AIAA JOURNAL
Vol. 44, No. 2, February 2006

Supervised Learning Approach to Parametric Computer-Aided Design Geometry Repair

András Sóbester* and Andy J. Keane†
University of Southampton, Southampton, England SO17 1BJ, United Kingdom

Multidisciplinary optimization systems rely increasingly on parametric CAD engines to supply the geometries required by their analysis components. Such parametric geometry models usually result from an uneasy compromise between high flexibility, that is, the ability to morph into a wide variety of topologies and shapes, and robustness, the ability to produce feasible, sensible topologies and shapes throughout most of the design space. It is argued that a possible means of achieving both objectives is via a supervised learning system attached to the CAD model. It is shown that such a model can capture some of the engineering and geometrical judgment of the designer and can thereafter be used to repair design variable sets that lead to infeasible CAD models.

Nomenclature

D	=	design search space
g	=	number of geometrical constraints applied to repair problem
H_1^{orig}, V_1^{orig}	=	original (prerepair) values of coordinates of first spline knot
$\boldsymbol{I}$	=	identity matrix
k	=	number of design variables
N	=	number of training/validation designs
Q	=	deficiency measure threshold value
q	=	number of cross-validation subsets
r	=	distance of current point from radial basis function center
w_i	=	final layer weight of ith kernel
$\boldsymbol{x}$	=	vector of design variables
$\boldsymbol{x}^{\text{orig}}$	=	original (prerepair) design
$\hat{y}$	=	geometry deficiency predictor
σ	=	width parameter of Gaussian kernel
$\Phi_{i,j}$	=	element of Gram matrix (row i, column j)

I. Introduction

THE initial, conceptual phase of any design process is, essentially, a highly global search over the space of possible configurations, topologies, shapes, and dimensions. The solution chosen to progress to the preliminary design stage is ideally one that satisfies all of the constraints specified in the design brief and relevant regulations and optimizes some figure of merit related to performance, cost, revenue, or combinations of these. The multidisciplinary design optimization (MDO) architectures conceived for this selection task are numerous and show considerable diversity. We focus here, however, on an element most of them have in common, namely, the geometry engine. The essential task of this component is to supply the models required by the various numerical simulation codes that make up the multidisciplinary analysis capability of the system.

Bespoke, in-house geometry engines dominate the world of conceptual design. These tools are often tightly coupled with the meshing and analysis capability of the MDO system and are designed for relatively narrow classes of applications. Increasingly, however, we see commercial parametric CAD packages taking center stage in MDO systems as geometry providers,[1] and this is the perspective we adopt here; nonetheless, much of the following discussion is valid for other forms of geometry construction as well.

Arguably, the global nature of the conceptual design process requires a CAD model that encompasses a broad range of possible designs. In an ideal world, a seamless parameterization would be available that ensures that the optimizer can visit a wide variety of configurations and can move between them smoothly, as driven by any objectives. In reality, however, the uniform, flawless coverage of the design space by a generic CAD model is fraught with difficulties. Parameterization technology has come a long way since the first tentative numerical design optimization efforts of the 1970s. (See, for example, Samareh's survey[2] of developments in this field.) As a result, techniques are available today for robust parameterization of relatively simple entities (such as airfoils). Nevertheless, the reliable and flexible parameterization of more complex models is still a considerable challenge. In all but the most trivial or overconstrained cases, the construction of the geometrical model can fail in certain areas of the design space. Of course, if these infeasible areas are rectangular, they can simply be avoided by adjusting the bound constraints on the relevant variables, but this is rarely the case. Most of the time these regions will have complex, irregular shapes and are much harder to identify and avoid. The problem, therefore, usually boils down to the following tradeoff. One can place bound constraints on the design variables that are sufficiently tight to ensure that any possible combination of variables will lead to a feasible design. The drawback here is that the design space will be very limited. Alternatively, wide bound constraints can be used to enable the exploration of a wide variety of designs, with the risk that the model generation process will fail occasionally.

In the present work, we advocate the use of supervised learning systems based on radial basis function (RBF) networks to repair such failed geometries. Repairing bad design variable sets can enable more radical search ranges, as well as contributing to the reduction of design cycle time by avoiding expensive, pointless computations performed on bad geometries. In a conventional automated MDO system bad designs may be highlighted by the failure of the CAD tool to generate a meaningful model, for example, when a feature is defined by the intersection of entities that no longer actually intersect, in which case only a few minutes are lost. However, if the shape is geometrically meaningful, but merely unphysical, for example, when wildly snaking splines define a wing section, alarm bells only start ringing when, for example, the aerodynamic flow simulation fails to converge even after a large number of iterations, by which time hours of CPU time may have been wasted.

Here we argue that a supervised learning system that has captured some of the designer's knowledge of what is likely to constitute a bad geometry could often detect these designs beforehand and deploy

Received 16 April 2005; revision received 2 September 2005; accepted for publication 2 September 2005.

*Research Fellow, Computational Engineering and Design Group. Member AIAA.

†Professor of Computational Engineering and Chair, Computational Engineering and Design Group.

the main function of the RBF network, that of repairing infeasible designs or helping to build feasible designs given certain constraints. In an automated MDO system, the supervised learning model would serve as a tool for finding the healthy design most similar to one that was found to be faulty (either by the RBF network itself or as a result of a geometry building, meshing, or analysis failure), without the need for the user to be involved in the process.

In a manual, or semiautomated conceptual design system, the RBF learner can have another role (in addition to a user-assisted, gradual repair of bad geometries), that of helping the operator implement local design changes without these having a detrimental impact on the rest of the geometry, that is, repairing a geometry when changes had been made to some of its variables and these changes have not propagated sensibly to the rest of the design.

Before we discuss the suggested repair mechanism in more detail, we need to make two important clarifications. First, CAD geometries often exhibit other types of MDO-hindering failures that are not covered by the algorithm described in this paper. These include topological errors (such as missing or duplicate entities, zero-volume parts, or inconsistent surface orientations),[3] geometrical errors (sliver faces, minute edges, loop closure gaps, edge-vertex and edge-face gaps, etc.),[4] or entities with mathematical descriptions that are inappropriate for certain uses (most frequently for mesh generation).[5] Such flaws can also preclude the multidisciplinary analysis process, but they are not strictly related to the parameterization of the geometry and are often invisible. Here we look at higher level failures that are always visible and that are inextricably linked with the parameterization of the geometry. Namely, one can talk about flawed parameter sets as much as the flawed geometries that are generated from them.

Second, failed designs could simply be thrown away instead of investing additional effort into setting up the repair mechanism. After all, many MDO algorithms can cope with unsuccessful evaluations, in particular, population-based search methods (such as genetic algorithms) and statistical techniques (response surface modeling) are fairly robust in this sense. However, the best designs may lie on the cusp between feasibility and infeasibility, and it is, therefore, important that the optimization engine of the MDO system be able to explore these boundaries.

