES-022917, CBET-0756575, CCF-0305583, and DMS-0434390, and a GAANN Research Fellowship during my second year. I am grateful for travel awards from SIAM, the Graduate School at Cornell, and Front Range Scientific Computations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

This thesis is concerned with optimization algorithms for problems where the objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is distinguished by two features:

1. Only the function values of the function f (and not derivative information such as ∇f) are available to the algorithm, and
2. The function f is computationally expensive to evaluate.

Many of the functions motivating the present work arise from solving complex environmental engineering problems where the objective depends on the output of a numerical simulation of a physical process. These simulators are expensive to evaluate because they involve numerically solving systems of partial differential equations governing the underlying physical phenomena. Such simulators increasingly benefit from the advancement of parallel computing. However, function evaluation remains the dominant expense in many optimization problems since the savings in time are often offset by increased accuracy of the simulation (for example using finer spatial meshes or smaller time steps when simulating a physical process).

These simulators often only produce a single set of output, rarely providing the derivative of the output with respect to the decision variables of interest. Furthermore, these simulators often depend on legacy or proprietary codes, preventing the application of standard derivative estimation techniques such as Automatic Differentiation (AD) [7]. Estimation of gradient information by finite-differencing can also be problematic due to the fixed resolutions of the simulator or presence of computational noise. This noise is deterministic and often due to round-off error or finite tolerances on termination criteria used by procedures within the simulator. Similar techniques specialized for the simulation-based setting, such as Implicit Filtering [27], Pattern Search methods [44], or the Nelder-Mead method [54], often rely on sampling at $\mathcal{O}(n)$ points at each iteration. A central question guiding the present work is whether an algorithm can produce a better approximate minimizer than these methods when very few function evaluations are permitted.

We pursue *derivative-free* optimization algorithms, algorithms requiring only the ability to evaluate the objective function. The types of problems these algorithms can solve are extremely diverse in nature. Engineering problems in automotive, electrical, environmental, and biomedical engineering applications can be found in [11, 37, 4, 25, 55], with other examples coming from numerical linear algebra [35] and statistics [9].

In Chapter 2 we present an expanded version of [52], in which we propose a methodology for benchmarking derivative-free optimization algorithms. We extend the performance profiles in [21] to the derivative-free setting by introducing an attainable convergence test that does not depend on the gradient of the function. We also introduce the concept of *data profiles*, which measure the fraction of problems that can be solved in terms of the number of function evaluations. These profiles seek to capture a user's desire to achieve reduction in the function value from an initial value $f(x_0)$ within some fixed computational budget, which limits the number of function evaluations that can be performed.

We introduce collections of smooth, noisy, and piecewise-smooth problems and analyze the performance of three solvers on these problems. Our tests show that, on these problems, the NEWUOA solver [61] outperforms two direct search solvers.

Using a subset of the known function values, NEWUOA builds a quadratic approximation model, which is used to obtain new points for evaluation.

Encouraged by the success of a model-based method, in Chapter 3 (published in [75]) we propose a new algorithm for unconstrained derivative-free local optimization. Our algorithm, named ORBIT (Optimization by Radial Basis Interpolation in Trust-regions), relies on a *Radial Basis Function* (RBF) model interpolating the function at a set of points at which the function has been evaluated. Using a trust-region framework, the model is trusted to approximate the function within a local neighborhood of the best point found so far. The RBF property of *conditional positive definiteness* is used to ensure that the interpolation is unique and well-posed given as few as $n + 1$ function values.

Accordingly, ORBIT is able to build nonlinear models of the function given only a linear number (in the number of variables, n) of function values. We present numerical results on a set of test problems from Chapter 1 to motivate the use of ORBIT when only a relatively small number of expensive function evaluations are available. We also test ORBIT against alternative derivative-free methods on two environmental application problems. The first consists of finding optimal parameters for a watershed model in order calibrate the model to measured flow data. The second application problem involves determining optimal pumping rates in order to obtain a cost-effective bioremediation plan for a contaminated groundwater site. These results support the effectiveness of ORBIT on blackbox functions for which no special mathematical structure is known or available.

In Chapter 4 we address several theoretical issues in ORBIT. Following the recent work in [15], we prove that ORBIT converges to first-order critical points, with $\nabla f(x_*) = 0$, while requiring only minor assumptions on the function f.

Additional details of the intermediate lemmas are provided in Appendix A. We show that Taylor-like first-order error bounds can be obtained while only assuming that a model with a bounded Hessian interpolates a smooth function on a set of $n+1$ sufficiently affinely independent points. We also analyze the choice of radial function used by the RBF model and give conditions for the radial model to fit within the globally convergent framework of ORBIT.

We test the effect of choosing different radial functions on a set of noisy test problems from Chapter 1. We also illustrate ORBIT's flexibility in interpolating varying numbers of function values on these test problems. We apply ORBIT to a truly expensive application problem related to cleaning up contamination on a former naval ammunition depot in Hastings, Nebraska. In this case, each set of decision variables corresponds to a different pumping strategy and evaluating the simulator using one of these strategies requires almost an hour of CPU time on a Pentium 4 machine.

In Chapter 5 we present an expanded version of [74], in which the framework of ORBIT is extended to quadratic polynomial models. In order to allow for interpolation of fewer points than the dimension of quadratics requires, these models will primarily be underdetermined. Hence, in this chapter we will work with quadratic interpolation models whose Hessians are of minimum norm. We illustrate the resulting algorithm on a couple of test functions and show the benefits gained from the flexibility of interpolating variable numbers of function values. As with ORBIT, these benefits come at the expense of a greater linear algebraic cost of each optimization iteration, a negligible cost when the function evaluation is expensive.

We acknowledge that many simulation-based problems are not adequately addressed by local optimization theory and algorithms, and hence we turn to *global* *optimization* problems in Chapter 6 and the development of the GORBIT algorithm. Based on its effectiveness in unconstrained local optimization, in GORBIT we employ ORBIT within a multistart method. This multistart method explores the global domain by stochastically sampling the function to determine good candidates at which to start the local ORBITprocedure. ORBIT then refines estimates of local minima by focusing on local optimization from these candidate points. We introduce a multistart method MIPE (Maximum Information from Previous Evaluations) so that the global exploration phase can take advantage of the output from the local optimization runs. MIPE shares some features with Multi Level Single Linkage (MLSL) [40]. We also detail changes made to ORBIT, such as accounting for bound-constraints and using the information from the function values obtained during the global exploration phase in MLSL and MIPE.

The resulting global optimization algorithm GORBIT differs from existing multistart-based algorithms such as [65] in that it seeks to use as much of its own evaluation history as possible. We test variants of the algorithm on a set of global test problems in an effort to gain insight into the advantage gained when better-utilizing the function values obtained in the course of the optimization. We also apply these algorithms to the global optimization problem of finding matrices that result in large errors when solving a system by Gaussian elimination [22].

We have used radial basis functions and trust-region methods to develop local and global algorithms targeting computationally expensive simulation-based optimization problems. We have used the structure of RBFs and trust-region methods to better understand the theoretical properties of the local algorithm and extended our framework to an algorithm employing a quadratic polynomial model. We have also developed a testing methodology specifically for users with a computational budget. We have tested our algorithms on a variety of applications and shown that our algorithms perform well when relatively few expensive function evaluations are available.

CHAPTER 2

BENCHMARKING DERIVATIVE-FREE OPTIMIZATION ALGORITHMS*

Derivative-free optimization has experienced a renewed interest over the past decade that has encouraged a new wave of theory and algorithms. While this research includes computational experiments that compare and explore the properties of these algorithms, there is no consensus on the benchmarking procedures that should be used to evaluate derivative-free algorithms.

We explore benchmarking procedures for derivative-free optimization algorithms when there is a limited computational budget. The focus of the work in this chapter is the unconstrained optimization problem

$$\min \left\{ f(x) : x \in \mathbb{R}^n \right\}, \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ may be noisy (as discussed in Section 2.3) or non-differentiable and, in particular, in the case where the evaluation of f is computationally expensive. These expensive optimization problems arise in science and engineering because evaluation of the function f often requires a complex deterministic simulation based on solving the equations (for example, nonlinear eigenvalue problems, ordinary or partial differential equations) that describe the underlying physical phenomena. The computational noise associated with these complex simulations means that obtaining derivatives is difficult and unreliable. Moreover, these simulations often rely on legacy or proprietary codes and hence must be treated as black-box functions, necessitating a derivative-free optimization algorithm.

*THIS IS AN EXPANDED VERSION OF A PAPER OF THE SAME TITLE [52] COAUTHORED BY JORGE J. MORÉ.

Several comparisons have been made of derivative-free algorithms on noisy optimization problems that arise in applications. In particular, we mention [26, 31, 37, 56, 65]. The most ambitious work in this direction [26] is a comparison of six derivative-free optimization algorithms on two variations of a groundwater problem specified by a simulator. In this work algorithms are compared by their trajectories (plot of the best function value against the number of evaluations) until the solver satisfies a convergence test based on the resolution of the simulator.

Benchmarking derivative-free algorithms on selected applications with trajectory plots provides useful information to users with related applications. In particular, users can find the solver that delivers the largest reduction within a given computational budget. However, the conclusions in these computational studies do not readily extend to other applications.

Most researchers have relied on a selection of problems from the CUTEr [29] collection of optimization problems for their work on testing and comparing derivative-free algorithms. Work in this direction includes [14, 37, 48, 55, 61]. The performance data gathered in these studies is the number of function evaluations required to satisfy a convergence test when there is a limit μ_f on the number of function evaluations. The convergence test is sometimes related to the accuracy of the current iterate as an approximation to a solution, while in other cases it is related to a parameter in the algorithm. For example, a typical convergence test for trust region methods [14, 55, 61] requires that the trust region radius be smaller than a given tolerance.

Users with expensive function evaluations are often interested in a convergence test that measures the decrease in function value. In Section 2.1 we propose the convergence test

$$f(x_0) - f(x) \geq (1 - \tau)(f(x_0) - f_L), \tag{2.2}$$

where $\tau > 0$ is a tolerance, x_0 is the starting point for the problem, and f_L is computed for each problem as the smallest value of f obtained by any solver within a given number μ_f of function evaluations. This convergence test is well suited for derivative-free optimization because it is invariant to the affine transformation $f \mapsto \alpha f + \beta$ ($\alpha > 0$) and measures the function value reduction $f(x_0) - f(x)$ achieved by x relative to the best possible reduction $f(x_0) - f_L$.

The convergence test (2.2) was used by Marazzi and Nocedal [48] but with f_L set to an accurate estimate of f at a local minimizer obtained by a derivative-based solver. In Section 2.1 we show that setting f_L to an accurate estimate of f at a minimizer is not appropriate when the evaluation of f is expensive, since no solver may be able to satisfy (2.2) within the user's computational budget.

We use performance profiles [21] with the convergence test (2.2) to evaluate the performance of derivative-free solvers. Instead of using a fixed value of τ, we use $\tau = 10^{-k}$ with $k \in \{1, 3, 5, 7\}$ so that a user can evaluate solver performance for different levels of accuracy. These performance profiles are useful to users who need to choose a solver that provides a given reduction in function value within a limit of μ_f function evaluations.

To the authors' knowledge, previous work with performance profiles has not varied the limit μ_f on the number of function evaluations and has used large values for μ_f. The underlying assumption has been that the long-term behavior of the algorithm is of utmost importance. This assumption is not likely to hold, however, if the evaluation of f is expensive.

Performance profiles were designed to compare solvers and thus use a performance ratio instead of the number of function evaluations required to solve a problem. As a result, performance profiles do not provide the percentage of problems that can be solved (for a given tolerance τ) with a given number of function evaluations. This information is essential to users with expensive optimization problems and thus an interest in the short-term behavior of algorithms. On the other hand, the *data profiles* of Section 2.1 have been designed to provide this information.

The remainder of this chapter is devoted to demonstrating the use of performance and data profiles for benchmarking derivative-free optimization solvers. Section 2.1 reviews the use of performance profiles with the convergence test (2.2) and defines data profiles.

Section 2.2 provides a brief overview of the solvers selected to illustrate the benchmarking process: the Nelder-Mead NMSMAX code [34], the pattern-search APPSPACK code [30], and the model-based trust region NEWUOA code [61]. Since the emphasis of this chapter is on the benchmarking process, no attempt was made to assemble a large collection of solvers. The selection of solvers was guided mainly by a desire to examine the performance of a representative subset of derivative-free solvers.

Section 2.3 describes the benchmark problems used in the computational experiments. We use a selection of problems from the CUTEr [29] collection for the basic set; but since the functions f that describe the optimization problem are invariably smooth, with at least two continuous derivatives, we augment this basic set with noisy and piecewise-smooth problems derived from this basic set. The choice of noisy problems was guided by a desire to mimic simulation-based optimization problems.

The benchmarking results in Section 2.4 show that data and performance profiles provide complementary information that measures the strengths and weaknesses of optimization solvers as a function of the computational budget. Data profiles are useful, in particular, to assess the short-term behavior of the algorithms. The results obtained from the benchmark problems of Section 2.3 show that the model-based solver NEWUOA performs better than the direct search solvers NMSMAX and APPSPACK even for noisy and piecewise-smooth problems. These results also provide estimates for the performance differences between these solvers.

Standard disclaimers [21] in benchmarking studies apply to the results in Section 2.4. In particular, all solvers were tested with the default options, so results may change if these defaults are changed. In a similar vein, our results apply only to the current version of these solvers and may change with future versions of these solvers.

2.1 Benchmarking Derivative-Free Optimization Solvers

Performance profiles, introduced by Dolan and Moré [21], have proved to be an important tool for benchmarking optimization solvers. Dolan and Moré define a benchmark in terms of a set $\mathcal{P}$ of benchmark problems, a set $\mathcal{S}$ of optimization solvers, and a convergence test $\mathcal{T}$. Once these components of a benchmark are defined, performance profiles can be used to compare the performance of the solvers. In this section we first propose a convergence test for derivative-free optimization solvers and then examine the relevance of performance profiles for optimization problems with expensive function evaluations.

2.1.1 Performance Profiles

Performance profiles are defined in terms of a performance measure $t_{p,s} > 0$ obtained for each $p \in \mathcal{P}$ and $s \in \mathcal{S}$. For example, this measure could be based on the amount of computing time or the number of function evaluations required to satisfy the convergence test. Larger values of $t_{p,s}$ indicate worse performance. For any pair (p, s) of problem p and solver s, the performance ratio is defined by

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

Note that the best solver for a particular problem attains the lower bound $r_{p,s} = 1$. The convention $r_{p,s} = \infty$ is used when solver s fails to satisfy the convergence test on problem p.

The *performance profile* of a solver $s \in \mathcal{S}$ is defined as the fraction of problems where the performance ratio is at most α, that is,

$$\rho_s(\alpha) = \frac{1}{|\mathcal{P}|}\text{size}\Big\{p \in \mathcal{P} : r_{p,s} \leq \alpha\Big\}, \tag{2.3}$$

where $|\mathcal{P}|$ denotes the cardinality of $\mathcal{P}$. Thus, a performance profile is the probability distribution (assuming a uniform distribution on $\mathcal{P}$) for the ratio $r_{p,s}$. Performance profiles seek to capture how well the solver performs relative to the other solvers in $\mathcal{S}$ on the set of problems in $\mathcal{P}$. Note, in particular, that $\rho_s(1)$ is the fraction of problems for which solver $s \in \mathcal{S}$ performs the best and that for α sufficiently large, $\rho_s(\alpha)$ is the fraction of problems solved by $s \in \mathcal{S}$. In general, $\rho_s(\alpha)$ is the fraction of problems with a performance ratio $r_{p,s}$ bounded by α, and thus solvers with high values for $\rho_s(\alpha)$ are preferable.

Benchmarking gradient-based optimization solvers is reasonably straightforward once the convergence test is chosen. The convergence test is invariably based on the gradient, for example,

$$\|\nabla f(x)\| \leq \tau \|\nabla f(x_0)\|$$

for some $\tau > 0$ and norm $\|\cdot\|$. This convergence test is augmented by a limit on the amount of computing time or the number of function evaluations. The latter requirement is needed to catch solvers that are not able to solve a given problem.

Benchmarking gradient-based solvers is usually done with a fixed choice of tolerance τ that yields reasonably accurate solutions on the benchmark problems. The underlying assumption is that the performance of the solvers will not change significantly with other choices of the tolerance and that, in any case, users tend to be interested in solvers that can deliver high-accuracy solutions. In derivative-free optimization, however, users are interested in both low-accuracy and high-accuracy solutions. In practical situations, when the evaluation of f is expensive, a low-accuracy solution is all that can be obtained within the user's computational budget. Moreover, in these situations, the accuracy of the data may warrant only a low-accuracy solution.

Benchmarking derivative-free solvers requires a convergence test that does not depend on the gradient. We propose to use the convergence test

$$f(x) \leq f_L + \tau(f(x_0) - f_L), \tag{2.4}$$

where $\tau > 0$ is a tolerance, x_0 is the starting point for the problem, and f_L is computed for each problem $p \in \mathcal{P}$ as the smallest value of f obtained by any solver within a given number μ_f of function evaluations. The convergence test (2.4) can also be written as done in (2.2),

$$f(x_0) - f(x) \geq (1 - \tau)(f(x_0) - f_L),$$

and this shows that (2.4) requires that the reduction $f(x_0) - f(x)$ achieved by x be at least $1 - \tau$ times the best possible reduction $f(x_0) - f_L$.

The convergence test (2.4) was used by Elster and Neumaier [23] but with f_L set to an accurate estimate of f at a global minimizer. This test was also used by Marazzi and Nocedal [48] but with f_L set to an accurate estimate of f at a local minimizer obtained by a derivative-based solver. Setting f_L to an accurate estimate of f at a minimizer is not appropriate when the evaluation of f is expensive because no solver may be able to satisfy (2.4) within the user's computational budget. Even for problems with a cheap f, a derivative-free solver is not likely to achieve accuracy comparable to a derivative-based solver. On the other hand, if f_L is the smallest value of f obtained by any solver, then at least one solver will satisfy (2.4) for any $\tau \geq 0$.

An advantage of (2.4) is that it is invariant to the affine transformation $f \mapsto \alpha f + \beta$ where $\alpha > 0$. Hence, we can assume, for example, that $f_L = 0$ and $f(x_0) = 1$. There is no loss in generality in this assumption because derivative-free algorithms are invariant to the affine transformation $f \mapsto \alpha f + \beta$. Indeed, algorithms for gradient-based optimization (unconstrained and constrained) problems are also invariant to this affine transformation.

The tolerance $\tau \in [0, 1]$ in (2.4) represents the percentage decrease from the starting value $f(x_0)$. A value of $\tau = 0.1$ may represent a modest decrease, a reduction that is 90% of the total possible, while smaller values of τ correspond to larger decreases. As τ decreases, the accuracy of $f(x)$ as an approximation to f_L increases. The accuracy of x as an approximation to some minimizer depends on the growth of f in a neighborhood of the minimizer. As noted, users are interested in the performance of derivative-free solvers for both low-accuracy and high-accuracy solutions. A user's expectation of the decrease possible within their computational budget will vary from application to application.

The following result relates the convergence test (2.4) to convergence results for gradient-based optimization solvers.

Theorem 2.1. *Assume that $q : \mathbb{R}^n \mapsto \mathbb{R}$ is a strictly convex quadratic and that x^* is the unique minimizer of q. If $q_L = q(x^*)$, then $x \in \mathbb{R}^n$ satisfies the convergence test* (2.4) *if and only if*

$$\|\nabla q(x)\|_* \le \tau^{1/2} \, \|\nabla q(x_0)\|_* \tag{2.5}$$

for the norm $\| \cdot \|_*$ defined by

$$\|v\|_* = \|G^{-\frac{1}{2}} \, v\|_2,$$

and G is the Hessian matrix of q.

Proof. Since q is a quadratic, G is the Hessian matrix of q, and x^* is the unique minimizer,

$$q(x) = q(x^*) + \frac{1}{2}(x - x^*)^T G (x - x^*).$$

Hence, the convergence test (2.4) holds if and only if

$$(x - x^*)^T G (x - x^*) \le \tau (x_0 - x^*)^T G (x_0 - x^*),$$

which in terms of the square root $G^{\frac{1}{2}}$ is just

$$\|G^{\frac{1}{2}}(x - x^*)\|_2^2 \le \tau \|G^{\frac{1}{2}}(x_0 - x^*)\|_2^2.$$

We obtain (2.5) by noting that since x^* is the minimizer of the quadratic q and G is the Hessian matrix, $\nabla q(x) = G(x - x^*)$. □

Other variations on Theorem 2.1 are of interest. For example, it is not difficult to show, by using the same proof techniques, that (2.4) is also equivalent to

$$\frac{1}{2}\|\nabla f(x)\|_*^2 \le \tau\,(f(x_0) - f(x^*)). \tag{2.6}$$

This inequality shows, in particular, that we can expect that the accuracy of x, as measured by the gradient norm $\|\nabla f(x)\|_*$, to increase with the square root of $f(x_0) - f(x^*)$.

Similar estimates hold for the error in x because $\nabla f(x) = G(x - x^*)$. Thus, in view of (2.5), the convergence test (2.4) is equivalent to

$$\|x - x^*\|_\diamond \le \tau^{1/2}\,\|x_0 - x^*\|_\diamond,$$

where the norm $\|\cdot\|_\diamond$ is defined by

$$\|v\|_\diamond = \|G^{\frac{1}{2}} v\|_2.$$

In this case the accuracy of x in the $\|\cdot\|_\diamond$ norm increases with the distance of x_0 from x^* in the $\|\cdot\|_\diamond$ norm.

We now explore an extension of Theorem 2.1 to nonlinear functions that is valid for an arbitrary starting point x_0. The following result shows that the convergence test (2.4) is (asymptotically) the same as the convergence test (2.6).

Lemma 2.1. *If $f : \mathbb{R}^n \mapsto \mathbb{R}$ is twice continuously differentiable in a neighborhood of a minimizer x^* with $\nabla^2 f(x^*)$ positive definite, then*

$$\lim_{x \to x^*} \frac{f(x) - f(x^*)}{\|\nabla f(x)\|_*^2} = \frac{1}{2}, \tag{2.7}$$

where the norm $\|\cdot\|_*$ is defined in Theorem 2.1 and $G = \nabla^2 f(x^*)$.

Proof. We first prove that

$$\lim_{x \to x^*} \frac{\|\nabla^2 f(x^*)^{1/2}(x - x^*)\|}{\|\nabla f(x)\|_*} = 1. \tag{2.8}$$

This result can be established by noting that since $\nabla^2 f$ is continuous at x^* and $\nabla f(x^*) = 0$,

$$\nabla f(x) = \nabla^2 f(x^*)(x - x^*) + r_1(x), \qquad r_1(x) = o(\|x - x^*\|).$$

If λ_1 is the smallest eigenvalue of $\nabla^2 f(x^*)$, then this relationship implies, in particular, that

$$\|\nabla f(x)\|_* \geq \frac{1}{2}\lambda_1^{1/2}\|x - x^*\| \tag{2.9}$$

for all x near x^*. This inequality and the previous relationship prove (2.8). We can now complete the proof by noting that since $\nabla^2 f$ is continuous at x^* and $\nabla f(x^*) = 0$,

$$f(x) = f(x^*) + \frac{1}{2}\|\nabla^2 f(x^*)^{1/2}(x - x^*)\|^2 + r_2(x), \qquad r_2(x) = o(\|x - x^*\|^2).$$

This relationship, together with (2.8) and (2.9) complete the proof. □

Lemma 2.1 shows that there is a neighborhood $N(x^*)$ of x^* such that if $x \in N(x^*)$ satisfies the convergence test (2.4) with $f_L = f(x^*)$, then

$$\|\nabla f(x)\|_* \leq \gamma\, \tau^{1/2}\, (f(x_0) - f(x^*))^{1/2}, \tag{2.10}$$

where the constant γ is a slight overestimate of $2^{1/2}$. Conversely, if γ is a slight underestimate of $2^{1/2}$, then (2.10) implies that (2.4) holds in some neighborhood of x^*. Thus, in this sense, the gradient test (2.10) is asymptotically equivalent to (2.4) for smooth functions.

2.1.2 Data Profiles

We can use performance profiles with the convergence test (2.4) to benchmark optimization solvers for problems with expensive function evaluations. In this case

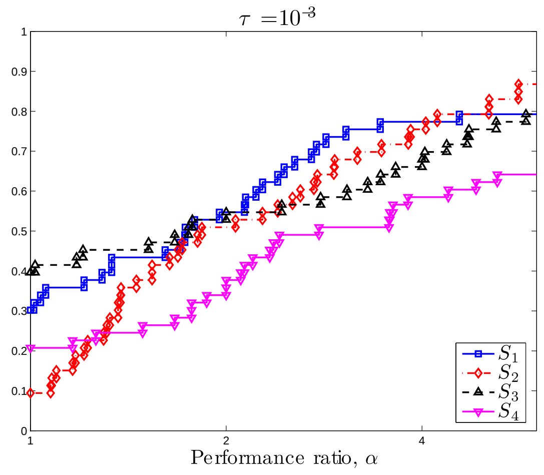


Figure 2.1: Sample performance profile $\rho_s(\alpha)$ (logarithmic scale) for derivative-free solvers S_1 to S_4, $\tau = 10^{-3}$.

the performance measure $t_{p,s}$ is the number of function evaluations because this is assumed to be the dominant cost per iteration. Performance profiles provide an accurate view of the relative performance of solvers within a given number μ_f of function evaluations. Performance profiles do not, however, provide sufficient information for a user with an expensive optimization problem.

Figure 2.1 shows a typical performance profile for derivative-free optimization solvers with the convergence test (2.4) and $\tau = 10^{-3}$. Users generally are interested in the best solver, and for these problems and level of accuracy, solver S_3 has the best performance. However, it is also important to pay attention to the performance difference between solvers. For example, consider the performance profiles ρ_1 and ρ_4 at a performance ratio of $\alpha = 2$, $\rho_1(2) \approx 55\%$ and $\rho_4(2) \approx 35\%$. These profiles show that solver S_4 requires more than twice the minimum number of function evaluations on roughly 20% more of the problems as solver S_1. This is a significant difference in performance.

The performance profiles in Figure 2.1 provide an accurate view of the performance of derivative-free solvers for $\tau = 10^{-3}$. However, these results were obtained with a limit of $\mu_f = 1300$ function evaluations and thus are not directly relevant to a user for which this limit exceeds their computational budget.

Users with expensive optimization problems are often interested in the performance of solvers as a function of the number of functions evaluations. In other words, these users are interested in *the percentage of problems that can be solved (for a given tolerance τ) with κ function evaluations.* We can obtain this information by letting $t_{p,s}$ be the number of function evaluations required to satisfy (2.4) for a given tolerance τ, since then

$$d_s(\kappa) = \frac{1}{|\mathcal{P}|}\text{size}\Big\{p \in \mathcal{P} : t_{p,s} \leq \kappa\Big\} \tag{2.11}$$

is the percentage of problems that can be solved with κ function evaluations. As usual, there is a limit μ_f on the total number of function evaluations, and $t_{p,s} = \infty$ if the convergence test (2.4) is not satisfied after μ_f evaluations.

Griffin and Kolda [32] were also interested in performance in terms of the number of functions evaluations and used plots of the total number of solved problems as a function of the number of (penalty) function evaluations to evaluate performance. They did not investigate how results changed if the convergence test was changed; their main concern was to evaluate the performance of their algorithm with respect to the penalty function.

This definition (2.11) of d_s is independent of the number of variables in the problem $p \in \mathcal{P}$. This is not realistic because, in our experience, the number of function evaluations needed to satisfy a given convergence test is likely to grow as the number of variables increases. We thus define the *data profile* of a solver $s \in \mathcal{S}$

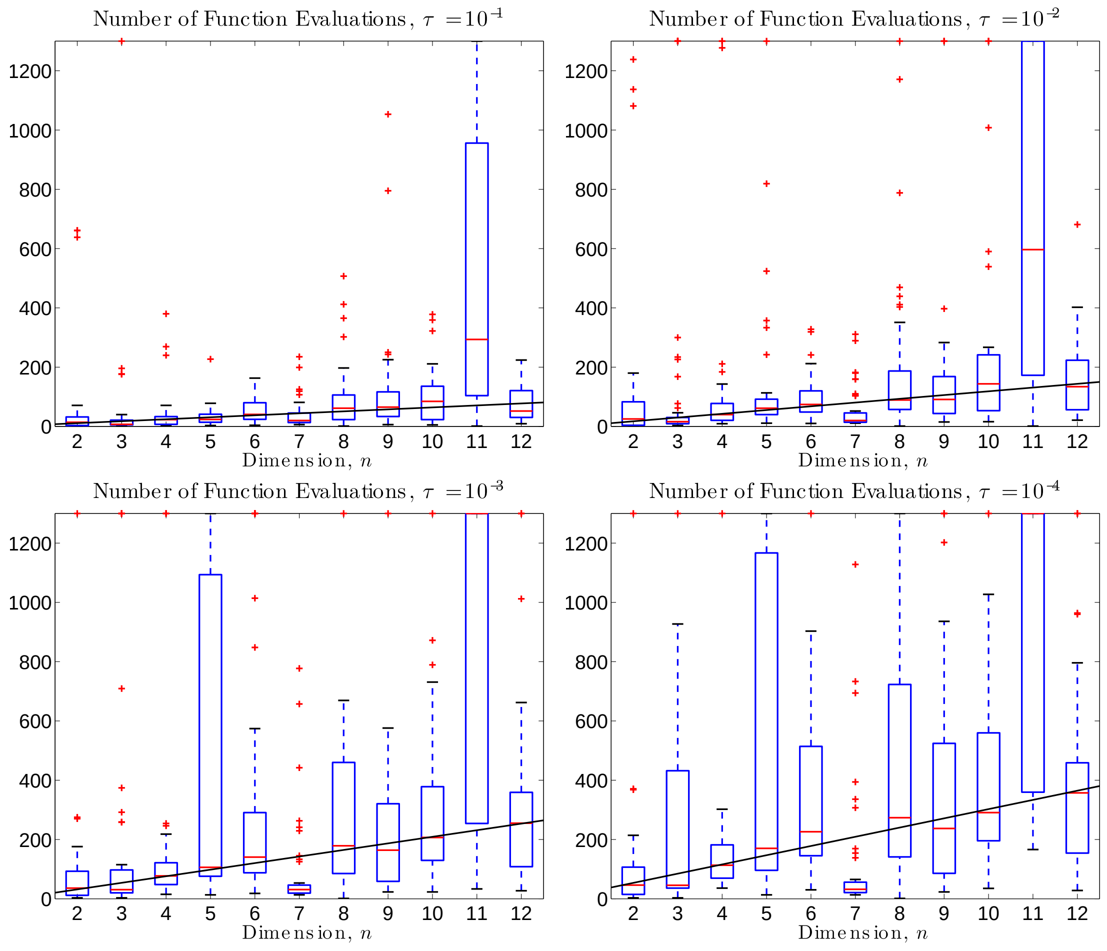


Figure 2.2: Grouping by problem dimension shows that the number of function evaluations required to satisfy the convergence test grows approximately linearly in the dimension (for the numerical results of Section 2.4).

by

$$d_s(\kappa) = \frac{1}{|\mathcal{P}|}\text{size}\Big\{p \in \mathcal{P} : \frac{t_{p,s}}{n_p + 1} \leq \kappa\Big\}, \tag{2.12}$$

where n_p is the number of variables in $p \in \mathcal{P}$. We refer to a plot of (2.12) as a data profile to acknowledge that its application is more general than the one used here and that our choice of scaling is for illustration only. For example, we note that the authors in [2] expect performance of stochastic global optimization algorithms to grow faster than linear in the dimension.

We note that there is support for choosing this linear scaling in the data profiles for the set of local optimization benchmark problems introduced later in Section 2.3. Figure 2.2 shows the number of function evaluations required by the solvers to be discussed in Section 2.2 to achieve a fixed level of accuracy grouped by the problem dimension. The boxes in Figure 2.2 represent the lower and upper quartiles, while the whiskers cover data values within 1.5 of the interquartile range and the crosses represent outliers. Through this box and whisker plot we see that the resulting data (less the dimensions $n = 7$ and $n = 11$), very closely fit a line increasing with the problem dimension.

With this scaling, the unit of cost is $n_p + 1$ function evaluations. This is a convenient unit that can be easily translated into function evaluations. Another advantage of this unit of cost is that $d_s(\kappa)$ can then be interpreted as the percentage of problems that can be solved with the equivalent of κ *simplex gradient estimates*, $n_p + 1$ referring to the number of evaluations needed to compute a one-sided finite-difference estimate of the gradient.

Performance profiles (2.3, Figure 2.1) and data profiles (2.12, Figure 2.3) are cumulative distribution functions, and thus monotone increasing, step functions with a range in $[0, 1]$. However, performance profiles compare different solvers, while data profiles display the raw data. In particular, performance profiles do not provide the number of function evaluations required to solve any of the problems. Also note that the data profile for a given solver $s \in \mathcal{S}$ is independent of other solvers; this is not the case for performance profiles.

Data profiles are useful to users with a specific computational budget who need to choose a solver that is likely to reach a given reduction in function value. The user needs to express the computational budget in terms of simplex gradients and

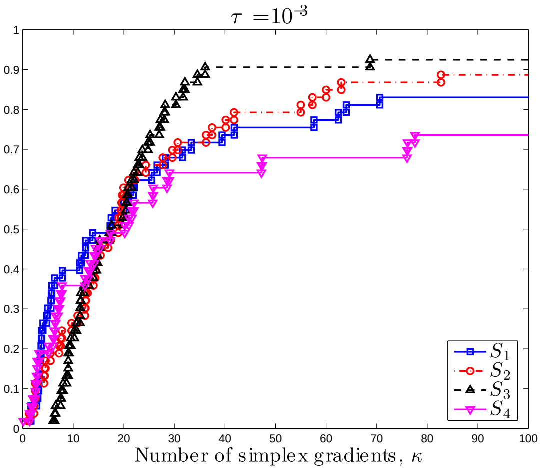


Figure 2.3: Sample data profile $d_s(\kappa)$ (2.12) for derivative-free solvers S_1 to S_4, $\tau = 10^{-3}$.

examine the values of the data profile d_s for all the solvers. For example, if the user has a budget of 50 simplex gradients, then the data profiles in Figure 2.3 show that solver S_3 solves 90% of the problems at this level of accuracy. This information is not available from the performance profiles in Figure 2.1.

We illustrate the differences between performance and data profiles with a synthetic case involving two solvers. Assume that solver S_1 requires k_1 simplex gradients to solve each of the first n_1 problems, but fails to solve the remaining n_2 problems. Similarly, assume that solver S_2 fails to solve the first n_1 problems, but solves each of the remaining n_2 problems with k_2 simplex gradients. Finally, assume that $n_1 < n_2$, and that $k_1 < k_2$. In this case,

$$\rho_1(\alpha) \equiv \frac{n_1}{n_1 + n_2}, \qquad \rho_2(\alpha) \equiv \frac{n_2}{n_1 + n_2},$$

for all $\alpha \geq 1$ if the maximum number of evaluations μ_f allows k_2 simplex gradients. Hence, $n_1 < n_2$ implies that $\rho_1 < \rho_2$, and thus solver S_2 is preferable. This is justifiable because S_2 solves more problems for all performance ratios. On the other hand,

$$d_1(\alpha) = \begin{cases} 0, & \alpha \in [0, k_1) \\ \dfrac{n_1}{n_1 + n_2}, & \alpha \in [k_1, \infty) \end{cases} \qquad d_2(\alpha) = \begin{cases} 0, & \alpha \in [0, k_2) \\ \dfrac{n_2}{n_1 + n_2}, & \alpha \in [k_2, \infty) \end{cases}$$

In particular, $0 = d_2(k) < d_1(k)$ for all budgets of k simplex gradients where $k \in [k_1, k_2)$, and thus solver S_1 is preferable under these budget constraints. This choice is appropriate because S_2 is not able to solve any problems with less than k_2 simplex gradients.

This example illustrates an extreme case, but this can happen in practice. For example, the data profiles in Figure 2.3 show that solver S_2 outperforms S_1 with a computational budget of k simplex gradients where $k \in [20, 100]$, though the differences are small. On the other hand, the performance profiles in Figure 2.1 show that S_1 outperforms S_2.

One other connection between performance profiles and data profiles needs to be emphasized. The limiting value of $\rho_s(\alpha)$ as $\alpha \to \infty$ is the percentage of problems that can be solved with μ_f function evaluations. Thus,

$$d_s(\hat{\kappa}) = \lim_{\alpha \to \infty} \rho_s(\alpha), \tag{2.13}$$

where $\hat{\kappa}$ is the maximum number of simplex gradients performed in μ_f evaluations. Since the limiting value of ρ_s can be interpreted as the reliability of the solver, we see that (2.13) shows that the data profile d_s measures the reliability of the solver (for a given tolerance τ) as a function of the budget μ_f.

2.2 Derivative-Free Optimization Solvers

The selection of solvers $\mathcal{S}$ that we use to illustrate the benchmarking process was guided by a desire to examine the performance of a representative subset of derivative-free solvers, and thus we included both direct search and model-based algorithms. No attempt was made to assemble a large collection of solvers, although we did consider more than a dozen different solvers. Users interested in the performance of other solvers (including SID-PSM [19] and UOBYQA [58]) can find additional results at `www.mcs.anl.gov/~more/dfo`. We note that some solvers were not tested because they require additional parameters outside the scope of this investigation, such as the requirement of bounds by imfil [27, 41].

We considered only solvers that are designed to solve unconstrained optimization problems using only function values, and with an implementation that is both serial and deterministic. We used an implementation of the Nelder-Mead method because this method is popular among application scientists. We also present results for the APPSPACK pattern search method because, in a comparison of six derivative-free methods, this code performed well in the benchmarking [26] of a groundwater problem. We used the model-based trust region code NEWUOA because this code performed well in a recent comparison [55] of model-based methods.

The NMSMAX code is an implementation of the Nelder-Mead method and is available from the Matrix Computation Toolbox [34]. Other implementations of the Nelder-Mead method exist, but this code performs well and has a reasonable default for the size of the initial simplex. All variations on the Nelder-Mead method update an initial simplex defined by $n + 1$ points via a sequence of reflections, expansions, and contractions. Not all of the Nelder-Mead codes that we examined, however, allow the size of the initial simplex to be specified in the calling sequence.

The NMSMAX code requires an initial starting point x_0, a limit on the number of function evaluations, and the choice of a starting simplex. The user can choose either a regular simplex or a right-angled simplex with sides along the coordinate axes. We used the right-angled simplex with the default value of

$$\Delta_0 = \max\{1, \|x_0\|_\infty\} \tag{2.14}$$

for the length of the sides. This default value performs well in our testing.

The APPSPACK code [30] is an asynchronous parallel pattern search method designed for problems characterized by expensive function evaluations. The code can be run in serial mode, and this is the mode used in our computational experiments. This code requires an initial starting point x_0, a limit on the number of function evaluations, the choice of scaling for the starting pattern, and an initial step size. We used unit scaling with an initial step size Δ_0 defined by (2.14) so that the starting pattern was defined by the right-angled simplex with sides of length Δ_0.

The model-based trust region code NEWUOA [61, 62] uses a quadratic model obtained by interpolation of function values at a subset of m previous trial points; the geometry of these points is monitored and improved if necessary. We used $m = 2n + 1$ as recommended by Powell [61]. The NEWUOA code requires an initial starting point x_0, a limit on the number of function evaluations, and the initial trust region radius. We used Δ_0 as in (2.14) for the initial trust region radius.

Our choice of initial settings ensures that all codes are given the same initial information. As a result, both NMSMAX and NEWUOA evaluate the function at the vertices of the right-angled simplex with sides of length Δ_0. The APPSPACK code, however, moves off this initial pattern as soon as a lower function value is obtained.

We effectively set all termination parameters to zero so that all codes terminate only when the limit on the number of function evaluations is exceeded. In a few cases the codes terminate early. This situation happens, for example, if the trust region radius (size of the simplex or pattern) is driven to zero. Since APPSPACK requires a strictly positive termination parameter for the final pattern size, we used 10^{-20} for this parameter.

2.3 Benchmark Problems

The benchmark problems we have selected highlight some of the properties of derivative-free solvers as they face different classes of optimization problems. We made no attempt to define a definitive set of benchmark problems, but these benchmark problems could serve as a starting point for further investigations. This test set is easily available, widely used, and allows us to easily examine different types of problems.

Our benchmark set comprises 22 of the nonlinear least squares functions defined in the CUTEr [29] collection. Each function is defined by m components $f_1, \dots, f_m$ of n variables and a standard starting point x_s.

The problems in the benchmark set $\mathcal{P}$ are defined by a vector (k_p, n_p, m_p, s_p) of integers. The integer k_p is a reference number for the underlying CUTEr function, n_p is the number of variables, m_p is the number of components, and $s_p \in \{0, 1\}$ defines the starting point via $x_0 = 10^{s_p} x_s$, where x_s is the standard starting point for this function. The use of $s_p = 1$ is helpful for testing solvers from a remote starting point because the standard starting point tends to be close to a solution for many of the problems.

Table 2.1: Distribution of problem dimensions.

n_p	2	3	4	5	6	7	8	9	10	11	12
Number of problems	5	6	5	4	4	5	6	5	4	4	5

The benchmark set $\mathcal{P}$ has 53 different problems. No problem is overrepresented in $\mathcal{P}$ in the sense that no function k_p appears more than six times. Moreover, no pair (k_p, n_p) appears more than twice. In all cases,

$$2 \leq n_p \leq 12, \quad 2 \leq m_p \leq 65, \qquad p = 1, \ldots, 53,$$

with $n_p \leq m_p$. The distribution of the dimensions n_p among all 53 problems is shown in Table 2.1, the median dimension being 7.

Users interested in the precise specification of the benchmark problems in $\mathcal{P}$ will find the source code for evaluating the problems in $\mathcal{P}$ at `www.mcs.anl.gov/~more/dfo`. This site also contains source code for obtaining the standard starting points x_s and, a file dfo.dat that provides the integers (k_p, n_p, m_p, s_p).

We use the benchmark set $\mathcal{P}$ defined above to specify benchmark sets for three problem classes: smooth, piecewise smooth, and noisy problems. The *smooth* problems $\mathcal{P}_S$ are defined by

$$f(x) = \sum_{k=1}^{m} f_k(x)^2. \tag{2.15}$$

These functions are twice continuously differentiable on the level set associated with x_0. Only two functions $(k_p = 7, 16)$ have local minimizers that are not global minimizers, but the problems defined by these functions appear only three times in $\mathcal{P}_S$.

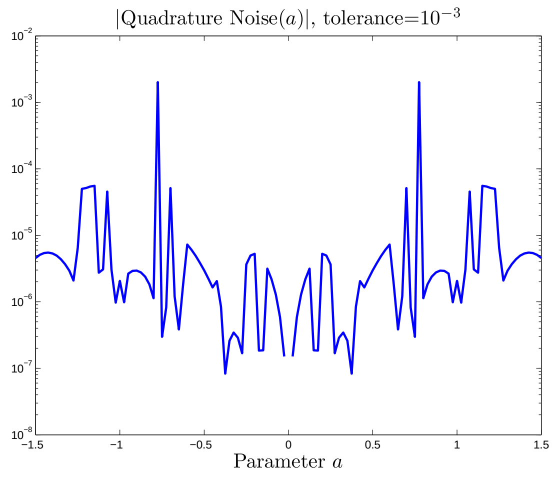


Figure 2.4: Relative error $\left|\frac{f(a)-\tilde{f}(a)}{f(a)}\right|$ for evaluating an integral using adaptive quadrature with an error tolerance 10^{-3}.

The second class of problems mimics simulations that are defined by an iterative process, for example, solving to a specified accuracy a differential equation where the differential equation or the data depends on several parameters. These simulations are not stochastic, but do tend to produce results that are generally considered noisy. We believe the noise in this type of simulation is better modeled by a function with both high-frequency and low-frequency oscillations.

As a example of this kind of deterministic noise, consider numerical evaluation of the integral

$$f(a) = \int_{-1}^{5} x^2 \sin^2(ax)dx,$$

for a parameter a varying over the interval $[-1.5, 1.5]$. By $\tilde{f}$ we denote the numerical value obtained in MATLAB using adaptive quadrature with an error tolerance 10^{-3}. In Figure 2.4 we show the relative error $\left|\frac{f(a)-\tilde{f}(a)}{f(a)}\right|$ and note that it has both high frequency and low frequency oscillations. The resulting noise is deterministic and of magnitude 10^{-3} and hence $\tilde{f}$ should only be trusted to three digits.

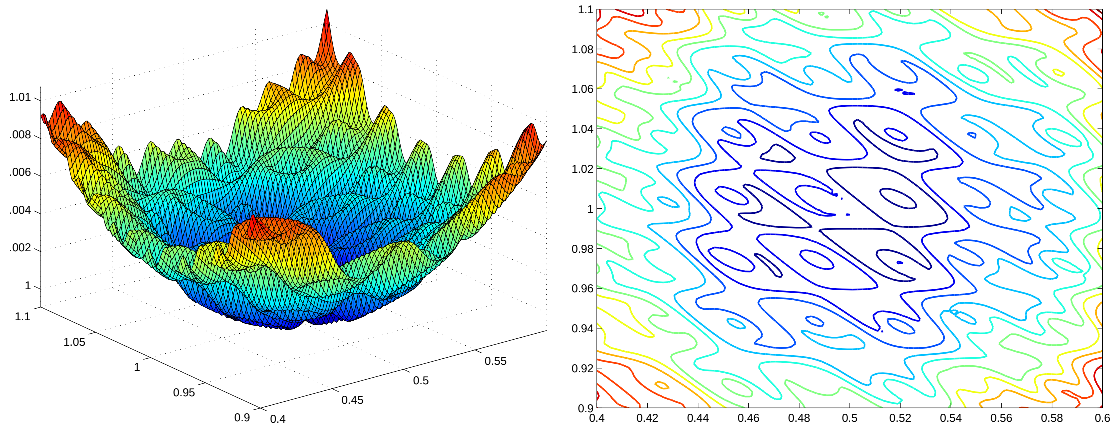


Figure 2.5: Plots of the noisy quadratic (2.19) on the box $[0.4, 0.6] \times [0.9, 1.1]$. Surface plots (left) and level sets (right) show the oscillatory nature of f.

