AIAA JOURNAL
Vol. 35, No. 6, June 1997

Automatic Differentiation in Robust Optimization

J. Su* and J. E. Renaud[†]
University of Notre Dame, Notre Dame, Indiana 46556

In this study the use of automatic differentiation techniques significantly reduces the number of analysis calls and the CPU time required for robust optimum design. Traditionally, robust optimum design procedures have relied on finite differencing techniques to evaluate sensitivities for use in gradient-based optimization. Robust objective functions and robust constraints are typically formulated as compound functions that invoke the original objective or constraint many times for each robust evaluation. In this research, automatic differentiation techniques are used to avoid the costly finite differencing of robust objective functions and robust constraints. Sensitivities calculated using automatic differentiation are exact and therefore enhance performance. Two new robust optimization extensions are developed to evaluate the utility of using automatic differentiation in robust optimization. One is a sensitivity-based procedure, and the other makes use of experimental design techniques. These two new robust optimization extensions are successfully implemented in application to an aircraft concept sizing test problem.

I. Introduction

INCREASING emphasis in engineering design is focused on accounting for manufacturing variations and functional variances in operation. Engineers have traditionally relied on postoptimality analysis to evaluate the sensitivities of their designs to variations in parameters or design variables.[1–4] Taguchi[5,6] introduced the concept of parameter design to improve the quality of a product whose manufacturing process involves significant variability. Taguchi's parameter design concept reduces variation in performance by reducing the sensitivity of an engineering design to sources of variation rather than controlling the sources. Taguchi techniques were based on direct experimentation. However, designers often use computer programs to evaluate performance functions instead of actual experiments. d'Entremont and Ragsdell[7] developed a nonlinear code that applies Taguchi's concepts to design optimization.

In robust or statistical optimization, we are interested not only in parameter sensitivities but also in finding the feasible combination of design variables that minimize both the objective function and its sensitivity to design variable variation. There is a need to incorporate the calculation of sensitivity derivatives during the optimization procedure rather than after optimization.

Robust optimum design procedures have been developed and implemented in several studies over the past seven years.[8–20]

One issue that has not been adequately addressed in previous investigations is the cost associated with obtaining sensitivities within robust optimization procedures. Most of the computational research in robust optimization has focused on developing efficient robust objective and robust constraint formulations. Researchers have developed robust formulations that do not require a priori knowledge of probability functions. In addition, the robust formulations avoid integration of the function multiplied by its probability function over the variation region to evaluate function variance. These robust formulations, although efficient at estimating variance, introduce significant costs when used in robust optimization.

The major cost introduced when performing robust optimization is the computational cost of obtaining derivatives (i.e., sensitivities) for optimization. Robust objective functions and robust constraints are typically formulated as compound functions that invoke the original objective or constraint on the order of $(n + 1)$ to $(2n + 1)$ times per evaluation, where n is the number of design variables in the optimization problem. Therefore, using finite differencing techniques (forward or central) to evaluate the gradients of these compound functions requires on the order of $(n + 1)^2$ to $(2n + 1)^2$ original function evaluations.

We are aware of two investigations[18,20] in which the computational costs of calculating sensitivities of the robust design objectives and constraints are addressed. In Ref. 20, the original objective and constraints are approximated by quadratic response surface models. Sensitivities of these response surface models can be calculated analytically by using the coefficients of the response surface model. In Ref. 18, a Plackett–Burman statistical test plan is used to generate a quadratic polynomial model to approximate the constraints. Once the approximation model is generated, the coefficients are used to calculate the first and second derivatives. Although these sensitivities are obtained inexpensively, there is some question about their utility in estimating function variance because the quadratic response surface models may not capture the variation of the original function.

In this research, the automatic differentiation tool ADIFOR[21] is used to reduce the computational expense associated with finite differencing in robust optimization. ADIFOR implements an extensional form of automatic differentiation. That is, it mainly serves as a preprocessor that compiles the original code and develops an enhanced code that can calculate both the original output and the sensitivities.

Sensitivities calculated by automatic differentiation are exact and therefore should enhance performance. To evaluate the utility of using automatic differentiation in robust optimization, two new robust optimization extensions are developed. One is a sensitivity-based procedure, and the other makes use of experimental design techniques.

II. Robust Optimization

In conventional optimization problems, the objective is to minimize/maximize a linear or nonlinear function of many variables subject to a set of constraints. In contrast, robust optimization aims to optimize not only the function value but also its sensitivity due to the variation of the design variables and parameters. Figure 1 illustrates the concepts of robust optimization for an unconstrained design space.

In Fig. 1, a single variable (x) objective function $f(x)$ is used for illustrative purposes. Equation (1) details the objective function formulation used:

$$f(x) = \sum^{9} a_i (x - 900)^{(i-1)} \qquad (1)$$

where $a_1 = -659.23$, $a_2 = 190.22$, $a_3 = -17.802$, $a_4 = 0.82691$, $a_5 = -0.021885$, $a_6 = 0.0003463$, $a_7 = -3.2446 \times 10^{-6}$, $a_8 = 1.6606 \times 10^{-8}$, and $a_9 = -3.5757 \times 10^{-11}$.

Points 1 and 2 are the conventional and robust optimums, respectively. At the conventional optimum (point 1), a variation of ± 10

Received Sept. 16, 1996; revision received Feb. 10, 1997; accepted for publication March 3, 1997. Copyright © 1997 by J. Su and J. E. Renaud. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

*Graduate Research Assistant, Aerospace and Mechanical Engineering.

[†]Assistant Professor, Aerospace and Mechanical Engineering. Member AIAA.

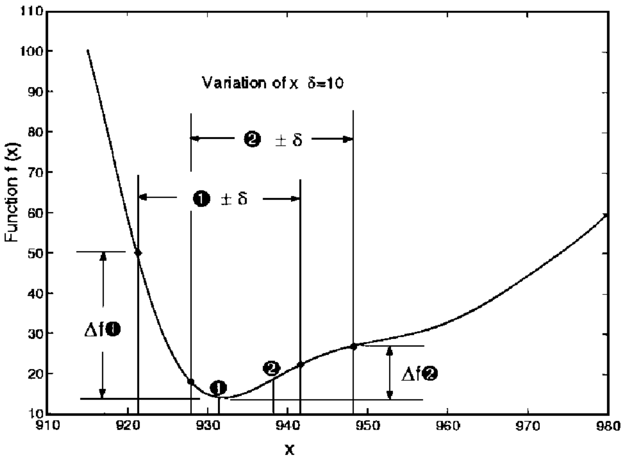


Fig. 1 Robust optimum.