On an abstract level, the approach put forward here is similar to the application of expert systems (ES) technology in fields such as condition monitoring, fault diagnosis, material selection, the planning of manufacturing operations, etc. (For a recent example, see the work by Cemal Cakir et al.[6]) Such systems mimic human experts in some sense: They capture domain-specific knowledge, which they then interrogate using some type of inference mechanism to produce conclusions. In the work presented here, the RBF network aims to capture the designer's ability to assess the degree of deficiency of a CAD geometry by means of a regression model that incorporates both explicit rules and observational data. Note that a variety of other means of knowledge representation can be found in the ES literature. See, for example, the recent comparative study of different rule-based representation methods by Kingston.[7] Here we have selected an RBF network representation for its ease of implementation (in terms of both setup and repair strategy inference), its ability to learn from a mixture of symbolic rules and observational data, as well as its robustness in terms of learning from incomplete or inexact data. (We will delve more deeply into this latter aspect later.)

In the next section, we discuss the technical aspects of constructing and training this supervised learning system. This will be followed by an application of the technique in Sec. III. In Sec. IV, we look back at the experimental studies and we formulate our conclusions.

II. Predicting Geometry Quality with RBF Networks

A. Preliminaries

Consider N instances of a parametric CAD geometry, with the corresponding sets of design variables used to control them denoted by $[\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}]$, where $\boldsymbol{x}^{(i)} \in D \subset \mathbb{R}^k$, $\forall i \in \{1, \ldots, N\}$. Further assume that each of these geometries has been assigned a tag $y^{(i)} \in \mathbb{R}$, which is a measure of its fitness for whatever the purpose of the geometry may be, for example, it could describe its suitability as an aerodynamic component. For the remainder of this paper, we adopt the arbitrary convention that this is a deficiency value, that is, the lower the tag value, the better the design. Let $\boldsymbol{y} = [y^{(1)}, y^{(2)}, \ldots, y^{(N)}]$ denote the vector containing these tags. As an aside, it is usually considered good practice to normalize the $\boldsymbol{x}$ and y data into the $(0, 1)^{N+1}$ unit cube, as this often facilitates the model validation process (more on which later).

Assuming that the geometry deficiency data were generated by an underlying nonlinear mapping $y: D \to \mathbb{R}$, we attempt to learn y by using a single hidden layer feedforward RBF network with linear output transfer functions and Gaussian transfer functions $\phi(r) = \exp(-r^2/2\sigma^2)$ on the hidden nodes.[8] We obtain the output, an approximation to y, in the form of a linear combination of parametric radial basis functions:

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{N} w_i \phi\left(\left\|\boldsymbol{x} - \boldsymbol{x}^{(i)}\right\|\right) \tag{1}$$

The vector of unknown final layer weights $\boldsymbol{w} = [w_1, w_2, \ldots, w_N]^T$ can be found by computing $\boldsymbol{w} = (\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y}^T$, where the Gram matrix $\boldsymbol{\Phi}$ has the form $\boldsymbol{\Phi}_{i,j} = \phi(\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|)$. The role of the regularization parameter λ is to control the amount of regression. (See Ref. 9 for a detailed analysis of the regularization approach.) This should, ideally, be set to the standard deviation of the noise in the response data[10] $\boldsymbol{y}$, but usually neither this, nor the kernel width parameter σ, is known at this stage. We will return to this issue shortly.

Note that in our preceding discussion we have made the tacit assumption that $\Phi + \lambda \boldsymbol{I}$ is nonsingular. This is generally the case. (Under certain assumptions positive definite kernels, such as the Gaussian used here, lead to symmetric positive definite Gram matrices.[11]) However, $\boldsymbol{\Phi}$ can still, theoretically, become ill-conditioned. In such cases Jones et al.[12] recommend augmenting Eq. (1) with an additional polynomial term.

We can now use the model to predict the deficiency score of a new geometry $\boldsymbol{x}$ by computing $\hat{y}(\boldsymbol{x}) = \boldsymbol{\phi w}$, where $\boldsymbol{\phi} = [\phi(\|\boldsymbol{x} - \boldsymbol{x}^{(1)}\|), \phi(\|\boldsymbol{x} - \boldsymbol{x}^{(2)}\|), \ldots, \phi(\|\boldsymbol{x} - \boldsymbol{x}^{(N)}\|)]$.

B. Training, Validation, and Testing

Received wisdom suggests that, in an ideal world, where plentiful training data can be obtained with minimal computational and human effort, one should allocate approximately one-half of the available $\boldsymbol{x} \to y$ pairs for actually fitting the model (training), around 25% for model selection based on prediction error estimation (validation), with the remaining quarter used for the assessment of the predictive capabilities of the learning system (testing).[13] However, the reality of short design cycle times and the high cost of multiple reconstructions and evaluations of geometries demand a more economical solution, where subsets of the training data are reused for model validation.

Training data reuse takes different forms. In the work described here, we select the shape of the model by cross validating it over the available data, which also serve as the training set. Cross validation involves splitting the training data (randomly) into q roughly equal subsets, then removing each of these subsets in turn and fitting the model to the remaining, aggregated, $q - 1$ subsets. A loss function L can then be computed that measures the error between the predictor and the points in the subset we set aside at each iteration. The contributions to L are then summed over the q iterations.

More specifically, if a mapping $\xi : \{1, \ldots, N\} \to \{1, \ldots, q\}$ describes the allocation of the N training points to one of the q subsets, and $\hat{y}^{-\xi(i)}(\boldsymbol{x})$ is the value (at $\boldsymbol{x}$) of the predictor obtained by removing the subset $\xi(i)$, that is, the subset to which observation i belongs, the cross-validation measure, which we employ here as an estimate of the prediction error, is

$$CM = \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, \hat{y}^{-\xi(i)}(\boldsymbol{x}_i)\right) \tag{2}$$

Introducing squared error in the role of the loss function and recalling from the preceding section that our model $\hat{y}$ is still a generic one, depending on the two parameters we left undetermined there, we can rewrite Eq. (2) as

$$CM(\sigma, \lambda) = \frac{1}{N} \sum_{i=1}^{N} \left[y_i - \hat{y}^{-\xi(i)}(\boldsymbol{x}_i, \sigma, \lambda) \right]^2 \tag{3}$$

The goal of the validation process is, of course, to find the exact form of the Gaussian RBF predictor (1) that minimizes the generalization error (sometimes referred to as the true risk). Therefore, those values of σ and λ are chosen that minimize Eq. (3). To what extent Eq. (3) is an unbiased estimator of true risk depends on the choice of q. It can be shown that if $q = N$, CM is an almost unbiased estimator of true risk. However, the variance of this leave-one-out measure can be very high due to the N subsets being very similar to each other. Hastie et al.[13] recommend compromise values of $q = 5$ or 10. In practical terms, using fewer subsets has the added bonus of reducing the computational cost of the cross-validation process by reducing the number of models that have to be fitted. Note that the construction of an RBF approximation requires $\mathcal{O}(N^3)$ operations, where Φ is usually a dense symmetric matrix.

