Managing Approximation Models in Optimization

J. E. Dennis and Virginia Torczon*

June 30, 1995

Abstract

It is standard engineering practice to use approximation models in place of expensive simulations to drive an optimal design process based on nonlinear programming algorithms. This paper uses well-established notions from the literature on trust-region methods and a powerful global convergence theory for pattern search methods to manage the interplay between optimization and the fidelity of the approximation models to insure that the process converges to a reasonable solution of the original design problem. We present a specific example from the class of algorithms outlined here, but many other interesting options exist that we will explore in later work.

The algorithm we present as an example of the management strategies we propose is based on a family of pattern search algorithms developed by the authors. Pattern search methods can be successfully applied when only ranking (ordinal) information is available and when derivatives are either unavailable or unreliable. Since we are interested here in using approximations to provide arguments for the objective function, our choice seems relevant.

This work is in support of the Rice effort in a collaboration with Boeing and IBM to look at the problem of designing helicopter rotor blades.

1 Introduction

A consistent theme in the engineering optimization literature [18] is that the time and cost required for the detailed analysis of a single design is often so great that it becomes prohibitive to implement a "black-box" optimization approach to the design problem. Because of the cost of running these detailed analyses (full simulations), the point of using approximation techniques is to reduce the number of full, or detailed, analyses required during optimization while maintaining the salient features of the design problem. A recent survey with 85 references to the vast literature on approximation concepts for optimum structural design can be found in the review by Barthelemy and Haftka [1]. Additional references are given in the paper by Yesilyurt and Patera [20], which offers a point of view similar to our own.

*This research was sponsored by the Air Force Office of Scientific Research grant F 49620–95–1–0210.

We do not propose to develop new modeling techniques. Rather, the open question we study is how to manage the interplay between the optimization and the fidelity of the approximation models to insure that the process converges to a solution of the original design problem. In short, we give a procedure that is straightforward to implement and yet converges to a solution to the original design problem. Of course, the efficiency of the optimization depends rather directly on how faithful the model can be made to the original design problem.

For simplicity in the following discussion, we assume that we are interested in solving a problem of the form

$$\begin{array}{ll} \min & f(x, y(x)) \\ \text{s.t.} & l \le x \le u, \end{array}$$

where $f : I\!R^n \to I\!R$, $y(x)$ represents system variables and $l, u \in I\!R^n$ represent upper and lower bounds on the decision variables. The idea, then, is to use simplified analysis models of the original complicated analyses represented by $y(x)$. We assume that f is inexpensive to evaluate once a value for $y(x)$ is known, so $f(x, y(x))$ is computed directly, though it is certainly possible within the framework we propose to model $f(x, y(x))$ instead.

We assume a family of models $\mathcal{M} = \{M^\alpha : \alpha \in \mathcal{A}\}$, where α is an index into a set $\mathcal{A}$ of possible approximation models. Thus $M^\alpha(x)$ represents an approximation model for $y(x)$. We use $M^*(x)$ to denote the result we get if we do everything reasonable to obtain the most accurate value of $y(x)$.

We note that a detailed analysis will itself usually be a discretization of a continuous model of the underlying physical process. Thus $y(x)$ may represent the underlying physical process while $M^*(x)$ is a discretization of the continuous problem. Ultimately, the optimization problem we will solve is the one defined by $f(x, M^*(x))$, since that is the problem for which it is assumed that reasonably accurate values can be computed, and we use $\mathcal{M}$ to model $M^*(x)$.

For the purposes of this discussion, we require $M^\alpha(x) = M^*(x)$ for all $\alpha \in \mathcal{A}$ at any point x for which $M^*(x)$ has been computed. This has not proven to be an onerous restriction in the design of experiment interpolation modeling (see [2] and [19]) we have been investigating as part of our joint research with Boeing and IBM to look at the problem of designing helicopter rotor blades; any new data point (i.e., any new point x for which $f(x, M^*(x))$ is known) is simply added to the correlation matrix used to define the surrogate model, thus insuring an exact match regardless of any other modifications that might be made to the model.

Later, we will investigate ways to integrate into our framework for model management some heuristics for approximating $M^*(x)$ when it is not defined as above by running a single detailed analysis; for example, when one wishes to approximately solve the continuous problem without declaring some "final accuracy" in the simulation [21]. These strategies are straightforward modifications of heuristics used in adaptive gridding for differential equation solutions. Although we do not wish to make strong claims for these heuristics when x_* is a solution to the continuous problem defined by $y(x)$, nonetheless, they do seem reasonable in that case.

The general framework for model management that we outline in §2 is based on trust-region strategies for globalizing Newton's method. Although trust-region methods are an active research area, and they are incorporated into various widely used software packages, it is our impression that they are likely to be unfamiliar to some readers in the engineering design community. It is impossible to survey adequately in such a short space a research area undergoing such rapid developments. Furthermore, its relevance to the general framework for model management given in §2 is more subtle than first appears. Therefore, we give an incomplete introduction to this area in §3 directed toward providing a context for our general framework for the management of approximation models. In §4 we then give an incomplete introduction to pattern search methods and the key elements of the global convergence analysis that we intend to exploit to develop specific algorithms. In §5, we give a specific algorithm to which we are confident this analysis applies. We devote §6 to some conclusions and a summary of future work.

2 The model management framework

Our goal is to solve the optimization problem defined by $f(x, M^*(x))$. But we would rather not run the full simulation $M^*(x)$ to compute $f(x, M^*(x))$ for every intermediate point x considered during the course of a single step of an optimization procedure. The idea then, regardless of how the approximation is done, is to use the model $M^\alpha(x)$ and compute the value of the objective function as $f(x, M^\alpha(x))$ to drive each step of an optimization procedure. The reasoning is that if it takes a significant amount of computational time to simulate the full physics of a particular process—for instance, a large case analysis for the helicopter rotor design problem can require a few hours on a Cray-YMP [11]—then the optimization may be able to progress more quickly if conducted using surrogate functions with less fidelity but much quicker computational turn-around.

