JOURNAL OF AIRCRAFT
Vol. 46, No. 2, March–April 2009

Robust Aerodynamic Design Optimization Using Polynomial Chaos

Michael Dodson* and Geoffrey T. Parks†
University of Cambridge, Cambridge, England CB2 1PZ United Kingdom

DOI: 10.2514/1.39419

This paper investigates the potential of polynomial chaos methods, when used in conjunction with computational fluid dynamics, to quantify the effects of uncertainty in the computational aerodynamic design process. The technique is shown to be an efficient and accurate means of simulating the inherent uncertainty and variability in manufacturing and flow conditions and thus can provide the basis for computationally feasible robust optimization with computational fluid dynamics. This paper presents polynomial chaos theory and the nonintrusive spectral projection implementation, using this to demonstrate polynomial chaos as a basis for robust optimization, focusing on the problem of maximizing the lift-to-drag ratio of a two-dimensional airfoil while minimizing its sensitivity to uncertainty in the leading-edge thickness. The results demonstrate that the robustly optimized designs are significantly less sensitive to input variation, compared with nonrobustly optimized airfoils. The results also indicate that the inherent geometric uncertainty could degrade the on-design as well as the offdesign performance of the nonrobust airfoil. This leads to the further conclusion that the global optimum for some design problems is unreachable without accounting for uncertainty.

Nomenclature

a^d	=	deterministic realization of process a
CV	=	number of control variables
L/D	=	lift-to-drag ratio
n	=	number of random variables
p	=	polynomial chaos order
ζ	=	general random variable
μ	=	mean (Gaussian)
$\boldsymbol{\xi}$	=	Gaussian random variable
σ	=	standard deviation (Gaussian)
Φ	=	Askey polynomial
Ψ	=	Hermite polynomial

I. Introduction

COMPUTATIONAL fluid dynamics (CFD) has evolved considerably since emerging in the 1960s, tracking improvements in computational hardware and algorithm development. CFD produces results faster and is generally less expensive than wind-tunnel testing. Advanced codes can approach the accuracy of wind-tunnel experiments [1]. However, because deterministic CFD only produces a single solution for a given set of input parameters [2], it cannot model the uncertainty and variation inherent in real-world problems [3].

Many approaches to classify and quantify uncertainty in simulations have been developed [3]. The most accurate of these is Monte Carlo simulation (MCS), but this often requires thousands of deterministic simulations to obtain uncertainty statistics, which is infeasible for computationally expensive CFD simulations. Recently, polynomial chaos (PC) methods have emerged as promising candidates for uncertainty quantification, showing accuracy comparable with MCS at a fraction of the cost [3,4].

Received 29 June 2008; revision received 9 November 2008; accepted for publication 16 November 2008.

*M.Phil. Candidate, Engineering Design Centre, Department of Engineering. Student Member AIAA.

†University Senior Lecturer, Engineering Design Centre, Department of Engineering.

Uncertainty quantification is also a growing concern in the design optimization field. Optimized designs are often sensitive to small variations in manufacturing or operating conditions [5], which can lead to inferior performance in real-world conditions [6]. The solution is to include sensitivity to uncertainty in the optimization: that is, minimize the design's sensitivity while optimizing some other observable(s). This inherently requires uncertainty quantification during each iteration of the optimization. Unfortunately, accurate methods for uncertainty quantification in high-fidelity CFD simulations (namely, MCS) are generally considered to be computationally infeasible for use within already computationally expensive optimization processes. As a result, designers often use lower-order surrogate models for robust optimization, decreasing the overall accuracy of the simulation and thus of the optimization. The desire for computationally feasible, high-fidelity, robust optimization is the impetus for the primary aim of this work: to demonstrate that PC methods are capable and efficient means of uncertainty quantification for CFD used in robust optimization.

II. Background

Many attempts have been made to classify error and uncertainty. Walters and Huyse [3] described error as deterministic: a recognizable deficiency not due to lack of knowledge, which is theoretically correctable. Conversely, uncertainty is stochastic: a potential deficiency due to lack of knowledge, which therefore cannot be eliminated. Oberkampf et al. [7] further divided uncertainty into aleatoric uncertainty, which is intrinsically variable and often represented by probability density functions, and epistemic uncertainty, which arises from a deliberate simplification or lack of understanding, for example, ignoring coupling effects to simplify models. Oberkampf et al. [8] provided means of detection and quantification of several sources of uncertainty and error, focusing on physical modeling and discretization errors. Walters and Huyse [3] further divided the physical modeling category, highlighting the importance of uncertainty in the geometry and turbulence models.

The difference between uncertainty and error is often blurred in the literature, but their adverse effect on CFD solutions is clear [3]. The need for validation and uncertainty quantification was highlighted by the 2001 AIAA CFD Drag Prediction Workshop (DPW) [9], which showed discrepancies among CFD solutions and between CFD and experimental results [10]. Subsequently, discussion of uncertainty quantification methods has become more prevalent in the literature. Hosder et al. [2] discussed the DPW results and described a method

to quantify uncertainty from several sources, including discretization error, geometry representation, turbulence model, and boundary conditions, as well as the interaction between these sources. Walters and Huyse [3] published an overview of the uncertainty analysis field for CFD users, explaining several methods, including interval analysis, MCS, moment methods, and PC, and used examples to highlight the benefits and drawbacks of each. However, Karniadakis et al. [4] showed most of them to be insufficient for computationally feasible, accurate uncertainty quantification.

Uncertainty quantification is also a growing concern in the design optimization field. Huyse et al. [5] described the sensitivity of optimized designs to small manufacturing errors or fluctuations in operating conditions. The former can be alleviated in the short term with tighter manufacturing tolerances, but this cannot reduce variations in operating conditions and cannot prevent inevitable geometric variations induced by standard operation, such as erosion or creep. Deterministic optimization with discrete operating conditions (i.e., ignoring uncertainty) can result in dramatically inferior performance under actual operating conditions. Garzon and Darmofal [6] highlighted this, studying the effect of geometric variability on axial compressor performance and showing up to a 1% decrease in efficiency due to manufacturing deviations from the optimal blade design. The alternative is robust optimization: creating designs that are insensitive to variation. Zang et al. [11] presented this as a multiobjective optimization problem: optimizing the mean performance while minimizing the response variation. They used Taguchi-based methods to show that performance improvements from robust optimization were possible in structural dynamics. Kumar et al. [12] used genetic algorithms and surrogate modeling to design an optimal compressor blade robust to erosion. Huyse et al. [5] compared and contrasted deterministic and nondeterministic optimization approaches to robust airfoil optimization, focusing on expected value and moment methods, and showed that robust methods overcame the inherent pitfalls of deterministic design optimization. Molina-Cristobal et al. [13] demonstrated that PC could provide the foundation of robust optimization, designing an optimized resistor–inductor–capacitor circuit that is robust to variations in the input voltage and resistance.

Polynomial chaos has received much attention recently as a technique for uncertainty analysis [3,4]. The method is attractive because it produces high-order response information and is computationally inexpensive compared with techniques with similar capabilities, such as MCS [13]. PC is rooted in the homogeneous chaos of Wiener [14], defined as a span of Hermite polynomial functionals of a Gaussian process. Cameron and Martin [15] showed that the homogeneous chaos expansion converged to any L_2 functional in the L_2 sense; that is, it exactly represents any process with a finite second-order moment.

The most widely used PC implementation was developed by Ghanem and Spanos [16] for uncertainty quantification in solid mechanics. They converted Wiener's [14] integral formulation to a discrete infinite summation. Truncating the summation and applying a Galerkin projection creates a system of deterministic equations, the solution of which is a set of coefficients defining the stochastic behavior of a random process to arbitrary order. Ghanem [17,18] expanded on the theme, developing a spectral, stochastic, finite element method. PC was first applied to fluid problems by Xiu et al. [19], who used PC with the incompressible Navier–Stokes equations to quantify uncertainty in fluid–structure interactions, demonstrating that their model was as accurate as MCS for quantifying uncertainty and was thousands of times less computationally expensive.

The original PC paired Hermite polynomials with Gaussian random variables. Xiu and Karniadakis [20] generalized PC by proving that the same exponential convergence behavior of the Cameron–Martin theorem [15] could be achieved by pairing polynomials from the Askey scheme with different probability distributions. In [21], they applied this generalized polynomial chaos (GPC) to steady-state diffusion problems and to pressure-driven channel flow in [22]. Others have applied PC to the full Navier–Stokes equations [23,24]. Mathelin et al. [25] extended this work to include turbulence model uncertainty. PC has also been applied to model geometric uncertainty [10,26]. Perez and Walters [27] formulated a stochastic Euler solver and applied it to several supersonic problems involving geometric uncertainty.

Several drawbacks of PC have been noted, however. The most apparent is the increased cost associated with dimensionality and nonlinearity [20,25,28–31]. Deviation from a basic probability density function (PDF) requires a larger number of random dimensions, and nonlinearity requires higher-order polynomials, both of which increase the size of the system of equations produced by the Galerkin projection. Walters and Huyse [3] showed that the exponential convergence rate for discretized equations was dependent on grid resolution, and higher-order chaos was required for denser grids. Several researchers have also noted the difficulties associated with PC's inherent intrusiveness [2,27,31–33], requiring substantial modifications to deterministic codes. Additionally, Wan and Karniadakis [34] showed that PC can behave poorly for long-term flow simulations unless high-order chaos is used, increasing the computational expense. Debusschere et al. [35] described the evaluation of nonpolynomial functions, such as exponentials and logarithms, which is not possible with the standard Galerkin projection, and difficulties with high-order powers of random variables, which can result in large truncation errors.

