ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

Solving Derivative-Free Nonlinear Least Squares Problems with POUNDERS[1]

Stefan M. Wild

Mathematics and Computer Science Division

Preprint ANL/MCS-P5120-0414

April 2014 *(Revised June 2015)*

[1]This work was supported by the applied mathematics and SciDAC activities within the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research program under contract number DE-AC02-06CH11357.

Chapter 1

Solving Derivative-Free Nonlinear Least Squares with POUNDERS

Much of the research and software tools in derivative-free optimization focus on *blackbox optimization problems*. These are problems where the objective is effectively a blackbox function, such as the scalar-valued output of an executable-only code, of the inputs. In practice, however, one often has additional knowledge (e.g., sparsity structures, partial separability, nonlinearity, convexity, form of nonsmoothness) about the problem, while still not having access to complete derivative information. This knowledge, which we characterize as defining a *greybox optimization problem*, can be exploited to reduce the solve time and/or obtain more accurate solutions. On the other hand, when designing software for derivative-free optimization, one must balance the customization needed to exploit this type of information with the ease of use for a nonexpert user. Achieving this balance is especially challenging in derivative-free optimization because of the size and diversity of the user pool, which includes many application scientists and engineers. For these nonexperts, derivative-free optimization is often a gateway to algorithmic differentiation, nonlinear programming, and other areas of structure-exploiting optimization.

Here we address nonlinear least squares, a particular form of structural information that occurs frequently in derivative-free optimization and that requires minimal input from a nonexpert. Formally, we seek local solutions to

$$\min \left\{ f(x) = \frac{1}{2}\|F(x)\|_2^2 = \frac{1}{2}\sum_{i=1}^{p} F_i(x)^2 : x \in \Omega \subseteq \mathbb{R}^n \right\}, \tag{1.1}$$

where $F : \mathbb{R}^n \to \mathbb{R}^p$ defines the system of nonlinear equations/residuals and the Jacobian $\nabla_x F(x)$ is unavailable. We focus our discussion here on bound-constrained problems, where $\Omega = \{x \in \mathbb{R}^n : l_i \le x_i \le u_i, i = 1, \ldots, n\}$ and the bounds $l_i < u_i$ are not necessarily finite.

These problems arise frequently in settings such as nonintrusive model calibration, where one has data d_i at design sites θ^i and wishes to estimate the parameters x of a nonlinear model $S(\theta; x)$ that best fit the data. A maximum likelihood approach directly yields (1.1), with $F_i(x) = S(\theta^i; x) - d_i$, while many Bayesian approaches can be captured through weighted residuals $F_i(x) = \frac{1}{w_i}\left(S(\theta^i; x) - d_i\right)$.

Other examples of algebraic structures that have been exploited in derivative-free optimization include partial separability [5], nonsmoothness of constraints [9], and bilevel problems [7]. The algorithm we describe here is called POUNDERS (Practical Optimization Using No Derivatives for sums of Squares). Other derivative-free approaches to least-squares problems include Implicit Filtering [10, 11] and DFLS [20, 21], both of which are described later, and LMDIF [15], which is a finite-difference-based implementation of the Levenberg-Marquardt method. As with these three methods, the usual caveats apply with POUNDERS. In particular, we are seeking local solutions to a potentially multimodal problem (1.1), and convergence theory (which is not our focus here) is generally limited to problems with sufficiently smooth residuals (despite the derivatives of these residuals being unavailable).

This chapter is organized as follows:

- § 1.1 introduces the interpolation-based models employed;
- § 1.2 describes the basic algorithm;
- § 1.3 details the implementation in TAO;
- § 1.4 describes numerical results on physics calibration problems.

1.1 Smooth Residual Models

The main device that POUNDERS uses to exploit the known structure of (1.1) is a collection of smooth surrogate models, one for each residual F_i. We briefly review the general use of models in derivative-free optimization and introduce our approach for exploiting the known structure.

1.1.1 Quadratic interpolation models

Many forms of models have been employed for model-based optimization, from classical polynomials [6] to radial basis functions [19] to sparse polynomials [2]. Here we focus on quadratic models

$$q_k(x) = c + (x - x^k)^\top g + \frac{1}{2}(x - x^k)^\top H (x - x^k), \tag{1.2}$$

where we have intentionally centered these models around a point $x^k \in \Omega$. The model q_k is defined by the $\frac{(n+1)(n+2)}{2}$ parameters $c \in \mathbb{R}$, $g \in \mathbb{R}^n$, $H = H^\top \in \mathbb{R}^{n \times n}$, where we have dropped the explicit dependence of these parameters on x^k.

Given a set $\mathcal{Y} = \left\{y^1, \cdots, y^{|\mathcal{Y}|}\right\} \subseteq \Omega$ and corresponding function values $\{f(y^i) : y^i \in \mathcal{Y}\}$, one can demand that the quadratic model interpolate the function f on $\mathcal{Y}$ by determining parameters (c, g, H) such that

$$q_k(y^i) = f(y^i), \quad i = 1, \ldots, |\mathcal{Y}|. \tag{1.3}$$

If the point x^k belongs to the interpolation set $\mathcal{Y}$, one can easily show that $c = f(x^k)$.

