51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference<BR> 18th
12 - 15 April 2010, Orlando, Florida

AIAA 2010-3088

Statistical Benchmarking of Surrogate-Based and Other Optimization Methods Constrained by Fixed Computational Budget

Patrick H. Reisenthel[*] and Daniel J. Lesieutre[†]
Nielsen Engineering & Research, Inc., Santa Clara, California 95054

The aerodynamic design of an asymmetric oversized payload fairing subject to stability constraints was used as an example of a derivative-free, expensive black box function to benchmark the relative performance of 16 different optimization methods, ranging from gradient-based to simulated annealing and genetic algorithms/evolution strategies, including four methods with surrogate-based accelerators. The focus of the present paper is on the practical attainability of getting an acceptable solution quickly. The various algorithms are compared using performance benchmarking in a statistical sense, yielding an "efficient frontier" with special emphasis on the case when designers are confronted with small computational budgets.

Nomenclature

`argmax`	=	argument of the maximum
CMO	=	cumulative metamodel optimization
C_m	=	pitching moment coefficient
DOE	=	design of experiment
E[.]	=	expected value
EI	=	expected improvement function
f	=	shape function
F	=	surrogate function
LHS	=	latin hypercube sampling
NL	=	nonlinear functional
OBJV	=	objective function value
p	=	probability
RBF	=	radial basis function
$\boldsymbol{x},\boldsymbol{X}$	=	design variables vector
y,Y	=	expensive, true function
σ	=	standard error
^	=	estimated
*	=	best real performer
<.>	=	average across realizations

I. Introduction / Motivation

Supporting dynamic, complex/nonlinear, and multidisciplinary optimization tasks requires novel methodologies capable of assimilating data from disparate (heterogeneous) sources in a potentially high-dimensional parameter space. Recent progress in cumulative metamodel technology, for example, fast multidimensional nonparametric response surface technology, has suggested improved multidisciplinary optimization methodologies capable of combining a priori mathematical models, numerical predictions, and noisy experimental data. Optimization methods which can perform robustly, consistently, and economically are of great interest to engineering and scientific design in MDO applications. The point of view adopted in this paper is the following: given finite computational resources and expensive function evaluations, it is critical that one selects the optimization method that is most likely to yield a satisfactory design with as few design evaluations as possible. To this end, Cumulative Global Metamodels (CGM) have been proposed as an aid for conceptual design of highly integrated flight vehicle and air space concepts.[1] These methods can be used with or without derivative information. However, frequently,

[*] Vice President & Chief Scientist, 2700 Augustine Drive, Suite 200, Santa Clara, CA 95054, Senior Member.

[†] Senior Research Engineer, 2700 Augustine Drive, Suite 200, Santa Clara, CA 95054, Senior Member.

single- and multidisciplinary analysis methods use expensive legacy codes that do not supply derivative information. Also, the results of high-fidelity analyses tend to be noisy, due to grid dependence and convergence issues. Therefore, the focus of the present paper is on derivative-free expensive "black box" optimization, specifically, the performance benchmarking of such methods in a statistical sense when the user is confronted with a fixed computational budget.

The present paper is organized as follows. Section II describes the surrogate-based optimization schemes used and, in particular, cumulative metamodel optimization (CMO). Section III considers the relative performance of various surrogate-based optimization schemes implementing CMO using a 1-D tutorial. An example of these methods is applied to an aerodynamic optimization problem, which is described in Section IV. The results of the optimization are given in Section V and summarized and discussed in Section VI.

II. Methods

It is well known that response surface-based surrogates can be used for automated searches and are naturally well-suited to the acceleration of optimization tasks[2-4] and rapid strategy evaluation. Given the specification of objectives and constraints, the basic optimization algorithm (**Fig. 1**) consists typically of the following steps: (1) initialization of metamodel parameters, (2) seeding of initial design space using design of experiments (DOE) methods, (3) metamodel/surrogate identification, (4) global search of current surrogate, (5) acquisition/evaluation of new data points ("infill" stage), (6) termination and/or model updates, and (7) determination of final solution quality.

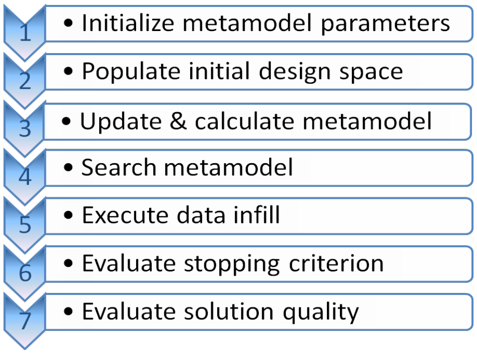
1
• Initialize metamodel parameters
2
• Populate initial design space
3
• Update & calculate metamodel
4
• Search metamodel
5
• Execute data infill
6
• Evaluate stopping criterion
7
• Evaluate solution quality

Figure 1. Outline of metamodel-based optimization algorithm.

When choosing a metamodel type, one must consider a number of issues, including assumption requirements, the expected design landscape complexity (if known), and the time spent training or retraining the metamodel. The optimization suitability of a given type of surrogate model is problem-dependent and is affected by factors such as the model's ability to fit complex local behavior, and the basic effort associated with the infill stage. References 4 and 5 provide excellent reviews of surrogate-based optimization which can be used to this effect.

Surrogate model types range in complexity from polynomial regression and moving least squares to Kriging and support vector regression. The approach used in this paper centers on the use of radial basis function (RBF) models, either fixed basis RBF or parametric RBF. The choice of these models is motivated by their qualities in terms of modeling ability, flexibility, generalization properties, and the significant advantages they offer in terms of performance.[6] The RBF class of surrogates covers a wide range of methods, from simple fixed basis RBF, to fully parametric RBF approaching the complexity of Kriging. Within this class, one can trade generality for performance, depending on which parameters are solved by optimization (parametric RBF) versus which parameters are fixed (simple RBF). In this paper, we will refer to two variants of these methods: (1) fixed parameters RBF, and (2) quasi-parametric RBF, in which the metamodel parameters are automatically updated (**Fig. 1**, Step 3) throughout the optimization.

A. Radial Basis Function Model

In N-dimensional design space, the sought-after surrogate is a function $F:\mathrm{R}^N \to \mathrm{R}$ which is constructed by satisfying data constraints at P available data points. If this response surface acts as an interpolant, then the function F must satisfy the constraints

$$F(\boldsymbol{X_i}) = Y_i, \;\; i = 1,\ldots,P \tag{1}$$

where each $\boldsymbol{X_i}$ is an N-dimensional vector containing the design variables and Y_i the corresponding dependent variable. In the case where F represents, instead, a regression model fit to the data, then the response surface is required to minimize in the least squares sense the distance $\| F(\boldsymbol{X_i}) - Y_i \|$, $i = 1,\ldots,M$, $M \geq P$.

In the radial basis function approach, the metamodel F is expanded into basis functions Φ_k which are radially symmetric about their control point, $\boldsymbol{\Gamma_k}$:

$$F(\boldsymbol{X}) = \sum_k c_k \, \Phi_k(\boldsymbol{X}, \boldsymbol{\Gamma_k}, b_k) \qquad \Phi_k = f(\|\boldsymbol{X} - \boldsymbol{\Gamma_k}\|, b_k) \tag{2}$$

where f is a scalar shape function (for example, a Gaussian), b_k is an adjustable scale or stiffness parameter, and $\|.\|$ designates the Euclidean norm. For example, if f is chosen to be a Gaussian:

$$\Phi_k(\boldsymbol{X}, \boldsymbol{\Gamma_b}, b_k) = \exp\left(- \frac{(\boldsymbol{X} - \boldsymbol{\Gamma_k})^{\mathrm{T}} (\boldsymbol{X} - \boldsymbol{\Gamma_k})}{b_k^2} \right) \tag{3}$$

With the additional assumptions that (a) the stiffness b_k is uniform, and (b) the control points are chosen among the available data points, the linear model regression design matrix equation [**A**][**c**] = [**Y**] is given by:

$$\begin{bmatrix} f(\|\boldsymbol{X}_1 - \boldsymbol{X}_1\|) & f(\|\boldsymbol{X}_1 - \boldsymbol{X}_2\|) & \dots & f(\|\boldsymbol{X}_1 - \boldsymbol{X}_{\mathbf{P}}\|) \\ f(\|\boldsymbol{X}_2 - \boldsymbol{X}_1\|) & f(\|\boldsymbol{X}_2 - \boldsymbol{X}_2\|) & \dots & f(\|\boldsymbol{X}_2 - \boldsymbol{X}_{\mathbf{P}}\|) \\ \vdots & \vdots & \vdots & \vdots \\ f(\|\boldsymbol{X}_{\mathrm{M}} - \boldsymbol{X}_1\|) & f(\|\boldsymbol{X}_{\mathrm{M}} - \boldsymbol{X}_2\|) & \dots & f(\|\boldsymbol{X}_{\mathrm{M}} - \boldsymbol{X}_{\mathbf{P}}\|) \end{bmatrix} . \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{\mathrm{P}} \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_{\mathrm{M}} \end{bmatrix} \tag{4}$$

In the case where uncertainty intervals for the Y_i are available, and provided these intervals correspond to random uncorrelated noise (variance σ_0^2), the variance of the surrogate prediction at point $\boldsymbol{X}$ is given by

$$\hat{\sigma}^2(\boldsymbol{X}) = \sigma_0^2 . \boldsymbol{\Phi}(\boldsymbol{X})^{\mathrm{T}} (\boldsymbol{A}^{\mathrm{T}} \boldsymbol{A})^{-1} \boldsymbol{\Phi}(\boldsymbol{X}) \tag{6}$$

where $\boldsymbol{\Phi}(\boldsymbol{X}) = [\Phi_1, \Phi_2, ..., \Phi_P]^{\mathrm{T}}$ (see Ref. 1). Although there are fundamental limitations to this kind of error modeling (since it pertains to repeatability errors), it will be used nonetheless to illustrate the use of the expected improvement function in the tutorial examples of Section III.

