Fall 2011

Inherent and model-form uncertainty analysis for CFD simulation of synthetic jet actuators

Daoru Frank Han
Missouri University of Science and Technology, handao@mst.edu

Department:

INHERENT AND MODEL-FORM UNCERTAINTY ANALYSIS FOR CFD SIMULATION OF SYNTHETIC JET ACTUATORS

by

DAORU HAN

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN AEROSPACE ENGINEERING

2011

Approved by

Dr. Serhat Hosder, Advisor
Dr. David W. Riggins
Dr. Xiaoping Du

ABSTRACT

A mixed aleatory (inherent) and epistemic (model-form) uncertainty quantification (UQ) analysis method was applied to a computational fluid dynamics (CFD) modeling problem of synthetic jet actuators. A test case, (Case 3, flow over a hump model with synthetic jet actuator control) from the CFDVAL2004 workshop was selected to apply the Second-Order Probability framework implemented with a stochastic response surface obtained from Quadrature-Based Non-Intrusive Polynomial Chaos (NIPC). Three uncertainty sources were considered: (1) epistemic uncertainty in turbulence model, (2) aleatory uncertainty in free stream velocity and (3) aleatory uncertainty in actuation frequency. Uncertainties in both long-time averaged and phase averaged quantities were quantified using a fourth order polynomial chaos expansion (PCE). Results were compared with experimental data available. A global sensitivity analysis with Sobol indices was utilized to rank the importance of each uncertainty source to the overall output uncertainty. The results indicated that for the long-time averaged separation bubble size, the uncertainty in turbulence model had a dominant contribution, which was also observed in the long-time averaged skin friction coefficients at three selected locations. For long-time averaged pressure coefficient, the contributions from free stream velocity and turbulence model are depending on the locations. The mixed UQ results for phase averaged x-velocity distributions at three selected locations showed that the 95% confidence intervals (CI) could generally envelope the experimental data. The Sobol indices showed that near the wall, the turbulence model had a main influence on the x-velocity, while approaching the main stream, the uncertainty in free stream velocity became a larger contributor. The uncertainty in frequency was found to have a very small contribution to both long-time averaged and phase averaged quantities with the range of variance considered.

ACKNOWLEDGMENT

I would like to express my thanks to my advisor, Dr. Serhat Hosder, for his supervision and guide on my study and research project, in addition to his help in the courses of Introduction to Hypersonic Flow and Applied Numerical Methods. The knowledge and inspiration I gained from the thought-provoking trainings will be of great help in my future endeavors.

I would also like to thank my degree committee members, Dr. David Riggins and Dr. Xiaoping Du for their time and comments on reviewing my thesis manuscript. Also, I want to thank Dr. David Riggins for his magnificent course of Aerospace Propulsion System.

I also want to thank Dr. Arindam Banerjee and Dr. Joshua L. Rovey for their helpful instruction in the courses of Viscous Fluid Flow and Gas Dynamics I, respectively, and their efforts and time in supporting recommendation letters for my academic future.

I would like to acknowledge the University of Missouri Research Board for providing financial support of a Graduate Research Assistantship to my project. In addition, I would also take this opportunity to thank the people in the Department of Mechanical and Aerospace Engineering for a valuable experience during my Master's study. I want to thank Dr. Lokeswarappa R. Dharani, Dr. Hai-Lung Tsai, Dr. Kelly Homan, Dr. Zhi Liang and Dr. Yukun Han for their help in my study and especially I want to thank Dr. Daniel S. Stutts for the opportunity of a Graduate Teaching Assistantship in his Mechanical Systems Laboratory.

Furthermore, I would like to thank Benjamin Robert Bettis, Srikanth Adya, Yi Zhang, Jocelyn Briana Bailey and Thomas Kelsey West IV for all of their support and inspiring talks while working in the Computational Fluid Dynamics and Aerospace

Design Laboratory. Especially, I want to thank Benjamin Robert Bettis for his helpful suggestions in formatting my thesis manuscript.

In addition, I would deliver my appreciation to all my friends in Rolla, with special thanks to Dr. Xiaoming He, Jing Hu, Xinsheng Zhang for their great support and encouragement during a hard time in my life.

Finally, I want to thank my family and Dr. Charles Zhou, whose love and support make this work possible.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

NOMENCLATURE

Symbol	Description
c	Characteristic reference length
C_f	Skin friction coefficient
C_p	Pressure coefficient
D	Statistical variance
f	Frequency (Hz)
K	Factor in S-A turbulence model
n	Number of random variables
N_t	Number of output modes
p	Pressure (N/m^2) or Order of polynomial chaos expansion
ref	Reference condition (subscript)
S	Sobol index
S_T	Total Sobol index
U	Velocity in x-direction (m/s)
V	Velocity in y-direction (m/s)
w	Wall condition (subscript)
α	Spectral modes
α^*	Stochastic output variable
μ_t	Turbulent (eddy) viscosity ($kg/(m{\cdot}s)$)
ξ	Standard random variable
$\vec{\xi}_a$	Standard aleatory random variable(s)
$\vec{\xi}_e$	Standard epistemic random variable(s)
τ	Shear stress (N/m^2)
Ψ	Random basis function
∞	Free stream condition (subscript)

1. INTRODUCTION

In this section, the brief working principles of synthetic jet actuators are introduced, followed by the descriptions of the motivation, objectives and contributions of the current research project. The outline of this thesis manuscript is given at the end of this section.

1.1. SYNTHETIC JET ACTUATORS

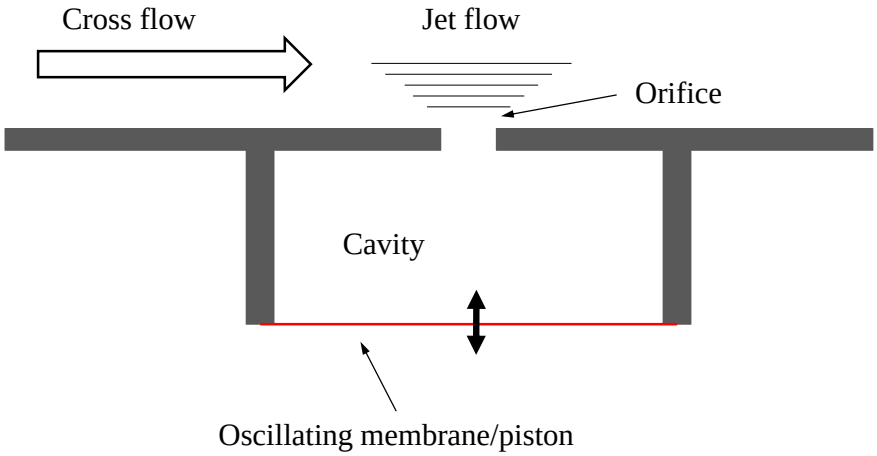

Figure 1.1. Working Principle of Synthetic Jet Actuators.

Synthetic jet actuators are among the most frequently studied active flow control devices. In general configurations (Figure 1.1), a synthetic jet device employs a moving membrane or piston to produce favorable changes in the flow field such as to delay or advance transition, to suppress or enhance turbulence or to prevent or provoke flow separation depending on the applications, e.g., drag reduction, lift enhancement or mixing augmentation [1]. A unique aspect of synthetic jet is that it is

formed by the working fluid from the flow field where it is employed hence eliminating the need of additional duct systems. Based on this characteristic, synthetic jet is also referred as "zero net-mass flux" or "zero efflux" jet. During the blowing period of the jet, the flow separates at the edges of the orifice mounted at the control surface, issuing into the surrounding regions. In a flow field with cross flow (main stream), the vortices can convect downstream entraining fluid from the main stream region. This interaction between periodically ejected jet and main stream flow can affect the velocity and/or pressure distributions of surrounding regions especially near the wall, resulting in favorable changes in the overall flow field characteristics. A detailed review of the recent works in synthetic jet flows, both experimentally and numerically, can be found in Glezer and Amitay [2].

1.2. MOTIVATION FOR UNCERTAINTY QUANTIFICATION

Among all the investigations in the flow phenomenon involved in a synthetic jet application, computational fluid dynamics (CFD) simulations are becoming more and more important with the aim of accurately predicting the flow field quantities and being able to perform robust and reliable designs. In order to assess the state-of-the-art CFD modeling of synthetic jet flows, a validation workshop [3] (referred as "CFDVAL2004" in the following of this manuscript) was held in 2004 where several synthetic jet configurations were selected as test cases. Summary of the workshop results and conclusions can be found in Rumsey et al. [4]. One of the conclusions was that, due to the unknown uncertainties in the modeling of the configuration, the CFD results failed to consistently agree with the experiments [5]. The time-dependent flow field quantities, such as phase averaged velocities, as well as the long-time averaged quantities can be affected by the uncertainties in the modeling of the unsteady boundary condition and physical models (e.g., turbulence model). In addition, the flow field can be also affected by the operating conditions such as main

stream velocity in the presence of a cross flow. All the uncertainties in the parameters of the CFD modeling motivate an uncertainty quantification (UQ) study.

1.3. OBJECTIVES OF THE CURRENT STUDY

One of the objectives of this study is to introduce and demonstrate an efficient methodology for the quantification of uncertainties and global non-linear sensitivity analysis in CFD modeling of synthetic jet actuators, which include both inherent (aleatory) and model-form (epistemic) uncertainty sources. Another unique aspect of the current study is to quantify and rank the contribution of each uncertainty source to the overall uncertainty in a selected output quantity.

The Second-Order Probability Theory can be used to propagate mixed (aleatory and epistemic) uncertainties through a simulation code. However, numerical computations can be intensive for this particular method due to the use of nested loops, especially if the simulation code is expensive to evaluate (such as a high-fidelity CFD code). To address this issue, a more efficient approach to Second-Order Probability is described and utilized in this work. In particular, a stochastic response surface which is obtained using a Non-Intrusive Polynomial Chaos (NIPC) method (Hosder et al. [6]) is utilized in the Second-Order Probability framework. The stochastic response surface is a surrogate model for the original simulation code, and is computationally less expensive to evaluate. Therefore, the utilization of the stochastic response surface, formulated with NIPC methods, enables the propagation of mixed uncertainties through the simulation code with much less computational cost compared to the expense of a traditional direct sampling approach (e.g., Monte Carlo method).

In the current work, a synthetic jet issued into a cross flow over a two-dimensional wall-mounted hump-shaped body (Case 3 of CFDVAL2004) is selected as the CFD modeling problem with both epistemic and aleatory uncertainties. Three uncertainty

variables are considered: turbulent (eddy) viscosity coefficient obtained from the turbulence model (epistemic or model-form uncertainty), free stream velocity (aleatory or inherent uncertainty), and the frequency used in the unsteady velocity-inlet boundary condition imposed at the bottom of the synthetic jet actuator cavity (aleatory or inherent uncertainty), which represents the variations in the frequency of the moving piston. Both aleatory uncertain inputs are described with uniform probability distributions, whereas the uncertainty in the turbulent viscosity is represented with an interval due to its epistemic nature. The quantities of interest for uncertainty quantification in the CFD simulations include the long-time averaged separation bubble characteristics, pressure and skin friction coefficients, as well as the phase averaged x-velocity distributions at selected locations. A previous UQ study on the same synthetic jet case (Adya et al. [7]) treated the free stream velocity as an aleatory uncertain variable with uniform distribution, showing that the phase averaged velocity were obviously affected by this uncertain input, however, the long-time averaged separation bubble size was found to be insensitive to the uncertainty in free stream velocity. The present work is built upon the experience obtained from that study, with a more comprehensive treatment of uncertainty sources including the turbulence model.

1.4. CONTRIBUTIONS OF THE CURRENT STUDY

Rcently, Bettis [8] and Adya [9] conducted research projects applying the NIPC uncertainty analysis approach to the CFD problems of hypersonic re-entry flows and synthetic jet into quiescent air, respectively. Compared to the latter project, the current study has contributions in two main topics:

Firstly, this study extends the synthetic jet flow configuration to the presence of a cross flow, introducing both long-time averaged and phase averaged quantities in the flow field.

Secondly and more importantly, this study is the first to treat both model-form (epistemic) and inherent (aleatory) uncertainty sources in the stochastic CFD problem of synthetic jet flow simulations. It should be noted that the term "model-form" in the title of this thesis reflects the uncertainty in the CFD modeling (i.e., turbulence model), while the inherent (aleatory) uncertainty variables represent the input uncertainties of the CFD simulation.

Furthermore, a non-linear global sensitivity analysis method, Sobol indices, is utilized in this study to rank the importance of each uncertainty source to the overall output uncertainty of a specific quantity.

1.5. THESIS OUTLINE

This thesis manuscript is composed of six main sections. Following the first main section of introduction, the second main section is a literature review study describing relevant work conducted on the topics of mixed uncertainty quantification methods and CFD modeling of synthetic jet flows.

The third main section describes the mixed aleatory and epistemic uncertainty propagation approach employed in this study, especially the Second-Order Probability framework and Quadrature-Based Non-Intrusive Polynomial Chaos methods. The method of global sensitivity analysis using Sobol indices is also presented.

In the fourth main section, the computational model of synthetic jet flows and the description of the stochastic CFD problem are outlined. The definitions of long-time averaged and phase averaged quantities to perform uncertainty analysis are described as well.

The uncertainty results obtained from the stochastic CFD problem are presented in the fifth main section. The global sensitivity analysis is also performed showing the relevant importance of each uncertainty variable to the overall uncertainty of each output quantity.

Finally, the conclusions and discussions on future work are given in the sixth main section.

The CFD simulation setup procedure and the main part of the post-processing source code are attached in the appendices.

2. LITERATURE REVIEW

This literature review includes two subsections. The first subsection is a review of recent studies on mixed aleatory-epistemic uncertainty quantification analysis applied to both model problems and computational simulations. The second subsection is mainly a review on the participations of the CFDVAL2004 workshop and some preliminary uncertainty studies derived from the results of the workshop.

2.1. MIXED UNCERTAINTY QUANTIFICATION

Among several investigations on the topic of propagating mixed aleatory-epistemic uncertainties through a simulation, one study conducted by Eldred et al. [10] provided an extensive summary of efficient algorithms for mixed aleatory and epistemic uncertainty quantification analysis. They proposed using Second-Order Probability for quantifying the effects of mixed uncertainties, which particularly separated the aleatory and epistemic input uncertainties into an inner and outer sampling loop, respectively. Hence it became easy to identify the overall uncertainty due to both aleatory and epistemic input uncertainties. Furthermore, they also applied this method to a problem of plastic analysis of a short column which was represented as a simple analytical function, which represented an ideal test case due to the analytical and inexpensive fact of the evaluation. Therefore, this study provided an analytical benchmark for validating simulation code used for mixed aleatory-epistemic uncertainty quantification.

Swiler et al. [11] also performed a similar study describing the method of using Second-Order Probability for mixed uncertainty quantification. They provided several convenient and helpful diagrams about the Second-Order Probability framework, and also applied the method to a simple model problem to validate simulation code.

Recently, Bettis et al. [12], [13] applied the Second-Order Probability method to the mixed aleatory-epistemic uncertainty analysis of a hypersonic flow problem. Particularly, they constructed a stochastic response surface to represent the simulation due to the computational expenses. They also performed the Second-Order Probability method to a model problem originating from the physics of hypersonic flows. These studies showed the effectiveness and efficiency of propagating mixed uncertainties using stochastic response surface implemented to the Second-Order Probability framework.

2.2. CFD MODELING OF SYNTHETIC JET FLOWS

As mentioned previously in the Introduction section, a number of investigations have been spent on the flow phenomenon of synthetic jet, both experimentally and computationally. Especially, a dedicated workshop (CFDVAL2004 [3]) was held in March 2004 at NASA Langley Research Center to assess the state-of-the-art CFD modeling of synthetic jet flows. Three configurations were selected as test cases: synthetic jet into quiescent air (Case 1), synthetic jet into a cross flow (Case 2) and flow over a hump model with synthetic jet actuator control (Case 3). A detailed documentation can be found via the website of the CFDVAL2004 workshop [3].

Among the results and conclusions obtained from the workshop, one main lesson was that, the CFD results failed to consistently agree with the experimental measurements [5], due to the unknown uncertainties in the modeling of the configuration (e.g., boundary conditions, physical models). A study conducted by Rumsey [14] involving Case 3 of the CFDVAL2004 workshop also introduced a new uncertainty source in the turbulence model when trying to match the separation bubble size with the experiment.

The uncertainty sources considered in the current study are mainly motivated by the challenges faced in the CFD modeling of synthetic jet flows. Case 3 of the

CFDVAL2004 workshop is selected as the test case due to the computational expenses (two-dimensional) and the available experimental data of quantities of interest. A previous study by Adya et al. [7] treated the free stream velocity as an aleatory uncertain variable with uniform distribution, showing that the phase averaged velocity distribution were obviously affected by this uncertain input, however, the long-time averaged separation bubble size was found to be insensitive to the uncertainty in free stream velocity. The current study is an extension of that work, in addition, the actuation frequency (boundary condition) and turbulent viscosity (physical model) are also treated as uncertainty sources.

3. UNCERTAINTY QUANTIFICATION APPROACH

In this section, types of uncertainties are briefly discussed and the method of propagating mixed aleatory and epistemic uncertainties are described.

3.1. TYPES OF UNCERTAINTIES IN COMPUTATIONAL SIMULATIONS

Generally, there can be three types of uncertainty and error in a computational simulation: (1) aleatory (inherent) uncertainty, (2) epistemic (model-form) uncertainty and (3) numerical error. A detailed description of these uncertainties can be found in Oberkampf et al. [15].

Aleatory uncertainty, or inherent uncertainty, originates from the random nature of a physical system and thus can be mathematically represented by a probability density function (PDF) if knowledge is available to estimate the distribution type (uniform, normal, etc.). Such knowledge can be from the substantial experimental data, statistical study of the survey, etc. The uncertainty in free stream velocity or geometry can be treated as examples of aleatory uncertainties in a stochastic aerodynamics problem.

Epistemic uncertainty, also referred as model-from uncertainty, is due to ignorance, lack of knowledge or incomplete information of some characteristics from a non-deterministic system. By this feature, an increase in knowledge or understanding of a system can lead to a decrease in the epistemic uncertainty. Similar to aleatory uncertainty, epistemic uncertainty can be also modeled with probabilistic approach. However, studies have shown that this may cause inaccurate predictions in the amount of uncertainty in the responses (see [16] for details). Another method to treat the epistemic uncertain variables is to use intervals by giving the lower and upper bounds

with the information from limited experimental data or expert judgment or empirical model. In CFD modeling problems, the turbulence model utilized to solve the Reynolds-Averaged Navier-Stokes (RANS) equations can be treated as a source of epistemic uncertainty.

Numerical error is from the deficiency in the modeling of the simulation. Different from epistemic uncertainty, numerical error is recognizable. Usually the discretization error in spatial or temporal domain when solving governing partial differential equations (PDEs) of a physical model on a computational mesh can be treated as an example of numerical error.

3.2. MIXED UNCERTAINTY PROPAGATION

In this study, Second-Order Probability framework implemented with Non-Intrusive Polynomial Chaos is employed to propagate the mixed uncertainties. The global sensitivity analysis method using Sobol indices is also described in this subsection.

3.2.1. Second-Order Probability. In the current work, Second-Order Probability [10], [17] is employed to propagate mixed aleatory and epistemic uncertainty through CFD simulations. As described in Bettis et al. [13], Second-Order Probability approach uses an outer loop where a specific value of the epistemic variable is selected and an inner loop where any traditional aleatory uncertainty quantification method can be performed for uncertainty analysis with each specific value of the epistemic variable. Each iteration of the outer loop will produce a cumulative distribution function (CDF) based on the aleatory uncertainty analysis in the inner loop. So the Second-Order Probability approach will give interval bounds of the output at different probability levels. Since the epistemic and aleatory variables are treated in different loops, it is easy to separate and identify each of them from the output horsetail plots. However, this method can be relatively computationally expensive

due to the two sampling loops especially when traditional sampling approach such as Monte Carlo is used.

In this study, a response surface (function of both epistemic and aleatory variables) of quantity of interest is first obtained using a Quadrature-Based NIPC method (described below). Then the Second-Order Probability is employed by sampling the epistemic uncertain variables in the outer loop and then sampling the aleatory uncertain variables in the inner loop (with fixed values of epistemic uncertain variables) and then evaluating the samples using the obtained response surface approximation. The procedure is shown in Figure 3.1.

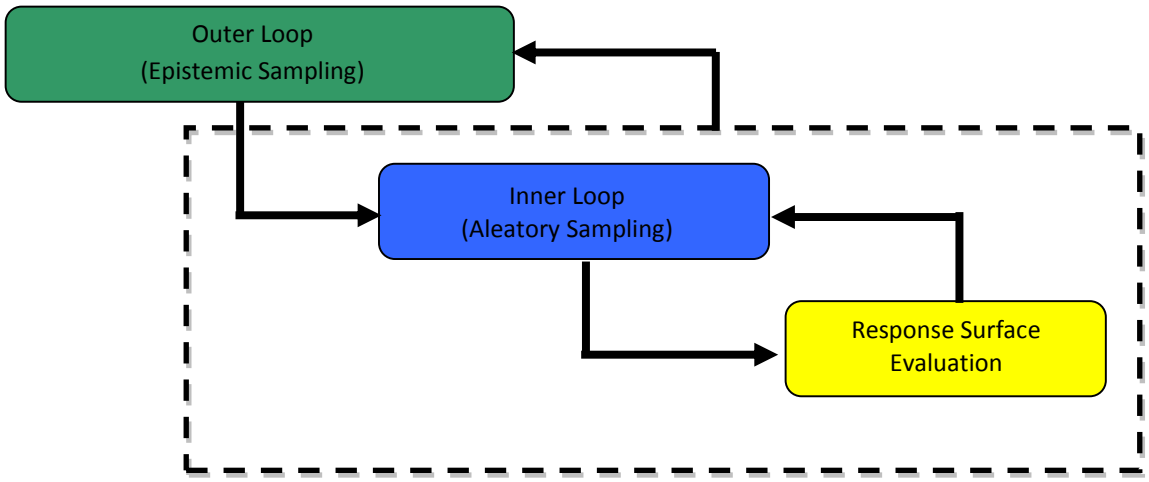

Figure 3.1. Schematic of Second-Order Probability.

3.2.2. Basics of Polynomial Chaos. In the current study, the Quadrature-Based Non-Intrusive Polynomial Chaos is employed which is derived from the polynomial chaos theory based on the spectral representation of the uncertainty. As an important aspect of spectral representation of uncertainty, one can decompose a random function or variable into separable deterministic and stochastic components as

shown in Equation (1), where α^* can be any random variable of interest such as long-time averaged or phase averaged velocity, pressure or skin friction coefficient in a stochastic fluid dynamics problem.

$$\alpha^*(t, \vec{x}, \vec{\xi}) \approx \sum_{j=0}^{P} \alpha_j(t, \vec{x}) \Psi_j(\vec{\xi}) \tag{1}$$

In the equation above, $\alpha_j(t, \vec{x})$ is the deterministic component and $\Psi_j(\vec{\xi})$ is the random basis function corresponding to the j^{th} mode. Here α^* is assumed to be a function of the independent deterministic variable vector $(t, \vec{x})$ and the n-dimensional random variable vector $\vec{\xi} = (\xi_1, ..., \xi_n)$, which can include both aleatory and epistemic uncertain variables. In theory, the polynomial chaos expansion (PCE) given by Equation (1) should have infinite number of terms, however in practice a discrete summation is taken over a finite number of output modes. For a total order expansion, the number of output modes (N_t) is given by,

$$N_t = P + 1 = \frac{(n+p)!}{n!p!} \tag{2}$$

which is a function of the order of PCE (p) and the number of random dimensions (n). Ideally, the basis function takes the form of multi-dimensional Hermite Polynomial to span the n-dimensional random space when the input uncertainty is Gaussian (unbounded), which was first introduced by Wiener [18] in his original work of polynomial chaos. To extend the application of the polynomial chaos theory to the propagation of continuous non-normal-distributed input uncertainties, Xiu and Karniadakis [19] used a set of polynomials known as the *Askey scheme* to obtain the *Wiener-Askey Generalized Polynomial Chaos*. The weight and density functions for several of the most commonly studied polynomials are shown in Table 3.1.

As shown in a study by Huyse et al. [20], the Hermite, Legendre and Laguerre polynomials are the optimal basis functions, in terms of the convergence of statistics,

Table 3.1. Density and Weight Functions of Several Commonly Used Univariate Optimal Basis Functions.

Input Distribution	Density Function	Polynomial Name	Weight Function	Support Range (R)
Normal	$\frac{1}{\sqrt{2\pi}}e^{\frac{-\xi^2}{2}}$	Hermite, $H_n(\xi)$	$e^{\frac{-\xi^2}{2}}$	$[-\infty, \infty]$
Uniform	$\frac{1}{2}$	Legendre, $Le_n(\xi)$	1	$[-1, 1]$
Exponential	$e^{-\xi}$	Laguerre, $La_n(\xi)$	$e^{-\xi}$	$[0, \infty]$

for input uncertainty variables having Gaussian (normal), bounded (uniform) and semi-bounded (exponential) distributions, respectively. The optimal basis functions are derived upon the inner product of the weight functions corresponding to the standard probability density functions (PDF) of a given input uncertainty variable. The standard PDF must meet the requirement that the integral of the PDF over the support range is exactly one. A direct result of this requirement is the constant multiplicative factor between the weight function and density function in Table 3.1.

For problems having more than one uncertainty variable, the multivariate basis functions can be obtained from the product of univariate orthogonal polynomials (see Eldred et al. [21] for details). For example, a multivariate Hermite polynomial can be constructed as,

$$H_n(\xi_{i_1}, \ldots, \xi_{i_n}) = H_n(\vec{\xi}) = e^{\frac{1}{2}\vec{\xi}^T\vec{\xi}}(-1)^n \frac{\partial^n}{\partial\xi_{i_1}\ldots\partial\xi_{i_n}} e^{-\frac{1}{2}\vec{\xi}^T\vec{\xi}} \tag{3}$$

which can also be obtained from one-dimensional Hermite Polynomials ($\psi_{m_i^j}(\xi_i)$) by

using the multi-index m_i^j, as shown in Equation (4):

$$H_n\left(\xi_{i_1},\ldots,\xi_{i_n}\right)=\Psi_j(\vec{\xi})=\prod_{i=1}^{n}\psi_{m_i^j}\left(\xi_i\right) \tag{4}$$

The main objective of the polynomial chaos method is to determine the coefficients $\alpha_j\left(t,\vec{x}\right)$ from Equation (1). Then, the statistics of the stochastic output can be calculated using these coefficients and the corresponding optimal basis functions. For example, Hosder et al. [6] showed that the mean value of a stochastic solution is given by,

$$\mu_{\alpha^*}=\bar{\alpha}^*\left(t,\vec{x}\right)=E_{PC}\left(\alpha^*(t,\vec{x},\vec{\xi})\right)=\int_R\alpha^*(t,\vec{x},\vec{\xi})p(\vec{\xi})d\vec{\xi}=\alpha_0\left(t,\vec{x}\right) \tag{5}$$

which demonstrates that the mean, or expected value, of the output $\alpha^*(t,\vec{x},\vec{\xi})$ is simply the zeroth coefficient (or mode) of the polynomial chaos expansion. Hosder et al. [6] also gave the expression for the variance of the output:

$$\sigma_{\alpha^*}^2=Var_{PC}\left[\alpha^*(t,\vec{x},\vec{\xi})\right]=\int_R\left(\alpha^*(t,\vec{x},\vec{\xi})-\bar{\alpha}_0^*(t,\vec{x})\right)^2p(\vec{\xi})d\vec{\xi}=\sum_{j=1}^{P}\left[\alpha_j^2\left\langle\Psi_j^2\right\rangle\right] \tag{6}$$

Equations (5) and (6) utilize the fact that $\langle\Psi_j\rangle=0$ for $j>0$ and $\langle\Psi_i\Psi_j\rangle=\left\langle\Psi_j^2\right\rangle\delta_{ij}$, where δ_{ij} is the Kronecker delta function. Furthermore, the inner product of $\Psi_i(\vec{\xi})$ and $\Psi_j(\vec{\xi})$ over the support range R is defined as:

$$\left\langle\Psi_i(\vec{\xi})\Psi_j(\vec{\xi})\right\rangle=\int_R\Psi_i(\vec{\xi})\Psi_j(\vec{\xi})p(\vec{\xi})d\vec{\xi} \tag{7}$$

where $p(\vec{\xi})$ is the density function.

If the probability distribution of each random variable is different, the optimal multivariate basis functions can be again obtained from the product of univariate orthogonal polynomials employing the optimal univariate polynomial at each random

dimension. In this approach, it is required that the input uncertainties are independent standard random variables, which also allows the calculation of the multivariate weight functions by taking the product of univariate weight functions associated with the probability distribution at each random dimension. More detailed information on polynomial chaos expansion can be found in Walters and Huyse [22], Najm [23], and Hosder and Walters [6].

Generally there are *intrusive* and *non-intrusive* approaches to model the uncertainty propagation in computational simulation via PCE. In the intrusive approach, all dependent variables and random parameters in the governing equations are replaced with their PCEs. Taking the inner product of the equations, or projecting each equation onto j^{th} basis, yields $P+1$ times the number of deterministic equations which can be solved by the same numerical methods applied to the original deterministic system (computational simulation). Although straightforward in theory, an intrusive formulation for complicated problems can be relatively difficult, expensive, and time-consuming to implement. To overcome such inconveniences associated with the intrusive approach, non-intrusive polynomial chaos (NIPC) formulations are considered for uncertainty propagation in this study.

The NIPC method uses spectral projection to find the polynomial coefficients $\alpha_k = \alpha_k\left(t, \vec{x}\right)$ in Equation (1). Projecting Equation (1) onto the k^{th} basis yields:

$$\left\langle \alpha^*(t, \vec{x}, \vec{\xi}), \Psi_k(\vec{\xi}) \right\rangle = \left\langle \sum_{j=0}^{P} \alpha_j(t, \vec{x}) \Psi_j(\vec{\xi}) \Psi_k(\vec{\xi}) \right\rangle \tag{8}$$

then, from the virtue of orthogonality,

$$\left\langle \alpha^*(t, \vec{x}, \vec{\xi}), \Psi_k(\vec{\xi}) \right\rangle = \alpha_k(t, \vec{x}) \left\langle \Psi_k^2(\vec{\xi}) \right\rangle \tag{9}$$

which leads to,

$$\alpha_k(t,\vec{x}) = \frac{\left\langle \alpha^*(t,\vec{x},\vec{\xi}), \Psi_k(\vec{\xi}) \right\rangle}{\left\langle \Psi_k^2(\vec{\xi}) \right\rangle} = \frac{1}{\left\langle \Psi_k^2(\vec{\xi}) \right\rangle} \int_R \alpha^*(t,\vec{x},\vec{\xi}) \Psi_k(\vec{\xi}) p(\vec{\xi}) \mathrm{d}\vec{\xi} \qquad (10)$$

The objective of the spectral projection method is to predict the polynomial coefficients by evaluating the numerator ($\left\langle \alpha^*(t,\vec{x},\vec{\xi}), \Psi_k(\vec{\xi}) \right\rangle$) in Equation (10), while the denominator ($\left\langle \Psi_k^2(\vec{\xi}) \right\rangle$) can be computed analytically for multivariate orthogonal polynomials. As described by Hosder et al. [6], there are three main NIPC methods: sampling-based, quadrature-based and point-collocation. The current study uses the quadrature-based approach to calculate the polynomial coefficients as will be further described.

3.2.3. Quadrature-Based Non-Intrusive Polynomial Chaos. In the quadrature-based NIPC approach, the multi-dimensional integral in the numerator of Equation (10) is evaluated with numerical quadrature [21] in the support range (R) where input uncertain variables are defined. For the integration of one-dimensional problems, the straightforward approach will be to use Gaussian quadrature points, which are zeros of the orthogonal polynomials that are optimal for the given input uncertainty distribution (e.g., Gauss-Hermite, Gauss-Legendre, and Gauss-Laguerre points for normal, uniform, and exponential distributions, respectively). The extension of this approach to multi-dimensional problems can be achieved via tensor product of one-dimensional quadrature formulas. In one-dimensional problems, a Gauss quadrature formula of n_p points will evaluate a polynomial of degree $2n_p - 1$ (or less) exactly and, the polynomial degree of the product of the function approximation and the basis in the integrand of the numerator in Equation (10) will be $2p$ for the evaluation of the coefficient of the highest degree if the degree of the PCE is chosen as p. Therefore, the minimum number of quadrature points required to exactly evaluate the integral will be $p + 1$. For stochastic problems with relatively small number of

input uncertain variables (i.e., $n \leq 4$), this approach will be computationally efficient compared to a typical Monte Carlo approach. However, for multi-dimensional problems with large number of uncertain variables, the computational expense may become significant due to its exponential growth with the number of random dimensions, since the required number of deterministic function evaluations will be $(p+1)^n$ for a stochastic problem with n random variables having the same degree of PCE (p) in each dimension. It is important to emphasize that the computational expense of propagating mixed input uncertainties can be high even if the number of aleatory and epistemic uncertain variables is not large when the "deterministic function evaluation" is actually CFD simulation. Therefore constructing and evaluating a stochastic (polynomial chaos) response surface will significantly reduce the required number of deterministic CFD simulations for the propagation of mixed uncertainties.