Motivated by this type of noise, we defined the *noisy* problems $\mathcal{P}_N$ by

$$f(x) = (1 + \varepsilon_F \phi(x)) \sum_{k=1}^{m} f_k(x)^2, \tag{2.16}$$

with ε_F is the relative noise level and the noise function $\phi : \mathbb{R}^n \mapsto [-1, 1]$ is defined in terms of the cubic Chebyshev polynomial T_3 by

$$\phi(x) = T_3(\phi_0(x)), \qquad T_3(\alpha) = \alpha(4\alpha^2 - 3), \tag{2.17}$$

where

$$\phi_0(x) = 0.9 \sin(100\|x\|_1) \cos(100\|x\|_\infty) + 0.1 \cos(\|x\|_2). \tag{2.18}$$

The function ϕ_0 defined by (2.18) is continuous and piecewise continuously differentiable with $2^n n!$ regions where ϕ_0 is continuously differentiable. The composition of ϕ_0 with T_3 eliminates the periodicity properties of ϕ_0 and adds stationary points to ϕ at any point where ϕ_0 coincides with the stationary points $\left(\pm\frac{1}{2}\right)$ of T_3.

Figure 2.5 illustrates the properties of the noisy function (2.16) when the underlying smooth function ($\varepsilon_F = 0$) is a quadratic function. In this case

$$f(x) = \left(1 + \frac{1}{2}\|x - x_0\|^2\right)(1 + \varepsilon_F \phi(x)), \tag{2.19}$$

where $x_0 = \left[\frac{1}{2}, 1\right]$, and noise level $\varepsilon_F = 10^{-3}$. The graph on the left shows f on the two-dimensional box around x_0 and sides of length $\frac{1}{2}$, while the graph on the right shows the contours of f. Both graphs show the oscillatory nature of f, and that f seems to have local minimizers near the global minimizer. Evaluation of f on a mesh shows that, as expected, the minimal value of f is 0.99906, that is, $1 - \varepsilon_F$ to high accuracy.

Our interest centers on smooth and noisy problems, but we also wanted to study the behavior of derivative-free solvers on piecewise-smooth problems. An advantage of the benchmark problems $\mathcal{P}$ is that a set of *piecewise-smooth* problems $\mathcal{P}_{PS}$ can be easily derived by setting

$$f(x) = \sum_{k=1}^{m} |f_k(x)|. \tag{2.20}$$

These problems are continuous, but the gradient does not exist when $f_k(x) = 0$ and $\nabla f_k(x) \neq 0$ for some index k. They are twice continuously differentiable in the regions where all the f_k do not change sign. There is no guarantee that the problems in $\mathcal{P}_{PS}$ have a unique minimizer, even if (2.15) has a unique minimizer. However, we found that for all but six functions all local minimizers were in fact global minimizers. These six functions had only global minimizers, provided the variables were restricted to the positive orthant . Hence, for these six functions ($k_p = 8, 9, 13, 16, 17, 18$) the piecewise-smooth problems are defined by

$$f(x) = \sum_{k=1}^{m} |f_k(x_+)|, \tag{2.21}$$

where $x_+ = \max(x, 0)$. This function is piecewise-smooth and agrees with the function f defined by (2.20) for $x \geq 0$.

2.4 Computational Experiments

We now present the results of computational experiments with the performance measures introduced in Section 2.1. We used the solver set $\mathcal{S}$ consisting of the three algorithms detailed in Section 2.2 and the three problem sets $\mathcal{P}_S$, $\mathcal{P}_N$, and $\mathcal{P}_{PS}$ that correspond, respectively, to the smooth, noisy, and piecewise-smooth benchmark sets of Section 2.3.

The computational results center on the short-term behavior of derivative-free algorithms. We decided to investigate the behavior of the algorithms with a limit of 100 simplex gradients. Since the problems in our benchmark sets have at most 12 variables, we set $\mu_f = 1300$ so that all solvers can use at least 100 simplex gradients.

Data was obtained by recording, for each problem and solver $s \in \mathcal{S}$, the function values generated by the solver at each trial point. All termination tolerances were set as described in Section 2.2 so that solvers effectively terminate only when the limit μ_f on the number of function evaluations is exceeded. In the exceptional cases where the solver terminates early after $k < \mu_f$ function evaluations, we set all successive function values to $f(x_k)$. This data is then processed to obtain a history vector $h_s \in \mathbb{R}^{\mu_f}$ by setting

$$h_s(x_k) = \min \left\{ f(x_j) : 0 \leq j \leq k \right\},$$

so that $h_s(x_k)$ is the best function value produced by solver s after k function evaluations. Each solver produces one history vector for each problem, and these history vectors are gathered into a history array H, one column for each problem. For each problem, $p \in \mathcal{P}$, f_L was taken to be the best function value achieved by any solver within μ_f function evaluations, $f_L = \min_{s \in \mathcal{S}} h_s(x_{\mu_f})$.

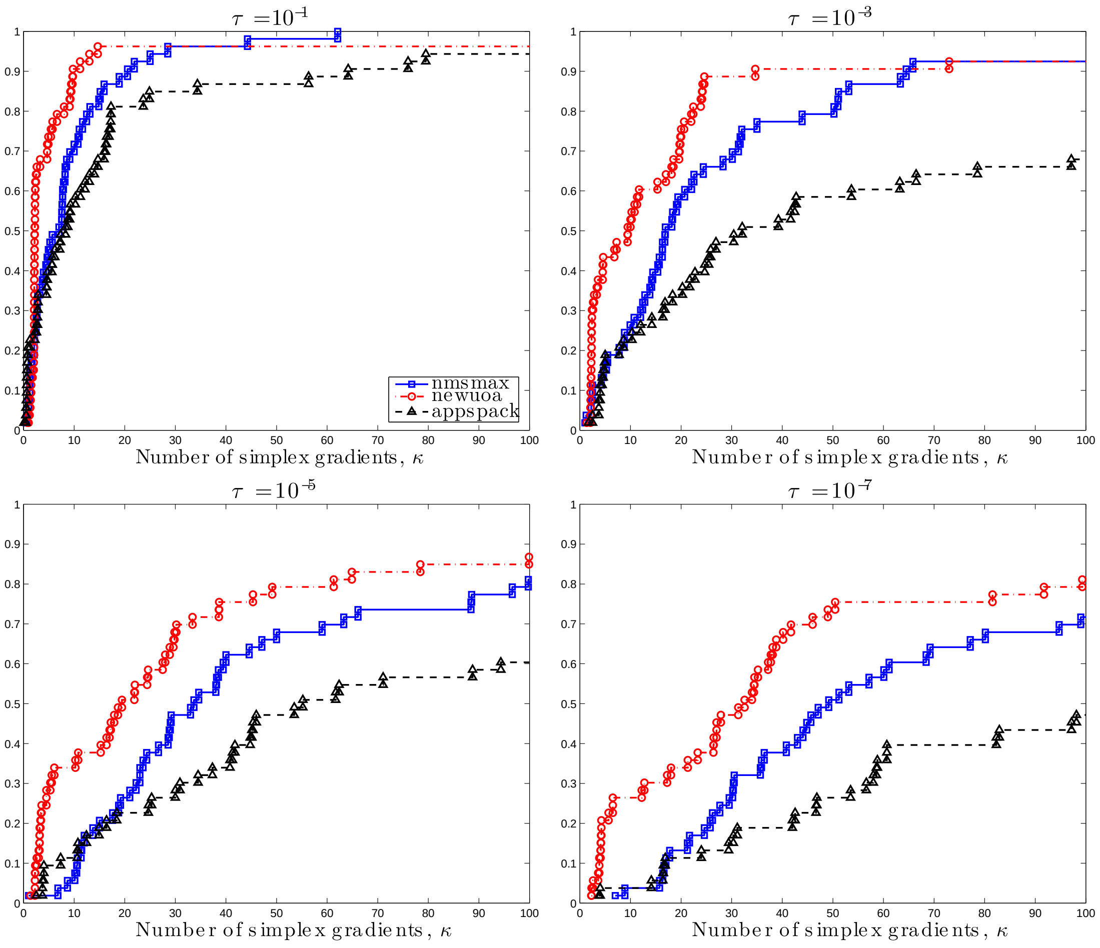


Figure 2.6: Data profiles $d_s(\kappa)$ for the smooth problems $\mathcal{P}_S$ show the percentage of problems solved as a function of a computational budget of simplex gradients.

We present the data profiles for $\tau = 10^{-k}$ with $k \in \{1, 3, 5, 7\}$ because we are interested in the short-term behavior of the algorithms as the accuracy level changes. We also present performance profiles for only $\tau = 10^{-k}$ with $k \in \{1, 3, 5, 7\}$. The interested reader will find a comprehensive set of results provided at `www.mcs.anl.gov/~more/dfo`.

We comment only on the results for an accuracy level of $\tau = 10^{-5}$ and use the other plots to indicate how the results change as τ changes. This accuracy level is mild compared to classical convergence tests based on the gradient. We support this claim by noting that (2.10) implies that if x satisfies the convergence test (2.4) near a minimizer x^*, then

$$\|\nabla f(x)\|_* \leq 0.45 \cdot 10^{-2} \, (f(x_0) - f(x^*))^{1/2}$$

for $\tau = 10^{-5}$ and for the norm $\| \cdot \|_*$ defined in Theorem 2.1. If the problem is scaled so that $f(x^*) = 0$ and $f(x_0) = 1$, then

$$\|\nabla f(x)\|_* \leq 0.45 \cdot 10^{-2}.$$

This test is not comparable to a gradient test that uses an unscaled norm. It suggests, however, that for well-scaled problems, the accuracy level $\tau = 10^{-5}$ is mild compared to that of classical convergence tests.

2.4.1 Smooth Problems

The data profiles in Figure 2.6 show that NEWUOA solves the largest percentage of problems for all sizes of the computational budget and levels of accuracy τ. This result is perhaps not surprising because NEWUOA is a model-based method based on a quadratic approximation of the function, and thus could be expected to perform well on smooth problems. However, the performance differences are noteworthy.

Performance differences between the solvers tend to be larger when the computational budget is small. For example, with a budget of 10 simplex gradients and $\tau = 10^{-5}$, NEWUOA solves almost 35% of the problems, while both NMSMAX and APPSPACK solve roughly 10% of the problems. Performance differences between NEWUOA and NMSMAX tend to be smaller for larger computational budgets. For example, with a budget of 100 simplex gradients, the performance difference be-

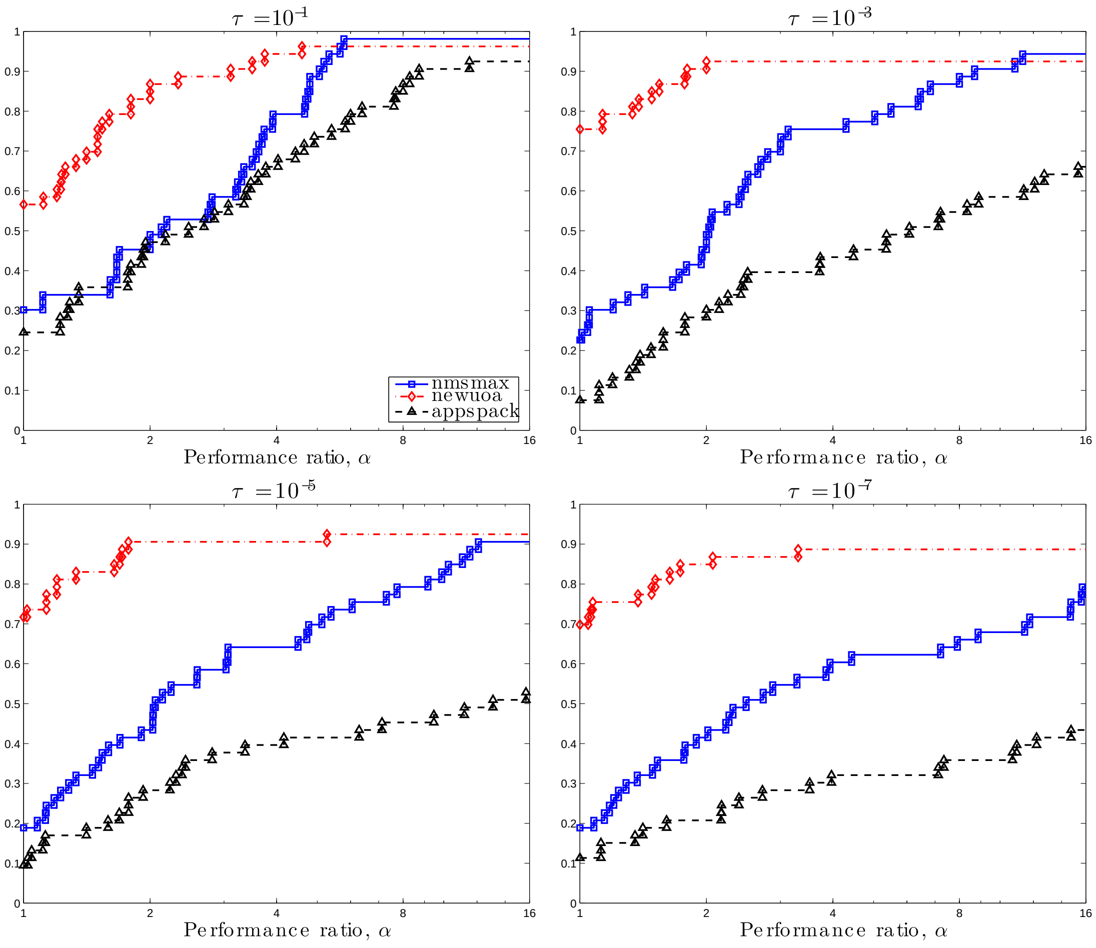


Figure 2.7: Performance profiles $\rho_s(\alpha)$ (logarithmic scale) for the smooth problems $\mathcal{P}_S$.

tween NEWUOA and NMSMAX is less than 10%. On the other hand, the difference between NEWUOA and APPSPACK is more than 25%.

A benefit of the data profiles is that they can be useful for allocating a computational budget. For example, if a user is interested in getting an accuracy level of $\tau = 10^{-5}$ on at least 50% of problems, the data profiles show that NEWUOA, NMSMAX, and APPSPACK would require 20, 35, and 55 simplex gradients, respectively. This kind of information is not available from performance profiles because they rely on performance ratios.

The performance profiles in Figure 2.7 are for the smooth problems with a logarithmic scale. Performance differences are also of interest in this case. In particular, we note that both of these plots show that NEWUOA is the fastest solver in at least 55% of the problems, while NMSMAX and APPSPACK are each the fastest solvers on fewer than 30% of the problems.

Both plots in Figure 2.7 show that the performance difference between solvers decreases as the performance ratio increases. Since these figures are on a logarithmic scale, however, the decrease is slow. For example, both plots show a performance difference between NEWUOA and NMSMAX of at least 40% when the performance ratio is two. This implies that for at least 40% of the problems NMSMAX takes at least twice as many function evaluations to solve these problems. When $\tau = 10^{-5}$, the performance difference between NEWUOA and APPSPACK is larger, at least 50%.

2.4.2 Noisy Problems

We now present the computational results for the noisy problems $\mathcal{P}_N$ as defined in Section 2.3. We used the noise level $\varepsilon_F = 10^{-3}$ with the non-stochastic noise function ϕ defined by (2.17,2.18). We consider this level of noise to be about right for simulations controlled by iterative solvers because tolerances in these solvers are likely to be on the order of 10^{-3} or smaller. Smaller noise levels are also of interest. For example, a noise level of 10^{-7} is appropriate for single-precision computations.

Arguments for a non-stochastic noise function were presented in Section 2.3, but here we add that a significant advantage of using a non-stochastic noise function in benchmarking is that this guarantees that the computational results are

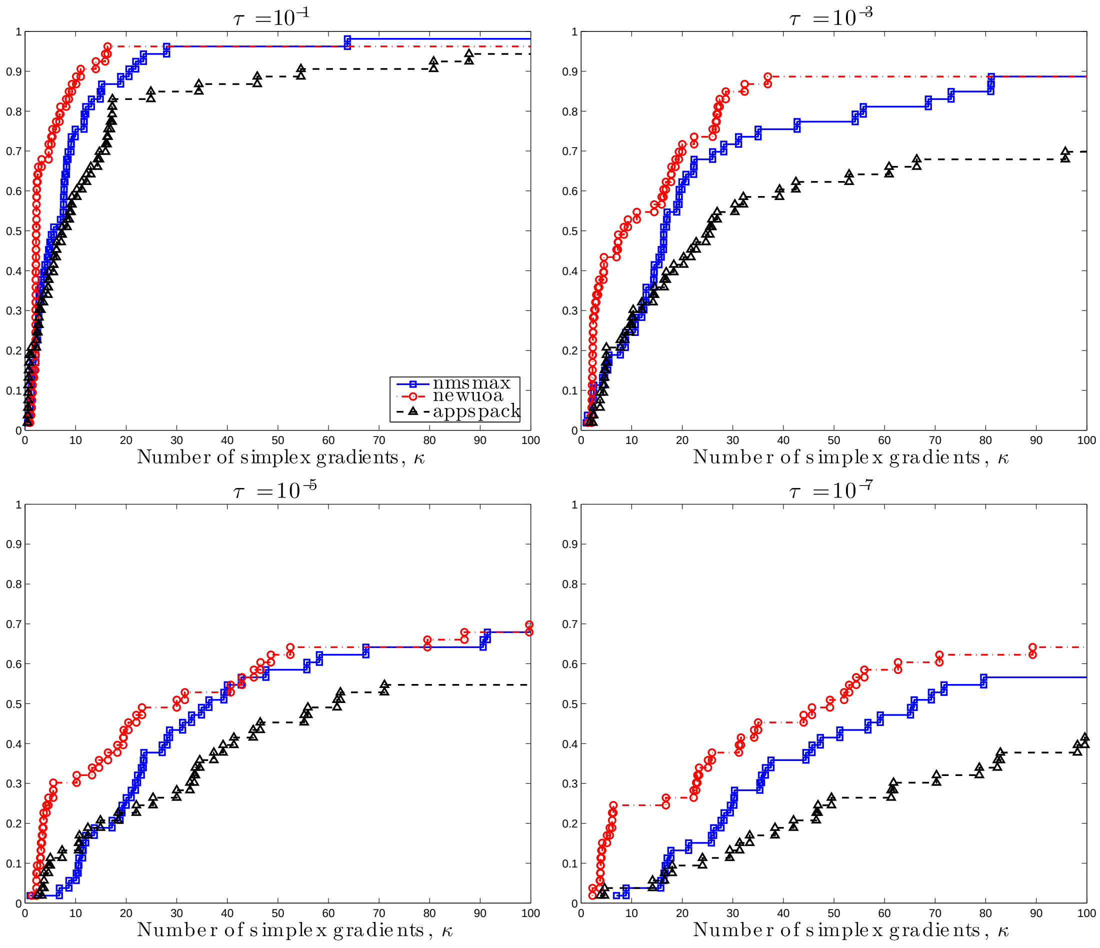


Figure 2.8: Data profiles $d_s(\kappa)$ for the noisy problems $\mathcal{P}_N$ show the percentage of problems solved as a function of a computational budget of simplex gradients.

reproducible up to the precision of the computations. We also note that the results obtained with a noise function ϕ defined by a random number generator are similar to those obtained by the ϕ defined by (2.17,2.18). For these (stochastic) noisy problems $\mathcal{P}_N^s$, we have replaced $\phi(x)$ in (2.16) by a random variable generated uniformly in $[-\varepsilon_F, \varepsilon_F]$. The downside of this definition is that the corresponding function is now stochastic, returning a possibly different output $f(x)$ when evaluated at a point x.

The performance profiles and data profiles for these problems are provided in

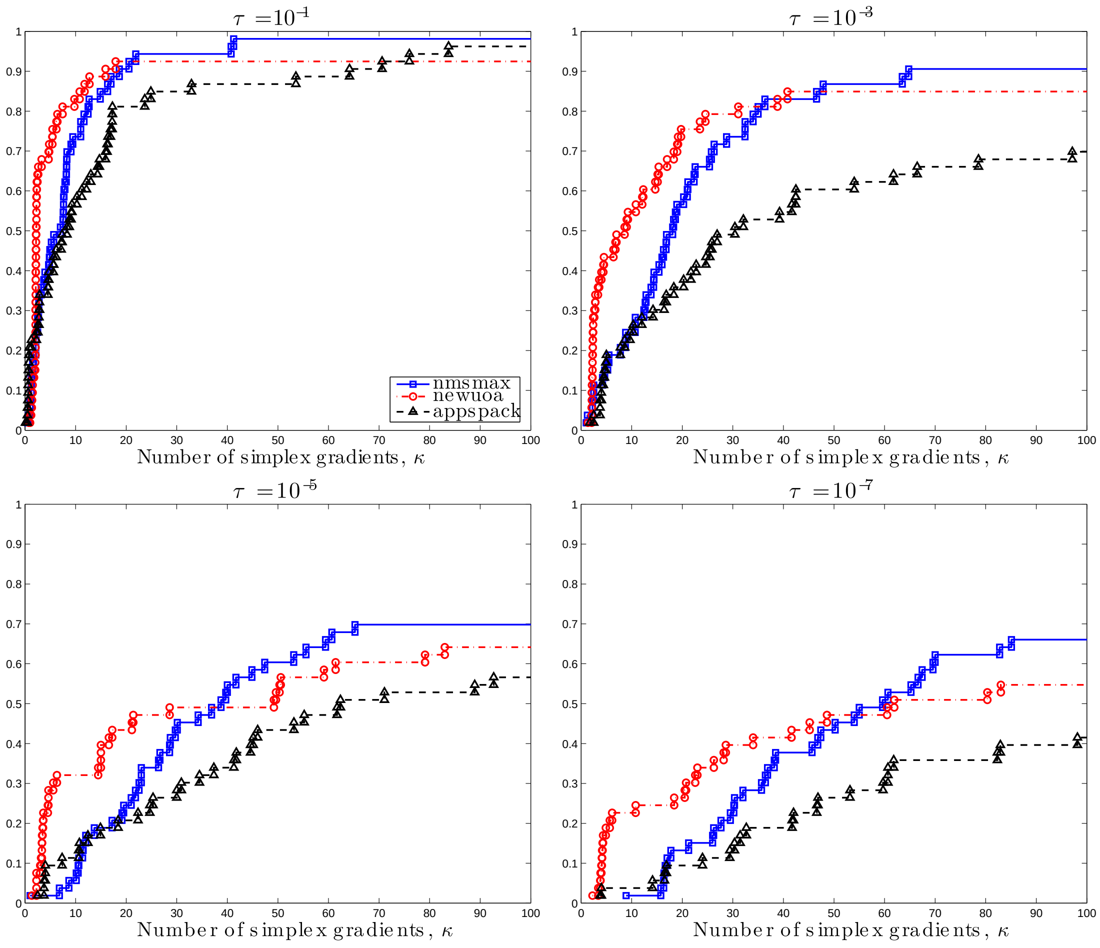


Figure 2.9: Data profiles $d_s(\kappa)$ for the (stochastic) noisy problems $\mathcal{P}_N^s$ show the percentage of problems solved as a function of a computational budget of simplex gradients.

Figures 2.8 and 2.10, respectively. Here we primarily note that these figures resemble their analogs for the (deterministic) noisy problems $\mathcal{P}_N$ in Figures 2.8 and 2.10. In both sets of profiles, we note that NEWUOA performs relatively worse on the stochastic problems for high levels of accuracy because it is relying on interpolation models fit to inconsistent data. As discussed in Section 2.3, our emphasis is on noise arising from deterministic iterative processes and hence we focus on the deterministic noisy problems $\mathcal{P}_N$.

The data profiles for the noisy problems, shown in Figure 2.8, are surprisingly similar to those obtained for the smooth problems. The degree of similarity between Figures 2.6 and 2.8 is much higher for small computational budgets and the smaller values of τ. This similarity is to be expected for direct search algorithms because the behavior of these algorithm depends only on logical comparisons between function values, and not on the actual function values. On the other hand, the behavior of NEWUOA is affected by noise because the model is determined by interpolating points and is hence sensitive to changes in the function values. Since NEWUOA depends on consistent function values, a performance drop can be expected for stochastic noise of magnitudes near a demanded accuracy level.

An interesting difference between the data profiles for the smooth and noisy problems is that solver performances for large computational budgets tend to be closer than in the smooth case. However, NEWUOA still manages to solve the largest percentage of problems for virtually all sizes of the computational budget and levels of accuracy τ.

Little similarity exists between the performance profiles for the noisy problems $\mathcal{P}_N$ when $\tau = 10^{-5}$, shown in Figure 2.10 and those for the smooth problems. In general these plots show that, as expected, noisy problems are harder to solve. For $\tau = 10^{-5}$, NEWUOA is the fastest solver on about 60% of the noisy problems, while it was the fastest solver on about 70% of the smooth problems. However, the performance differences between the solvers are about the same. In particular, both plots in Figure 2.10 show a performance difference between NEWUOA and NMSMAX of about 30% when the performance ratio is two. As we pointed out earlier, performance differences are an estimate of the gains that can be obtained when choosing a different solver.

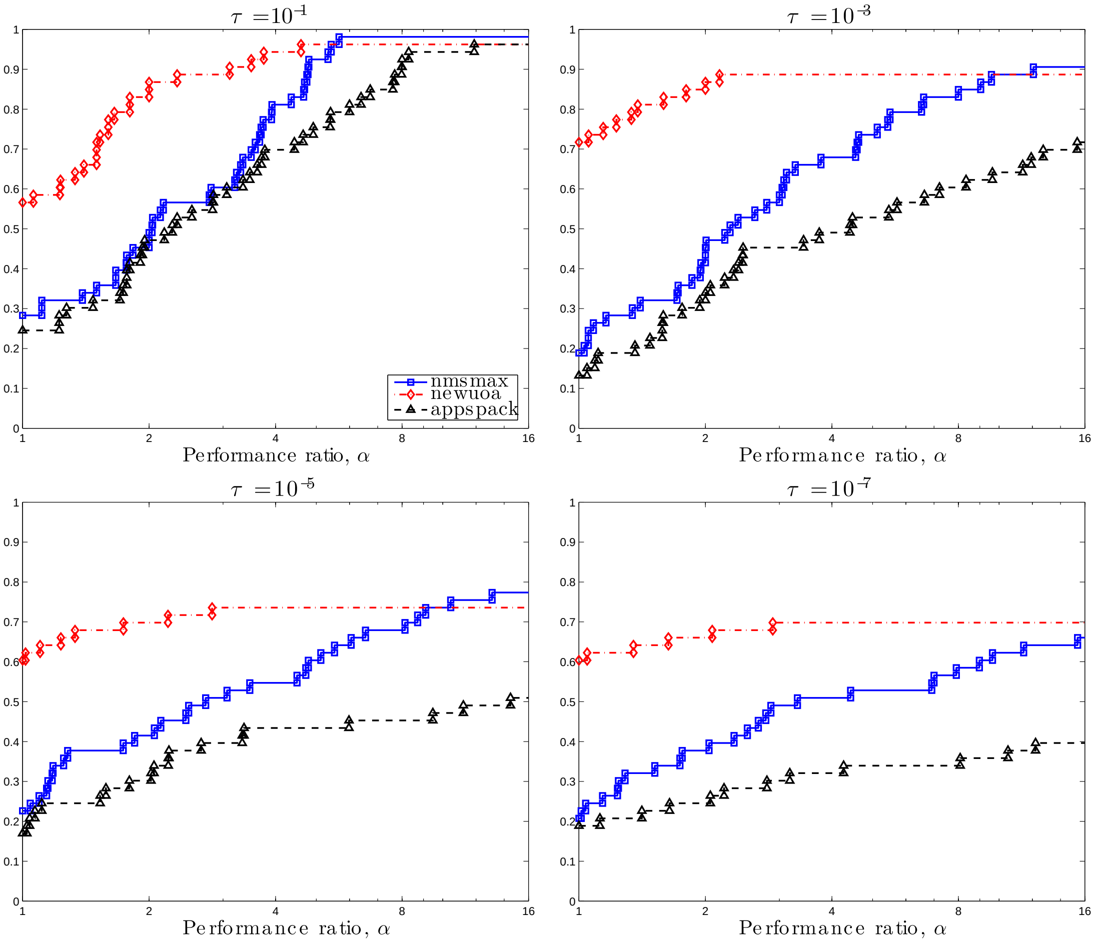


Figure 2.10: Performance profiles $\rho_s(\alpha)$ (logarithmic scale) for the noisy problems $\mathcal{P}_N$.

2.4.3 Piecewise-Smooth Problems

The computational experiments for the piecewise-smooth problems $\mathcal{P}_{PS}$ measure how the solvers perform in the presence of non-differentiable kinks. There is no guarantee of convergence for the tested methods in this case. We note that recent work has focused on relaxing the assumptions of differentiability [3].

The data profiles for the piecewise-smooth problems, shown in Figure 2.11, show that these problems are more difficult to solve than the noisy problems $\mathcal{P}_N$

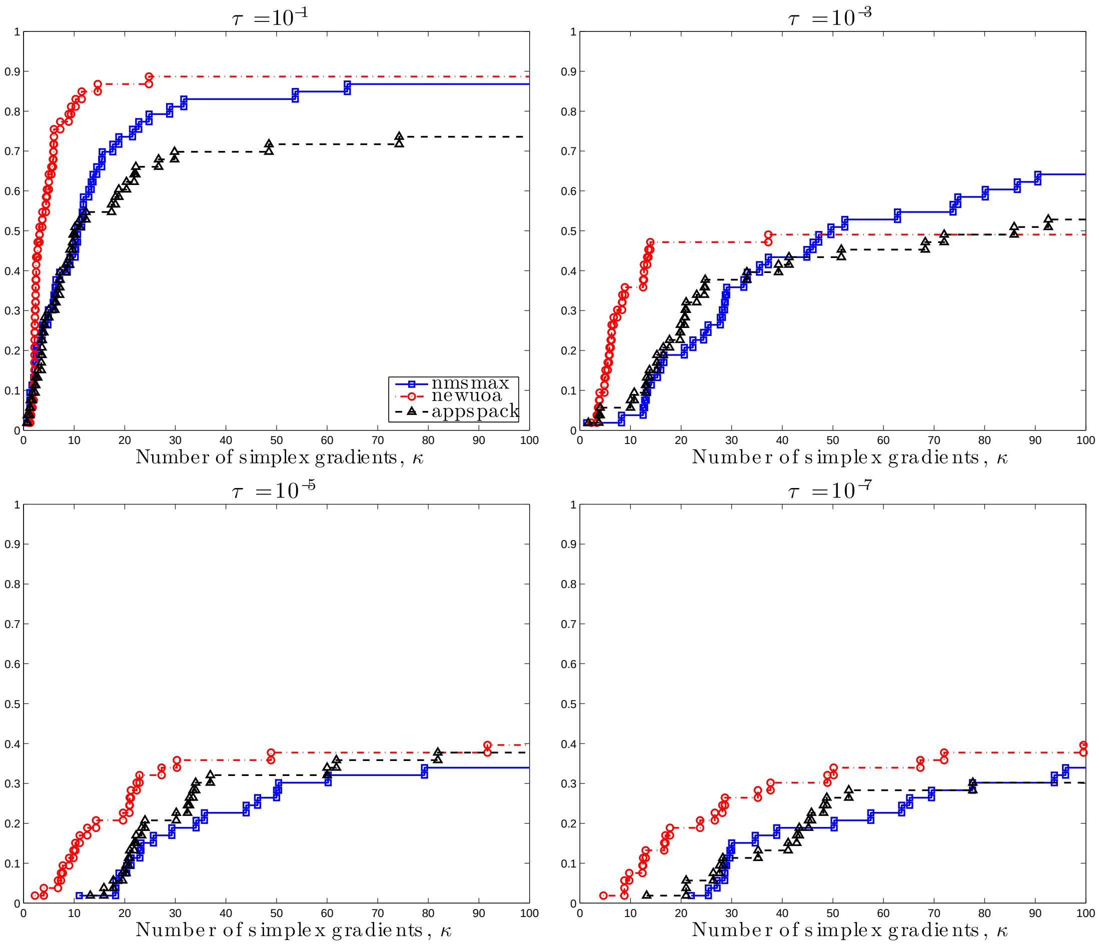


Figure 2.11: Data profiles $d_s(\kappa)$ for the piecewise-smooth problems $\mathcal{P}_{PS}$ show the percentage of problems solved as a function of a computational budget of simplex gradients.

and the smooth problems $\mathcal{P}_S$. In particular, we note that no solver is able to solve more than 40% of the problems in $\mathcal{P}_{PS}$ with a computational budget of 100 simplex gradients and $\tau = 10^{-5}$. By contrast, almost 70% of the noisy problems in $\mathcal{P}_N$ and 90% of the smooth problems in $\mathcal{P}_S$ can be solved with this budget and level of accuracy. Differences in performance are also smaller for the piecewise smooth problems. NEWUOA solves the most problems in almost all cases, but the performance difference between NEWUOA and the other solvers is smaller than in the noisy or smooth problems.

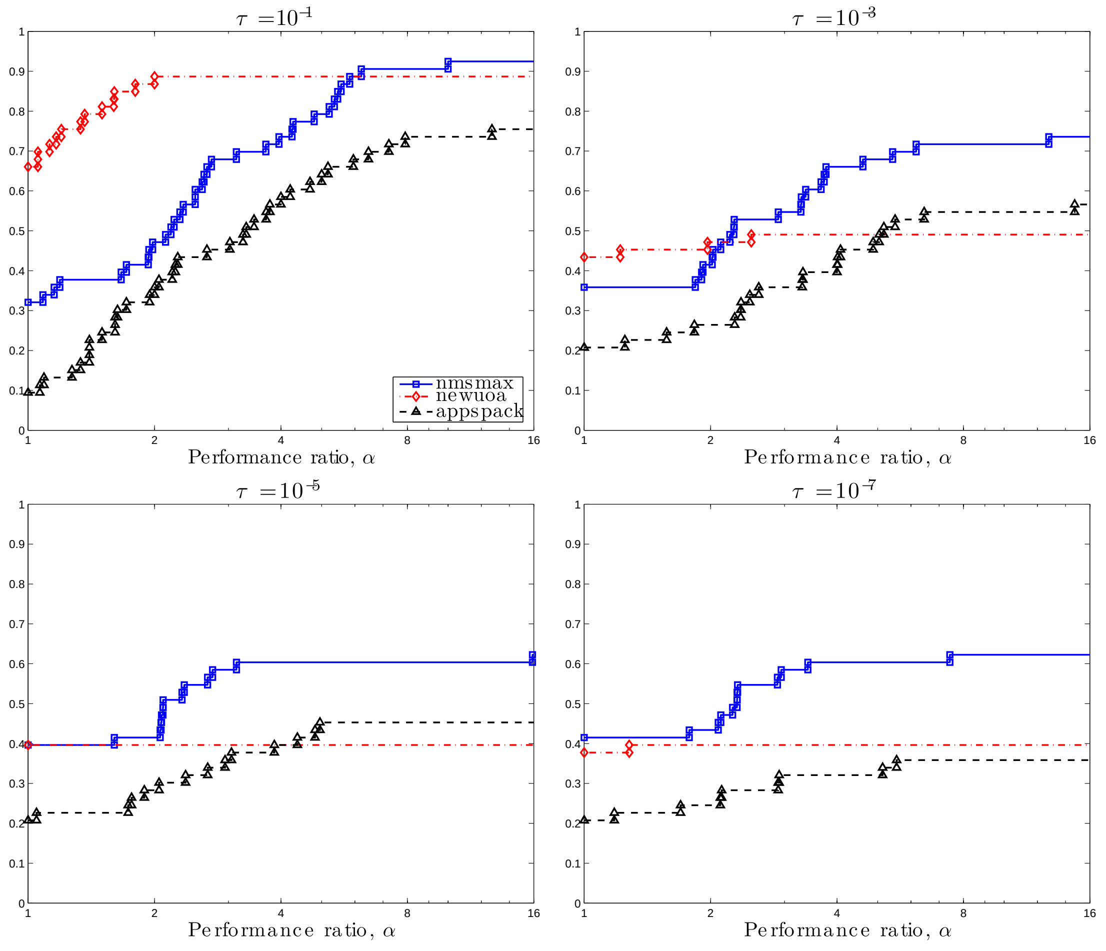


Figure 2.12: Performance profiles $\rho_s(\alpha)$ (logarithmic scale) for the piecewise-smooth problems $\mathcal{P}_{PS}$.

Another interesting observation on the data profiles is that APPSPACK solves more problems than NMSMAX with $\tau = 10^{-5}$ for all sizes of the computational budget. This in contrast to the results for smooth and noisy problem where NMSMAX solved more problems than APPSPACK.

The performance profiles for the piecewise-smooth problems $\mathcal{P}_{PS}$ appear in Figure 2.12. The results for $\tau = 10^{-5}$ show that NEWUOA, NMSMAX, and APPSPACK are the fastest solvers on roughly 50%, 30%, and 20% of the problems, respectively. This performance difference is maintained until the performance ratio is near $r = 2$.

The same behavior can be seen in the performance profile with $\tau = 10^{-1}$, but now the initial difference in performance is larger, more than 40%. Also note that for $\tau = 10^{-5}$ NEWUOA either solves the problem quickly or does not solve the problem within μ_f evaluations. On the other hand, the reliability of both NMSMAX and APPSPACK increases with the performance ratio, and NMSMAX eventually solves more problems than NEWUOA.

Finally, note that the performance profiles with $\tau = 10^{-5}$ show that NMSMAX solves more problems than APPSPACK, while the data profiles in Figure 2.11 show that APPSPACK solves more problems than NMSMAX for a computational budget of k simplex gradients where $k \in [25, 100]$. As explained in Section 2.1, this reversal of solver preference can happen when there is a constraint on the computational budget.

2.5 Concluding Remarks

Our interest in derivative-free methods is motivated in large part by the computationally expensive optimization problems that arise in engineering and science problems, including those in DOE's SciDAC initiative. These applications give rise to the noisy optimization problems that have been the focus of this work.

We have used the convergence test (2.4) to define performance and data profiles for benchmarking unconstrained derivative-free optimization solvers. This convergence test relies only on the function values obtained by the solver and caters to users with an interest in the short-term behavior of the solver. Data profiles provide crucial information for users who are constrained by a computational budget and complement the measures of relative performance shown by performance plots.

Our computational experiments show that the performance of the three solvers considered varied from problem class to problem class, with the worst performance on the set of piecewise-smooth problems $\mathcal{P}_{PS}$. While NEWUOA generally outperformed the NMSMAX and APPSPACK implementations in our benchmarking environment, the latter two solvers may perform better in other environments. For example, our results did not take into account APPSPACK's ability to work in a parallel processing environment where concurrent function evaluations are possible.

This work can be extended in several directions. For example, data profiles can also be used to benchmark solvers that use derivative information. In this setting we could use a gradient-based convergence test or the convergence test (2.4). Below we outline four other possible future research directions.

Performance on larger problems. The computational experiments in Section 2.4 used problems with at most $n_p = 12$ variables. Performance of derivative-free solvers for larger problems is of interest, but this would require a different set of benchmark problems.

Performance on application problems. Our choice of noisy problems mimics simulations that are defined by an iterative process, for example, solving a set of differential equations to a specified accuracy. We plan to validate this claim in future work. Performance of derivative-free solvers on other classes of simulations is also of interest.

Performance of other derivative-free solvers. As mentioned before, our emphasis is on the benchmarking process, and thus no attempt was made to assemble a large collection of solvers. Users interested in the performance of other solvers can find additional results at `www.mcs.anl.gov/~more/dfo`. Results for additional solvers can be added easily.

Performance with respect to input and algorithmic parameters. Our computational experiments used default input and algorithmic parameters, but we are aware that performance can change for other choices. For example, our experience is that the performance of NMSMAX deteriorates significantly as the initial size of the simplex decreases.

Acknowledgments

The authors are grateful to Christine Shoemaker for useful conversations throughout the course of this work and to Josh Griffin and Luís Vicente for valuable comments on an earlier draft. Lastly, we are grateful for the developers of the freely available solvers we tested for providing us with source code and installation assistance: Genetha Gray and Tammy Kolda (APPSPACK[30]), Nick Higham (MDSMAX and NMSMAX [34]), Tim Kelley (nelder and imfil [41]), Michael Powell (NEWUOA [61] and UOBYQA [58]), and Ana Custódio and Luís Vicente (SID-PSM [19]).

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357 and by a DOE Computational Science Graduate Fellowship under grant number DE-FG02-97ER25308.

CHAPTER 3

OPTIMIZATION BY RADIAL BASIS FUNCTIONS IN TRUST-REGIONS*

In this chapter we address unconstrained local minimization,

$$\min_{x \in \mathbb{R}^n} f(x), \tag{3.1}$$

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 chapter, 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 [30, 44].

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 (3.1). However, in addition to the potential presence of computational noise, as

*THIS CHAPTER IS THE PAPER OF THE SAME TITLE [75] COAUTHORED BY ROMMEL G. REGIS AND CHRISTINE A. SHOEMAKER.

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 [13, 56, 58, 60]). However, they are often tailored to minimize the underlying computational complexity, as in [60], or to yield global convergence, as in [13]. 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 [61] and Oeuvray and Bierlaire [56], 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 [16].

To minimize computational overhead, in both [61] and [56], 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 [8]. Our algorithmic framework works for a wide variety of radial basis functions. While we have recently established a global convergence result in [76] (Chapter 4), the focus of this chapter is on implementation details and the success of ORBIT in practice.

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 3.1. In Section 3.2 we introduce interpolating models based on RBFs. The computational details of ORBIT are outlined in Section 3.3. In Section 3.4 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 3.5.

3.1 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 iterate x_k. We define this so-called *trust-region* for an implied (center, radius) pair $(x_k, \Delta_k > 0)$ as:

$$\mathcal{B}_k = \{x \in \mathbb{R}^n : \|x - x_k\|_k \leq \Delta_k\}, \tag{3.2}$$

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:

$$\min \{m_k(x_k + s) : x_k + s \in \mathcal{B}_k\}. \tag{3.3}$$

As an example, in the upper left of Figure 3.1 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 (3.3), the pair (x_k, Δ_k) is updated according to the ratio of actual to predicted improvement,

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}. \tag{3.4}$$

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 Algorithm 3.1. The design of the trust-region algorithm ensures that f is sampled only within the relaxed level set:

$$\mathcal{L}(x_0) = \{y \in \mathbb{R}^n : \|x - y\|_k \leq \Delta_{\max} \text{ for some } x \text{ with } f(x) \leq f(x_0)\}. \tag{3.5}$$

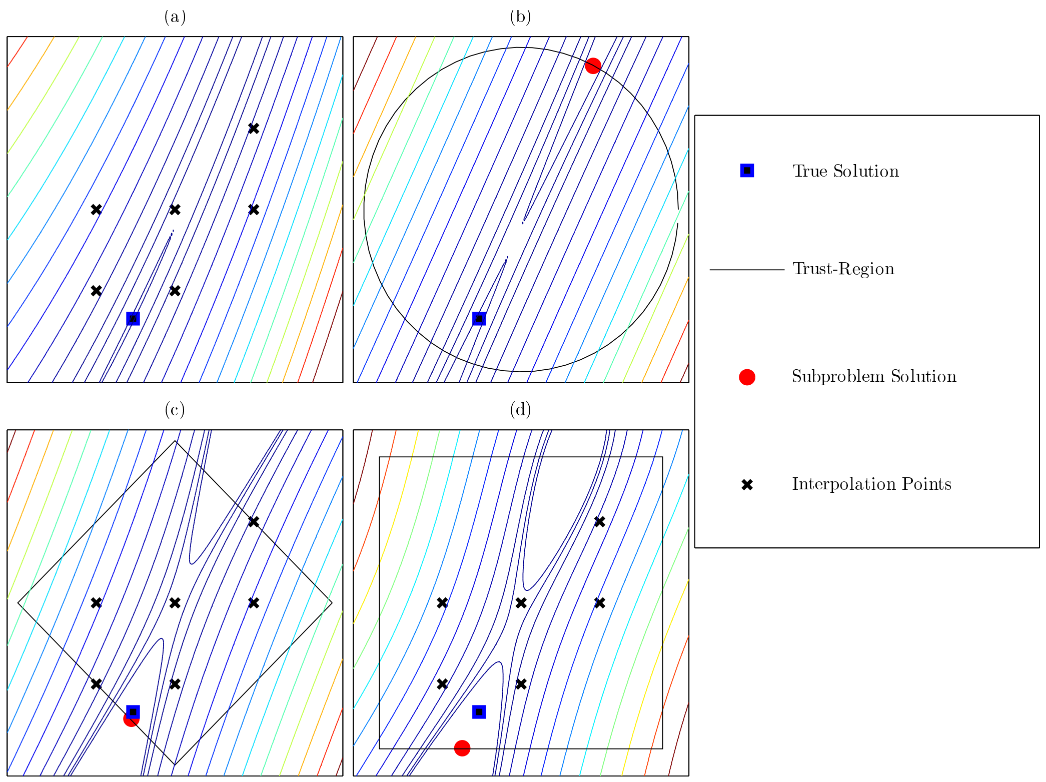


Figure 3.1: Trust-region subproblem solutions (all have same axes): (a) true function, (b) quadratic Taylor model in 2-norm trust-region, (c) quadratic interpolation model in 1-norm trust-region, (d) Gaussian RBF interpolation model in ∞-norm trust-region.