For a basis ϕ for quadratic functions on $\mathbb{R}^n$ (for example, the monomial basis $\left\{1, x_1, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_n^2\right\}$), satisfying the interpolation conditions (1.3) is equivalent to satisfying the linear system

$$\Phi(\mathcal{Y})z = \begin{bmatrix} \phi(y^1) \\ \vdots \\ \phi(y^{|\mathcal{Y}|}) \end{bmatrix} z = \begin{bmatrix} f(y^1) \\ \vdots \\ f(y^{|\mathcal{Y}|}) \end{bmatrix} = f(\mathcal{Y}), \tag{1.4}$$

where the solution z defines the model parameters (c, g, H). Whether this system has a unique solution for arbitrary function values $f(\mathcal{Y})$ depends solely on the interpolation set $\mathcal{Y}$; in particular we note that, unlike when doing univariate interpolation, having $\frac{(n+1)(n+2)}{2}$ distinct points in $\mathcal{Y}$ is not a sufficient condition for a unique solution to (1.4) when $n \geq 2$. The conditioning of this system clearly depends on the basis employed, and most methods take this into consideration when selecting particular forms of $\mathcal{Y}$ and/or ϕ. Measures of the quality of the sample set $\mathcal{Y}$ and the approximation properties of the resulting model are discussed in [6]. Here a primary concern is the ability of such models to approximate a function in a neighborhood of x^k; doing so requires that the sample points be within a certain proximity of x^k.

Interpolation is not the only form of model that one could consider. Other forms just correspond to different ways of "solving" (1.4). For example, for overdetermined ($|\mathcal{Y}| > \frac{(n+1)(n+2)}{2}$) sample sets one could obtain a regression-based quadratic by minimizing $\|\Phi(\mathcal{Y})z - f(\mathcal{Y})\|$, and for underdetermined sample sets one could find the interpolating quadratic for which $\|z\|$ is minimized.

For expensive problems, we generally find that we have fewer than $\frac{(n+1)(n+2)}{2}$ nearby points. A popular way of resolving the extra degrees of freedom is the approach of Powell [17], which finds the interpolating quadratic whose Hessian is closest to the prior model's Hessian, H^{k-1}:

$$\min_{c,g,H=H^\top} \left\{\|H - H^{k-1}\|_F^2 : q_k(y^i) = f(y^i), i = 1, \ldots, |\mathcal{Y}|\right\}. \tag{1.5}$$

This corresponds to minimizing a seminorm of z for a particular choice of basis and again places certain demands on the interpolation set $\mathcal{Y}$. We refer the interested reader to [18] for details of the solution procedure used in POUNDERS for solving (1.5).

We note that a common measure of approximation quality for interpolation-based models in derivative-free optimization is based on Taylor-like conditions [6]. For example, given a continuously differentiable function f, a model m is said to be "a fully linear approximation on $\mathcal{B}(x^k,\Delta)=\{x\in\Omega:\|x-x^k\|\le\Delta\}$ of f" if

$$|f(x)-m(x)|\le\nu_1\Delta^2 \quad\text{and}\quad |\nabla f(x)-\nabla m(x)|\le\nu_2\Delta \qquad \forall x\in\mathcal{B}(x^k,\Delta), \tag{1.6}$$

for some positive constants ν_1,ν_2 (independent of x and Δ).

1.1.2 Modeling residuals

Given that we have a vector mapping F, in POUNDERS we form a quadratic model

$$q_k^{(i)}(x)=c^{(i)}+(x-x^k)^\top g^{(i)}+\frac{1}{2}(x-x^k)^\top H^{(i)}(x-x^k) \tag{1.7}$$

of each residual $F_i(x)$ for $i=1,\dots,p$. At first glance, it may appear that we have substantially increased the linear algebraic overhead of determining the $\frac{(n+1)(n+2)}{2}$ coefficients for these p models. If we demand that the models employ a common interpolation set $\mathcal{Y}$, however, the system (1.4) becomes

$$\Phi(\mathcal{Y})Z=\begin{bmatrix}\phi(y^1)\\ \vdots\\ \phi(y^{|\mathcal{Y}|})\end{bmatrix}\begin{bmatrix} z^1 & \cdots & z^p\end{bmatrix}=\begin{bmatrix}F_1(y^1) & \cdots & F_p(y^1)\\ \vdots & & \vdots\\ F_1(y^{|\mathcal{Y}|}) & \cdots & F_p(y^{|\mathcal{Y}|})\end{bmatrix}=F(\mathcal{Y})^\top. \tag{1.8}$$

Thus, the coefficients for this collection of models are determined from a single linear system, with multiple right-hand sides. In practice, the basis matrix $\Phi(\mathcal{Y})$, and derived matrices associated with the approach in (1.5), is dense and therefore solved with a direct method. Since the main expense will be forming the inverse of this matrix (or a factorization of this inverse), the cost of obtaining Z grows slowly in p.

