Optimization in SciDAC applications

Jorge J. Moré, Todd S. Munson, and Jason Sarich
Mathematics and Computer Science Division
Argonne National Laboratory, Argonne, Illinois 60439

E-mail: more@mcs.anl.gov,tmunson@mcs.anl.gov,sarich@mcs.anl.gov

Abstract. We present a brief overview of optimization tools that are being developed for SciDAC applications. We emphasize derivative-free and gradient-based methods since these tools make minimal demands on the user and the application. We discuss the performance of these tools and point out developments that have led to significant improvements in performance. A parameter estimation problem that arises in nuclear fission is used to illustrate the challenges that arise as we attack nonlinear, noisy, computationally-intensive optimization applications.

1. Introduction

The design of engineering systems and the simulation of scientific phenomena of DOE interest invariably require the optimization of a mathematical model with respect to design parameters and constraints. At present, however, there is a lack of optimization tools for high-performance, parallel architectures, and thus scientists are severely limited in their ability to study and solve complex simulation problems.

To address the need for optimization tools on high-performance, parallel architectures, we have been developing TAO [4, 18] (Toolkit for Advanced Optimization). In this note we discuss and emphasize derivative-free and gradient-based methods because these tools make minimal demands on the user and the application. We discuss the performance of these tools and point out developments that have led to significant improvements in performance.

We start by mentioning the wide range of optimization problems being addressed by members of the TOPS Center for Enabling Technology. We then illustrate the challenges that arise as we attack nonlinear, noisy, computationally intensive optimization applications by discussing a parameter estimation problem that arises in nuclear fission. We end by discussing the scalability of gradient-based optimization methods on mesh-based applications. In particular, we show that the number of function and gradient evaluations required to solve a problem to a given accuracy scales linearly with the number of variables.

2. Background

The mission of the TOPS Center for Enabling Technology is to enable scientists and engineers to take full advantage of petascale hardware by overcoming the scalability bottlenecks that traditional solvers impose, and assist them to move beyond one-off simulations to validation and optimization As part of this mission, TOPS has been developing and maintaining a collection

of toolkits for scientific computing. These toolkits (Hypre, PETSc, SUNDIALS, SuperLU, TAO) address most of the important areas in scientific computing.

At present, TOPS is working on a range of optimization problems (listed in Figure 2.1) in SciDAC applications. We focus on a nonlinear estimation problem in nuclear physics to illustrate some of the challenges we face as we start to work on petaflop applications.

- ◇ Nuclear Physics
  - Nonlinear eigenvalues
  - Parameter estimation
  - Least action pathways
- ◇ Groundwater Flow
  - Parameter estimation
- ◇ Quantum Chemistry
  - Energy minimization
  - Transition states
- ◇ Accelerator Design
  - Shape optimization
  - Nonlinear eigenvalues

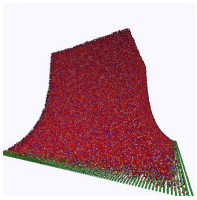
Figure 2.1. Optimization problems in SciDAC applications

TAO [4, 18] contains sophisticated optimization tools for SciDAC applications. TAO uses object-oriented techniques to leverage the parallel linear algebra infrastructure (parallel sparse matrix data structures, preconditioners, solvers) offered by PETSc [14]. Other optimization toolkits have employed object-oriented design, but their work does not address the reuse of linear algebra toolkits and is restricted to uniprocessor environments.

TAO is available as open-source software, and has been tested on architectures ranging from laptops and high-end workstations to high-performance clusters like the IBM BG/L system. TAO is also being incorporated into other toolkits for computational science problems. We are aware of three major toolkits that use TAO:

- ◇ TADM, a toolkit for estimating the parameters of discriminative models,
- ◇ BUSTER, a toolkit for the X-ray crystallographic determination of protein structures, and
- ◇ ELEFANT, a software framework for statistical machine learning.

The connection of TAO to statistical machine learning is of special importance, since this area is likely to be of increasing importance in SciDAC applications.