Our interest is in using the agreement between the values returned by the approximation models (as measured by the function value f) and the values returned by the detailed analyses for a putative new best design (again, as measured by the function value f) to determine how far to go in each optimization cycle on the approximate problem before computing another set of detailed analyses to check progress. We use this same measure of agreement to determine when and where a refined model is needed for further progress on the optimization of $f(x, M^*(x))$.

With this setting in mind, we start by proposing the following general framework for an optimization method that uses approximation models. Later we will discuss how pattern search and gradient-based methods can be inserted in Steps 2 and 4 in the framework for model management given below.

We assume that the user has selected a variable fidelity family of models $\mathcal{M} = \{M^\alpha : \alpha \in \mathcal{A}\}$ for $y(x)$ and an initial fidelity index α,[1] as well as an initial (feasible) design for which a full simulation has been executed.[2]

[1]For example, α might indicate mesh size in a PDE code.

[2]Of course, this initial design may have been selected by some auxiliary initialization procedure, such as solving the box-constrained problem with the approximate objective function $f(x, M^\alpha(x))$ from some initial point that may not have been feasible. But however it was selected, we start our discussion at a point x for which $M^*(x)$ is available.

In the discussion that follows, x_k denotes the current iterate (the best design confirmed to date), x_{k+} denotes a trial iterate (a putative new best design) produced by the optimization algorithm using the current approximation model, x_{k+1} denotes the iterate accepted to finish the current iteration and start the next one, y_k^α denotes the value of $M^\alpha(x_k)$, and y_k^* denotes the value of $M^*(x_k)$. We again stress that we assume that $M^\alpha(x) = M^*(x)$ at any point for which $M^*(x)$ has been computed; in particular, $M^\alpha(x_k) = M^*(x_k)$ for all k.

We also use the constants $0 < \eta_1 < \eta_2$ to judge how well an approximation model M^α matches the full simulation M^* at a given iterate, as measured by f. What is important is that if our current approximation model M^{α_k} is doing a good job of producing trial iterates such that $f(x_{k+}, M^*(x_{k+})) < f(x_k, M^*(x_k))$, then we might consider using an even cheaper model, subject to the requirement that the value of the model matches the value at $M^*(x)$ at any point x for which $M^*(x)$ has been computed. However, if the current model is doing a poor job of predicting decrease from $f(x_k, M^*(x_k))$, then it may be advisable to consider choosing a better model from the family of models $\mathcal{M}$. The actual conditions are somewhat arbitrary; reasonable choices of η_1 and η_2 might be 10^{-4} and 0.75. (See [6].)

The General Framework for the Management of Approximation Models

Given M^*, M^α, x_0, y_0^*, and $0 < \eta_1 < \eta_2$, for $k = 0, 1, \cdots$, do

1. Check for convergence. Otherwise, continue.

2. Apply an optimization algorithm to the approximate problem to find an x_{k+} for which $f(x_{k+}, y_{k+}^\alpha)$ satisfies an appropriate decrease condition for $f(x, M^\alpha(x))$ from x_k.

   Compute $pred_k \equiv f(x_{k+}, y_{k+}^\alpha) - f(x_k, y_k^*)$ (the predicted reduction).

3. Compute $y_{k+}^* = M^*(x_{k+})$ (by a detailed analysis or adaptive heuristics).

   Compute $ared_k \equiv f(x_{k+}, y_{k+}^*) - f(x_k, y_k^*)$ (the actual reduction).

   Compute $r_k = \frac{ared_k}{pred_k}$.

4. If $r_k \leq 0$ (improvement was predicted but not achieved), then

   Set $x_{k+1} = x_k$ and $y_{k+1}^* = y_k^*$. (Reject the step.)

   Allow less optimization on the approximate problem at the next iteration.

   Consider getting a more faithful model M^α anchored at x_k. (Refine the model.)

   Else, if $0 < r_k < \eta_1$ (much more improvement was predicted than achieved), then

   Set $x_{k+1} = x_{k+}$ and $y_{k+1}^* = y_{k+}^*$. (Accept the step.)

   Allow less optimization on the approximate problem at the next iteration.

   Consider keeping the current approximation model M^α.

   Else, if $\eta_1 \leq r_k < \eta_2$ (the prediction was satisfactory), then

   Set $x_{k+1} = x_{k+}$ and $y_{k+1}^* = y_{k+}^*$. (Accept the step.)

Allow the same optimization on the approximate problem at the next iteration.

Consider keeping the current approximation model M^α.

Else, $\eta_2 \le r_k$ (prediction was excellent, or more decrease was obtained than predicted),

Set $x_{k+1} = x_{k+}$ and $y^*_{k+1} = y^*_{k+}$. (Accept the step.)

Allow more optimization on the approximate problem at the next iteration.

Consider using a *less* accurate approximation model. (Relax the model.)

5. Return to Step 1.

To finish the specification for the general algorithm, we must determine

- how to find a trial step x_{k+},
- what constitutes an appropriate decrease condition for $f(x, M^\alpha(x))$ from x_k, and
- how to update the decrease requirement on the optimization of the approximate problem in Step 2.

We discuss these issues below in context.

3 Trust-region motivations for the general framework

In this section, we present an introduction to trust-region philosophy, which motivated our model management framework given in the previous section. However, we warn the reader that the powerful trust-region convergence theory does not apply immediately to this new context. In order to keep this distinction clear, we first state a generic trust-region algorithm for unconstrained optimization in a form that emphasizes the connections between the trust-region philosophy and our general framework for managing approximation models. We then follow this with a discussion of some important differences.

