

Multi-Level Modeling for Engineering Design Optimization*

Thomas Ellman John Keane Mark Schwabacher Ke-Thia Yao
Department of Computer Science, Hill Center for Mathematical Sciences
Rutgers University, Piscataway, NJ 08855
{ellman,keane,schwabac,kyao}@cs.rutgers.edu

Abstract

Physical systems can be modeled at many levels of approximation. The right model depends on the problem to be solved. In many cases, a combination of models will be more effective than a single model alone. Our research investigates this idea in the context of engineering design optimization. We present a family of strategies that use multiple models for unconstrained optimization of engineering designs. The strategies are useful when multiple approximations of an objective function can be implemented by compositional modeling techniques. We show how a compositional modeling library can be used to construct a variety of locally calibratable approximation schemes that can be incorporated into the optimization strategies. We analyze the optimization strategies and approximation schemes to formulate and prove sufficient conditions for correctness and convergence. We also report experimental tests of our methods in the domain of sailing yacht design. Our results demonstrate dramatic reductions in the CPU time required for optimization, on the problems we tested, with no significant loss in design quality.

* *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 11, 5, 1997.

1 Introduction

Physical systems can be modeled at many levels of approximation. For example, the behavior of a sailing yacht can be modeled in terms of steady-state motion or time-dependent motion. Furthermore, the forces acting on a sailing yacht can be modeled by simple algebraic equations, or by computational fluid dynamics. The right model will depend, in general, on the problem to be solved. For example, the sort of model needed to synthesize a new design may be different from the sort needed to accurately predict the performance of a single proposed design or to diagnose problems with an existing design. Even in the context of a particular problem-solving task, a single model may not suffice. Problem solving may proceed in stages, with different modeling requirements at each stage. Given a problem whose solution requires a computational model of a physical artifact, an intelligent system must choose one or more suitable models from the range of all models possible, and use them in a coordinated fashion to solve the problem.

A number of investigators have recently developed knowledge-based methods for using multiple models to reason about physical systems. A technique known as “compositional modeling” has emerged as the dominant paradigm for this research [Falkenhainer and Forbus, 1991]. In compositional modeling, each model is constructed from a library of model fragments. An entire space of models results from systematically enumerating combinations of model fragments. A number of methods have been proposed for utilizing such a space of models to carry out problem-solving tasks. Most of this work has focused on problems whose solution involves simulation of a single artifact. Very little effort, if any, has gone into investigating the use of compositional modeling in a design process that involves searching through a space of artifacts.

We are investigating the use of multiple-models in the specific context of computer-aided design optimization. Design optimization involves search through a space of artifacts. This presents a special opportunity for an intelligent system to exploit multiple models of an artifact. An optimization algorithm can often utilize relatively simple models to make search control decisions, and rely on complex models only when needed to verify optimality of a solution and satisfaction of constraints. For this reason, a combination of simple and complex models can lead to designs as good as those resulting from a single complex model, but at far lower computational cost.

We have developed a family of optimization strategies that are based on this insight. For example, one strategy uses a simple model to get near an optimum, and falls back on a complex model only during the last stage of optimization. Another strategy interleaves optimization with periodic calibration of an approximate model. A third strategy is especially useful for problems with multiple optima. It uses an approximate model to find the basin of attraction of the global optimum. It then uses the exact model to close in on the optimum.

Our research may be understood in the larger context of previous work in Artificial Intelligence. A large body of research has investigated the use of multiple levels of modeling in problem-solving activities. These include the use of abstraction in planning [Sacerdoti, 1974], decomposition in design [Simon, 1981] and approximation in state-space search [Pearl, 1984], among many others. We are attempting to extend this body of research to the arena of engineering design optimization.

We will begin, in section 2, by describing our testbed domain and reviewing some of the modeling choices that arise in that domain. In section 3, we will introduce our family of multi-model optimization strategies. In section 4, we will formulate and prove theorems about the correctness

and convergence of these strategies. In section 5, we will present results from experimental tests of the strategies in the testbed domain. In section 6, we will discuss several areas of related work, both within and outside the Artificial Intelligence community. Finally, in section 7, we will summarize our contributions.

2 The Sailing Yacht Design Problem

The domain of sailing yacht design is serving as a testbed for our research on multi-level modeling. As a case study, we are attempting to reconstruct the process that led to the design of the “Stars and Stripes ’87”. A picture of the hull of this yacht is shown in Figure 1. The hull includes a “canoe body” (the main part of the hull); a “keel” (the appendage descending from the bottom of the canoe body) and the “winglets” (the appendages attached to the bottom of the keel). The Stars and Stripes ’87 won the America’s Cup yacht race in 1987 [Letcher *et al.*, 1987]. Racing yachts are normally designed to meet objectives involving speed, rating and cost, among others. Speed is defined by the time required for a yacht to sail around a specified race course in specified wind conditions. Rating refers to a set of constraints on the hull and sails of a yacht that must be satisfied before the yacht can be admitted into a particular racing class. In our research, we have chosen to focus on objectives of speed and rating, ignoring considerations of cost. We use a “Velocity Prediction Program” (VPP) to evaluate yacht designs. VPP is a program that we wrote ourselves, based on a commercial software package for evaluating yacht designs [Letcher, 1991]. The organization of VPP is described in Figure 2. VPP takes as input a vector $\bar{x} = (x_1, \dots, x_n)$ of design parameters. In our current work, we are using a five-parameter design space, including *Length*, *Beam* and *Draft* (dimensions of the canoe body); as well as *KeelHeight* and *WingletSpan*. VPP uses these parameters to generate B-Spline surfaces representing the geometry of the yacht hull. It begins by using the “hull processing models” to extract critical quantities impacting the speed of the yacht, e.g., wave resistance (Rw), friction resistance (Rf), effective draft (D), vertical center of gravity (Vcg) and vertical center of pressure (Zcp), among others. VPP then uses these quantities in a “velocity prediction model” to set up nonlinear equations describing the balance of forces and torques on the yacht. The velocity prediction model uses an iterative method to solve these equations and thereby determine the “velocity polar”, i.e., a table giving the steady-state velocity of the yacht under various wind speeds and directions of heading. Finally, the “race model” uses the velocity polar to determine the total time to traverse the given course, assuming the given wind speed. The rating constraint is handled implicitly by the VPP program. When given a set of hull geometry parameters, VPP automatically computes the size of the largest sail that will satisfy the rating constraint. By so incorporating the rating constraint into the VPP program, we formulate the yacht design problem as unconstrained minimization of the course time function $T(\bar{x})$.

Numerical optimization codes offer a direct method of attempting to solve the yacht design problem. One such code is CFSQP [Lawrence *et al.*, 1995]. This code solves constrained nonlinear optimization problems: Given an objective function $e(\bar{x})$; a collection of inequality constraints $g_i(\bar{x}) \leq 0$ ($i = 1 \dots M$); a collection of equality constraints $h_i(\bar{x}) = 0$ ($i = 1 \dots N$), and a seed design \bar{s} , CFSQP attempts to find a value of \bar{x} that optimizes (minimizes or maximizes) $e(\bar{x})$ while satisfying each constraint. CFSQP is a Quasi-Newton method. It operates by repeatedly constructing a quadratic approximation to the Lagrangian of the functions $e(\bar{x})$, $\{g_i(\bar{x})|i = 1 \dots M\}$ and $\{h_i(\bar{x})|i = 1 \dots N\}$, in the neighborhood of a point \bar{x}_i , and subsequently optimizing the quadratic function to

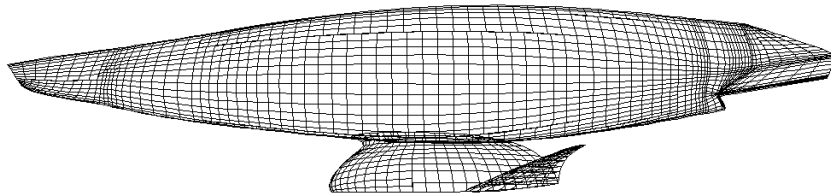


Figure 1: The Stars and Stripes '87 Hull

obtain a new point \bar{x}_{i+1} . The iteration process is repeated until convergence is achieved. The method is “quasi” because the (inverse) Hessian of $e(\bar{x})$ needed to construct the quadratic approximation is not computed directly. It is approximated by a series of gradients $\nabla e(\bar{x})$ generated as the iteration process moves through the design space.

In our yacht application, we use CFSQP to minimize the course time function $T(\bar{x})$. Numerical minimization of this function is difficult for several reasons. The first difficulty is the cost of evaluation. Each computation of $T(\bar{x})$ requires about one hour of CPU time, if one elects to use the most exact model available. A typical optimization from a single seed in a five dimensional space using CFSQP requires about 100 evaluations of the objective function, i.e., about 100 CPU hours to conduct an optimization. An even greater difficulty results from the unreliability of optimization starting from a single seed. In the yacht domain, the course time function $T(\bar{x})$ exhibits numerous pathologies, such as ridges and discontinuities, resulting from numerical discretization used in VPP, among other things. Furthermore, $T(\bar{x})$ is unevaluatable throughout large regions of the design space, for example, because the balance of force equations fail to be solvable. Our system catches such failures and arranges for extremely bad values of the objective function to be returned to CFSQP. This allows optimization to continue when such failures are encountered; however, it introduces sharp discontinuities into the objective function. These pathologies make optimization from a single seed extremely unreliable, even for finding a local optimum [Gill *et al.*, 1981]. A commonly used remedy is to conduct many optimizations, from randomly selected seeds; however, this approach dramatically raises the already high cost of optimization. In order to make reliable yacht design optimization feasible, some alternative must be found.

Approximation methods offer a means of overcoming the difficulties of direct optimization. Especially fruitful opportunities for approximation arise in the context of local optimization problems. Consider that in order to verify a local optimum, a search algorithm need only obtain accurate evaluations of the objective function, or its gradient, in the neighborhood of a solution. Evaluations that occur along a path toward a solution need only be accurate enough to guide the search. Newton and Quasi-Newton methods like CFSQP exploit approximations in just this fashion. They use quadratic approximations of the objective function to guide search along the path toward a solution. They fall back on the exact objective function to test for convergence.

Purely numerical methods, like CFSQP, are not always able to construct the most cost-effective approximations available. They are limited to treating the objective function as a “black box”, i.e., they use it to evaluate designs, but they do not look inside to examine its internal structure. More cost-effective approximations often result from exploiting the internal structure of the objective function. For example, consider the structure of VPP shown in Figure 2. Instead of approximating

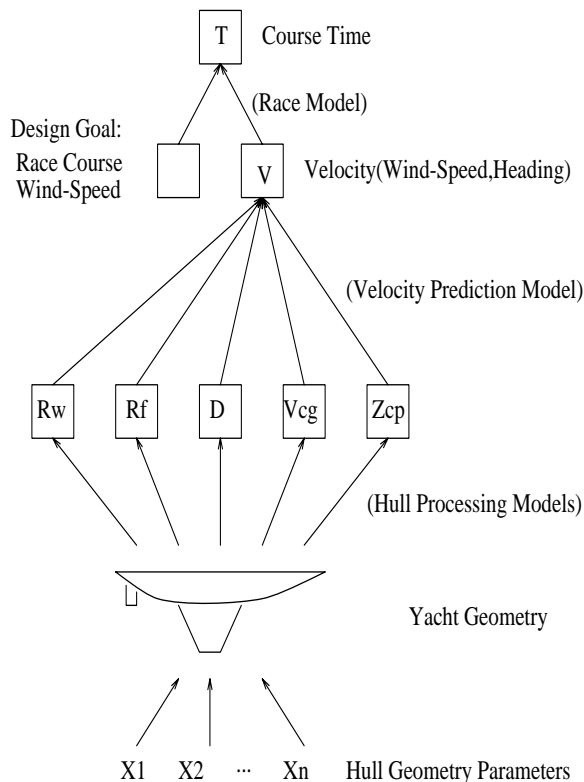


Figure 2: Velocity Prediction Program

the entire function $T(\bar{x})$, (as CFSQP does), one might choose to approximate a module that computes an intermediate quantity such as wave resistance R_w or effective draft D . These two modules are especially good candidates for approximation because they both involve time-consuming computational fluid-dynamics (CFD) codes. For example, effective draft D can be computed using a potential-flow code called “PMARC” [Katz and Plotkin, 1991], [Ashby *et al.*, 1992]. It can also be approximated by a simple algebraic formula. Likewise, wave resistance R_w can be computed using a slender-body flow code called “SLAW” [Weems *et al.*, 1994]. It can also be approximated by a formula that interpolates data from wave tank tests. Such *internal approximations* are not available to purely numerical optimization methods. They can only be constructed by a system that has access to the internal structure of the objective function.

Consider the choice between the PMARC potential flow code and an algebraic approximation for computing effective draft D . Effective draft is a measure of the amount of drag produced by the keel as a result of the lift it generates. An accurate estimate of this quantity is quite important for analyzing the performance of a sailing yacht. Although PMARC is the most accurate means of estimating effective draft, it can also be estimated using an algebraic approximation that involves the maximum keel depth S and the midship cross section area A , as follows:

$$D = K \sqrt{S^2 - 2A/\pi}$$

This formula is based on an approximation that treats a sailing yacht hull as an infinitely long cylinder and treats the keel of the yacht as an infinitely thin fin protruding from the cylinder [Newman and Wu, 1973],[Letcher, 1975]. The constant K may be chosen to fit the algebraic model

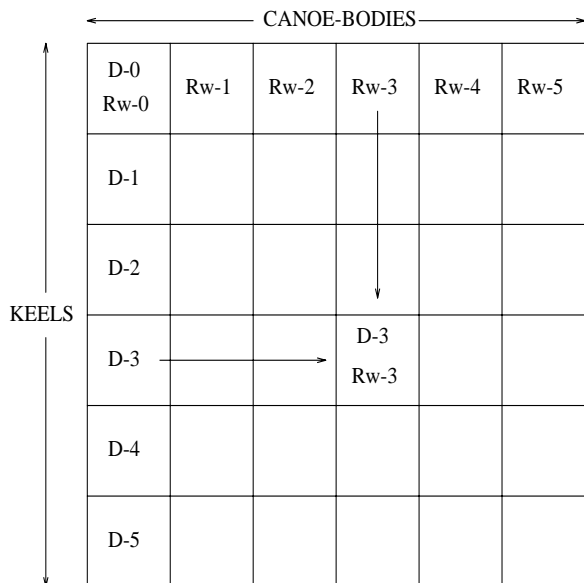


Figure 3: Reuse of Prior Results

to data obtained from wave tank tests, or sample runs using PMARC. (When the formula is fit to PMARC a value of $K = 0.83$ results.) Although the algebraic approximation is comparatively easy to use, its results are not as accurate as those produced by PMARC.

Another internal approximation involves the reuse of results from prior evaluations in the course of an optimization process. Some physical quantities may not change significantly when a design is modified. Values of such quantities can be retrieved from prior candidate designs. For example, suppose an algorithm is systematically exploring combinations of canoe-bodies and keels of a sailing yacht. (See Figure 3.) In principle, VPP must evaluate the wave resistance Rw and the effective draft D of each candidate design. Wave resistance depends mainly on the canoe-body of the yacht and is not significantly influenced by the keel. When only the keel is modified, wave resistance will not change much. Instead of recomputing wave resistance for the new yacht, the system can reuse the prior value. On the other hand, effective draft depends mainly on the keel of the yacht and is not significantly influenced by the canoe-body. When only the canoe-body is modified, effective draft will not change much. Instead of recomputing effective draft for the new yacht, the system can reuse the prior value. In fact, the entire matrix of yachts can be evaluated by computing wave resistance for a single row and computing effective draft for a single column. By intelligently deciding when to reuse prior evaluation results, one can significantly lower the computational costs of design.