In addition to GPC, there have been several modifications to the original PC scheme to address its drawbacks. Lucor and Karniadakis [30] developed adaptive GPC, which uses a significantly smaller system of equations to alleviate the high cost of adding dimensionality without sacrificing accuracy. Wan and Karniadakis [34] developed multi-element GPC (ME-GPC) which accepts arbitrary PDFs by numerically constructing the polynomial basis functions. They found that ME-GPC could handle long-term integration without the increased chaos order that GPC requires. Perez and Walters [27] constructed a compact notation for standard GPC, which makes conversion from deterministic to stochastic equations more intuitive and readable, with the potential for automation.

To address the difficulty of implementing intrusive PC by modifying large and complex codes, researchers developed several nonintrusive methods. All rely on a series of deterministic solves, treating the deterministic code as a black box, to approximate the PC coefficients. The methods vary in the projection and integration techniques that are used to determine the PC coefficients from the deterministic solutions. LeMaître et al. [36] outlined the nonintrusive spectral projection (NISP) method that is used in this work. NISP approximates PC coefficients by combining a series of deterministic solves at collocation points corresponding to Gauss–Hermite (GH) quadrature points. The method works well for complex distributions and systems with several random dimensions, with pronounced deviations (from the exact PC coefficients) occurring only in highly coupled modes. Mathelin and Hussaini [33] developed a stochastic collocation algorithm with better properties for systems with higher dimensionality and coupling. Hosder et al. [2] developed the nonintrusive point collocation method, using a supersonic wedge and cosine airfoil to demonstrate the method's accuracy. Loeven et al. [32] developed a competing chaos collocation method that was similarly nonintrusive and was shown to maintain the exponential convergence of intrusive PC with good long-term integration properties.

III. Polynomial Chaos

In this section, the generalized polynomial chaos formulation is presented and illustrated on a simple example problem. The nonintrusive spectral projection method is then described and applied to the same problem.

A. Generalized Polynomial Chaos Formulation

Polynomial chaos is an expansion based on the homogeneous chaos developed by Wiener [14]. Wiener used Hermite polynomials in terms of Gaussian random variables as the basis for a spectral expansion of random processes. The Cameron–Martin theorem [15]

proved that the expansion could represent any second-order random process in terms of orthogonal polynomials.

Using Hermite polynomials, a second-order random process $X(\theta)$ can be expanded as

$$X(\theta) = a_0 I_0 + \sum_{i_1=1}^{\infty} a_{i_1} H_1(\xi_{i_1}(\theta)) + \sum_{i_1=1}^{\infty} \sum_{i_2=1}^{i_1} a_{i_1 i_2} H_2(\xi_{i_1}(\theta), \xi_{i_2}(\theta)) + \sum_{i_1=1}^{\infty} \sum_{i_2=1}^{i_1} \sum_{i_3=1}^{i_2} a_{i_1 i_2 i_3} H_3(\xi_{i_1}(\theta), \xi_{i_2}(\theta), \xi_{i_3}(\theta)) + \ldots \quad (1)$$

where $H_n(\xi_{i_1}, \ldots, \xi_{i_n})$ represents the Hermite chaos function of order n in terms of the random vector $\boldsymbol{\xi} = (\xi_{i_1}, \ldots, \xi_{i_n})$. The Hermite polynomials H_n are given by

$$H_n(\xi_{i_1}, \ldots, \xi_{i_n}) = e^{\frac{1}{2}\boldsymbol{\xi}^T \boldsymbol{\xi}} (-1)^n \frac{\partial^n}{\partial \xi_{i_1} \cdots \partial \xi_{i_n}} e^{-\frac{1}{2}\boldsymbol{\xi}^T \boldsymbol{\xi}} \quad (2)$$

For notational convenience, Eq. (1) is rewritten as

$$X(\theta) = \sum_{j=0}^{\infty} \hat{a}_j \Psi_j(\boldsymbol{\xi}) \quad (3)$$

where there is a one-to-one correspondence between the functions $H_n(\xi_{i_1}, \ldots, \xi_{i_n})$ and $\Psi_j(\boldsymbol{\xi})$. Note that the summation now begins at $j = 0$ [20].

The Hermite polynomials form a complete basis and are inherently orthogonal. The PC orthogonality relation takes the form

$$\langle \Psi_i \Psi_j \rangle = \langle \Psi_i^2 \rangle \delta_{ij} \quad (4)$$

where δ_{ij} is the Kronecker delta, and $\langle \cdot | \cdot \rangle$ denotes the ensemble average, the inner product in the Hilbert space of the Gaussian variables $\boldsymbol{\xi}$, defined by

$$\langle f(\boldsymbol{\xi}) g(\boldsymbol{\xi}) \rangle = \int f(\boldsymbol{\xi}) g(\boldsymbol{\xi}) W(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \quad (5)$$

where $W(\boldsymbol{\xi})$ is the weighting function corresponding to the polynomial basis $\{\Psi_i\}$. For the Hermite polynomials, this has the form of an n-dimensional PDF:

$$W(\boldsymbol{\xi}) = \frac{1}{\sqrt{(2\pi)^n}} e^{-\frac{1}{2}\boldsymbol{\xi}^T \boldsymbol{\xi}} \quad (6)$$

Hermite polynomials are paired with Gaussian distributions because they are in terms of Gaussian variables, and the polynomials are orthogonal to the weighting function $W(\boldsymbol{\xi})$ [20].

Deviations from Gaussian distributions adversely affect the convergence properties of standard PC. For example, the exponential convergence of the Cameron–Martin theorem does not hold when representing a uniform distribution with Hermite chaos. In response, Xiu and Karniadakis [20] proved that there were several additional pairings of basis polynomials and distributions, which were possible with the Askey scheme of polynomials, thus generalizing the standard PC to what they termed Askey chaos, also known as generalized PC. Table 1 shows the pairings. The preceding equations apply for GPC, but the symbols are often changed in the literature to distinguish it from standard PC.

B. PC Example

The procedure for representing and solving stochastic systems with PC is illustrated by an example adapted from [20].

Consider the ordinary differential equation (ODE):

$$\frac{dy(t)}{dt} = -ky, \qquad y(0) = \hat{y} \quad (7)$$

where the decay rate is a random variable $k(\theta)$, represented by a Gaussian distribution with a mean value μ_k and a standard deviation σ_k. The aim is to find the probabilistic behavior of y based on the known probabilistic behavior of k.

Based on the given mean and standard deviation, a single random realization of k is represented as

$$k(\theta) = \mu_k + \sigma_k \xi_1(\theta) \quad (8)$$

where θ is the random dimension and $\xi_1(\theta)$ is the Gaussian random variable; $\xi_1(\theta)$ is often notated as ξ_1 for convenience. The subscript is necessary to distinguish between multiple uncorrelated random variables for a problem with several random dimensions.

The root of the problem is the fact that the deterministic equation (7) cannot account for the probabilistic behavior of $k(\theta)$. MCSs are considered to be an exact means of determining the statistical properties (mean, standard deviation, skew, etc.) of such a stochastic system. In this case, the Gaussian variable $k(\theta)$ would be sampled and substituted into Eq. (7), then the equation would be solved with an ODE solver. This process of sampling and solving would continue until the desired statistical properties of $y(t)$, such as mean, standard deviation, etc., converged (so that further sampling and solving would not increase the accuracy of their values significantly). For simple problems such as this example, the process may require less than 100 samples to converge, but for more complex problems and simulations, many thousands of iterations may be required to obtain accurate statistical data.

Polynomial chaos, on the other hand, requires some mathematical adjustments to the problem formulation, but requires very few deterministic solves to obtain accurate statistical data. To account for probabilistic behavior, the independent and dependent variables must be replaced by equivalent PC expansions. In this case, PC expansions are applied to both the stochastic input $k(\theta)$ and the dependent variable $y(t; \theta)$:

$$k(\theta) = \sum_{i=0}^{\infty} k_i \Psi_i(\xi_1), \qquad y(t; \theta) = \sum_{i=0}^{\infty} y_i(t) \Psi_i(\xi_1) \quad (9)$$

Note that the random dimension θ is added to both the input and the dependent variable. The expansion of $y(t; \theta)$ does not require a second random dimension because it is dependent on $k(\theta)$ and therefore on ξ_1. The crucial element of the spectral expansion is the separation of the stochastic processes into deterministic coefficients k_i and $y_i(t)$ and random basis polynomials $\{\Psi_i(\xi_1)\}$.

