UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERING SCIENCE AND MATHEMATICS

School of Engineering Sciences

Robust Design Methodologies:
Application to Compressor Blades

by

Apurva Kumar

Thesis for the degree of Doctor of Philosophy

November 2006

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING SCIENCE AND MATHEMATICS

SCHOOL OF ENGINEERING SCIENCES

Doctor of Philosophy

ROBUST DESIGN METHODOLOGIES: APPLICATION TO COMPRESSOR BLADES

by Apurva Kumar

Compressor blades are subtle aerodynamic shapes designed after years of research and insight. They inevitably show deviations from their desired shapes due to manufacturing errors, erosion or foreign object damage. In the present study we focus on seeking compressor blade geometries, that are robust in performance in the presence of geometric uncertainty. Sophisticated tools for representing and propagating uncertainty are employed. Novel method for modeling eroded blade geometry and simulating manufacturing variations with process capability data are presented. These are combined with an automatic meshing routine and a high fidelity viscous flow solver for performance analysis. A combination of Design of Experiment techniques and Gaussian Process emulators are employed to develop efficient surrogate models for uncertainty analysis and exploring the design space.

Efficient multiobjective optimization based robust design methodologies are presented. The robust design methods in conjunction with the surrogate model are used to seek blades that have less variation in performance in the presence of erosion and manufacturing variations. Main effects and sensitivity analysis are also performed to understand the effect of each noise variable on the performance. The performance of the robust blades obtained are compared to that of deterministic optimal blades in the presence of the uncertainties. The robust optimal blades exhibit considerably less variability and mean shift in performance as compared to the optimal blades. Finally, a probabilistic framework is developed to deal with randomness in objectives during multiobjective optimization and is applied in conjunction with Gaussian Process emulators for robust design.

to my parents ....

Contents

List of Figures

List of Tables

Declaration of authorship

I, **Apurva Kumar**, declare that the thesis entitled **Robust Design Methodology: Application to Compressor Blades** and the work presented in it are my own. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published and some articles are under review as:

  1. A. Kumar, A.J. Keane, P.B. Nair and S. Shahpar. 'Robust Design of Compressor Blades against Erosion.' *Journal of Mechanical Design*, Volume 128, Issue 4, pp. 864-873, July 2006.
  2. A. Kumar, P.B. Nair, A.J. Keane and S. Shahpar. 'Robust Design using Bayesian Monte Carlo.' *International Journal for Numerical Methods in Engineering* - Submitted for Review.
  3. A. Kumar, P.B. Nair and A.J. Keane. 'Probabilistic Multi-objective Optimization Algorithm with Gaussian Process Emulators.' (In Preparation).

4. A. Kumar, A.J. Keane, P.B. Nair, and S. Shahpar. 'Robust Design for Compressor Fan Blades against Manufacturing Variations.' *In Proceedings of ASME IDETC/CIE*, **DETC2006-99304**, Pennsylvania, Sept 2006
5. A. Kumar, A.J. Keane, P.B. Nair and S. Shahpar. 'Efficient Robust Design for Manufacturing Process Capability.' *In Proceedings of 6th ASMO/ISSMO International Conference*, pp. 242-250, Oxford, July 2006
6. A. Kumar, A.J. Keane, P.B. Nair and S. Shahpar. 'Efficient GA based Robust Design Method for Compressor Fan Blades.' *In Proceedings of ASME IDETC/CIE*, **DETC2005-85042**, Long Beach (CA), September 2005.
7. A. Kumar, P.B. Nair, A.J. Keane and S. Shahpar. 'Probabilistic Performance Analysis of Eroded Compressor Fan Blades.' *In Proceedings of ASME Power Conference*, **PWR2005-50070**, Chicago, April 2005.

Signed:.....

Date:.....

Acknowledgements

I would like to thank Prof A. J. Keane for his support and for inspiring and motivating me from time to time. I am also very thankful to Dr. P. B. Nair for his guidance, advice and constant encouragement. I consider myself very fortunate to have both of them as my supervisors. I am thankful to Dr. S. Shahpar of Rolls Royce for his technical advice and helping me with PADRAM and HYDRA.

Special mention to my parents, bhaiya and bhabhi for their support and blessings without which this PhD would have been impossible. I have spent many moments to cherish during these three years with Abhi, Arturo, Arun, Bhaskar, Chincha, David, Dmitra, Mexican Twins, Nikita, Pola, Rishi, Sachin, Sree, Surya and BOSS! I would always remembers Abhi's prophecies, Chincha's debugging and bugging, Sachin's idiosyncracies, Sree's biryani, Surya's help, BOSS' cognac and Nikita's emotional support. Many thanks to Diya for rejuvenating me with her chuckles, laughters and gibberings.

This work was supported by the University Technology Partnership for Design, a collaboration between Rolls-Royce Plc., BAE Systems and the Universities of Sheffield, Cambridge and Southampton. I would like to take this opportunity to acknowledge and thank the Computational Engineering and Design Group for providing me with excellent research facilities and conducive working conditions.

Nomenclature

$\mathbf{\Gamma}$	Covariance matrix
$\boldsymbol{\theta}$	Vector of Hyperparameters
$\boldsymbol{\xi}$	Vector of random noise factors
χ	Control factor space
Δh	Tolerance in manufacturing process
Δh_t	Total specific enthalpy per unit mass
Γ	Covariance between two random variables
$\langle . \rangle$	Expectation operator
$\langle \mathcal{M} \rangle$	Bayesian Monte Carlo estimate of mean
$\mathbb{R}^p$	p-dimensional real space
$\mathbf{F}$	Set of vectors of objectives $\mathbf{f}$
$\mathbf{F}^*$	Pareto Set of vectors of objectives
$\mathbf{f}^{(i)}$	Vector of objective function at i^{th} design point
$\mathbf{x}$	Vector of control (design) factors
$\mathcal{C}(x, x')$	Posterior covariance between x and x'
$\mathcal{D}_n$	Training dataset with n design points
$\mathcal{E}$	Noise factor space
$\mathcal{F}^{(i)}$	i random objective vector

$\mathcal{GP}$ Gaussian Process

$\mathcal{L}$ Loglikelihood function

$\mathcal{N}(m, \Gamma)$ Gaussian random field with mean m and a covariance Γ

$\mathcal{O}$ Order of complexity

$\mathcal{P}$ Parent generation in Genetic Algorithm

$\mathcal{Q}$ Offspring generation in Genetic Algorithm

$\text{erfc}(x)$ complementary error function

$\text{erf}(x)$ error function

$\mathscr{F}$ Family of random vectors

$\mathscr{F}^*$ Pareto set of a family of random vectors $\mathscr{F}$

$\mathscr{P}$ Probability of occurance

ω wheel speed

$\phi(\mathbf{X}^*)$ Baye's risk

σ_p Manufacturing process variance

$\sigma_Y^2(x)$ Posterior variance at x

$\sigma_{\mathcal{M}}^2$ Bayesian Monte Carlo error in estimation of mean

σ_{share} Sharing parameter

σ_y Sample estimate of the variance of y

$\succ$ Dominance opertaor

υ Flow velocity

ϑ Flow turning angle

$\widehat{y}$ Approximate response

$\widetilde{y}$ Monte-Carlo estimate

A Parameter for depth of erosion

F_s Factor of safety

$m_i(x)$ posterior mean of the i^{th} output response at x

P Probability Density Function (PDF)

P_l Pressure loss coefficient

S_{i_k} k^{th} order global sensitivity index for the i^{th} variable

t_1 Parameter for location of erosion

t_2 Parameter for width of erosion

W_i Weight factors for each hick-henne function for manufacturing problem

w_i weight coefficients in weighted sum optimization

x_{M_i} Location of maxima of hick-henne function

y Response process

$Y(x)$ Gaussian Stochastic Process Model

$Y_{-i}(x)$ Gaussian Stochastic Process Model trained without i^{th} training point

$Z(x)$ Gaussian function

$\mathbf{X}^*$ Baye's decision

ANOVA Analysis of Variance

BMCS Bayesian Monte Carlo Simulation

CA Combined Array

CFD Computational Fluid Dynamics

DHC Dynamic Hill Climbing

DOE Design of Experiments

FON Fonseca function

FORM First Order Reliability Method

GA Genetic Algorithm

LHS Latin Hypercube Sampling

MCS Monte-Carlo Simulation

MSD Mean Squared Deviation

NSGA-II Non-dominated Sorting Genetic Algorithm

OA Orthogonal Array

PADRAM Parametric Design and Rapid Meshing

PDF Probability Density Function

POI Probability of Improvement

RANS Reynolds Averaged Navier Stokes

SCH Schaffer test function

SCVR Standard Cross-Validated Residual

SNR Signal to Noise Ratio

SORM Second Order Reliability Method

WS Weighted Sum

ZDT Zitzler Deb Thiele function

Chapter 1

Introduction

1.1 Motivation

The performance and integrity of compressor blades are central to the behaviour of modern gas turbine engines. These blades are subtle airfoils designed after years of insight and research, however, they inevitably exhibit deviations from their designed shape due to manufacturing errors, erosion and damage. This can cause considerable deterioration in aerodynamic performance and may even lead to replacement of the expensive blades. Recent changes in the aviation power industry and new market paradigms such as *Power By The Hour* by Rolls-Royce, which offer operational guarantees on items such as performance retention, reliability, in-flight shutdowns, maintainability, maintenance cost, etc; make the need for robust design very important in the present scenario. This motivates our current work of developing methods to seek compressor blades that have robust performance in the face of erosion and manufacturing errors.

The presence of uncertainty in designing any engineering system is inevitable. Uncertainty could be present for many reasons, including lack of knowledge about material properties, uncertain boundary conditions and operating conditions, manufacturing tolerances, etc. Traditional methods of deterministic design ignore the presence of uncertainty which can lead to undesirable variations in the design performance. Probabilistic design methods can be employed to quantify, propagate and manage uncertainty in the design process. More often than not it can be very expensive or perhaps impossible to eliminate the sources of uncertainty in a system, such as manufacturing precision, operating temperature or humidity levels. One of the popular probabilistic methods is robust design which seeks designs that are less sensitive to the presence of uncertainty in the system, without removing the source of uncertainty.

It exploits the interactions between the design variables and uncertain parameters to propose designs that are robust (Phadke, 1989). Robust design essentially aims at reducing the variability in the performance of a system, along with seeking improvement in the mean performance (Taguchi & Wu, 1980). There are many methods for conducting robust design optimization and the most common are the Taguchi methods, Metamodel based methods, Statistical decision theory based methods and Multi-objective optimization methods (Nair, 1992; Sanchez, 2000; Trosset, 1996).

There is a wide body of literature dealing with application of robust design methods to engineering systems. These contributions have been mainly in the area of experimental set up, defining variability metrics and objective formulations. Unfortunately, robust design methods are still not widely used in industry. The primary reason being that, industries like those in the aerospace sector, are continuously moving towards higher fidelity models to simulate the design performance. For example, in the field of aerodynamic design, there is an increasing push from low fidelity codes like panel methods, to high fidelity methods like Reynolds Averaged Navier Stokes solvers (RANS), Large Eddy Simulation (LES) etc. Such high fidelity models incur high computational cost and quantifying uncertainty in the face of uncertainty can be prohibitive. The focus of the present research is to propose efficient surrogate model based robust design methods which can be used effectively in a sequential optimization framework to arrive at robust designs on a limited computational budget. The research also focuses on explicitly exploiting the trade-offs between mean and standard deviation of the performance in a sophisticated multiobjective framework. The strategy is essentially 1) to formulate efficient surrogate based robust design methods for high fidelity analysis problems, 2) to devise strategies to systematically improve the quality of the surrogate to predict robust solutions. The resulting surrogate can then be used to produce robust and high quality designs.

1.2 Background

Traditional methods in design optimization aim to optimize the nominal performance of the system. This may lead to designs which have good performance at the design points but may have poor off-design performance. A classical example in aerodynamic optimization is to seek an airfoil which has good performance across a range of Mach numbers. Researchers have solved this problem from a deterministic point of view using multi-point optimization

techniques. These methods are computationally very expensive and though one may achieve designs with good performance at the multiple design points there is no guarantee that the design would have good performance at some other intermediate point (Drela, 1998).

Non-deterministic methods which systematically manage and propagate uncertainty through the system are best suited to solve such problems. Two major class of probabilistic methods are *Robust Design Methods* and *Reliability Based Design Methods*. Robust design methods seek to reduce the variability in the performance of the system while reliability based methods suggest designs with the probability of failure lower than a prescribed value. Robinson (1998) provides a survey of probabilistic methods for performing reliability and uncertainty analysis of complex engineering systems. Unlike traditional methods using factors of safety and knockdown factors, the probabilistic methods provide a systematic and quantitative approach. The most commonly used reliability based methods are first and second order reliability methods (FORM and SORM) (Madsen *et al.*, 1986; Chandu & Grandhi, 1995; Allen & Maute, 2004). One of the early approaches to reduce variations in performance was the *Six Sigma Quality* method, which required that the ± 6 standard deviations of the performance lie between the mean and a prescribed value (Harry, 1997). The Six Sigma Quality method has been employed by many companies like Motorola and General Electric to improve their product performance (Pande *et al.*, 2000).

In the early 1970's Taguchi re-emphasized the need to reduce the variation in the performance of products (Taguchi & Wu, 1980). He proposed to exploit the interactions between the design variables and noise variables to achieve robust designs (Taguchi, 1986; Phadke, 1989). Taguchi's methods found widespread applications in the manufacturing and electronic industry, but these methods had several limitations for complex design problems (Nair, 1992; Box, 1988; Tsui, 1996). With the growing interest in robust design many improvements on Taguchi's methods were presented. Welch *et al.* (1990); Welch & Sacks (1991) proposed a combined array and metamodel based method for robust design using computer experiments. Ben-Tal & Nemirovski (1997) used min-max method, based on statistical decision theory, for robust design studies. A genetic algorithm based robust solution searching scheme was proposed by Tsutsui & Ghosh (1997). Robust design methods of these forms have also been widely used in the industry.

Probabilistic methods like robust design have found limited applications to problems involving computationally expensive analysis such as aerodynamic design. This is primarily because they demand many evaluations of these high fidelity models which turn out to be

prohibitively expensive. In the last few years there has been a resurgent increase in probabilistic design methods in aerospace design (Zang *et al.*, 2002). Huyse & Lewis (2001) studied the classical airfoil optimization problem of minimizing drag while allowing for uncertainty in operating conditions. They used a free form airfoil optimization approach with an inviscid Euler solver to estimate the drag. In (Huyse, 2001), they compared the results obtained from deterministic single point and multi point optimization methods, with a robust design based method. They employed a statistical decision theory based min-max approach method which seeks to optimize the worst case scenario of the performance (Li *et al.*, 2002). Putko *et al.* (2001) use sensitivity derivatives in an approximate statistical moment method to seek robust solutions. They used a quasi 1-D Euler code for CFD simulation and sought nozzle designs robust to geometric variability and uncertainty in input conditions. Mavris & Bandte (1997) applied probabilistic methods applied to aircraft system design for a high speed civil transport system. Mavris *et al.* (1998) compared Monte Carlo Simulation (MCS), Response Surface Methodology and fast probability integration methods for probabilistic design of a commercial aircraft engine cycle.

Despite the previously cited work probabilistic methods have not found widespread application in aerothermal analysis and design of turbomachinery components. In the last decade these methods have been applied to optimize engine cycle parameters and individual engine components (Egorov, 1992; Egorov & Kretinin, 1993; Garzon & Darmofal, 2001, 2003). Egorov (1992); Egorov & Kretinin (1993) proposed a procedure for optimizing multistage axial compressor parameters in a stochastic setup. Turbomachinery components like compressor fan blades are central to the performance of modern gas turbine engines. These blades are sensitive shapes and are manufactured under tight tolerancing. However, finished airfoils have deviations from their intended shape due to manufacturing errors or erosion and damage (Roberts, 1984; Balan & Tabakoff, 1984).

During operation, compressor fan blades are exposed to a number of erosion processes (Metwally *et al.*, 1995). This can lead to reduction of the blade chord, alteration in the shape and increase in the surface roughness (Balan & Tabakoff, 1984; Hamed *et al.*, 1998). In their study, Balan & Tabakoff (1984) use a semi-empirical erosion model, derived from erosion tests of material samples at different particulate flow conditions, to predict the blade erosion patterns and locations. Roberts (1984) has shown that geometric variability in the form of leading edge erosion in compressor airfoils may account for an increase of 3% or more on the thrust specific fuel consumption.

Manufacturing uncertainty in end products always occur due to variations in production conditions like temperature, humidity, pressure and other deviations due to tool wear and human errors. Departure from *datum* design may also occur due to limitations of the manufacturing process or existence of tolerances. It is very important to understand the effect of such manufacturing uncertainty on the performance of the product. Garzon & Darmofal (2001) proposed a principal component analysis (PCA) based method to minimize the effect of shape variability due to manufacturing errors on the performance of the compressor cascade. They use random MCS combined with response surface methods and probabilistic quadrature methods to seek robust compressor designs. They report up to 15% decrease in the shift of mean loss from the nominal loss in a compressor cascade (Garzon & Darmofal, 2003). In this study we aim to seek compressor blades that are less sensitive to erosion and manufacturing errors.

1.3 Scope and Objectives of the Thesis

The main objectives of this work are to 1) develop efficient methods to represent and propagate geometric uncertainty in compressor blades in order to quantify their impact on aerodynamic performance, 2) develop efficient methods for seeking robust optimal compressor blades that have reduced sensitivities to geometric uncertainty on a limited computational budget.

1.3.1 Uncertainty Representation and Propagation

Geometric uncertainty can be present in compressor blades due to manufacturing errors, erosion or damage. In the present study we focus on representing and propagating uncertainty due to erosion and manufacturing errors in blades. Firstly, we set up the design analysis framework for the compressor blade erosion problem to conduct robust design studies. We develop a parametric geometry definition of the compressor blade, which can be exploited during design optimization to explore the design space. Next, we develop a novel parametric model for geometric representation of eroded compressor blades and set up an automatic grid generation routine for subsequent high fidelity CFD analysis. The next step at hand is to select and train a surrogate model for efficient probabilistic analysis. We select an appropriate Design of Experiments (DOE) method and train the surrogate, which is subsequently used in lieu of the computationally expensive CFD simulation. Finally, we focus on conducting an

efficient probabilistic optimization study based on the surrogate to analyze the performance of blades in the presence of geometric uncertainty arising due to erosion. Similarly, for the manufacturing problem we first set up the design analysis framework. Process capability data for the blade manufacturing process is employed in conjunction with a parametric geometry model to quantify and simulate manufacturing uncertainty. Again surrogate models and DOE techniques are used to expedite the analysis process. This set up is then employed for probabilistic analysis and robust design studies.

1.3.2 Robust Design

We seek compressor fan blades that have robust performance in the presence of erosion and manufacturing uncertainty. The two problems 1) geometric uncertainty due to erosion (environment variables), and 2) geometric variations due to manufacturing uncertainty (design variables), are essentially different in nature and we propose different techniques for them. We begin with the aim to develop a generic robust design methodology for dealing with external noise (e.g. operating conditions or erosion etc.). This methodology is applied to the erosion problem where high fidelity CFD is employed for design analysis. Robust design methods with high fidelity models (CFD) can be computationally very expensive. We employ a sophisticated metamodel as a cheap surrogate to the expensive analysis solver in the robust design process. More often than not, reductions in variation of the performance is achieved at the cost of deterioration of the mean performance. The objective here is to develop a formulation which ensures an effective trade-off between both objectives.

In the case of manufacturing uncertainty, the noise variables (manufacturing uncertainty) is a subset of the design variables themselves. We employ a combined array based experiment for this problem. We present a new methodology based on Bayesian Monte Carlo technique to manage such problems. Bayesian Monte Carlo methods are employed to predict the statistics of interest more accurately and precisely. Finally, we develop a probabilistic framework for multiobjective optimization which exploits the errors in prediction of the surrogate models.

The focus of the thesis is to achieve robustness with high accuracy on a limited computational budget. Our scope is limited to developing efficient robust design methods and effective exploitation of surrogate models in a robust design framework for probabilistic analysis for high fidelity complex design analysis models. Improvements in the analysis model, CFD in our case, for modeling flow around complex eroded blades or blades with manufacturing defects are beyond the scope of this work. The issues in measurement of data and

subsequent development of input models for uncertainty analysis are also not tackled. In all the problems in this study, a simplified probabilistic representation of the input uncertainty is assumed.

1.4 Layout of the Thesis

The thesis is arranged as follows:

Chapter 2 presents an overview of probabilistic methods used in engineering design. Various methods for uncertainty propagation and quantification are discussed. The chapter also provides an overview of the more commonly used robust design methods. The multiobjective robust design formulation for robust design is also discussed.

Chapter 3 analyzes the effect of geometric uncertainty due to erosion and manufacturing variations on the performance of compressor blades. A novel parametric geometry model for eroded blades is presented. Process capability data based novel method for simulating manufacturing variations around the blade is also proposed. The CFD analysis solver HYDRA and the grid generator PADRAM is also introduced. Monte Carlo Simulations are conducted to estimate the performance variations and build a case for robust design studies

Chapter 4 presents a multi-objective formulation based robust design method for compressor fan blades against erosion. Design of Experiment techniques are used to train Gaussian Stochastic Process models to predict the mean and variance of the performance. Main effect studies are also conducted to understand the effect of each parameter on the output performance. The Non-dominated Sorting Genetic Algorithm (NSGA-II) is employed for the multi-objective optimization to seek the Pareto-optimal front. The performance of the robust blade and the deterministically optimized blade, in the presence of erosion, are compared.

Chapter 5 deals with robust design against manufacturing uncertainty for compressor blades. An efficient Bayesian Monte Carlo Simulation based robust design methodology is proposed. The methodology again uses Gaussian Process emulators in conjunction with NSGA-II for explicit trade-off between mean and variance in the output performance. Analysis of Variance based sensitivity analysis studies are conducted to understand the effect of noise variables on the blade performance. Finally, robust design studies are conducted on the blade and the performance of the robust blade is compared to that of the deterministically optimal blade.

Chapter 6 starts with discussing the issues involved in surrogate based multiobjective

robust design methods. A probabilistic framework is presented to extend the definitions of *dominance* and *Pareto Optimality* for random objectives. An efficient probabilistic ranking based multiobjective algorithm is proposed to conduct robust design studies. The algorithm is tested on a suite of test problems and the results are presented. Finally, we revisit the erosion and manufacturing uncertainty problem.

Chapter 7 summarizes the contributions and major conclusions of this research. Some directions for future research are also outlined.

Chapter 2

Overview of Uncertainty Modeling and Robust Design

In this chapter we present an overview of uncertainty quantification and robust design methods. We first discuss methods of representing uncertainty and various techniques of propagating uncertainty in an analysis. Subsequently we discuss approaches for design in the presence of uncertainty. Finally, we present a brief account of the most commonly used methods for probabilistic and non-probabilistic robust design.

2.1 Uncertainty Representation

Uncertainty representation is a prerequisite for the use of uncertainty based design methods. Oberkampf *et al.* (1998) categorized uncertainties into three different classes. *Variability* refers to "the inherent variations associated with the physical system or the environment under consideration." *Uncertainty* is "a potential deficiency in any phase or activity of the modeling process that is due to lack of knowledge." *Error* is "a recognizable deficiency in any phase or activity of modeling and simulation that is not due to lack of knowledge." Uncertainties can also be broadly classified into two groups to distinguish between their origin: *aleatory* and *epistemic* uncertainty (Helton *et al.*, 2004). Aleatory uncertainty is also referred to as irreducible uncertainty, inherent uncertainty or stochastic uncertainty. Aleatory uncertainty is used to describe the inherent variations associated with the physical system or the environment under consideration. Epistemic uncertainty is also referred to as reducible uncertainty, subjective uncertainty and model form uncertainty. Epistemic uncertainty derives from some level of ignorance or incomplete information of the system or the surrounding

environment.

Uncertainty is also widely categorized as *parametric* and *model form* uncertainty (Zang *et al.*, 2002). Parametric uncertainty is the uncertainty associated either with the input data of a computational model, such as input conditions or boundary conditions, or with the basic parameters which define a computational process. Model form uncertainties are associated with the model validity, such as the inadequacy of a mathematical model to capture the physics of a process. Another taxonomy of uncertainty in computer code outputs is given by O'Hagan & Oakley (2004); Haylock & O'Hagan (1996). They classify the uncertainties in four classes: *parametric*, *model*, *residual* and *code* uncertainty. The first two classifications are similar to the ones mentioned above. Residual uncertainty pertains to the variations in the results predicted by a model even when the conditions are fully specified, whereas code uncertainty is associated with the practical inability to run the code at all the relevant input configurations (for example sampling error in MCS).

The most commonly used probabilistic models for representing uncertainty in design are random variable, random field and time dependent stochastic processes. Probabilistic models are appropriate when sufficient data or information about the uncertain quantity is available. In some situations with limited data, non-probabilistic approaches may be required for uncertainty quantification. Non-probabilistic approaches which have attracted attention in the design community in recent years include evidence theory, possibility theory, interval analysis and convex modeling. A detailed study and application of these methods can be found in Ben-Haim & Elishakoff (1990); Rao & Chen (1998).

2.2 Uncertainty Propagation

Once any uncertainties are classified and represented, the next step is to propagate the uncertainty through the analysis code. In this section, we discuss approaches for uncertainty propagation through deterministic computational models. Given a set of input vectors, the computational model can be considered as a black box which provides outputs of interest. When probabilistic models with specified statistics are used to represent uncertainty, the uncertainty propagation problem essentially involves computing the statistics of the outputs of interest. When interval or convex models are employed to represent uncertainty, the uncertainty propagation involves evaluating the bounds on the outputs. The aim of the methods discussed in this section is to extract as much information as possible from a limited

set of experiments. Such methods are broadly referred to as Design of Experiments (DOE) techniques.

2.2.1 Monte Carlo Simulation

Monte Carlo Simulation (MCS) is the most popular DOE technique and can be applied to compute the statistics of the response quantities of interest with high accuracy, provided sufficient number of samples is used. MCS (Hurtado & Barbat, 1998; Gentle, 1998) employs a random number generator to select points say $\xi^{(1)}$,$\xi^{(2)}$,...,$\xi^{(n)}$, in the design space where the response quantity $y(\xi)$ is evaluated using a computational model. The statistics can be expressed as:

$$\langle y(\xi) \rangle \approx \widetilde{y} = \frac{1}{n} \sum_{i=1}^{n} y(\xi^{(i)}). \tag{2.1}$$

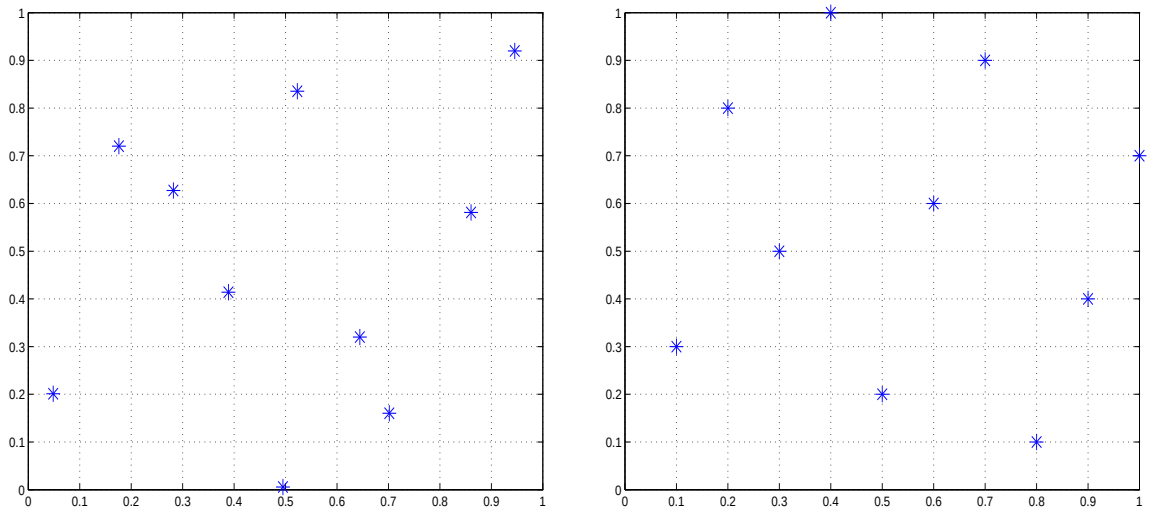
where $\widetilde{y}$ is referred to as the Monte Carlo estimate. The variance of the Monte Carlo estimate is:

$$Var(\widetilde{y}) = \frac{1}{n(n-1)} \sum_{i=1}^{n} \left(y(\xi^{(i)}) - \widetilde{y}\right)^2 = \frac{{\sigma_y}^2}{n}, \tag{2.2}$$

where σ_y is the sample estimate of the variance of $y(\xi)$. The variance computed using equation (2.2) can be used to evaluate the accuracy of the Monte Carlo estimate. It can be deduced from equation (2.2) that the standard error of $\widetilde{y}$ is independent of the dimension of the design space and is given by $\sigma_y/\sqrt{n}$. Hence, MCS estimate has a convergence rate of $\mathcal{O}(1/\sqrt{n})$.

MCS is employed as a method of last resort as it can be computationally very expensive to use with high fidelity computational models like CFD, FEM etc. The other major drawback of MCS is that the samples generated for evaluation are not essentially space filling. Unlike field and laboratory experiments, which have randomness and non-repeatability, computational models are deterministic. Therefore, to extract the most information it is important to choose training points which fill the design space in an optimal sense (Sacks *et al.*, 1989). Several pseudo MCS methods have been proposed to address the deficiency of basic MCS sampling. Some of the widely used pseudo MCS methods are Stratified MCS, Latin Hypercube Sampling (LHS) (Mckay *et al.*, 1979) and orthogonal array (OA) sampling. The underlying idea of Stratified MCS (Koehler & Owen, 1996) and LHS is to divide the design space into regions of equal probability (bins) and generate pseudo random points, such that no two points lie

in the same bin. Even so, the space-filling characteristics are not guaranteed to be good all the time. This has motivated the development of optimal LHS design which have a more uniform coverage of the design space, for example the algorithm used by Audze & Eglais (1977), see Figure 2.1. Conceptually, OA shares many similarities with LHS and the OA



LHS Optimized LHS

FIGURE 2.1: Spatial Distribution of points using standard and optimized LHS

algorithm can be used to produce latin hypercube samples (Hedayat *et al.*, 1999). The key feature of OA sampling is that it produces a set of samples that yield uniform sampling in any t-dimensional projection of an d-dimensional design space (where $t < d$).

2.2.2 Quasi Monte Carlo Simulation

Another class of DOE techniques for uncertainty propagation are the Quasi-Monte Carlo methods. These methods employ a deterministic algorithm to generate samples in an d-dimensional space. Sobol & Stanikov (1981) introduced the concept of *discrepancy* to allow quantitative assessment of the uniformity of a sequence of points. It refers to the measure of how much the distribution of samples deviates from an ideal uniform distribution. Hence, low discrepancy is a desired feature of this class of sampling methods. The most commonly used approach is the LP_τ method based on Sobol sequences (Sobol, 1994). Figure 2.2 shows the distribution of 100 points generated using the Sobol sequence.

LP_τ gives a mechanism for generating a deterministic sequence of points in space which is uniformly distributed. An important feature of LP_τ sampling is that it provides a way

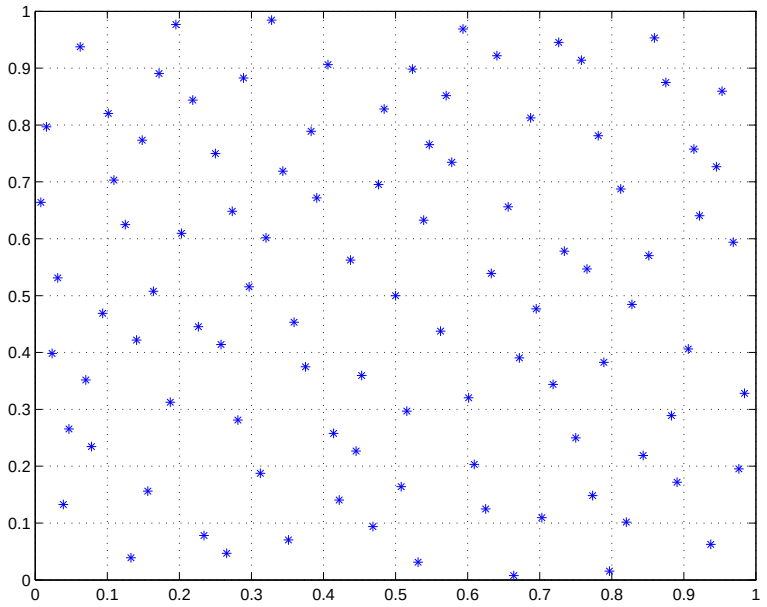


FIGURE 2.2: Spatial distribution of points generated from Sobol Sequence

DOE Method	Convergence
Random MCS	$\mathcal{O}(1/\sqrt{n})$
Pseudo MCS	$\mathcal{O}([\frac{\log\log n}{n}]^{1/2})$
Quasi MCS	$\mathcal{O}(\frac{[\log n]^d}{n})$

TABLE 2.1: Comparison between convergence of different MCS techniques.

to add more points to the initially sampled points with the same uniform characteristics. The actual computation of LP_τ sequences for any dimension can be done by consulting the tables provided by Sobol and Statnikov . Any sequence of uniform numbers can be chosen to generate LP_τ sequences. For an overview of the modern design of experiments methods for computational simulations the reader is referred to Giunta *et al.* (2003). Table 2.1 shows the comparison of the convergence for evaluating the statistics using various discussed MCS method. It should be noted, however, that for the quasi MCS methods the convergence is dependent on the number of dimensions of the problem (d) and hence is not preferred for high dimensional problems. For low dimensional problems ($d << n$) all sampling sequences that aim for uniform or low discrepancy sampling converge at the rate of $\mathcal{O}(1/n)$.

2.2.3 Surrogate Model Based Monte Carlo Simulation

The advent of faster computers gave hopes that probabilistic analysis using MCS would be easily achieved. However, the design community has also moved to more precise computational analysis models which are computationally expensive. For example, in the aerospace design community panel codes used to be the analysis tools whereas, now Reynolds Averaged Navier Stokes (RANS) models are used extensively in the industry. The computational cost associated with such high fidelity models make the use of MCS virtually impossible for probabilistic analysis. This motivates the use of approximate models which can be employed as computationally cheap surrogates to the original high fidelity analysis model for MCS.

Surrogate modeling uses the basic idea of analyzing an initial set of design points to generate data which can be used to construct approximations of the original high fidelity model. The high-fidelity model can be represented by a functional relationship $y = f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^p$ is the vector of inputs to the simulation code and y is the output. The objective is to construct an approximate model $\widehat{y} = \widehat{f}(\mathbf{x}, \alpha) \approx f(\mathbf{x})$, that is computationally cheaper to evaluate. α is a vector of undetermined parameters which is to be estimated using the observed data.There are a wide range of techniques like Response Surface Models, Radial Basis Function Approximations, Gaussian Stochastic Process Model, Neural Networks etc., to construct surrogate models from observed data; see for example (Vapnik, 1998; Mackay, 2003; Keane & Nair, 2005).

These surrogate models can be classified into two major categories, namely parametric and non-parametric models. The most widely used and simplest parametric model is the polynomial response surface model. This method employs DOE techniques with interpolation or regression technique for model building (Box & Draper, 1987; Myers & Montgomery, 2002). To illustrate the polynomial model, let $(\mathbf{x}^{(i)}, y^{(i)}, i = 1, 2, ..., n)$ denote the training dataset obtained by running the high fidelity analysis code at a set of initial points, which in turn can be obtained using any DOE technique. A quadratic response surface model for y can be expressed as

$$\widehat{y} = c_0 + \sum_{1 \le j \le p} c_j x_j + \sum_{1 \le j \le p, k \ge j} c_{j(p-j+2)+k-1} x_j x_k, \tag{2.3}$$

where c_0, c_j and $c_{p-1+j+k}$ are the undetermined coefficients in the model which can be evaluated by using an interpolation or regression technique. Hence once the parameters are evaluated there is no need to refer to the observed dataset again.

An alternative approach is the Gaussian stochastic process method which was developed

in the field of geostatistics (where this model is referred to as Kriging) and has been in use since the early 1960's (Matheron, 1963). It is also widely used in the neural network community where it is referred to as Gaussian process regression (Neal, 1996; MacKay, 1997). A Gaussian Stochastic Process model is a non-parametric model and the structure typically used to approximate the relationship $y = f(x)$ can be expressed as

$$Y(x) = g(x) + Z(x). \tag{2.4}$$

It can be seen as a combination of a global model ($g(x)$) and a local model ($Z(x)$). $g(x)$ is usually a linear or quadratic polynomial function, however, a constant $g(x) = \beta$ is often found to be sufficient for modeling complex functions. $Z(x)$ is a Gaussian random function with zero mean and non-zero covariance and is used to model the local deviations from the global model. The main advantage of Gaussian stochastic process model is that the user can get an estimate of the errors in prediction. This can be exploited in optimization procedures to update the surrogate model. Gaussian Stochastic Process Models have been widely used by researchers in optimization studies (Ong *et al.*, 2003; Keane, 2003a; Ong *et al.*, 2004). The basic idea of surrogate model based MCS is to replace y in equation (2.1) by $\widehat{y}$ obtained using the computationally less expensive surrogate.

2.3 Uncertainty in Design

The presence of some degree of uncertainty in characterizing any real engineering system is inevitable. Uncertainty in a system can be present at a macro scale due to changes in environment and operating condition, or at a micro scale due to non-homogeneous microstructure in materials. In aerodynamic design, the most common uncertainties observed are variations in nominal geometry due to erosion, icing, damage or manufacturing tolerances and uncertainty in loading due to changes in flight speed and other nondeterministic operating environment conditions. Traditional design methods tend to optimize designs for the nominal performance of the engineering system. Such methods, which assume deterministic values of the involved uncertainty in the system, are referred to as deterministic design methods. The deterministic methods currently in use for product design either tend to produce solutions that perform well at design points but may have poor off-design characteristics. As a consequence, an optimum design obtained without accounting for uncertainty can potentially be a high risk solution which is likely to violate design requirements or lead to failure when the final system is tested.

The conventional practices used to account for uncertainty are based on combinations of factors of safety (F_s) and knockdown factors. These methods aim to safeguard against uncertainty by imposing stringent conditions on the design. Typical value of F_s range from 1.2 to 3 and are applied to the loads, while knockdown factors take a value less than one and are applied to the strengths. The actual values are often decided on the basis of prior experience with the material being used and similar design concepts. These methods can be logically inconsistent and have several shortcomings. Deciding on F_s depends on the designers previous experience and knowledge which can not be readily applied to radical and unconventional designs. These methods also do not provide one with any measure of the robustness or reliability of the engineering system. Due to the constraints imposed by these methods, the design process can lead to designs with performance penalties without worthwhile improvements in reliability or robustness.