3 Multi-Model Optimization Strategies

A family of strategies for unconstrained multi-model optimization is presented in Figure 4. Each strategy (potentially) takes as input a seed design representing a starting point for the design optimization process. Each returns a new design that results from using a nonlinear optimization code, such as CFSQP, in combination with exact and approximate versions of the objective function. Definitions of these are shown in Figure 5. The function $e(\bar{x})$ is the so-called exact objective function. The function $\tilde{e}(\bar{x})$ is a fixed approximation of $e(\bar{x})$. The approximation $\tilde{e}(\bar{x})$ is called “fixed” because

the function \tilde{e} itself is not modified during the optimization process. In contrast, the function $\hat{e}(\bar{x}, \bar{y})$ is a locally calibratable approximation. It can be modified during the optimization process.

3.1 Compositional Modeling of Approximate Objective Functions

An approximate objective function $\tilde{e}(\bar{x})$ can be constructed from a library of model fragments using compositional modeling techniques. Our approach to compositional modeling is illustrated in Figure 6. The evaluation function $e(\bar{x})$ is defined by an expression that references functions $F_1 \dots F_n$. Each F_i computes an intermediate quantity needed in the computation of $e(\bar{x})$. For example, in the yacht domain, course time $T(\bar{x})$ is defined by an expression $S(Rw(\bar{x}), Rf(\bar{x}), D(\bar{x}), Vcg(\bar{x}), Zcp(\bar{x}))$ that references intermediate functions Rw , Rf , D , Vcg , and Zcp , which define hull processing models. (See Figure 7.) In general, each intermediate function F_i may be defined in several ways. Different definitions of F_i implement different approximations. For instance, effective draft can be defined by the function $D(\bar{x})$ that uses PMARC or by the function $\tilde{D}(\bar{x})$ that uses the algebraic formula. Likewise, wave resistance can be defined by the function $Rw(\bar{x})$ that uses SLAW or by the function $\tilde{Rw}(\bar{x})$ that uses curves fitted to tank data. In general, each version of $\tilde{e}(\bar{x})$ results from selecting a tuple from the Cartesian product of the definitions of the intermediate quantity functions. For example, in the yacht domain, $\tilde{e}(\bar{x})$ can be instantiated by any of the approximations T_1 , T_2 and T_3 that use simplified models for computing effective draft and/or wave resistance.

3.2 A Simple Two-Level Strategy

The simplest multi-model strategy (MLM_1) is based on the idea of dividing the optimization process into two phases. The first phase optimizes the fixed approximate objective function $\tilde{e}(\bar{x})$, starting at a seed \bar{s}_{old} to find a design \bar{s}_{temp} . The second phase starts at \bar{s}_{temp} and optimizes the exact objective function $e(\bar{x})$ to obtain a final design. Strategy MLM_1 can be justified by the following rationale: First, one expects that the distance from \bar{s}_{temp} to the final design will be less than the distance from \bar{s}_{old} to the final design. Second, one expects the required number of objective function evaluations to decrease with the distance from the seed to the optimum. Therefore, strategy MLM_1 should require fewer evaluations of the exact objective function $e(\bar{x})$ than are needed in a strategy of directly optimizing $e(\bar{x})$. The performance of strategy MLM_1 will thus depend, in part, on the cost-effectiveness of the approximate objective function $\tilde{e}(\bar{x})$: How close is the optimum of $\tilde{e}(\bar{x})$ to the optimum of $e(\bar{x})$? How much faster is $\tilde{e}(\bar{x})$ than $e(\bar{x})$? The performance of MLM_1 will also depend, in part, on the properties of the numeric method (e.g., CFSQP) that is used to carry out each stage of optimization. These factors will be investigated analytically in Section 4 and empirically in Section 5. It turns out that strategy MLM_1 often fails to achieve dramatic computational savings. The reason is that optimization programs typically spend most of their time conducting search very close to the final stopping point. Getting close to the stopping point therefore achieves little savings. These considerations motivate more sophisticated multi-model strategies.

3.3 Recalibration of Approximate Objective Functions

The multi-model strategy MLM_2 is based on the idea of repeatedly calibrating an approximate objective function to fit local regions of the design space. This strategy uses an objective function $\hat{e}(\bar{x}, \bar{y})$ whose domain is the Cartesian product of the original design space with itself. The parameter

Strategy $MLM_1(\bar{s}_{old})$:

1. Optimize $\tilde{e}(\bar{x})$ starting at \bar{s}_{old} to obtain \bar{s}_{temp} .
2. Optimize $e(\bar{x})$ starting at \bar{s}_{temp} to obtain and return \bar{s}_{new} .

Strategy $MLM_2(\bar{s}_{old})$:

- *InnerOptimization*(\bar{s}): Return the result of optimizing $\hat{e}_{\bar{s}}(\bar{x})$ starting at \bar{s} .
 - *LineSearch*($\bar{s}_{old}, \bar{s}_{new}$): If $|(\bar{s}_{new} - \bar{s}_{old})| \leq \delta$ then return \bar{s}_{old} otherwise if $e(\bar{s}_{new})$ is better than $e(\bar{s}_{old})$ then return \bar{s}_{new} otherwise return $LineSearch(\bar{s}_{old}, (\bar{s}_{old} + \bar{s}_{new})/2)$.
 - *Converged?*($\bar{s}_{old}, \bar{s}_{new}$): Return $|(e(\bar{s}_{new}) - e(\bar{s}_{old}))/e(\bar{s}_{old})| < \epsilon$.
1. Let $\bar{s}_{new} = InnerOptimization(\bar{s}_{old})$.
 2. If *DoLineSearch?* then let $\bar{s}_{new} = LineSearch(\bar{s}_{old}, \bar{s}_{new})$.
 3. If *Converged?*($\bar{s}_{old}, \bar{s}_{new}$) then return \bar{s}_{new} else return $MLM_2(\bar{s}_{new})$.

Strategy $MLM_3()$:

1. Randomly select seed designs: $\{\bar{s}_1, \dots, \bar{s}_n\}$.
2. For each seed \bar{s}_i , optimize $\tilde{e}(\bar{x})$ starting at \bar{s}_i to obtain \bar{t}_i .
3. Let \bar{s}_{new} be a member of $\{\bar{t}_1, \dots, \bar{t}_n\}$ that is optimal according to $\tilde{e}(\bar{x})$.
4. Return $MLM_2(\bar{s}_{new})$.

Figure 4: Strategies for Unconstrained Optimization Using Multiple Models

\bar{x} represents a design to be evaluated. The parameter \bar{y} represents a point at which calibration takes place. Strategy MLM_2 consists of two nested optimization loops. The *InnerOptimization* procedure takes a seed point \bar{s} as an argument. It begins by constructing the function $\hat{e}_{\bar{s}}(\bar{x})$, i.e., the restriction of $\hat{e}(\bar{x}, \bar{y})$ to the subspace defined by $\bar{y} = \bar{s}$. The function $\hat{e}_{\bar{s}}(\bar{x})$ is intended to be a locally accurate approximation of the exact objective function, i.e., $\hat{e}_{\bar{s}}(\bar{x}) \approx e(\bar{x})$ whenever $|\bar{x} - \bar{s}|$ is small. The *InnerOptimization* procedure then optimizes $\hat{e}_{\bar{s}}(\bar{x})$ using the seed $\bar{x} = \bar{s}$ as a starting point. The outer loop of strategy MLM_2 calls *InnerOptimization* repeatedly to generate a series of designs $(\bar{s}_0, \bar{s}_1, \dots, \bar{s}_n, \dots)$, where $\bar{s}_{i+1} = InnerOptimization(\bar{s}_i)$. The outer loop terminates when a convergence test detects that the relative improvement in the exact objective function $e(\bar{x})$ between successive iterates has fallen below a threshold.

The locally calibratable objective function $\hat{e}(\bar{x}, \bar{y})$ can be constructed from a library of model fragments using compositional modeling techniques. Examples of such constructions are shown in Figure 8. Six different types of calibratable approximation are shown. Each results from combining an approximate version ($\tilde{F}_i(\bar{x})$) and an exact version ($F_i(\bar{x})$) of some intermediate function. Functions \hat{e}_1 and \hat{e}_2 are based on zeroth and first order approximations of F_i about the calibration

- Design space: \mathfrak{R}^n
- Exact objective function: $e : \mathfrak{R}^n \rightarrow \mathfrak{R}$.
- Fixed objective approximation: $\tilde{e} : \mathfrak{R}^n \rightarrow \mathfrak{R}$.
- Locally calibratable objective approximation: $\hat{e} : \mathfrak{R}^n \times \mathfrak{R}^n \rightarrow \mathfrak{R}$.
- Exact internal function: $F_i : \mathfrak{R}^n \rightarrow \mathfrak{R}$.
- Fixed internal approximation: $\tilde{F}_i : \mathfrak{R}^n \rightarrow \mathfrak{R}$.
- Locally calibratable internal approximation: $\hat{F}_i : \mathfrak{R}^n \times \mathfrak{R}^n \rightarrow \mathfrak{R}$.

Figure 5: Definitions of Exact and Approximate Objective Functions

$$\begin{aligned}
 e(\bar{x}) &= G(F_1(\bar{x}), F_2(\bar{x}), \dots, F_n(\bar{x})) \\
 F_1 &= F_1 \mid \tilde{F}_1 \mid \hat{F}_1 \dots \\
 F_2 &= F_2 \mid \tilde{F}_2 \mid \hat{F}_2 \dots \\
 &\dots \\
 F_n &= F_n \mid \tilde{F}_n \mid \hat{F}_n \dots
 \end{aligned}$$

Figure 6: Organization of Library of Models and Submodels

point. Notice that these are purely numerical approximations. They do not actually require any predefined internal approximation \tilde{F}_i from the library of model fragments. Functions \hat{e}_3 and \hat{e}_4 are based on an assumption that \tilde{F}_i has a predictable absolute error with respect to F_i . In particular, \hat{e}_3 implements an assumption of constant absolute error, in the neighborhood of the calibration point. In contrast, \hat{e}_4 implements an assumption that the absolute error is a linear function of the design parameters. Functions \hat{e}_5 and \hat{e}_6 are based on an assumption that \tilde{F}_i has predictable relative error with respect to F_i . In particular, \hat{e}_5 implements an assumption of constant relative error, in the neighborhood of the calibration point. In contrast, \hat{e}_6 implements an assumption that the relative error is a linear function of the design parameters. Notice that functions \hat{e}_1 and \hat{e}_2 may be seen as special case versions of \hat{e}_3 and \hat{e}_4 by taking $\tilde{F}_i(\bar{x}) = 0$. Alternatively, they may be seen as special case versions of \hat{e}_5 and \hat{e}_6 by taking $\tilde{F}_i(\bar{x}) = 1$.

Consider how strategy MLM_2 might be instantiated in the domain of sailing yachts. Various different locally calibratable objective functions result from different choices of intermediate functions to play the role of F_i in Figure 8. For example, if F_i represents effective draft D , then the calibratable approximations $\hat{e}_1, \dots, \hat{e}_6$ represent different ways of periodically recalibrating the algebraic formula for effective draft using PMARC as a basis for recalibration. Likewise, if F_i represents

Exact Objective Function:

$$\begin{aligned} e(\bar{x}) &= T(\bar{x}) \\ T(\bar{x}) &= S(Rw(\bar{x}), Rf(\bar{x}), D(\bar{x}), Vcg(\bar{x}), Zcp(\bar{x})) \end{aligned}$$

Fixed Approximation Objective Function:

$$\begin{aligned} \tilde{e}(\bar{x}) &= \tilde{T}(\bar{x}) \\ \tilde{T}_1(\bar{x}) &= S(Rw(\bar{x}), Rf(\bar{x}), \tilde{D}(\bar{x}), Vcg(\bar{x}), Zcp(\bar{x})) \\ \tilde{T}_2(\bar{x}) &= S(\tilde{R}w(\bar{x}), Rf(\bar{x}), D(\bar{x}), Vcg(\bar{x}), Zcp(\bar{x})) \\ \tilde{T}_3(\bar{x}) &= S(\tilde{R}w(\bar{x}), Rf(\bar{x}), \tilde{D}(\bar{x}), Vcg(\bar{x}), Zcp(\bar{x})) \end{aligned}$$

T Computes course time from the design parameters.

S Computes course time from the results of the hull processing modules.

Rw Computes wave resistance using SLAW.

$\tilde{R}w$ Computes wave resistance using curves fitted to wave tank data.

D Computes effective draft using PMARC.

\tilde{D} Computes effective draft using the algebraic formula.

Figure 7: Definitions of Yacht Domain Functions

wave resistance Rw , then the calibratable approximations $\hat{e}_1, \dots, \hat{e}_6$ represent different ways of periodically recalibrating the curves fitted to tank data using SLAW as a basis for recalibration. Notice that in either case each approximation $\hat{e}_i(\bar{x}, \bar{y})$ includes some subexpressions that depend on the calibration parameters \bar{y} , and not on the design parameters \bar{x} . These subexpressions need only be computed at each calibration point \bar{s} , when the inner optimization procedure constructs the function $\hat{e}_{\bar{s}}(\bar{x})$. They need not be recomputed each time $\hat{e}_{\bar{s}}(\bar{x})$ is evaluated inside the inner optimization. For example, in the yacht domain examples described above, the function $F_i(\bar{x})$ (which calls PMARC or SLAW) is applied only to the calibration point \bar{s} . These CFD codes need not be invoked from within the numeric optimization code (e.g., CFSQP) invoked by the inner optimization.

The behavior of strategy MLM_2 is illustrated by the diagram in Figure 9. MLM_2 follows a path through the space on which the locally calibratable function $\hat{e}(\bar{x}, \bar{y})$ is defined, i.e., the Cartesian product of the design space and the calibration parameter space. Movement along the vertical axis corresponds to calibration or recalibration, i.e., setting the calibration parameter \bar{y} to the design parameter seed \bar{s} and constructing the function $\hat{e}_{\bar{s}}(\bar{x})$. During calibration, only the calibration parameter \bar{y} in $\hat{e}(\bar{x}, \bar{y})$ is changed. The design parameter \bar{x} remains fixed. Calibration always moves vertically to the line defined by $\bar{x} = \bar{y}$. Movement along the horizontal axis corresponds to the numeric optimization that occurs inside the *InnerOptimization* procedure. During the inner optimization, only the design parameter \bar{x} in $\hat{e}(\bar{x}, \bar{y})$ is changed. The calibration parameter \bar{y} remains

Exact Objective Function:

$$e(\bar{x}) = G(F_1(\bar{x}), \dots, F_{i-1}(\bar{x}), F_i(\bar{x}), F_{i+1}(\bar{x}), \dots, F_n(\bar{x}))$$

Fixed Approximation Objective Function:

$$\tilde{e}(\bar{x}) = G(F_1(\bar{x}), \dots, F_{i-1}(\bar{x}), \tilde{F}_i(\bar{x}), F_{i+1}(\bar{x}), \dots, F_n(\bar{x}))$$

Locally Calibratable Objective Functions:

$$\begin{aligned} \hat{e}_j(\bar{x}, \bar{y}) &= G(F_1(\bar{x}), \dots, F_{i-1}(\bar{x}), \hat{F}_{ij}(\bar{x}, \bar{y}), F_{i+1}(\bar{x}), \dots, F_n(\bar{x})) \\ \hat{F}_{i1}(\bar{x}, \bar{y}) &= F_i(\bar{y}) \\ \hat{F}_{i2}(\bar{x}, \bar{y}) &= F_i(\bar{y}) + (\bar{x} - \bar{y}) \cdot \nabla F_i(\bar{y}) \\ \hat{F}_{i3}(\bar{x}, \bar{y}) &= \tilde{F}_i(\bar{x}) + [F_i(\bar{y}) - \tilde{F}_i(\bar{y})] \\ \hat{F}_{i4}(\bar{x}, \bar{y}) &= \tilde{F}_i(\bar{x}) + [(F_i(\bar{y}) - \tilde{F}_i(\bar{y})) + (\bar{x} - \bar{y}) \cdot \nabla [F_i(\bar{y}) - \tilde{F}_i(\bar{y})]] \\ \hat{F}_{i5}(\bar{x}, \bar{y}) &= \tilde{F}_i(\bar{x}) \frac{F_i(\bar{y})}{\tilde{F}_i(\bar{y})} \\ \hat{F}_{i6}(\bar{x}, \bar{y}) &= \tilde{F}_i(\bar{x}) \left[\frac{F_i(\bar{y})}{\tilde{F}_i(\bar{y})} + (\bar{x} - \bar{y}) \cdot \nabla \left(\frac{F_i(\bar{y})}{\tilde{F}_i(\bar{y})} \right) \right] \end{aligned}$$

Figure 8: Exact and Approximate Objective Functions for Use in Multi-Model Strategies

fixed. Evaluations of the locally calibratable approximation $\hat{e}(\bar{x}, \bar{y})$ occur only at points along the solid lines in the diagram. Evaluations of the exact objective function $e(\bar{x})$, or subexpressions of $e(\bar{x})$ need occur only in the neighborhoods of calibration points, i.e., points that lie along the line $\bar{y} = \bar{x}$ and are marked by circles on the diagram. Notice that strategy MLM_2 never evaluates the exact objective function $e(\bar{x})$ inside the execution of a numeric optimization code.