Table 1 Correspondence of random variable distributions and chaos polynomials (N is a finite, nonnegative integer) [20]

Distribution type	Random variable ζ	Chaos polynomial $\{\Phi(\zeta)\}$	Support
Continuous	Gaussian	Hermite chaos	$(-\infty, \infty)$
	Gamma	Laguerre chaos	$[0, \infty)$
	Beta	Jacobi chaos	$[a, b]$
	Uniform	Legendre chaos	$[a, b]$
Discrete	Poisson	Charlier chaos	$\{0, 1, 2, \ldots\}$
	Binomial	Krawtchouk chaos	$\{0, 1, \ldots, N\}$
	Negative binomial	Meixner chaos	$\{0, 1, 2, \ldots\}$
	Hypergeometric	Hahn chaos	$\{0, 1, \ldots, N\}$

The expansions are then substituted into the differential equation:

$$\sum_{i=0}^{\infty} \frac{dy_i(t)}{dt}\Psi_i = -\sum_{i=0}^{\infty}\sum_{j=0}^{\infty} k_i y_j(t)\Psi_i\Psi_j \tag{10}$$

The summations must be truncated for practical calculation, and the upper summation limit P depends on the number of random dimensions n and the desired order of the polynomials p as

$$P+1 = \frac{(p+n)!}{p!n!} \tag{11}$$

Truncation introduces error, and a Galerkin projection is used to ensure that the error is projected perpendicularly to the reduced Hermite polynomial basis $\{\Psi_l\}$, where $l = 0, 1, \ldots, P$. The projection [shown in Eq. (12)] is based on Eq. (5). Each polynomial in the reduced basis $\{\Psi_l\}$ is projected against the truncated version of Eq. (10) to form $P+1$ *deterministic* differential equations, which can be solved as a system:

$$\left\langle \sum_{i=0}^{P} \frac{dy_i(t)}{dt}\Psi_i, \Psi_l \right\rangle = \left\langle -\sum_{i=0}^{P}\sum_{j=0}^{P} k_i y_j(t)\Psi_i\Psi_j, \Psi_l \right\rangle \qquad l = 0, 1, \ldots, P \tag{12}$$

Because of the orthogonality condition (4), Eq. (12) reduces to

$$\frac{dy_l(t)}{dt} = -\frac{1}{\langle\Psi_l^2\rangle}\sum_{i=0}^{P}\sum_{j=0}^{P} k_i y_j(t) e_{ijl} \qquad l = 0, 1, \ldots, P \tag{13}$$

where $e_{ijl} = \langle\Psi_i\Psi_j\Psi_l\rangle$, and e_{ijl} and $\langle\Psi_l^2\rangle$ can be precomputed using the inner product definition (5).

The case in which $l = 0$ is expanded and projected explicitly as an illustrative example. The first projection in Eq. (12) involves Ψ_0:

$$\left\langle \sum_{i=0}^{P} \frac{dy_i(t)}{dt}\Psi_i, \Psi_0 \right\rangle = \left\langle -\sum_{i=0}^{P}\sum_{j=0}^{P} k_i y_j(t)\Psi_i\Psi_j, \Psi_0 \right\rangle \tag{14}$$

Because of the orthogonality condition, $\langle\Psi_i, \Psi_0\rangle = 0$ except when $i = 0$, and so the left-hand side reduces to

$$\frac{dy_0(t)}{dt}\langle\Psi_0^2\rangle$$

The right-hand side does not reduce, however, because the orthogonality relation does not apply to the inner product of three polynomials. So Eq. (14) simplifies to

$$\frac{dy_0(t)}{dt}\langle\Psi_0^2\rangle = -\sum_{i=0}^{P}\sum_{j=0}^{P} k_i y_j(t) e_{ij0} \tag{15}$$

This is then rearranged into the format more commonly found in the literature by bringing $\langle\Psi_0^2\rangle$ to the right-hand side. This example pairs Hermite polynomials with Gaussian random variables, and so Eq. (2) is used to compute e_{ij0} and $\langle\Psi_0^2\rangle$ using Eq. (5). Once the inner products are computed, all random elements from the equation have been removed. The equation can be further simplified because the behavior of $k(\theta)$, and thus the behavior of the coefficients k_i, are known:

$$k(\theta) = \mu_k + \sigma_k\xi_1(\theta) = \sum_{i=0}^{P} k_i\Psi_i \tag{16}$$

Expanding the right-hand side and substituting in the first three Hermite polynomials ($\Psi_0 = 1$, $\Psi_1 = \xi$, and $\Psi_2 = \xi^2 - 1$) results in

$$k(\theta) = \mu_k + \sigma_k\xi_1 = k_0(1) + k_1(\xi_1) + k_2(\xi_1^2 - 1) + \cdots + k_P\Psi_P \tag{17}$$

Thus, it can be seen that $k_0 = \mu_k$, $k_1 = \sigma_k$, and $k_i = 0$ for $i > 1$, and so Eq. (15) reduces according to Eq. (18):

$$\begin{aligned}\frac{dy_0(t)}{dt} &= -\frac{1}{\langle\Psi_0^2\rangle}\sum_{i=0}^{P}\sum_{j=0}^{P} k_i y_j(t) e_{ij0} \\ &= -\frac{1}{\langle\Psi_0^2\rangle}\left\{k_0\sum_{j=0}^{P} y_j(t)e_{0j0} + k_1\sum_{j=0}^{P} y_j(t)e_{1j0}\right. \\ &\quad \left. + k_2\sum_{j=0}^{P} y_j(t)e_{2j0} + \ldots + k_P\sum_{j=0}^{P} y_j(t)e_{Pj0}\right\} \\ &= -\frac{1}{\langle\Psi_0^2\rangle}\left\{\mu_k\sum_{j=0}^{P} y_j(t)e_{0j0} + \sigma_k\sum_{j=0}^{P} y_j(t)e_{1j0}\right\}\end{aligned} \tag{18}$$

The next step is to decide the polynomial order. If second-order polynomials are used, and recalling $n = 1$ here, $P = 2$ based on Eq. (11). Equation (18) can then be expanded to form

$$\begin{aligned}\frac{dy_0(t)}{dt} = -\frac{1}{\langle\Psi_0^2\rangle}\{&\mu_k[y_0(t)e_{000} + y_1(t)e_{010} + y_2(t)e_{020}] \\ &+ \sigma_k[y_0(t)e_{100} + y_1(t)e_{110} + y_2(t)e_{120}]\}\end{aligned} \tag{19}$$

The other P equations in the system are similarly formed by projecting the remaining P Hermite polynomials from Eq. (12). The result is a deterministic system of $P+1$ equations in which the only unknowns are the $P+1$ PC coefficients $\{y_j\}$, for $j = 0, \ldots, P$, which can be found using an ODE solver [20]. Once found, the PC coefficients can be substituted back into the PC expansion of $y(t;\theta)$ to calculate its stochastic response:

$$\begin{aligned}y(t;\theta) &= \sum_{i=0}^{2} y_i\Psi_i(\xi_1) = y_0(t)\Psi_0 + y_1(t)\Psi_1 + y_2(t)\Psi_2 \\ &= y_0(t)(1) + y_1(t)\xi_1 + y_2(t)(\xi_1^2 - 1)\end{aligned} \tag{20}$$

The mean response of the system, $\bar{y}(t)$, can be calculated as

$$\bar{y}(t) = E[y(t;\theta)] \tag{21}$$

which simplifies to y_0, and so $\bar{y}(t)$ can be determined without further calculation. The system standard deviation can be calculated as

$$\sigma(t) = \sqrt{E[(y(t;\theta) - \bar{y}(t))^2]} \tag{22}$$

Skew can also be calculated as the third moment about the mean. For higher-order statistics, higher-order chaos is necessary. In this case, if $k(\theta)$ is assumed to have zero mean and unit standard deviation, the deterministic response will be constant, whereas the mean stochastic response will increase with time, emphasizing the weakness of deterministic modeling of stochastic systems [20].

C. Nonintrusive Spectral Projection Formulation

The previous sections described an intrusive PC scheme requiring direct modification of a deterministic code. As codes become more complex and computation-intensive, modification becomes a difficult and time-consuming task. Also, implementing PC with nonpolynomial functions is not straightforward and is still an active research topic [35]. In response, nonintrusive methods, which approximate the PC coefficients while treating the deterministic code as a black box, have been developed and tested extensively. These methods allow the user to construct stochastic CFD tools using well-known and well-tested deterministic CFD codes with little additional coding.

One such method, NISP, is based on the observation that PC coefficients for a variable a (i.e., $\{a_i\}$, for $i = 0, \ldots, P$) can be obtained by projecting the deterministic solution onto the PC basis, $\{\Psi_i\}$. If $a^d(\boldsymbol{\xi})$ represents the deterministic solution corresponding to a particular realization $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)$, where n is the number of random dimensions, then the PC coefficients are defined by [36]

$$a_i = \frac{\langle a^d(\boldsymbol{\xi})\Psi_i\rangle}{\langle \Psi_i \Psi_i\rangle} \equiv \int_{-\infty}^{\infty} \mathrm{d}\xi_1 \cdots \int_{-\infty}^{\infty} \mathrm{d}\xi_n \left[a^d(\boldsymbol{\xi}) \frac{\Psi_i(\boldsymbol{\xi})}{\langle \Psi_i \Psi_i\rangle} \prod_{k=1}^{n} \frac{\exp(-\xi_k^2/2)}{\sqrt{2\pi}} \right] \quad (23)$$

For moderate values of n, this integral can be solved with GH quadrature [36,37]. Using m quadrature points for each basis polynomial in the set $\{\Psi_i\}$, the integral can be approximated by

$$a_i = \sum_{m_1=1}^{m} \cdots \sum_{m_n}^{m} a^d(x_{m_1}, \ldots, x_{m_n}) \frac{\Psi_i(x_{m_1}, \ldots, x_{m_n})}{\langle \Psi_i \Psi_i\rangle} \prod_{k=1}^{n} w_{m_k} \quad (24)$$

where x_k and w_k, with $k = 1, \ldots, m$, are the one-dimensional GH integration points and weights. This quadrature rule is exact when the integrand is a polynomial with a degree less than $2m$. The integrand has one explicit polynomial Ψ_i, and it also has in a^d an implicit polynomial of equal order, because a can be approximated as a PC expansion of the polynomials $\{\Psi_i\}$: that is,

$$a = \sum_{i=0}^{P} a_i \Psi_i$$

[see Eq. (3)]. Thus, the order of the integrand is actually twice the order p of the polynomial Ψ_i. Because the quadrature rule is exact, if the integrand has a degree less than $2m$, then m must be greater than or equal to $p + 1$. Therefore, Eq. (24) requires $(p + 1)^n$ solves of the deterministic system. Notably, this will always be greater than or equal to the $P + 1$ solves required for intrusive PC [see Eq. (11)], but because NISP requires little code modification and can accurately approximate the PC coefficients, it is an attractive option, considering the extensive modifications required for intrusive PC [36].