Unlike traditional deterministic methods, probabilistic design rationally accommodates uncertainties during the design process to arrive at reliable and robust designs. This is in contrast to the factor of safety based approach where parameter uncertainty is not explicitly incorporated in the design formulation - rather uncertainties arising from all sources are lumped into a single parameter. The two major classes of probabilistic design problems are *Robust Design* problems and *Reliability-based Design* problems. A robust design problem is

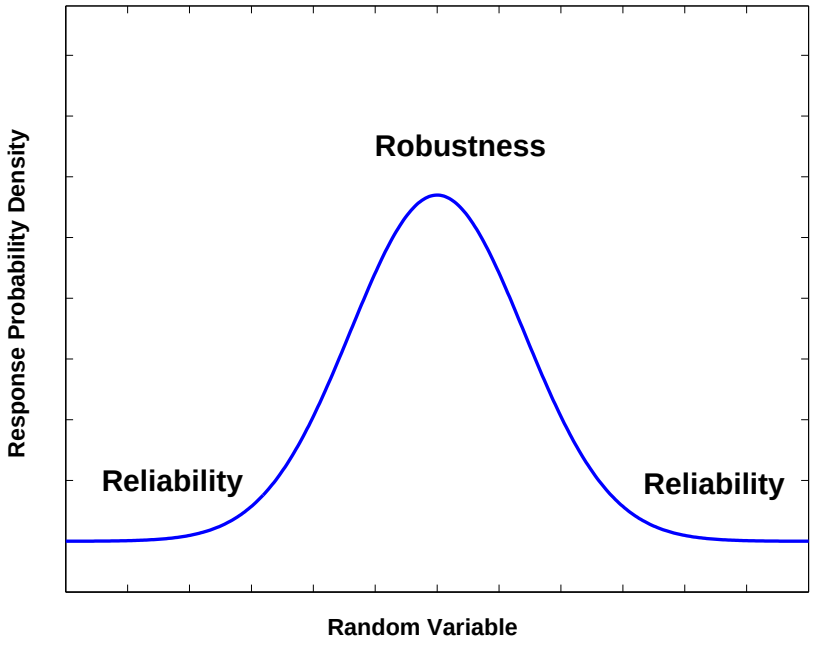


FIGURE 2.3: Reliability vs Robustness (Zang et al., 2002)

one in which a design is sought that is relatively insensitive to small changes in uncertain parameters. A reliability-based problem is one in which a design is sought that has a probability of failure less than a prescribed value. Hence, robust design is concerned with the event distribution near the mean of the probability distribution function (pdf) of the output, whereas reliability-based design is concerned with the event distribution in the tails of the pdf of output, see figure 2.3. The most commonly used reliability based methods are first and second order reliability methods (FORM and SORM) (Madsen *et al.*, 1986; Chandu & Grandhi, 1995; Allen & Maute, 2004). Robinson (1998) provides a survey of probabilistic methods for performing reliability and uncertainty analysis on complex engineering system. We do not discuss reliability methods further in this thesis. In the next section we discuss the robust design approach in more detail.

2.4 Robust Design

Robust Design aims at seeking designs that have reduced sensitivities to variations. These variations could be due to uncertainty in operating conditions, manufacturing process or the geometry due to erosion, icing, etc. Such uncertainty is inevitable and more often than not expensive to control. Traditional optimization methods do not take uncertainty into account and may lead to designs that have poor performance characteristics in their presence (sensitive designs). Figure 2.4 compares a robust design and a sensitive design. The fundamental idea of most of the robust design methods is to exploit the interactions between the parameters of the system to reduce variability, without reducing the sources of uncertainty (Phadke, 1989). In these method, the system parameters are classified into two groups 1) control factors ($\mathbf{x} = \{x_1, x_2, ..., x_p\} \in \chi$) and 2) noise factors ($\boldsymbol{\xi} = \{\xi^{(1)}, \xi^{(2)}, ..., \xi^{(q)}\} \in \mathcal{E}$). Control factors are the parameters which can be easily controlled by the designer whereas noise factors are the parameters that are difficult or expensive to control.

2.4.1 Taguchi Methods

In the early 1970's, Taguchi emphasized the need to reduce the variations in product performance (Taguchi & Wu, 1980). The Taguchi method consists of three stages of *system design*, *parameter design* and *tolerance design* (Taguchi, 1986). In system design one identifies the design space for exploration and defines the system configuration. The parameter design stage involves identifying the control parameters that reduce the system sensitivity to

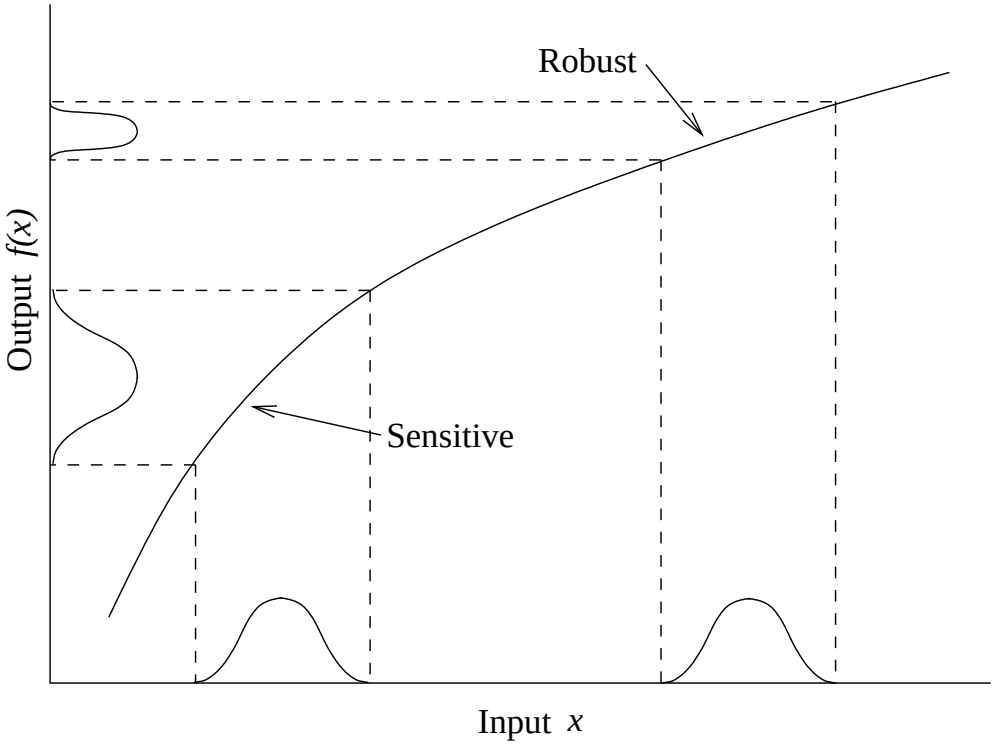


FIGURE 2.4: A comparison of robust and sensitive design (Keane and Nair, 2005)

the noise factors. Tolerance design involves fine tuning the optimal design in the parameter design stage. The parameter design stage is also referred to as robust design.

Taguchi's methods employ a Signal to Noise Ratio (SNR) to define a quantitative measure of performance variability. The SNR is defined as $-10\log_{10}(\text{MSD})$, where MSD is the mean squared deviation. The MSD depends on the problem formulation, for example if $\{y_1, ..., y_n\}$ denote the observed results obtained by varying $\boldsymbol{\xi}$ for a fixed $\mathbf{x}$ and the target for quality is to achieve y close to s, then

$$MSD = \frac{1}{n}\sum_{i=1}^{n}(y_i - s)^2; \tag{2.5}$$

and, if target is to maximize y, then

$$MSD = \frac{1}{n}\sum_{i=1}^{n}{y_i}^{-2}. \tag{2.6}$$

The objective is to maximize the SNR which leads to a reduction in variations. For the efficient use of SNR in robust design the reader is referred to Phadke (1989). To solve the optimization problem inherent in robust design, Taguchi methods use DOE techniques. Orthogonal Arrays (OA) are employed to create a cross product of two separate arrays of control factors (inner array) and noise factors (outer array). This set up is referred to as *product array* and using this set up, for each set of $\mathbf{x}$ in the inner array, the SNR is obtained

using the outer array $\boldsymbol{\xi}$. Finally, the data obtained from the experiment are evaluated using standard analysis of variance methods to obtain the optimum combination of control factors for robustness.

Though Taguchi methods are widely used in industry they have several limitations. Taguchi's method of modeling the signal to noise ratios may lead to non-optimal solutions (Tsui, 1994). Shoemaker *et al.* (1991) showed that the SNR model approach leads to information loss, since it lumps all the data into a single function and hence does not provide information on the effects of individual noise factors. Taguchi methods decide the values of the control and noise factors *apriori* and hence do not permit a sequential optimization process. This may lead to spending important computational effort in areas of no interest to the designer (Trosset, 1996). Another major criticism of the Taguchi method is the use of a cross product array of control and noise factors which leads to a large number of points at which observations must be made (Welch *et al.*, 1990; Box, 1988). For a detailed critical overview of Taguchi methods the reader is referred to Nair (1992); Tsui (1996).

2.4.2 Response Model Based Robust Design

Welch *et al.* (1990) and Box & Jones (1992) suggested the use of combined array and metamodeling techniques to alleviate some of the limitations of Taguchi's method. In the *combined array* approach both the control and noise factors are varied together using DOE techniques. This saves computational effort as compared to the cross product array method. In the combined array (CA) based metamodel strategy for robust design a DOE (Design of Experiments) is performed over the product space $\mathbf{x} \times \boldsymbol{\xi}$. Let us denote the variables in this product space by the vector $\tilde{\mathbf{x}} \in \mathbb{R}^{p+q}$, then the response at the DOE points obtained using the analysis code can be represented as $[\tilde{\mathbf{x}}^{(i)}, y(\tilde{\mathbf{x}}^{(i)})]$. This dataset can be utilized for finding the robust design by minimizing some loss function. The dataset obtained using the CA method can also be used to train a metamodel ($\widehat{y}(\tilde{\mathbf{x}}) = \widehat{y}(\mathbf{x}, \boldsymbol{\xi})$) for predicting the response quantity $y(\tilde{\mathbf{x}})$ (Welch & Sacks, 1991; Myers *et al.*, 1992). Using the metamodel, the robust design problem can be formulated as

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \int_{\mathcal{E}} \widehat{y}(\mathbf{x}, \boldsymbol{\xi}) P(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}. \tag{2.7}$$

Since the metamodel directly models the response over the control and noise factors, it can explain how the control factors dampen the effects of the individual noise factors (Shoemaker *et al.*, 1991). This can be done using Analysis of Variance (ANOVA) techniques (Chen *et al.*,

2005). Trosset *et al.* (2003) proposed a four step procedure to improve the metamodel based approach. The steps are (1) choose initial set of points by varying the design and noise variables $\tilde{\mathbf{x}}$, (2) compute the function $y(\tilde{\mathbf{x}})$ at these points, (3) construct a surrogate objective function $\widehat{y}$, (4) evaluate the integral in equation (2.7) and minimize.

The major limitation of this method is that like Taguchi's method the points in the combined array are decided beforehand. Further, there are no rational guidelines for sequentially updating the surrogate during the optimization process. Hence the designer might end up spending important computational effort in regions of no interest. Another shortcoming is that the quality of the robust designs are critically dependent on the accuracy of the predictions made by the surrogate (Tsui, 1996). Hence without options for further updates, this can be considered as a serious shortcoming for modeling complex interactions between system parameters.

2.4.3 Decision Theory Based Robust Design

Statistical decision theory has also been used for robust design. The objective can be formulated so as to mitigate the effect of the worst-case performance. This strategy is referred to as the minimax method and can be employed to seek a design with the optimal worst case performance (Ben-Tal & Nemirovski, 1997). If $f(\mathbf{x}, \boldsymbol{\xi})$ is the function to be optimized then the minimax formulation can be expressed as

$$\mathbf{x}^* = \arg\min_{\chi} \phi(\mathbf{x}), \quad \text{where} \quad \phi(\mathbf{x}) = \max_{\mathcal{E}} f(\mathbf{x}, \boldsymbol{\xi}), \tag{2.8}$$

where $f(\mathbf{x}, \boldsymbol{\xi})$ is a bounded function. The minimax principle is conservative as it seeks to protect the decision maker against the worst case scenario (Trosset *et al.*, 2003). Huyse & Lewis (2001); Huyse (2001) use a maximum expected value criterion which seeks a design with the best expected performance in the presence of uncertainty. This formulation uses the Bayes's principle and the best design or decision is referred to as the *Bayes' decision*. This can be expressed as

$$\mathbf{x}^* = \arg\min_{\chi} \phi(\mathbf{x}), \quad \text{where} \quad \phi(\mathbf{x}) = \int_{\mathcal{E}} f(\mathbf{x}, \boldsymbol{\xi}) P(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}. \tag{2.9}$$

where $P(\boldsymbol{\xi})$ is the probability distribution function of the uncertain parameters, $\phi(\mathbf{x}^*)$ is the Bayes' risk, and $\mathbf{x}^*$ is the so called Bayes' decision.

All the above mentioned methods require evaluation of complex integrals of the form $\int_{\mathcal{E}} f(\mathbf{x}, \boldsymbol{\xi}) P(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}$; see for example equations (2.7) and (2.9). More often than not, the above

integrals are very difficult or impossible to evaluate analytically. In most cases, techniques such as Monte Carlo Simulation (MCS) are used which can be computationally expensive, particularly when used with high fidelity analysis codes. Later in this chapter, we will discuss the use of efficient surrogate based methods to alleviate this issue. It should also be noted that these methods focus on minimizing the expected value of the performance in the presence of uncertainty. Robust design is inherently a multiobjective problem with the objective to improve the expected value and reduce the variance. In the next subsection, we introduce a multiobjective formulation for robust design to allow for explicit trade-off between the mean and standard deviation of the performance.

2.4.4 Multi-Objective Optimization Based Robust Design

Many researchers have proposed to minimize a single objective to achieve robust design. These methods can be classified into two groups where the aim is to minimize 1) the expectation of the objective function in its neighborhood (Tsutsui & Ghosh, 1997; Tsutsui *et al.*, 1996; Huyse, 2001; Li *et al.*, 2002), 2) the variance of the objective function (Parkinson, 1997). Das (2000) discussed the drawbacks of minimizing the expectation of the objective function. They argued that positive and negative deviations in the function value in the neighborhood of a target may cancel each other and lead to a non-robust optimal design. Minimizing the variability function alone can lead to designs that are robust but not optimal and hence not desirable (Das, 2000; Jin & Branke, 2003). Therefore, it is desirable to optimize both the expectation and variance of the objective function. Robust design can be formulated as a multiobjective problem with the goals of minimizing the mean and variance of the performance. This method allows explicit trade-off between the mean and variability of the performance. The multiobjective formulation can be expressed as to simultaneously minimize both the mean and standard deviation:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\chi}\{\mu(\mathbf{x}), \sigma(\mathbf{x})\} \tag{2.10}$$

where μ and σ are the performance statistics in the presence of noise variables $\boldsymbol{\xi} \in \mathcal{E}$ and are given by

$$\mu(\mathbf{x}) = \int_{\mathcal{E}} \widehat{y}(\mathbf{x}, \boldsymbol{\xi}) P(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \tag{2.11}$$

and

$$\sigma^2(\mathbf{x}) = \int_{\mathcal{E}} (\widehat{y} - \mu)^2 P(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}. \tag{2.12}$$

The objective of this formulation is expressed in figure 2.5 where the abscissa represents vari-

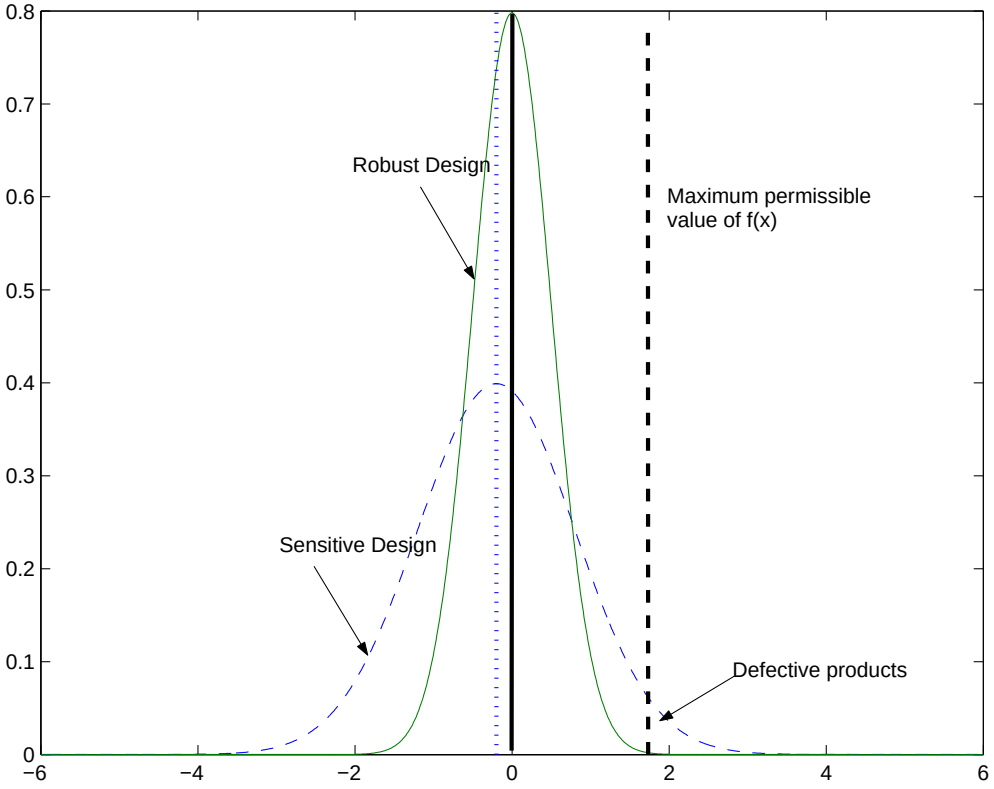


FIGURE 2.5: Robust design using multiobjective formulation (Keane and Nair, 2005)

ations in a performance parameter, the bell-shaped dashed curve represents the probability density function (PDF) of a sensitive design and the dashed line at its center is its mean. The bell-shaped solid curve is the PDF of the robust design and its mean is represented by the solid line. It can be observed from figure 2.5 that robust design has less variability as compared to the sensitive design, whereas the mean performance of the sensitive design is marginally better than the mean performance of robust design. The improvement in the variance has been obtained at the cost of a (small) reduction in the mean performance.

More often than not there is a conflict between optimization of mean and variance and a trade off is required to choose the best designs. The presence of multiple conflicting objectives in a problem leads to a set of optimal solutions, rather than a single solution. Later in the thesis, we will discuss various ways to solve and approach this multiobjective optimization problem.

Chapter 3

Effect of Geometry Variations on Compressor Blades

3.1 Introduction

The behaviour of compressor blades is central to the performance of modern gas turbine engines. These airfoils inevitably exhibit deviation from their intended shape and size. Geometric uncertainty may be introduced for instance, by manufacturing errors, icing or operational wear and damage. The effect of geometric variability, primarily caused by manufacturing errors, on the performance of axial compressors has been studied by Garzon (2003). The probabilistic model used by them was based on Principal-Component Analysis of the blade surface measurements (Garzon & Darmofal, 2003). They employed a full scale Monte Carlo Simulation to understand the effects of manufacturing uncertainties on the aerodynamic and aerothermal properties of high-pressure axial compression systems. Their study suggests that the overall compressor efficiency could deteriorate by approximately 1% due to blade passage effects arising from representative manufacturing variability.

Bragg (1986); Bragg & Khodadoust (1989) studied the aerodynamic performance of airfoils with added ice shapes. They conducted numerical studies to investigate the effect of leading-edge ice shapes and simulated ridge ice shapes on the aerodynamic behaviour of airfoils and wings, over a range of Reynolds numbers and Mach numbers. The effect of ice shape size and location has been studied on the stall condition, lift and drag values by Pan *et al.* (2003). They validated their RANS based studies with experimental data and suggested use of higher fidelity methods, such as Large Eddy Simulations or Detached Eddy Simulations,

for post-stall analysis where the flow tends to be unsteady and separated. In this thesis, we focus on geometric uncertainty due to erosion and manufacturing variations.

In this chapter our aim is to understand the effect of erosion and manufacturing variation on compressor blade performance. A novel parametric CAD representation of compressor blade sections with geometric variations due to erosion and manufacturing variations is presented. The method uses cubic polynomials, splines and Hicks-Henne functions (Hicks & Henne, 1978). This is combined with the geometric modeling and CFD meshing tool PADRAM (Shahpar & Lapworth, 2003) (Parametric Design and Rapid Meshing) to generate hybrid O-H mesh. The multigrid RANS solver HYDRA is then used to carry out CFD simulations at the DOE candidate points. The Spalart Allmaras model is used for modeling turbulence. Since running a full scale Monte Carlo Simulation (Hurtado & Barbat, 1998) would be computationally very expensive, the space filling DOE techniques discussed in the previous chapter are used for the probabilistic analysis. Here LP_τ techniques (Statnikov & Matusov, 2002) are used to create the initial candidate points.

3.2 Aerodynamic Analysis

3.2.1 Background

The aerodynamic performance of a compressor blade may be summarized by the changes in total enthalpy and entropy in the flow across the blade row. Applying the first law of thermodynamics and conservation of angular momentum to a control volume surrounding a stream-tube across the blade passage leads to the Euler turbine equation

$$\Delta h_t = \omega(r_2 v_2 - r_1 v_1), \tag{3.1}$$

where Δh_t denotes the change in total enthalpy; ω, r and v denote wheel speed, radius and circumferential flow velocity, and the subscripts 1 and 2 denote the inlet and exit respectively. Equation (3.1) relates the change in total specific enthalpy (per unit mass) Δh_t to the specific angular momentum across the passage. For small changes in the radial direction across the blade row, equation (3.1) can be expressed as

$$\Delta h_t = \omega r[u_1 tan\beta_1 - u_2 tan(\beta_1 + \vartheta)], \tag{3.2}$$

where u and β are the axial flow velocity and relative flow angle respectively and ϑ represents the flow turning angle, $\vartheta = \beta_2 - \beta_1$.

An appropriate choice for estimating loss in an adiabatic machine is by using entropy generation. The increase in entropy results in a decrease of stagnation pressure rise compared to the isentropic value. Pressure loss coefficient is used by Rolls-Royce to estimate the performance of their blades. The pressure loss coefficient can can be defined as the drop in total pressure from the ideal value at passage exit normalized by the difference between inlet total and static pressure, In this study we use the pressure loss coefficient, which is extracted using PAX post-processor (see figure 3.3), as the criterion for evaluating the performance of the compressor fan blade. Here, we employ CFD to evaluate the pressure loss coefficient. PADRAM is used for grid generation and HYDRA is used as the CFD solver.

3.2.2 Geometry and Grid Generation: PADRAM

The Rolls-Royce propriety code PADRAM is employed for creating the geometry and for grid generation. The basic idea of the PADRAM system is to parametrically change the geometry and rapidly mesh the resultant geometry. This capability is central for solving optimization problems, where the exploration of the design space is dependent on the efficiency of automatically creating and meshing new geometries. A popular approach to manage such problems is to perturb the existing computational grid. However, if the changes in geometry are significant this method may lead to poor mesh quality. In this study, we generate a completely new mesh for the variations in airfoil geometry.

PADRAM is capable of creating 2D, quasi-3D, 3D, single passage and multi-passage meshes suitable for viscous and inviscid CFD calculations. The meshes are based on O-H multi-blocks which are then pre-processed to write input files for the Hydra CFD solver. It makes use of both transfinite interpolation and elliptic grid generation to generate hybrid C-O-H meshes. An orthogonal body fitted O mesh is used to capture the viscous region of the airfoil whilst an H mesh is used near the boundary where stretched cells are required, for example in the wake region. The schematic of a single passage mesh is shown in figure 3.1 Figure 3.2 shows a typical mesh created for CFD analysis using PADRAM. Another benefit of using PADRAM is that it offers the user the flexibility to incorporate new geometry modules. Unlike most existing commercial grid generators, the availability of the source code allows new geometry modeling techniques to be incorporated for modeling erosion and manufacturing variations. We discuss these models later in this chapter.

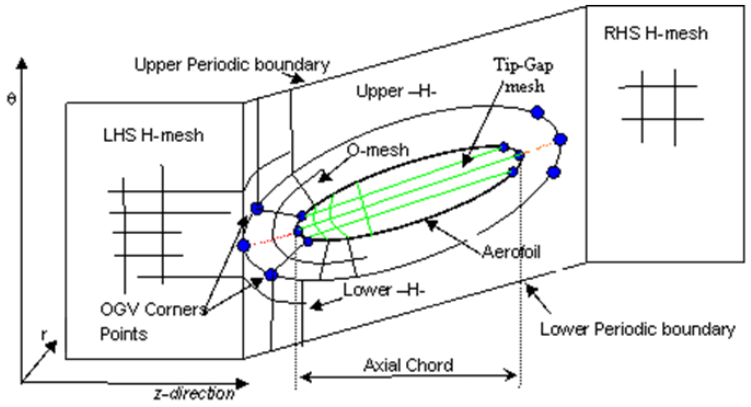


FIGURE 3.1: Schematic of a Single Passage Mesh

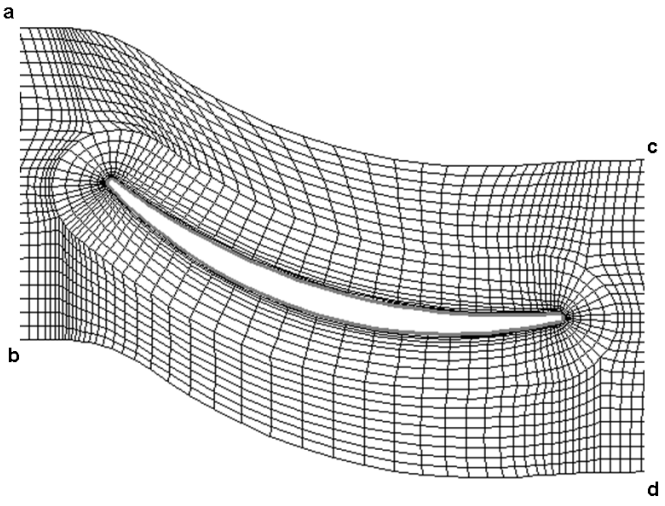


FIGURE 3.2: CFD mesh for compressor blade (O-H mesh)

3.2.3 CFD Analysis: HYDRA

The viscous flow solver HYDRA is used for the CFD simulation. Hydra is a collection of linear, non-linear and adjoint CFD solvers. It was initially developed by researchers at the Rolls-Royce University Technology Centre (UTC) in CFD at the University of Oxford. The code continues to be actively developed by Rolls-Royce and the UTCs for a wide range of aerospace, marine and industrial applications. In this study, the steady non-linear solver is used (Moinier & Giles, 1998). HYDRA uses an edge-based data structure to give the

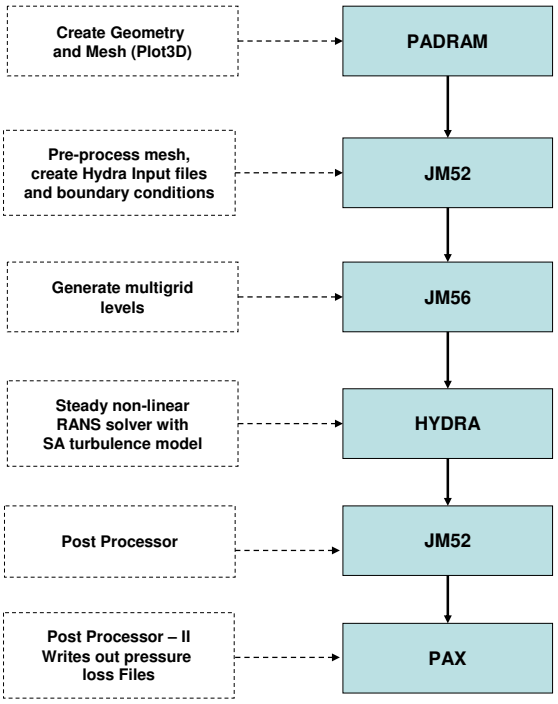


FIGURE 3.3: Flowchart of Hydra routines used in this study

flexibility to run on different mesh types. A pre-processing tool is employed to compute the edge weights and provide the connectivity information. In addition, it uses multigrid adaptation (Moinier *et al.*, 1999) in order to accelerate convergence. In this approach, the multigrid scheme is achieved by edge collapsing, removing certain edges to selectively coarsen the mesh and damping out high frequency error modes.

Figure 3.3 gives an overview of the various elements of the Hydra user suite. PADRAM is employed to generate the geometry with mesh in this study. This mesh is preprocessed by Rolls-Royce proprietary code JM52 to produce a hybrid unstructured mesh. JM52 also

prepares the Hydra input files and is used to define the boundary conditions for the solver. This includes the definition of the boundary surface types as inviscid or viscous walls, inflow or outflow, periodic surfaces, etc. Following this the Rolls-Royce proprietary code JM56 is used to generate multigrid levels for Hydra. It is also employed to compute the edge connectivity and weights for each mesh in the multigrid sequence. The next step is to run HYDRA itself. All these codes can be run in batch mode, which make them ideally suited for sequential optimization studies.

As already noted HYDRA solves the steady Reynolds Averaged Navier-Stokes equations with the Spalart-Allmaras turbulence model. The non-linear solver employs an explicit time marching scheme based on the five step *Runge-Kutta* stepping procedure. To reduce computational effort, the multigrid approach and parallel processing based on the master slave system are adopted. The CFD domain used here is shown in figure 3.2 by *abdc*, where boundary *ab* is the inlet and boundary *cd* is the exit. The inlet boundary conditions for the CFD analysis are Total temperature = 290 Kelvin, Total Pressure = 63400 Pascal, Whirl Angle = -37.28 Degrees and the outlet boundary condition is Static Pressure = 52000 Pascal. An initial uniform flow condition with Density = 0.7675 kg/m^3, Velocity = 0 m/sec and Pressure = 66932 Pascal is considered. These input conditions are provided by Rolls-Royce and are representative of what they employ for their analysis. A four level multigrid is used for the present simulations. The converged CFD solution is used to calculate the pressure loss coefficient across the blade geometry.

3.3 Analysis of Eroded Compressor Blade

During operation, compressor fan blades are exposed to a number of erosion processes (Metwally *et al.*, 1995), which can lead to reduction of the blade chord, alteration in the shape and increase in the surface roughness (Hamed *et al.*, 1998). This is critical to the blade performance and can lead to degraded overall engine efficiency. Roberts (1984) has shown that geometric variability in the form of leading edge erosion in compressor airfoils may account for an increase of 3% or more on the thrust specific fuel consumption. However, replacing the eroded compressor fan blades is expensive. Hence, it is desirable to understand the effect of erosion on the blade performance. In this section, we propose a method to generate simplified models of eroded blade geometries. The aim is to use these models to understand the effect of a range of erosion patterns on the aerodynamic performance of the blade.

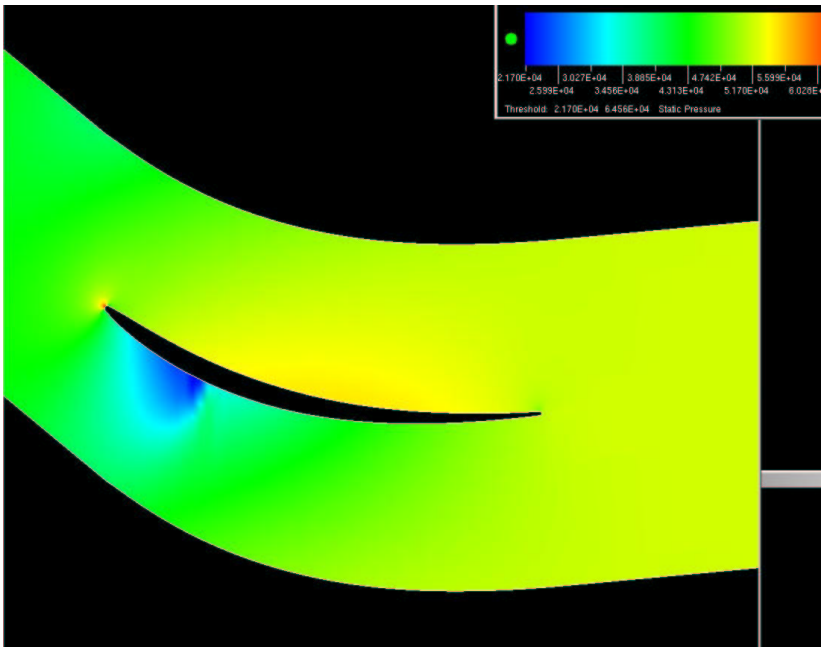


FIGURE 3.4: Static pressure distribution for Baseline geometry

3.3.1 Geometry Modeling of Eroded Blades

Erosion leads to blade surface deterioration and causes a depression in the original airfoil. Hence for modeling eroded geometries, an approach that can model local dents in the original airfoil shape is required. Hicks-Henne functions (Hicks & Henne, 1978) provide a flexible tool to model local variation in the form of bumps. Erosion patterns observed in compressor fan blades can be very complex. A combination of piece-wise cubic polynomial and Hicks-Henne function is used here to create a simplified model of such erosion patterns. The eroded compressor fan blade section is parametrized in terms of the location, depth and the width of the eroded section. The Hicks-Henne function can be expressed as:

$$b(x) = A\left[sin\left(\pi x^{\frac{log0.5}{logt_1}}\right)\right]^{t_2}, 0 \leq x \leq 1. \tag{3.3}$$

Here, A is the maximum bump magnitude, t_1 is the position of the maximum of the bump, and t_2 controls the shape of the bump. For modelling eroded geometries we keep the values of $t_1 = 0.5$ and $t_2 = 2$ fixed. This implies that the erosion is symmetric about its maxima and behaves as $\sin^2$. We parametrize the eroded blade in terms of the location of erosion on the blade, the width and the depth of the erosion on the blade geometry. All these quantities are denoted as percentage of the chord length. This provides us with a parametric model of eroded blades with three variables - location (l), width (w) and depth (A). Once the depth of erosion is fixed, we use equation (3.3) to get the erosion shape. This shape is then

superimposed on the baseline shape at the given location with the prescribed depth. The details of the implementation of this model is given in Appendix A. Figure 3.5 shows the parametrized model.

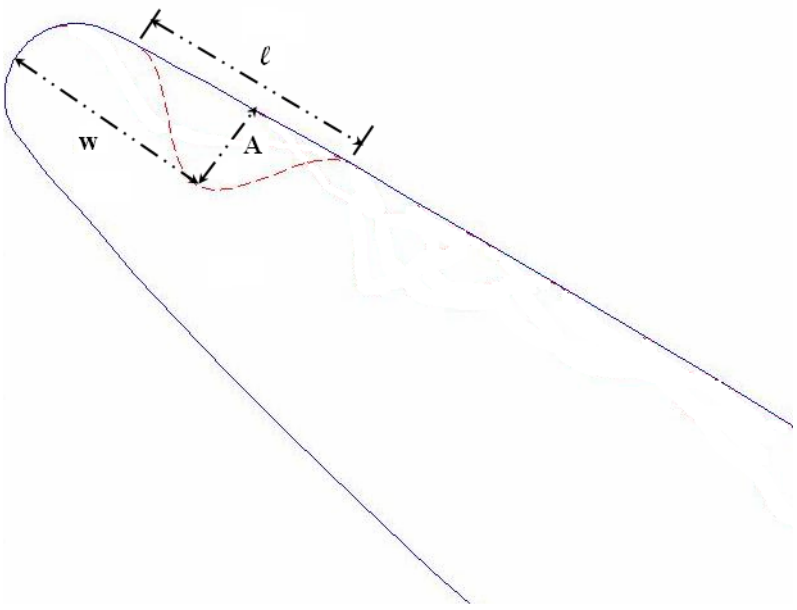


FIGURE 3.5: Parametric model of eroded compressor blade

It has to be mentioned here that this model is not a true representation of the complex eroded shapes observed in practice. The proposed model is a simplified representation of erosion due to several limitations. The major limitations is lack of data of eroded blade geometries. In the absence of relevant data, this method was limited to visual verification with a few available eroded geometries. It is to be noted here that though our model is capable of modeling erosion on both pressure and suction surfaces, we only model erosion on the pressure surface. It has been observed that due to the rotatary motion and placement of the blades, erosion is primarily observed on the pressure surface. As mentioned earlier, we employed PADRAM for automatic and rapid grid generation. Though a new mesh was generated for each eroded geometry, complex erosion shapes could lead to poor mesh quality or failure of the grid generation process due to negative areas which more often than not may need human interaction. However, for probabilistic analysis such interactions can be computationally prohibitive. So, the grid generation process had to be conducted in batch mode and this imposes additional constraints on the erosion model. Finally, as we employ a steady non-linear solver HYDRA, it was not advisable to analyze complex shapes with high curvature changes (for example corner wedges), since such geometries may cause unsteady

flow behaviour. Figure 3.6 shows some typical eroded blade shapes that can be obtained using this method.

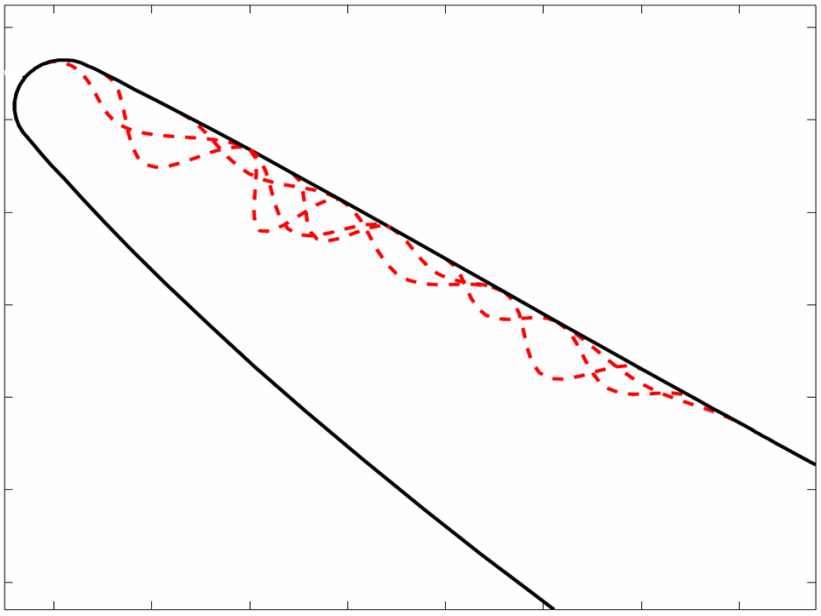


FIGURE 3.6: Typical eroded blade patterns obtained using the parametric model

3.3.2 CFD Analysis of Eroded Blades

The parametric erosion model discussed above is incorporated in PADRAM. The grid generator in PADRAM is then employed for automatic mesh generation. The HYDRA computational grid was made fine enough to give y^+ values between 20 and 50 (normally recommended for Spalart-Allmaras turbulence model). This required a mesh size of more than 26000 cells in two dimensions. This would ensure that a high quality solution is obtained using the Spalart-Allmaras transport equation turbulence model as boundary layer is modeled accurately for calculating the wall shear stress. The CFD analysis was performed on the eroded geometry for two grids of mesh size 26,000 and 32,000 respectively to understand the effect of mesh size on the CFD results. Table 3.1 shows the comparison of the prediction of pressure loss coefficient using the two grids. The fine mesh calculations are not very different from the coarse mesh (difference of 0.114%). Hense the coarse mesh was chosen for the present study. Figure 3.7 shows a zoomed in view of the selected mesh. This mesh was preprocessed using JM52 with the specified boundary conditions. JM56 was used for generating four multigrid levels. Finally, the HYDRA CFD solver is run to obtain the pressure loss coefficient. The CFD analysis takes approximately 20 minutes on a Intel(R) Xeon(TM) 3.06GHz processor.

Mesh	No. of Cells	Pressure Loss Coeff
Coarse Mesh	26,344	2.3924
Fine Mesh	32,224	2.3897

TABLE 3.1: Comparison of pressure loss coefficient predicted on coarse and fine grid.