Strategy MLM_2 includes an optional procedure $LineSearch(\bar{s}_{old}, \bar{s}_{new})$, which may be called after each invocation of $InnerOptimization(\bar{s})$, depending on the flag $DoLineSearch?$ specified by the user. This procedure conducts a search in the one-dimensional space defined by the input \bar{s}_{old} and the output \bar{s}_{new} of $InnerOptimization$. It generates the series of points $(\bar{t}_0, \bar{t}_1, \dots, \bar{t}_i, \dots)$ where $\bar{t}_0 = \bar{s}_{new}$ and $\bar{t}_i = (\bar{s}_{old} + \bar{t}_{i-1})/2$, for all $i > 0$. It terminates when finding a point \bar{t}_i that is better than \bar{s}_{old} according to $e(\bar{x})$, the exact objective function. In most cases the point \bar{s}_{new} generated by $InnerOptimization$ will immediately meet this criterion; however, this condition is not guaranteed, since the inner optimization uses only an approximation of the exact objective function. Notice that $LineSearch$ conducts a *satisficing* search, along the line between \bar{s}_{old} and \bar{s}_{new} . It does not conduct an *optimizing* search. This feature is motivated by a desire to minimize the use of the exact objective function $e(\bar{x})$. Observe that the two procedures $InnerOptimization$ and $LineSearch$ are analogous to two computation steps that occur in optimization codes based on Newton or Quasi-Newton methods: (1) constructing and optimizing a quadratic approximation of the objective function; (2) conducting a line search between the start and end points of the quadratic optimization [Lawrence *et al.*, 1995].

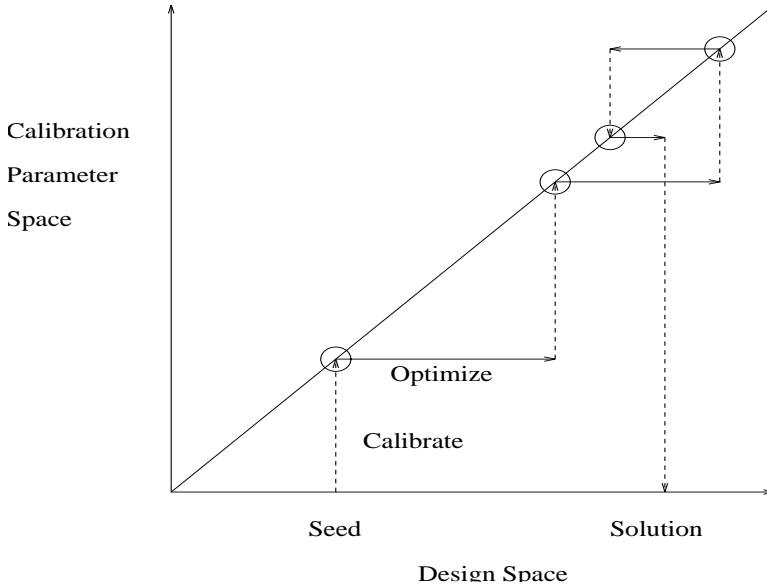


Figure 9: Path Followed by Interleaved Optimization and Calibration

The line search procedure is especially important in applications, such as the yacht domain, in which the code that evaluates the exact objective function $e(\bar{x})$ may fail to return a value in some regions of the design space. The local approximation $\hat{e}_{\bar{s}}(\bar{x})$ may move the location of the unevaluable region by a small amount. The design \bar{s}_{new} returned by *InnerOptimization* may actually be unevaluable using the exact objective function $e(\bar{x})$. (In our system a call to $e(\bar{x})$ would return an extremely bad numerical value.) This unfortunate situation tends to arise when the optimal design is obtained by pushing the design up against physical constraints that are not explicit in the problem statement, but which must be met in order for $e(\bar{x})$ to be evaluable. For example, in the yacht domain, the balance of force and torque equations must be solvable in order to compute steady state velocity on each leg of the race course. Near the optimum, it often happens that equations are nearly unsolvable for one leg of the course. This occurs because the design is well optimized for other legs of the course and is just barely able to sail at all on the leg that results in a near failure of the objective function. In situations like this, the *LineSearch* procedure serves to move \bar{s}_{new} back into the evaluable region of the design space. In all of the experiments we report in this paper, the flag *DoLineSearch?* was set to *True* so that the *LineSearch?* routine was invoked after each inner optimization. In experiments on an aircraft design problem, and a simpler version of the yacht design problem, we obtained good results without using the line search routine [Ellman *et al.*, 1996].

A possible modification to strategy MLM_2 is suggested by analogy with “trust region” optimization techniques [Dennis and Schnabel, 1983]. A trust region method fits an approximation $\hat{e}_{\bar{s}}(\bar{x})$ to the objective function $e(\bar{x})$ at the design point $\bar{s} = \bar{s}_{old}$. It then optimizes $\hat{e}_{\bar{s}}(\bar{x})$ over a hypersphere centered at s_{old} to obtain a new design s_{new} . The hypersphere is chosen to be a region over which $\hat{e}_{\bar{s}}(\bar{x})$ is a good approximation of $e(\bar{x})$. The optimum s_{new} is then used to define a direction $s_{new} - s_{old}$ for a line search. Strategy MLM_2 could be modified to use this approach. Suppose we have an estimate δ of the size of a region over which the local approximation $\hat{e}_{\bar{s}}(\bar{x})$ is valid. We obtain a trust region technique by modifying the inner optimization routine so that it optimizes

$\hat{e}_{\bar{s}}(\bar{x})$ subject to the constraint $|\bar{x} - \bar{s}| \leq \delta$.

A number of interesting variations on strategy MLM_2 arise when the locally calibratable approximation $\hat{e}(\bar{x}, \bar{y})$ involves two or more internal approximations. For example, if two internal functions are approximated and recalibrated, then $\hat{e}(\bar{x}, \bar{y})$ has the form:

$$\hat{e}_{kl}(\bar{x}, \bar{y}) = G(F_1(\bar{x}), \dots, \hat{F}_{ik}(\bar{x}, \bar{y}), \dots, \hat{F}_{jl}(\bar{x}, \bar{y}), \dots, F_n(\bar{x}))$$

The function $\hat{e}_{kl}(\bar{x}, \bar{y})$ uses scheme number k for approximating F_i and uses scheme number l for approximating F_j . (See Figure 8.) Suppose, for example, that F_i is wave resistance Rw and F_j is effective draft D . Suppose further that we use $\hat{e}_{22}(\bar{x}, \bar{y})$, i.e., we fit linear functions to $Rw(\bar{x})$ and $D(\bar{x})$ at calibration points. A question arises regarding how to coordinate the recalibration of each of these linear approximations. If both internal approximations are accurate over roughly the same region, it may pay to use strategy MLM_2 directly and recalibrate both after each inner optimization. Suppose on the other hand that one internal approximation is valid over a wider region than the other. It may be useful to use a doubly nested strategy in which one approximation is calibrated at the beginning of each inner optimization, and the other is calibrated at the beginning of each “inner-inner” optimization. On the other hand, the two internal approximations may be accurate in different directions. For example, an approximation of effective draft is likely to remain accurate when the canoe body is changed and the keel is fixed. Likewise, an approximation of wave resistance is likely to remain accurate when the keel is changed and the canoe body remains fixed. This suggests using a scheme for approximating each internal quantity in which the approximation is constant in one subspace and linear in the complementary space. The resulting strategy will behave in a manner similar to that illustrated in Figure 3.

3.4 Dealing with Pathological Objective Functions

The last multi-model strategy, MLM_3 , may be seen as a combination of the simple two-level strategy, MLM_1 , and the recalibrating strategy, MLM_2 . It is useful when the objective function has pathologies such as discontinuities, ridges, unevaluable points and multiple-local optima — features that prevent numerical optimization codes from reliably reaching global, or even local, optima. Strategy MLM_3 is based on the hope that optimization using $\tilde{e}(\bar{x})$ from multiple seeds can find a design lying in the basin of attraction of the true global optimum, even though it does not produce a true global (or local) optimum itself. MLM_3 thus begins by selecting a random set of seed designs. It optimizes each using $\tilde{e}(\bar{x})$, the fixed approximation. This results in a set of “approximate optima”. MLM_3 then selects one of the approximate optima that is best according to $\tilde{e}(\bar{x})$ and uses this as a seed for strategy MLM_2 .

Strategy MLM_3 is particularly useful in the yacht domain. In this application, pathological features of the objective function are a major problem. Unreliable optimization, due to ridges, discontinuities and unevaluable regions, can result in a dramatic loss in design quality — even greater than losses that result from using approximations like the algebraic formulae for effective draft and wave resistance. Consider that many optimizations using the algebraic approximations can be run in the time it takes to conduct a single optimization using the CFD codes. A computational advantage therefore results from conducting optimizations from many seeds using the algebraic formulae, and then using the best result as a seed for a single optimization using CFD codes.

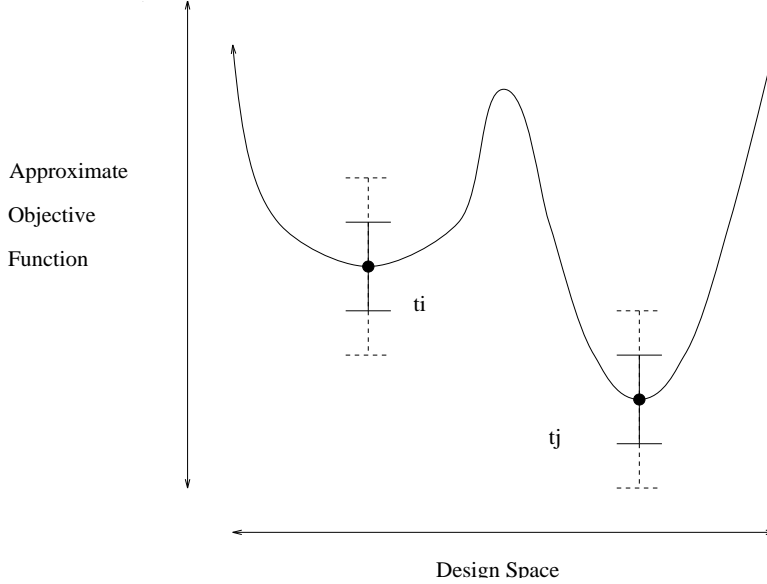


Figure 10: Use of Error Bars in Selecting Among Seeds

3.5 Using Error Estimates to Guide Model Selection

Estimates of the error of approximate models may be available from a variety of sources. For example, in the yacht domain, the error of the algebraic effective draft formula \tilde{D} may be obtained as a byproduct of the process of fitting the constant K that appears in the formula, whether the formula is fit to data from wave tanks or to data from sample runs of PMARC. (When the formula is fit to PMARC, the root mean square error ϵ is 0.51 feet. A typical effective draft is about 10.5 feet.) The error in effective draft $D(\bar{x})$ can be converted into an error in the course time function $T(\bar{x})$. If the true effective draft lies in the interval $[\tilde{D} - \epsilon, \tilde{D} + \epsilon]$, then the true course time lies in the interval $[S(Rw, Rf, (\tilde{D} + \epsilon), Vcg, Zcp), S(Rw, Rf, (\tilde{D} - \epsilon), Vcg, Zcp)]$. This conclusion is based on the fact that course time is a monotonically decreasing function of effective draft. The error in course time $T(\bar{x})$ corresponding to an error of ϵ in effective draft will depend on the specific design \bar{x} , as well as the environment conditions, i.e., race course and windspeed.

When error estimates are available, they can be used to make refinements in the multi-model optimization strategies described above. The first refinement involves strategy MLM_3 , which conducts multiple optimizations using the fixed approximation $\tilde{e}(\bar{x})$ to find a good starting point for the recalibrating strategy MLM_2 . (See Figure 4.) Notice that strategy MLM_3 uses the fixed approximation $\tilde{e}(\bar{x})$ to select among the designs $\{\bar{t}_1, \dots, \bar{t}_n\}$ resulting from the first stage of optimization. This may not result in choosing a seed for the second stage that is best according to the exact objective $e(\bar{x})$. An alternative approach could use the exact objective function $e(\bar{x})$ to choose among the designs $\{\bar{t}_1, \dots, \bar{t}_n\}$; however, this might be too computationally expensive. A third approach is illustrated by Figure 10. When comparing two designs t_i and t_j , one uses the approximation $\tilde{e}(\bar{x})$ to evaluate and compare $\tilde{e}(\bar{t}_i)$ and $\tilde{e}(\bar{t}_j)$ and checks whether the error bars overlap. If not (as illustrated by the solid error bars) then the comparison process is complete. If the error bars overlap (as illustrated by the dotted error bars) one falls back on the exact objective function $e(\bar{x})$ to evaluate and compare $e(\bar{t}_i)$ and $e(\bar{t}_j)$. This approach uses the approximation on all the clear cut cases and relies on the exact function only when needed.

A complication arises when a large number of starting points is used during the first stage of strategy MLM_3 . It may happen that two designs \bar{t}_i and \bar{t}_j lie very close to the same local optimum. In such a case, their error bars will likely overlap — recommending use of the exact objective function to discriminate between them. Nevertheless, the choice between two such designs will probably not matter very much. The computation expended to evaluate them exactly would be wasted effort. A possible solution is to cluster the designs $\{\bar{t}_1, \dots, \bar{t}_n\}$ into equivalence classes such that two designs are in the same class whenever they are near each other in design space. One then carries out the comparison between designs using only one representative from each class.

A second refinement involves the use of error estimates to choose convergence criteria. Numeric optimization codes often test for convergence by comparing the evaluations of successive iterates. For example, one convergence criterion available in CFSQP tests whether the absolute change in the design quality is below a user-specified threshold. In the context of multi-level modeling, error estimates provide a basis for choosing such a threshold. For example, in strategy MLM_1 , when the first stage of optimization (using the fixed approximation $\tilde{e}(\bar{x})$) calls a numeric optimization code, the average error of $\tilde{e}(\bar{x})$ would be a natural choice for the convergence threshold. Likewise, in strategy MLM_2 , when the inner optimization (using the recalibratable approximation $\hat{e}(\bar{x}, \bar{y})$) calls a numeric optimization code, an estimate of the average error of $\hat{e}(\bar{x}, \bar{y})$ would be a natural choice for the convergence threshold. In practice, the actual choice of threshold should probably depend on both the estimated error and the costs of computation of approximate objective functions. For example, in the yacht domain, the algebraic formulae for effective draft and wave resistance are much faster than the PMARC and SLAW CFD codes. Therefore relatively little savings in CPU time results from early termination of an initial optimization (or inner optimization) using these algebraic formulae. In such cases, optimizations using approximate objective functions can be run to convergence levels beyond the error estimates with little sacrifice of computational resources.