Usually a quadratic model,

$$m_k(x_k + s) = f(x_k) + g_k^T s + \frac{1}{2} s^T H_k s, \tag{3.6}$$

is employed and the approximate solution, s_k, to the subproblem (3.3) is required to satisfy a sufficient decrease condition of the form

$$m_k(x_k) - m_k(x_k + s_k) \geq \frac{\kappa_d}{2} \|g_k\|_k \min\left\{ \frac{\|g_k\|_k}{\|H_k\|_k}, \Delta_k \right\}, \tag{3.7}$$

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 (3.7) and the update of Δ_{k+1} in the basic

3.1.1. Build model m_k approximating f in the trust-region $\mathcal{B}_k$.
3.1.2. Solve subproblem (3.3).
3.1.3. Evaluate $f(x_k + s_k)$ and compute ρ_k using (3.4).
3.1.4. Adjust trust-region according to:

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

Algorithm 3.1: Iteration k of a basic trust-region algorithm.

algorithm in Algorithm 3.1. 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 [12]. In the derivative-free setting, other models must be constructed.

3.1.1 Derivative-Free Trust-Region Models

The quadratic model in (3.6) is attractive because, with it, the subproblem in (3.3) 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.

Table 3.1: 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

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

$$m_k(x_k + y_i) = f(x_k + y_i) \qquad \text{for all } y_i \in \mathcal{Y}. \tag{3.8}$$

The idea of forming quadratic models by interpolation for optimization without derivatives was proposed by Winfield in the late 1960's [77] and revived in the mid-1990's independently by Powell (UOBYQA [58]) and Conn, Scheinberg, and Toint (DFO [13]).

These methods rely heavily on results from multivariate interpolation, a problem much more difficult than its univariate counterpart [73]. 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 (3.8) to be well-posed. These geometric conditions have received recent treatment in [16], 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 3.1.

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 3.1 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 (3.8) uniquely by certain underdetermined quadratics [60]. 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 [61]. A similar approach is used in a later version of the DFO package [14].

3.1.2 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 [16] and note that three similar conditions define *fully quadratic* models.

Definition 3.1. *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* <u>*fully linear on* $\mathcal{B}$</u> *if for all* $x \in \mathcal{B}$*:*

$$
\begin{aligned}
|f(x) - m(x)| &\leq \kappa_f \Delta^2, & (3.9)\\
\|\nabla f(x) - \nabla m(x)\| &\leq \kappa_g \Delta. & (3.10)
\end{aligned}
$$

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 [16], fully linear interpolation models are defined by geometric conditions on the interpolation set. In Section 3.3 we will explore the conditions (3.9) and (3.10) for the radial basis function models introduced next.

3.2 Radial Basis Functions

Quadratic surrogates of the form (3.6) 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 (3.8) for the interpolation points $\{y_j\}_{j=1}^{|\mathcal{Y}|}$. One possible model is of the form

$$
m_k(x_k + s) = \sum_{j=1}^{|\mathcal{Y}|} \lambda_j \phi(\|s - y_j\|) + P(s), \tag{3.11}
$$

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 3.2.

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,

$$\nabla m_k(x_k+s)=\sum_{i=1}^{|\mathcal{Y}|} \lambda_i \phi'(\|s-y_i\|)\frac{s-y_i}{\|s-y_i\|}+\nabla P(s), \tag{3.12}$$

and Hessian, provided in (3.32), of the model.

In addition to being sufficiently smooth, these radial functions in Table 3.2 all share the property of *conditional positive definiteness* [73].

Definition 3.2. *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* <u>*Conditionally Positive Definite (CPD) of order d*</u> *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 (3.11) 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 $\mathcal{P}_{d-1}^n$* if the zero polynomial is the only polynomial in $\mathcal{P}_{d-1}^n$ which vanishes at all $\hat{p}$ points. Conditional positive definiteness of a function is usually proved by Fourier transforms [10, 73] 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 $\hat{d} \geq d$.

3.2.1 Obtaining Model Parameters

We now illustrate one method for obtaining the parameters defining an RBF model that interpolates data as in (3.8) 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:

$$\begin{bmatrix} \Phi & \Pi^T \\ \Pi & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \nu \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}. \tag{3.13}$$

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 $\text{rank}(\Pi) = \text{dim}\mathcal{P}_{d-1}^n = \hat{p}$. It is then easy to see that for CPD functions of order d, a sufficient condition for the nonsingularity of (3.13) is that the points in $\mathcal{Y}$ are distinct and yield a Π^T of full column rank.

Table 3.2: 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, $e^{-\frac{r^2}{\gamma^2}}$

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 [6]) for solving the symmetric system in (3.13). 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 (3.13), 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 (3.13) reduces to

$$Z^T \Phi Z \omega = Z^T f, \tag{3.14}$$

$$R\nu = Q^T(f - \Phi Z \omega). \tag{3.15}$$

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

$$Z^T \Phi Z = LL^T \tag{3.16}$$

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

$$\|\lambda\| = \left\| Z L^{-T} L^{-1} Z^T f \right\| \leq \left\| L^{-1} \right\|^2 \|f\|, \tag{3.17}$$

which will prove useful for the analysis in Section 3.3.2.

3.2.2 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 [10]. Their use to date has been mainly confined to global optimization [8, 33, 65]. 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 3.1. 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 [55]. His algorithm, BOOSTERS, was motivated by problems in the area of medical image registration and was subsequently modified to include gradient information when available [56]. Convergence theory was based on the literature available at the time [12].

3.3 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 Algorithm 3.2.

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:

$$d_i \in \mathcal{D}_k \iff f(x_k + d_i) \text{ is known.} \tag{3.18}$$

```
3.2.1: Find $n+1$ affinely independent points:
    AffPoints($\mathcal{D}_k, \theta_0, \theta_1, \Delta_k$) (detailed in Algorithm 3.3)
3.2.2: Add up to $p_{\max} - n - 1$ additional points to $\mathcal{Y}$:
    AddPoints($\mathcal{D}_k$,$\theta_2$,$p_{\max}$)
3.2.3: Obtain RBF model parameters from (3.14) and (3.15).
3.2.4: 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 $\Delta_k = \frac{\epsilon_g}{2\kappa_g}$.
3.2.5: Approximately solve subproblem (3.3) to obtain a step $s_k$ satisfying (3.35),
    Evaluate $f(x_k + s_k)$.
3.2.6: Update trust-region parameters:
```

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

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

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

Algorithm 3.2: Iteration k of the ORBIT algorithm.

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 [56, 58, 61], 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 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 3.2.1 of Algorithm 3.2, 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 3.3.1. Additional points are added to $\mathcal{Y}$ in Step 3.2.2 (discussed in Section 3.3.2) 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 3.2.3 using the previously discussed solution techniques.

In Step 3.2.4 a termination criteria is checked. If the model gradient is small enough, the method detailed in Section 3.3.1 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 (3.10):

$$\|\nabla f(x_k)\| \leq \|\nabla m_k(x_k)\| + \|\nabla f(x_k) - \nabla m_k(x_k)\| \leq \frac{\epsilon_g}{2} + \kappa_g \left(\frac{\epsilon_g}{2\kappa_g}\right) = \epsilon_g \quad (3.19)$$

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 3.2.5 as discussed in Section 3.3.3. In Step 3.2.6, the trust-region parameters are updated. The given procedure coincides with the derivative-based procedure in Algorithm 3.1 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 3.2.7 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.

3.3.1 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 (3.11) 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

$$\Pi = \begin{bmatrix} y_2 & \cdots & y_{|\mathcal{Y}|} & 0 \\ 1 & \cdots & 1 & 1 \end{bmatrix} = \begin{bmatrix} Y & 0 \\ e^T & 1 \end{bmatrix}, \tag{3.20}$$

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 [16] and is proved in Chapter 4 and [76].

Lemma 3.1. *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 (3.8) at a set of points $\mathcal{Y} = \{y_1 = 0, y_2, \ldots, y_{n+1}\} \subseteq \mathcal{B} - x_k$ such that $\|Y^{-1}\| \leq \frac{\Lambda_Y}{c_k \Delta}$, where c_k (introduced in Section 3.1) 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 3.1 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 (3.9) and (3.10) 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 3.2. *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 $\{y_2, \dots, y_{n+1}\} \subseteq \mathcal{B} - x_k$, 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

$$\sigma_n \sigma_1^{n-1} \geq \prod_{i=1}^{n} \sigma_i = |\det(\hat{Y})| = |\det(R)| = \prod_{i=1}^{n} |r_{ii}|. \tag{3.21}$$

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

$$\left\| Y^{-1} \right\| = \frac{1}{c_k \Delta} \left\| \hat{Y}^{-1} \right\| = \frac{1}{c_k \Delta} \frac{1}{\sigma_n} \leq \frac{1}{c_k \Delta} \frac{n^{\frac{n-1}{2}}}{\theta_1^n}. \tag{3.22}$$

□

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 3.1 and 3.2 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 3.1.

The subroutine AffPoints given in Algorithm 3.3 details our method of constructing a model which is fully linear on $\mathcal{B}_k$. We note that the projections in

```
3.3.0. Input $\mathcal{D} = \{d_1, \dots, d_{|\mathcal{D}|}\} \subset \mathbb{R}^n$, constants $\theta_0 \geq 1$, $\theta_1 \in (0, \theta_0^{-1}]$, $\Delta \in (0, \Delta_{\max}]$.
3.3.1. Initialize $\mathcal{Y} = \{0\}$, $Z = I_n$.
3.3.2. For all $d_j \in \mathcal{D}$ such that $\|d_j\|_k \leq \theta_0 \Delta$ (if no such $d_j$, continue to 3.3.3b):
    If $\left|\mathrm{proj}_Z\left(\frac{1}{\theta_0 \Delta} d_j\right)\right| \geq \theta_1$,
        $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{d_j\}$,
        Update $Z$ to be an orthonormal basis for $\mathcal{N}\left([y_1 \cdots y_{|\mathcal{Y}|}]\right)$.
3.3.3a. If $|\mathcal{Y}| = n + 1$, set linear=true.
3.3.3b. 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 $\left|\mathrm{proj}_Z\left(\frac{1}{\theta_0 \Delta} d_j\right)\right| \geq \theta_1$,
            $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{d_j\}$,
            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 $Z$,
        $\mathcal{Y} \leftarrow \mathcal{Y} \cup Z$.
```

Algorithm 3.3: AffPoints($\mathcal{D}, \theta_0, \theta_1, \Delta$): Algorithm for obtaining fully linear models.

Steps 3.3.2 and 3.3.3b 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 3.1 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 3.2 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 (3.15) to be numerically stable, the matrix Π^T defined in (3.20) must be well-conditioned. In particular we note that

$$\Pi^{-T} = \begin{bmatrix} Y^{-T} & -Y^{-T}e \\ 0 & 1 \end{bmatrix}, \tag{3.23}$$

and hence

$$\left\|\Pi^{-T}\right\| \leq \left\|Y^{-1}\right\| \sqrt{n+1} + 1 \tag{3.24}$$

provides an easily obtainable bound based on $\|Y^{-1}\|$. 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 Algorithm 3.3 yields an interpolation set of $n+1$ points suitably poised for linear interpolation.

3.3.2 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 (3.13), and hence the RBF model in (3.11) 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 3.1, 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 (3.14). In [55] 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 (3.13). ORBIT's method largely follows the development in [8] and directly addresses the conditioning of the system used by our solution techniques.

Employing the notation of Section 3.2.1, 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:

$$\Phi_y = \begin{bmatrix} \Phi & \phi_y \\ \phi_y^T & \phi(0) \end{bmatrix}, \qquad \Pi_y^T = \begin{bmatrix} \Pi^T \\ \pi(y) \end{bmatrix}. \tag{3.25}$$

As suggested by Lemmas 3.1 and 3.2, 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, eg.- $\pi\left(\frac{y}{\Delta_k}\right)$. This scaling does not affect the analysis of the algorithm using linear tails in Chapter 4 and [76].

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

$$Z_y = \begin{bmatrix} Z & Q\tilde{g} \\ 0 & \hat{g} \end{bmatrix}, \tag{3.26}$$

where, as in Section 3.2.1, Z is any orthogonal basis for $\mathcal{N}(\Pi)$. Hence, $Z_y^T \Phi Z_y$ is of the form

$$Z_y^T \Phi Z_y = \begin{bmatrix} Z^T \Phi Z & v \\ v^T & \sigma \end{bmatrix}, \tag{3.27}$$

and it can easily be shown that

$$L_y^T = \begin{bmatrix} L^T & L^{-1}v \\ 0 & \sqrt{\sigma - \|L^{-1}v\|^2} \end{bmatrix}, \qquad L_y^{-T} = \begin{bmatrix} L^{-T} & \frac{-L^{-T}L^{-1}v}{\sqrt{\sigma - \|L^{-1}v\|^2}} \\ 0 & \frac{1}{\sqrt{\sigma - \|L^{-1}v\|^2}} \end{bmatrix} \tag{3.28}$$

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

$$\begin{aligned} v &= Z^T\left(\Phi Q\tilde{g} + \phi_y \hat{g}\right), & (3.29) \\ \sigma &= \tilde{g}^T Q^T \Phi Q \tilde{g} + 2\tilde{g}^T Q^T \phi_y \hat{g} + \phi(0)\hat{g}^2. & (3.30) \end{aligned}$$

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:

$$\|v\| \leq \sqrt{|\mathcal{Y}|(|\mathcal{Y}|+1)}\max\{|\phi(r)| : r \in [0, 4c_k\Delta_{\max}]\}. \tag{3.31}$$

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 (3.17). 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 3.1.

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

$$\nabla^2 m_k(x_k + s) = \sum_{i=1}^{|\mathcal{Y}|} \lambda_i \left[\frac{\phi'(\|z_i\|)}{\|z_i\|} I_n + \left(\phi''(\|z_i\|) - \frac{\phi'(\|z_i\|)}{\|z_i\|} \right) \frac{z_i}{\|z_i\|} \frac{z_i^T}{\|z_i\|} \right], \quad (3.32)$$

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

$$b_2(\overline{\Delta}) = \max \left\{ 2 \left| \frac{\phi'(r)}{r} \right| + |\phi''(r)| : r \in [0, \overline{\Delta}] \right\}, \quad (3.33)$$

which is again bounded whenever $\overline{\Delta}$ is for all of the radial functions considered in Table 3.2.

The following Lemma is a consequence of the preceding remarks and is proved formally in Chapter 4 and [76].

Lemma 3.3. *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 (3.11) interpolating f on $x_k + \mathcal{Y}$, we have that for all $x \in \mathcal{B}$:*

$$\left\|\nabla^2 m_k(x)\right\| \le |\mathcal{Y}| \left\|L^{-1}\right\|^2 b_2(4 c_k \Delta_{\max}) f_{\max} =: \kappa_H. \quad (3.34)$$

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

Algorithm 3.4: Backtracking algorithm for obtaining a sufficient decrease in Step 3.2.5 of Algorithm 3.2 ($\alpha \in (0,1)$, $\kappa_d \in (0,1]$).

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 3.1 for our RBF models.

3.3.3 Solving the Subproblem

The trust-region subproblem (3.3) is made considerably more difficult using the RBF model in (3.11). Given that the radial function ϕ is chosen from Table 3.2, 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 (3.3).

Since the RBF model may be multimodal, an optimal solution to (3.3), 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 (3.7) for some fixed $\kappa_d \in (0,1]$:

$$m_k(x_k) - m_k(x_k+s) \geq \frac{\kappa_d}{2}\|\nabla m_k(x_k)\| \min\left\{\frac{\|\nabla m_k(x_k)\|}{\kappa_H}, \frac{\|\nabla m_k(x_k)\|}{\|\nabla m_k(x_k)\|_k}\Delta_k\right\}. \tag{3.35}$$

Algorithm 3.4 gives a simple algorithm for backtracking line search in the direction of steepest descent. Since subproblem solutions are calculated in Algorithm 3.2 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 $\max \left\{ \log_\alpha \frac{2\Delta_k \kappa_H}{\epsilon_g}, 0 \right\}$ 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 $m_k(x_k + \tilde{s}) \leq m_k(x_k + \hat{s})$.

The sufficient decrease condition in (3.35) guarantees that we can efficiently obtain an approximate solution to the trust-region subproblem. Further, it allows us to establish the global convergence in Chapter 4 and [76] of ORBIT to first-order critical points satisfying $\nabla f(x_*) = 0$.

3.4 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

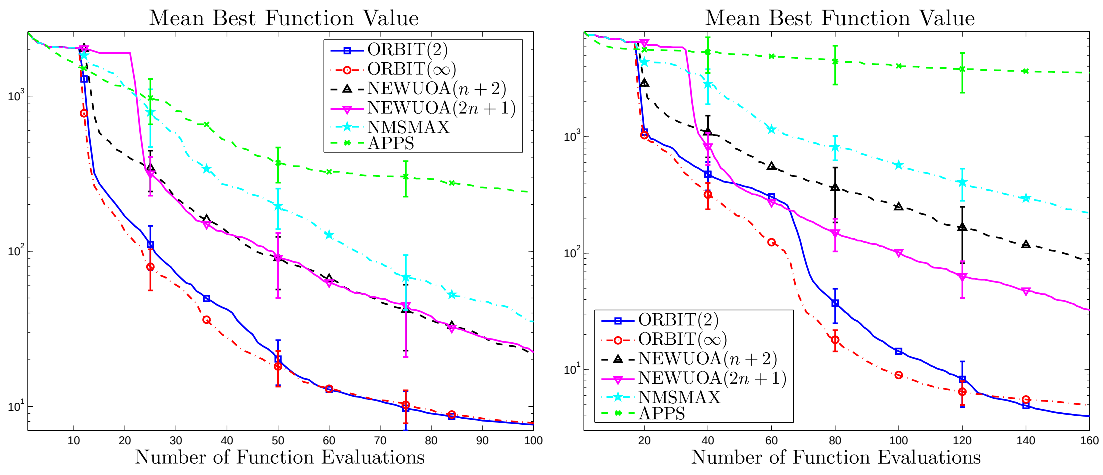


Figure 3.2: Average of 30 Starting Points on Two Functions from $\mathcal{P}$ (log_{10} scale, lowest line is best): (a) Extended Rosenbrock ($n = 10$); (b) Extended Powell Singular ($n = 16$).

algorithms for computationally expensive optimization are infrequent. In [55] and [56] the number of function evaluations needed to reach some convergence goal is reported, while in [26] and [65] means plots similar to those shown in Figure 3.2 are given.

Using 30 different starting points (see Section 3.4.3), Figure 3.2 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 using the same starting point. We now discuss one way to complement the means plots in Figure 3.2.

3.4.1 Performance and Data Profiles

In [21], 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

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}} \tag{3.36}$$

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:

$$\rho_s(\alpha) = \frac{1}{|\mathcal{P}|}\text{size}\Big\{p \in \mathcal{P} : r_{p,s} \leq \alpha\Big\}. \tag{3.37}$$

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 [52] (Chapter 2). The convergence test used there is

$$f(x_0) - f(x) \geq (1-\tau)(f(x_0) - f_L), \tag{3.38}$$

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 (3.38) 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 Chapter 2 and [52]. The data profile,

$$d_s(\kappa) = \frac{1}{|\mathcal{P}|}\text{size}\left\{p \in \mathcal{P} : \frac{t_{p,s}}{n_p + 1} \leq \kappa\right\}, \tag{3.39}$$

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 (3.38).

3.4.2 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 [55, 56] 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 [26]. APPSPACK is an asynchronous parallel pattern search method [30, 42], 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 [34], 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 [55, 56]. Powell's Fortran NEWUOA code [61] 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 (Algorithm 3.2) 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 Algorithm 3.4, we set $\kappa_d = 10^{-4}$ and $\alpha = .9$. In the ORBIT implementation tested here, we also relied on the FMINCON routine from MATLAB [68] which is based on a sequential quadratic programming (SQP) method.

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 only viable for sufficiently expensive objective functions.

Our choice of inputs ensures that ORBIT, NMSMAX, and NEWUOA initially

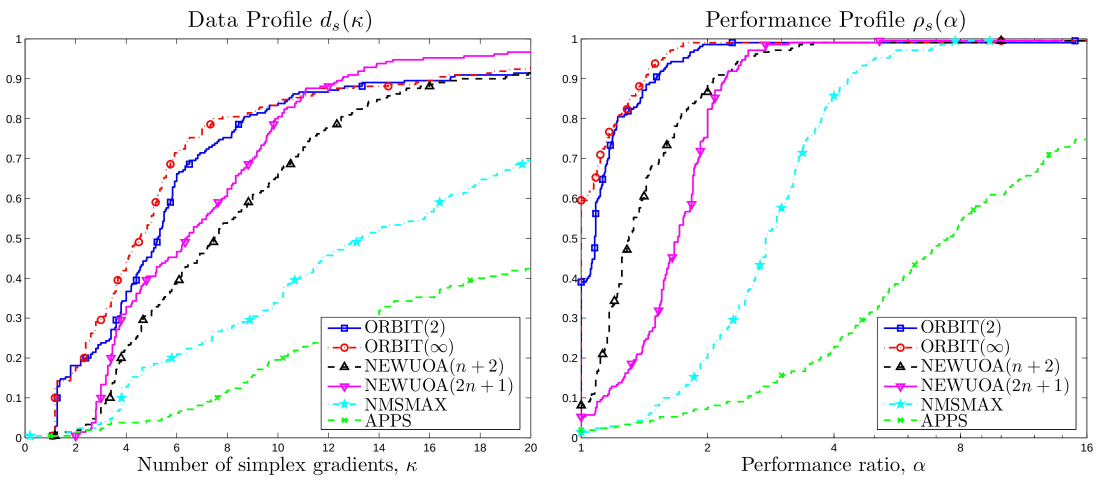


Figure 3.3: Set $\mathcal{P}$ of smooth test problems: (a) data profile $d_s(\kappa)$ shows the percentage of problems solved, $\tau = .01$; (b) performance profile $\rho_s(\alpha)$ shows the relative performance, $\tau = .1$.

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.

3.4.3 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 [50]: 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 3.3 lists the hypercubes, each chosen by the authors to contain the corresponding solution (see [50]) 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 Figure 3.2 (a) and (b), respectively.

Table 3.3: 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

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 3.3 (a). Here we see the ORBIT 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 3.3 (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 Chapter 2 and [52], which are "noisy" variants of a set of CUTEr problems [29]. The functions which these profile problems are based on are distinct

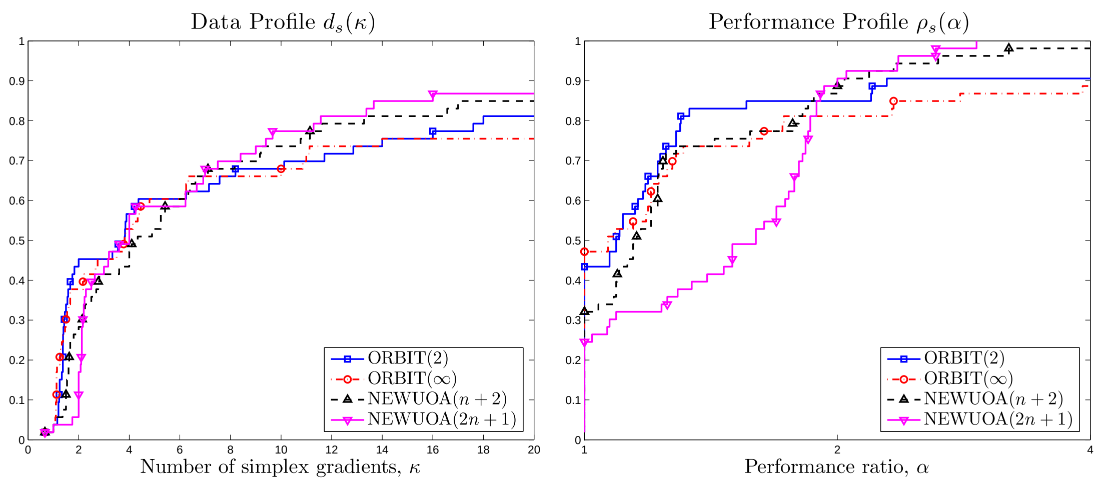


Figure 3.4: Set $\mathcal{P}_N$ of noisy test problems: (a) data profile $d_s(\kappa)$ shows the percentage of problems solved, $\tau = .01$; (b) performance profile $\rho_s(\alpha)$ shows the relative performance, $\tau = .1$.

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 Chapter 2 and [52].

In the data profiles for $\tau = .01$ shown in Figure 3.4 (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 3.4 (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.

3.5 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 3.3.

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 [53] 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 [46] can take over 2 hours per simulation.

3.5.1 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 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 [69], 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:

$$\min \left\{ \sum_{t=1}^{1096} (Q_t^{meas} - Q_t^{sim}(x))^2 : x_i^{\min} \leq x_i \leq x_i^{\max},\ i = 1, \ldots, n \right\}. \tag{3.40}$$

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 3.5 (a) shows the mean of the best function value for 30 different starting points generated uniformly within the bound-constrained region by MATLAB's

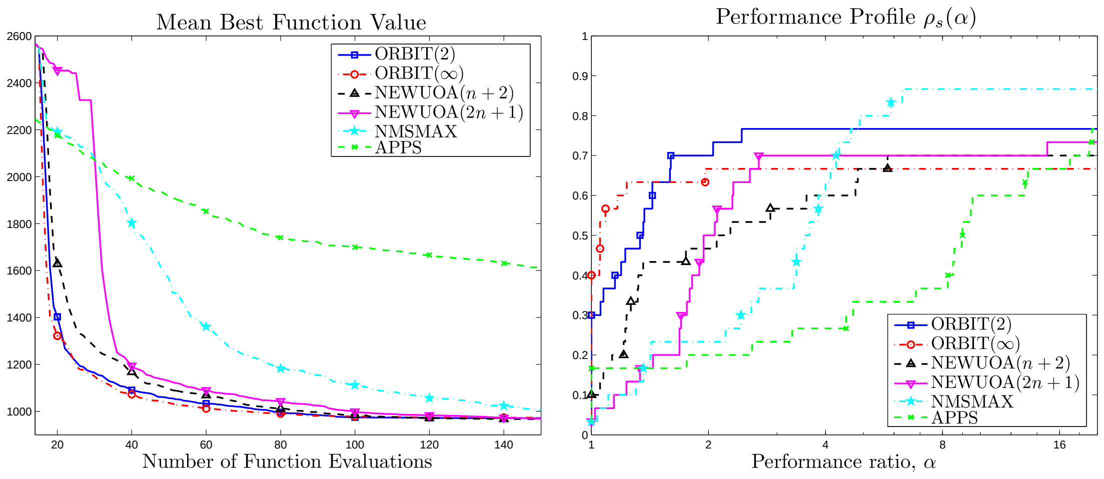


Figure 3.5: Town Brook problem ($n = 14$): (a) mean best function value (30 trials); (b) performance profile $\rho_s(\alpha)$ shows the relative performance, $\tau = .2$.

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 3.5 (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 Chapter 2 and [52] where APPSPACK

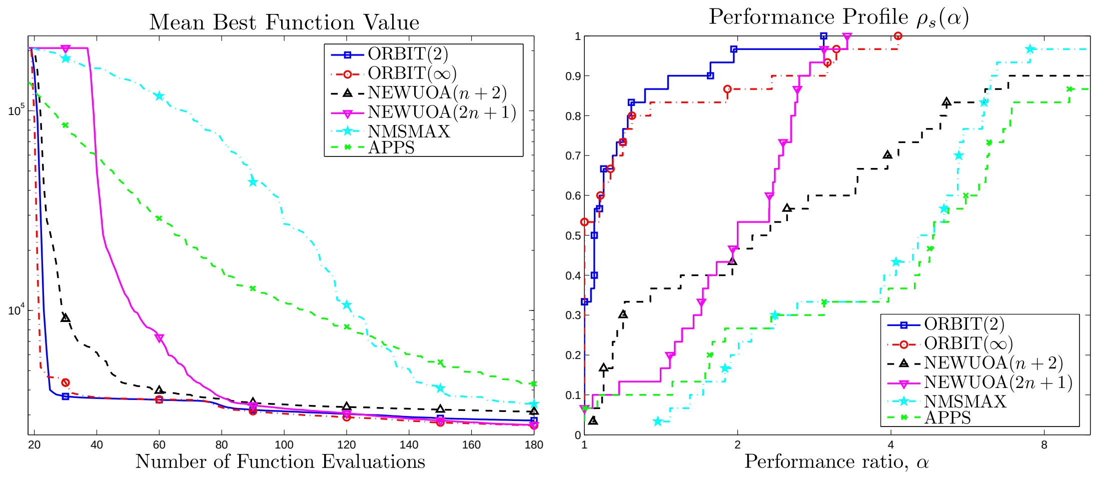


Figure 3.6: GWB18 Problem ($n = 18$): (a) ($\log_{10}$ scale) Mean Best Function Value (30 trials); (b) Performance Profile $\rho_s(\alpha)$ Shows the Relative Performance, $\tau = .01$.

and NMSMAX were successful at finding higher accuracy solutions for particularly messy functions if given many function evaluations.

3.5.2 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 [78]. We refer to this problem as GWB18.

Figure 3.6 (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 3.6 (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.

3.6 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 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 3.2 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 [30, 42], 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 Chapter 4 and [76].

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 chapters [69] and [78] used by us here as application problems. We are also grateful to Andy Conn and two anonymous referees for their helpful suggestions.

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.

CHAPTER 4

GLOBAL CONVERGENCE OF RADIAL BASIS FUNCTION TRUST-REGION ALGORITHMS

In this chapter we analyze trust-region algorithms for solving the unconstrained problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{4.1}$$

using Radial Basis Function (RBF) models. The deterministic real-valued function f is assumed to be continuously differentiable with a Lipschitz gradient ∇f and bounded from below, but we assume that all derivatives of f are either unavailable to the algorithm or intractable to compute. This chapter is driven by our work on the ORBIT algorithm introduced in [75] and Chapter 3, and highlights the key theoretical conditions needed for such algorithms to converge to first-order critical points. As a consequence, we will see that the popular thin-plate spline RBFs do not fit in this globally convergent framework. Further, our numerical results show that the Gaussian RBFs popularly used in kriging [38, 39] are not as effective in our algorithms as alternative RBF types.

A keystone of the present work is that we assume that the function is *computationally expensive* to evaluate. For our purposes, a computationally expensive function is one whose time for evaluation yields a bottleneck for classical techniques (the expense of evaluating the function at a single point outweighing any other expense/overhead of the algorithm). In some applications this could mean that function evaluation requires a few seconds on a state-of-the-art machine (in contrast to the milliseconds required by the overhead of an optimization algorithm), up to functions which, even when parallelized, require several hours on a large cluster. The functions which drive our work usually depend on complex deterministic computer simulations including those numerically solving systems of PDEs governing underlying physical phenomena.

The *derivative-free optimization* problem in (4.1) has received renewed interest in recent years. Research has primarily focused on developing methods which do not rely on finite-difference estimates of the function's gradient or Hessian. These methods can generally be categorized into those based on systematic sampling of the function along well-chosen directions [37, 42, 44, 45], and those employing a trust-region framework with a local approximation of the function [13, 48, 57, 58, 59, 61].

The methods in the former category are particularly popular with engineers for their ease of implementation and include the Nelder-Mead simplex algorithm [45] and Pattern/Direct Search [44]. These methods also admit natural parallel methods [37, 42] where different poll directions are sent to different processors for evaluation and hence have also proved to be attractive for high performance computing applications.

Methods in the latter category use prior function evaluations to construct a model, which approximates the function in compact neighborhoods of a current iterate. These models (for example, fully quadratic [13, 48, 58], underdetermined or structured quadratic [59, 61], or radial basis functions [57, 75]) yield computationally attractive derivatives and are hence easy to optimize over within the neighborhood.

The algorithm driving the present work, named ORBIT, belongs to the latter category. ORBIT is a trust-region algorithm relying on a radial basis function model with a *linear polynomial tail*, which interpolates the function at a set of previously-evaluated points [75]. A primary distinction between ORBIT and the previously proposed RBF-based algorithm in [57] is the management of this interpolation set. In contrast to [57], the expense of our objective function allows us to effectively ignore the computational complexity of the overhead of building and maintaining the RBF model.

In Section 4.1 we review the multivariate interpolation problem and derive a new result showing that the local error between the function (and its gradient) and a fairly general interpolation model (and its gradient) can be bounded using a simple condition on $n+1$ of the interpolation points. This work is in the spirit of recent work on the geometry of sample sets [16], and shows that such interpolation models can be made *fully linear* with an easy procedure requiring at most n additional function evaluations.

Since a computational budget is usually the limiting factor when optimizing computationally expensive functions, the focus of ORBIT is primarily on making rapid progress towards reducing the function value. In this chapter we further require that the algorithm converge to first-order stationary points. In Section 4.2 we review derivative-free trust-region methods and the conditions necessary for global convergence when fully linear models are employed. For this convergence analysis we benefit from the recent results in [15]. Additional proofs are provided in Appendix A.

Our analysis reveals the conditions we believe are necessary for obtaining a globally convergent trust-region method using an interpolating RBF-based model. In Section 4.3 we introduce radial basis functions and the fundamental property of *conditional positive definiteness*, which we rely on to construct uniquely-defined RBF models with bounded coefficients. To accomplish this we develop a procedure for iteratively selecting interpolation points from the points that the algorithm has previously evaluated, while ensuring that the resulting model has a bounded Hessian. We also give necessary and sufficient conditions for different RBF types to fit within our framework.

In Section 4.4 we illustrate the effect of selecting from three different popular radial basis functions covered by the theory by running the resulting algorithm on a set of smooth test functions. We also examine the effect of varying the maximum number of interpolation points. Finally, we motivate the use of ORBIT on computationally expensive functions with an application problem (requiring nearly 1 CPU hour per evaluation on a Pentium 4 machine) arising from cleaning up contaminated groundwater. We remark that additional computational results, both on a set of test problems and on two applications from Environmental Engineering, as well as more practical considerations are addressed in [75].

4.1 Interpolation Models

We begin our discussion on models that interpolate a set of scattered data with an introduction to the polynomial models that are heavily relied on by the derivative-free trust-region methods in the literature [13, 48, 58, 59, 61].

We first collect the notation conventions used throughout the chapter. $\mathbb{N}_0^n$ will denote n-tuples from the natural numbers including zero. A vector $x \in \mathbb{R}^n$ will be written in component form as $x = [\chi_1, \ldots, \chi_n]^T$ to differentiate it from a particular point $x_i \in \mathbb{R}^n$. For $d \in \mathbb{N}_0$, let $\mathcal{P}_{d-1}^n$ denote the space of n-variate polynomials of total degree no more than $d-1$ with the convention that $\mathcal{P}_{-1}^n = \emptyset$. Let $\mathcal{Y} = \{y_1, y_2, \ldots, y_{|\mathcal{Y}|}\} \subset \mathbb{R}^n$ denote an interpolation set of $|\mathcal{Y}|$ data points where (y_i, f_i) is known. For ease of notation, we will often assume interpolation relative to some base point $x_b \in \mathbb{R}^n$, made clear from the context, and employ the set notation $x_b + \mathcal{Y} = \{x_b + y : y \in \mathcal{Y}\}$. Lastly, we will be working with a general norm $\|\cdot\|_k$ that we are careful to distinguish from the 2-norm $\|\cdot\|$, and will rely on the positive constants c_1, c_2 satisfying:

$$\|\cdot\| \leq c_1 \|\cdot\|_k \qquad \forall k, \tag{4.2}$$

$$\|\cdot\|_k \leq c_2 \|\cdot\| \qquad \forall k, \tag{4.3}$$

respectively, c_1 and c_2 depending only on the dimension n.

The multivariate polynomial interpolation problem is to find a polynomial $P \in \mathcal{P}^n_{d-1}$ such that

$$P(y_i) = f_i, \qquad \forall y_i \in \mathcal{Y}, \tag{4.4}$$

for arbitrary values $f_1, \ldots, f_{|\mathcal{Y}|} \in \mathbb{R}$. Spaces where unique polynomial interpolation is possible given an appropriate number of distinct data points are called *Haar spaces.* A classic theorem of Mairhuber and Curtis (cf. [73, pg 19]) states that Haar spaces do not exist for $n \geq 2$. Hence additional (geometric) conditions are necessary for the multivariate interpolation problem in (4.4) to be well-posed. This necessitates the following definition.

Definition 4.1. *The data points $\mathcal{Y}$ are $\underline{\mathcal{P}^n_{d-1}\text{-unisolvent}}$ if the only polynomial in $\mathcal{P}^n_{d-1}$ that vanishes at all points in $\mathcal{Y}$ is the zero polynomial.*

The monomials $\{\chi_1^{\alpha_1} \cdots \chi_n^{\alpha_n} : \alpha \in \mathbb{N}_0^n, \sum_{i=1}^n \alpha_i \leq d-1\}$ form a basis for $\mathcal{P}^n_{d-1}$ and hence any polynomial $P \in \mathcal{P}^n_{d-1}$ can be written as a linear combination of such monomials. In general, for a basis $\{\pi_i(x)\}_{i=1}^{\hat{p}}$ with $\pi_i : \mathbb{R}^n \to \mathbb{R}$ we will use the representation $P(x) = \sum_{i=1}^{\hat{p}} \nu_i \pi_i(x)$, where $\hat{p} = \dim \mathcal{P}^n_{d-1} = \binom{n+d-1}{n}$. Hence finding an interpolating polynomial $P \in \mathcal{P}^n_{d-1}$ is equivalent to finding coefficients $\nu \in \mathbb{R}^{\hat{p}}$ for which (4.4) holds.

Defining the matrix $\Pi \in \mathbb{R}^{\hat{p} \times |\mathcal{Y}|}$ as $\Pi_{i,j} = \pi_i(y_j)$, it follows that $\mathcal{Y}$ is $\mathcal{P}^n_{d-1}$-unisolvent if and only if Π is of full row rank, rank$\Pi = \hat{p}$. Further, the interpolation in (4.4) is unique for arbitrary right-hand-side values $f_1, \ldots, f_{|\mathcal{Y}|} \in \mathbb{R}$ if and only if $|\mathcal{Y}| = \hat{p}$ and Π is in fact invertible. In this case, the unique polynomial that solves (4.4) is defined by the coefficient vector:

$$\nu = \Pi^{-T} f. \tag{4.5}$$

It is easy to see that the existence and uniqueness of an interpolant is independent of the particular basis $\{\pi_i(x)\}_{i=1}^{\hat{p}}$ employed. However, the conditioning of the corresponding matrix Π depends strongly on the basis chosen, as noted (for example) in [16].

Based on these observations, we see that in order to uniquely fit a polynomial of degree $d-1$ to function values, the function f must be evaluated on at least $\hat{p} = \dim \mathcal{P}^n_{d-1} = \binom{n+d-1}{n}$ points. When f is computationally expensive and n is not very small, then the computational expense of evaluating f to repeatedly fit even a quadratic (with $\hat{p} = \frac{(n+1)(n+2)}{2}$), and as is done in [13] and [58], may be better spent in other ways. To this end, we will focus on interpolation models that can be formed using as few as a linear (in the problem dimension, n) number of function values.

4.1.1 Fully Linear Models

We now explore a class of *fully linear* interpolation models, which can be formed using as few as $n+1$ function evaluations. Since such models are heavily tied to Taylor-like error bounds, we will require additional assumptions on the function f as in this definition from [16].

Definition 4.2. *Suppose that $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_b\|_k \leq \Delta\}$ and $f \in C^1[\mathcal{B}]$. For fixed $\kappa_f, \kappa_g > 0$, a model $m \in C^1[\mathcal{B}]$ is said to be* <u>*fully linear on $\mathcal{B}$*</u> *if for all $x \in \mathcal{B}$:*

$$|f(x) - m(x)| \leq \kappa_f \Delta^2, \tag{4.6}$$

$$\|\nabla f(x) - \nabla m(x)\| \leq \kappa_g \Delta. \tag{4.7}$$

This definition ensures that Taylor-like bounds exist for both the model's error and the gradient of the model's error within the compact neighborhood $\mathcal{B}$. For example, if $f \in C^1[\mathbb{R}]$, ∇f has Lipschitz constant γ_f, and m is the derivative-based linear model $m(x_b + s) = f(x_b) + \nabla f(x_b)^T s$, then m is fully linear with constants $\kappa_g = \kappa_f = \gamma_f$ for any bounded region $\mathcal{B}$.

Since the function's gradient is unavailable in our setting, our focus is on models that interpolate the function at a set of points:

$$m(x_b + y_i) = f(x_b + y_i) \qquad \text{for all } y_i \in \mathcal{Y} = \{y_1 = 0, y_2, \dots, y_{|\mathcal{Y}|}\} \subset \mathbb{R}^n. \tag{4.8}$$

While we may have interpolation at more than $n+1$ points, we will for the moment work with a subset of exactly $n+1$ points and will always enforce interpolation at the base point x_b so that $y_1 = 0 \in \mathcal{Y}$. It will also be useful to place the remaining n (nonzero) points in a square matrix:

$$Y = \begin{bmatrix} y_2 & \cdots & y_{n+1} \end{bmatrix}. \tag{4.9}$$

We can now state the following Theorem, which is a generalization of the error bounds found in [16] and [17].

Theorem 4.1. *Suppose that f and m are continuously differentiable in $\mathcal{B} = \{x : \|x - x_b\|_k \leq \Delta\}$ 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 (4.8) at a set of points* $\{y_1 = 0, y_2, \ldots, y_{n+1}\} \subseteq \mathcal{B} - x_b$ *such that* $\|Y^{-1}\| \leq \frac{\Lambda_Y}{c_2\Delta}$*, for a fixed constant* $\Lambda_Y < \infty$*. Then for any* $x \in \mathcal{B}$*:*

$$|m(x) - f(x)| \leq \sqrt{n}c_2^2(\gamma_f + \gamma_m)\left(\frac{5}{2}\Lambda_Y + \frac{1}{2}\right)\Delta^2 = \kappa_f\Delta^2, \quad (4.10)$$

$$\|\nabla f(x) - \nabla m(x)\| \leq \frac{5}{2}\sqrt{n}\Lambda_Y c_2(\gamma_f + \gamma_m)\Delta = \kappa_g\Delta. \quad (4.11)$$

Proof. Define:

$$e^m(s) = m(x_b + s) - f(x_b + s), \quad (4.12)$$

$$e^g(s) = \nabla m(x_b + s) - \nabla f(x_b + s), \quad (4.13)$$

and let $\langle \cdot, \cdot \rangle$ be the standard dot product on $\mathbb{R}^n$. A first-order Taylor expansion (see, for example [20, pg 71]) about $x_b + s$ for both f and m gives:

$$\begin{aligned}\langle e^g(s), y_i - s\rangle = & \int_0^1 \langle \nabla f(x_b + s + t(y_i - s)) - \nabla f(x_b + s), y_i - s\rangle dt \\ & - \int_0^1 \langle \nabla m(x_b + s + t(y_i - s)) - \nabla m(x_b + s), y_i - s\rangle dt \\ & - e^m(s) \end{aligned} \quad (4.14)$$

for $i = 1, \ldots, n+1$. Subtracting the equation associated with $y_1 = 0$ yields:

$$\begin{aligned}\langle e^g(s), y_i\rangle = & \int_0^1 \langle \nabla f(x_b + s + t(y_i - s)) - \nabla f(x_b + s), y_i - s\rangle dt \\ & - \int_0^1 \langle \nabla m(x_b + s + t(y_i - s)) - \nabla m(x_b + s), y_i - s\rangle dt \\ & - \int_0^1 \langle \nabla f(x_b + s - ts)) - \nabla f(x_b + s), -s\rangle dt \\ & + \int_0^1 \langle \nabla m(x_b + s - ts)) - \nabla m(x_b + s), -s\rangle dt,\end{aligned}$$

for each y_i, $i = 2, \ldots, n+1$. We now examine how large the first right hand side term can get when $s \in \mathcal{B} - x_b$:

$$\begin{aligned}
&\left| \int_0^1 \langle \nabla f\left(x_b + s + t(y_i - s)\right) - \nabla f(x_b + s), y_i - s \rangle dt \right| \\
\leq \;& \int_0^1 \left\| \nabla f\left(x_b + s + t(y_i - s)\right) - \nabla f(x_b + s) \right\| \left\| y_i - s \right\| dt \\
\leq \;& \int_0^1 \gamma_f c_2^2 \left\| y_i - s \right\|_k^2 t dt \\
\leq \;& 2\gamma_f c_2^2 \Delta^2,
\end{aligned}$$

since $\|y_i - s\|_k^2 \leq 4\Delta^2$ when $s, y_i \in \mathcal{B} - x_b$. Similar expressions may be obtained for the other 3 terms and hence the right hand side may be bounded by:

$$\left\| Y^T e^g(s) \right\| \leq \sqrt{n} \left\| Y^T e^g(s) \right\|_\infty \leq \sqrt{n} \left(\frac{5}{2} \gamma_f c_2^2 \Delta^2 + \frac{5}{2} \gamma_m c_2^2 \Delta^2 \right), \tag{4.15}$$

since $\|\cdot\| \leq \sqrt{n} \|\cdot\|_\infty$. Lastly, since $\left\| Y^{-T} \right\| = \|Y^{-1}\| \leq \frac{\Lambda_Y}{c_2 \Delta}$, we obtain (4.11) by noting

$$\|e^g(s)\| \leq \left\| Y^{-T} \right\| \left\| Y^T e^g(s) \right\| \leq \frac{5}{2} \sqrt{n} \Lambda_Y c_2 \left(\gamma_f + \gamma_m \right) \Delta. \tag{4.16}$$

Returning to $e^m(s)$ in (4.14) we then have for $i = 1$ ($y_1 = 0$):

$$\begin{aligned}
|e^m(s)| \;&\leq\; \|e^g(s)\| \, \|s\| + \frac{1}{2} \sqrt{n} \left(\gamma_f c_2^2 \Delta^2 + \gamma_m c_2^2 \Delta^2 \right) \\
&\leq\; \sqrt{n} c_2^2 \left(\gamma_f + \gamma_m \right) \left(\frac{5}{2} \Lambda_Y + \frac{1}{2} \right) \Delta^2.
\end{aligned}$$

□

We note that Theorem 4.1 holds for very general interpolation models, requiring only a minor degree of smoothness and conditions on the points being interpolated. In particular, Theorem 4.1 provides the constants $\kappa_f, \kappa_g > 0$ in (4.10) and (4.11) such that conditions (4.6) and (4.7) are satisfied, and hence m is fully linear in a neighborhood containing the $n+1$ interpolation points.