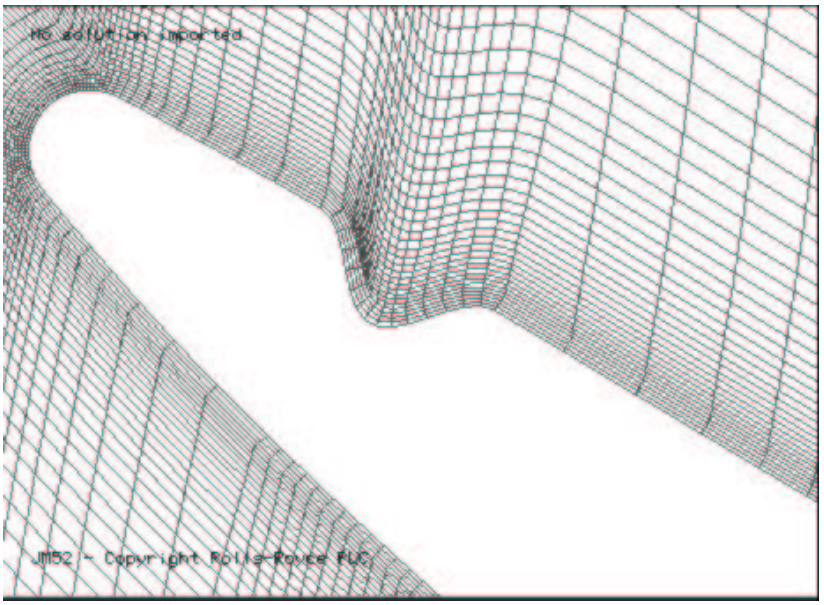


FIGURE 3.7: Zoomed in view of the mesh around the eroded blade geometry

Figure 3.8 shows the static pressure distribution for an eroded blade geometry. Next we show the zoomed in plots for x velocity u, relative y velocity v, relative angular velocity w and static pressure for a typical CFD solution on the eroded blade in figure 3.9.

3.3.3 Probabilistic Analysis of Eroded Blades

We use the analysis system described in the previous subsection to conduct probabilistic performance analysis of the compressor blade in the presence of erosion. Figure 3.10 outlines the sequence of steps involved. In the first step we define the range of the erosion parameters - location (l), width (w) and depth (A). In the absence of measurement data, the range for each parameter was decided by visual inspection and the limitations imposed by the grid generator and CFD solver. The space filling DOE technique LP_τ is employed for providing a set of candidate points, assuming a uniform distribution of the erosion parameters over the selected range. Table 3.2 gives the range of values for each erosion parameter. This dataset

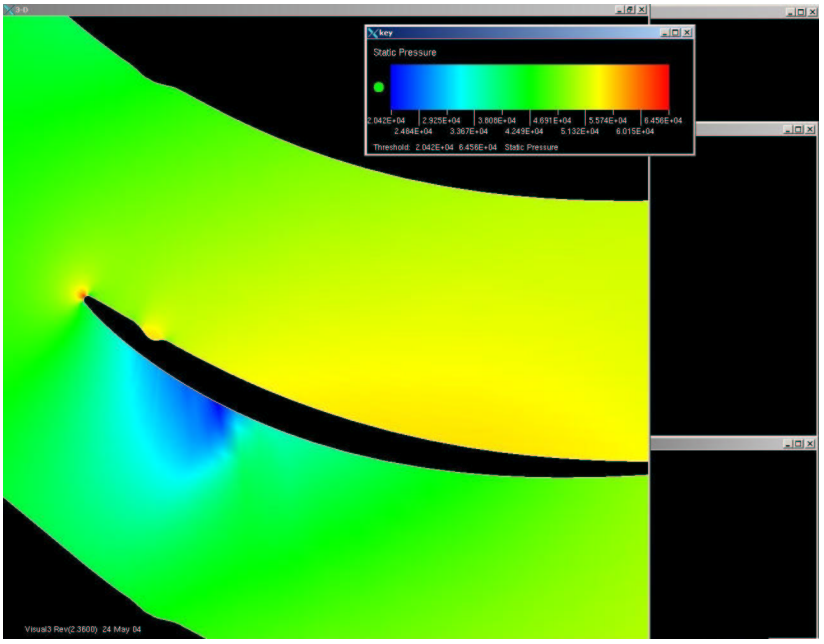


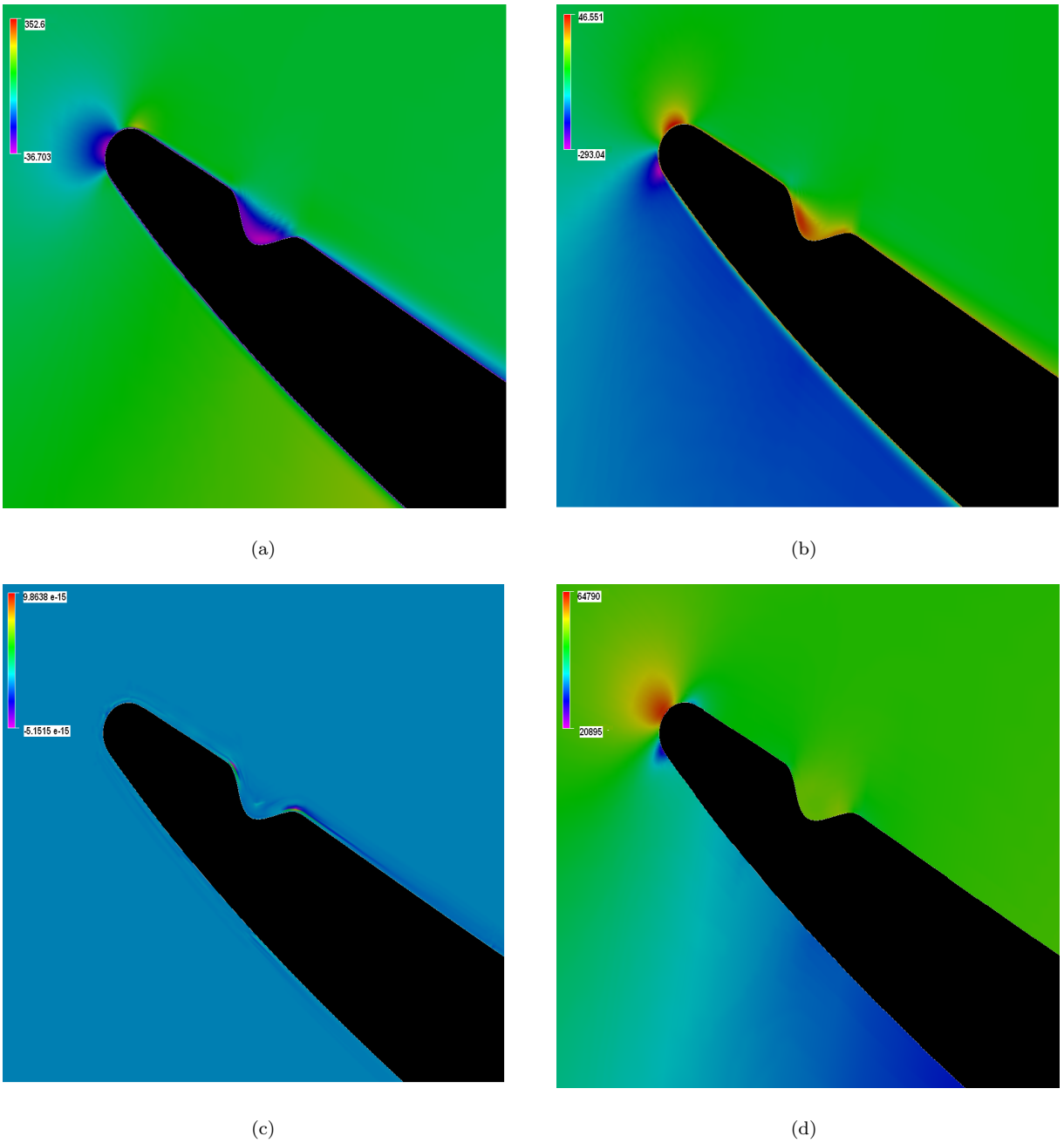
FIGURE 3.8: Static pressure distribution for an eroded geometry

Erosion Parameter	Lower Bound	Upper Bound
A	0%	3%
l	3%	40%
w	2%	6%

TABLE 3.2: Upper and lower bounds of erosion parameters in terms of percentage of chord.

is fed into PADRAM to create geometries and subsequent CFD meshes.

A 100 point DOE is run over this parameter range of which 11 analysis failed due to problems in grid generation or divergence of CFD solutions. A common practice is to select at least 10 points for each variable to understand their effect on the performance. We are also limited by the high computational cost involved in selecting a huge sample for analysis and hence 89 points for 3 variables were selected. Figure 3.11 shows the normalized scatter in performance of the eroded blade shapes at the selected points. It can be observed that the pressure loss for most of the eroded geometries has deteriorated. Up to 5% degradation in pressure loss coefficient can be observed. This emphasizes the need to consider the effect of erosion on blades during the design stage. In the next chapter, we propose a methodology for robust design of compressor blades in the presence of such erosion.



(a) (b)

(c) (d)

FIGURE 3.9: Zoomed in plots near the erosion for the compressor blade using CFD for (a) u velocity (b) relative y velocity v (c) relative angular velocity w (d) static pressure

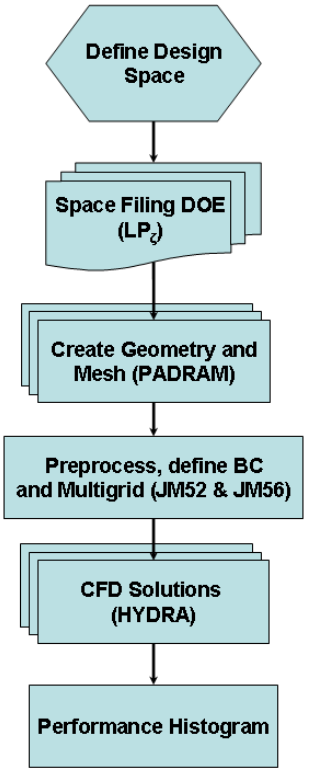


FIGURE 3.10: Flowchart for probabilistic analysis of compressor blade

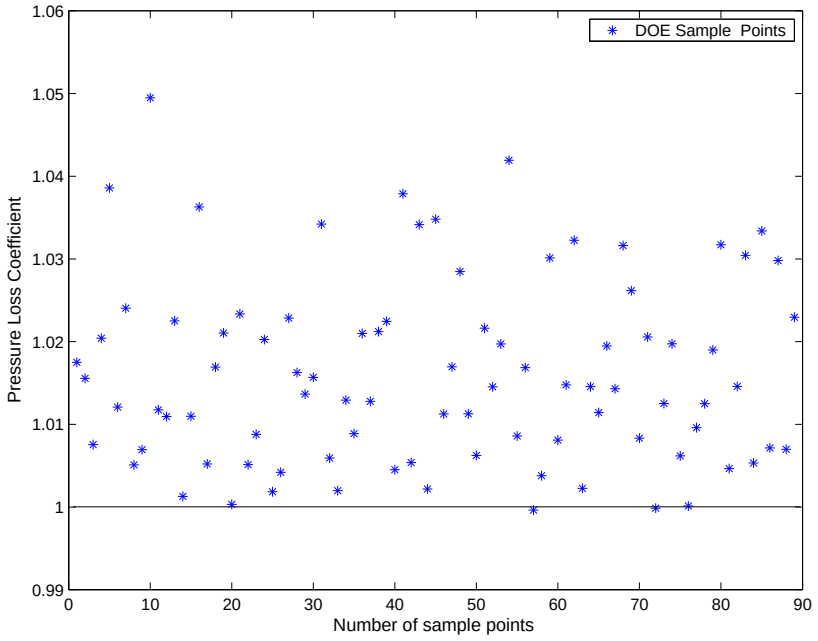


FIGURE 3.11: Scatter observed in performance due to erosion of compressor blade. The horizontal line represents the nominal value.

3.4 Analysis of Blades with Manufacturing Variations

Manufacturing variations can also lead to loss in quality due to performance degradation, non-conformance to specifications, high cost of redesign or scrap and failure. Due to such variations, the manufactured product may differ from the proposed ideal design and its performance may be sensitive to manufacturing uncertainty. To assure that a product meets the design specification, manufacturers select tight tolerances or a higher precision manufacturing process, which can lead to considerable increase in manufacturing cost. Hence, it is important to consider the effect of manufacturing variations on a product during the design phase and select a design that is *robust*.

3.4.1 Process Capability Data

More often than not design engineers do not have enough information about the downstream manufacturing process capability (Tata & Thornton, 1999). In recent years there have been many attempts to provide the designer with process capability data (Kotz & Johnson, 2002). Process Capability is the expected probability distribution of the manufactured products using a manufacturing process in ideal conditions (Bothe, 2001). Numerous methods for measuring process capability exist in the manufacturing literature; for example see Chase & Parkinson (1991); Boyles (1991); Kotz & Lovelace (1998); Bothe (2001). The most common quantitative definition of process capability is the process spread, or $6\sigma_p$. The performance of a manufacturing process can be measured using the dimensions of parts produced by the process. In the limiting case, the process is assumed to have a normal distribution with standard deviation σ. In practice, the observed variations in the process will be greater than that predicted by process capability due to temperature variations, tool wear, material properties etc, see figure 3.12. Moreover they are never strictly Gaussian in nature.

Probabilistic analysis have shown substantial degradation in performance of compressor blades in presence of manufacturing variations, errors and tolerances (Garzon, 2003; Lamb, 2005). These studies re-emphasize the need to consider robustness of compressor blade performance against manufacturing uncertainty in the design stage. The most common processes used for manufacturing compressor blades are flank milling and point milling and their process capability data can be readily made available to the designers. Next we discuss how to use such data for simulating manufacturing process variability and to understand its effect on aerodynamic performance of compressor blades.

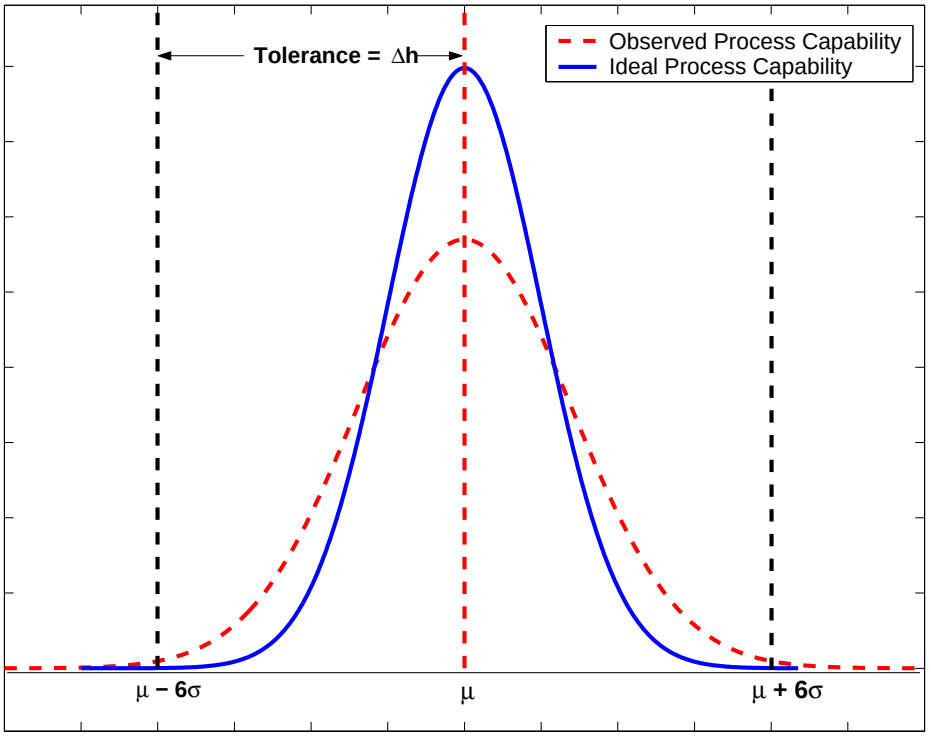


FIGURE 3.12: Ideal and Observed Process Capability

3.4.2 Geometry Modeling of Manufacturing Variations

Once a manufacturing process is fixed (flank or point milling) and the process capability is known, manufacturing variations can be modeled. Figure 3.13 shows a typical manufacturing uncertainty band around the nominal compressor blade. The task at hand is to simulate a manufacturing process such that the observed manufactured blades have a normal distribution with $6\sigma_p = \Delta h$. To model manufacturing uncertainty precisely, a parametric geometry model is sought which can describe geometry variations in the given tolerance band around the nominal geometry. These could be variations in chord, camber and thickness. Here we present an efficient method using a combination of Hick-Hennes functions and splines for modeling manufacturing variations. To parametrize the blade section geometry we use a linear combination of Hicks-Henne functions super-imposed on a baseline shape. For the problem under consideration, we have used 10 Hicks-Henne functions to parametrize the compressor fan blade. The Hicks-Henne shape functions can also be expressed as

$$b_i(x) = sin^2(\pi x^{m_i}) \ , \quad m_i = ln(0.5)/ln(x_{M_i}) \quad i = 1, 2, ...n \tag{3.4}$$

where x is the normalized chord-wise coordinate starting from the trailing edge encompassing the whole airfoil and back to the trailing edge. $(0 \leq x \leq 1)$, x_{M_i} are preselected values

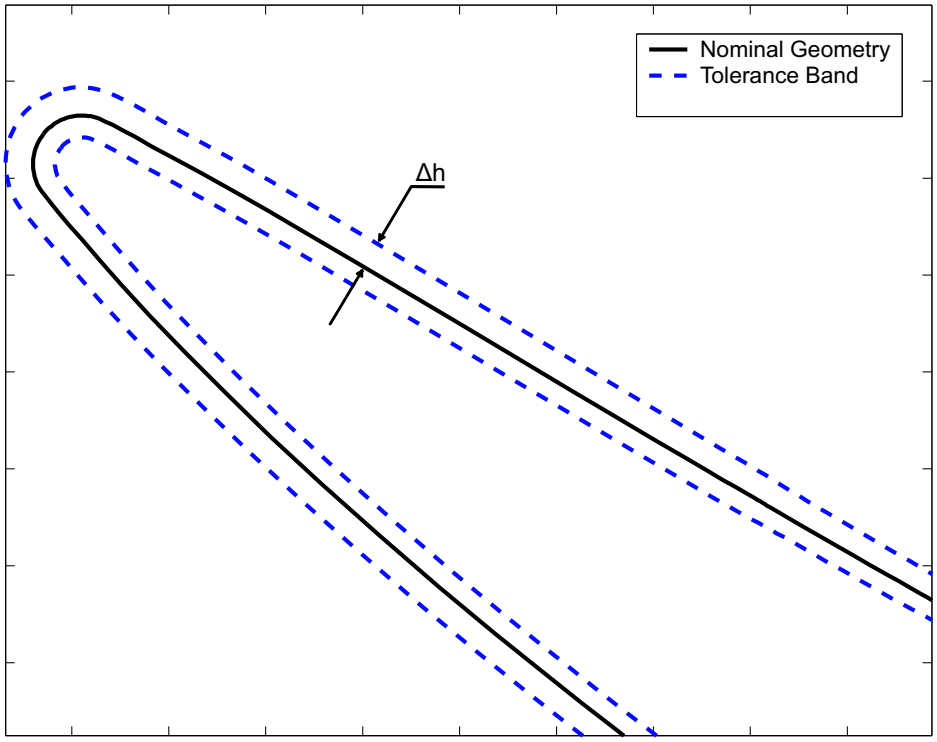


FIGURE 3.13: Manufacturing uncertainty band in compressor blade

corresponding to the location of the maxima and n is the number of Hicks-Henne functions used.

In the present study, the locations of x_{M_i} for $i = 1, 2, ..., 10$ are chosen in a manner to ensure clustering near the leading edge. This ensures more points where the curvature is higher and thus more variety in shapes near the leading edge. Each Hicks-Henne function is multiplied by a weight factor $W_i \mid i = 1, 2, ..., 10$, which controls the amplitude (maxima) of the respective functions. These weight factors give us 10 design variables to parametrize manufacturing uncertainty in the blade geometry. Note that though each W_i would have a effect on the whole shape of the blade, its maximum influence would be near the region corresponding to the location of its respective maxima (x_{M_i}). Figure 3.14 shows the location of the maximum influence point due to each variable on the blade shape. Furthermore, two control points required are chosen to be the two points at the cusp on the trailing edge. This ensures that we also achieve deviations in the chord length of the airfoil. Some typical patterns of blade geometries obtained using the above method is shown in figure 3.15.

Figure 3.16 shows zoomed in view near the leading and trailing edge for two typical shapes obtained by using the above mentioned technique. Note that we obtain changes in chord length, which is not possible in the existing Hick-Henne based techniques. This is

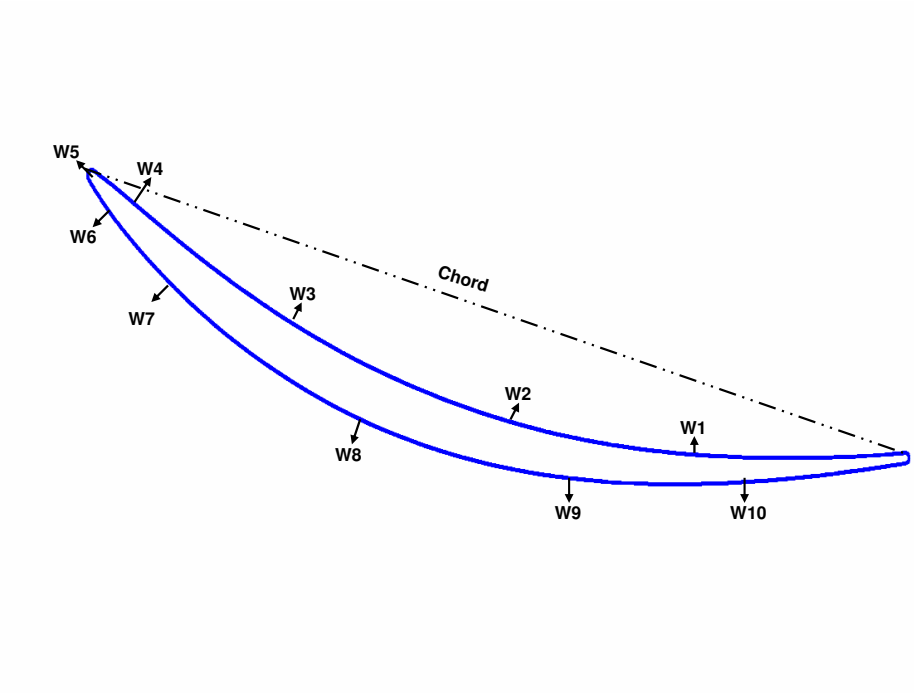


FIGURE 3.14: Maximum impact location due to each variable on the blade shape

achieved by having both the control points at the cusp of the trailing edge. The details of the implementation of this technique is fiven in Appendix B.

3.4.3 Probabilistic Analysis in Presence of Manufacturing Uncertainty

Once the manufacturing geometry simulation model is developed we combine it with PADRAM and the HYDRA suite of codes. The process defined in the flowchart given in figure 3.10 is executed. An LP_τ based DOE of 150 points (shapes) is executed to understand the effect of manufacturing variations on the aerodynamic performance of the compressor blade. The choice of 150 points is again based on the fact that we are limited by the high computational cost involved in running the CFD solver and that we need at least 10 points for each design variable.

The CFD analysis for all these blade shapes is conducted using HYDRA. Figure 3.17 shows the normalized scatter in the aerodynamic performance due to manufacturing uncertainty.

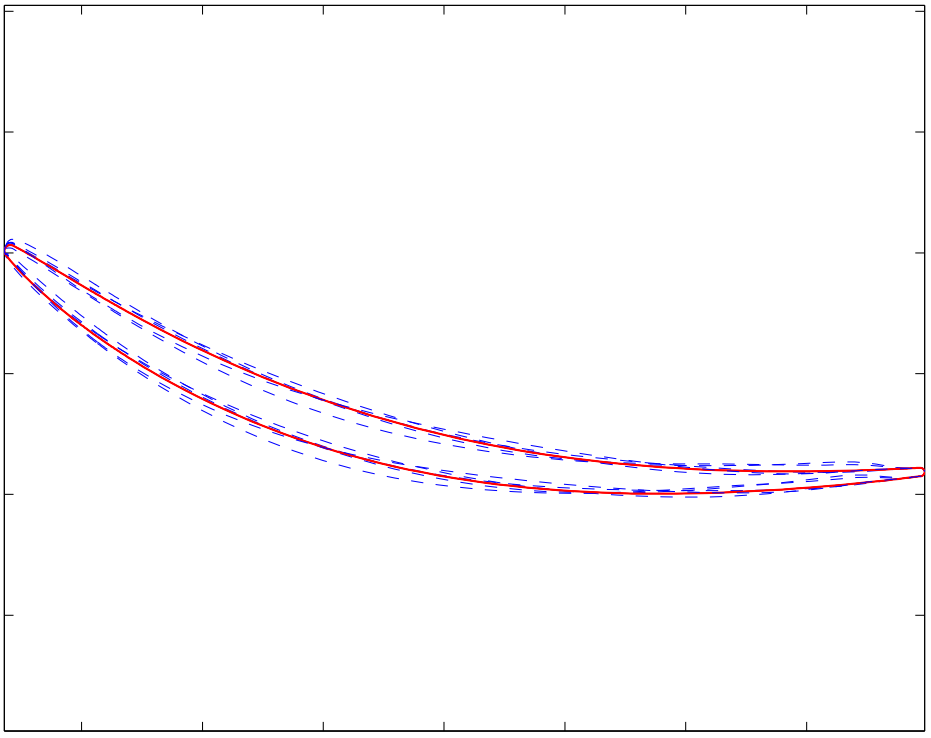


FIGURE 3.15: Some typical manufacturing variations shape s obtained using the parametric model

It can be observed that for the selected value of process capability (σ_p) we observe upto 6% deterioration in the pressure loss coefficient. Later in the thesis we will discuss methods to alleviate this issue. It is also interesting to observe that some blades show improvement in performance.

3.5 Summary

In this chapter we introduced the geometric and computational model which is central to analysis of blades throughout this thesis. Novel methods to model erosion and manufacturing uncertainty was also presented. A parametric representation of these geometric models were developed which could be exploited during uncertainty analysis and design search. We also discussed the limitations of our geometric models. In the case of eroded blades, these limitations are imposed due to lack of data. The steady state CFD solver and the grid generator also limits us to use smooth geometries. On the other hand the manufacturing variations model is more realistic as the variations due to manufacturing uncertainty are smooth and hence easier to model. We assumed a normal distribution for the manufacturing variations however, in practice these uncertainty are never strictly normal.

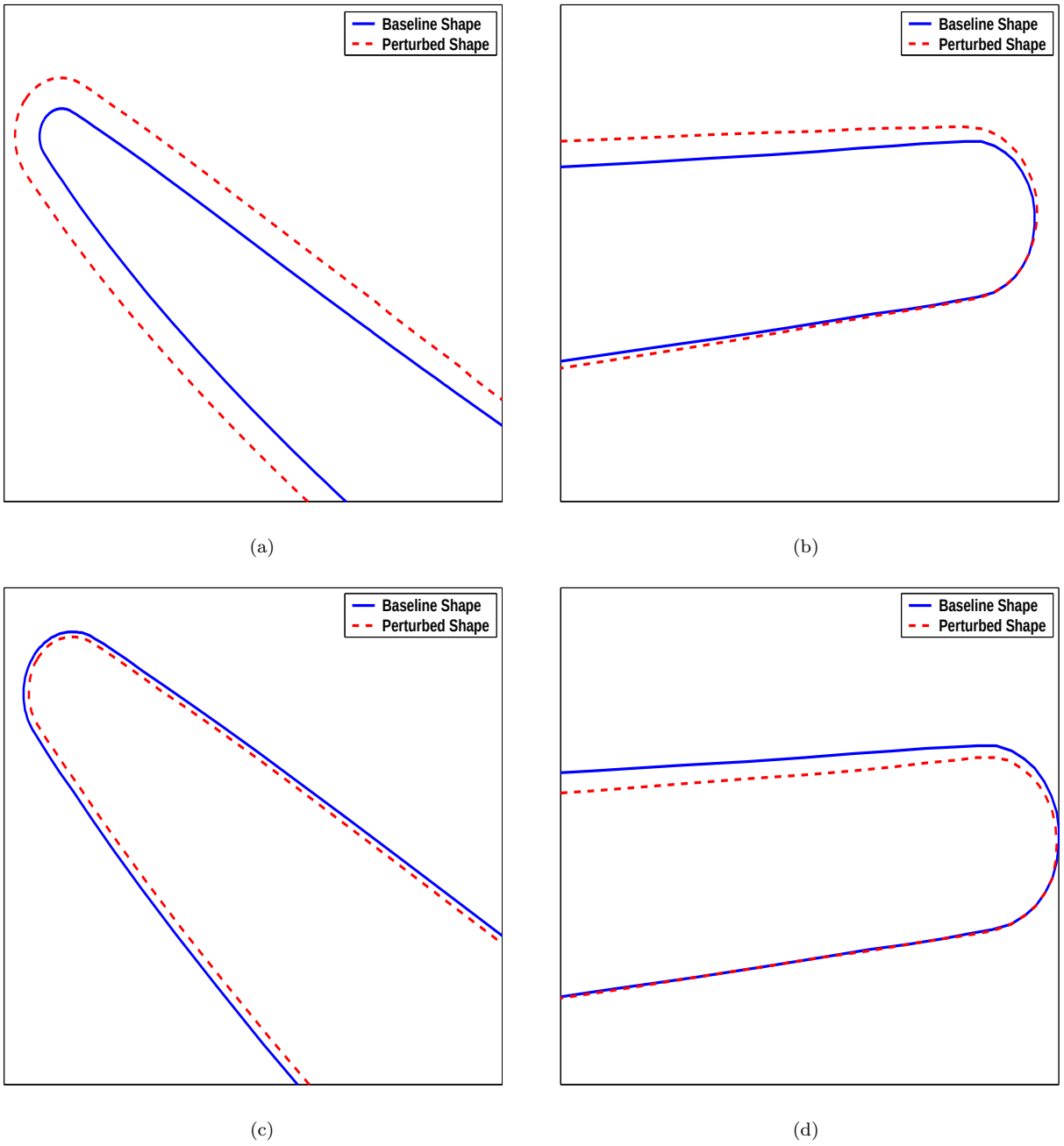


FIGURE 3.16: Zoomed in plots near the leading and trailing edge using the manufacturing uncertainty parametrization technique (a) leading edge for shape1 (b) trailing edge for shape 1 (c) leading edge for shape 2 (d) trailing edge for shape 2

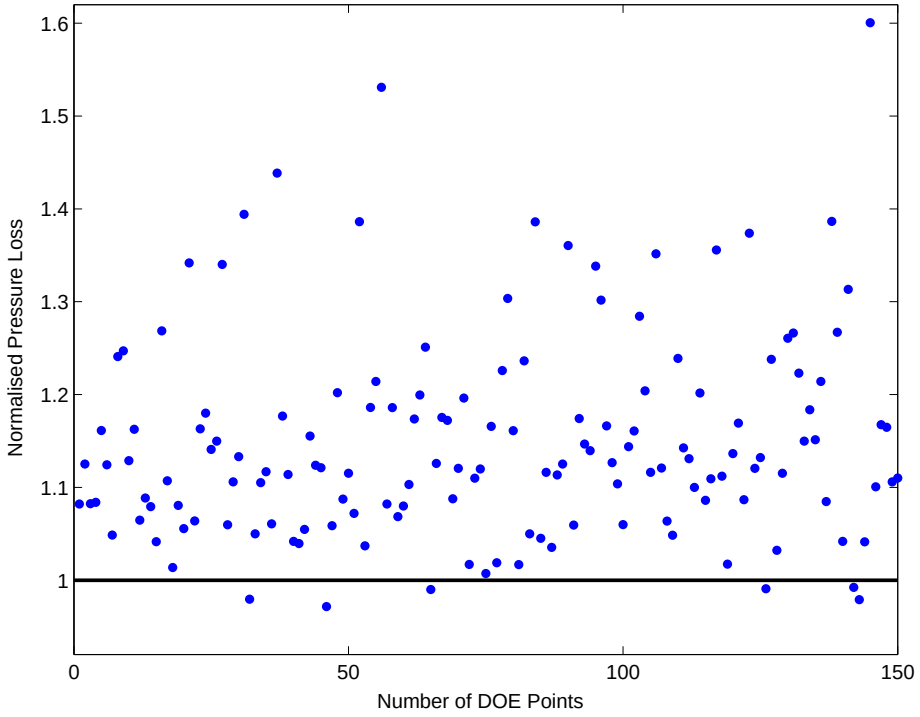


FIGURE 3.17: Scatter in pressure loss coefficient of compressor blade due to manufacturing uncertainty

Probabilistic studies on performance analysis of eroded blades and blades with manufacturing uncertainty were presented. We observed that blade performance can deteriorate significantly in the presence of uncertainty. This reemphasized the case for robust design against geometric uncertainty during blade design. The robust design methods require evaluation of the statistics of the performance over the *noise space* which can be computationally expensive. The aim of the remaining part of the thesis is to propose and implement robust design methods to alleviate the effect of geometric uncertainties on the aerodynamic performance of compressor blades. The focus would be to make these analysis computationally efficient and effective.

Chapter 4

Robust Design of Compressor Blades against Erosion

4.1 Introduction

In recent years, there has been a resurgent interest in computational analysis and design methods that rationally accommodate uncertainty arising from sources such as varying operating conditions, inaccurate system parameters etc. (Zang *et al.*, 2002). Aircraft engines can be subjected to severe conditions during operation in flights all around the world. Examination of compressor blades of gas turbine engines after operations have shown that these blades undergo critical shape changes along the surface of the blades due to erosion. This erosion is primarily caused due to ingestion of foreign particles which then impinge and damage the stationary and rotating blade surfaces. Over time, the erosion process can severely alter the blade surface, which in turn can significantly deteriorate the overall performance of the engine. In this chapter, we propose a method to design compressor blades that are less sensitive in aerodynamic performance to variations in shape due to the above-mentioned erosion processes.

The proposed approach combines a multiobjective genetic algorithm with geometry modelling methods and high-fidelity computational fluid dynamics to arrive at robust designs. Direct application of high fidelity analysis solvers for robust design studies can be computationally expensive. We employ surrogate models based on Gaussian Stochastic Process Emulators to achieve our objectives on a limited computational budget. A detailed Gaussian emulator based probabilistic performance analysis is carried out to quantify and understand

the effect of erosion. This is followed by numerical studies on robust design of a typical compressor fan blade section to illustrate the proposed methodology. The performance of a selected robust optimal solution on the Pareto front is compared to a deterministic optimal solution to demonstrate that significant improvements in the mean shift and variance can be achieved.

This chapter is organized as follows, Section 4.2 describes the Gaussian Stochastic Process Model. The details of surrogate model based probabilistic performance analysis to understand the effect of erosion on the aerodynamic performance of the blade are presented in section 4.3. In section 4.4 we present the robust design methodology and define the erosion problem. A brief overview of the Non-dominated Sorting Genetic Algorithm (NSGA-II) used for solving the multiobjective optimization problem is presented in section 4.5. Finally, in section 4.6 we present the numerical studies which are followed by a summary of the chapter.

4.2 Surrogate Model

The computational cost associated with high-fidelity simulation generally rules out the direct application of MCS based techniques for probabilistic analysis. Surrogate modelling uses the basic idea of analyzing an initial set of design points to generate data which can be used to construct computationally cheap approximations of the original high-fidelity model. A brief overview of surrogate models was given earlier in section 2.2.3. In this study, we employ a Gaussian Stochastic Process model, a non-parametric model based on combination of a local and global models. It has been shown that this model offers sufficient flexibility for modelling smooth as well as nonlinear functions (Nair *et al.*, 2001). One key advantage of this approach is that the user can get an estimate of the prediction error, which can be exploited in optimization procedures to sequentially update the surrogate model and hence improve its predictive capability (Keane, 2003a; Keane & Nair, 2005; Ong *et al.*, 2003). Next we present this model in detail.

4.2.1 Gaussian Stochastic Process Modelling

In general, Gaussian Stochastic Process modelling involves the following steps: (1) data generation, (2) model structure selection, (3) parameter estimation and (4) model validation. Let $\mathcal{D}_n = (\boldsymbol{\xi}^{(i)}, y(\boldsymbol{\xi}^{(i)})|i = \ 1...n)$ denote the initial dataset which comprises the candidate points obtained using DOE techniques and the respective observed output responses from

a computational analysis model. Our objective is to formulate a method that utilizes the information in the observed dataset $\mathcal{D}_n$ and subsequently provide us a computationally less expensive surrogate to replace the high fidelity analysis model (CFD, FEA) for uncertainty analysis. The underlying functional relationship between the input (noise factors) and the output is denoted by the function $Y(\boldsymbol{\xi})$. The goal is to model this function using the observed dataset.

In the next step, we assume that the output response $Y(\boldsymbol{\xi})$ is a Gaussian process (Cressie, 1998; Neal, 1996). O'Hagan *et al.* (1999) have extensively employed Gaussian process emulators to model complex functions . A Gaussian process can be thought of as the generalisation of a Gaussian distribution over a finite vector space to an infinite dimension function space (Mackay, 2003; Girard, 2004). It is a random field which is fully characterized by its mean and covariance function; i.e. $Y(\boldsymbol{\xi}) \sim \mathcal{GP}(m, \Gamma)$ where, m is the mean (expectation) function and, Γ is the covariance function. m is a function of $\boldsymbol{\xi}$ ($m = \langle Y(\boldsymbol{\xi})\rangle$), however, m is often taken as the zero function for the sake of convenience. It has been shown that this choice does not lead to any loss of generality and can be employed to model complex functions (Girard (2004)). The covariance function should ideally depend on the observed dataset $\mathcal{D}_n$ and we shall discuss this issue later in this section. For the time being, we will proceed with our discussion assuming that $m = 0$ and an appropriate covariance function has been specified.