3.6 Generalizing to an Arbitrary Number of Levels

The strategies MLM_1 , MLM_2 and MLM_3 in Figure 4 can all be generalized to utilize an arbitrary number of models, each at a different level of approximation. For example, strategy MLM_1 can be generalized from two levels to N levels by replacing the sequence of two optimization steps with a sequence of N optimization steps. Likewise, strategy MLM_2 can be made to operate with N levels by replacing the *InnerOptimization* step with a version of MLM_2 that operates with $N - 1$ levels. The generalization of strategy MLM_3 is more complicated. One approach would optimize some number n of seed designs $\{\bar{s}_1, \dots, \bar{s}_n\}$ at each level to obtain n “optimal” points $\{\bar{t}_1, \dots, \bar{t}_n\}$. A fraction of the optimal points would be passed down as seeds for the the next level. At the lowest level, the strategy returns the result that is best, according to the “exact” model.

Such multi-level strategies may or may not be useful in practice. Suppose the overall cost of a two-level optimization is dominated by the computations that take place at the bottom level. The addition of a third level, at the top of the hierarchy, may speed up the processing that takes place at the middle level; however, it will do little to speed up the processing that takes place at the bottom level. In contrast to this, suppose the overall cost of a two-level optimization is dominated by the computations that take place at the top level. The addition of a third level, at the top of the hierarchy, may well speedup the overall solution process.

4 Theoretical Analysis

4.1 Analysis of the Simple Two-Level Strategy

The simple two-level optimization strategy MLM_1 uses the fixed approximation $\tilde{e}(\bar{x})$ to get close to the optimum. It uses the exact objective function $e(\bar{x})$ to move the remaining distance to the optimum. In this section, we investigate the question of when MLM_1 will run faster than a strategy that simply uses $e(\bar{x})$ in a single phase of optimization. Strategy MLM_1 will require fewer evaluations of $e(\bar{x})$ if two conditions are met: (1) The first phase of MLM_1 generates an intermediate design \bar{s}_{temp} that is closer than the seed \bar{s}_{old} to the true optimum, i.e., $|\bar{s}_{new} - \bar{s}_{temp}| < |\bar{s}_{new} - \bar{s}_{old}|$. (2) The number of evaluations of $e(\bar{x})$ needed by the numeric optimization code decreases with the distance from the seed to the optimum. The performance of MLM_1 will depend in part on the order of convergence [Dahlquist and Bjorck, 1974] of the optimization method. Consider what happens when the method has linear convergence. Let ϵ_i be the distance of the i th iteration from the optimum.

$$\begin{aligned}\epsilon_{i+1} &\approx k\epsilon_i \\ \epsilon_n &\approx k^n \epsilon_0 \\ n &\approx \log_k \left(\frac{\epsilon_n}{\epsilon_0} \right)\end{aligned}$$

The cost of carrying out an optimization to achieve a desired error of ϵ_n grows as the logarithm of the error ϵ_0 of the starting point. Suppose the initial seed s_{old} is a distance α from the optimum. Suppose further that the first stage of optimization results in a design s_{temp} at a distance β from the optimum. If we assume that the computational cost of the first stage is negligible, then the relative savings S obtained by using strategy MLM_1 is given by:

$$\begin{aligned}S &\approx \frac{n(\alpha) - n(\beta)}{n(\alpha)} \\ &\approx \frac{\log_k(\epsilon_n/\alpha) - \log_k(\epsilon_n/\beta)}{\log_k(\epsilon_n/\alpha)}\end{aligned}$$

For example, in order to achieve a factor of two speedup, the first stage of strategy MLM_1 must move to a point s_{temp} whose error is the geometric mean of the error of the initial point s_{old} and the desired error of the final design s_{new} , i.e., $\beta \approx \sqrt{\alpha\epsilon_n}$. This puts a rather stringent requirement on the fixed approximation $\tilde{e}(\bar{x})$. Now consider what happens when the optimization method has quadratic convergence.

$$\begin{aligned}\epsilon_{i+1} &\approx k\epsilon_i^2 \\ \epsilon_n &\approx \frac{1}{k}(k\epsilon_0)^{2^n} \\ n &\approx \log_2 \left(\frac{\log(k\epsilon_n)}{\log(k\epsilon_0)} \right)\end{aligned}$$

The cost of carrying out an optimization to achieve a desired error of ϵ_n grows as the double logarithm of error ϵ_0 of the starting point. The relative savings S obtained by using strategy

MLM_1 is given by:

$$\begin{aligned} S &\approx \frac{n(\alpha) - n(\beta)}{n(\alpha)} \\ &\approx \frac{\log_2 [\log(k\epsilon_n)/\log(k\alpha)] - \log_2 [\log(k\epsilon_n)/\log(k\beta)]}{\log_2 [\log(k\epsilon_n)/\log(k\alpha)]} \end{aligned}$$

For example, in order to achieve a factor of two speedup, if we take $k = 1$, the first stage of strategy MLM_1 must move to a point s_{temp} whose log-error is the geometric mean of the log-error of the initial point s_{old} and the desired log-error of the final design s_{new} , i.e., $\log(\beta) \approx \sqrt{\log(\alpha)\log(\epsilon_n)}$. This puts a somewhat less stringent requirement on the fixed approximation.

4.2 Analysis of the Recalibrating Strategy

Strategy MLM_2 uses the recalibratable objective function $\hat{e}(\bar{x}, \bar{y})$ in the inner optimization. It uses the exact objective function $e(\bar{x})$ or portions of $e(\bar{x})$ only during calibration, or recalibration, in between successive inner optimizations. This section investigates several questions about the behavior of strategy MLM_2 : (1) Is an optimum found by MLM_2 always an optimum of the exact objective function? (2) Is MLM_2 guaranteed to converge? (3) Does MLM_2 run faster than a strategy that simply uses $e(\bar{x})$ in a single phase of optimization? We investigate these questions by considering an idealized version of strategy MLM_2 . The idealization is based on an assumption that *LineSearch* routine is explicitly turned off (*DoLineSearch?* = *False*) or else the *LineSearch* routine operates as a null operation, i.e., it always returns its second argument, \bar{s}_{new} , the local optimum of $\hat{e}_s(\bar{x})$ found by the inner optimization routine. Later in this section we shall discuss the degree to which our analysis applies when we relax this assumption.

4.2.1 Analysis of the Idealized Strategy

We begin with two definitions.¹ The first definition concerns how closely $\hat{e}(\bar{x}, \bar{y})$ approximates $e(\bar{x})$. The second concerns properties of $\hat{e}(\bar{x}, \bar{y})$ itself. These definitions characterize conditions under which we can prove correctness and convergence of the idealized version of strategy MLM_2 :

Definition 1 A function $\hat{e} : \mathfrak{R}^n \times \mathfrak{R}^n \rightarrow \mathfrak{R}$ is a locally calibratable approximation scheme of order p for a function $e : \mathfrak{R}^n \rightarrow \mathfrak{R}$, if $e(\bar{x})$ and $\hat{e}(\bar{x}, \bar{y})$ are continuous and p times differentiable in \bar{x} and $(\forall \bar{x}, \bar{y} \in \mathfrak{R}^n)$ and $(\forall i \in [0 \dots p])$, $\bar{x} = \bar{y}$ implies that $\partial^i \hat{e}(\bar{x}, \bar{y}) / \partial \bar{x}^i = \partial^i e(\bar{x}) / \partial \bar{x}^i$.

This definition can be understood in terms of the graph shown in Figure 9. If an approximation scheme $\hat{e}(\bar{x}, \bar{y})$ has order zero, the value of $\hat{e}(\bar{x}, \bar{y})$ will fit the value of $e(\bar{x})$ in the subspace defined by $\bar{y} = \bar{x}$. If an approximation scheme $\hat{e}(\bar{x}, \bar{y})$ has order one, the value and gradient of $\hat{e}(\bar{x}, \bar{y})$ (with respect to \bar{x}) will fit the value and gradient of $e(\bar{x})$, in the subspace defined by $\bar{y} = \bar{x}$.

¹Notation: A closed hyperrectangle with opposite corners \bar{l} and \bar{u} is denoted by $[\bar{l}, \bar{u}]$. An open hyperrectangle is denoted by (\bar{l}, \bar{u}) . The gradient of a multivariate scalar function $f(\bar{x})$ is a column vector denoted by $\partial f / \partial \bar{x}$. The Hessian of $f(\bar{x})$ is a matrix denoted by $\partial^2 f(\bar{x}) / \partial \bar{x}^2$. The mixed derivative of $f(\bar{x}, \bar{y})$ is a matrix $[a_{ij}]$ with n (length of \bar{x}) rows and m (length of \bar{y}) columns where $a_{ij} = \partial^2 f / \partial y_j \partial x_i$, and is denoted by $\partial^2 f(\bar{x}, \bar{y}) / \partial \bar{y} \partial \bar{x}$. The norm of a matrix A is denoted by $|A|$.

Definition 2 A locally calibratable approximation scheme $\hat{e} : \mathfrak{R}^n \times \mathfrak{R}^n \rightarrow \mathfrak{R}$ is admissible for unconstrained minimization from a seed in $[\bar{l}, \bar{u}] \subset \mathfrak{R}^n$ if:

1. The function $\hat{e}(\bar{x}, \bar{y})$ is continuous and twice differentiable in \bar{x} and \bar{y} on $[\bar{l}, \bar{u}] \times [\bar{l}, \bar{u}]$.
2. $(\forall \bar{s} \in [\bar{l}, \bar{u}]), \partial^2 \hat{e}(\bar{x}, \bar{y}) / \partial \bar{x}^2$ is positive definite whenever $\bar{x} = \bar{y} = \bar{s}$.
3. $(\exists c), (\forall \bar{x}, \bar{y} \in [\bar{l}, \bar{u}]), |(\partial^2 \hat{e}(\bar{x}, \bar{y}) / \partial \bar{x}^2)^{-1} \cdot (\partial^2 \hat{e}(\bar{x}, \bar{y}) / \partial \bar{y} \partial \bar{x})| \leq c < 1$.
4. $(\forall \bar{s} \in [\bar{l}, \bar{u}]), (\exists \bar{r}) \epsilon(\bar{l}, \bar{u})$ such that \bar{r} is a local minimum of $\hat{e}_{\bar{s}}(\bar{x})$.

In the context of an admissible approximation scheme $\hat{e}(\bar{x}, \bar{y})$, we may speak of a function $I : [\bar{l}, \bar{u}] \rightarrow [\bar{l}, \bar{u}]$ such that $I(\bar{s})$ is the unique local minimum of $\hat{e}_{\bar{s}}(\bar{x})$ in $[\bar{l}, \bar{u}]$. The existence of such a local minimum is guaranteed by condition four. Its uniqueness is guaranteed by the following argument: The admissibility of $\hat{e}(\bar{x}, \bar{y})$ implies that the Hessian $\partial^2 \hat{e}(\bar{x}, \bar{y}) / \partial \bar{x}^2$ is positive definite on the diagonal $\bar{x} = \bar{y}$ and varies continuously throughout $[\bar{l}, \bar{u}] \times [\bar{l}, \bar{u}]$. If there were some point (\bar{x}_0, \bar{y}_0) in $[\bar{l}, \bar{u}] \times [\bar{l}, \bar{u}]$ at which the Hessian were not positive definite, then there would necessarily be some point on the line segment from (\bar{x}_0, \bar{x}_0) to (\bar{x}_0, \bar{y}_0) at which the Hessian would be singular. This would violate condition two. Therefore the Hessian $\partial^2 \hat{e}(\bar{x}, \bar{y}) / \partial \bar{x}^2$ is positive definite throughout $[\bar{l}, \bar{u}] \times [\bar{l}, \bar{u}]$. Since $\partial^2 \hat{e}(\bar{x}, \bar{s}) / \partial \bar{x}^2 = \partial^2 \hat{e}_{\bar{s}}(\bar{x}) / \partial \bar{x}^2$, this guarantees uniqueness of the local minimum of $\hat{e}_{\bar{s}}(\bar{x})$. (This argument was taken from [Kachiyan, 1997]). We may think of the function $I(\bar{s})$ as representing the *InnerOptimization* routine in strategy *MLM*₂. We may define the idealized version of strategy *MLM*₂ to be an algorithm that accepts a seed \bar{s}_0 as input, and repeats the iteration $\bar{s}_{i+1} = I(\bar{s}_i)$ until reaching a fixed point $\bar{s}_f = I(\bar{s}_f)$. We can thus formulate and prove theorems about the idealized procedure *MLM*₂, in terms of behavior of the function $I(\bar{s})$.

Theorem 1 Let $\hat{e}(\bar{x}, \bar{y})$ be a locally calibratable approximation scheme of order $p \geq 1$ for $e(\bar{x})$ that is admissible for unconstrained minimization from a seed in $[\bar{l}, \bar{u}] \subset \mathfrak{R}^n$. Let $e(\bar{x})$ be convex on $[\bar{l}, \bar{u}]$. Then for any $\bar{s} \in [\bar{l}, \bar{u}]$, \bar{s} is a local minimum of $e(\bar{x})$ if and only if $\bar{s} = I(\bar{s})$.

Proof: Suppose $\bar{s} \in [\bar{l}, \bar{u}]$ is a local minimum of $e(\bar{x})$. Then the derivative $\partial e(\bar{x}) / \partial \bar{x} = \bar{0}$ when $\bar{x} = \bar{s}$, since $e(\bar{x})$ is continuous and differentiable. Then $\partial \hat{e}(\bar{x}, \bar{y}) / \partial \bar{x} = \bar{0}$ when $\bar{x} = \bar{y} = \bar{s}$ since $\hat{e}(\bar{x}, \bar{y})$ is a locally calibratable approximation scheme of order $p \geq 1$. Then the function $\hat{e}_{\bar{s}}(\bar{x})$ has zero derivative at $\bar{x} = \bar{s}$. Also, $\partial^2 \hat{e}(\bar{x}, \bar{y}) / \partial \bar{x}^2$ is positive definite when $\bar{x} = \bar{y} = \bar{s}$ since $\hat{e}(\bar{x}, \bar{y})$ is admissible for minimization from a seed in $[\bar{l}, \bar{u}]$. Since $\partial^2 \hat{e}(\bar{x}, \bar{s}) / \partial \bar{x}^2 = \partial^2 \hat{e}_{\bar{s}}(\bar{x}) / \partial \bar{x}^2$, it follows that \bar{s} is a local minimum of $\hat{e}_{\bar{s}}(\bar{x})$. Furthermore, the minimum \bar{s} is unique in $[\bar{l}, \bar{u}]$ since $\hat{e}(\bar{x}, \bar{y})$ is admissible for minimization from a seed in $[\bar{l}, \bar{u}]$. Therefore $\bar{s} = I(\bar{s})$. Now suppose that $\bar{s} \in [\bar{l}, \bar{u}]$ and $\bar{s} = I(\bar{s})$. Then \bar{s} is a local minimum of $\hat{e}_{\bar{s}}(\bar{x})$. Then the function $\hat{e}_{\bar{s}}(\bar{x})$ has zero derivative at $\bar{x} = \bar{s}$, since $\hat{e}(\bar{x}, \bar{y})$ is continuous and differentiable. Then $\partial \hat{e}(\bar{x}, \bar{y}) / \partial \bar{x} = \bar{0}$ when $\bar{x} = \bar{y} = \bar{s}$. Then $\partial e(\bar{x}) / \partial \bar{x} = \bar{0}$ when $\bar{x} = \bar{s}$, since $\hat{e}(\bar{x}, \bar{y})$ is a locally calibratable approximation scheme of order $p \geq 1$. Since $e(\bar{x})$ is convex on $[\bar{l}, \bar{u}]$, it follows that \bar{s} is a local minimum of $e(\bar{x})$. \square

When the conditions of our first theorem are met, every fixed point $\bar{s} \in [\bar{l}, \bar{u}]$ of $I(\bar{s})$ is a local optimum of $e(\bar{x})$. Likewise, any local optimum $\bar{s} \in [\bar{l}, \bar{u}]$ of $e(\bar{x})$ is a fixed point of $I(\bar{s})$. Since fixed-point iteration of $I(\bar{s})$ corresponds to our idealized version of strategy *MLM*₂, these conditions tell us when a design returned by *MLM*₂ is guaranteed to be a local optimum for the original problem, assuming the inner optimization operates in an ideal fashion. Among other things the

conditions require that $\hat{e}(\bar{x}, \bar{y})$ fit the gradient $\nabla e(\bar{x})$ exactly at all possible calibration points $\bar{x} = \bar{y}$ for $\bar{x}, \bar{y} \in [\bar{l}, \bar{u}]$.