3.2.4. Implementation of NIPC in Second-Order Probability. As described previously, the current study utilizes Second-Order Probability approach to propagate the mixed aleatory-epistemic uncertainties. With this approach, the stochastic response (e.g., the long-time averaged separation bubble size) is represented with a PCE as a function of both aleatory and epistemic uncertain variables. The optimal basis functions are used for the aleatory variables while Legendre polynomials are used for the epistemic variables. It should be noted that the use of Legendre polynomials should not imply uniform probability distributions of the epistemic uncertain variables. This choice is made due to the bounded nature of epistemic uncertain variables considered in this study. Once the stochastic response surface is constructed, the stochastic responses can be evaluated for a large number of samples randomly produced based on the probability distributions of the aleatory input uncertainties (inner loop of Second-Order Probability) with fixed values of epistemic uncertain variables. Each iteration in the outer loop will produce a single cumulative distribution function (CDF). By repeating the inner loop for a large number of

epistemic uncertain variables sampled from their corresponding intervals (outer loop of Second-Order Probability), a population of CDFs can be obtained and plotted ("horsetail" plot) thus the bounds of the stochastic response at different probability levels as well as confidence intervals (CI) can be calculated. Figure 3.2 gives an example of a conservative calculation of 95% confidence interval from the horsetail plot.

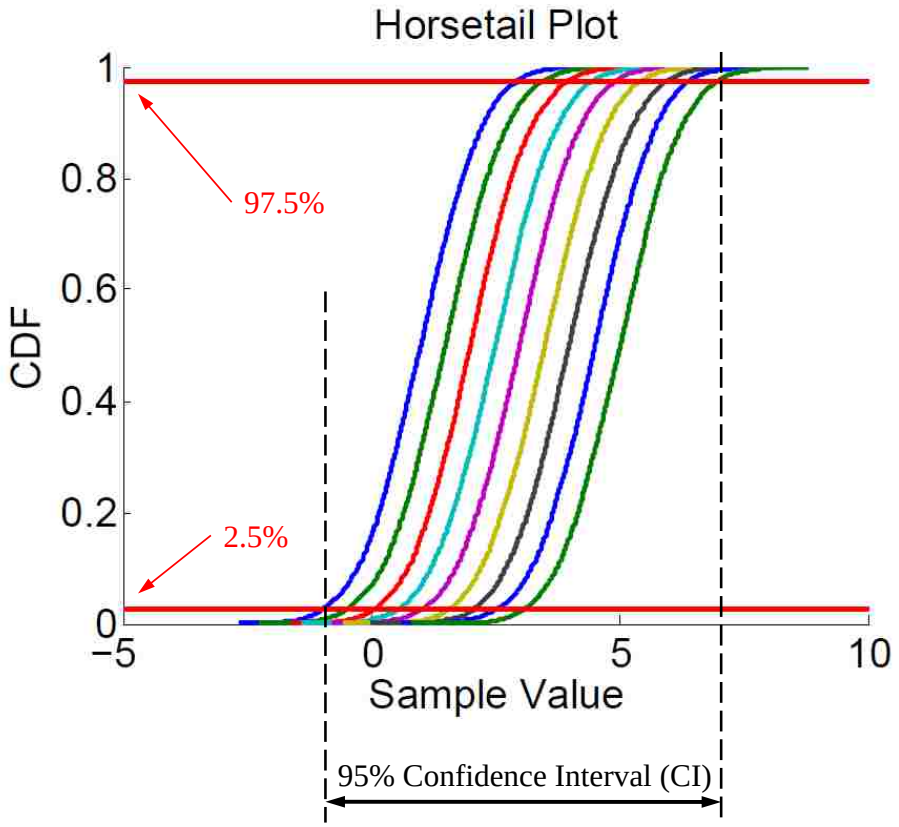

Figure 3.2. A Conservative Calculation of 95% Confidence Interval.

Due to the analytical nature (polynomials) of the stochastic response surface, the described procedure will be computationally efficient, especially compared to the direct Monte Carlo sampling which requires a large number of deterministic CFD

simulations. Figure 3.3 shows the flowchart of the entire procedure of mixed aleatory-epistemic uncertainty analysis employed in this study.

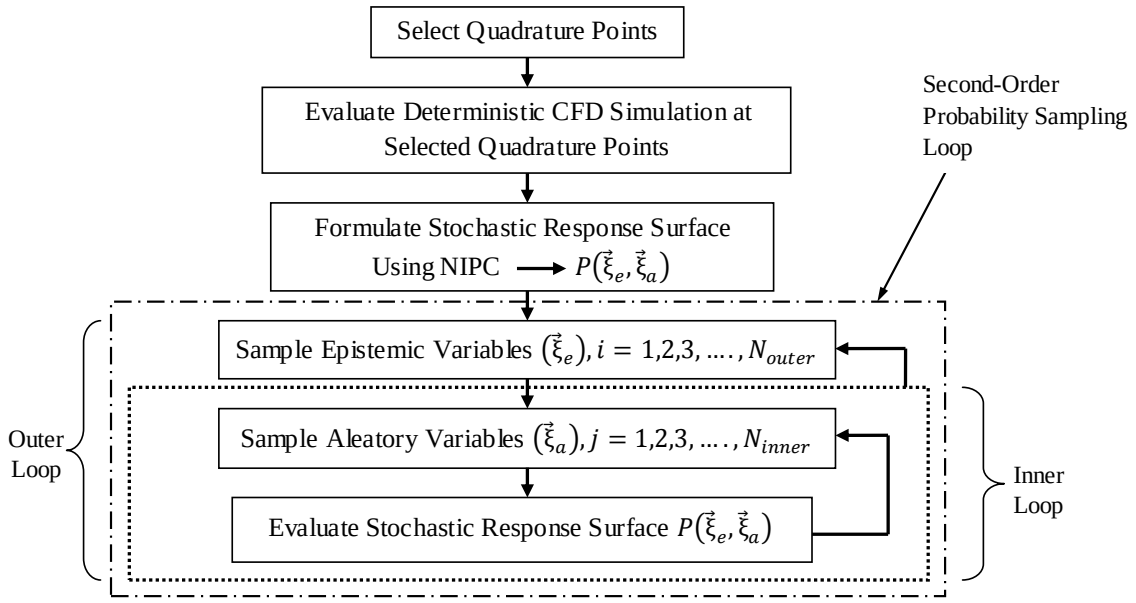

Figure 3.3. Flowchart Showing the Procedure of Propagating Mixed Aleatory-Epistemic Uncertainty with Second-Order Probability and Quadrature-Based NIPC.

3.2.5. Global Sensitivity Analysis with Sobol Indices. In a system where multiple uncertain variables are present, it is often useful to demonstrate and rank the relative importance of each input uncertain variable to the overall output quantity of interest using a global sensitivity analysis approach. In current study, Sobol [24] indices are used to perform the this analysis.

Once the PCE for an output uncertain quantity is formed, Sobol indices can be derived via *Sobol Decomposition* which is a variance-based global sensitivity analysis method. First, the total variance (D) can be written in terms of the PCE:

$$D = \sum_{j=1}^{P} \alpha_j^2(t, \vec{x}) \left\langle \Psi_j^2(\vec{\xi}) \right\rangle \tag{11}$$

Then, as shown by Sudret [25], Crestaux et al. [26], and Ghaffari et al. [27], the total variance can be decomposed as:

$$D = \sum_{i=1}^{i=n} D_i + \sum_{1 \le i < j \le n}^{i=n-1} D_{i,j} + \sum_{1 \le i < j < k \le n}^{i=n-2} D_{i,j,k} + \cdots + D_{1,2,\ldots,n} \tag{12}$$

where the partial variances ($D_{i_1,\ldots,i_s}$) are given by:

$$D_{i_1,\ldots,i_s} = \sum_{\beta \in \{i_1,\ldots,i_s\}} \alpha_\beta^2(t,\vec{x}) \left\langle \Psi_\beta^2(\vec{\xi}) \right\rangle, \qquad 1 \le i_1 < \ldots < i_s \le n \tag{13}$$

Then the Sobol indices ($S_{i_1 \cdots i_s}$) are defined as,

$$S_{i_1 \cdots i_s} = \frac{D_{i_1,\ldots,i_s}}{D} \tag{14}$$

which satisfy the following equation:

$$\sum_{i=1}^{i=n} S_i + \sum_{1 \le i < j \le n}^{i=n-1} S_{i,j} + \sum_{1 \le i < j < k \le n}^{i=n-2} S_{i,j,k} + \cdots + S_{1,2,\ldots,n} = 1.0 \tag{15}$$

The Sobol indices provide a sensitivity measure due to individual contribution from each input uncertain variable (S_i), as well as the mixed contributions ($\{S_{i,j}\}, \{S_{i,j,k}\}, \cdots$). As shown by Sudret [25] and Ghaffari et al. [27], the total (combined) effect (S_{T_i}) of an input parameter i is defined as the summation of the partial Sobol indices that include the particular parameter:

$$S_{T_i} = \sum_{L_i} \frac{D_{i_1,\ldots,i_s}}{D}; \quad L_i = \{(i_1,\ldots,i_s) : \exists\, k,\ 1 \le k \le s,\ i_k = i\} \tag{16}$$

For example, with $n = 3$, the total contribution to the overall uncertainty from

the first uncertain variable ($i = 1$) can be written as:

$$S_{T_1} = S_1 + S_{1,2} + S_{1,3} + S_{1,2,3} \tag{17}$$

From these formulations, it can be seen that the Sobol indices can be used to provide a relative ranking of each input uncertainty to the overall variation in the output with consideration of non-linear correlation between input variables and output quantities of interest. One of the goals of the current work is to calculate Sobol indices with PCE and then use them to rank the relative importance of each input uncertain variable to a specific output quantity of interest.

4. COMPUTATIONAL MODEL

In this section, the CFD modeling of the flow configuration is described as well as the stochastic CFD simulation problem. The definition and calculation of quantities of interest are also presented.

4.1. INTRODUCTION TO COMPUTATIONAL FLUID DYNAMICS

Computational fluid dynamics (CFD) is becoming more and more important in the design and analysis of aerospace systems. The main goal of CFD is to provide an accurate representation of flow field, and extract valuable flow field quantities (e.g., velocity, pressure, skin friction) at any location in the entire computational domain. A main advantage of CFD over experiment is that it is easy and inexpensive to perform, thus becoming a more and more popular tool to perform the aerospace design and analysis.

As the first step to numerically approximate the flow field over an arbitrary geometry in most of the CFD methods, the governing equations, in the form of highly coupled partial differential equations (PDEs), are to be discretized. The most general form of fluid dynamic governing equations are known as Navier-Stokes, and these equations are implemented into most of the modern CFD codes. A common numerical scheme for approximating the governing equations is the finite volume method, which is utilized in most CFD codes (such as ANSYS FLUENT). This particular method requires a discretized computational domain (mesh) having a sufficient amount of volume surrounding the geometry of interest. Depending on the geometry and/or capacities of the CFD code, either structured or unstructured mesh can be used. Once a computational mesh has been constructed, one must specify the boundary and initial conditions to be used in the simulation. Furthermore, in any type of process involving

the numerical approximation of the flow field, it is crucial to appropriately model all relevant boundary conditions for the problem at hand. For example, one may specify the surface of the vehicle to be an adiabatic "non-slip" wall boundary condition for many low speed aerodynamic applications (such as the synthetic jet flow considered in this study). Also, the user should specify the most appropriate physical models (such as turbulence model) in the simulation. Lastly, the user should specify all relevant methods to be used in the numerical scheme utilized within the CFD code. These may includes things such as inviscid flux modeling, limiters, and parallel computing options. In the time-dependent flow field simulation, a time domain scheme is also required to advance the solution in time.

To model the turbulence nature of the flow considered in this study, turbulence models are needed to solve the Navier-Stokes equations. Generally, there are three commonly used methods to model the turbulence: (1) Direct Numerical Simulation (DNS), (2) Large Eddy Simulation (LES) and (3) Reynolds-Averaged Navier-Stokes (RANS). The DNS method resolves the entire range of turbulent length scales thus is extremely expensive to implement. The LES method only resolves the largest and most important turbulent scales by filtering smaller scales whose effect is modeled using sub-grid scale models. This method is computationally cheaper than DNS but still more expensive than RANS method, in which the Reynolds stresses are introduced to solve the governing equations (Navier-Stokes). In RANS method with Boussinesq hypothesis, the Reynolds stresses can be obtained from an algebraic equation determining the turbulent viscosity. There are several models calculating the turbulent viscosity, depending on the sophistication of the models, such as zero-equation models (e.g., Mixing Length Model) and two-equation models (e.g., k-ϵ models). In this study, one commonly used one-equation model (Spalart-Allmaras [28]) is used in the CFD simulation, with a consideration of the use of different turbulence models (see Section 4.3 for details).

To ensure the CFD simulation produces accurate results, one must ensure that both the physical modeling error and discretization error are kept at a minimum. The physical modeling errors can be minimized by selecting the most appropriate models, while the discretization error can be minimized via a grid convergence study since it is directly related to the grid density of the computational mesh used within the CFD simulation. It is a good practice to perform grid convergence studies to ensure that the CFD solution is independent to the grid size so the discretization error is kept at a minimum. By doing these studies, the accuracy of CFD results can be increased for the design and analysis of complex aerospace systems and compared to the experiments.

4.2. CFD SIMULATIONS

4.2.1. Physical Models and Geometry. The synthetic jet configuration studied in this paper is flow over a two-dimensional wall-mounted hump-shaped body which is labeled as "hump model (Case 3)" in CFDVAL2004 workshop [3].

Figure 4.1 shows the experimental configuration [3] of the hump model which is mounted between two glass endplate frames. The width of the tunnel test section is 28" and the nominal test section height is 15.032". As the workshop indicated, this experiment was nominally two-dimensional except the side wall effects near the endplates.

The characteristic reference length of the model is defined as the length of the bump on the wall which is 16.536". The model itself is 23" wide between the endplates at both sides and 2.116" high at its maximum thickness point. All the experiment test flows were considered under the free stream conditions of Mach = 0.1 at a Reynolds number of 9.36×10^5. The model experienced a fully-developed turbulent boundary layer during the test, which separated over the concave section in the aft part of the hump body. A slot opening was located at approximately 65%

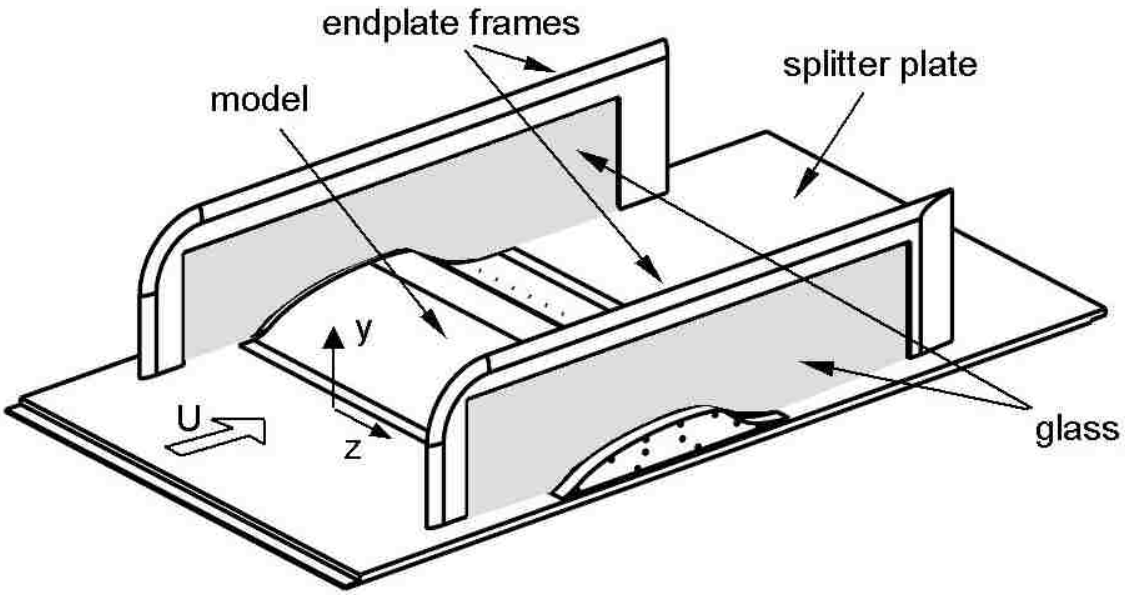

Figure 4.1. Experimental Configuration.

chord station, extending across the entire span of the hump. In the oscillatory part of the experiment, the two-dimensional oscillatory blowing was achieved by means of a rigid piston spanning the model with a actuation frequency of 138.5 Hz. More detailed test conditions are documented on the website of CFDVAL2004 workshop [3].

4.2.2. Numerical Scheme and Computational Grid. The commercial CFD software, ANSYS FLUENT 12 [29], is used for the simulation of the flow field. The unsteady Reynolds-Averaged Navier-Stokes (RANS) equations coupled with Spalart-Allmaras [28] turbulence model are solved to compute the unsteady, turbulent, two-dimensional flow field including the cavity and main flow region. Periodic solutions are obtained to calculate the phase averaged and long-time averaged quantities in the flow field. A second-order accurate implicit time-integration scheme is used to advance the solution in time. The inviscid fluxes are approximated with a second-order upwind scheme in space and the viscous terms are approximated with second-order central differencing. The general setup procedure for ANSYS FLUENT is given in Appendix A.

The grid employed in this paper is labeled as "STRUCTURED 2D GRID #4" on the workshop website (210,060 grid points total), where top wall shape is adjusted to approximately account for the side plate blockage effect. In this grid, the computational domain extends upstream to -6.39 chord length which yields a "run" long enough to get the approximate boundary layer thickness matching experimental data. The internal slot and cavity are also included in the grid. Figure 4.2 shows the local zoom-in view of the grid near the slot [3].

To get the time-accurate solution, 360 time steps per period (time step size of order 2×10^{-5} seconds) are used with 20 inner iterations per time step [14]. The choice of 360 time steps also makes it more convenient to calculate the phase averaged quantities. All the simulation results presented are taken from cycles when periodicity is obtained. Phase averaged and long-time averaged data are calculated to compare the results of CFD simulations with available experimental data. The reference phase is defined as the maximum blowing occurring at a phase angle of 170° and maximum suction at a phase angle of 350°.

4.2.3. Boundary Conditions. At the floor and model surfaces, as well as the inner side of the cavity, solid non-slip wall conditions are applied. At the location $x/c = -6.39$ where velocity-inlet boundary condition is applied, uniform velocity profile is used to get a naturally-developed full-turbulent boundary layer so that it reaches the approximate boundary layer thickness as experimental data measured at the location of $x/c = -2.14$ [3]. At the downstream boundary, pressure-outlet boundary type is applied with the pressure $p/p_{ref} = 0.99962$ where p_{ref} is the free stream reference pressure. At top wall of the tunnel, inviscid wall condition is applied for the consideration of side plate blockage effect. The boundary condition at the bottom of the cavity is set as velocity-inlet where the components of the velocity are given as follows:

$$U = 0 \tag{18}$$

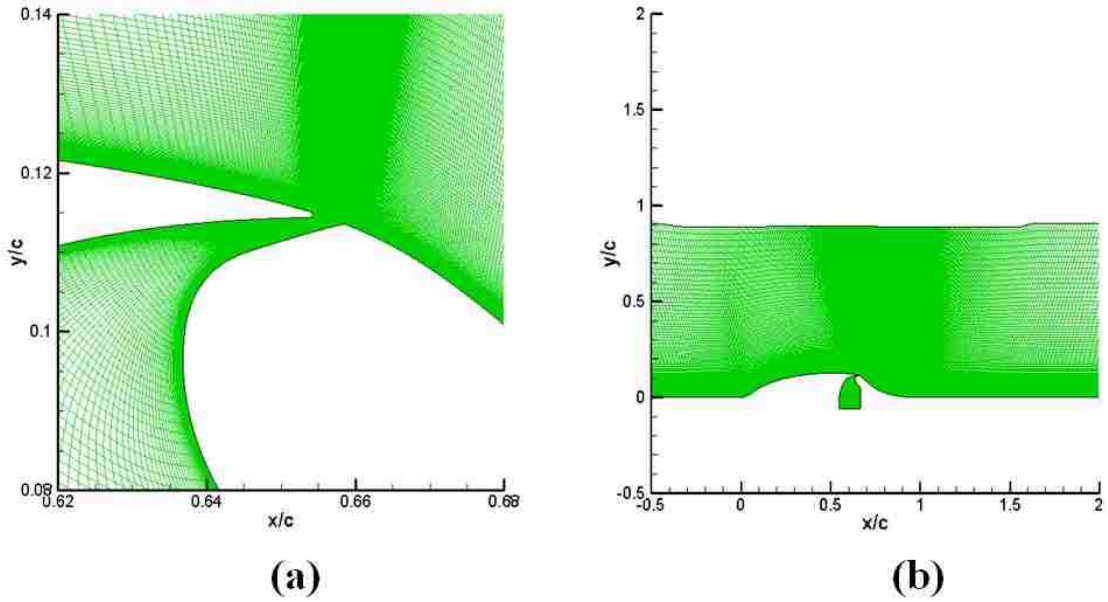

Figure 4.2. Computational Grid: (a) Zoom-in View of Slot Region and (b) Main Flow Domain.

$$V = A_0 cos(2\pi f t) \tag{19}$$

where the amplitude A_0 is picked to match the peak velocity out of slot during blowing part of cycle in the experiment [14]. Figure 4.3 shows the schematics of the boundary conditions applied in the CFD simulation.

4.3. DESCRIPTION OF THE STOCHASTIC PROBLEM

For this study, the free stream velocity (U_∞) and frequency in the unsteady velocity-inlet boundary condition (f) imposed to the cavity bottom are modeled as aleatory uniformly distributed uncertain variables with a coefficient of variance (CoV) of 10% from their baseline values. The turbulent viscosity coefficient within the Spalart-Allmaras (S-A) turbulence model [28] is treated as a source of epistemic uncertainty through the introduction of a factor K as shown below:

$$\mu_t = K\mu_{t_{SA}} \tag{20}$$

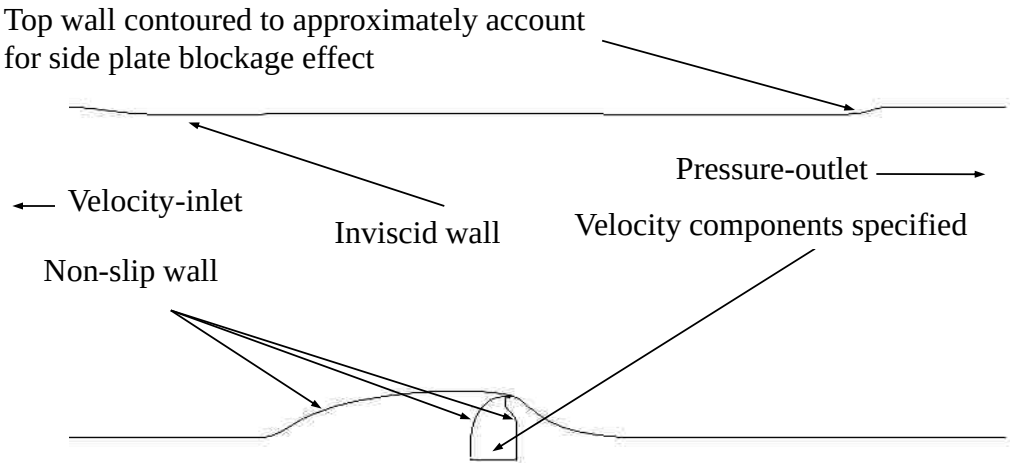

Figure 4.3. Overview of Boundary Conditions Applied in CFD Simulation.

where $\mu_{t_{SA}}$ is the turbulent viscosity originally calculated in the S-A model and then scaled by factor K as the turbulent viscosity used in the whole computational domain in the CFD simulations. The range of this factor K is chosen based on the turbulent viscosities calculated from different turbulence models (i.e., standard k-ϵ, standard k-ω and SST k-ω) [29] for the baseline case to reflect the uncertainty due to the use of different turbulence models. All other parameters in the CFD simulations are kept constant at their baseline values. An overview of the ranges of the uncertain parameters considered in this study is shown in Table 4.1.

Table 4.1. Uncertainty Ranges for Parameters Used in CFD Simulations.

Uncertain Parameter	Uncertainty Type	Uncertainty Range
U_∞	Aleatory (Uniform)	[31.14, 38.06] m/s
f	Aleatory (Uniform)	[124.65, 152.35] Hz
K	Epistemic	[0.5, 2.0]

In the following section, the quantities (both long-time averaged and phase averaged) of interest selected to perform uncertainty quantification analysis are introduced.

4.4. QUANTITIES OF INTEREST

In the CFD simulation, twenty cycles of the synthetic jet actuation are calculated in time domain and periodicity is obtained. To calculated the long-time averaged and phase averaged quantities, twenty data sets are saved for each cycle. Figure 4.4 shows an example of the solution history of the monitored maximum x-velocity out of the slot.

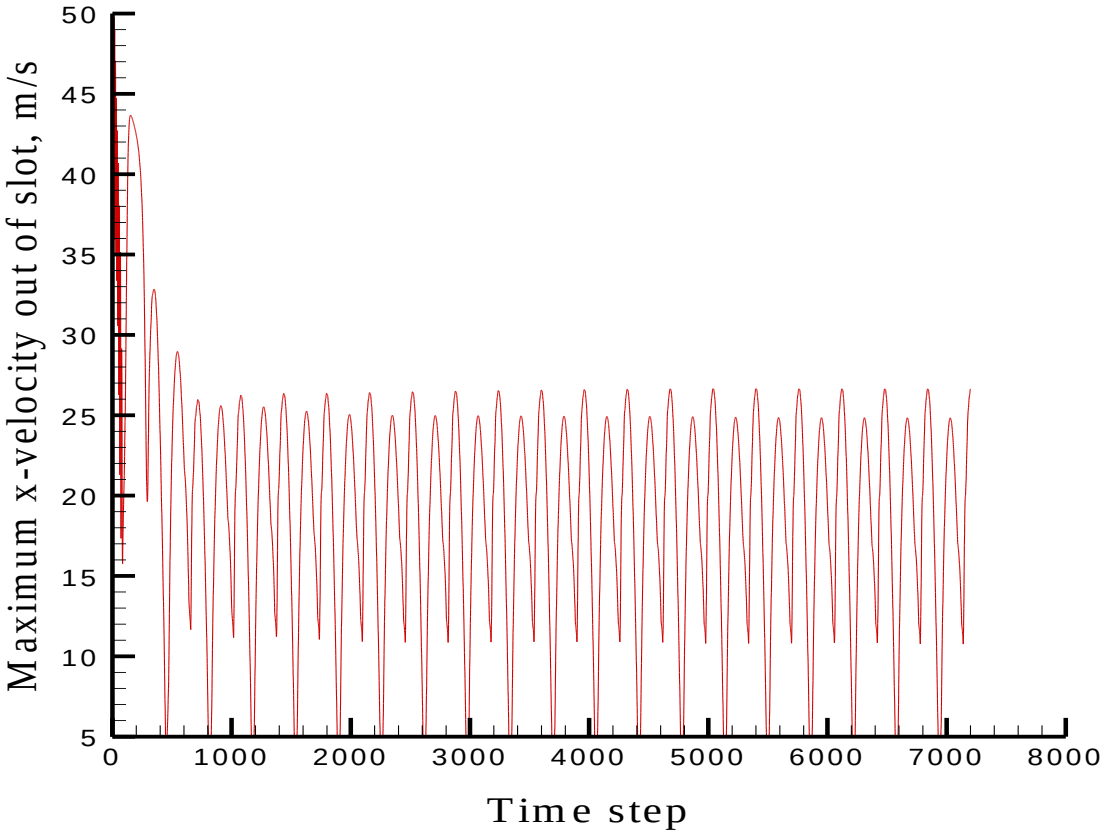

Figure 4.4. An Example of Solution History of Maximum X-velocity Out of Slot.

The definition and calculation of long-time averaged and phase averaged quantities are described below.

4.4.1. Long-time Averaged Quantities. To calculate the long-time averaged data, the quantities of interest extracted from all saved data sets of the last cycle are averaged. In the current study, long-time averaged pressure and skin friction coefficients are calculated as follows:

$$\overline{C_p} = \frac{\overline{p}(\vec{x}, \vec{\xi}) - p_\infty}{\frac{1}{2}\rho_\infty {U_\infty}_m{}^2} \tag{21}$$

$$\overline{C_f} = \frac{\overline{\tau_w}(\vec{x}, \vec{\xi})}{\frac{1}{2}\rho_\infty {U_\infty}_m{}^2} \tag{22}$$

where ${U_\infty}_m$ is the mean value of the free stream velocity.

The long-time averaged bubble characteristics are obtained from the skin friction coefficients. The results are also to be compared with available experimental data.

4.4.2. Phase Averaged Quantities. The CFDVAL2004 workshop also provided the experimental data for phase averaged quantities. In the simulation, once the periodicity is obtained, any single point during a cycle corresponds to a phase averaged data. For different values of frequency (one of uncertainty sources considered), the solution time step size is calculated from each actual value of frequency (not mean or baseline value). The reference phase is defined as maximum blowing at phase = 170° and maximum suction at phase = 350° (see Figure 4.5 for an example). The phase averaged x-velocity distributions scaled with mean value of free stream velocity (i.e., $U/{U_\infty}_m$) at selected locations are to be performed uncertainty quantification analysis and compared with experimental data.

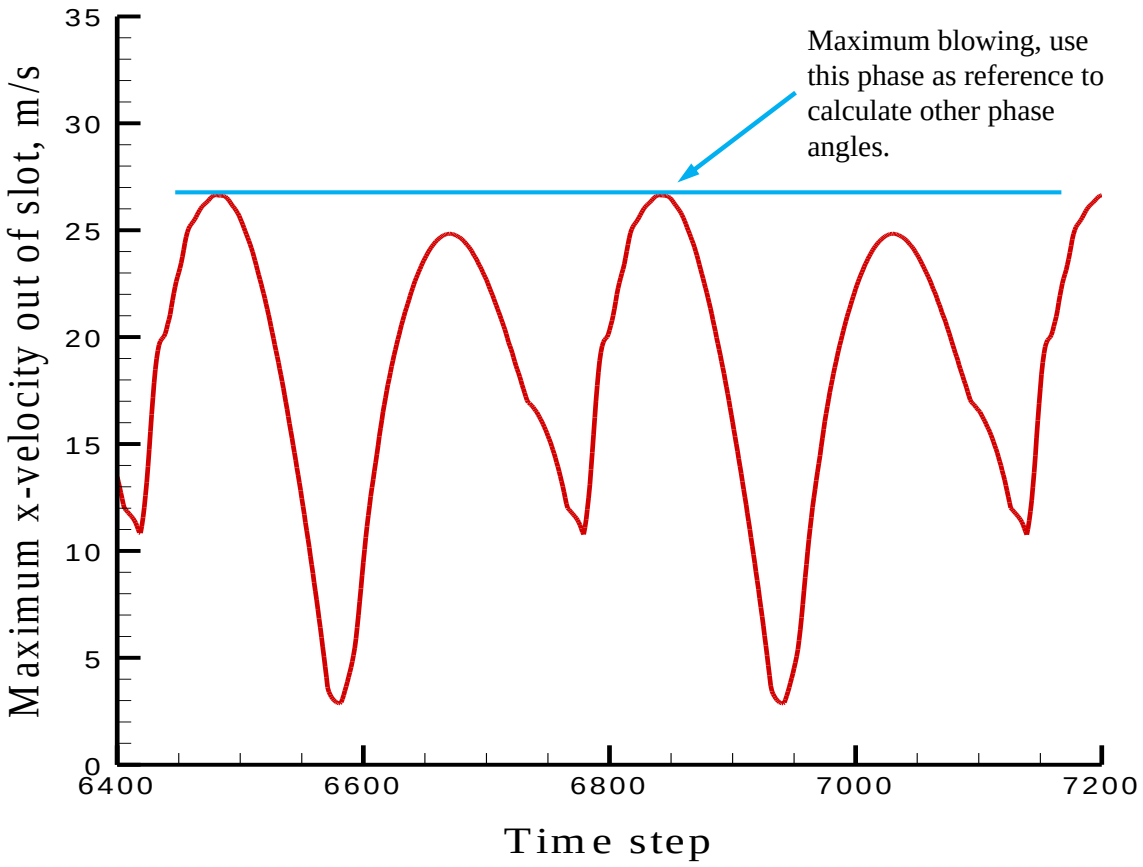

Figure 4.5. Definition of Reference Phase.

5. UNCERTAINTY QUANTIFICATION RESULTS AND DISCUSSIONS

In this section, the uncertainty quantification analysis results of the stochastic CFD problem described in previous sections are presented and discussed.