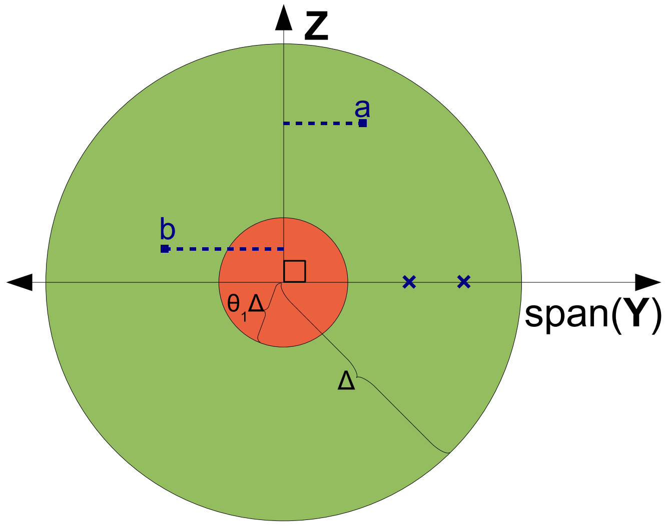


Figure 4.1: Finding sufficiently affinely independent points: a is acceptable, b is not.

In Theorem 4.1 we have isolated the key condition for a model to be fully linear: interpolation at $y_1 = 0$ and a set of n points resulting in a matrix Y^{-1} of bounded norm. This condition is equivalent to requiring that the points $\{y_1, y_2, \ldots, y_{n+1}\}$ are sufficiently affinely (linear) independent (or equivalently, that the set $\{y_2, \ldots, y_{n+1}\}$ is sufficiently linearly independent).

It is easy to iteratively construct a set of such points given a set of candidates $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\} \subset \{x \in \mathbb{R}^n : \|x - x_b\|_k \leq \Delta\}$ using LU- and QR-like algorithms as noted in [16]. Points from $\mathcal{D}$ will be added to the interpolation set $\mathcal{Y}$ one-at-a-time using a QR-like variant as illustrated in Figure 4.1.

In Figure 4.1 we show the subspace spanned by the interpolation points currently in hand and the orthogonal complement, Z, of the corresponding matrix Y defined in (4.9). A new candidate from $\mathcal{D}$ is added to $\mathcal{Y}$ if and only if it yields a projection onto Z that is sufficiently large (as measured by the constant $\theta \in (0, 1]$). Using the θ shown in Figure 4.1, the point b would not be included in $\mathcal{Y}$ since its

```
4.1.0. Input $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\} \subset \mathbb{R}^n$, constants $\theta \in (0,1]$, $\Delta > 0$.
4.1.1. Initialize $\mathcal{Y} = \{0\}$, $Z = I_n$.
4.1.2. For all $d_j \in \mathcal{D}$ such that $\|d_j\|_k \leq \Delta$:
    If $\left\|\mathrm{proj}_Z\left(\frac{1}{\Delta}d_j\right)\right\| \geq \theta$,
        $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{d_j\}$,
        Update $Z$ to be an orthonormal basis for $\mathcal{N}\left([y_1 \cdots y_{|\mathcal{Y}|}]\right)$.
4.1.3. If $|\mathcal{Y}| = n+1$, then $m_k$ is fully linear, otherwise $m_k$ is not fully linear.
```

Algorithm 4.1: AffPoints($\mathcal{D}, \theta, \Delta$): Algorithm for obtaining fully linear models.

projection is of magnitude less than $\theta\Delta$, while the point a would be. The formal procedure is provided as Algorithm 4.1.

Note that if Algorithm 4.1 is exited with $|\mathcal{Y}| < n+1$, m_k is not fully linear and the final orthonormal basis Z will be nonempty. However, the model can be made fully linear if the interpolation set $\mathcal{Y}$ is augmented by including scaled versions of the remaining columns of Z, $\{\Delta z_1, \ldots, \Delta z_{n+1-|\mathcal{Y}|}\}$. These points belong to $\{x \in \mathbb{R}^n : \|x - x_b\|_k \leq \Delta\}$ and are perfectly conditioned with respect to the projection done in Algorithm 4.1.

It remains only to show that if $|\mathcal{Y}| = n+1$ upon exiting Algorithm 4.1, the points in $\mathcal{Y}$ satisfy $\|Y^{-1}\| \leq \frac{\Lambda_Y}{c_2\Delta}$. The following intuitive Lemma (proved in [75]) provides a bound on $\|Y^{-1}\|$ based on the pivots of a QR factorization.

Lemma 4.1. *Let $QR = \frac{1}{c_2\Delta}Y$ denote a QR factorization of a matrix $\frac{1}{c_2\Delta}Y$ whose columns satisfy $\left\|\frac{Y_j}{c_2\Delta}\right\| \leq 1$, $j = 1, \ldots, n$. If $r_{ii} \geq \theta > 0$ for $i = 1, \ldots, n$, then $\|Y^{-1}\| \leq \frac{\Lambda_Y}{c_2\Delta}$ for a constant Λ_Y depending only on n and θ.*

We note that this result does not require the thresholding procedure in Step

4.1.2. of Algorithm 4.1 to add points in the order of their pivot values. Hence a user is free to first attempt to add more recently sampled points, which are often closer to x_b and thus usually have lower pivot values.

4.2 Derivative-Free Trust-Region Methods

The interpolation models of the previous section were constructed to approximate a function in a local neighborhood of a point x_b. The natural algorithmic extensions of such models are trust-region methods, whose general form we now briefly review.

Trust-region methods generate a sequence of iterates $\{x_k\}_{k\geq 0} \subseteq \mathbb{R}^n$ by employing a surrogate model $m_k : \mathbb{R}^n \to \mathbb{R}$, assumed to approximate f within a neighborhood of the current iterate x_k. For a (center, radius) pair $(x_k, \Delta_k > 0)$ define the *trust-region*:

$$\mathcal{B}_k = \{x \in \mathbb{R}^n : \|x - x_k\|_k \leq \Delta_k\}, \tag{4.17}$$

where we are careful to distinguish the trust-region norm (at iteration k), $\|\cdot\|_k$, from other norms used here. New points are obtained by solving subproblems of the form:

$$\min \{m_k(x_k + s) : x_k + s \in \mathcal{B}_k\}. \tag{4.18}$$

Given an approximate solution s_k to (4.18), the pair (x_k, Δ_k) is updated according to the ratio of actual to predicted improvement,

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}. \tag{4.19}$$

Given a maximum trust-region radius $\Delta_{\max}$, the design of the trust-region algorithm ensures that f is only sampled within the relaxed level set:

$$\mathcal{L}(x_0) = \{y \in \mathbb{R}^n : \|x - y\|_k \leq \Delta_{\max} \text{ for some } x \text{ with } f(x) \leq f(x_0)\}, \tag{4.20}$$

hence one really only requires that f be sufficiently smooth within $\mathcal{L}(x_0)$. Trust-region algorithms (primarily when derivative information is available) are given full treatment in [12].

When exact derivatives are unavailable, smoothness of the function f is no longer sufficient for guaranteeing that a model m_k approximates the function locally. Hence the main difference between classical and derivative-free trust-region algorithms is the addition of safeguards to account for and improve models of poor quality.

Historically, the most frequently used model is a quadratic,

$$m_k(x_k + s) = f(x_k) + g_k^T s + \frac{1}{2} s^T H_k s, \tag{4.21}$$

the coefficients g_k and H_k being found by enforcing interpolation at $\frac{(n+1)(n+2)}{2}$ points as in (4.8). Alternatively, the coefficients g_k and H_k could be found by finite difference methods using $\frac{(n+1)(n+2)}{2} - 1$ nearby function evaluations, but this is often impractical because these evaluations are only useful for estimating derivatives at the current center, x_k. Furthermore, in many applications finite difference estimates are unreliable due to computational noise.

Quadratic interpolation models form the basis for the derivative-free algorithms in [13, 48, 58, 59, 61, 77]. As discussed in Section 4.1, these models rely heavily on results from multivariate interpolation. Quadratic models are attractive in practice because the resulting subproblem in (4.18), for a 2-norm trust-region, is one of the only nonlinear programs for which *global* solutions can be efficiently computed.

A downside of quadratic models in our computationally expensive setting is that the number of interpolation points (and hence function evaluations) required is quadratic in the dimension of the problem. Noting that it may be more efficient to use additional function evaluations for forming subsequent models, Powell's NEWUOA code [60, 61] relies on least-change quadratic models interpolating fewer than $\frac{(n+1)(n+2)}{2}$ points.

4.2.1 Fully Linear Derivative-Free Models

Recognizing the difficulty (and possible inefficiency) of maintaining geometric conditions on a quadratic number of points, we will focus on using the fully linear models introduced in Section 4.1. These models can be formed with a linear number of points while still maintaining the local approximation bounds in (4.6) and (4.7).

We will follow the recent general trust-region algorithmic framework introduced for linear models by Conn et. al [15] in order to arrive at a similar convergence result for the types of models considered here. Given standard 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 the additional constants $\kappa_d \in (0,1), \kappa_f > 0, \kappa_g > 0, \epsilon > 0, \mu > \beta > 0, \alpha \in (0,1)$, the general first-order derivative-free trust-region algorithm is shown in Algorithm 4.2 with the final criticality subroutine shown in Algorithm 4.3. Both of these are discussed in [15] and we note these form an infinite loop, a recognition that termination in practice is often as a result of exhausting a budget of expensive function evaluations.

A benefit of working with more general fully linear models is that they allow for nonlinear modeling of f. Hence, we will primarily be interested in models with nontrivial Hessians, $\nabla^2 m_k \neq 0$, which are uniformly bounded by some constant κ_H.

4.2.1. Criticality test

If $\|\nabla m_k(x_k)\| \leq \epsilon$ and either m_k is not fully linear or $\Delta_k > \mu \|\nabla m_k(x_k)\|$:

Obtain m_k and $\tilde{\Delta}_k$ by running the final criticality subroutine in Algorithm 4.3.

Update $\Delta_k = \max\{\tilde{\Delta}_k, \beta \|\nabla m_k(x_k)\|\}$.

4.2.2. Obtain step s_k satisfying a sufficient decrease condition

4.2.3. Evaluate $f(x_k + s_k)$

4.2.4. Adjust trust-region according to ratio $\rho_k = \frac{f(x_k)-f(x_k+s_k)}{m_k(x_k)-m_k(x_k+s_k)}$:

$$\Delta_{k+1} = \begin{cases} \min\{\gamma_1 \Delta_k, \Delta_{\max}\} & \text{if } \rho_k \geq \eta_1 \text{ and } \Delta_k < \beta \|g_k\| \\ \Delta_k & \text{if } \rho_k \geq \eta_1 \text{ and } \Delta_k \geq \beta \|g_k\| \\ \Delta_k & \text{if } \rho_k < \eta_1 \text{ and } m_k \text{ is not fully linear} \\ \gamma_0 \Delta_k & \text{if } \rho_k < \eta_1 \text{ and } m_k \text{ is fully linear} \end{cases} \tag{4.22}$$

$$x_{k+1} = \begin{cases} x_k + s_k & \text{if } \rho_k \geq \eta_1 \\ x_k + s_k & \text{if } \rho_k > \eta_0 \text{ and } m_k \text{ is fully linear} \\ x_k & \text{else} \end{cases} \tag{4.23}$$

4.2.5. Improve m_k if $\rho_k < \eta_1$ and m_k is not fully linear.

4.2.6. Form new model m_{k+1}

Algorithm 4.2: Iteration k of a first-order (fully linear) derivative-free algorithm [15].

The sufficient decrease condition that we will use in Step 4.2.2. then takes the form:

$$m_k(x_k) - m_k(x_k + s) \geq \frac{\kappa_d}{2} \|\nabla m_k(x_k)\| \min\left\{ \frac{\|\nabla m_k(x_k)\|}{\kappa_H}, \frac{\|\nabla m_k(x_k)\|}{\|\nabla m_k(x_k)\|_k} \Delta_k \right\}. \tag{4.24}$$

for some pre-specified constant $\kappa_d \in (0,1)$. This condition is similar to those found in the trust-region setting when general norms are employed [12].

It is important to note that we will always be able to find an approximate solution, s_k, to the subproblem (4.18) that satisfies condition (4.24). The following Lemma guarantees this.

```
4.3.1. Set $\tilde{\Delta}_k = \Delta_k$
4.3.2. Update $m_k$ so that it is fully linear on $\{x : \|x - x_k\|_k \le \tilde{\Delta}_k\}$.
4.3.3. While $\tilde{\Delta}_k > \mu \|\nabla m_k(x_k)\|$:
           Set $\tilde{\Delta}_k \leftarrow \alpha\tilde{\Delta}_k$ and update $m_k$ and so that it is fully linear on $\{x : \|x - x_k\|_k \le \tilde{\Delta}_k\}$.
```

Algorithm 4.3: Final criticality subroutine.

Lemma 4.2. *Let $\mathcal{B}_k = \{x \in \mathbb{R}^n : \|x - x_k\|_k \le \Delta_k\}$. If:*

- *$m_k \in C^2(\mathcal{B}_k)$,*
- *and $\kappa_H > 0$ satisfies*

$$\infty > \kappa_H \ge \max_{x \in \mathcal{B}_k} \left\|\nabla^2 m_k(x)\right\|, \tag{4.25}$$

then for any $\kappa_d \in (0, 1)$ there exists an $s \in \mathcal{B}_k - x_k$ satisfying (4.24).

Proof. Without loss of generality we assume that $\nabla m_k(x_k) \neq 0$ (otherwise $s = 0$ trivially satisfies (4.24)). We will construct such a point by considering the step:

$$s(j) = -\kappa_d^j \frac{\Delta_k}{\|\nabla m_k(x_k)\|_k} \nabla m_k(x_k). \tag{4.26}$$

First note that if $j \ge 0$, $\|s(j)\|_k = \kappa_d^j \Delta_k \le \Delta_k$ and hence $s(j) \in \mathcal{B}_k - x_k$. Now note that since $\kappa_d < 1$, for the (finite) index

$$j = \max\left\{0, \left\lceil \log_{\kappa_d} \frac{\|\nabla m_k(x_k)\|_k}{\Delta_k \kappa_H} \right\rceil\right\}. \tag{4.27}$$

we have that:

$$\kappa_d^j \le \frac{\|\nabla m_k(x_k)\|_k}{\Delta_k \kappa_H}. \tag{4.28}$$

Applying Taylor's Theorem, we have that for some $\xi \in \mathcal{B}_k$:

$$\begin{aligned} m_k(x_k) - m_k(x_k + s(j)) &= -\nabla m_k(x_k)^T s(j) - \frac{1}{2} s(j)^T \nabla^2 m_k(\xi) s(j) \\ &\geq -\nabla m_k(x_k)^T s(j) - \frac{1}{2} \|s(j)\|^2 \kappa_H \\ &= \left(1 - \frac{\kappa_d^j}{2} \frac{\Delta_k \kappa_H}{\|\nabla m_k(x_k)\|_k}\right) \frac{\|\nabla m_k(x_k)\|^2}{\|\nabla m_k(x_k)\|_k} \Delta_k \kappa_d^j \\ &\geq \frac{1}{2} \frac{\|\nabla m_k(x_k)\|^2}{\|\nabla m_k(x_k)\|_k} \Delta_k (\kappa_d)^j, \end{aligned} \tag{4.29}$$

where the last inequality follows by rearranging (4.28). If $j = 0$, the result now follows immediately from (4.29). If $j \geq 1$, then by the choice of j in (4.27), the fact that $(\kappa_d)^j$ is strictly decreasing since $\kappa_0 < 1$, and because $\lceil a \rceil - 1 < a$ for all $a \in \mathbb{R}$, we have:

$$(\kappa_d)^{j-1} > \frac{\|\nabla m_k(x_k)\|_k}{\Delta_k \kappa_H}, \tag{4.30}$$

and hence

$$(\kappa_d)^j = \kappa_d (\kappa_d)^{j-1} > \kappa_d \frac{\|\nabla m_k(x_k)\|_k}{\Delta_k \kappa_H}. \tag{4.31}$$

Thus (4.29) and (4.31) give:

$$m_k(x_k) - m_k(x_k + s(j)) \geq \frac{\kappa_d}{2} \frac{\|\nabla m_k(x_k)\|^2}{\kappa_H}, \tag{4.32}$$

and (4.24) again follows. □

The previous Lemma is our variant of similar ones in [12] and provides a simple way for computing a step that yields a model reduction that is at least a fraction of that achieved by the Cauchy point. Thus a simple back-tracking line search algorithm will yield a sufficient decrease in a finite number of steps. We immediately have the following Corollary, which states that the size of this step is bounded from zero if $\|\nabla m_k(x_k)\|_k$ and Δ_k are.

Corollary 4.1. *There exists $s \in \mathcal{B}_k - x_k$ satisfying (4.24) such that*

$$\|s\|_k \geq \min\left\{\Delta_k, \kappa_d \frac{\|\nabla m_k(x_k)\|_k}{\kappa_H}\right\}. \tag{4.33}$$

Proof. From Lemma 4.24 we know that the step $s(j)$ in (4.26) satisfies (4.24) for the j in (4.27). If $j = 0$ then $\|s(j)\|_k = \Delta_k$. Otherwise $(\kappa_d)^j$ satisfies (4.31) and hence $\|s(j)\|_k \geq \kappa_d \frac{\|\nabla m_k(x_k)\|_k}{\kappa_H}$. Combining these two cases we have (4.33). $\square$

Reluctance to use nonpolynomial models in practice can primarily be attributed to the difficulty of solving the trust-region subproblem (4.18). We will show that by using the sufficient decrease condition (4.24), guaranteed to be easily attainable by the preceding Lemma, first-order convergence is still possible. This result is independent of the number of local or global minima that the trust-region subproblem may have as a result of using multimodal models.

Further, we assume that the twice continuously differentiable model used in practice will have first- and second- order derivatives available to more accurately solve (4.18). Using a more sophisticated solver technique for the subproblem may be an especially attractive option when f is computationally expensive, since this additional expense will be negligible when compared with the time required to evaluate f at the subproblem solution.

We now state the convergence result for our models of interest and Algorithm 4.2.

Theorem 4.2. *Suppose that the following two assumptions hold:*

(AF) *$f \in C^1[\Omega]$ for some open $\Omega \supset \mathcal{L}(x_0)$ (with $\mathcal{L}(x_0)$ defined in (4.20)), ∇f is Lipschitz continuous on $\mathcal{L}(x_0)$, and f is bounded on $\mathcal{L}(x_0)$.*

(AM) *For all $k \geq 0$, $m_k \in C^2[\mathcal{B}_k]$, $\infty > \kappa_H \geq \max_{x\in\mathcal{B}_k} \|\nabla^2 m_k(x)\|$, and m_k can be made (and verified to be) fully linear by some finite procedure.*

Then for the sequence of iterates generated by Algorithm 4.2, we have:

$$\lim_{k\to\infty} \nabla f(x_k) = 0 \tag{4.34}$$

Proof. This follows in large part from the corresponding lemmas in [15] with minor changes made to accommodate our sufficient decrease condition and the more general trust-region norm employed. These lemmas, and further explanation where needed, are provided in Appendix A. □

Theorem 4.2 outlines the fundamental conditions on the model m_k that are needed to show convergence. The next section is devoted to analyzing which radial basis function models satisfy these conditions.

4.3 Radial Basis Functions

Throughout this section we will drop the dependence of the model on the iteration number, k, but note that we intend for the model m and base point x_b to be the kth model and iterate, m_k and x_k, in the trust-region algorithm of the previous section.

An alternative to higher degree polynomial models is an interpolating surrogate that is a linear combination of nonlinear nonpolynomial basis functions. One such model is of the form:

$$m(x_b + s) = \sum_{j=1}^{|\mathcal{Y}|} \lambda_j \phi(\|s - y_j\|) + P(s), \tag{4.35}$$

where $s \in \mathbb{R}^n$, $\phi : \mathbb{R}_+ \to \mathbb{R}$ is a univariate function, and $P \in \mathcal{P}_{d-1}^n$ is a polynomial as in Section 4.1. Such models are called *radial basis functions* (RBFs) because $m(x_b + s) - P(s)$ is a linear combination of shifts of a function $\phi(\|x\|)$, which is constant on spheres in $\mathbb{R}^n$.

Interpolation by RBFs of scattered data has only recently gained popularity in practice [10]. In the context of optimization methods, RBF models have been primarily used for global optimization [8, 33, 65]. In global optimization such models are attractive because they are able both to model multimodal/nonconvex functions and to interpolate a large number of points (relative to the dimension) in a numerically stable manner.

To the authors' knowledge, Oeuvray was the first to employ RBFs in a local optimization algorithm. He introduced BOOSTERS, a derivative-free trust-region algorithm using a cubic RBF model with a linear tail, in his 2005 dissertation [55]. Oeuvray was motivated by medical image registration problems and was particularly interested in "doping" his algorithm with gradient information when it becomes available [57].

The advent of the recent results in [15] has allowed us to generalize the theory for globally convergent first-order RBF trust-region algorithms and obtain an algorithm [75] that works particularly well in practice for computationally expensive functions. In this chapter we seek to analyze the properties of RBFs needed to fit in the framework presented in Section 4.2.

Table 4.1: 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, $e^{-\frac{r^2}{\gamma^2}}$

4.3.1 Conditional Positive Definite Functions

The fundamental property we rely on is *conditional positive definiteness*, which we now define using the notation of Wendland [73].

Definition 4.3. *Let π be a basis for $\mathcal{P}^n_{d-1}$, with the convention that $\pi = \emptyset$ if $d = 0$. A function ϕ is said to be* <u>*conditionally positive definite of order d*</u> *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.*

Some examples of popular radial functions and their orders of conditional positive definiteness are given in Table 4.1. We note that if a radial function ϕ is conditionally positive definite of order d, then it is also conditionally positive definite of order $\hat{d} \geq d$ [73, pg 98].

We now use the property of conditional positive definiteness to uniquely determine an RBF model that interpolates data on a set $\mathcal{Y}$. Let $\Phi_{i,j} = \phi(\|y_i - y_j\|)$ define the square matrix $\Phi \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|}$ and let Π be the polynomial matrix $\Pi_{i,j} = \pi_i(y_j)$ as in Section 4.1. Provided that $\mathcal{Y}$ is $\mathcal{P}^n_{d-1}$-unisolvent (as in Definition 4.1), we have the equivalent nonsingular symmetric linear system:

$$\begin{bmatrix} \Phi & \Pi^T \\ \Pi & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \nu \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}. \tag{4.36}$$

The top set of equations corresponds to the interpolation conditions in (4.8) for the RBF model in (4.35) while the lower set ensures uniqueness of the solution.

As in Section 4.1 for polynomial models, for conditionally positive definite functions of order d, a sufficient condition for the nonsingularity of (4.36) is that the points in $\mathcal{Y}$ are distinct and yield a Π^T of full column rank. Clearly this condition is *geometric*, depending only on the location of (but not function values at) the data points.

Note that the saddle point problem in (4.36) will generally be indefinite [6]. However, we employ a null-space method that directly relies on the conditional positive definiteness of ϕ. If Π^T is full rank, then $R \in \mathbb{R}^{n+1 \times n+1}$ is nonsingular from the truncated QR factorization $\Pi^T = QR$. By the lower set of equations in (4.36) we must have $\lambda = Z\omega$ for $\omega \in \mathbb{R}^{|\mathcal{Y}|-n-1}$ and any orthogonal basis Z for $\mathcal{N}(\Pi)$. Hence (4.36) reduces to:

$$\begin{aligned} Z^T \Phi Z \omega &= Z^T f & (4.37) \\ R\nu &= Q^T(f - \Phi Z \omega). & (4.38) \end{aligned}$$

Given that Π^T is full rank and the points in $\mathcal{Y}$ are distinct, we are now in a setting where Definition 4.3 directly implies that $Z^T \Phi Z$ is positive definite for any ϕ that is conditionally positive definite of at most order d. The positive definiteness of $Z^T \Phi Z$ guarantees the existence of a nonsingular lower triangular Cholesky factor L such that:

$$Z^T \Phi Z = LL^T, \tag{4.39}$$

and the isometry of Z gives the bound:

$$\|\lambda\| = \left\|ZL^{-T}L^{-1}Z^T f\right\| \leq \left\|L^{-1}\right\|^2 \|f\|. \tag{4.40}$$

4.3.2 Fully Linear RBF Models

Thus far we have maintained a very general RBF framework. We now focus on a more specific set of radial functions that satisfy two additional conditions:

- $\phi \in C^2[\mathbb{R}_+]$ and $\phi'(0) = 0$,
- ϕ conditionally positive definite of order 2 or less.

The first condition ensures that the resulting RBF model is twice continuously differentiable. The second condition is useful for restricting ourselves to models of the form (4.35) with a linear tail $P \in \mathcal{P}_1^n$.

For RBF models that are both twice continuously differentiable and have a linear tail, we have:

$$\begin{aligned} \nabla m(x_b + s) &= \sum_{\{y_i \in \mathcal{Y}: y_i \neq s\}} \lambda_i \phi'(\|s - y_i\|) \frac{s - y_i}{\|s - y_i\|} + \nabla P(s), && (4.41) \\ \nabla^2 m(x_b + s) &= \sum_{i=1}^{|\mathcal{Y}|} \lambda_i \Theta(\|s - y_i\|), && (4.42) \end{aligned}$$

with

$$\Theta(r) = \begin{cases} \frac{\phi'(\|r\|)}{\|r\|} I_n + \left(\phi''(\|r\|) - \frac{\phi'(\|r\|)}{\|r\|}\right) \frac{r}{\|r\|} \frac{r^T}{\|r\|}, & \text{if } r \neq 0, \\ \phi''(0) I_n & \text{if } r = 0, \end{cases} \tag{4.43}$$

where we have explicitly defined these derivatives for the special case when s is one of the interpolation knots in $\mathcal{Y}$.

The following lemma is a consequence of an unproven result in Oeuvray's dissertation [55], which we could not locate in the literature. It provides necessary and sufficient conditions on ϕ for the RBF model m to be twice continuously differentiable.

Lemma 4.3. *The model m defined in* (4.35) *is twice continuously differentiable on $\mathbb{R}^n$ if and only if $\phi \in C^2[\mathbb{R}_+]$ and $\phi'(0) = 0$.*

Proof. We begin by noting that the polynomial tail P and composition with the sum over $\mathcal{Y}$ are both smooth. Further, away from any of the points in $\mathcal{Y}$, m is clearly twice continuously differentiable if and only if $\phi \in C^2[\mathbb{R}_+]$. It now remains only to treat the case when $s = y_i \in \mathcal{Y}$.

For ∇m in (4.41) to be continuous, we note that since $\frac{s-y_i}{\|s-y_i\|}$ is always of bounded magnitude but the limit as $s \to y_i$ does not exist, we must have that ϕ' is continuous and vanishes at 0.

This condition is in fact necessary and sufficient for the continuity of $\nabla^2 m$. To see this, recall from L'Hôpitals rule in calculus that $\lim_{a\to 0} \frac{g(a)}{a} = g'(0)$, provided that $g(0) = 0$ and g is differentiable at 0. Applying this result with $g = \phi'$, we have that

$$\lim_{s\to y_i} \frac{\phi'(\|s-y_i\|)}{\|s-y_i\|} = \phi''(0).$$

Hence the second term in the expression for Θ in (4.43) vanishes as $r \to 0$, leaving only the first, that is $\lim_{r\to 0} \Theta(r) = \phi''(0) I_n$ exists. $\square$

Having established conditions for the twice differentiability of the radial portion of m in (4.35), we now focus on the linear tail P. Without loss of generality, we assume that the base point x_b is an interpolation point so that $y_1 = 0 \in \mathcal{Y}$.

Employing the standard linear basis and permuting the points we then have that the polynomial matrix $\Pi_{i,j} = \pi_i(y_j)$ is of the form:

$$\Pi = \begin{bmatrix} Y & 0 & y_{n+2} & \cdots & y_{|\mathcal{Y}|} \\ e^T & 1 & 1 & \cdots & 1 \end{bmatrix}, \tag{4.44}$$

where e is the vector of ones and Y denotes the matrix (4.9) of n particular nonzero points in $\mathcal{Y}$.

Recall that, in addition to the distinctness of the points in $\mathcal{Y}$, an equivalent condition for the nonsingularity of the RBF system (4.36) is that the first $n+1$ columns of Π in (4.44) are linearly independent. This latter condition is exactly the condition needed for the fully linear interpolation models in Section 4.1, where bounds for the matrix Y were provided.

In order to fit RBF models with linear tails into the globally convergent trust-region framework of Section 4.2, it remains only to show that the model Hessians are bounded by some fixed constant κ_H.

From (4.42) and (4.43), it is clear that the magnitude of the Hessian depends only on the quantities λ, $\left|\frac{\phi'(r)}{r}\right|$, and $|\phi''(r)|$. As an example, Table 4.2 provides bounds on the latter two quantities for the radial functions in Table 4.1 when r is restricted to lie in the interval $[0, \Delta]$. In particular, these bounds provide an upper bound for:

$$h_\phi(\Delta) = \max\left\{2\left|\frac{\phi'(r)}{r}\right| + |\phi''(r)| : r \in [0, \Delta]\right\}. \tag{4.45}$$

From (4.40) we also have a bound on λ provided that the appropriate Cholesky factor L is of bounded norm. We will bound $\|L^{-1}\|$ inductively by building up the interpolation set $\mathcal{Y}$ one point at a time. This inductive method lends itself well to a practical implementation and was inspired by the development in [8].

Table 4.2: Upper bounds on RBF components (Assumes $\gamma > 0$, $r \in [0, \Delta]$).

$\phi(r)$	$\lvert\phi(r)\rvert$	$\left\lvert\frac{\phi'(r)}{r}\right\rvert$	$\lvert\phi''(r)\rvert$
r^β, $\beta \in (2,4)$	Δ^β	$\beta\Delta^{\beta-2}$	$\beta(\beta-1)\Delta^{\beta-2}$
$(\gamma^2+r^2)^\beta$, $\beta \in (1,2)$	$(\gamma^2+\Delta^2)^\beta$	$2\beta(\gamma^2+\Delta^2)^{\beta-1}$	$2\beta(\gamma^2+\Delta^2)^{\beta-1}\left(1+\frac{2(\beta-1)\Delta^2}{\gamma^2+\Delta^2}\right)$
$-(\gamma^2+r^2)^\beta$, $\beta \in (0,1)$	$(\gamma^2+\Delta^2)^\beta$	$2\beta\gamma^{2(\beta-1)}$	$2\beta\gamma^{2(\beta-1)}$
$(\gamma^2+r^2)^{-\beta}$, $\beta > 0$	$\gamma^{-2\beta}$	$2\beta\gamma^{-2(\beta+1)}$	$2\beta\gamma^{-2(\beta+1)}$
$e^{-\frac{r^2}{\gamma^2}}$	1	$\frac{2}{\gamma^2}$	$\frac{2}{\gamma^2}$

To start this inductive argument, we assume that $\mathcal{Y}$ consists of $n+1$ points that are $\mathcal{P}_1^n$-unisolvent (possibly as a result of Algorithm 4.1). Given only these $n+1$ points, $\lambda = 0$ is the unique solution to (4.36) and hence the RBF model is linear. To include an additional point $y \in \mathbb{R}^n$ in the interpolation set $\mathcal{Y}$ (beyond the initial $n+1$ points), we appeal to the following Lemma.

Lemma 4.4. *Suppose that $\mathcal{Y}$ is such that Π is full rank and $LL^T = Z^T\Phi Z$ as in (4.39). If an arbitrary point $y \in \mathbb{R}^n$ is added to $\mathcal{Y}$, then the new Cholesky factor L_y is of the form*

$$L_y = \begin{bmatrix} L & 0 \\ (L^{-1}v_y)^T & \sqrt{\sigma_y - \|L^{-1}v_y\|^2} \end{bmatrix}. \tag{4.46}$$

Proof. Let the resulting RBF and polynomial matrices be denoted by

$$\Phi_y = \begin{bmatrix} \Phi & \phi_y \\ \phi_y^T & \phi(0) \end{bmatrix}, \qquad \Pi_y^T = \begin{bmatrix} \Pi^T \\ \pi(y) \end{bmatrix}, \tag{4.47}$$

respectively. 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:

$$Z_y = \begin{bmatrix} Z & Q\tilde{g} \\ 0 & \hat{g} \end{bmatrix}, \tag{4.48}$$

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

$$Z_y^T \Phi Z_y = \begin{bmatrix} Z^T \Phi Z & v_y \\ v_y^T & \sigma_y \end{bmatrix}, \tag{4.49}$$

for

$$v_y = Z^T \left(\Phi Q \tilde{g} + \phi_y \hat{g} \right) \tag{4.50}$$

$$\sigma_y = \tilde{g}^T Q^T \Phi Q \tilde{g} + 2\tilde{g}^T Q^T \phi_y \hat{g} + \phi(0)\hat{g}^2. \tag{4.51}$$

It can now be directly verified that $L_y L_y^T = Z_y^T \Phi Z_y$. $\square$

Only the last row of the Cholesky factor is affected by the addition of the new point y. We also note that the key quantity of interest is the lower right component of (4.46),

$$\tau(y) = \sqrt{\sigma_y - \|L^{-1} v_y\|^2}, \tag{4.52}$$

which must be bounded away from zero in order for the matrix of interest,

$$L_y^{-1} = \begin{bmatrix} L^{-1} & 0 \\ \frac{-v_y^T L^{-1} L^{-T}}{\tau(y)} & \frac{1}{\tau(y)} \end{bmatrix}, \tag{4.53}$$

to exist and be suitably well-conditioned. The following Lemma bounds the norm of the resulting Cholesky factor L_y^{-1} as a function of the previous factor L^{-1}, v_y, and the quantity $\tau(y)$.

```
4.4.0. Input $\mathcal{D} = \{d_1, \dots, d_{|\mathcal{D}|}\} \subset \mathbb{R}^n$, $\mathcal{Y}$ consisting of $n+1$ sufficiently affinely
       independent points, constants $\theta > 0$, $\Delta > 0$, and $p_{\max} \geq n+1$.
4.4.1. Using $\mathcal{Y}$, compute the Cholesky factorization $LL^T = Z^T \Phi Z$ as in (4.39).
4.4.2. For all $y \in \mathcal{D}$ such that $\|y\|_k \leq \Delta$:
       Compute $\Phi_y$ and $\Pi_y$ (4.47), $L_y$ (4.46), and $\tau(y)$ (4.52)
       If $\tau(y) \geq \theta$,
           $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{y\}$,
           Update $Z \leftarrow Z_y$, $L \leftarrow L_y$,
           If $|\mathcal{Y}| = p_{\max}$, return.
```

Algorithm 4.4: AddPoints($\mathcal{D}, \mathcal{Y}, \theta, \Delta, p_{\max}$): Algorithm for adding additional interpolation points.

Lemma 4.5. *If* $\|L^{-1}\| \leq \kappa$ *and* $\tau(y) \geq \theta > 0$, *then:*

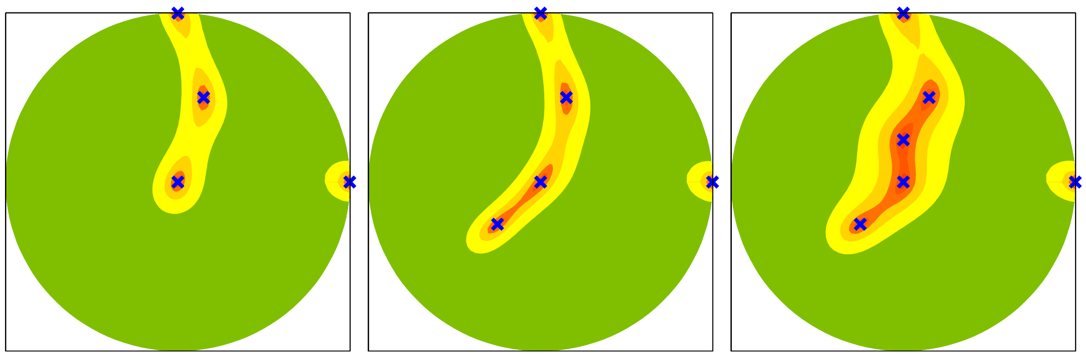
$$\left\|L_y^{-1}\right\|^2 \leq \kappa + \frac{1}{\theta^2}\left(1 + \|v_y\|\kappa^2\right)^2 \tag{4.54}$$

Proof. Let $w_y = (w, \tilde{w}) \in R^{|\mathcal{Y}|+1}$ be an arbitrary vector with $\|w_y\| = 1$. Then

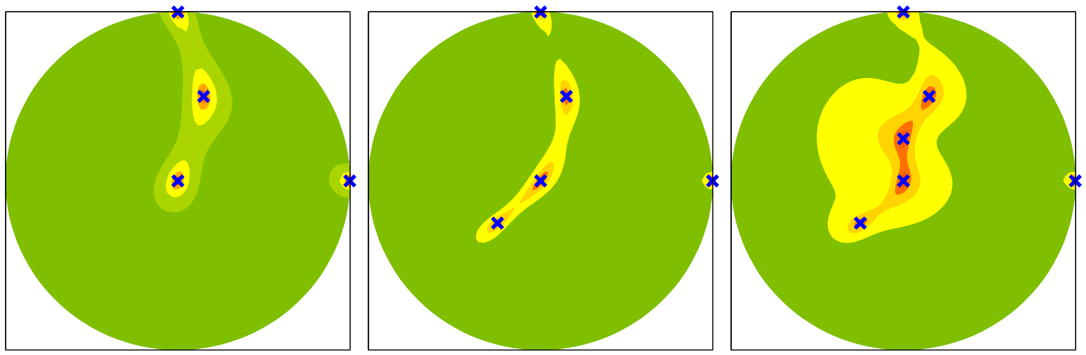
$$\begin{aligned}
\left\|L_y^{-1} w_y\right\|^2 &= \left\|L^{-1} w\right\|^2 + \frac{1}{\tau(y)^2}\left(\tilde{w} - v_y^T L^{-T} L^{-1} w\right)^2 \\
&\leq \kappa + \frac{1}{\theta^2}\left(\tilde{w}^2 - 2\tilde{w} v_y^T L^{-T} L^{-1} w + \left(v_y^T L^{-T} L^{-1} w\right)^2\right) \\
&\leq \kappa + \frac{1}{\theta^2}\left(1 + 2\left\|L^{-1} v_y\right\| \left\|L^{-1} w\right\| + \left(\left\|L^{-1} v_y\right\| \left\|L^{-1} w\right\|\right)^2\right) \\
&\leq \kappa + \frac{1}{\theta^2}\left(1 + \|v_y\|\kappa^2\right)^2.
\end{aligned}$$

□

Lemma 4.55 suggests the procedure given in Algorithm 4.4. We use Algorithm 4.4 to iteratively add points that have previously been evaluated to the interpolation set. Before this algorithm is called, we assume that the interpolation set consists of $n+1$ sufficiently affinely independent points generated as described in Section 4.1 and hence the initial L matrix is empty.



(a) $\tau(y)^{-1}$



(b) $\left\|L_y^{-1}\right\|^2$

Figure 4.2: Contours for (a) $\tau(y)^{-1}$ (4.52) and (b) $\left\|L_y^{-1}\right\|^2$ (4.53) values for a multiquadric RBF surface interpolating 4, 5, and 6 points in $\mathbb{R}^2$ (logarithmic scale). The quantity shown grows as the shading progresses from green to red.

Figure 4.2 (a) gives an example of $\tau(y)^{-1}$ values for different interpolation sets in $\mathbb{R}^2$. In particular we note that $\tau(y)$ approaches zero as y approaches any of the points already in the interpolation set $\mathcal{Y}$. Figure 4.2 (b) shows the behavior of $\left\|L_y^{-1}\right\|^2$ for the same interpolation sets and illustrates the relative correspondence between the values of $\tau(y)^{-1}$ and $\left\|L_y^{-1}\right\|^2$.

We now assume that both $\mathcal{Y}$ and the point y being added to the interpolation set belong to some bounded domain $\{x \in \mathbb{R}^n : \|x\|_k \leq \Delta\}$. Thus the quantities $\{\|x - z\| : x, z \in \mathcal{Y} \cup \{y\}\}$ are all of magnitude no more than $2c_1\Delta$ since $\|\cdot\| \leq c_1 \|\cdot\|_k$. The elements in $\Phi_{i,j} = \phi(\|y_i - y_j\|)$ and $\phi_y =$

$[\phi(\|y-y_1\|),\ldots,\phi(\|y-y_{|\mathcal{Y}|}\|)]^T$ are bounded by $k_\phi(2c_1\Delta)$, where:

$$k_\phi(2c_1\Delta) = \max\{|\phi(r)| : r \in [0, 2c_1\Delta]\}. \tag{4.55}$$

Bounds for the specific ϕ functions of the radial basis functions of interest are provided in Table 4.2. Using the isometry of Z_y we hence have the bound:

$$\|v_y\| \leq \sqrt{|\mathcal{Y}|(|\mathcal{Y}|+1)}k_\phi(2c_1\Delta), \tag{4.56}$$

independent of where in $\{x \in \mathbb{R}^n : \|x\|_k \leq \Delta\}$ the point y lies, which can be used in (4.54) to bound $\left\|L_y^{-1}\right\|$.

The following Theorem gives the resulting bound.

Theorem 4.3. *Let $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_b\|_k \leq \Delta\}$. Let $\mathcal{Y} \subset \mathcal{B} - x_b$ be a set of distinct interpolation points, $n+1$ of which are affinely independent and $|f(x_b + y_i)| \leq f_{\max}$ for all $y_i \in \mathcal{Y}$. Then for a model of the form (4.35), with a bound h_ϕ as in (4.45), interpolating f on $x_b + \mathcal{Y}$, we have that for all $x \in \mathcal{B}$:*

$$\left\|\nabla^2 m(x)\right\| \leq |\mathcal{Y}| \left\|L^{-1}\right\|^2 h_\phi(2c_1\Delta) f_{\max} =: \kappa_H. \tag{4.57}$$

Proof. Let $r_i = s - y_i$ and note that when s and $\mathcal{Y}$ both belong to $\mathcal{B} - x_b$, $\|r_i\| \leq c_1 \|r_i\|_k \leq 2c_1\Delta$ for $i = 1, \ldots, |\mathcal{Y}|$. Thus for an arbitrary vector w with $\|w\| = 1$:

$$\begin{aligned}
\left\|\nabla^2 m(x_b+s)w\right\| &\leq \sum_{i=1}^{|\mathcal{Y}|} |\lambda_i| \left\| \frac{\phi'(\|r_i\|)}{\|r_i\|} w + \left(\phi''(\|r_i\|) - \frac{\phi'(\|r_i\|)}{\|r_i\|}\right) \frac{r_i^T w}{\|r_i\|} \frac{r_i}{\|r_i\|} \right\|, \\
&\leq \sum_{i=1}^{|\mathcal{Y}|} |\lambda_i| \left[2\left|\frac{\phi'(\|r_i\|)}{\|r_i\|}\right| + |\phi''(\|r_i\|)| \right] \\
&\leq \|\lambda\|_1 h(2c_1\Delta) \\
&\leq \sqrt{|\mathcal{Y}|} \left\|L^{-1}\right\|^2 \|f\| \, h(2c_1\Delta),
\end{aligned}$$

where the last two inequalities follow from (4.45) and (4.40), respectively. Noting that $\|f\| \leq \sqrt{|\mathcal{Y}|} f_{\max}$ gives the desired result. □

```
4.5.0. Input $\mathcal{D} =\subset \mathbb{R}^n$, constants $\theta_2 > 0$, $\theta_4 \geq \theta_3 \geq 1$, $\theta_1 \in (0, \frac{1}{\theta_3}]$, $\Delta_{\max} \geq \Delta_k > 0$, and $p_{\max} \geq n+1$.
4.5.1. Seek affinely independent interpolation set $\mathcal{Y}$ by running AffPoints($\mathcal{D}, \theta_1, \theta_3\Delta_k$) (Algorithm 4.1),
       Save $z_1$ as a model-improving direction for use in Step 4.2.5. of Algorithm 4.2.
       If $|\mathcal{Y}| < n+1$ (and hence $m_k$ is not fully linear):
           Seek to get $n+1-|\mathcal{Y}|$ additional points in $\mathcal{Y}$ by running AffPoints($\mathcal{D}, \theta_1\frac{\theta_3}{\theta_4}, \theta_4\Delta_{\max}$) (Algorithm 4.1).
           If $|\mathcal{Y}| < n+1$, evaluate $f$ at remaining $n+1-|\mathcal{Y}|$ model points so that $|\mathcal{Y}| = n+1$.
4.5.2. Attempt to use additional points by running AddPoints($\mathcal{D}$,$\mathcal{Y}$,$\theta_2$,$\theta_4\Delta_{\max}$,$p_{\max}$) (Algorithm 4.4),
4.5.3. Obtain model parameters by (4.37) and (4.38).
```

Algorithm 4.5: RBFModel($\mathcal{D}, \theta_1, \theta_2, \theta_3, \theta_4, \Delta_k, \Delta_{\max}, p_{\max}$): Algorithm for constructing model m_k.

Algorithm 4.5 summarizes the RBF model formation we use in Step 4.2.6. of Algorithm 4.2. Several details of this algorithm demand further explanation.

Note that in Algorithm 4.5 we require that the interpolation points in $\mathcal{Y}$ lie within some constant factor of the largest trust-region $\Delta_{\max}$. In particular, all of the points in $\mathcal{Y}$ are contained in

$$\mathcal{B}_{\max} = \{y \in \mathbb{R}^n : \|y\|_k \leq \theta_4\Delta_{\max}\}. \tag{4.58}$$

This region is chosen to be larger than the current trust-region so that the algorithm can make use of more points previously evaluated in the course of the optimization.