B. Objective Function

The examples discussed in this paper consider the unconstrained minimization of a computationally expensive function $Y(\boldsymbol{X})$ over a simply connected domain D:

$$\begin{array}{cc} \underset{\{x_1, x_2, \dots, x_N\}}{\text{minimize}} & Y(\boldsymbol{X}) \\ \text{subject to} & \boldsymbol{X} \in D \end{array} \tag{7}$$

A key aspect of the surrogate-based optimization process is the search (**Fig. 1**, Step 4) of the metamodel, since $F(\boldsymbol{X})$ evaluations cost little in comparison to evaluations of the true computational function $Y(\boldsymbol{X})$.

Additionally, in the case where an uncertainty model (e.g., (6)) is available, it is possible to make use of the expected improvement function (EI) traditionally used in Kriging.[2] This function has shown considerable promise[5] in terms of driving the optimization and being able to balance exploitation and exploration efficiently.

Assuming a normally distributed error at point $\boldsymbol{X}$ with standard error $\sigma(\boldsymbol{X})$, and defining $\boldsymbol{X}^*$ as the current best solution, the expectation of improving upon $F(\boldsymbol{X}^*)$ can be expressed analytically[2] as

$$\begin{aligned} \mathrm{EI}(\boldsymbol{X}) \; = \; E\left[\max\left(F(\boldsymbol{X}^*) - F(\boldsymbol{X}), 0\right)\right] \; &= \; \left(F(\boldsymbol{X}^*) - F(\boldsymbol{X})\right) . \boldsymbol{\psi}\left(F(\boldsymbol{X}^*) - F(\boldsymbol{X}), \sigma(\boldsymbol{X})\right) \\ &+ \quad \sigma(\boldsymbol{X}) . \boldsymbol{\phi}\left(F(\boldsymbol{X}^*) - F(\boldsymbol{X}), \sigma(\boldsymbol{X})\right) \end{aligned} \tag{8}$$

where $\boldsymbol{\phi}$ is the probability density function

$$\boldsymbol{\phi}(y, \sigma) \; = \; \frac{1}{\sqrt{(2\pi)}} \exp\left(- \frac{y^2}{2\sigma^2} \right) \tag{9}$$

and $\boldsymbol{\psi}$ is the normal cumulative distribution function,

$$\boldsymbol{\psi}(y, \sigma) \; = \; \frac{1}{2} \left[1 + \mathrm{erf}\left(\frac{y}{\sqrt{2}\sigma} \right) \right] \tag{10}$$

Thus, Problem (7) is replaced by the following alternative problem involving the surrogate F and its standard error σ:

$$\begin{array}{cc} \underset{\{x_1, x_2, \dots, x_N\}}{\text{maximize}} & OBJV(\boldsymbol{X}) \\ \text{subject to} & \boldsymbol{X} \in D \end{array} \tag{11}$$

where the objective function value $OBJV(\boldsymbol{X})$ is either {EI$(\boldsymbol{X})$} if an uncertainty model $\sigma(\boldsymbol{X})$ is available, or {$-F(\boldsymbol{X})$} otherwise.[‡]

[‡] Note that, in the case where $\sigma(\boldsymbol{X})$ is not available, using $OBJV(\boldsymbol{X}) = -F(\boldsymbol{X})$ is equivalent to using $OBJV(\boldsymbol{X}) = \mathrm{EI}(\boldsymbol{X})$ with $\sigma_0 = 0$. Tests using infinitesimally small input variance (e.g., $\sigma_0 = 10^{-10}$) have been shown to produce virtually identical results, although using $OBJV(\boldsymbol{X}) = -F(\boldsymbol{X})$ is more computationally efficient.

C. Update/Infill Criterion

Problem (7) is solved by making use of a sequentially constructed radial basis function surrogate $F(\boldsymbol{X})$. The surrogate is initialized using a sparse design of experiment (**Fig. 1**, Step 2). The metamodel search algorithm used in this study is a form of generalized pattern search[4] which uses a gradient-free, multistart, steepest ascent, hill climbing algorithm to maximize the expected improvement. At each iteration of the optimization, local optima resulting from the metamodel search are sorted based on objective function performance, and the top m virtual performers ($\boldsymbol{X}_i = \mathtt{argmax}\,[OBJV]$, $i = 1,\ldots,m$) are then used to spawn m real computational analyses $Y(\boldsymbol{X}_{i\,=\,1,\ldots,m})$.

These new computational analyses augment the RBF set of potential regressors $\Phi_{k=1..P}$ as well as the set of potential constraints $\{\boldsymbol{X}, Y(\boldsymbol{X})\}_{i\,=\,1,\ldots,M}$. In addition, the metamodel parameters b_k can be dynamically adjusted based on cross-validation error minimization or other criteria. The update stage (**Fig. 1**, Step 3) therefore results in solving a new linear system (4) for $F(\boldsymbol{X})$. The results presented in this paper use $m = 1$, corresponding to a standard, serial implementation of what Ref. 5 refers to as a "two-stage" infill.

III. A One-Dimensional Example

Prior to describing the application of the above methods to an aerodynamic design function involving CFD of unusual configurations, it is useful to understand how the methods perform on a simpler, one-dimensional problem.

A. Choice of 1-D Function

The data used in the problem described in Sections IV and V are obtained from Navier-Stokes CFD computations. Given the nature of the statistical benchmarking targeted in this research, rather than making CFD calls on demand, which would be prohibitively expensive, the present study is made possible by interrogating an existing CFD database. This CFD database thus represents the expensive function $Y(\boldsymbol{X})$.

Because database evaluations are the result of an interpolation of CFD data, the one-dimensional test function considered here is based on the same interpolation scheme as the one used in the CFD database (see Section IV). By doing so, the function is not only useful as a tutorial, but is also representative of the problem considered in Section V. The one-dimensional version of the expensive function, denoted $y(x)$, is considered over the domain $D = [0,1]$ and is the three-bucket function shown in **Fig. 2**.[§]

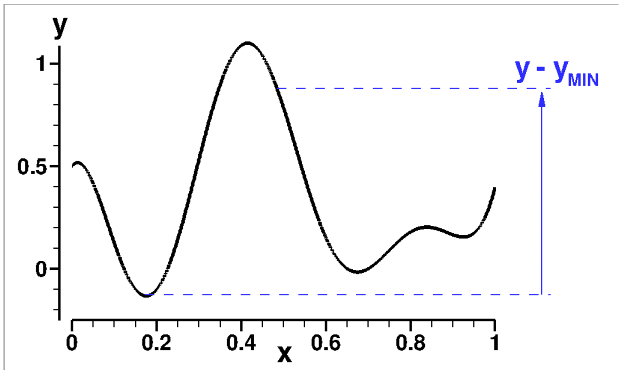

Figure 2. One-dimensional test function.

B. Cumulative Metamodel Optimization Variants

The methods described in Section II are referred to here as Cumulative Metamodel Optimization (CMO). To better understand how the variants of the method may affect their probabilistic outcome, this section considers four different variations. The first three are dependent on the existence of an uncertainty model. The fourth represents the particular case in which no uncertainty is assumed and is analogous to the approach used in Section V.

Figure 3 compares the results of six optimizations representing three optimization strategies, two uncertainty models, and three different sets of initial conditions. Each row in **Fig. 3** corresponds to a given optimization run. For each run, four plots are shown. The first three plots provide snapshots of the surrogate-based optimization as it progresses. The first snapshot corresponds to the state of the optimization at the beginning of the first iteration, i.e., after the DOE step and after the calculation of the initial metamodel (end of Step 3, Iteration 1 in **Fig. 1**). Each design of experiment consists of only two points, which are selected based on a latin hypercube sampling strategy. At the end of the 18th iteration, 20 function evaluations $y(\boldsymbol{x})$ have been performed (two as part of the initial design, and 18 infills). The corresponding "late stage" portrait is shown as the third snapshot. The second snapshot (labeled "intermediate stage") corresponds to the optimization after anywhere from 4 to 9 iterations. Each snapshot indicates the following: (a) the true function $y(\boldsymbol{x})$, labeled "database" (black line), (b) prior function evaluations (whether infills or DOE, shown as □ symbols, (c) the current surrogate $F(x)$ (solid blue line), (d) the uncertainty on the current surrogate, depicted as $F(\boldsymbol{x}) \pm \sigma(\boldsymbol{x})$ (dashed blue lines), and (e) the value of the expected improvement function (red line, log scale). The fourth plot in each row is a summary performance profile for the given optimization. Each performance profile indicates the evolution of the best current minimum y^* as a function of the total number of function evaluations, n_i.

[§] Although not strictly identical, a good approximation of this function over the interval $\boldsymbol{x} \in [0,1]$ is given by the polynomial $y(x) = 0.520558 - 0.07002374\boldsymbol{x} - 24.1693\boldsymbol{x}^2 - 638.5461\boldsymbol{x}^3 + 7418.51\boldsymbol{x}^4 - 29340.51\boldsymbol{x}^5 + 57419.98\boldsymbol{x}^6 - 60358.23\boldsymbol{x}^7 + 32720.3\boldsymbol{x}^8 - 7197.405\boldsymbol{x}^9$.