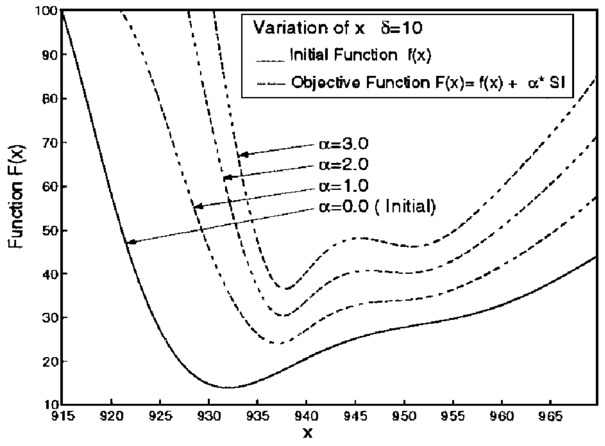


Fig. 2 Robust objective function.

units in the design variable x (approximately 1% variation) results in a range of possible variation for the objective function, depicted as $\Delta f(1)$. At the robust optimum (point 2) a much smaller range for the change in objective $\Delta f(2)$ is observed for the same variation of ± 10 units in the design variable x. It is obvious that the robust optimum is much less sensitive to variations in the design variable x.

Robust optimization strategies consider not only the objective function value but also its variation when the design variables and parameters have fluctuations. Equation (2) defines one of the robust objective functions used in this research. It is developed in this section to illustrate the concept of robust optimization:

$$F(x) = f_c(x) + \alpha \mathrm{SI} \tag{2}$$

where

$$\mathrm{SI} = \sqrt{\frac{1}{ns}\sum^{ns}(f_i - f_c)^2} \tag{3}$$

Here f_c is the current design's (x^c) function value and SI is the sensitivity index, which was defined by Sundaresan et al.[8] SI is calculated from objective function sampling in the variation region near the current design. In our example (Fig. 1), the function values f_i of Eq. (3) are evaluated at sampling points

$$x^i = (x^c \pm 10)$$

located about the current design. In this example, the number of sample points ns equals 2, and α is a weighting factor for SI.

Figure 2 depicts the robust objective function $F(x)$ of Eq. (2) applied to the objective function $f(x)$ of Eq. (1) for different values of α where a variation δ in x of ± 10 is used and two sample points $(x + \delta)$ and $(x - \delta)$ are evaluated for this one-dimensional problem. Note that the robust optimum (2) denoted in Fig. 1 is obtained by using a value of $\alpha = 1.0$ in the robust objective function F.

There is a tradeoff between reducing the variation of the function and minimizing the function value itself. Robust optima are less sensitive to the variation of the design variables, but the function value is larger than that at conventional optima. The engineer must be willing to trade off an increase in the function value for a decrease in variation when moving from the conventional optimum (point 1) to the robust optimum (point 2) in Fig. 1.

III. Literature Review

Significant research in the area of robust optimization has been ongoing for the past seven years. This section provides an overview and comparison of several strategies that are related to those developed in this investigation. Sundaresan et al.[8–10] developed robust optimization methods that account for manufacturing and operational variations. They made use of concepts from statistical design of experiments to approximate the expected value of performance. Ramakrishnan and Rao[12, 13] formulated the robust design problem as a nonlinear optimization problem with Taguchi's loss function as the objective. They used many statistical concepts, such as the expected value of the objective function, the variance of the objective function, and the expected value of constraints. Each of these quantities requires sensitivity information of either the objective function or the constraints to approximate their variations. Mistree et al.[14] developed a robust design methodology within a decision support problem[22] using Monte Carlo simulations with orthogonal arrays. Chen et al.[20] extended this earlier work by integrating the response surface method within the robust compromise decision support problem. Chen et al.[20] developed a fitted response surface model of the objective function. Using this response surface model, they derived a sensitivity-based estimate of variance. The variance was then used as an input to compromise the decision support problem. They suggested use of a response surface instead of an expensive computer simulation for robust optimization. Using this approach, there is always a tradeoff between the number of experiments used and the accuracy of the estimated model. Teng et al.[18] and Emch and Parkinson[17] addressed robust design from a different point of view. They developed a feasibility robustness problem in which they considered the variations of system variables and parameters to be fixed or maximized while the design remains feasible within a specified probability. To consider feasibility robustness, they accounted for the constraint variation by using either a first-order or a second-order sensitivity model based on a statistical or worst-case analysis. Two methods were used to generate the required partial derivatives to calculate the transmitted variations in the constraints. One method uses finite differencing to calculate the first-order and the second-order derivatives. The other method uses a Plackett–Burman statistical test plan to generate a quadratic polynomial model to approximate the constraints. Once the approximation model is generated, the coefficients are used to calculate the first and the second derivatives.

IV. Computational Considerations in Robust Optimization

Robust optimization significantly increases the computational requirements in comparison to traditional optimization. The robust objective functions and robust constraint formulations used in robust optimization are compound functions that require invoking the conventional objective or constraints on the order of $n + 1$ times, where n is the number of design variables. For example, consider a constrained optimization problem that has 10 design variables, each of which has some variation.

If the method of Sundaresan et al.[8–10] is used, assuming an L-16 orthogonal array, then one robust objective function $F(x)$ evaluation will require $16 + 1$ evaluations of the original objective $f(x)$. When applying a conventional gradient-based optimizer for the robust optimization of $F(x)$, the number of original function evaluations $f(x)$ escalates because of the optimizer invoking finite difference calculations to obtain the gradient of $F(x)$. Central differencing of $F(x)$ requires $(20 \times 16) = 320$ original function evaluations $f(x)$ to obtain the gradient information of $F(x)$ for just one iteration of the robust optimizer. Evaluation of the gradients of the robust constraints $g_j^*(\mathbf{x})$ in the method of Sundaresan et al.[8–10] introduces similar costs. The robust constraints $g_j^*(\mathbf{x})$ must be evaluated an additional 20 times to evaluate the gradients of the constraints by central differencing. For each of these evaluations, an additional 20 evaluations of the original constraints $g_j(x)$ are required to estimate by central differencing the sensitivities required in evaluating the robust constraint $g_j^*(x)$. This results in $(20 \times 20) = 400$ original constraint evaluations to obtain gradients of the robust constraints. This can be computationally prohibitive in engineering problems.

If the method of Ramakrishnan and Rao[12,13] is used, the sensitivities of the original function $f(x)$ are required in evaluating the robust objective. Evaluation of the gradient by central differencing requires $20 \times 20 = 400$ original function calls per optimization iteration. Evaluation of the gradients of the robust constraints are even more expensive because second-order sensitivities are required in evaluating the expected value of the constraints.