In Algorithm 4.5 we declare a model to be fully linear if $n+1$ points within $\{y \in \mathbb{R}^n : \|y\|_k \leq \theta_3\Delta_k\}$ result in pivots larger than θ_1, where the positive constant θ_1 is chosen so as to be attainable by the model directions (scaled by Δ_k) discussed in Section 4.1.

If not enough points are found, the model will not be fully linear, but we expand the search for affinely independent points within the larger region $\mathcal{B}_{\max}$. If still fewer than $n+1$ points are available, we must evaluate at the directions in Z obtained by Algorithm 4.1 to ensure that $\mathcal{Y}$ is $\mathcal{P}_1^n$-unisolvent.

Finally, additional available points within $\mathcal{B}_{\max}$ are added to the interpolation set $\mathcal{Y}$ provided that they keep $\tau(y) \geq \theta_2 > 0$, until a maximum of $p_{\max}$ points are in $\mathcal{Y}$.

Since we have assumed f to be bounded on $L(x_0)$ and that $\mathcal{Y} \subset \mathcal{B}_{\max}$, the bound (4.57) holds for all models used by the trust-region algorithm, regardless of whether or not they are fully linear. Provided that the radial function ϕ is chosen to satisfy the requirements of Lemma 4.3, m will be twice continuously differentiable and hence it follows that ∇m is Lipschitz continuous on $\mathcal{B}_{\max}$ and κ_H in (4.25) is one possible Lipschitz constant. When combined with the results of Section 4.1 showing that such interpolation models can be made fully linear in a finite procedure, we have that Theorem 4.2 guarantees that $\lim_{k\to\infty} \nabla f(x_k) = 0$ for trust-region algorithms using these RBFs.

4.4 Computational Experiments

In order to better understand what effect model selection has in the algorithmic framework of the previous section, we now present the results of several numerical experiments.

We will follow the benchmarking procedures detailed in [52] and Chapter 2, where the proposed derivative-free convergence test is

$$f(x_0) - f(x) \geq (1-\tau)(f(x_0) - f_L), \tag{4.59}$$

where $\tau > 0$ is a tolerance, x_0 is the starting point, and f_L is the smallest value of f obtained by any tested solver within a fixed number, μ_f, of function evaluations. We note that in (4.59), a problem is "solved" when the achieved reduction from the initial value, $f(x_0) - f(x)$, is at least $1 - \tau$ times the best possible reduction, $f(x_0) - f_L$.

We use this convergence test when benchmarking a set of solvers $\mathcal{S}$ on a set of problems $\mathcal{P}$. For each $s \in \mathcal{S}$ and $p \in \mathcal{P}$, we define $t_{p,s}$ as the number of function evaluations required by s to satisfy the convergence test (4.59) on p, with the convention that $t_{p,s} = \infty$ if s does not satisfy the convergence test on p within μ_f evaluations.

If we assume that:

(i) the differences in times for solvers to determine a point for evaluation of $f(x)$ are negligible relative to the time to evaluate the function, and

(ii) the function requires the same amount of time to evaluate at any point in its domain,

then differences in the measure $t_{p,s}$ will roughly correspond to differences in computing time. Assumption (i) is reasonable for the computationally expensive simulation-based problems, which is the primary target of ORBIT.

Given this measure, we define the *data profile* $d_s(\alpha)$ for solver $s \in \mathcal{S}$ as:

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} : \frac{t_{p,s}}{n_p + 1} \leq \alpha \right\} \right|, \tag{4.60}$$

where n_p is the number of variables in problem $p \in \mathcal{P}$. We note that the data profile $d_s : \mathbb{R} \rightarrow [0, 1]$ is a nondecreasing step function that is independent of the data profiles of the other solvers $\mathcal{S} \backslash \{s\}$. By this definition, $d_s(\kappa)$ is the percentage of problems that can be solved within κ *simplex gradient estimates.*

4.4.1 Smooth Test Problems

We begin by considering the test set $\mathcal{P}_S$ of 53 smooth nonlinear least squares problems defined in [52] and Chapter 2. Each unconstrained problem is defined by a starting point x_0 and a function

$$f(x) = \sum_{i=1}^{k} f_i(x)^2, \tag{4.61}$$

comprised of a set of smooth components. The functions vary in dimension from $n = 2$ to $n = 12$, with the 53 problems being roughly uniformly distributed across these dimensions. The maximum number of function evaluations is set to $\mu_f = 1300$ so that at least the equivalent of 100 simplex gradient estimates could be obtained on all the problems in $\mathcal{P}_S$. The initial trust-region Δ_0 was chosen for each problem based on the initial point so that:

$$\Delta_0 = \max \{1, \|x_0\|_\infty\}. \tag{4.62}$$

The ORBIT implementation illustrated here relies on a 2-norm trust-region with the following parameter values (as in [75] and Chapter 3): $\eta_0 = 0$, $\eta_1 = .2$, $\gamma_0 = \frac{1}{2}$, $\gamma_1 = 2$, $\Delta_{\max} = 10^3 \Delta_0$, $\epsilon = 10^{-10}$, $\kappa_d = 10^{-4}$, $\alpha = .9$, $\mu = 2000$, $\beta = 1000$, $\theta_1 = 10^{-3}$, $\theta_2 = 10^{-7}$, $\theta_3 = 10$, $\theta_4 = \max(\sqrt{n}, 10)$. In addition to the backtracking line search detailed here, we used an augmented Lagrangian method to find an approximate solution to the trust-region subproblem.

The first solver set we consider is the set $\mathcal{S}_A$ consisting of three different radial basis function types for ORBIT:

Multiquadric : $\phi(r) = -\sqrt{\gamma^2 + r^2}$, $\gamma = 1$, with $p_{\max} = 2n + 1$;

Cubic : $\phi(r) = r^3$, with $p_{\max} = 2n + 1$;

Gaussian : $\phi(r) = e^{-\frac{r^2}{\gamma^2}}$, $\gamma = 1$, with $p_{\max} = 2n + 1$,

where the common theme among these models is that they interpolate at most $p_{\max} = 2n + 1$ points. The number $2n + 1$ was chosen since this is the number of interpolation points recommended by Powell for the NEWUOA algorithm [61]. We tested alternative values of the parameter γ used by multiquadric and Gaussian RBFs but found that $\gamma = 1$ worked well for both.

In Figure 4.3 we present the data profiles for $\tau = 10^{-k}$ with $k \in \{1, 3, 5, 7\}$. Recall that, for example, $\tau = .1$ corresponds to a 90% reduction relative to the best possible reduction in $\mu_f = 1300$ function evaluations. As discussed in [52] and Chapter 2, data profiles are used to see which solver is likely to achieve a given reduction of the function within a specific computational budget. For example, when the equivalent of 15 simplex gradients ($15(n + 1)$ function evaluations) are available, we see that the cubic, multiquadric, and Gaussian variants are respectively able to solve 60%, 54%, and 49% of problems to an accuracy level of $\tau = 10^{-3}$.

For all four accuracy levels shown, we see that the cubic variant is generally best (especially given relatively small budgets) while the Gaussian variant is generally worst. Not surprisingly, these differences are smaller than those seen in the benchmarking work in [52] and Chapter 2, where three very different solvers were tested.

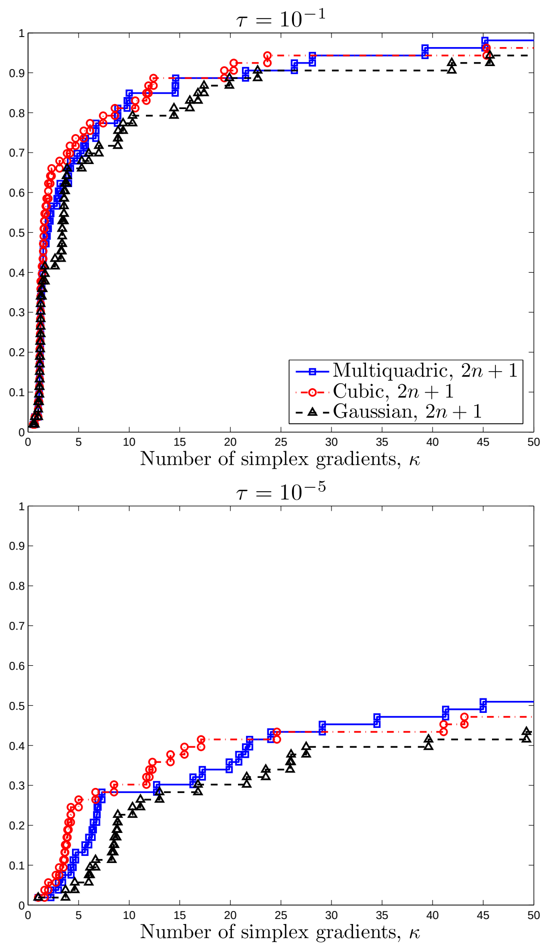


Figure 4.3: Data profiles $d_s(\kappa)$ for different RBF types with $p_{\max} = 2n+1$ on the smooth problems $\mathcal{P}_S$. These profiles show the percentage of problems solved as a function of a computational budget of simplex gradients.

The second solver set, $\mathcal{S}_B$, consists of the same three radial basis function types:

Multiquadric : $\phi(r) = -\sqrt{\gamma^2 + r^2}$, $\gamma = 1$, with $p_{\max} = \frac{(n+1)(n+2)}{2}$;

Cubic : $\phi(r) = r^3$, with $p_{\max} = \frac{(n+1)(n+2)}{2}$;

Gaussian : $\phi(r) = e^{-\frac{r^2}{\gamma^2}}$, $\gamma = 1$, with $p_{\max} = \frac{(n+1)(n+2)}{2}$,

but with the maximum number of points being interpolated now set to $p_{\max} =$

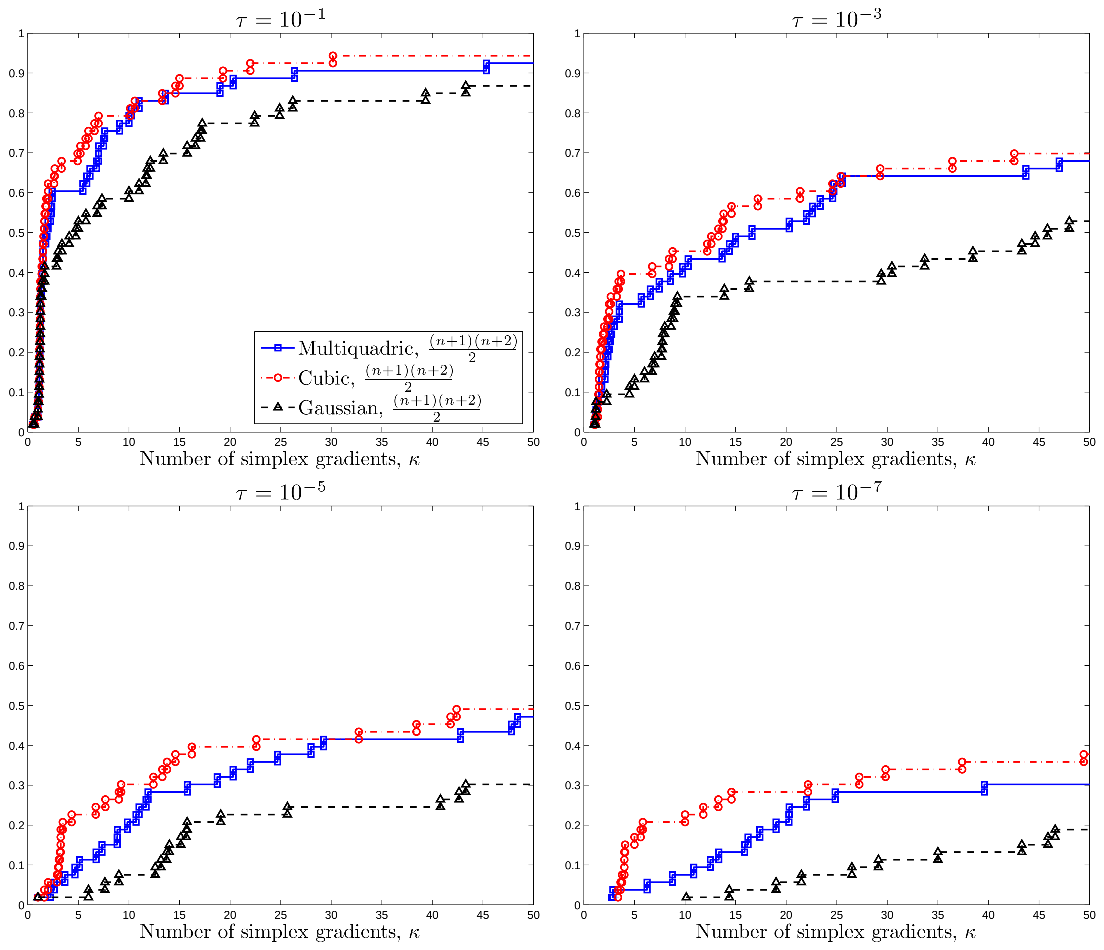


Figure 4.4: Data profiles $d_s(\kappa)$ for different RBF types with $p_{\max} = \frac{(n+1)(n+2)}{2}$ on the smooth problems $\mathcal{P}_S$. These profiles show the percentage of problems solved as a function of a computational budget of simplex gradients.

$\frac{(n+1)(n+2)}{2}$, corresponding to the number of points needed to uniquely fit an interpolating quadratic model.

Figure 4.4 shows the data profiles for the same accuracy levels considered in Figure 4.3. Again we see that the cubic variant is generally best (especially given relatively small budgets) while the Gaussian variant is generally the worst, but there are now larger differences between the three variants. When the equivalent of 15 simplex gradients are available, we see that the cubic, multiquadric, and

Gaussian variants are respectively able to now solve 56, 47, and 36% of problems to an accuracy level of $\tau = 10^{-3}$. We note that the raw data shown in Figure 4.4 should not be quantitatively compared against that shown in Figure 4.3 since the best function value found for each problem is obtained from only the solvers tested (in $\mathcal{S}_A$ or $\mathcal{S}_B$) and hence the convergence tests will differ.

Our final test on these smooth test problems is to compare between the best variants for the two different maximum numbers of interpolation points. Thus the third solver set, $\mathcal{S}_C$, consists of ORBIT with the following RBFs:

Cubic A : $\phi(r) = r^3$, with $p_{\max} = 2n + 1$;

Cubic B : $\phi(r) = r^3$, with $p_{\max} = \frac{(n+1)(n+2)}{2}$.

Figure 4.5 shows that these two variants perform comparably. As expected, the difference between the two variants tends to grow as the number of function evaluations grows, with the variant that is able to interpolate more points performing better than the variant interpolating at most $2n+1$ points. The $\frac{(n+1)(n+2)}{2}$ variant also performs better when higher accuracy levels are demanded, and we attribute this to the fact that the model interpolating more points is generally a better approximation of the function f. The main downside of interpolating more points is that the linear systems in Section 4.3 will also grow, resulting in a higher linear algebraic cost per trust-region iteration. As we will see in the next set of tests, for many applications, this cost may be viewed as negligible relative to the cost of evaluating the function f.

We are, however, surprised to see that Cubic A, the $2n + 1$ variant, performs better for several ranges of smaller budgets. For example, this variant performs slightly better between 5 and 15 simplex gradient estimates when $\tau = 10^{-3}$, and

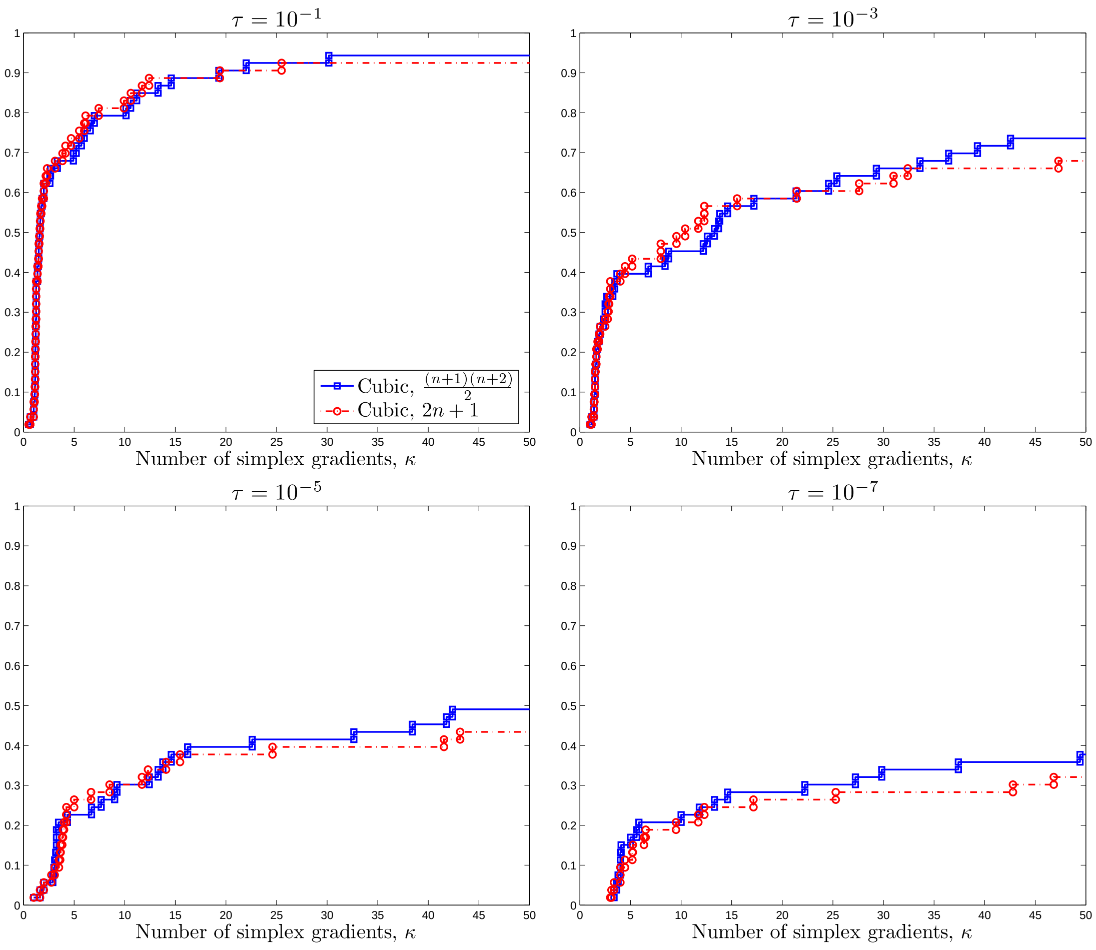


Figure 4.5: The effect of changing the maximum number of interpolation points, $P_{\max}$, on the data profiles $d_s(\kappa)$ for the smooth problems $\mathcal{P}_S$.

between 4 and 9 simplex gradient estimates when $\tau = 10^{-5}$. Since the initial $n+1$ evaluations are common to both variants and the parameter $p_{\max}$ has no effect on the subroutine determining the sufficiently affinely independent points, we might expect that the variant interpolating more points would do at least as well as the variant interpolating fewer points.

Further results comparing ORBIT (in 2-norm and ∞-norm trust-regions) against NEWUOA on a set noisy test problems are provided in [75] and Chapter 3.

4.4.2 Environmental Application(s)

We now illustrate the use of RBF models on a computationally expensive application problem.

The Blaine Naval Ammunition Depot comprises 48,800 acres just east of Hastings, Nebraska. In the course of producing nearly half of the Naval ammunition used in World War II, much toxic waste was generated and disposed of on the site. Among other contaminants, both Trichloroethylene (TCE), a probable carcinogen, and Trinitrotoluene (TNT), a possible carcinogen, are present in the groundwater.

As part of a collaboration [5, 79] between environmental consultants, academic institutions, and governmental agencies, several optimization problems were formulated. Here we focus on one of the simpler formulations, where we have control over 15 injection and extraction wells located at fixed positions within the site. At each of these wells we can either inject clean water or extract contaminated water, which is then treated. Each instance of the decision variables hence corresponds to a pumping strategy that will run over a 30-year time horizon. For scaling purposes, each variable is scaled so that range of realistic pumping rates maps to the interval $[0, 1]$.

The objective is to minimize the cost of the pumping strategy (the electricity needed to run the pumps) plus a penalty associated with exceedance of the constraints on maximum allowable concentration of TCE and TNT over the 30-year planning horizon. For each pumping strategy, these concentrations are obtained by running a pair of coupled simulators, MODFLOW 2000 [72] and MT3D [80], which simulate the underlying contaminant transport and transformation. For a given set of pumping rates, this process required more than 45 minutes on a Pentium 4 dual-core desktop.

Comparison Solvers Tested

In the spirit of [52] (Chapter 1), in addition to ORBIT we considered three solvers designed to solve unconstrained serial optimization problems using only function values. The solvers considered were thus:

NMSMAX is an implementation of the Nelder-Mead method and is due to Higham [34]. We specified that the initial simplex have sides of length Δ_0. Since NMSMAX is defined for maximization problems, it was given the negative of f.

SID-PSM is a pattern search solver due to Custódio and Vicente [19]. It is especially designed to make use of previous function evaluations. The initial step size was set to Δ_0.

ORBIT used the same parameter values as used on the test functions, with a cubic RBF, initial trust-region radius $\Delta_0 = .1$, and maximum number of interpolation points $p_{\max} = \frac{(n+1)(n+2)}{2}$.

Each of these three solvers also require a starting point x_0 and a maximum number of allowable function evaluations, μ_f. A common selection of $\Delta_0 = .1$ was made to standardize the initial evaluations across the collection of solvers, hence each solver except SID-PSM evaluated the same initial $n+1$ points. SID-PSM moves off this initial pattern of points once it sees a reduction. All other inputs were set to their default values except that we effectively set all termination parameters to zero to ensure that the solvers only terminate after exhausting the budget μ_f function evaluations.

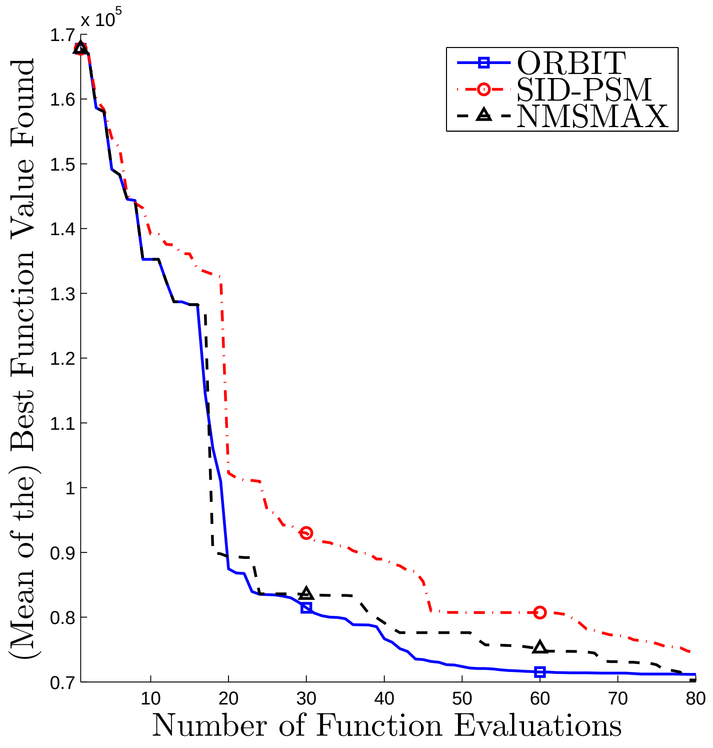


Figure 4.6: Mean (in 8 trials) of the best function value found for the first 80 evaluations on the Blaine problem.

The maximum number of function evaluations was chosen to be $\mu_f = 10(n + 1) = 160$. Since each evaluation requires more than 45 minutes, this means that a single run of one solver requires roughly 5 CPU days. Since this problem is noisy, we chose to run each solver from the same 8 starting points generated uniformly at random within the hypercube $[0, 1]^{15}$ of interest. Thus each solver trajectory over these 8 starting points required roughly 6 CPU weeks to obtain.

Figure 4.6 shows the average of the best function value obtained over the course of the first 80 function evaluations. We note that, by design, all solvers start from the same function value. The ORBIT solver does best initially, obtaining a function value of 72000 within 50 evaluations. The ORBIT trajectory quickly flattens out as it is the first to find a local minima with an average value of 70500. However, in this case the solver (on average) gets trapped in a local minimum that has a higher function value than the local minimum found by the NMSMAX and SID-PSM solvers after $\mu_f = 160$ evaluations. Hence, on these tests, NMSMAX and SID-PSM are especially good at finding a good minimum for a noisy function. Given $\mu_f = 160$ evaluations, NMSMAX finds a point with $f \approx 67000$ and SID-PSM finds a point with $f \approx 69500$.

The Blaine problem highlights the fact that solvers will have different performance on different functions, and that many application problems contain computational noise. This noise can prevent globally convergent local methods from finding good solutions. Comparisons between ORBIT and alternative derivative-free algorithms on two different problems from environmental engineering can be found in [75] and Chapter 3.

4.5 Conclusions

In this chapter we have introduced and analyzed first-order derivative-free trust-region algorithms based on radial basis functions, which are globally convergent. We first showed that, provided a function and a model are sufficiently smooth, interpolation on a set of sufficiently affinely independent points is enough to guarantee Taylor-like error bounds for both the model and its gradient. In Section 4.3 we introduced procedures for bounding a radial basis function model's Hessian. These two results allowed us to take advantage of the recent derivative-free trust-region framework in [15] to show convergence of algorithms using very general radial basis function models to stationary points.

The central element of a radial basis function is the radial function $\phi : \mathbb{R}_+ \to$

$\mathbb{R}$. We have illustrated the results with a few different types of radial functions, however, the results presented here are wide-reaching, requiring only the following conditions on ϕ:

1. ϕ is twice continuously differentiable on $[0, u)$, for some $u > 0$,
2. $\phi'(0) = 0$, and
3. ϕ is conditionally positive definite of order 2.

While the last condition seems to be the most restrictive, it is actually only the first that eliminates the thin-plate spline radial function $\phi(r) = r^2 \log(r)$, popular in other applications of RBFs, from our analysis.

Our numerical results are aimed at illustrating the effect of using different types of radial functions ϕ in the ORBIT algorithm [75]. We saw that the cubic radial function slightly outperformed the multiquadric radial function, while the Gaussian radial function performed worse. These results are interesting because Gaussian radial basis functions are the only ones among those tested that are conditionally positive definite of order 0, requiring neither a linear or constant term to uniquely interpolated scattered data. Gaussian RBFs are usually used in kriging [18], which forms the basis for the global optimization methods in [38, 39].

We also ran ORBIT on a computationally expensive environmental engineering problem, requiring 5 CPU days for a single run of $\mu_f = 160$ evaluations. On this problem we saw that ORBIT obtained a good solution within 50 expensive evaluations and quickly found a local minimum.

While the focus of this work was primarily on the theoretical implications associated with using radial basis function models in a trust-region framework, in the future we intend to pursue “large-step” variants of ORBIT designed to step over noise such as that encountered in the Blaine problem. This problem has also motivated our development of global optimization methods in Chapter 6. We also intend to work on parallel implementations akin to APPS Pattern Search [37].

Lastly, we note that the theory presented here can also be extend to models of other forms. We mention quadratics in Chapter 5, but note that we could also have used higher order polynomial tails for better approximation bounds. For example, methods using a suitably conditioned quadratic detail could be expected to converge to second-order local minima. However, these methods require function values at many more points than is often feasible for the computationally expensive functions driving our work.

Acknowledgments

The authors are grateful to Amandeep Singh for providing the code for the application problem and to Rommel Regis for comments on an earlier draft. This work was supported by a Department of Energy Computational Science Graduate Fellowship, grant number DE-FG02-97ER25308 and by NSF grants BES-022917, CBET-0756575, CCF-0305583, and DMS-0434390.

CHAPTER 5

OPTIMIZATION BY MINIMUM NORM HESSIAN (MNH) QUADRATICS*

In this chapter we address unconstrained optimization,

$$\min \{ f(x) : x \in \mathbb{R}^n \}, \tag{5.1}$$

of a function whose derivatives are unavailable. Our work is motivated by functions that are computationally expensive to evaluate, usually as a result of the need to run some underlying complex simulation model. These simulations often provide the user solely with the simulation output, creating the need for a *derivative-free* optimization algorithm. Examples of derivative-free optimization applied to these types of problems in electrical, environmental, and biomedical engineering can be found in [37, 55, 75] and Chapters 3 and 4.

When f is computationally expensive, a user is typically constrained by a computational budget that limits the number of function evaluations available to the optimization algorithm. We view the data gained from each function evaluation as contributing to a *bank* of insight into the function. As the optimization is carried out, more points are evaluated and this bank will grow. How to most effectively manage the data contained in the bank is a central driving force behind this chapter.

Our approach is inspired by the recent work of Powell [60, 61] using quadratic models interpolating fewer than a quadratic (in the dimension n) number of points. This strategy allows the underlying optimization to begin sooner and make more rapid progress in fewer function evaluations. These models are assumed to locally

*THIS CHAPTER IS AN EXPANDED VERSION OF THE CONFERENCE PAPER [74].

approximate the function while being computationally inexpensive to evaluate and optimize over.

In this chapter we introduce a new algorithm, MNH, that contributes two new features. First, unlike previous algorithms [55, 61], which were driven by a desire to keep linear algebraic overhead to $\mathcal{O}(n^3)$ operations per iteration, our algorithm views overhead as negligible relative to the expense of function evaluation. This allows greater flexibility in using points from the bank.

Second, our models are formed from interpolation sets in a computationally stable manner which guarantees that the models are well-behaved. In fact, both our model and its gradient are able to approximate the function and its gradient arbitrarily well. Consequently, the recent convergence results in [15] and Chapter 4 guarantee that our algorithm will converge to first-order critical points. Our goal in this chapter is to extend the framework using RBFs in ORBIT to an algorithm relying on quadratic models.

Encouraged by preliminary results, we hope that this convergence result and our way of using points from the bank will yield a theoretically sound algorithm that is both relatively simple and works well in practice.

This chapter is organized as follows. In Section 5.1 we review derivative-free trust-region algorithms. Section 5.2 introduces the special quadratic models employed by our algorithm. The MNH algorithm is discussed in Section 5.3 and preliminary numerical findings are presented in Section 5.4.

5.1 Derivative-Free Trust-Region Methods

Our algorithm is built upon a trust-region framework that we now review. A trust-region method is an iterative method that optimizes over a surrogate model m_k assumed to approximate f within a neighborhood of the current iterate x_k, the *trust-region*

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

for a radius $\Delta_k > 0$. New candidate points are obtained by solving the subproblem

$$\min \left\{ m_k(x_k + s) : x_k + s \in \mathcal{B}_k \right\}. \tag{5.2}$$

In fact, it suffices to only solve (5.2) approximately, provided that the resulting step s_k satisfies a sufficient decrease condition. After the function is evaluated at $x_k + s_k$, the pair (x_k, Δ_k) is updated according to the ratio of actual to predicted decrease,

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

ρ_k values close to 1 corresponding to good model prediction.

Given an initial point x_0 and a maximum radius $\Delta_{\max}$, the design of the trust-region algorithm ensures that f is only sampled within the relaxed level set

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

A quadratic model,

$$m_k(x_k + s) = f(x_k) + g_k^T s + \frac{1}{2} s^T H_k s, \tag{5.3}$$

is typically employed, with $g_k = \nabla f(x_k)$ and $H_k = \nabla^2 f(x_k)$ when these derivatives are available. The quadratic model in (5.3) is attractive because global solutions to the subproblem in (5.2) can then be efficiently computed [51]. When the gradient

∇f is exactly available, global convergence to local minima is possible under mild assumptions. Full treatment is given in [12].

When only function values are available, the model m_k can be obtained by interpolating the function at a set of distinct data points $\mathcal{Y} = \{y_1 = 0, y_2, \dots, y_{|\mathcal{Y}|}\} \subset \mathbb{R}^n$:

$$m_k(x_k + y_j) = f(x_k + y_j) \qquad \text{for all } y_j \in \mathcal{Y}. \tag{5.4}$$

This approach was taken with both quadratic [13, 58] and radial basis function (RBF) [55, 75] models (see also Chapter 3).

A primary concern in the study of interpolation model-based derivative-free methods is the quality of the model within $\mathcal{B}_k$. In [16], Taylor-like error bounds are established based on the geometry of the interpolation set $\mathcal{Y}$. These results motivate a class of so-called *fully linear* models for approximating functions that are reasonably smooth. In particular, we will assume that $f \in C^1[\Omega]$ for some open $\Omega \supset \mathcal{L}(x_0)$, ∇f is Lipschitz continuous on $\mathcal{L}(x_0)$, and f is bounded on $\mathcal{L}(x_0)$.

Definition 5.1. *For fixed $\kappa_f, \kappa_g > 0$ and $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_k\| \leq \Delta\}$, a model $m \in C^1[\Omega]$ is said to be $\underline{\text{fully linear (f.l.) on } \mathcal{B}}$ if for all $x \in \mathcal{B}$:*

$$|f(x) - m(x)| \leq \kappa_f \Delta^2, \tag{5.5}$$

$$\|\nabla f(x) - \nabla m(x)\| \leq \kappa_g \Delta. \tag{5.6}$$

The two conditions in Definition 5.1 ensure that approximations to the function and its gradient can achieve any desired degree of precision within a small enough neighborhood of x_k. Provided that m_k can be made fully linear (for fixed κ_f and κ_g) in finitely many steps, Algorithm 5.1 was recently shown to be globally convergent to a stationary point $\nabla f(x_*) = 0$, given an appropriate criticality test [15].

```
Input $x_0 \in \mathbb{R}^n$, $0 < \Delta_0 \le \Delta_{\max}$, $m_0$, $0 \le \eta_0 \le \eta_1 < 1$ ($\eta_1 \neq 0$), $0 < \gamma_0 < 1 < \gamma_1$,
$\epsilon_g > 0$, $\kappa_d \in (0,1)$.
Iteration $k \ge 0$:

5.1.1 If $\|\nabla m_k\| \le \epsilon_g$, test for criticality.
5.1.2 Solve $\min\{m_k(x_k+s) : \|s\| \le \Delta_k\}$ for $s_k$ satisfying

        $m_k(x_k) - m_k(x_k+s_k) \ge \frac{\kappa_d}{2} \|\nabla m_k(x_k)\| \min\left\{ \frac{\|\nabla m_k(x_k)\|}{\kappa_H}, \Delta_k \right\}$,    (5.7)

      where $\kappa_H > 0$ satisfies $\infty > \kappa_H \ge \max_{x \in \mathcal{B}_k} \|\nabla^2 m_k(x)\|$.
      Set $x_+ = x_k + s_k$.
5.1.3 Evaluate $f(x_+)$ and $\rho_k = \frac{f(x_k)-f(x_+)}{m_k(x_k)-m_k(x_+)}$ and update the center:

        $x_{k+1} = \begin{cases} x_+ & \text{if } \rho_k \ge \eta_1 \\ x_+ & \text{if } \eta_1 > \rho_k > \eta_0 \text{ and } m_k \text{ f.l. on } \mathcal{B}_k \\ x_k & \text{else.} \end{cases}$

5.1.4 If $\rho_k < \eta_1$ and $m_k$ not f.l. on $\mathcal{B}_k$, improve the model by evaluating at a
      model-improving point. Hence or otherwise update the model to $m_{k+1}$.
5.1.5 Update the trust-region radius

        $\Delta_{k+1} = \begin{cases} \min\{\gamma_1 \Delta_k, \Delta_{\max}\} & \text{if } \rho_k \ge \eta_1 \\ \Delta_k & \text{if } \rho_k < \eta_1 \text{ and } m_k \text{ not f.l. on } \mathcal{B}_k \\ \gamma_0 \Delta_k & \text{if } \rho_k < \eta_1 \text{ and } m_k \text{ f.l. on } \mathcal{B}_k. \end{cases}$
```

Algorithm 5.1: Basic first-order derivative-free trust-region algorithm.

Before proceeding we note that for any fixed constant $\kappa_d \in (0,1)$ and twice-continuously differentiable model m_k, a step satisfying the sufficient decrease condition (5.7) can be efficiently found by Lemma 4.2 (see also [51]).

5.2 Minimum Norm Quadratic Interpolation Models

In this chapter we are interested in quadratic models of the form (5.3) with the parameters g_k and H_k such that m_k satisfies the interpolation conditions (5.4). To this end, we define

$$\begin{aligned} \mu(x) &= [1, \chi_1, \cdots, \chi_n], && (5.8)\\ \nu(x) &= \left[\frac{\chi_1^2}{2}, \cdots, \frac{\chi_n^2}{2}, \frac{\chi_1\chi_2}{\sqrt{2}}, \cdots, \frac{\chi_{n-1}\chi_n}{\sqrt{2}}\right], && (5.9)\end{aligned}$$

where χ_i denotes the ith component of the argument $x \in \mathbb{R}^n$. When taken together, $[\mu(x), \nu(x)]$ forms a basis for the linear space of quadratics in n variables, $\mathcal{Q}^n$. Thus any quadratic $m_k \in \mathcal{Q}^n$ can be written as

$$m_k(x - x_k) = \alpha^T \mu(x - x_k) + \beta^T \nu(x - x_k), \tag{5.10}$$

for coefficients $\alpha \in \mathbb{R}^{n+1}$ and $\beta \in \mathbb{R}^{n(n+1)/2}$. We note that any bijection of this basis would also yield a quadratic and so the form of the quadratic model in (5.10) may seem unusual at first glance. We propose to use this particular form of model because it lends itself well to our solution procedure.

Abusing notation, we let f denote the vector of function values so that (5.4) can be written as

$$\begin{bmatrix} M_{\mathcal{Y}} \\ N_{\mathcal{Y}} \end{bmatrix}^T \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = f, \tag{5.11}$$

where we define $M_{\mathcal{Y}} \in \mathbb{R}^{n+1 \times |\mathcal{Y}|}$ and $N_{\mathcal{Y}} \in \mathbb{R}^{n(n+1)/2 \times |\mathcal{Y}|}$, by $M_{i,j} = \mu_i(y_j)$ and $N_{i,j} = \nu_i(y_j)$, respectively. We explicitly note the dependence of these matrices on the interpolation set $\mathcal{Y}$.

The interpolation problem in (5.4) for multivariate quadratics is significantly more difficult than its univariate counterpart [73]. These points must satisfy additional geometric conditions that are summarized in the following Lemma, which follows immediately from the fact that $[\mu(x),\, \nu(x)]$ form a basis for $\mathcal{Q}^n$.

Lemma 5.1. *The following are equivalent:*

1. *For any $f \in \mathbb{R}^{|\mathcal{Y}|}$, there exists $m_k \in \mathcal{Q}^n$ satisfying (5.4).*

2. *$\{[\mu(y_j),\, \nu(y_j)]\}_{j=1}^{|\mathcal{Y}|}$ is linearly independent.*

3. *$\dim\{q \in \mathcal{Q}^n : q(x_k + y_i) = 0\, \forall y_j \in \mathcal{Y}\} = \frac{(n+1)(n+2)}{2} - |\mathcal{Y}|$.*

Proof. (1. $\Rightarrow$ 2.) Suppose that $f \neq 0$ gives a linear dependence relation on $\{[\mu(y_j),\, \nu(y_j)]\}_{j=1}^{|\mathcal{Y}|}$. Thus $\sum_{j=1}^{|\mathcal{Y}|}[\mu(y_j),\, \nu(y_j)]f_j = 0$, or equivalently $\begin{bmatrix} M_{\mathcal{Y}} \\ N_{\mathcal{Y}} \end{bmatrix} f = 0$. In order to obtain a contradiction, suppose there then exists a vector $g = (\alpha, \beta) \in \mathbb{R}^{\frac{(n+1)(n+2)}{2}}$ satisfying (5.11). However, then

$$0 = \left(f^T \begin{bmatrix} M_{\mathcal{Y}} \\ N_{\mathcal{Y}} \end{bmatrix}^T \right) g = f^T f = \|f\|^2,$$

which is a contradiction since $f \neq 0$.

(2. $\Rightarrow$ 1.) By linear independence, there are index sets $\mathcal{I}$ and $\mathcal{J}$ with $|\mathcal{I}|+|\mathcal{J}| = |\mathcal{Y}|$ such that

$$\left\{[\mu_i(y_1), \ldots, \mu_i(y_{|\mathcal{Y}|})] : i \in \mathcal{I}\right\} \cup \left\{[\nu_j(y_1), \ldots, \nu_j(y_{|\mathcal{Y}|})] : j \in \mathcal{J}\right\}$$

is linearly independent. Hence for any $f \in \mathbb{R}^{|\mathcal{Y}|}$ there is a unique $(\alpha_{\mathcal{I}}, \beta_{\mathcal{J}})$ such that

$$\sum_{i \in \mathcal{I}} \alpha_i [\mu_i(y_1), \ldots, \mu_i(y_{|\mathcal{Y}|})]^T + \sum_{j \in \mathcal{J}} \beta_j [\nu_j(y_1), \ldots, \nu_j(y_{|\mathcal{Y}|})] = f.$$

Letting $\alpha_i = 0$ for all $i \notin \mathcal{I}$ and $\beta_j = 0$ for all $j \notin \mathcal{J}$ we have a $(\alpha, \beta) \in \mathbb{R}^{\frac{(n+1)(n+2)}{2}}$ satisfying (5.11), and hence determining an interpolating quadratic through (5.10).

(2. $\iff$ 3.) Since $\begin{bmatrix} M_{\mathcal{Y}} \\ N_{\mathcal{Y}} \end{bmatrix} \in \mathbb{R}^{\frac{(n+1)(n+2)}{2} \times |\mathcal{Y}|}$, the Fundamental Theorem of Linear Algebra [36, pg 5] states that

$$\dim \left\{ \begin{bmatrix} M_{\mathcal{Y}} \\ N_{\mathcal{Y}} \end{bmatrix} u : u \in \mathbb{R}^{|\mathcal{Y}|} \right\}$$
$$+ \dim \left\{ (\alpha, \beta) \in \mathbb{R}^{\frac{(n+1)(n+2)}{2}} : \begin{bmatrix} M_{\mathcal{Y}} \\ N_{\mathcal{Y}} \end{bmatrix}^T \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = 0 \right\} = \frac{(n+1)(n+2)}{2}.$$

The result then follows by noting that 2. is equivalent to

$$\dim \left\{ \begin{bmatrix} M_{\mathcal{Y}} \\ N_{\mathcal{Y}} \end{bmatrix} u : u \in \mathbb{R}^{|\mathcal{Y}|} \right\} = |\mathcal{Y}|,$$

and that $[\mu(x), \nu(x)]$ form a basis for $\mathcal{Q}^n$. $\square$

The third condition in Lemma 5.1 reveals that these conditions are geometric, requiring that the subspace of quadratics disappearing at all of the data points be of sufficiently low dimension. For example, this prevents interpolation of arbitrary function values using 6 points lying on a circle in $\mathbb{R}^2$.

We illustrate this geometric condition in Figure 5.1 for the 2-dimensional case where the dimension of the space of quadratics is $\frac{(n+1)(n+2)}{2} = 6$. Shown is the taboo region corresponding to the level curve of a quadratic disappearing at all 5 of the interpolation points. Hence interpolation of arbitrary data values would not be possible if a sixth interpolation point lying on this taboo region were added.

Lemma 5.1 implies that quadratic interpolation is only feasible for arbitrary right hand side values if $[M_{\mathcal{Y}}^T, N_{\mathcal{Y}}^T]$ is full row rank. Further, this interpolation is only unique if $|\mathcal{Y}| = \frac{(n+1)(n+2)}{2}$ (the dimension of quadratics in $\mathbb{R}^n$) and $[M_{\mathcal{Y}}^T, N_{\mathcal{Y}}^T]$ is nonsingular.

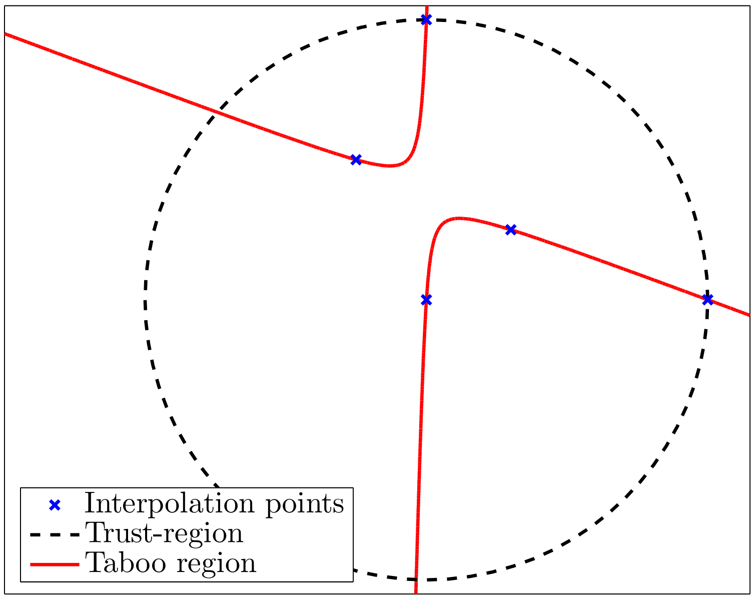


Figure 5.1: Level curve of a quadratic vanishing at 5 interpolation points in $\mathbb{R}^2$. A sixth interpolation point cannot lie on this taboo region in order to guarantee unique interpolation of arbitrary data values.

When $|\mathcal{Y}| < \frac{(n+1)(n+2)}{2}$, and $[M_{\mathcal{Y}}^T, N_{\mathcal{Y}}^T]$ is full rank, the interpolation problem (5.11) will have an infinite number of solutions. In this chapter we will focus on solutions to (5.11) that are of minimum norm with respect to the vector β. Hence we require the solution (α, β) of

$$\min \left\{ \frac{1}{2}\|\beta\|^2 : M_{\mathcal{Y}}^T \alpha + N_{\mathcal{Y}}^T \beta = f \right\}. \tag{5.12}$$

This minimum seminorm solution is of interest because it represents the quadratic whose Hessian matrix is of minimum Frobenius norm since $\|\beta\| = \|\nabla_{x,x}^2 m_k(x)\|_F$. While other "minimal norm" quadratics could be found, we are drawn to those with Hessians of minimal norm because the resulting solution procedure will have a natural tie-in to the fully linear models considered in Chapters 3 and 4.