Courtesy P. Bauman

Courtesy P. Joshi

Figure 2.2. TAO applications: Semiconductor modeling (left) and variational surfaces (right)

TAO solvers are having an impact on the computational science community with applications in a wide variety of areas, including semiconductor modelling, magnetic nanostructures, subsurface remediation, and variational surfaces. Figure 2.2 shows images associated with semiconductor modeling and variational surfaces. The image on the left is a cube of polymer material with colored particles representing different monomer molecules. The bonds between the particles are not visualized, but these bonds cause the deformation of the material. This

image is the result of preliminary parallel calculations that are being used as a basis for comparison of multiscale models where particle models are coupled to continuum models (in this case nonlinear elastic continuua). Additional details on most of these applications can be found in the TAO site [18].

The focus of our research has been on large-scale problems that require high-performance architectures. In particular, we have been working with computational chemists at both Pacific Northwest National Laboratory and Sandia National Laboratories on optimization algorithms for molecular geometry optimization. As a result of this collaboration TAO can be used as a component in `NWChem` [13], a large (2.5 million lines of code) software suite that encompasses multiple theories, algorithms, and methods in computational chemistry, and we have shown [10] reductions in simulation times of more that 40% compared to the stand-alone chemistry packages. This is a significant reduction because large molecular geometry simulations take hours on massively parallel architectures.

3. Parameter Estimation Problems in Nuclear Fission

The UNEDF project plans to use optimization techniques to determine parameters in the energy density functional to fit experimental data . As we shall see, problems of interest require massive amounts of computing that can only be provided by petascale architectures.

We emphasize parameter estimation problems associated with the HFODD code [7], which solves the nuclear Skyrme-Hartree-Fock or Skyrme-Hartree-Fock-Bogolyubov problem by using the Cartesian deformed harmonic-oscillator basis.

Assume that $\xi_1, \ldots, \xi_n$ are the parameters that need to be determined in the optimization procedure. We gather all these parameters into a vector $x \in \mathbb{R}^n$, where $\mathbb{R}^n$ is n-dimensional Euclidean space.

The output of HFODD for the kth nucleus is a set of observables $f_k(x)$. The number of observables depends on the nucleus. The following diagram formalizes this relationship.

$$x \quad \Longrightarrow \quad \boxed{\text{HFODD}} \quad \Longrightarrow \quad f_k(x)$$

In all cases a data vector y_k is associated with this nucleus, and we would like to determine the vector of parameters x such that $f_k(x)$ is close to y_k. In the least-squares (or χ^2) approach the optimization procedure seeks to minimize

$$f(x) = \sum_{k=1}^{m} \sigma_k \|f_k(x) - y_k\|^2, \tag{3.1}$$

where $\|\cdot\|$ is the Euclidean norm and σ_k is a set of weights, but other approaches are possible. Determining a vector of parameters x that minimizes f, where f is defined by (3.1), is a standard optimization problem. Since each evaluation of f is expensive, however, significant computational challenges arise:

- ⋄ Expensive evaluations of $f_k(x)$ ($U_{236} \approx 12$ hours)
- ⋄ Large memory requirements ($U_{236} \approx 0.5$ GB)
- ⋄ Many nuclei (about 2,000)
- ⋄ A wide range of observables (binding energy, ...)
- ⋄ Noisy function evaluations
- ⋄ Lack of derivatives with respect to parameters
- ⋄ Several minima with different predictive power

In particular, for this problem the evaluation of f at a trial x may require 1,000 CPU days (2000×12 hours). Hence, we cannot afford many function evaluations, even on petascale architectures.

What are the best optimization techniques for these nonlinear, noisy, computationally intensive problems? We review the state of the art in the next section, but we want to end this section by examining the performance of HFODD in more detail.