5.1. UNCERTAINTY QUANTIFICATION IN LONG-TIME AVERAGED SEPARATION BUBBLE SIZE

Figure 5.1 is a sample CFD result from the CFDVAL2004 workshop showing the main flow structure near the separation bubble region. In this study, the long-time averaged separation bubble size (calculated with separation and reattachment locations) is chosen to represent the bubble characteristics. The preliminary results of the current project, as well as Rumsey [14], showed that the location of separation is relatively insensitive to the parameters considered. The reattachment location is found to have a large variance. The separation bubble size is the difference between separation and reattachment locations.

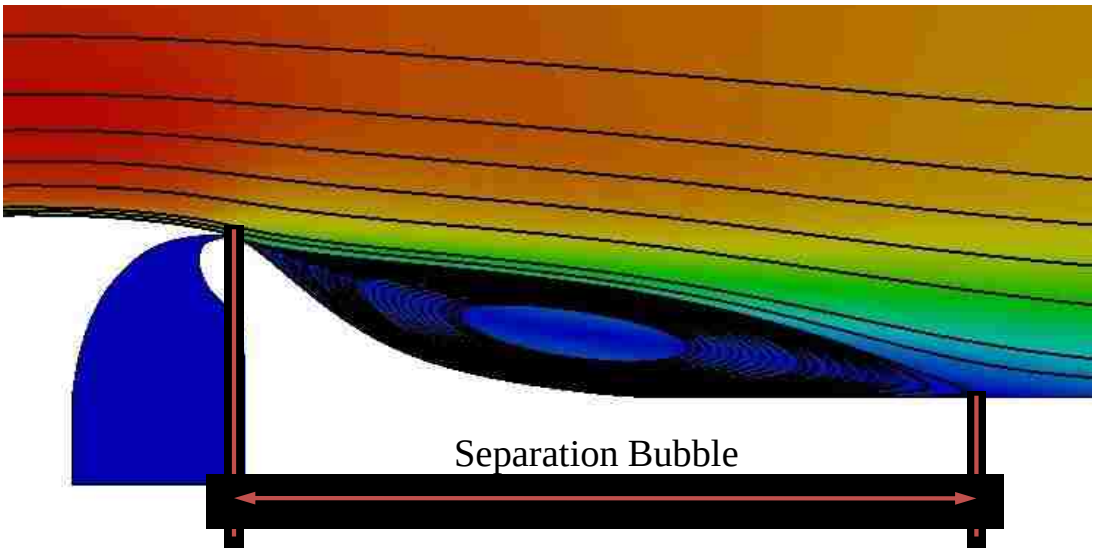

Figure 5.1. A Sample CFD Result from CFDVAL2004 Workshop [3].

5.1.1. Results with Pure Aleatory Uncertainty Assumption. Before mixed aleatory-epistemic uncertainty quantification, the analysis with pure aleatory uncertainty assumption is conducted where all three uncertain input variables are treated as aleatory with uniform uncertainty distributions with the bounds given in previous section. The CDFs obtained from different degrees of PCE are compared. The Quadrature-Based NIPC method described in previous section is used to construct the response surface as a function of all three uncertain input variables. It is important to ensure that the order of PCE is high enough to capture the non-linear relations between input and output quantities of interest. Therefore, a degree convergence check study is performed where the PCE order is increased up to 4 and the response surface is constructed and evaluated at each order. Figure 5.2 gives an example of the CDF plots to show the degree convergence of the PCE with respect to long-time averaged separation bubble size. 10,000 randomly produced samples are selected to evaluate the constructed stochastic response surface. The figure shows that there is no obvious difference in the CDFs between 3^{rd} and 4^{th} orders of PCE. Thus it can be concluded that the response surface obtained via the Quadrature-Based NIPC converges at the 3^{rd} order PCE.

5.1.2. Results with Mixed (Aleatory-Epistemic) Uncertainty Assumption. For the mixed aleatory-epistemic uncertainty quantification, Second-Order Probability approach described previously is used with the same response surface for pure aleatory uncertainty assumption (4^{th} order PCE). Random samples from the specified bounds (Table 4.1) are utilized for the epistemic uncertain variable in the outer loop while in the inner loop, for each specific value of the epistemic uncertain variable, samples of aleatory uncertain variables based on the uniform probability distributions are utilized to evaluate the stochastic response surface. Two sets of samples are selected to check the sample size independence. The first set takes 100 samples in the outer (epistemic) loop and 1,000 samples in the inner (aleatory) loop; the second

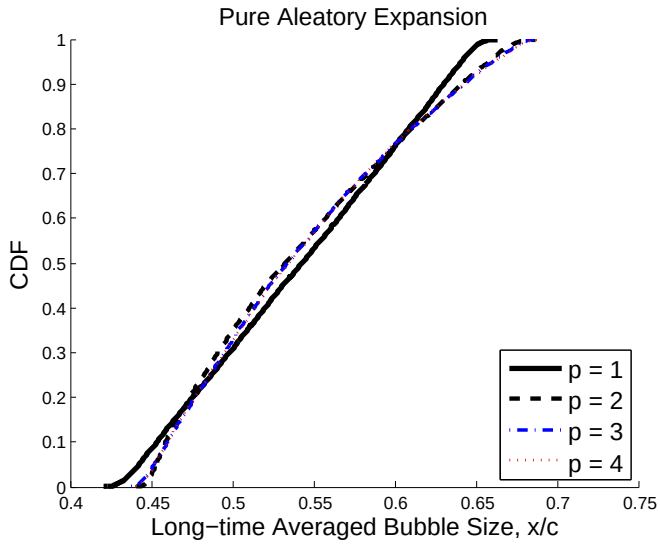

Figure 5.2. PCE Degree Convergence Check of Long-time Averaged Bubble Size.

set takes 1,000 and 10,000 samples in the outer and inner loops, respectively. The CDFs produced are shown in Figure 5.3. For each set of samples, it is obvious that at a particular probability level, the variation in the long-time averaged separation bubble size is due to the epistemic uncertain input (K factor), which is represented by the interval bounded by the minimum and maximum values obtained from the CDFs at the same probability level. The width of the interval is nearly constant at each probability level. The overall agreement of the horsetail plots obtained from the two sets of samples also shows the sample size independence in the Second-Order Probability framework. Thus in the following part of this study, 100 samples for the epistemic loop and 1,000 samples for the aleatory loop are utilized considering the computational expenses.

5.1.3. Global Sensitivity Analysis with Sobol Indices. In order to have a relative ranking of the importance of each input uncertain variable on the overall output uncertainty in the long-time averaged separation bubble size, global sensitivity analysis with Sobol indices is conducted to quantitatively account for the non-linear dependencies between input and output uncertainties. These indices obtained from

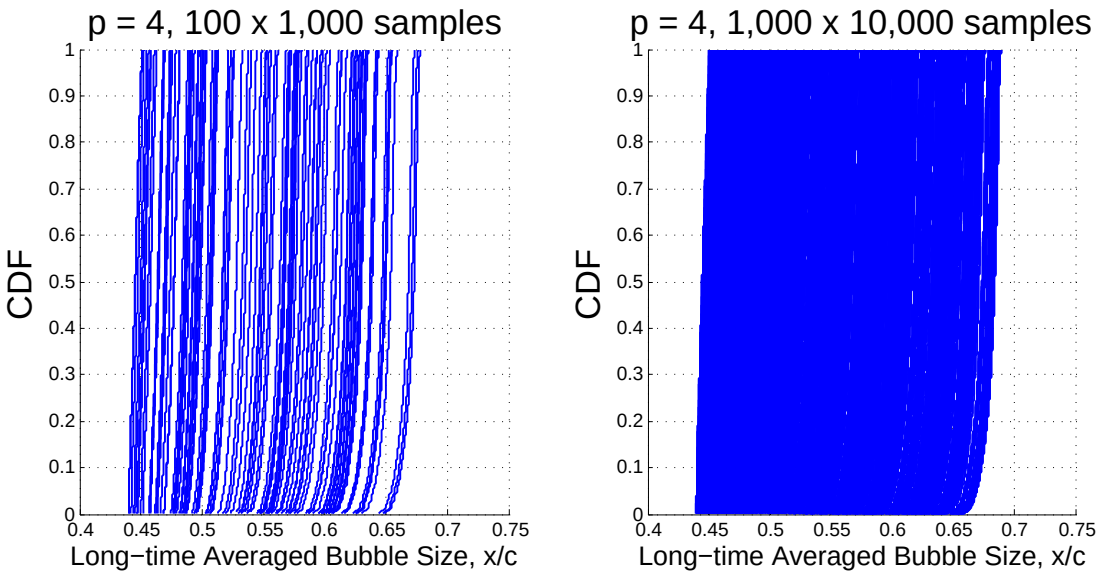

Figure 5.3. Horsetail Plot for Long-time Averaged Bubble Size.

the same order of PCE (4^{th}) in previous sections are shown in Table 5.1. The results show that the epistemic input uncertain variable K factor has a dominant influence on the output uncertainty in the long-time averaged bubble size while the two aleatory input uncertain variables have much less contributions to the output uncertainty. This is consistent with the observation from the horsetail plots shown in Figure 5.3. This result also agrees qualitatively well with a similar study by Rumsey [14].

Furthermore, the results also indicate that the combined contributions (see Section 3.2.5 for details) of different uncertainty sources (e.g., $S_{1,2}$) are very small when comparing the total indices (e.g., S_{T_1}) with their corresponding individual (non-combined) indices (e.g., S_1).

5.2. UNCERTAINTY QUANTIFICATION IN LONG-TIME AVERAGED PRESSURE AND SKIN FRICTION COEFFICIENTS

5.2.1. Results with Pure Aleatory Uncertainty Assumption. Similar to long-time averaged bubble size, the analysis with pure aleatory uncertainty

Table 5.1. Sobol Indices of Each Uncertain Input in Long-time Averaged Bubble Size.

Index	Parameter	Sobol Indices
S_1	K	0.9948
S_2	U_∞	0.0022
S_3	f	0.0025
S_{T_1}	K	0.9951
S_{T_2}	U_∞	0.0026
S_{T_3}	f	0.0029

assumption is also conducted where all three uncertain input variables are treated as aleatory with uniform uncertainty distributions with the bounds given in previous section. The CDFs obtained from different degrees of PCE are also compared. Again the Quadrature-Based NIPC method described in previous section is used to construct the response surface as a function of all three uncertain input variables. A degree convergence check study is performed where the PCE order is increased up to 4 and the response surface is constructed and evaluated at each order. Figure 5.4, 5.5, 5.6 are the CDF plots showing the degree convergence of the PCE with respect to long-time averaged pressure and skin friction coefficients at three selected locations ($x/c = 0.62693$, upstream separation bubble; $x/c = 0.994$, inside separation bubble; $x/c = 1.5212$, downstream separation bubble), respectively. It is shown again that there is no obvious difference in the CDFs between 3^{rd} and 4^{th} orders of PCE. Thus it can be concluded that the response surfaces obtained via the Quadrature-Based NIPC for long-time averaged pressure and skin friction coefficients again converge at the 3^{rd} order PCE.

5.2.2. Results with Mixed (Aleatory-Epistemic) Uncertainty Assumption. For the mixed (aleatory-epistemic) uncertainty quantification with respect to long-time averaged pressure and skin friction coefficients along the wall, Second-Order Probability approach is used again with the same response surfaces for pure

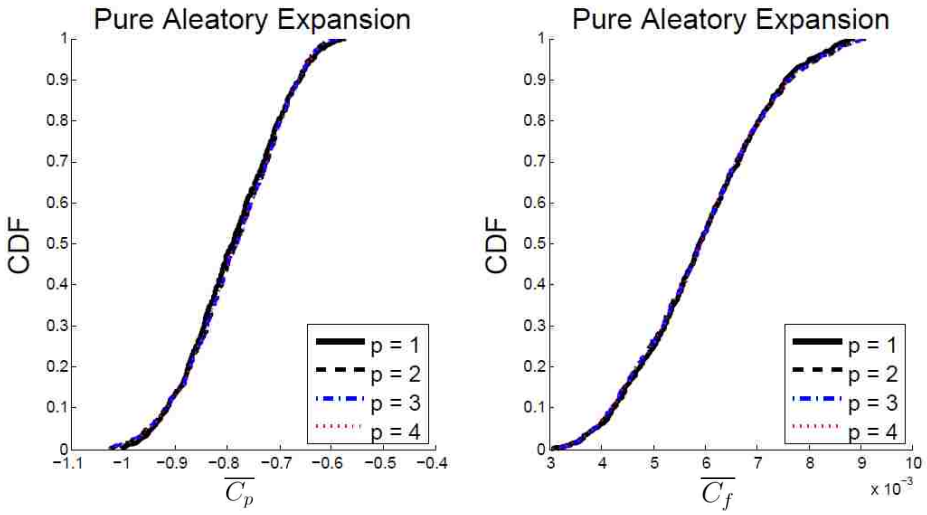

Figure 5.4. PCE Degree Convergence Check of Long-time Averaged Pressure Coefficient at Location $x/c = 0.62693$ (upstream separation bubble).

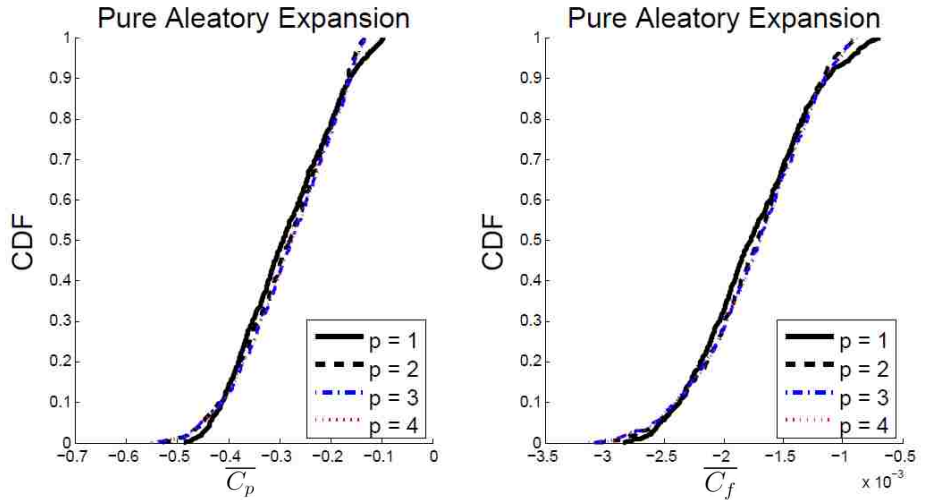

Figure 5.5. PCE Degree Convergence Check of Long-time Averaged Pressure Coefficient at Location $x/c = 0.994$ (inside separation bubble).

aleatory uncertainty assumption (4^{th} order PCE). 100 random samples from the specified bounds (Table 4.1) are selected for the epistemic uncertain variable in the outer loop. In the inner loop, for each specific value of the epistemic uncertain variable, 1,000 random samples of aleatory uncertain variables based on uniform probability

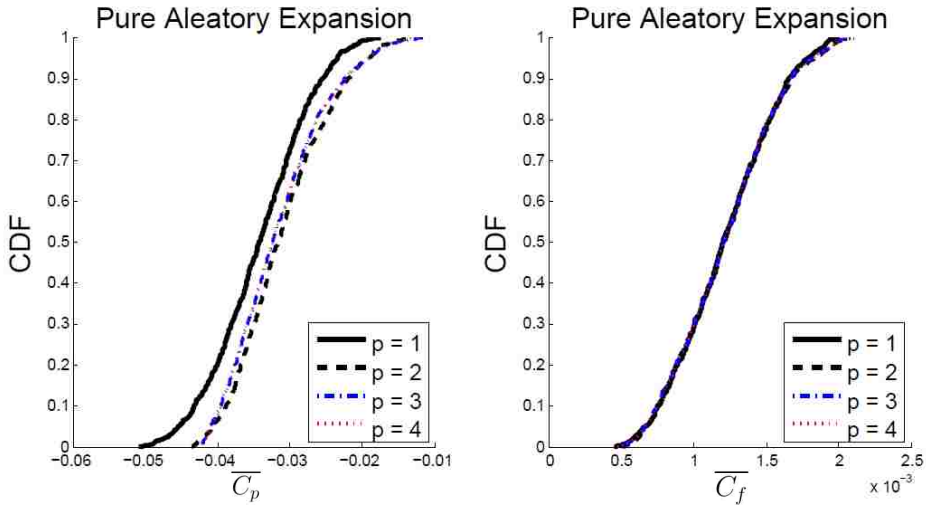

Figure 5.6. PCE Degree Convergence Check of Long-time Averaged Pressure Coefficient at Location $x/c = 1.5212$ (downstream separation bubble).

distributions are utilized to evaluate the stochastic response surface. This procedure produces 100 CDFs which are then evaluated to find the lower and upper bounds of the interval at each probability level.

The interval bounds for pressure and skin friction coefficients distributions are plotted at selected points along the wall at 2.5%, 50% and 97.5% probability levels as shown in Figure 5.7 and Figure 5.8, respectively, including the available experimental data [3]. The results with pure aleatory uncertainty assumption are also plotted in the figures at the corresponding probability levels. It should be noted that the pure aleatory result is just a single value at each probability level at each selected point along the wall while the mixed uncertainty results are intervals. As expected, the results from the pure aleatory uncertainty assumption lies within the bounds of the mixed uncertainty results. At probability level of 2.5%, the pure aleatory results stay almost in the bottom portion of the mixed uncertainty intervals, especially for the long-time averaged skin friction coefficient (note that the axis for long-time averaged pressure coefficient is reversed). At probability level of 50%, the pure aleatory values

move towards the center of the mixed uncertainty intervals while at probability level of 97.5%, the pure aleatory results are very close to the upper limits of the mixed uncertainty intervals. It is also obvious that for long-time averaged pressure coefficient, the mixed uncertainty intervals are significant only near the separation bubble region.

In addition, the 95% confidence interval (CI) plots of both pressure and skin friction coefficients are shown in Figure 5.9 and Figure 5.10, respectively.

5.2.3. Global Sensitivity Analysis with Sobol Indices. Similar to long-time averaged bubble size, Sobol indices of each uncertain input with respect to the long-time averaged pressure and skin friction coefficients at the three selected locations are also computed with the 4^{th} order PCE. These indices are shown in Table 5.2, 5.3, 5.4, respectively.

Table 5.2. Sobol Indices of Each Uncertain Input in Long-time Averaged Pressure and Skin Friction Coefficients at Location $x/c = 0.62693$ (upstream separation bubble).

Index	Parameter	Sobol Indices for $\overline{C_p}$	Sobol Indices for $\overline{C_f}$
S_1	K	0.2535	0.7656
S_2	U_∞	0.7423	0.2249
S_3	f	3.1460E-4	6.1673E-5
S_{T_1}	K	0.2574	0.7750
S_{T_2}	U_∞	0.7462	0.2343
S_{T_3}	f	3.3027E-4	6.6998E-5

The results indicate again that the combined contributions of different uncertainty sources (e.g., $S_{1,2}$) are very small when comparing the total indices (e.g., S_{T_1}) with their corresponding individual (non-combined) indices (e.g., S_1) for both long-time averaged pressure and skin friction coefficients at all three locations. Furthermore, it can be seen that at a location upstream separation bubble (Table 5.2), the

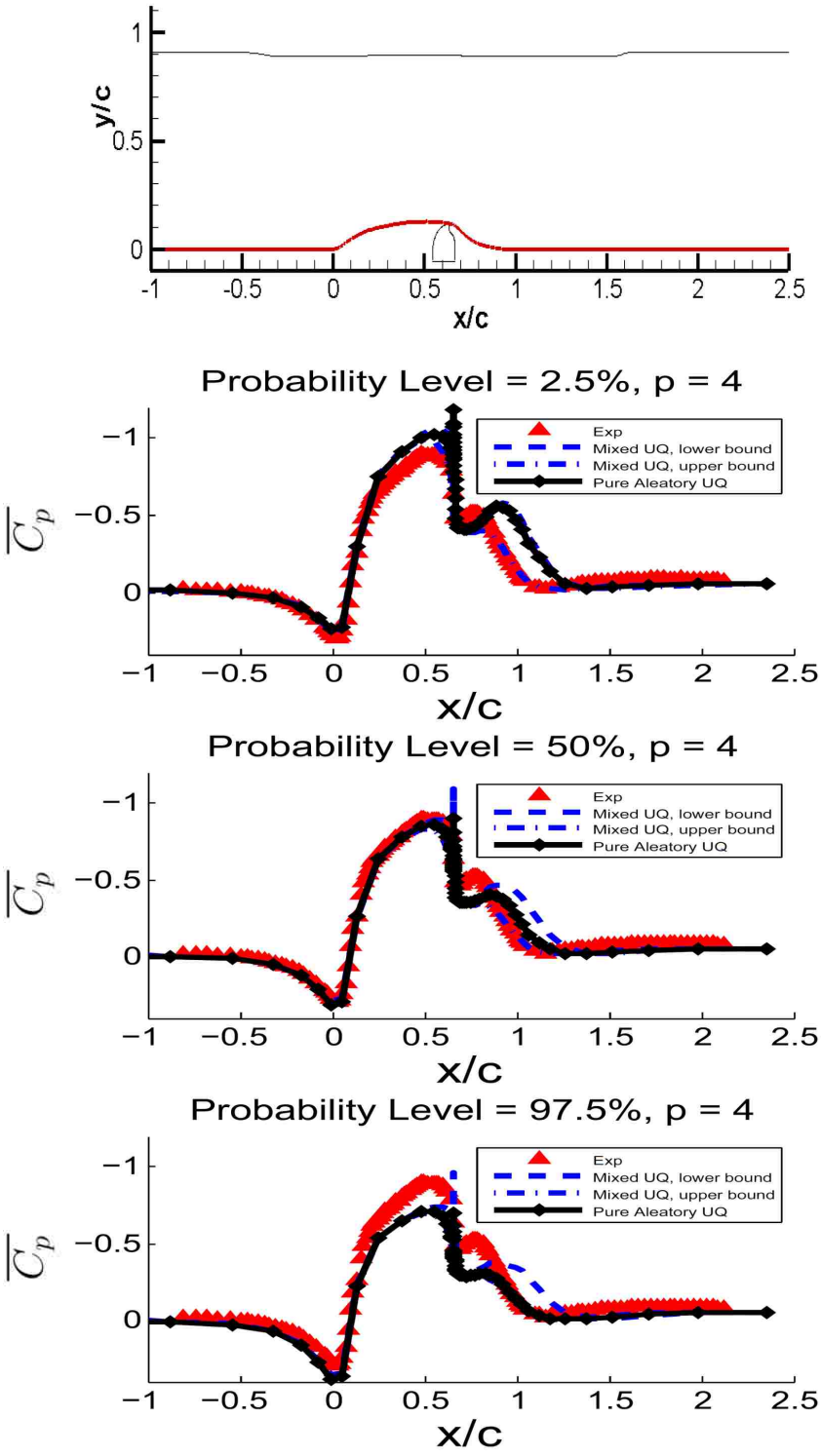

Figure 5.7. Mixed UQ for Long-time Averaged Pressure Coefficient.

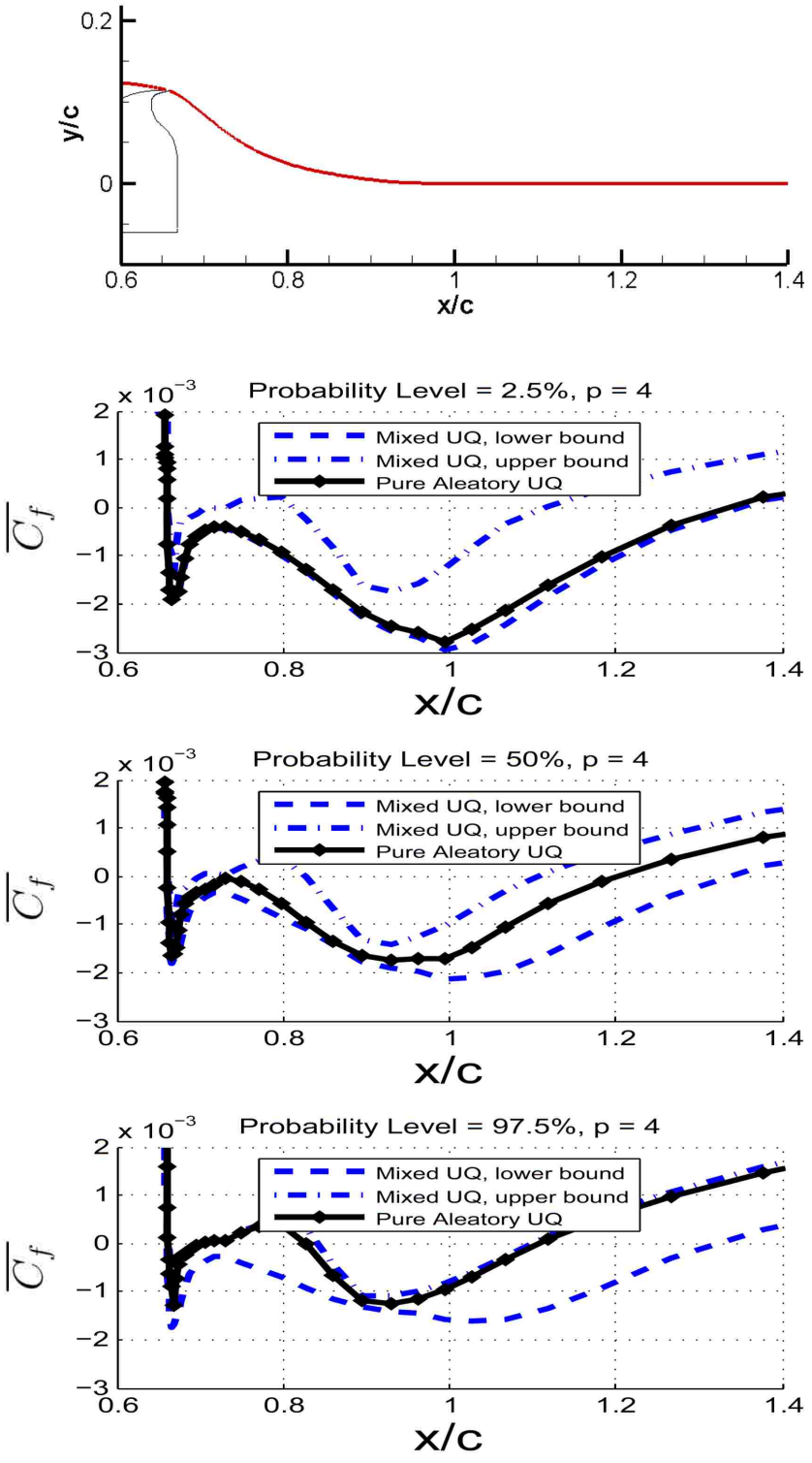

Figure 5.8. Mixed UQ for Long-time Averaged Skin Friction Coefficient.

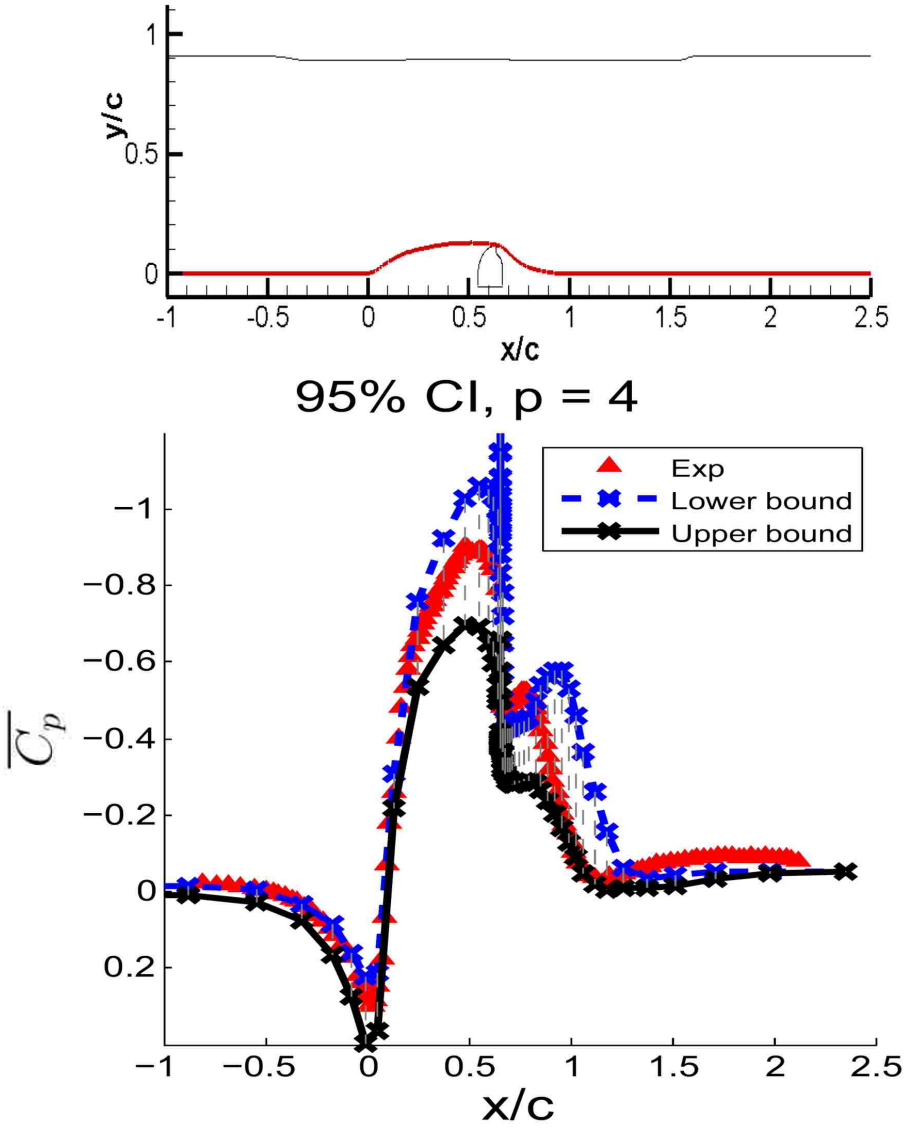

Figure 5.9. 95% Confidence Interval for Long-time Averaged Pressure Coefficient.

aleatory uncertainty input variable free stream velocity, U_∞, is the main contributor to the output uncertainty in long-time averaged pressure coefficient while at a location inside separation bubble (Table 5.3), the epistemic uncertainty variable, K factor, becomes the main contributor. The contributions from U_∞ and K factor to the long-time averaged pressure coefficient are comparable at a location downstream separation bubble (Table 5.4). For long-time averaged skin friction coefficient, the

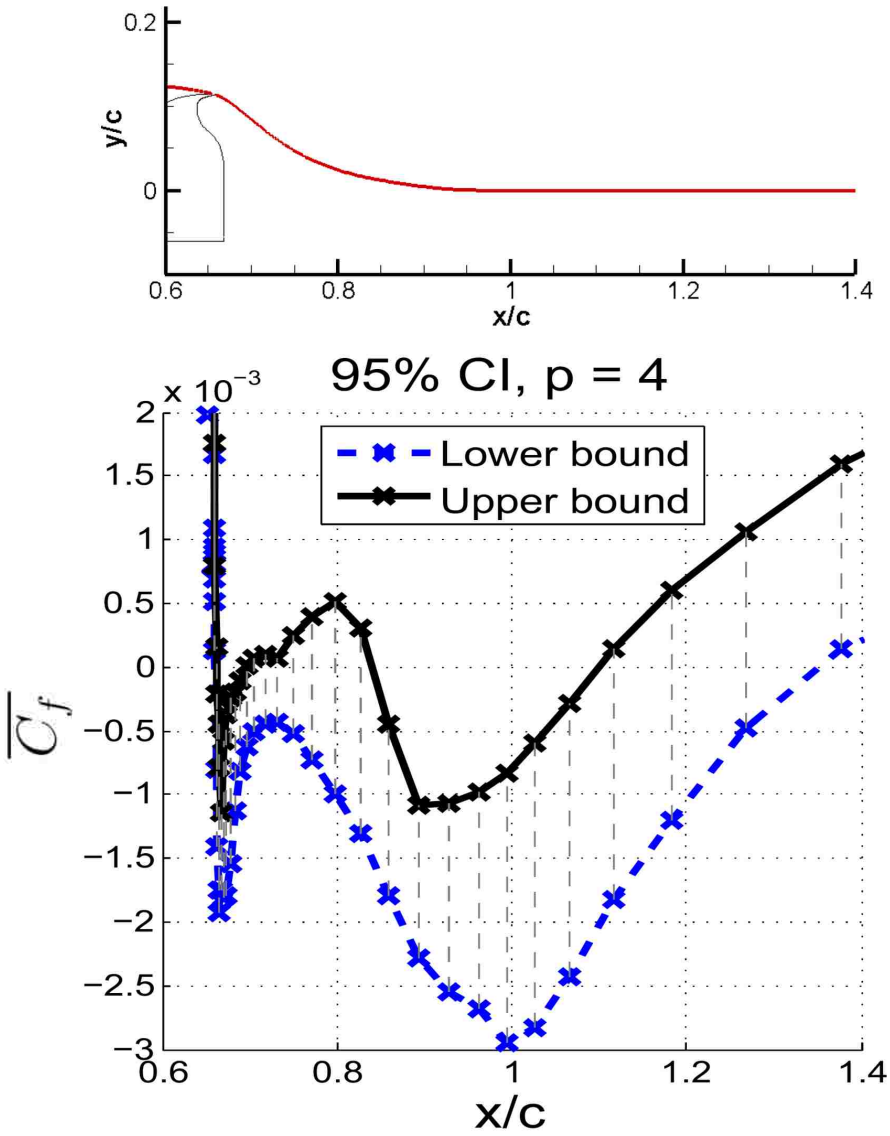

Figure 5.10. 95% Confidence Interval for Long-time Averaged Skin Friction Coefficient.

epistemic uncertainty input variable, K factor, has a main contribution at all three selected locations. It is also noticeable that the uncertainty in the aleatory input variable frequency, f, has the least influence on both long-time averaged pressure and skin friction coefficients at all three selected locations.