The Gaussian process prior implies that, for any given set of inputs $(\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, ..., \boldsymbol{\xi}^{(n)})$, the corresponding random variables $\mathcal{Y}_n = [Y(\boldsymbol{\xi}^{(1)}), Y(\boldsymbol{\xi}^{(2)}), ..., Y(\boldsymbol{\xi}^{(n)})]^T$ have an n-dimensional normal distribution. The randomness can be thought to arise from the fact that one cannot afford to evaluate $y(\boldsymbol{\xi})$ at every $\boldsymbol{\xi} \in \mathcal{E}$. This is usually the case when the output is expensive to evaluate using high fidelity computational models like CFD. This Gaussian process model can be symbolically expressed as $\mathcal{Y}_n \sim \mathcal{N}(0, \boldsymbol{\Gamma})$. Here, $\boldsymbol{\Gamma}$ is an $n \times n$ matrix of covariances between all pairs of points. The ijth element of the covariance matrix is given by $\Gamma(\boldsymbol{\xi}^{(i)}, \boldsymbol{\xi}^{(j)}) = \mathrm{Cov}(\boldsymbol{\xi}^{(i)}, \boldsymbol{\xi}^{(j)})$ which expresses the covariance between the values of $Y(\boldsymbol{\xi})$ at the points $\boldsymbol{\xi}^{(i)}$ and $\boldsymbol{\xi}^{(j)}$. The covariance matrix $\boldsymbol{\Gamma}_n$ can be written in expanded form as

$$\boldsymbol{\Gamma}_n = \begin{bmatrix} \Gamma(\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(1)}) & \cdots & \Gamma(\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(n)}) \\ \vdots & \vdots & \vdots \\ \cdots & \Gamma(\boldsymbol{\xi}^{(i)}, \boldsymbol{\xi}^{(j)}) & \cdots \\ \vdots & \vdots & \vdots \\ \Gamma(\boldsymbol{\xi}^{(n)}, \boldsymbol{\xi}^{(1)}) & \cdots & \Gamma(\boldsymbol{\xi}^{(n)}, \boldsymbol{\xi}^{(n)}) \end{bmatrix} \tag{4.1}$$

Next we use the Gaussian process prior for inferencing at a new point given the dataset $\mathcal{D}_n$

4.2.2 Predicting at a new point

Once the joint probability distribution function $\mathcal{N}(0, \mathbf{\Gamma})$ is obtained from the specified covariance function, the information available in the existing dataset can be used for predicting at any new data point using *Bayesian Inference*. The conditional probability for $Y(\boldsymbol{\xi}^*)$ at any new noise factor set $\boldsymbol{\xi}^*$ given the observed data $\mathcal{D}_n$ is

$$P(Y(\boldsymbol{\xi}^*)|\mathcal{D}_n) = \frac{P(Y(\boldsymbol{\xi}^*), \mathcal{D}_n)}{P(\mathcal{D}_n)}. \tag{4.2}$$

Having assumed a Gaussian process prior over the response outputs, the output at any new point and the outputs in $\mathcal{D}_n$ are jointly Gaussian. Hence, the conditional probability $P(Y(\boldsymbol{\xi}^*)|\mathcal{D}_n)$ given by equation (4.2) is also Gaussian. The posterior distribution is given by

$$P(Y(\boldsymbol{\xi}^*)|\mathcal{D}_n) \propto \exp\left[-\frac{1}{2}\mathcal{Y}_{n+1}\mathbf{\Gamma}_{n+1}^{-1}\mathcal{Y}_{n+1}^T\right], \tag{4.3}$$

where $\mathbf{\Gamma}_{n+1}$ is the $(n+1)\times(n+1)$ covariance matrix for the vector $\mathcal{Y}_{n+1} = [Y(\boldsymbol{\xi}^{(1)}), ..., Y(\boldsymbol{\xi}^{(n)}), Y(\boldsymbol{\xi}^*)]^T$. We can evaluate the expected value and posterior variance of $Y(\boldsymbol{\xi}^*)$ by inverting the $\mathbf{\Gamma}_{n+1}$ matrix, which can be expressed in expanded form as

$$\begin{bmatrix} [\mathbf{\Gamma}_n] & [\mathbf{k}] \\ [\mathbf{k}^T] & [\kappa] \end{bmatrix}, \tag{4.4}$$

where, $\kappa = \Gamma(\boldsymbol{\xi}^*, \boldsymbol{\xi}^*)$ and $[\mathbf{k}] = [\Gamma(\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^*), \Gamma(\boldsymbol{\xi}^{(2)}, \boldsymbol{\xi}^*), ..., \Gamma(\boldsymbol{\xi}^{(n)}, \boldsymbol{\xi}^*)]$ is an n element vector with each term being the covariance between all the observed points $(\boldsymbol{\xi}^{(i)}|i = 1...n)$ and the new point $\boldsymbol{\xi}^*$ under consideration. Substituting the inverse of the matrix $\mathbf{\Gamma}_{n+1}$ into equation (4.3) gives

$$P(Y(\boldsymbol{\xi}^*)|\mathcal{D}_n) = \frac{1}{Z}\exp\left[-\frac{(Y(\boldsymbol{\xi}^*) - \widehat{Y}(\boldsymbol{\xi}^*))^2}{2\sigma_Y^2(\boldsymbol{\xi}^*)}\right], \tag{4.5}$$

where

$$\widehat{Y}(\boldsymbol{\xi}^*) = \mathbf{k}^T\mathbf{\Gamma}_n^{-1}\mathcal{Y}_n, \tag{4.6}$$

and

$$\sigma_Y^2(\boldsymbol{\xi}^*) = \kappa - \mathbf{k}^T\mathbf{\Gamma}_n^{-1}\mathbf{k}. \tag{4.7}$$

Equations (4.6) and (4.7) can be viewed as a Gaussian process emulator of the original function $y(\boldsymbol{\xi})$. The posterior mean $\widehat{Y}$ can be interpreted as an estimate of the high-fidelity function value at $\boldsymbol{\xi}^*$, while the posterior variance σ_Y^2 can be interpreted as an estimate of the uncertainty involved in predicting the output using the dataset $\mathcal{D}_n$. Note that this

uncertainty arises from the fact that the dataset contains a finite number of training points. The posterior covariance can be written as

$$\mathcal{C}(\boldsymbol{\xi}^*, \boldsymbol{\xi}') = \Gamma(\boldsymbol{\xi}^*, \boldsymbol{\xi}') - \mathbf{k}^T \boldsymbol{\Gamma}_n^{-1} \mathbf{k}. \tag{4.8}$$

Note that the posterior variance $\sigma_Y^2(\boldsymbol{\xi}^*)$ is given by $\mathcal{C}(\boldsymbol{\xi}^*, \boldsymbol{\xi}^*)$. In this step we selected the model structure for the surrogate model. In the next section we discuss the choice of covariance function and how to tune the covariance function to a given training dataset.

4.2.3 Covariance Function Selection and Tuning

It can be observed from equations (4.6) and (4.7) that predictions made using the above model depend on the choice of the covariance function Γ used in the Gaussian process prior. In the ideal case the covariance function should depend on the observed dataset (Mackay, 2003). A very convenient and widely used covariance function is of the form

$$\Gamma(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}) = \theta_1 \exp\left[-\frac{1}{2}\sum_{i=1}^{q} \frac{(\xi_i - \xi_i')^2}{\theta_{2+i}^2}\right] + \theta_2 \tag{4.9}$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_{q+2})$ are a set of hyperparameters, that can be tuned to the training dataset $\mathcal{D}_n$. This parameterized covariance function belongs to the stationary family and obeys the product correlation rule. In the next chapter, we will discuss in detail the merits of selecting this covariance function. The next task is to estimate unknown model hyperparameters of the covariance function Γ using the observed dataset $\mathcal{D}_n$. This is also referred to as covariance tuning or tuning of the hyperparameters. The maximum likelihood estimation approach can be employed to seek values of the undetermined hyperparameters $\boldsymbol{\theta}$ that are most likely to have generated the observed dataset $\mathcal{D}_n$. Since the observed outputs have a Gaussian distribution, the log likelihood function can be written as

$$\mathcal{L} = -\frac{1}{2}\ln|\boldsymbol{\Gamma}_n| - \frac{1}{2}\mathcal{Y}_n^T \boldsymbol{\Gamma}_n^{-1} \mathcal{Y}_n - \frac{n}{2}\ln 2\pi. \tag{4.10}$$

To find the most probable value of $\boldsymbol{\theta}$, we can solve the optimization problem given below:

$$\boldsymbol{\theta}_{\text{opt}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}, \quad \text{where} \quad \boldsymbol{\theta} = (\theta_i | i = 1, 2, ..., q+2), \tag{4.11}$$

and where $\mathcal{L}$ is given by equation (4.10). In practice, any standard optimization technique can be employed to solve equation (4.11). For a detailed discussion of the computational aspects, the reader is referred to Keane & Nair (2005); Mackay (2003).

As shown in this section, the use of a Gaussian process prior over the response quantities gives us a statistical foundation for exploiting the information in the existing dataset for

further prediction. At the same time we have built an emulator which can act as a computationally cheap surrogate to the high fidelity simulator (CFD analysis in our case). In the next subsection we discuss methods to validate the accuracy of the surrogate model.

4.2.4 Model Validation

To assess the quality of the surrogate model we need to perform validation studies. This assessment study can be performed in many ways; two common methods are : 1) the accuracy of predicting the output at a number of additional points and 2) a leave-one-out type cross-validation procedure. The first method can become computationally prohibitive for high-fidelity models such as CFD simulation of a geometry with significant mesh size. A cross-validation procedure which does not need additional observation data can be a computationally cheap alternative. This method involves leaving the ith training point out and reusing the surrogate model. The new surrogate model is then employed to compute the posterior mean at the ith point, given by $\widehat{Y}_{-i}(\boldsymbol{\xi}^{(i)})$. By plotting the computed values against the original values from the training dataset, the quality of the surrogate model can be assessed. In a recent study, Meckesheimer *et al.* (2002) noted that the leave-one-out procedure may significantly underestimate the actual prediction error and suggested that a k-fold cross-validation scheme (with $k = 0.1n$ or $\sqrt{n}$) may be a better indicator of the model quality; i.e. where k elements are left out at each step.

Another measure of the quality of the Gaussian stochastic surrogate model would be to validate our basic assumption that a Gaussian process prior is appropriate for the CFD simulation code used as a black box here. Jones *et al.* (1998) have discussed the use of Standardized Cross-Validated Residual (SCVR) which is defined as

$$SCVR_i = \frac{y(\boldsymbol{\xi}^{(i)}) - \widehat{Y}_{-i}(\boldsymbol{\xi}^{(i)})}{\sigma^2_{-i}(\boldsymbol{\xi}^{(i)})}, i = 1, 2, ..., n, \tag{4.12}$$

where $\widehat{Y}_{-i}(\boldsymbol{\xi}^{(i)})$ and $\sigma^2_{-i}(\boldsymbol{\xi}^{(i)})$ denote the mean and variance of the metamodel prediction at a point $\boldsymbol{\xi}^{(i)}$ without using the ith training point. $SCVR_i$ can be computed for all the training points by removing the contribution of the corresponding points from the correlation matrix $\boldsymbol{\Gamma}$. In the next section, we train the Gaussian stochastic process model to a candidate dataset of points. After model validation studies, we use this surrogate model for probabilistic analysis and, later in the chapter for robust design studies.

4.3 Surrogate Based Probabilistic Analysis

In section 3.3.3 we employed a Quasi-Monte Carlo Method (LP_τ) in conjunction with the high fidelity CFD analysis system for probabilistic analysis. In this section, we train a Gaussian Stochastic Process model to the initial dataset of 90 points used in that study. The range of the erosion parameters are shown in table 3.2 and figure 3.11 shows the scatter of the points. The optimization problem, involved in tuning the hyperparameters, is solved using a hybrid Genetic Algorithm (GA) and Dynamic Hill Climbing (DHC) search method (Keane, 2003b).

The quality of the surrogate model is assessed by a leave-one-out type cross-validation procedure. As discussed in the previous section, this method involves leaving the ith training point out and computing the posterior mean at the ith point. The computed values using the surrogate are plotted against the original values from the training dataset in figure 4.1. The regression coefficient for a linear model fit to the leave-one-out results is $R^2 = 0.954$. A Standardized Cross-Validated Residual test is also performed over the surrogate model.

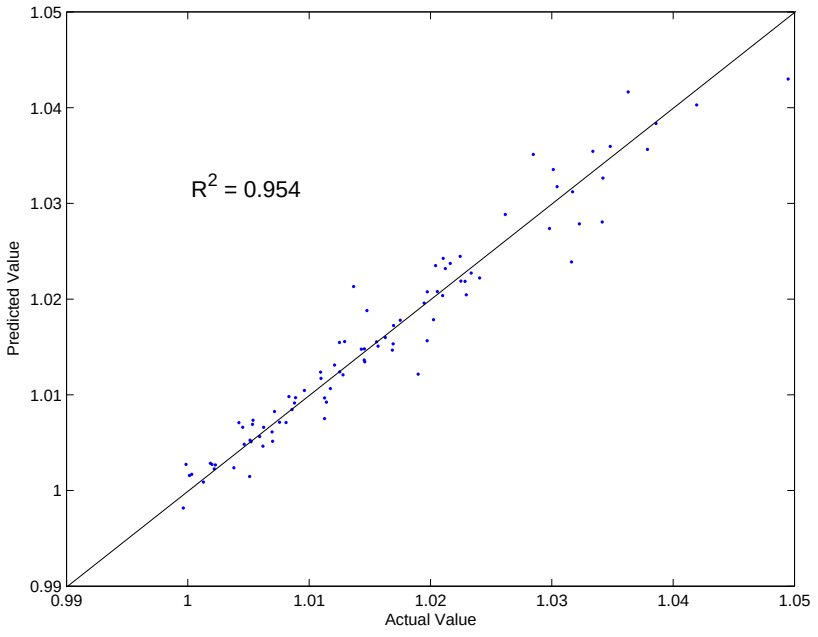


FIGURE 4.1: Predicted posterior mean versus original values

Figure 4.2 show the plot of posterior mean and SCVR_i values predicted by the leave-one-out method. It can also be observed that most of the SCVR_i in Fig. 4.2 lie within the range $[-3\sigma, +3\sigma]$ and hence the error bars computed using the surrogate are reasonably tight.

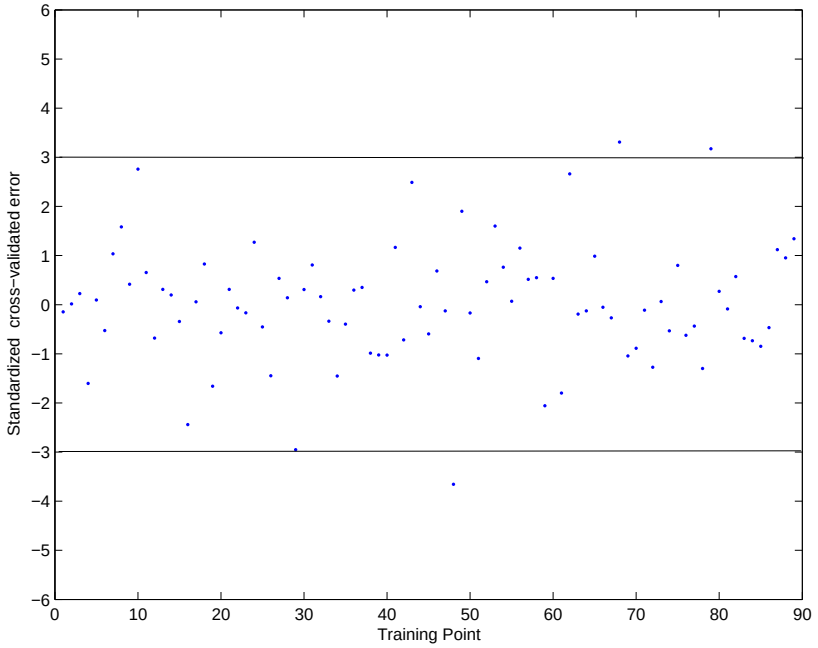


FIGURE 4.2: $SCVR_i$ Values using Leave-One-Out Validation

4.3.1 Probabilistic Analysis

Having established the quality of the surrogate model we use it to study the effect of erosion parameters on the blade at modest computational cost. A 10,000 point MCS is run on the surrogate model to generate the probability distribution of the pressure loss assuming a uniform distribution for the erosion parameters *location, width, depth* (A). The histogram for the pressure loss evaluated using MCS is shown in Fig. 4.3. The MCS using the surrogate model takes less than 4 seconds for carrying out 10,000 evaluations on an Intel(R) Xeon(TM) CPU 3.06GHz dual processor machine. This shows substantial savings in computational time, as compared to using the high-fidelity CFD model for probabilistic studies. The histogram shows that there has been considerable shift in mean performance from the nominal performance (Mean Shift). The predicted mean shift is close to 2%. It can also be observed that the performance degradation is as large as 6%.

Figure 4.4 shows the variations in the pressure loss for different settings of the erosion parameters *location, width, depth*, using the surrogate model. The parameters are varied in the range shown in Table 3.2. All variables are normalized between 0 to 1 for ease of presentation. Each row in figure 4.4 shows the contour plot of pressure loss for two variables at a different settings of the third variable. The setting of the third variable is shown by Z, which increases with the increase in column number. The values of the hyperparameter

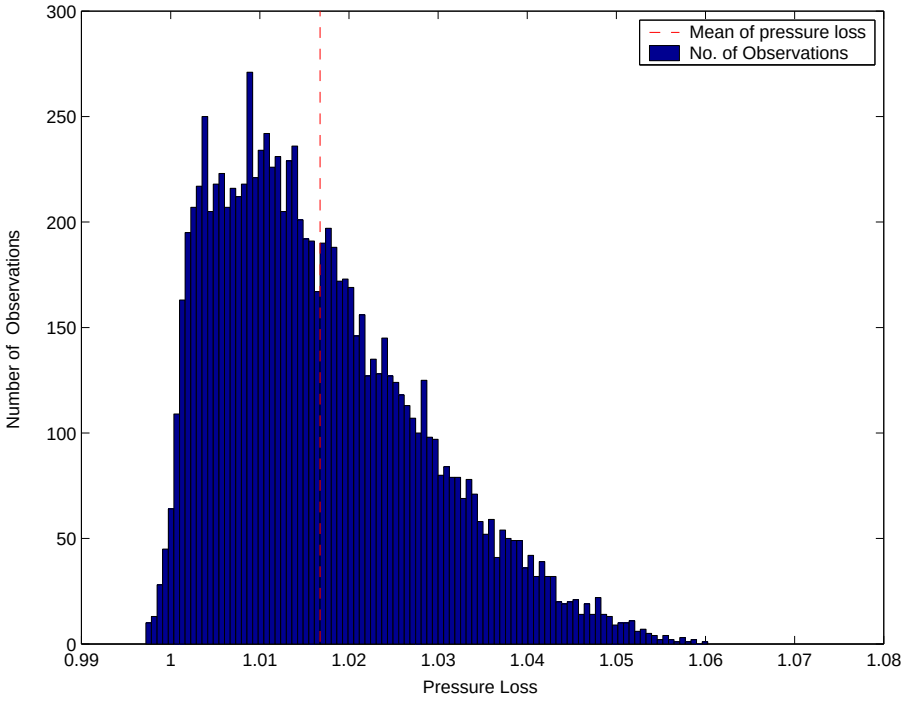


FIGURE 4.3: Probability Distribution of the Pressure Loss using MCS on the Surrogate Model

$\boldsymbol{\theta} = [\theta_{location}, \theta_{width}, \theta_{depth}]$ can be used to understand the relative importance of the erosion parameters on the pressure loss. The higher the θ value the more influence the respective parameter has on the objective. We found the $\boldsymbol{\theta}$ values corresponding to location, width and depth to be 1.413, 0.227 and 1.623, respectively. Since, there was no significant difference in these values, no insight into the relative importance of the input variables could be made. A more rigorous way to understand the effect of each parameter is the main effect analysis and is presented in the next subsection.

4.3.2 Main Effect Analysis

One of the limitations of non-parametric techniques like Gaussian stochastic process modelling, as compared to parametric models, is that the underlying input-output relationship is not available in a readily interpretable form. Hence, it is not straightforward to understand the effects of each input variable on the output performance. In order to understand the underlying functional relationship, the effect of each input parameter has to be isolated. The main effect of an input can be evaluated by integrating out the effect of the other input variables (Sacks *et al.*, 1989; Schonlau, 1997). The main effect of the ith variable can be

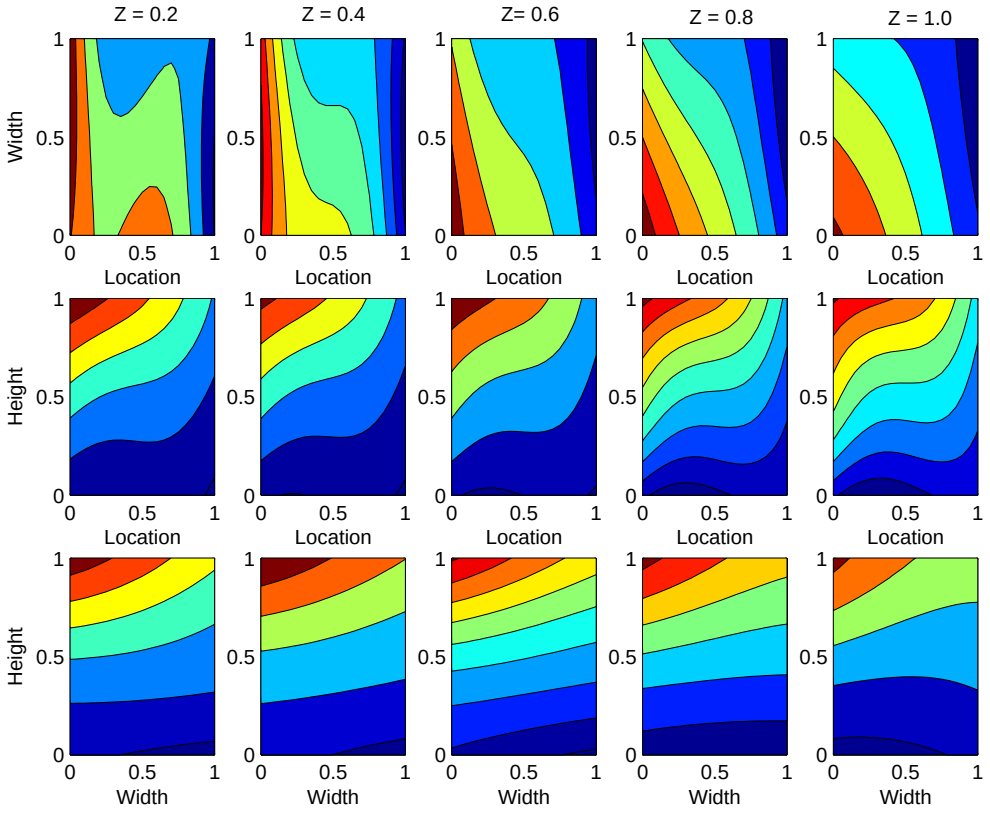


FIGURE 4.4: Contour Plot of the pressure loss for various settings of noise variables

expressed as

$$\mu(\boldsymbol{\xi}^{(i)}) = \frac{1}{V} \int y(\boldsymbol{\xi}) \prod_{h \neq i} \mathrm{d}\xi^{(h)}. \tag{4.13}$$

In most cases it is cumbersome to evaluate the above integral analytically. However, a numerical approximation can be obtained by performing a summation over a set of m discrete points $[\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, ..., \boldsymbol{\xi}^{(m)}]$. The m points can be generated by using any standard DOE technique. Note that since the Gaussian Stochastic Process model also provides the posterior variance, error bars on the main effects can be readily computed (Schonlau, 1997).

Main effect studies were carried out by numerically integrating out the effect of the other variables. The main effect plots for $[location, width, depth]$ are shown in Figures 4.5, 4.6 and 4.7. The main effect plots for *location* and *width* of the erosion are relatively flat as compared to the main effect plot for *height*. This suggests that *depth* of the erosion is the most influential input variable. The main effects plots also suggest that there is an approximately linear relationship between pressure loss and the *location* and *width* of the erosion. However, a relatively non-linear underlying relationship between the *depth* of the erosion and pressure loss is predicted. The probabilistic analysis suggested up to 6 %

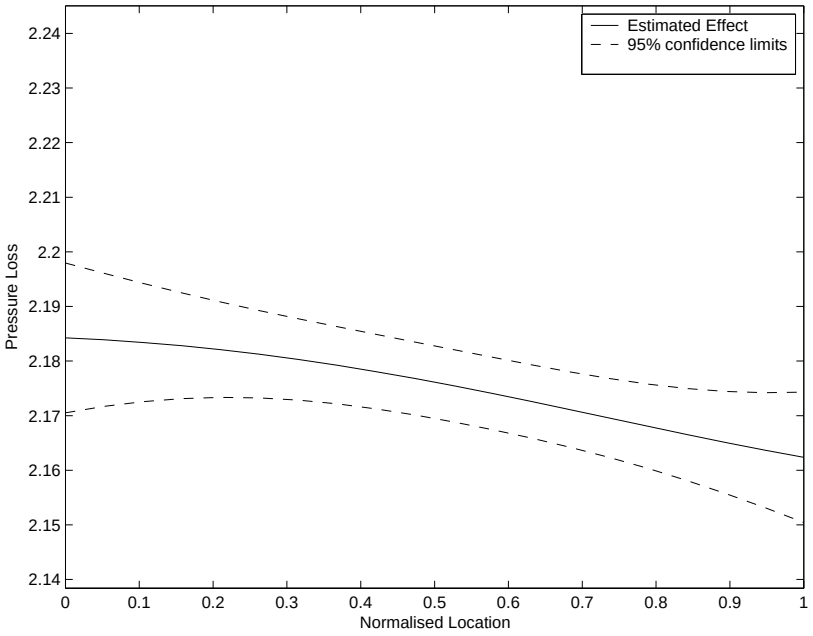


FIGURE 4.5: Main effect plot for location

degradation in pressure loss which emphasizes the need for blades which are robust against erosion. Since no one variable shows less effect on the performance as compared to the other two variables, we consider all of them for the robust design studies.

4.4 Robust Design Methodology

This sections discusses the methodology proposed for robust design of compressor blade sections against erosion. In the first step we define the control and noise variables. Next we present the objective functions in terms of the statistics of the aerodynamic performance. This is followed by a description of the multiobjective optimizer used in this study. Then we present the methodology employed for robust design optimization.

4.4.1 Control and Noise Factors

Robust design methods require the definition of noise factors and control (design) factors in the system design. The parameters used to model the eroded blades in section 3.3.1 are taken as the noise variables. This provides us with three noise factors - location (l), width (w) and depth (A). The compressor blade geometry itself also needs to be parametrized to create a set of control factors. To parametrize the blade section geometry we use a linear combination of Hicks-Henne functions. Wu *et al.* (2003) have presented and compared the

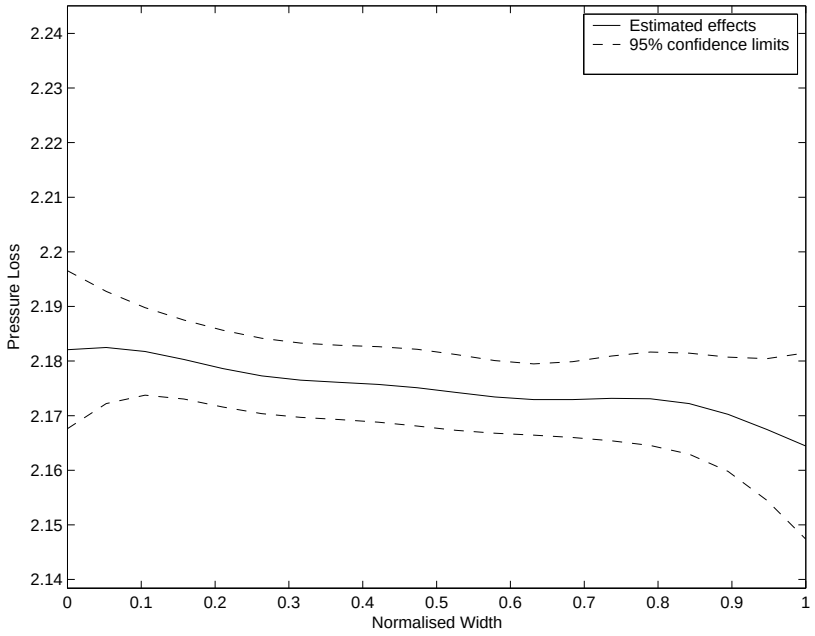


FIGURE 4.6: Main effect plot for width

efficacy of Hicks-Henne shape functions to other methods for modelling compressor blade sections. For the problem under consideration, we have used 10 Hicks-Henne functions, five each for the upper and lower airfoil section, to parametrize the compressor fan blade. The Hicks-Henne shape functions can also be expressed as

$$b_i(x) = sin^2(\pi x^{m_i}) \ , \quad m_i = ln(0.5)/ln(x_{M_i}) \quad \text{where} \quad i = 1, 2, ...n \tag{4.14}$$

where x is the normalized chord-wise coordinate from leading edge to the trailing edge ($0 \leq x \leq 1$), x_{M_i} are preselected-selected values corresponding to the location of the maxima and n is the number of Hicks-Henne functions used. In the present study, the locations of x_{M_i} for $i = 1, 2, ..., 5$ are chosen to be $0.5[1-cos(\beta_i)]$, where $\beta_i = \pi i/6$. This choice of x_{M_i} ensures that the distribution is denser near the leading edge and trailing edge, where the curvature is high. These shape functions are multiplied by an weight function $W_i \mid i = 1, 2, ..., 10$ and added to a typical Rolls-Royce compressor fan blade section to obtain new shapes. The amplitudes of these shape functions are used as design variables. Hence, for our robust design study we have 10 design variables which can be treated as the control factors. Some typical shapes obtained by using this method are shown in figure 4.8. Figure 4.9 also shows zoomed in view near the leading edge for some typical shapes obtained using these control factors

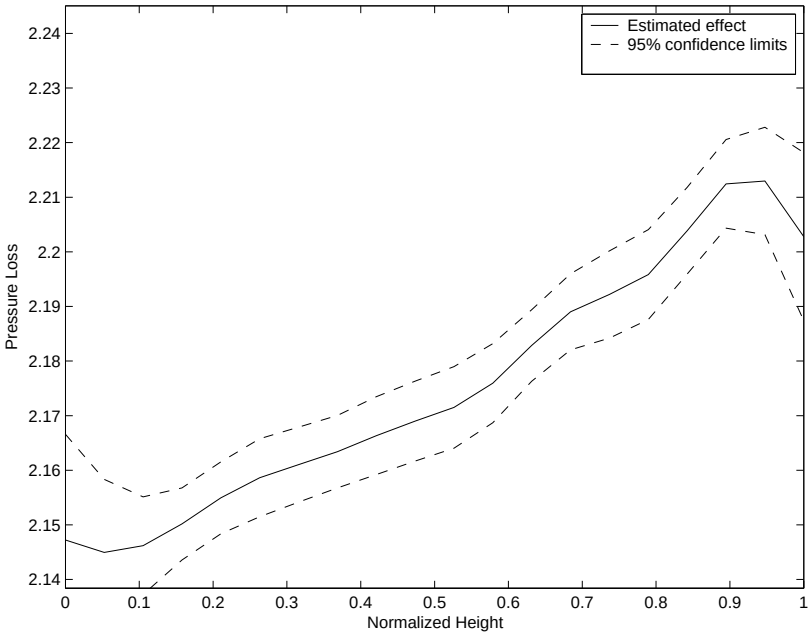


FIGURE 4.7: Main effect plot for depth

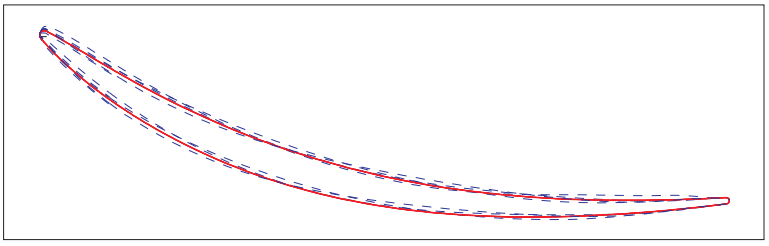


FIGURE 4.8: Typical shapes obtained by varying the control factors

4.4.2 Multiobjective Optimization: NSGA-II

Robust design is inherently a bi-objective problem since we seek to control both the mean and standard deviation of performance. The presence of multiple objectives in a problem leads to a set of optimal solutions, rather than a single solution. Such a solution set is referred to as Pareto-optimal set in the optimization literature. Each point in the set is optimal in the sense that no improvement can be achieved in one objective without worsening the other objective. In the absence of further information about the relative importance of the objectives, it is not possible to decide which design is better than the rest. Hence, it is important to find as many Pareto-optimal solutions as possible for the benefit of the designer.

Classical optimization methods suggest converting the multiple objective optimization problem to a single objective optimization problem emphasizing one particular Pareto-

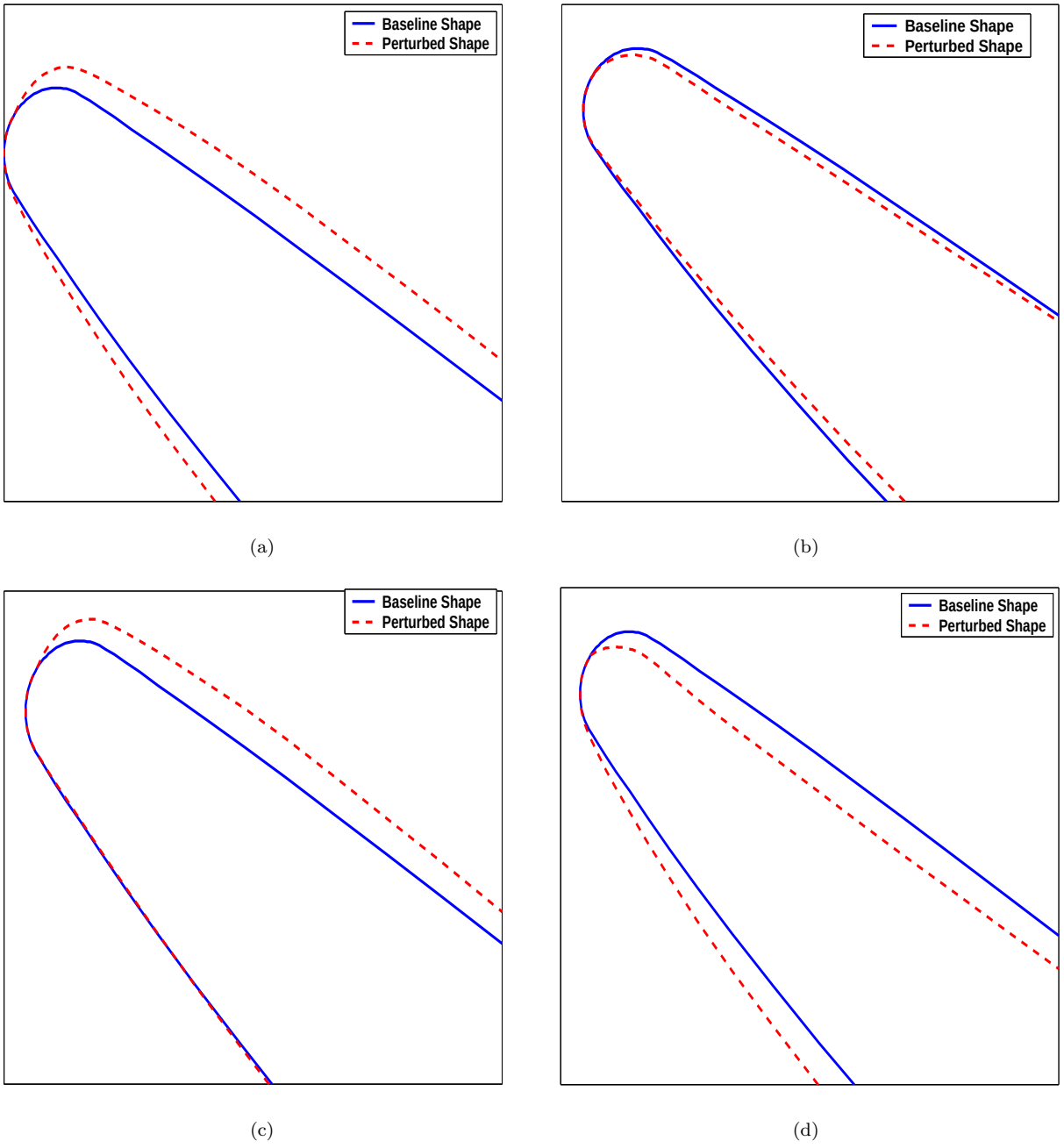


FIGURE 4.9: Zoomed in view near the leading edge for some typical shapes obtained by varying the control factors

optimal solution at a time. Such methods prove to be computationally expensive and do not ensure convergence to true optimal Pareto sets in non-convex problems (Athan & Papalambros, 1996; Koski, 1985). In contrast, Genetic Algorithms are inherently suited for multi-objective problems as they have the ability to find multiple Pareto-optimal solutions in one simulation run. Since GAs work with a population of solutions, it is easier to extend them to maintain high diversity in finding multiple Pareto-optimal solutions at each stage, while moving toward the true Pareto-optimal region (Fonseca & Fleming, 1993).

In recent years several approaches have been proposed to solve Multi-objective problems using GAs (Horn *et al.*, 1994; Deb, 1999). The elitism based Non-dominated Sorting Genetic Algorithm (NSGA-II) proposed by Deb *et al.* (2002) is employed here to seek the true Pareto-optimal front. The NSGA-II method is fast as it has a computational complexity of $O(MN^2)$ (where M is the number of objectives and N is the population size) when compared to other non-dominated GA with computational complexity $O(MN^3)$. The NSGA-II method also uses elitism to enhance the performance of the GA and prevent the loss of good solutions once they are found. Traditional GA methods ensure diversity in a population by relying on the concept of sharing. In such methods it is necessary to specify the sharing parameter (σ_{share}) beforehand by the user. The performance of sharing functions in ensuring diversity, is dependent upon the choice of σ_{share}. In practice it is not very obvious how to select the best σ_{share}. In NSGA-II the sharing function approach is replaced by a crowded comparison approach. The crowded comparison approach has a better computational complexity and eliminates any user defined parameter for maintaining diversity among population members. In this study NSGA-II is employed in conjunction with surrogate models to identify the true Pareto-optimal front to seek robust designs. We discuss the underlying algorithm in NSGA-II in detail later in this thesis.

4.4.3 Robust Design Method

For the present study the robust design objective can be can be expressed as

$$\begin{aligned} &\text{Minimize}: \quad \mu = \tfrac{1}{k}\textstyle\sum_{i=1}^{k} Pl_i \quad \text{and} \\ &\text{Minimize}: \quad \sigma = \sqrt{\tfrac{1}{k-1}\textstyle\sum_{i=1}^{k}\left(Pl_i - \mu\right)^2}, \quad i = 1, 2, ...k. \end{aligned} \tag{4.15}$$

We present an efficient method to achieve the above mentioned objective using the various tools described earlier in this chapter. Figure 4.10 presents the flowchart for the proposed robust design method. In the first step we select the design space and use DOE techniques

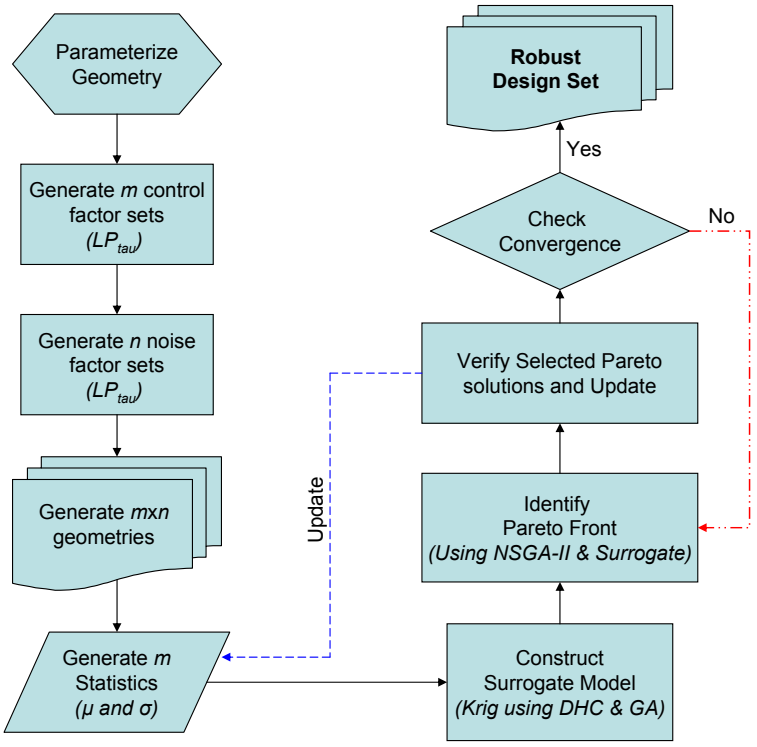


FIGURE 4.10: Flowchart for the Robust Design Methodology proposed

to rationally choose a set of compressor fan blade sections as initial m candidate points. Subsequently a second level of DOE is run to suggest n different erosion geometries on each candidate compressor fan blade section. This is very similar to the inner control factor array of m points and noise factor array of n points used in Taguchi's system design. The eroded compressor blades are modeled using the novel parameterization technique based on Hicks-Henne functions, discussed earlier in section 3.3.1. The Parametric Design and Rapid Meshing (PADRAM) tool is used to produce high quality hybrid meshes. The multigrid Reynolds-Averaged Navier Stokes (RANS) solver HYDRA with Spalart Allmaras turbulence model is used for Computational Fluid Dynamics (CFD) simulations to calculate the total pressure loss over the compressor blade section at each of the $m \times n$ points. The mean and standard deviation of the total pressure loss (over n erosion types settings) is calculated for all the m blade sections.