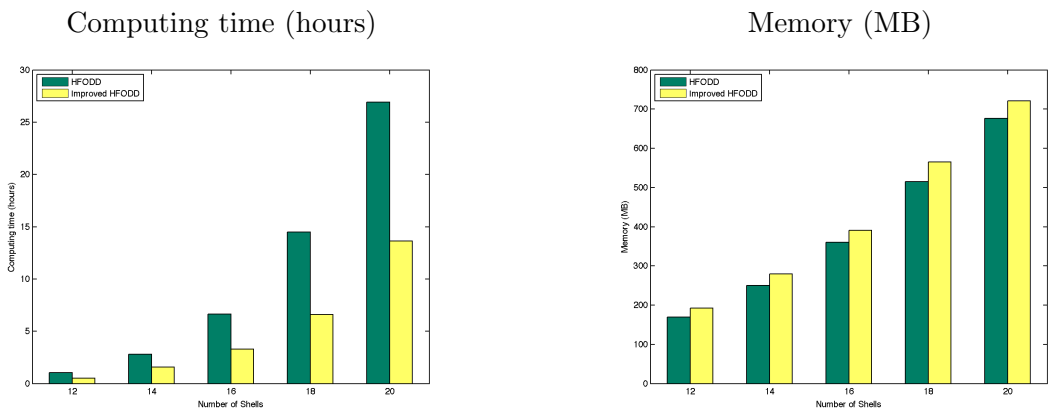


Figure 3.1. HFODD performance with U_{236}

We were able to restructure the most intensive computations in HFODD to take advantage of level-2 and level-3 BLAS routines, and as a result we obtained substantial decreases in computation time with only a minor increase in memory usage. This is ongoing work, but the data in Figure 3.1 (obtained by Carlos Bertulani) already shows a decrease of at least 45% in computing time, with about a 10% increase in memory usage. Any time reduction is particularly significant because almost all of the time needed to solve the parameter estimation problem (3.1) will be spent in HFODD calculations.

4. Noisy Bound-Constrained Optimization

The parameter estimation problem defined by the function (3.1), where f_k requires the evaluation of a computationally-intensive code (HFODD) is an example of a nonlinear, noisy optimization problem. In general there may be bounds on the parameters that we need to determine, and thus we consider the general bound-constrained optimization problem

$$\min \{ f(x) : x_L \leq x \leq x_U \} ,$$

where the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is nonlinear and noisy.

Algorithms for dealing with these noisy, nonlinear optimization problems are usually called black-box methods because no assumptions are made on the code that is needed to evaluate f. The most promising approaches for dealing with such problems fall under the general category of derivative-free methods. These methods require the user to provide only function evaluations and thus are easy to use. However, each evaluation of f is usually costly in terms of computing time, and thus methods are needed that require a satisfactory level of accuracy with relatively few function evaluations.

Our review of derivative-free methods shows serious shortcomings with most of the current techniques. The techniques that have received the most attention in the optimization community

are *geometry-based* methods. The main reason for this interest is that there is a general convergence theory for many (pattern search [19], generating-set search [11], mesh-adaptive [1]) of these methods. However, geometry-based methods rely on a comparison of the current function value f with previous values to determine algorithmic behavior at the current trial value. Hence, these methods do not tend to model a function accurately and are likely to require a large number of function evaluations to reach a satisfactory level of accuracy. More sophisticated techniques are needed that can obtain a satisfactory level of accuracy with significantly fewer function evaluations.

Model-based methods for derivative-free optimization construct a model of the optimization problem based on function values at previous iterates and then use that model to generate the next iterate. These models, usually called response surface models, are approximations to the function that are considerably easier to evaluate than the original function. The traditional response surface literature [12] constructs the response surface as a least-squares approximation of low-order polynomials, but tensor-products of polynomials and splines are also used.

A model-based optimization method evaluates the function f at points $x_1, \ldots, x_k$ in $\mathbb{R}^n$ and constructs a model of the function f based on this data. The next point x_{k+1} is then determined by minimizing the model subject to additional constraints on the final location of x_{k+1}. In our experience model-based methods that use polynomial interpolation (Powell [15, 16] and Conn, Scheinberg, and Toint [5, 6]) have the best performance among direct search methods.