D. Nonintrusive Spectral Projection Example

The NISP method can also be illustrated for the example based on Eq. (7). In this problem, k is the uncertain variable, defined by Eq. (8), with one random dimension. For second-order PC, the number of GH quadrature points must be $m = p + 1 = 3$. Equation (24) becomes

$$y_i = \sum_{m_1=1}^{m} y^d(x_{m_1}) \frac{\Psi_i(x_{m_1})}{\langle \Psi_i \Psi_i\rangle} w_{m_1} \quad (25)$$

where x_{m_1} represents a single realization of the Gaussian random variable, and so $k = \mu_k + \sigma_k \xi_1$ becomes $k = \mu_k + \sigma_k x_{m_1}$, which is substituted for the deterministic value of k in solving for y. Any ODE solver can be used to solve the deterministic problem for $y^d(x_{m_1})$. Then Eq. (25) can be used to solve for each PC mode $\{y_i\}$, where $i = 0, \ldots, P$. At first glance, it appears that the m deterministic solves must be computed for each of the $i = 0, \ldots, P$ PC coefficients, but because y^d is only dependent on the GH quadrature point x_{m_1}, the same m deterministic solves can be used to calculate each $\{y_i\}$, resulting in $(p + 1)^n$ deterministic solves, rather than $(P + 1)(p + 1)^n$.

IV. Robust Optimization

A. Motivation

The basic task of any computational optimization routine is to maneuver through a search space to minimize or maximize a user-defined objective. Experience in the last decade has shown that optimized designs can be sensitive to variability in manufacturing and operating conditions, resulting in severely degraded performance of supposedly optimized designs [5]. That is, a slight variation in a design variable from its optimal value can cause a significant deterioration in the objective-function value.

One solution is to ensure that there is no input variability, but this is not realistic because uncertainty and variability are inherent parts of the production and operating environments. Another solution is to create optimized designs that are insensitive to variation: a robustly optimized design. Figure 1 contrasts the sensitivity of robust and nonrobust designs.

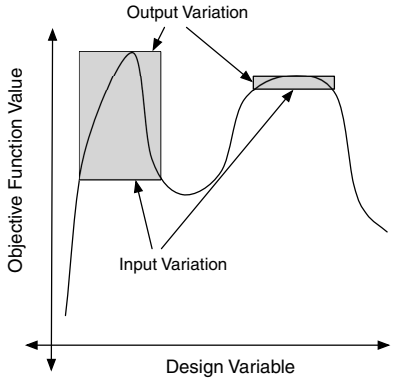


Fig. 1 Contrasting robust (on the right) and nonrobust design points.

The robust design point may not be the global optimum, but the danger of suboptimal performance due to variation is mitigated. This is an acceptable or necessary tradeoff for many applications.

B. Basic Robust Optimization Description

1. *Multiobjective Optimization*

Robust optimization involves optimizing (minimizing or maximizing) the mean of some desired observables *and* optimizing (always minimizing) the variance in those observables. Thus, robust optimization is inherently multiobjective: even if there is only a single main observable to optimize, its variance must be minimized, resulting in at least two objectives for any robust optimization problem.

Multiobjective optimization does not usually produce a single solution or design point, but rather a set of *nondominated* solutions. For an optimization problem with two objectives, two design points (1 and 2) are nondominated with respect to each other if 1 has a superior value for one objective but an inferior value for the other. For example, when maximizing lift-to-drag ratio L/D and minimizing its variance, two solutions are nondominated if $(L/D)_1 > (L/D)_2$ *and* $\sigma_1 > \sigma_2$. Nondominated solutions are often represented as a Pareto set that shows tradeoffs between two or more objectives.

2. *Stochastic Systems*

Because robust optimization involves optimizing both the mean and variance of one or more observables, the underlying system must compute such information from similar input information. In the context of this work, for each objective-function evaluation, the underlying system accepted Gaussian random variables with a defined mean and variance, representing geometric uncertainty in the edge thicknesses of an airfoil, and computed mean and variance information for the airfoil's L/D.

Obtaining the variance information required involves modeling a stochastic system. There are several methods for modeling stochastic systems (see Sec. II), of which MCS is the most accurate and straightforward, but it usually requires on the order of 10^3 deterministic runs to obtain accurate output statistics for a single design point. Optimization routines usually require several hundred or several thousand design-point evaluations. If MCS is used to obtain the mean and variance, the computational cost becomes prohibitive.

Another solution is to find an efficient means of obtaining variance information for a system, and nonintrusive PC is a promising candidate. A single objective-function evaluation still requires several deterministic runs, but this is on the order of 10^0 or 10^1, rather than 10^3, making robust optimization computationally feasible.

C. Problem Statement

The primary aim of this work is to investigate the hypothesis that PC methods are efficient means of obtaining stochastic information for use in robust optimization. Validating this hypothesis requires

showing both that the PC method is efficient and that it can be used to create a robust design that performs better than a nonrobust design under uncertainty. The former has been adequately demonstrated by the current body of PC literature (see Sec. II). The latter requires two different optimization problems. The first is a standard nonrobust problem: optimizing a design with respect to some observable(s). In this problem, the inputs and outputs are deterministic. The second problem is identical in every way except that the design is robustly optimized: optimizing some observables and minimizing their variance. In the second problem, the inputs are stochastic and PC is used to obtain output mean and variance information. To demonstrate the robustness of the second design (or designs in the Pareto set), the nonrobust design, which was optimized with deterministic inputs, is tested with the same stochastic inputs used in the robust design problem. The method is validated if the robust designs have similar mean performance but are less sensitive to the input uncertainty.

The particular application chosen involves maximizing L/D using a NACA 0009 airfoil as the base design. The robust aspect of the optimization will focus on geometric uncertainty analogous to the tolerances of a manufacturing process. Specifically, the airfoil's leading-edge thickness will be considered to be uncertain with a defined mean and variance. The deterministic core of the simulation is the XFOIL airfoil design code.[‡] Using XFOIL demonstrates how a well-known simulation code can be combined with PC *without modification to the code* for uncertainty quantification and robust optimization.

This study demonstrates a method, not a particular application. Thus, it is the number of deterministic runs, not the actual runtime, which is relevant when measuring efficiency. Because the method can be applied to any deterministic optimizable process, a user who knows the time required for a single deterministic run of their application can gauge whether this method is computationally feasible based on the number of deterministic runs noted here.

D. Setup

There are three main modules to this robust optimization framework. The deterministic solver is the base module. In this case, it includes XFOIL, the script used to interact with XFOIL, and the base airfoil geometry. The second is the uncertainty quantification module, which uses PC to propagate uncertain inputs through the deterministic process to obtain stochastic output information. The final module is the optimizing algorithm, which uses the mean and variance data from the uncertainty quantification to drive design changes, attempting to find an optimized robust design. These three modules are independent of the underlying deterministic process being optimized. The nonrobust method, in contrast, simply requires the underlying deterministic process and an optimization routine.

1. Deterministic Solver

The deterministic solver was essentially a MATLAB framework, abbreviated here as MFVP (Manufacturing and Flow Variation Project), that was specifically designed to simulate manufacturing and flow variation and was capable of "wrapping" different simulation cores with minimal code changes, making it a flexible tool for simulating variation.

The basic modules included the following:

1) The geometry generation module generates the basic geometry.

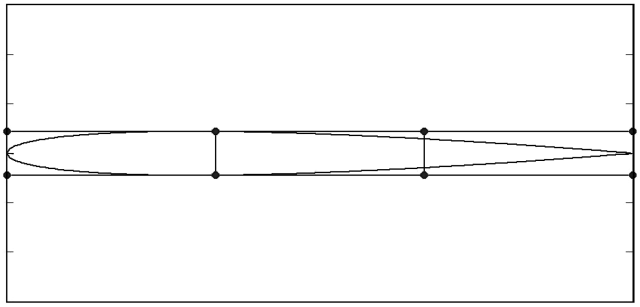
2) The geometry manipulation module uses free-form deformation to manipulate the geometry, mainly for optimization purposes.

3) The manufacturing variation module uses free-form deformation to simulate small perturbations from manufacturing.

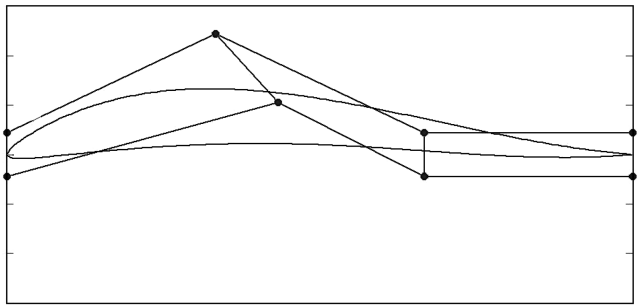
4) The grid generation module generates a CFD grid around the geometry.

5) The flow variation module perturbs the flow to simulate physical phenomena.

6) The CFD solver module is the basic flow solver.



a) Nominal geomatry with FFD control points



b) Deformed geometry with FFD control points

Fig. 2 NACA 0009 airfoil geometry modified with FFD.

7) The postprocessing module processes results from the CFD solver.

Because not every application required all modules, the framework efficiently skipped those that were unnecessary.

The geometry generation module returned Cartesian coordinates for the desired airfoil. In this case, a NACA airfoil was generated based on inputs for the maximum camber, location of the maximum camber, and the airfoil thickness. This was passed to the geometry manipulation routine, which used free-form deformation (FFD) (see Sec. IV.D.4) to modify the nominal geometry. It accepted control parameters from the optimizer and returned the resulting modified airfoil geometry. Figure 2 shows the nominal NACA 0009 airfoil and a modified geometry based on manipulated control points.