The KKT conditions for (5.12) can be written as

$$\begin{bmatrix} N_{\mathcal{Y}}^T N_{\mathcal{Y}} & M_{\mathcal{Y}}^T \\ M_{\mathcal{Y}} & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \alpha \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}, \tag{5.13}$$

with $\beta = N_{\mathcal{Y}}\lambda$. This system closely resembles the system (3.13) for the radial basis function interpolation models of Chapters 3 and 4. As we did there, we solve this saddle point problem with a null space method by letting Z be an orthogonal basis for the null space $\mathcal{N}(M_{\mathcal{Y}})$ and $QR = M_{\mathcal{Y}}^T$ be a QR factorization. Since λ must belong to $\mathcal{N}(M_{\mathcal{Y}})$, we write $\lambda = Z\omega$ for $\omega \in \mathbb{R}^{|\mathcal{Y}|-n-1}$ so that (5.13) reduces to the $|\mathcal{Y}|$ equations:

$$Z^T N_{\mathcal{Y}}^T N_{\mathcal{Y}} Z\omega = Z^T f \tag{5.14}$$

$$R\alpha = Q^T(f - N_{\mathcal{Y}}^T N_{\mathcal{Y}} Z\omega), \tag{5.15}$$

with $\beta = N_{\mathcal{Y}} Z\omega$. We note the quivalent radial basis function expressions in (3.14) and (3.15).

The following Theorem establishes that the quadratic program (5.12) will yield a unique solution given geometric conditions on $\mathcal{Y}$.

Theorem 5.1. *For $n \geq 2$, if:*

(Y1) rank$(M_{\mathcal{Y}}) = n + 1$, and

(Y2) $Z^T N_{\mathcal{Y}}^T N_{\mathcal{Y}} Z$ is positive definite,

then, for any $f \in \mathbb{R}^{|\mathcal{Y}|}$, there exists a unique solution (α, β) to the quadratic program (5.12)*.*

Proof. $Z^T N_{\mathcal{Y}}^T N_{\mathcal{Y}} Z$ is positive definite if and only if $N_{\mathcal{Y}} Z$ is full rank. Since $n \geq 2$, $N_{\mathcal{Y}} Z$ is full rank if and only if $\mathcal{N}(N_{\mathcal{Y}} Z) = \{0\}$. Lastly, since Z is a basis for

$\mathcal{N}(M_{\mathcal{Y}})$, this is equivalent to $\mathcal{N}(N_{\mathcal{Y}}) \cap \mathcal{N}(M_{\mathcal{Y}}) = \{0\}$, which says that $[M_{\mathcal{Y}}^T \, N_{\mathcal{Y}}^T]$ is full rank. By Lemma 5.1, we then have that the feasible region of (5.12) is nonempty.

Since (5.12) is a convex (in β) quadratic program whose feasible region is nonempty, both β and the Lagrange multipliers λ associated with the constraints are unique [24]. Finally, we note that the coefficients α are then also uniquely determined from $M_{\mathcal{Y}}^T \alpha = f - N_{\mathcal{Y}}^T \beta$ since $M_{\mathcal{Y}}^T$ is full rank. □

In the proof of Theorem 5.1 we note that the system (5.13) being nonsingular is analogous to the constrained optimization optimality condition that the reduced Hessian of the objective function is positive definite.

If $Z^T N_{\mathcal{Y}}^T N_{\mathcal{Y}} Z$ is positive definite, it admits the Cholesky factorization

$$Z^T N_{\mathcal{Y}}^T N_{\mathcal{Y}} Z = LL^T, \tag{5.16}$$

for a nonsingular lower triangular L. Since Z is orthogonal we have the bound

$$\|\lambda\| = \|Z\omega\| \left\|ZL^{-T}L^{-1}Z^T f\right\| \leq \left\|L^{-1}\right\|^2 \|f\| = \frac{\|f\|}{\sigma_{\min}^2(L)}, \tag{5.17}$$

where $\sigma_{\min}(L)$ is the smallest singular value of L. This relationship will allow us to bound the coefficients $\beta = N_{\mathcal{Y}}\lambda$, and hence bound the Hessians of the model m_k.

Lemma 5.2. *If $\|y_i\|_\infty \leq \bar{\Delta}$ for all $y_i \in \mathcal{Y}$, then the model m_k obtained by solving* (5.12) *satisfies*

$$\left\|\nabla_{x,x}^2 m_k(x)\right\|_F \leq \frac{\|f\|}{\sigma_{\min}^2(L)} \sqrt{\frac{|\mathcal{Y}|}{2}} n^2 \bar{\Delta}^2. \tag{5.18}$$

Proof. Recalling that $\|\beta\| = \|\nabla^2_{x,x} m_k(x)\|_F$ and $\beta = N_{\mathcal{Y}}\lambda$, the bound in (5.17) yields:

$$\begin{aligned}
\|\nabla^2_{x,x} m_k(x)\|_F = \|N_{\mathcal{Y}}\lambda\| &\leq \|N_{\mathcal{Y}}\| \frac{\|f\|}{\sigma^2_{\min}(L)} \\
&\leq \sqrt{|\mathcal{Y}|}\|N_{\mathcal{Y}}\|_1 \\
&\leq \sqrt{|\mathcal{Y}|}\left(\frac{n}{2}\bar{\Delta}^2 + \frac{n(n-1)}{\sqrt{2}}\bar{\Delta}^2\right),
\end{aligned}$$

where the last inequality follows from the form of $\mu(y_i)$ in (5.9) and the fact that $\|y_i\|_\infty \leq \bar{\Delta}$. The result follows by slightly overestimating the final term. □

5.3 The MNH Algorithm

Theorem 5.1 offers a constructive way of obtaining an interpolation set $\mathcal{Y}$ that uniquely defines an underdetermined quadratic model whose Hessian is of minimum norm. We first collect $n+1$ affinely independent points and then add more points while keeping $\sigma_{\min}(L)$ bounded from zero.

We will always keep $y_1 = 0$ in the set $\mathcal{Y}$ to enforce interpolation at the current center. Thus we only need to find n linearly independent points $y_2, \ldots, y_{n+1}$. The resulting points will serve a secondary purpose of providing approximation guarantees for the model. This is formally stated in the following generalization of similar Taylor-like error bounds found in [16].

Theorem 5.2 (Chapter 4). *Suppose that f and m_k are continuously differentiable in $\mathcal{B} = \{x : \|x - x_k\| \leq \Delta\}$ and that ∇f and ∇m_k are Lipschitz continuous in $\mathcal{B}$ with Lipschitz constants γ_f and γ_m, respectively. Further suppose that m_k satisfies the interpolation conditions in* (5.4) *at a set of points $\mathcal{Y} = \{y_1 = 0, y_2, \ldots, y_{n+1}\} \subseteq \mathcal{B} - x_k$ such that $\left\|[y_2, \cdots, y_{n+1}]^{-1}\right\| \leq \frac{\Lambda_Y}{\Delta}$. Then for any $x \in \mathcal{B}$:*

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

2. $\|\nabla m_k(x) - \nabla f(x)\| \leq \frac{5}{2}\sqrt{n}\Lambda_Y\,(\gamma_f + \gamma_m)\,\Delta$.

Proved in Chapter 4 and [76], Theorem 5.2 says that if a model with a Lipschitz continuous gradient interpolates a function on a sufficiently affinely independent set of nearby points, there exist constants $\kappa_f, \kappa_g > 0$ independent of Δ such that conditions (5.5) and (5.6) are satisfied. In our case, the model m_k will be twice continuously differentiable and hence the following Lemma yields a Lipschitz constant.

Lemma 5.3. *For the model m_k defined in* (5.10)*, $\nabla m_k(x)$ is $\|\beta\|$-Lipschitz continuous on $\mathbb{R}^n$.*

Proof. Since m_k is a quadratic, $\nabla m_k(x) - \nabla m_k(y) = \nabla^2 m_k(x)(x-y)$ for all $x, y \in \mathbb{R}^n$. Recalling that $\|\nabla^2 m(x)\|_F = \|\beta\|$ we have

$$\|\nabla m_k(x) - \nabla m_k(y)\| \leq \|\nabla^2 m_k(x)\|\|x-y\| \leq \|\nabla^2 m_k(x)\|_F\|x-y\| = \|\beta\|\|x-y\|,$$

establishing the result. □

5.3.1 Finding Affinely Independent Points

We now show that we can obtain n points such that $\|\left[y_2, \cdots, y_{n+1}\right]^{-1}\|$ is bounded by a quantity of the form $\frac{\Lambda_Y}{\Delta}$ as required in Theorem 5.2. We ensure this by working with a QR factorization of the normalized points $Y = \left[\frac{y_2}{\Delta}, \cdots, \frac{y_{n+1}}{\Delta}\right]$. If we require that these points satisfy $\left\|\frac{y_j}{\Delta}\right\| \leq 1$, and that the resulting pivots satisfy $|R_{j,j}| \geq \theta_1 > 0$, then it is straightforward to show that $\|Y^{-1}\| \leq \Lambda_Y$ for a constant Λ_Y depending only on n and θ_1 (see Lemma 4.1 in Chapter 4 and [76]).

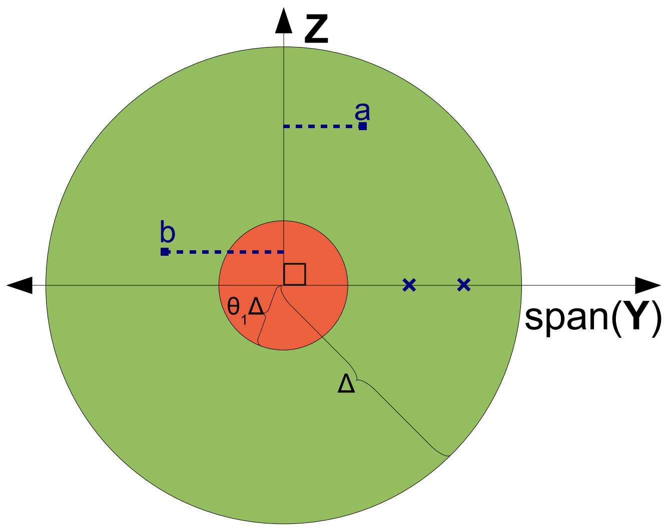


Figure 5.2: Finding sufficiently affinely independent points: a is acceptable, b is not (see also Figure 4.1).

Figure 5.2 illustrates our procedure graphically. From our bank of points at which the function has been evaluated, we examine all those within Δ of the current center. These points are iteratively added to $\mathcal{Y}$ provided that their projection onto the current null space $Z = \mathcal{N}([y_2, \cdots y_{|\mathcal{Y}|}])$ is at least of magnitude $\theta_1 \Delta$. In Figure 5.2 the **x**'s denote the current points in $\mathcal{Y}$, while the projections of two available candidate points, **a** and **b**, show that only **a** would be added to $\mathcal{Y}$.

In practice, we work with an enlarged region with radius $\Delta = \theta_0 \Delta_k$ (for a parameter $\theta_0 \geq 1$), to ensure the availability of some previously evaluated points. Our procedure is detailed formally in Algorithm 5.2 and also used by ORBIT in Chapters 3 and 4.

This procedure also guarantees that such an interpolation set can be constructed for any value of the constant $\theta_1 \leq 1$. In particular, if Z is an orthogonal basis for $\mathcal{N}([y_2, \cdots y_{|\mathcal{Y}|}])$, its columns are directions that result in unit pivots, $|R_{j,j}| = 1$. We call $\pm\Delta z_j$ *model-improving points* because they can be included in

```
5.2.0. Input $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\} \subset \mathbb{R}^n$, constants $\theta_0 \geq 1$, $\theta_1 \in (0, \theta_0^{-1}]$, $\Delta \in (0, \Delta_{\max}]$.
5.2.1. Initialize $\mathcal{Y} = \{y_1 = 0\}$, $Z = I_n$.
5.2.2. 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 $\mathcal{N}\left([y_1 \cdots y_{|\mathcal{Y}|}]\right)$.
```

Algorithm 5.2: AffPoints($\mathcal{D}, \theta_0, \theta_1, \Delta$): Algorithm for obtaining a fully linear model (See also Algorithm 3.3).

$\mathcal{Y}$ to make m_k fully linear on $\mathcal{B}$.

Such an approach was also taken in Chapter 4 and [76] but here we note that in both the MNH and ORBIT algorithms we can also choose an orientation on the direction based on the model values: $m_k(\Delta z_j) = \min\{m_k(\pm \Delta z_j)\}$. Intuitively, this means that our algorithm selects the direction that the model predicts the function is smaller. As a result, either the algorithm sees the decrease in f as predicted, or it obtains additional information that the function is not as fruitful in the direction previously predicted to be good.

Upon termination of Algorithm 5.2, the set $\mathcal{Y}$ either contains $n+1$ points (including the initial point 0) which certifies that the model is fully linear on a ball of radius $\theta_0 \Delta_k$, or there will be nontrivial model-improving directions in Z which can be evaluated to obtain such a model.

While the trust-region framework in Algorithm 5.1 does not prescribe a fully linear model at each iteration, Theorem 5.1 requires that $\mathcal{Y}$ include $n+1$ affinely independent points. Hence, if a model is not fully linear, we will rerun Algorithm 5.2 with a larger θ_0. This has the effect of searching for points in the bank within a larger region. If still an insufficient number of points are available, the directions in the resulting Z must be evaluated to ensure that the interpolation set allows for a unique model.

5.3.2 Adding More Points

After running Algorithm 5.2, and possibly evaluating f at additional points, the interpolation set $\mathcal{Y}$ consists of $n+1$ sufficiently affinely independent points. If no other points are added to $\mathcal{Y}$, we will have $\beta = 0$ and hence m_k would be a linear model. Adding additional points to $\mathcal{Y}$ will not affect the first condition (Y1) of Theorem 5.1, thus our goal is to add more points from the bank to $\mathcal{Y}$ while ensuring the second condition (Y2) is satisfied and (5.15) remains well-conditioned.

We now consider what happens when $d \in \mathbb{R}^n$ is added to the interpolation set $\mathcal{Y}$ and denote the resulting basis matrices by $\tilde{M_{\mathcal{Y}}}$ and $\tilde{N_{\mathcal{Y}}}$:

$$\tilde{M_{\mathcal{Y}}} = \left[\begin{array}{cc} M_{\mathcal{Y}} & \mu(d) \end{array}\right], \qquad \tilde{N_{\mathcal{Y}}} = \left[\begin{array}{cc} N_{\mathcal{Y}} & \nu(d) \end{array}\right].$$

By applying $n+1$ Givens rotations to the full QR factorization of $M_{\mathcal{Y}}^T$, we obtain an orthogonal basis for $\mathcal{N}(M_{\mathcal{Y}})$ of the form:

$$\tilde{Z} = \left[\begin{array}{cc} Z & Q\tilde{g} \\ 0 & \hat{g} \end{array}\right],$$

where Z is any orthogonal basis for $\mathcal{N}(M_{\mathcal{Y}})$. Hence, $\tilde{N_{\mathcal{Y}}}\tilde{Z}$ consists of the previous factors $N_{\mathcal{Y}}Z$ and one additional column:

$$\tilde{N_{\mathcal{Y}}}\tilde{Z} = \left[\begin{array}{cc} N_{\mathcal{Y}}Z & N_{\mathcal{Y}}Q\tilde{g} + \hat{g}\nu(d) \end{array}\right]. \tag{5.19}$$

While beyond the scope of this chapter, we note that (5.19) suggests that the resulting Cholesky factorization $\tilde{L}\tilde{L}^T = (\tilde{N_{\mathcal{Y}}}\tilde{Z})^T\tilde{N_{\mathcal{Y}}}\tilde{Z}$ could be updated using the

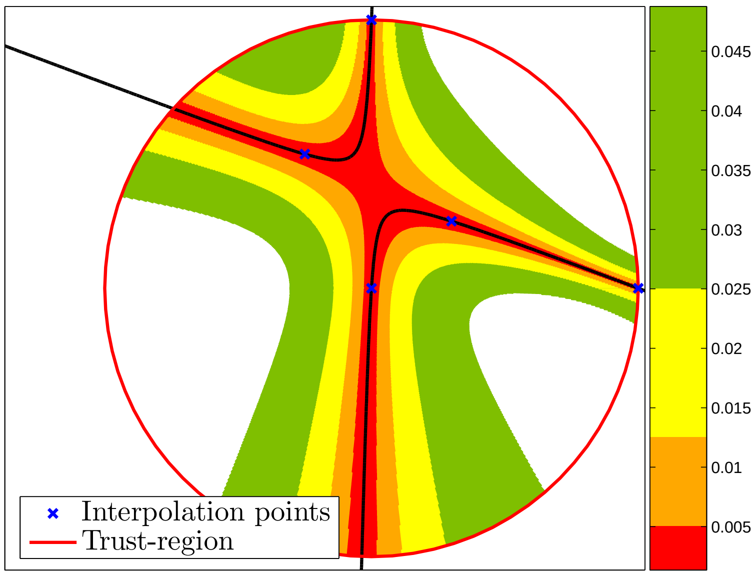


Figure 5.3: Illustrating the value of $\sigma_{\min}(N_{\mathcal{Y}}Z)$ for points inside a trust-region when $\mathcal{Y}$ consists of 5 points in $\mathbb{R}^2$. Small values $\sigma_{\min}(N_{\mathcal{Y}}Z)$ occur near the taboo region (shown in Figure 5.1) and should be avoided to prevent ill-conditioning.

previous factorization, $LL^T = Z^T N_{\mathcal{Y}}^T N_{\mathcal{Y}} Z$. In the present work, we require only a mechanism for bounding $\sigma_{\min}(L)$ for use in the bound (5.17). Since $\sigma_{\min}(N_{\mathcal{Y}}Z) = \sigma_{\min}(L)$, it will suffice to enforce $\sigma_{\min}(N_{\mathcal{Y}}Z) \geq \theta_2$ for a constant $\theta_2 > 0$.

In Figure 5.3 we illustrate the level sets of $\sigma_{\min}(N_{\mathcal{Y}}Z)$ for points inside the trust-region when the interpolation set $\mathcal{Y}$ consists of 5 points. Recall from Figure 5.1 that the taboo region corresponds to a quadratic curve disappearing at all 5 points and must be avoided to allow for unique interpolation of arbitrary function values. We note that the level sets shown in Figure 5.3 appear as relaxations of this taboo region for the specific basis that we employ. An additional point would only be added to $\mathcal{Y}$ if it were outside of the level set $\sigma_{\min}(N_{\mathcal{Y}}Z) < \theta_2$.

The bound on λ in (5.17) will be used to bound $\|\beta\| = \|N_{\mathcal{Y}}\lambda\|$, which from Lemma 5.3, serves as a Lipschitz constant for m_k, thus justifying our use of fully

```
5.3.0. Input $\mathcal{Y}$, $\mathcal{D} = \{d_1, \dots, d_{|\mathcal{D}|}\} \subset \mathbb{R}^n$, constants $\theta_0 \geq 1$, $\theta_2 > 0$, $\Delta_k \in (0, \Delta_{\max}]$.
5.3.1. Initialize $QR = M_{\mathcal{Y}}^T$, $Z = \emptyset$.
5.3.2. For all $d_j \in \mathcal{D} \backslash \mathcal{Y}$ such that $\|d_j\| \leq \theta_0 \Delta_k$:
    Compute $\tilde{N_{\mathcal{Y}}}\tilde{Z}$ as in (5.19).
    If $\sigma_{\min}\left(\tilde{N_{\mathcal{Y}}}\tilde{Z}\right) \geq \theta_2$:
        $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{d_j\}$,
        Update $Z = \tilde{Z}$ and $N_{\mathcal{Y}} = \tilde{N_{\mathcal{Y}}}$.
```

Algorithm 5.3: MorePoints($\mathcal{D}, \theta_0, \theta_2, \Delta_k$): Algorithm for using additional points in $\mathcal{Y}$.

linear models. By the discussion in Section 5.2, the interpolation set must always obey the bound $|\mathcal{Y}| = \frac{(n+1)(n+2)}{2}$ since otherwise $N_{\mathcal{Y}}Z$ would be rank-deficient. Hence, in order to bound $\|N_{\mathcal{Y}}\|$, it suffices to keep the points in $\mathcal{Y}$ within a bounded region. We will again assume that this region is contained in a ball of radius $\theta_0\Delta_k$ for some $\theta_0 \geq 1$. Algorithm 5.3 then specifies the resulting subroutine, analogous to Algorithm 4.4 used for radial basis function models.

By Theorem 5.1, once we have the interpolation set resulting from Algorithms 5.2 and 5.3, we can uniquely obtain a quadratic model whose Hessian is of minimal Frobenius norm. Furthermore, by construction, we can obtain the model parameters α and β in a computationally stable way by solving the system in (5.14) and (5.15). We summarize the theoretical implications in the following Theorem, which is similar to Theorem 4.2 for the radial basis function models used by ORBIT.

Theorem 5.3. *Suppose that:*

(AF) *$f \in C^1[\Omega]$ for some open $\Omega \supset \mathcal{L}(x_0)$, ∇f is Lipschitz continuous on $\mathcal{L}(x_0)$, and f is bounded on $\mathcal{L}(x_0)$ such that* $\max\{|f(x)| : x \in \mathcal{L}(x_0)\} = f_{\max} < \infty$, *and that every interpolation set $\mathcal{Y}$ obeys:*

(a) *$n+1$ points in $\mathcal{Y}$ are affinely independent,*

(b) *$|\mathcal{Y}| \leq p_{\max}$,*

(c) *$\|y_i\|_\infty \leq \bar{\Delta}$ for all $y_i \in \mathcal{Y}$, and*

(d) *$\sigma_{\min}(N_{\mathcal{Y}} Z) \geq \theta_2 > 0$ for the L defined in (5.16).*

Then we have that every quadratic model (5.10) formed satisfies:

$$\left\|\nabla^2_{x,x} m_k(x)\right\|_F \leq \frac{p_{\max} f_{\max}}{\theta_2^2} \frac{1}{\sqrt{2}} n^2 \bar{\Delta}^2. \tag{5.20}$$

Furthermore, given an appropriate criticality test (such as Algorithm 4.3 in Chapter 4 or the test in [15]), the sequence of iterates generated by Algorithm 5.1 converge to a first-order critical point:

$$\lim_{k\to\infty} \nabla f(x_k) = 0 \tag{5.21}$$

Proof. Using assumptions (a) and (d), Theorem 5.1 guarantees existence and uniqueness of the model m_k. The first half of the theorem then follows directly from assumptions (b), (c), and (d), and Lemma 5.2. Thus we have that the model m_k has a bounded Hessian and hence $\nabla_x m_k(x)$ is Lipschitz continuous on $\mathbb{R}^n$. Theorem 5.2 thus applies and the model m_k can be made fully linear by the finite procedure described in Algorithm 5.2. The result then follows immediately from Theorem 4.2 in Chapter 4 (see also [15]). □

5.4 Preliminary Numerical Experiments

We have recently completed an initial implementation of the MNH algorithm. In this section we present the results of preliminary numerical tests to illustrate the strengths and weaknesses of our current implementation and outline directions of future work.

We are particularly interested in how MNH performs compared to the NEWUOA [61] and UOBYQA [58] codes of Powell. NEWUOA was shown to have the best short-term performance on both smooth and mildly noisy functions in a test of three frequently-used derivative-free optimization algorithms [52] (see also Chapter 2). UOBYQA requires more initial function evaluations but forms more accurate models in the long term.

Both are trust-region methods that use quadratic interpolation models. NEWUOA works with updates of the Hessian which are of minimal norm and a fixed number of interpolation points $p \in \{n+2, \ldots, \frac{(n+1)(n+2)}{2}\}$, the value $p = 2n+1$ being recommended by Powell. Hence each time a newly evaluated point is added to the interpolation set, another point must be removed and will never return to the interpolation set. Including these points whenever possible, as is done by MNH, may provide the model with additional useful information. UOBYQA uses full quadratic models and thus always interpolates at $\frac{(n+1)(n+2)}{2}$ points.

We considered two smooth test functions from the set detailed in Chapter 2 and [52]. For each, we generated 30 random starting points within the unit hypercube and gave all codes the same starting point and trust-region radius. In Figure 5.4 we show the mean trajectory of the best function value obtained as a function of the number of evaluations of f. The interpretation here is that each solver would

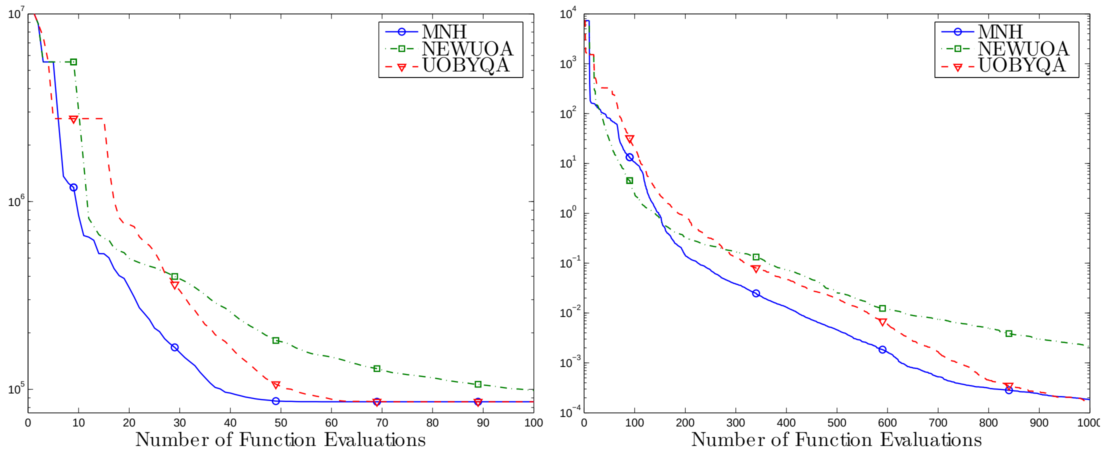


Figure 5.4: Mean of the best function value in 30 trials ($\log_{10}$-scale, lowest is best): (a) Brown and Dennis function ($n = 4$); (b) Watson function ($n = 9$).

output the value shown as its approximate solution given this number of function evaluations.

In Figure 5.4 (a) we show the results for the ($n = 4$)-dimensional Brown and Dennis function. Note that MNH, NEWUOA, and UOBYQA require initializations of $n+1 = 5$, $2n+1 = 9$, and $\frac{(n+1)(n+2)}{2} = 15$ function values, respectively. We see that MNH obtains an initial lead because of its shorter initialization and then continues to make marked progress, yielding the best approximate solution for virtually all numbers of evaluations.

In Figure 5.4 (b) we show the results for the ($n = 9$)-dimensional Watson function. We see that MNH again has a slight initial advantage over NEWUOA and UOBYQA because it begins solving trust-region subproblems after $n+1$ evaluations. Further, given between 155 and 1000 evaluations, MNH obtains the best solution on average. For these numbers of function evaluations MNH often has the ability to use a full quadratic number $\left(\frac{(n+1)(n+2)}{2} = 55\right)$ of points from the bank while NEWUOA is always using only $2n + 1 = 19$ points. This allows MNH to form models based on more information. That NEWUOA outperforms MNH between 30 and 155 evaluations is interesting, and we hope that as our implementation matures we may better understand this difference.

For one run on the Watson problem, Figure 5.5 shows the number of points at which the MNH model interpolates the function and the inverse of the trust-region radius Δ_k, scaled for visibility. We note that MNH is able to make efficient use of the bank of points, $|\mathcal{Y}|$ growing from $n + 1 = 10$ to the upper bound of 55, using a full quadratic model for the majority of the iterations. The iterations when this upper bound is not achieved usually correspond to those where the trust-region radius Δ_k has experienced considerable decrease.

For the same run, Figure 5.6 shows the distribution of the distances from the interpolation points to the current iterate x_k. Here we see that the interpolation set consists of points which are close to x_k. As expected, the distribution tends toward larger distances after periods of larger trust-regions and the models are constructed in smaller neighborhoods as the algorithm progresses.

5.5 Conclusions and Future Work

In this chapter we have outlined a new algorithm for derivative-free optimization. The quadratic models employed resemble those used by Powell in [60] but our method of constructing the interpolation set allows for a convergence result (Theorem 5.3) that is not currently established for NEWUOA. Our method is also able to take advantage of more data in the bank of previously evaluated points, often employing a full quadratic number of them as interpolation points in our tests.

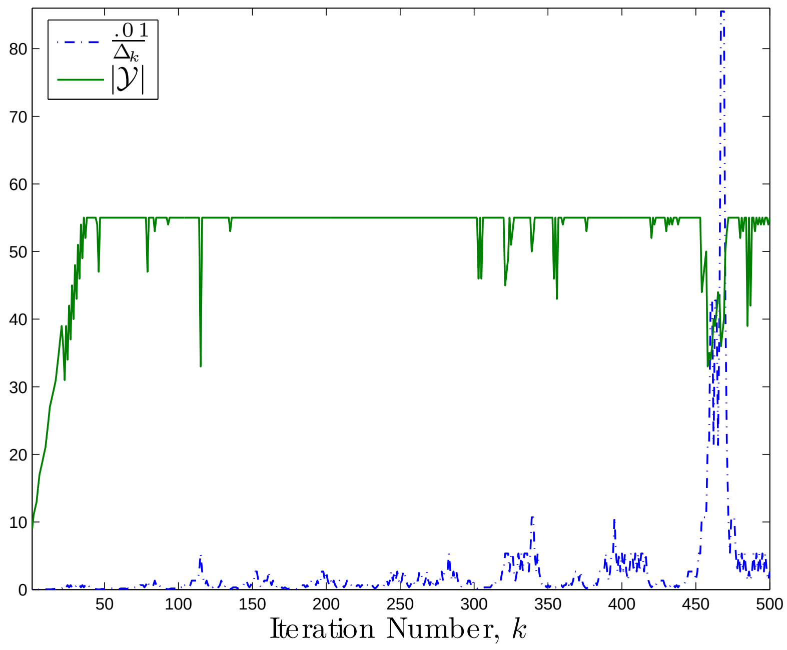


Figure 5.5: One run on the Watson function: Inverse of the trust-region radius and number of interpolation points.

Our preliminary results are encouraging and we expect these to improve as our code matures.

The approach outlined can also be extended to other types of interpolation models, from higher order polynomials to different forms of underdetermined quadratics. Regarding the latter we note that it may be advantageous to obtain a better estimate of the gradient than via the system in (5.15). For example, one could obtain the coefficients α using only $n+1$ nearby points and then form the minimal norm Hessian given this fixed α. This is just one of many areas of future work inspired by the approach introduced here.

In the future, we hope to do comprehensive numerical comparisons between MNH and ORBIT to determine if there are classes of problems where one significantly outperforms the other. We also intend to extend the framework introduced here to algorithms using minimum change models similar to those employed by

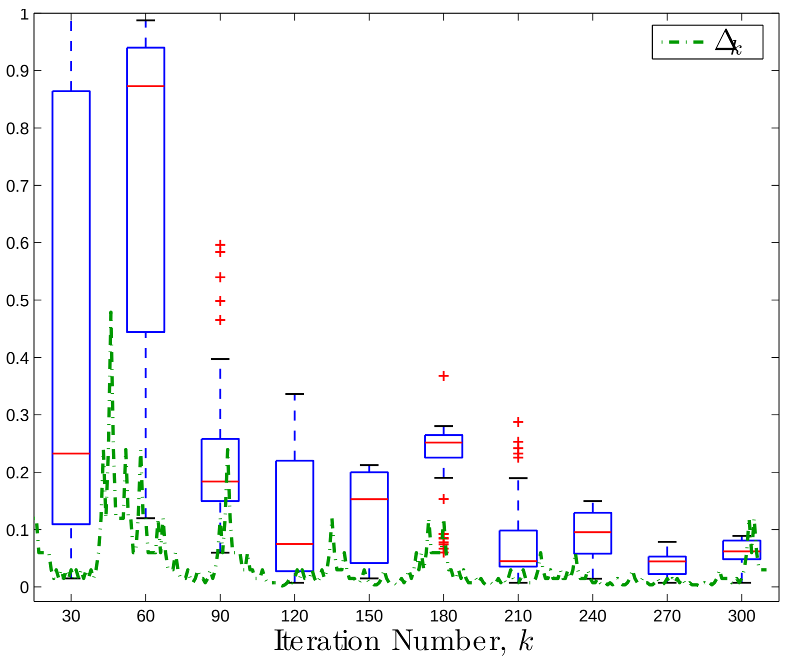


Figure 5.6: One run on the Watson function: The box and whisker plots show the distribution of the distances to the interpolation points and the dashed lines represent the size of the trust-region radius.

NEWUOA to see if this strategy can further benefit the algorithm.

Acknowledgements

The author is grateful to Jorge Moré and Christine Shoemaker for helpful comments. This work was supported by a Department of Energy Computational Science Graduate Fellowship, grant number DE-FG02-97ER25308.

CHAPTER 6

GLOBAL OPTIMIZATION BY GORBIT

In this chapter we address ways in which a global optimization algorithm can best profit from its own history of evaluations and propose a new algorithm, GORBIT. First president of the United States George Washington once declared:

We ought not to look back unless it is to derive useful lessons from past errors, and for the purpose of profiting by dear-brought experience.

We believe this adage holds true in computational science as much as it does in life. The saying is particularly appropriate when working with computationally expensive functions, where each value of the objective function obtained is especially "dear-brought." The algorithm we propose will in fact often rely just as much on previous poor function values as it will on favorable ones. As Washington suggests, we will also examine what is gained from this history and find that the performance of the resulting algorithm depends on how this history is used.

We propose a new algorithm, GORBIT, which uses a modified version of the local optimization solver ORBIT [75] in a multistart procedure. We aim to solve the global minimization problem,

$$f_* = \min \left\{ f(x) : x \in \mathcal{D} \right\}, \tag{6.1}$$

of a real-valued computationally expensive function f over a compact domain $\mathcal{D} \subset \mathbb{R}^n$. Our initial focus will be on bound-constrained domains $\mathcal{D}$. The function f is assumed to be deterministic and continuous on $\mathcal{D}$. By Weierstrass' Theorem, continuity and compactness guarantee that the minimum value f_* is attained by some $x_* \in \mathcal{D}$. While additional smoothness properties of f are expected to be more favorable for the local algorithm we will employ, we assume that all derivatives of f are either unavailable or intractable to compute or approximate directly.

We are motivated by application problems depending on a so-called *blackbox simulator*, where the simulator usually only returns a single output $S(\hat{x})$ at a specified point $\hat{x}$. We are particularly interested in simulators that are computationally expensive to evaluate, requiring anywhere from several seconds to many weeks of CPU time to evaluate at a single $\hat{x}$. Such simulators increasingly benefit from the advancement of parallel computing. However, function evaluation remains the dominant expense in many optimization problems since the savings in wall clock are often offset by increased expectations for accuracy of the simulation (for example using finer spatial meshes or smaller time steps when simulating a physical process).

As a result of the computational expense of the function, in practice a global optimization algorithm has a computational budget that restricts the number of function evaluations that can be performed. We target the user seeking the greatest reduction of function value within this budget of evaluations. Encouraged by the results when using approximation models in derivative-free local optimization [52, 75], we take a similar approach here, using a computationally-attractive approximating model to generate new points for evaluation of the expensive function.

Our approach to global optimization is to use a local approximation coupled with a multistart method. When only function values are available, many algorithms build global models of the function f. In particular we mention [8, 33, 38, 39, 63, 64, 65]. We are driven by the results in [65], where an algorithm using a more localized model outperformed the analogous algorithm with a model using global knowledge of the function. A third approach is taken in [1], where a substantially different modeling approach is employed.

The algorithm we propose here will be in the class of *two-phase methods*, which Schoen categorizes as having exploration and refinement phases [67]. In the exploration phase, sampling is done throughout the domain. Then a global decision is made as to which local minima estimates should be refined. In our case, this refinement takes the form of running a local model-based optimization algorithm.

We will focus on using the local optimization algorithm ORBIT, which shows promise for unconstrained local optimization when f is computationally expensive [75] (and Chapter 3). ORBIT is a trust-region algorithm using radial basis function interpolation models. While there are several ways in which a local solver can be used as part of a global algorithm, we pursue two primary goals:

1. Rapid function value reduction given a limited number of function evaluations, and
2. Convergence to a global minimum.

We emphasize these goals because they may be viewed as competing, especially in light of the following theorem.

Theorem 6.1 (Törn and Žilinskas [70])**.** *An algorithm converges to the global minimum f_* of* (6.1) *for any continuous f if and only if the sequence of points visited by the algorithm is dense in $\mathcal{D}$.*

Theorem 6.1 states that in order to achieve our second goal, the algorithm must be able to sample everywhere in $\mathcal{D}$ if given enough evaluations. The basic idea of Theorem 6.1 is that if iterates of an algorithm are not dense when run on a function f, and hence no points within a distance $\delta > 0$ of some point

$\hat{x} \in \mathcal{D}$ are evaluated, the function f can be continuously augmented on the set $\{x : \|x - \hat{x}\| \leq \delta\}$ so that the global minimum occurs at $\hat{x}$ and the new function has $\hat{f}(\hat{x}) = \min\{f(x) : x \in \mathcal{D}\} - \epsilon$ for some $\epsilon > 0$. For this augmented function $\hat{f}$ the sequence of iterates would be the same but would have no chance of obtaining function values within ϵ of $\hat{f}(\hat{x})$. While our focus is on computationally expensive functions, we acknowledge that an algorithm should be able to use additional function evaluations if the computational budget suddenly grows.

Our Global ORBIT algorithm, GORBIT, will rely on running ORBIT from different starting points in order to ensure convergence to a global minimum. Given this multistart framework, our goal is get the best possible decrease in function value in the fewest possible expensive function evaluations. Unlike previous approaches, our strategy will be to make use of the function values obtained in every stage of the algorithm. Each of these function values were costly to obtain and, in our view, are often underutilized relative to their expense.

Our strategy for using the points evaluated in the course of running GORBIT has two major ingredients:

1. Give the multistart procedure knowledge of the points and function values obtained in the local optimization runs, and
2. Use a local solver which is efficient at getting rapid reduction in function value and can use the points and function values from the multistart sampling procedure and previous runs of the local solver.

Before we begin, we collect notation that will be used throughout the chapter. For iteration $k \geq 1$ of a global algorithm we define the following sets consisting of points from $\mathcal{D}$:

S_k: the set of sample points accumulated in the exploration phases up to the kth iteration,

C_k: the set of candidate points during the kth iteration,

$M_{i,k}$: the set of points evaluated by the ith local minimization run in the kth iteration, and

A_k: the set of points evaluated by all local minimizations up to the kth iteration, $A_k = \cup_{j \leq k} \cup_i M_{i,j}$.

In particular we note that after the kth iteration, the global algorithm will have evaluated the function f at the set of points $S_k \cup A_k$.

This chapter is organized as follows. In Section 6.1 we begin with a review of the Multi Level Single Linkage (MLSL) multistart framework, which we use to start a local solver from different starting points. In Section 6.2 we detail our new Maximum Information from Previous Evaluations (MIPE) procedure for using the function values obtained in the course of running the local solver. In Section 6.3 we review the local ORBIT algorithm and the modifications needed to address simple bound-constrained domains and the use of points obtained outside of the local ORBIT. We conclude in Section 6.4 with preliminary numerical results, which suggest default parameter values for GORBIT. We also illustrate the use of GORBIT on the problem of finding large growth factors when solving linear systems using Gaussian elimination.

6.1 Multistart Using Multi-Level Single Linkage

Any local optimization algorithm $\mathcal{A}$ can be extended to a global optimization algorithm by a *multistart* procedure, running the algorithm $\mathcal{A}$ from multiple starting points. In fact, if these starting points are generated in a random manner such that the probability of selecting any point in $\mathcal{D}$ is strictly positive, the resulting global algorithm will converge to f_* by Theorem 6.1. This simple multistart algorithm will not be particularly efficient since it does not use the function values at the sample points to guide the refinement phase.

GORBIT is designed to use one of the following multistart methods: a) *Multi Level Single Linkage (MLSL)* [40] and b) MIPE, described in the next section. MLSL is a stochastic algorithm, obtaining starting points for a local algorithm $\mathcal{A}$ by sampling points at random within $\mathcal{D}$. However, it also attempts to make use of the function values obtained as part of this sampling and the subsequent local minimizations to better delineate regions of attraction within $\mathcal{D}$. The interested reader is directed to [47] in which some of the original assumptions of MLSL are relaxed.

A general iteration of the basic MLSL algorithm is given in Algorithm 6.1. In addition to a local optimization algorithm $\mathcal{A}$ and a procedure for generating a sample of size N, it requires a *critical distance* r_k and a thresholding parameter γ. Both of these parameters are aimed at reducing the number of local minimization runs started, in an effort to avoid finding the same local minimizer repeatedly.

The parameter $\gamma \in (0, 1]$ controls the number of sample points that are candidates for starting a new local minimization run. At each iteration only the sample points whose function values lie in the lower $\gamma-$quantile are labeled as candidate

6.1.1. Evaluate f at N new sample points $x_{(k-1)N+1}, \ldots, x_{kN}$, which are included in the sample set S_k

6.1.2. Order all the kN sample points so that $f(x_1) \leq \cdots \leq f(x_{kN})$

6.1.3. Define the reduced sample set of candidate points $C_k = \{x_1, \ldots, x_{\lceil \gamma kN \rceil}\}$

6.1.4. Start $\mathcal{A}$ at each $x_i \in C_k$ for which both:

1. no local procedure has previously been started, and
2. no other point $x_j \in C_k$ with $f(x_j) < f(x_i)$ is within a distance r_k

6.1.5. Update the critical distance r_k (eg.- using (6.2))

Algorithm 6.1: Iteration k of a basic MLSL algorithm (given inputs N, γ, a critical distance r_k, a sampling procedure, and a local algorithm $\mathcal{A}$).

points and put in C_k. These candidate points are then further reduced through a clustering procedure that relies on the following definition.

Definition 6.1. *An ordered set of points $\{x_i : i \in \mathcal{I}_D\}$ for an index set $\mathcal{I}_D = i_1, i_2, \ldots, i_{|\mathcal{I}_D|}$, is called an $\underline{r_k\text{-descent path}}$ if for all $i_j \in \mathcal{I}_D$ $(j \neq 1)$ we have that $f(x_{i_j}) \leq f(x_{i_{j-1}})$ and $\|x_{i_j} - x_{i_{j-1}}\| \leq r_k$.*

The general idea is to use these descent paths to delineate the basins of attraction of a local descent algorithm. The local algorithm $\mathcal{A}$ will not be started from any candidate point lying on the same descent path as another candidate point with a lower function value (as determined in Step 6.1.4.2 of Algorithm 6.1). Initially $r_k > 0$ is large enough to usually limit starting local minimization runs from only those sample points whose function value are best. We note that as r_k tends to zero, two points x_i and x_j will only be neighbors on an r_k-descent path if the function f is monotone on the line segment $x_i + t(x_j - x_i)$, $t \in [0, 1]$.

The properties of a particular MLSL implementation depend on how the decreasing sequence of $\{r_k\}_{k \geq 1}$ is chosen. A special case of the critical distance measure used in [40] at iteration k is

$$r_k = \frac{1}{\sqrt{\pi}} \sqrt[n]{\Gamma\left(1+\frac{n}{2}\right) \operatorname{vol}(\mathcal{D}) \frac{5\log(kN)}{kN}}, \tag{6.2}$$

where $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ is the gamma function and $\operatorname{vol}(\mathcal{D})$ denotes the volume of $\mathcal{D}$. The following theorem gives the key property that drew us toward MLSL and this critical distance r_k.

Theorem 6.2 (Rinnooy Kan and Timmer [40]). *If the critical distance r_k is chosen by* (6.2) *then, even if sampling continues forever, the total number of local searches started by MLSL is finite with probability 1.*

We expect that each local minimization run will require (many) more than N evaluations of the function, and hence we would like to limit the number of unnecessary runs. Theorem 6.2 says that choosing a good critical distance measure ensures that (with probability 1) we do not do an infinite number of these local minimizations.

6.1.1 Illustration of Basic MLSL

We illustrate the basic version of MLSL in two dimensions on the scaled and tilted two-dimensional Branin function over the unit cube $\mathcal{D} = [0,1]^2$. The contours over this cube are shown in Figure 6.1 and we note that there are three local minima, the leftmost minima being the global minimum.

We generate $N = 50$ sample points, S_1, at which we evaluate f and then select the best $100\gamma = 50\%$ of these as the first set of candidates, $C_1 \subset S_1$, for starting a local minimization run. Following (6.2), the initial critical distance is $r_1 = \sqrt{\frac{\log(50)}{10\pi}} \approx .353$, and hence we cycle through the $\gamma N = 25$ candidate points

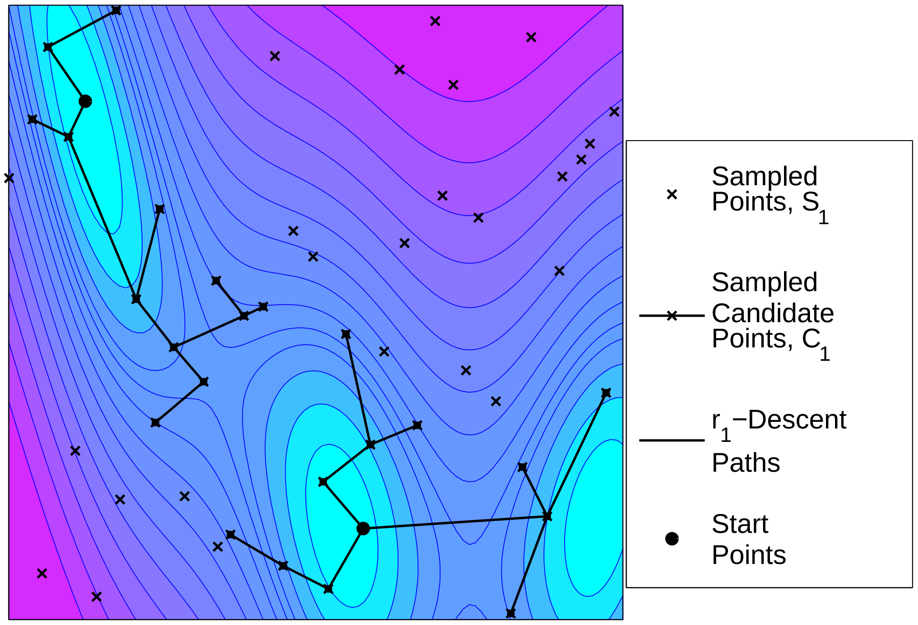


Figure 6.1: Initial MLSL iteration on the scaled and tilted Branin function ($r_1 \approx .353$).