3.1 Trust-region methods

There are strong convergence results for constrained and unconstrained trust-region methods for smooth and nonsmooth problems. See [12] for a survey of earlier work on smooth unconstrained problems, [9] for a survey of earlier work on nonsmooth problems, [15] for the highly regarded bundle trust-region method for nonsmooth problems, and [3], [5], [10], [4], [8] for work on smooth constrained problems.

For our purposes, it seems best to illustrate the concepts using the following smooth unconstrained optimization problem:

$$\min_{x \in I\!R^n} \phi(x) \tag{1}$$

where $\phi : I\!R^n \to I\!R$ and $\nabla\phi$ is uniformly continuous. A basic notion for trust-region algorithms is that of a local quadratic model about x_k for $\phi(x_k + s)$ given by $q_k(s) =$

$\phi(x_k) + \nabla\phi(x_k)^T s + \frac{1}{2}s^T H_k s$. For the global convergence analysis, the sequence of approximate Hessians $\{H_k\}$ is required only to be uniformly bounded. This condition on the Hessian approximations gives great latitude in theory and practice; for example, even the zero matrix can be used if one wishes to avoid some of the numerical linear algebra in the step computations mentioned below.

A Generic Trust-Region Framework

```
Given $q_0$, $x_0$, $\phi(x_0)$, $\delta_0$, $0 < \gamma < 1$, and $0 < \eta_1 < \eta_2 < 1$
(possible choices for these constants are $\gamma = 0.5$, $\eta_1 = 10^{-4}$, and $\eta_2 = 0.75$);
for $k = 0, 1, \cdots$, do

  1. Check for convergence. Otherwise, continue.

  2. Find an *appropriately accurate* approximate solution $s_k$ to the trust-region subproblem
     $\min q_k(s)$ subject to $\|s\| \le \delta_k$.
     Set $x_{k+} = x_k + s_k$.
     Compute $pred_k \equiv q_k(s_k) - \phi(x_k)$ (the predicted reduction).

  3. Compute $\phi(x_{k+})$.
     Compute $ared_k \equiv \phi(x_{k+}) - \phi(x_k)$ (the actual reduction).
     Compute $r_k = \frac{ared_k}{pred_k}$.

  4. If $r_k \le 0$ (improvement was predicted but not achieved), then
         Set $x_{k+1} = x_k$. (Reject the step.)
         Set $\delta_{k+1} = \gamma\|s_k\|$. (Decrease the trust radius.)
     Else, if $0 < r_k < \eta_1$ (much more improvement was predicted than achieved), then
         Set $x_{k+1} = x_{k+}$. (Accept the step.)
         Set $\delta_{k+1} = \gamma\|s_k\|$. (Decrease the trust radius.)
     Else, if $\eta_1 \le r_k < \eta_2$ (prediction was satisfactory), then
         Set $x_{k+1} = x_{k+}$. (Accept the step.)
         Set $\delta_{k+1} = \delta_k$. (Keep the same trust radius.)
     Else, $\eta_2 \le r_k$ (prediction was excellent, or more decrease was obtained than predicted),
         Set $x_{k+1} = x_{k+}$. (Accept the step.)
         Set $\delta_{k+1} = \frac{\delta_k}{\gamma}$. (Increase the trust radius.)

  5. Update the quadratic model $q_k$ and return to Step 1.
```

To finish the specification for the trust-region algorithm, we must say what we mean by finding an *appropriately accurate* solution in Step 2. The answer is that for global convergence we only need satisfy a condition we call *fraction of Cauchy decrease* (*fCd*): s_k must be determined to satisfy $pred_k \geq \rho[q(s_k^{Cp}) - \phi(x_k)]$ for any fixed $\rho \in (0,1]$. This is a trivial condition to satisfy. We call s_k^{Cp} the Cauchy step because it solves the constrained steepest descent trust-region subproblem

$$\min_{t \in [0,\delta_k]} q_k \left(-t \frac{\nabla \phi(x_k)}{\|\nabla \phi(x_k)\|} \right).$$

The trust-region paradigm is so powerful that this *fCd* condition is enough, even without any convexity assumptions, to ensure global convergence in the sense that $\liminf \|\nabla \phi(x_k)\| = 0$. See [14] for the unconstrained problem and [5] for the equality constrained problem.

The fraction of Cauchy decrease condition allows us to prove that the local convergence rate is linear. If we want to prove a faster convergence rate, say second order convergence, we need to use $H_k = \nabla^2 \phi(x_k)$ and to satisfy a condition known as *fraction of optimal decrease* (*fod*). For this condition, s_k must be determined to satisfy $pred_k \geq \rho[q(s_k^{opt}) - \phi(x_k)]$ for any fixed $\rho \in (0,1]$. See [13] for the unconstrained problem and [10] for the equality constrained problem. This *fod* condition is satisfied by a computational solution of the following full trust-region subproblem whose exact solution is s_k^{opt}:

$$\min_{\|s\| \leq \delta_k} q_k(s).$$

Of course, *fod* is not so trivial to satisfy in practice as *fCd*; for example, in nonlinear least squares with H_k taken to be the Gauss-Newton approximation, s_k^{opt} is the familiar Levenberg-Marquardt step.

3.2 Similarities and differences between trust-region methods and the framework for the management of approximation models

We have borrowed for our model management framework from trust-region methods the notions of:

1. using an underlying model,
2. requiring a trial step to provide decrease for that model, and
3. using a comparison of the predicted and actual decrease caused in the model objective and the actual objective by the trial step to manage the interaction of the model and actual problem.