Furthermore, since the conditioning of (1.5) depends solely on $\mathcal{Y}$ (and the basis ϕ), approximation properties satisfied by any one of the models will be shared by the collection of models, provided that the residuals $\{F_i\}$ satisfy common regularity conditions. For example, if $q^{(1)}$ satisfies (1.6), then every $q^{(i)}$ determined in a similar manner—whether from (1.8) or the analog of (1.5)—is a fully linear approximation on $\mathcal{B}(x^k,\Delta)$ of the corresponding F_i, provided that all the residuals belong to the same class of functions (i.e., with regard to smoothness and boundedness).

The left two plots in Figure 1.1 illustrate quadratic interpolation models using a common interpolation set $\mathcal{Y}$ on a one-dimensional problem. The rightmost plot illustrates a coupling of these models, described next.

1.1.3 Master model for nonlinear least squares

Given a quadratic model for each residual, several ways exist for constructing a model for the objective f in (1.1). For example, direct substitution using $q^{(i)}\approx F_i$ in (1.1) would yield the fourth-order polynomial $\pi(x)=\frac{1}{2}\sum_{i=1}^p q^{(i)}(x)^2$.

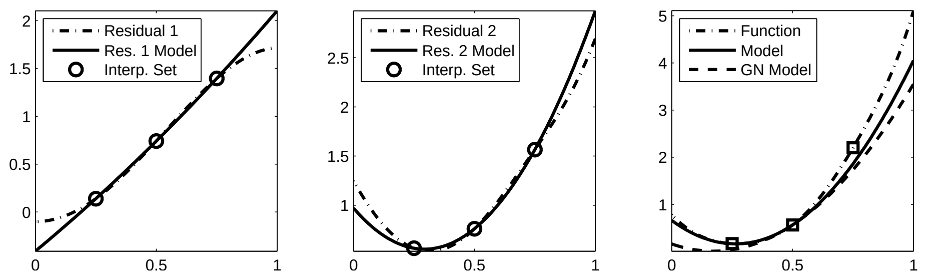

Figure 1.1: Illustration of quadratic interpolation models on a $n = 1$-dimensional problem with two residuals *(left two plots)*. The master model no longer interpolates the objective on the interpolation set; the Gauss-Newton model (GN) neglects the model Hessian terms *(rightmost plot)*.

Provided that the residuals are twice differentiable, we have that the first- and second-order derivatives of the objective are

$$\nabla f(x) = \sum_{i=1}^{p} \nabla F_i(x) F_i(x) \quad \text{and} \quad \nabla^2 f(x) = \sum_{i=1}^{p} \nabla F_i(x) \nabla F_i(x)^\top + \sum_{i=1}^{p} F_i(x) \nabla^2 F_i(x),$$

respectively. The implicit filtering method [10, 11] uses a Gauss-Newton model, whereby the second term in the Hessian is neglected and the gradient of a linear model is used in place of each (unavailable) ∇F_i.

In POUNDERS we employ the full second-order information for the Hessian of f and define the master model

$$m_k(x^k + \delta) = f(x^k) + \delta^\top \sum_{i=1}^{p} F_i(x^k) g(i) + \frac{1}{2}\delta^\top \sum_{i=1}^{p} \left(g^{(i)} (g^{(i)})^\top + F_i(x^k) H^{(i)} \right) \delta, \tag{1.9}$$

where the first term assumes that x^k belongs to the interpolation set $\mathcal{Y}^k$ and hence

$$\frac{1}{2} \sum_{i=1}^{p} \left(q^{(i)}(x^k) \right)^2 = \frac{1}{2} \sum_{i=1}^{p} \left(F_i(x^k) \right)^2 .$$

We refer to this model as the "full-Newton" model.

An approach that falls somewhere between the Gauss-Newton and full-Newton models is proposed in [20], whereby the model Hessian is regularized. The key strength of this approach in the unconstrained case is that this regularization yields fast local convergence for a class of zero residual problems [21]. POUNDERS and the DFLS algorithm in [20] both employ a trust-region framework but were developed independently and hence with different design goals. In particular, POUNDERS expects to be run primarily on problems where the residuals remain nontrivial. POUNDERS also avoids defining the breakpoints used in DFLS to determine when to use which second-order term.

An interesting observation about both the master model in (1.9) and the regularized model used in [20] is that these models no longer interpolate the nonlinear least squares objective f on the set $\mathcal{Y}$. For a univariate example, the rightmost plot in Figure 1.1 shows that the master model in (1.9) interpolates f only at the designated center point (in this case $x = 0.5$) but not at the remaining points in $\mathcal{Y}$. This plot also illustrates potential differences between the model in (1.9) and a Gauss-Newton model using an interpolation-based Jacobian estimate.