C. Data Preparation Practical Considerations

As mentioned in the Introduction, the main purpose of the tool described here is to repair unsuitable geometries, having also the somewhat serendipitous capability of detecting such failures before they cause disruption in the MDO system. We now tackle the issue of setting up the learning model, which will provide the basis for both of these functions.

The first entry in our recipe for producing a geometry classification and repair tool is the generation of the initial data that the RBF model will be built on. There is no general rule regarding the number of observations needed for an effective geometry deficiency predictor. Note, however, that the curse of dimensionality demands exponentially increasing numbers of observations as the number of design variables increases. Ultimately, the first attempts at repairing faulty geometries will show whether the data were sufficient or not.

It is clearer, however, that the training/validation data have to fill the design space in a uniform manner. (For the sake of simplicity, we substitute, for the remainder of this section, the term "training data" for both of these categories.) A vast amount has been written in the statistics literature on what space filling really means and how an experimental design can achieve it. The interested reader should consult, for example, the authoritative text by Montgomery.[14] Here we use an experimental design that offers a good compromise between favorable projective properties and good space-filling. That is, it covers all dimensions of the design space as uniformly as possible. To ensure that we meet the first requirement, we start from a Latin hypercube (LH) design.[15] To make sure that the design space is filled uniformly, even in cases where the dimensionality of the space is high relative to the available training point budget, we modify this using the iterative technique by Morris and Mitchell,[16] whereby we optimize a ranking coefficient based on the minimum distance between any two points within the hypercube (maximin criterion).

The next stage in the construction of the repair system is the tagging of the data. The tags are the known, training values of the response y that measure the geometrical deficiency of the training designs. We consider y as having the form

$$y(\boldsymbol{x}) = \begin{cases} T_1 & \text{if} \quad R_1(\boldsymbol{x}) \le 0 \\ \cdots & \cdots\cdots\cdots\cdots \\ T_t & \text{if} \quad R_t(\boldsymbol{x}) \le 0 \\ T(\boldsymbol{x}) & \text{otherwise} \end{cases} \tag{4}$$

where $T_1, \ldots, T_t$ are a set of tag values that we can assign to designs that obey some explicit, known rules $R_1, \ldots, R_t$ and where $T : D \to \mathbb{R}$ is an unknown function that remains to be approximated by the RBF learner. In other words, in the regions of the search space where the explicit rules hold, we do not need to approximate the designer's likely deficiency assessments. We already know them and can, thus, fill in the blanks in the relevant parts of y. The rulebase $R_1, \ldots, R_t$ comprises relationships involving one or more of the design variables, usually discovered by using engineering or geometrical judgment. For example, given a spline on which the x coordinates of two consecutive control nodes are x_1 and x_2, we may not want to restrict the ranges of these two variables, but we know that if $R_1 = x_1 - x_2 \ge 0$, the spline will have a loop, which renders the geometry useless. Therefore, a high value of T_1 can be assigned to that training point without the need for actually examining that geometry. These rules can, of course, be more complex, and, in practice, their evaluation may even require the actual construction of the geometry, for example, if we have imposed a constraint on the curvature of the surface and we wish to use the CAD engine to calculate this.

In general, there will be a large number of designs that we will not be able to tag simply based on the symbolic, explicit rules. After all, we would not need an approximation model if we had an analytical way of determining most of the responses. These remaining designs will have to be assessed individually, typically by visual examination, giving the term supervised learning a very literal meaning. Most CAD packages have an automated catalog generation capability, which can be very useful for this task. More specifically, a catalog of all of these designs can be generated, and all the operator has to do is to view the catalog design by design and enter the corresponding ratings in a spreadsheet (generally, the same spreadsheet that contains the design table used by the CAD tool to generate the catalog).

A note is in order here regarding the computational resource allocation aspects of generating these training models. If the CAD model is complex, instantiating it for hundreds of design variable combinations may be a fairly time-consuming process. Though this can be run as a background or overnight job (thanks to the automated cataloging facilities of some CAD packages), for the method advocated here to be viable, the overall computational cost of the training process has to be small in comparison with the cost of the analyses performed throughout the design process. The average time it takes for analysis failures to become apparent (through, for example, divergence of a flow solution) should also be considered because in the rare cases when failure is always immediately obvious the additional overheads involved in implementing the technique may not be worthwhile. Nevertheless, even in such situations, one must not underestimate the value of obtaining repair information, which, of course, we would not have without building the RBF network. A further factor that could tip the balance in favor of setting up the supervised learner is the potential need to incorporate engineering knowledge that is not related to analysis failures.

Let us now return to the actual design scoring process. To use an academic comparison, this is akin to marking large numbers of students' assignments: A clear set of benchmarks has to be established (for example, in the form of a marking matrix, as illustrated in Table 1), after which the same, uniform standards have to be maintained throughout the tagging process.

When very complex geometries and, thus, vast training sets are dealt with, this tagging may be (or may have to be) conducted by several people. In such cases extra care is required to maintain consistent marking standards throughout the pool of operators performing the tagging. This could be achieved, for example, by reciprocal supervision between the members of the team for part of the marking process. We note here that such a team approach to the training of the supervised learning model is very much in the spirit of ES because

Table 1 Scoring matrix devised for nacelle design example

Failure mode	Spline snaking at front	Spline snaking in fan casing section	Piccolo tube too far from leading edge
None	0	0	0
Mild	1	1	1
Moderate	2	2	2
Severe	3	3	3

they are often viewed as instruments of capturing the knowledge of several individuals and subsequently making inferences based on this aggregated knowledge base. In the case of our geometry repair advisor, this is not limited to the collection of observational data (the labeling process). The rules $R_1, \ldots, R_t$ would also be typically provided by different individuals or departments.

A final task related to the preparation of the initial data is the establishment of a threshold value Q of the geometry deficiency measure, above which we consider a model infeasible. (Depending on the MDO architecture, such models can then be rejected or repaired.) The reliability vs globality question arises here again, in a different guise. The smaller the value of Q, the more certain we can be that designs with deficiency predictions below it will be feasible, but the added confidence will carry a search space reduction penalty. Of course, should the regression model fail to detect a bad geometry, we can still repair it once the failure has been highlighted by errors appearing in the analysis modules.