The manufacturing variation module also used FFD, but on a smaller scale. It made small perturbations to the leading- and trailing-edge thicknesses to simulate potential variations from manufacturing. The routine treated these perturbations as Gaussian random variables with zero mean and a set variance according to

$$\Delta t_{\mathrm{LE}} = \mu_{\Delta t} + \sigma_{\Delta t}\xi = 0 + \sigma_{\Delta t}\xi \tag{26}$$

where Δt_{LE} is the perturbation to the leading-edge thickness, $\mu_{\Delta t}$ is the mean perturbation (nominally zero), and $\sigma_{\Delta t}$ is the standard deviation of the perturbation. The input to the module was the random realization ξ, and the module calculated Δt, applying it to the geometry via FFD.

For XFOIL, the grid generation module was skipped because geometry representations were internal to XFOIL. For other applications, the framework can interact with a grid generation routine to produce any grid coordinates needed by the core flow solver.

The flow variation module handles variations in flow parameters that are similar to the manufacturing variations in the leading-edge thickness. For example, the module could treat the angle of attack α as a Gaussian random variable with a defined mean and variance.

For the XFOIL application, flow parameters such as the angle of attack, Mach number, and Reynolds number were saved and used as inputs to the script that interacted with XFOIL. The core flow solver included XFOIL and a script written using the scripting program Expect,[§] which handled XFOIL's interactive interface. The script

[‡]XFOIL is a well-validated airfoil design code. It generates accurate viscous solutions for subsonic flows using inviscid-viscous interaction [38].

[§]Data available online at http://expect.nist.gov/ [retrieved 8 November 2008].

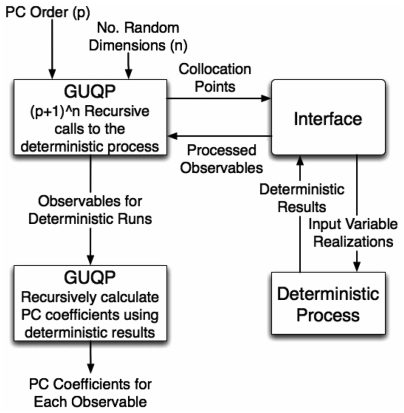


Fig. 3 Basic flow of the GUQP.

navigated XFOIL's menus to load the airfoil geometry, set the viscous parameters, set the angle of attack, then write the flow results to a file. The script included elements to handle nonconvergence and unexpected errors. Typical output from XFOIL was a file listing basic flow parameters and the lift, drag, and moment coefficients.

Finally, the postprocessing module processed raw output from the core flow solver. For XFOIL, this involved reading lift and drag information from the output file and calculating L/D. The power of this framework is its ability to interact with a variety of higher-level applications, because it accepted a common set of input parameters and returned a consistent set of outputs, hiding the implementation and simulation details.

2. *Uncertainty Quantification*

The uncertainty quantification module implemented PC using nonintrusive spectral projection (Sec. III.B), which approximated PC coefficients while treating the deterministic solver as a black box. NISP will always be less efficient than intrusive PC methods, but intrusive PC can be complex and time-consuming to implement and must be implemented separately for every code, whereas NISP requires very little coding and does not require modifications to the deterministic code, making it ideal as the basis of an uncertainty quantification tool [36].

A separate framework was created within MATLAB for Gaussian uncertainty quantification. The framework, abbreviated here as GUQP (Gaussian Uncertainty Quantification Project), was designed to interface with any deterministic process for which uncertainty information was desired. It was limited to modeling uncertainty as perfect Gaussian random variables, but this was sufficient for the present work. Figure 3 shows the general flow of the tool and its interface with the deterministic process. GUQP recursively calculated the summation in Eq. (24). Recursion provided intuitive flexibility for problems of arbitrary dimension (note this is random, not spatial, dimensions).

The first part of the flow diagram in Fig. 3 indicates that GUQP recursively called the deterministic process via an interface. This section of the tool only handled the deterministic runs and stored the results for later use in computing Eq. (24). The section recursively constructed the vector of collocation points $(x_{m_1}, \ldots, x_{m_n})$ and sent it to the interface. This used the vector of random realizations to calculate realizations for each uncertain input, then called the deterministic process. Finally, the interface returned all observables for which uncertainty information was desired. For example, for a problem using XFOIL with two random dimensions (e.g., leading- and trailing-edge thicknesses), the interface would accept two collocation points from GUQP and calculate realizations of the two random inputs. The interface would then send these two inputs to the framework, which would run XFOIL for the specific geometry with the calculated leading- and trailing-edge thicknesses. Having stored observables for each deterministic run, GUQP would compute the PC coefficients according to Eq. (24). It again worked recursively to move through each summation, using the stored observables to compute each PC coefficient based on the PC order p, specified by the user.

The use of two separate recursive segments may appear inefficient at first. However, it actually increased the speed by a factor equal to the desired maximum chaos index P. As mentioned in Sec. III.D, the deterministic solution only depends on the collocation points, and so all deterministic solutions could be computed in advance of calculating the $P + 1$ PC coefficients. This was the work of the first recursive section. The second recursive section then computed the PC coefficients in $(P + 1)(p + 1)^n$ recursive iterations using the stored deterministic data. The deterministic runs were, by far, the limiting part of the computation, and so the cost of each GUQP run was $(p + 1)^n$, rather than $(P + 1)(p + 1)^n$.

Once the PC coefficients were calculated for each observable, they were used to compute corresponding Gaussian modes (i.e., mean, standard deviation, skewness, etc.), which were more intuitive and useful for robust optimization than raw PC coefficients.

3. *Optimization*

The application in this study focused on optimizing a single observable: L/D for an airfoil based on the NACA 0009 airfoil. The application was used to test whether PC methods were an efficient basis for calculating stochastic information that is necessary for robust optimization. Such a validation required two separate optimization problems: a single objective, nonrobust problem and a multiobjective robust problem. Thus, two separate optimization algorithms were necessary.

a. Differential Evolution. Differential evolution (DE) was selected as the basis for both the single- and multiobjective algorithms because of its well-established, efficient, and simple-to-code implementations for both.

The basic layout of the single-objective implementation follows [39]:

1) Create initial population of size NP.
2) Evaluate each member of the population.
3) Until some stopping criterion is met:
   a) Randomly enumerate the population.
   b) For each member of the population P_i (where $i = 1, \ldots,$ NP):
      1. Choose three members P_a, P_b, and P_c (where all are distinct to each other and to P_i).
      2. Create a mate $M = P_a + F(P_b - P_c)$ (where F is a user-defined factor in the range of 0–1).
      3. Breed the parent P_i with the mate M to create a child through crossover.
      4. Evaluate the child.
      5. If the child is better than the parent, replace the parent immediately.
      6. If the child is not better than the parent, reject it.

Each member of the population was represented as a chromosome holding the values for each control variable being manipulated in the optimization problem. When mate M is bred with parent P_i, random numbers in the range of 0–1 are selected, corresponding to each control variable. If an individual random number is greater than C (the crossover probability) the corresponding control variable from M is selected for the child; otherwise, the value from P_i is used. This algorithm has been extensively tested and shown to have excellent convergence properties [39].

b. Differential Evolution for Multiobjective Optimization. The multiobjective implementation DEMO (differential evolution for multiobjective optimization) was analogous to the single-objective method. It was developed by Robic and Filipic [39] and compared favorably against other multiobjective optimization algorithms when run on standard benchmark problems [39].

The main differences between the single- and multiobjective implementations are due to the nature of multiobjective optimization and the property of nondominance in particular. If a child dominates a parent, the parent is replaced. If a parent dominates a child, the child is rejected. However, if parent and child are nondominating with respect to each other, the child is *added* to the population. Thus, at the end of a generation there are between NP and 2NP population

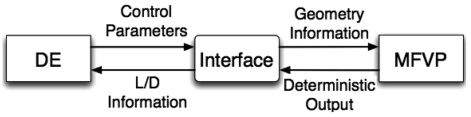


Fig. 4 Flow diagram for the differential evolution algorithm using MFVP.

members. These are sorted based on their Pareto ranking,[¶] and then the members of each front are sorted according to crowding distance, a measure of how close each member of a Pareto front is to the other members of the front. Finally, the population is truncated back to NP members. This process has two benefits. First, the best (rank 1) solutions are grouped together, and so truncation removes inferior solutions. Second, sorting by crowding distance before truncation helps to maintain diversity in the population, because members with low crowding distances are more likely to be lost in truncation. The algorithm tends to converge rapidly to the optimal Pareto front and then spread out along it, discovering more diverse nondominated solutions [39].

c. Search Space Exploration. The design space consisted of geometric modifications to a NACA 0009 airfoil. The modifications were created using free-form deformation (see Sec. IV.D.4). An eight-point FFD box was set up around the airfoil permitting up to 16 control parameters (2 degrees of freedom for each control point). Figure 2 shows an example of an airfoil modified by an eight-point FFD box.

d. Interface Implementation. Both the single- (DE) and multiobjective (DEMO) optimization implementations used application-specific interfaces to connect with MFVP and GUQP, respectively, hiding the details of the problem being optimized from the DE and DEMO algorithms. Figures 4 and 5 present flow diagrams for both implementations.

The DE algorithm only required deterministic information, and so it was interfaced directly with MFVP. The interface accepted control parameter information and called MFVP for a single deterministic run with the given geometry modifications. The lift-to-drag information was returned to the DE implementation by the XFOIL MFVP variant as the main observable.