The robust constraint formulations of Teng et al.[18] introduce similar costs associated with finite differencing. To address this, they use a Plackett–Burman statistical test plan to generate a quadratic polynomial model to approximate the constraints. Once the approximation model is generated, the coefficients are used to calculate the first and second derivatives. They calculate this second-order information only once and invoke several constraint calls to develop the polynomial approximation.

Chen et al.[20] use response surfaces to eliminate the costs associated with obtaining sensitivity information of the original functions within the robust objective and constraint formulations. The use of response surfaces may have a negative impact if the variation of the actual function is lost in the response surface model established by a quadratic least-squares regression. Although sensitivities can be extracted easily from a response surface model, they may be inaccurate. In addition, the method requires that one invoke several function calls to develop the response surface approximation.

The automatic differentiation tool ADIFOR[21] is used in this research to eliminate the computational expense associated with finite differencing in robust optimization. The basic premise of automatic differentiation is that all computations eventually can be broken down into the elementary operations (addition, subtraction, multiplication, and division) and functions (trigonometric, exponential, etc.). Therefore, if all of the basic derivative components of these operations and functions are known, which they readily are, then the chain rule can easily be applied to numerically propagate the derivative through a complicated equation or set of equations. Automatic differentiation provides a very efficient tool for sensitivity calculations in robust optimization.

V. Robust Optimization Extensions

Because robust optimization is a relatively new area of research, there exists a lack of a common language in the existing body of literature. Many concepts currently being used are ambiguous. Some authors define their own concepts or have borrowed names (e.g., statistical objective function, sensitivity index, expected value of the function, variance, feasibility robust, parameter design, tolerance design, etc.) from other research areas. In this research, an attempt has been made to define each term used.

Two new robust optimization extensions are developed for application studies. One is sensitivity based, and the other makes use of experimental design techniques. In each method, an objective function and constraint formulation for robust optimization will be constructed. Each of these objective functions and constraints for robust optimization consists of two parts, the original or conventional function combined with an estimate of the variation of the function. We introduce a weighting factor that allows the user to weigh the two parts according to the problem requirements.

In this research, δ is referred to as the variation of design variables. If the standard deviation of the design variables defined by statistical analysis is σ, then the variation δ of the design variable assumed in robust optimization studies can be set to σ, 2σ, or 3σ depending on the confidence required by the user.

The variation of the objective function $vrt[f(x)]$ and the variation of constraints $\Delta g(x)$ are defined by formulas in the following sections. They are terms related to the common terms of variance and deviation in statistical analysis but are not exactly the same.

The expression "worst-case combination of design variables" refers to designs in which each of the design variables has been perturbed relative to the current design, either as a positive variation $(+\delta)$ or as a negative variation $(-\delta)$. The experimental design-based robust optimization procedure evaluates designs at these worst-case combination sites.

Conventional Optimization

The general formulation for a conventional optimization problem is as follows.

Minimize

$$f(x) \qquad x = [x_1, x_2, \ldots, x_n]^T \tag{4}$$

subject to

$$h_k(x) = 0 \qquad k = 1, \ldots, K \tag{5}$$

$$g_j(x) \geq 0 \qquad j = 1, \ldots, J \tag{6}$$

$$x_i^L \leq x_i \leq x_i^U \qquad i = 1, \ldots, n \tag{7}$$

Note that the equality constraints cannot be satisfied in robust optimization. That is, it is almost impossible for an optimum point to be located where the equality constraint is insensitive to variations. The equality constraints could be relaxed so that robust optimization could be performed. Only inequality constraints are addressed in this research.

Sensitivity-Based Robust Optimization

In this section, a robust optimization strategy is developed. It makes use of sensitivity information, i.e., gradient information for the objective function and constraints. The sensitivity information is used to approximate variations in the objective function and constraints due to variation in the design variables $(x$. Equations (8–14) detail our sensitivity-based robust optimization formulation.

Minimize

$$F(x) = f(x) + \alpha vrt[f(x)] \tag{8}$$

subject to

$$g_j^*(x) = g_j(x) - \Delta g_j \geq 0.0 \qquad j = 1, \ldots, J \tag{9}$$

$$x_{\text{new}}^L \leq x \leq x_{\text{new}}^U \tag{10}$$

where

$$vrt[f(x)] = \sqrt{\sum^n \left(\frac{\partial f}{\partial x_i} \delta_i \right)^2} \tag{11}$$

$$\Delta g_j = \sum^n \left| \frac{\partial g_i}{\partial x_i} \delta_i \right| \tag{12}$$

$$x_{\text{new}_i}^L = x_i + \delta_i \tag{13}$$

$$x_{\text{new}_i}^U = x_i - \delta_i \tag{14}$$

The robust objective [Eq. (8)] developed here considers both the function and its variation. It is an approximation of standard deviation of the objective function. Equation (11) assumes that the function value does not change severely in the deviation range $(\pm\delta)$ of the design, so that the first-order Taylor series terms at the current point provide a good approximation of the variation. The weight α can be adjusted by the user in the robust optimization procedure. The term α allows the user to trade off a reduction in $f(x)$ relative to a reduction in the variation. In well-scaled optimization problems, a value of $\alpha = 1$ has worked well.

Equation (9) considers constraint variation by a formulation similar to the methods of Teng et al.[18] and Emch and Parkinson.[17] The worse-case variation of the constraints is approximated in Eq. (12). These new constraints account for the effect of the variation of the design variables and parameters. Adding variation to constraints has the effect of reducing the size of the feasible region. The optimum design is moved into the reduced feasible region so that the design will stay feasible when subjected to variation in design variables or parameters.

To account for variations that lead to violations of the upper and lower variable bounds of the original optimization problem, the upper and lower bounds of the design variables are shifted by the amount of the worst-case variation. This ensures that all variations are within the original problem bounds [Eqs. (13) and (14)].