We reinforce this motivation for the model management framework of §2 and for the specific algorithm of §5 by using standard trust-region notation in their definition.

The crucial difference is that our framework for managing approximation models must allow for trial steps chosen for a model problem $f(x, M^\alpha(x))$ which is likely to be much less accurate around x_k than a Taylor series as an approximation to $f(x, M^*(x))$. When the trial step is rejected, in either the model management framework or the trust-region framework, the problem clearly is that the model objective function on which the algorithm is based is not sufficiently accurate at the trial point x_{k+}. This requires a more complex remedy in the model management framework than it does in the trust-region framework. This leads to the most subtle point in the paper.

In the general framework for managing approximation models there can be two causes for poor agreement between the predicted reduction and the actual reduction. It could be that the optimization algorithm applied to the model problem in Step 2 has *over-optimized* the model problem and taken the decision variable x outside a region where the optimization model problem is accurate. The fix is to do less optimization on the model problem and perhaps improve the accuracy of the model. It could also be that the model problem must be made more accurate if even a small optimization step for the model problem is to cause decrease for $f(x, M^*(x))$. The fix is to get a more accurate model, or to terminate the computation. In §2 we attempted to apply these fixes to either situation in what seemed to us to be reasonable ways, given our measure of predicted versus actual decrease.

The situation is simpler in the trust-region framework; as the trust radius δ is decreased, it decreases both the amount of quadratic model optimization required of the trial step and the error in the approximate objective function. The latter is just calculus, and from the definitions of s_k^{Cp} and s_k^{opt}, it can be seen that the respective decrease conditions are less stringent for smaller δ.

3.3 Implications for the management of approximation models

A trial iterate x_{k+} in the model management framework might be chosen by applying a trust-region method to the approximate problem $f(x, M^\alpha(x))$. If we were to use a trust-region method as the optimization method within the general framework outlined in §2, then we would be adding an extra modeling layer in which the model of $f(x, M^*(x))$ given by $f(x, M^\alpha(x))$ is in turn modeled locally by a quadratic Taylor series type model. We postpone the investigation of this particular approach to future work.

It would be premature to discuss in much detail the myriad possibilities for deciding how much optimization to do on the model problem of minimizing $f(x, M^\alpha(x))$ before checking progress by going on to Step 3 of the general framework given in §2. In all cases, we will follow the standard trust-region notion of doing more optimization on the model problem at the next step if the agreement between the actual and predicted reduction at the current step is good and less if the agreement is bad.

With this general framework, any reasonable suite of models should work in practice and yield to a trust-region approach to analysis. Some examples are to go to Step 3 when:

(a) $x_{k+} - x_k$ is as large as we think prudent currently. In Step 4, based on the value of r_k, increase or decrease this allowable change for the next iteration.

(b) x_{k+} is the minimizer of $f(x, M^\alpha(x))$ in some current subregion of design space. In Step 4, based on the value of r_k, update this subregion for the next iteration.

(c) x_{k+} is a single successful optimization iteration from x_k, on minimizing $f(x, M^\alpha(x))$.

(*d*) $x_{k+} - x_k$ satisfies some strong Armijo condition, say,

$$f(x_{k+}, y^{\alpha}_{k+}) - f(x_k, y^*_k) \leq 0.5 \nabla f(x_k, y^{\alpha}_k)^T (x_{k+} - x_k).$$

But now having introduced these possible ways to choose trial steps for trust-region methods, we see that they all carry over directly to line-search methods. An interesting addition might be to go to Step 3 when:

(*e*) x_{k+} is a minimizer of $f(x, M^{\alpha}(x))$ along the direction of search from x_k within some current estimate of a prudent allowable change, which is updated based on r_k in Step 4.

Our model management framework is flexible enough to handle the extreme case where the user evaluates the model $f(x, M^{\alpha}(x))$ at points on a grid and x_{k+} is the best grid point. This flexibility will be useful for the specific algorithm we propose in §5, which will employ a pattern search method, rather than a gradient-based method, as the optimization algorithm in Step 2. Our choice of a pattern search method is motivated by our intent to extend the global convergence theory of pattern search methods to cover a specific algorithm that follows the model management strategy we propose. But this choice of a pattern search method highlights the distinction between the trust-region philosophy we have borrowed and the trust-region convergence theory, which does not apply in this new context. In particular, pattern search methods are not based on local models. Thus there is no well-tested notion of a principle of sufficient predicted decrease to guide us in deciding when to stop applying a pattern search algorithm to the approximate optimization problem of minimizing $f(x, M^{\alpha}(x))$ and declare the best point found so far to be the trial step x_{k+}. Instead, we rely on simpler strategies, as we discuss further in the next section.

4 Pattern search motivations for specific algorithms

In this section, we present an introduction to pattern search methods for optimization. While the trust-region motivation provides a general framework for the model management strategy we advocate, the pattern search methods give us practical algorithms to use immediately within this general framework. And the advantage of using pattern search methods is that the powerful convergence theory for pattern search methods appears to apply, without modification, within this new context. We will present the generalized pattern search algorithm for unconstrained optimization, explain the key conditions that make a global convergence analysis possible, and then discuss why the resulting combination fits so neatly within our general framework for managing approximation models.