We illustrate these ideas with the model-based methods of Conn, Scheinberg, and Toint [5, 6]. These methods use quadratic models, that is, functions of the form

$$q(x) = c + g^T(x - x_0) + \tfrac{1}{2}(x - x_0)^T G(x - x_0),$$

where $g \in \mathbb{R}^n$ is the gradient of the quadratic q at the base point x_0 and $G \in \mathbb{R}^{n\times n}$ is the Hessian matrix of the quadratic. This approach requires evaluation of the function at $\frac{1}{2}(n+1)(n+2)$ evaluations of f, since this is the number of evaluations needed to determine all the terms in the quadratic. Unfortunately, this implies that the required number of evaluations grows quickly. For example, if $n = 10$, then the quadratic model requires 66 evaluations. This is prohibitive for the kind of computationally intensive optimization problems that we are considering.

In joint work with C. Shoemaker and S. Wild we have been exploring minimal norm quadratics as defined by Powell [16]. Given m points $x_1, \ldots, x_m$, we determine a minimal norm quadratic q by requiring interpolation of the function at these points, and by requiring that the Hessian matrix G of q satisfy

$$\min \left\{ \|G - G_0\|_F^2 : q(x_k) = f(x_k), \quad 1 \le k \le m \right\},$$

where G_0 is the Hessian matrix of a previous model. If the points $x_1, \ldots, x_m$ satisfy a mild geometric restriction, then this minimal change quadratic exists and is unique. Moreover, the quadratic is well defined for any number of points m with $m \ge n+1$. Clearly, if $m = n+1$, then $G = G_0$, so that the Hessian matrix of the new quadratic is unchanged, while if $m > n+1$, we can reasonably expect an improved Hessian matrix.

Preliminary experience with these ideas is encouraging. Below we illustrate the performance of both geometry-based and model-based methods on a watershed model. Of course, performance on one problem is not indicative of performance in general, but in our view, the results in Figure 4.1 are representative. More testing will be needed to establish the validity of this claim.

The watershed model is a nonlinear, noisy parameter estimation problem with several local minimizers and 14 variables. The results in Figure 4.1 show the value of the fit $f(x_k)$ as a function of the trial value x_k for four methods. Two geometry-based methods are illustrated: the pattern-search method is shown in black, and the Nelder-Mead method is shown in cyan.

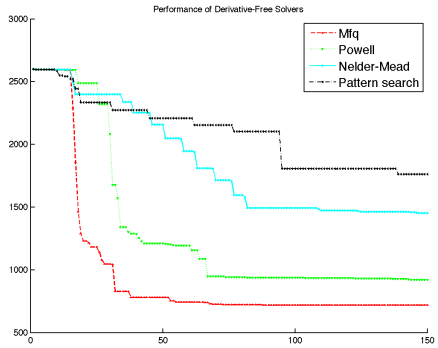


Figure 4.1. Performance of derivative-free methods on the watershed model

We are using the implementations available in MATLAB. Figure 4.1 also shows two model-based methods: the Powell [17] code is shown in green, and our implementation of an optimization method that uses minimal norm quadratics is shown in red.

Figure 4.1 illustrates various properties of derivative-free methods. Note, for example, that both the pattern-search method and the Nelder-Mead method have long stretches where the function value remains constant. This behavior can be expected in general, since geometry-based methods do not take into account the relative values of the function and thus are slower at adapting to nonlinear behavior. On the other hand, the model-based methods are able to approximate the function and thus are likely to achieve a reduction in the function at most iterations.

5. Gradient-Based Optimization Methods

In many nonlinear optimization problems, the user is able to provide the gradient ∇f of the function (even if the problem is noisy, although in such cases we would not expect a highly accurate gradient). We thus consider the bound-constrained optimization problem

$$\min \{f(x) : x_L \leq x \leq x_U\} \tag{5.1}$$

under the assumption that the gradient ∇f is available.

A survey of the optimization literature shows that the most common methods for attacking these problems are nonlinear conjugate gradient methods and that, although there are many variations on these methods, the most popular and efficient methods are the following:

- ◇ Fletcher-Reeves (FR)
- ◇ Polak-Rivière (PR)
- ◇ Limited-memory variable-metric methods (LMVM)

FR and PR methods are well known to application scientists (see, for example, *Numerical Recipes*), but LMVM methods are relatively unknown to many application scientists.