Once the entire training set is tagged, the RBF network can be validated and trained (as described earlier) and the tool is ready to fulfill its purpose, that is, to provide a geometry deficiency landscape that serves as the basis for the repair algorithm.

D. Repair

With the RBF network built and correctly trained on a sufficient amount of data, using it for bad geometry detection and repair is straightforward.

For any $\boldsymbol{x} \in D$, if $\hat{y}(\boldsymbol{x}) \leq Q$, then $\boldsymbol{x}$ is expected to be feasible, otherwise it can be rejected. Note that if this type of classification were our only goal (as in Ref. 17), we could have tagged the training designs simply with one of two class labels, for example, 1 for feasible and -1 for infeasible. The reason why the framework presented here is based on a scale of values instead is to give the repair algorithm as much deficiency landscape gradient information as possible. (If we only needed to classify designs and, thus, tag the training set with -1 and 1, certain automated methods of training could be considered, for example, a low-cost mesh generator and/or flow solver could be run on each design, and the geometries that lead to convergence errors could be classified as failed.)

As we mentioned in the Introduction, the repair of faulty geometries can be viewed as a mechanism for exploring the boundaries of the feasible design space. It is, therefore, essential that we identify the smallest possible repair alteration (SPRA), that is, the vector of design variable increments that will make the design feasible while keeping changes to a minimum. Let the following definition clarify what we mean by this.

Definition: $\Delta\boldsymbol{x} \in \mathbb{R}^k$ is defined as an SPRA made to a geometry $\boldsymbol{x}^{\text{orig}} \in D$ iff $\hat{y}(\boldsymbol{x}^{\text{orig}} + \Delta\boldsymbol{x}) \leq Q$ and $\forall \boldsymbol{x}$, where $\hat{y}(\boldsymbol{x}) \leq Q$, $\|\boldsymbol{x}^{\text{orig}} - \boldsymbol{x}\| \geq \|\Delta\boldsymbol{x}\|$.

Therefore, in a fully automated MDO system, should we encounter a design predicted to be infeasible $[y(\boldsymbol{x}^{\text{orig}}) > Q]$, we would seek to identify $\Delta\boldsymbol{x}$, so that we could replace the failed design with $\boldsymbol{x}^{\text{orig}} + \Delta\boldsymbol{x}$. From a practical standpoint, it is perhaps easiest to search directly for this repaired design.

Problem Formulation 1: We seek the value of $\boldsymbol{x}$ that minimizes the distance metric $\|\boldsymbol{x}^{\text{orig}} - \boldsymbol{x}\|$, subject to the inequality constraint $\hat{y}(\boldsymbol{x}) \leq Q$.

As far as the exact search method employed is concerned, the choice is fairly open because both the objective (the distance metric) and the constraint (the deficiency predictor) are cheap to evaluate, therefore, optimizer efficiency is relatively unimportant. Furthermore, the predictor is a linear combination of continuous basis functions; therefore, the constraint boundary can also be expected to be continuous.

A broadly similar formulation can be used for user-assisted repair (UAR) in manual or semiautomated MDO systems. In the automated system scenario, we were searching for the nearest feasible design, where feasibility was determined by $\hat{y}$ being below the threshold Q. This does not have to be the case here. UAR offers more freedom to the user in the sense that the alteration to the design does not necessarily have to be the SPRA. The designer has visual control over the process in this case and, thus, has the ability to balance fidelity to the original concept against feasibility. In other words, solutions are sought to the double-objective problem of simultaneously minimizing the distance metric and the deficiency predictor.

It is worth discussing multiobjective search problems here, to be more precise, problems with multiple conflicting objectives. Such problems only have a single, well-determined solution when we have some means of assigning a precise weighting to each objective, that is, we can quantify their relative importance. In such cases the problem can be reduced to a single-objective formulation, where we are optimizing a linear combination of the initial objectives. For example, in the present case, if we could express the relative importance of being close to the original design and having a low deficiency value through a coefficient κ, we could simply optimize the function $\kappa\,\hat{y}(\boldsymbol{x}) + (1-\kappa)\|\boldsymbol{x}^{\text{orig}} - \boldsymbol{x}\|$. This, however, is rarely the case. Typically we have no way of capturing the relative weight of our goals and, therefore, we seek a range of compromise solutions. More specifically, we need to identify those nondominated solutions (also known as Pareto designs), that can only be improved on for any objective at the detriment of one or more of the other objectives. (A design is said to dominate another if it is better on one objective and not worse on any other.) In terms of our double-objective problem, we need to identify those geometries that can only be made more similar to the original design at the detriment of their deficiency rating and can only be improved on from the deficiency point of view by being moved away in the design space from the original concept.

Problem Formulation 2: We seek values of $\boldsymbol{x}$ that satisfy a set of constraints $\Gamma_i(\boldsymbol{x}) \geq 0$, $\forall i \in \{1, \ldots, g\}$ and for which any change $\delta\boldsymbol{x} \in \mathbb{R}$ leading to $\|\boldsymbol{x}^{\text{orig}} - (\boldsymbol{x} + \delta\boldsymbol{x})\| < \|\boldsymbol{x}^{\text{orig}} - \boldsymbol{x}\|$ will also result in $\hat{y}(\boldsymbol{x} + \delta\boldsymbol{x}) > \hat{y}(\boldsymbol{x})$ and for which any change $\delta\boldsymbol{x}' \in \mathbb{R}$ leading to $\hat{y}(\boldsymbol{x} + \delta\boldsymbol{x}') < \hat{y}(\boldsymbol{x})$ will also result in $\|\boldsymbol{x}^{\text{orig}} - (\boldsymbol{x} + \delta\boldsymbol{x}')\| > \|\boldsymbol{x}^{\text{orig}} - \boldsymbol{x}\|$. (Note that $\delta\boldsymbol{x}$ and $\delta\boldsymbol{x}'$ denote here changes of any magnitude, that is, we are looking for globally nondominated solutions.)

The repair process is, therefore, a review of such Pareto-optimal solutions (which are collectively referred to as the Pareto front), the balance between the two objectives giving the ultimate solution $\boldsymbol{x}$, being determined (implicitly) by the judgment of the designer. In other words, generating multiple solutions, which the designer can choose from, circumvents the need for an explicit, numerical weighting of the objective.

We note here that the design sought in Problem Formulation 1 is, in fact, one of these Pareto-optimal solutions, specifically, the design whose feasibility objective is $\hat{y}(\boldsymbol{x}) = Q$ and distance objective is the norm of the SPRA ($\|\Delta\boldsymbol{x}\|$). As per the Definition, any reduction in the distance metric would cause $\hat{y}$ to increase; therefore, the design is on the Pareto front.