The DEMO implementation for robust optimization used GUQP because it required stochastic information for each objective-function evaluation. The DEMO algorithm sent control parameter information to the GUQP interface, which called the GUQP code with the desired geometry modifications. However, instead of a single deterministic run, GUQP conducted $(p+1)^n$ deterministic runs. Finally, the desired Gaussian modes (mean and variance) of L/D were calculated and returned to the optimizer.

4. Free-Form Deformation

The free-form deformation implementation used in this work was based on that of Sederberg and Parry [40]. FFD is a technique for deforming solid geometric models. The algorithm is capable of "deforming any type or degree of surface primitive including planes, quadrics, parametric surface patches, or implicitly defined surfaces" [40]. Sederberg and Parry [40] showed that the algorithm was ideally suited for both global and local deformations and that local deformations could be conducted to an arbitrary degree of derivative continuity. This made it ideal for both the large-scale deformations required by optimization (see Fig. 2), as well as the small-scale deformations used to simulate manufacturing perturbations and variation.

A physical analogy to FFD is a box filled with clear, flexible plastic in which geometric objects are embedded. Assuming that the geometric objects are of the same flexible plastic and are bound to their plastic surroundings, deforming the plastic box will deform the enclosed objects in an intuitive manner. The intuitive deformation is one of the powerful aspects of the method [40].

[¶]Nondominated members of the population are rank 1; rank 2 solutions are those dominated only by rank 1 solutions, etc.

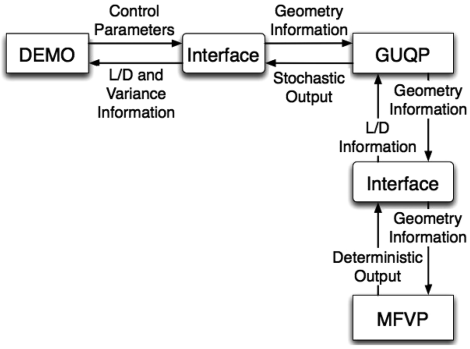


Fig. 5 Flow diagram for the differential evolution for multiobjective optimization using GUQP.

Mathematically, FFD is based on trivariate Bernstein polynomials. Similar to Bezier curves and patches, the control points define the geometry of the coordinate lines and planes, and so adjusting control points deforms the coordinate system. The internal geometry, including the embedded object, is then mapped onto the new coordinate system. Because moving the control points creates an intuitive change in the coordinate planes, the deformation in the geometry is intuitive as well [40].

E. Results

Initial tests focused on a limited problem with two control variables and a single random dimension. Several runs were conducted to ensure that both the robust and nonrobust optimization problems were converging. Once satisfactory results were achieved within this framework, scalability tests were conducted by increasing the number of random dimensions and the number of control variables.

The control variables related to the control points of the FFD box surrounding the airfoil geometry. For the main problem, the control variables were restricted to independent vertical movement in the second set of points from the leading edge. For tests with a higher numbers of control variables, the other middle points were freed for vertical movement, resulting in four control variables.

The single random dimension corresponded to the thickness of the leading edge. The same FFD box was used for creating variation in the leading edge. The points used to effect this change were the same as those used for the preceding larger-scale modifications, but here they were restricted to *horizontal* motion and displaced the same distance toward the leading or trailing edge, rather than being varied independently. As the points moved forward, they pushed the geometry toward the leading edge, making it thicker. Conversely, as they moved backward, they effectively thinned out the leading edge by pulling the geometry toward the trailing edge. For tests with a higher number of random dimensions, the other pair of middle points was freed in the horizontal direction, moving together to affect the trailing-edge thickness in a similar manner.

Within this section, the nonrobust solutions are presented first as a baseline, followed by the test results from the robust optimization runs. These results are then compared to determine whether the robust optimization successfully discovered designs that are less sensitive to input variation.

1. Primary Tests: Two Control Variables with One Random Dimension

MFVP was set up to use XFOIL as its deterministic solver with an angle of attack of 5.0 deg, a Reynolds number of 600,000, and a Mach number of 0.3. The Reynolds and Mach numbers were constant throughout all tests. The only concern was remaining below the transonic range, because XFOIL is not designed to operate in that flow regime.

GQUP was set up for a single random dimension and third-order polynomial chaos. (Second-order chaos was tried but did not produce the desired level of accuracy in σ.) The increase from second to third order increased the number of deterministic runs for uncertainty

quantification from three to four. This is a significant increase, considering that a single optimization run can require several thousand calls to the GUQP module, highlighting the potential expense of scaling with PC methods.

For the nonrobust optimization problem, the DE algorithm used a population size NP of 20 and an upper limit on the number of generations NG of 150. NP was recommended by [41] to be 10 times the number of control variables CV. NG was set high enough to ensure convergence, with the potential to break early if the population remained unchanged for a user-defined number of generations. The factors F and C were both set to 0.8, as recommended by [41].

The robust optimization algorithm (DEMO) parameters were similar. NP was again set at 20. NG was set to 100, rather than 150, after a first run showed convergence in much fewer than 150 generations. F and C were both set to 0.8 again.

NP and NG determined the upper limit on the number of deterministic runs for the optimization problem. Table 2 shows a breakdown of how many deterministic runs were required for the nonrobust and robust optimizations. In each generation, each population member was used once as the primary parent for a child, and so each generation required NP solution evaluations (to evaluate each child). For the nonrobust problem, each evaluation involved a single call to the deterministic MFVP code, resulting in 3000 total deterministic runs after 150 generations. For the robust problem, each evaluation involved a single call to the GUQP code, which required several deterministic MFVP runs for uncertainty quantification. In the case of a single random dimension and third-order PC, GQUP required four calls to MFVP for each GUQP call. Thus, each evaluation required four deterministic runs, resulting in 8000 deterministic runs for 100 generations. The potential expense of robust optimization is apparent. Compared with MCS methods, however, which can require several thousand deterministic runs *per evaluation*, the PC method is computationally inexpensive.

a. Nonrobust Results. Table 3 lists the control variable values for the optimal design from three optimization runs, along with the lift-to-drag value generated by the geometry. The three runs produced nearly identical geometries. Figure 6 shows one of these geometries. The near-identical geometries indicate that this was the best design that could be reached with the current algorithm, constraints, and objective function. As in all evolutionary optimization problems, this does not guarantee that the global optimum has been reached, but provides confidence that the result was a very good solution that was close to the global optimum.

b. Robust Results. Figure 7 shows the Pareto front found in one of the three robust optimization runs. Similar to the nonrobust problem, the results were nearly identical, indicating that all three runs converged before reaching the maximum number of generations. The results show a strong tradeoff between lift-to-drag ratio and sensitivity to input variation. Table 4 lists the maximum and minimum L/D for each run, along with its corresponding variation. Figure 8 shows the corresponding airfoil geometries from the first run.

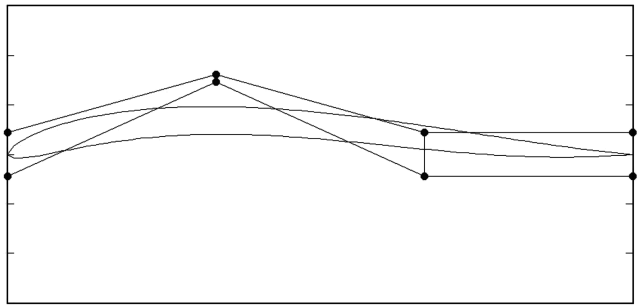


Fig. 6 Optimal airfoil from the first of three nonrobust optimization runs.

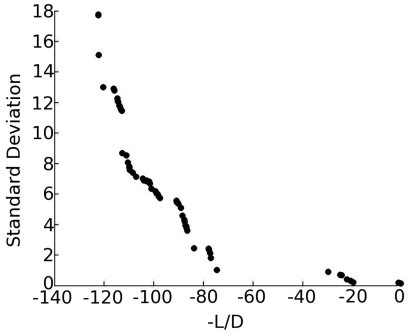


Fig. 7 Pareto front from one of the three robust optimization runs.

c. Comparison. To compare the robust and nonrobust optimized designs, the nonrobust solutions were evaluated with input uncertainty; that is, their geometries were provided as input to GUQP, with the same uncertainty parameters used in the robust optimization problem for the leading-edge variation. In Fig. 9, all three nonrobust optimal designs (which appear as a single point) are compared with one of the three Pareto fronts in terms of the resulting mean and variance in L/D.

There are three observations that can be made. First, all nonrobust optimal designs have a higher variance than even the most sensitive robust solution. Second, the mean lift-to-drag ratios of the three nonrobust designs are lower when evaluated under uncertainty, as shown in Table 5, and so not only are they very sensitive to input variation, but the input variation deteriorated their average performance. Finally, and most important, although the original L/D values of the nonrobust designs correspond to the maximum L/D in Fig. 9, their L/D values under uncertainty are significantly less than the maximum, indicating that the global maximum *for this problem* could not be reached unless input variation was considered.

Notably, these results are specific to the observable (L/D in this case). Other observables may show different characteristics, especially with respect to the global maximum only being reachable when variation is considered. However, in all cases in which there is a tradeoff between the mean and variance of an observable, robust optimization should find designs with less sensitivity than the nonrobust optimal design.