Unlike conventional robust design procedures we do not limit ourselves to just searching

the initial design space. A Gaussian stochastic process model is employed to generate a computationally less expensive surrogate to predict the mean and standard deviation of the total pressure loss. Note that the Gaussian process emulator maps the design variables W_i to the output statistics (mean and standard deviation). These output statistics are estimated for each blade using MCS of size n over the erosion parameters. Note that this MCS of n size is run over the high fidelity solver and not the surrogate model. The hyperparameters $\boldsymbol{\theta}$ of the surrogate model are estimated via maximum likelihood estimation. NSGA-II is then used to search the entire design space using this surrogate to obtain Pareto-optimal solutions. The errors in the prediction can be estimated from the posterior variance given by the surrogate model at each point. The prediction, using the surrogate model, at the points on the Pareto-front are then verified by running full scale CFD simulations. A low-crowding algorithm is used to select points for update at which the full CFD model is run in the noise space (n runs to estimate the statistics). The surrogate model is then updated at the suggested points; i.e. the new data is updated to the existing training dataset and the hyperparameters are re-tuned. This process is performed iteratively until some convergence criteria are satisfied or the specified computational budget is exhausted. In this study, the process is terminated if for more than two update iterations, there is no improvement in Pareto front as compared to the previous best Pareto front. The designer can subsequently use the resulting optimal design set to trade off between mean performance and variance to obtain robust designs.

4.5 Numerical Studies and Results

The robust design method discussed above is next applied to seek compressor fan blades with robust aerodynamic performance in the presence of erosion. We compare the performance of the robust blade to another blade obtained by deterministic optimization method. In the deterministic optimization process we use the same design variables but no noise variables (no erosion) and simply seek the blade with the best nominal pressure loss. The blade is also compared to the baseline blade from which both optimization process started.

4.5.1 Robust Design Studies

The robust design method discussed earlier is applied to a typical Rolls-Royce compressor fan blade section. An LP_τ based DOE is employed to create an initial control factor set

of m ($m = 50$) compressor fan blade shapes. This is followed by another set of LP_τ based DOE to create n (n=15) types of erosion on each of the m blade shapes. The noise factors - location (t_1), width (t_2) and depth (A) are represented by a uniform distribution. PADRAM is employed for generating high quality CFD meshes and a multigrid RANS based CFD simulation using HYDRA is performed at the $m \times n$ (50×15) candidates points to evaluate the mean and variance of the pressure losses corresponding to the m designs. Note that the mean ans standard deviation have been estimated using a sample size of 15 points. NSGA-II is used in conjunction with Gaussian stochastic process models for the mean and variance of the loss coefficient to seek Pareto optimal solutions. After each search iteration 10 new points are selected on the Pareto front obtained using the surrogate models using a low crowding algorithm. Exact runs of the CFD code are carried out at these points and the resulting data is used to update the baseline surrogate models for the mean and variance

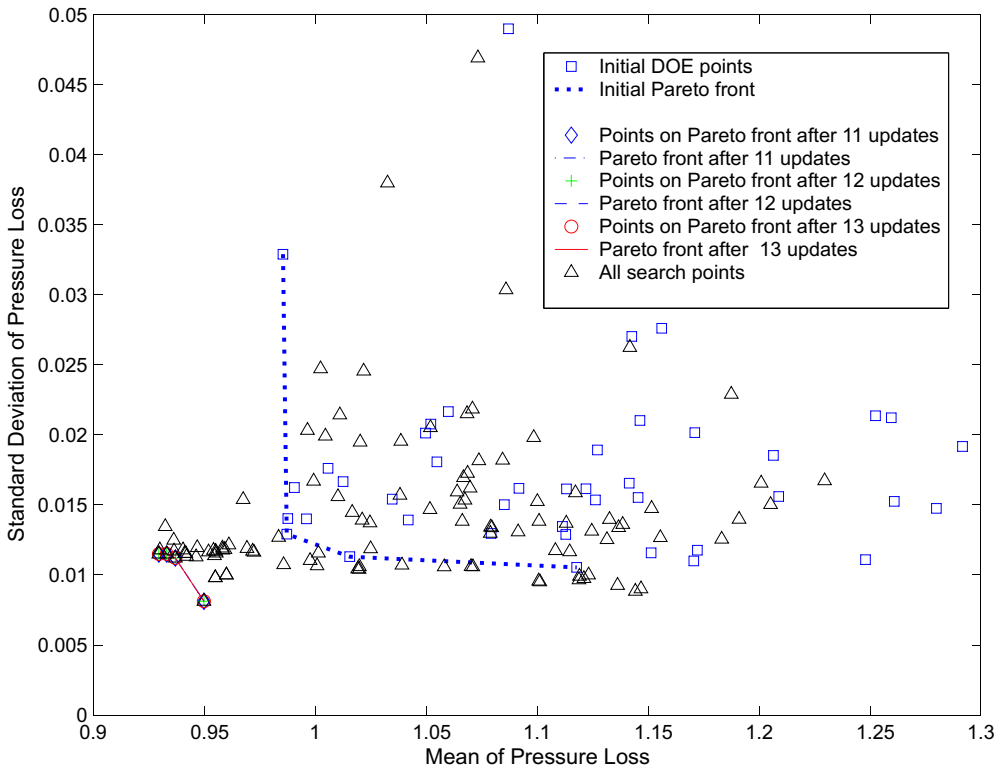


FIGURE 4.11: Plot of the initial dataset and the initial Pareto front. The plot also shows the last three Pareto fronts after which the search was terminated. Note that the $11^{th}, 12^{th}$ and 13^{th} Pareto fronts are the same and overlap, hence they are not distinguishable in the plot

The NSGA-II algorithm with the update method is employed for 13 updates to find the optimal robust design set (i.e. 180×15 CFD runs are used in total). The search is terminated

when there is no further improvement in the Pareto front. Figure 4.11 shows the convergence of the Pareto fronts. It can be observed from Fig. 4.11 that the last three Pareto fronts are same, and hence the search was terminated. It should be noted that this termination criterion does not ensure convergence to the true Pareto front. However, our primary objective is to seek design improvements on a limited computational budget and this has been achieved. Figure 4.11 shows the initial dataset with the initial Pareto front (dotted line). The solid line shows the final Pareto front. It can be noted from Fig. 4.11 that significant improvement in the Pareto front has been achieved.

A 50 point LP_τ based DOE with noise factors - location (t_1),width (t_2) and depth (A) is executed for a selected geometry on the final Pareto front. The data is used to train a surrogate model, which is further used for an MCS. An MCS of 10,000 runs is executed for the selected blade geometry and the histograms of the pressure loss are generated. Figure 4.12 shows the robust geometry as compared to the baseline geometry. The histogram of

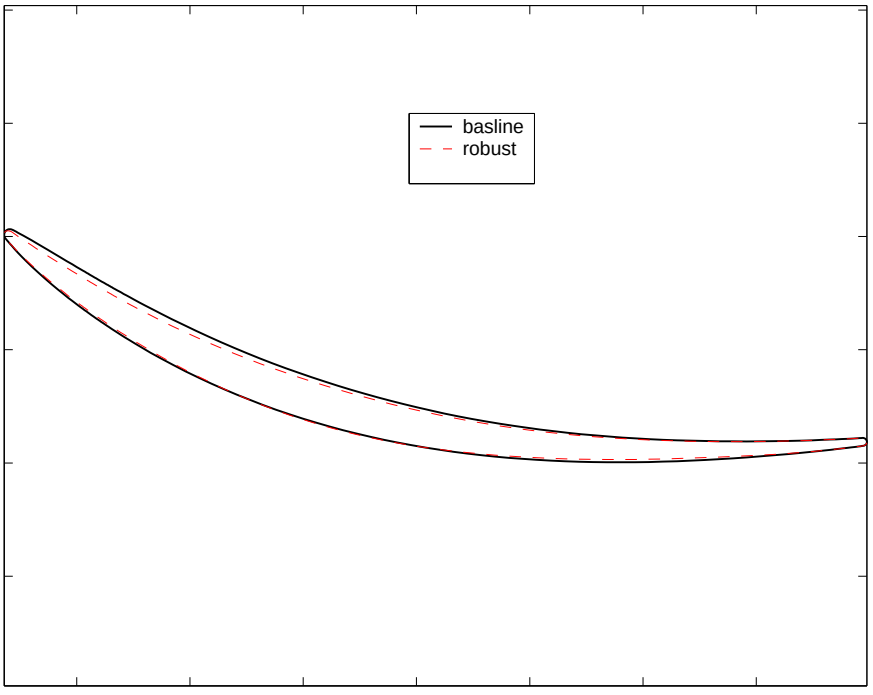


FIGURE 4.12: Shape of the robust geometry and the baseline geometry

pressure loss for the robust geometry is shown in figure 4.13.

4.5.2 Deterministic Design

To provide a benchmark against which the results of a multiobjective robust design search can be compared, we begin with a traditional deterministic optimization approach. Deterministic

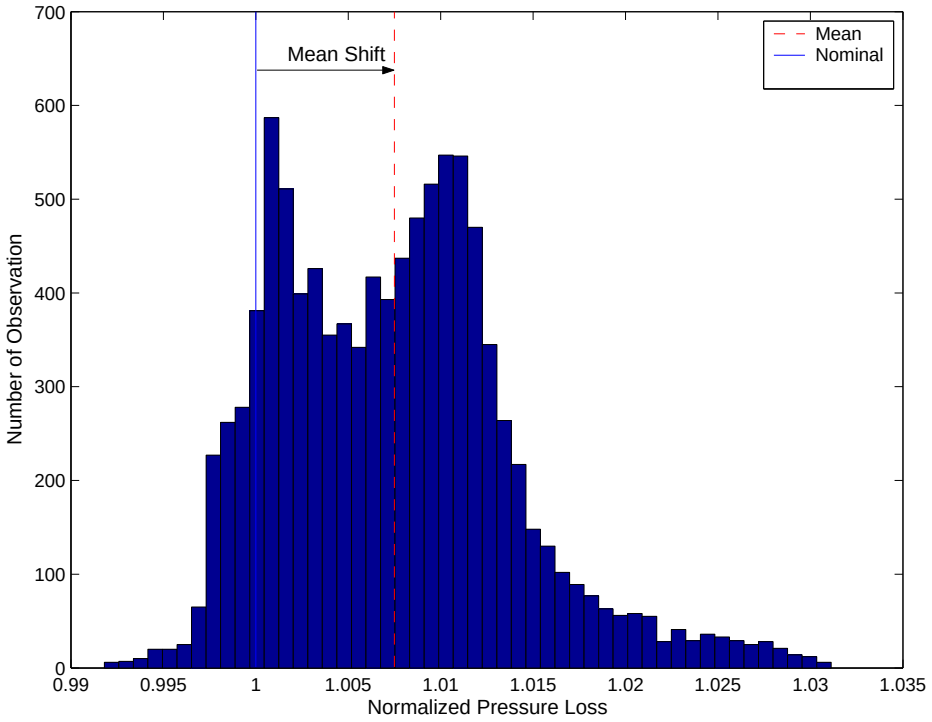


FIGURE 4.13: Histogram of Pressure Loss of the Robust Geometry

design methods seek to optimize the nominal performance of the system, i.e., optimize blade geometries for low pressure loss coefficients. An LP_τ based DOE is employed to generate an initial set of training points containing different blade geometries whose performances are evaluated using the CFD code. The initial dataset is subsequently used to construct a baseline surrogate model which is then used in lieu of the CFD simulator during optimization. During the optimization iterations, the surrogate models are updated in a sequential fashion to improve its accuracy. The approach employed is illustrated by the flowchart in Fig. 4.14.

We employ the Dynamic Hill Climbing based optimization method in OPTIONS software framework for the design search (Keane, 2003b). The initial DOE points and all the points explored during the optimization search are shown in figure 4.15. The figure shows the convergence history and the deterministic optimal point chosen for comparison. Once the optimal blade is found we then analyze its behaviour in the presence of erosion. A probabilistic analysis, similar to the study performed earlier, is employed to study the deviation in coefficient of pressure loss for the resulting design. A 50 point LP_τ DOE survey is performed over the values of the erosion parameters *location, width, depth* and CFD calculations using HYDRA are executed at these points. The initial dataset is used to train a Gaussian stochastic process model. An MCS of 10,000 runs is employed to obtain the statistics of the

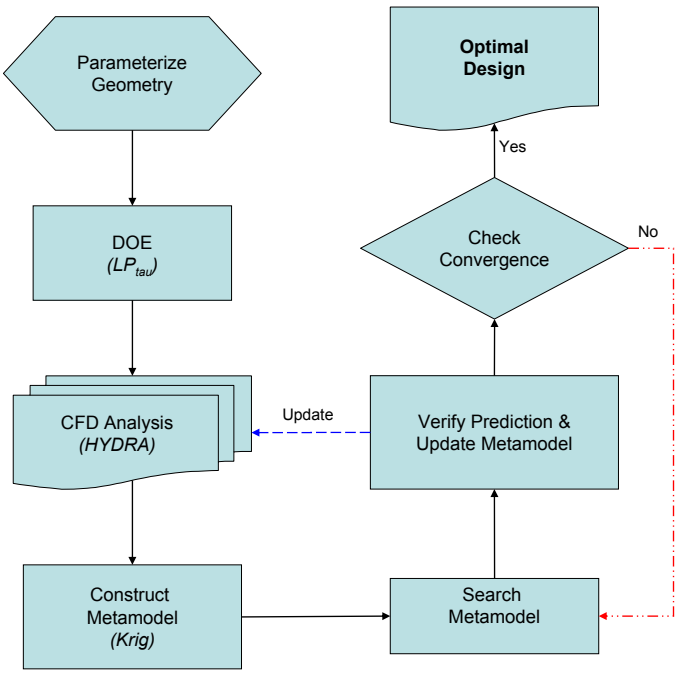


FIGURE 4.14: Flowchart for deterministic surrogate-assisted design optimization

blade performance.

We compare the performance of the geometry which was optimized for pressure loss coefficient using the deterministic method with the selected robust geometry. Figure 4.16 shows the histograms for both geometries with nominal pressure loss coefficient and the mean of the pressure loss coefficient in the presence of erosion. The histogram of the robust geometry shows less variability in pressure loss coefficient as compared to the design obtained using a deterministic approach. There is considerably lower shift in the mean performance of 0.7% from the nominal performance for the robust blade geometry as compared to almost 2% for the deterministic optimal blade geometry. It can be observed that low variability has been achieved in the robust design at the expense of a marginal reduction in the mean performance. The worst case performance of the robust blade geometry is also considerably better than the deterministic optimal blade geometry.

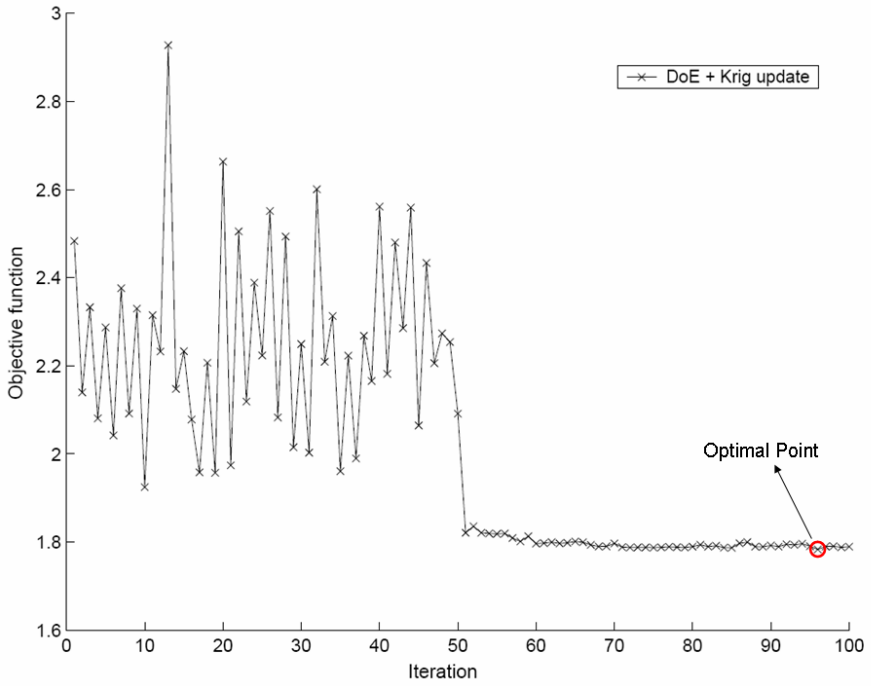


FIGURE 4.15: Convergence plot for deterministic surrogate-assisted design optimization using DHC

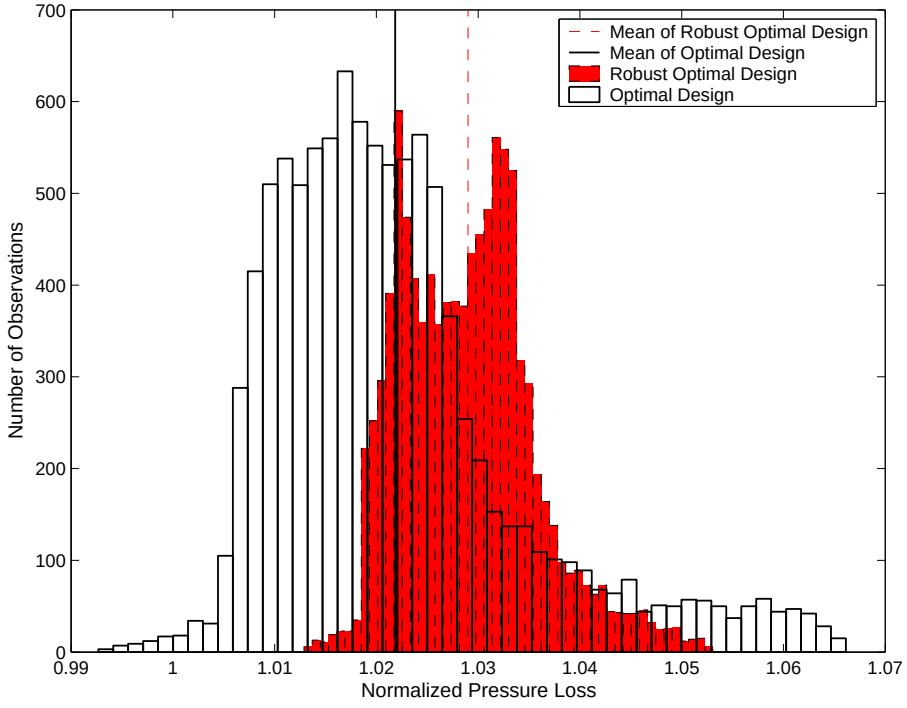


FIGURE 4.16: Histograms of robust and deterministic optimal geometries

4.6 Summary

In this chapter we have presented an efficient surrogate model and genetic algorithm based robust design methodology. An LP_τ based DOE technique was used to construct an initial inner control array and outer noise array. A parametric grid generation routine was used to automate geometry creation and grid generation to construct high quality CFD meshes which were then solved using a RANS code. Gaussian stochastic process models were used as computational surrogates to the high-fidelity CFD simulations in order to ensure convergence close to the true Pareto front using a limited number of exact function evaluations.

The robust design problem was formulated as a multi-objective problem. An elitism based non-dominated sorting genetic algorithm was employed in conjunction with the surrogate model to search the design space. A Pareto-optimal set was identified for trade off between the mean and standard deviation of the pressure loss. The efficiency of the proposed solutions would depend on the quality of the surrogate model used. The Pareto-optimal set suggested by the NSGA-II, using the surrogate model, was verified using CFD simulations and few points were selected from the final Pareto front for further analysis and updating the surrogate models. An MCS based on surrogate models was executed for the selected blades and the results were found to be considerably more robust than the design obtained using a deterministic optimization approach. The method presented can be employed to seek robust optimal sets which can be presented to the designers to find compressor blade designs that are robust to erosion processes.

The major limitation in this chapter was the number of sample points taken in the noise space. This was due to the high computational cost involved in taking a large sample. Recall that then we employed MCS to estimate the statistics of pressure loss for each selected blade geometry using this sample. It is well known that the MCS is very sensitive to the sample size employed for predicting the statistics of response (Convergence rate is $\mathcal{O}(1/\sqrt{n})$). In the next chapter, we will tackle this issue by employing Bayesian Monte Carlo methods. On a different note, the CFD solutions for the eroded compressor blades may have significant errors. This may be due lack of verification of the CFD solver with experimental data for these eroded geometries. In such case, we cannot rely much on our results as the contributions of the CFD solution errors to the pressure loss variations may be comparable to the contributions due to geometric uncertainty. However, the main aim was to achieve better robustness in performance on a limited computational budget and it was achieved.

Chapter 5

Robust Design against Manufacturing Variations

In this chapter, we propose an efficient strategy for robust design based on Bayesian Monte Carlo simulation. Robust design is formulated as a multiobjective problem with the mean performance and variability of the performance as the objectives to be minimized. Bayesian Monte Carlo simulation is employed to efficiently estimate these performance statistics for optimization. The proposed method is applied to compressor blade design in the presence of manufacturing uncertainty. Process capability data is utilized in conjunction with a parametric geometry model for manufacturing uncertainty quantification. High fidelity computational fluid dynamics simulations are used to evaluate the aerodynamic performance of the compressor blade.

A probabilistic analysis for estimating the effect of manufacturing variations on the aerodynamic performance of the blade is performed. Sensitivity analysis studies are conducted to understand the effect of the design and noise variables on the performance output. The proposed approach is applied to robust design of compressor blades and a selected design from the final Pareto set is compared with an optimal design obtained by minimizing the nominal performance alone. The selected robust blade has substantial improvement in robustness against manufacturing variations in comparison to the deterministic optimal blade. Significant savings in computational effort using the proposed method as compared to MCS are also illustrated.

5.1 Introduction

Manufacturing uncertainty is inevitable due to the presence of varying process conditions, tool wear, human errors or tolerancing limitations. In Chapter 3, we discussed the role of process capability data in quantifying the uncertainty present in manufacturing processes. We also presented a methodology for utilizing process capability data to represent manufacturing uncertainty in compressor blades. It was observed, that such uncertainty can cause significant deterioration in the aerodynamic performance of the blade. This established the need for robust design studies in the early design phase.

We left the earlier chapter mentioning some limitations of the robust design method used for erosion study; one of the major limitations being the choice of sample size used for the inner noise loop. In this chapter, we present Bayesian Monte Carlo based Monte Carlo Simulation (BMCS) method for efficient integration over the noise variables to evaluate the performance statistics. Also manufacturing variations cause smooth geometry changes and we expect the CFD solver may not contribute *bias* in the performance statistics. In section 5.1, we present the BMCS method and compare it with existing methods using a simple example problem. This is followed by a definition of the manufacturing uncertainty problem in the context of robust design. In section 5.3, we conduct a robust design study against manufacturing uncertainty on a typical compressor blade geometry. The probabilistic aerodynamic performance of the robust design obtained is compared to the performance of a deterministic optimal design in the presence of manufacturing variations.

5.2 Why Bayesian Monte Carlo?

In chapter 2 we presented an overview of Monte Carlo Simulation to compute the statistics of the response quantities of interest. In this section, we look at MCS from the perspective of robust design studies; that is for evaluating the integrals in equations (2.7), (2.11) and (2.12) (see section 2.4). For every set of control factors ($\mathbf{x} \in \chi$), MCS employs a random number generator to select a set of points in the noise space ($\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, ..., \boldsymbol{\xi}^{(m)}$), where $\boldsymbol{\xi}^{(i)} = \{\xi_1^{(i)}, \xi_2^{(i)}, ..., \xi_q^{(i)}\}$. Once the response quantity $y(\mathbf{x}, \boldsymbol{\xi}^{(i)})$ is evaluated using a computational analysis model at these points, the mean of the output can be estimated by equation 2.1 and the variance can be estimated by equation 2.2.

As mentioned in Chapter 2, the standard MCS technique has many limitations. First, a large sample size is required for accurate inferencing, which rules out its application to

computationally expensive high fidelity simulation models. Another drawback of MCS is that the random samples generated for evaluation may not be space filling, which is important with computational analysis models as they are deterministic. In Chapter 2 we presented Pseudo and Quasi-MCS based methods that may be used to address these issues. We also presented surrogate based MCS to reduce the high computational cost involved with robust design. A Gaussian Stochastic Process emulator was employed as a surrogate in the previous chapter. However, another major drawback of MCS is that it uses only the observations of the output response quantity $[y^{(1)}, y^{(2)}, ..., y^{(m)}]$ for estimating the statistics and does not take into account the points $(\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, ..., \boldsymbol{\xi}^{(m)})$ at which they were observed (O'Hagan, 1987). Here, we present a Bayesian approach that does not suffer from these limitations. In the previous chapter, we employed Gaussian stochastic process model as an emulator for probabilistic analysis. We presented a statistical method that utilizes the information in the observed dataset $\mathcal{D}_n$ to build an emulator, given by (4.6) and (4.7), which can act as a cheap surrogate to the high fidelity analysis code. In the next section, we extend this idea to the case when the inputs to this emulator are random.

5.2.1 Random Inputs: BMCS

Here, we revisit the problem mentioned earlier, where we need to evaluate the multidimensional integral $\int_{\mathcal{E}} f(\boldsymbol{\xi})P(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi}$. This is the case when the input $\boldsymbol{\xi}$ is random with distribution $P(\boldsymbol{\xi})$, and one aims to estimate the statistics of the response output $f(\boldsymbol{\xi})$. Recollect that such problems are central to robust design formulations where the aim is to evaluate the statistics of design performance in the presence of uncertainty due to noise variables. For example, in the case of manufacturing uncertainty, the inputs are given by a probability density function. In the limiting case the probability density function is Gaussian, with the mean given by baseline geometry and variance specified by the process capability variance σ_p. In the Gaussian process modeling approach described earlier we had a set of inputs and the corresponding outputs from an analysis code represented by $\mathcal{D}_n = (\boldsymbol{\xi}^{(i)}, y(\boldsymbol{\xi}^{(i)})|i = 1...n)$. Using this method we approximated the output response by a Gaussian random field given by

$$Y(\boldsymbol{\xi}) \sim \mathcal{N}(\widehat{Y}(\boldsymbol{\xi}), \mathcal{C}(\boldsymbol{\xi}, \boldsymbol{\xi}')). \tag{5.1}$$

The Gaussian field $Y(\boldsymbol{\xi})$ can be employed to approximate the integral $\int_{\mathcal{E}} f(\boldsymbol{\xi})P(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi}$. This method is also known as the Bayesian Monte Carlo Simulation (BMCS) (Rasmussen, 2003). Since $Y(\boldsymbol{\xi})$ is a Gaussian random field its expected value will be a Gaussian random variable

and is given by

$$\mathcal{M} = \int_{\mathcal{E}} Y(\boldsymbol{\xi}) P(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \sim \mathcal{N}(\langle \mathcal{M} \rangle, \sigma^2_{\mathcal{M}}), \tag{5.2}$$

where

$$\langle \mathcal{M} \rangle = \int_{\mathcal{E}} \widehat{Y}(\boldsymbol{\xi}) P(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \quad \text{and} \quad \sigma^2_{\mathcal{M}} = \int_{\mathcal{E}} \int_{\mathcal{E}} \mathcal{C}(\boldsymbol{\xi}, \boldsymbol{\xi}') P(\boldsymbol{\xi}) P(\boldsymbol{\xi}') \mathrm{d}\boldsymbol{\xi} \mathrm{d}\boldsymbol{\xi}'. \tag{5.3}$$

Here, $\langle \mathcal{M} \rangle$ can be interpreted as an estimate of the mean of $Y(\boldsymbol{\xi})$, while $\sigma^2_{\mathcal{M}}$ is a probabilistic error bar which is analogous to the variance of the MCS (equation (2.2)), and can be used to estimate the accuracy of the BMCS prediction. The integrals in equation (5.3) can be estimated using simulation techniques. This is feasible since predicting $\widehat{Y}(\boldsymbol{\xi})$ and $\mathcal{C}(\boldsymbol{\xi}, \boldsymbol{\xi}')$ at a new point using the Gaussian process emulator given by Equations (4.6) and (4.8) is computationally very cheap incurring only $\mathcal{O}(n^2)$ operations.

In the special case when the input variable distribution $P(\boldsymbol{\xi})$ is Gaussian and the covariance function $\Gamma(\boldsymbol{\xi}, \boldsymbol{\xi}')$ obeys the product correlation rule (for example Gaussian), the integral in equation (5.3) can be evaluated analytically (O'Hagan *et al.*, 1999; Rasmussen & Ghahramani, 2003). Let $P(\boldsymbol{\xi}) \sim \mathcal{N}(\mathbf{b}, \mathbf{B})$ and the covariance function be of the form $\Gamma = \mathcal{N}(a_i = \boldsymbol{\xi}^{(i)}, \mathbf{A} = \mathrm{diag}(\theta_1^2, ..., \theta_{q+1}^2))$ given by

$$\Gamma(\boldsymbol{\xi}, \boldsymbol{\xi}') = \theta_1 \mathrm{exp}\left[-\frac{1}{2} \sum_{i=1}^{q} \frac{(\xi_i - \xi_i')^2}{\theta_{i+1}^2} \right]. \tag{5.4}$$

Then the first integral in Equation (5.3) simplifies to

$$\langle \mathcal{M} \rangle = z^T \boldsymbol{\Gamma}^{-1} \mathcal{Y}_n, \tag{5.5}$$

where

$$z = \theta_1 |\mathbf{A}^{-1}\mathbf{B} + \mathbf{I}|^{-1/2} \mathrm{exp}\left[-0.5(\mathbf{a} - \mathbf{b})^T (\mathbf{A} + \mathbf{B})^{-1} (\mathbf{a} - \mathbf{b}) \right]. \tag{5.6}$$

Similarly, the second integral in Equation (5.3) which gives an error bar for $\langle \mathcal{M} \rangle$ simplifies to

$$\sigma^2_{\mathcal{M}} = \theta_1 |2\mathbf{A}^{-1}\mathbf{B} + \mathbf{I}|^{-1/2} - z^T \boldsymbol{\Gamma}^{-1} z. \tag{5.7}$$

As shown in this section, the use of a Gaussian process prior over the response quantities provides a statistical approach that fully utilizes the information in the existing dataset for uncertainty analysis. At the same time we have an emulator which can act as a computationally cheap surrogate to the high fidelity simulator (CFD analysis in our case). Hence, the above proposed methodology alleviates the drawbacks of the standard MCS technique discussed earlier. In the next subsection we apply the BMCS based methodology to a simple test case.

5.2.2 Case Study: Rastrigin Function

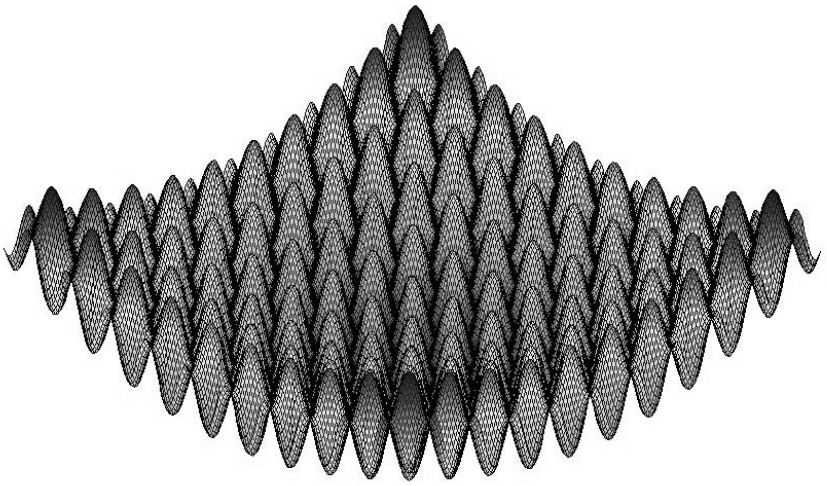


FIGURE 5.1: The plot of Rastrigin function

We look at the convergence rates of random MCS, quasi MCS (LP_τ) and Bayesian MCS methods to predict the statistics of the output when the underlying relation between the input and outputs is a complex function. The highly nonlinear *Rastrigin function*, see figure 5.1, is used to study the convergence of the above mentioned methods. For two independent variables, the Rastrigin function can be expressed as

$$R(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2). \tag{5.8}$$

The random inputs x_1 and x_2 are assumed to be uniformly distributed; i.e. $x_1 \in [-1, 1]$ and $x_2 \in [-1, 1]$. To obtain a benchmark for comparison, an initial random MCS of 10,000 runs is executed. This is employed to calculate the mean and standard deviation of the output $R(x_1, x_2)$ using equations (2.1) and (2.2). The mean and standard deviation using the random and quasi MCS are estimated incrementally. This implies that the number of points in the sample, for estimating the statistics, are incremented after each step. In the random MCS the points are incremented randomly, whereas, in the quasi MCS the new point is selected using the LP_τ method based on Sobol sequences. The choice of using LP_τ method is due to the fact that it generates a deterministic DOE dataset. This property makes it ideal for our study as we can create a dataset initially and can employ it to increment the sample size at each step without generating a new DOE dataset at every step.

For the case of Bayesian MCS, at each step, we increment the sample using the LP_τ DOE technique. Then a Gaussian stochastic process model is trained to the updated sample data

to obtain a surrogate model as discussed in Chapter 4. This emulator is employed for BMCS as discussed in the previous section. The statistics of interest can then be evaluated using the integrals in equation (5.3). Since the inputs are not Gaussian (uniform in this case), we do not use analytical solutions for these integrals. Instead a MCS of 10,000 runs is further executed using the emulator model to estimate the integral at each step. Figure 5.2 and 5.3 show the convergence plots for the above mentioned methods. These figures clearly illustrate

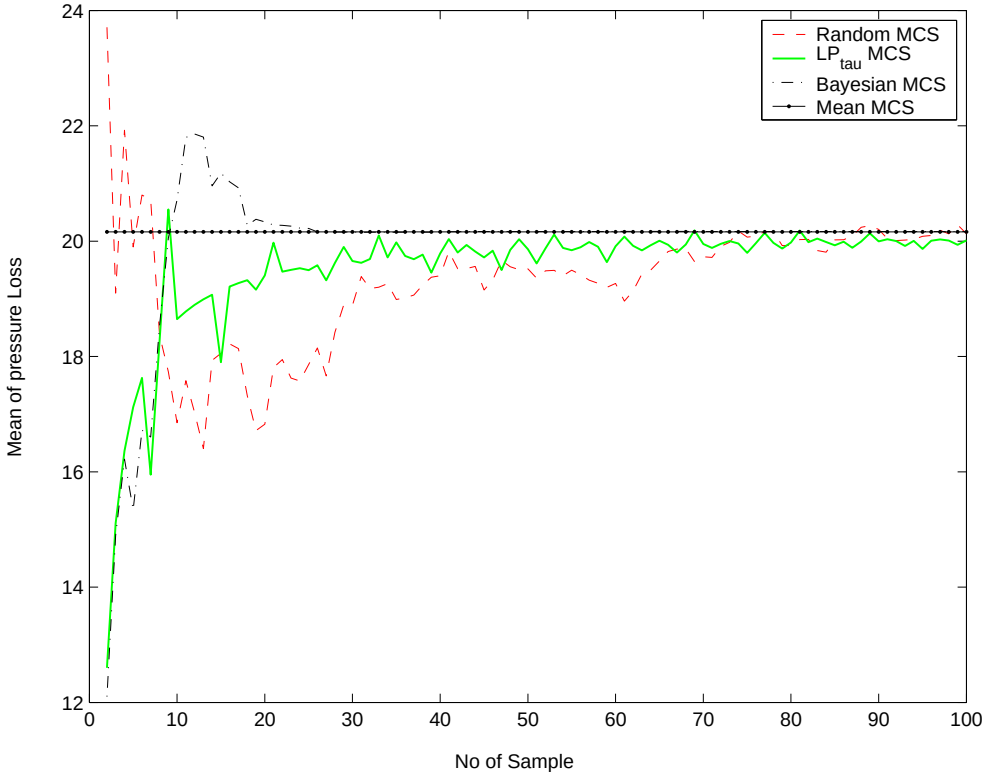


FIGURE 5.2: Convergence study for predicting the mean of the Rastrigin function. The horizontal line is the benchmark result for mean

that BMCS can predict statistics of the response quantity with far fewer samples. It should be noted that, for the BMCS study we had to run the surrogate model 10,000 times at each step. This incurred extra computational cost as compared to the random and quasi MCS methods. However, the run time associated with the surrogate model is very small. This argument becomes very apparent when the underlying relationship between the inputs and outputs is given by a high fidelity analysis code like CFD. Hence, one would prefer many more evaluations of the surrogate as compared to running the computationally expensive CFD simulations many times over. Hence, this study suggests far better convergence and computational savings when using Bayesian MCS to predict the output statistics.

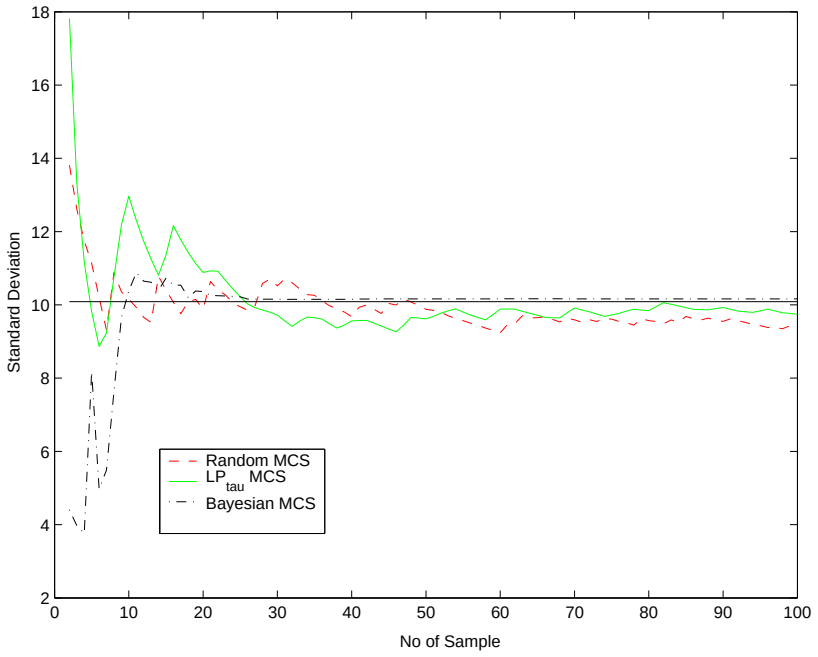


FIGURE 5.3: Convergence study for predicting standard deviation of the Rastrigin function. The horizontal line is the benchmark result for standard deviation

5.3 Application: Manufacturing Uncertainty in Compressor Blade

In Chapter 3 we proposed a method for modeling manufacturing uncertainty in compressor blades. We also showed that manufacturing uncertainty can significantly deteriorate the aerodynamic performance of a compressor blade. In this section we will further build a case for robust design by conducting a surrogate model based probabilistic analysis. We will employ the ideas presented earlier in this chapter (BMCS) to efficiently estimate the performance statistics. Firstly, we train a Gaussian Stochastic Process emulator to an initial candidate dataset. This surrogate model is validated using the methods discussed in section 4.2.4. The next step is to conduct a BMCS based probabilistic analysis which is followed by the main effect studies and sensitivity analysis.