Experimental Design-Based Robust Optimization

In this section, an experimental design-based robust optimization strategy is developed. This strategy makes use of Taguchi's orthogonal arrays to specify an experimental design plan for evaluating the variation of the objective function and constraints. A penalty function formulation [Eq. (17)] is adopted in this method to avoid the undesired overhead associated with robust constraint gradient calls as in the method of Sundaresan et al.[8–10]

Minimize

$$F(x) = f_p(x) + \alpha vrt[f_p(x)] \tag{15}$$

subject to

$$x_{\text{new}}^L \leq x \leq x_{\text{new}}^U \tag{16}$$

where

$$f_p(x) = f(x) + \Omega[R, g(x)] \tag{17}$$

$$vrt[f_p(x)] = \sqrt{\frac{1}{ns}\sum_{i=1}^{ns}(f_{p_i} - f_{p_c})^2} \tag{18}$$

$$f_{p_i} = f_p(x + \delta_i) \tag{19}$$

$$f_{p_c} = f_p(x) \tag{20}$$

$$x_{\text{new}_i}^L = x_i + \delta_i \tag{21}$$

$$x_{\text{new}_i}^U = x_i - \delta_i \tag{22}$$

$$\Omega = R[\hat{g}(x)]^2 \tag{23}$$

$$\hat{g}(x) = \begin{pmatrix} g(x) & \text{if} \quad g(x) < 0.0 \\ 0 & \text{if} \quad g(x) \geq 0.0 \end{pmatrix} \tag{24}$$

$$g_j(x) \geq 0 \qquad j = 1, \ldots, J \tag{25}$$

The robust objective function [Eq. (15)] has two parts, the function and its variation. The variation $vrt[f_p(x)]$ in this formulation is similar to the SI in the method of Sundaresan et al.[8–10] It is defined as the root-mean-square value of the difference between the value of the sample points and the function value at the current point. Note that there are no explicit constraints used in this formulation. Constraints are considered in Eq. (17) as penalty terms. This formulation simultaneously accounts for objective and constraint variations.

The sample designs i in Eqs. (18) and (19) are selected by using an orthogonal array to determine the combination of design variables for fractional factorial experiments. These sampling points are composed of combinations of the worst-case variation of the design variables. Their use in Eqs. (17) and (18) provides an estimate of the variation of the penalty function value f_{p_i} in the worst-case region of variation.

The variable boundaries are treated the same as in the sensitivity-based method. The constraints are not considered formally in this robust strategy as they are already accounted for in the penalty function. In this method, variations in constraints are considered simultaneously with variation in the objective function through use of the familiar penalty function approach[23] for optimization.

VI. Automatic Differentiation Implementation Details

In this research, ADIFOR is applied successfully for robust optimization in both the experimental design and the sensitivity-based procedures.

In the experimental design-based method [Eqs. (8–14)], ADIFOR is used to process the source code for $F(x)$ to obtain the new source code, which can calculate both the function value $F(x)$ and its gradient $\nabla F(x)$ simultaneously. These gradients can be used directly by the optimizer and therefore the costly gradient calculations associated with finite differencing of $F(x)$ are avoided.

In the sensitivity-based method, ADIFOR has been used twice. First ADIFOR is used to compile the source code, which calculates the original function $f(x)$ and constraints $g(x)$, to obtain a new code, which can calculate those terms plus their gradients $\nabla f(x)$ and $\nabla g(x)$. Note that these terms are required to evaluate the sensitivity-based robust objective and robust constraint formulations [Eqs. (8) and (9)].

This code serves as the basis for the programs that calculate $F(x)$ [Eq. (8)] and $g*(x)$ [Eq. (9)]. Recall that the optimizer requires gradients for $F(x)$ and $g*(x)$. Therefore, ADIFOR is used again to compile the $F(x)$ and $g*(x)$ source code to obtain a new code that calculates those terms and their gradients $\nabla F(x)$ and $\nabla g*(x)$. This avoids all of the finite differencing that would be required when not using automatic differentiation.

In this research, automatic differentiation technology is used to obtain sensitivity information for robust optimization. Both the experimental design-based robust optimization strategy and the sensitivity-based procedure make use of automatic differentiation in application to a multidisciplinary optimization test problem.

VII. Implementation Studies

The two new robust optimization extensions developed in this research are implemented in application to a multidisciplinary design optimization test problem. A generalized reduced gradient (GRG) optimizer is used to solve the robust optimization problems.

Aircraft Concept Sizing Test Problem

The test problem selected for this study involves preliminary sizing of a general aviation aircraft (for complete details, see Ref. 24). The design vector for this aircraft concept sizing problem is composed of variables relating to the geometry of the aircraft, propulsion and aerodynamic characteristics, and flight regime. In all, 10 design variables exist for the test problem: x_1 = aspect ratio of the wing, x_2 = wing area, $x_3 = C_{L_{\max}}$ (maximum lift coefficient of the wing), x_4 = propeller efficiency, x_5 = specific fuel consumption, x_6 = fuselage length, x_7 = fuselage diameter, x_8 = density of air at cruise altitude, x_9 = cruise speed, and x_{10} = fuel weight.

Six system states are calculated for this design problem as illustrated in Fig. 3. These states are as follows: y_1 = maximum lift-to-drag ratio, y_2 = total aircraft wetted area, y_3 = empty weight, y_4 = gross takeoff weight, y_5 = stall speed, and y_6 = aircraft range.

As illustrated in the dependency diagram of Fig. 4, five feed-forward couplings and one feedback coupling exist among the

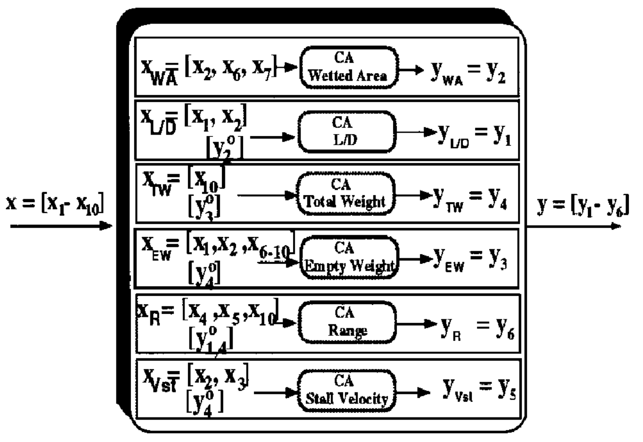


Fig. 3 System analysis module.

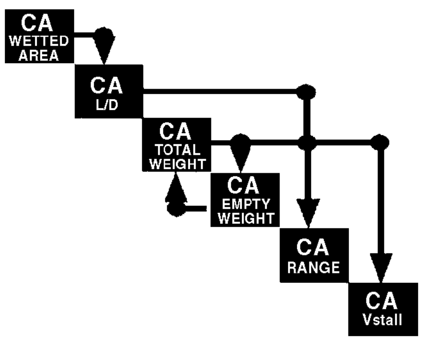


Fig. 4 Dependency diagram.

contributing analyses (CAs) for this problem. The feedback coupling is due to the dependence of empty weight on total weight, and iteration among these CAs is necessary to converge to a consistent set of system states.

In implementation, the system analysis is carried out through execution of Fortran codes applying the empiricism for this class of aircraft as discussed above. A tolerance of 0.01% was set on the weight iteration between the empty and total weight CAs.