Table 5.3. Sobol Indices of Each Uncertain Input in Long-time Averaged Pressure and Skin Friction Coefficients at Location $x/c = 0.994$ (inside separation bubble).

Index	Parameter	Sobol Indices for $\overline{C_p}$	Sobol Indices for $\overline{C_f}$
S_1	K	0.8805	0.6589
S_2	U_∞	0.0951	0.2773
S_3	f	4.3054E-4	0.0214
S_{T_1}	K	0.9042	0.7002
S_{T_2}	U_∞	0.1181	0.3134
S_{T_3}	f	0.0017	0.0291

Table 5.4. Sobol Indices of Each Uncertain Input in Long-time Averaged Pressure and Skin Friction Coefficients at Location $x/c = 1.5212$ (downstream separation bubble).

Index	Parameter	Sobol Indices for $\overline{C_p}$	Sobol Indices for $\overline{C_f}$
S_1	K	0.5976	0.8272
S_2	U_∞	0.3522	0.1611
S_3	f	0.0300	0.0024
S_{T_1}	K	0.6102	0.8361
S_{T_2}	U_∞	0.3721	0.1705
S_{T_3}	f	0.0385	0.0028

5.3. UNCERTAINTY QUANTIFICATION IN PHASE AVERAGED VELOCITY DISTRIBUTIONS

For the phase averaged quantities, the x-velocity distributions at three selected locations are picked to perform the uncertainty quantification analysis. At locations of $x/c = 0.66$ (just downstream of slot), $x/c = 0.80$ (inside separation bubble) and $x/c = 1.00$ (near end of separation bubble), phase averaged x-velocity distributions at phase angles of 80°, 170°, 260° and 350° are analyzed with mixed aleatory-epistemic uncertainty assumption. Figure 5.11 shows a schematic of the three x/c locations.

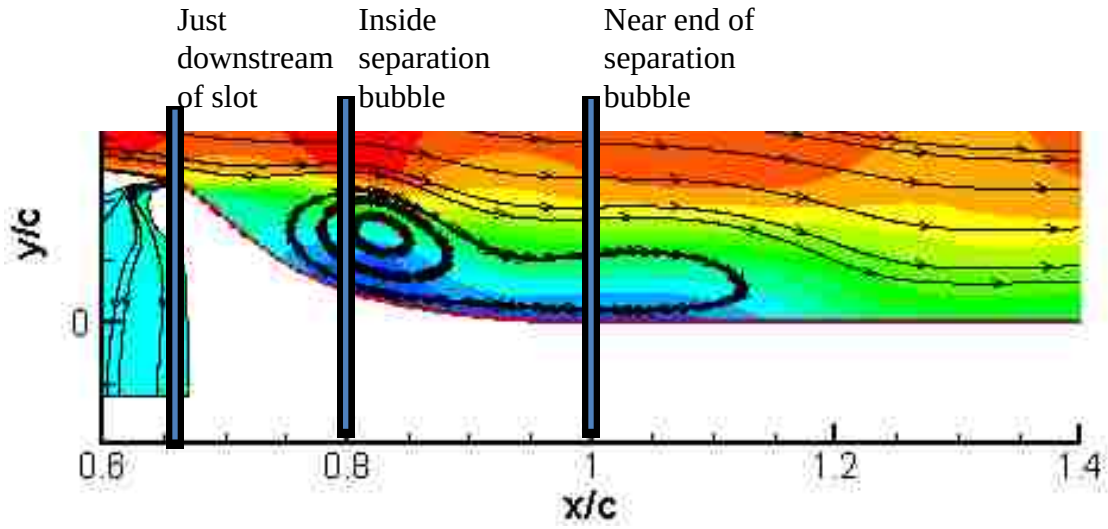

Figure 5.11. Schematic of Three Selected Locations to Perform UQ Analysis of Phase Averaged X-velocity Distributions.

5.3.1. Results with Mixed (Aleatory-Epistemic) Uncertainty Assumption. Similar to the uncertainty quantification analysis approach for the long-time averaged pressure and skin friction coefficients, phase averaged x-velocity distributions at selected locations are picked to perform the mixed aleatory-epistemic uncertainty quantification. Second-Order Probability approach is used again with a constructed stochastic response surface (4^{th} order PCE). 100 and 1,000 random samples are utilized in outer and inner loops of Second-Order Probability framework, respectively, to evaluate the stochastic response surface. This procedure produces 100 CDFs which can be evaluated to find the lower and upper bounds of the interval at each probability level, as well as 95% confidence intervals.

Figure 5.12, 5.13, 5.14 show the Second-Order Probability results for the phase averaged x-velocity distributions at locations of $x/c = 0.66, 0.80, 1.00$, respectively. For comparison, the experimental data [3] are also included in the plots. The results

show that with the uncertainty ranges considered, the output statistics are able to generally envelope the experimental data with 95% confidence intervals (CI) especially at locations of $x/c = 0.66$ and $x/c = 0.80$. At location of $x/c = 1.00$, some of the experimental data are still not captured by the confidence intervals at phase angles of 80° and 350°.

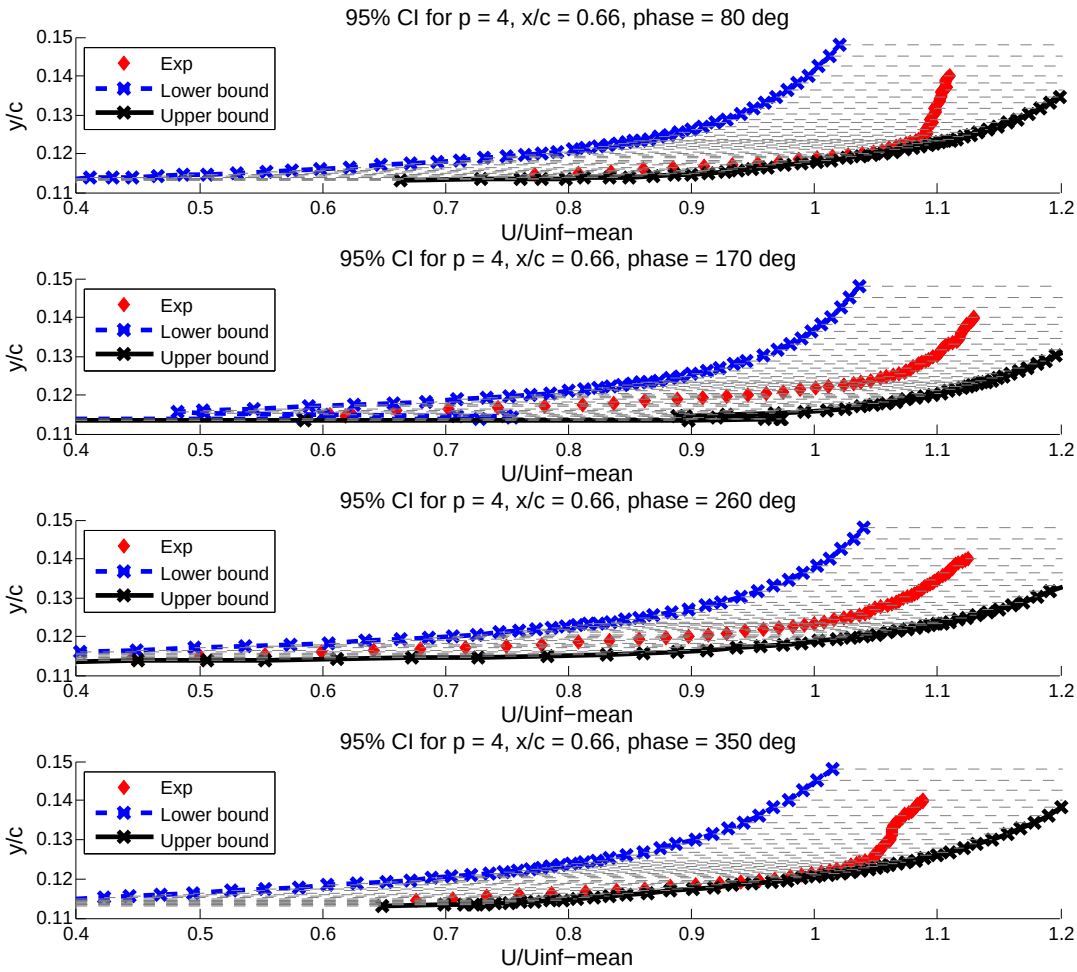

Figure 5.12. 95% CI for Phase Averaged X-velocity Distribution at Location $x/c = 0.66$.

5.3.2. Global Sensitivity Analysis with Sobol Indices. For the global sensitivity analysis, the x-velocity distributions at three selected points at each x/c

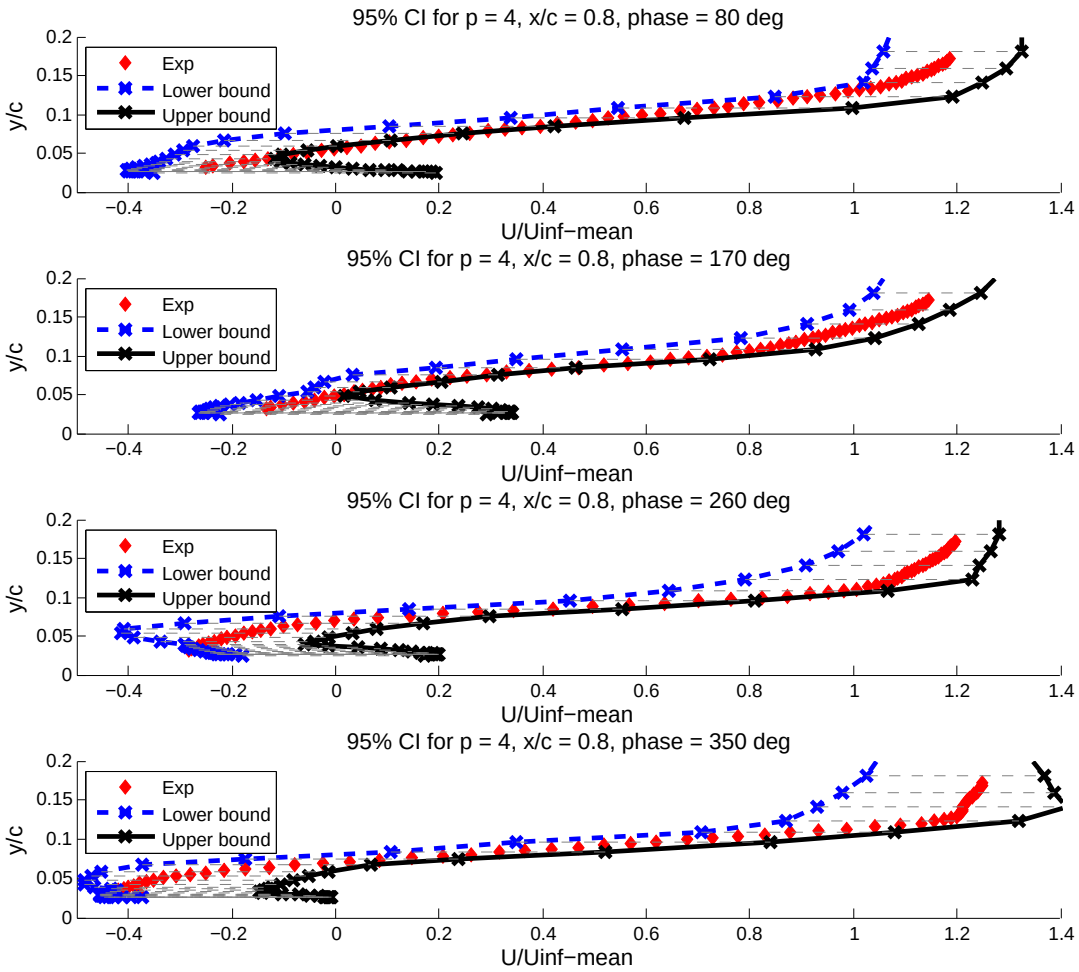

Figure 5.13. 95% CI for Phase Averaged X-velocity Distribution at Location $x/c = 0.80$.

location and phase angle are picked to calculate the Sobol indices with 4^{th} order PCE. The results are tabulated in Table 5.5, 5.6, 5.7, respectively.

Similar to results presented before, the combined contributions of different uncertainty sources (e.g., $S_{1,2}$) are relatively small, except for some data sets at location of $x/c = 0.80$ ($y/c = 0.026463$, phase 80° and $y/c = 0.10869$, phase 350°). These uncommon data may be due to that the non-linear combined terms of different uncertainty variables play a large role in the polynomial chaos expansion of the output quantity.

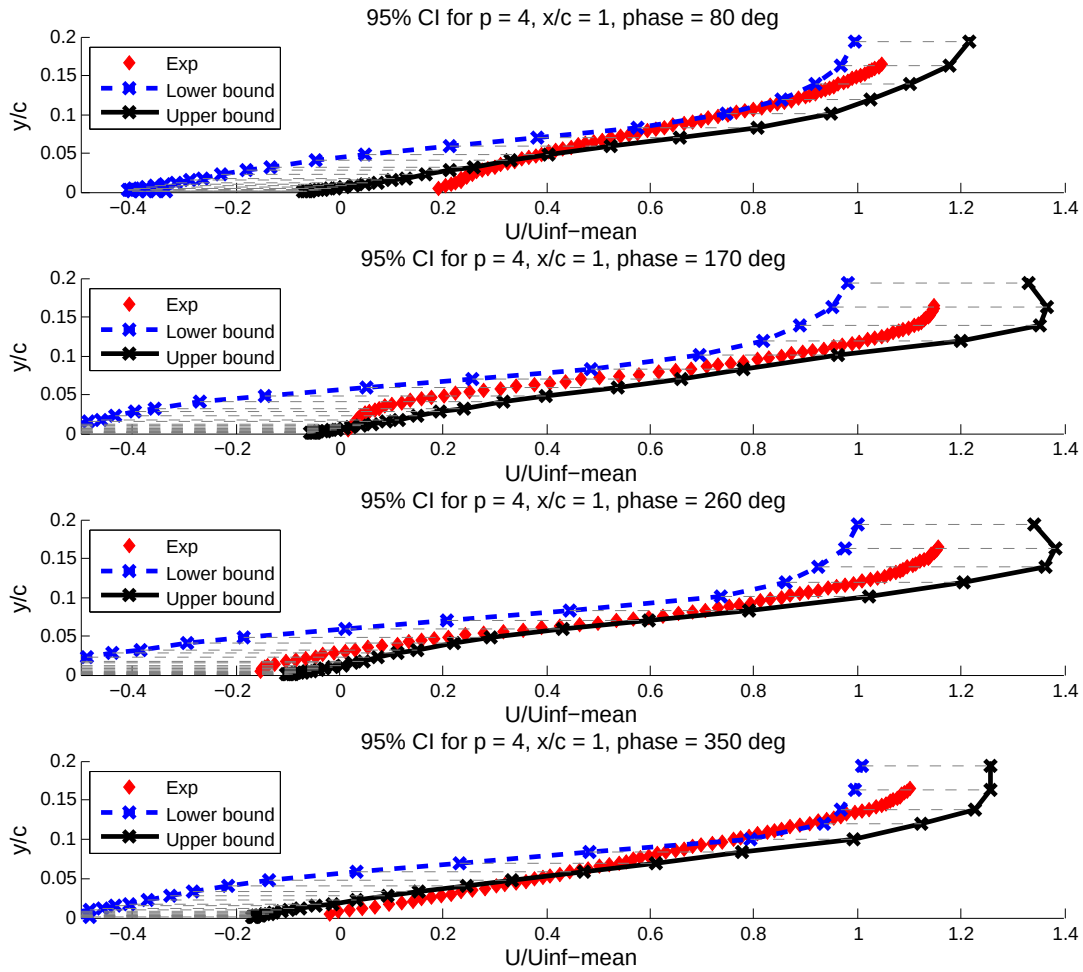

Figure 5.14. 95% CI for Phase Averaged X-velocity Distribution at Location $x/c = 1.00$.

The results also show that, for each of the three x/c locations, at a y/c location near the wall, the epistemic uncertain variable, K factor (turbulence model), has a main contribution to the uncertainty in the output x-velocity. Approaching the the main stream, the contribution from the aleatory uncertain input variable, free stream velocity, U_∞, becomes larger while at a location near the main stream, the free stream velocity becomes the dominant contributor to the uncertainty in the output x-velocity. The results also show that the contribution from the other aleatory uncertain variable, frequency, f, is significantly small compared to the other two uncertain variables.

Table 5.5. Sobol Indices of Each Uncertain Input in Phase Averaged X-velocity Distribution at Location $x/c = 0.66$.

Phase	Index	Parameter	y/c Locations		
			$y/c = 0.11322$ (near the wall)	$y/c = 0.11876$ (between wall and main stream)	$y/c = 0.14792$ (near main stream)
80°	S_1	K	0.8615	0.2820	0.0025
	S_2	U_∞	0.1330	0.7160	0.9973
	S_3	f	6.5717E-4	0.0012	3.1616E-5
	S_{T_1}	K	0.8663	0.2827	0.0026
	S_{T_2}	U_∞	0.1378	0.7167	0.9974
	S_{T_3}	f	7.1667E-4	0.0013	1.1044E-4
170°	S_1	K	0.9366	0.5365	0.0032
	S_2	U_∞	0.0035	0.4598	0.9967
	S_3	f	1.2345E-4	1.8301E-4	9.5108E-5
	S_{T_1}	K	0.9961	0.5400	0.0032
	S_{T_2}	U_∞	0.0614	0.4631	0.9967
	S_{T_3}	f	0.0031	4.8004E-4	1.0968E-4
260°	S_1	K	0.9668	0.7286	8.0203E-4
	S_2	U_∞	0.0010	0.2624	0.9987
	S_3	f	0.0070	0.0058	1.2189E-4
	S_{T_1}	K	0.9919	0.7318	0.0012
	S_{T_2}	U_∞	0.0063	0.2654	0.9990
	S_{T_3}	f	0.0272	0.0060	2.5496E-4
350°	S_1	K	0.9090	0.4746	0.0090
	S_2	U_∞	0.0870	0.5245	0.9902
	S_3	f	6.0224E-4	3.4279E-5	6.7424E-4
	S_{T_1}	K	0.9123	0.4754	0.0091
	S_{T_2}	U_∞	0.0899	0.5249	0.9903
	S_{T_3}	f	0.0012	6.0735E-4	7.0411E-4

Table 5.6. Sobol Indices of Each Uncertain Input in Phase Averaged X-velocity Distribution at Location $x/c = 0.80$.

Phase	Index	Parameter	y/c Locations		
			$y/c = 0.026463$ (near the wall)	$y/c = 0.10869$ (between wall and main stream)	$y/c = 0.20204$ (near main stream)
80°	S_1	K	0.6630	0.8548	0.0231
	S_2	U_∞	0.0107	0.0052	0.9728
	S_3	f	0.0905	0.1204	5.8208E-4
	S_{T_1}	K	0.8786	0.8704	0.0258
	S_{T_2}	U_∞	0.1630	0.0230	0.9749
	S_{T_3}	f	0.1969	0.1270	0.0029
170°	S_1	K	0.8750	0.4539	0.0015
	S_2	U_∞	0.0283	0.5095	0.9919
	S_3	f	0.0569	0.0145	0.0056
	S_{T_1}	K	0.9122	0.4747	0.0025
	S_{T_2}	U_∞	0.0582	0.5184	0.9925
	S_{T_3}	f	0.0719	0.0291	0.0060
260°	S_1	K	0.9017	0.0148	0.0033
	S_2	U_∞	0.0528	0.8018	0.9944
	S_3	f	0.0188	0.0970	2.8282E-4
	S_{T_1}	K	0.9181	0.0932	0.0051
	S_{T_2}	U_∞	0.0712	0.8492	0.9957
	S_{T_3}	f	0.0411	0.1492	0.0014
350°	S_1	K	0.4949	0.3395	0.0466
	S_2	U_∞	0.3657	0.2548	0.9422
	S_3	f	0.1201	0.0257	0.0075
	S_{T_1}	K	0.5113	0.7053	0.0502
	S_{T_2}	U_∞	0.3834	0.4914	0.9449
	S_{T_3}	f	0.1296	0.2004	0.0086

Table 5.7. Sobol Indices of Each Uncertain Input in Phase Averaged X-velocity Distribution at Location $x/c = 1.00$.

Phase	Index	Parameter	y/c Locations		
			$y/c = 0.0064552$ (near the wall)	$y/c = 0.082942$ (between wall and main stream)	$y/c = 0.19335$ (near main stream)
80°	S_1	K	0.8052	0.2719	0.0168
	S_2	U_∞	0.0403	0.6813	0.9616
	S_3	f	0.1311	0.0014	0.0145
	S_{T_1}	K	0.8262	0.3108	0.0209
	S_{T_2}	U_∞	0.0595	0.7156	0.9679
	S_{T_3}	f	0.1416	0.0306	0.0204
170°	S_1	K	0.8120	0.7659	0.1005
	S_2	U_∞	0.0701	0.1264	0.8568
	S_3	f	0.0308	0.0521	0.0230
	S_{T_1}	K	0.8711	0.7879	0.1172
	S_{T_2}	U_∞	0.1401	0.1791	0.8708
	S_{T_3}	f	0.0767	0.0945	0.0320
260°	S_1	K	0.7434	0.5741	0.1529
	S_2	U_∞	0.1749	0.4010	0.8360
	S_3	f	0.0727	0.0137	0.0052
	S_{T_1}	K	0.7492	0.5848	0.1554
	S_{T_2}	U_∞	0.1823	0.4050	0.8411
	S_{T_3}	f	0.0797	0.0230	0.0106
350°	S_1	K	0.9209	0.2916	0.0584
	S_2	U_∞	0.0126	0.6515	0.8996
	S_3	f	0.0293	0.0104	0.0270
	S_{T_1}	K	0.9296	0.3370	0.0708
	S_{T_2}	U_∞	0.0426	0.6718	0.9077
	S_{T_3}	f	0.0656	0.0378	0.0366

6. CONCLUSIONS AND FUTURE WORK

In this section, all the main relevant conclusions from the current study are presented and a vista of future work is also discussed.

6.1. CONCLUSIONS

In this study, an efficient methodology for the quantification of uncertainties in CFD modeling of synthetic jet actuators is introduced and demonstrated, which include both inherent (aleatory) and model-form (epistemic) uncertainty sources. A global non-linear sensitivity analysis is also utilized to quantify the contribution of each uncertainty source to the overall uncertainty in a selected output quantity.

From the current results obtained and presented, the following conclusions can be addressed:

1) For the uncertainty in long-time averaged bubble size, the K factor which reflects the turbulence model uncertainty through the turbulent viscosity plays a dominant role compared to the other two uncertainty input variables.

2) For long-time averaged pressure coefficient, at a location upstream the flow separation, the aleatory uncertainty input variable free stream velocity, U_∞, is the main contributor while at a location inside separation bubble, the epistemic uncertainty variable, K factor, becomes the main contributor. The contributions from U_∞ and K factor are comparable at a location downstream flow separation. For long-time averaged skin friction coefficient, the epistemic uncertainty input variable, K factor, has a main contribution at all three selected locations. It is also noticeable that the uncertainty in the aleatory input variable, actuation frequency, f, has the least influence on both long-time averaged pressure and skin friction coefficients at all three selected locations.

3) For the phase averaged x-velocity distribution, at each of the three selected x/c locations, the epistemic uncertain input variable, K factor (turbulence model), affects the x-velocity most significantly at the near-wall region. Approaching the the main stream, the contribution from the aleatory uncertain input variable, free stream velocity, U_∞, becomes larger while at the near-main-stream region, the free stream velocity becomes the dominant contributor to the uncertainty in the output x-velocity. The contribution from the other aleatory uncertain variable, actuation frequency, f, is significantly small compared to the other two uncertain variables.

4) Overall, for both long-time averaged and phase averaged quantities, the uncertainty in actuation frequency, f, is found to have a very small contribution to the uncertainty in the output.

5) For most of the output uncertainties (both long-time averaged and phase averaged), the combined contributions from different uncertainty sources are relatively small compared to the individual (non-combined) contribution of each uncertainty source.

6.2. FUTURE WORK

The results obtained from the current study show the potential of Second-Order Probability framework implemented with Non-Intrusive Polynomial Chaos as an efficient tool to perform the mixed uncertainty quantification analysis. There are several tasks remaining for further investigations.

Firstly, more uncertainty sources in the CFD modeling of synthetic jet flows, both model-form (e.g., physical models, boundary condition types) and inherent (e.g., amplitude of actuation, operating conditions), could be included in the uncertainty quantification analysis. A more comprehensive parametric study could help in finding more sources affecting the output quantities of interest.

Secondly, a more efficient scheme in the Second-Order Probability framework

with stochastic response surface could be developed and implemented. The current study utilizes sampling method to determine the ranges of the output quantities at each probability level thus requiring a large number of samples to evaluate the stochastic response surface constructed with NIPC. Considering the analytical nature of the polynomial chaos expansion (function of both aleatory and epistemic variables), an optimization scheme could be employed at both outer and inner loops of Second-Order Probability framework to determine the maximum and minimum of the output quantities of interest. This method requires much fewer function evaluations than sampling techniques, thus improving the computational efficiency even further comparing the original Monte Carlo method of CFD simulations.

Furthermore, the developed generic uncertainty quantification framework could be integrated to the analysis and robust design of synthetic jet applications. Due to the non-intrusive virtue, the uncertainty quantification method presented in this study could also be applied to other complex aerospace systems.

APPENDIX A

CFD SIMULATION SETUP PROCEDURE

In this appendix, the main setup procedure of the CFD simulation is presented. More detailed information can be found in ANSYS FLUENT User Manual [29].

GRID GENERATION

The grid used in the simulation was taken from the CFDVAL2004 workshop (labeled as "STRUCTURED 2D GRID #4"). The grid file can be pre-processed via GAMBIT to make a mesh file ready for ANSYS FLUENT.

ANSYS FLUENT CASE SETUP

Considering the computational expenses, all the simulation runs were performed on a high performance computing cluster. This section describes the main procedure of configuring simulation case via an interactive ANSYS FLUENT interface.

Launching Parallel ANSYS FLUENT

To run a parallel ANSYS FLUENT application interactively, an interactive session must be requested through a PBS scheduler. An example of requesting multiple nodes to run interactive ANSYS FLUENT is as follows:

qsub -I -X -l nodes $= 8 : ppn = 2$ *-l walltime* $= 40 : 00 : 00$ *-q hos_cpu@nic-cluster.mst.edu*

This will request 16 processors (8×2) for a wall time of 40 hours on the nodes labeled with "hos_cpu" dedicated to the Computational Fluid Dynamics and Aerospace Design Laboratory. It may take a couple of minutes before the requested nodes become available. Once the nodes are assigned, parallel ANSYS FLUENT can be launched interactively at the specified working directory using the following two lines of command:

cd /path_of_working_directory

$fluent$ *2ddp -t*16 *-pethernet -cnf=*$PBS\_NODEFILE$ *-ssh*

Case Setup

The following figures of screen-shot show the main steps to set up the ANSYS FLUENT case.

Step 1: Read in mesh file and save as case file.

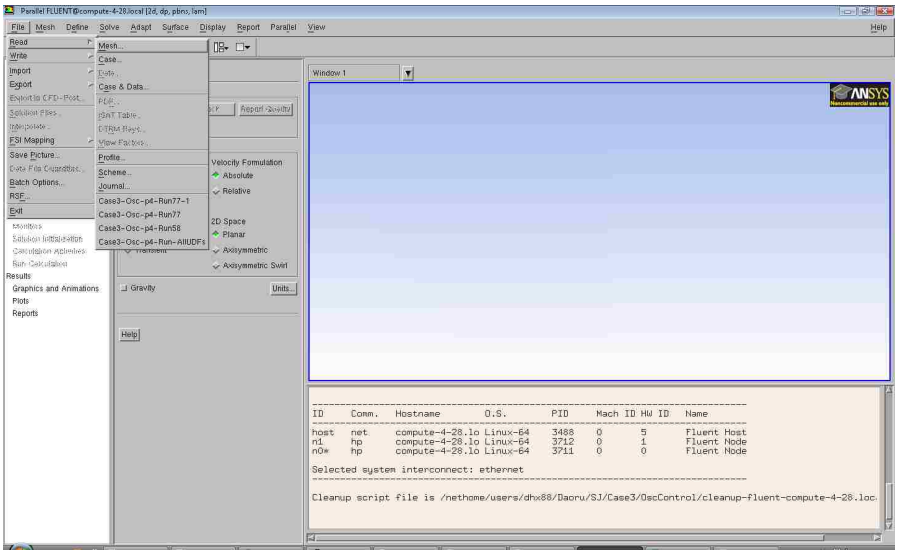

Figure A.1. Read mesh file into ANSYS FLUENT (Step 1).

Step 2: Set up the solver.

"Pressure-Based" solver was selected as well as the "Transient" time option due to the unsteady nature of the flow field studied. Velocity formulation was set as "Absolute" and "Planar" 2D space option was selected.

Step 3: Compile User-Defined Functions (UDF).

The unsteady boundary condition at the bottom of the actuator cavity and turbulence model were configured with User-Defined Functions (UDF). The UDF files were written in C language and compiled in ANSYS FLUENT. It should be noted that the directory of the UDF library file to be built and the working directory of ANSYS FLUENT should be the same in order to successfully compile the UDF.

Step 4: Set up fluid model.

The properties of the working fluid (air) were modified according to CFD-VAL2004 workshop test conditions.

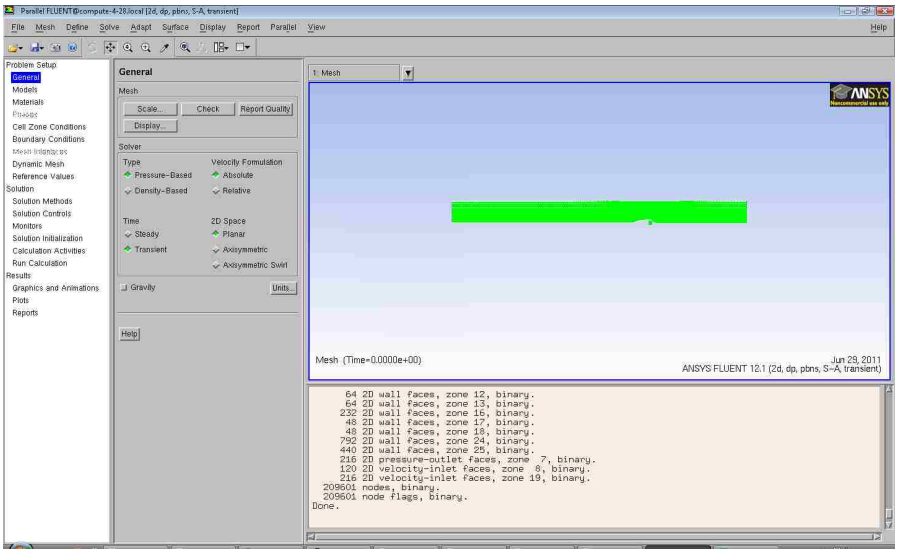

Figure A.2. Set up solver (Step 2).

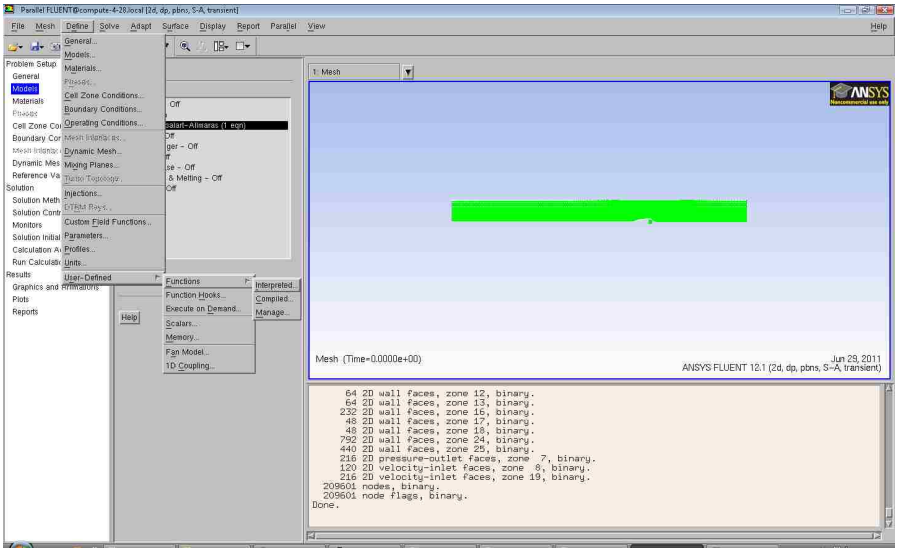

Figure A.3. Compile User-Defined Functions (Step 3).