5.3.1 Surrogate Model Training and Validation

We employ the BMCS method to estimate the effect of manufacturing uncertainty on the aerodynamic performance of a baseline design. A set of 100 points in vicinity of the baseline blade (i.e. the noise space defined by process capability data) is first generated using the LP_τ technique. CFD analysis is carried out at these points to create an initial training dataset.

The Gaussian Stochastic process emulator is trained to this initial dataset using the steps described in Chapter 4. Once we have tuned the hyperparameters, the next task at hand is to validate the surrogate model.

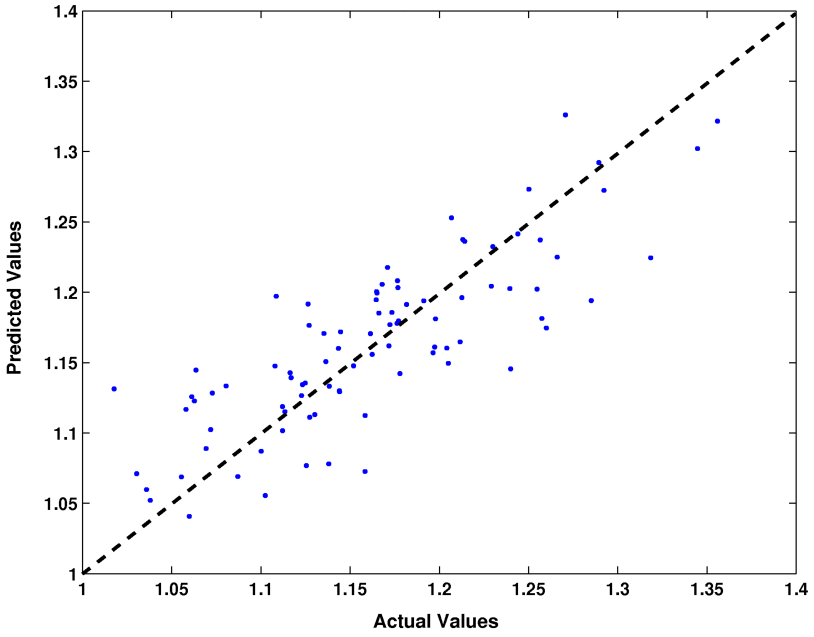


FIGURE 5.4: Predicted posterior mean versus original values for initial emulator for modeling manufacturing variations

The quality of the surrogate model can be assessed by a leave-one-out type cross-validation procedure. As shown in previous chapter, this method involves leaving the ith training point out and computing the posterior mean at the ith point. The computed values using the surrogate are plotted against the original values from the training dataset in figure 5.4. The regression coefficient for a linear model fit to the leave-one-out results is $R^2 = 0.916$. A Standardized Cross-Validated Residual test is also performed over the surrogate model. Figure 5.5 show the plot of posterior mean and SCVR_i values predicted by the leave-one-out method. It can be observed that most of the SCVR_i in Fig. 5.5 lie within the range $[-3\sigma, +3\sigma]$ and hence the error bars computed using the surrogate are reasonably tight.

5.3.2 Probabilistic Analysis

A normal distribution for the noise variables is assumed with the tolerance limits assigned to lie at $[-6\sigma, +6\sigma]$, obtained from the process capability data. The histogram of the pressure loss is shown in figure 5.6. A total of 100,000 samples were used in conjunction with the

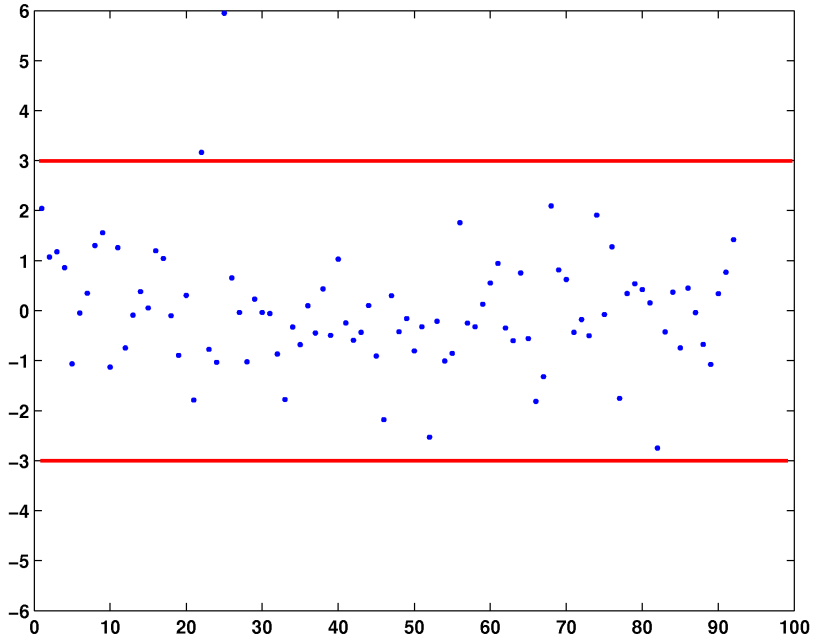


FIGURE 5.5: $SCVR_i$ values using Leave-One-Out validation for initial emulator for modeling manufacturing variations

Gaussian process emulator to generate this figure. Uncertainty analysis using the Gaussian process emulator takes around 13 seconds on an Intel(R) Xeon(TM) 3.06GHz processor. This represents a substantial saving in computational time as compared to using the high-fidelity CFD model for probabilistic studies, which would have taken approximately 2,000,000 minutes on the machine. Note here that if only the first two statistical moments are of interest (which is the case in our robust design formulation), then the computational cost is less than a second, after the covariance function of the Gaussian process prior has been tuned for the baseline dataset. It can be seen from figure 5.6 that manufacturing variations can lead to significant deterioration in the aerodynamic performance of a blade. In the worst case there can be upto 14% degradation in pressure loss coefficient. It also suggests an almost 4% shift in mean performance from the nominal performance. These figures re-emphasize the need for robust design of compressor blades against manufacturing variations.

5.3.3 Main Effects and Sensitivity Analysis

One of the important parts of a robustness analysis is to understand the effect of each variable on the performance. Main effect studies, as discussed in section 4.3.2, were carried out by numerically integrating out the effect of other variables. The main effect plots for the ten

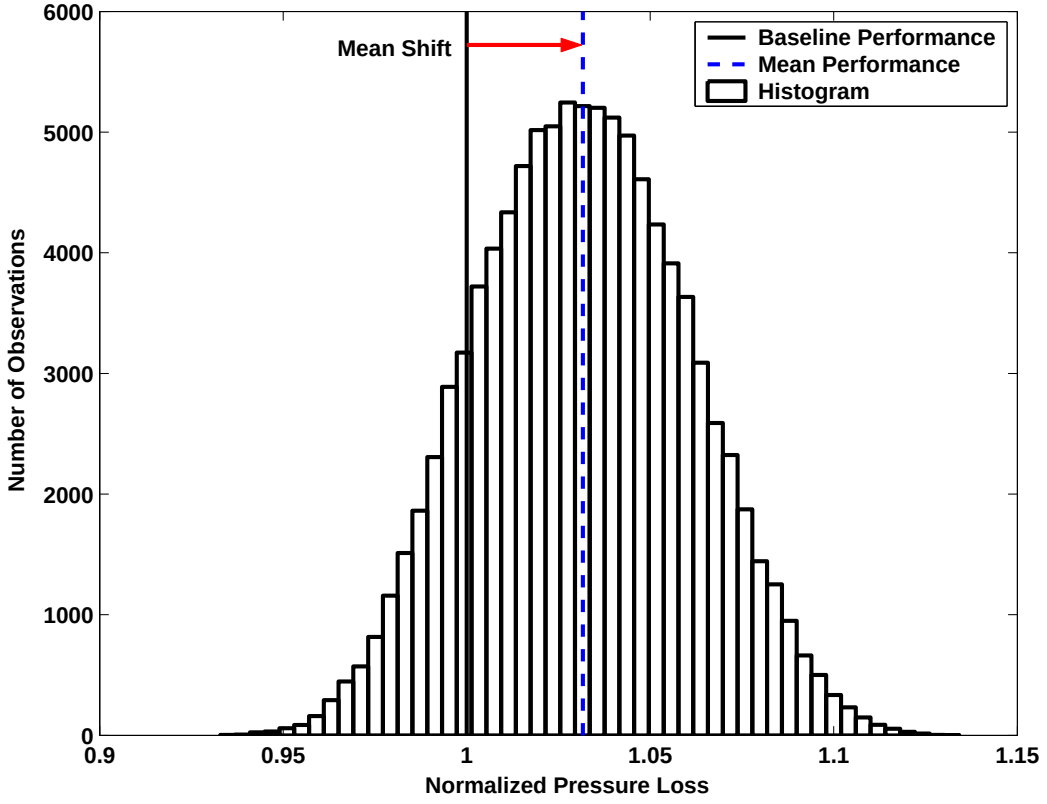


FIGURE 5.6: Histogram of performance of baseline geometry with manufacturing uncertainty

variables are shown in Figure 5.7. $W_i \mid i = 1, 2, ..., 10$ refer to all the variables and we plot the estimated effect and the 95% confidence intervals. Close inspection of these plots reveal that variables W_2, W_4, W_5, W_7 and W_8 have a larger effect on the performance. However, it is very difficult to understand the relative importance of each variable from these plots. Also it does not provide us any metric to rank variables. In such cases we are limited to visual comparison between many plots. It is also limited by the fact that it does not give any insight in to the interaction effects. Here we will present a more precise method for understanding the effect of each variable on the performance output.

Sensitivity analysis has been widely proposed by researchers to quantify the effect of variables on model output. For a detailed review of existing sensitivity analysis methods the reader is referred to (Saltelli *et al.*, 1993; Sobol', 1993; Chan *et al.*, 1997). Sobol' (1993) proposed a general method for sensitivity analysis for nonlinear mathematical models. He also also presented global sensitivity indices using Monte Carlo estimates in (Sobol', 2001). The main idea in Sobols' method of global sensitivity indices is that underlying function in the analysis model can be decomposed into functions of increasing orders of variables. We adopt this idea for our problem, in our case the Gaussian emulator $\widehat{Y}(\boldsymbol{\xi})$ can be decomposed

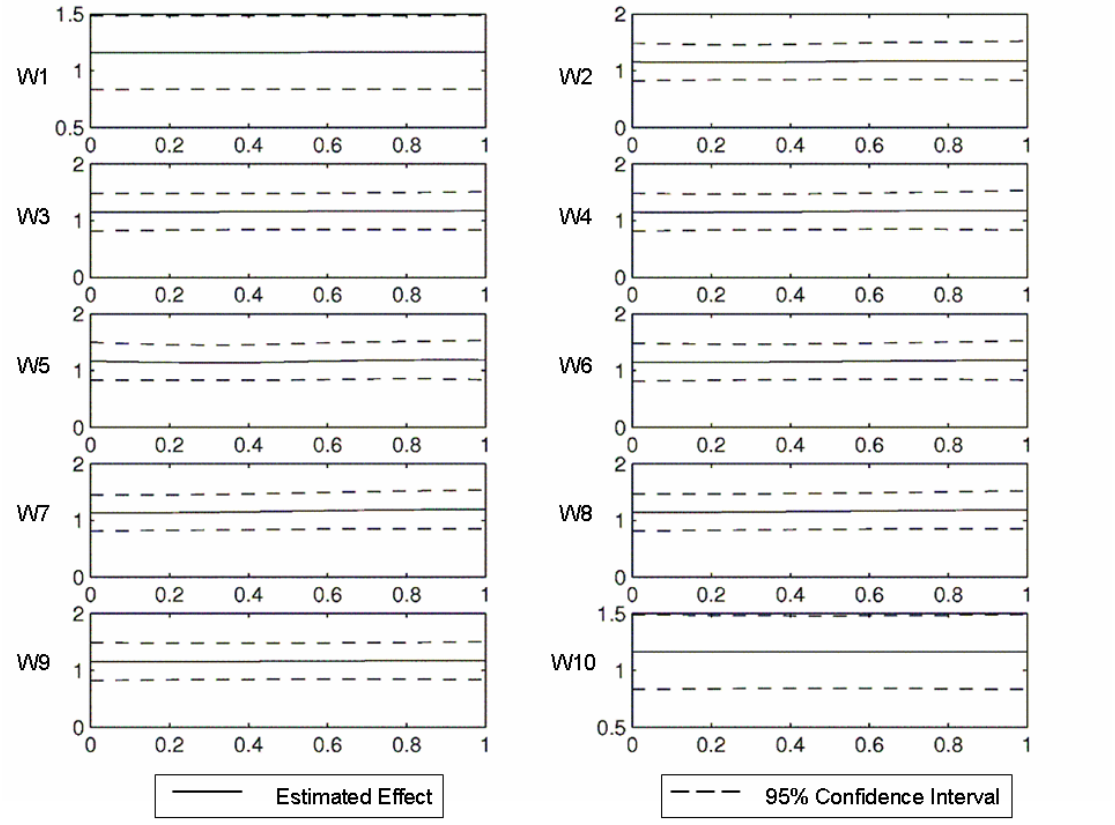


FIGURE 5.7: Main effect plot for all the design variables in the manufacturing uncertainty problem.

as

$$\widehat{Y}(\xi^{(1)},...,\xi^{(q)}) = \widehat{Y}_o + \sum_{i=1}^{q} \widehat{Y}_i(\xi^{(i)}) + \sum_{i=1}^{q}\sum_{j=i+1}^{q} \widehat{Y}_{ij}(\xi^{(i)},\xi^{(j)}) + ... + \widehat{Y}_{1,2,...,q}(\xi^{(i)},...,\xi^{(q)}). \quad (5.9)$$

where, $\widehat{Y}_o$ is a constant and q is the number of variables (noise variables) in this case. It can be seen that total number of summands will be 2^q. Equation (5.9) can also be expressed compactly as

$$\widehat{Y}(\boldsymbol{\xi}) = \widehat{Y}_o + \sum_{s=1}^{q} \sum_{i_1<...<i_s}^{q} \widehat{Y}_{i_1...i_s}(\xi^{(i_1)},...,\xi^{(i_s)}), \quad (5.10)$$

where $1 \leq i_1 < ... < i_s \leq q$. For equation 5.9 to be unique, the integral of every summand over any of its own variables must vanish, i.e.

$$\int_0^1 \widehat{Y}_{i_1...i_s}(\xi^{(i_1)},...,\xi^{(i_s)})\mathrm{d}\xi^{(k)} = 0 \quad \text{for} \quad k = i_1,...,i_s. \quad (5.11)$$

Equation (5.11) also enforces that all the summand in equation (5.9) are orthogonal and can be expressed as integrals of $\widehat{Y}(\xi)$; due to Sobol' (2001). The expression in equation (5.9) is also called Analysis of Variance (ANOVA) decomposition. If we assume that $\widehat{Y}(\xi)$

is square integrable, then all the $\widehat{Y}_{i_1 \ldots i_s}$ in equation (5.9) are square integrable in the noise space ($\xi \in \mathbb{R}^q$). Squaring the terms in equation (5.9) and integrating over the noise space we get

$$\int \widehat{Y}^2(\boldsymbol{\xi})\mathrm{d}\xi - \widehat{Y}_o^2 = \sum_{s=1}^{q} \sum_{i_1<\ldots<i_s}^{q} \int \widehat{Y}^2_{i_1 \ldots i_s} \mathrm{d}\xi_{i_1} \ldots \mathrm{d}\xi_{i_s}. \tag{5.12}$$

In this expression the constant terms are

$$D = \int \widehat{Y}^2 \mathrm{d}\boldsymbol{\xi} - \widehat{Y}_o^2, \quad \text{and} \quad D_{i_1 \ldots 1_s} = \int \widehat{Y}^2_{i_1 \ldots i_s} \mathrm{d}\xi_{i_1} \ldots \mathrm{d}\xi_{i_s}. \tag{5.13}$$

These terms are the variances in the sense that if the input $\boldsymbol{\xi}$ is random, the output $\widehat{Y}(\boldsymbol{\xi})$ is random. This in turn would imply from equation (5.9) that terms like $\widehat{Y}_{i_1 \ldots i_s}(\xi^{(i_1)}, \ldots, \xi^{(i_s)})$ are random. In such case their variance is given by D and $D_{i_1 \ldots 1_s}$. The global sensitivity indices are defined as

$$S_{i_1 \ldots i_s} = \frac{D_{i_1 \ldots i_s}}{D}. \tag{5.14}$$

In equation (5.14), s is referred to as the order of the sensitivity index. The global sensitivity indices can be employed for ranking of variables and fixing the unessential variables in $\widehat{Y}(\xi^{(1)}, \ldots, \xi^{(q)})$. They can also be used in dropping higher order members in equation (5.9) to conduct a more detailed sensitivity analysis over other terms.

In this study we carry out a first order sensitivity analysis. We evaluate the terms $D_1, \ldots . D_q$ using MCS estimates for the respective integrals. The first step in this method is similar to the method employed in main effect analysis in section 4.3.2. We first numerically solve the following equation:

$$m(\xi^{(i)}) = \int \widehat{Y}(\xi^{(1)}, \ldots, \xi^{(q)}) \prod_{j=1; j\neq i}^{q} \mathrm{d}\xi^{(j)}. \tag{5.15}$$

We employ a 10,000 points LP_τ based MCS in $q-1$ dimensions to estimate $m(\xi^{(i)})$. Once we integrate out the effect of all variables *sans* the i^{th} variable, we conduct another LP_τ based MCS over the i^{th} variable to estimate the variance of $m(\xi^{(i)})$, which is in turn the value of D_i. Figure 5.8 shows the plot of the indices $\frac{D_i}{\sum_{i=1}^{q} D_i}$. In our example the variables $[\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(q)}]$ are the weights of the shape function used in parameterizing the blade geometry and are given by $[W_1, W_2, \ldots, W_{10}]$. The parametrization technique and the effect of the weights variables on the geometry was explained in section 3.4. Recall that, each variable has its maximum influence near the region corresponding to the location of its maxima given by x_{M_i}; see equation (3.4). The pie chart in figure 5.8 shows that the variables W_1 and W_{10} also have less effect on the performance output. As expected, these variables

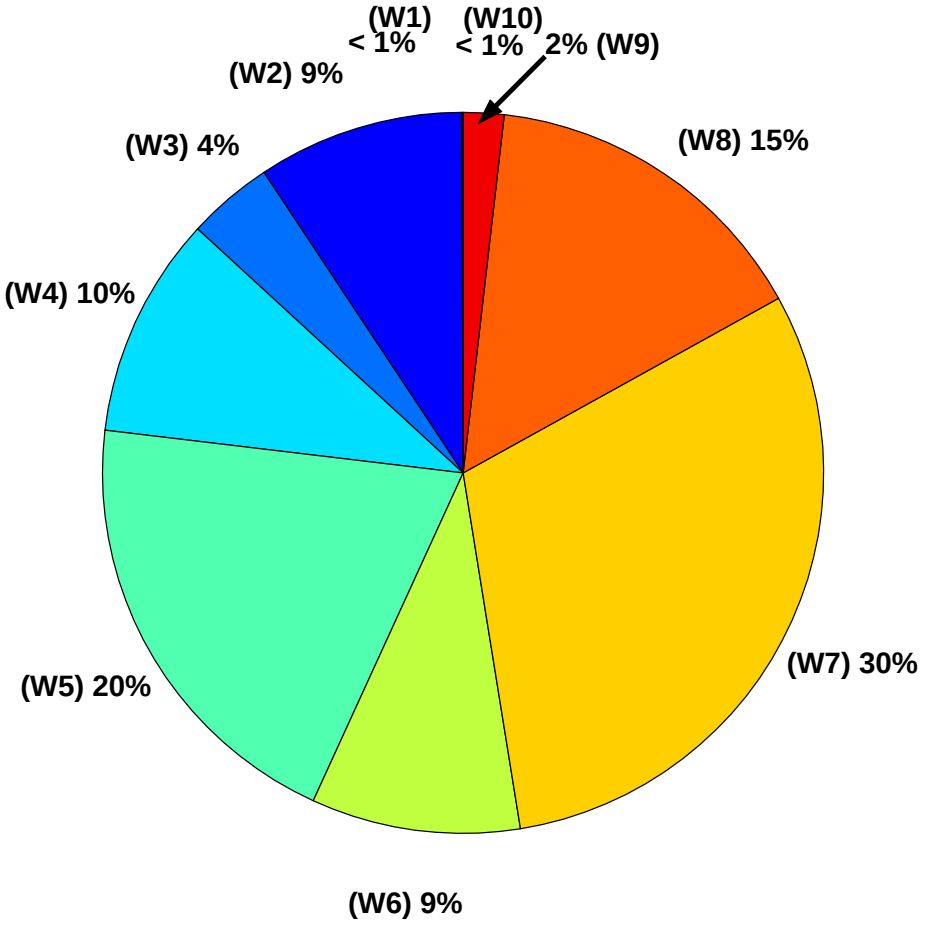


FIGURE 5.8: Graphical representation using pie chart of the 1st order sensitivity analysis

cause shape changes near the trailing edge and should not influence pressure loss much. The variables W_4, W_5, W_6, W_7 and W_8 have the maximum effect on the aerodynamic performance.

Table 5.1 rank the variables with their first order sensitivity indices. It also shows the location of the maximum effect of the variable as percentage of chord. The term PS along with the location stands for pressure surface and SS stands for suction surface. It can be noted from the table that the variables that have higher ranking of sensitivity index are in general those that cause shape changes near the leading edge and in turn are expected to have high influence on the aerodynamic performance. Figure 3.14 shows the location of the maximum influence point due to each variable on the blade shape. One would have expected W_5 to have the maximum impact on the output performance as it is effects the leading edge the most. It is interesting to note that W_7 and W_8, which effects the pressure side of the blade near the trailing edge, have a higher first order sensitivity index as compared to not only W_5, but also W_4 and W_6 which are closer to the leading edge. This is consistent with

Rank of Variables	Variable Name	Sensitivity Index (as %) $\frac{D_i}{\sum_{i=1}^{q} D_i}$	Location of Peak from LE
1	W7	30.391	20% (PS)
2	W5	20.354	0% (SS)
3	W8	14.934	60% (PS)
4	W4	9.8952	10% (SS)
5	W6	9.3935	10% (PS)
6	W2	9.2958	50% (SS)
7	W3	3.7892	30% (SS)
8	W9	1.8732	60% (PS)
9	W1	0.0707	70% (SS)
10	W10	0.0033	80% (PS)

TABLE 5.1: Ranking of first order sensitivity indices for the variables with their maximum effect location

the many observations made in transonic airfoil design optimization, where designers find that this region near the trailing edge on the pressure surface of the airfoil influences the aerodynamic performance (Huyse *et al.*, 2002; Li *et al.*, 2006; Zingg & Elias, 2006)

However, there may be high interaction effects present in our model. From equations (5.12) and (5.13) we have

$$D = \sum_{s=1}^{q} \sum_{i_1<\ldots<i_s}^{q} D_{i_1 \ldots 1_s}. \tag{5.16}$$

From this we can evaluate the interaction effects as the difference between the total global variance D and the sum of the first order variances given by $\sum_{i=1}^{q} D_i$. The total global variance D is evaluated using a MCS over all the variables. Figure 5.9 shows the plot of these indices given by $\frac{D_i}{D}$ and the sensitivity index of the interaction effects given by $\frac{D-\sum_{i=1}^{q} D_i}{D}$. This plot shows that there is large impact due to interaction effects on the model output. To understand the effect of interaction effects one can conduct a higher order sensitivity analysis. This can be implemented by using the method proposed above for different combinations of variables instead of just one variable. Once the variable combinations are selected a two step MCS simulation similar to the one conducted earlier in this section needs to be performed.

Higher order sensitivity calculations can be computationally demanding. One can use the

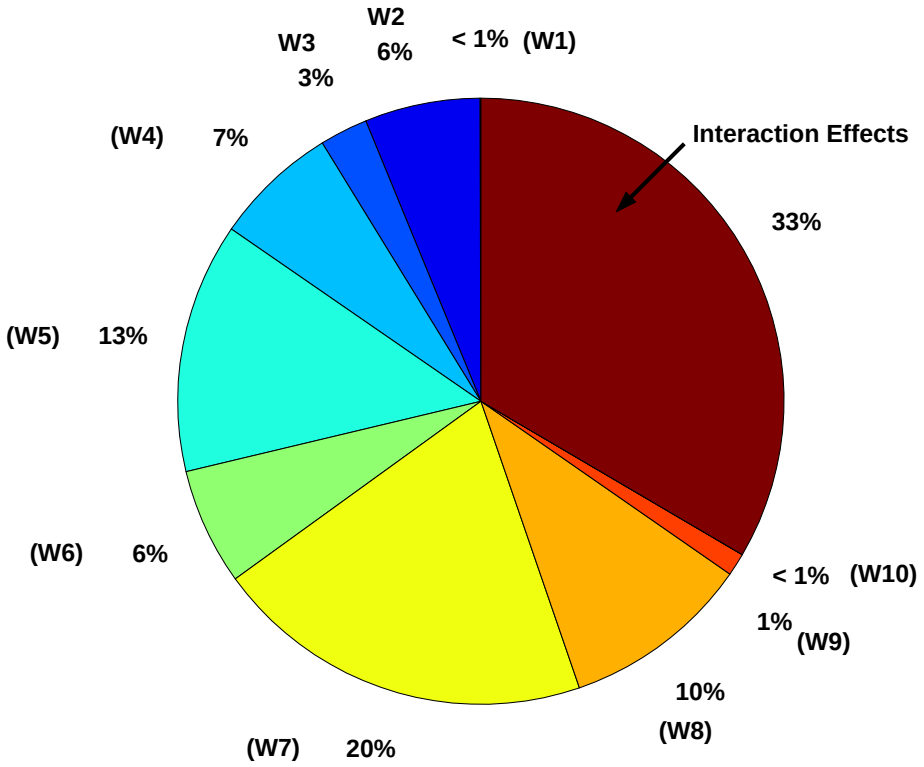


FIGURE 5.9: Pie chart of the 1st order sensitivity analysis with interaction effects

first order sensitivity analysis ranking as shown in Table 5.1 to drop variables with lower ranks. Again at each stage one can employ the idea in equation (5.16) to estimate higher order interaction effects. For example after one estimates the first and second order variance, the sum of these variances can be subtracted from the total variance D to estimate the effect of higher order effects. If this quantity is a small fraction of total variance further index calculations can be omitted. This can lead to reduction in computational effort for the user. The sensitivity index ranking in table 5.1 shows that variables such as $[W_1, W_3, W_9, W_{10}]$ have much less effect on model output. These terms can be dropped from further analysis. However, in our shape model these variables also contribute to maintaining smooth shape changes to the blade geometry, and thus are retained for robust design studies in the next section.

5.3.4 BMCS based Robust Design Method

The tools and methods discussed in earlier sections are employed next to build an efficient method for robust design. We call this method Bayesian Monte Carlo based Robust Design and the steps involved are shown in figure 5.10. In this method we start with an initial

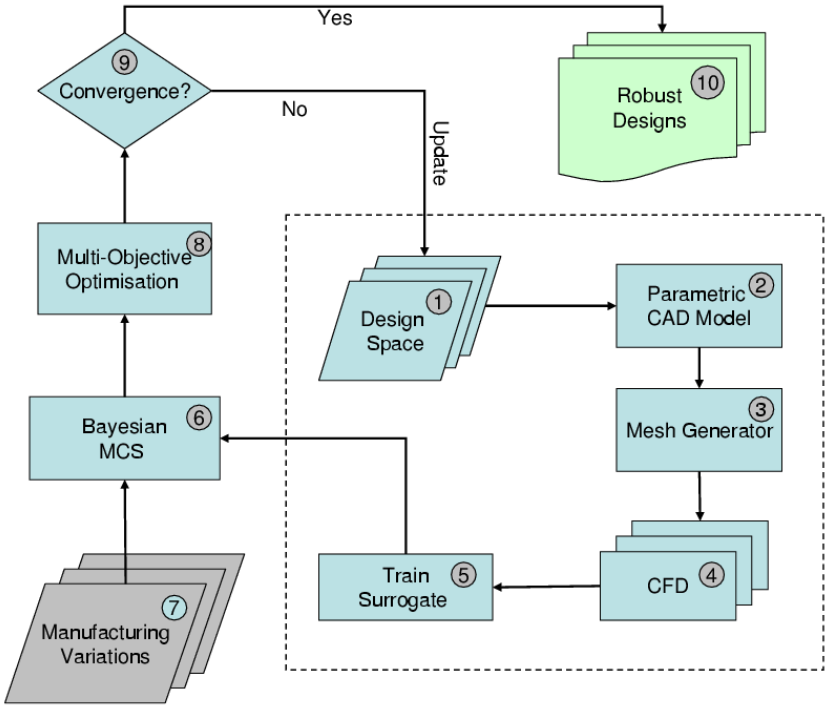


FIGURE 5.10: Flowchart for Robust Design of Compressor Blades against Manufacturing Uncertainty

combined array dataset $\mathcal{D}_n \in \tilde{\mathbf{x}} \in \mathbb{R}^{p+q}$, generated using the LHS technique. The choice of a combined array is intuitive for the problem under consideration since the noise factors (manufacturing variations) are a subset of the design factors (design space), i.e. $\mathcal{E} \in \chi$. The grid generator and CFD solver are used to generate the initial dataset $[\mathbf{X}_n, \mathcal{Y}_n]$. The dataset is employed for tuning the covariance function as discussed in section 3.4. The BMCS method is employed, using the emulator obtained from the previous step, to calculate the mean and standard deviation of the pressure loss. This is followed by multiobjective optimization to seek the Pareto optimal set. The CFD analysis is performed on points on the Pareto front and these new points are appended to the existing dataset, which is subsequently used to update the baseline Gaussian process emulator. This process is repeated till we meet some predefined convergence criteria.

The above mentioned approach alleviates most of the concerns expressed regarding existing robust design strategies discussed earlier. Specifically, the present approach uses a computationally cheap emulator to improve the efficiency of robust design search. The method

also provides the option of updates which ensures that the designer spends effort in the area of interest (near the Pareto front) and the emulator prediction improves with each iteration. We will illustrate this improvement in the numerical studies section. In the next section, we discuss the results obtained by applying the proposed method to a compressor blade design problem.

5.4 Numerical Studies and Results

We next conduct a robust design optimization study for a compressor blade. We employ the Bayesian Monte Carlo based method proposed in the flowchart shown in figure 5.10. To start with, we use the Gaussian process emulator which was trained to initial dataset of 100 points. NSGA-II is then employed in conjunction with this emulator to search the entire design space. BMCS is carried out for all points in the population at each generation to evaluate the performance statistics (design objectives). The objectives are then ranked to obtain Pareto-optimal solutions. We use a population size of 100 points with 100 generations for this study.

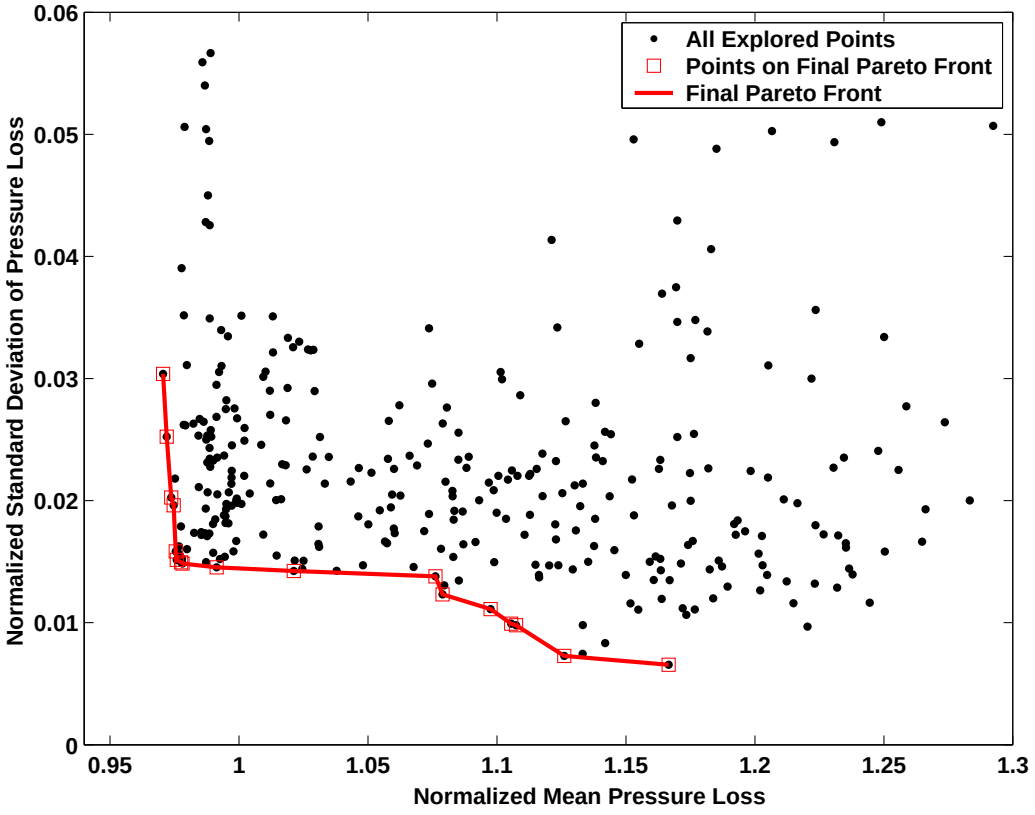


FIGURE 5.11: Final Pareto front with all explored points

A low-crowding algorithm, which maximizes the Euclidean distance between the Pareto points, is used to select points which are then verified by running full scale CFD simulations Kumar *et al.* (2006). These points are updated to the existing dataset and the Gaussian process emulator is re-trained on the new dataset. The Gaussian emulator based NSGA-II searches are repeated for updates until a convergence criterion is met. Here the convergence criterion is a function of the improvement made in predicting the Pareto front after each update. When the observed improvement is less than a user defined value the search is terminated. Figure 5.11 contains the initial dataset and subsequent updates points which are collectively shown as the explored points. The figure also shows the final Pareto Front after ten updates. Note that exact aerodynamic analysis using the high fidelity CFD code was conducted only at the 336 points shown in the figure.

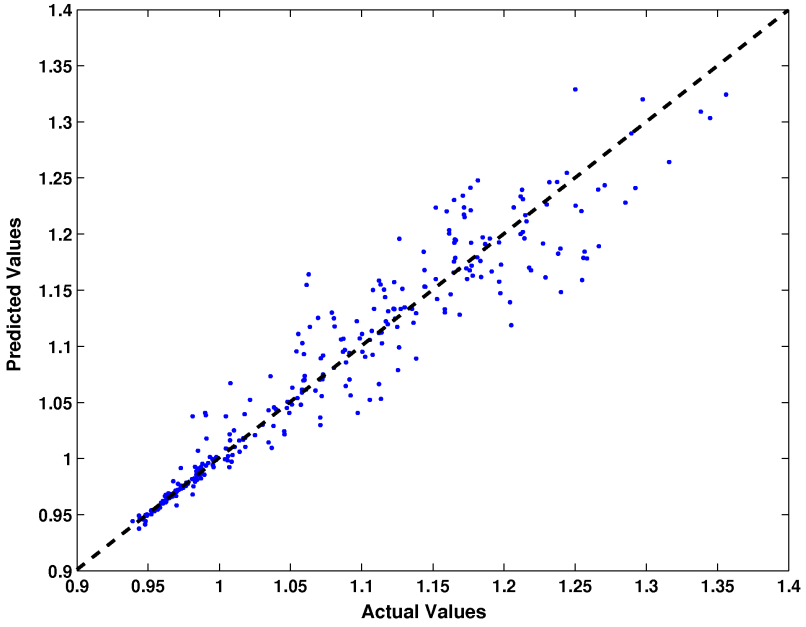


FIGURE 5.12: Predicted posterior mean versus original values for final emulator for modeling manufacturing variations

We had mentioned earlier that at each iteration the surrogate model is updated with CFD solutions at the points selected from the Pareto front. Hence, at the final iteration we had 336 points for training the Gaussian emulator. We conducted model validation on the final surrogate model to illustrate improvements from the initial surrogate model and build confidence in our predictions. Figure 5.12 shows the plot of predicted values versus actual CFD values using the leave-one-out validation test. The regression coefficient for a linear

model fit to the leave-one-out results is $R^2 = 0.978$, which has considerably improved as compared to $R^2 = 0.916$ for the initial dataset. An SCVR test was also conducted for the final surrogate model. Figure 5.13 shows the SCVR plot. It can be observed that most of the points lie between $+1\sigma$ and -1σ. Though there are a few outliers this suggests a good model fit. Figure 5.14 compares the initial Pareto front with the final converged front. It

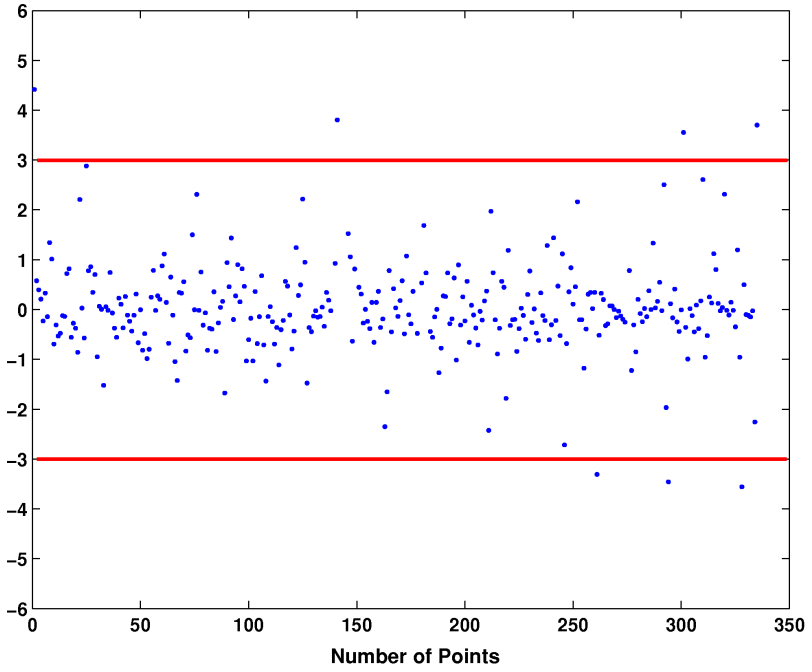


FIGURE 5.13: $SCVR_i$ values using Leave-One-Out validation for final emulator for modeling manufacturing variations

can be noted that significant improvement in the Pareto front is obtained.

5.4.1 Comparison with Deterministic Design

We compare the robust blades with an optimal geometry obtained by minimizing the nominal performance to understand the trade-offs obtained. To conduct this comparison, we perform a standard deterministic optimization study. Deterministic design methods seek to optimize the nominal performance of the system, i.e., to optimize blade geometries simply for low pressure loss coefficients. In order to ensure computational efficiency, the search is performed in conjunction with the Gaussian process emulator constructed using the initial dataset. A robust design from the final Pareto Set is selected for comparison with the deterministic optimal design. The selected robust design is highlighted in figure 5.14.