The objective in this problem is to determine the least gross takeoff weight design within the bounded design space subject to two performance constraints. The first constraint is that the stall speed must be less than or equal to some prescribed maximum stall speed, and the second constraint is that the aircraft range must be greater than a specified requirement. In standard form, the system optimization problem is given in Eqs. (26–29).

Minimize

$$f(x) = \text{weight} = y_4 \tag{26}$$

subject to

$$g_1 = 1 - \frac{y_5}{V_{\text{stall}_{\text{req}}}} \geq 0.0 \tag{27}$$

$$g_2 = 1 - \frac{\text{range}_{\text{req}}}{y_6} \geq 0.0 \tag{28}$$

$$x^L \leq x \leq x^U \tag{29}$$

where

$$V_{\text{stall}_{\text{req}}} = 70\,\text{ft/s} \qquad \text{range}_{\text{req}} = 565\,\text{miles}$$

Sensitivity-Based Robust Optimization

In the sensitivity-based method [Eqs. (8–14)], ADIFOR can be used twice as discussed earlier. Table 1 compares the number of function evaluations, the number of constraint evaluations, the number of system analyses, and the CPU time required for three different implementations of the sensitivity-based robust optimization procedure. Each procedure converges to the same robust minimum. Procedure 0 uses central differentiation for calculation of all the gradient information, including $\nabla f(x)$, $\nabla g(x)$, $\nabla F(x)$, and $\nabla g^*(x)$. Procedure 1 uses ADIFOR once to obtain $\nabla f(x)$ and $\nabla g(x)$ and then uses central differencing to calculate $\nabla F(x)$ and $\nabla g^*(x)$ for the optimizer. Procedure 2 uses ADIFOR twice to calculate all of the gradient information, including $\nabla f(x)$, $\nabla g(x)$, $\nabla F(x)$, and $\nabla g^*(x)$. The number of system analyses refers to evaluations of the coupled system as depicted in Figs. 3 and 4.

Table 1 details the significant savings obtained when using automatic differentiation (ADIFOR) for sensitivity calculations in sensitivity-based robust optimization. Using traditional central differentiation on the robust objective function and constraints requires 7056 system analyses. Using ADIFOR once requires only 336 system calls and using ADIFOR twice reduces the number of system analyses to 244. The reductions in number of system analyses required for robust optimization are 95 and 97%, respectively. There is some CPU cost associated with using ADIFOR for the gradient calculations. Although the number of the system analyses decreases on the order of 95–97%, the reductions of CPU time are observed to be 81 and 85%, respectively.

Figure 5 details the computational history of these three procedures. Significant savings in the number of system analyses required and CPU time required for robust optimization are obtained by ADIFOR.

Figure 6 illustrates the convergence history of the robust objective function $F(x)$, the original objective function $f(x)$, and the variation of the function $vrt[f(x)]$ vs the number of robust function evaluations within the robust optimizer (ADIFOR twice). In the robust optimization procedure, both the original function $f(x)$ and its variation are observed to decrease. The convergence histories are similar when using ADIFOR once or central differencing.

Table 1 ADIFOR comparison[a]

Procedure	nfe	nce	nsa	CPU, s
CD 0	146	336	7056	21.92
ADIFOR 1	146	336	336	4.08
ADIFOR 2	67	239	244	3.29

[a]nfe, no. of function evaluations; nce, no. of constraint evaluations; nsa, no. of system analyses; CD, central differentiation.

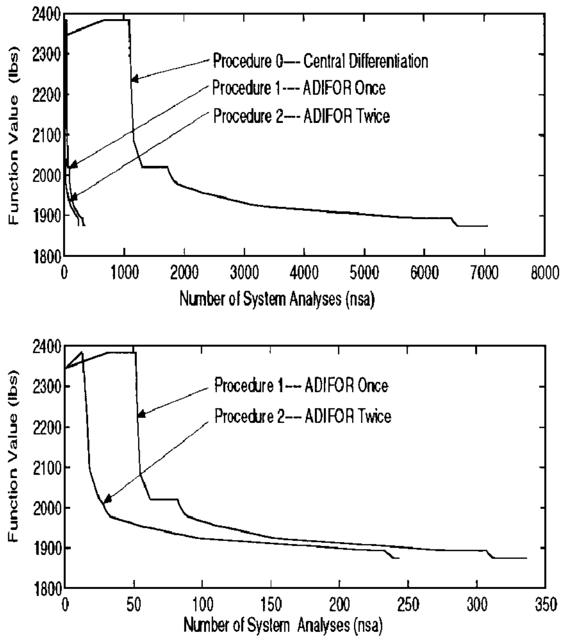


Fig. 5 ADIFOR performance.

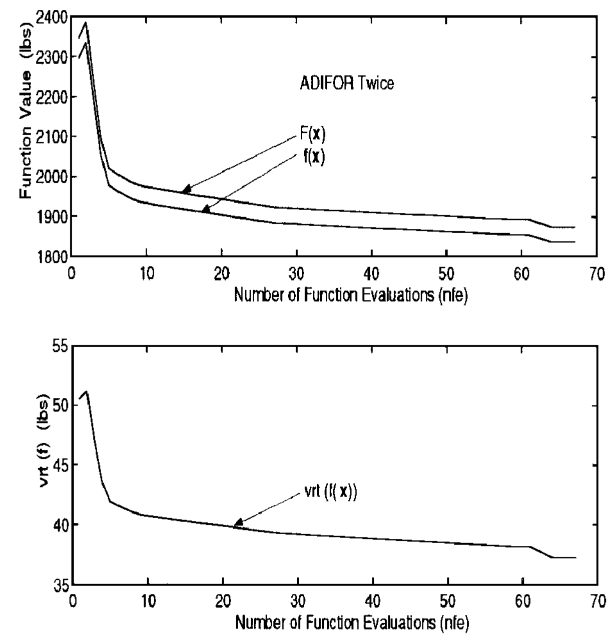


Fig. 6 Objective convergence history.

Figure 7 illustrates the convergence history of the constraints vs the number of constraint evaluations within the robust optimizer (ADIFOR twice).

Recall that g_1 and g_2 are the conventional constraint formulations, and g_1^* and g_2^* are the robust formulations. Note that g_2 has been pushed away from being active to make violation of g_2 insensitive to design variable variation. Constraint g_1 is less sensitive to design variable variation and remains nearly active throughout the optimization. The convergence histories are similar when using ADIFOR once or central differencing.

The oscillation of the constraints in Fig. 7 is attributed to the GRG optimization procedure. Newton updates of active constraints within the GRG method are oscillatory in nature.