1.2 The POUNDERS Algorithm

We now discuss the basic form of the algorithm underlying POUNDERS. We use a trust-region framework, wherein the master model m_k in (1.9) is used as a quadratic surrogate for the objective f in a local neighborhood of the current iterate x^k.

Algorithm 1.1. Iteration k of Model-Based Algorithm Underlying POUNDERS.

Given $\mathcal{H}^k$,$x^k \in \Omega$,$\Delta_k > 0$ and constants $\eta > 0$, $\varepsilon > 0$, $\Delta_{\max} \leq \min_i\{u_i - l_i\}$:

1. Define $\mathcal{Y}^k$ based on $\mathcal{H}^k$, form m_k, and determine if m_k **valid**.

2. If $\|P(\nabla m_k(x^k), x^k, l, u)\| \leq \varepsilon$, check criticality; otherwise proceed to 3.

3. Solve the trust-region subproblem (1.13) to obtain x^+.

4. If $\|x^k - x^+\| \geq \frac{1}{100}\Delta_k$ or m_k is **valid**, proceed to 4a; otherwise, set $x^{k+1} = x^k$, $\Delta_{k+1} = \Delta_k$, $\rho_k = -1$, and go to 5.

   4a. Evaluate f at x^+ and compute $\rho_k = \frac{f(x^k)-f(x^+)}{m_k(x^k)-m_k(x^+)}$.

   4b. Update the trust region via

$$x^{k+1} = \begin{cases} x^+ & \text{if } \rho_k \geq \eta \\ x^+ & \text{if } \eta > \rho_k > 0 \text{ and } m_k \textbf{ valid} \\ x^k & \text{otherwise,} \end{cases} \tag{1.10}$$

$$\Delta^{k+1} = \begin{cases} \min\{2\Delta_k, \Delta_{\max}\} & \text{if } \rho_k \geq \eta \text{ and } \|x^k - x^+\| > \frac{3}{4}\Delta_k \\ \frac{\Delta_k}{2} & \text{elseif } m_k \textbf{ valid} \\ \Delta_k & \text{otherwise.} \end{cases} \tag{1.11}$$

5. If m_k is not **valid** and $\rho_k < \eta$, evaluate f at model-improving point and iterate; otherwise iterate.

The steps specified in Algorithm 1.2 are repeated until a specified budget of function evaluations has been exhausted or the criticality test (step 2) has been satisfied.

We collect the history of points for which the residual values are available in $\mathcal{H}^k$. In each iteration, an interpolation set $\mathcal{Y}^k \subseteq \mathcal{H}^k$ for the submodels (1.7) is constructed, from this history, based on the current iterate x^k and trust-region radius Δ_k. Similarly, we say that the model m_k is **valid** if it satisfies certain approximation guarantees (such as (1.6)) based on (x^k, Δ_k). Our technique for selecting points from $\mathcal{H}^k$ also determines the validity of the master model m_k and is discussed in [19]. Here we note only that the interpolation set is constructed in a way that ensures that any "model-improving points" evaluated (in step 5) since the last iterate change are included in $\mathcal{Y}^k$. This ensures that in no more than n consecutive iterations will model-improving points be evaluated before the resulting master model is deemed **valid**.

The criticality test is applied depending on a measure of the projected gradient step, with the ith component being defined by

$$[P(g,x,l,u)]_i = \begin{cases} 0 & \text{if } x_i = l_i \text{ and } g_i \geq 0 \\ 0 & \text{if } x_i = u_i \text{ and } g_i \leq 0 \\ g_i & \text{otherwise.} \end{cases} \tag{1.12}$$

The tolerance ϵ is specified by a user as input. If $\|P(\nabla m_k(x^k), x^k, l, u)\| \leq \varepsilon$ and m_k is not **valid**, then the trust region is maintained, $(x^{k+1}, \Delta_{k+1}) = (x^k, \Delta_k)$; we set $\rho_k = -1$; and we go to step 5 to evaluate a model-improving point. On the other hand, if $\|P(\nabla m_k(x^k), x^k, l, u)\| \leq \varepsilon$ and m_k is **valid**, then we must ensure that the trust-region (to which approximation quality is deeply tied, see (1.6)) is sufficiently small. In this case, either $\Delta^k \leq \epsilon$ and we terminate, or $\Delta^k > \epsilon$ and we set $(x^{k+1}, \Delta_{k+1}) = (x^k, \epsilon)$ and iterate (proceeding to step 1).

In each iteration where the criticality test is not invoked, a candidate point $x^+ \in \Omega$ is obtained by solving the trust-region subproblem

$$\min\left\{m_k(x) : x \in \mathcal{B}(x^k, \Delta_k)\right\}, \tag{1.13}$$

where we recall that the definition of the trust region $\mathcal{B}(x^k, \Delta_k)$ includes any bound constraints, thus ensuring that $x^+ \in \Omega$. We are purposely ambiguous about the norm $\|\cdot\|$ defining the trust region because, as discussed in the next section, in POUNDERS this norm (e.g., ℓ_2, ℓ_∞) depends on whether bound constraints are present and which subproblem solver is employed.