There is vast literature on algorithms for Pareto-front identification. It is beyond the scope of the present work to review the plethora of algorithms available for such searches. The reader may wish to consult, for example, recent work by Deb et al.[18] Note, however, that the hallmark of a good multiobjective search algorithm is the ability to produce a uniform spread of points along the Pareto front. Identifying clusters of very similar Pareto-optimal (nondominated) points with large gaps in between is rarely desirable.

Also note that, as in the case of Problem Formulation 1, the two functions are continuous, and the distance metric is also unimodal; therefore, the multiobjective problem should be reasonably easy to solve.

The equality or inequality constraints $\Gamma(\boldsymbol{x})$ referred to in Problem Formulation 2 are an additional instrument at the disposal of the user for making ad hoc design decisions, for example, by fixing certain variables during the repair process or constraining them to smaller ranges. There is no reason why such constraints could not be applied to Problem Formulation 1 as well. We merely mention them here because they are more likely to be useful in a UAR setting.

We now show the two problem formulations described by means of an example, namely, the design of a jet engine nacelle.

III. Application: Design of Jet Engine Nacelle

A. Problem Setup

The CATIA® model of the geometry we propose to examine is shown in Fig. 1. We focus on the handling of the geometry of the

Table 2 Bound constraints and values of the 10 design variables used for designs shown in Figs. 1 and 2

Design	H_1, m	V_1, m	H_2, m	V_2, m	H_3, m	V_3, m	H_4, m	V_4, m	H_5, m	V_5, m
Lower bound	0.25	0.59	0.4	0.5	0.6	0.45	0.4	0.45	0.1	0.45
Upper bound	0.5	0.63	0.8	0.62	1.0	0.62	0.8	0.55	0.5	0.55
Fig. 1 example	0.365	0.628	0.603	0.606	0.948	0.472	0.642	0.478	0.440	0.489
Fig. 2 example	0.282	0.617	0.573	0.523	0.787	0.579	0.532	0.458	0.330	0.463

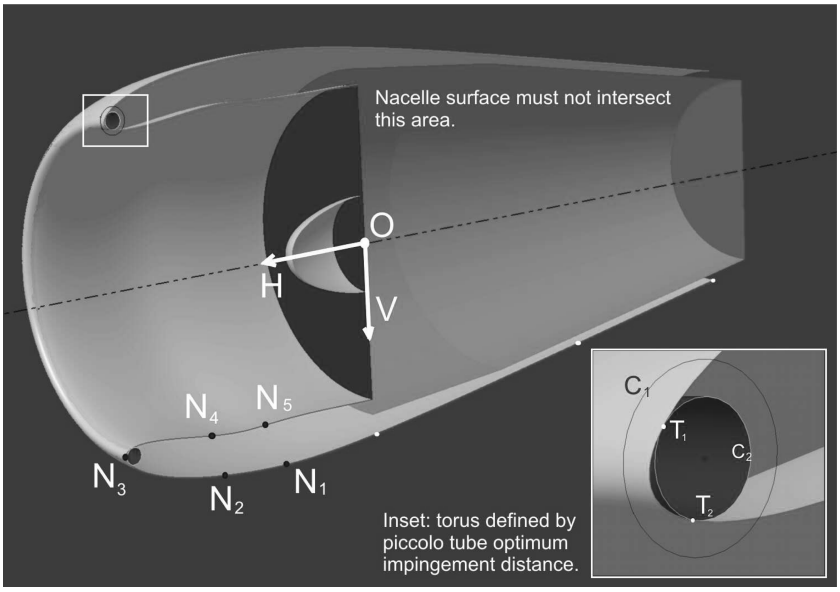

Fig. 1 Longitudinal section through CATIA model of jet engine nacelle.

nacelle lip surface. This is a B-spline, defined by seven knots, the first five of which (N_1–N_5) have two degrees of freedom each. (They are allowed to move in the symmetry plane.) The surface of the nacelle is a surface of revolution, whose generator is the spline going through the five variable control points and the three fixed knots defining the aft part of the cowl. The end of the spline nearest to N_5 is additionally constrained to be normal to the fan face.

Table 2 contains the bounds placed on the design variables, as well as their values for the instances shown in Figs. 1 and 2. H_i and V_i denote the horizontal and vertical coordinates of control point N_i, measured in an axis system with its origin in the fan face center, the H axis pointing upstream and the V axis pointing downward.

The geometry can fail by violating any of the following constraints. First, we assume that the dark area behind the fan face is the external envelope of the fan casing and the bypass air duct and, therefore, must not be intersected by the nacelle surface. Such intersections could be due to excessive snaking of the spline aft of the fan face. High-amplitude snaking may render the design infeasible even if the bump is on the outside, that is, if no intersection occurs.

The second constraint relates to de-icing requirements: We define a toroidal region whose radius is equal to the optimum impingement distance of the piccolo tube, which has to fit inside the leading edge of the lip. (See Ref. 19 for a discussion of issues related to piccolo tube design.) The CAD model has been constructed in such a way that it automatically finds a location for this torus as near to the leading edge as possible, without causing interference between the torus and the nacelle surface. We assume that the optimum impingement distance (and, therefore, the radius of C_2, Fig. 1 inset) is 0.025 m and that no part of the nacelle surface between the two tangency points T_1 and T_2 should deviate by more than 0.015 m from this optimum distance. Therefore, the radius of circle C_1 is 0.04 m and the leading edge of the lip must be inside this circle for the geometry to be considered feasible. In other words, sharp leading edges must be avoided because they may not accommodate the optimum impingement distance torus, and, thus, the piccolo tube would have to be too far from the leading edge.

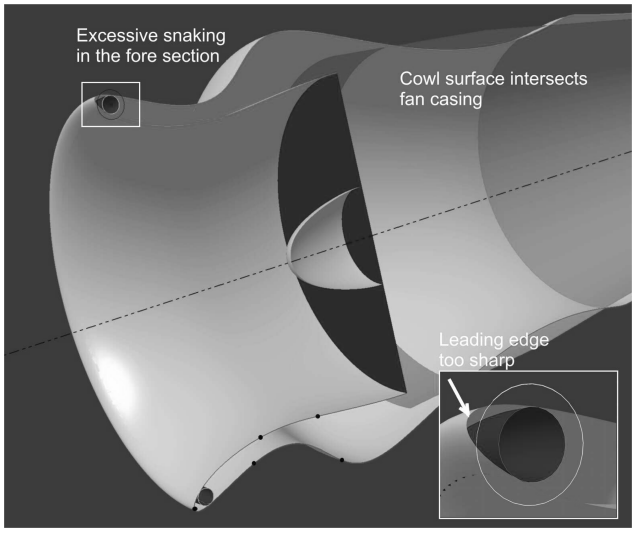

Fig. 2 Geometry that exemplifies all three types of failure.