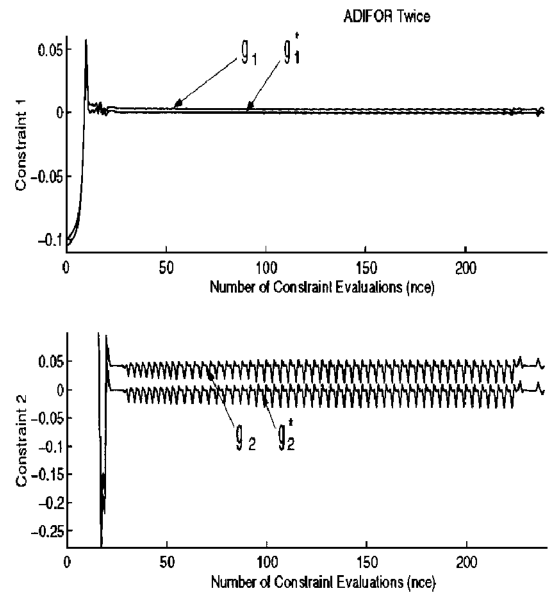


Fig. 7 Constraint convergence history.

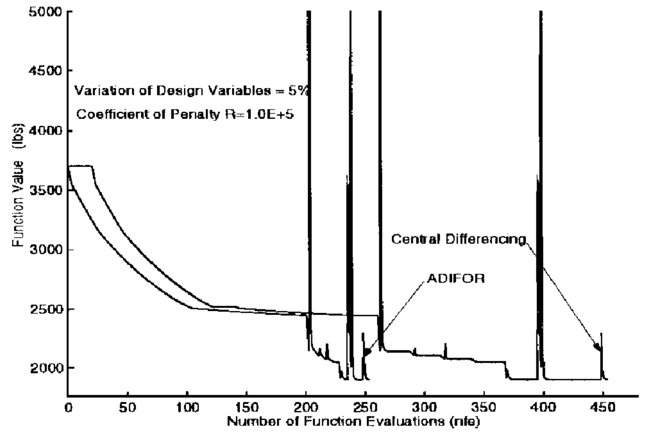


Fig. 8 Penalty function convergence.

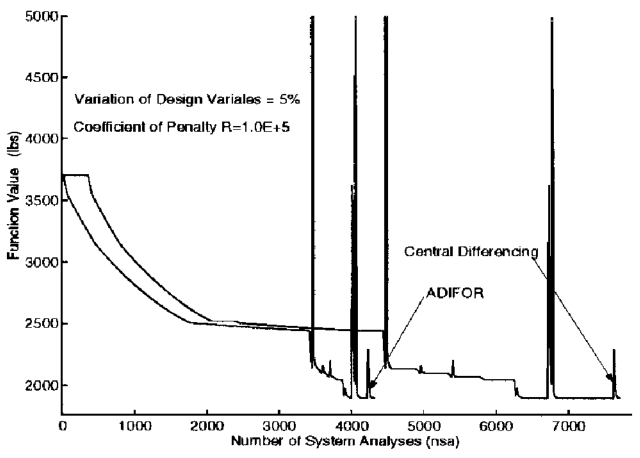


Fig. 9 Penalty function convergence.

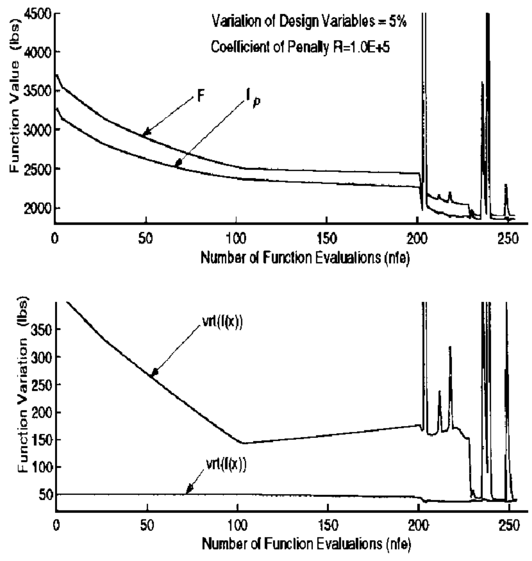


Fig. 10 Penalty and variation history.

Experimental Design-Based Robust Optimization

In the experimental design-based method [Eqs. (15–25)], ADIFOR is used to process the code for the robust objective $F(x)$ [Eq. (15)] to obtain a new source code that calculates both the function value $F(x)$ and its gradient $\nabla F(x)$ simultaneously.

This strategy makes use of Taguchi's orthogonal arrays in developing an approximation for variation in the objective function. A penalty function formulation [Eq. (17)] is adopted for use in this method to avoid the unwanted overhead associated with robust constraint gradient calls as in the method of Sundaresan et al.[8–10] A penalty coefficient, R, equal to $1.0E + 5$ is used in this trial.

Figure 8 details the significant savings obtained when using automatic differentiation (ADIFOR) for gradient calculations in experimental design-based robust optimization.

Using traditional central differencing on the robust objective function $F(x)$ requires the optimizer to make 453 objective function $F(x)$ calls, whereas using ADIFOR requires only 253 function $F(x)$ calls. This reduction is expected because each trial required 10 GRG iterations for convergence. Each GRG iteration invokes a gradient calculation for $\nabla F(x)$. The elimination of the central differencing function evaluations (20 per iteration) for 10 GRG iterations results in a savings of 200 function evaluations when using automatic differentiation.

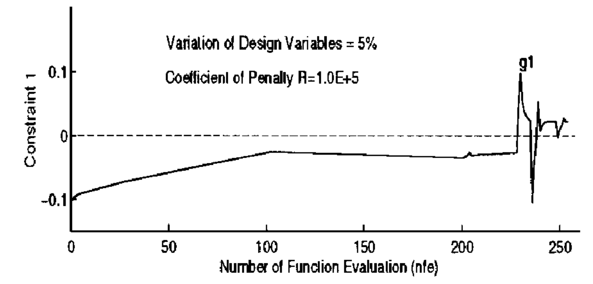
This reduction in robust function evaluations is even greater when one considers that each robust objective function call requires 16 addition evaluations of the original objective (i.e., system analyses). There are several big spikes that occur after 200 function evaluations in Fig. 8. These occur because some of the fractional factorial sampling points and the current design are infeasible. The large value of the penalty term increases the robust objective function value dramatically. To keep these plots in a proper scale, the spikes have been truncated.