In typical trust-region algorithms, the candidate point x^+ is then evaluated. We avoid performing this evaluation, however, if both the resulting step is small and the current model is not deemed **valid**. In this case we instead perform the evaluation at a model-improving point. In all other cases, the candidate point is evaluated, and the usual ratio of actual decrease to predicted decrease (ρ_k) is computed. Using (1.10), the iterate is updated if this ratio is sufficiently large or if the model was **valid** and a strict decrease in the function value was obtained. Using (1.11), the trust-region radius is increased only if both the ratio and the steplength are sufficiently large. The radius is decreased only if the model is **valid**.

We note that if m_k is not **valid** and $\rho_k < \eta$, then the trust region (and hence the implied definition of validity) remains unchanged. In this case we evaluate F at a model-improving point, which in POUNDERS is actually generated at the same time as the model is determined not to be **valid** (in step 1). A key difference between the procedure employed in [19] and that used in POUNDERS, however, occurs when finite bounds are imposed. In the development of POUNDERS, a concerted effort was made to ensure that the model-improving points respect these bounds. This decision was made because these bounds are unrelaxable for many problems in practice; violating a bound could mean crashing the corresponding simulation evaluation (e.g., when a negative hydraulic conductivity is passed to a subsurface flow simulator). Because of our requirement that all points evaluated by the algorithm remain in Ω, we must bound $\Delta_{\max}$, the maximum trust-region radius. If the trust-region radius were allowed to grow significantly, one could not ensure that a model m_k is **valid** solely by using model-improving points that respect the bound constraints.

1.3 POUNDERS in TAO

We now describe further details of a specific implementation of POUNDERS.

1.3.1 The Toolkit for Advanced Optimization

The Toolkit for Advanced Optimization (TAO, [16]) is a software package designed for solving optimization problems on high-performance architectures. TAO has an open-source license and is available at `http://www.mcs.anl.gov/tao/`. The Portable, Extensible Toolkit for Scientific Computation (PETSc, [1]) provides the core scalable data structures and linear algebra routines that enable the parallel scalability of TAO. Consequently, TAO is used to solve problems on machines ranging from single-core laptops to massively parallel leadership-class supercomputers.

In addition to POUNDERS, the current version of TAO includes solvers for unconstrained optimization (e.g., limit-memory, variable metric quasi-Newton; Newton line search; and Newton trust-region methods), bound-constrained optimization (e.g., TRON, interior-point Newton method), PDE-constrained optimization, and complementarity problems. The use of parallel data structures and linear algebra routines makes the solvers in TAO especially amenable to solving large-scale problems.

For our purposes, however, the key benefit of these parallel capabilities in TAO is not for linear algebraic operations but for the objective function evaluation. In particular, the TAO separable objective functionality allows one to evaluate the residual F at a single point x using parallel resources. For example, if each residual component F_i can benefit from shared-memory parallelism to scale up to c cores, the separable objective capabilities allow for internode parallelism of the p residuals so that the wall-clock time for an objective evaluation can exhibit a potential speedup of cp. Examples of this functionality are included with TAO.

1.3.2 POUNDERS inputs

POUNDERS is available in TAO as the solver `tao_pounders`. The solver requires a minimal number of inputs:

- $x^0 \in \Omega$, an initial starting point;
- $\Delta_0 > 0$, the initial trust-region radius;
- a routine for evaluating the residual vector F at any given $x \in \Omega$;
- convergence criteria (e.g., a maximum number of function evaluations, a model gradient tolerance $\epsilon > 0$, a desired function value).

The values for the remaining internal constants in Algorithm 1.2 that are used by POUNDERS are $\eta = \frac{1}{10}$ and $\Delta_{\max} = \min\left\{\frac{1}{2}\min_i\{u_i - l_i\}, 1000\Delta_0\right\}$.

Advanced users can modify several POUNDERS options:

- How the trust-region subproblem is solved, including different solver types (e.g., an interior-point Newton method, the GQT routine from MINPACK-2), different subproblem tolerances, and different norms for the trust-region radius. By default, an infinity-norm trust region is used, and the resulting bound-constrained quadratic program is solved by the TAO solver TRON.
- A maximum number of interpolation points $|\mathcal{Y}|$ (in $\left\{n+2, \dots, \frac{(n+1)(n+2)}{2}\right\}$). By default, this is set to $2n+1$.
- Finite lower and/or upper bounds. By default, the problem is assumed to be unconstrained.
- A set of points $\mathcal{H}^0$ (and the corresponding residuals) at which the residual vector F has been evaluated prior to the call to POUNDERS. By default, this set is assumed to be empty.