Consider how this theorem can be applied to the calibratable approximations described in Figure 8. Functions $\hat{e}_1(\bar{x}, \bar{y})$, $\hat{e}_3(\bar{x}, \bar{y})$ and $\hat{e}_5(\bar{x}, \bar{y})$ are locally calibratable approximation schemes of order zero, i.e., they fit the objective function $e(\bar{x})$, at the calibration point. They are not guaranteed to fit the gradient $\nabla e(\bar{x})$ at the calibration point. They may fit the gradient by fortuitous accident. For example, if it happens that $\nabla \tilde{F}_i(\bar{x}) = \nabla F_i(\bar{x})$ for all \bar{x} , then $\hat{e}_3(\bar{x}, \bar{y})$ will fit the gradient of $e(\bar{x})$ at the calibration point. Likewise, if $\nabla \tilde{F}_i(\bar{x})(F_i(\bar{x})/\tilde{F}_i(\bar{x})) = \nabla F_i(\bar{x})$ for all \bar{x} , then $\hat{e}_5(\bar{x}, \bar{y})$ will fit the gradient of $e(\bar{x})$ at the calibration point. In any case, these three functions should not normally be expected to result in exactly correct solutions when used in strategy MLM_2 . On the other hand, functions $\hat{e}_2(\bar{x}, \bar{y})$, $\hat{e}_4(\bar{x}, \bar{y})$, $\hat{e}_6(\bar{x}, \bar{y})$ are locally calibratable approximation schemes of order one, i.e., they fit both the objective function $e(\bar{x})$ and the gradient $\nabla e(\bar{x})$ at the calibration point. Assuming these functions satisfy the other admissibility conditions, they should return exactly correct solutions when used in strategy MLM_2 , subject to ordinary numerical errors.

Theorem 2 *Let $\hat{e}(\bar{x}, \bar{y})$ be a locally calibratable approximation scheme for $e(\bar{x})$ that is admissible for unconstrained minimization from a seed in $[\bar{l}, \bar{u}] \subset \mathfrak{R}^n$. Then the function $I(\bar{s})$ has a unique fixed point $\bar{s}_f \in (\bar{l}, \bar{u})$, and $(\forall \bar{s}_0 \in [\bar{l}, \bar{u}])$, the series $(\bar{s}_0, \bar{s}_1, \dots, \bar{s}_i, \dots)$ generated by the rule $\bar{s}_{i+1} = I(\bar{s}_i)$ converges to \bar{s}_f .*

Proof: By the fixed-point theorem [Conte and de Boor, 1980], there exists a fixed point $\bar{s}_f \in (\bar{l}, \bar{u})$ of the function $I(\bar{s})$ such that \bar{s}_f is unique in (\bar{l}, \bar{u}) and \bar{s}_f is the limit of the series $(\bar{s}_0, \bar{s}_1, \dots, \bar{s}_i, \dots)$ defined by $\bar{s}_{i+1} = I(\bar{s}_i)$, provided three conditions are met: (1) $\bar{s}_0 \in [\bar{l}, \bar{u}]$; (2) $I(\bar{s})$ maps $[\bar{l}, \bar{u}]$ into itself; (3) $I(\bar{s})$ is a contraction mapping, i.e., for all $\bar{s}_1, \bar{s}_2 \in [\bar{l}, \bar{u}]$, $|I(\bar{s}_1) - I(\bar{s}_2)| < c|\bar{s}_1 - \bar{s}_2|$ for some constant $c < 1$. The first condition is true by assumption. The second condition is guaranteed by the admissibility of $\hat{e}(\bar{x}, \bar{y})$. The third may be demonstrated by showing that the Jacobian of $I(\bar{s})$ exists and satisfies the relation $|\partial I(\bar{s})/\partial \bar{s}| \leq c$ for some constant $c < 1$, for all $\bar{s} \in [\bar{l}, \bar{u}]$. Notice that $I(\bar{s})$ is the unique value of \bar{x} in $[\bar{l}, \bar{u}]$ that solves the equation $\bar{\gamma}(\bar{x}, \bar{s}) = \bar{0}$ where $\bar{\gamma}(\bar{x}, \bar{y}) = \partial \hat{e}(\bar{x}, \bar{y})/\partial \bar{x}$. By the implicit function theorem [Rudin, 1964], the Jacobian of $I(\bar{s})$ exists and is given by $\partial I(\bar{s})/\partial \bar{s} = -(\partial \bar{\gamma}/\partial \bar{x})^{-1} \cdot (\partial \bar{\gamma}/\partial \bar{y}) = -(\partial^2 \hat{e}(\bar{x}, \bar{y})/\partial \bar{x}^2)^{-1} \cdot (\partial^2 \hat{e}(\bar{x}, \bar{y})/\partial \bar{y} \partial \bar{x})$ when $\bar{x} = I(\bar{s})$ and $\bar{y} = \bar{s}$. The admissibility of $\hat{e}(\bar{x}, \bar{y})$ guarantees that the norm of this product is less than one for all $\bar{x}, \bar{y} \in [\bar{l}, \bar{u}]$. Therefore $I(\bar{s})$ is a contraction mapping, satisfying the third condition of the fixed-point theorem. \square

Our second theorem asserts conditions under which the idealized version of strategy MLM_2 is guaranteed to converge. Among other things, the norm of the product of the mixed second derivative of $\hat{e}(\bar{x}, \bar{y})$ and unmixed second derivative (in \bar{x}) of $\hat{e}(\bar{x}, \bar{y})$ must be less than one. As an illustration of this condition, consider the example of an objective function that is the sum of two terms $f(x)$ and $g(x)$. Suppose that $g(x)$ is relatively expensive to compute. One might choose an approximation scheme that fits a linear function to $g(x)$ at each calibration point:

$$\begin{aligned} e(x) &\equiv f(x) + g(x) \\ \hat{e}(x, y) &\equiv f(x) + g(y) + g'(y)(x - y) \\ \frac{\partial^2 \hat{e}(x, y)}{\partial x^2} &= f''(x) \\ \frac{\partial^2 \hat{e}(x, y)}{\partial y \partial x} &= g''(y) \end{aligned}$$

In this example, the admissibility condition on the derivatives of $\hat{e}(x, y)$ reduces to the following: $(\forall x, y \in [l, u]) |g''(y)| < |f''(x)|$. The term $g(x)$, which is subjected to the linear approximation, must be closer to linear than the term $f(x)$, which is left alone.

As a second illustration of the convergence condition for strategy MLM_2 , consider how it can be applied to the calibratable approximations for the yacht domain. (See Figure 8.) First we rewrite $\hat{e}(\bar{x}, \bar{y})$ in the following form:

$$\hat{e}(\bar{x}, \bar{y}) = R(\bar{E}(\bar{x}), \hat{D}(\bar{x}, \bar{y}))$$

The function $\bar{E}(\bar{x})$ represents all of the hull processing models other than the one that computes effective draft. Notice that $\bar{E}(\bar{x})$ does not depend on the calibration parameter \bar{y} , i.e., it is not subject to approximation. Using the chain rule twice, we derive formulae for the derivatives appearing in the convergence condition:

$$\begin{aligned} \frac{\partial^2 \hat{e}(\bar{x}, \bar{y})}{\partial \bar{y} \partial \bar{x}} &= \frac{\partial R}{\partial \hat{D}} \left[\frac{\partial^2 \hat{D}}{\partial \bar{y} \partial \bar{x}} \right] + \frac{\partial^2 R}{\partial \hat{D}^2} \frac{\partial \hat{D}}{\partial \bar{x}} \frac{\partial \hat{D}}{\partial \bar{y}} + \left[\frac{\partial^2 R}{\partial \bar{E} \partial \hat{D}} \right] \frac{\partial \bar{E}}{\partial \bar{x}} \frac{\partial \hat{D}}{\partial \bar{y}} \\ \frac{\partial^2 \hat{e}(\bar{x}, \bar{y})}{\partial \bar{x}^2} &= \frac{\partial R}{\partial \bar{E}} \frac{\partial^2 \bar{E}}{\partial \bar{x}^2} + \frac{\partial R}{\partial \hat{D}} \frac{\partial^2 \hat{D}}{\partial \bar{x}^2} + \frac{\partial^2 R}{\partial \bar{E}^2} \left[\frac{\partial \bar{E}}{\partial \bar{x}} \right]^2 + \frac{\partial^2 R}{\partial \hat{D}^2} \left[\frac{\partial \hat{D}}{\partial \bar{x}} \right]^2 + 2 \left[\frac{\partial^2 R}{\partial \bar{E} \partial \hat{D}} \right] \frac{\partial \bar{E}}{\partial \bar{x}} \frac{\partial \hat{D}}{\partial \bar{x}} \end{aligned}$$

We can simplify these formulae if we assume that essentially all of the nonlinearity in $\hat{e}(\bar{x}, \bar{y})$ appears in $\bar{E}(\bar{x})$ and $D(\bar{x})$, the hull processing models. In this case, all of the second derivatives involving R vanish:

$$\begin{aligned} \frac{\partial^2 \hat{e}(\bar{x}, \bar{y})}{\partial \bar{y} \partial \bar{x}} &= \frac{\partial R}{\partial \hat{D}} \left[\frac{\partial^2 \hat{D}}{\partial \bar{y} \partial \bar{x}} \right] \\ \frac{\partial^2 \hat{e}(\bar{x}, \bar{y})}{\partial \bar{x}^2} &= \frac{\partial R}{\partial \bar{E}} \frac{\partial^2 \bar{E}}{\partial \bar{x}^2} + \frac{\partial R}{\partial \hat{D}} \frac{\partial^2 \hat{D}}{\partial \bar{x}^2} \end{aligned}$$

If we further assume that $\hat{e}(\bar{x}, \bar{y})$ satisfies the second admissibility condition (i.e., $\partial^2 \hat{e}(\bar{x}, \bar{y}) / \partial \bar{x}^2$ is strictly positive definite) then the convergence condition will be satisfied if $\partial^2 \hat{D} / \partial \bar{y} \partial \bar{x}$ is zero. Now consider what happens when $\hat{D}(\bar{x}, \bar{y})$ is implemented by each of the six locally calibratable internal approximations F_{i1}, \dots, F_{i6} in Figure 8. Notice that the mixed derivative $\partial^2 \hat{D} / \partial \bar{y} \partial \bar{x}$ is zero in the cases of F_{i1} and F_{i3} . Likewise, the mixed derivative $\partial^2 \hat{D} / \partial \bar{y} \partial \bar{x}$ is zero in the cases of F_{i2} and F_{i4} , provided (respectively) that $F_i(\bar{x})$ is linear and that $F_i(\bar{x}) - \tilde{F}_i(\bar{x})$ is linear. In the case of F_{i5} , it suffices that $F_i(\bar{x}) / \tilde{F}_i(\bar{x})$ be constant. The corresponding condition in the case of F_{i6} is more complicated. Our analysis thus indicates that key factors governing convergence include the relative distribution of nonlinearity in different parts of the objective function, and the degree of nonlinearity in the error of the internal approximation.

4.2.2 Extending the Correctness and Convergence Theorems

Our analysis so far has applied only to the idealized version of strategy MLM_2 , in which the *LineSearch* routine is either disabled or operates as a null operation. We can extend our analysis to include the line search in the following way. Define the function $L(\bar{s}_{old}, \bar{s}_{new})$ with the following behavior. Consider the series of points $(\bar{t}_0, \bar{t}_1, \dots, \bar{t}_i, \dots)$ where $\bar{t}_0 = \bar{s}_{new}$ and $\bar{t}_{i+1} = (\bar{s}_{old} + \bar{t}_i) / 2$, for

all $i \geq 0$. Let $L(\bar{s}_{old}, \bar{s}_{new})$ be t_i where i is the smallest integer such that $e(\bar{t}_i)$ is better than $e(\bar{s}_{old})$ and $|\bar{t}_i - \bar{s}_{old}| > \delta$, if such an integer exists. Let $L(\bar{s}_{old}, \bar{s}_{new})$ be \bar{s}_{old} , otherwise. We can now analyze the behavior of strategy MLM_2 by considering the series of points generated by the composition of the *InnerOptimization* and *LineSearch* routines: $C(\bar{s}) = L(\bar{s}, I(\bar{s}))$.

Under the assumptions of our original correctness and convergence theorems, it is possible to prove that for all \bar{s}_0 in $[\bar{l}, \bar{u}]$ the series $(\bar{s}_0, \bar{s}_1, \dots, \bar{s}_i, \dots)$ generated by the rule $\bar{s}_{i+1} = C(\bar{s}_i)$ converges to a limit \bar{s}_l . Notice first that $\bar{s}_{i+1} = (1 - h_i)\bar{s}_i + (h_i)I(\bar{s}_i)$ where each h_i lies in the interval $[0, 1]$. Now consider two cases: (1) Suppose $\sum_{i=0}^{\infty} h_i = \infty$. First we define $r_i = |\bar{s}_i - \bar{s}_f|$, where \bar{s}_f is the unique fixed point of $I(\bar{s})$ in $[\bar{l}, \bar{u}]$, from our original correctness proof. We can then show that $r_{i+1} \leq \alpha_i r_i$, where $\alpha_i = 1 - (1 - c)h_i$, by using simple algebra, along with the fact that $\bar{s}_f = I(\bar{s}_f)$, and the fact that $|I(\bar{s}_i) - I(\bar{s}_f)| < c|\bar{s}_i - \bar{s}_f|$ for the constant $c < 1$, from our original convergence proof. We can then show that $\prod_{i=0}^{\infty} \alpha_i = 0$ by taking the logarithm of this product and using the fact that $\ln(1 - x) < -x$. It follows that the series $(r_0, r_1, \dots, r_i, \dots)$ converges to zero, and the series $(\bar{s}_0, \bar{s}_1, \dots, \bar{s}_i, \dots)$ converges to the limit $\bar{s}_l = \bar{s}_f$. (2) Suppose $\sum_{i=0}^{\infty} h_i < \infty$. We can apply the Cauchy convergence criterion [Rudin, 1964] to this summation: Thus for every $\epsilon > 0$, there is some integer N such that $\sum_{i=m+1}^n h_i < \epsilon$ whenever $n > m > N$. Notice also that $|\bar{s}_{i+1} - \bar{s}_i| = h_i |I(\bar{s}_i) - \bar{s}_i| \leq h_i |\bar{u} - \bar{l}|$. Using the triangle inequality, it follows that $|\bar{s}_n - \bar{s}_m| \leq (\sum_{i=m+1}^n h_i) |\bar{u} - \bar{l}|$. Thus for every $\epsilon > 0$, there is some integer N such that $|\bar{s}_n - \bar{s}_m| < \epsilon$ whenever $n > m > N$. Applying the Cauchy criterion again, it follows that series $(\bar{s}_0, \bar{s}_1, \dots, \bar{s}_i, \dots)$ converges to a limit \bar{s}_l . (This proof was taken from [Kachiyan, 1997].)