An extensive discussion of LMVM is not appropriate here, so we just note that these methods determine an (inverse) metric H_k from information gathered during the previous m iterations such that

- ◇ H_k is positive definite,
- ◇ storage of H_k requires $2mn$ locations, and
- ◇ computation of $H_k \nabla f(x_k)$ costs $(8m + 1)n$ flops.

Given the matrix H_k, the next iterate is obtained via

$$x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k),$$

with α_k determined by a line search. The user is free to set the number m of extra vectors to use, but in our experience, a relatively small number of vectors ($m = 5$) provides the best performance.

The main claim that we make in this section is that LMVM methods vastly outperform FR and PR methods if we use the number of evaluations of $(f, \nabla f)$ as the metric. We use performance profiles [8] to compare the TAO implementations of these methods on a set of nonlinear variational problems with dimensions ranging from $2,500$ to $40,000$. The results are shown in Figure 5.1.

Given a set of problems $\mathcal{P}$ and a set of solvers $\mathcal{S}$, a performance profile is a plot of the probability distribution function

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

where $|\mathcal{P}|$ is the number of problems, $r_{p,s}$ is the performance ratio

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

and $\mu_{p,s}$ is the number of evaluations of $(f, \nabla f)$ required for convergence. Clearly, the fastest solver has $r_{p,s} = 1$, and the solver fails if $r_{p,s} = \infty$.

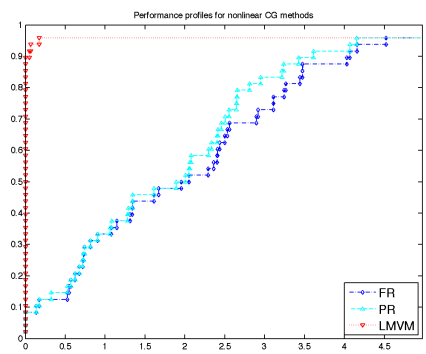


Figure 5.1. Performance profiles for the TAO methods (logarithmic scale)

Figure 5.1 shows that LMVM is the fastest solver on about 95% of the problems. Additional information can be obtained by examining the difference in performance between these methods for other values of the performance ratio τ. For example, Figure 5.1 also shows that the FR and PR methods are at least twice as slow as LMVM on about 60% of the problems. Moreover, the FR and PR methods are at least four times as slow as LMVM on about half of the problems.

6. Scalability on Mesh-Based Optimization Problems

We now examine how LMVM methods perform on mesh-based optimization problems that arise as discretizations of infinite-dimensional problems of the form

$$\min \{ f(v) : v_L \leq v \leq v_U \}, \tag{6.1}$$

where $v : \mathbb{R}^p \mapsto \mathbb{R}^q$ is a function defined on some domain $\mathcal{D} \subset \mathbb{R}^p$, and v_L and v_U are, respectively, the lower and upper (pointwise) bounds on v. In these cases, discretization of the domain $\mathcal{D}$ leads to a large optimization problem of the form (5.1) where the variables x represent the values of v at the mesh points of the discretization of $\mathcal{D}$. If there are m mesh points, then the number of variables is $n = mq$. We use the term *mesh-based* optimization problem to refer to problems of the form (6.1) that require determination of a function v on a mesh.

An interesting mesh-based variational problem is that of minimizing the Gibbs free energy for a homogeneous superconductor with a vector potential perpendicular to the superconductor. This (Ginzburg-Landau) problem requires the minimization of the functional

$$\int_{\mathcal{D}} \left\{ -|v(x)|^2 + \tfrac{1}{2}|v(x)|^4 + \|[\nabla - iA(x)]\, v(x)\|^2 + \kappa^2 \, \|(\nabla \times A)(x)\|^2 \right\} dx$$

where $v : \mathbb{R}^2 \to \mathbb{C}$ is the order parameter and $A : \mathbb{R}^2 \to \mathbb{R}^2$ is the vector potential. Figure 6.1 is a plot of the magnitude $|v|$ of the order parameter. This problem has four variables per mesh point, two variables for the real and complex parts of the order parameter, and two variables for the two components of the vector potential. Hence, this problem has $4n$ variables for a mesh with n mesh points.