Since the trust-region norms employed in POUNDERS treat each variable identically, the scaling of variables is an important consideration. Other solvers in TAO employ gradient information in order to scale each of the variables. Since POUNDERS does not have access to actual derivative information, the user must ensure that the problem is well-scaled. If the bounds have been selected so that the residual variation across these bounds is similar for each variable, then we advocate scaling the problem so that the bounds correspond to the unit hypercube $[0,1]^n$. For unconstrained variables, one would similarly scale each variable so that unit changes in each variable result in similar order-of-magnitude changes.

1.4 Calibrating Energy Density Functionals

We now illustrate the application of POUNDERS on problems arising in the UNEDF low-energy physics project [4].

A grand-challenge problem in low-energy nuclear physics is to determine an energy density functional (EDF) that describes properties of atomic nuclei across the nuclear landscape (see, e.g., Figure 1.2). One of the focuses of the

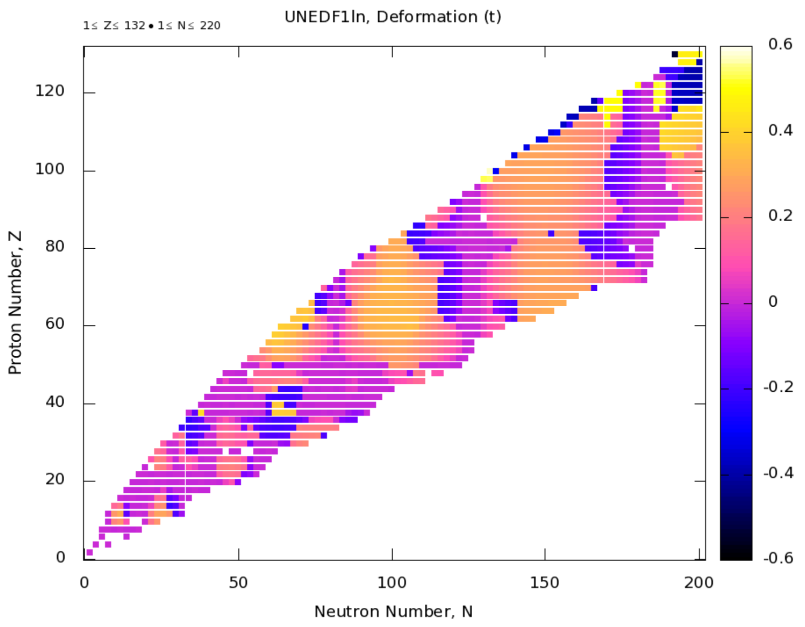

Figure 1.2: The nuclear landscape as shown by a table of nuclides (each column representing an element and its isotopes). The shading for each nuclei shows the total deformation as computed by an EDF code using the UNEDF$_1$ functional optimized by POUNDERS [14].

UNEDF project was developing EDFs based on density functional theory. Such EDF approaches depend on phenomenological constants, for example, in the form of low-energy coupling constants. Values for these constants are typically optimized based on fits to available experimental data and pseudo-data (derived from experiment and/or ab initio approaches). Selection of these fit observables depends on which coupling constants must be determined and on the desired properties for the resulting functional. A starting point for the optimization, which can play a critical role in the solution quality obtained by a local optimization solver, is often readily available in these calibration problems, for example, from the values of a previous-generation EDF optimization or from “natural values” for the coupling constants.

The number of optimization variables is typically on the order of a dozen; see Table 1.1. Although some coupling constants have natural ranges, the majority are effectively free: the physics-based view is that they are “constrained by the fit observables.” In practice, however, the simulation codes that evaluate a particular observable are not expected to produce meaningful output for arbitrary values of coupling constants. In the worst case, the code can even fail (e.g., because the underlying self-consistent-equation solver or eigensolver fails to converge). Thus, for computational reasons, bound constraints are often specified by application users as a way to restrict the domain in which the op-

Table 1.1: Problem characteristics.

Problem	UNEDF_0	UNEDF_1	UNEDF_2	NNLO_{opt}	BPW_{opt}
Reference	[12]	[14]	[13]	[8]	[3]
# Variables, n	12	12	14	14	17
# Residuals, p	108	115	130	2173	2049
# Nuclei Calc.	72	79	98	2173	2049

timization solver is allowed to operate. This is one of the reasons POUNDERS respects bound constraints, not only for the trust-region subproblem (1.13), but also for model-improving points. As a result, for these EDF calibration problems, the majority of the bounds are expected to be inactive at the solution. For example, for the UNEDF_2 solution [13], only two of the fourteen variables attained one of their bounds.

Observables used in these optimizations have included a wide range of nuclear properties. For example, in the UNEDF_2 study [13], the $p = 108$ residuals involved 47 deformed binding energies, 29 spherical binding energies, 28 proton point radii, 13 OES values, 4 fission isomer excitation energies, and 9 single-particle level splittings; in the BPW_{opt} study [3], only binding energies were considered but pseudo-data from over 2,000 nuclei were used. In all problems, the residual vector passed to POUNDERS corresponds to the scaled difference between a simulated observable of a particular nucleus and its corresponding experimental data or pseudo-data value. The scaling weights are typically based on the uncertainty in the data and simulation and the effects of these weights are typically analyzed at the solution to the optimization; see [12].