Under the assumptions of our original correctness and convergence theorems, it is also possible to prove that a point \bar{s} in $[\bar{l}, \bar{u}]$ is a local optimum of $e(\bar{x})$ if and only if \bar{s} is a fixed point of $C(\bar{s})$, provided the threshold δ used in the definition of $L(\bar{s}_{old}, \bar{s}_{new})$ is set to zero. Unfortunately, the function $C(\bar{s})$ might not be continuous. Consequently, the limit \bar{s}_l of the series $(\bar{s}_0, \bar{s}_1, \dots, \bar{s}_i, \dots)$ generated by the rule $\bar{s}_{i+1} = C(\bar{s}_i)$ might not be a fixed point of $C(\bar{s})$, and therefore not a local optimum of $e(\bar{x})$. Although the composition of the *InnerOptimization* and *LineSearch* is guaranteed to converge to a limit, the result may not be a correct solution to the original optimization problem. We can overcome this limitation by using a more sophisticated line search algorithm, e.g., one that satisfies the ‘‘Goldstein-Armijo’’ conditions [Dennis and Schnabel, 1983]. Unfortunately, a more sophisticated line search would likely result in greater use of the exact objective function $e(\bar{x})$, and correspondingly greater computational expense.

Our analysis does allow us to make the following guarantee about the correctness of strategy MLM_2 , even when using our original line search routine. After the procedure terminates at a point \bar{s}_l , we determine whether the condition *Converged?* $(\bar{s}_l, \text{InnerOptimization}(\bar{s}_l))$ is satisfied. (Since MLM_2 uses *Converged?* $(\bar{s}_l, \text{LineSearch}(s_l, \text{InnerOptimization}(\bar{s}_l)))$ as its convergence test, this condition may or may not be satisfied.) When the condition is satisfied, we have some level of assurance that \bar{s}_l is close to a fixed point of the *InnerOptimization* function. The level of assurance depends on the tolerance ϵ used in the *Converged?* test. According to our original correctness theorem, any fixed point of *InnerOptimization* is also a local optimum of $e(\bar{x})$, the exact objective function. We therefore have the same level of assurance that \bar{s}_l is close to a correct solution to the original optimization problem.

4.2.3 Practical Applicability of the Theoretical Analysis

For many applications, it will be difficult or impossible to establish the conditions of our theorems about the correctness and convergence of strategy MLM_2 . For example, in the yacht domain, the objective function is sufficiently complicated that one cannot analytically verify the conditions of the theorems. The conditions might be checked empirically by sampling points in the design space; however, sampling would incur considerable computational cost and would still not provide any absolute guarantees. For these reasons, we expect our theorems to have at most limited value for a priori prediction of correctness and convergence. The theorems may nevertheless be useful in other ways. For example, suppose one uses strategy MLM_2 to solve a problem, and the algorithm does not appear to converge. Our convergence theorem suggests ways to overcome this difficulty. In particular, one might help the algorithm to converge by switching to an approximation scheme that is more likely to satisfy the condition on the mixed derivative, as discussed above. On the other hand, suppose the algorithm does converge, but that subsequent evaluations of the objective function in the neighborhood of the solution demonstrate that the algorithm did not find a local optimum. Our correctness theorem suggests ways to overcome this difficulty. In particular, one might cure a problem with correctness by switching from an approximation scheme of order zero to an approximation scheme of order one.

4.2.4 Performance of the Recalibrating Strategy

The overall performance of MLM_2 will depend a variety of issues: the accuracy of $\hat{e}(\bar{x}, \bar{y})$ at locations removed from calibration points; the computational cost of $\hat{e}(\bar{x}, \bar{y})$ in comparison to the cost of $e(\bar{x})$; and the computational cost of recalibrating $\hat{e}(\bar{x}, \bar{y})$. Little can be said in general about these tradeoffs. On the other hand, the *asymptotic* behavior of MLM_2 can be characterized more precisely. In particular, the order of convergence [Dahlquist and Bjorck, 1974] of MLM_2 is equal to the smallest derivative of $I(\bar{s})$ with a nonzero norm at the optimum. (Assuming suitably many derivatives of $I(\bar{s})$ are defined and continuous.) Thus if $|\partial I(s)/\partial \bar{s}| > 0$ at the optimum, then MLM_2 will have linear convergence. It will therefore be asymptotically inferior to a single-stage strategy that uses a Newton or Quasi-Newton method in combination with the exact objective function. There will be some sufficiently low error tolerance that can be achieved faster by the single-stage strategy. On the other hand, if $|\partial I(s)/\partial \bar{s}| = 0$ at the optimum, MLM_2 will have quadratic or better convergence and will be asymptotically as good or better than the default strategy. Of course, the desired error may not be one that leads the algorithms into regions of behavior where asymptotic comparisons are relevant. Asymptotic comparisons have limited value in practice.

4.2.5 Extending the Analysis to Constrained Optimization Problems

We observe in passing one more way in which these analytic results might be generalized. The theorems are limited to problems of unconstrained optimization. They could be extended to constrained optimization in two ways: (1) recalibratable approximation of the objective function; and (2) recalibratable approximation of the constraints. Our proofs of convergence and correctness might be generalized to handle such approximation methods for constrained optimization problems by using the Karush-Kuhn-Tucker conditions [Peressini *et al.*, 1988] to characterize constrained optima of $e(\bar{x})$ and $\hat{e}(\bar{x}, \bar{y})$; however, we have not yet attempted to generalize our results in this fashion.

5 Experimental Results

5.1 Implementation of Multi-Model Optimization Strategies

We have implemented our multi-model optimization strategies as part of the “Design and Modeling/Simulation Associate” (DA-MSA) [Ellman *et al.*, 1992],[Ellman *et al.*, 1995],[Ellman *et al.*, 1996]. The DA-MSA is a system that supports interactive construction of numerical models for simulation of engineering design artifacts. It also supports construction of design optimization strategies that use simulation models to solve design problems. Simulation models and optimization strategies are represented visually as second-order dataflow graphs. Nodes represent functions like root-finding, integration or optimization. Arcs represent flow of data and control. The dataflow graphs are actually implemented as executable LISP functions that wrap numerical C routines from [Press *et al.*, 1986]. The DA-MSA also implements compositional modeling in the following way: It maintains a library of functions that may be used to define simulation models and optimization strategies. The library is organized in the manner indicated in Figure 6. Each function in the system is (potentially) implemented in several different versions. Each version embodies a different approximation of the function.

The user begins a session with the DA-MSA by hand-coding an initial simulation model and optimization strategy. The initial model includes the most accurate version of each function. The initial strategy includes only a single stage of optimization. The user subsequently modifies the initial model and strategy using a catalog of transformations. Each transformation replaces one dataflow graph (or sub-graph) with a new one. Transformations implement a variety of changes, such as substituting one version of a function for another, approximating functions and introducing multiple stages of optimization. For example, the system includes transformations that replace a single-stage strategy with multi-stage strategy MLM_1 , MLM_2 or MLM_3 . It also includes transformations that construct each of the locally calibratable objective functions $\hat{e}_1(\bar{x}, \bar{y}), \dots, \hat{e}_6(\bar{x}, \bar{y})$ shown in Figure 8. A complete description of the DA-MSA transformation system is beyond the scope of this paper. For additional information, see [Ellman *et al.*, 1995],[Ellman *et al.*, 1996].

5.2 Setup of Experiments in the Yacht Domain

Our experiments in the yacht domain were intended to address the following questions: (1) How do the multi-model strategies compare to two alternative strategies: (a) optimization using only the “exact” objective function; (b) optimization using only an approximate objective function? (2) How do the multi-model strategies MLM_1 , MLM_2 and MLM_3 compare to each other? (3) How do the calibratable approximation schemes $\hat{e}_1(\bar{x}, \bar{y}), \dots, \hat{e}_6(\bar{x}, \bar{y})$ compare to each other?

We set up our experiments by using the DA-MSA to construct several versions of strategies MLM_1 , MLM_2 and MLM_3 , each instantiated in the domain of sailing yacht design. We chose the effective draft computation as a focus for experimenting with approximations in the yacht domain. The reasons for this choice are as follows: The most expensive parts of an “exact” course time computation are the PMARC (effective draft) and SLAW (wave resistance) CFD codes; however, for the class of sailing yachts that includes the Stars and Stripes ’87, the curves fitted to tank data give nearly as accurate answers as the SLAW code, but at much lower cost. Yachts in this class can be designed equally well using SLAW or the fitted wave resistance curves. In contrast, the algebraic approximation of effective draft is not nearly as accurate as PMARC. Yachts in this class cannot

be properly designed using the algebraic approximation alone. Considerations of cost-effectiveness thus led us to focus on multi-model strategies involving PMARC and the algebraic approximation of effective draft. When we constructed the fixed approximation $\tilde{e}(\bar{x})$ and the locally calibratable approximations $\hat{e}_1(\bar{x}, \bar{y}), \dots, \hat{e}_6(\bar{x}, \bar{y})$, we let the PMARC effective draft model play the role of $F_i(\bar{x})$ (the internal quantity to be approximated) and we let the algebraic formula for effective draft play the role of $\tilde{F}_i(\bar{x})$ (the fixed internal approximation). We used forward differencing to implement the gradient computations required in approximation schemes $\hat{e}_2(\bar{x}, \bar{y}), \hat{e}_4(\bar{x}, \bar{y})$ and $\hat{e}_6(\bar{x}, \bar{y})$.

We conducted two groups of experiments. The first group was carried out using an interpolated version of the function $D(\bar{x})$ that computes effective draft using PMARC. The interpolated version of $D(\bar{x})$ was constructed by doing $3^5 = 243$ PMARC runs to generate a table mapping design parameters to effective draft. The table has one dimension for each of the five yacht design parameters: *Length*, *Beam*, *Draft*, *KeelHeight* and *WingletSpan*. The table was then used to construct a cubic-spline interpolation function that computes effective draft for arbitrary values of these parameters. The interpolated version of $D(\bar{x})$ was needed in order to experiment with strategies that use random seeds to initialize optimization. It allowed us to conduct a relatively large number of experimental optimization runs and to average the results. The table of PMARC evaluations is necessary only as a part of our experimental apparatus. The expensive fitting process itself does not occur in any of the optimization strategies our experiments are designed to evaluate. The second group of experiments was carried out using PMARC itself under the control of various design optimization strategies. In both groups of experiments, the necessary PMARC runs were set up using a fully automatic panelization system [Yao and Gelsey, 1996]. The panelization program was run on a Sun Microsystems Sparcstation 2. PMARC itself was run on (one processor of) a three-processor DEC Alpha 2100. Each PMARC run required approximately one hour of total CPU time to do the panelization and run the flow code.

We used CFSQP to perform all the underlying numerical optimizations in our experiments. CFSQP carried out both the first and second stages of optimization in each experiment with strategy MLM_1 . Likewise, CFSQP carried out the inner optimizations in each experiment with strategy MLM_2 . In addition, CFSQP was used to carry out all the single-level optimizations, which we used as a baseline to measure the relative speedup obtained by our multi-level strategies. Although CFSQP is capable of solving constrained optimization problems, we used CFSQP as an unconstrained optimizer. As a formality, the CFSQP code requires us to supply upper and lower bounds on each design parameter; however, we have observed experimentally that the bounds do not actively constrain the solutions we find.

5.3 Results from Interpolated PMARC

A portion of our results from experiments using the interpolated version of PMARC are found in Tables 1 and 2. These tables compare the performance of several strategies on the following two problems: Design a yacht with minimum course time on an America’s Cup race course sailing in a 10 kt wind, and a 20 kt wind.² The row labeled MLM_1 refers to a strategy that uses two stages of optimization. Stage one uses the algebraic formula for effective draft. Stage two uses interpolated PMARC. The row labeled “Pure-Algebraic” refers to a single-stage strategy that uses

²Our VPP program does not model tacking and therefore cannot immediately handle race courses that involve sailing directly upwind. We therefore modified the America’s Cup course by replacing the upwind leg with one that serves to estimate the crew’s tacking strategy.

Strategy	PMARC Evals (Per Seed)	Course Time (Median)	Seed Designs $N(0.99, 0.01)$	PMARC Evals (Total)
<i>Pure-Algebraic</i>	1.0	3.363	-	-
<i>Pure-PMARC</i>	80.00	3.250	5.61	448.75
<i>MLM₁</i>	24.52	3.247	2.86	70.16

Table 1: Comparison of Strategies Using Interpolated PMARC (10 kt wind)

Strategy	PMARC Evals (Per Seed)	Course Time (Median)	Seed Designs $N(0.99, 0.01)$	PMARC Evals (Total)
<i>Pure-Algebraic</i>	1.0	2.402	-	-
<i>Pure-PMARC</i>	100.88	2.363	6.27	632.52
<i>MLM₁</i>	42.56	2.364	4.51	191.95

Table 2: Comparison of Strategies Using Interpolated PMARC (20 kt wind)

the algebraic formula to compute effective draft. Likewise, the row labeled “Pure-PMARC” refers to a single-stage strategy that uses interpolated PMARC to compute effective draft. All strategies were run multiple times using 25 randomly selected seeds to initialize each run. For each strategy, we recorded the average number of PMARC evaluations per seed and the median value of the final course time. We also computed a statistic that estimates the performance of each strategy when used repeatedly with multiple seed designs. The statistic $N(p, \epsilon)$ represents the number of seed designs that each strategy would require in order to have a probability p of finding a design whose course time is within a fraction ϵ of the best design found for this problem. Using elementary probability theory:

$$N(p, \epsilon) = \frac{\log(1 - p)}{\log(1 - r(\epsilon))}$$

where $r(\epsilon)$ is the probability that an optimization from a single seed results in a design within ϵ of the best design. The value of $r(\epsilon)$ for each strategy was obtained from a histogram of the final course times resulting from the 25 seeds used in the experiments. A missing entry for $N(p, \epsilon)$ indicates that $r(\epsilon) = 0$ in the experimental data, i.e., none of the seeds resulted in a design within ϵ of the best design. An estimate of the overall CPU time needed to come within ϵ of the best design results from multiplying $N(p, \epsilon)$ by the average number of PMARC evaluations per seed. This product is recorded in the last column of Tables 1 and 2.

Notice first that the “Pure-Algebraic” strategy failed to find a design within 1% of the best on any of the 25 trials, on either of the test problems. (In the case of the “Pure-Algebraic” strategy, the table records the average final course time as measured using interpolated PMARC, and records the one interpolated PMARC evaluation conducted at the end of each optimization to evaluate the final design.) On the first test problem (10 kt wind) the “Pure-PMARC” strategy requires an average of 448.75 PMARC evaluations to have 99% chance of coming within 1% of the best design. In contrast, the two-model, two-stage strategy *MLM₁* requires an average of 70.16 PMARC

Strategy	PMARC Evals (Per Seed)	Inner-Opts. (Per Seed)	Course Time (Median)	Seed Designs $N(0.99, 0.01)$	PMARC Evals (Total)
$MLM_2 : \hat{e}_1$	8.64	2.40	3.570	-	-
$MLM_2 : \hat{e}_2$	13.24	2.08	3.258	6.27	83.07
$MLM_2 : \hat{e}_3$	2.60	2.32	3.253	4.51	11.72
$MLM_2 : \hat{e}_4$	13.2	2.12	3.259	6.27	82.82
$MLM_2 : \hat{e}_5$	2.72	2.44	3.257	4.51	12.26
$MLM_2 : \hat{e}_6$	15.24	2.40	3.248	5.61	85.49

Table 3: Comparison of Strategies Using Interpolated PMARC (10 kt wind)

evaluations to achieve the same degree of reliability, i.e., only about 15.63% of the CPU time, or a reduction factor of 6.40 in comparison to the Pure-PMARC strategy. On the second test problem (20 kt wind) the “Pure-PMARC” strategy requires an average of 632.52 PMARC evaluations to have 99% chance of coming within 1% of the best design. In contrast, the two-model, two-stage strategy MLM_1 requires an average of 191.95 PMARC evaluations to achieve the same degree of reliability, i.e., only about 30.35% of the CPU time, or a reduction factor of 3.30 in comparison to the Pure-PMARC strategy.