To start with, 50 points are sampled in the noise space around each blade and, the

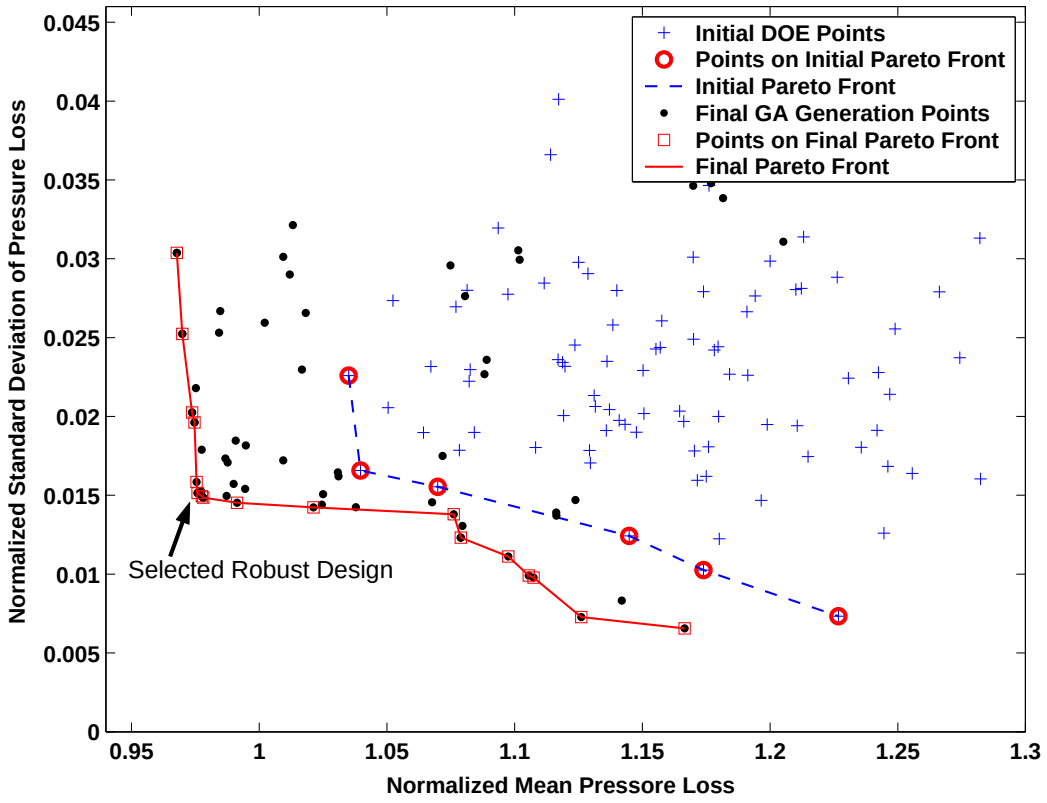


FIGURE 5.14: Improvement in Pareto Front from initial Pareto Front

high-fidelity CFD solver is executed to create two new datasets. These datasets are then appended to the final updated dataset for training Gaussian emulators. Finally, probabilistic performance analysis for the robust and the deterministically optimal blade is conducted by employing the respective emulators. Figure 5.15 compares the histogram of the pressure loss coefficient of the robust blade with that of the deterministically optimal blade. This figure was generated by sampling the respective Gaussian process emulator (100,000 samples) in the noise space around the robust and the deterministically optimal blades. As expected, the histogram of the robust geometry shows less variability in pressure loss coefficient as compared to the design obtained using a deterministic approach. Table 5.2 shows the comparison between the *Nominal*, *Mean* and *Worst Case* performance of the baseline, deterministic and robust design blades. It also shows the respective *Standard Deviation* and *Mean Shift* predicted using the above mentioned method. There is a considerably lower shift in the mean performance of 1.05% from the nominal performance for the robust blade geometry as compared to almost 3.59% for the deterministic optimal blade geometry, see Table 5.2. The standard deviation of the robust blade [$\sigma = 0.0231$] is significantly less than that of the deterministically optimal blade [$\sigma = 0.0335$]. It can be observed that low variability has

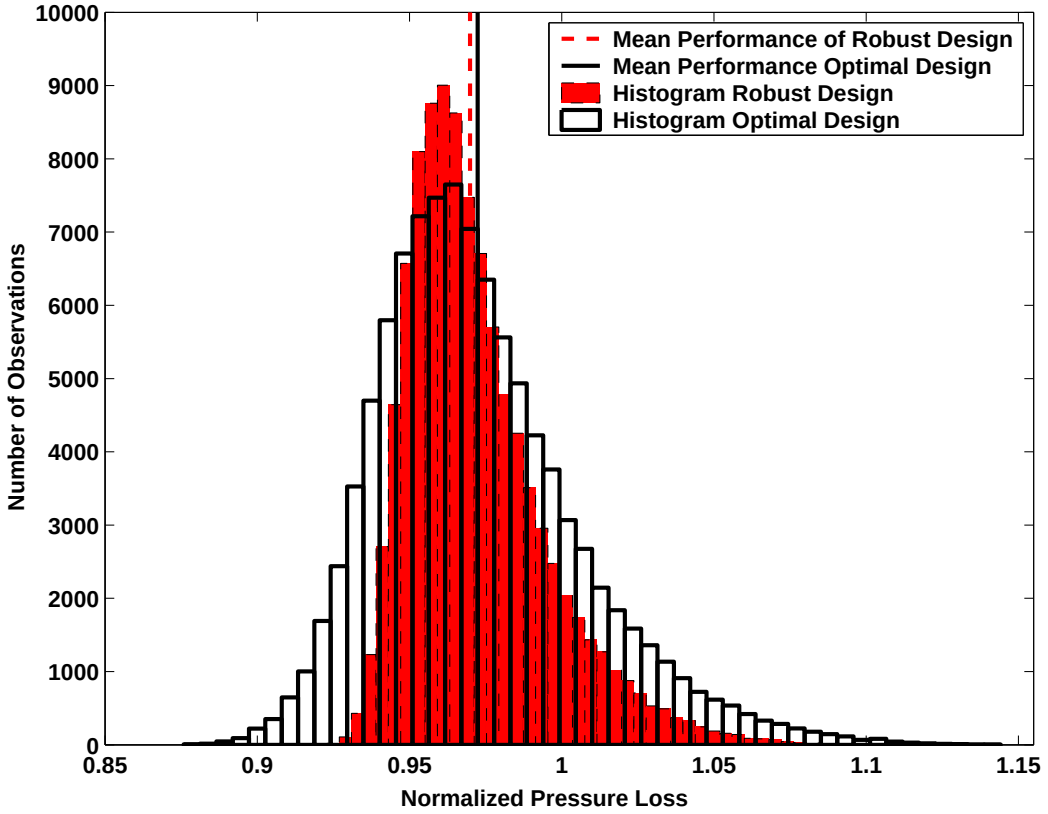


FIGURE 5.15: Histograms for Comparison between Robust Design and Deterministic Optimal Design

Blade Designs	Nominal Performance	Mean Performance	Standard Deviation	Mean Shift	Worst Case Performance
Baseline	1.0	1.0317	0.0295	3.17%	1.1461
Deterministic	0.9387	0.9724	0.0335	3.59%	1.1442
Robust	0.9610	0.9711	0.0231	1.05%	1.1124

TABLE 5.2: Comparison between the Baseline, Deterministic and Robust Design

been achieved in the robust design at the expense of a marginal reduction in the nominal performance. However, the mean performance of the robust blade [$\mu = 0.9711$] is better than the mean performance for the deterministically optimal blade [$\mu = 0.9724$]. The worst case performance of the robust blade geometry is also almost 3% better than the deterministic optimal blade geometry. Clearly the robust blade is better in all measures than the baseline geometry, except in nominal performance. Figure 5.16 shows the zoomed in view of the baseline, deterministic optimal and robust design.

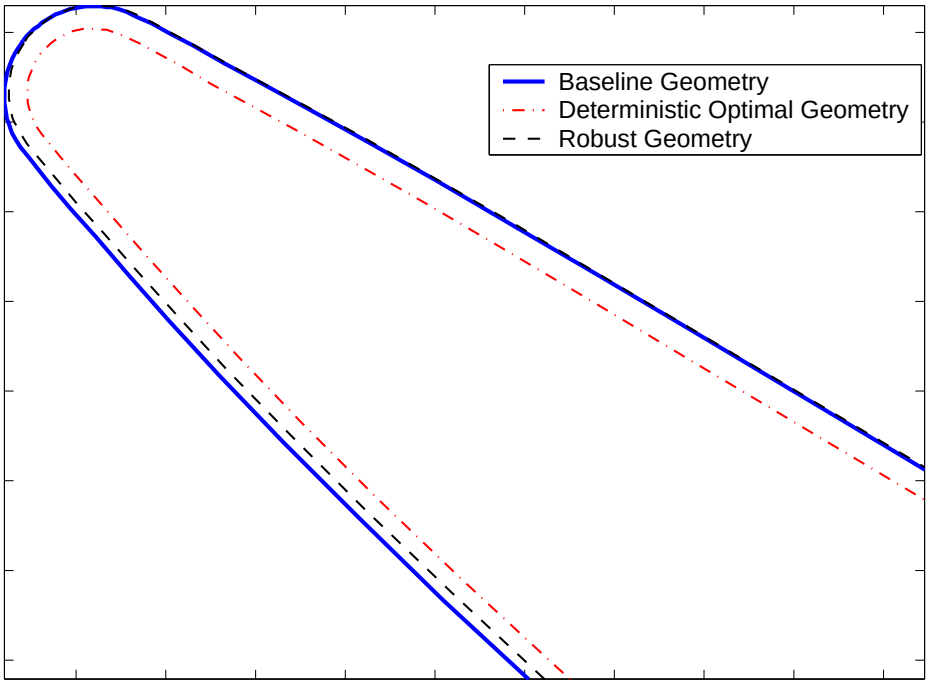


FIGURE 5.16: Zoomed in view of Baseline, Deterministic and Robust Design Blades

5.5 Summary

In this chapter a Bayesian Monte Carlo Simulation method was presented for efficient estimation of statistics of output. The limitations of existing pseudo and quasi MCS methods were presented and an argument for using the BMCS method for robust design studies was presented. The method was shown to be more efficient than existing methods. This idea was further employed to propose a Bayesian Monte Carlo based robust design methodology. The proposed method also employed a multiobjective robust design formulation to ensure explicit trade-offs between mean and standard deviation of the performance. Gaussian process emulators were used to construct computational cheap surrogate of the high-fidelity CFD

simulations and for carrying out BMCS to estimate the effect of manufacturing uncertainty on the aerodynamic performance of compressor blades. Main effect and sensitivity analysis for understanding the effect of variables were conducted. The sensitivity analysis provided us with a metric to rank the variables according to their influence on the aerodynamic performance of the blade. It also gave us an insight into the interaction effects. Subsequently, multiobjective genetic algorithm based robust design studies were conducted to seek the Pareto optimal solutions.

A robust design from the final Pareto set was selected for comparison with a deterministically optimal blade design. BMCS was carried out for the selected blades and the robust design was found to be considerably less sensitive to manufacturing variations as compared to the deterministic optimal design. The robust design was also found to have better performance than deterministic optimal design in all respects (Mean, Standard Deviation, Mean Shift, Worst Case Performance), except the nominal performance. Significant computational savings for the robust design studies gained by employing efficient Gaussian process emulators instead of high fidelity CFD simulations were also reported.

One issue that we faced while using the proposed robust design methodology in this study was the density (diversity) of points on the Pareto Front. The reason is that the multiobjective optimizers seeks the Pareto front using the surrogate model (Gaussian emulator). Even after many updates the model is not 100% accurate. This leads to errors in the prediction of points on the pareto front. Consequently, points when verified by CFD solutions tend to move away from the predicted Pareto front. In the next chapter we will discuss and propose a method to alleviate this problem.

Chapter 6

Probabilistic Ranking Based Multiobjective Robust Design

In the previous chapters we employed surrogate models based on Gaussian emulators to expedite the multiobjective optimization process. Surrogate models have been extensively used to reduce computation cost when using optimization methods with high fidelity analysis codes. However, these approximate models may have poor predictability in certain regions of the design space which could mislead the optimization process. Many researchers have proposed methods to manage this issue in single objective optimization. Some widely used methods are based on the concepts of *Probability of Improvement*, *Expected Improvement Criteria* and the *Trust Region approach* (Dennis & Torczon, 1997; Jones *et al.*, 1998; Alexandrov *et al.*, 1998; Jin *et al.*, 2003; Ong *et al.*, 2003; Keane, 2003a; Keane & Nair, 2005). However, very few methods exist for improvement of surrogate based multiobjective optimization.

A commonly used method to manage this concern in multiobjective optimization, is to update the surrogate model during the optimization search. The idea is to select points from the approximate Pareto set predicted by the surrogate and execute the high fidelity analysis model at these points to create a new dataset. This new dataset is appended to the initial dataset and the surrogate is retrained. We employed this idea in the earlier chapters with some success. However, we observed two major issues (1) wasting computational effort in updating regions which may not be of interest and (2) lack of diversity in the predicted approximate Pareto set.

Recently some methods have been proposed to alleviate these concerns for multiobjective problems. An efficient method, based on the concept of *Expected Improvement Criteria*,

to manage these issues in multiobjective optimization was proposed by Keane (2006). In this method a probabilistic metric (probability of improvement) allows trade-off between exploring the design space and seeking the Pareto optimal set. The major advantage of this method is that multiobjective problems can be solved using existing single objective optimization methods (e.g. gradient based methods) without any need of weight factors. Emmerich *et al.* (2006) used the information of error in the approximate model to expedite the multiobjective optimization process. They used the confidence intervals provided by Gaussian emulator based surrogate models to deduce the *lower confidence bound* of the predicted value. They also proposed other metrics based on *Probability of Improvement* and *Expected Improvement Criteria*, and showed that these methods perform better than direct surrogate based optimization methods.

In this chapter, we propose a generic method for efficient multiobjective optimization with surrogate models. The proposed method is based on a probabilistic framework and ensures that the uncertainty in predictions are accounted for during the optimization search. For the sake of clarity, we first discuss in detail the concepts of non-dominated sorting and Pareto optimality which are central to multiobjective optimization algorithms. We then extend these definitions for dealing with models with uncertainty. We propose a probabilistic multiobjective optimization algorithm and implement it using an existing multiobjective optimization method (NSGA-II). We apply this tool to a suite of test problems based on complex functions whose true Pareto front is known apriori. We compare the proposed methodology with optimization methods which directly search the true function and a simple Gaussian emulator assisted search algorithm. Finally, we present a generic method for efficient robust design with surrogates and revisit the erosion and manufacturing uncertainty problems.

6.1 Introduction

Many engineering problems involve more than one objective which more often than not compete with each other. Multiobjective optimization involves simultaneous optimization of these objectives. Let $\mathbf{x} = [x_1, x_2, ..., x_m] \in \mathbb{R}^m$ be a vector of m design variables and the n objectives be given by $\mathbf{f} = [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_n(\mathbf{x})] \in \mathbb{R}^n$. Then the multiobjective minimization problem can be expressed as seek $\mathbf{x}^* = [x_1^*, x_2^*, ..., x_m^*]$ to

$$\underset{\mathbf{x}}{\text{minimize}} \quad \{f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_n(\mathbf{x})\}. \tag{6.1}$$

For the sake of simplicity we will consider unconstrained optimization problems through out this chapter. Without any loss of generality, we present our definitions for a minimization problem in which the aim is to minimize all the objectives simultaneously. These ideas can be easily extended to maximization problems or combinations of minimization and maximization problems.

Over the last few decades, several techniques have been employed to solve the nonlinear optimization problem given by equation (6.1). The simplest approach is to use a combination of all the objectives to get a single objective. The major problem with this approach is in deciding the weights assigned to each objective function. Without any prior information about the behaviour and importance of each objective, it is not very clear on how to decide these weights. The most common approach in this class is the weighted sum approach. This method consists of adding all the objective function with different weight combinations. The new single objective problem can be expressed as

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \sum_{i}^{n} w_i f_i(\mathbf{x}), \tag{6.2}$$

where $\sum_i^n w_i = 1$ and $w_i \geq 0$. w_i are the weight coefficients representing the relative importance of each objective. Equation (6.2) is solved for different combinations of w_i to seek the optimal solutions which can be then presented to the designer for selection.

The principles of multiobjective optimization are different from that of single objective optimization. In single objective optimization problems, the goal is to find the optimal or best design, which corresponds to the minimum or maximum value of the objective function. However, in multiobjective optimization, there may not be any single optimal solution. This is the case when there is a *conflict* between the objectives, i.e. an improvement in one objective may lead to worsening of other objectives. In such cases, one seeks the compromised solutions which are also known as the Pareto Optimal Solutions. There exist numerous methods in literature to solve the multiobjective optimization; for an extensive review of such methods the reader is referred to (Fonseca & Fleming, 1993, 1995; Deb, 1995; Keane & Nair, 2005). The idea of dominance and Pareto optimality are central to multiobjective optimization methods. Next we present these concepts and ideas in more detail

6.2 Pareto Optimality: Concepts and Definitions

The concept of Pareto optimality was first formulated in the area of economics in the early 19^{th} century by Vilferdo Pareto (Pareto, 1896). The definition of the Pareto optimum is as

follows

Definition 1 (Pareto Optimum)

Let $\hat{\mathbf{x}}, \mathbf{x}^* \in \mathbb{R}^m$. *Then, for a minimization problem,* $\mathbf{x}^*$ *is Pareto optimal if* $\forall \ \hat{\mathbf{x}} \neq \mathbf{x}^*$ *one of the following holds*

- $\forall i \in \{1, 2, \ldots, n\}: \quad f_i(\hat{\mathbf{x}}) = f_i(\mathbf{x}^*)$
- $\exists$ *at least one* $i \in \{1, 2, \ldots, n\}$ *such that* $f_i(\hat{\mathbf{x}}) > f_i(\mathbf{x}^*)$

According to the definition, the design vector $\mathbf{x}^*$ is Pareto optimal if there exists no feasible vector $\mathbf{x}$ which would decrease some objective without causing a simultaneous increase in at least one other objective. However, this definition more often than not leads to a set of solutions rather than a single solution. Such set is referred to the set of *non-dominated* solution. We define $\mathbf{f}^{(i)}$ as a vector of objectives given by $\mathbf{f}^{(i)} = [f_1^{(i)}, f_2^{(i)}, ..., f_n^{(i)}]$. Next we define the concept of dominance.

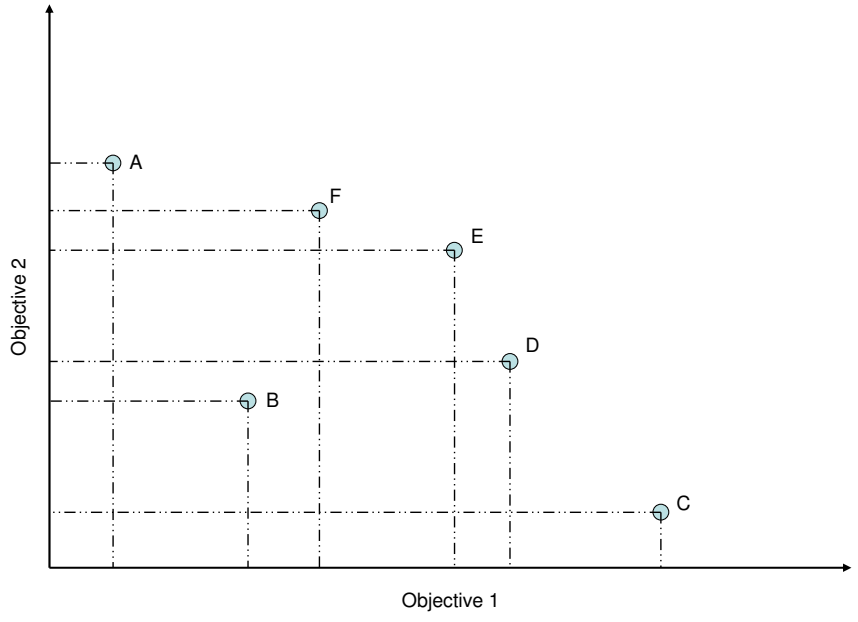


FIGURE 6.1: Concept of dominance using a bi-objective minimization problem

Definition 2 (Dominance)

Let $\mathbf{f}^{(1)}, \mathbf{f}^{(2)} \in \mathbb{R}^n$ *be two vectors of objectives. Then, for a minimization problem,* $\mathbf{f}^{(1)}$ *is said to dominate* $\mathbf{f}^{(2)}$ *(i.e.* $\mathbf{f}^{(1)} \succ \mathbf{f}^{(2)}$*), if both the following conditions are true*

- $\forall i \in \{1, 2, \ldots, n\}: \quad f_i^{(1)} \leq f_i^{(2)}$
- $\exists\ j \in \{1, 2, \ldots, n\}: \quad f_j^{(1)} < f_j^{(2)}$

This definition implies that if the solution $\mathbf{f}^{(1)}$ is said to dominate $\mathbf{f}^{(2)}$, then the solution $\mathbf{f}^{(1)}$ is no worse than $\mathbf{f}^{(2)}$ in all objectives and at the same time the solution $\mathbf{f}^{(1)}$ is strictly better than $\mathbf{f}^{(2)}$ in at least one objective. To illustrate this notion,we consider a bi-objective problem. Figure 6.1 shows a set of solutions for the bi-objective minimization problem. In figure 6.1 point B dominates points F, E and D as it is better than these points in both objectives. However, point B does not dominate point A. This follows from the definition of dominance, though point B is better than point A in Objective 1, it is worse than point A in Objective 2. Using the same argument point C is not dominated by point B and the points F, E and D are not dominated by points A and C. It should be noted that points A, B and C are not dominated by any other points. We now use the concept of dominance to define *Pareto Set* and the Pareto front.

Definition 3 (Pareto Set)

Let $\mathbf{F}, \mathbf{F}^* \subseteq \mathbb{R}^n$ *be a set of vectors. Then the Pareto set* $\mathbf{F}^*$ *of* $\mathbf{F}$ *is defined as follows:* $\mathbf{F}^*$ *contains all vectors* $\mathbf{f}^{(*)} \in \mathbf{F}$ *which are not dominated by any vector* $\mathbf{f} \in \mathbf{F}$*, i.e.*

- $\mathbf{F}^* := \{\mathbf{f}^{(*)} \in \mathbf{F} |\ \nexists\ \mathbf{f} \in \mathbf{F} : \mathbf{f} \succ \mathbf{f}^{(*)}\}$

It can be deduced from this definition that any vector $\mathbf{f} \in \mathbf{F}$ is dominated by at least one vector belonging to the Pareto set ($\mathbf{f}^{(*)} \in \mathbf{F}^*$). We illustrate this idea using a bi-objective minimization problem as shown in figure 6.2. According to the definition, all the points A, B and C are members of the Pareto set as they are not dominated by any other points shown in the figure. Note that points A and C do not dominate any of the other points B, F or D but are still members of the Pareto set. The Pareto front is an curve (e.g. a polynomial interpolant though points A, B and C as shown in figure 6.2) on which all points from the Pareto set lie. In the next section, we generalise these definitions in a probabilistic setting, addressing the concerns raised in the beginning of this chapter.

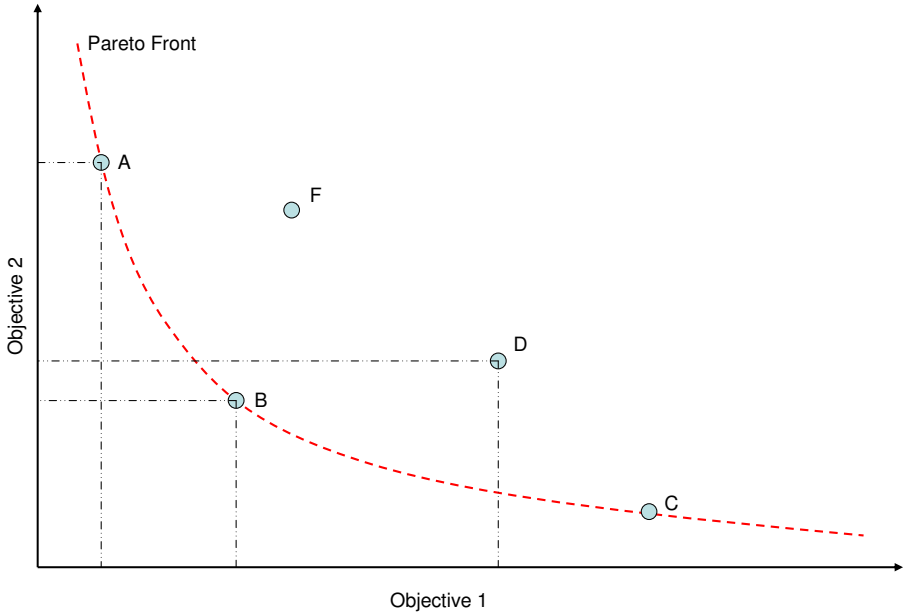


FIGURE 6.2: Concept of Pareto Set using a bi-objective minimization problem

6.3 Gaussian Emulator assisted Multiobjective Optimization

In the previous section we presented the definitions of *Dominance* and *Pareto Set* for deterministic quantities. In this section, we consider the case where Gaussian Stochastic Process models are used to expedite the Multiobjective optimization. For simplicity of presentation we present our final formulation on bi-objective problems, although the proposed approach can be readily extended to any arbitrary number of objectives. Recall from 4.2.1 that we employed Gaussian emulators as surrogate to the high fidelity CFD analysis solver. For our bi-objective problem, we approximated both the objectives with Gaussian processes so that, $f_1 \sim \mathcal{GP}_1(m_1(\mathbf{x}), \mathcal{C}_1(\mathbf{x}, \mathbf{x}'))$ and $f_2 \sim \mathcal{GP}_2(m_2(\mathbf{x}), \mathcal{C}_2(\mathbf{x}, \mathbf{x}'))$. For each objective, the realization of the Gaussian process at any two points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \in \mathbb{R}^m$ are correlated Gaussian random variables. The mean vector for objective 1 at points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}$ is given by

$$\left\{ \begin{array}{c} m_1(\mathbf{x}^{(1)}) \\ m_1(\mathbf{x}^{(2)}) \end{array} \right\}, \tag{6.3}$$

and the covariance matrix is given by

$$\left\{ \begin{array}{ll} \mathcal{C}_1(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}) & \mathcal{C}_1(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) \\ \mathcal{C}_1(\mathbf{x}^{(2)}, \mathbf{x}^{(1)}) & \mathcal{C}_1(\mathbf{x}^{(2)}, \mathbf{x}^{(2)}) \end{array} \right\}. \tag{6.4}$$

Similarly for objective 2 at points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}$ the mean vector is

$$\left\{ \begin{array}{l} m_2(\mathbf{x}^{(1)}) \\ m_2(\mathbf{x}^{(2)}) \end{array} \right\}, \tag{6.5}$$

and the covariance matrix is given by

$$\left\{ \begin{array}{ll} \mathcal{C}_2(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}) & \mathcal{C}_2(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) \\ \mathcal{C}_2(\mathbf{x}^{(2)}, \mathbf{x}^{(1)}) & \mathcal{C}_2(\mathbf{x}^{(2)}, \mathbf{x}^{(2)}) \end{array} \right\}. \tag{6.6}$$

Since we are now dealing with random objectives (an approximation of the original deterministic objectives), we can not directly use the definition for *Dominance* and *Pareto set* which were applicable for deterministic variables.

6.3.1 Probabilistic Definitions

To begin with we redefine the concepts of Dominance and Pareto set for random objectives.

Definition 4 (Probabilistic Dominance)

Let $\mathcal{F}^{(1)}, \mathcal{F}^{(2)} \in \mathbb{R}^n$ be two vectors of random objectives. Then $\mathcal{F}^{(1)}$ is said to dominate $\mathcal{F}^{(2)}$ (i.e. $\mathcal{F}^{(1)} \succ \mathcal{F}^{(2)}$), if both the following conditions are true

- $\forall i \in \{1, 2, \ldots, n\}: \quad \mathscr{P}[\mathcal{F}_i^{(1)} \leq \mathcal{F}_i^{(2)}] \geq \gamma$
- $\exists\ j \in \{1, 2, \ldots, n\}: \quad \mathscr{P}[\mathcal{F}_j^{(1)} < \mathcal{F}_j^{(2)}] > \gamma$

where $0.5 < \gamma \leq 1$ is defined as the *confidence factor*. γ is an important parameter and has to be chosen judiciously. The higher the γ value, the lower the chance of selecting a point with high uncertainty in prediction. We discuss the effect of γ on the Pareto optimal set in more detail later in the chapter. $\mathscr{P}[\mathcal{X}]$ denotes the probability of occurrence of $\mathcal{X}$. This definition implies that if the solution $\mathcal{F}^{(1)}$ is said to dominate $\mathcal{F}^{(2)}$, then the probability $\mathscr{P}$ that $\mathcal{F}^{(1)}$ is no worse than $\mathcal{F}^{(2)}$ in all objectives is greater than or equal to γ and at the same time the probability that $\mathcal{F}^{(1)}$ is strictly better than $\mathcal{F}^{(2)}$ in at least one objective is greater than γ. The definition of probabilistic Pareto set follows as

Definition 5 (Probabilistic Pareto Set)

Let $\mathscr{F} \subseteq \mathbb{R}^n$ be a set of random vectors. Then the Pareto set $\mathscr{F}^*$ of $\mathscr{F}$ is defined as follows: $\mathscr{F}^*$ contains all random vectors $\mathcal{F}^{(*)} \in \mathscr{F}$ which are not dominated by any random vector $\mathcal{F} \in \mathscr{F}$, i.e.

- $\mathscr{F}^* := \{\mathcal{F}^{(*)} \in \mathscr{F} |\ \nexists\, \mathcal{F} \in \mathscr{F} : \mathcal{F} \succ \mathcal{F}^{(*)}\}$

It can be deduced from this definition that any random vector $\mathcal{F} \in \mathscr{F}$ is dominated by at least one random vector belonging to the Pareto set ($\mathcal{F}^{(*)} \in \mathscr{F}^*$). These definitions are very general and can be used for any multiobjective optimization problem dealing with stochastic objectives. Next we illustrate the application of these probabilistic definitions to a bi-objective optimization problem.

6.3.2 Comparing Random Objectives

In order to use the definition of *Probabilistic Dominance* we need to evaluate the probability given by $\mathscr{P}[\mathcal{F}_i^{(1)} \leq \mathcal{F}_i^{(2)}]$. This essentially involves evaluating the statistics of the random variable given by $\mathcal{F}_i^{(1)} - \mathcal{F}_i^{(2)}$. For the case of a bi-objective minimization problem, let $\mathcal{Z}_1$ be a random variable given by the difference in objective 1 at two points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}$, i.e. $\mathcal{Z}_1 = \mathcal{F}_1^{(1)} - \mathcal{F}_1^{(2)}$. In the first step we estimate the expected value of $\mathcal{Z}_1$, denoted as $m_{\mathcal{Z}_1}$ by

$$\int \mathcal{Z}_1 P(\mathcal{Z}_1) \mathrm{d}\mathcal{Z}_1, \tag{6.7}$$

where $P(\mathcal{Z}_1)$ is the probability density function (PDF) of the random variable $\mathcal{Z}_1$. Equation (6.7) can also be expressed as

$$\int \mathcal{F}_1^{(1)} \mathrm{d}\mathcal{F}_1^{(1)} \int P_{12}(\mathcal{F}_1^{(1)}, \mathcal{F}_1^{(2)}) \mathrm{d}\mathcal{F}_1^{(2)} - \int \mathcal{F}_1^{(2)} \mathrm{d}\mathcal{F}_1^{(2)} \int P_{12}(\mathcal{F}_1^{(1)}, \mathcal{F}_1^{(2)}) \mathrm{d}\mathcal{F}_1^{(1)} \tag{6.8}$$

where $P_{12}(\mathcal{F}_1^{(1)}, \mathcal{F}_1^{(2)})$ is the joint PDF of the random variables $\mathcal{F}_1^{(1)}$ and $\mathcal{F}_1^{(2)}$. From the definition of marginal probability density function we know that $\int P_{12}(\mathcal{F}_1^{(1)}, \mathcal{F}_1^{(2)}) \mathrm{d}\mathcal{F}_1^{(2)} = P(\mathcal{F}_1^{(1)})$ and $\int P_{12}(\mathcal{F}_1^{(1)}, \mathcal{F}_1^{(2)})) \mathrm{d}\mathcal{F}_1^{(1)} = P(\mathcal{F}_1^{(2)})$. Substituting this we get

$$\int \mathcal{F}_1^{(1)} P(\mathcal{F}_1^{(1)}) \mathrm{d}\mathcal{F}_1^{(1)} - \int \mathcal{F}_1^{(2)} P(\mathcal{F}_1^{(2)}) \mathrm{d}\mathcal{F}_1^{(2)} \tag{6.9}$$

This leads to the final expression for the expected value of $\mathcal{Z}_1$ as

$$m_{\mathcal{Z}_1} = m_1(\mathbf{x}^{(1)}) - m_1(\mathbf{x}^{(2)}). \tag{6.10}$$

Next we derive the second central moment of the random variable $\mathcal{Z}_1$, the variance $\sigma^2_{\mathcal{Z}_1}$, which can be written as

$$\int (\mathcal{Z}_1 - m_{\mathcal{Z}_1})^2 P(\mathcal{Z}_1) \mathrm{d}\mathcal{Z}_1 \tag{6.11}$$

which can also be expressed as

$$\iint (\mathcal{F}_1^{(1)} - \mathcal{F}_1^{(2)})^2 P(\mathcal{F}_1^{(1)}, \mathcal{F}_1^{(2)}) \mathrm{d}\mathcal{F}_1^{(1)} \mathrm{d}\mathcal{F}_1^{(2)} \quad - \quad m_{\mathcal{Z}_1}^2 \ . \tag{6.12}$$

After rearranging we get

$$\int {\mathcal{F}_1^{(1)}}^2 P(\mathcal{F}_1^{(1)}) \mathrm{d}\mathcal{F}_1^{(1)} + \int {\mathcal{F}_1^{(2)}}^2 P(\mathcal{F}_1^{(2)}) \mathrm{d}\mathcal{F}_1^{(2)} - 2 \iint \mathcal{F}_1^{(1)} \mathcal{F}_1^{(2)} P_{12}(\mathcal{F}_1^{(1)}, \mathcal{F}_1^{(2)}) \mathrm{d}\mathcal{F}_1^{(1)} \mathrm{d}\mathcal{F}_1^{(2)} - m_{\mathcal{Z}_1}^2 \ . \tag{6.13}$$

Using equations (6.3) and (6.4) in conjunction with the definitions for covariance and second central moments we can finally deduce that

$$\sigma_{\mathcal{Z}_1}^2 = \mathcal{C}_1(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}) + \mathcal{C}_1(\mathbf{x}^{(2)}, \mathbf{x}^{(2)}) - 2\mathcal{C}_1(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) \ . \tag{6.14}$$

Similarly, for the second objective we can define $\mathcal{Z}_2 = \mathcal{F}_2^{(1)} - \mathcal{F}_2^{(2)}$. The mean of $\mathcal{Z}_2$ can be written as

$$m_{\mathcal{Z}_2} = m_2(\mathbf{x}^{(1)}) - m_2(\mathbf{x}^{(2)}) \ . \tag{6.15}$$

and the variance can be expressed as

$$\sigma_{\mathcal{Z}_2}^2 = \mathcal{C}_2(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}) + \mathcal{C}_2(\mathbf{x}^{(2)}, \mathbf{x}^{(2)}) - 2\mathcal{C}_2(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) \ . \tag{6.16}$$

These expressions are in general applicable for any random quantities. Note that in our problem $\mathcal{F}$ are Gaussian random variables. As $\mathcal{Z}$ is a linear combination of two Gaussian random variables, it is also Gaussian. Hence, the random variables $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are completely defined by their first and second moments; $\mathcal{Z}_1 \sim \mathcal{N}(m_{\mathcal{Z}_1}, \sigma_{\mathcal{Z}_1}^2)$ and $\mathcal{Z}_2 \sim \mathcal{N}(m_{\mathcal{Z}_2}, \sigma_{\mathcal{Z}_2}^2)$. The next step is to calculate the probability that $\mathcal{Z} > 0$. We can express this as

$$\mathscr{P}(\mathcal{Z} > 0) = 1 - \int_{-\infty}^{0} \frac{1}{\sigma_{\mathcal{Z}}\sqrt{2\pi}} e^{-\frac{(\mathcal{Z} - m_{\mathcal{Z}})^2}{2\sigma_{\mathcal{Z}}^2}} \mathrm{d}\mathcal{Z}. \tag{6.17}$$

The second term on the right hand side in equation (6.17) is the cumulative distribution function which can be expressed in terms of the error function ($\mathrm{erf}(x) = \int_0^x \frac{2}{\sqrt{\pi}} e^{-t^2} \mathrm{d}t$) as

$$\frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{-m_{\mathcal{Z}}}{\sigma_{\mathcal{Z}}\sqrt{2}}\right)\right]. \tag{6.18}$$

Substituting equation (6.18) in equation (6.17) we finally get

$$\mathscr{P}(\mathcal{Z} > 0) = \frac{1}{2}\mathrm{erfc}\left(\frac{-m_{\mathcal{Z}}}{\sigma_{\mathcal{Z}}\sqrt{2}}\right), \tag{6.19}$$

where $\mathrm{erfc}(x)$ is the complementary error function. This derivation is showed in detail in Appendix C. The next is to evaluate if $\mathscr{P}(\mathcal{Z} > 0)$ is greater than γ, where $0.5 < \gamma \leq 1$. In

the case of Gaussian random variables, which are symmetric about their mean, the choice of γ close to 0.5 would imply that we compare just the mean values and the standard deviation is more or less ignored in probabilistic ranking. Higher values of γ would put a stringent condition on deciding if a Gaussian random objective dominates the other. This would lead to a cloud of non-dominated points rather than a front. We will illustrate this idea while applying the proposed algorithm to some test functions in the numerical studies carried out later in this chapter

These derivations illustrated how to employ the probabilistic definitions for ranking multiple objectives. In the next section we propose a probabilistic multiobjective optimization algorithm based on NSGA-II. Note that the proposed ideas are not limited to use with Gaussian Stochastic Process model or NSGA-II. The concept is very general and can find applications in many cases. For example, a very interesting application would be to use these definitions when a multiobjective optimizer is used in conjunction with a stochastic solver where, each objective would be random with a given probability density function.

6.3.3 Probabilistic NSGA-II

We present an algorithm based on the Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) proposed by Deb *et al.* (2002). A schematic of the NSGA-II algorithm is shown in figure 6.3. Apart from having the standard GA operators like *Reproduction, Crossover* and *Mutation*, NSGA-II uses an elite preservation strategy. NSGA-II also employs an explicit diversity-preserving mechanism using a niching strategy. This niching strategy is based on local crowding distance, which ensures that new members reside in the least crowded region of the Pareto front. In the proposed algorithm we extend NSGA-II to include Gaussian emulator based search with an efficient update strategy. We also modify the ranking procedure by introducing the probabilistic dominance formulation we presented in the earlier sections. We refer to this new algorithm as **Probabilistic NSGA-II** and the key steps are summarised in Algorithm 1.