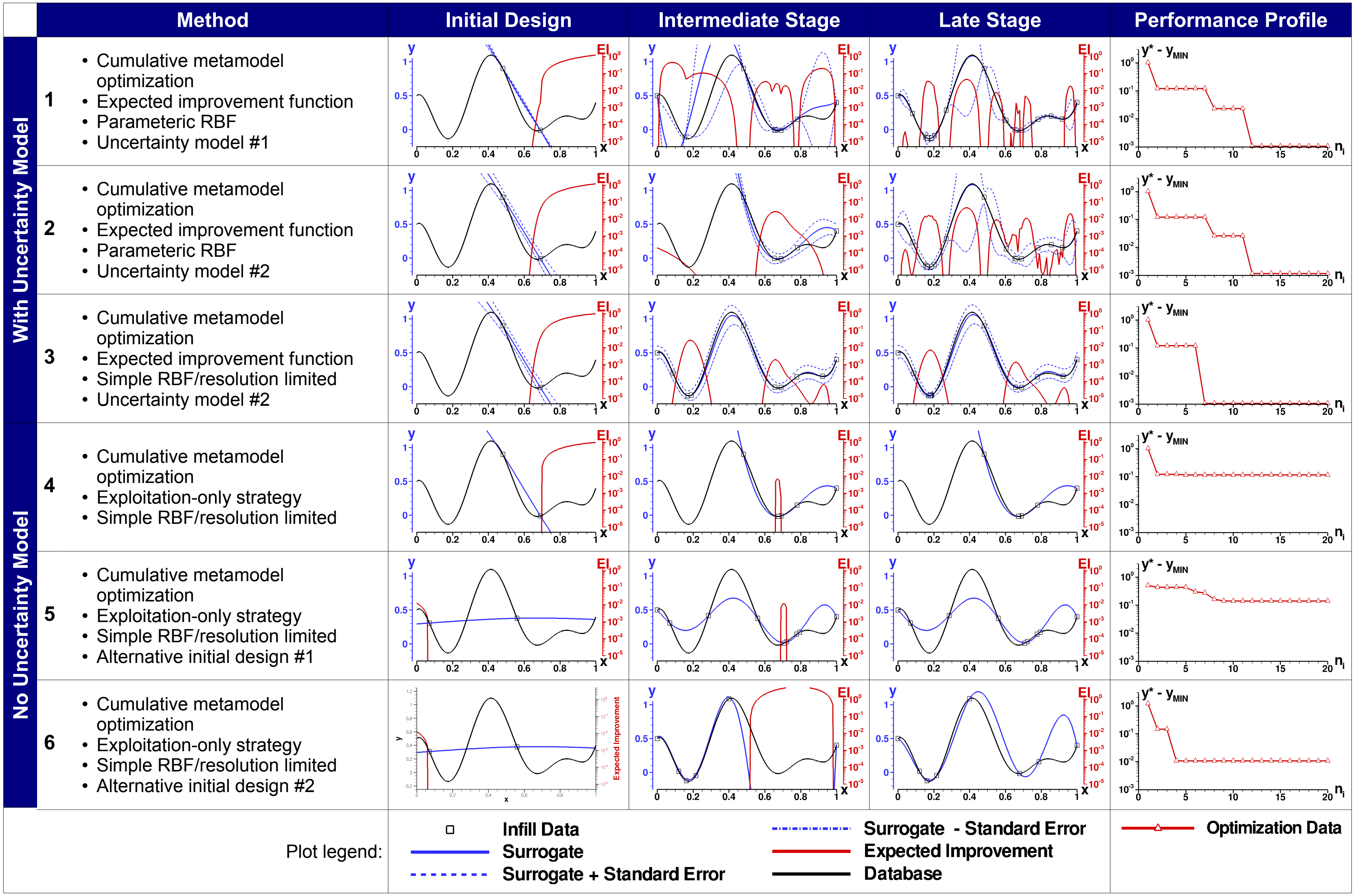

Figure 3. Examples of Surrogate Optimization Strategies.

All six optimizations in **Fig. 3** are expected improvement function driven, i.e., *OBJV(**x**)*= EI*(**x**)*. For the first three optimizations, a value of $\sigma_0 = 0.1$ is assumed, while a value of $\sigma_0 = 10^{-10}$ is used in the last three optimizations to emulate, using the same method, the case where there is no uncertainty model available. Cases 1 through 4 (**Fig. 3**) use the same DOE as initial conditions. Cases 5 and 6 are repeats of Case 4 using different initial conditions. All cases employ an RBF surrogate with regressors of the form

$$\Phi_k(\boldsymbol{X},\boldsymbol{\Gamma_b},b_k) = \frac{1}{\sqrt{(\boldsymbol{X}-\boldsymbol{\Gamma_k})^{\mathrm{T}}(\boldsymbol{X}-\boldsymbol{\Gamma_k}) - b_k^2}} \qquad (12)$$

For Cases 3-6, the b_k parameters are fixed from iteration to iteration. By contrast, Cases 1 and 2 benefit from an adaptation algorithm which modifies the b_k parameters from iteration to iteration (**Fig. 1**, step 3).

Cases 1 and 2 in **Fig. 3** differ by the uncertainty model they use. While Cases 2 through 6 use the uncertainty propagation model ("uncertainty model #1") described in Eq. (6), Case 1 simulates the case of an uncertainty model associated with computational experiments having zero uncertainty at the computed points but nonzero uncertainty inbetween ("uncertainty model #2"). Such uncertainty intervals will result, for example, from variations between fully populated regression models which exactly interpolate all the data.[¶] The result of each optimization iteration is the detection of $\boldsymbol{x}$ = `argmax` [EI] which is used as the next infill point before updating the metamodel using {***x**,y(**x**)*}. It can be seen from the last stage snapshot of Cases 1 and 2 that all minima of the function are found, and that the surrogate accurately approximates the underlying function. Case 3 also detects all minima, but with a lesser global accuracy. This is due to the lack of adaptivity of this variant, which has two effects: (1) less exploration, and (2) limited fitting ability.

Case 4 is identical to Case 3 except that Case 4 has no uncertainty model. The late stage optimization snapshot shows a poor approximation of the underlying function for this set of initial conditions, and, accordingly, a solution that is trapped in a local minimum. The outcomes of two alternative sets of initial conditions are shown as Cases 5 and 6. Case 5 optimization results in the solution being trapped in the same local optimum around $\boldsymbol{x} = 0.675$ and a poor quality surrogate. Case 6 optimization, on the other hand, ends up finding the global minimum.

C. Ensemble Processing

The sample results shown in **Fig. 3** (Cases 4 through 6) suggest 1:3 odds in favor of finding the global minimum when using a simple a simple pure-exploitation strategy. The results of Cases 1 through 3 also suggest that a strategy which appropriately balances exploitation and exploration has the greatest probability of finding the global optimum. This is, of course, a well-established fact (see, e.g., Ref. 5), and the expected improvement function is designed to do so. However, for *OBJV(**x**)*= EI*(**x**)* to be an effective driver, the uncertainty model/levels must be correct.

Too often, optimization methods are compared on the basis of a limited number of realizations. In the case of surrogate-based optimization, there is a strong coupling between successive infills and the evolution of the surrogate. This is also true of the dependence on the initial DOE, especially in data starved situations. While it is clear that a richer DOE coverage can mitigate the dependence on initial conditions, the choice of a two-point DOE in one dimension is representative of data-starved situations, which occur with either large numbers of design variables, or limited resources, or both.

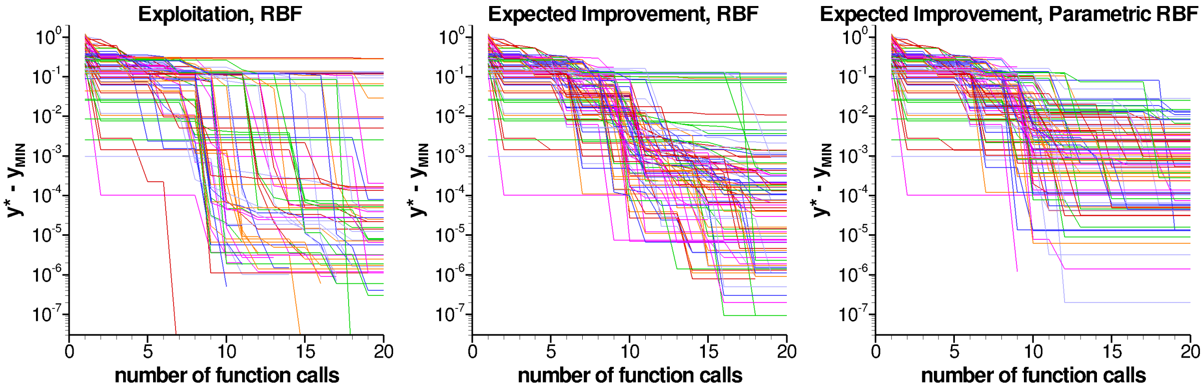

Figure 4. Comparison of optimization data profiles resulting from 100 realizations for three different strategies. Left: pure exploitation with simple RBF. Center: expected improvement function with simple RBF. Right: expected improvement function with parametric RBF.