4.1 Pattern search methods

We return to the smooth unconstrained problem defined in (1). A basic component of the definition for pattern search methods is the *pattern* of points P_k about x_k from which one considers possible steps. The pattern is coupled with an *Exploratory Moves* algorithm that decides which steps defined by the pattern will be considered and in what order the evaluations are to be done. The scalar δ_k serves as a step length control parameter. The particular choice of a pattern and an Exploratory Moves algorithm distinguishes the individual pattern search methods. There is tremendous flexibility in the way a pattern search algorithm can be specified, though there are technical restrictions, some of which we touch on below, that must be satisfied to ensure global convergence in the sense that $\liminf \|\nabla \phi(x_k)\| = 0$. (See [17] for a complete discussion.) It is this great latitude in the way in which steps are chosen for evaluation that we seek to exploit within the general framework we propose for the management of approximation models.

The Generalized Pattern Search

Given P_0, x_0, $\phi(x_0)$, δ_0, and $0 < \gamma < 1$ (the usual choice for this constant is $\gamma = 0.5$);
for $k = 0, 1, \cdots$, do

```
1. Check for convergence. Otherwise, continue.

2. Use an *Exploratory Moves* algorithm to determine a step $s_k$ from among those steps
   defined by $\delta_k P_k$ for which $\phi(x_k + s_k) \le \phi(x_k)$.
   Set $x_{k+} = x_k + s_k$.

3. Compute $ared_k \equiv \phi(x_{k+}) - \phi(x_k)$ (the actual reduction).

4. If $ared_k \le 0$ (improvement was not achieved), then
       Set $x_{k+1} = x_k$. (Reject the step.)
       Set $\delta_{k+1} = \gamma \delta_k$ (Decrease the step length control factor.)
   Else, $ared_k > 0$ (improvement was achieved)
       Set $x_{k+1} = x_{k+}$. (Accept the step.)
       Set $\delta_{k+1} = \delta_k$ (Keep the same step length control factor.)
       *Or* consider setting $\delta_{k+1} = \frac{\delta_k}{\gamma}$. (*Or* consider increasing it.)

5. Update the pattern $P_k$ and return to Step 1.
```

To finish the specification for the pattern search algorithm, we must choose both a pattern and an Exploratory Moves algorithm.

The pattern can be viewed as a matrix $P_k \in I\!R^{n \times m}$, where $m > 2n$. The columns of P_k define all possible steps allowed at the current iteration. We require $m > 2n$ because for the global convergence analysis, the columns of P_k must contain a core pattern defined by $[\Gamma_k, -\Gamma_k]$, where $\Gamma_k \in I\!R^{n \times n}$ and nonsingular. For convenience, we also allow for the possibility that $s_k = 0$ so that the condition $\phi(x_k + s_k) \le \phi(x_k)$ can be satisfied. Thus the pattern matrix must have a minimum of $2n+1$ columns, but in fact there is no upper bound on the number of columns that are allowed as long as the remaining columns satisfy certain minimal restrictions. (These are covered in detail in [17].) The core pattern ensures that if the current iterate x_k is not a critical point of the function, then at least one of the steps defined by the core pattern lies along a direction of descent from x_k, though we have no way of predicting a priori which direction this might be.

To ensure global convergence, we use the pattern matrix to place the following two conditions on the Exploratory Moves:

1. The step s_k must be defined by $\delta_k P_k$.

2. If there exists a step σ_k, defined by δ_k and some one of the $2n$ steps in the core pattern $[\Gamma_k, -\Gamma_k]$, for which $\phi(x_k + \sigma_k) < \phi(x_k)$, then the Exploratory Moves must return a step s_k for which $\phi(x_k + s_k) < \phi(x_k)$.

   There is no requirement that s_k must be defined by the core pattern, or that all $2n$ steps defined by the core pattern must be evaluated at every iteration k, or even that the step returned give the greatest decrease possible among all m steps defined by $\delta_k P_k$.

Thus, a legitimate Exploratory Moves algorithm would be one that somehow "guesses" which of the m steps defined by $\delta_k P_k$ will produce decrease on $\phi(x_k)$ and then evaluates ϕ at that single step. At the other extreme, an equally legitimate choice for the Exploratory Moves algorithm would be one that evaluates ϕ at all m steps defined by $\delta_k P_k$ and returns the step that produced the least function value.

The questions then become: How can we extend the trust-region strategy for managing a quadratic model of the true function to the more general setting of managing any family of approximation models that define the true function? Can we combine this with the simple decrease principle of pattern search methods? And do so in a way that will guarantee convergence of the optimization process to a solution of the true problem?

4.2 Pattern search methods as part of a model management strategy

We propose to introduce the trust-region philosophy of using a model—though not necessarily a quadratic model—into the simple and flexible algorithmic framework of pattern search methods in the following way. We use an optimization algorithm to satisfy an appropriate decrease condition for $f(x, M^{\alpha}(x))$ from x_k and to *predict* which one of the steps defined by the pattern matrix is most likely to produce decrease on the true objective function $f(x, M^{*}(x))$. If the model—however it is derived—is reasonably accurate in the neighborhood of the search, then the prediction should be adequate. Now, instead of "guessing" which of the m steps defined by $\delta_k P_k$ will produce simple decrease for $f(x, M^{*}(x))$ from x_k, we should be able to predict such a step—and thus only have to compute a single function value requiring a full set of expensive simulations to verify that this prediction is correct. We assess the quality of our model exactly as we would assess the quality of the quadratic model in a standard trust-region method: by comparing the amount of decrease predicted by the model to the amount of decrease actually realized. We then use the same heuristics employed by trust-region methods to suggest when either further refinement of the model or less optimization on the model problem is indicated to make further progress on the optimization.

To preserve the convergence properties of pattern search methods, we must also respect the second condition on the Exploratory Moves. In the worst case, if our model problem has not been able to predict a point that produces descent on the true function, then we may need to poll the steps defined by the core pattern to ensure that we have not overlooked a possible direction of descent. However, there is nothing in this condition that says we must necessarily look at all $2n$ steps defined by the core pattern; as soon as we find a step that improves the objective function, we are free to resume the basic strategy outlined above. And it makes sense to use the model problem to predict the order in which the columns of the core pattern should be polled. We will refer to the polling of the steps defined by the core pattern as our fallback strategy—something to be avoided if possible until near a solution, but necessary if we are to guarantee global convergence.