Figure 9 compares the number of system analyses required for experimental design-based robust optimization. Figure 10 details the convergence history of the robust objective function $F(x)$, the penalty function $f_p(x)$, and the variation of the penalty function $vrt[f_p(x)]$ vs the number of robust function evaluations in the optimizer. To illustrate the influence of the penalty terms, the variation of the original function $vrt[f(x)]$ is also calculated and shown in this plot.

The impact of the penalty term can be observed in the convergence history of the variation of the penalty function, where positive spikes indicate constraint violations at the fractional factorial experimental sites evaluated in Eqs. (18) and (19) or the current design [Eq. (20)]. The robust optimizer acts to push the design back into the feasible region. The results in this plot are consistent with the constraint results shown in Fig. 11.

The difference between the variation of the original function $vrt[f(x)]$ and variation of the penalty function $vrt[f_p(x)]$ highlights the effect of the penalty term. The variation of the original function $vrt[f(x)]$ decreases 24% during the optimization procedure.

Figure 11 details the constraint histories for this problem vs the number of robust objective function evaluations. Both constraints are driven away from being active at the robust optimum design. The use of the penalty formulation pushes constraints away from



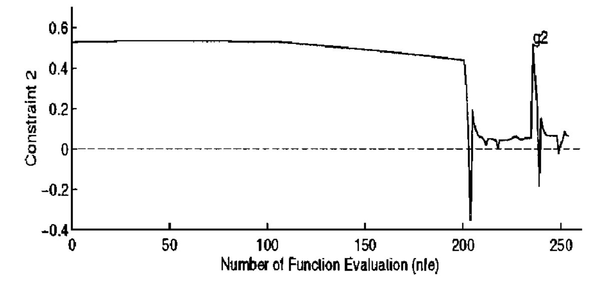


Fig. 11 Constraint convergence history.

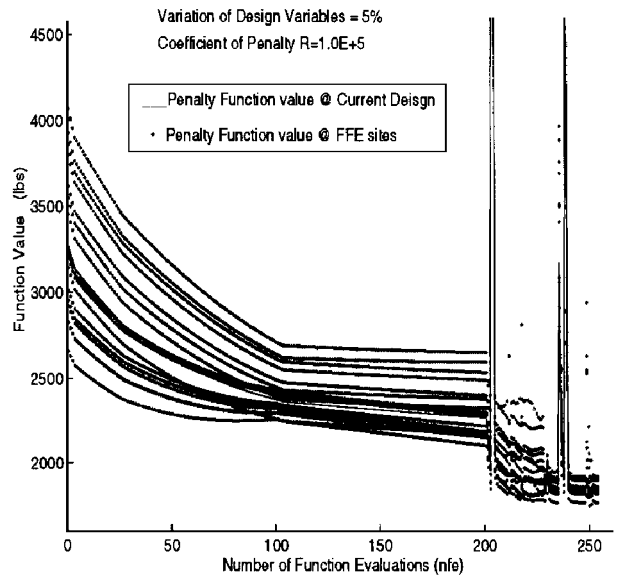


Fig. 12 Fractional factorial (L-16) evaluations.

being active to make violation of that constraint insensitive to design variable variation. Future studies will be directed toward developing a strategy for actively setting the penalty coefficient R to ensure robust constraint feasibility. In this study, a value of R equal to $1.0E + 5$ is used and is sufficiently large to ensure proper convergence.

Figure 12 details the convergence history of the objective function $f_p(x)$ at the current design and the individual penalty values $f_{p_i}(x)$ $(i = 1, \ldots, 16)$ at each of the L-16 fractional factorial experimental sites for each robust objective calculation. Remember that each robust objective calculation [Eqs. (15) and (18)] requires calculation of 16 penalty functions [Eq. (17)] at the fractional factorial experimental design sites.

Figure 12 shows not only the function value at the current design but also the function value at the worst-case sample points during the robust optimization procedure. Except for a few spikes, the range of the distribution of the functions at the sample points is observed to decrease during the optimization procedure. Figure 12 provides a good picture that visually verifies the goals of experimental design-based robust optimization.

Note that Figs. 10–12 detail convergence histories for the experimental design-based robust optimization procedure where ADIFOR has been used for gradient calculations. Similar trends would be observed when using central differencing for gradient calculation.

VIII. Summary and Conclusions

Most of the computational research in robust optimization has focused on developing efficient robust objective and robust constraint formulations. These robust formulations, although efficient at estimating variance, introduce significant costs when used for robust optimization. The major cost introduced when robust optimization is performed is the computational cost of obtaining derivatives (i.e., sensitivities) for optimization.

We are aware of two investigations[18,20] in which the computational costs of calculating sensitivities of the robust design objectives and constraints are addressed. In those studies, the original objective and constraints were approximated by using quadratic response surface models[20] and quadratic polynomials.[18] Sensitivities of these approximate models were calculated analytically by using the coefficients of the approximate models. Although these sensitivities are obtained inexpensively, there is some question about their utility in calculating function variance because the quadratic approximations may not capture the variation of the original functions.

In this research, automatic differentiation techniques are used to avoid the costly finite differencing of robust objective functions and robust constraints. Sensitivities calculated by automatic differentiation are exact and therefore enhance performance. To evaluate the utility of using automatic differentiation in robust optimization, two new robust optimization extensions are developed. One is a sensitivity-based procedure, and the other makes use of experimental design techniques. These two new robust optimization extensions are implemented in application to an aircraft concept sizing test problem.

Automatic differentiation greatly increases the efficiency of robust optimization strategies. Using automatic differentiation in the sensitivity-based method, one observes a 95–97% reduction in the number of system analysis required for robust optimization and an 81–85% reduction in the CPU time required when compared to central differencing for gradient calculations. Using the experimental design-based method, there is a 44% reduction in the number of system analysis and a 21% reduction in the CPU time compared with central differencing. The savings in CPU time is less than the savings in system analyses for both methods because of the computational overhead incurred when automatic differentiation is used.

The sensitivity-based strategy is observed to be significantly faster when using either ADIFOR once or ADIFOR twice compared to the experimental design-based strategy. This is to be expected because the use of ADIFOR once in the sensitivity-based method to calculate the sensitivities of the original objective $f(x)$ and constraints $g(x)$ does not have an equivalent in the experimental design-based procedure. The experimental design-based strategy calculates variation in the robust objective by evaluating the robust objective at several (ns) locations in the vicinity of the current design. This cost cannot be removed by automatic differentiation. Therefore, the sensitivity-based strategy is the preferred method for robust optimization in situations where the variation can be reasonably estimated with exact derivatives.