To further clarify the relative performance of the methods shown in **Fig. 3** , the results of Cases 2, 3, and 4 are each repeated 100 times using different initial design of experiments. Each design of experiment corresponds to a two-point latin hypercube sampling, and the same 100 initial conditions are used in each case. The corresponding aggregate of

[¶] approximated here using the artefact $\hat{\sigma}^2(\boldsymbol{X}) = \sigma_0^2.\left|\boldsymbol{\Phi}(\boldsymbol{X})^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{\Phi}(\boldsymbol{X}) - 1\right|$ as an alternative to (6).

individual optimization data profiles is shown in **Fig. 4**. Although large DOE-to-DOE variations occur, several observations can be made from the raw data. Knowing that the difference between the primary and secondary minima is $\Delta \approx 0.11$, it can be seen from the center plot, for example, that approximately 10% of realizations do not fall in the basin of attraction of the global optimum when using fixed RBF. By contrast, all 100 realizations succeed in finding the global optimum region when using adaptive refinement (**Fig. 4**, right hand side plot).

Plots such as those shown in **Fig. 4** can be postprocessed as cumulative distribution functions corresponding to either vertical or horizontal "cuts" through the data. Examples are shown in **Figs. 5** and **6**. **Figure 5** compares the three optimizations of **Fig. 4** in terms of their cumulative probability distribution function after 18 iterations. The cumulative distribution functions indicate the probability $p(y^* \le y)$ that the best real function evaluation y^* not exceed a given threshold value y. The value of the threshold is shown on the abscissa, referenced to the value of the global minimum, y_{MIN}. As expected, there is a less than 90% probability $p(y^* \le \Delta)$ of escaping the secondary minimum for the "expected improvement, RBF" method. **Figure 5** reveals that this probability drops to less than 65% if one uses an exploitation-only strategy. It is also interesting to note that, for low values of the threshold $(y - y_{MIN} \le 10^{-2})$, the fixed RBF methods achieve a higher probability of success.

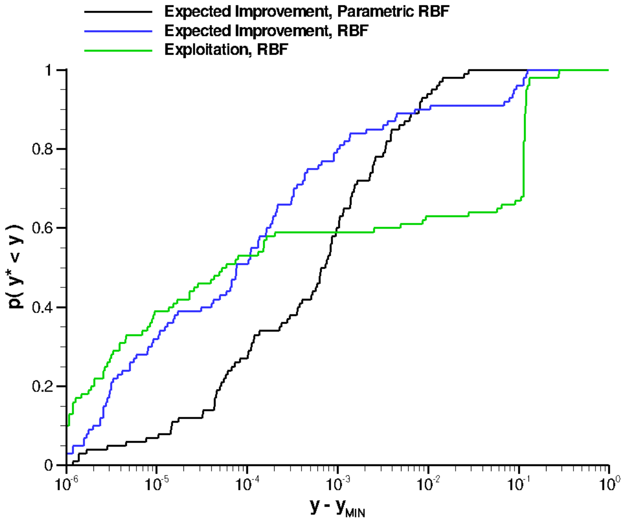

Figure 5. Cumulative distribution functions after $n_i = 20$ function evaluations.

This last observation is confirmed in **Fig. 6**, which depicts the probability that $y^* - y_{MIN}$ not exceed 10^{-1} (solid lines) and 10^{-5} (dashed lines), respectively, as a function of the number of function calls. The results indicate that, while the "expected improvement, parametric RBF" method is thorough, in terms of finding the correct basin of attraction every time, there is a price to pay, in terms of how deep a minimum is found for the given number of iterations. By contrast, the methods which spend less time exploring and, accordingly, more time exploiting ("expected improvement, RBF" and "exploitation, RBF"), are more likely to find a deep minimum, as exhibited by the $p(y^*-y_{MIN} \le 10^{-5})$ results.

In other words, an adaptively refined, expected improvement-driven method is a better approach for "degrossing" the problem, i.e., focussing the method on the right area of the design space. Although further research is required, the present results suggest that, once in the correct basin of attraction, a simple infill strategy may be most efficient.

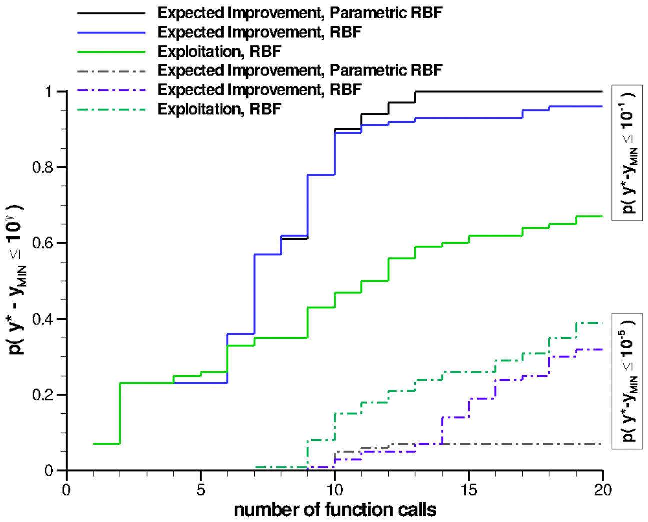

Figure 6. Probability of reaching the global optimum as a function of the number of function evaluations for two different objective function thresholds.

D. Application to Aerodynamic Design

With a few exceptions (see **Table 1**), the "exploitation, RBF" method and example described above are the one-dimensional analog of the problem considered in Section V.

Table 1. Summary of differences between the optimization problems considered in Sections III and V.

Test Cases	# variables (N)	Design Functions	RBF Surrogates	Objective Function
Fig. 3, Cases 4-6	1	$y(\mathbf{x})$	$F[y(\mathbf{x})]$	$OBJV = F$
Section V	5	$Cm_1(\mathbf{X})$, $Cm_2(\mathbf{X})$	$F_1[Cm_1(\mathbf{X})]$, $F_2[Cm_2(\mathbf{X})]$	$OBJV = NL(F_1,F2)$

In Section V, ensemble processing similar to **Figs. 5** and **6** is used to reduce the variability of optimization results when comparing between different methods.

IV. CFD Database Background

In Ref. 7, the problem of designing a payload fairing large enough to encompass a spacecraft with an optical mirror up to twice the diameter of an expendable launch vehicle (EELV) was considered. Optimization and computational fluid dynamics (CFD) methods were used to design a nonaxisymmetric exterior shape for the fairing that meets specific aerodynamic design goals. These aerodynamic goals include a balance of low lateral force on the fairing and smooth variations in that force with respect to angle of attack across a range of Mach numbers at transonic conditions. In this application, the multiple design objectives are met by means of subsystem metamodels which are combined into a single performance index.

The fairing surface is defined principally in terms of analytical functions. The cost of performing design optimization increases significantly with the number of independent design variables; therefore, it was important to minimize the number of control variables needed to parameterize the surface shape. This is accomplished with an analytical surface shape definition, rather than a surface defined by discrete points. Also, the approach seeks to impose the designer's understanding of aerodynamics on the optimization process, and to focus the design on shapes that trigger flow separation gradually, to alleviate excessive aerodynamic forces on the fairing.

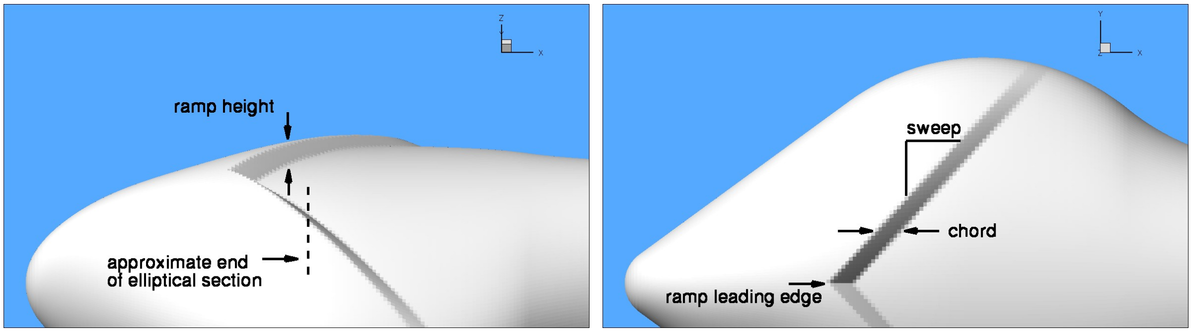

Figure 7. Side and top views of payload fairing geometry.

Three generations of the aerodynamic design were considered. Only the first two are discussed here. In the first-generation family of designs, the payload fairing geometry was initially parameterized using nine free design variables, corresponding to a two-dimensional (unswept) version of the ramp shown in **Fig. 7**. The nine control variables were the longitudinal minor axis and focus for the nose, the depths of the minor and major axes, three variables controlling the height, location, and slope of the separation trigger, and two geometry transition parameters. A main outcome of optimizing this first-generation family of designs was to confirm that flow separation was needed in order to achieve the aerodynamic goals. The geometry of the second-generation family of designs is illustrated in **Fig. 7**, which shows side and top views of the three-dimensional payload fairing geometry, along with some of the geometric design parameters. This fairing geometry was controlled by five independent variables, which were (1) the length of the elliptical section of the nose (in side view), (2) the location of the apex of the swept rearward-facing ramp (denoted "ramp leading edge" in the top view), (3) the ramp sweep angle, (4) the length of the backward-facing ramp ("chord" in top view), and (5) the height of the ramp.

Because the flow around the payload fairing involves flow separation, a Navier-Stokes CFD solver (Overflow[8]) was used to predict the steady-state forces and moments on the launch vehicle. The turbulence model used was the Spalart-Allmaras model.[9] The active independent variables were used to automatically remesh the vehicle's surface and recompute a new volume grid using Hypgen.[10] Grid convergence studies were conducted to determine the optimal grid resolution needed. Using this grid resolution, individual CFD calculations took between two and six hours, depending on the rate of convergence, which itself depended on the flow physics of individual cases.