Additional results from experiments using the interpolated version of PMARC are presented in Tables 3 and 4. This data was obtained using the same experimental setup as described above (America’s Cup Course, 10 and 20 kt winds, 25 seeds for each strategy). These tables compare the performance of various instantiations of MLM_2 , the recalibrating strategy. The rows labeled $MLM_2 : \hat{e}_1, \dots, MLM_2 : \hat{e}_6$ contain results of using strategy MLM_2 instantiated with the corresponding one of the six locally calibratable approximation schemes $\hat{e}_1(\bar{x}, \bar{y}), \dots, \hat{e}_6(\bar{x}, \bar{y})$, to combine PMARC and the algebraic formula. Consider first how well each strategy performed in terms of design quality. Strategy $MLM_2 : \hat{e}_1$ failed to come within 1% of the best solution on all 25 trials, on either of the test problems. All of the other strategies, $MLM_2 : \hat{e}_2, \dots, MLM_2 : \hat{e}_6$ came within 1% of the best on over half of the 25 trials, on both of the test problems. Consider now how well each strategy performed in terms of CPU time, i.e., number of (interpolated) PMARC evaluations per seed. Strategies $MLM_2 : \hat{e}_3$ and $MLM_2 : \hat{e}_5$ were the best in this respect, each requiring about three PMARC evaluations per seed on the 10 kt wind problem, and about seven PMARC evaluations per seed on the 20 kt wind problem. Finally, consider the number of PMARC evaluations needed by each strategy to have a 99% probability of getting within 1% of the best design. Using this measure, the best performance results from strategy $MLM_2 : \hat{e}_3$ (an average of 11.72 PMARC evaluations on the 10 kt wind problem and 29.95 PMARC evaluations on the 20 kt wind problem) and from strategy $MLM_2 : \hat{e}_5$ (an average of 12.26 PMARC evaluations on the 10 kt wind problem and 36.42 PMARC evaluations on the 20 kt wind problem). In comparison to the Pure-PMARC strategy, $MLM_2 : \hat{e}_3$ requires only 2.61% as much CPU time on the 10 kt wind problem and 4.73% as much CPU time on the 20 kt wind problem. Likewise, in comparison to the Pure-PMARC strategy, $MLM_2 : \hat{e}_5$ requires only 2.73% as much CPU time on the 10 kt wind problem and 5.76% as much CPU time on the 20 kt wind problem. Summarizing the results in Tables 3 and 4, we see that strategies $MLM_2 : \hat{e}_2 \dots MLM_2 : \hat{e}_6$ require from 2.61% to 19.94 as much CPU time, or reduction factors ranging from 38.3 to 5.01, in comparison to the Pure-PMARC strategy.

Strategy	PMARC Evals (Per Seed)	Inner-Opts. (Per Seed)	Course Time (Median)	Seed Designs $N(0.99, 0.01)$	PMARC Evals (Total)
$MLM_2 : \hat{e}_1$	12.12	3.72	2.640	-	-
$MLM_2 : \hat{e}_2$	18.96	3.00	2.364	6.27	118.88
$MLM_2 : \hat{e}_3$	6.64	3.80	2.363	4.51	29.95
$MLM_2 : \hat{e}_4$	20.12	3.16	2.366	6.27	126.15
$MLM_2 : \hat{e}_5$	7.24	4.36	2.364	5.03	36.42
$MLM_2 : \hat{e}_6$	15.92	2.52	2.363	5.61	89.31

Table 4: Comparison of Strategies Using Interpolated PMARC (20 kt wind)

Strategy	PMARC Evals	Inner Opts.	Course Time
$MLM_3 : \text{Pure-PMARC}$	25	-	3.280
$MLM_3 : \hat{e}_1$	12	1	3.285
$MLM_3 : \hat{e}_2$	12	2	3.273
$MLM_3 : \hat{e}_3$	2	2	3.275
$MLM_3 : \hat{e}_4$	8	1	3.285
$MLM_3 : \hat{e}_5$	2	2	3.279
$MLM_3 : \hat{e}_6$	7	1	3.285

Table 5: Performance of Multi-Model Strategies Using Real PMARC (10 kt wind)

The numbers of PMARC evaluations used by strategies $MLM_2 : \hat{e}_1, \dots, MLM_2 : \hat{e}_6$ require explanation. (See Tables 3 and 4.) The approximation schemes \hat{e}_1, \hat{e}_3 and \hat{e}_5 each require one PMARC evaluation per calibration. The approximation schemes \hat{e}_2, \hat{e}_4 and \hat{e}_6 each require six PMARC evaluations per calibration (one plus the number of design parameters) in order to numerically compute the gradient of effective draft. If we multiply the average number of inner optimizations per seed by the number of PMARC evaluations needed per calibration, we get a result that is *lower* than the average number of PMARC evaluations per seed that were actually used. The difference is due to evaluations of PMARC that occurred during the *LineSearch* procedure of strategy MLM_2 . In some cases, line searches resulted in more PMARC evaluations than were needed for calibration.

5.4 Results from Real PMARC

Results from a set of experiments using PMARC itself are found in Tables 5 and 6. This data was generated using the same test problems as described above. The rows labeled $MLM_3 : \hat{e}_1, \dots, MLM_3 : \hat{e}_6$ contain results of using strategy MLM_3 , i.e., a two-stage strategy that optimizes $N = 25$ randomly generated seeds using the fixed approximation $\tilde{e}(\bar{x})$ and then optimizes the best result using strategy MLM_2 . In each case, the first stage used the algebraic formula for effective draft. In each case, the second stage used one of the six locally calibratable approximation schemes $\hat{e}_1(\bar{x}, \bar{y}), \dots, \hat{e}_6(\bar{x}, \bar{y})$ to combine PMARC and the algebraic formula. Finally, the row labeled $MLM_3 : \text{Pure-PMARC}$ contains results from a variation of MLM_3 . In this variation, the second stage (using MLM_2) was replaced with a single optimization using PMARC to compute effective draft. Notice that all of the strategies found designs within 1% of the best design on the 10 kt wind

Strategy	PMARC Evals	Inner Opts.	Course Time
$MLM_3 : \textit{Pure-PMARC}$	50	-	2.376
$MLM_3 : \hat{e}_1$	15	2	2.357
$MLM_3 : \hat{e}_2$	20	3	2.372
$MLM_3 : \hat{e}_3$	4	3	2.363
$MLM_3 : \hat{e}_4$	12	2	2.363
$MLM_3 : \hat{e}_5$	15	4	2.350
$MLM_3 : \hat{e}_6$	13	2	2.380

Table 6: Performance of Multi-Model Strategies Using Real PMARC (20 kt wind)

Strategy	PMARC Evals (Per Seed)	Inner-Opts. (Per Seed)	Course Time (Median)	Seed Designs $N(0.99, 0.01)$	PMARC Evals (Total)
$MLM_2 : \hat{e}_3$ (S)	2.04	1.00	3.257	5.03	10.25
$MLM_2 : \hat{e}_3$ (M)	2.60	2.32	3.253	4.51	11.72
$MLM_2 : \hat{e}_5$ (S)	2.04	1.00	3.262	6.27	12.80
$MLM_2 : \hat{e}_5$ (M)	2.72	2.44	3.257	4.51	12.26

Table 7: Comparison of Single and Multiple Recalibration Using Interpolated PMARC (10 kt wind)

problem, and within 2% of the best design on the 20 kt wind problem. Strategies $MLM_3 : \hat{e}_3$ and $MLM_3 : \hat{e}_5$ were the fastest on the 10 kt wind problem, requiring only two PMARC evaluations, i.e., 8.0% of the 25 PMARC evaluations used by strategy $MLM_3 : \textit{Pure-PMARC}$. Strategy $MLM_3 : \hat{e}_3$ was the fastest on the 20 kt wind problem, requiring only four PMARC evaluations, i.e., 8.0% of the fifty PMARC evaluations used by strategy $MLM_3 : \textit{Pure-PMARC}$. Notice also that strategy $MLM_3 : \textit{Pure-PMARC}$ did not return the best design on either of the two test problems. We can explain this result in the following way: PMARC involves lots of numerical discretization. This introduces discontinuities and nonsmoothness into any objective function that calls PMARC. These pathologies apparently cause strategy $MLM_3 : \textit{Pure-PMARC}$ to stop at a false local optimum. In contrast, a locally calibrated approximation $\hat{e}_s(\bar{x})$ has relatively fewer pathologies, because it does not call PMARC (after calibration). The relative absence of pathology apparently enables some of the recalibrating strategies to avoid getting stuck at false local optima.

5.5 Evaluating the Impact of Recalibration

We conducted an additional set of experiments to evaluate the importance of recalibration of approximate objective functions. In particular, we modified the recalibrating strategies reported in Tables 3, 4, 5 and 6 so that each would terminate after a single calibration of the objective function, and a single inner optimization. Results from running these modified strategies on the same test problems as described above are summarized in Tables 7 and 8 (interpolated PMARC) and Tables 9 and 10 (real PMARC). The rows labeled ‘‘S’’ correspond to the strategies that do a single calibration. The rows labeled ‘‘M’’ correspond to the strategies that perform multiple calibrations/recalibrations. (The ‘‘M’’ data was simply copied from Tables 3 and 4 and Tables 5 and 6

Strategy	PMARC Evals (Per Seed)	Inner-Opts. (Per Seed)	Course Time (Median)	Seed Designs $N(0.99, 0.01)$	PMARC Evals (Total)
$MLM_2 : \hat{e}_3$ (S)	2.24	1.00	2.412	16.78	37.59
$MLM_2 : \hat{e}_3$ (M)	6.64	3.80	2.363	4.51	29.95
$MLM_2 : \hat{e}_5$ (S)	2.20	1.00	2.415	14.02	30.84
$MLM_2 : \hat{e}_5$ (M)	7.24	4.36	2.364	5.03	36.42

Table 8: Comparison of Single and Multiple Recalibration Using Interpolated PMARC (20 kt wind)

Strategy	PMARC Evals	Inner Opts.	Course Time
$MLM_3 : \hat{e}_3$ (S)	2	1	3.275
$MLM_3 : \hat{e}_3$ (M)	2	2	3.275
$MLM_3 : \hat{e}_5$ (S)	2	1	3.279
$MLM_3 : \hat{e}_5$ (M)	2	2	3.279

Table 9: Comparison of Single and Multiple Recalibration using Real PMARC (10 kt wind)

above.) First consider the data obtained using interpolated PMARC (Tables 7 and 8). Notice that the recalibrating strategies found better designs than the strategies that perform a single calibration, at the price of additional PMARC computations. An estimate of the number of PMARC evaluations needed to have 99% probability of getting within 1% of the best design is found in the last column of the table. According to this measure, the recalibrating strategies are comparable to the strategies that perform a single calibration, each requiring about 10 to 13 PMARC evaluations for the 10 kt wind problem, and about 30 to 38 PMARC evaluations for the 20 kt wind problem. Now consider the data obtained using real PMARC (Tables 9 and 10). On the 10 kt wind problem, the same overall numbers of PMARC evaluations were used by the recalibrating strategies and the strategies performing a single calibration. In addition, the single calibrating strategies found designs that were identical to the corresponding recalibrating strategies, even though the recalibrating strategies carried out an additional inner optimization. This result is explained by the fact that in both cases the second inner optimization failed to make any improvement over the seed design, and thus returned the seed as the solution. On the 20 kt wind problem, recalibration made a difference. Both of the recalibrating strategies resulted in better designs than the strategies performing a single calibration, at the price of additional PMARC computations.

Strategy	PMARC Evals	Inner Opts.	Course Time
$MLM_3 : \hat{e}_3$ (S)	3	1	2.379
$MLM_3 : \hat{e}_3$ (M)	4	3	2.363
$MLM_3 : \hat{e}_5$ (S)	11	1	2.382
$MLM_3 : \hat{e}_5$ (M)	15	4	2.350

Table 10: Comparison of Single and Multiple Recalibration using Real PMARC (20 kt wind)

5.6 Conclusions to Be Drawn from Experimental Results

Our experiments support the following general conclusions about the performance of our multi-model strategies in the sailing yacht domain. The best performing strategies used the recalibratable approximation schemes \hat{e}_3 and \hat{e}_5 . These schemes use both the algebraic formula and PMARC to compute effective draft in searching for a good design, i.e., they really use multiple models. They outperformed the Pure-PMARC strategy and the approximation schemes \hat{e}_1 and \hat{e}_2 , all of which use only a single model (PMARC) and fail to utilize the algebraic formula at all. Our results thus demonstrate the value of using multiple models in a design optimization process. Approximation schemes \hat{e}_3 and \hat{e}_5 (which assume locally constant error) also outperformed approximation schemes \hat{e}_4 and \hat{e}_6 (which assume locally linear error). This might be surprising, considering that schemes \hat{e}_3 and \hat{e}_5 both have order zero, while schemes \hat{e}_4 and \hat{e}_6 both have order one. Apparently the algebraic formula for effective draft by itself does a reasonably good job of fitting the gradient of the true effective draft, as computed by PMARC. When instantiated in the yacht domain, using the algebraic formula, the approximation schemes \hat{e}_3 and \hat{e}_5 fortuitously have order one. The computational expense incurred in schemes \hat{e}_4 and \hat{e}_6 for fitting the gradient is apparently not justified by improved fitting accuracy. (Since we used forward-differencing to compute these gradients, we suspect that our results might have been different if a more efficient technique for computing the gradient of effective draft had been available.) Finally, recalibration was sometimes important to the success of our multi-model strategies. On one of our two test problems, the strategies that periodically recalibrate approximations found better designs than strategies that simply perform a single initial calibration.

Our experiments relied on CFSQP to carry out all the underlying numerical optimizations required by the strategies we tested. Of course, we could have used any one of a large number of other codes for this purpose. A good survey of optimization software is found in [Moré and Wright, 1993]. One might ask whether our results would have been different if we had used a different numerical optimization code. In order to address this question, consider what would happen if all optimization codes returned the same solution when started from the same seed point. In such an ideal situation, the choice of optimization code would not impact the input/output behavior of the inner optimization routine used in strategies MLM_2 and MLM_3 . The choice of code would also not affect the sequence of evaluations of the “exact” objective function $e(\bar{x})$ invoked by strategies MLM_2 and MLM_3 , since $e(\bar{x})$ is not used in the internal optimization steps of these strategies. Since our cost metric ignores evaluations of the approximate objective function $\hat{e}(\bar{x}, \bar{y})$, the choice of optimization code would also not affect the overall cost of running either strategy MLM_2 or strategy MLM_3 , as measured by the number of evaluations of $e(\bar{x})$. Therefore the actual algorithm used for each internal optimization would be irrelevant to the absolute performance of strategies MLM_2 and MLM_3 . Of course, the choice of optimization code would still influence the cost of a single-level strategy, which we used as a baseline to measure the relative speedup obtained by our multi-level strategies. Furthermore, in real life, different numerical codes will return different solutions, even when started from the same seed. For this reason, we cannot guarantee that our experiments would have demonstrated the same relative improvements in performance if we switched from CFSQP to another numerical optimization code.