Finally, excessive snaking and sharp transitions (steep gradients) in the spline forward of the fan face can also render the geometry infeasible.

The scoring system we use for this example is based on these three modes of failure. The deficiency measure of the design ranges from 0 (satisfies all three requirements, like the instance shown in Fig. 1) to 9 (severe failure on all three counts). The corresponding marking matrix is shown in Table 1. As a further example, Fig. 2 shows a failed design with a score of 4.5. In Fig. 2, 1 penalty point is due to the inlet leading edge being slightly outside the maximum allowed impingement distance of the piccolo tube (highlighted in the bottom right-hand corner of Fig. 2), a further 1.5 penalty points have been assigned to the geometry because of the excessive snaking in the

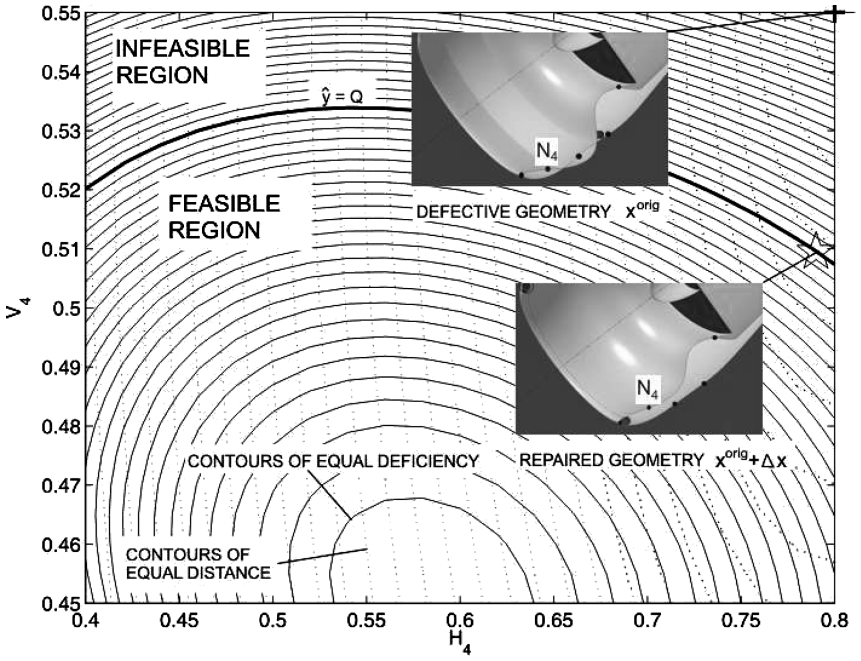

Fig. 3 Contour plots of deficiency predictor $\hat{y}$ and distance metric, where H_4 and V_4 are allowed to vary and remaining eight variables are held constant at the values corresponding to +, defective geometry and ☆, geometry obtained by adding SPRA (Δx) to defective geometry.

fore section, and, finally, 2 points are due to the snaking nacelle surface intersecting the fan casing.

Note from the lower and upper bounds of the design variables (Table 2) that their ranges often overlap, which leads to the possibility of some of the knots of the spline coming very close to each other (or even overtaking each other), thus leading to highly unstable and/or unphysical geometries. (As an example of unstable behavior, consider the situation when the horizontal distance between knots N_1 and N_2 diminishes to, for example, 5% of the value measured in Fig. 1. With such a tight spacing, any small change in V_1 and/or V_2 can clearly lead to dramatic overall shape changes due to the high sensitivity with respect to V_1 and V_2 of the local gradient of the spline.) Also, if V_1 and V_5 or V_2 and V_4 are very similar to each other, the thickness of the inlet lip may become very small and, thus, structurally undesirable. We distill these examples of geometrical and engineering judgment into five explicit rules that we incorporate into the deficiency measure function by assigning them a penalty value of 9:

$$y(\boldsymbol{x}) = \begin{cases} 9 & \text{if} \quad H_5 + 0.075 \geq H_4 \\ & \text{or} \quad H_1 + 0.075 \geq H_2 \\ & \text{or} \quad V_1 \leq V_5 + 0.06 \\ & \text{or} \quad V_2 \leq V_4 + 0.06 \\ & \text{or} \quad \max\{H_2, H_4\} + 0.03 \geq H_3 \\ T(\boldsymbol{x}) & \text{otherwise} \end{cases} \tag{5}$$

B. Training RBF Network

We have trained and validated the RBF model on 2500 designs arranged in a 10-dimensional, Morris–Mitchell[16] optimal LH. After assigning the deficiency value of 9 to those elements, which obeyed at least one of the rules included in Eq. (5), we had 667 untagged points left (26.7% of the total). We instantiated these geometries and collated them into the catalog file, which we then used to conduct the visual tagging process, as described earlier. This took one of the authors approximately 6 h. Additionally, we have determined during this process a feasibility threshold value of $Q = 1$. Note that the time required for the visual inspection process does not, usually, depend on the complexity of the model. Thus, 6 h can be considered a short period of time when viewed with respect to the time it takes to build a good parametric CAD geometry.

With the entire set thus tagged, the RBF network could be trained and validated. We opted for a 10-fold ($q = 10$) cross-validation procedure to do this.

C. Repair

Let us begin the demonstration of the repair capabilities of the learning system with an automated repair scenario, based on Problem Formulation 1. The two designs shown in Fig. 3 originate from the geometry represented in Fig. 1. To allow us a clear pictorial description of the repair process, we have kept most of the variables the same as on the original design (Fig. 1), with only H_4 and V_4 being allowed to vary. Thus, we can plot the equal deficiency contours (shown with continuous curves in Fig. 3) and the equal distance metric contours (shown with dotted curves in Fig. 3). The latter are the level curves of a sphere centered around a failed design $\boldsymbol{x}^{\text{orig}}$, located in the upper right-hand corner of Fig. 3. Alongside it, the CATIA rendering of the geometry is also represented. It can clearly be seen that the inside and outside surfaces of the nacelle intersect due to excessive snaking, and, thus, the piccolo tube can only be accommodated a long way away from the leading edge. (The predicted deficiency of this geometry is $\hat{y} = 5.1$.)

Repair, based on the earlier discussion, is equivalent here to implementing Problem Formulation 1, that is, attempting to locate the constrained minimum of the distance metric function. The feasible and infeasible parts of this design space reduced to two dimensions are delimited by the $\hat{y}(\boldsymbol{x}) = Q = 1$ curve of equal deficiency as highlighted in Fig. 3. (Recall that during the visual tagging process we have established one as being the threshold below which we can consider designs acceptable.) The minimum of the feasible part of the distance metric landscape is indicated in Fig. 3 by a star, with the corresponding CATIA rendering shown alongside. It is clear that this geometry is much improved and is a viable design alternative.