Multiple response surfaces (metamodels), each corresponding to one dependent variable, were combined into a single objective function on which the search was performed (see **Table 1**). The individual dependent variables were the overall pitching moments, C_m, at multiple aerodynamic conditions. These multiple aerodynamic conditions were chosen to correspond to combinations of Mach number, angle of attack, and could potentially include roll angle.

At Mach 1.0, it was found that the load C_m could be minimized relatively easily, to the point of zero load or even load reversal. Such a condition, however, was the result of massive flow separation over the surface of the fairing and, consequently, tended to exhibit poor aerodynamic characteristics (i.e., a nonsmooth response with respect to angle of attack). Another problem was that minimizing the load at Mach 1.0 (the nominal specification) frequently resulted in configurations that exhibited excessive loads at slightly supersonic Mach numbers, due to the reduced extent of flow separation in supersonic flow. Both difficulties were addressed by carrying out the optimization based on a measure of global aerodynamic performance, rather than on a single load. This measure of global aerodynamic performance was specified as an objective function designed to penalize the undesired behaviors. This objective function (*OBJV*) combined multiple nonlinear "bucket" functions, one per explicitly optimized aerodynamic condition.

One of the outcomes of this optimization process was the generation of 320 separate fairing configurations. These were used in the present study as the support data of a "black box" function, referred hereafter as "the CFD database." This CFD database is an approximate representation of the launch vehicle aerodynamics, which agrees exactly with the CFD at 320 points of the design space. While the fidelity of this model is uncertain away from the support data, we expect it to hold reasonably well near the real CFD points. It is important not to confuse the reference, fixed "CFD database" with the dynamic metamodel being updated during the course of the optimization (**Fig. 1**, Step 3).

V. Benchmarking of Optimization Methods

The present section begins with a summary of the goals and methods of surrogate-based optimization, followed by a description of the specific objectives of this study and the emphasis on positive outcome at finite resources rather than asymptotic convergence. The optimization methods used in this study are then tabulated and the method used for their comparison is described.

A. Introduction

In surrogate-based optimization approaches, the optimization is carried out on analytical surrogates in lieu of the computational analyses themselves. While it is not difficult to understand the performance advantages of metamodel optimization when the surrogates approximate the actual problem with reasonable accuracy, an important question is whether this has any practical value if the knowledge of the design space is sparse or imprecise. Indeed, an argument can be made that the knowledge of the design space is necessarily imprecise at the early stages of design, and that this lack of certainty is inherent to high-dimensional spaces, since they cannot be sufficiently populated *a priori.* For this reason, most metamodel optimization approaches include a critically important last step, which is to *verify* the optimum (i.e., characterize the neighborhood around the presumed optimum using real analysis).

Efficient surrogate-based optimization seeks to make use of the limited (and often imprecise) information contained in early and intermediate versions of the surrogate model. The point of view adopted here is one where not only are the computational analyses expensive "black box" function calls, but the computational and time resources are limited by budgetary constraints. It is desirable to make use of as few real computational analyses as possible in order to solve the optimum (or reach an acceptable optimum). To do this, a metamodel surrogate must be formulated at the earliest stages of the design, typically from an initial (low-count) design of experiment (DOE).[#] This metamodel is searched, and results of the search are then used to spawn one or more real computational analyses which provide the infill data used to update the surrogate.

CMO can be described as an implementation of surrogate-based optimization based on a pruning-free fully-populated regression model. As such, the underlying metamodel is constantly updated using new real information, making it, in turn, more accurate. The updated metamodel becomes the object of a new metamodel search, which in turn spawns new support data, and so on (see **Fig. 1**), until an acceptable optimum is reached (and, ultimately, verified). The CMO approach is similar to other adaptive response surface approaches. Every adaptive response model is based on two elements: (1) flexible approximations to the underlying physics-based model, and (2) smart/efficient sampling methods. The present sampling strategy is goal-driven, rather than primarily concerned with model accuracy (Refs. 11-14) *per se*. Local (or even global) accuracy may result from the addition of support data in regions of interest, but it is not a necessary condition *a priori* (compare, for example, the late stage surrogates in **Fig. 3**). The only thing that matters is that it improves the design, i.e., drives the design (real, not virtual) towards the goals expressed in the objective function formulated by the user.

B. Objective

While the idea that adaptive response surfaces can be used to accelerate the design process is not new,[4,15] it is important to quantify the relative performance of optimization methods on problems other than "toy problems" such as the one considered in Section III. The design of an oversized asymmetric fairing described in Section IV provides a viable target for benchmarking because it is a difficult problem relevant to the aerospace community and because it is computationally tractable, since the CFD database is itself a surrogate for Navier-Stokes calculations.

An interesting aspect of the CMO method using radial basis functions is the occasionally erratic nature of the optimization trajectories/data profiles (see **Fig 3**). In our experience, the approach is likely to cause initial *divergence*, resulting in an efficiency penalty which must be overcome. In other words, it cannot be taken for

[#] In the RBF method, the optimization can start with as little as *one* function evaluation.

granted that a *net* acceleration should always be achieved. The fact that the dynamically updated metamodel will end up approximating the physics may be true, but only asymptotically, and provided there is sufficient coverage, which depends on the infill strategy. The main focus of this paper is not on asymptotics (*that* number of CFD calculations could not possibly be afforded!), but, instead, concerns the practical attainability of getting an acceptable solution quickly.

C. Approach

To perform this benchmarking study, 16 optimization methods were applied to the aerodynamic design of a nonaxisymmetric payload fairing, as described in Section IV. This test problem was chosen because of its suitable complexity and relevance to the aerospace community. The salient characteristics of this problem are as follows:

- five independent variables controlling the shape of the fairing (see **Fig. 7**),
- two dependent variables (corresponding to the overall pitching moment at different Mach number and angle of attack combinations), treated as individual subsystems,
- nonaxisymmetric shock structure and three-dimensional flow separation and reattachment, resulting in highly nonlinear, highly sensitive responses for each subsystem,
- stability constraint enforced via nonlinear objective function (used to combine the two subsystem response surfaces into one figure of merit, see **Table 1**),
- global optimization problem (large number of local minima),
- encapsulated CFD model based on a database of 640 Navier-Stokes calculations.[**]

It is well-known that the suitability of one optimization method over another depends on the problem at hand. Therefore, a wide class of optimization methods (gradient-based, genetic algorithms, simulated annealing, evolution strategies) were examined, with the goal of quantifying their relative efficiency on this problem. This was accomplished by exercising the optimization methods provided within the modeFRONTIER™ software.[16] A summary of the optimization methods used in this study is provided in **Table 2**.

Table 2. Summary of optimization methods.

Name	Description
SIMPLEX	Nelder & Mead Simplex algorithm for nonlinear optimization problems (Ref. 17).
B-BFGS	Bounded Broyden-Fanno-Fletcher-Goldfarb-Shanno gradient algorithm (Ref. 18).
SA	Simulated Annealing (Ref. 19).
MOGA-II	Multi Objective Genetic Algorithm (Ref. 16).
ARMOGA	Adaptive Range Multi-Objective Genetic Algorithm (Ref. 20).
MOSA (discrete)	Multi Objective Simulated Annealing, discrete design space (Ref. 21).
MOSA (continuous)	Multi Objective Simulated Annealing, continuous design space (Ref. 21).
MACK	Multivariate Adaptive Crossvalidating Kriging (Ref. 16).
NSGA-II	Non-dominated Sorting Genetic Algorithm II (Ref. 22).
FMOGA-II	Fast Multi Objective Genetic Algorithm using internal response surface acceleration (Ref. 16).
FSIMPLEX	Fast Simplex algorithm using internal response surface acceleration (Ref. 16).
NLPQLP	Sequential Quadratic Programming solver (Ref. 23).
MMES	Multimembered Multiobjective Evolution Strategy (Ref. 16).
DES	Derandomized Evolution Strategy (Refs. 24-26).
1P1-ES	(1+1)-Evolution Strategy (Refs. 24-25).
CMO-1	Cumulative Metamodel Optimization (Ref. 27).

Each scheduler in modeFRONTIER™ consists of a (DOE, optimization method) pair. Each optimization method, whether gradient-, or population-, or evolutionary-based, uses an initial design-of-experiment table. This initial DOE table provides either the initialization for the search path (e.g., SIMPLEX, B-BFGS, $*$SA, NLPQLP, CMO-1), or an initial population from which to evolve (e.g., $*$GA and $*$ES methods). The same space-filling Latin Hypercube Sampling (LHS) design method was used as the DOE method for all of the optimization methods investigated. The number of LHS samples was kept fixed at 15 points. For a given trial, the *same* 15 points were used by all methods. Special care was taken to export the DOE table to make sure that the CMO-1 method

[**] 320 configurations calculated at two different flow conditions

(exercised outside modeFRONTIER™) used exactly the same initial conditions as all other methods. Regardless of optimization method, the querying of the CFD database was executed using a simple shell script.