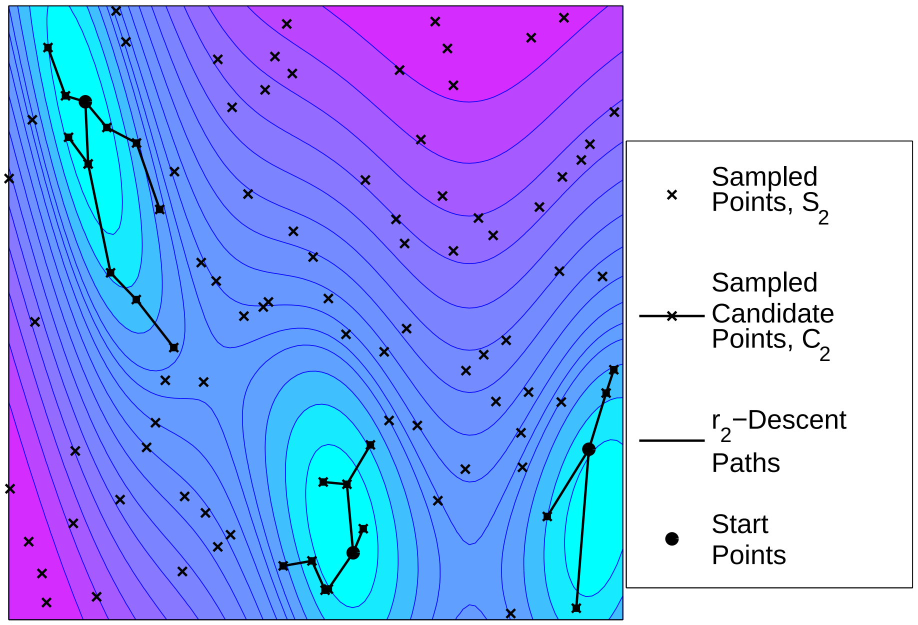


Figure 6.2: Second MLSL iteration on the scaled and tilted Branin function ($r_2 \approx .271$).

to determine whether there is another candidate within a distance r_1 with a lower function value. In the MLSL algorithm we do not need to compute the r_1-descent paths. The algorithm only requires the set of points in C_1 that are not within a distance r_1 of a lower-valued point in C_1. However, we note that this procedure ends up clustering the points based on r_1-descent paths. One possible set of r_1-descent paths is shown in Figure 6.1. Here we see that two clusters emerge, and hence there are two acceptable points in C_1 from which a local algorithm $\mathcal{A}$ would be started.

Figure 6.2 shows the points obtained during the second iteration ($k = 2$) of MLSL when 50 more points are sampled. We note that the γkN candidate points in C_k are taken from the entire set, S_k, of kN sample points thus far. In this case, some of the sample points that were candidates in the previous iteration are no longer candidates in this iteration because their function values are no longer in the bottom half of values from S_2. The resulting critical distance is now $r_2 \approx .271$ and one possible set of r_2-descent paths is shown in Figure 6.2. We note that now three clusters emerge and that two new local optimization algorithms would be started, the third cluster not resulting in a new local optimization because the algorithm $\mathcal{A}$ was run from the best point in this cluster in the previous MLSL iteration.

We note that starting a local algorithm from the best point in the middle cluster in Figure 6.2 will result in finding the same local minimum as found when starting a local algorithm from the best point in the analogous cluster in Figure 6.1. In the next section, we show how we can take advantage of the information gained in the course of running a local algorithm to avoid unnecessarily starting a run of the local algorithm.

If the algorithm has terminated early (in our case, because some maximum number of function evaluations has occurred), the best approximate local minimizer is output as the approximate global minimum. We note that it is possible that approximate local minima are improved in the course of running the algorithm, since eventually a sample point may be obtained that is even closer to the true local minimum and has a smaller function value.

6.2 MIPE: Maximum Information from Previous Evaluations

We developed an alternate multistart method, MIPE (Maximum Information from Previous Evaluations). MIPE shares some features with MLSL but differs in that it seeks to take advantage of the information gained from using the function evaluations done by the local minimization algorithm $\mathcal{A}$.

The central idea of MLSL is to cluster points together if they were believed to belong to the same basin of attraction based on the function values of the sample points. The method for clustering was based on the critical distance r_k: at iteration k, sample points lie within a cluster if they are connected by an r_k-descent path, possibly through other sample points. Within each resulting cluster, a local minimization run is started from the sample point of minimum function value.

In the course of running a local algorithm $\mathcal{A}$, we produce a set of points $M_{i,k}$ (for the i local minimization in the kth multistart iteration) whose function values are being ignored during the clustering. Suppose now that we put the following restriction on the local algorithm $\mathcal{A}$:

(A1) Every point $x \in M_{i,k}$ is either within a distance r_k of a point $y \in M_{i,k}$ with $f(y) < f(x)$, or $f(x) = \min\{f(y) : y \in M_{i,k}\}$.

This condition simply states that any maximal r_k-descent path on the collection $M_{i,k}$ ends at a point achieving the value of the approximate local minimum found by $\mathcal{A}$.

While not technically necessary for the clustering, this condition ensures that any points produced in the course of a local optimization in iteration k would in fact be clustered within the same cluster as the sample point from which the local optimization was started. Hence, the points in $M_{i,k}$ will become part of an r_k-descent path that terminates at an approximate local minimum (found by the local minimization run). The result is the addition of a condition to the basic MLSL algorithm in Algorithm 6.1, which depends on the values at the previous optimization points in A_k. The resulting MIPE algorithm is formally detailed in Algorithm 6.2. We note that this does not affect the sampling procedure nor the update of the critical distance, it simply further restricts the number of local optimization runs started.

We again illustrate the intuition behind MIPE on the scaled and tilted Branin function. Figure 6.3 (left) shows the paths obtained as a result of running a local algorithm (with a maximum of $\mu_l = 50$ function evaluations) from each of the two starting points obtained as a result of the clustering in Figure 6.1. We pay particular attention to the approximate local minimizer found in the lower center of Figure 6.3 (left) and note that this point (and its surrounding neighbors in $M_{1,1}$ from the optimization) would eliminate the need to run an additional local optimization algorithm in the second MLSL iteration shown in Figure 6.2. This is

```
6.2.1. Evaluate $f$ at $N$ new sample points $x_{(k-1)N+1}, \ldots, x_{kN}$, which are included
       in the sample set $S_k$
6.2.2. Order all the $kN$ sample points so that $f(x_1) \leq \cdots \leq f(x_{kN})$
6.2.3. Define the reduced sample set of candidate points $C_k = \{x_1, \ldots, x_{\lceil \gamma kN \rceil}\}$
6.2.4. For each $x_i \in C_k$ at which a local run has not been started:
           If no other point $x_j \in C_k$ with $f(x_j) < f(x_i)$ is within a distance $r_k$
           then:
           6.2.4.a If a point $x^{\mathcal{A}} \in A_k$ evaluated in a previous local run and within
                   a distance $r_k$ of $x_i$ has $f(x^{\mathcal{A}}) < f(x_i)$:
                       Start $\mathcal{A}$ at a point $x \in A_k$ whose function value is least among
                       all points connected to $x^{\mathcal{A}}$ through an $r_k$-descent path, except
                       when $x$ was previously determined by $\mathcal{A}$ to be a local minimum
           6.2.4.b Otherwise start $\mathcal{A}$ at $x_i$
6.2.5. Update the critical distance $r_k$ (eg.- using (6.2))
```

Algorithm 6.2: Iteration k of the MIPE algorithm (given inputs N, γ, a critical distance r_k, a sampling procedure, and a local algorithm $\mathcal{A}$).

because the new candidate point in Figure 6.2 is within a distance r_2 of the local minimizer found by $\mathcal{A}$.

Our multistart procedure is called *MIPE* (Maximum Information from Previous Evaluations) to reflect that the clustering process is now using all of the function values obtained through both exploration (global sampling) and refinement (local minimization). While it is clear that MIPE will result in at most as many local optimization runs as the basic MLSL, we note that it is now possible that a local optimization run is started from a point obtained in the course of a previous local optimization run. This is because as the critical distance for clustering decreases, the maximal r_j-descent paths from optimization points obtained in iteration $j < k$, may not be r_k-descent paths terminating at a previous approximate local minimizer. In this case we will start a local optimization in iteration

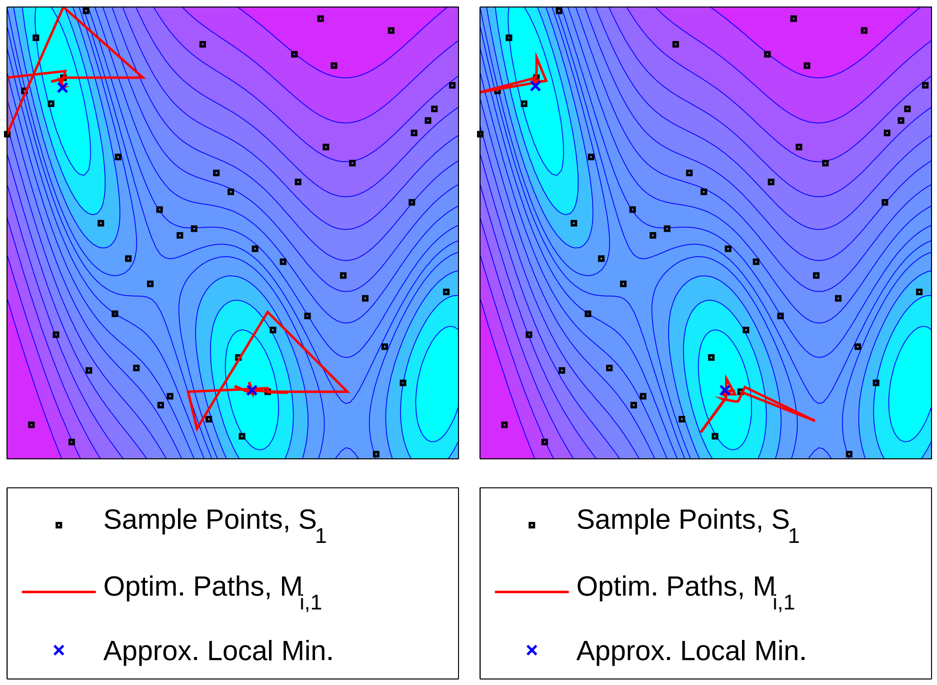


Figure 6.3: Local optimization paths on the scaled and tilted Branin function: (left) without use of the sample points, and (right) using the sample points.

k from this final point.

There is some ambiguity Step 6.2.4.a of Algorithm 6.2 since there could be several points from previous minimization runs in A_k within r_k of x_i and with a lower function value. Further, these points could in fact lie in separate clusters. In practice, we choose to take the point whose function value is minimum among all the optimization points within a distance r_k of x_i.

Since the clustering is only initiated from points within C_k and the resulting clusters of candidate sample points are supersets of the analogous clusters obtained by the basic MLSL clustering procedure, MIPE enjoys the same theoretical property of MLSL given in Theorem 6.1. Recall that our remaining goal is to achieve rapid function value reduction within a fixed computational budget. Since more optimization runs could be started in a MLSL iteration than in a MIPE iteration with the same sample points, a global algorithm using MIPE would have more evaluations available for additional global exploration and local refinement in future iterations.

6.3 The ORBIT Algorithm For Local Optimization

The second way in which we try to achieve rapid decrease in the function value is by using an efficient derivative-free local minimization algorithm. We will modify the ORBIT algorithm, introduced as an unconstrained derivative-free local optimization algorithm in [75], for use in the multistart global optimization algorithm GORBIT. We first describe the ORBIT algorithm and then describe changes, which were made to make ORBIT competitive in this global optimization setting.

ORBIT relies on a trust-region framework whereby the computationally expensive local problem $\min\{f(x) : x \in \mathbb{R}^n\}$ is replaced with the much simpler problem

$$\min \{m(x) : x \in \mathcal{B}\}, \tag{6.3}$$

where $m : \mathbb{R}^n \to \mathbb{R}$ is a model approximating f on $\mathcal{B}$. While the original function f is computationally expensive and has no available derivatives, the model m is computationally-attractive: it is inexpensive to evaluate and has analytical derivatives, so that it can be efficiently optimized over. We will trust the model m to approximate the function f within a ball of radius $\Delta > 0$ centered about the current iterate $\hat{x}$, $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - \hat{x}\| \leq \Delta\}$, called the *trust-region.*

The next point for evaluating f is obtained by solving the trust-region subproblem (6.3). The parameters $\hat{x}$ and Δ defining the trust-region $\mathcal{B}$ are updated

Table 6.1: Popular twice continuously differentiable RBFs that ORBIT works with.

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

based on the ratio of actual improvement to predicted improvement. This simple framework works well for smooth unconstrained problems whether derivatives are available or not.

When derivatives are unavailable the central question relates to how to form a model and ensure it sufficiently approximates the function within $\mathcal{B}$. ORBIT uses a *radial basis function* (RBF) model of the form

$$m(\hat{x}+s) = \sum_{j=1}^{|\mathcal{Y}|} \lambda_j \phi(\|s-y_j\|) + \nu_0 + \nu^T s, \tag{6.4}$$

where $s \in \mathbb{R}^n$ is a displacement from the current iterate $\hat{x}$, which interpolates the function at a set of points $\mathcal{Y} = \{y_1, \dots, y_{|\mathcal{Y}|}\}$. The model m is a linear combination of nonlinear basis functions plus a linear polynomial tail. Examples of common choices for the univariate function ϕ are given in Table 6.1. These functions are called radial because $\phi(\|x\|)$ is constant on spheres.

We find the model parameters λ, ν_0, and ν by requiring that the model interpolate the function at a set of points $\mathcal{Y}$:

$$m(\hat{x}+y_i) = f(\hat{x}+y_i) \qquad \forall y_i \in \mathcal{Y}. \tag{6.5}$$

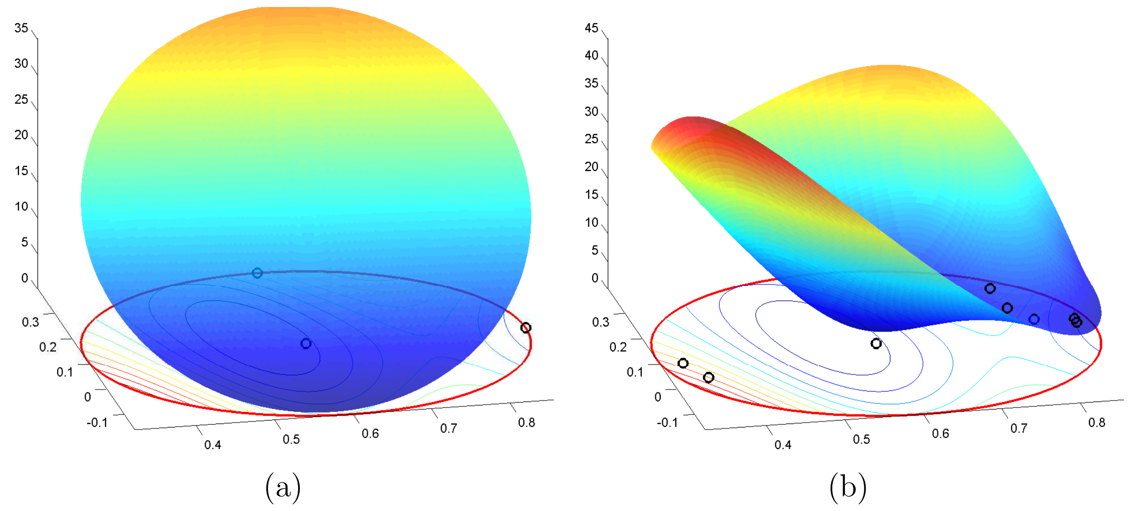


Figure 6.4: RBF models (represented by the colorful surface) interpolating a function (represented by the contour) over a 2-norm trust-region: (a) Interpolating 3 points; (b) Interpolating 8 sample points.

Though multivariate interpolation of scattered data is typically a difficult problem, for all of the ϕ functions given in Table 6.1, we can actually uniquely interpolate arbitrary functions as long as two conditions are met:

(Y1) The points in $\mathcal{Y}$ are distinct, and

(Y2) $n + 1$ of the points in $\mathcal{Y}$ are affinely independent.

Figure 6.4 shows two examples of a Gaussian RBF (represented by the surface), with $\phi(r) = e^{-r^2}$, interpolating a function (represented by the contour) of two variables. In Figure 6.4 (a) $|\mathcal{Y}| = n + 1 = 3$ points are interpolated, while in Figure 6.4 (b) $|\mathcal{Y}| = 8$ points are interpolated.

A major component of the algorithm is determining the interpolation set $\mathcal{Y}$ (required to satisfy **(Y1)** and **(Y2)** so that the model parameters are uniquely defined). ORBIT does this at every iteration based on the set of points that have been evaluated in its previous iterations. Of the history of points evaluated so far, it only considers those within a constant factor of the trust-region, those within $n\Delta$ of the current iterate $\hat{x}$. This is because we are only interested in the model approximating the function locally and including points that are farther away does not add to the quality of the model but may introduce ill-conditioning in our solution approach.

In fact, the interpolation conditions **(Y1)** and **(Y2)**, along with minor additional assumptions on the function f and model m, are sufficient for guaranteeing Taylor-like error bounds for our RBF models in [76] and Chapter 4.

Using this result, we have established global convergence of ORBIT to first order critical points, $\lim_{k\to\infty} \nabla f(x_k) = 0$, in [76] and Chapter 4, when $\mathcal{D} = \mathbb{R}^n$. The result is established using the recent general convergence theory of Conn, Scheinberg, and Vicente in [15]. This result ensures that if we have a budget that allows for enough evaluations, we can get to a stationary point with ORBIT.

ORBIT has been shown to work well locally on unconstrained functions, especially only a limited number of function evaluations is available. In our experience, this is because ORBIT greedily forms a model by taking full advantage of its previous function evaluations [75].

We made two major extensions to ORBIT in order for it to be suitable as the efficient local solver used in GORBIT:

1. allow for bound constraints, and
2. take advantage of externally-supplied, previously-evaluated points.

In this case, the previously-evaluated points will come from the multistart sampling points, S_k, and the points from previous local runs, A_k. For clarity, we will call the local algorithm with these extensions ORBIT-g.

6.3.1 Bound constraints in ORBIT-g

In the first modification, we note that ORBIT was designed to solve the unconstrained problem $\min\{f(x) : x \in \mathbb{R}^n\}$ but that we now would like ORBIT-g to solve the constrained problem (6.1). Throughout the remainder of this chapter it will be useful to focus on specific forms of the domain $\mathcal{D}$. For simplicity we will confine ourselves to a hyperrectangle defined by simple box constraints:

$$\mathcal{D} = \{x \in \mathbb{R}^n : l \leq x \leq u\}, \tag{6.6}$$

where the inequalities are taken component-wise and $l \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$ are vectors of finite lower and upper bounds, respectively. Without loss of generality we assume that $l_i < u_i$ for $i = 1, \ldots, n$ and hence $\text{vol}(\mathcal{D}) = (u_1 - l_1) \cdots (u_n - l_n) > 0$. Other domains are possible provided that appropriate changes are made to the parameters of the multistart method and the local algorithm $\mathcal{A}$.

If we would like a trust-region algorithm to only evaluate f within the bound constraints, the trust-region subproblem becomes

$$\min\{m(x) : x \in \mathcal{B} \cap \mathcal{D}\}. \tag{6.7}$$

We note that this problem is the same as the original trust-region subproblem (6.3), except when the current iterate is close to one of the bounds. Since $\mathcal{D}$ consists of simple bounds, it is relatively straightforward to approximately solve this subproblem using a projected gradient method following the approach detailed in [12]. However, unlike in [12], our model gradient is no longer the function's gradient, and hence there is currently no convergence theory for the resulting bound-constrained algorithm. This theoretical issue is beyond the scope of the present global algorithm and is left as future work. We note that the resulting local algorithm will only be run for a finite number of evaluations, μ_l. Further, the MLSL theory does not in fact demand that the local procedure converge to a local minimum of (6.1).

The trickier part about introducing bound constraints is that ORBIT needs to be able to evaluate at so-called *model-improving points*, which could lie anywhere within the trust-region $\mathcal{B}$. These points are necessitated by ORBIT's assumption that at least $n+1$ of the interpolation points are sufficiently affinely independent. As detailed in [75, 76], ORBIT obtains these model-improving points by looking at directions orthogonal to the span of nearby interpolation points and model-improving points must be evaluated on occasion if the quality of the RBF model has deteriorated.

In particular, for a trust-region radius $\Delta \in (0, \Delta_{\max}]$, a matrix of j acceptable nonzero displacements from $\hat{x}$, $Y = [y_2, \ldots, y_{j+1}]$, and parameter $\theta \in (0, 1]$, the projection of a model point $z \in \mathcal{B}$ onto the orthogonal complement of the span of points, $\mathcal{N}(Y^T)$, must be at least of magnitude $\theta\Delta$. In the unconstrained case, such a point is always available since the feasible point $z = \Delta Z_i$, where Z_i is a normalized basis vector for $\mathcal{N}(Y^T)$, always yields a projection of magnitude Δ. Such model-improving points are ideal, in the sense that they yield the maximum pivot value (of any point in $\mathcal{B}$) in the QR-like procedure used by ORBIT.

In the constrained case, these ideal model-improving points, ΔZ_i, may not be contained in $\mathcal{D}$. We illustrate this in Figure 6.5 (a), where we show a trajectory of recent optimization points, denoted by the x's, and a ball of radius $\theta\Delta$ extending around the current iterate. In this case, $\mathcal{N}(Y^T)$ consists of a single direction.

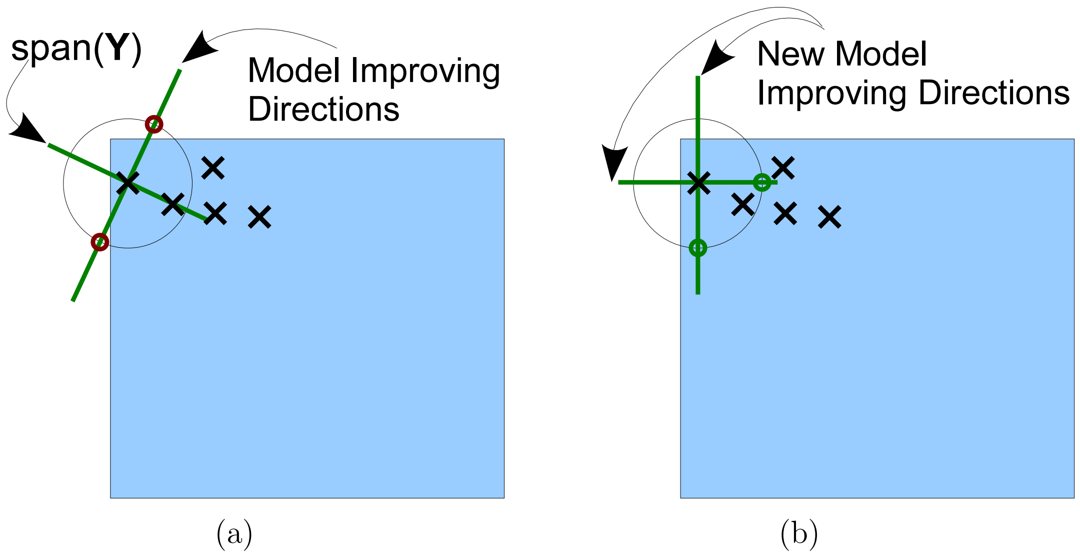


Figure 6.5: Model-Improving directions and bounds: (a) Standard model-improving points are infeasible; (b) Rotated model-improving points are feasible.

However, no point along this direction lies both within $\mathcal{D}$ and has a magnitude of more than $\theta\Delta$. The points that would be used in the unconstrained case are denoted by o's.

To fix this, we look at new directions that conform with the nearby boundaries. A similar approach is taken by constrained pattern search algorithms [43]. In this case we choose the coordinate directions and, as long as the trust-region radius is not too big, we will always have at least one point along these directions that is feasible. The resulting model-improving points are shown in Figure 6.5 (b). The downside of this approach is that it may mean that the algorithm must evaluate up to n additional points since the points previously in the interpolation matrix Y may not yield sufficiently large projections under this new set of directions.

In any case, we will restrict the maximum trust-region size to be bounded by half the length of the smallest side of the hyperrectangle $\mathcal{D}$, $\Delta_{\max} \leq \frac{1}{2}\min_i(u_i - l_i)$.

This will ensure it will always be possible to find an acceptable model-improving point using the rotated directions. We note that, in practice, we scale the variables so that we are effectively working with a domain corresponding to the unit hypercube, and hence each side will have unit length.

6.3.2 Using external points in ORBIT-g

The other main change from ORBIT to ORBIT-g was to allow the local procedure to take advantage of the additional data from the multistart sampling and previous local runs.

Since our procedure was developed for computationally expensive functions, it would make little sense to ignore the function values at the multistart sampling points in S_k. ORBIT constructs interpolation sets at each of its iterations by selecting nearby points at which the function has previously been evaluated during its current run. These points are added to the interpolation set provided that their locations satisfy geometric conditions. Since ORBIT does not care where these points come from, it is straightforward to expand the set of points considered.

Since ORBIT-g will be started many times within GORBIT, we allow ORBIT-g to additionally use nearby points previously evaluated in the course of the multistart sampling, S_k, and points previously evaluated in previous local minimization runs, A_{k-1}. This is a different approach than the basic MLSL with a local optimizer, since the local minimization runs are essentially independent of each other in MLSL.

This approach can also benefit the initial model used by ORBIT-g. By default, ORBIT initially builds a linear model (a special case of an RBF model) using the $n+1$ vertices of the standard simplex starting at the initial point $\hat{x}$ with sides of length Δ_0. However, these initial $n + 1$ evaluations can be avoided if an initial model can be formed using points close to $\hat{x}$ that have been evaluated in the course of the multistart sampling (or even other local runs).

In Figure 6.4 we return to the 2-dimensional example showing the Gaussian RBF model obtained when (a) the standard 3 initial simplex points are interpolated, and (b) 8 points obtained in the course of multistart sampling are interpolated. Once an initial model has been formed, ORBIT can begin its usual iteration of minimizing the model within the current trust-region. In Figure 6.4 (a) the linear model is a relatively inaccurate approximation of the function over the initial trust-region. In Figure 6.4 (b) we see that the RBF model is now interpolating a total of 8 points (the center plus 7 other sample points) and we are getting much better agreement between the RBF model (represented by the colorful surface) and the function (represented by the contours). In the worst case, there would be no previously-evaluated points close to the initial point and we would need to initialize as before.

We illustrate how using these sample points helps the local minimization algorithm by returning to the example of the scaled and tilted Branin function in Figure 6.3. Here we see that the optimization trajectory obtained using the default linear model initialization (shown in Figure 6.3 (left)) makes many more misguided steps away from the local minima before having enough points to better approximate the function and hence reduce the function value. In contrast, Figure 6.3 (right) shows the trajectories that result from using the nearby sample points. Since these points allowed for the construction of better initial models, a more direct path is taken to the local minima.

Just as the previously evaluated points were used in this example to yield a better initial RBF model (saving evaluations in the process), it could also be that other previously evaluated points can also be used as the trust-region moves from iteration to iteration. The benefit of using these points is that as ORBIT-g moves to areas that it has not previously explored, it may be able to avoid doing wasteful evaluations by knowing the function values at the points known at the global level by accessing information gained in previous multistart iterations.

We note that one of the reasons why we may want to avoid using the previously evaluated points is that they may in fact be from a previous local optimization and attract the local run out of a new basin of attraction and into one previously explored. Regis and Shoemaker were particularly concerned that reusing any function values [65] would result in this behavior. This issue is a particular concern of the present work where we hypothesize that controlling the size of the trust-region should help avoid this case.

This brings to light the last key ingredient of using ORBIT-g as the local minimization algorithm. We must set the maximum trust-region size $\Delta_{\max}$ to less than or equal to the current critical distance r_k. This is to ensure that the MIPE clustering procedure, which we have extended to use the points from the local run, will still obey the desired properties for r_k-descent paths. Hence in practice we will always ensure that the trust-region parameters obey $\Delta_0 \leq \Delta_{\max} \leq \min\left(\frac{1}{2}\min_i(u_i - l_i), r_k\right)$.

6.4 Preliminary Numerical Results

In the previous sections we have introduced the global algorithm GORBIT, which uses a multistart procedure for exploration (either MLSL or MIPE) and a modified version of ORBIT for refinement. As with many global optimization algorithms, GORBIT has a number of parameters, which the user must set before beginning an optimization run. We begin this section with a set of numerical experiments meant to determine the effect of some of these parameters in order to determine preliminary default parameters for GORBIT.

6.4.1 Performance Profiles

We follow the benchmarking notation of [52] and Chapter 2, where an algorithm is said to have converged if it obtains an iterate x satisfying

$$f(x_0) - f(x) \geq (1-\tau)(f(x_0) - f_L), \tag{6.8}$$

for a tolerance $\tau > 0$, a starting point x_0, and an attainable estimate of the true solution f_L. As discussed in [52] and Chapter 2, the convergence test (6.8) means that a solver has reduced the function value by at least $1-\tau$ times the best possible reduction, $f(x_0) - f_L$. The value f_L is usually set to the least function value obtained by any of the algorithms tested within μ_f evaluations.

We will refer to a *solver* as a specific implementation of an algorithm and use this convergence test when benchmarking a set of solvers $\mathcal{S}$ on a set of test problems $\mathcal{P}$. For each $s \in \mathcal{S}$ and $p \in \mathcal{P}$, we define $t_{p,s}$ as the number of function evaluations needed for s to satisfy (6.8) on p. The *performance profile* for solver $s \in \mathcal{S}$ is then defined as:

$$\rho_s(\alpha) = \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} : \frac{t_{p,s}}{\min\left\{t_{p,s} : s \in \mathcal{S}\right\}} \leq \alpha \right\} \right|. \tag{6.9}$$

The performance profile $\rho_s : \mathbb{R} \to [0,1]$ is a monotone increasing function, with higher values corresponding to better performance, and represents the fraction of problems for which the number of evaluations required by s is within a factor α of the solver that requires the fewest evaluations to solve p. Hence, $\rho_s(1)$ corresponds to the fraction of problems for which s required the fewest evaluations and $\rho_s(\infty)$ corresponds to the fraction of problems for which s was able to satisfy the convergence test (6.8) within μ_f evaluations.

6.4.2 Test Functions

Our test set consists of 67 different problems formed from the 54 different functions (some functions yielding two problems of differing dimensions) listed in Table 6.2. The first 50 functions are the test set considered in [2], while the last four functions are from [52] and [65]. These test functions are not expensive to evaluate but they share some key features of real world objective functions: they are multimodal and possess some detectable trends or patterns. They range in dimension from 2 to 20, the median being 5, and range from having a single difficult-to-find local minimum to having hundreds of local minima that are not global minima.

In order to minimize the effect of scaling, throughout these tests we will assume that (possibly after an affine transformation of the original domain) $\mathcal{D}$ is the unit hypercube in $\mathbb{R}^n$. However, the sampling procedure that we use in GORBIT is in fact invariant to this kind of scaling.

6.4.3 Results on the Test Functions

In each multistart iteration our implementation of GORBIT generates a *Latin Hypercube Sample* (LHS) of size N. Latin hypercube sampling is a sample stratifica-

Table 6.2: Test functions for the computational experiments.

Test Function	n	Domain
Ackley	10	$[-30, 30]^{10}$
Aluffi-Pentini's	2	$[-10, 10]^2$
Becker and Lago	2	$[-10, 10]^2$
Bohachevsky 1	2	$[-50, 50]^2$
Bohachevsky 2	2	$[-50, 50]^2$
Branin	2	$[-5, 10] \times [10, 15]$
(3-hump) Camel back	2	$[-5, 5]^2$
(6-hump) Camel back	2	$[-5, 5]^2$
Cosine Mixture	2, 4	$[-1, 1]^n$
Dekkers and Aarts	2	$[-20, 20]^2$
Easom	2	$[-10, 10]^2$
Epistatic Michalewicz	5,10	$[0, \pi]^n$
Exponential	10	$[-1, 1]^{10}$
Goldstein-Price	2	$[-2, 2]^2$
Griewank	5,10	$[-600, 600]^n$
Gulf Research	3	$[.1, 100]\times$ $[0, 25.6]\times$ $[0, 5]$
Hartman3	3	$[0, 1]^3$
Hartman6	6	$[0, 1]^6$
Helical Valley	3	$[-10, 10]^3$
Hosakie	2	$[0, 5] \times [0, 6]$
Kowalik	4	$[0, .42]^4$
Levy and Montalvo 1	3	$[-10, 10]^3$
Levy and Montalvo 2	5,10	$[-5, 5]^n$
McCormick	2	$[-1.5, 4] \times [-3, 3]$
Meyer and Roth	3	$[-10, 10]^3$
Miele and Cantrell	4	$[-1, 1]^4$
Modified Langerman	5,10	$[0, 10]^n$
Modified Rosenbrock	2	$[-5, 5]^2$
Multi-Gaussian	2	$[-2, 2]^2$
Neumaier 2	4	$[0, 4]^4$
Neumaier 3	10,15	$[-n^2, n^2]^n$
Odd Square	10,20	$[-15, 15]^n$
Paviani	10	$[2.01, 9.99]^{10}$
Periodic	2	$[-10, 10]^2$
Powell's Quadratic	4	$[-10, 10]^4$
Price's Transistor	9	$[-10, 10]^9$
Rastrigin	5,10	$[-5.12, 5.12]^n$
Extended Rosenbrock	10	$[-30, 30]^{10}$
Salomon	5,10	$[-100, 100]^n$
Schaffer 1	2	$[-100, 100]^2$
Schaffer 2	2	$[-100, 100]^2$
Shubert	2	$[-10, 10]^2$
Schwefel	10	$[-500, 500]^{10}$
Shekel5	4	$[0, 10]^4$
Shekel7	4	$[0, 10]^4$
Shekel10	4	$[0, 10]^4$
Shekel's foxholes	5,10	$[0, 10]^n$
Sinusoidal	10,20	$[0, \pi]^n$
Storn's Tchebychev 1	9	$[-128, 128]^9$
Storn's Tchebychev 2	17	$[-32768, 32768]^{17}$
Wood's Function	4	$[-10, 10]^4$
Osborne 2	11	$[0, 2]^{11}$
Broyden Tridiagonal	10	$[-2, 2]^{10}$
Broyden Banded	10	$[-2, 2]^{10}$
Schoen	7,12	$[0, 1]^n$

```
6.3.1. For $i = 1, \ldots, N$
    6.3.1a. For $j = 1, \ldots, n$
            Generate $U \sim$Uniform$[0,1]$, and
            Set $X_{i,j} = \frac{U+i-1}{N}$
    6.3.1b. Randomly permute the elements of the $i$th row of $X$, $[X_{i,1}, \ldots, X_{i,n}]$
6.3.2. Output each of the $N$ rows of the matrix $X$ as a sample point
```

Algorithm 6.3: Generating a (uniform) Latin hypercube sample of size N over the unit hypercube in $\mathbb{R}^n$ [49].

tion strategy seeking to ensure that the domain is better covered [49]. Unlike when sampling uniformly within $\mathcal{D}$, the sample size N must now be known a priori. Algorithm 6.3 shows the procedure we use for obtaining a matrix of N n-dimensional points in $\mathcal{D}$. It should be noted that, from this sampling alone, Theorem 6.1 thus guarantees that the basic MLSL and MIPE algorithms will produce a sequence of iterates that converge to the global minimum of any continuous function provided $N \geq 1$ points are sampled in each iteration.

For our implementations, we made one change to the basic MLSL and MIPE algorithms in Algorithms 6.1 and 6.2. In both algorithms we now no longer eliminate a candidate point $x_i \in C_k$ if a local minimization run has previously been started. This is because our limit on the maximum number of evaluations per local minimization, μ_l, may prevent the local minimizer from converging. Hence, we will not start a run at such a point only if the the local algorithm certified that it was a local minima. Otherwise we will start the run from the approximate minima obtained when the local algorithm terminated after reaching the maximum number of evaluations, μ_l (and hence further refine that minima estimate).

The ORBIT algorithm upon which our ORBIT-g is built is detailed in [75] and

```
6.4.1. Let $\kappa_{mc} = \sum_{\{i:\hat{\chi}_i=u_i,[\nabla m(\hat{x})]_i<0\}\cup\{i:\hat{\chi}_i=l_i,[\nabla m(\hat{x})]_i>0\}} [\nabla m(\hat{x})]_i^2$
6.4.1. If $\kappa_{mc} \leq \epsilon$ and $m$ is fully linear on $\{x \in \mathcal{D} : \|x - \hat{x}\| \leq \epsilon\Delta_0\}$:
        Return from ORBIT-g with approximate local minimizer $\hat{x}$
```

Algorithm 6.4: Criticality/Termination test used by ORBIT-g in the numerical tests.

Chapter 3). We used an ∞-norm trust-region and the following parameter values: $\Delta_{\max} = \min\left(r_k, \frac{1}{2}\min_i(u_i - l_i)\right)$, $\Delta_0 = \frac{1}{10}\Delta_{\max}$, $\eta_0 = 0$, $\eta_1 = .2$, $\gamma_0 = \frac{1}{2}$, $\gamma_1 = 2$, $\epsilon = 10^{-5}$, $\kappa_d = 10^{-4}$, $\alpha = .9$, $\theta_1 = 10^{-3}$, $\theta_2 = 10^{-7}$, $\theta_3 = 10$, $\theta_4 = 10$, $p_{\max} = 2n+1$, with the starting point, x_0, determined by the multistart procedure. The remaining parameters (the type of radial basis function, $\phi(r)$, and the maximum number of evaluations per local optimization run μ_l) are treated as inputs to GORBIT in addition to N, γ, and the maximum total number of evaluations, μ_f. The ORBIT algorithm detailed in [75] and Chapter 3 uses a criticality test that could result in an infinite loop once a stationary point is discovered. Here we use the modified criticality/termination test shown in Algorithm 6.4. If $\hat{x}$ is an interior point and satisfies the test in Algorithm 6.4, we have that $\|\nabla f(\hat{x})\| \leq \epsilon(1 + \kappa_g\Delta_0)$, where κ_g is a constant associated with the model being fully linear. The reader interested in the concept of a fully linear model ad resulting approximation bounds is directed to Chapter [76] and Chapter 4.

Since our goal is to eventually apply GORBIT to expensive functions, we will only generate a small number of sample points ($N = 10$) and reduce the cumulative sample by ($\gamma = .5$) to select the starting points for the local optimization runs. Our goal will be to limit the total number of evaluations spent by the global algorithm, due to both the sampling procedure and the local optimization runs.

Using inexpensive test problems allows us to examine difficult cases and run the algorithm for 30 trials since it relies on a stochastic sampling component. For each of the 30 trials of the 67 problems, we recorded the best function value as a function of the number of times the function was evaluated for a maximum of $\mu_f = 1200$ function evaluations. Hence each solver run corresponds to roughly 2.4 million evaluations. Since performance profiles measure performance relative to best solver on a particular run, our problem set $\mathcal{P}$ will in fact consist of the $30 * 67 = 2010$ (problem,trial) pairs. Each trial is defined by a different seed of the random number generator used by the Latin hypercube sampling procedure in GORBIT. As the local solver ORBIT-g is deterministic, this means that, for a specific trial, each different version of GORBIT is encountering the same set of sampled points. Hence, differences in performance can be solely attributed to the differences due to the parameter selection.

The first solver set we consider is the set $\mathcal{S}_1$ consisting of four different global algorithms:

MIPE GORBIT: Both the multistart procedure and ORBIT-g have knowledge of the entire evaluation history;

MLSL GORBIT: ORBIT-g has knowledge of the sampling evaluations done by previous local minimizations and the multistart procedure;

MIPE using ORBIT: The multistart procedure has knowledge of the evaluations done by ORBIT;

MLSL using ORBIT: Both ORBIT and the multistart procedure do not know the evaluation history, each using the same LHS and a local algorithm using a cubic radial function, $\phi(r) = r^3$, interpolating at most $p_{\max} = 2n + 1$ points for a maximum of $\mu_l = 100$ function evaluations in the local minimization.

Figure 6.6 presents performance profiles for the solver set $\mathcal{S}_1$ for $\tau = 10^{-k}$ with $k \in \{1, 3, 5, 7\}$, corresponding to a $100(1-\tau)\%$ reduction relative to the best possible reduction in $\mu_f = 1200$ function evaluations. For example, in the upper right plot we see that when a 99.9% reduction is demanded ($\tau = 10^{-3}$, both MIPE GORBIT and MLSL GORBIT are the fastest solvers (require the fewest evaluations to attain this reduction level) on 50% of the problems ($\rho_s(1) = .5$). Further, MLSL GORBIT is able to achieve this reduction level on over 78% of the problems within a factor of 4 times the fewest evaluations of the best solver for each problem ($\rho_s(4) \approx .78$).

These results indicate that the primary split between these four variants is based on whether the multistart procedure uses ORBIT (without any information from the evaluation history) or GORBIT (with complete knowledge of both the local optimization and global exploration history). It also appears that for less accurate levels ($\tau = 10^{-1}, 10^{-3}$, shown in the top half of Figure 6.6) the MLSL procedure does better than the the MIPE procedure, while for higher accuracy levels ($\tau = 10^{-5}, 10^{-7}$, shown in the bottom half of Figure 6.6) the converse is true.

We note that one of the settings in which MLSL may perform better than MIPE is on functions where the distance between local minima is much smaller than the initial critical distance r_1 in (6.2). On such functions, MIPE may rule out starting a local minimization at a candidate point if it is within the critical distance of a previously found local minimum with a lower function value than the candidate

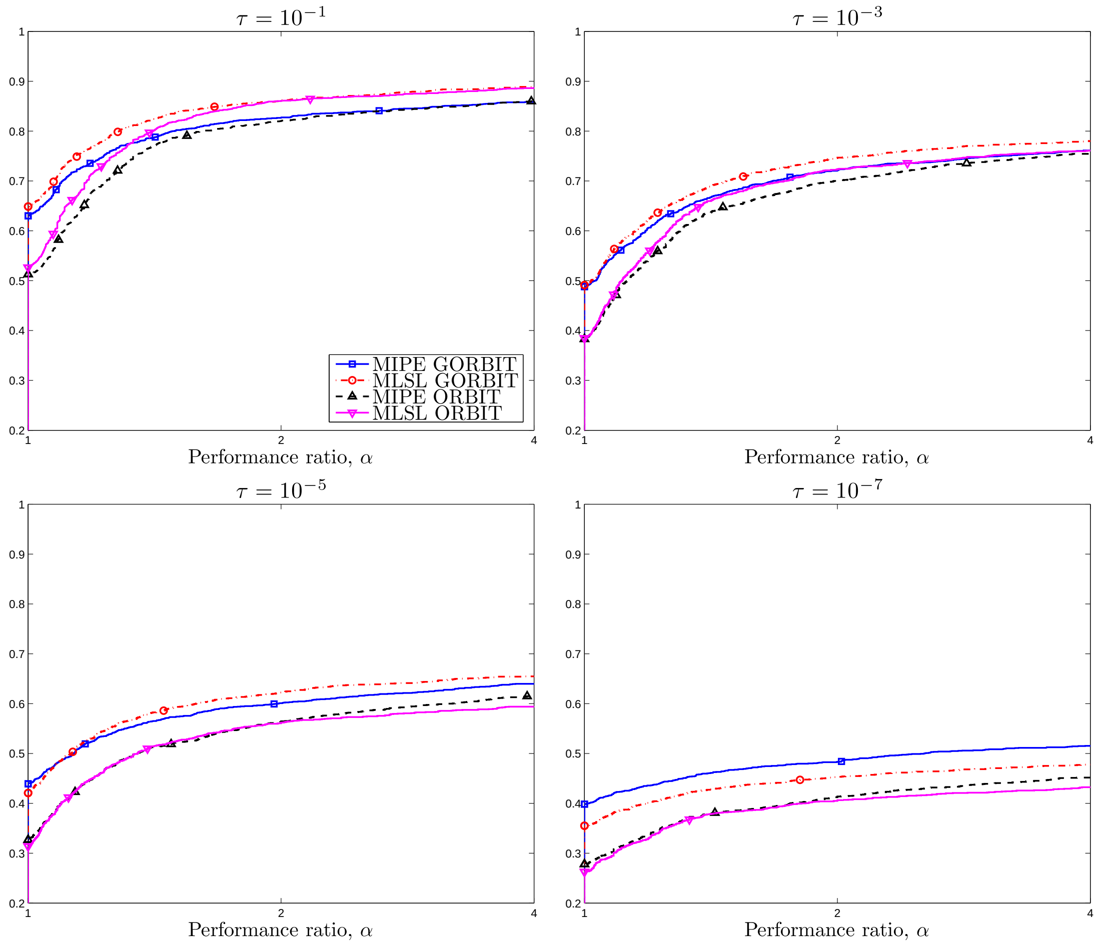


Figure 6.6: Performance profiles $\rho_s(\alpha)$ (6.9) for different combinations of global and local strategies with the solver set $\mathcal{S}_1$ using $p_{\max} = 2n+1$ and $\mu_l = 100$ ($\log_2$-scale for the x-axis) for the problems listed in Table 6.2. These profiles show the fraction of problems for which a solver is within a factor α of the best solver on each problem (in terms of the number function evaluations).

point. Since MLSL does not use the optimization run for the clustering, it would start a local minimization at this candidate point and could find a different local minimum. However, we note that (as higher accuracy solutions are demanded) the MIPE procedure seems to be more efficient, both with ORBIT-g and ORBIT.