As discussed in Sec. II, the UAR process offers the designer a control of the tradeoff between implementing the original design change (which led to the failure being dealt with) as closely as possible and improving the health of the geometry. The information provided by this repair process (as per Problem Formulation 2) is more than just a final design: It is a collection of Pareto-optimal solutions that offer a variety of different tradeoffs between predicted deficiency and similarity to the original design.

To illustrate this, let us consider the failed initial geometry in the bottom left-hand corner of Fig. 4. Allowing the coordinates of node N_3 to change, we turn to Problem Formulation 2, where we define

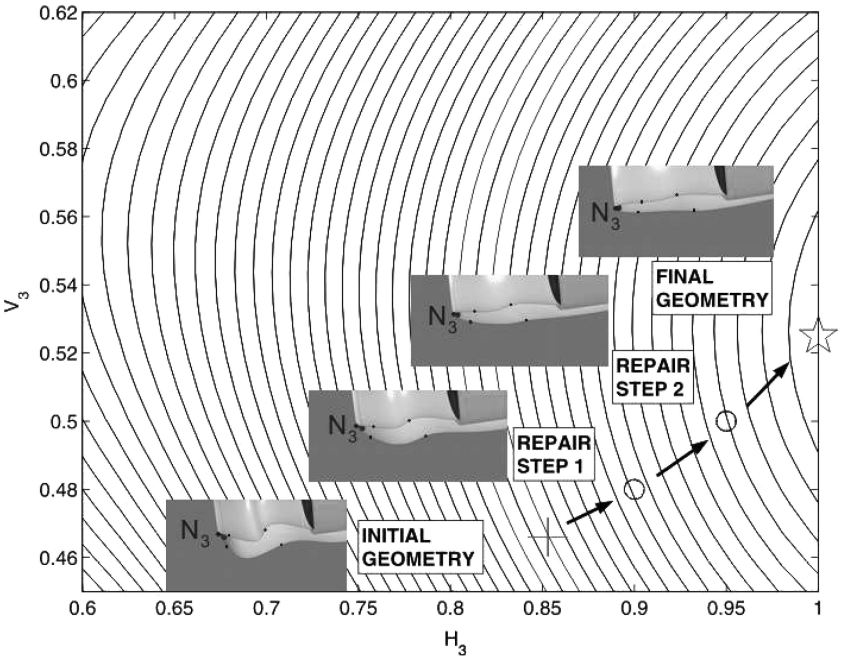

Fig. 4 Contour plot of deficiency predictor $\hat{y}$, where H_3 and V_3 are allowed to vary and the remaining eight variables are held constant at values corresponding to the initial geometry: ○ and ☆, two intermediate geometries and the final geometry, respectively, the latter corresponding to the basin optimum (against the upper bound constraint on H_3); ○, repair step 1 and repair step 2 and ☆, final version, are all nondominated solutions of multiobjective repair problem, as per Problem Formulation 2.

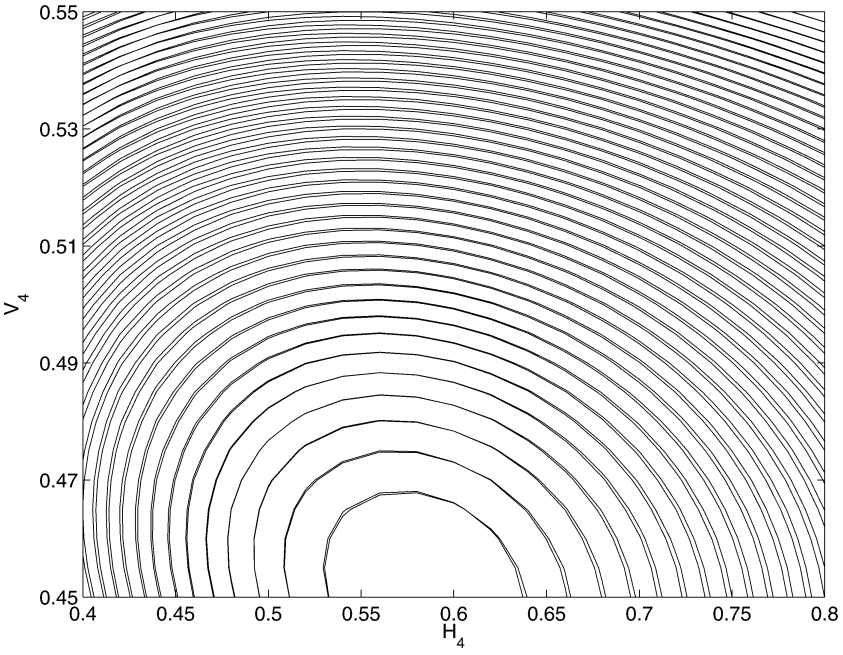

Fig. 5 Superimposed contour plots of $\hat{y}$ before and after artificial corruption of the design deficiency score data.

a set of equality constraints Γ fixing all variables except H_3 and V_3. ($\Gamma_1 : H_1 - H_1^{\text{orig}} = 0$ and $\Gamma_2 : V_1 - V_1^{\text{orig}} = 0$, etc.) Solving the multiobjective optimization problem, we can obtain a string of nondominated geometries of increasing smoothness, three of which are shown in Fig. 4. Moving any of these points along the contours of equal deficiency will move them farther away from the original design (note the differently scaled axes that give a slightly distorted picture of any distances), and, conversely, moving any of them closer to the original design will mean climbing onto higher deficiency level curves. In other words, they are nondominated solutions.

The nacelle labeled final geometry marks the minimum of the deficiency landscape, but, out of the highlighted Pareto-optimal solutions, it is the farthest from the initial geometry.

The designer, thus, has the ability to review all of these designs (usually in the order of increasing distance from the original design, as shown by the arrows in Fig. 4) before making a decision as to which solution is the most suitable.

Of course, in the cases presented here, the astute reader will have been able to predict, at least roughly, which way the knots of the spline would have to move to improve the geometry. This, however, is unlikely to be the case for the much more complex geometries often encountered in aerospace design. Here our intention was merely to illustrate the process on more clear-cut cases, where the workings of the repair process are not obscured by the complexity of the parameterization.

D. Fault Tolerance: Human Factors Perspective

The supervised learning process of transferring elements of the engineering knowledge of the designer to the RBF network is based on a certain level of trust in his or her ability to give consistent marks to the designs examined in the network training phase. In an ideal world, given the universal approximation properties of RBF models, the neural network could be taught to give almost exactly the same answers (deficiency scores), when confronted with new designs, as the human operator who trained it. (Of course, an astronomical number of training points would be required.) This is, however, in practice, not achievable because the underlying value model in the human brain is corrupted by noise. Instead of a neuroscientific analysis of the meaning of the term noise in this context, we offer the following example by way of illustration. If we inserted two identical geometries at random locations into the training set, unless they ended up being very close to each other (according to the order in which they are assessed), it is far from certain that the designer would give them exactly the same mark. This is especially true for complex scoring schemes. This inherent noise, which is the reason why we based the learning algorithm on a regression model in the first place (instead of an interpolating one), could be compounded by environmental factors, such as interruptions, lapses of concentration, etc.