The benchmarking exercise was carried out on a single-objective optimization problem, namely the minimization of the objective function *OBJV* referred to in Section IV. The efficiency of the various methods is compared in the following manner. During the optimization process, various regions of the design space are visited and the resulting design objective function values are stored in a design database. Each design ID corresponds to one geometric configuration, which, in this study, corresponds to two CFD calculations (one per Mach number and angle of attack combination). Since we are concerned with optimization using expensive computational analyses, the convention throughout this report is to equate the cost (time) of optimization with the number of design IDs, denoted *Niter* in the following figures. Note that the function count is conservative, in the sense that, even in the case of optimization methods that use gradients and for which the gradients had to be approximated by finite differences, only the nominal design is counted. And the same goes, of course, for any optimization that is assisted by an internal response surface accelerator, in which case only the real design IDs are counted; response surface or virtual ones are not.

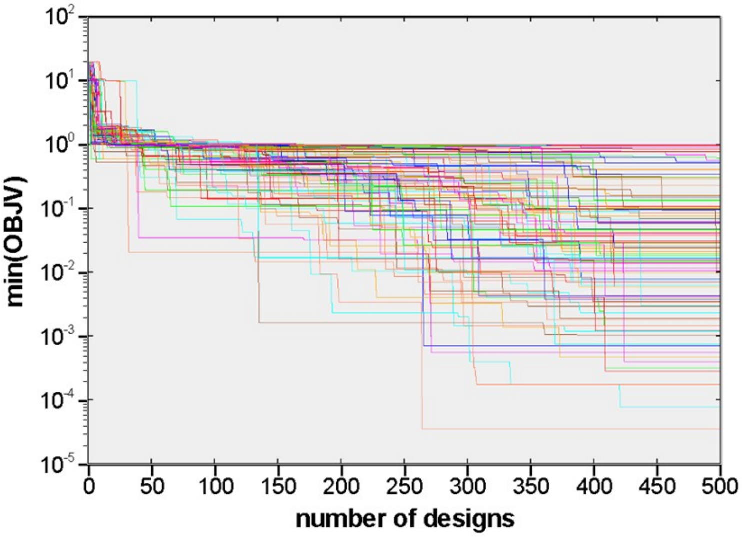

Figure 8. Variability of *min(OBJV)* data profiles plotted as a function of the number of real function evaluations.

Thus, the ultimate metric in this study is the number of designs (real, not virtual) that need to be evaluated in order to achieve a desired performance objective. Specifically, we are interested in comparing the various methods in terms of how many function calls are required to meet the design objectives. To do so, we keep track of the "current best" solution (i.e., minimum *OBJV*) at any given iteration of the design. Naturally, this produces a monotonically decreasing function exhibiting discrete steps each time a lower minimum is found. As demonstrated in Section III, the location of these steps and how deep a minimum is reached depends not only on the optimization method used, but also on the initial DOE. The latter dependence is a consequence of the multimodality of the problem being solved, i.e., the existence of a *multiplicity* of local minima. Due to this multimodality, a hypothetical "Method A" could be more efficient than another hypothetical "Method B" for a given initial DOE, but their relative ranking could be reversed when initiating the calculations from a different initial DOE. For this reason, optimizations need to be carried out using repeated independent random trials (realizations), where each trial corresponds to one 15-point initial DOE. To maximize statistical significance, 100 random trials were conducted, for a total of 1,600 optimizations. Scheduler parameters were chosen so that each method within modeFRONTIER™ would run for approximately 500 real designs (i.e., 500 CFD database queries). An example of the raw optimization results data for the case of the MOGA-II method is given in **Fig. 8** in the form of data profiles[28] for 100 realizations of the *min(OBJV)* function versus the number of function calls.

For each method, the stochastic results are processed in two ways. The first consists of averaging the results of individual realizations of *min(OBJV)*, yielding a smoother representation of the average scenario. While informative, a problem of this representation is that, since the underlying probability distributions are not symmetrical about the mean, the average can be misleading. The second type of processing computes the probabilities of exceeding preset *OBJV* thresholds based on the accumulated distribution histograms.

By construction, the *OBJV* values in the present application range from 20 (worst) to 0 (best). It follows that values of *OBJV* below a threshold of 0.5 can be thought of as being associated with the top 2.5% performers in the design space. Similarly, those below 0.2 represent the top 1% performers, and so on.

D. Results

Figure 9 summarizes the performance of all 16 optimization methods. The results are presented in the form of (a) the average across realizations of *min(OBJV)* (denoted <*OBJV*>, top graph), and (b) the probability, denoted $p(OBJV \leq 0.5)$, that *min(OBJV)* not exceed a 0.5 threshold value (bottom graph). Both are plotted as a function of the number of designs/real function evaluations, *Niter*.

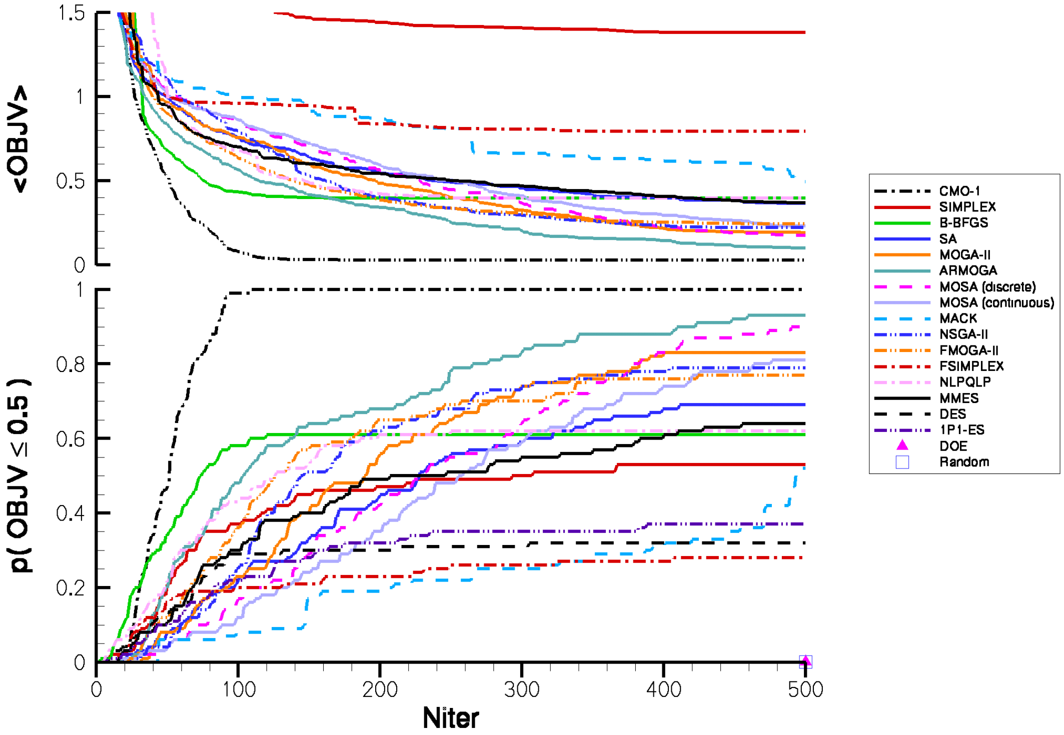

Figure 9. Comparison of mean of *min(OBJV)* and probability *p(min(OBJV)* ≤ 0.5) as a function of the number of designs.

Looking at the averages (**Fig. 9**, top graph), it is clear that some of the evolution strategies, along with methods such as SIMPLEX, FSIMPLEX, and MACK do rather poorly on this problem, at least within the first 500 design iterations. The best performer amongst the standard methods initially is B-BFGS, which lowers <*min(OBJV)*> faster than any of the other methods early on, but stalls after approximately 120 iterations. The reason for this is that further iterations only contribute to finding deeper (more precise) local minima, but do not improve the odds of finding *better* optima. After approximately 150 iterations, ARMOGA dominates the other methods. Specifically, <*min(OBJV)*> in ARMOGA reaches 0.1 in approximately 490 iterations. By 500 iterations, the runner-ups (MOSA (discrete), MOGA-II, NSGA-II, MOSA (continuous), and FMOGA-II) only reach between 0.17 and 0.25, while <*min(OBJV)*> for all other methods remains larger than 0.36.

In contrast to these methods, CMO-1 quickly drives <*min(OBJV)*> to low values, reaching 0.08 before 100 iterations (see **Table 3**). It must be stressed that this comparison applies identical initial conditions for all methods, and that the initial conditions are those randomly generated within modeFRONTIER$^{\text{TM}}$'s DOE module.

Table 3. Ranking of optimizers based on *min(OBJV)* performance at *Niter* = 500.

Name	<OBJV>			
	Niter = 50	*Niter* = 100	*Niter* = 200	*Niter* = 500
CMO-1	0.519	0.078	0.026	0.026
ARMOGA	0.835	0.566	0.339	0.098
MOSA (discrete)	1.023	0.851	0.564	0.172
MOGA-II	0.964	0.756	0.484	0.190
NSGA-II	1.105	0.797	0.400	0.221
MOSA (continuous)	1.010	0.861	0.594	0.229
FMOGA-II	0.899	0.637	0.383	0.243
SA	0.992	0.756	0.544	0.363
MMES	0.938	0.700	0.541	0.366
NLPQLP	1.014	0.678	0.419	0.394
B-BFGS	0.679	0.433	0.395	0.395

Name	*<OBJV>*			
	Niter = 50	*Niter* = 100	*Niter* = 200	*Niter* = 500
MACK	1.106	1.000	0.867	0.494
FSIMPLEX	1.004	0.958	0.838	0.791
SIMPLEX	1.715	1.535	1.440	1.379
1P1-ES	5.205	4.779	4.651	4.569
DES	5.311	5.167	5.120	5.100

Another way to examine the results is to consider the probability of reaching our objective, expressed by *p(min(OBJV)* ≤ *threshold)*. The lower plot in **Fig. 9** depicts this probability for an operational threshold value of 0.5. The CMO-1 method outperforms all other methods, reaching a 99% probability of success within 93 function calls on average (and 100% success in less than 110 function calls). By comparison, at 100 iterations, the overall runner-up (B-BFGS) only has a 58% probability of success, with the next largest probability of success obtained with ARMOGA, which gradually increases to 93% by *Niter* = 500. All other methods perform worse on this problem.