Step 5: Set up turbulence model.

One-equation Spalart-Allmaras turbulence model was selected and modified using UDF when needed.

Step 6: Set up boundary conditions.

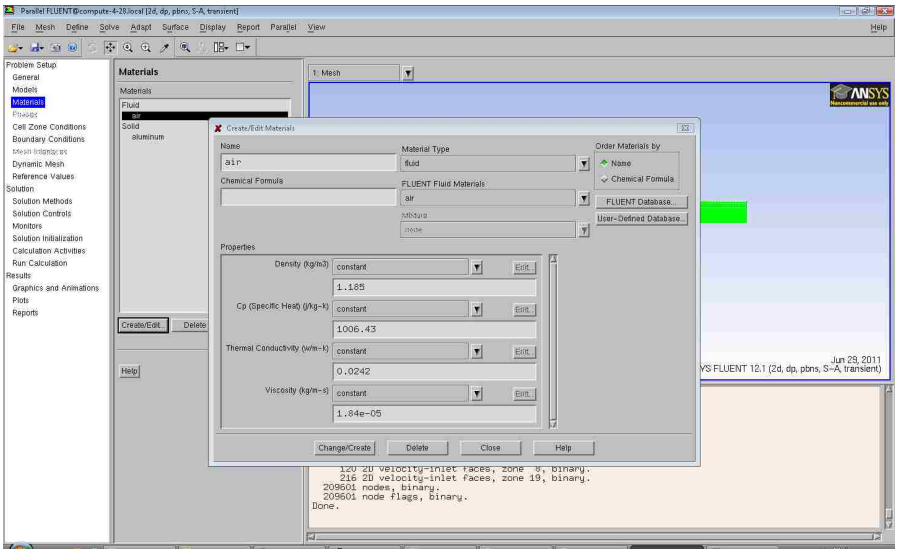

Figure A.4. Modify air properties (Step 4).

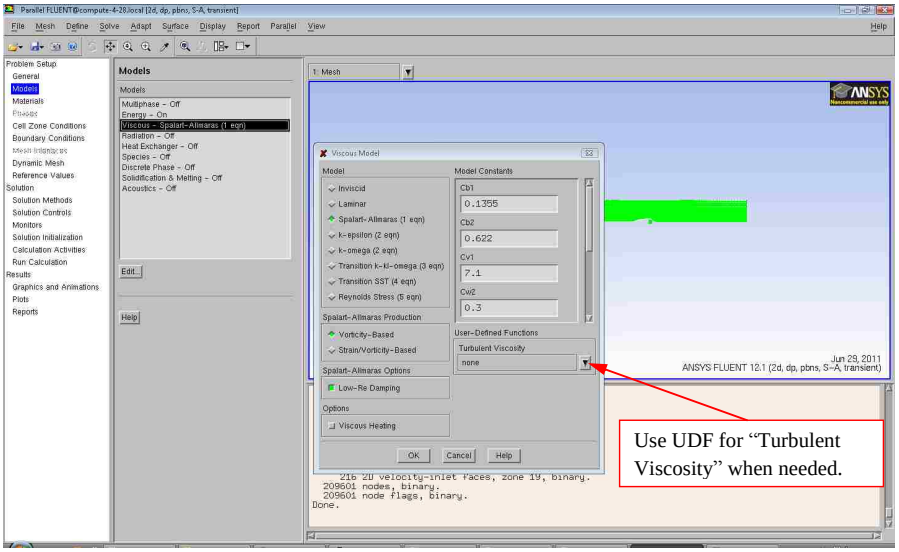

Figure A.5. Set up turbulence model (Step 5).

The example of setting up boundary condition at bottom of cavity is given in Figure A.6. The other boundary conditions were specified according to the descriptions in Section 4.2.3.

Step 7: Set up time step size.

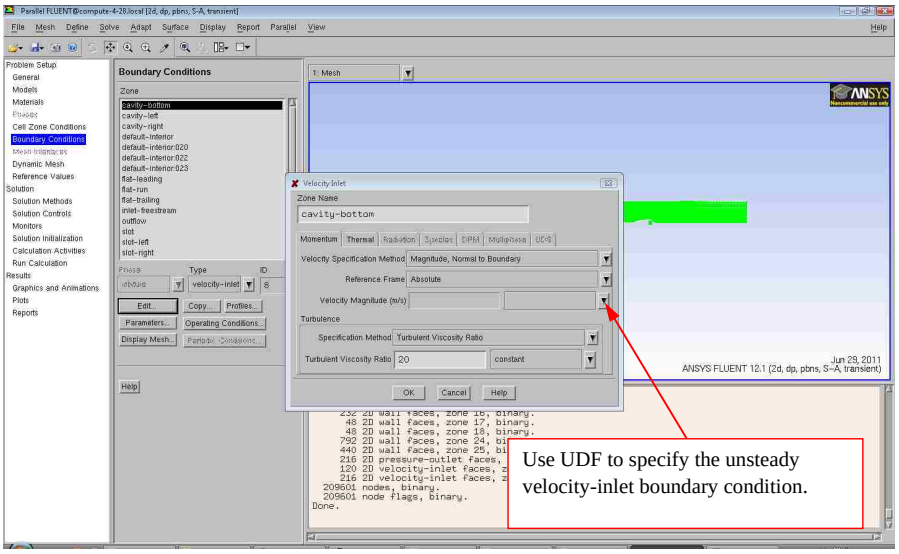

Figure A.6. Set up boundary conditions at bottom of cavity (Step 6).

The time step size was specified based on the frequency. 360 time steps were used within every cycle of actuation.

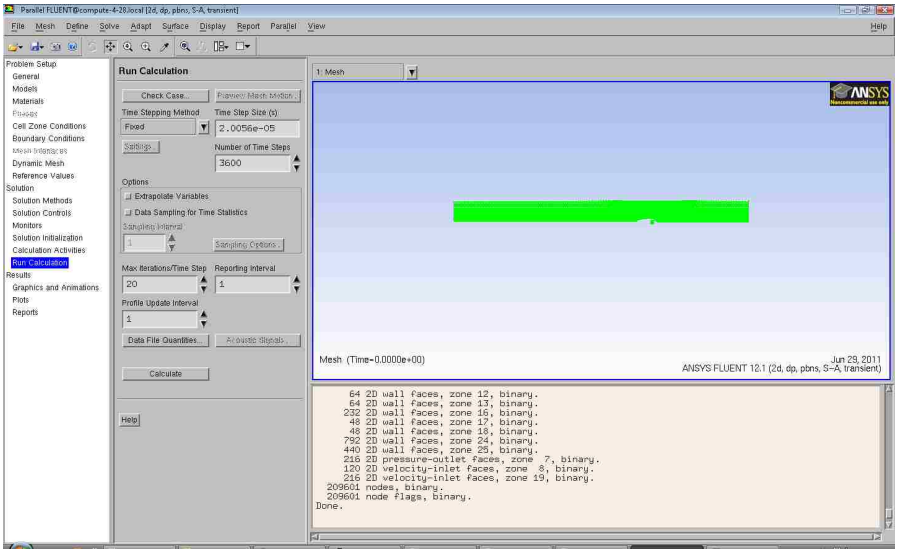

Figure A.7. Set up time step size (Step 7).

Step 8: Set up auto-save option.

Data were auto-saved during the calculation for the post-processing.

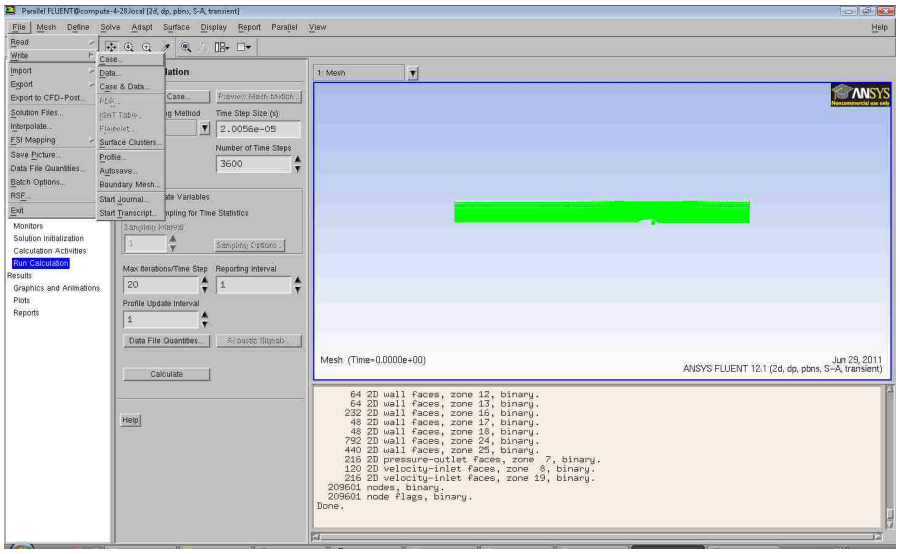

Figure A.8. Set up auto-save option (Step 8).

Step 9: Set up monitors.

Both residual monitors and surface monitors were selected to monitor and record solution history.

Step 10: Initialize the flow field.

The solver needs an initial flow field to start the calculation. In this study, initialization from free stream was selected to get the converged *steady* solution, then, all the unsteady calculations were started from the same (baseline) initial flow field.

Step 11: Set up solution methods.

The solution methods setup followed the descriptions in Section 4.2.2, as shown in Figure A.11.

Step 12: Check case file and save.

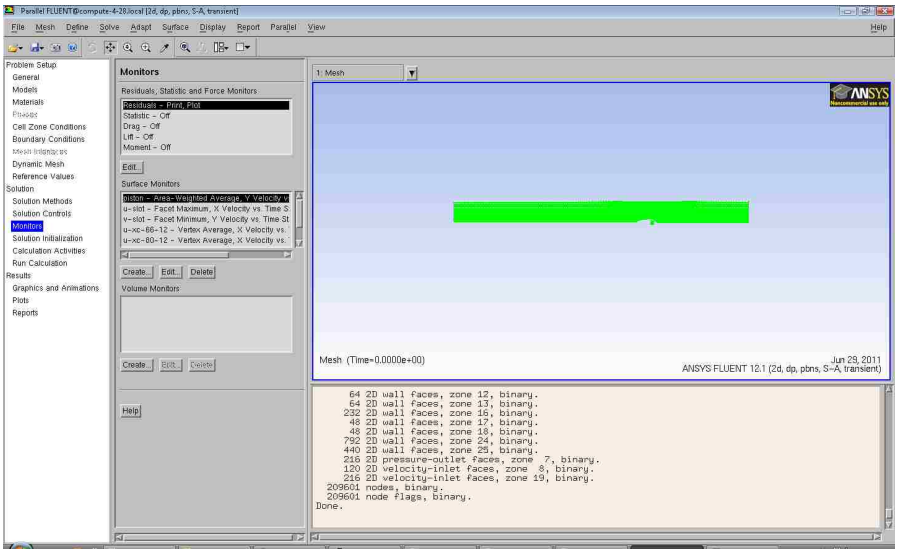

Figure A.9. Set up monitors (Step 9).

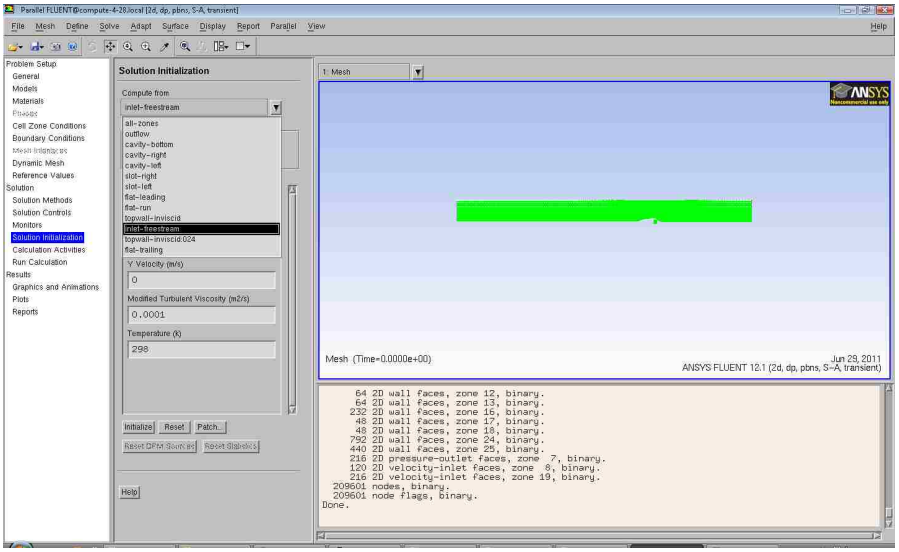

Figure A.10. Initialize the flow field (Step 10).

Once the setup was complete, save the case file and run the simulations in a batch mode.

RUNING ANSYS FLUENT WITH BATCH JOBS

Once the ANSYS FLUENT case is ready, it is more convenient to run all the

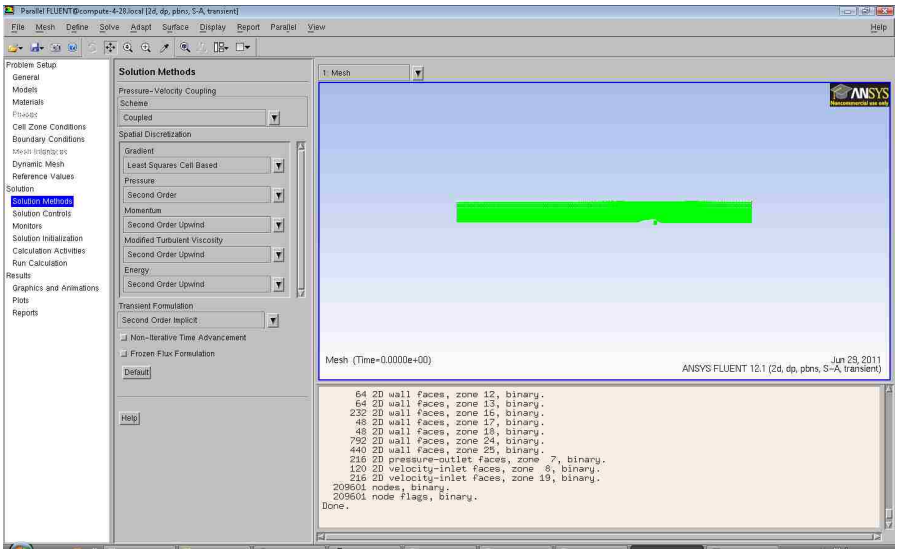

Figure A.11. Set up solution methods (Step 11).

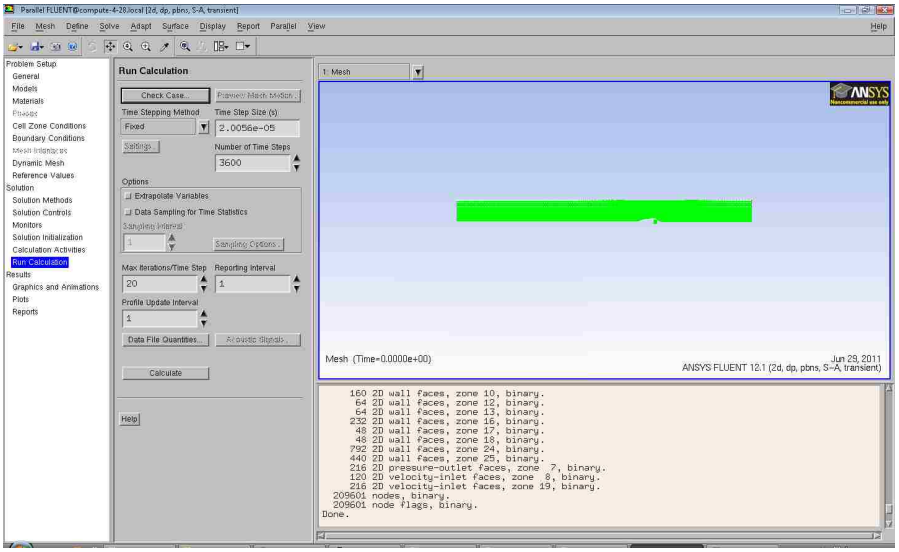

Figure A.12. Check case file (Step 12).

simulations in a batch mode by submitting job files. An example of the job file is as follows:

```
#!/bin/bash
```

```
#PBS -q hos_cpu@nic-cluster.mst.edu
#PBS -m abe
#PBS -M username@mail.mst.edu
#PBS -l nodes=3:ppn=2
#PBS -l walltime=100:00:00
#PBS -d /working_directory
fluent 2ddp -t6 -pethernet -cnf=$PBS_NODEFILE -g -ssh < /command_file
```

The above job file will launch the ANSYS FLUENT using the specified nodes and commands. A command file looks like the following:

```
/file/rc /working_directory/Case3-Osc-p4-RunS.cas
/file/rd /working_directory/Case3-Osc-Baseline_initial.dat
/file/autosave/data-frequency 0
/solve/dual-time-iterate
6480
20
/file/autosave/data-frequency 18
/solve/dual-time-iterate
720
20
/exit
```

APPENDIX B

MATLAB SOURCE CODE: POLYNOMIAL CHAOS EXPANSION AND SOBOL INDICES

CALCULATION OF COEFFICIENTS IN PCE

```
function Alpha=PolyCoef(p,AlphaStar)
%% This function calculate the coefficients (Alpha) in the PCE.
%% expansion using Tensor-Product Quadrature method.
%% Legendre polynomial due to uniform distribution
%% created 3/23/2011
%% revised 4/5/2011: Added p=2 case
%% revised 5/5/2011: Added p=3 case

%% revised 6/29/2011: Put into the appendix in my thesis.

%% Input 'AlphaStar', which is actually read from a text file.
%% Input 'p' is polynomial orders
%% for test use only
% close all
% clear all
% clc
% AlphaStar=[1:8];
%% Evaluate the denominator and numerator
% probability density function for uniform distribution included in the
% 'InnerProduct' function
%% number of terms in the polynomial chaos expansion
if p==1
    P=3;
elseif p==2
    P=9;
elseif p==3
    P=19;
elseif p==4
    P=34;
else
    display('Error! Not enough code information for this order of PCE!');
end
for j=0:P
    %% denominator, analytically
    Denominator(j+1)=LegendreInnerProduct(j);
    %% numerator, Gauss Quadrature rule
    Numerator(j+1)=LegendreGaussQuadrature(p,j,AlphaStar);
end
% Denominator
% Numerator

Alpha=Numerator./Denominator;
%% clear variables for speeding up
%% added 5/17/2011
% clearvars -except Alpha;
end

function InnerProduct=LegendreInnerProduct(i)
%% This function calculate the inner product of the 3-d Legendre
%% Polynomial.
%% revised 3/23/2011: Legendre Polynomial updated for 3 variables
%% revised 4/5/2011: Added terms for p=2 case.
%% revised 6/25/2011: Added terms for p=4 case.
```

```
%% Input 'i' is the term number in the Polynomial Chaos Expansion
%% i==0: p=0
%% i==1,2,3: p=1
%% i==4,5,...,9: p=2
%% i==10,11,...,19: p=3
%% i==20,21,...,34: p=4
if i==0
    InnerProduct=(LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))^3;
elseif i==1
    InnerProduct=(LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))^2 ;
elseif i==2
    InnerProduct=(LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))^2 ;
elseif i==3
    InnerProduct=(LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))^2 ;
elseif i==4
    InnerProduct=(LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))^2;
elseif i==5
    InnerProduct=(LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))^2*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));
elseif i==6
    InnerProduct=(LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))^2*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));
elseif i==7
    InnerProduct=(LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))^2;
elseif i==8
    InnerProduct=(LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))^2*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));
elseif i==9
    InnerProduct=(LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))^2;
elseif i==10
    InnerProduct=(LegendreIntegrated(3,1)-LegendreIntegrated(3,-1))*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))^2;
elseif i==11
    InnerProduct=(LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                 (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));
elseif i==12
    InnerProduct=(LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                 (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));
elseif i==13
    InnerProduct=(LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                 (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));
elseif i==14
    InnerProduct=(LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))^3;
elseif i==15
    InnerProduct=(LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
```

```
                (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));
elseif i==16
    InnerProduct=(LegendreIntegrated(3,1)-LegendreIntegrated(3,-1))*...
                (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))^2;
elseif i==17
    InnerProduct=(LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));
elseif i==18
    InnerProduct=(LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));
elseif i==19
    InnerProduct=(LegendreIntegrated(3,1)-LegendreIntegrated(3,-1))*...
                (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))^2;
%% p=4 terms
elseif i==20% (4 0 0)
    InnerProduct=(LegendreIntegrated(4,1)-LegendreIntegrated(4,-1))*...
                (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))*...
                (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));
elseif i==21% (3 1 0)
    InnerProduct=(LegendreIntegrated(3,1)-LegendreIntegrated(3,-1))*...
                (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));
elseif i==22% (3 0 1)
    InnerProduct=(LegendreIntegrated(3,1)-LegendreIntegrated(3,-1))*...
                (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))*...
                (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1));
elseif i==23% (2 2 0)
    InnerProduct=(LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                (LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));

elseif i==24% (2 1 1)
    InnerProduct=(LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1));
elseif i==25% (2 0 2)
    InnerProduct=(LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))*...
                (LegendreIntegrated(2,1)-LegendreIntegrated(2,-1));
elseif i==26% (1 3 0)
    InnerProduct=(LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                (LegendreIntegrated(3,1)-LegendreIntegrated(3,-1))*...
                (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));
elseif i==27% (1 2 1)
    InnerProduct=(LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                (LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1));
elseif i==28% (1 1 2)
    InnerProduct=(LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                (LegendreIntegrated(2,1)-LegendreIntegrated(2,-1));
```

```
elseif i==29% (1 0 3)
    InnerProduct=(LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))*...
                 (LegendreIntegrated(3,1)-LegendreIntegrated(3,-1));
elseif i==30% (0 4 0)
    InnerProduct=(LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))*...
                 (LegendreIntegrated(4,1)-LegendreIntegrated(4,-1))*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1));
elseif i==31% (0 3 1)
    InnerProduct=(LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))*...
                 (LegendreIntegrated(3,1)-LegendreIntegrated(3,-1))*...
                 (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1));
elseif i==32% (0 2 2)
    InnerProduct=(LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))*...
                 (LegendreIntegrated(2,1)-LegendreIntegrated(2,-1))*...
                 (LegendreIntegrated(2,1)-LegendreIntegrated(2,-1));
elseif i==33% (0 1 3)
    InnerProduct=(LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))*...
                 (LegendreIntegrated(1,1)-LegendreIntegrated(1,-1))*...
                 (LegendreIntegrated(3,1)-LegendreIntegrated(3,-1));
elseif i==34% (0 0 4)
    InnerProduct=(LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))*...
                 (LegendreIntegrated(0,1)-LegendreIntegrated(0,-1))*...
                 (LegendreIntegrated(4,1)-LegendreIntegrated(4,-1));
end
%% clear variables for speeding up
%% added 5/17/2011
% clearvars -except InnerProduct;

end

function I=LegendreIntegrated(i,x)
%% This function is the intergral of Legendre polynomial
%% orignially created for AIAA 2010-4411 work, Point-Collocation method
%% revised 3/23/2011: Legendre Polynomial updated for 3 variables
%% revised 4/5/2011: Added terms for p=2 case.
%% revised 5/5/2011: Added terms for p=3 case.
%% revised 6/25/2011: Added terms for p=4 case.
%% Integrated function (including the probability term 1/2)
%% 'i' is the term order in the Polynomial Chaos Expansion
if i==0
    I=1/2*(x);
elseif i==1
    I=1/6*(x^3);
elseif i==2
    I=1/8*((9/5)*x^5-2*x^3+x);
elseif i==3
    I=1/8*(25/7*x^7-6*x^5+3*x^3);
elseif i==4
    I=1/128*(35^2/9*x^9-300*x^7+222*x^5-60*x^3+9*x);
end
%% clear variables for speeding up
%% added 5/17/2011
% clearvars -except I;
```

```
end
```

CONSTRUCTION OF RESPONSE SURFACE

```
function R=ResponseSurface(Alpha,xi)
%% This function constructs the Response Surface using PCE
%% created 3/24/2011
%% Input 'AlphaStar' is read from the written file
%% Input 'xi' is a vector
%% for test use only
% close all
% clear all
% clc
% AlphaStar=[1:8];
% Alpha=PolyCoef(AlphaStar);
%%
%how many terms in the PCE
Nt=length(Alpha);

sum=0;
for j=1:Nt
    sum=sum+Alpha(j)*LegendrePolyTerm(j-1,xi);
end
R=sum;
%% clear variables for speeding up
%% added 5/17/2011
% clearvars -except R;

end

function Psi=LegendrePolyTerm(j,xi)
%% This function return the Legendre Polynomials Term
%% originially created for AIAA 2010-4411 work
%% revised 3/24/2011: update the terms for 3 variables
%% revised 4/5/2011: Added terms for p=2 case.
%% revised 5/5/2011: Added terms for p=3 case.
%% revised 6/25/2011: Added terms for p=4 case.
%% Input 'j' is order of the polynomial
%% Input 'xi' is a vector, xi(1),xi(2),xi(3)
%%
if j==0
    Psi=1;
elseif j==1
    Psi=xi(1);
elseif j==2
    Psi=xi(2);
elseif j==3
    Psi=xi(3);
elseif j==4
    Psi=0.5*(3*xi(1)^2-1);
elseif j==5
    Psi=xi(1)*xi(2);
elseif j==6
    Psi=xi(1)*xi(3);
elseif j==7
```

```
    Psi=0.5*(3*xi(2)^2-1);
elseif j==8
    Psi=xi(2)*xi(3);
elseif j==9
    Psi=0.5*(3*xi(3)^2-1);
elseif j==10
    Psi=0.5*(5*xi(1)^3-3*xi(1));
elseif j==11
    Psi=0.5*(3*xi(1)^2-1)*xi(2);
elseif j==12
    Psi=0.5*(3*xi(1)^2-1)*xi(3);
elseif j==13
    Psi=xi(1)*0.5*(3*xi(2)^2-1);
elseif j==14
    Psi=xi(1)*xi(2)*xi(3);
elseif j==15
    Psi=xi(1)*0.5*(3*xi(3)^2-1);
elseif j==16
    Psi=0.5*(5*xi(2)^3-3*xi(2));
elseif j==17
    Psi=0.5*(3*xi(2)^2-1)*xi(3);
elseif j==18
    Psi=xi(2)*0.5*(3*xi(3)^2-1);
elseif j==19
    Psi=0.5*(5*xi(3)^3-3*xi(3));
   %% p=4 case
elseif j==20
    Psi=(1/8)*(35*xi(1)^4-30*xi(1)^2+3);
elseif j==21
    Psi=0.5*(5*xi(1)^3-3*xi(1))*xi(2);
elseif j==22
    Psi=0.5*(5*xi(1)^3-3*xi(1))*xi(3);
elseif j==23
    Psi=0.5*(3*xi(1)^2-1)*0.5*(3*xi(2)^2-1);
elseif j==24
    Psi=0.5*(3*xi(1)^2-1)*xi(2)*xi(3);
elseif j==25
    Psi=0.5*(3*xi(1)^2-1)*0.5*(3*xi(3)^2-1);
elseif j==26
    Psi=xi(1)*0.5*(5*xi(2)^3-3*xi(2));
elseif j==27
    Psi=xi(1)*0.5*(3*xi(2)^2-1)*xi(3);
elseif j==28
    Psi=xi(1)*xi(2)*0.5*(3*xi(3)^2-1);
elseif j==29
    Psi=xi(1)*0.5*(5*xi(3)^3-3*xi(3));
elseif j==30
    Psi=(1/8)*(35*xi(2)^4-30*xi(2)^2+3);
elseif j==31
    Psi=0.5*(5*xi(2)^3-3*xi(2))*xi(3);
elseif j==32
    Psi=0.5*(3*xi(2)^2-1)*0.5*(3*xi(3)^2-1);
elseif j==33
    Psi=xi(2)*0.5*(5*xi(3)^3-3*xi(3));
```

```
elseif j==34
    Psi=(1/8)*(35*xi(3)^4-30*xi(3)^2+3);
end
%% clear variables for speeding up
%% added 5/17/2011
% clearvars -except Psi;
end
```

CALCULATION OF CDF

```
function CDF=MyCDF(R)
%% Input R is sorted already.
%% Comulative Distribution Function
%% This is my CDF function
%% originally created for AIAA 2010-4411 work
%% revised 3/24/2011
%% modified 5/17/2011 - write my own CDF function
%% This is the original built-in function
% R_cdf=unifcdf(R,min(R),max(R));
%% This is my function
N=length(R);
CDF=zeros(1,N);
for j=1:N
    CDF(j)=j/N;
end

%% clear the memory
% clearvars -except CDF;
end
```

CALCULATION OF SOBOL INDICES

```
function [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
%% This function calculate the Sobol indices.
%% created 5/12/2011 in MST library
%% revised 6/25/2011: Added the terms for p=4 case.

%% revised 6/29/2011: Put in the appendix of my thesis.
%%
P=length(Alpha)-1;% number of terms in PCE
%% Total variance (D)
D=0;
for j=1:P
    D=D+Alpha(j+1)^2*LegendreInnerProduct(j);% note the Psi_0 term
end
% D
%% Partial variance
if P==3% p=1 case
    D1=Alpha(2)^2*LegendreInnerProduct(1);
    D2=Alpha(3)^2*LegendreInnerProduct(2);
    D3=Alpha(4)^2*LegendreInnerProduct(3);
    % S
    S1=D1/D;
    S2=D2/D;
    S3=D3/D;
    ST1=S1;
```

```
    ST2=S2;
    ST3=S3;
    %% check the unity
    disp('%%%%%check the sum of all Sobol indicesk,p=1%%%%%');
    S_Sum=S1+S2+S3
elseif P==9% p=2 case
    D1=Alpha(2)^2*LegendreInnerProduct(1)...
        +Alpha(5)^2*LegendreInnerProduct(4);
    D2=Alpha(3)^2*LegendreInnerProduct(2)...
        +Alpha(8)^2*LegendreInnerProduct(7);
    D3=Alpha(4)^2*LegendreInnerProduct(3)...
        +Alpha(10)^2*LegendreInnerProduct(9);
    D12=Alpha(6)^2*LegendreInnerProduct(5);
    D13=Alpha(7)^2*LegendreInnerProduct(6);
    D23=Alpha(9)^2*LegendreInnerProduct(8);
    % S
    S1=D1/D;
    S2=D2/D;
    S3=D3/D;
    S12=D12/D;
    S13=D13/D;
    S23=D23/D;
    ST1=S1+S12+S13;
    ST2=S2+S12+S23;
    ST3=S3+S13+S23;
    %% check the unity
    disp('%%%%%check the sum of all Sobol indices,p=2%%%%%');
    S_Sum=S1+S2+S3+S12+S13+S23
elseif P==19% p=3 case
    D1=Alpha(2)^2*LegendreInnerProduct(1)...
        +Alpha(5)^2*LegendreInnerProduct(4)...
        +Alpha(11)^2*LegendreInnerProduct(10);
    D2=Alpha(3)^2*LegendreInnerProduct(2)...
        +Alpha(8)^2*LegendreInnerProduct(7)...
        +Alpha(17)^2*LegendreInnerProduct(16);
    D3=Alpha(4)^2*LegendreInnerProduct(3)...
        +Alpha(10)^2*LegendreInnerProduct(9)...
        +Alpha(20)^2*LegendreInnerProduct(19);
    % combined - two
    D12=Alpha(6)^2*LegendreInnerProduct(5)...
        +Alpha(12)^2*LegendreInnerProduct(11)...
        +Alpha(14)^2*LegendreInnerProduct(13);
    D13=Alpha(7)^2*LegendreInnerProduct(6)...
        +Alpha(13)^2*LegendreInnerProduct(12)...
        +Alpha(16)^2*LegendreInnerProduct(15);
    D23=Alpha(9)^2*LegendreInnerProduct(8)...
        +Alpha(18)^2*LegendreInnerProduct(17)...
        +Alpha(19)^2*LegendreInnerProduct(18);
    % combined - three
    D123=Alpha(15)^2*LegendreInnerProduct(14);
    % S
    S1=D1/D;
    S2=D2/D;
    S3=D3/D;
```

```
    S12=D12/D;
    S13=D13/D;
    S23=D23/D;
    S123=D123/D;
    ST1=S1+S12+S13+S123;
    ST2=S2+S12+S23+S123;
    ST3=S3+S13+S23+S123;
    %% check the unity
    disp('%%%%%check the sum of all Sobol indices,p=3%%%%%');
    S_Sum=S1+S2+S3+S12+S13+S23+S123
elseif P==4% p=4 case
    %% For the firt 19 terms (Psi_0 to Psi_19), the same as 'p=3' case,
    %% just need to add the terms from 4th order PCE.
    D1=Alpha(2)^2*LegendreInnerProduct(1)...
        +Alpha(5)^2*LegendreInnerProduct(4)...
        +Alpha(11)^2*LegendreInnerProduct(10)...
        +Alpha(21)^2*LegendreInnerProduct(20);
    D2=Alpha(3)^2*LegendreInnerProduct(2)...
        +Alpha(8)^2*LegendreInnerProduct(7)...
        +Alpha(17)^2*LegendreInnerProduct(16)...
        +Alpha(31)^2*LegendreInnerProduct(30);
    D3=Alpha(4)^2*LegendreInnerProduct(3)...
        +Alpha(10)^2*LegendreInnerProduct(9)...
        +Alpha(20)^2*LegendreInnerProduct(19)...
        +Alpha(35)^2*LegendreInnerProduct(34);
    % combined - two
    D12=Alpha(6)^2*LegendreInnerProduct(5)...
        +Alpha(12)^2*LegendreInnerProduct(11)...
        +Alpha(14)^2*LegendreInnerProduct(13)...
        +Alpha(22)^2*LegendreInnerProduct(21)...
        +Alpha(24)^2*LegendreInnerProduct(23)...
        +Alpha(27)^2*LegendreInnerProduct(26);
    D13=Alpha(7)^2*LegendreInnerProduct(6)...
        +Alpha(13)^2*LegendreInnerProduct(12)...
        +Alpha(16)^2*LegendreInnerProduct(15)...
        +Alpha(23)^2*LegendreInnerProduct(22)...
        +Alpha(26)^2*LegendreInnerProduct(25)...
        +Alpha(30)^2*LegendreInnerProduct(29);
    D23=Alpha(9)^2*LegendreInnerProduct(8)...
        +Alpha(18)^2*LegendreInnerProduct(17)...
        +Alpha(19)^2*LegendreInnerProduct(18)...
        +Alpha(32)^2*LegendreInnerProduct(31)...
        +Alpha(33)^2*LegendreInnerProduct(32)...
        +Alpha(34)^2*LegendreInnerProduct(33);
    % combined - three
    D123=Alpha(15)^2*LegendreInnerProduct(14)...
        +Alpha(25)^2*LegendreInnerProduct(24)...
        +Alpha(28)^2*LegendreInnerProduct(27)...
        +Alpha(29)^2*LegendreInnerProduct(28);
    % S
    S1=D1/D;
    S2=D2/D;
    S3=D3/D;
    S12=D12/D;
```

```
    S13=D13/D;
    S23=D23/D;
    S123=D123/D;
    ST1=S1+S12+S13+S123;
    ST2=S2+S12+S23+S123;
    ST3=S3+S13+S23+S123;
    %% check the unity
    disp('%%%%%check the sum of all Sobol indices,p=4%%%%%');
    S_Sum=S1+S2+S3+S12+S13+S23+S123
    %% also print the result
    str=['S1, S2, S3, ST1, ST2, ST3'];
    %disp(str);
    data=[S1,
        S2,
        S3,
        ST1
        ST2,
        ST3];
    %disp(data);
end

%% clear variables for speeding up
%% added 5/17/2011
% clearvars -except S1 S2 S3 ST1 ST2 ST3;
end
```