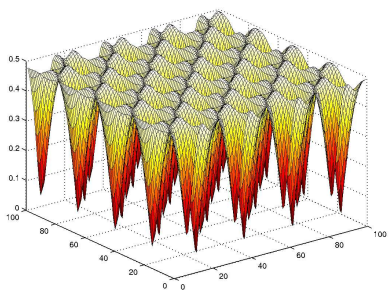


Figure 6.1. Order potential for the Ginzburg-Landau problem

We explore *scalable* algorithms for mesh-based optimization problems in the sense that the number of evaluations of $(f, \nabla f)$ required to solve the problem grows linearly with the number of variables n. Note that LMVM methods may require more than one evaluation of $(f, \nabla f)$ at each iteration because the line search may not find an acceptable point at the first trial point.

The advantage of using evaluations of $(f, \nabla f)$ as a measure of scalability is that this measure is independent of the architecture; that is, we obtain the same results on both workstations and clusters. We also require that the computing time grow linearly with n; for LMVM methods this follows if the evaluation of f and ∇f is linear with n.

Results for several benchmark problems [2] are shown in Figure 6.2. These results show that the number of operations grow linearly with n. Surprisingly, the number of evaluations of $(f, \nabla f)$ also grow linearly with n. We expect this kind of behavior from Newton's method because there are mesh independence results for Newton's method [9], but we are not aware of mesh independence results for gradient-based optimization methods.

For these results we started with a coarse mesh with $5^2 = 25$ mesh points and refined the mesh by doubling the number of grid points in each direction. The finest mesh had $1535^2 \approx 2.3\ 10^6$ mesh points.

The results in Figure 6.2 were obtained by using mesh sequencing; that is, an interpolated coarse mesh solution was used as the initial starting point for a finer mesh. This is a standard technique for solving systems of nonlinear (partial differential) equations, but with few exceptions it is not used for solving mesh-based optimization problems. See Bank, Gill and Marcia [3] for a recent exception.

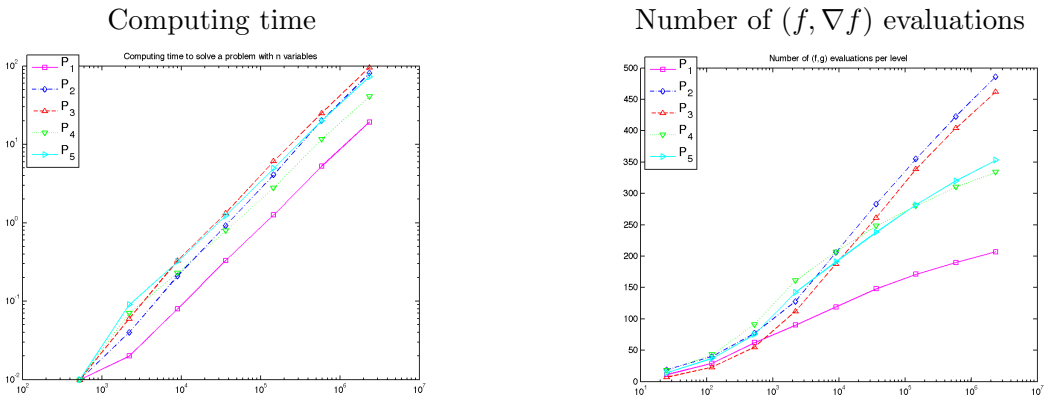


Figure 6.2. Scalability of limited-memory variable-metric methods

The termination criteria at each level could be defined in terms of the gradient ∇f, but we prefer a termination test in terms of the function value. Thus, given a tolerance $\tau_f > 0$ (in our results we used $\tau_f = 10^{-8}$), we terminate the LMVM algorithm if

$$|f(x_k + \alpha_k p_k) - f(x_k)| \leq \tau_f |f(x_k)|, \qquad \alpha_k |\nabla f(x_k)^T p_k| \leq \tau_f |f(x_k)|.$$

This test guarantees that the function value has achieved a reasonable level of accuracy on each level. Note, in particular, that this test is not sensitive to the mesh size in the sense because the function value will not change significantly as the mesh size is decreased.