The graph shown in the lower portion of **Fig. 9** is a measure of systematic efficiency. It does not necessarily say that a local objective function minimum of less than 0.5 cannot be reached, for example, by a gradient or simplex-type method. It simply says that these methods are less *likely* to do so. Specifically, it quantifies how much less likely they are to do so than CMO. Indeed, even a random sampling strategy is not immune to finding a low minimum; it is simply less likely to do so than a rational optimization method. For reference, when sampling the design space randomly 50,000 times (the equivalent of 100 trials with 500 random numbers each), only 72 designs are found such that *OBJV* ≤ 0.5. Similarly, a sampling of the design space using a 50,000-point Latin Hypercube DOE yields 73 acceptable designs, representing a less than 0.15% probability of success. Both of these are indicated by the symbols in the lower right-hand corner of **Fig. 9**. If one lowers the *OBJV* threshold from 0.5 to 0.1 (from top 2.5% performers to top 0.5% performers), the random and DOE odds of success decrease further to 0.03%.

It is noteworthy that this simple implementation of CMO represents at least a five-fold increase in efficiency factor over the best-performing standard method (ARMOGA), in the sense that a 90% probability of reaching the objective is achieved in 1/5th the number of evaluations. Most importantly, as the threshold decreases (corresponding to a more demanding objective), this efficiency factor grows further, as shown in **Figs. 10** and **11** for threshold values of 0.2 and 0.1, respectively. This trend towards further increased efficiency is consistent with the results of Han and Chatterjee.[15]

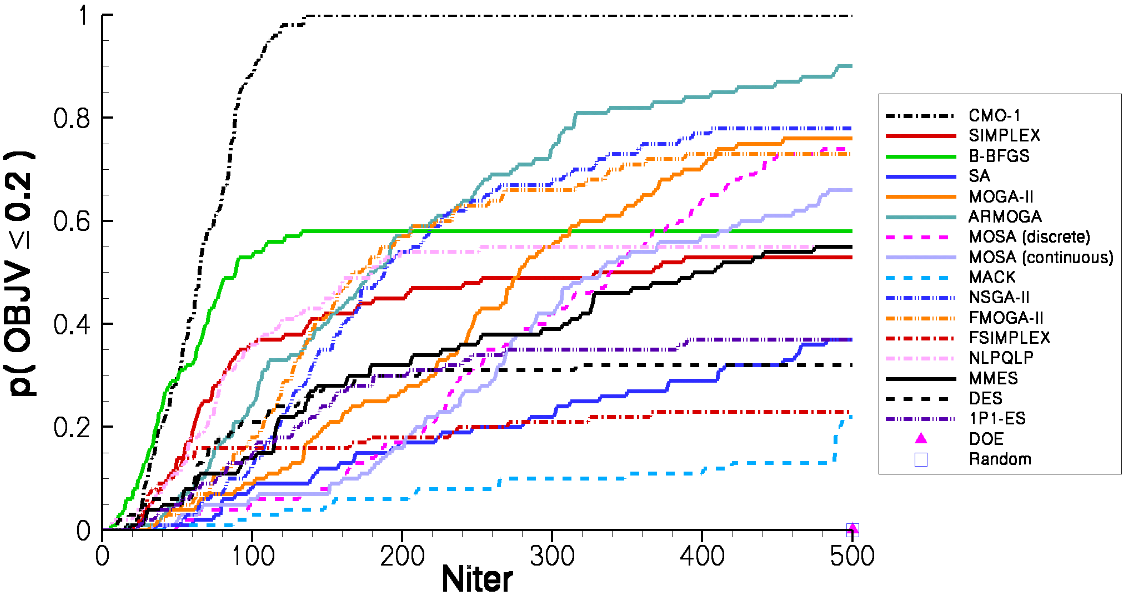

Figure 10. Comparison of probability *p(min(OBJV)* ≤ 0.2) as a function of the number of designs.

A summary of the relative performance of each optimization method, ranked according to *p(min(OBJV)*≤0.2) at *Niter* = 500 is given in **Table 4**. Based on the results of **Figs. 9-11**, it is clear that no single method dominates unconditionally. Which method performs best statistically for a given number of iterations depends on the expectation or definition of "success." For example, in the unlikely event that "success" is defined as a greater than 25% probability of reaching *min(OBJV)* ≤ 0.1, then, according to **Fig. 11**, B-BFGS dominates (reaching this likelihood in just 45 designs). If, on the other hand, "success" is defined as a 99% or more probability of reaching the same threshold, *min(OBJV)* ≤ 0.1, then CMO-1 is the only successful method,[††] reaching this level in 135 designs.

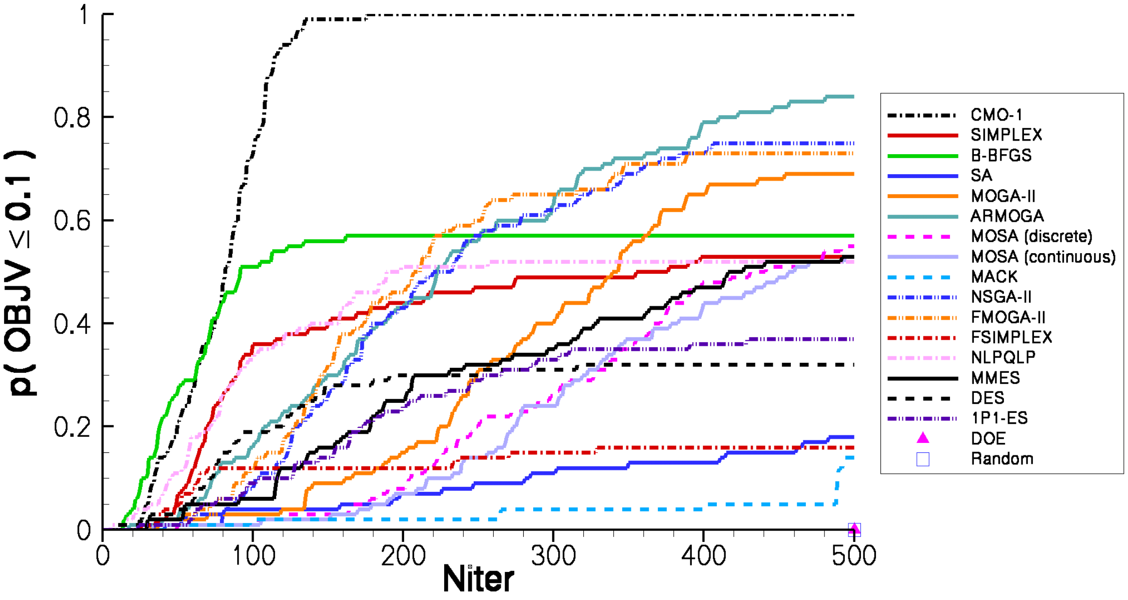

Figure 11. Comparison of probability *p(min(OBJV)* $\le$ 0.1) as a function of the number of designs.

Table 4. Ranking of optimizers based on *p(min(OBJV)* $\le$ *0.2)* performance at *Niter* = 500.

Name	*p(min(OBJV)≤0.2)*			
	Niter = 50	*Niter* = 100	*Niter* = 200	*Niter* = 500
CMO-1	0.30	0.89	1.00	1.00
ARMOGA	0.05	0.24	0.57	0.90
NSGA-II	0.01	0.12	0.54	0.78
MOGA-II	0.04	0.09	0.27	0.76
MOSA (discrete)	0.01	0.05	0.17	0.74
FMOGA-II	0.03	0.18	0.57	0.73
MOSA (continuous)	0.02	0.06	0.16	0.66
B-BFGS	0.29	0.53	0.58	0.58
NLPQLP	0.11	0.36	0.54	0.55
MMES	0.05	0.14	0.32	0.55
SIMPLEX	0.14	0.36	0.45	0.53
SA	0.01	0.08	0.17	0.37
1P1-ES	0.05	0.15	0.30	0.37
DES	0.06	0.21	0.30	0.32
FSIMPLEX	0.12	0.16	0.18	0.23
MACK	0.01	0.03	0.06	0.22

Regardless of the details of which method dominates in which *Niter* range, what is important, in this "optimization of optimizations" is the fact that there exists an "efficient frontier." By definition, a point residing on the efficient frontier (seen as the envelope of the 15 standard method curves in **Figs. 9-11**) is such that (a) for a

[††] (based on the maximum of 500 iterations used in this exercise)

given number of function calls, no other method yields a higher probability of success, and (b) for a given probability of success, all other methods require a greater number of iterations. In other words, any point that is not on the efficient frontier is either riskier (lower probability of success) or costlier (higher number of function calls). It can be seen from the results of **Figs. 9-11** that the present cumulative metamodel optimization method appreciably stretches the efficient frontier.

Further, accepting *de facto* that response surface acceleration techniques are a given, one can limit the comparison of the CMO-1 method to only those optimization methods that use an internal response surface accelerator. Three of the benchmarking methods examined in this study (MACK, FMOGA II, and FSIMPLEX) do so. FMOGA II and FSIMPLEX use an internal adaptive response surface (Kriging) to speed up the search path, thus reducing the overall number of real designs (which are, again, the only ones being counted). The MACK (Multivariate Adaptive Crossvalidating Kriging) method is different: its goal is to adaptively sample the design space where the interpolation is less accurate. The comparison of CMO-1 to these three methods is shown in **Fig. 12**.