One might also ask whether our results would have been different if we had carried out experiments in some application domain other than yacht design. We cannot provide a definite answer, without actually trying our methods on some other application. On the other hand, we can try to

identify the features of the yacht design problem that enabled our methods to be successful. In the yacht domain, the “exact” objective function $e(\bar{x})$ includes one subroutine (PMARC) that dominates the cost of evaluation. This subroutine computes a function (effective draft) that is simple enough to be approximated locally by an algebraic formula. In addition, most of the nonlinearity in the objective function $e(\bar{x})$ results from subroutines other than the one being approximated. These facts characterize the situations in which we expect *internal approximations* to be most cost-effective. We therefore expect our methods to be successful on problems that exhibit these characteristics.

6 Related Work

A considerable amount of research on multi-level design optimization has been carried out by investigators working on structural design or multi-disciplinary design in the aerospace community [Sobieszczanski-Sobieski, 1982], [Sobieszczanski-Sobieski and Hatfka, 1996]. Most of this research has focused on problems in which optimization is computationally expensive due more to the presence of many design parameters than to the expense of individual evaluations of objective or constraint functions. Accordingly, such work has focused on decomposing design spaces into nearly independent subspaces. In this work, each “level” of a multi-level design process corresponds to a factor space of the original design space. In contrast, our work is focused on problems in which optimization is computationally expensive due more to the cost of evaluation than to the presence of many design parameters. We leave the design space fixed. In our work, each “level” corresponds to a model of the objective function that uses a particular combination of approximations.

A statistical approach to multi-level engineering design is presented in [Osio and Amon, 1996]. This technique applies to problems for which a series M^1, \dots, M^q of computational models of increasing accuracy and cost is available to evaluate candidate designs. It operates by constructing a series Y^1, \dots, Y^q of “surrogate models”. During the i th stage of operation, the model M^i is evaluated at a set of design points. These points are chosen by a data-adaptive optimal sampling technique. The evaluations are used to construct the surrogate Y^i through a Bayesian updating process in which the previous surrogate Y^{i-1} is used to define a prior distribution. This approach is similar to ours in the sense that it uses cheap, approximate models to select points at which more expensive, accurate models will be evaluated. It differs from ours in the sense that sampling is focused on regions of the design space where errors are large and systematic, regardless of whether such regions contain good designs. In contrast to this, our system uses successive optimizations to focus sampling on regions of the design space that are close to locally optimal designs.

A method of constructing and optimizing approximate objective functions is outlined in [Vanderplaats, 1984]. The method operates by moving through a design space and evaluating the exact objective function $e(\bar{x})$ at a series $(\bar{s}_0, \bar{s}_1, \dots, \bar{s}_n)$ of points. It periodically uses the evaluations of these points to fit an approximate version $\hat{e}(\bar{x}, \bar{s}_0, \bar{s}_1, \dots, \bar{s}_n)$ of the objective function. Depending on whether the number k of free parameters in the class of fitting functions, the fitting process may either interpolate the evaluations of $(\bar{s}_0, \bar{s}_1, \dots, \bar{s}_n)$ exactly, or do a least squares approximation. In either case, the approximation $\hat{e}(\bar{x}, \bar{s}_0, \bar{s}_1, \dots, \bar{s}_n)$ is then used in a fast optimization process, resulting in one or more new design points and a new fitting process, etc. This technique is similar to ours in the sense that it involves a process of interleaved fitting and optimizing of approximate objective functions. There are two main differences. Our technique is based on internal approximations,

rather than approximations of the objective function as a whole. Furthermore, our technique uses multiple domain-specific models in the fitting process, rather than a single domain-specific model in combination with a domain-independent fitting technique.

Our recalibrating strategies are similar in spirit to numerical continuation methods for solving equations or optimizing functions [Allgower and Georg, 1990]. A continuation method for finding a root of a function $\bar{f}(\bar{x})$ uses a continuous function $\bar{g}(\bar{x}, p)$ to define a family of approximations parameterized by p in much the same manner as a locally calibratable approximation scheme $\hat{e}(\bar{x}, \bar{y})$ defines a family of approximations parameterized by \bar{y} . The family $\bar{g}(\bar{x}, p)$ is chosen so that $\bar{f}(\bar{x})$ is the restriction of $\bar{g}(\bar{x}, p)$ to the subspace defined by $p = 0$, and so that a solution to the equation $\bar{g}(\bar{x}, 1) = \bar{0}$ is known or easy to find. A solution to the original problem is found by gradually varying p from 1 to 0 and repeatedly solving the equation $\bar{g}(\bar{x}, p) = \bar{0}$, using the previous solution as a starting point. The chief difference between our recalibration method and a standard continuation method lies in how the fitting parameter p is used. In a continuation method, the fitting parameter p is controlled by an independent process. In our recalibration method, the parameter \bar{y} is controlled by the optimization strategy itself.

Research on knowledge-based design optimization has been pursued by several investigators in the Artificial Intelligence community. One portion of this work has focused on automating the choice of the underlying numerical algorithm, and numerical parameters to that algorithm [Orelup *et al.*, 1988]. Another portion of this work has attempted to use numerical methods in combination with rule-based inference to guide the search for an optimal design [Tong, 1990], [Powell, 1990]. This work is similar to ours inasmuch as we are both motivated, in part, by a desire to deal with pathological objective functions. In contrast to our work, however, these efforts have adhered to a paradigm in which there is a single model of the objective function, which remains fixed during the optimization process.

A method of adapting standard optimization techniques to deal with modeling error is reported in [Cagan and Williams, 1993]. In this work, the authors develop a modification of the Karush-Kuhn-Tucker conditions for local optimality. In particular, they present a method of modifying the objective function (for unconstrained optimization) and the Lagrangian (for constrained optimization). Their modification allows one to include robustness of the design with respect to modeling error as part of the objective to be optimized. The method relies on human supplied weights to decide how to balance competing concerns of optimal performance and robustness. This work is similar to our own inasmuch as it provides a means of utilizing approximate models in an optimization process; however, it differs in focusing on use of a single model rather than multiple models. Furthermore, it aims to improve robustness of the design rather than lowering the computation cost of the design process.

Methods of reasoning with multiple models have been investigated by the Qualitative Physics community. Some of this work is similar to ours in that it has studied methods of dynamically selecting among multiple models to choose one that is well suited to the task at hand. Most of this work has taken considerations of gross relevance as the sole criterion for selecting among models [Falkenhainer and Forbus, 1991], [Nayak, 1994]. In this context, methods of dependency tracing suffice for deciding on which model to use to answer a given query. More sophisticated methods reason about the sign of the error of approximate models [Addanki *et al.*, 1991], [Weld, 1990]. In contrast to our work, considerations of numerical accuracy are not used at all to guide the choice of a suitable model.

Methods of using multiple models to assure the quality of numerical simulations are reported

by our colleague in [Gelsey, 1995]. This work is similar to ours inasmuch as we both used the yacht domain and the PMARC code as experimental testbeds; however, this work focused on improving the reliability of simulations of individual designs. Unlike our work, it did not address the use of multiple models in an overall design process.

In our own previous work [Ellman *et al.*, 1993] we developed a technique called “Gradient Magnitude Model Selection” (GMMS) for dynamically choosing between exact and approximate models during design optimization. GMMS uses explicit accuracy estimates to decide which model to use for each numerical comparison of evaluations that takes place during a design optimization process. The utility of GMMS is limited by the fact that it requires a special numerical optimization algorithm. It cannot be combined with an arbitrary numerical optimization code without extensive rewriting of that code. In our present work, we have attempted to formulate multi-model strategies that can be more easily integrated with arbitrary numerical optimization codes, i.e., by calling such codes as black-box subroutines.

7 Summary

We have developed, analyzed and tested a family of strategies for using multiple models to optimize engineering designs. Our strategies are useful when multiple approximations of the objective function can be implemented using compositional modeling techniques. Compositional modeling is important in this context, because it enables selective approximation of internal components of the objective function. We showed how a compositional modeling library can be used to construct a variety of locally calibratable approximation schemes, each of which can be used in our family of optimization strategies. We analyzed the schemes and strategies to formulate and prove sufficient conditions for correctness and convergence. Correctness depends in part on whether the internal approximation accurately fits the gradient of the function being approximated. Convergence depends, roughly, on the distribution of nonlinearity in the objective function, and the degree of nonlinearity in the error of the internal approximation. We also tested our approximation schemes and optimization strategies experimentally in the domain of sailing yacht design. Our results demonstrate that our strategies achieve dramatic reductions, ranging from a factor of five to a factor of thirty-eight, in the CPU time required for optimization, on the sailing yacht problems we tested, with no significant loss in design quality. The greatest reduction depends on the use of multiple models of internal components of the objective function. Preservation of design quality depends on the use of periodic recalibration of approximations during the optimization process.

8 Acknowledgments

The research reported in this paper is supported by the Hypercomputing and Design Project at Rutgers University, which is sponsored by the Advanced Research Projects Agency of the Department of Defense through contract ARPA-DABT 63-93-C-0064. It has benefited from discussions with Saul Amarel, Gene Bouchard, Vasek Chvatal, Martin Fritts, Andrew Gelsey, Haym Hirsh, Leonid Kachiyan, John Letcher, Mike Meinhold, Gerry Richter, Nils Salvesen, Don Smith, Lou Steinberg and the anonymous referees.

References

- [Addanki *et al.*, 1991] S. Addanki, R. Cremonini, and J. Scot. Graphs of models. *Artificial Intelligence*, 50, 1991.
- [Allgower and Georg, 1990] E. Allgower and K. Georg. *Numerical Continuation Methods*. Springer-Verlag, 1990.
- [Ashby *et al.*, 1992] Dale L. Ashby, Michael R. Dudley, Steve K. Iguchi, Lidsey Browne, and Joseph Katz. *Potential Flow Theory and Operation Guide for the Panel Code PAMRC 12*, 1992.
- [Cagan and Williams, 1993] J. Cagan and B. Williams. First order necessary conditions for robust optimality. In *Proceedings of the ASME Design Automation Conference*, 1993.
- [Conte and de Boor, 1980] S. Conte and C. de Boor. *Elementary Numerical Analysis: An Algorithmic Approach*. McGraw-Hill, 1980.
- [Dahlquist and Bjorck, 1974] G. Dahlquist and A. Bjorck. *Numerical Methods*. Prentice-Hall, 1974.
- [Dennis and Schnabel, 1983] J. Dennis and R. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, New Jersey, 1983.
- [Ellman *et al.*, 1992] T. Ellman, J. Keane, and M. Schwabacher. The Rutgers CAP project design associate. Technical Report CAP-TR-7, Department of Computer Science, Rutgers University, New Brunswick, NJ, 1992.
- [Ellman *et al.*, 1993] T. Ellman, J. Keane, and M. Schwabacher. Intelligent model selection for hillclimbing search in computer-aided design. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, DC, 1993.
- [Ellman *et al.*, 1995] T. Ellman, J. Keane, M. Schwabacher, and T. Murata. A transformation system for interactive reformulation of design optimization strategies. In *Proceedings of the Tenth Knowledge-Based Software Engineering Conference*, Boston, MA, 1995.
- [Ellman *et al.*, 1996] T. Ellman, J. Keane, A. Banerjee, and G. Armhold. A transformation system for interactive reformulation of design optimization strategies. Accepted with revisions to *Research in Engineering Design*, 1996.
- [Falkenhainer and Forbus, 1991] B. Falkenhainer and K. Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 50, 1991.
- [Gelsey, 1995] A. Gelsey. Intelligent automated quality control for computational simulation. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 9, 1995.
- [Gill *et al.*, 1981] P. Gill, W. Murray, and M. Wright. *Practical Optimization*. Academic Press, London, England, 1981.
- [Kachiyani, 1997] L. Kachiyani. Personal communication. 1997.

- [Katz and Plotkin, 1991] Joseph Katz and Allen Plotkin. *Low-speed aerodynamics: From wing theory to panel methods*. McGraw-Hill, 1991.
- [Lawrence *et al.*, 1995] C. Lawrence, J. Zhou, and A. Tits. User's guide for CFSQP version 2.3: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Technical Report TR-94-16r1, Institute for Systems Research, University of Maryland, August 1995.
- [Letcher *et al.*, 1987] J. Letcher, J. Marshall, J. Oliver, and N. Salvesen. Stars and Stripes. *Scientific American*, 257(2), August 1987.
- [Letcher, 1975] J. Letcher. Sailing hull hydrodynamics, with reanalysis of the Antiope data. *Transactions of the Society of Naval Architects and Marine Engineers*, 83, 1975.
- [Letcher, 1991] J. Letcher. *The Aero/Hydro VPP Manual*. Aero/Hydro, Inc., Southwest Harbor, ME, 1991.
- [Moré and Wright, 1993] Jorge J. Moré and Stephen J. Wright. *Optimization Software Guide*. SIAM, Philadelphia, 1993.
- [Nayak, 1994] P. Nayak. Causal approximations. *Artificial Intelligence*, 70, 1994.
- [Newman and Wu, 1973] J. Newman and T. Wu. A generalized slender body theory for fish-like forms. *J. Fluid Mechanics*, 57(4), 1973.
- [Orelup *et al.*, 1988] M. F. Orelup, J. R. Dixon, P. R. Cohen, and M. K. Simmons. Dominic ii: Meta-level control in iterative redesign. In *Proceedings of the National Conference on Artificial Intelligence*, pages 25–30, St. Paul, MN, 1988.
- [Osio and Amon, 1996] G. Osio and C. Amon. An engineering design methodology with multistage Bayesian surrogates and optimal sampling. *Research in Engineering Design*, 8:189–206, 1996.
- [Pearl, 1984] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading, MA, 1984.
- [Peressini *et al.*, 1988] A. Peressini, F. Sullivan, and J. Uhl. *The Mathematics of Nonlinear Programming*. Springer-Verlag, New York, NY, 1988.
- [Powell, 1990] D. Powell. Inter-gen: A hybrid approach to engineering design optimization. Technical report, Rensselaer Polytechnic Institute, Department of Computer Science, December 1990. Ph.D. Thesis.
- [Press *et al.*, 1986] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes*. Cambridge University Press, New York, NY, 1986.
- [Rudin, 1964] W. Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 1964.
- [Sacerdoti, 1974] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

- [Simon, 1981] H. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, MA, 1981.
- [Sobieszczanski-Sobieski and Hatfka, 1996] J. Sobieszczanski-Sobieski and R. Hatfka. Multidisciplinary aerospace design optimization: Survey of recent developments. Technical Report AIAA 96-0711, American Institute of Aeronautics and Astronautics, 1996.
- [Sobieszczanski-Sobieski, 1982] J. Sobieszczanski-Sobieski. A linear decomposition method for large optimization problems—blueprint for development. Technical Report NASA TM-83248, National Aeronautics and Space Administration, 1982.
- [Tong, 1990] S. S. Tong. Coupling symbolic manipulation and numerical simulation for complex engineering designs. In *Intelligent Mathematical Software Systems*, pages 241–252. North-Holland, New York, NY, 1990.
- [Vanderplaats, 1984] G. Vanderplaats. *Numerical Optimization Techniques for Engineering Design: With Applications*. McGraw-Hill, New York, New York, 1984.
- [Weems *et al.*, 1994] K. Weems, W. Lin, and H. Chen. Calculations of viscous nonlinear free-surface flows using an interactive zonal approach. In *Proceedings of the CFD Workshop of the Ship Research Institute of Japan*, 1994.
- [Weld, 1990] D. Weld. Approximation reformulations. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, 1990. MIT Press.
- [Yao and Gelsey, 1996] Ke-Thia Yao and Andrew Gelsey. Intelligent automated grid generation for numerical simulations. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 10(3), 1996.