We close by noting that we still have been vague in our specification of what it means to satisfy "an appropriate decrease condition for $f(x, M^\alpha(x))$ from x_k." We are applying a pattern search method to the problem of minimizing $f(x, M^*(x))$. The strategy behind the Exploratory Moves we employ is to use the model problem $f(x, M^\alpha(x))$ to try and produce a single step s_k at which we must run the full simulation $M^*(x_k + s_k)$. An obvious way to accomplish this is to use an optimization algorithm to find a step that produces decrease on $f(x, M^\alpha(x))$ from x_k, but we are free to do this in any way we desire as long as the step returned by the Exploratory Moves for the true problem satisfies the two conditions we have outlined above. To the extent that the step returned by the Exploratory Moves must be defined by the columns of $\delta_k P_k$ if we are to guarantee global convergence, it makes sense to use another pattern search method—with the same pattern search matrix P_k and the same step length control parameter δ_k—on the problem defined by $f(x, M^\alpha(x))$. But we would advocate using a far more aggressive Exploratory Moves algorithm for the model problem, given that the model has been designed so that the function evaluations are much less expensive to compute. It is this strategy we will pursue in the specific algorithm given in the next section. However, another strategy that makes sense if the model has been constructed to be smooth and have derivatives that are easy to obtain, would be to use another optimization method—perhaps a gradient-based method—to find a step that gives decrease on the model problem. As long as corrective measures are taken to compensate for the fact that most methods do not place the same restrictions on the form of the step as do pattern search methods (i.e., there is no guarantee that the step returned by a trust-region method applied to the model problem will be defined by $\delta_k P_k$), so that both conditions on the Exploratory Moves are satisfied, then the global convergence theory for pattern search methods will still hold. This second strategy will be the subject of future work.

5 A specific algorithm

The theory for pattern search methods grew out of our work designing, implementing, and testing the family of direct search algorithms first presented in [7]. We call this family of algorithms *parallel direct search* (PDS) both because they were designed to be implemented on either sequential or parallel machines and because no derivatives are required. We refer the interested reader to [7] for details on the parallel direct search algorithms.

Because we have an implementation of PDS ([16]) that will allow us to define a pattern matrix P_k that we can also use for the problem defined by $f(x, M^*(x))$ and that returns a step s_k of a form consistent with our requirement that s_k be defined by $\delta_k P_k$, our initial testing will be done using some number of steps of PDS applied to the model problem to satisfy the "appropriate decrease on $f(x, M^\alpha(x))$ from x_k" discussed above. We will use the variable i_k to give an upper bound on the number of iterations of PDS allowed to produce a step s_k and then modify i_k to govern whether more or less optimization should be allowed on the model problem. We will modify i_k by assessing both how many successful steps were completed for the model problem and how well the predicted improvement matched the actual reduction seen for $f(x, M^*(x))$ from x_k.

At this juncture we include some specific termination conditions: we will stop the search when either the step length falls below a user-prescribed tolerance *tol* or the total number of full evaluations of the expensive simulations M^* reaches a user-defined limit *maxfevals*. For simplicity, δ_k is used to judge the length of the steps and *fevals* is used to count the number of of expensive simulations.

Exploratory Moves

Given M^*, M^{α_k}, P_k, x_k, $f(x_k, M^*(x_k))$, δ_k, i_k, γ, *tol*, *fevals*, and *maxfevals*:

```
1. Set $s_k = 0$. Set $j = 0$, $P' = P_k$, $\delta' = \delta_k$, $x' = x_k$, and $f' = f(x_k, M^*(x_k))$.

2. Repeat

   (a) Using $P'$ and $\delta'$, take one iteration of PDS on $f(x, M^{\alpha_k}(x))$ from $x'$ to produce a
       step $s'$ such that either $f(x' + s', M^{\alpha_k}(x' + s')) < f'$ or $s' = 0$.
   (b) Set $s_k = s_k + s'$.
   (c) If $(f(x_k + s_k, M^{\alpha_k}(x_k + s_k)) < f(x_k, M^*(x_k)))$, then
           Set $j = j + 1$.
           Set $f' = f(x' + s', M^{\alpha_k}(x' + s'))$.
           Set $x' = x' + s'$.
           Update $P'$ and $\delta'$.
       else
         i. Order points defined by the core pattern $v_i = x_k + s_k^i$, $i = 1, 2n$ based on
            $f(v_i, M^{\alpha_k}(v_i))$, so that $f(v_1, M^{\alpha_k}(v_1)) \leq \cdots \leq f(v_{2n}, M^{\alpha_k}(v_{2n}))$.
        ii. Set $i = 0$.
            Repeat
                Set $i = i + 1$.
                If $f(v_i, M^*(v_i))$ has not yet been evaluated, then
                    Compute $f(v_i, M^*(v_i))$.
                    Set $fevals = fevals + 1$.
                    Consider updating the model $M^{\alpha_k}$ and reordering $v_{i+1}, \ldots, v_{2n}$ so that
                    $f(v_{i+1}, M^{\alpha_k}(v_{i+1})) \leq \cdots \leq f(v_{2n}, M^{\alpha_k}(v_{2n}))$.
                endif.
            until $((f(v_i, M^*(v_i)) < f(x_k, M^*(x_k))$ OR $(i = 2n)$ OR $(fevals = maxfevals))$.
       iii. If $(f(v_i, M^*(v_i)) < f(x_k, M^*(x_k))$ then
                Set $x_k = v_i$ and $f(x_k, M^*(x_k)) = f(v_i, M^*(v_i))$.
                Set $x' = v_i$ and $f' = f(v_i, M^*(v_i))$.
                Update $P'$.
            else, if ($i = 2n$), then
                Set $\delta_k = \gamma\delta_k$.
                Set $\delta' = \delta_k$.
            endif.
        endif.
    until (($j = i_k$) $OR$ ($\delta' < \delta_k$) $OR$ ($fevals = maxfevals$) $OR$ ($\delta_k < tol$)).
3. Set $x_{k+} = x_k + s_k$. Set $\delta_{k+1} = \delta_k$.
```