APPENDIX C

MATLAB SOURCE CODE: UNCERTAINTY QUANTIFICATION OF LONG-TIME AVERAGED SEPARATION BUBBLE CHARACTERISTICS

```
function Bubble
%% This function investigates the separation bubble characteristics
%% created 3/24/2011

%% revised 3/30/2011: bubble size

%% revised 5/11/2011:
%% 1) Polynomial Chaos Expansion Degree Convergence: return the mean value
%% in the pure aleatory expansion.
%% 2) Return the CDF in the pure aleatory expansion to check the degree
%% convergence.

%% revised 5/17/2011: plot 'median' value rather than 'mean' value for the
%% convergence check

%% revised 6/26/2011: Update the figures for the thesis.

%% revised 6/29/2011: Put in the appendix of my thesis.
%%
tic;
close all
clear all
clc
% format long;
format short;
p=4;
p=1:4;
FigNum=1;
% the order of sample number (aleatory), i.e. '2' means 'E2' samples
SampleNumOrder=4;
%%
pNum=length(p);
%% preallocate the arrays
Median_s=zeros(1,pNum);% separation
Median_r=zeros(1,pNum);% reattachment
Median_bubble=zeros(1,pNum);% bubble size
R_s=zeros(pNum,10^SampleNumOrder);
R_r=zeros(pNum,10^SampleNumOrder);
R_bubble=zeros(pNum,10^SampleNumOrder);
CDF_s=zeros(pNum,10^SampleNumOrder);
CDF_r=zeros(pNum,10^SampleNumOrder);
CDF_bubble=zeros(pNum,10^SampleNumOrder);
for i=1:pNum
    [FigNum,Median_s(i),R_s(i,:),CDF_s(i,:)]...
        =SeparationLocation(p(i),FigNum,SampleNumOrder);
    [FigNum,Median_r(i),R_r(i,:),CDF_r(i,:)]...
        =ReattachmentLocation(p(i),FigNum,SampleNumOrder);
    [FigNum,Median_bubble(i),R_bubble(i,:),CDF_bubble(i,:)]...
        =BubbleSize(p(i),FigNum,SampleNumOrder);
end
% Median_s
% Median_r
% Median_bubble
%% plot the PCE degree convergence check
```

```
%% separation location
figure(FigNum);
hold all;
plot(p,Median_s,'bo--');
axis([1 4 0.660 0.664]);
xlabel('Polynomial Chaos Expansion Degree, p','FontSize',14);
ylabel('Median Separation Location, x/c','FontSize',14);
title('Pure Aleatory Expansion','FontSize',14);
set(gca,'FontSize',18);
FigNum=FigNum+1;% update the figure number
figure(FigNum);
hold all;
plot(R_s(1,:),CDF_s(1,:),'k-','LineWidth',3.0);
plot(R_s(2,:),CDF_s(2,:),'k--','LineWidth',2.5);
plot(R_s(3,:),CDF_s(3,:),'b-.','LineWidth',2.0);
plot(R_s(4,:),CDF_s(4,:),'r:','LineWidth',1.5);
xlabel('Long-time Averaged Separation Location, x/c','FontSize',14);
ylabel('CDF','FontSize',14);
title('Pure Aleatory Expansion','FontSize',14);
set(gca,'FontSize',18);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot...
    =legend(legend1,legend2,legend3,legend4,'Location','SouthEast');
set(legend_plot,'FontSize',14);
FigNum=FigNum+1;% update the figure number
%% reattachment location
figure(FigNum);
hold all;
plot(p,Median_r,'bo--');
axis([1 4 1.18 1.22]);
xlabel('Polynomial Chaos Expansion Degree, p','FontSize',14);
ylabel('Median Reattachment Location, x/c','FontSize',14);
title('Pure Aleatory Expansion','FontSize',14);
set(gca,'FontSize',18);
FigNum=FigNum+1;% update the figure number
figure(FigNum);
hold all;
plot(R_r(1,:),CDF_r(1,:),'k-','LineWidth',3.0);
plot(R_r(2,:),CDF_r(2,:),'k--','LineWidth',2.5);
plot(R_r(3,:),CDF_r(3,:),'b-.','LineWidth',2.0);
plot(R_r(4,:),CDF_r(4,:),'r:','LineWidth',1.5);
xlabel('Long-time Averaged Reattachment Location, x/c','FontSize',14);
ylabel('CDF','FontSize',14);
title('Pure Aleatory Expansion','FontSize',14);
set(gca,'FontSize',18);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot...
    =legend(legend1,legend2,legend3,legend4,'Location','SouthEast');
```

```
set(legend_plot,'FontSize',14);
FigNum=FigNum+1;% update the figure number
%% bubble size
figure(FigNum);
hold all;
plot(p,Median_bubble,'bo--');
axis([1 4 0.53 0.55]);
xlabel('Polynomial Chaos Expansion Degree, p','FontSize',14);
ylabel('Median Bubble Size, x/c','FontSize',14);
title('Pure Aleatory Expansion','FontSize',14);
set(gca,'FontSize',18);
FigNum=FigNum+1;% update the figure number
figure(FigNum);
hold all;
plot(R_bubble(1,:),CDF_bubble(1,:),'k-','LineWidth',3.0);
plot(R_bubble(2,:),CDF_bubble(2,:),'k--','LineWidth',2.5);
plot(R_bubble(3,:),CDF_bubble(3,:),'b-.','LineWidth',2.0);
plot(R_bubble(4,:),CDF_bubble(4,:),'r:','LineWidth',1.5);
xlabel('Long-time Averaged Bubble Size, x/c','FontSize',14);
ylabel('CDF','FontSize',14);
title('Pure Aleatory Expansion','FontSize',14);
set(gca,'FontSize',18);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot...
    =legend(legend1,legend2,legend3,legend4,'Location','SouthEast');
set(legend_plot,'FontSize',14);
FigNum=FigNum+1;% update the figure number
toc;
end

function [FigNum,Median,R_Aleatory,CDF]...
    =SeparationLocation(p,FigNum,SampleNumOrder)
%% This function construct the response surface for separation location
%% and also use the Second-Order Probability sampling to creat the CDFs.
%% created 3/24/2011

%% revised 5/11/2011: Polynomial Chaos Expansion Degree Convergence check
%% returning mean value and CDF in pure aleatory expansion.

%% revised 5/12/2011: add the Sobol indices part

%% revised 5/16/2011: clear variables for speeding

%% revised 5/17/2011: return median value rather than mean value

%% revised 6/29/2011: Put in the appendix of my thesis.

%% Input 'p' is the order of polynomial chaos expansion
%% Input 'FigNum' is the figure number
%% for test use only
% close all
```

```
% clear all
% clc
% p=1;
% FigNum=1;
% format long
% format short
%% Read the separation location data
path=['C:\Users\dhx88\Daoru\Case3\C3\POST\ThesisWork\'];
str=['LongSeparationLocation-p',num2str(p),'.txt'];
filestr=[path,str];
str=['separation location data from the runs, p = ',num2str(p),':'];
disp(str);
[RunNum,xc]=textread(filestr,'%d %12.8f','delimiter',',');
%% just to test the code
% xc=0.662*ones((p+1)^3,1);
%%
Alpha=PolyCoef(p,xc);
%% Sobol Indices part
%% added 5/12/2011
display('Sobol Indices for Seperation Location:');
[S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)

% RunNum
% xc
%% Second-Order Probability sampling with Constructed Response Surface
%% Epistemic loop
xi1=[-1.00:0.5:1.00];
%% sample epistemic loop
SampleStr_1D='SamplesE2_1D.txt';
SampleFile_1D=[path,SampleStr_1D];
xi1=textread(SampleFile_1D,'%f');
%%
EpistemicNum=length(xi1);
figure(FigNum);
hold all
for i=1:EpistemicNum
    %% Aleatory loop
    %% read the samples
    % SampleStr=['SamplesE3.txt'];
    SampleStr=['SamplesE',num2str(SampleNumOrder),'.txt'];
    SampleFile=[path,SampleStr];
    [xi2,xi3]=textread(SampleFile,'%f %f','delimiter',',');
    SampleNum=length(xi2);
%     xi2
%     xi3
    %% response surface evaluation
    R=zeros(1,SampleNum);
    for j=1:SampleNum
        xi=[xi1(i),xi2(j),xi3(j)];
        R(j)=ResponseSurface(Alpha,xi);
    end
%     R
    %% Sort and calculate the CDF
    %% Use my own CDF function - modified 5/17/2011
```

```
    R=sort(R);
    R_cdf=MyCDF(R);
    % [R_cdf,R]=Sortxy(R_cdf,R); - No need to sort again
    plot(R,R_cdf,'b-','LineWidth',1);
    xlabel('Long-time Averaged Separation Location, x/c','FontSize',18);
    ylabel('CDF','FontSize',18);
    titlestr=['Polynomial Chaos Degree p = ',num2str(p)];
    title(titlestr,'FontSize',18);
    set(gca,'FontSize',18);
    axis([0.66,0.664,0,1.0]);
end% epistemic loop

%% Pure aleatory expansion, return the mean value to check the PCE degree
%% convergence.
%% added 5/11/2011
%% read the samples
% SampleStr_Aleatory=['SamplesE2_3D.txt'];
SampleStr_Aleatory=['SamplesE',num2str(SampleNumOrder),'_3D.txt'];
SampleFile=[path,SampleStr_Aleatory];
[xi1,xi2,xi3]=textread(SampleFile,'%f %f %f','delimiter',',');
SampleNum_Aleatory=length(xi1);
%% response surface evaluation
R_Aleatory=zeros(1,SampleNum_Aleatory);
for j=1:SampleNum_Aleatory
    xi_Aleatory=[xi1(j),xi2(j),xi3(j)];
    R_Aleatory(j)=ResponseSurface(Alpha,xi_Aleatory);
end
%% median value
Median=median(R_Aleatory);
%% Use my own CDF function - modified 5/17/2011
R_Aleatory=sort(R_Aleatory);
CDF=MyCDF(R_Aleatory);
% [CDF,R_Aleatory]=Sortxy(CDF,R_Aleatory);
% [R_Aleatory,CDF]=Sortxy(R_Aleatory,CDF)
%% update the figure number
FigNum=FigNum+1;
%% histogram
figure(FigNum);
hold all;
hist(R_Aleatory,100);
xlabel('Long-time Averaged Separation Location, x/c','FontSize',18);
titlestr=['Polynomial Chaos Degree p = ',num2str(p)];
title(titlestr,'FontSize',18);
FigNum=FigNum+1;
%% clear variables for speeding up
%% added 5/16/2011
% clearvars -except FigNum Median R_Aleatory CDF;
end

function [FigNum,Median,R_Aleatory,CDF]...
    =ReattachmentLocation(p,FigNum,SampleNumOrder)
%% This function construct the response surface for reattachment location
%% and also use the Second-Order Probability sampling to creat the CDFs.
%% created 3/24/2011
```

```
%% revised 5/11/2011: Polynomial Chaos Expansion Degree Convergence check
%% returning mean value and CDF in pure aleatory expansion.

%% revised 5/12/2011: add the Sobol indices part

%% revised 5/16/2011: clear variables for speeding

%% revised 5/17/2011: return median value rather than mean value

%% revised 6/29/2011: Put in the appendix of my thesis.

%% Input 'p' is the order of polynomial chaos expansion
%% Input 'FigNum' is the figure number
%% for test use only
% close all
% clear all
% clc
% p=1;
% FigNum=1;
% format long
% format short
%% Read the reattachment location data
path=['C:\Users\dhx88\Daoru\Case3\C3\POST\ThesisWork\'];
str=['LongReattachmentLocation-p',num2str(p),'.txt'];
filestr=[path,str];
str=['reattachment location data from the runs, p = ',num2str(p),':'];
disp(str);
[RunNum,xc]=textread(filestr,'%d %12.8f','delimiter',',')
str=['Coef. for Reattachment Location'];
% disp(str);
Alpha=PolyCoef(p,xc);
Alpha_T=Alpha';% for Yi's use only
%% Sobol Indices part
%% added 5/12/2011
display('Sobol Indices for Reattachment Location:');
[S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)

% RunNum
% xc
%% Second-Order Probability sampling with Constructed Response Surface
%% Epistemic loop
xi1=[-1.00:0.5:1.00];
%% sample epistemic loop
SampleStr_1D='SamplesE2_1D.txt';
SampleFile_1D=[path,SampleStr_1D];
xi1=textread(SampleFile_1D,'%f');
%%
EpistemicNum=length(xi1);
figure(FigNum);
hold all
for i=1:EpistemicNum
    %% Aleatory loop
    %% read the samples
```

```
    % SampleStr=['SamplesE3.txt'];
    SampleStr=['SamplesE',num2str(SampleNumOrder),'.txt'];
    SampleFile=[path,SampleStr];
    [xi2,xi3]=textread(SampleFile,'%f %f','delimiter',',');
    SampleNum=length(xi2);
%      xi2
%      xi3
    %% response surface evaluation
    R=zeros(1,SampleNum);
    for j=1:SampleNum
        xi=[xi1(i),xi2(j),xi3(j)];
        R(j)=ResponseSurface(Alpha,xi);
    end
%     R
    %% Sort and calculate the CDF
    %% Use my own CDF function - modified 5/17/2011
    R=sort(R);
    R_cdf=MyCDF(R);
    % R_cdf=UniCDF(R);
    % [R_cdf,R]=Sortxy(R_cdf,R); - No need to sort again
    plot(R,R_cdf,'b-','LineWidth',1);
    xlabel('Long-time Averaged Reattachment Location, x/c','FontSize',18);
    ylabel('CDF','FontSize',18);
    titlestr=['Polynomial Chaos Degree p = ',num2str(p)];
    title(titlestr,'FontSize',18);
    set(gca,'FontSize',18);
    axis([1.0,1.4,0,1.0]);
end% epistemic loop

%% Pure aleatory expansion, return the mean value to check the PCE degree
%% convergence.
%% added 5/11/2011
%% read the samples
% SampleStr_Aleatory=['SamplesE2_3D.txt'];
SampleStr_Aleatory=['SamplesE',num2str(SampleNumOrder),'_3D.txt'];
SampleFile=[path,SampleStr_Aleatory];
[xi1,xi2,xi3]=textread(SampleFile,'%f %f %f','delimiter',',');
SampleNum_Aleatory=length(xi1);
%% response surface evaluation
R_Aleatory=zeros(1,SampleNum_Aleatory);
for j=1:SampleNum_Aleatory
    xi_Aleatory=[xi1(j),xi2(j),xi3(j)];
    R_Aleatory(j)=ResponseSurface(Alpha,xi_Aleatory);
end
%% median value
Median=median(R_Aleatory);
%% Use my own CDF function - modified 5/17/2011
R_Aleatory=sort(R_Aleatory);
CDF=MyCDF(R);
% CDF=UniCDF(R_Aleatory);
% [CDF,R_Aleatory]=Sortxy(CDF,R_Aleatory);
%% update the figure number
FigNum=FigNum+1;
%% histogram
```

```
figure(FigNum);
hold all;
hist(R_Aleatory,100);
xlabel('Long-time Averaged Reattachment Location, x/c','FontSize',18);
titlestr=['Polynomial Chaos Degree p = ',num2str(p)];
title(titlestr,'FontSize',18);
FigNum=FigNum+1;
%% clear variables for speeding up
%% added 5/16/2011
% clearvars -except FigNum Median R_Aleatory CDF;
end

function [FigNum,Median,R_Aleatory,CDF]=BubbleSize(p,FigNum,SampleNumOrder)
%% This function construct the response surface for bubble size and
%% also use the Second-Order Probability sampling to creat the CDFs.
%% created 3/30/2011: modified from 'ReattachmentLocation'

%% revised 4/7/2011: Added the sample distribution plot for bubble size.

%% revised 4/8/2011: Added the sample distribution plot for
%% both separation and reattachment locations.

%% revised 5/11/2011: Polynomial Chaos Expansion Degree Convergence check
%% returning mean value and CDF in pure aleatory expansion.

%% revised 5/12/2011: add the Sobol indices part

%% revised 5/16/2011: clear variables for speeding

%% revised 5/17/2011: return median value rather than mean value

%% revised 6/23/2011: subplot of a sample size convergence study

%% Input 'p' is the order of polynomial chaos expansion
%% Input 'FigNum' is the figure number
%% for test use only
% close all
% clear all
% clc
% p=4;
% FigNum=1;
% SampleNumOrder=3;
% format long;
% format short;
%% Read the reattachment location data
path=['C:\Users\dhx88\Daoru\Case3\C3\POST\ThesisWork\'];
str=['LongReattachmentLocation-p',num2str(p),'.txt'];
filestr=[path,str];
[RunNum,xc_reattachment]=textread(filestr,'%d %12.8f','delimiter',',');
% RunNum
% xc
%% Read the separation location data
path=['C:\Users\dhx88\Daoru\Case3\C3\POST\ThesisWork\'];
str=['LongSeparationLocation-p',num2str(p),'.txt'];
```

```
filestr=[path,str];
[RunNum,xc_separation]=textread(filestr,'%d %12.8f','delimiter',',');
% RunNum
% xc
%% calculate the bubble size
bubblesize=xc_reattachment-xc_separation;
Alpha_separation=PolyCoef(p,xc_separation);% separation
Alpha_reattachment=PolyCoef(p,xc_reattachment);% reattachment
Alpha_bubblesize=PolyCoef(p,bubblesize);% bubble size
Alpha=Alpha_bubblesize;
%% Sobol Indices part
%% added 5/12/2011
display('Sobol Indices for Bubble Size:');
[S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)

%% Second-Order Probability sampling with Constructed Response Surface
%% Epistemic loop
xi1=[-1.00:0.02:1.00];
%% sample epistemic loop
SampleStr_1D='SamplesE2_1D.txt';
SampleFile_1D=[path,SampleStr_1D];
xi1=textread(SampleFile_1D,'%f');
%%
EpistemicNum=length(xi1);
HorsetailPlot=figure(FigNum);
hold all;
%% ========================================================================
%% When performing the sample size convergence study, use the following
%% script and the 'subplot'.
%% added 6/23/2011
% set the position of the plot
% [left bottom width height]
% by default, [440 378 560 420]
% This size fits the pdf file of A4 paper.
% set(HorsetailPlot,'Position',[440 378 900 420]);
% subplot(1,2,1);
% hold all;
%% ========================================================================
for i=1:EpistemicNum
    %% Aleatory loop
    %% read the samples
    % SampleStr=['SamplesE3.txt'];
    SampleStr=['SamplesE',num2str(SampleNumOrder),'.txt'];
    SampleFile=[path,SampleStr];
    [xi2,xi3]=textread(SampleFile,'%f %f','delimiter',',');
    SampleNum=length(xi2);
%       xi2
%       xi3
    %% response surface evaluation
    R=zeros(1,SampleNum);
    for j=1:SampleNum
        xi=[xi1(i),xi2(j),xi3(j)];
        R(j)=ResponseSurface(Alpha,xi);
    end
```

```
%     R
    %% Sort and calculate the CDF
    %% Use my own CDF function - modified 5/17/2011
    R=sort(R);
    R_cdf=MyCDF(R);
    % R_cdf=UniCDF(R);
    % [R_cdf,R]=Sortxy(R_cdf,R); - No need to sort again
    plot(R,R_cdf,'b-','LineWidth',1);
    xlabel('Long-time Averaged Bubble Size, x/c','FontSize',18);
    ylabel('CDF','FontSize',18);
    titlestr=['Polynomial Chaos Degree p = ',num2str(p)];
    title(titlestr,'FontSize',18);
    set(gca,'FontSize',18);
    grid on;
    %axis([0.10,0.50,0,1.0]);
end% epistemic loop
FigNum=FigNum+1;
%% Sample distribution for the Pure Aleatory Expansion
% SampleStr_Aleatory=['SamplesE4_3D.txt'];
SampleStr_Aleatory=['SamplesE',num2str(SampleNumOrder),'_3D.txt'];
SampleFile=[path,SampleStr_Aleatory];
[xi1,xi2,xi3]=textread(SampleFile,'%f %f %f','delimiter',',');
SampleNum_Aleatory=length(xi2);
%% project to the actual value
%factor K in the turbulent viscosity
a=0.5;
b=2.0;
%project to [0.5,2.0] interval
x1=(b+a)/2+(b-a)/2.*xi1;
K=x1;
%free stream velocity and actuator frequency
a=0.9;
b=1.1;
x2=(b+a)/2+(b-a)/2.*xi2;
%free stream velocity and frequency have the same interval and bounds
x3=x2;
Uinf=34.6*x2;
f=138.5*x3;
%% Pure Aleatory response surface evaluation and plot
SamplePlot=figure(FigNum);
%% set the figure window size
%get(SamplePlot)
set(SamplePlot,'Position',[50 50 1024 768]);
R_Aleatory_separation=zeros(1,SampleNum_Aleatory);
R_Aleatory_reattachment=zeros(1,SampleNum_Aleatory);
R_Aleatory_bubblesize=zeros(1,SampleNum_Aleatory);
for j=1:SampleNum_Aleatory
    %% It is faster to store first and then plot!!!
    xi_Aleatory=[xi1(j),xi2(j),xi3(j)];
    R_Aleatory_separation(j)...
        =ResponseSurface(Alpha_separation,xi_Aleatory);
    R_Aleatory_reattachment(j)...
        =ResponseSurface(Alpha_reattachment,xi_Aleatory);
    R_Aleatory_bubblesize(j)...
```

```
        =ResponseSurface(Alpha_bubblesize,xi_Aleatory);
end
%% separation location
%% xi1, 'K' factor
subplot(3,3,1);
hold all;
plot(K,R_Aleatory_separation,'b.','MarkerSize',2);
xlabel('K factor','FontSize',12);
ylabel('Separation Location, x/c','FontSize',12);
title_plot=['p = ',num2str(p)];
title(title_plot,'FontSize',12);
axis([0.5,2.0,0.66,0.67]);
%% Uinf
subplot(3,3,2);
hold all;
plot(Uinf,R_Aleatory_separation,'b.','MarkerSize',2);
xlabel('Free Stream Velocity, m/s','FontSize',12);
ylabel('Separation Location, x/c','FontSize',12);
title_plot=['p = ',num2str(p)];
title(title_plot,'FontSize',12);
axis([30,40,0.66,0.67]);
%% frequency
subplot(3,3,3);
hold all;
plot(f,R_Aleatory_separation,'b.','MarkerSize',2);
xlabel('Frequency, Hz','FontSize',12);
ylabel('Separation Location, x/c','FontSize',12);
title_plot=['p = ',num2str(p)];
title(title_plot,'FontSize',12);
axis([120,160,0.66,0.67]);
%% reattachment location
%% xi1, 'K' factor
subplot(3,3,4);
hold all;
plot(K,R_Aleatory_reattachment,'b.','MarkerSize',2);
xlabel('K factor','FontSize',12);
ylabel('Reattachment Location, x/c','FontSize',12);
title_plot=['p = ',num2str(p)];
title(title_plot,'FontSize',12);
axis([0.5,2.0,1.0,1.4]);
%% Uinf
subplot(3,3,5);
hold all;
plot(Uinf,R_Aleatory_reattachment,'b.','MarkerSize',2);
xlabel('Free Stream Velocity, m/s','FontSize',12);
ylabel('Reattachment Location, x/c','FontSize',12);
title_plot=['p = ',num2str(p)];
title(title_plot,'FontSize',12);
axis([30,40,1.0,1.4]);
%% frequency
subplot(3,3,6);
hold all;
plot(f,R_Aleatory_reattachment,'b.','MarkerSize',2);
xlabel('Frequency, Hz','FontSize',12);
```

```
ylabel('Reattachment Location, x/c','FontSize',12);
title_plot=['p = ',num2str(p)];
title(title_plot,'FontSize',12);
axis([120,160,1.0,1.4]);
%% bubble size
%% xi1, 'K' factor
subplot(3,3,7);
hold all;
plot(K,R_Aleatory_bubblesize,'b.','MarkerSize',2);
xlabel('K factor','FontSize',12);
ylabel('Bubble Size, x/c','FontSize',12);
title_plot=['p = ',num2str(p)];
title(title_plot,'FontSize',12);
axis([0.5,2.0,0.4,0.8]);
%% Uinf
subplot(3,3,8);
hold all;
plot(Uinf,R_Aleatory_bubblesize,'b.','MarkerSize',2);
xlabel('Free Stream Velocity, m/s','FontSize',12);
ylabel('Bubble Size, x/c','FontSize',12);
title_plot=['p = ',num2str(p)];
title(title_plot,'FontSize',12);
axis([30,40,0.4,0.8]);
%% frequency
subplot(3,3,9);
hold all;
plot(f,R_Aleatory_bubblesize,'b.','MarkerSize',2);
xlabel('Frequency, Hz','FontSize',12);
ylabel('Bubble Size, x/c','FontSize',12);
title_plot=['p = ',num2str(p)];
title(title_plot,'FontSize',12);
axis([120,160,0.4,0.8]);

%% Pure aleatory expansion, return the mean value to check the PCE degree
%% convergence.
%% added 5/11/2011
%% read the samples
% SampleStr_Aleatory=['SamplesE2_3D.txt'];
SampleStr_Aleatory=['SamplesE',num2str(SampleNumOrder),'_3D.txt'];
SampleFile=[path,SampleStr_Aleatory];
[xi1,xi2,xi3]=textread(SampleFile,'%f %f %f','delimiter',',');
SampleNum_Aleatory=length(xi1);
%% response surface evaluation
R_Aleatory=zeros(1,SampleNum_Aleatory);
for j=1:SampleNum_Aleatory
    xi_Aleatory=[xi1(j),xi2(j),xi3(j)];
    R_Aleatory(j)=ResponseSurface(Alpha,xi_Aleatory);
end
%% median value
Median=median(R_Aleatory);
%% Use my own CDF function - modified 5/17/2011
R_Aleatory=sort(R_Aleatory);
CDF=MyCDF(R);
% CDF=UniCDF(R_Aleatory);
```

```
% [CDF,R_Aleatory]=Sortxy(CDF,R_Aleatory);
%% update the figure number
FigNum=FigNum+1;
%% histogram
figure(FigNum);
hold all;
hist(R_Aleatory,100);
xlabel('Long-time Averaged Bubble Size, x/c','FontSize',18);
titlestr=['Polynomial Chaos Degree p = ',num2str(p)];
title(titlestr,'FontSize',18);
FigNum=FigNum+1;
%% clear variables for speeding up
%% added 5/16/2011
% clearvars -except FigNum Median R_Aleatory CDF;
%% 3-D plot the samples
% figure(FigNum);
% hold all
% plot3(xi1,xi2,xi3,'b.','MarkerSize',2);
% xlabel('xi1','FontSize',18);
% ylabel('xi2','FontSize',18);
% zlabel('xi3','FontSize',18);
% %update the figure number
% FigNum=FigNum+1;

end
```