To assess the impact of such factors or, in other words, to measure the fault tolerance of the supervised learning system, we have devised the following experiment. We added a Gaussian random perturbation with zero mean and a standard deviation of two marks (as per the $0, \ldots, 9$ scale used in the example) to a random sample of 100 of the 667 designs tagged by visual examination (15% of the total). With the response vector $\boldsymbol{y}$ thus altered, we rebuilt the RBF model, and we set up the repair scenario depicted in Fig. 3 again. The result is shown in Fig. 5. Two superimposed contour plots are represented here, both of $\hat{y}$, as in the first repair example described earlier. One of them is reproduced exactly from Fig. 3, whereas the other is based on the corrupted training data. Clearly, there is little change in the shape of the response surface model.

Although this simple experiment does not allow us to draw definitive conclusions, we can conclude that there is some empirical evidence that the method is robust enough to cope with reasonable levels of human fallibility.

IV. Conclusions

An effective CAD geometry engine is becoming the sine qua non of many modern MDO processes, and the smoothness of its operation can have a significant impact on the entire design process. Building a parametric CAD geometry is not an easy task by any means because it usually involves finding the elusive best compromise between the conflicting goals of robustness and flexibility. The work described suggests a solution for improving the former, while not compromising the latter. We advocate the use of an RBF network as a means of capturing the geometrical and engineering judgment of the designer, with the ultimate goal of using the rules learned in this process to repair faulty geometries. By faulty, here we mean those geometries that the concept generator (the module of the MDO framework that produces the sets of design variables) and the CAD engine would see as being healthy, but the trained eyes of an engineer would immediately identify as being unphysical or infeasible in some other, practically unquantifiable way.

We have shown that the RBF learning machine is not only a feasible and effective tool for this task, but also that it is relatively straightforward to implement. It is robust, and the additional overheads are small enough to make it a worthwhile design time investment.

The system is essentially a means of capturing the knowledge of an expert and deploying it automatically as and when required. In addition to eliminating the need for the designer to be present when geometry repairs need to be carried out (highly impractical considering the drive toward automated MDO systems), it often produces inferences (complex ways of repairing geometries) that the human expert, who trained the system, may not be able to devise in a reasonable amount of time.

At the present time, off-the-shelf CAD engines are indispensable as detailed design tools and are frequently used at the preliminary design stage as well. Conceptual design, however, is still dominated by purpose-built, in-house geometry generators, which are costly to set up and difficult to integrate into the overall design loop. We believe that part of the reason why commercial CAD packages are not used more widely at the conceptual design stage lies in the difficulty of balancing CAD model robustness against design search scope (flexibility). It is hoped that the repair mechanism described here may go some way toward increasing the uptake of parametric CAD at the conceptual design level.

Acknowledgments

This work has been jointly supported by BAE Systems and the Engineering and Physical Sciences Research Council (United Kingdom) as part of the Integrated Programme of Research in Aeronautical Engineering. The authors also thank Prasanth Nair, Alexander Forrester, and Nicola Hoyle for their useful suggestions.

References

[1]La Rocca, G., Krakers, L., and van Tooren, M. J. L., "Development of an ICAD Generative Model for Blended Wing Body Aircraft Design," AIAA Paper 2002-5447, Sept. 2002.

[2]Samareh, J. A., "Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization," *AIAA Journal*, Vol. 39, No. 5, 2001, pp. 877–883.

[3]Barequet, G., Duncan, A. C., and Kumar, S., "RSVP: A Geometric Toolkit for Controlled Repair of Solid Models," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 4, No. 2, 1998, pp. 162–177.

[4]Samareh, J. A., "Status and Future of Geometry Modeling and Grid Generation for Design and Optimization," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 97–104.

[5]Ribó, R., Bugeda, G., and Oñate, E., "Some Algorithms to Correct a Geometry in Order to Create a Finite Element Mesh," *Computers and Structures*, Vol. 80, No. 16–17, 2002, pp. 1399–1408.

[6]Cemal Cakir, M., Irfan, O., and Cavdar, K., "An Expert System Approach for Die and Mold Making Operations," *Robotics and Computer-Integrated Manufacturing*, Vol. 21, No. 2, 2005, pp. 175–183.

[7]Kingston, J., "High Performance Knowledge Bases: Four Approaches to Knowledge Acquisition, Representation and Reasoning for Workaround Planning," *Expert Systems with Applications*, Vol. 21, No. 4, 2001, pp. 181–190.

[8]Lowe, D., "Radial Basis Function Networks and Statistics," *Statistics and Neural Networks—Advances at the Interface*, Oxford Univ. Press, Oxford, 1999, pp. 65–95.

[9]Poggio, T., and Girosi, F., "Regularization Algorithms for Learning that are Equivalent to Multilayer Networks," *Science*, Vol. 247, No. 4945, 1990, pp. 978–982.

[10]Keane, A. J., and Nair, P. B., *Computational Approaches to Aerospace Design: the Pursuit of Excellence*, Wiley, Chichester, England, U.K., 2005, Chap. 5.

[11]Vapnik, V., *Statistical Learning Theory*, Wiley, New York, 1998, Sec. 1.

[12]Jones, D., Schonlau, M., and Welch, W., "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, Vol. 13, No. 4, 1998, pp. 455–492.

[13]Hastie, T., Tibshirani, R., and Friedman, J., *The Elements of Statistical Learning*, Springer-Verlag, New York, 2001, Chap. 3.

[14]Montgomery, D. C., *Design and Analysis of Experiments*, Wiley, Chichester, England, U.K., 2000.

[15]McKay, M. D., Beckman, R. J., and Conover, W. J., "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, Vol. 21, No. 2, 1979, pp. 239–245.

[16]Morris, M. D., and Mitchell, T. J., "Exploratory Designs for Computer Experiments," *Journal of Statistical Planning and Inference*, Vol. 43, No. 3, 1995, pp. 381–402.

[17]Sóbester, A., and Keane, A. J., "Classifier Systems Can Reduce Conceptual Design Cycle Time," Centre of Excellence for Integrated Aircraft Technology, Paper 2005-0022, Aug. 2005.

[18]Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 2002, pp. 182–197.

[19]Wright, W. B., "An Evaluation of Jet Impingement Heat Transfer Correlations for Piccolo Tube Application," AIAA Paper 2004-0062, Jan. 2004.

E. Livne
Associate Editor