The main goal of the Exploratory Moves algorithm is to use PDS on the model M^{α_k} to produce a nonzero step s_k at which the value of $f(x_k + s_k, M^*(x_k + s_k))$ is *predicted* by $f(x_k + s_k, M^{\alpha_k}(x_k + s_k))$ to be strictly less than the value of $f(x, M^*(x))$ at the current iterate x_k. The only conditions under which the Exploratory Moves algorithm would fail to do so are when one of the termination criteria set by the user has been met.

An auxiliary goal of the Exploratory Moves algorithm is to avoid the expensive calculation of $M^*(x)$ unless necessary to further progress on the optimization. If the model M^{α_k} can be used to produce a nonzero step s_k, then the Exploratory Moves algorithm will not require a computation of $M^*(x)$—that is left to the algorithm given below to assess the progress of the search. However, should the model fail to produce such a step for the given value of δ_k, then the purpose of Steps 2(c)i–2(c)iii is to enforce the fallback strategy required to satisfy the second condition on the Exploratory Moves discussed in §4.1. The fallback strategy may require additional expensive simulations, but we ask for these as sparingly as possible.

Approximation models with pattern search

```
Given $M^*$, $M^\alpha$, $x_0$, $y_0^*$, $\delta_0$, $i_0$, $0 < \gamma < 1$, and $0 < \eta_1 < \eta_2 < 1$,
(possible choices for these constants are $\gamma = 0.5$, $\eta_1 = 10^{-4}$, and $\eta_2 = 0.75$);
for $k = 0, 1, \cdots$, do

    1. Check for convergence. Otherwise, continue.

    2. Apply the Exploratory Moves to find an $x_{k+}$ for which $f(x_{k+}, y_{k+}^{\alpha_k}) \le f(x_k, y_k^*)$.
       Compute $pred_k \equiv f(x_{k+}, y_{k+}^{\alpha_k}) - f(x_k, y_k^*)$ (the predicted reduction).

    3. If $x_{k+} \ne x_k$ then

        (a) Compute $y_{k+}^* = M^*(x_{k+})$. Set $fevals = fevals + 1$.
        (b) Compute $ared_k \equiv f(x_{k+}, y_{k+}^*) - f(x_k, y_k^*)$ (the actual reduction).
        (c) Compute $r_k \equiv \frac{ared_k}{pred_k}$.

       else $r_k = 0$.

    4. If $r_k \le 0$ (improvement was predicted but not achieved), then

            Set $x_{k+1} = x_k$ and $y_{k+1}^* = y_k^*$. (Reject the step.)
            Set $i_{k+1} = \max\{1, j - 1\}$. (Allow less optimization at the next iteration.)
        Consider refining the model.

    Else, if $0 < r_k < \eta_1$ (much more improvement was predicted than achieved), then

        Set $x_{k+1} = x_{k+}$ and $y^*_{k+1} = y^*_{k+}$. (Accept the step.)

        Set $i_{k+1} = j$. (Allow no more optimization then was successful at this iteration
        for the next iteration.)

        Consider keeping the current approximation model.

    Else, if $\eta_1 \leq r_k < \eta_2$ (prediction was satisfactory), then

        Set $x_{k+1} = x_{k+}$ and $y^*_{k+1} = y^*_{k+}$. (Accept the step.)

        Set $i_{k+1} = i_k$. (Allow the same upper bound on the number of optimization steps
        allowed at the next iteration.)

        Consider keeping the current approximation model.

    Else, $\eta_2 \leq r_k$ (prediction was excellent, or more decrease was obtained than predicted)

        Set $x_{k+1} = x_{k+}$ and $y^*_{k+1} = y^*_{k+}$. (Accept the step.)

        Set $i_{k+1} = i_k + 1$. (Allow more optimization at the next iteration.)

        Consider using a *less* accurate approximation model.

5. Update the pattern $P_k$ (if $r_k \leq 0$ then set $P_{k+1} = P_k$, else set $P_{k+1} = P'$) and return
   to Step 1.
```

6 Conclusions and future work

We have presented an outline for a general framework to manage the use for optimization of approximation models of expensive simulations. Our approach is based on the highly successful trust-region framework for global convergence to local solutions of nonlinear optimization problems and the structure and global convergence theory of pattern search methods. This leads us to believe that we have found a path to effective practical algorithms that we will be able to support by rigorous convergence theorems. Furthermore, we believe that our work is compatible with the engineering insight that has gone into the vast literature on approximation models.

We are not seeking new ways to construct approximation models. Rather we are interested in designing effective, robust optimization algorithms that make use of these models. We expect our contribution to the modeling effort to be strategies indicating when model refinement is needed. These conditions will be developed to ensure that the accompanying optimization algorithms progress toward a minimizer for the objective based on the detailed analyses.

In future work, we will implement, test, and refine our strategy using the pattern search algorithms given here. We also intend to test variants that use trust-region algorithms to do the model optimization. The convergence analysis for these algorithms with the double layer of modeling will be more delicate, but should be possible.