APPENDIX D

MATLAB SOURCE CODE: UNCERTAINTY QUANTIFICATION OF LONG-TIME AVERAGED PRESSURE AND SKIN FRICTION COEFFICIENTS

```
function FigNum=LongCpCf_UQ(FigNum)
%% Mixed UQ for long-time avearged Cp and Cf vs. x/c
%% created 4/3/2011
%% revised 4/5/2011: Added p=2 case.
%% refer to Ben's plot (Fig. 13, AIAA 2011-252)

%% revised 5/11/2011: Polynomial Chaos Expansion Degree Convergence check
%% return the mean value and CDF from pure aleatory expansion.

%% revised 5/17/2011: Check the median value rather than the mean value
%% when checking the CDF convergence at three selected locations.

%% revised 6/18/2011: subplots for the thesis.

%% revised 6/18/2011: add bar in the 95% CI plot.

%% revised 6/27/2011: Update the plots for thesis use.
%% E2 for epistemic samples and E3 for aleatory samples.

%% revised 6/29/2011: Put in the appendix of my thesis.

%% for test use only
tic;
close all;
clear all;
clc;
format short;
FigNum=1;
p=4;
p=1:4;
SampleNumOrder=3;% the order of sample number, i.e. '2' means 'E2' samples
%%
pNum=length(p);
Mean_Cp=zeros(pNum,71);
Mean_Cf=zeros(pNum,71);
%% preallocate the arrays
R_XC1_Cp=zeros(pNum,10^SampleNumOrder);% location 1
R_XC2_Cp=zeros(pNum,10^SampleNumOrder);% location 2
R_XC3_Cp=zeros(pNum,10^SampleNumOrder);% location 3
CDF_XC1_Cp=zeros(pNum,10^SampleNumOrder);% location 1
CDF_XC2_Cp=zeros(pNum,10^SampleNumOrder);% location 2
CDF_XC3_Cp=zeros(pNum,10^SampleNumOrder);% location 3
R_XC1_Cf=zeros(pNum,10^SampleNumOrder);% location 1
R_XC2_Cf=zeros(pNum,10^SampleNumOrder);% location 2
R_XC3_Cf=zeros(pNum,10^SampleNumOrder);% location 3
CDF_XC1_Cf=zeros(pNum,10^SampleNumOrder);% location 1
CDF_XC2_Cf=zeros(pNum,10^SampleNumOrder);% location 2
CDF_XC3_Cf=zeros(pNum,10^SampleNumOrder);% location 3
for i=1:pNum
    [FigNum,XC_plot_Cp,Mean_Cp(i,:),...
        R_XC1_Cp(i,:),CDF_XC1_Cp(i,:),...
        R_XC2_Cp(i,:),CDF_XC2_Cp(i,:),...
        R_XC3_Cp(i,:),CDF_XC3_Cp(i,:),]...
        =LongCp_UQ(p(i),FigNum,SampleNumOrder);
```

```
    [FigNum,XC_plot_Cf,Mean_Cf(i,:),...
        R_XC1_Cf(i,:),CDF_XC1_Cf(i,:),...
        R_XC2_Cf(i,:),CDF_XC2_Cf(i,:),...
        R_XC3_Cf(i,:),CDF_XC3_Cf(i,:),]...
        =LongCf_UQ(p(i),FigNum,SampleNumOrder);
    %% subplots for thesis use, added 6/19/2011
    FigNum=Plot_LongCpCfUQ(FigNum,p(i));
end
% size(Mean_Cp)
% size(Mean_Cf)
%% plot the PCE degree convergence check
%% LongCp
figure(FigNum);
hold all;
%% read the experimental data
[X_exp,Cp_exp]=ReadLongCp_exp;
plot(X_exp,Cp_exp,'r^',...
    'MarkerEdgeColor','r',...
    'MarkerFaceColor','r',...
    'MarkerSize',6);
plot(XC_plot_Cp,Mean_Cp(1,:),'b--o');
plot(XC_plot_Cp,Mean_Cp(2,:),'k--x');
plot(XC_plot_Cp,Mean_Cp(3,:),'m--*');
plot(XC_plot_Cp,Mean_Cp(4,:),'g-.+');
%% axis and ticks
axis([-1 2 -1.2 0.4]);
set(gca,'FontSize',12);
%% reverse the y-axis direction
set(gca,'YDir','reverse');
legend_exp=('Exp');
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend_exp,legend1,legend2,legend3,legend4,...
    'Location','NorthEast');
set(legend_plot,'FontSize',14);
xlabel('x/c','FontSize',18);
ylabel('Mean Long-time Averaged Cp','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);
FigNum=FigNum+1;% update the figure number
%% location 1
figure(FigNum);
hold all;
plot(R_XC1_Cp(1,:),CDF_XC1_Cp(1,:),'k-','LineWidth',3.0);
plot(R_XC1_Cp(2,:),CDF_XC1_Cp(2,:),'k--','LineWidth',2.5);
plot(R_XC1_Cp(3,:),CDF_XC1_Cp(3,:),'b-.','LineWidth',2.0);
plot(R_XC1_Cp(4,:),CDF_XC1_Cp(4,:),'r:','LineWidth',1.5);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend1,legend2,legend3,legend4,...
    'Location','SouthEast');
```

```
set(legend_plot,'FontSize',14);
xlabel('LongCp @ Location 1','FontSize',18);
ylabel('CDF','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);
FigNum=FigNum+1;% update the figure number
%% location 2
figure(FigNum);
hold all;
plot(R_XC2_Cp(1,:),CDF_XC2_Cp(1,:),'k-','LineWidth',3.0);
plot(R_XC2_Cp(2,:),CDF_XC2_Cp(2,:),'k--','LineWidth',2.5);
plot(R_XC2_Cp(3,:),CDF_XC2_Cp(3,:),'b-.','LineWidth',2.0);
plot(R_XC2_Cp(4,:),CDF_XC2_Cp(4,:),'r:','LineWidth',1.5);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend1,legend2,legend3,legend4,...
    'Location','SouthEast');
set(legend_plot,'FontSize',14);
xlabel('LongCp @ Location 2','FontSize',18);
ylabel('CDF','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);
FigNum=FigNum+1;% update the figure number
%% location 3
figure(FigNum);
hold all;
plot(R_XC3_Cp(1,:),CDF_XC3_Cp(1,:),'k-','LineWidth',3.0);
plot(R_XC3_Cp(2,:),CDF_XC3_Cp(2,:),'k--','LineWidth',2.5);
plot(R_XC3_Cp(3,:),CDF_XC3_Cp(3,:),'b-.','LineWidth',2.0);
plot(R_XC3_Cp(4,:),CDF_XC3_Cp(4,:),'r:','LineWidth',1.5);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend1,legend2,legend3,legend4,...
    'Location','SouthEast');
set(legend_plot,'FontSize',14);
xlabel('LongCp @ Location 3','FontSize',18);
ylabel('CDF','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);
FigNum=FigNum+1;% update the figure number
%% ========================================
%% LongCf
figure(FigNum);
hold all;
plot(XC_plot_Cf,Mean_Cf(1,:),'b--o');
plot(XC_plot_Cf,Mean_Cf(2,:),'k--x');
plot(XC_plot_Cf,Mean_Cf(3,:),'m--*');
plot(XC_plot_Cf,Mean_Cf(4,:),'r-.+');
%% axis and ticks
axis([0.6 1.4 -0.003 0.002]);
set(gca,'FontSize',12);
legend1=('p = 1');
legend2=('p = 2');
```

```
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend1,legend2,legend3,legend4,...
    'Location','SouthEast');
set(legend_plot,'FontSize',14);
xlabel('x/c','FontSize',18);
ylabel('Mean Long-time Averaged Cf','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);
FigNum=FigNum+1;% update the figure number
%% location 1
figure(FigNum);
hold all;
plot(R_XC1_Cf(1,:),CDF_XC1_Cf(1,:),'k-','LineWidth',3.0);
plot(R_XC1_Cf(2,:),CDF_XC1_Cf(2,:),'k--','LineWidth',2.5);
plot(R_XC1_Cf(3,:),CDF_XC1_Cf(3,:),'b-.','LineWidth',2.0);
plot(R_XC1_Cf(4,:),CDF_XC1_Cf(4,:),'r:','LineWidth',1.5);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend1,legend2,legend3,legend4,...
    'Location','SouthEast');
set(legend_plot,'FontSize',14);
xlabel('LongCf @ Location 1','FontSize',18);
ylabel('CDF','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);
FigNum=FigNum+1;% update the figure number
%% location 2
figure(FigNum);
hold all;
plot(R_XC2_Cf(1,:),CDF_XC2_Cf(1,:),'k-','LineWidth',3.0);
plot(R_XC2_Cf(2,:),CDF_XC2_Cf(2,:),'k--','LineWidth',2.5);
plot(R_XC2_Cf(3,:),CDF_XC2_Cf(3,:),'b-.','LineWidth',2.0);
plot(R_XC2_Cf(4,:),CDF_XC2_Cf(4,:),'r:','LineWidth',1.5);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend1,legend2,legend3,legend4,...
    'Location','SouthEast');
set(legend_plot,'FontSize',14);
xlabel('LongCf @ Location 2','FontSize',18);
ylabel('CDF','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);
FigNum=FigNum+1;% update the figure number
%% location 3
figure(FigNum);
hold all;
plot(R_XC3_Cf(1,:),CDF_XC3_Cf(1,:),'k-','LineWidth',3.0);
plot(R_XC3_Cf(2,:),CDF_XC3_Cf(2,:),'k--','LineWidth',2.5);
plot(R_XC3_Cf(3,:),CDF_XC3_Cf(3,:),'b-.','LineWidth',2.0);
plot(R_XC3_Cf(4,:),CDF_XC3_Cf(4,:),'r:','LineWidth',1.5);
legend1=('p = 1');
legend2=('p = 2');
```

```
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend1,legend2,legend3,legend4,...
    'Location','SouthEast');
set(legend_plot,'FontSize',14);
xlabel('LongCf @ Location 3','FontSize',18);
ylabel('CDF','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);
FigNum=FigNum+1;% update the figure number
%% ========================================
%% subplots for the thesis use
%% Location 1
Plot_LongCpCf1=figure(FigNum);
% [left bottom width height]
% by default, [440 378 560 420]
% This size fits the pdf file of A4 paper.
% set(PlotU,'Position',[50 -400 900 1150]);
set(Plot_LongCpCf1,'Position',[200 378 900 420]);
hold all;
% LongCp
subplot(1,2,1);
hold all;
plot(R_XC1_Cp(1,:),CDF_XC1_Cp(1,:),'k-','LineWidth',3.0);
plot(R_XC1_Cp(2,:),CDF_XC1_Cp(2,:),'k--','LineWidth',2.5);
plot(R_XC1_Cp(3,:),CDF_XC1_Cp(3,:),'b-.','LineWidth',2.0);
plot(R_XC1_Cp(4,:),CDF_XC1_Cp(4,:),'r:','LineWidth',1.5);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend1,legend2,legend3,legend4,...
    'Location','SouthEast');
set(legend_plot,'FontSize',14);
xlabel('LongCp @ Location 1','FontSize',18);
ylabel('CDF','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);
% LongCf
subplot(1,2,2);
hold all;
plot(R_XC1_Cf(1,:),CDF_XC1_Cf(1,:),'k-','LineWidth',3.0);
plot(R_XC1_Cf(2,:),CDF_XC1_Cf(2,:),'k--','LineWidth',2.5);
plot(R_XC1_Cf(3,:),CDF_XC1_Cf(3,:),'b-.','LineWidth',2.0);
plot(R_XC1_Cf(4,:),CDF_XC1_Cf(4,:),'r:','LineWidth',1.5);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend1,legend2,legend3,legend4,...
    'Location','SouthEast');
set(legend_plot,'FontSize',14);
xlabel('LongCf @ Location 1','FontSize',18);
ylabel('CDF','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);
```

```
FigNum=FigNum+1;% update the figure number
%% Location 2
Plot_LongCpCf2=figure(FigNum);
% [left bottom width height]
% by default, [440 378 560 420]
% This size fits the pdf file of A4 paper.
% set(PlotU,'Position',[50 -400 900 1150]);
set(Plot_LongCpCf2,'Position',[200 378 900 420]);
hold all;
% LongCp
subplot(1,2,1);
hold all;
plot(R_XC2_Cp(1,:),CDF_XC2_Cp(1,:),'k-','LineWidth',3.0);
plot(R_XC2_Cp(2,:),CDF_XC2_Cp(2,:),'k--','LineWidth',2.5);
plot(R_XC2_Cp(3,:),CDF_XC2_Cp(3,:),'b-.','LineWidth',2.0);
plot(R_XC2_Cp(4,:),CDF_XC2_Cp(4,:),'r:','LineWidth',1.5);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend1,legend2,legend3,legend4,...
    'Location','SouthEast');
set(legend_plot,'FontSize',14);
xlabel('LongCp @ Location 2','FontSize',18);
ylabel('CDF','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);
% LongCf
subplot(1,2,2);
hold all;
plot(R_XC2_Cf(1,:),CDF_XC2_Cf(1,:),'k-','LineWidth',3.0);
plot(R_XC2_Cf(2,:),CDF_XC2_Cf(2,:),'k--','LineWidth',2.5);
plot(R_XC2_Cf(3,:),CDF_XC2_Cf(3,:),'b-.','LineWidth',2.0);
plot(R_XC2_Cf(4,:),CDF_XC2_Cf(4,:),'r:','LineWidth',1.5);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend1,legend2,legend3,legend4,...
    'Location','SouthEast');
set(legend_plot,'FontSize',14);
xlabel('LongCf @ Location 2','FontSize',18);
ylabel('CDF','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);

FigNum=FigNum+1;% update the figure number

%% Location 3
Plot_LongCpCf3=figure(FigNum);
% [left bottom width height]
% by default, [440 378 560 420]
% This size fits the pdf file of A4 paper.
% set(PlotU,'Position',[50 -400 900 1150]);
set(Plot_LongCpCf3,'Position',[200 378 900 420]);
hold all;
```

```
% LongCp
subplot(1,2,1);
hold all;
plot(R_XC3_Cp(1,:),CDF_XC3_Cp(1,:),'k-','LineWidth',3.0);
plot(R_XC3_Cp(2,:),CDF_XC3_Cp(2,:),'k--','LineWidth',2.5);
plot(R_XC3_Cp(3,:),CDF_XC3_Cp(3,:),'b-.','LineWidth',2.0);
plot(R_XC3_Cp(4,:),CDF_XC3_Cp(4,:),'r:','LineWidth',1.5);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend1,legend2,legend3,legend4,...
    'Location','SouthEast');
set(legend_plot,'FontSize',14);
xlabel('LongCp @ Location 3','FontSize',18);
ylabel('CDF','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);
% LongCf
subplot(1,2,2);
hold all;
plot(R_XC3_Cf(1,:),CDF_XC3_Cf(1,:),'k-','LineWidth',3.0);
plot(R_XC3_Cf(2,:),CDF_XC3_Cf(2,:),'k--','LineWidth',2.5);
plot(R_XC3_Cf(3,:),CDF_XC3_Cf(3,:),'b-.','LineWidth',2.0);
plot(R_XC3_Cf(4,:),CDF_XC3_Cf(4,:),'r:','LineWidth',1.5);
legend1=('p = 1');
legend2=('p = 2');
legend3=('p = 3');
legend4=('p = 4');
legend_plot=legend(legend1,legend2,legend3,legend4,...
    'Location','SouthEast');
set(legend_plot,'FontSize',14);
xlabel('LongCf @ Location 3','FontSize',18);
ylabel('CDF','FontSize',18);
title('Pure Aleatory Expansion','FontSize',18);

FigNum=FigNum+1;% update the figure number
%% ========================================
%% clear variables for speeding up
%% added 5/17/2011
% clearvars -except FigNum;
toc;
end

function [FigNum,XC_plot,Mean,R_XC1,CDF_XC1,R_XC2,CDF_XC2,R_XC3,CDF_XC3]...
    =LongCp_UQ(p,FigNum,SampleNumOrder)
%% Mixed UQ for long-time avearged Cp vs. x/c
%% created 3/30/2011

%% revised 4/3/2011: Pure Aleatory UQ
%% refer to Ben's plot (Fig. 13, AIAA 2011-252)

%% revised 4/5/2011: Added p=2 case.

%% revised 5/11/2011: Return the mean value in the pure aleatory expansion
```

```
%% for the PCE degree convergence check. Also return the XC_plot and CDF
%% in pure aleatory expansion at three selected x/c locations.

%% revised 5/12/2011: add the Sobol indices part

%% revised 6/7/2011: add the 95% CI plot.

%% revised 6/18/2011: subplots for the thesis.

%% revised 6/18/2011: add bar in the 95% CI plot.

%% revised 6/19/2011: write the UQ data to file to make the subplots for
%% thesis use.

%% revised 6/29/2011: Put in the appendix of my thesis.

%% for test use only
% close all
% clear all
% clc
% FigNum=1;
% % p is the order of polynomial chaos expansion
% p=3;
% SampleNumOrder=2;
%% creat the x/c matrix and Cp matrix (Quadrature Points data)
%RunNum is the Run number of the quadrature points
RunNums=(p+1)^3;
RunNum=[1:RunNums];
RunNums=length(RunNum);
%% read the written file
Cp=zeros(723,RunNums);
for i=1:RunNums
    [XC,Cp(:,i)]=ReadLongCp(p,RunNum(i));
end
% XC;
% CP;
%Totally 723 points in x/c direction
PointsNumber=length(XC);
% size(CP)
ProbabilityLevel=[0.025,0.5,0.975];
ProbabilityLevelNum=length(ProbabilityLevel);
path=['C:\Users\dhx88\Daoru\Case3\C3\POST\ThesisWork\'];
%% Mixed UQ and Pure Aleatory UQ
PointIndex=1;
for index=1:10:701%x/c index
    AlphaStar=Cp(index,:);
    Alpha=PolyCoef(p,AlphaStar);
    %% Mixed UQ: Second-Order Probability sampling
    %% with Constructed Response Surface
    %% Epistemic loop
    % xi1=[-1.00:0.05:1.00];
    %% sample epistemic loop
    SampleStr_1D='SamplesE2_1D.txt';
    SampleFile_1D=[path,SampleStr_1D];
```

```
xi1=textread(SampleFile_1D,'%f');
%%
EpistemicNum=length(xi1);
%% preallocate Cp_Point
Cp_Point=zeros(ProbabilityLevelNum,EpistemicNum);
for i=1:EpistemicNum
    %% Aleatory loop
    %% read the samples
    % SampleStr=['SamplesE2.txt'];
    SampleStr=['SamplesE',num2str(SampleNumOrder),'.txt'];
    SampleFile=[path,SampleStr];
    [xi2,xi3]=textread(SampleFile,'%f %f','delimiter',',');
    SampleNum=length(xi2);
    %% response surface evaluation
    R=zeros(1,SampleNum);
    for j=1:SampleNum
        xi=[xi1(i),xi2(j),xi3(j)];
        R(j)=ResponseSurface(Alpha,xi);
    end
    %% Sort and calculate the CDF
    %% Mixed UQ
    %% use my own CDF function, modifed 5/17/2011
    R=sort(R);% sort
    R_cdf=MyCDF(R);
    % R_cdf=UniCDF(R);
    % [R_cdf,R]=Sortxy(R_cdf,R);
    %% Find the value for the corresponding probability level
    for k=1:ProbabilityLevelNum%each probability level
        for kk=1:SampleNum%find the value
            %% For Mixed UQ
            if R_cdf(kk) > ProbabilityLevel(k)
                %% linear interpolation
                RR=R(kk)-...
                    (R_cdf(kk)-ProbabilityLevel(k))...
                    /(R_cdf(kk)-R_cdf(kk-1))...
                   *(R(kk)-R(kk-1));
                Cp_Point(k,i)=RR;
                break;
            end
        end
        Cp_Mixed_min(k,PointIndex)=min(Cp_Point(k,:));
        Cp_Mixed_max(k,PointIndex)=max(Cp_Point(k,:));
    end%Probability Level loop
end%Epistemic loop for Mixed UQ
%% Aleatory UQ
%% read the samples
% SampleStr_Aleatory=['SamplesE2_3D.txt'];
SampleStr_Aleatory=['SamplesE',num2str(SampleNumOrder),'_3D.txt'];
SampleFile=[path,SampleStr_Aleatory];
[xi1,xi2,xi3]=textread(SampleFile,'%f %f %f','delimiter',',');
SampleNum_Aleatory=length(xi1);
%% response surface evaluation
R_Aleatory=zeros(1,SampleNum_Aleatory);
for j=1:SampleNum_Aleatory
```

```
    xi_Aleatory=[xi1(j),xi2(j),xi3(j)];
    R_Aleatory(j)=ResponseSurface(Alpha,xi_Aleatory);
end
%% return the mean value: added 5/11/2011
Mean(PointIndex)=mean(R_Aleatory);
%% Sort and calculate the CDF
%% Pure Aleatory UQ
%% use my own CDF function
R_Aleatory=sort(R_Aleatory);
R_cdf_Aleatory=MyCDF(R_Aleatory);
% R_cdf_Aleatory=UniCDF(R_Aleatory);
% [R_cdf_Aleatory,R_Aleatory]=Sortxy(R_cdf_Aleatory,R_Aleatory);
%% Select three x/c locations for PCE degree convergence check using
%% CDFs.
if index==201% location 1, x/c = 0.62693
    % XC(index)
    R_XC1=R_Aleatory;
    CDF_XC1=R_cdf_Aleatory;
    %% Sobol Indices part
    %% added 5/12/2011
    str=['Sobol Indices for LongCp at x/c = ',num2str(XC(index)),...
        ' (upstream separation bubble)'];
    disp(str);
    [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
end
if index==601% location 2, x/c = 0.994
    % XC(index)
    R_XC2=R_Aleatory;
    CDF_XC2=R_cdf_Aleatory;
    %% Sobol Indices part
    %% added 5/12/2011
    str=['Sobol Indices for LongCp at x/c = ',num2str(XC(index)),...
        ' (inside separation bubble)'];
    disp(str);
    [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
end
if index==671% location 3, x/c = 1.5212
    % XC(index)
    R_XC3=R_Aleatory;
    CDF_XC3=R_cdf_Aleatory;
    %% Sobol Indices part
    %% added 5/12/2011
    str=['Sobol Indices for LongCp at x/c = ',num2str(XC(index)),...
        ' (downstream separation bubble)'];
    disp(str);
    [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
end
%% Find the value for the corresponding probability level
Cp_Point_Aleatory=zeros(1,ProbabilityLevelNum);
for k=1:ProbabilityLevelNum%each probability level
    for kk=1:SampleNum_Aleatory%find the value
        %% For Aleatory UQ
        if R_cdf_Aleatory(kk) > ProbabilityLevel(k)
            %% linear interpolation
```

```
                RR_Aleatory=...
                    R_Aleatory(kk)...
                    -(R_cdf_Aleatory(kk)-ProbabilityLevel(k))...
                    /(R_cdf_Aleatory(kk)-R_cdf_Aleatory(kk-1))...
                        *(R_Aleatory(kk)-R_Aleatory(kk-1));
                Cp_Point_Aleatory(k)=RR_Aleatory;
                break;
            end
        end
        Cp_Aleatory(k,PointIndex)=Cp_Point_Aleatory(k);
    end%Probability Level loop
    %Cp_Point
    XC_plot(PointIndex)=XC(index);%for plot use
    PointIndex=PointIndex+1;
end%x/c loop
%% Check the results, for test use
% XC_plot
% Cp_Mixed_min
% Cp_Mixed_max
% Cp_Aleatory
%% ========================================================
%% write the data to files, added 6/19/2011
for i=1:ProbabilityLevelNum% each probability level
    % aleatory
    filename=['LongCpUQ_p',num2str(p),'_',...
        num2str(ProbabilityLevel(i)*1000),'_Al.txt'];
    WriteLongCpUQ(filename,XC_plot,Cp_Aleatory(i,:));
    % mixed
    filename=['LongCpUQ_p',num2str(p),'_',...
        num2str(ProbabilityLevel(i)*1000),'_L.txt'];
    WriteLongCpUQ(filename,XC_plot,Cp_Mixed_min(i,:));
    filename=['LongCpUQ_p',num2str(p),'_',...
        num2str(ProbabilityLevel(i)*1000),'_U.txt'];
    WriteLongCpUQ(filename,XC_plot,Cp_Mixed_max(i,:));
end
%% ========================================================
%% Plot the results including experimental data
%% read the experimental data
[X_exp,Cp_exp]=ReadLongCp_exp;
for i=1:ProbabilityLevelNum
    figure(FigNum+i-1);
    hold all;
    %% experimental data
    plot(X_exp,Cp_exp,'r^',...
        'MarkerEdgeColor','r',...
        'MarkerFaceColor','r',...
        'MarkerSize',6);
    plot(XC_plot,Cp_Mixed_min(i,:),'b--','LineWidth',2.5);
    plot(XC_plot,Cp_Mixed_max(i,:),'b-.','LineWidth',2.5);
    plot(XC_plot,Cp_Aleatory(i,:),'kd-','LineWidth',2.5,...
        'MarkerEdgeColor','k',...
        'MarkerFaceColor','k',...
        'MarkerSize',4);
    %% title
```

```
    title_plot=['Long-time Averaged Cp, Probability Level = ',...
                num2str(ProbabilityLevel(i)*100),'% for p = ',num2str(p)];
    title(title_plot,'FontSize',14)
    %% axis and ticks
    axis([-1 2.5 -1.2 0.4]);
    set(gca,'FontSize',12);
    %% reverse the y-axis direction
    set(gca,'YDir','reverse');
    %% labels and legend
    xlabel('x/c','FontSize',18)
    ylabel('Cp','FontSize',18)
    legend1=['Exp'];
    legend2=['Mixed UQ, lower bound'];
    legend3=['Mixed UQ, upper bound'];
    legend4=['Pure Aleatory UQ'];
    legend_plot...
        =legend(legend1,legend2,legend3,legend4,'Location','NorthEast');
    set(legend_plot,'FontSize',8);
end
FigNum=FigNum+i;
%% plot the 95% C.I.
%% added 6/7/2011
Cp_CI_min=Cp_Mixed_min(1,:);% the first row
Cp_CI_max=Cp_Mixed_max(ProbabilityLevelNum,:);% the last row
%% ========================================================
%% write the data to files, added 6/19/2011
% lower bounds
filename=['LongCpUQ_CI_p',num2str(p),'_L.txt'];
WriteLongCpUQ(filename,XC_plot,Cp_CI_min);
% upper bounds
filename=['LongCpUQ_CI_p',num2str(p),'_U.txt'];
WriteLongCpUQ(filename,XC_plot,Cp_CI_max);
%% ========================================================
figure(FigNum);
hold all;
plot(X_exp,Cp_exp,'r^',...
    'MarkerEdgeColor','r',...
    'MarkerFaceColor','r',...
    'MarkerSize',6);
plot(XC_plot,Cp_CI_min,'b--x','LineWidth',2.5,'MarkerSize',10);
plot(XC_plot,Cp_CI_max,'k-x','LineWidth',2.5,'MarkerSize',10);
%% 95% CI bar plot, added 6/18/2011
PointNum=size(XC_plot,2);% number of points
for BarIndex=1:PointNum% each bar
    XX=[XC_plot(BarIndex),XC_plot(BarIndex)];
    YY=[Cp_CI_min(BarIndex),Cp_CI_max(BarIndex)];
    line(XX,YY,'LineStyle','--','Color',[0.5 0.5 0.5]);% gray
    % line(XX,YY,'LineStyle','--','Color',[0 1 1]);% cyan
    % line(XX,YY,'LineStyle','--','Color',[0 1 0]);% green
    % line(XX,YY,'LineStyle','--','Color',[1 0 1]);% magenta
end

%% title
title_plot=['Long-time Averaged Cp, 95% CI for p = ',num2str(p)];
```

```
title(title_plot,'FontSize',14);
%% axis and ticks
axis([-1 2.5 -1.2 0.4]);
set(gca,'FontSize',12);
%% reverse the y-axis direction
set(gca,'YDir','reverse');
%% labels and legend
xlabel('x/c','FontSize',18);
ylabel('Cp','FontSize',18);
legend1=['Exp'];
legend2=['Lower bound'];
legend3=['Upper bound'];
legend_plot=legend(legend1,legend2,legend3,'Location','NorthEast');
set(legend_plot,'FontSize',8);
FigNum=FigNum+1;
%% ========================================================
%% subplots of the UQ for thesis use
%% added 6/18/2011
Plot_LongCp_UQ=figure(FigNum);
% [left bottom width height]
% by default, [440 378 560 420]
% This size fits the pdf file of A4 paper.
% set(PlotU,'Position',[50 -400 900 1150]);
set(Plot_LongCp_UQ,'Position',[200 -500 450 1150]);
for i=1:ProbabilityLevelNum
    subplot(3,1,i);
    hold all;
    %% experimental data
    plot(X_exp,Cp_exp,'r^',...
        'MarkerEdgeColor','r',...
        'MarkerFaceColor','r',...
        'MarkerSize',6);
    plot(XC_plot,Cp_Mixed_min(i,:),'b--','LineWidth',2.5);
    plot(XC_plot,Cp_Mixed_max(i,:),'b-.','LineWidth',2.5);
    plot(XC_plot,Cp_Aleatory(i,:),'kd-','LineWidth',2.5,...
        'MarkerEdgeColor','k',...
        'MarkerFaceColor','k',...
        'MarkerSize',4);
    %% title
    title_plot=['Probability Level = ',...
                num2str(ProbabilityLevel(i)*100),'%, p = ',num2str(p)];
    title(title_plot,'FontSize',14)
    %% axis and ticks
    axis([-1 2.5 -1.2 0.4]);
    set(gca,'FontSize',12);
    %% reverse the y-axis direction
    set(gca,'YDir','reverse');
    %% labels and legend
    xlabel('x/c','FontSize',18)
    ylabel('Cp','FontSize',18)
    legend1=['Exp'];
    legend2=['Mixed UQ, lower bounds'];
    legend3=['Mixed UQ, upper bounds'];
    legend4=['Pure Aleatory UQ'];
```

```
    legend_plot...
        =legend(legend1,legend2,legend3,legend4,'Location','NorthEast');
    set(legend_plot,'FontSize',6);
end
FigNum=FigNum+1;
%%
%% clear variables for speeding up, added 5/17/2011
% clearvars -except...
%     FigNum XC_plot Mean R_XC1 CDF_XC1 R_XC2 CDF_XC2 R_XC3 CDF_XC3;
end

function [FigNum,XC_plot,Mean,R_XC1,CDF_XC1,R_XC2,CDF_XC2,R_XC3,CDF_XC3]...
    =LongCf_UQ(p,FigNum,SampleNumOrder)
%% Mixed UQ for long-time avearged Cf vs. x/c
%% created 3/30/2011: modified from 'LongCp_UQ'

%% revised 4/3/2011: Pure Aleatory UQ
%% refer to Ben's plot (Fig. 13, AIAA 2011-252)

%% revised 4/5/2011: Added p=2 case.

%% revised 5/11/2011: Return the mean value in the pure aleatory expansion
%% for the PCE degree convergence check. Also return the XC_plot and CDF
%% in pure aleatory expansion at three selected x/c locations.

%% revised 5/12/2011: add the Sobol indices part

%% revised 6/7/2011: add the 95% CI plot.

%% revised 6/18/2011: subplots for the thesis.

%% revised 6/18/2011: add bar in the 95% CI plot.

%% revised 6/19/2011: write the UQ data to file to make the subplots for
%% thesis use.

%% revised 6/29/2011: Put in the appendix of my thesis.

%% for test use only
% close all
% clear all
% clc
% FigNum=1;
% % p is the order of polynomial chaos expansion
% p=3;
% SampleNumOrder=2;

%% creat the x/c matrix and Cf matrix (Quadrature Points data)
%RunNum is the Run number of the quadrature points
RunNums=(p+1)^3;
RunNum=[1:RunNums];
RunNums=length(RunNum);
%% read the written file
Cf=zeros(723,RunNums);
```

```
for i=1:RunNums
    [XC,Cf(:,i)]=ReadLongCf(p,RunNum(i));
end
% XC;
% Cf;
%Totally 723 points in x/c direction
PointsNumber=length(XC);
% size(CP)
ProbabilityLevel=[0.025,0.5,0.975];
ProbabilityLevelNum=length(ProbabilityLevel);
path=['C:\Users\dhx88\Daoru\Case3\C3\POST\ThesisWork\'];
%% Mixed UQ and Pure Aleatory UQ
PointIndex=1;
for index=1:10:701%x/c index
    AlphaStar=Cf(index,:);
    Alpha=PolyCoef(p,AlphaStar);
    %% Mixed UQ: Second-Order Probability sampling
    %% with Constructed Response Surface
    %% Epistemic loop
    % xi1=[-1.00:0.05:1.00];
    %% sample epistemic loop
    SampleStr_1D='SamplesE2_1D.txt';
    SampleFile_1D=[path,SampleStr_1D];
    xi1=textread(SampleFile_1D,'%f');
    %%
    EpistemicNum=length(xi1);
    %% preallocate Cf_Point
    Cf_Point=zeros(ProbabilityLevelNum,EpistemicNum);
    for i=1:EpistemicNum
        %% Aleatory loop
        %% read the samples
        % SampleStr=['SamplesE2.txt'];
        SampleStr=['SamplesE',num2str(SampleNumOrder),'.txt'];
        SampleFile=[path,SampleStr];
        [xi2,xi3]=textread(SampleFile,'%f %f','delimiter',',');
        SampleNum=length(xi2);
        %% response surface evaluation
        R=zeros(1,SampleNum);
        for j=1:SampleNum
            xi=[xi1(i),xi2(j),xi3(j)];
            R(j)=ResponseSurface(Alpha,xi);
        end
        %% Sort and calculate the CDF
        %% Mixed UQ
        %% use my own CDF function, modified 5/17/2011
        R=sort(R);
        R_cdf=MyCDF(R);
        % R_cdf=UniCDF(R);
        % [R_cdf,R]=Sortxy(R_cdf,R);
        %% Find the value for the corresponding probability level
        for k=1:ProbabilityLevelNum%each probability level
            for kk=1:SampleNum%find the value
                %% For Mixed UQ
                if R_cdf(kk) > ProbabilityLevel(k)
```

```
                    %% linear interpolation
                    RR=R(kk)-...
                        (R_cdf(kk)-ProbabilityLevel(k))...
                        /(R_cdf(kk)-R_cdf(kk-1))...
                       *(R(kk)-R(kk-1));
                    Cf_Point(k,i)=RR;
                    break;
                end
            end
            Cf_Mixed_min(k,PointIndex)=min(Cf_Point(k,:));
            Cf_Mixed_max(k,PointIndex)=max(Cf_Point(k,:));
        end%Probability Level loop
    end%Epistemic loop for Mixed UQ
    %% Aleatory UQ
    %% read the samples
    % SampleStr_Aleatory=['SamplesE2_3D.txt'];
    SampleStr_Aleatory=['SamplesE',num2str(SampleNumOrder),'_3D.txt'];
    SampleFile=[path,SampleStr_Aleatory];
    [xi1,xi2,xi3]=textread(SampleFile,'%f %f %f','delimiter',',');
    SampleNum_Aleatory=length(xi2);
    %% response surface evaluation
    R_Aleatory=zeros(1,SampleNum_Aleatory);
    for j=1:SampleNum_Aleatory
        xi_Aleatory=[xi1(j),xi2(j),xi3(j)];
        R_Aleatory(j)=ResponseSurface(Alpha,xi_Aleatory);
    end
    %% return the mean value: added 5/11/2011
    Mean(PointIndex)=mean(R_Aleatory);
    %% Sort and calculate the CDF
    %% Pure Aleatory UQ
    R_Aleatory=sort(R_Aleatory);
    R_cdf_Aleatory=MyCDF(R_Aleatory);
    % R_cdf_Aleatory=UniCDF(R_Aleatory);
    % [R_cdf_Aleatory,R_Aleatory]=Sortxy(R_cdf_Aleatory,R_Aleatory);
    %% Select three x/c locations for PCE degree convergence check using
    %% CDFs.
    if index==201% location 1, x/c = 0.62693
        % XC(index)
        R_XC1=R_Aleatory;
        CDF_XC1=R_cdf_Aleatory;
        %% Sobol Indices part
        %% added 5/12/2011
        str=['Sobol Indices for LongCf at x/c = ',num2str(XC(index))...
            ,' (upstream separation bubble)'];
        disp(str);
        [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
    end
    if index==601% location 2, x/c = 0.994
        % XC(index)
        R_XC2=R_Aleatory;
        CDF_XC2=R_cdf_Aleatory;
        %% Sobol Indices part
        %% added 5/12/2011
        str=['Sobol Indices for LongCf at x/c = ',num2str(XC(index))...
```

```
            ,' (inside separation bubble)'];
        disp(str);
        [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
    end
    if index==671% location 3, x/c = 1.5212
        % XC(index)
        R_XC3=R_Aleatory;
        CDF_XC3=R_cdf_Aleatory;
        %% Sobol Indices part
        %% added 5/12/2011
        str=['Sobol Indices for LongCf at x/c = ',num2str(XC(index))...
            ,' (downstream separation bubble)'];
        disp(str);
        [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
    end
    %% Find the value for the corresponding probability level
    Cf_Point_Aleatory=zeros(1,ProbabilityLevelNum);
    for k=1:ProbabilityLevelNum%each probability level
        for kk=1:SampleNum_Aleatory%find the value
            %% For Aleatory UQ
            if R_cdf_Aleatory(kk) > ProbabilityLevel(k)
                %% linear interpolation
                RR_Aleatory=R_Aleatory(kk)...
                    -(R_cdf_Aleatory(kk)-ProbabilityLevel(k))...
                    /(R_cdf_Aleatory(kk)-R_cdf_Aleatory(kk-1))...
                            *(R_Aleatory(kk)-R_Aleatory(kk-1));
                Cf_Point_Aleatory(k)=RR_Aleatory;
                break;
            end
        end
        Cf_Aleatory(k,PointIndex)=Cf_Point_Aleatory(k);
    end%Probability Level loop
    %Cf_Point
    XC_plot(PointIndex)=XC(index);%for plot use
    PointIndex=PointIndex+1;
end%x/c loop
%% Check the results, for test use
% XC_plot
% Cf_Mixed_min
% Cf_Mixed_max
% Cf_Aleatory
%% ========================================================
%% write the data to files, added 6/19/2011
for i=1:ProbabilityLevelNum% each probability level
    % aleatory
    filename=['LongCfUQ_p',num2str(p),'_',...
        num2str(ProbabilityLevel(i)*1000),'_Al.txt'];
    WriteLongCpUQ(filename,XC_plot,Cf_Aleatory(i,:));
    % mixed
    filename=['LongCfUQ_p',num2str(p),'_',...
        num2str(ProbabilityLevel(i)*1000),'_L.txt'];
    WriteLongCpUQ(filename,XC_plot,Cf_Mixed_min(i,:));
    filename=['LongCfUQ_p',num2str(p),'_',...
        num2str(ProbabilityLevel(i)*1000),'_U.txt'];
```

```
    WriteLongCpUQ(filename,XC_plot,Cf_Mixed_max(i,:));
end
%% ========================================================
%% Plot the results
for i=1:ProbabilityLevelNum
    figure(FigNum+i-1);
    hold all;
    grid on;
    plot(XC_plot,Cf_Mixed_min(i,:),'b--','LineWidth',2.5);
    plot(XC_plot,Cf_Mixed_max(i,:),'b-.','LineWidth',2.5);
    plot(XC_plot,Cf_Aleatory(i,:),'kd-','LineWidth',2.5,...
        'MarkerEdgeColor','k',...
        'MarkerFaceColor','k',...
        'MarkerSize',4);
    %% title
    title_plot=['Long-time Averaged Cf, Probability Level = ',...
                num2str(ProbabilityLevel(i)*100),'% for p = ',num2str(p)];
    title(title_plot,'FontSize',10)
    %% axis and ticks
    axis([0.6 1.4 -0.003 0.002]);
    set(gca,'FontSize',12);
    %% labels and legend
    xlabel('x/c','FontSize',18)
    ylabel('Cf','FontSize',18)
    legend1=['Mixed UQ, lower bound'];
    legend2=['Mixed UQ, upper bound'];
    legend3=['Pure Aleatory UQ'];
    legend_plot=legend(legend1,legend2,legend3,'Location','North');
    set(legend_plot,'FontSize',10);
end
FigNum=FigNum+i;
%% plot the 95% C.I.
%% added 6/7/2011
%% same idea as LongCp
Cf_CI_min=Cf_Mixed_min(1,:);% the first row
Cf_CI_max=Cf_Mixed_max(ProbabilityLevelNum,:);% the last row
%% ========================================================
%% write the data to files, added 6/19/2011
% lower bounds
filename=['LongCfUQ_CI_p',num2str(p),'_L.txt'];
WriteLongCpUQ(filename,XC_plot,Cf_CI_min);
% upper bounds
filename=['LongCfUQ_CI_p',num2str(p),'_U.txt'];
WriteLongCpUQ(filename,XC_plot,Cf_CI_max);
%% ========================================================
figure(FigNum);
hold all;
grid on;
plot(XC_plot,Cf_CI_min,'b--x','LineWidth',2.5,'MarkerSize',10);
plot(XC_plot,Cf_CI_max,'k-x','LineWidth',2.5,'MarkerSize',10);
%% 95% CI bar plot, added 6/18/2011
PointNum=size(XC_plot,2);% number of points
for BarIndex=1:PointNum% each bar
    XX=[XC_plot(BarIndex),XC_plot(BarIndex)];
```

```
    YY=[Cf_CI_min(BarIndex),Cf_CI_max(BarIndex)];
    line(XX,YY,'LineStyle','--','Color',[0.5 0.5 0.5]);% gray
    % line(XX,YY,'LineStyle','--','Color',[0 1 1]);% cyan
    % line(XX,YY,'LineStyle','--','Color',[0 1 0]);% green
    % line(XX,YY,'LineStyle','--','Color',[1 0 1]);% magenta
end
%% title
title_plot=['Long-time Averaged Cf, 95% CI for p = ',num2str(p)];
title(title_plot,'FontSize',14);
%% axis and ticks
axis([0.6 1.4 -0.003 0.002]);
set(gca,'FontSize',12);
%% labels and legend
xlabel('x/c','FontSize',18);
ylabel('Cf','FontSize',18);
legend1=['Lower bound'];
legend2=['Upper bound'];
legend_plot=legend(legend1,legend2,'Location','North');
set(legend_plot,'FontSize',12);
FigNum=FigNum+1;
%% ========================================================
%% subplots of the UQ for thesis use
%% added 6/18/2011
Plot_LongCf_UQ=figure(FigNum);
% [left bottom width height]
% by default, [440 378 560 420]
% This size fits the pdf file of A4 paper.
% set(PlotU,'Position',[50 -400 900 1150]);
set(Plot_LongCf_UQ,'Position',[200 -500 450 1150]);
for i=1:ProbabilityLevelNum
    subplot(3,1,i);
    hold all;
    grid on;
    plot(XC_plot,Cf_Mixed_min(i,:),'b--','LineWidth',2.5);
    plot(XC_plot,Cf_Mixed_max(i,:),'b-.','LineWidth',2.5);
    plot(XC_plot,Cf_Aleatory(i,:),'kd-','LineWidth',2.5,...
        'MarkerEdgeColor','k',...
        'MarkerFaceColor','k',...
        'MarkerSize',4);
    %% title
    title_plot=['Probability Level = ',...
                num2str(ProbabilityLevel(i)*100),'%, p = ',num2str(p)];
    title(title_plot,'FontSize',14)
    %% axis and ticks
    axis([0.6 1.4 -0.003 0.002]);
    set(gca,'FontSize',12);
    %% labels and legend
    xlabel('x/c','FontSize',18)
    ylabel('Cf','FontSize',18)
    legend1=['Mixed UQ, lower bounds'];
    legend2=['Mixed UQ, upper bounds'];
    legend3=['Pure Aleatory UQ'];
    legend_plot=legend(legend1,legend2,legend3,'Location','North');
    set(legend_plot,'FontSize',10);
```

```
end
FigNum=FigNum+1;
%%
%% clear variables for speeding up, added 5/17/2011
% clearvars -except...
%      FigNum XC_plot Mean R_XC1 CDF_XC1 R_XC2 CDF_XC2 R_XC3 CDF_XC3;
end
```