Table 2 Number of deterministic runs required for robust and nonrobust optimization

Optimization	p	n	CV	NP	Runs/child	NG	Total runs
Nonrobust	——	——	2	20	1	150	3000
Robust	3	1	2	20	4	100	8000
Robust	3	1	2	20	4	150	12,000
Robust	3	2	2	20	16	100	32,000
Robust	3	1	4	40	4	100	16,000

Table 3 Control variables and lift-to-drag values for optimal designs from three test runs

Run	Upper CV	Lower CV	L/D
1	0.1171	0.1914	123.2174
2	0.1172	0.1915	123.2174
3	0.1172	0.1917	123.2174

Table 4 Comparison of airfoil performance characteristics at the extremes of each Pareto front for each optimization run

Run	$(L/D)_{\max}$	$(L/D)_{\min}$	$\sigma(L/D)_{\max}$	$\sigma(L/D)_{\min}$
1	122.27	0.48114	17.766	0.13496
2	122.60	1.8365	18.985	0.077596
3	123.10	20.634	12.908	0.11662

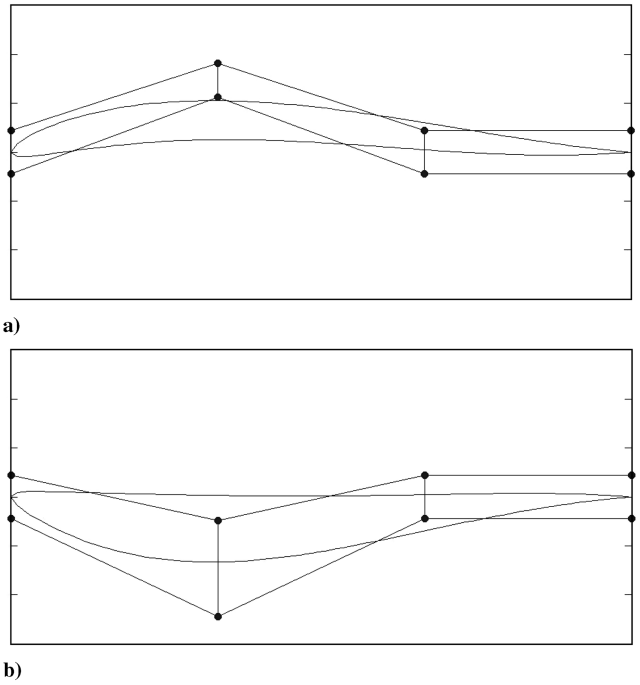


Fig. 8 Airfoil geometries from the extremes of the Pareto front in Fig. 7: a) $(L/D)_{max}$ and b) $(L/D)_{min}$.

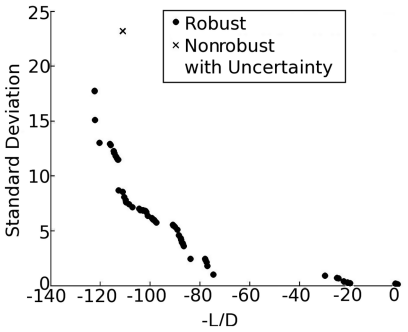


Fig. 9 Comparison of nonrobust, optimally designed airfoils with robust designs.

2. Secondary Tests: Scaling

The primary tests confirmed that the robust optimization method using PC was capable of producing designs that are less sensitive to variation than nonrobust optimal designs. The problem was limited in nature, however, being restricted to two control variables and a single random dimension. Secondary tests were therefore conducted to ensure that the method could scale to problems with more control variables and random dimensions while producing similar results.

a. Four Control Variables with One Random Dimension. For this problem, all four middle points in the eight-point FFD box were free to move independently in the vertical direction. With CV increased from 2 to 4, NP increased from 20 to 40, according to the recommendation of [41]. The increase in NP doubled the number of deterministic runs required, and so efforts were made to increase the simulation speed. The angle of attack was reduced from 5.0 to 3.0 deg, and the range of motion for the control variables was halved, both of which made the XFOIL solution easier and faster. The changes were implemented in both the nonrobust and robust optimization routines, and so the comparison was unaffected.

Table 5 Comparison of L/D values with and without uncertainty for each of the nonrobustly designed optimal airfoils along with the calculated standard deviation

Run	$(L/D)_{without}$	$(L/D)_{with}$	$\sigma(L/D)$
1	123.22	110.75	22.93
2	123.22	110.69	22.89
3	123.22	110.92	23.03

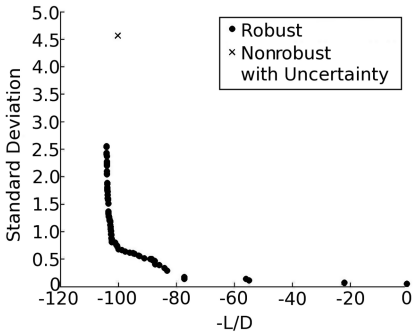


Fig. 10 Comparison of robust Pareto front with nonrobust optimal design for four control variables.

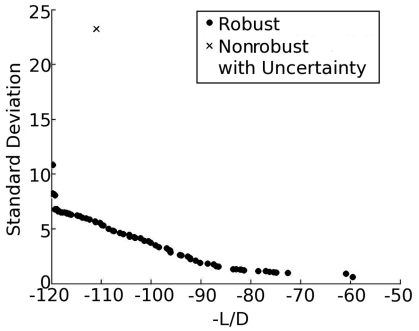


Fig. 11 Comparison of robust Pareto front with nonrobust optimal design for two random dimensions.

Figure 10 shows the Pareto front for the robust optimization problem with the nonrobust optimal design point added. The results are similar to those for two control variables, with the nonrobust design again showing much greater sensitivity to input variation, indicating that the method scaled well with CV.

Though not shown in Fig. 10, uncertainty degraded the mean performance of the nonrobust design by nearly 40%, significantly more than in the formulation with two control variables. This is not necessarily an effect of scaling, but indicates clearly that there are some designs that are highly sensitive to variation and that a nonrobust optimization risks finding such solutions.

b. Two Control Variables with Two Random Dimensions. For this problem, the original two control variables were used for searching the design space, but the trailing-edge thickness was considered to be uncertain, along with the leading edge. This was facilitated by allowing all four middle points in the eight-point FFD box to move horizontally in pairs to either fatten or squeeze the leading and trailing edges, as described earlier. Importantly, an increase from one to two random dimensions with third-order PC increased the number of deterministic runs for each call to GUQP from 4 to 16, increasing the required number of deterministic runs for the optimization problem by 400%. This was still significantly less expensive than MCS (on the order of 10 instead of 1000 deterministic runs for each uncertainty analysis), but demonstrated a significant scaling weakness in PC methods.

Other than the number of random dimensions, all parameters for both the robust and nonrobust problems remained the same as for the primary tests. Figure 11 shows the Pareto front for the robust optimization with the nonrobust optimal design point superimposed. The results are similar to those with one random dimension, indicating that the method scaled well with n.

V. Discussion

The primary aim of this work was to show that PC methods are a capable basis for uncertainty quantification in robust optimization. The demonstration involved one main problem formulation and two additional formulations that established the scalability of the method with both the numbers of control variables and random dimensions. The control variables manipulated the airfoil design, and the random dimensions related to uncertainty in the thicknesses of the leading and trailing edges.

Robust optimization should produce a set of nondominated solutions that are less sensitive to input variation than a nonrobustly optimized design. In this work, the focus was maximizing L/D for an airfoil, and so the robust designs should exhibit less variance in their L/D than a nonrobustly optimized airfoil for the same uncertainty in the leading- and trailing-edge thicknesses. This was precisely what the results showed.

In the primary problem formulation, the variance of the nonrobustly optimized airfoil, when simulated with uncertainty, was significantly larger than that of all the Pareto-optimal robust designs. The results for the secondary problems were similar, demonstrating the method worked for larger numbers of control variables and random dimensions.

The tests also show two unexpected and related results. The nominal (i.e., without uncertainty) L/D value of the nonrobust optimal design was about 23% greater than when the same design was tested under uncertainty, indicating that the input uncertainty degraded the airfoil's mean performance. Degraded performance *offdesign* was expected (i.e., high variance), but the extent of degraded performance *on-design* (i.e., decreased mean value) was unexpected. The second unexpected result was that the nonrobust optimal design did not yield the maximum mean L/D, indicating that the global optimum for this problem was not attainable without considering input uncertainty. These results show that nonrobust optimization not only risks producing designs that perform poorly under uncertainty, but also risks overpredicting performance characteristics and missing the globally optimal design.

With MCS, uncertainty quantification can require thousands of deterministic runs per evaluation, increasing the cost of the optimization, which is already high, by three orders of magnitude. PC methods, in contrast, often require only $\mathcal{O}(10^0)$ or $\mathcal{O}(10^1)$ deterministic runs per evaluation, and so the cost of optimization might increase by less than an order of magnitude. PC is clearly much more efficient, though the actual feasibility is always dependent on the runtime cost of a single deterministic simulation.

VI. Conclusions

The results of the tests presented in this paper demonstrate that polynomial chaos methods are an efficient and accurate basis for uncertainty quantification in robust optimization. They also demonstrate that uncertain conditions can degrade the mean, or on-design, performance of a nonrobustly optimized airfoil, and for some optimization problems, the global optimum cannot be attained without considering uncertainty. These findings highlight the value of including measures of robustness directly within the design optimization process. Polynomial chaos methods allow accurate uncertainty quantification to be achieved at a fraction of the cost of Monte Carlo simulations and thus make robust optimization with high-fidelity computational fluid dynamics feasible.

As the current work was limited to optimizing a single observable, the lift-to-drag ratio, further work should include testing the method with other observables and with multiple observables. Additional scaling work would also be useful, especially with respect to random variables: adding non-geometry-related uncertainty, such as in the viscosity or angle of attack, would provide insight into the method's generality. Finally, the method was limited to Gaussian variation, and extending the tools to function with non-Gaussian random variables would provide more realistic options for variation and uncertainty quantification.

References

[1] Aeschliman, D. P., and Oberkampf, W. L., "Experimental Methodology for Computational Fluid Dynamics Code Validation," Sandia National Labs., Rept. SAND95-1189, Albuquerque, NM, Sept. 1997.