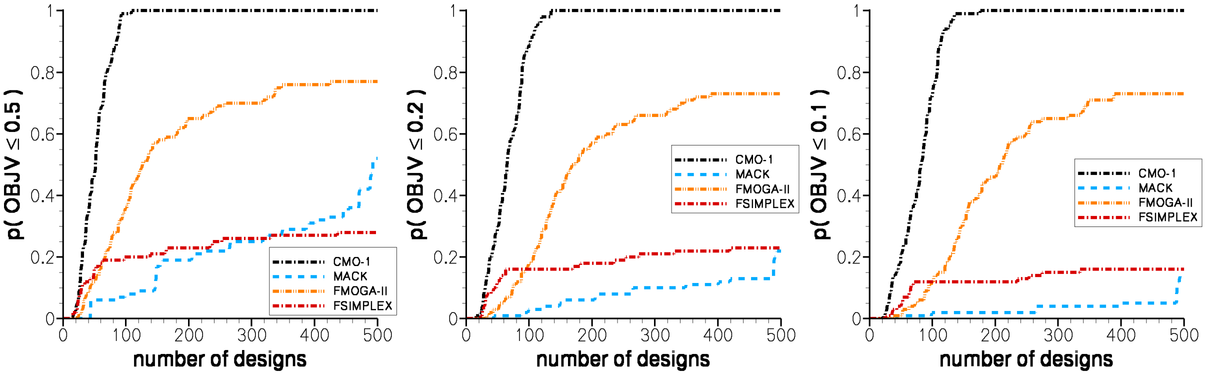

Figure 12. Objective function threshold-based performance comparison of four different response surface-assisted optimization methods based on 100 random trials. Left: *p(min(OBJV)* ≤ 0.5), center: *p(min(OBJV)* ≤ 0.2), right: *p(min(OBJV)* ≤ 0.1).

VI. Concluding Remarks / Discussion

The aerodynamic design of an asymmetric oversized payload fairing subject to stability constraints was used as an example of a derivative-free, expensive black box function to compare the relative performance of 16 different optimization methods. The challenging multimodal nature of this optimization problem required examining the results in a statistical sense, using rigorous application of identical initial conditions for each optimization method, and a conservative count of function calls.

The main result of the benchmarking study is a measure of systematic efficiency, quantifying how much more likely the cumulative metamodel optimization method is of reaching the design objective, compared to any of the other methods tested. This includes methods which use an internal adaptive response surface (Kriging) to speed up the search path. Specifically, an acceleration by at least a factor of five was found in terms of systematic efficiency, based on an "apples-to-apples," comparison of optimization data profiles.

As a cautionary note, several qualifiers are in order. The first is the fact that the present study was performed with each method's default parameters. It is possible, indeed likely, that fine-tuning the adjustable parameters of some of these methods would change their ranking. Second, the present results, while thorough in a statistical/ensemble sense, are for one specific application. This application (the design of a large asymmetric fairing subject to stability constraints) presented specific challenges, such as severe nonlinearity and a multiplicity of local optima. How well the CMO method will perform on different applications and test problems is a question that is left for future research.

Acknowledgments

The authors gratefully acknowledge the support of the optimization benchmarking portion of this work under NASA SBIR Phase I contract number NNX07CA06P. The supporting CFD database was developed under prior funding by the Air Force Research Laboratory (SBIR contracts FA9453-04-M-0132 and FA9453-05-C-0056). Finally, we would like to express our appreciation to ESTECO North America for providing access to the modeFRONTIER^{TM} software for the comparison portion of this study.

References

[1] Reisenthel, P. H., Love, J. F., Lesieutre, D. J., and Childs, R. E., "Cumulative Global Metamodels with Uncertainty - a Tool for Aerospace Integration," The Aeronautical Journal, Vol. 110, No. 1108, Jun. 2006, pp. 375-384.

[2] Jones, D. R., Schonlau, M., and Welch, W. J., "Efficient Global Optimization of Expensive Black-Box Functions," *J. Global Opt.*, Vol. 13, 1998, pp. 455-492.

[3] Sóbester, A., Leary, S. J., and Keane, A. J., "On the Design of Optimization Strategies Based on Global Response Surface Approximation Models," *J. Global Opt.*, Vol. 33, No. 1, 2005, pp. 31-59.

[4] Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., and Tucker, P. K., "Surrogate-Based Analysis and Optimization," *Progress in Aerospace Sciences*, Vol. 41, 2005, pp. 1-28.

[5] Forrester, A. I. J. and Keane, A. J., "Recent Advances in Surrogate-Based Optimization," *Prog. Aero. Sci.*, Vol. 45, Issues 1-3, Jan.-Apr. 2009, pp. 50-79.

[6] Reisenthel, P. H., Allen, T. T., Lesieutre, D. J., and Lee, S. H., "Development of Multidisciplinary, Multifidelity Analysis, Integration and Optimization of Aerospace Vehicles," NEAR TR 657, Nielsen Engineering & Research, Santa Clara, CA, 2010.

[7] Reisenthel, P. H., Childs, R. E., and Higgins, J. E., "Surrogate-Based Design Optimization of a Large Asymmetric Launch Vehicle Payload Fairing," AIAA-2007-361.

[8] Buning, P. G., Jespersen, D. C., Pulliam, T. H., Klopfer, G. H., Chan, W. M., Slotnick, J. P., Krist, S. E., and Renze, K. J., "Overflow User's Manual Version 1.8r," NASA Ames Research Center, 2000.

[9] Spalart, P. R. and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA-92-0439.

[10] Chan, W. M. and Steger, J. L., "Enhancements of a Three-Dimensional Hyperbolic Grid Generation Scheme," *Appl. Math. and Comput.*, Vol. 51, 1992, pp. 181-205.

[11] Macke, M., Roos, D., and Riedel, J., "An Adaptive Response Surface Method Utilizing Error Estimates," 8th ASCE Specialty Conference on Probabilistic Mechanics and Structural Reliability, PMC2000-068, 24-16 July, 2000.

[12] Costa, M. C., Coulomb, J.-L., Marechal, Y., Dietrich, A. B., and Nabeta, S. I., "Diffuse-Element Method and Quadtrees: Two "Ingredients" for an Adaptive Response Surface," *IEEE Transactions on Magnetics*, Vol. 38, No. 2, Pt. 1, Mar. 2002, pp. 1085-1088.

[13] Wang, G. G., "Adaptive Response Surface Method Using Inherited Latin Hypercube Design Points," *J. Mech. Design*, Vol. 125, No. 2, Jun. 2003, pp. 210-220.

[14] Scheibe, T. D., Engel, D. W., Liebetrau, A. M., Jarman, K. D., Ferryman, T. A., and Didier, B. T., "Iterative Response Surface Methods for Quantifying Uncertainty in Environmental Models," PNNL-SA-41690, IDS-Water Americas, 2004.

[15] Han, D. and Chatterjee, A., "Adaptive Response Surface Modeling-based Method for Analog Circuit Sizing," *IEEE Proc. SOC* Conference, pp. 109-112, Sep. 2004.

[16] http://www.esteco.com/products.jsp .

[17] Nelder, J. A. and Mead, R., "A Simplex Method for Function Minimization," *Comput. J.* **7**, 1965, pp. 308-313.

[18] Rigoni, E., "Bounded BFGS," ESTECO TR 2003-007, ES.TEC.O S.r.l., Dec. 2003.

[19] Kirkpatrick, S., Gelatt, C. D., Jr., and Vecchi, M. P., "Optimization by Simulated Annealing," *Science*, 220, 4598, 1983, pp. 671-680.

[20] Sasaki, D. and Obayashi, S., "Low-Boom Design Optimization for SST Canard-Wing-Fuselage Configuration," AIAA 2003-3432.

[21] Rigoni, E., "MOSA Multi-Objective Simulated Annealing," ESTECO TR 2003-003, ES.TEC.O S.r.l., May 2003.

[72] Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T., "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," KanGAL Report 200001, Indian Institute of Technology, Kanpur, India, 2000.

[23] Schittkowski, K., "NLPQLP: A Fortran Implementation of a Sequential Quadratic Programming Algorithm with Distributed and Non-monotone Line Search - User's Guide, Version 2.2," Report, Department of Computer Science, University of Bayreuth, 2006.

[24] Rechenberg, I., *Evolutionsstrategie: Optimierung technischer Systeme und Prinzipien der biologischen Evolution* , Frommann-Holzboog, Stuttgart, 1973.

[25] Schwefel, H.-P., *Numerical Optimization of Computer Models* , Wiley, Chichester, 1981.

[27] Hansen, N. and Ostermeier, A., "Completely Derandomized Self-Adaptation in Evolution Strategies," *Evolutionary Computation*, 9(2), 2001, pp. 159-195.

[27] Reisenthel, P. H. and Lesieutre, D. J., "Cumulative Metamodeling With Uncertainty Estimation: A New Approach to Risk-Based Optimization of Aerospace Vehicles," NEAR TR 632, Nielsen Engineering & Research, Mountain View, CA, Jul. 2007.

[28] Moré, J. J. and Wild, S. M., "Benchmarking Derivative-Free Optimization Algorithms," *SIAM J. Optim.*, Vol. 20, No. 1, pp. 172-191, 2009.