APPENDIX E

MATLAB SOURCE CODE: UNCERTAINTY QUANTIFICATION OF PHASE-AVERAGED X-VELOCITY DISTRIBUTIONS

```
function FigNum=PhaseU_UQ(p,FigNum,SampleNumOrder)
%% Phase averaged U profile at three locations: x/c=0.66;0.80;1.00
%% Mixed UQ and Pure Aleatory UQ
%% Input 'p' is the order of PCE
%% Input 'FigNum' is figure number
%% created 4/22/2011: based on 'PhaseU'

%% revised 5/12/2011: Add the Sobol Indices analysis

%% revised 5/19/2011: Re-pick the points at x/c = 0.80.

%% revised 6/7/2011: add the 95% CI plot.

%% revised 6/18/2011: add bar in the 95% CI plot.

%% revised 6/27/2011: Update the plots for thesis.

%% revised 6/29/2011: Put in the appendix of my thesis.

%% for test use only
tic;
close all;
clear all;
clc;
p=4;
FigNum=1;
SampleNumOrder=3;
%%
% x/c and its flag in the file name
XC=[0.66,0.80,1.00];
X=[66,80,100];
% XC=[0.66];
% X=[66];
Phase=[80,170,260,350];
XCNum=length(XC);
PhaseNum=length(Phase);
RunNums=(p+1)^3;
RunNum=1:RunNums;
%% probability level
ProbabilityLevel=[0.025,0.5,0.975];
ProbabilityLevelNum=length(ProbabilityLevel);
path=['C:\Users\dhx88\Daoru\Case3\C3\POST\ThesisWork\'];
%%
for i=1:XCNum% x/c location loop
    %% Preallocate U matrix for different locations
    if (X(i)==66)
        Y=zeros(494,1);
        U=zeros(494,RunNums);
        FirstPoint=101;
        LastPoint=351;
    elseif (X(i)==80)
        Y=zeros(368,1);
        U=zeros(368,RunNums);
        FirstPoint=51;
```

```
        LastPoint=201;
    elseif (X(i)==100)
        Y=zeros(261,1);
        U=zeros(261,RunNums);
        FirstPoint=51;
        LastPoint=201;
    end
    %%
%     PlotU=figure(FigNum);
%     hold all;
%     % set the position of the plot
%     set(PlotU,'Position',[50 50 1024 768]);
    % subplot number
    SubPlotNum=1;
    for j=1:PhaseNum% phase angle loop
        %% go back to this 'first' figure to fit the CI plot
        PlotU=figure(FigNum);
        hold all;
        % set the position of the plot
        % [left bottom width height]
        % set(PlotU,'Position',[50 50 1024 768]);
        % This size fits the pdf file of A4 paper.
        set(PlotU,'Position',[50 -400 900 1150]);
        %% read the experimental data
        path_exp=['C:\Users\dhx88\Daoru\Case3\C3\POST\EXP\'];
        filestr_exp...
            =['U_exp_x',num2str(X(i)),'-',num2str(Phase(j)),'d.dat'];
        filename_exp=[path_exp,filestr_exp];
        data_exp=load(filename_exp);
        % reduced velocity and y/c coordinate
        % data information in the experimental data
        % display...
        % ('"x/c","y/c","u/Uinf","v/Uinf","uu/Uinf^2",...
        % "vv/Uinf^2","uv/Uinf^2"')
        U_exp=data_exp(:,3);
        Y_exp=data_exp(:,2);
        %% Plot
        %% creat the U matrix (Quadrature Points data)
        for k=1:RunNums
            [Y,U(:,k)]=ReadPhaseU(p,RunNum(k),X(i),Phase(j));
            % size(Y)
            % size(U)
            [Y,U(:,k)]=Sortxy(Y,U(:,k));% sort the points
        end
        % Y
        % U
        % size(U)
        %% Totally 494 points at x/c = 0.66
        %% Totally 368 points at x/c = 0.80
        %% Totally 261 points at x/c = 1.00
        PointNumber=length(Y);
        %% Mixed UQ
        PointIndex=1;% This is used for plot
        Y_plotNum=(LastPoint-FirstPoint)/5+1;
```

```
Y_plot=zeros(Y_plotNum,1);
U_Mixed_min=zeros(Y_plotNum,ProbabilityLevelNum);
U_Mixed_max=zeros(Y_plotNum,ProbabilityLevelNum);
U_CI_min=zeros(Y_plotNum,PhaseNum);
U_CI_max=zeros(Y_plotNum,PhaseNum);
U_Aleatory=zeros(Y_plotNum,ProbabilityLevelNum);
for index=FirstPoint:5:LastPoint
    % y/c index at a fixed x/c location index
    AlphaStar=U(index,:);
    Alpha=PolyCoef(p,AlphaStar);
    %% Select three locations for the Sobol indices analysis
    %% added 5/12/2011
    if (XC(i)==0.66)
        % pick up a y/c location between wall and main stream.
        if index==101
            % near the wall
            % y/c = 0.11322 for x/c = 0.66 location
            % y/c = 0.031276 for x/c = 0.80 location
            % y/c = 0.0064552 for x/c = 1.0 location
            str=['Sobol Indices for Phase-averaged U/U_inf',...
                ' at x/c = ',num2str(XC(i)),...
                ', y/c = ',num2str(Y(index)),...
                ' (near the wall)'...
                ', Phase = ',num2str(Phase(j)),...
                ' degree'];
            disp(str);
            [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
        end
        if index==201
            % y/c = 0.11876, in the middle of BL for x/c = 0.66
            str=['Sobol Indices for Phase-averaged U/U_inf',...
                ' at x/c = ',num2str(XC(i)),...
                ', y/c = ',num2str(Y(index)),...
                ' (between wall and main stream)'...
                ', Phase = ',num2str(Phase(j)),...
                ' degree'];
            disp(str);
            [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
        end
        if index==351
            % y/c = 0.14792, near the main stream for x/c = 0.66
            str=['Sobol Indices for Phase-averaged U/U_inf',...
                ' at x/c = ',num2str(XC(i)),...
                ', y/c = ',num2str(Y(index)),...
                ' (near main stream)'...
                ', Phase = ',num2str(Phase(j)),...
                ' degree'];
            disp(str);
            [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
        end
    elseif (XC(i)==0.80)% x/c = 0.80
        if index==71
            % near the wall
            % y/c = 0.026463 for x/c = 0.80 location
```

```
            str=['Sobol Indices for Phase-averaged U/U_inf',...
                ' at x/c = ',num2str(XC(i)),...
                ', y/c = ',num2str(Y(index)),' (near the wall)'...
                ', Phase = ',num2str(Phase(j)),...
                ' degree'];
            disp(str);
            [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
        end
        if index==166
            % y/c = 0.10869, between wall and main stream
            % for x/c = 0.80
            % y/c = 0.082942, between wall and main stream
            % for x/c = 1.0
            str=['Sobol Indices for Phase-averaged U/U_inf',...
                ' at x/c = ',num2str(XC(i)),...
                ', y/c = ',num2str(Y(index)),...
                ' (between wall and main stream)'...
                ', Phase = ',num2str(Phase(j)),...
                ' degree'];
            disp(str);
            [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
        end
        if index==191
            % near the main stream
            % y/c = 0.20204 for x/c = 0.80
            % y/c = 0.19335 for x/c = 1.0
            str=['Sobol Indices for Phase-averaged U/U_inf',...
                ' at x/c = ',num2str(XC(i)),...
                ', y/c = ',num2str(Y(index)),...
                ' (near main stream)'...
                ', Phase = ',num2str(Phase(j)),...
                ' degree'];
            disp(str);
            [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
        end
    else% x/c = 1.0
        if index==101
            % near the wall
            % y/c = 0.11322 for x/c = 0.66 location
            % y/c = 0.031276 for x/c = 0.80 location
            % y/c = 0.0064552 for x/c = 1.0 location
            str=['Sobol Indices for Phase-averaged U/U_inf',...
                ' at x/c = ',num2str(XC(i)),...
                ', y/c = ',num2str(Y(index)),...
                ' (near the wall)'...
                ', Phase = ',num2str(Phase(j)),...
                ' degree'];
            disp(str);
            [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
        end
        if index==166
            % y/c = 0.10869, near the middle of BL for x/c = 0.80
            % y/c = 0.082942, near the middle of BL for x/c = 1.0
            str=['Sobol Indices for Phase-averaged U/U_inf',...
```

```
            ' at x/c = ',num2str(XC(i)),...
            ', y/c = ',num2str(Y(index)),...
            ' (between wall and main stream)'...
            ', Phase = ',num2str(Phase(j)),...
            ' degree'];
        disp(str);
        [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
    end
    if index==191
        % near the main stream
        % y/c = 0.20204 for x/c = 0.80
        % y/c = 0.19335 for x/c = 1.0
        str=['Sobol Indices for Phase-averaged U/U_inf',...
            ' at x/c = ',num2str(XC(i)),...
            ', y/c = ',num2str(Y(index)),...
            ' (near main stream)'...
            ', Phase = ',num2str(Phase(j)),...
            ' degree'];
        disp(str);
        [S1,S2,S3,ST1,ST2,ST3]=SobolIndices(Alpha)
    end
end
%% Mixed UQ: Second-Order Probability Sampling with Response
%% Surface
%% Epistemic loop
% xi1=-1.00:0.05:1.00;
%% sample epistemic loop
SampleStr_1D='SamplesE2_1D.txt';
SampleFile_1D=[path,SampleStr_1D];
xi1=textread(SampleFile_1D,'%f');
%%
EpistemicNum=length(xi1);
U_Point=zeros(ProbabilityLevelNum,EpistemicNum);
for ii=1:EpistemicNum% Epistemic loop
    %% Aleatory loop
    %% read the samples
    % SampleStr=['SamplesE2.txt'];
    SampleStr=['SamplesE',num2str(SampleNumOrder),'.txt'];
    SampleFile=[path,SampleStr];
    [xi2,xi3]=textread(SampleFile,'%f %f','delimiter',',');
    SampleNum=length(xi2);% number of sample points
    %% response surface evaluation
    R=zeros(1,SampleNum);
    for jj=1:SampleNum
        xi=[xi1(ii),xi2(jj),xi3(jj)];
        R(jj)=ResponseSurface(Alpha,xi);
    end
    %% Sort and calculate CDF
    %% use my own CDF function, modified 5/17/2011
    R=sort(R);
    R_cdf=MyCDF(R);
    % R_cdf=UniCDF(R);
    % [R_cdf,R]=Sortxy(R_cdf,R);
    %% Find the value for the corresponding probabity level
```

```
    for kp=1:ProbabilityLevelNum% at each probability level
        for kk=1:SampleNum% find the value
            if R_cdf(kk) > ProbabilityLevel(kp)
                % A=kk
                % B=R_cdf(kk)
                %% linear interpolation
                RR=R(kk)...
                    -(R_cdf(kk)-ProbabilityLevel(kp))...
                    /(R_cdf(kk)-R_cdf(kk-1))...
                    *(R(kk)-R(kk-1));
                U_Point(kp,ii)=RR;
                break;
            end
        end
        U_Mixed_min(PointIndex,kp)=min(U_Point(kp,:));
        U_Mixed_max(PointIndex,kp)=max(U_Point(kp,:));
    end% at each probability level
end% Epistemic loop for Mixed UQ
%% Pure Aleatory UQ
%% read the samples
% SampleStr_Aleatory=['SamplesE2_3D.txt'];
SampleStr_Aleatory...
    =['SamplesE',num2str(SampleNumOrder),'_3D.txt'];
SampleFile=[path,SampleStr_Aleatory];
[xi1,xi2,xi3]...
    =textread(SampleFile,'%f %f %f','delimiter',',');
SampleNum_Aleatory=length(xi1);
%% response surface evaluation
R_Aleatory=zeros(1,SampleNum_Aleatory);
for jj=1:SampleNum_Aleatory
    xi_Aleatory=[xi1(jj),xi2(jj),xi3(jj)];
    R_Aleatory(jj)=ResponseSurface(Alpha,xi_Aleatory);
end
%% Sort and calculate the CDF
R_Aleatory=sort(R_Aleatory);
R_cdf_Aleatory=MyCDF(R_Aleatory);
% R_cdf_Aleatory=UniCDF(R_Aleatory);
% [R_cdf_Aleatory,R_Aleatory]...
% =Sortxy(R_cdf_Aleatory,R_Aleatory);
%% Find the value for the corresponding probability level
U_Point_Aleatory=zeros(ProbabilityLevelNum,1);
for kp=1:ProbabilityLevelNum%each probability level
    for kk=1:SampleNum_Aleatory%find the value
        %% For Aleatory UQ
        if R_cdf_Aleatory(kk) > ProbabilityLevel(kp)
            %% linear interpolation
            RR_Aleatory=R_Aleatory(kk)...
                -(R_cdf_Aleatory(kk)...
                -ProbabilityLevel(kp))...
                /(R_cdf_Aleatory(kk)-R_cdf_Aleatory(kk-1))...
                *(R_Aleatory(kk)-R_Aleatory(kk-1));
            U_Point_Aleatory(kp)=RR_Aleatory;
            break;
        end
```

```
            end
            U_Aleatory(PointIndex,kp)=U_Point_Aleatory(kp);
        end%Probability Level loop
        %U_Aleatory
        %% for plot use
        Y_plot(PointIndex)=Y(index);% for plot use
        PointIndex=PointIndex+1;
    end% y/c loop at a fixed x/c location
    % Y_plot
    %% Plot
    %% at each probability level
    for kp=1:ProbabilityLevelNum
        subplot(4,3,SubPlotNum);
        hold all;
        %% experimental data
        plot(U_exp,Y_exp,'rd',...
            'MarkerSize',6,'MarkerFaceColor','r');
        %% Mixed UQ
        % sort
        %[U_Mixed_min(:,kp),Y_plot]=Sortxy(U_Mixed_min(:,kp),Y_plot);
        %[U_Mixed_max(:,kp),Y_plot]=Sortxy(U_Mixed_max(:,kp),Y_plot);
        plot(U_Mixed_min(:,kp),Y_plot,'b--','LineWidth',2.5);
        plot(U_Mixed_max(:,kp),Y_plot,'b-.','LineWidth',2.5);
        %% Pure Aleatory UQ
        %Y_plot
        %U_Aleatory
        % sort
        %[U_Aleatory(:,kp),Y_plot]=Sortxy(U_Aleatory(:,kp),Y_plot);
        plot(U_Aleatory(:,kp),Y_plot,'kd-','LineWidth',2.5,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','k',...
            'MarkerSize',4);
        %% labels and title
        xlabel('U/Uinf-mean','FontSize',12);
        ylabel('y/c','FontSize',12);
        TitleStr=['p = ',num2str(p),', x/c = ',num2str(XC(i)),...
            ', phase = ',num2str(Phase(j)),...
            ' deg, ',num2str(ProbabilityLevel(kp)*100),'%'];
        title(TitleStr,'FontSize',8);
        %% Legend
        legend1=['Exp'];
        legend2=['Mixed UQ, lower bounds'];
        legend3=['Mixed UQ, upper bounds'];
        legend4=['Pure Aleatory UQ'];
        legend_exp=...
            legend(legend1,legend2,legend3,legend4,...
            'Location','NorthWest');
        set(legend_exp,'FontSize',6);
        %% axis bounds
        if (X(i)==66)
            a=0.4;
            b=1.2;
            c=0.11;
            d=0.15;
```

```
        else
            a=-0.5;
            b=1.4;
            c=0.0;
            d=0.2;
        end
        axis([a b c d]);
        %% update subplot number
        SubPlotNum=SubPlotNum+1;
    end% probability level loop in the plot
    %% plot the 95% C.I.
    %% added 6/7/2011
    U_CI_min(:,j)=U_Mixed_min(:,1);% the first column
    U_CI_max(:,j)=U_Mixed_max(:,ProbabilityLevelNum);% the last column
    %% update the figure number
    %% FigNum=FigNum+1;
    PlotU_CI=figure(FigNum+1);
    hold all;
    % set the position of the plot
    % [left bottom width height]
    % This size fits the pdf file of A4 paper.
    set(PlotU_CI,'Position',[50 -400 900 1150]);
    subplot(4,1,j);
    hold all;
    %% experimental data
    plot(U_exp,Y_exp,'rd',...
        'MarkerSize',6,'MarkerFaceColor','r');
    plot(U_CI_min(:,j),Y_plot,'b--x','LineWidth',2.5,...
        'MarkerSize',10);
    plot(U_CI_max(:,j),Y_plot,'k-x','LineWidth',2.5,...
        'MarkerSize',10);
    %% 95% CI bar plot, added 6/18/2011
    PointNum=size(Y_plot,1);
    X1=zeros(1,PointNum);
    X2=zeros(1,PointNum);
    X1=U_CI_min(:,j);% lower bound
    X2=U_CI_max(:,j);% upper bound
    for BarIndex=1:PointNum% each bar
        XX=[X1(BarIndex),X2(BarIndex)];
        YY=[Y_plot(BarIndex),Y_plot(BarIndex)];
        line(XX,YY,'LineStyle','--','Color',[0.5 0.5 0.5]);% gray
        % line(XX,YY,'LineStyle','--','Color',[0 1 1]);% cyan
        % line(XX,YY,'LineStyle','--','Color',[0 1 0]);% green
        % line(XX,YY,'LineStyle','--','Color',[1 0 1]);% magenta
    end
    %% labels and title
    xlabel('U/Uinf-mean','FontSize',12);
    ylabel('y/c','FontSize',12);
    TitleStr=['95% CI for p = ',num2str(p),...
        ', x/c = ',num2str(XC(i)),...
        ', phase = ',num2str(Phase(j)),' deg'];
    title(TitleStr,'FontSize',12);
    %% Legend
    legend1=['Exp'];
```

```
        legend2=['Lower bound'];
        legend3=['Upper bound'];
        legend_exp=legend(legend1,legend2,legend3,'Location','NorthWest');
        set(legend_exp,'FontSize',10);
        %% axis bounds
        if (X(i)==66)
            a=0.4;
            b=1.2;
            c=0.11;
            d=0.15;
        else
            a=-0.5;
            b=1.4;
            c=0.0;
            d=0.2;
        end
        axis([a b c d]);
        %%
    end% phase angle loop
    %% save current figure to pdf file.
    filename=['XC',num2str(X(i)),'_CI.pdf'];
    saveas(gcf,filename,'pdf');
    %% update the figure number
    FigNum=FigNum+2;
end% x/c location loop
%% clear the variables for speeding up, added 5/17/2011
% clearvars -except FigNum;
toc;
end
```

BIBLIOGRAPHY

[1] Mohamed Gad el Hak. *Flow Control, Passive, Active and Reactive Flow Management*. Cambridge University Press, first edition, 2000.

[2] A. Glezer and M. Amitay. Synthetic jets. *Annual Review of Fluid Mechanics*, 34:503–529, 2002.

[3] Nasa langley research center workshop: Cfd valication of synthetic jets and turbulent separation control, http://cfdval2004.larc.nasa.gov/, March 2004.

[4] C. L. Rumsey, T. B. Gatski, W. L. Sellers, III, V. N. Vatsa, and S. A. Viken. Summary of the 2004 computational fluid dynamics validation workshop on synthetic jets. *AIAA Journal*, 44(2):194–207, 2006.

[5] C. L. Rumsey. Successes and challenges for flow control simulations (invited), AIAA 2008-4311. In 4^{th} *AIAA Flow Control Conference*, Seattle, WA, June 2008.

[6] S. Hosder and R. W. Walters. Non-Intrusive polynomial chaos methods for uncertainty quantification in fluid dynamics, AIAA 2010-129. In 48^{th} *AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Orlando, FL, January 2010.

[7] S. Adya, D. Han, and S. Hosder. Quantification of uncertainty integrated to the cfd modeling of synthetic jet actuators, AIAA 2010-4411. In 5^{th} *AIAA Flow Control Conference*, Chicago, IL, June 2010.

[8] Benjamin Robert Bettis. Quantification of uncertainty in aerodynamic heating of a reentry vehicle due to uncertain wall and freestream conditions. Master's thesis, Missouri Universtiy of Science and Technology, 2010.

[9] Srikanth Adya. Uncertainty quantification integrated to computational fluid dynamic modeling of synthetic jet actuators. Master's thesis, Missouri Universtiy of Science and Technology, 2011.

[10] M. Eldred and L. Swiler. Efficient algorithms for mixed Aleatory-Epistemic uncertainty quantification with application to Radiation-Hardened electronics. *Sandia National Laboratories Report*, SAND2009-5805, September 2009.

[11] L. P. Swiler and A. A. Giunta. Aleatory and epistemic uncertainty quantification for engineering applications. *Sandia National Laboratories Report*, SAND2007-2670C, July 2007.

[12] B. Bettis and S. Hosder. Quantification of uncertainty in aerodynamic heating of a reentry vehicle due to uncertain wall and freestream conditions, AIAA 2010-4642. In 10^{th} *AIAA Joint Thermophysics and Heat Transfer Conference*, Chicago, IL, June 2010.

[13] B. Bettis and S. Hosder. Quantification of uncertainty in aerodynamic heating of a reentry vehicle due to uncertain wall and freestream conditions, AIAA 2011-252. In 49^{th} *AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Orlando, FL, January 2011.

[14] C. L. Rumsey. Reynolds-averaged navier-stokes analysis of zero efflux flow control over a hump model, AIAA 2006-1114. In 44^{th} *AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 2006.

[15] W. L. Oberkampf, J. C. Helton, and K. Sentz. Mathematical representation of uncertainty, AIAA 2001-1645. In 3^{rd} *Non-Deterministic Approaches Forum*, Seattle, WA, April 2001.

[16] W. L. Oberkampf and J. C. Helton. Investigation of evidence theory for engineering applications, AIAA 2002-1569. In 4^{th} *Non-Deterministic Approaches Forum*, Denver, CO, April 2002.

[17] L. Swiler, T. Paez, R. Mayes, and M. Eldred. Epistemic uncertainty in the calculation of margins, AIAA 2009-2249. In 50^{th} *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Palm Springs, CA, May 2009.

[18] N. Wiener. The homogeneous chaos. *American Journal of Mathematics*, 60(4):897–936, 1938.

[19] D. Xiu and G. E. Karniadakis. Modeling uncertainty in flow simulations via generalized polynomial chaos. *Journal of Computational Physics*, 187(1):137–167, May 2003.

[20] J. B. Pleming D. S. Riha C. Waldhart L. Huyse, A. R. Bonivtch and B. H. Thacker. Verification of stochastic solutions using polynomial chaos expansions, AIAA 2006-1994. In 47^{th} *AIAA/ASME/ASCE/AHS/ASC Structures, Structual Dynamics, and Materials Conference*, Newport, RI, May 2006.

[21] M. S. Eldred, C. G. Webster, and P. G. Constantine. Evaluation of Non-Intrusive approaches for Wiener-Askey generalized polynomial chaos, AIAA 2008-1892. In 10^{th} *AIAA Non-Deterministic Approaches Forum*, Schaumburg, IL, April 2008.

[22] R. W. Walters and L. Huyse. Uncertainty analysis for fluid mechanics with applications. Technical report, ICASE 2002-1, NASA/CR-2002-211449, NASA Langley Research Center, Hampton, VA, 2002.

[23] H. N. Najm. Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics. *Annual Review of Fluid Mechanics*, 41:35–52, January 2009.

[24] I.M. Sobol'. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and Computers in Simulation*, 55:271–280, 2001.

[25] B. Sudret. Global sensitivity analysis using polynomial chaos expansion. *Reliability Engineering and System Safety*, 93(7):964–979, July 2008.

[26] T. Crestaux, O. L. Maître, and J.-M. Martinez. Polynomial chaos expansion for sensitivity analysis. *Reliability Engineering and System Safety*, 2009.

[27] S. Ghaffari, T. Magin, and G. Iaccarino. Uncertainty quantification of radiative heat flux modeling for titan atmospheric entry, AIAA 2010-239. In 48^{th} *AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Orlando, FL, January 2010.

[28] P. R. Spalart and S. R. Allmaras. A one-equation turbluence model for aerodynamic flows. *La Recherche Aerospatiale*, (1):5–21, 1994.

[29] *ANSYS FLUENT User Manual*.

VITA

Daoru Han was born on March 5, 1988, in the city of Zhoukou, Henan Province, of People's Republic of China (PRC). In September of 2005, he started his undergraduate study in Nanjing University of Aeronautics and Astronautics in the city of Nanjing, Jiangsu Province of PRC. He earned a Bachelor of Engineering degree in Power Engineering of Aircraft (Aeronautical Propulsion) in June 2009, and then began his graduate study at Department of Mechanical and Aerospace Engineering in Missouri University of Science and Technology (Missouri S&T) from August 2009. During his graduate career he served as the President 2010 of Chinese Students and Scholars Association (CSSA) at Missouri S&T in addition to a Graduate Research Assistant and Graduate Teaching Assistant. He defended in July 2011 receiving a degree of Master of Science in Aerospace Engineering from Missouri University of Science and Technology.