We also will consider ways to proceed when $M^*(x)$ is not some detailed analysis code, but rather $y^*(x_{k+})$ is approximated in some more case-specific manner. This seems to us to be extremely important for identification, design, and control. A fairly standard technique in the numerical solutions of differential equations is to manage the discretization adaptively by comparing values of discretized solutions at a point from successive refinements of the mesh. We suggest that, since we are just trying to move the optimization along, we really care about $f(x_{k+}, y^*(x_{k+}))$ and not y^*_{k+}. Thus, we will test the notion of approximating y^*_{k+} by comparing successive values of $f(x_{k+}, y^{\alpha'}(x_{k+}))$ and $f(x_{k+}, y^{\alpha''}(x_{k+}))$ computed using the family of models we are given. We emphasize, however, that this procedure for approximating y^*_{k+} is conceptually separate from decisions in Step 4 of the model management framework concerning the selection of a model to generate trial steps for x. One of our first goals in this direction will be to recast the algorithm of [21] in the general framework for the management of approximation models given above in §2.

Acknowledgments

The authors thank David Serafini and Michael Trosset for their many careful comments and useful observations, particularly during the development of the specific algorithm given in §5. We also thank Natalia Alexandrov for the invitation to present this work at the ICASE/LaRC Workshop on Multidisciplinary Optimization.

References

[1] J.-F. M. Barthelemy and R. T. Haftka, *Recent advances in approximation concepts for optimum structural design*, Tech. Report 104032, NASA Langley Research Center, Hampton, Virginia 23665–5225, March 1991.

[2] A. J. Booker, *DOE for computer output*, Tech. Report BCSTECH–94–052, Boeing Computer Services, Research and Technology, M/S 7L–68, Seattle, Washington 98124–0346, December 1994.

[3] R. H. Byrd, R. B. Schnabel, and G. A. Shultz, *A trust region algorithm for nonlinearly constrained optimization*, SIAM Journal on Numerical Analysis, 24 (1987), pp. 1152–1170.

[4] R. G. Carter, *Multi-Model Algorithms for Optimization*, Ph.D. thesis, Rice University, 1986. Available as TR86–03, Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77251–1892.

[5] J. E. Dennis, M. El-Alem, and M. C. Maciel, *A global convergence theory for general trust–region–based algorithms for equality constrained optimization*, Tech. Report CRPC–TR92247, Center for Research in Parallel Computation, Rice University, 1992. Revised September 1994. Submitted for publication.

[6] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983. See §6.4.3, pp. 143–147.

[7] J. E. Dennis, Jr. and V. Torczon, *Direct search methods on parallel machines*, SIAM J. Optimization, 1 (1991), pp. 448–474.

[8] J. E. Dennis Jr., M. Heinkenschloss, and L. Vicente, *Trust-region interior-point algorithms for a class of nonlinear programming problems*, Tech. Report TR94–45, Rice University, Department of Computational and Applied Mathematics, Houston, Texas, 1994. Submitted for publication.

[9] J. E. Dennis Jr., S. B. Li, and R. A. Tapia, *A unified approach to global convergence of trust-region methods for nonsmooth optimization*, Mathematical Programming, 68 (1995), pp. 319–346.

[10] J. E. Dennis Jr. and L. Vicente, *On the convergence theory of trust-region-based algorithms for equality constrained optimization*, Tech. Report CRPC–TR94478, Rice University, Department of Computational and Applied Mathematics, Houston, Texas, 1994. Submitted for publication.

[11] P. D. Frank, *Brief description of the helicopter rotor blade design project.* Informal report, September 1994.

[12] J. J. Moré, *Recent developments in algorithms and software for trust region methods*, in Mathematical Programming, The State of the Art, A. Bachem, M. Grotschel, and G. Korte, eds., Springer–Verlag, 1983, pp. 256–287.

[13] J. J. Moré and D. C. Sorensen, *Computing a trust region step*, SIAM Journal on Statistical and Scientific Computing, 4 (1983), pp. 553–572.

[14] M. J. D. Powell, *Convergence properties of a class of minimization algorithms*, in Nonlinear Programming 2, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds., Academic Press, 1975, pp. 1–27.

[15] H. Schramm and J. Zowe, *A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results*, SIAM J. Optimization, 2 (1992), pp. 121–152.

[16] V. Torczon, *PDS: Direct search methods for unconstrained optimization on either sequential or parallel machines*, Tech. Report 92–9, Department of Mathematical Sciences, Rice University, Houston, TX 77251–1892, 1992. Submitted to *acm Transactions on Mathematical Software.*

[17] ——, *On the convergence of pattern search methods*, Tech. Report 93–10, Department of Computational and Applied Mathematics, Rice University, Houston, TX 77251–1892, 1993. Under revision for *SIAM Journal on Optimization.*

[18] G. N. Vanderplaats, *Numerical Optimization Techniques for Engineering Design: With Applications*, McGraw–Hill, Inc., New York, 1984.

[19] W. J. Welch, R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell, and M. D. Morris, *Screening, predicting, and computer experiments*, Technometrics, 34 (1992), pp. 15–25.

[20] S. Yesilyurt and A. T. Patera, *Surrogates for numerical simulations; optimization of eddy-promoter heat exchangers*, Tech. Report 93–50, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA 23681-0001, August 1993. Submitted to *Computer Methods in Applied Mechanics and Engineering.*

[21] D. P. Young, W. P. Huffman, R. G. Melvin, M. B. Bieterman, C. L. Hilmes, and F. T. Johnson, *Inexactness and global convergence in design optimization – AIAA 94-4386.* Proceedings of the Fifth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, September 1994. Panama City, Florida.