Computationally, the overwhelming expense in running POUNDERS on such problems can be attributed to the time required to evaluate the residual vector. The CPU time required to perform a single nucleus simulation (which results in multiple observable outputs for some problems, see Table 1.1) at a particular $x \in \Omega$ value ranged from 10 seconds in [3] to 12 minutes in [12].

Taking advantage of the separable function capabilities in TAO, one can reduce the wall-clock time needed to evaluate the residual vector by simulating each of the nuclei concurrently. If the simulator can itself exploit parallelism, then each nucleus calculation can be sped up further. For example, in the UNEDF_1 study, each of the 79 nuclei calculations employed a single node consisting of 8 cores, with the master TAO driver operating on another node. Using these 640 cores, the optimization evaluated 218 points in 5.67 hours of wall-clock time. The same run performed on a single core would have consumed roughly 70 days (the speedup being less than ideal because of load imbalance and the 8 cores not being perfectly utilized). For the UNEDF_2 run, the computational footprint grew to nearly 1,600 cores (with 16 cores for each of 98 nuclei calculations).

However, benefiting from larger computational resources is not the only factor accelerating the solution of expensive EDF calibration problems. Algorithmic improvements, which result in reductions in the number of points at which

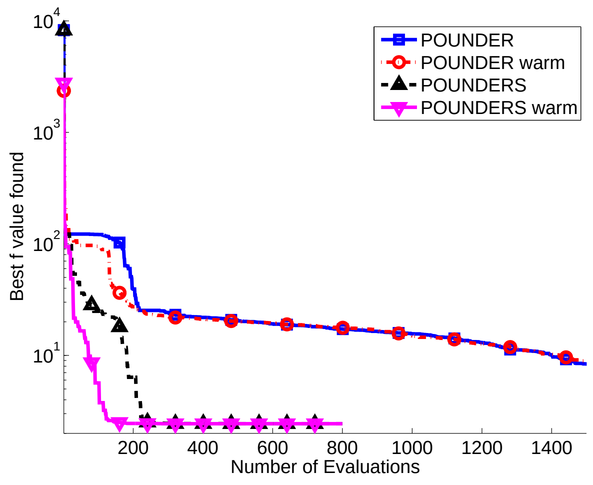

Figure 1.3: Best function value found (log scale) for the energy density calibration problem in calibration problem in [3]. This figure shows the benefit of exploiting the least-squares structure (POUNDERS) over not doing so (POUNDER); this benefit persists if both methods are warm-started with evaluations for an initial space filling design.

the simulators must be run in the course of the optimization, are responsible for a multiplicative scaling of this speedup. For the BPW_{opt} problem, Figure 1.3 shows the objective function values obtained with two different model-based trust-region algorithms: POUNDERS exploits the availability of the residual vector F, while POUNDER treats the scalar-valued aggregate f as a "black box". Comparable objective values are obtained in a factor 10 fewer evaluations when the structure is exploited. An even larger factor was seen when comparing POUNDERS with TAO's Nelder-Mead code on the UNEDF_0 problem [12].

Furthermore, POUNDERS has the ability to exploit residual evaluations done externally, for example, as a result of a variable scaling study, an efficient initial sampling, or a globalization strategy. For the BPW_{opt} problem, Figure 1.3 illustrates the benefits of using this information to warmstart the initial submodels used by POUNDERS.

1.5 Discussion

The POUNDERS solver has been used to make a number of advances in EDFs by solving computationally expensive, nonlinear least-squares problems in the absence of Jacobian information. We attribute the algorithmic benefits of the approach to the model-based framework, which has proven effective in the general blackbox case, and to taking advantage of the additional information (in this case, the residual vector) provided in greybox problems. Because of their core similarities, we expect that the DFLS algorithm would perform similar to POUNDERS on unconstrained problems; we also observe that DFLS compares favorably to other derivative-free methods on a large set of mathematical test problems [21]. Lowering the barrier to running many simulations concurrently has also been a key strength of POUNDERS.

Acknowledgment

This work was supported by the applied mathematics and SciDAC activities within the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research program under contract number DE-AC02-06CH11357.