Having motivated the use of GORBIT relative to alternative procedures using ORBIT without knowledge of the evaluation history, we now seek to determine how sensitive the procedure is to the parameter μ_l determining the maximum number of evaluations available for each run of the local solver ORBIT. In practice the choice of this parameter should be guided by the expectation of how many local minima/stationary points are contained within $\mathcal{D}$. We note this parameter also determines the fraction of the total number of evaluations dedicated to global exploration versus local refinement. In particular, since a typical multistart iteration will start no more than γN local minimizations, a rough estimate of the percentage of evaluations in each iteration devoted to local exploration is:

$$\frac{\gamma N \mu_l}{\gamma N \mu_l + N} = \frac{\gamma \mu_l}{\gamma \mu_l + 1}, \tag{6.10}$$

the second term in the denominator corresponding to the number of sample points in each iteration. In our experience, (6.10) is an overestimate of this percentage since usually fewer than γN local minimizations are started in each global iteration and the local minimization could terminate before μ_l evaluations if the local convergence tolerance has been met. We note that if we expect to perform k multistart iterations with a budget of μ_f total evaluations, we could estimate the maximum number of evaluations in each local minimization as

$$\mu_l = \frac{\mu_f - N}{k \gamma N}. \tag{6.11}$$

Given no information on the number of minima or number of iterations expected, the next solver set we consider, $\mathcal{S}_2$, again uses a cubic RBF and consists of three different values of μ_l:

A. MIPE GORBIT with $\mu_l = 70$;

B. MIPE GORBIT with $\mu_l = 100$;

C. MIPE GORBIT with $\mu_l = 130$.

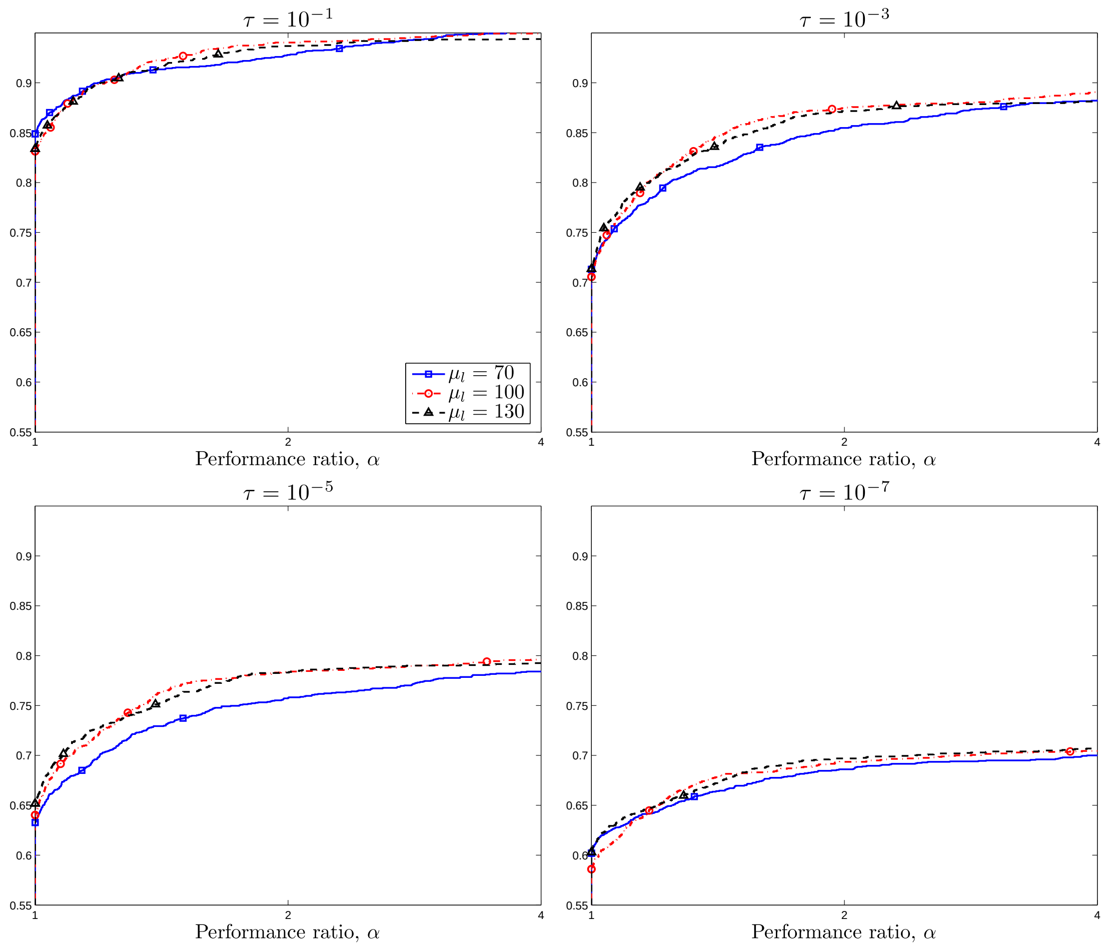


Figure 6.7: Performance profiles $\rho_s(\alpha)$ for the variants of MIPE GORBIT in $\mathcal{S}_1$, with different values of μ_l determining the maximum number of evaluations in each local minimization ($\log_2$-scale for the x-axis) for the problems listed in Table 6.2. These profiles show the fraction of problems for which a solver is within a factor α of the best solver on that problem (in terms of the number function evaluations).

Figure 6.7 shows the data profiles for solver set $\mathcal{S}_2$ again using the accuracy levels $\tau = 10^{-k}$, $k \in \{1, 3, 5, 7\}$. Here we see only minor differences in performance depending on which variant is used, with the variant using fewer evaluations per local minimization, $\mu_l = 70$, generally being slightly less efficient than the others. Recall that for a fixed budget of μ_f evaluations, lowering the parameter μ_l has the effect of being able to run more local minimizations (since each requires fewer evaluations), but that the local minima obtained in these minimizations may not be as accurate.

As we have found when doing unconstrained local optimization in the previous chapters, the number of evaluations needed to obtain a local minimum grows with the dimension, n, of the problem. Since the test problems in $\mathcal{P}$ (Table 6.2) varied in dimension, we look at the subset of problems for which $n \geq 6$ in Figure 6.8. As expected, the differences between the variants allowing for longer local minimization runs and the variant allowing only $\mu_l = 70$ evaluations are now more pronounced. This underscores the importance of accounting for the dimension of the problem when selecting parameters for GORBIT. The results in Figure 6.8 suggest that $\mu_l = 130$ is probably as good or better than smaller values of μ_l for problems of dimension $n \geq 6$.

6.4.4 Finding Large Growth Factors

We now illustrate the use of GORBIT on a classical global problem sharing characteristics with many simulation-based problems while being relatively inexpensive to evaluate, and thus allowing for more extensive testing.

Given a matrix $A \in \mathbb{R}^{d \times d}$, let $PA = LU$ denote an LU decomposition with row pivoting. Following [71], we define the growth factor

$$\psi(A) = \frac{\max_{i,j} |U_{i,j}|}{\max_{i,j} |A_{i,j}|}. \tag{6.12}$$

We note that the growth factor ψ obeys the bound $\psi(A) \leq 2^{d-1}$. As noted in [28] and [22], this growth factor measures how far off the solution $\hat{x}$ computed using

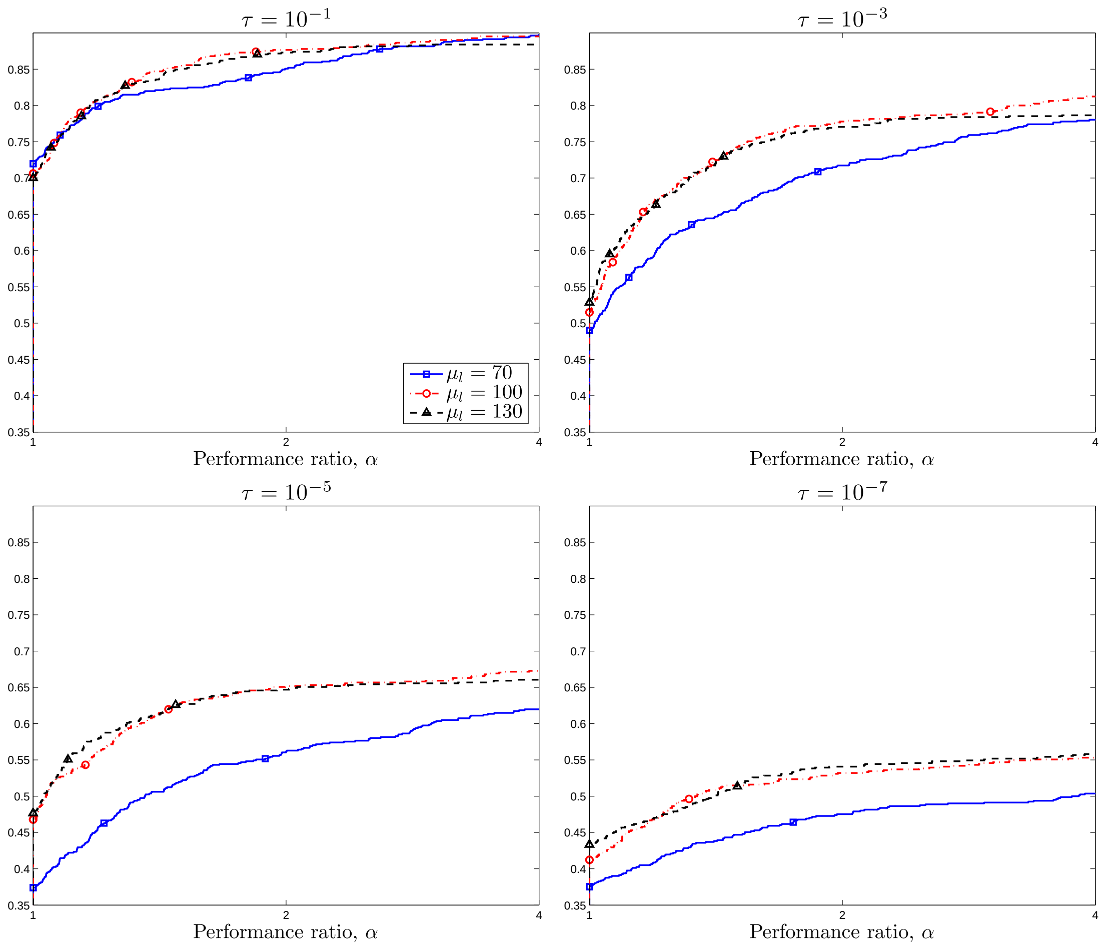


Figure 6.8: Performance profiles $\rho_s(\alpha)$ for the same conditions as in Figure 6.7 but only applied to those problems in Table 6.2 of dimension $n \geq 6$.

Gaussian elimination to solve $Ax = b$ can be:

$$(A + \delta A)\hat{x} = b \quad \text{for some } \delta A \text{ with } \|\delta A\|_\infty \leq 3n^3 u\psi(A)\|A\|_\infty, \tag{6.13}$$

where u denotes the unit roundoff of the computer.

Exponential growth factors are rare in practice and bounds such as (6.13) have prompted researchers to search for matrices near the upper bound 2^{d-1}. In [35], Higham searches for such matrices using direct search methods and notes that the problem has many local minima. Further, ψ is in fact discontinuous as a result of the row pivoting. The Monte Carlo studies in [22] suggest that large growth factors are exceptionally rare. The authors in [22] estimated that the probability of randomly drawing an 8×8 matrix with a growth factor of $\psi(A) = 40$ to be roughly 10^{-20}.

Inspired by [35], we seek to solve

$$\min \left\{ \frac{1}{\psi(A(x))} : x \in [-1, 1]^{d^2} \right\}, \tag{6.14}$$

where $A(x) \in \mathbb{R}^{d \times d}$ is the matrix formed by reshaping the $n = d^2$-dimensional vector x.

On this problem we ran both (MIPE and MLSL) variants of GORBIT for a total of $\mu_f = 3500$ evaluations. GORBIT used the same parameters as in the previous tests with a maximum of $\mu_l = 150$ evaluations per local optimization, a maximum of $p_{\max} = 2n+1 = 33$ interpolation points, $N = 20$ sample points per global sample and $\gamma = .25$, so that in the kth global iteration there were a total of $k\gamma N = 5k$ candidate points.

In addition to the two GORBIT variants, we present results for two other solvers.

Simulated annealing is a popular approach for the global optimization of continuous functions. We used an implementation due to Regis and Shoemaker in [65] of the algorithm developed by Sait and Youssef [66]. The parameters of the algorithm (the initial temperature T_0 and the cooling rate α) are obtained by an initial budget of $N = 20$ evaluations devoted to estimating good values as described in [65]. The $N = 20$ points used for this were the $N = 20$ sample points of the other algorithms.

The Nelder-Mead algorithm is a popular method for many application problems. NMSMAX is an implementation due to Higham [34] and was selected since it was one of the primary solvers used in [35]. We note that NMSMAX is only guaranteed to converge to local minima and hence a multistart approach would need to be used to guarantee finding the global minimum. The default parameters for NMSMAX were used with the termination criteria set to 0 to ensure that the algorithm would exhaust the computational budget of $\mu_f = 3500$ evaluations. Since NMSMAX is defined for maximization problems, it was given the negative of f.

As was done on the test problems, we ran multiple trials on this problem, each corresponding to a different seed of the random number generator used in the sampling procedures. NMSMAX is a deterministic algorithm and hence it was started at the point with the lowest function value of the initial $N = 20$ sample points. As a result, for each trial, each of the solvers tested had the same initial $N = 20$ data points. We ran a total of 30 trials for these tests.

In Figure 6.9 we show the trajectory of the function values obtained in one sample trial by three of the solvers (MLSL GORBIT showing similar behavior to the MIPE GORBIT trajectory shown). Here we see the multiple local optimization runs (each of length no more than 150 evaluations) of MIPE GORBIT, which run from evaluations 20 to 136, 137 to 225, and 226 to 366. We also see that MIPE GORBIT finds local minima (shown by the flat tails of each optimization run. NMSMAX consists of a single run (that could get caught in a local minimum that is not a global minimum) and we see that the trajectory is generally a single decreasing trend. The Simulated Annealing solver used shows much greater variation in function values but generally also tends to decrease (especially after more evaluations have been performed).

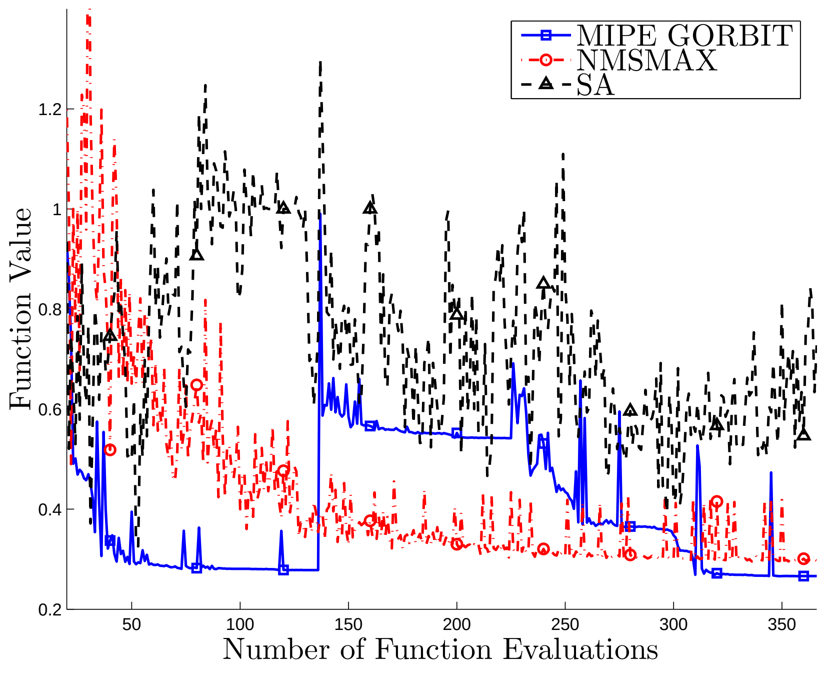


Figure 6.9: Sample function value trajectories for one trial of searching for large growth factors in (6.14). The local minimizations done by MIPE GORBIT occur from evaluations 20 to 136, 137 to 225, and 226 to 366.

In Figure 6.10 we collect the 30 trials performed and show the mean best function value obtained. Hence, for a fixed number of evaluations, the plot shows the mean value of the approximate minimizer that would be returned. We note that, by design, all four solvers had the same mean value for the first $N = 20$ evaluations. Next we note that the MIPE and MLSL GORBIT solvers on average yield lower function values for most all numbers of function evaluations. Further, these solvers perform similarly, the MLSL variant performing better initially and the MIPE variant obtaining lower function values in the long run. The NMSMAX solver is actually the best solver on average between 250 and 1500 function evaluations, but then sees very small reductions of the ψ value after 1400 evaluations, often getting stuck in a local minimum. Recall that in practice, NMSMAX would

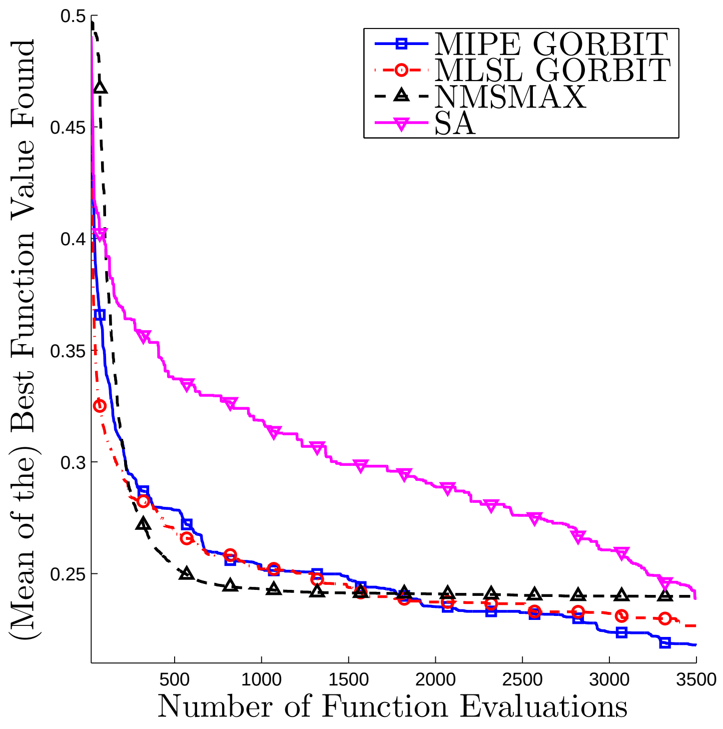


Figure 6.10: Mean in 10 trials of the best function value obtained when searching for large growth factors in (6.14).

need to be used with more of a global exploration phase to be guaranteed to find a global minimum. The Simulated Annealing solver tested required more evaluations to see significant reductions but on average found matrices with lower ψ values than NMSMAX after $\mu_f = 3500$ evaluations.

The best ψ value found in all of the trials in $\mu_f = 3500$ evaluations was less than .14, and hence more evaluations would be required to find ψ values closer to the lower bound of .125. Before concluding we again note that the function f has points of discontinuity.

6.5 Conclusions and Future Work

In this chapter we have introduced a new algorithm GORBIT for global optimization, which makes use of previously-evauated points to improve the efficiency of the global optimization. GORBIT uses a multistart procedure for exploration and a modified version, ORBIT-g, of the local minimization algorithm ORBIT for refinement. We have introduced a new multistart procedure MIPE, which utilizes the output from previous local minimization runs in an effort to avoid repeatedly finding the same local minimum. MIPE shares features with MLSL and also convergences to a global minimum using finitely many local minimization runs with probability 1.

Both MIPE and ORBIT-g were developed with the goal of using the history of evaluations performed in the course of running a global multistart algorithm. This algorithm development is motivated by problems where the objective function f is computationally expensive to evaluate and hence a considerable computational effort was paid for each of these evaluations. We note that MIPE could be used with other local optimization solvers and that ORBIT-g could be used with other multistart procedures.

In the course of our numerical experiments we have found that using full knowledge of the history of evaluations is not always beneficial. We have seen that on some of the test problems (for example, the top half of Figure 6.6)), MLSL achieves more efficient reductions of the function value, especially on problems characterized by tightly clustered local minima with significantly different function values. These same tests showed that the local minimization algorithm employed benefits significantly from knowing the function values at the points previously evaluated by the global sampling procedure and previous local minimization runs.

We are currently analyzing the theoretical implications of the bound-constrained variant of ORBIT used in GORBIT, with the goal of proving convergence to critical points in this setting.

In the future, we expect to do more numerical testing of the algorithm to better understand when MIPE or MLSL should be used. These numerical studies will also help us determine good default parameters for GORBIT. We believe that the performance of GORBIT can be improved by taking advantage of knowledge of the number of local minima, a topic saved for future work. Future studies will also be devoted to further analyzing the sensitivity of the performance relative to the parameter selection.

We also intend to pursue versions of GORBIT that can take advantage of parallel computing environments, where the local minimizations are run in parallel. In this context it will also be critical to determine the benefits of communication among these local minimization runs, especially if the path of their points approach one another.

Acknowledgements

The authors wish to thank Rommel Regis for writing the code for the Simulated Annealing algorithm tested here as well as many helpful conversations along the way. The authors are also grateful to Jorge Moré for suggesting the application of finding large growth factors. This work was supported by a Department of Energy Computational Science Graduate Fellowship, grant number DE-FG02-97ER25308 and by NSF grants CCF-0305583 and DMS-0434390.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this dissertation we have addressed general simulation-based optimization problems when the objective function is computationally expensive to evaluate. We have developed and analyzed new algorithms for both local and global optimization when only function values are available. Our algorithms make use of the expensive function evaluations done throughout the course of the optimization in order to obtain better approximate solutions in fewer expensive function evaluations than existing methods.

In Chapter 2 we proposed a methodology for analyzing how solvers perform on a set of test problems when there is a computational budget limiting the number of function evaluations and when derivatives are unavailable. We introduced data profiles for measuring the fraction of problems that can be solved in terms of the number of function evaluations. We also introduced collections of smooth, noisy, and piecewise-smooth problems and analyzed the performance of three solvers on these problems. These tests showed that a model-based method outperformed two popular director search methods.

Encouraged by these results, in the subsequent chapters we developed and analyzed new model-based algorithms. Chapter 3 introduced the new algorithm ORBIT (Optimization by Radial Basis Interpolation in Trust-regions) for solving unconstrained local optimization problems. ORBIT uses a Radial Basis Function (RBF) model interpolating the function on a set of points at which the function has been evaluated. ORBIT is a trust-region method, and hence the model is trusted to approximate the function within a local neighborhood of the best point found so far. We developed procedures for using the RBF property of conditional positive definiteness to ensure that the RBF interpolation is unique. As a result, ORBIT is able to build nonlinear models of the function given given as few as $n + 1$ function values. We also presented numerical results on sets of test problems and two environmental engineering applications. These results support the effectiveness of ORBIT, when relatively few function evaluations are available, on blackbox functions for which no special mathematical structure is known or available.

In Chapter 4 we pursued several theoretical issues within ORBIT. We showed that it is possible to obtain global convergence of trust-region methods using very general interpolation models. These models need only satisfy minimal smoothness conditions and interpolate on a set of sufficiently affinely independent points. We provided a procedure for obtaining such affinely independent sets as well as a procedure for using more of the previously-evaluated points while keeping the model well-conditioned. We also analyzed the choice of radial function used by the RBF model, giving conditions for the radial model to fit within the globally convergent framework of ORBIT and testing different radial functions on a set of noisy test problems from Chapter 1. In addition to numerical tests where we vary the maximum number of interpolation points, we also run ORBIT on a truly expensive application problem related to cleaning up contamination on a former naval ammunition depot in Hastings, Nebraska requiring almost 1 CPU hour per evaluation.

This general framework allows consideration of other models beyond the RBF ones used by ORBIT. In Chapter 5 we showed how our general framework can be extended to quadratic polynomial models. In order to allow for interpolation of fewer points than the dimension of quadratics requires, these models will primarily be underdetermined and we propose an algorithm using quadratic models whose

Hessians are of minimum norm. We illustrated this algorithm on test functions and showed the benefits gained from the flexibility of interpolating variable numbers of function values.

Many simulation-based problems are not adequately addressed by local optimization theory and algorithms, and for this reason we addressed global optimization problems in Chapter 6. We proposed a new algorithm, GORBIT, which employs ORBIT within a multistart method to solve bound-constrained global optimization problems. We introduced a new multistart method MIPE (Maximum Information from Previous Evaluations) so that the global exploration phase can take advantage of the function evaluations done by the local ORBIT algorithm. We also made modifications to ORBIT, allowing ORBIT to take advantage of both the function evaluations done in the global exploration and other minimization runs, and accounting for bound-constraints. GORBIT differs from existing multistart-based algorithms such as in that it seeks to use as much of its own evaluation history as possible. Our numerical results showed the effect of using this evaluation history on a set of global test problems. Finally, we applied these algorithms to the global optimization problem of finding matrices that result in large errors when solving a system by Gaussian elimination.

There remain many open avenues for future work and we briefly mention some areas of particular interest to us. First, many simulation-based problems are inherently *noisy*. Often this is due to computational noise associated with the discretizations and finite convergence parameters for many of the nonlinear systems and PDE solvers that the simulator relies on. There remains a demand for solving such noisy problems when relatively few function evaluations are available. Next, in this paper we have addressed serial optimization algorithms. Here we acknowledge that distributed computing is increasingly prevalent and that many simulation-based functions can only allow for a limited degree of parallelization. As a result, there are processors that a serial algorithm would leave idle while the function is being evaluated. In the future we hope to address how these additional processors could be utilized in our approach. Lastly we note that some real applications consist of computationally simple objective functions, but have nonlinear constraints that are computationally expensive to evaluate. Much work remains to be done in the area of derivative-free model-based methods to account for more general constraints.

APPENDIX A

INTERMEDIATE LEMMAS FOR CHAPTER 4

This appendix lists the intermediate lemmas used to establish Theorem 4.2. We seek to follow the corresponding progression of lemmas provided in [15]. Differences are primarily due to the use of a general trust-region norm $\|\cdot\|_k$ and the resulting sufficient decrease condition. Detailed proofs are only provided when this affects the content of the proof.

We begin by using the trust-region mechanics to ensure that the approximate solution to the subproblem will be accepted provided that the trust-region is small enough. The following Lemma relies on both our sufficient decrease condition and the property of a model being fully linear.

Lemma A.1. *If m_k is fully linear on $\mathcal{B}_k$ and*

$$\Delta_k \leq \|\nabla m_k(x_k)\|_k \min\left\{\frac{1}{\kappa_H}, (1-\eta_1)\frac{\kappa_d}{4\kappa_f}\frac{\|\nabla m_k(x_k)\|^2}{\|\nabla m_k(x_k)\|_k^2}\right\}, \tag{A.1}$$

then iteration k is successful.

Proof. (Largely follows [15, Lemma 5.2]) Since $\Delta_k \leq \frac{\|\nabla m_k(x_k)\|_k}{\kappa_H}$, the sufficient decrease condition (4.24) implies that the step in the kth iteration satisfies:

$$m_k(x_k) - m_k(x_k+s_k) \geq \frac{\kappa_d}{2}\frac{\|\nabla m_k(x_k)\|^2}{\|\nabla m_k(x_k)\|_k}\Delta_k. \tag{A.2}$$

Since m_k is fully linear on $\mathcal{B}_k$, (4.6) gives:

$$\begin{aligned}
1-\rho_k &\leq |\rho_k - 1| \\
&\leq \left|\frac{f_k(x_k+s_k) - m_k(x_k+s_k)}{m_k(x_k) - m_k(x_k+s_k)}\right| + \left|\frac{f_k(x_k) - m_k(x_k)}{m_k(x_k) - m_k(x_k+s_k)}\right| \\
&\leq 4\frac{\kappa_f}{\kappa_d}\frac{\|\nabla m_k(x_k)\|_k}{\|\nabla m_k(x_k)\|^2}\Delta_k \\
&\leq 1-\eta_1,
\end{aligned}$$

where the last inequality follows from (A.1). Hence $\rho_k \geq \eta_1$. $\square$

Next we show that if the model gradients are bounded from zero, then so is the trust-region radius.

Lemma A.2. *If $\|\nabla m_k(x_k)\| \geq \kappa_1 > 0$ for all k, then there exists $\kappa_2 > 0$ such that:*

$$\Delta_k > \kappa_2 \tag{A.3}$$

for all k.

Proof. (Largely follows [15, Lemma 5.3])

Recall that by the definition of $c_2 > 0$ in (4.3), $\frac{1}{c_2}\|x\|_k \leq \|x\|$ for all $x \in \mathbb{R}^n$ and k. Thus $\frac{\|\nabla m_k(x_k)\|^2}{\|\nabla m_k(x_k)\|_k^2} \geq \frac{1}{c_2^2}$ for all k.

By Lemma A.1 and the assumption that $\|\nabla m_k(x_k)\| \geq \kappa_1 > 0$ for all k, we know that whenever

$$\Delta_k \leq \kappa_1 \min\left\{\frac{1}{\kappa_H}, (1-\eta_1)\frac{\kappa_d}{4\kappa_f}\frac{1}{c_2^2}\right\}, \tag{A.4}$$

$\rho_k \geq \eta_1$ and hence $\Delta_{k+1} \geq \Delta_k$. By virtue of the trust-region radius update (4.22) we thus have that

$$\Delta_k \geq \gamma_0\kappa_1 \min\left\{\frac{1}{\kappa_H}, (1-\eta_1)\frac{\kappa_d}{4\kappa_f}\frac{1}{c_2^2}\right\}, \tag{A.5}$$

for all k, outside of when the criticality step is entered.

Inside the criticality test we have that $\Delta_k \geq \beta\|\nabla m_k(x_k)\| \geq \beta\kappa_1$. Combining these two gives the result with

$$\kappa_2 = \min\left\{\beta\kappa_1, \gamma_0\kappa_1 \min\left\{\frac{1}{\kappa_H}, (1-\eta_1)\frac{\kappa_d}{4\kappa_f}\frac{1}{c_2^2}\right\}\right\}. \tag{A.6}$$

$\square$

The following series of lemmas establishes the convergence of $\nabla f(x_k)$ along certain subsequences.

Lemma A.3. *If the number of successful iterations is finite then*

$$\lim_{k\to\infty} \|\nabla f(x_k)\| = 0. \tag{A.7}$$

Proof. This follows directly from [15, Lemma 5.4]. □

Lemma A.4. *We have that*

$$\lim_{k\to\infty} \Delta_k = 0, \tag{A.8}$$

and hence

$$\liminf_{k\to\infty} \|\nabla m_k(x_k)\| = 0. \tag{A.9}$$

Proof. (Largely follows [15, Lemmas 5.5 and 5.6]) Since Lemma A.3 addresses the other case, we need only consider what happens when $\rho_k \geq \eta_1$ for an infinite number of k. For all such k, the definition of ρ_k in (4.19) and the sufficient decrease condition in (4.24) give:

$$\begin{aligned} f(x_k) - f(x_{k+1}) &\geq \eta_1 \frac{\kappa_d}{2} \|\nabla m_k(x_k)\| \min\left\{ \frac{\|\nabla m_k(x_k)\|}{\kappa_H}, \frac{\|\nabla m_k(x_k)\|}{\|\nabla m_k(x_k)\|_k} \Delta_k \right\} && \text{(A.10)} \\ &\geq \eta_1 \frac{\kappa_d}{2} \min\left\{ \epsilon, \frac{\Delta_k}{\mu} \right\} \min\left\{ \frac{\min\left\{\epsilon, \frac{\Delta_k}{\mu}\right\}}{\kappa_H}, c_2 \Delta_k \right\}, && \text{(A.11)} \end{aligned}$$

where the second inequality follows by observing that $\|\nabla m_k(x_k)\| \geq \min\left\{\epsilon, \frac{\Delta_k}{\mu}\right\}$ due to the criticality step.

Since f is bounded from below, this final expression must converge to zero. Thus $\Delta_k \to 0$ along the subsequence of k for which $\rho_k \geq \eta_1$. Now note that Δ_k can only increase during such iterations and hence any other iteration l must satisfy $\Delta_l \leq \gamma_1 \Delta_k$ for the next iteration k that has $\rho_k \geq \eta_1$. Hence $\Delta_k \to 0$ along all iterations.

The second result then follows immediately by Lemma A.2 by noting that if $\|\nabla m_k(x_k)\| > \kappa_1 > 0$ for all k, then $\Delta_k > \kappa_2 > 0$ for all k. □

Lemma A.5. *For any subsequence $\{k_i\}$ such that*

$$\lim_{i \to \infty} \|\nabla m_{k_i}(x_{k_i})\| = 0, \tag{A.12}$$

it also holds that

$$\lim_{i \to \infty} \|\nabla f(x_{k_i})\| = 0. \tag{A.13}$$

Proof. This follows directly from [15, Lemma 5.7]. □

The final result stated in Theorem 4.2 follows from [15, Theorem 5.9] with the appropriate decrease condition (A.10).

BIBLIOGRAPHY

[1] Bernardetta Addis and Sven Leyffer. A trust-region algorithm for global optimization. *Computational Optimization and Applications*, 35(3):287–304, 2006.

[2] M. Montaz Ali, Charoenchai Khompatraporn, and Zelda B. Zabinsky. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization*, 31(4):635–672, 2005.

[3] Charles Audet and John E. Dennis. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, 2002.

[4] Astrid Battermann, Jörg M. Gablonsky, Alton Patrick, C. Tim Kelley, Kathleen R. Kavanagh, Todd Coffey, and Cass T. Miller. Solution of a groundwater control problem with implicit filtering. *Optimization and Engineering*, 3(2):189–199, 2002.

[5] David Becker, Barbara Minsker, Robert Greenwald, Yan Zhang, Karla Harre, Kathleen Yager, Chunmiao Zheng, and Richard Peralta. Reducing long-term remedial costs by transport modeling optimization. *Ground Water*, 44(6):864–875, November-December 2006.

[6] Michele Benzi, Gene H. Golub, and Jörg Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[7] Martin Berz, Christian Bischof, George Corliss, and Andreas Griewank, editors. *Computational Differentiation: Techniques, Applications and Tools*, Philadelphia, PA, 1996. SIAM.

[8] Mattias Björkman and Kenneth Holmström. Global optimization of costly nonconvex functions using radial basis functions. *Optimization and Engineering*, 1:373 – 397, 2000.

[9] Nikolay Bliznyuk, David Ruppert, Christine A. Shoemaker, Rommel G. Regis, Stefan M. Wild, and Pradeep Mugunthan. Bayesian calibration of computationally expensive models using optimization and radial basis function approximation. *Journal of Computational and Graphical Statistics*, 17(2):1–25, 2008.

[10] Martin D. Buhmann. *Radial Basis Functions: Theory and Implementations.* Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, England, 2003.

[11] Tony D. Choi, Owen J. Eslinger, C. Tim Kelley, J. W. David, and M. Etheridge. Optimization of automotive valve train components with implicit filtering. *Optimization and Engineering*, 1(1):9–27, 2000.

[12] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust-region methods.* MPS/SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[13] Andrew R. Conn, Katya Scheinberg, and Philippe L. Toint. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming*, 79(3):397–414, 1997.

[14] Andrew R. Conn, Katya Scheinberg, and Philippe L. Toint. A derivative free optimization algorithm in practice. In *Proceedings of 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1998.

[15] Andrew R. Conn, Katya Scheinberg, and Luís N. Vicente. Global convergence of general derivative-free trust-region algorithms to first and second order critical points. Technical Report Preprint 06-49, Departamento de Matemática, Universidade de Coimbra, Portugal, 2006.

[16] Andrew R. Conn, Katya Scheinberg, and Luís N. Vicente. Geometry of interpolation sets in derivative free optimization. *Mathematical Programming*, 111:141–172, 2008.

[17] Andrew R. Conn, Katya Scheinberg, and Luís N. Vicente. Geometry of sample sets in derivative free optimization: Polynomial regression and underdetermined interpolation. *IMA Journal of Numerical Analysis*, 2008.

[18] Noel A. Cressie. *Statistics for spatial data.* Wiley, New York, 1993.

[19] Ana Luísa Custódio and Luís N. Vicente. Using sampling and simplex derivatives in pattern search methods. *SIAM Journal on Optimization*, 18(2):537–555, 2007.

[20] John E. Dennis and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* SIAM, Philadelphia, 1996.

[21] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[22] Tobin A. Driscoll and Kara L. Maki. Searching for rare growth factors using multicanonical monte carlo methods. *SIAM Review*, 49(4):673–692, 2007.

[23] Clemens Elster and Arnold Neumaier. A grid algorithm for bound constrained optimization of noisy functions. *IMA Journal of Numerical Analysis*, 15(4):585–608, 1995.

[24] Roger Fletcher. *Practical Methods of Optimization.* John Wiley & Sons, New York, second edition, 1987.

[25] Kathleen R. Fowler, C. Tim Kelley, Cass T. Miller, Chris E. Kees, Robert W. Darwin, Jill P. Reese, Matthew W. Farthing, and Mark S. C. Reed. Solution of a well-field design problem with implicit filtering. *Optimization and Engineering*, 5(2):207–234, 2004.

[26] Kathleen R. Fowler, Jill P. Reese, Chris E. Kees, John E. Dennis, C. Tim Kelley, Cass T. Miller, Charles Audet, Andrew J. Booker, Gilles Couture, Robert W. Darwin, Matthew W. Farthing, Daniel E. Finkel, Jörg M. Gablonsky, Genetha A. Gray, and Tamara G. Kolda. A comparison of derivative-free optimization methods for water supply and hydraulic capture community problems. *Advances in Water Resources*, 31(5):743–757, 2008.

[27] Paul Gilmore and C. Tim Kelley. An implicit filtering algorithm for optimization of functions with many local minima. *SIAM Journal on Optimization*, 5(2):269–285, 1995.

[28] Gene H. Golub and Charles F. Van Loan. *Matrix computations (3rd ed.).* Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[29] Nicholas I. M. Gould, Dominique Orban, and Philippe L. Toint. CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software*, 29(4):373–394, 2003.

[30] Genetha A. Gray and Tamara G. Kolda. Algorithm 856: APPSPACK 4.0: Asynchronous parallel pattern search for derivative-free optimization. *ACM Transactions on Mathematical Software*, 32(3):485–507, 2006.

[31] Genetha A. Gray, Tamara G. Kolda, Ken Sale, and Malin M. Young. Optimizing an empirical scoring function for transmembrane protein structure determination. *INFORMS Journal of Computing*, 16(4):406–418, 2004.

[32] Joshua D. Griffin and Tamara G. Kolda. Nonlinearly-constrained optimization using asynchronous parallel generating set search. Technical Report SAND2007-3257, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, May 2007. Submitted for publication.

[33] Hans-Martin Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19:201–227, 2001.

[34] Nicholas J. Higham. The Matrix Computation Toolbox. `www.ma.man.ac.uk/~higham/mctoolbox`.

[35] Nicholas J. Higham. Optimization by direct search in matrix computations. *SIAM Journal on Matrix Analysis and Applications*, 14(2):317–333, 1993.

[36] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, New York, NY, USA, 1986.

[37] Patricia D. Hough, Tamara G. Kolda, and Virginia J. Torczon. Asynchronous parallel pattern search for nonlinear optimization. *SIAM Journal on Scientific Computing*, 23(1):134–156, 2001.

[38] D. Huang, T. T. Allen, W. I. Notz, and N. Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34(3):441–466, 2006.

[39] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[40] A. H. G. Rinnooy Kan and G. T. Timmer. Stochastic global optimization methods part II: Multilevel methods. *Mathematical Programming*, 39(1):57–78, 1987.

[41] C. Tim Kelley. *Users Guide for imfil version 0.5*. Available at `www4.ncsu.edu/~ctk/imfil.html`.

[42] Tamara G. Kolda. Revisiting asynchronous parallel pattern search for nonlinear optimization. *SIAM Journal on Optimization*, 16(2):563–586, 2005.

[43] Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. Stationarity results for generating set search for linearly constrained optimization. *SIAM Journal on Optimization*, 17(4):943–968, 2006.

[44] Tamara G. Kolda, Robert Michael Lewis, and Virginia J. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.

[45] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the Nelder-Mead simplex algorithm in low dimensions. *SIAM Journal on Optimization*, 9:112–147, 1998.

[46] 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 and Ecosystems Modeling*, 1:91–122, 2000.

[47] Marco Locatelli. Relaxing the assumptions of the multilevel singlelinkage algorithm. *Journal of Global Optimization*, 13(1):25–42, 1998.

[48] Marcelo Marazzi and Jorge Nocedal. Wedge trust region methods for derivative free optimization. *Mathematical Programming*, 91(2):289 – 305, 2002.

[49] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[50] Jorge J. Moré, Burton S. Garbow, and Kenneth E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7(1):17–41, 1981.

[51] Jorge J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983.

[52] Jorge J. Moré and Stefan M. Wild. Benchmarking derivative-free optimization algorithms. Technical Report ANL/MCS-P1471-1207, Argonne National Laboratory, MCS Division, 2007.

[53] Pradeep Mugunthan, Christine A. Shoemaker, and Rommel G. Regis. Comparison of function approximation, heuristic and derivative-based methods for automatic calibration of computationally expensive groundwater bioremediation models. *Water Resources Research*, 41, 2005.

[54] John A. Nelder and Roger Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[55] Rodrigue Oeuvray. *Trust-Region Methods Based on Radial Basis Functions with Application to Biomedical Imaging.* PhD thesis, EPFL, Lausanne, Switzerland, 2005.

[56] Rodrigue Oeuvray and Michel Bierlaire. A new derivative-free algorithm for the medical image registration problem. *International Journal of Modelling and Simulation*, 27(2):115–124, 2007.

[57] Rodrigue Oeuvray and Michel Bierlaire. Boosters: a derivative-free algorithm based on radial basis functions. Technical report, EPFL, 2008. Accepted in: International Journal of Modelling and Simulation, 2008.

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

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

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

[61] Michael J.D. Powell. The NEWUOA software for unconstrained optimization without derivatives. In G. Di Pillo and M. Roma, editors, *Large-Scale Nonlinear Optimization*, pages 255–297. Springer, 2006.

[62] Michael J.D. Powell. Developments of NEWUOA for minimization without derivatives. *IMA Journal of Numerical Analysis*, page drm047, 2008.

[63] Rommel G. Regis and Christine A. Shoemaker. Improved strategies for radial basis function methods for global optimization. *Journal of Global Optimization*, 37(1):113–135, 2007.

[64] Rommel G. Regis and Christine A. Shoemaker. Parallel radial basis function methods for the global optimization of expensive functions. *European Journal of Operational Research*, 127(2):514–535, October 2007.

[65] Rommel G. Regis and Christine A. Shoemaker. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal of Computing*, 19:457–509, 2007.

[66] S.M. Sait and H. Youssef. *Iterative Computer Algorithms with Applications in Engineering*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1999.

[67] Fabio Schoen. *Handbook of global optimization*, volume 2, chapter Two-phase methods for global optimization, pages 151–178. Kluwer Academic Publishers, Dordrecht, 2002.

[68] The Mathworks, Inc. *Optimization Toolbox for Use with MATLAB: User's Guide, V. 3*, 2004.

[69] Bryan A. Tolson and Christine A. Shoemaker. Cannonsville reservoir watershed swat2000 model development, calibration and validation. *Journal of Hydrology*, 337:68–86, 2007.

[70] Aimo Törn and Antanas Zilinskas. *Global optimization*. Springer-Verlag New York, Inc., New York, NY, USA, 1989.

[71] Lloyd N. Trefethen and III David Bau. *Numerical Linear Algebra*. SIAM, Philadelphia, 1997.

[72] U.S. Geological Survey. MODFLOW 2000, 2000. See `http://water.usgs.gov/nrp/gwsoftware/modflow2000/modflow2000.html`.

[73] Holger Wendland. *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, England, 2005.

[74] Stefan M. Wild. MNH: a derivative-free optimization algorithm using minimal norm hessians. In *Tenth Copper Mountain Conference On Iterative Methods*, April 2008. Available at `http://grandmaster.colorado.edu/~copper/2008/SCWinners/Wild.pdf`.

[75] Stefan M. Wild, Rommel G. Regis, and Christine A. Shoemaker. ORBIT: Optimization by radial basis function interpolation in trust-regions. *To appear in SIAM Journal on Scientific Computing*, 2008.

[76] Stefan M. Wild and Christine A. Shoemaker. Global convergence of radial basis function trust-region algorithms. Technical Report ORIE1470, Cornell University, School of ORIE, 2008.

[77] D.H. Winfield. Function minimization by interpolation in a data table. *Journal of the Institute of Mathematics and its Applications*, 12:339–347, 1973.

[78] Jae-Heung Yoon and Christine A. Shoemaker. Comparison of optimization methods for ground-water bioremediation. *Journal of Water Resources Planning and Management*, 125(1):54–63, 1999.

[79] Yan Zhang, Robert Greenwald, Barbara Minsker, Richard Peralta, Chunmiao Zheng, Karla Harre, Dave Becker, Laura Yeh, and Kathy Yager. Final cost and performance report application of flow and transport optimization codes to groundwater pump and treat systems. Technical Report TR-2238-ENV, Naval Facilities Engineering Service Center, Port Hueneme, California, January 2004.

[80] Chunmiao Zheng and P. Patrick Wang. MT3DMS, a modular three-dimensional multi-species transport model for simulation of advection, dispersion and chemical reactions of contaminants in groundwater systems – documentation and users guide. Technical Report Contract Report SERDP-99-1, U.S. Army Engineer Research and Development Center, 1999.