Acknowledgments

The description of the parameter estimation problem in Section 3 is based on discussions with Witek Nazarewicz, who also supplied references and pointers to the literature in nuclear physics. We are also grateful to Jacek Dobaczewski, the main author of HFODD, for numerous discussions on various aspect of this code. Carlos Bertulani deserves special thanks for providing the numerical results comparing the two versions of HFODD; he has also been the source of much additional information. We are all part of the UNEDF SciDAC project, and we hope that the ideas presented in this manuscript will prove useful in our ongoing collaboration.

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.

References

[1] M. A. Abramson, C. Audet, and J. E. Dennis, *Nonlinear programming by mesh adaptive direct searches*, Report TR05-13, Rice University, Texas, 2005.

[2] B. M. Averick, R. G. Carter, J. J. Moré, and G.-L. Xue, *The MINPACK-2 test problem collection*, Preprint MCS-P153-0694, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 1994.

[3] R. E. Bank, P. E. Gill, and R. F. Marcia, *Interior point methods for a class of elliptic variational inequalities*, in High Performance Algorithms and Software for Nonlinear Optimization, L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. Van Bloemen Waanders, eds., Springer-Verlag, 2003, pp. 218–235.

[4] S. Benson, L. C. McInnes, J. J. Moré, and J. Sarich, *TAO Users Manual*, Technical Memorandum ANL/MCS-TM-242 (Revision 1.4), Argonne National Laboratory, Argonne, Illinois, 2002.

[5] A. R. Conn, K. Scheinberg, and P. L. Toint, *On the convergence of derivative-free methods for unconstrained optimization*, in Approximation Theory and Optimization, M. D. Buhmann and A. Iserles, eds., Cambridge University Press, 1997, pp. 83–108.

[6] ———, *Recent progress in unconstrained nonlinear optimization without derivatives*, Math. Programming, 79 (1997), pp. 397–415.

[7] J. Dobaczewski and P. Olbratowski, *HFODD (version 2.08k)*, Comput. Phys. Commun., 167 (2005), pp. 214–216.

[8] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Programming, 91 (2002), pp. 201–213.

[9] A. L. Dontchev, W. W. Hager, and V. M. Veliov, *Uniform convergence and mesh independence of Newton's method for discretized variational problems*, SIAM J. Control Optim., 39 (2000), pp. 961–980.

[10] J. P. Kenny, S. J. Benson, Y. Alexeev, J. Sarich, C. L. Janssen, L. C. McInnes, M. Krishnan, J. Nieplocha, E. Jurrus, C. Fahlstrom, and T. L. Windus, *Component-based integration of chemistry and optimization software*, J. Computational Chemistry, 25 (2004), pp. 1717–1725.

[11] T. G. Kolda, R. M. Lewis, and V. Torczon, *Optimization by direct search: New perspectives on some classical and modern methods*, SIAM Rev., 45 (2000), pp. 385–482.

[12] R. H. Myers and D. C. Montgomery, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, John Wiley & Sons, 2002.

[13] NWChem, *High-performance computational chemistry software.* See `www.emsl.pnl.gov/docs/nwchem/`.

[14] PETSc, *Portable Extensible Toolkit for Scientific Computation.* See `www.mcs.anl.gov/petsc`.

[15] M. J. D. Powell, *UOBYQA: Unconstrained optimization by quadratic approximation*, Math. Programming, 92 (2002), pp. 555–582.

[16] ———, *Least Frobenius norm updating of quadratic models that satisfy interpolation conditions*, Math. Programming, 100 (2004), pp. 183–215.

[17] ———, *The NEWUOA software for unconstrained optimization without derivatives*, in Large Scale Nonlinear Optimization, G. Di Pillo and M. Roma, eds., Springer, Netherlands, 2006, pp. 255–297.

[18] TAO, *Toolkit for Advanced Optimization.* See `www.mcs.anl.gov/tao`.

[19] V. Torczon, *On the convergence of pattern search methods*, SIAM J. Optim., 7 (1997), pp. 1–25.