[2] Hosder, S., Walters, R. W., and Perez, R., "A Non-Intrusive Polynomial Chaos Method for Uncertainty Propagation in CFD Simulations," AIAA Paper 2006-891, Jan. 2006.

[3] Walters, R. W., and Huyse, L., "Uncertainty Analysis for Fluid Mechanics with Applications," NASA CR-2002-211449, Feb. 2002.

[4] Karniadakis, G., Su, C., Xiu, D., Lucor, D., Schwab, C., and Todor, R., "Generalized Polynomial Chaos Solution for Differential Equations with Random Inputs," ETH Zurich, Rept. 2005-01, Zurich, Jan. 2005.

[5] Huyse, L., Padula, S. L., Lewis, R. M., and Li, W., "Probabilistic Approach to Free-Form Airfoil Shape Optimization Under Uncertainty," *AIAA Journal*, Vol. 40, No. 9, 2002, pp. 1764–1772. doi:10.2514/2.1881

[6] Garzon, V. E., and Darmofal, D. L., "Impact of Geometric Variability on Axial Compressor Performance," *International Gas Turbine and Aeroengine Congress and Exhibition*, Vol. 125, American Society of Mechanical Engineers, New York, Oct. 2003, pp. 692–703.

[7] Oberkampf, W. L., Helton, J. C., and Sentz, K., "Mathematical Representation of Uncertainty," AIAA Paper 2001-1645, Mar. 2001.

[8] Oberkampf, W., Blottner, F., and Aeschliman, D., "Methodology for Computational Fluid Dynamics Code Verification/Validation," AIAA Paper 95-2226, June 1995.

[9] *AIAA CFD Drag Prediction Workshop*, AIAA, Reston, VA, 2001.

[10] Huyse, L., and Walters, R. W., "Random Field Solutions Including Boundary Condition Uncertainty for the Steady-State Generalized Burgers Equation," NASA CR-2001-211239, Oct. 2001.

[11] Zang, C., Friswell, M., and Mottershead, J., "A Review of Robust Optimal Design and its Application in Dynamics," *Computers and Structures*, Vol. 83, Nos. 4–5, 2005, pp. 315–326. doi:10.1016/j.compstruc.2004.10.007

[12] Kumar, A., Keane, A. J., Nair, P. B., and Shahpar, S., "Robust Design of Compressor Fan Blades Against Erosion," *Journal of Mechanical Design*, Vol. 128, No. 4, 2006, pp. 864–873. doi:10.1115/1.2202886

[13] Molina-Cristobal, A., Parks, G. T., and Clarkson, P. J., "Finding Robust Solutions to Multi-Objective Optimization Problems Using Polynomial Chaos," *Proceedings of the 6th ASMO UK/ISSMO Conference on Engineering Design Optimization* [CD-ROM], Univ. of Leeds, Leeds, England, U.K., July 2006.

[14] Wiener, N., "The Homogeneous Chaos," *American Journal of Mathematics*, Vol. 60, No. 4, 1938, pp. 897–936. doi:10.2307/2371268

[15] Cameron, R. H., and Martin, W. T., "The Orthogonal Development of Non-Linear Functionals in Series of Fourier-Hermite Functionals," *Annals of Mathematics*, Vol. 48, No. 2, 1947, pp. 385–392.

[16] Ghanem, R. G., and Spanos, P. D., *Stochastic Finite Elements: A Spectral Approach*, Dover, Mineola, NY, 1991.

[17] Ghanem, R., "Ingredients for a General Purpose Stochastic Finite Elements Implementation," *Computer Methods in Applied Mechanics and Engineering*, Vol. 168, Nos. 1–4, 1999, pp. 19–34. doi:10.1016/S0045-7825(98)00106-6

[18] Ghanem, R., "Stochastic Finite Elements with Multiple Random Non-Gaussian Properties," *Journal of Engineering Mechanics*, Vol. 125, No. 1, Jan. 1999, pp. 26–40. doi:10.1061/(ASCE)0733-9399(1999)125:1(26)

[19] Xiu, D., Lucor, D., Su, C., and Karniadakis, G. E., "Stochastic Modeling of Flow-Structure Interactions Using Generalized Polynomial Chaos," *Journal of Fluids Engineering*, Vol. 124, No. 1, 2002, pp. 51–59. doi:10.1115/1.1436089

[20] Xiu, D., and Karniadakis, G. E., "The Wiener-Askey Polynomial Chaos for Stochastic Differential Equations," *Journal of Scientific Computing*, Vol. 24, No. 2, 2002, pp. 619–644. doi:10.1137/S1064827501387826

[21] Xiu, D., and Karniadakis, G. E., "Modeling Uncertainty in Steady State Diffusion Problems via Generalized Polynomial Chaos," *Computer Methods in Applied Mechanics and Engineering*, Vol. 191, No. 43, 2002, pp. 4927–4948. doi:10.1016/S0045-7825(02)00421-8

[22] Xiu, D., and Karniadakis, G. E., "Modeling Uncertainty in Flow Simulations via Generalized Polynomial Chaos," *Journal of Computational Physics*, Vol. 187, No. 1, 2003, pp. 137–167. doi:10.1016/S0021-9991(03)00092-5

[23] Knio, O., and Le Maître, O., "Uncertainty Propagation in CFD Using Polynomial Chaos Decomposition," *Fluid Dynamics Research*,

Vol. 38, No. 9, 2006, pp. 616–640.
doi:10.1016/j.fluiddyn.2005.12.003

[24] Le Maître, O. P., Knio, O. M., Najm, H. N., and Ghanem, R. G., "A Stochastic Projection Method for Fluid Flow I: Basic Formulation," *Journal of Computational Physics*, Vol. 173, No. 2, 2001, pp. 481–511. doi:10.1006/jcph.2001.6889

[25] Mathelin, L., Hussaini, M. Y., Zang, T. A., and Bataille, F., "Uncertainty Propagation for a Turbulent, Compressible Nozzle Flow Using Stochastic Methods," *AIAA Journal*, Vol. 42, No. 8, 2004, pp. 1669–1675. doi:10.2514/1.5674

[26] Walters, R. W., "Towards Stochastic Fluid Mechanics via Polynomial Chaos," AIAA Paper 2003-0413, Jan. 2003.

[27] Perez, R., and Walters, R. W., "An Implicit Compact Polynomial Chaos Formulation for the Euler Equations," AIAA Paper 2005-1406, Jan. 2005.

[28] Emery, A., "Generating Simulated Responses for Stochastic Systems Using Polynomial Chaos and Wick Products," *Inverse Problems, Design and Optimization Symposium* [CD-ROM], Federal Univ. of Rio de Janeiro, Rio de Janeiro, Brazil, 2004.

[29] Keane, A. J., and Nair, P. B., *Computational Approaches for Aerospace Design: The Pursuit of Excellence*, Wiley, West Sussex, England, U.K., 2005.

[30] Lucor, D., and Karniadakis, G. E., "Adaptive Generalized Polynomial Chaos for Nonlinear Random Oscillators," *Journal of Scientific Computing*, Vol. 26, No. 2, 2004, pp. 720–735. doi:10.1137/S1064827503427984

[31] Xiu, D., and Hesthaven, J. S., "High-Order Collocation Methods for Differential Equations with Random Inputs," *SIAM Journal on Scientific Computing*, Vol. 27, No. 3, 2005, pp. 1118–1139. doi:10.1137/040615201

[32] Loeven, A., Witteveen, J., and Bijl, H., "Efficient Uncertainty Quantification Using a Two-Step Approach with Chaos Collocation," *European Conference on Computational Fluid Dynamics*, Egmond aan Zee, The Netherlands, 2006, http://proceedings.fyper.com/ecco-mascfd2006/ [retrieved 8 November 2008].

[33] Mathelin, L., and Hussaini, M. Y., "A Stochastic Collocation Algorithm for Uncertainty Analysis," NASA CF-2003-212153, Feb. 2003.

[34] Wan, X., and Karniadakis, G. E., "Long-Term Behavior of Polynomial Chaos in Stochastic Flow Simulations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, Nos. 41–43, 2006, pp. 5582–5596. doi:10.1016/j.cma.2005.10.016

[35] Debusschere, B. J., Najm, H. N., Pebay, P. P., Knio, O. M., Ghanem, R. G., and Le Maître, O. P., "Numerical Challenges in the Use of Polynomial Chaos Representations for Stochastic Processes," *Journal of Scientific Computing*, Vol. 26, No. 2, 2004, pp. 698–719. doi:10.1137/S1064827503427741

[36] Le Maître, O. P., Reagan, M. T., Najm, H. N., Ghanem, R. G., and Knio, O. M., "A Stochastic Projection Method for Fluid Flow 2: Random Processes," *Journal of Computational Physics*, Vol. 181, No. 1, 2002, pp. 9–44. doi:10.1006/jcph.2002.7104

[37] Abramowitz, M., and Stegun, I. A. (ed.), *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*, Dover, Mineola, NY, 1965.

[38] Drela, M., "XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils," *Low Reynolds Number Aerodynamics*, edited by T. J. Mueller, Lecture Notes in Engineering, Vol. 54, Springer–Verlag, New York, 1989, pp. 1–12.

[39] Robic, T., and Filipic, B., "DEMO: Differential Evolution for Multiobjective Optimization," *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, Springer, New York, 2005, pp. 520–533.

[40] Sederberg, T. W., and Parry, S. R., "Free-Form Deformation of Solid Geometric Models," *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, Association for Computing Machinery, New York, 1986, pp. 18–22.

[41] Price, K., and Storn, R., "Differential Evolution—A Simple Evolution Strategy for Fast Optimization," *Dr. Dobb's Journal*, Vol. 22, No. 4, 1997, pp. 18–24.