The two robust optimization extensions developed in this research are variations of previous formulations and have been designed to eliminate cited drawbacks. The penalty function used in the experimental design-based method eliminates redundant calculations of the constraint gradients. The sensitivity-based method combines the merits of previous work while eliminating shortcomings. Using the two strategies in application to the aircraft concept sizing problem, the design variables are pushed away from their bounds and the constraints are restricted from being active. Convergence histories of the function, constraints and variation confirm that the two robust optimization extensions are effective in decreasing the variation of the function while maintaining feasibility robustness.

Acknowledgments

This multidisciplinary research effort was supported in part by NASA Research Grant Award NAG-1-1561, National Science

Foundation Grants DMI93-08083 and DMI94-57179, and the Ford Motor Company.

References

[1]Fiacco, A. V., *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*, Academic, New York, 1991.

[2]Sobieski, J., Barthelemy, J., and Riley, K. M., "Sensitivity of Optimum Solutions of Problem Parameters," *AIAA Journal*, Vol. 20, No. 9, 1982, pp. 1291–1299.

[3]Beltracchi, T. J., and Gabriel, G. A., "A RQP Based Method for Estimating Parameter Sensitivity Derivatives," *Proceedings of Design Automation Conference*, DE Vol. 14, American Society of Mechanical Engineers, New York, 1988, pp. 155–164.

[4]Beltracchi, T. J., and Gabriel, G. A., "Observations on Extrapolations Using Parameter Sensitivity Derivatives," *Proceedings of Design Automation Conference*, DE Vol. 14, American Society of Mechanical Engineers, New York, 1988, pp. 165–173.

[5]Taguchi, G., "Off-Line and On-Line Quality Control Systems," *Proceedings of the International Conference on Quality Control* (Tokyo, Japan), 1978.

[6]Taguchi, G., *System of Experimental Design*, edited by D. Clausing, American Supplier Inst., Dearborn, MI, 1987.

[7]d'Entremont, K. L., and Ragsdell, K. M., "Design for Latitude Using TOPT," *Advances in Design Automation*, DE Vol. 14, American Society of Mechanical Engineers, New York, 1988, pp. 265–272.

[8]Sundaresan, S., Ishii, K., and Houser, D. R., "A Procedure Using Manufacturing Variance to Design Gears with Minimum Transmission Error," *Advances in Design Automation, Design Automation and Design Optimization*, DE Vol. 19-2, American Society of Mechanical Engineers, New York, 1989, pp. 145–151.

[9]Sundaresan, S., Ishii, K., and Houser, D. R., "Design Optimization for Robustness Using Performance Simulation Programs," *Advances in Design Automation, Design Automation and Design Optimization*, DE Vol. 32-1, American Society of Mechanical Engineers, New York, 1991, pp. 249–256.

[10]Sundaresan, S., Ishii, K., and Houser, D. R., "A Robust Optimization Procedure with Variations on Design Variables and Constraints," *Advances in Design Automation*, DE Vol. 65-1, American Society of Mechanical Engineers, New York, 1993, pp. 379–386.

[11]Otto, K. N., and Antonsson, E. K., "Extensions to the Taguchi Method of Product Design," *Design Theory and Methodology*, DE Vol. 31, American Society of Mechanical Engineers, New York, 1991, pp. 21–30.

[12]Ramakrishnan, B., and Rao, S. S., "A Robust Optimization Approach Using Taguchi's Loss Function for Solving Nonlinear Optimization Problems," *Advances in Design Automation*, DE Vol. 32-1, American Society of Mechanical Engineers, New York, 1991, pp. 241–248.

[13]Ramakrishnan, B., and Rao, S. S., "Efficient Strategies for the Robust Optimization of Large Scale Nonlinear Design Problems," *Advances in Design Automation*, DE Vol. 69-2, American Society of Mechanical Engineers, New York, 1994, pp. 25–35.

[14]Mistree, F., Lautenschalager, U., and Erikstad, S. O., "Simulation Relation Using the Taguchi Method," NASA CR 4542, 1993.

[15]Yu, J.-C., and Ishii, K., "A Robust Method for Systems with Significant Nonlinear Effects," *Advances in Design Automation*, DE Vol. 65-1, American Society of Mechanical Engineers, New York, 1993, pp. 379–386.

[16]Cagan, J., and Williams, B. C., "First-Order Necessary Conditions for Robust Optimality," *Advances in Design Automation*, DE Vol. 65-1, American Society of Mechanical Engineers, New York, 1993, pp. 539–549.

[17]Emch, G., and Parkinson, A., "Using Engineering Models to Control Variability: Feasibility Robustness for Worst-Case Tolerances," *Advances in Design Automation*, DE Vol. 65-1, American Society of Mechanical Engineers, New York, 1993, pp. 411–418.

[18]Teng, A., Free, J., and Parkinson, A., "An Approach to Robust Design of Dynamic Systems—Allocation of Variation Robust Performance Under Parametric Uncertainties," *Proceedings of the AIAA/USAF/NASA/OAI 4th Symposium on Multidisciplinary Analysis and Optimization*, AIAA, Washington, DC, 1992, pp. 810–819 (AIAA Paper 92-4700).

[19]Yurkovich, R., "The Use of Taguchi Techniques with The ASTRO Code for Optimum Wing Structural Design," *Proceedings of the AIAA/ASME/ASCE/AHS/ASC 35th Structures, Structural Dynamics, and Materials Conference*, AIAA, Washington, DC, 1994, pp. 1334–1342 (AIAA Paper 94-1484).

[20]Chen, W., Tsui, K., Allen, J. K., and Mistree, F., "Integration of the Response Surface Methodology with the Compromise Decision Support Problem in Developing a General Robust Design Procedure," *Advances in Design Automation*, DE Vol. 82, American Society of Mechanical Engineers, New York, 1995, pp. 485–492.

[21]Bischof, C., "ADIFOR Working Note 9: Getting Started with ADIFOR," ANL/MCS-TM-164, Argonne National Lab., Argonne, IL, June 1993.

[22]Mistree, F., Hughes, O. F., and Bras, B. A., "The Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm," *Structure Optimization: Status and Promise*, edited by M. P. Kamat, Vol. 150, Progress in Astronautics and Aeronautics, AIAA, Washington, DC, 1993, pp. 247–289.

[23]Reklatis, G. V., Ravindran, A., and Ragsdell, K. M., *Engineering Optimization Method and Applications*, Wiley-Interscience, New York, 1983, pp. 216–260.

[24]Wujek, B. A., and Renaud, J. E., "Design Flow Management and Multidisciplinary Design Optimization in Application to Aircraft Concept Sizing," AIAA Paper 96-0713, Jan. 1996.

A. D. Belegundu
Associate Editor