Bibliography

[1] S. Balay, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, and H. Zhang, *PETSc users manual*, Technical Memorandum ANL-95/11 - Revision 3.4, Argonne National Laboratory, 2013. (Cited on p. 8)

[2] A. S. Bandeira, K. Scheinberg, and L. N. Vicente, *Computation of sparse low degree interpolating polynomials and their application to derivative-free optimization*, Mathematical Programming, 134 (2012), pp. 223–257. (Cited on p. 2)

[3] M. Bertolli, T. Papenbrock, and S. M. Wild, *Occupation number-based energy functional for nuclear masses*, Physical Review C, 85 (2012), p. 014322. (Cited on pp. 11, 12)

[4] S. Bogner, A. Bulgac, J. Carlson, J. Enge, G. Fann, R. Furnstahl, S. Gandolfi, G. Hagen, M. Horoi, C. Johnson, M. Kortelainen, E. Lusk, P. Maris, H. Nam, P. Navratil, W. Nazarewicz, E. Ng, G. Nobre, E. Ormand, T. Papenbrock, J. Pei, S. C. Pieper, S. Quaglioni, K. Roche, J. Sarich, N. Schunck, M. Sosonkina, J. Terasaki, I. Thompson, J. Vary, and S. Wild, *Computational nuclear quantum many-body problem: The UNEDF project*, Computer Physics Communications, 184 (2013), pp. 2235–2250. (Cited on p. 9)

[5] B. Colson and P. L. Toint, *Optimizing partially separable functions without derivatives*, Optimization Methods and Software, 20 (2005), pp. 493–508. (Cited on p. 2)

[6] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*, MPS/SIAM Series on Optimization, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009. (Cited on pp. 2, 3)

[7] A. R. Conn and L. N. Vicente, *Bilevel derivative-free optimization and its application to robust optimization*, Optimization Methods and Software, 27 (2012), pp. 561–577. (Cited on p. 2)

[8] A. Ekström, G. Baardsen, C. Forssén, G. Hagen, M. Hjorth-Jensen, G. R. Jansen, R. Machleidt, W. Nazarewicz, T. Papenbrock, J. Sarich, and S. M. Wild, *Optimized chiral nucleon-nucleon interaction at next-to-next-to-leading order*, Physical Review Letters, 110 (2013), p. 192502. (Cited on p. 11)

[9] A. Kannan and S. M. Wild, *Benefits of deeper analysis in simulation-based groundwater optimization problems*, in Proceedings of the XIX International Conference on Computational Methods in Water Resources (CMWR 2012), June 2012. (Cited on p. 2)

[10] C. T. Kelley, *Implicit filtering and nonlinear least squares problems*, in System Modeling and Optimization XX, E. Sachs and R. Tichatschke, eds., vol. 130 of IFIP – The International Federation for Information Processing, Springer US, 2003, pp. 71–90. (Cited on pp. 2, 5)

[11] C. T. Kelley, *Implicit Filtering*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2011. (Cited on pp. 2, 5)

[12] M. Kortelainen, T. Lesinski, J. Moré, W. Nazarewicz, J. Sarich, N. Schunck, M. V. Stoitsov, and S. M. Wild, *Nuclear energy density optimization*, Physical Review C, 82 (2010), p. 024313. (Cited on pp. 11, 12)

[13] M. Kortelainen, J. McDonnell, W. Nazarewicz, E. Olsen, P.-G. Reinhard, J. Sarich, N. Schunck, S. M. Wild, D. Davesne, J. Erler, and A. Pastore, *Nuclear energy density optimization: Shell structure*, Preprint ANL/MCS-P5038-1113, Argonne Mathematics and Computer Science Division, 2013. (Cited on p. 11)

[14] M. Kortelainen, J. McDonnell, W. Nazarewicz, P.-G. Reinhard, J. Sarich, N. Schunck, M. V. Stoitsov, and S. M. Wild, *Nuclear energy density optimization: Large deformations*, Physical Review C, 85 (2012), p. 024304. (Cited on pp. 10, 11)

[15] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *User guide for MINPACK-1*, Tech. Rep. ANL-80-74, Argonne National Laboratory, Argonne, IL, Aug. 1980. (Cited on p. 2)

[16] T. Munson, J. Sarich, S. M. Wild, S. Benson, and L. Curfman McInnes, *TAO 2.0 users manual*, Technical Memorandum ANL/MCS-TM-322, Argonne National Laboratory, Argonne, IL, 2012. (Cited on p. 8)

[17] M. J. D. Powell, *Least Frobenius norm updating of quadratic models that satisfy interpolation conditions*, Mathematical Programming, 100 (2004), pp. 183–215. (Cited on p. 3)

[18] S. 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`. (Cited on p. 3)

[19] S. M. Wild and C. A. Shoemaker, *Global convergence of radial basis function trust-region algorithms for derivative-free optimization*, SIAM Review, 55 (2013), pp. 349–371. (Cited on pp. 2, 7, 8)

[20] H. Zhang, A. Conn, and K. Scheinberg, *A derivative-free algorithm for least-squares minimization*, SIAM Journal on Optimization, 20 (2010), pp. 3555–3576. (Cited on pp. 2, 5, 6)

[21] H. Zhang and A. R. Conn, *On the local convergence of a derivative-free algorithm for least-squares minimization*, Computational Optimization and Applications, 51 (2012), pp. 481–507. (Cited on pp. 2, 5, 13)