We first create the initial sample population $\mathcal{P}^0$ using any standard DOE technique. The high fidelity analysis solver is employed to evaluate the objective values $f_i \mid i = 1, ..., n$ at these points to get the initial dataset $\mathcal{D}_0$. This dataset is used to train the Gaussian emulator which is employed as the cheap surrogate to the expensive analysis model. At each generation, the offspring population $\mathcal{Q}^t$ is created by using *Reproduction, Crossover* and *Mutation* operators on the parent population $\mathcal{P}^t$. In the next step, the parent and offspring

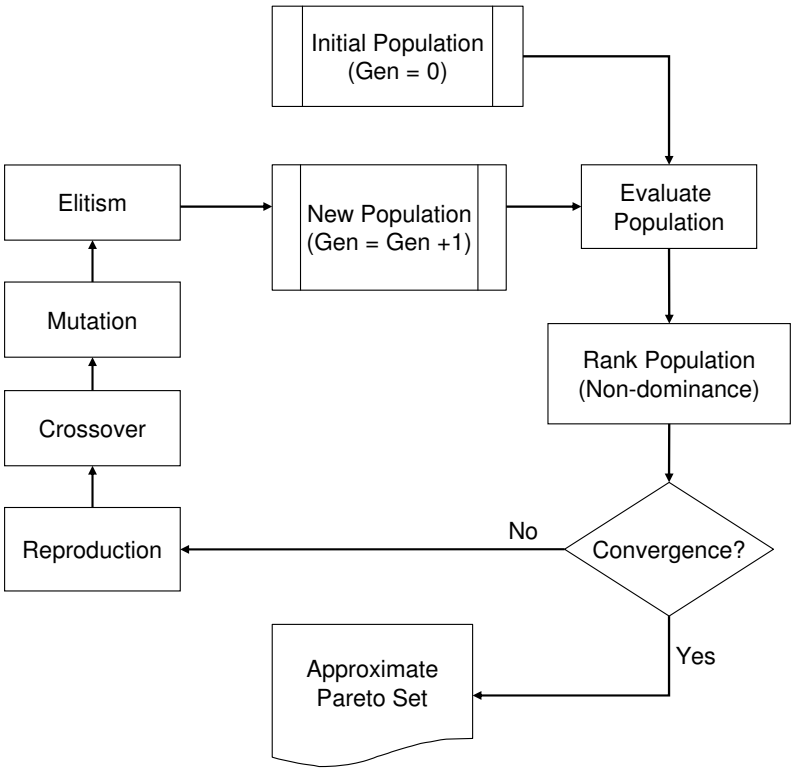


FIGURE 6.3: Flowchart for NSGA-II

populations are combined together to form $\mathcal{R}^t$. If both $\mathcal{P}^t$ and $\mathcal{Q}^t$ were of size N (population size), the new set $\mathcal{R}^t$ is of size $2N$. Probabilistic non-dominated ranking is then employed to classify the new population $\mathcal{R}^t$ to allow a global non-domination check among the offspring and parent solutions. This leads to identification of all the probabilistic non-dominated sets $\mathscr{F}^{t_k} \mid k = 1, 2, ...etc.$, where k denotes the ranking of the set. For example $k = 1$ is the best non-dominated set, $k = 2$ is the non-dominated set if we remove points with $k = 1$.

Once the probabilistic ranking is over, the new population is created by selecting solutions from the different non-dominated sets. The best non-dominated set ($k = 1$) is selected first, followed by the second non-dominated set ($k = 2$) and then the third non-dominated set ($k = 3$) and so on. This is shown in steps 7, 8 and 9 in Algorithm 1. Since the size of $\mathcal{R}^t$ is $2N$, all the sets cannot be accommodated in the new population. The last set which can be accommodated may also have more solutions that the remaining slots left in the new generation. A niching strategy based on the low crowding algorithm is employed to select

Algorithm 1 Probabilistic NSGA-II

1. Create initial random sample population $\mathcal{P}^0$

2. Evaluate the high fidelity analysis model at $\mathcal{P}^0$ to get the initial dataset $\mathcal{D}_0$

3. Assign $t = 0$, $k = 1$; *(where k is the rank of objectives)*

while (Convergence Criteria not met) **do**

4. Train Gaussian emulators for all objectives using training dataset $\mathcal{D}_t$

$\{ f_j \sim \mathcal{GP}_j(m_j(\mathbf{x}), \mathcal{C}_j(\mathbf{x}, \mathbf{x}')) \mid j = 1, ..., n;$ *n = number of objectives*$\}$

5. Create offspring population $\mathcal{Q}^t$ from parent population $\mathcal{P}^t$, where $|\mathcal{Q}^t| = N$

{*using Reproduction, Crossover and Mutation Operators;* N *= population size*}

6. Combine parent and offspring population $\mathcal{R}^t = \mathcal{P}^t \cup \mathcal{Q}^t$

7. Perform probabilistic non-dominated sorting to $\mathcal{R}^t$

{*Identify different probabilistic Pareto sets* $\mathscr{F}^{t_k}$}

while $(|\mathcal{P}^{t+1}| + |\mathscr{F}^{t_k}| < N)$ **do**

8. $\mathcal{P}^{t+1} = \mathcal{P}^{t+1} \cup \mathscr{F}^{t_k}$

{*Create new population set by adding probabilistic non-dominated sets*}

9. $k = k + 1$

end while

10. Perform Crowding Sort and on $\mathcal{P}^{t+1}$ to select u points for update $\mathcal{P}^u$

11. Evaluate the true function $\mathcal{P}^u$ and update training dataset $\mathcal{D}_{t+1} = \mathcal{D}_t + \mathcal{D}_u$

12. $t = t + 1$

end while

the remaining solutions. This strategy is also referred to as *Elitism*.

Once the new population is created, u number of points are selected for updating the surrogate model. A low crowding sorting approach which minimizes the Euclidean distance between the selected points is employed. The analysis solver is executed at these points to obtain the true objectives at these points ($\mathcal{P}^u$). The update training dataset $\mathcal{D}_u$ is then appended to the existing dataset $\mathcal{D}_t$ to get $\mathcal{D}_{t+1}$, which is then employed to retrain the Gaussian emulator. The emulator based optimization and the subsequent update process is repeated until a user specified convergence criteria is met. The convergence criteria could be a function of successive improvements made in the new approximate Pareto set or perhaps the maximum computational resources available for design search. In the next section we evaluate the performance of the proposed algorithm using a suit of test functions.

6.4 Numerical Studies

We compare the performance of the Probabilistic NSGA-II algorithm proposed above with direct NSGA-II search on the true function and the Gaussian emulator assisted NSGA-II search. The steps for Gaussian emulator assisted NSGA-II search are the same as the steps shown for Probabilistic NSGA-II in algorithm 1, except that there is no probabilistic dominance based ranking. It is a common practice to only use the posterior mean predicted by the Gaussian emulator, during non-dominance based ranking and ignore the error bars (posterior variance). The comparison is performed on a selected set of mathematical test problems whose true Pareto front are known. Note that for all test problems the objectives are deterministic. However, since we employ the Gaussian Stochastic Process model to approximate the existing deterministic objectives, the new approximate objectives are stochastic.

For all the experiments we fix the population size $N = 100$. For the direct NSGA-II, the GA is stopped after 50 generations. We employ 10 update steps for both the other methods. For probabilistic ranking the *Confidence Factor* is selected to be 0.7 ($\gamma = 0.7$). Every experiment was repeated 5 times, each with a different random number seed. For every experiment we plot the best predicted approximate Pareto set. This can be employed to check the diversity of the population and the coverage of the the true Pareto front by the approximate Pareto sets predicted by each method. We also compare a convergence metric which is a measure of the convergence of the approximate Pareto set toward the true Pareto front which we describe next.

6.4.1 Convergence Metric

We employ the convergence metric proposed by (Deb & Jain, 2002) for comparing the performance of the above mentioned methods. This metric evaluates the convergence towards a target reference set $\mathbf{F}^*$. The target set of points can either be the set of the true Pareto Optimal set or a set of non-dominated solutions obtained by another algorithm or a huge GA run. In our case, the true Pareto fronts for the test problems are known apriori. We define the convergence metric as follows. For the population $\mathcal{P}^{(t)}$ at any generation let the non-dominated set be $\mathbf{F}^t = \{\mathbf{f}^{(1^t)}, ..., \mathbf{f}^{(i^t)}, ..., \mathbf{f}^{(p^t)}\}$, where $p = |\mathbf{F}^t|$. If n is the number of objectives, each vector of the set can be expressed as $\mathbf{f}^{(i^t)} = [f_1^{(i^t)}, f_2^{(i^t)}, ..., f_n^{(i^t)}]$. Similarly for the target reference set $\mathbf{F}^*$, each vector of the set can be expressed as $\mathbf{f}^{(i^*)} = [f_1^{(i^*)}, f_2^{(i^*)}, ..., f_n^{(i^*)}]$. The steps involved in evaluating the convergence metric are shown in Algorithm 2. In the next

Algorithm 2 Evaluate Convergence Metric

1. Identify the target reference set $\mathbf{F}^*$

2. Identify the non-dominated set $\mathbf{F}^t$ of $\mathcal{P}^t$

for $i = 1$ to p **do**

$\{where\ p = |\mathbf{F}^t|\}$

for $j = 1$ to q **do**

$\{where\ q = |\mathbf{F}^*|\}$

3.Calculate $d_{ij} = \sqrt{\sum_{k=1}^{n} \left(\frac{f_k^{(i^t)} - f_k^{(j^*)}}{\max(f_k^{(i^t)}) - \min(f_k^{(j^*)})} \right)^2}$

$\{where\ n\ is\ number\ of\ objectives\}$

end for

4. Find minimum of $d_{ij} \Rightarrow d_i = \min\ \ d_{ij}$

end for

5. Calculate Convergence Metric $\text{Conv}(\mathbf{F}^t) = \frac{\sum d_i}{|\mathbf{F}^t|}$

sections we present the test problems and the results obtained.

6.4.2 Test Case 1: Generalized Schaffer Problem

As the first test problem we employ the generalized Schaffer's test function (Emmerich *et al.*, 2006). The test function can be expressed as

$$f_1(\mathbf{x}) = \frac{1}{d^{\alpha/2}} \left(\sum_{i=1}^{d} x_i^2 \right)^{\alpha/2} \tag{6.20}$$

$$f_2(\mathbf{x}) = \frac{1}{d^{\alpha/2}} \left(\sum_{i=1}^{d} (1 - x_i)^2 \right)^{\alpha/2} \tag{6.21}$$

where d is the dimension of the design space and $\mathbf{x} \in [0, 10]^d$. The aim is to minimize both the objectives f_1 and f_2 simultaneously. The curvature of the Pareto front is scalable by changing the parameter α. The true Pareto front for this class of problem is known and is given by

$$f_2^* = \left(1 - f_1^{*1/\alpha}\right)^{\alpha} \ , \quad f_1^* \in [0, 1]. \tag{6.22}$$

The choice of $\alpha = 1$ leads to a linear Pareto front, $\alpha < 1$ leads to concave Pareto fronts and, $\alpha > 1$ leads to convex Pareto fronts. In our case we select the value of α as fixed at 2. We employ all the algorithms mentioned above to solve this problem. We solve the problem for 4-d and 6-d case. The experiments are repeated 5 times with a different random seed.

We evaluate the mean *Convergence Metric* (averaged over the 5 experiments) and select the best approximate Pareto set predicted using each algorithm for comparison. Recall that the objectives are stochastic only because we employ the Gaussian Stochastic Process model to approximate them. The results obtained are presented in figures 6.4 and 6.5.

6.4.3 Test Case 2: FON Problem

This test function was proposed by Fonseca & Fleming (1995) and can expressed as to minimize simultaneously both the objectives given by

$$f_1(\mathbf{x}) = 1 - \exp\left(-\sum_{i=1}^{d}(x_i - \frac{1}{\sqrt{d}})^2\right) \tag{6.23}$$

$$f_2(\mathbf{x}) = 1 - \exp\left(-\sum_{i=1}^{d}(x_i + \frac{1}{\sqrt{d}})^2\right) \tag{6.24}$$

where $-4 \leq x_i \leq 4 \mid i = 1, 2, ..., d$ and d is the number of variables. The true Pareto solution to this problem is $x_i^* \in [-1/\sqrt{d}, 1/\sqrt{d}] \mid i = 1, 2, ..., d$. These solutions also satisfy the following relationship between the two objective functions

$$f_2^* = 1 - \exp\left(-\left[2 - \sqrt{-\ln(1 - f_1^*)}\right]^2\right) \tag{6.25}$$

in the range $0 \leq f_1^* \leq 1 - \exp(-4)$. The true Pareto front in this case is non-convex. We employ all the algorithms mentioned above to solve this problem. We solve the problem for 3-d and 6-d case. Like the previous problem the experiments are repeated 5 times with a different random seed. We evaluate the mean *Convergence Metric* and select the best approximate Pareto set predicted using each algorithm for comparison. The objectives for this problem are also deterministic. The stochastic nature is only because we approximate them using the Gaussian process emulator. The results obtained are presented in figure 6.6 and 6.7.

6.4.4 Test Case 3: ZDT1 Problem

The next test problem is selected from a suite of problems called Zitzler-Deb-Thiele's (ZDT) test problems (Zitzler *et al.*, 2000). We choose the ZDT1 problem which has a convex Pareto front. The problem involves minimizing the following objectives simultaneously

$$f_1\mathbf{x}, \quad \text{and} \quad f_2(\mathbf{x}) = g(\mathbf{x})h(f_1(\mathbf{x}), g(\mathbf{x})). \tag{6.26}$$

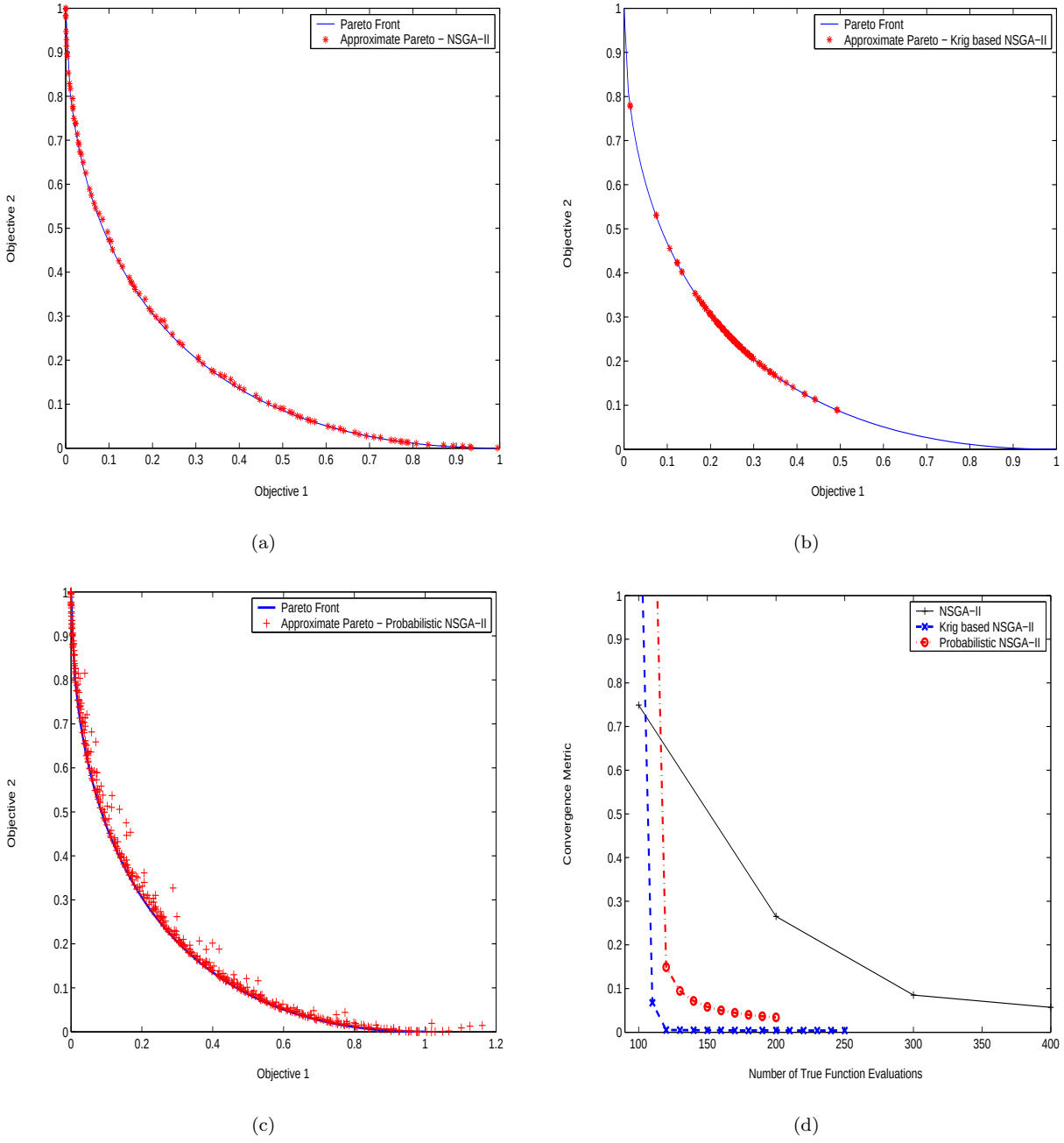


FIGURE 6.4: True Pareto front and approximate Pareto set for SCH function (4d) using (a) NSGA-II (b) Gaussian emulator based NSGA-II (c) Probabilistic Ranking and Gaussian emulator based NSGA-II (d) Convergence Metric for original, Gaussian emulator based and Gaussian emulator with probabilistic ranking based NSGA-II for SCH function (d=4)

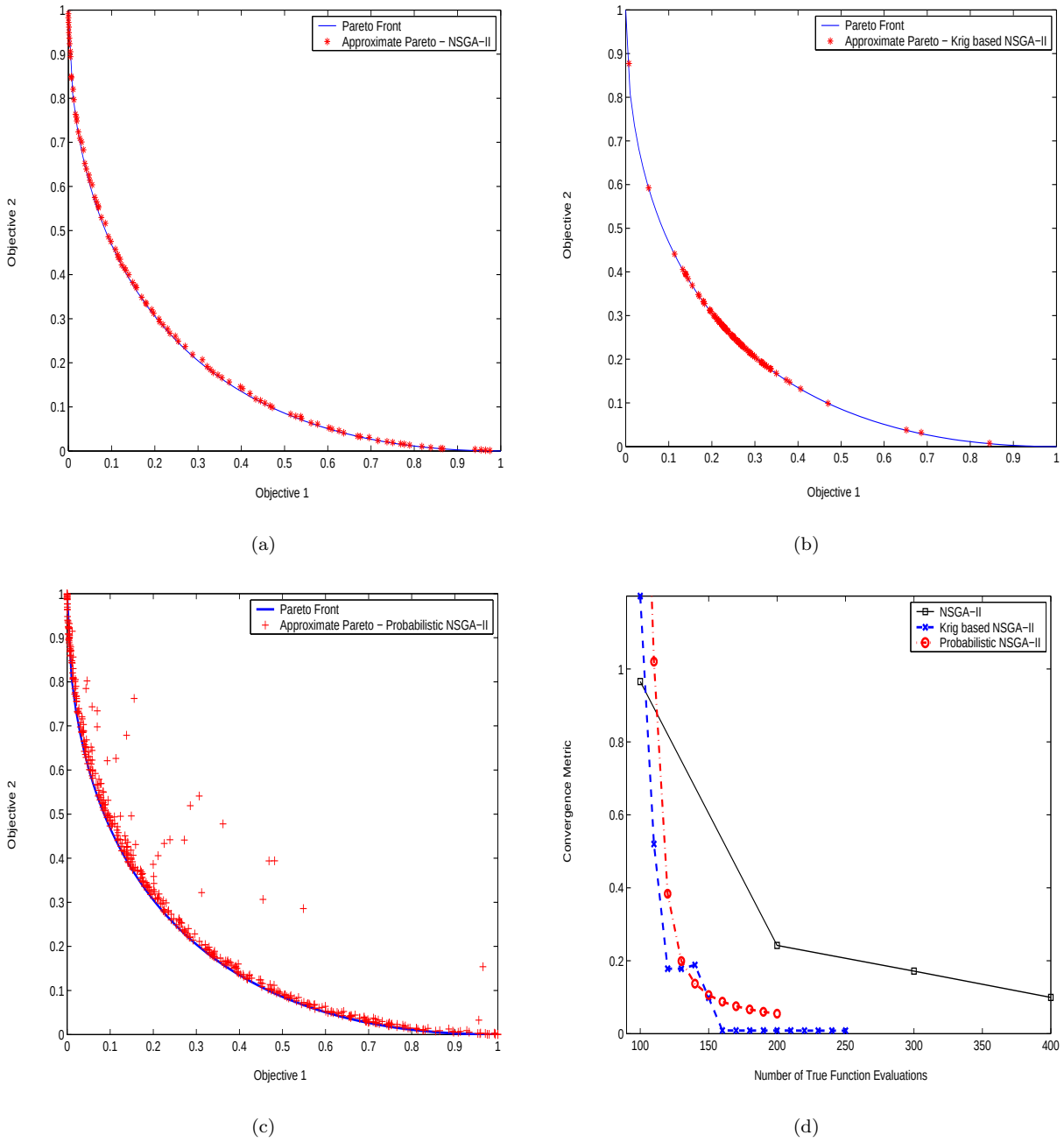


FIGURE 6.5: True Pareto front and approximate Pareto set for SCH function (6d) using (a) NSGA-II (b) Gaussian emulator based NSGA-II (c) Probabilistic Ranking and Gaussian emulator based NSGA-II (d) Mean Convergence Metric for original, Gaussian emulator based and Gaussian emulator with probabilistic ranking based NSGA-II for SCH function (d=6)

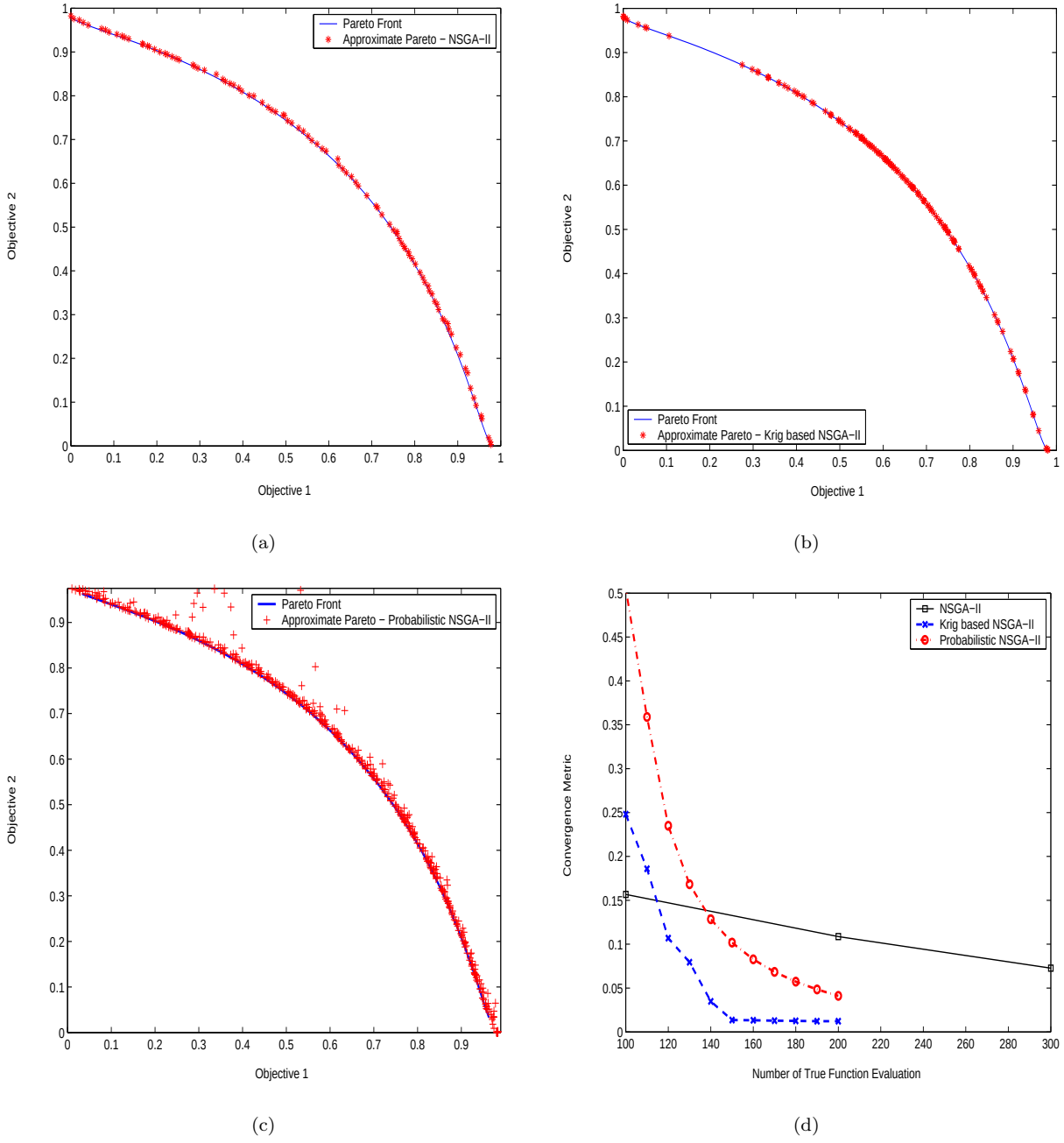


FIGURE 6.6: True Pareto front and approximate Pareto set for FON function (3d) using (a) NSGA-II (b) Gaussian emulator based NSGA-II (c) Probabilistic Ranking and Gaussian emulator based NSGA-II (d) Mean Convergence Metric for original, Gaussian emulator based and Gaussian emulator with probabilistic ranking based NSGA-II for FON function (d = 3)

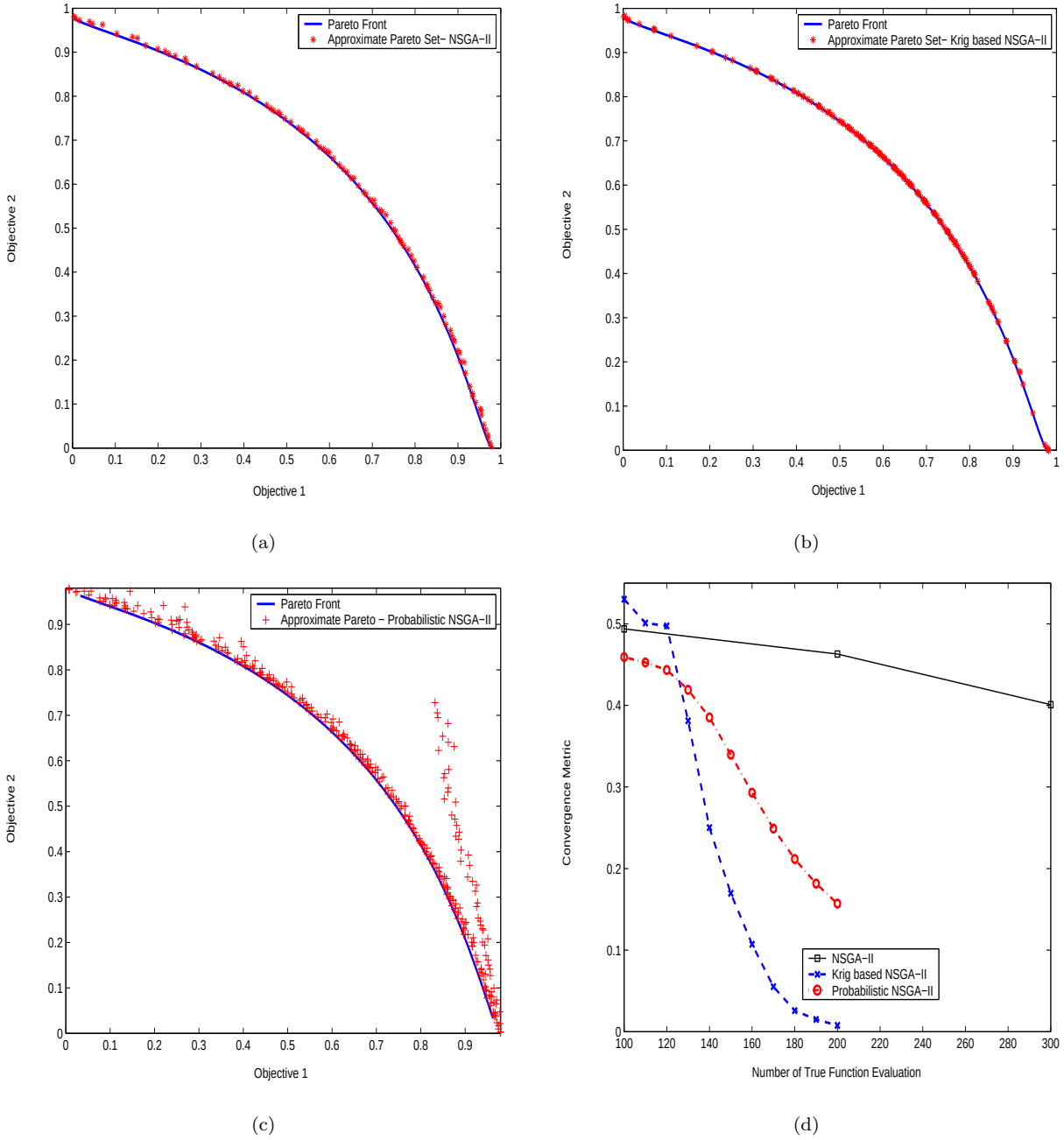


FIGURE 6.7: True Pareto front and approximate Pareto set for FON function (6-d) using (a) NSGA-II (b) Gaussian emulator based NSGA-II (c) Probabilistic Ranking and Gaussian emulator based NSGA-II (d) Mean Convergence Metric for original, Gaussian emulator based and Gaussian emulator with probabilistic ranking based NSGA-II for FON function (d = 6)

where $f_1(\mathbf{x})$, $g(\mathbf{x})$ and $h(f_1, g)$ is given by

$$f_1(\mathbf{x}) = x_1, \tag{6.27}$$

$$g(\mathbf{x}) = 1 + \frac{9}{d-1}\sum_{i=2}^{d} x_i, \tag{6.28}$$

$$h(f_1, g) = 1 - \sqrt{(f_1/g)}. \tag{6.29}$$

where $x_i \in [0,1]; i = 1, 2, ..., d$. The true Pareto front corresponds to $0 \leq x_1^* \leq 1$ and $x_i^* = 0 \mid i = 2, 3, ..., d$. We employ all the algorithms mentioned above to solve this problem. We solve the problem for 6-d and 10-d case. Again the experiments are repeated 5 times with a different random seed. We evaluate the mean *Convergence Metric* and select the best approximate Pareto set predicted using each algorithm for comparison. The randomness in the objectives is again attributed to the Gaussian Process emulator employed to approximate them. The results obtained are presented in figures 6.8 and 6.9

6.4.5 Results and Comments

In the last section we conducted experiments to compare the three methods for multiobjective optimization. The results obtained are summarised in figures 6.4 - 6.9 . We showed the plot of mean *Convergence Metric* versus the number of true function evaluations. It is very important to consider the number of true function evaluation rather the number of surrogate evaluations, when dealing with high fidelity analysis models. For example in our case each CFD evaluation takes approximately 20 minutes, whereas the Gaussian emulator evaluation takes a fraction of a second once the hyperparameters have been tuned.

It can be observed from Figure 6.4 - 6.9 that the probabilistic NSGA-II algorithm and the Gaussian emulator (krig) assisted algorithm have better mean convergence as compared to the direct NSGA-II algorithm. It can also be observed that Gaussian emulator assisted NSGA-II does ensure good diversity in the approximate Pareto set. This is obvious from figures 6.4 (b), 6.5 (b) and 6.6 (b). For low dimension of input variables (d), there is no marked difference in the mean convergence metric for the two methods though probabilistic NSGA-II shows better diversity in solutions. For high dimensional problems the Probabilistic NSGA-II performs better than the Gaussian emulator assisted NSGA-II in terms of diversity in Pareto Set; for example see figures 6.5 (d), 6.7 (d) and 6.9 (d). However, Gaussian emulator assisted NSGA-II has better convergence towards the Pareto front for these problems. This can explained by the fact that Gaussian emulators suffer from the curse of dimensionality and thus, for high dimensional problems the error in prediction (posterior variance) is high.

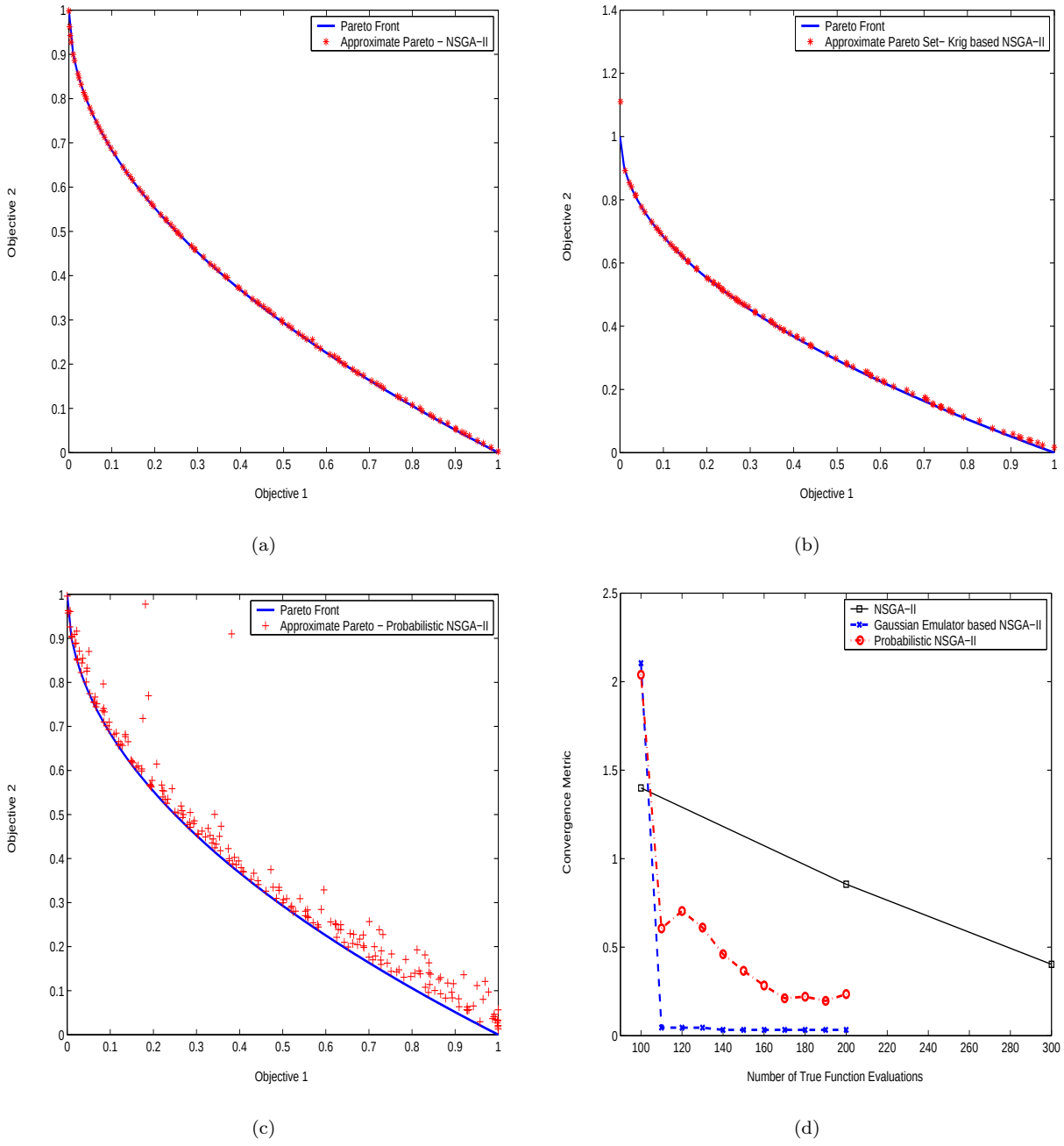


FIGURE 6.8: True Pareto front and approximate Pareto set for ZDT1 function (6d) using (a) NSGA-II (b) Gaussian emulator based NSGA-II (c) Probabilistic Ranking and Gaussian emulator based NSGA-II (d) Mean Convergence Metric for original, Gaussian emulator based and Gaussian emulator with probabilistic ranking based NSGA-II for ZDT1 function (6d)

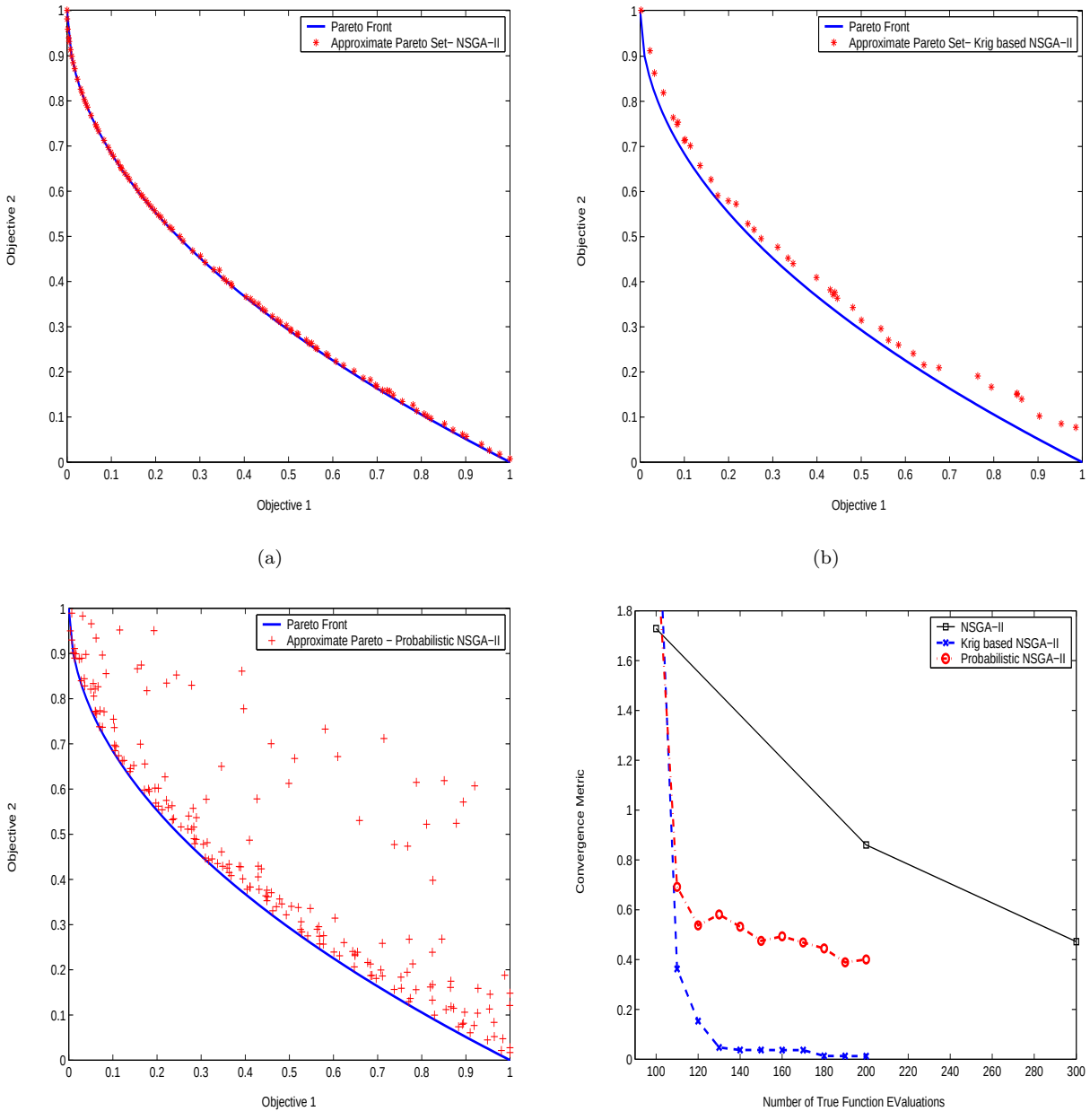


FIGURE 6.9: True Pareto front and approximate Pareto set for ZDT1 function (10d) using (a) NSGA-II (b) Gaussian emulator based NSGA-II (c) Probabilistic Ranking and Gaussian emulator based NSGA-II (d) Mean Convergence Metric for original, Gaussian emulator based and Gaussian emulator with probabilistic ranking based NSGA-II for ZDT1 function (10d)

With high *confidence factor* values ($\gamma > 0.7$), we put a strict condition for deciding if a point is dominated by others. Many points which are visually dominated by others are selected in the Pareto set, as they still satisfy the non-dominated criteria due to high posterior variance and the stricter condition imposed by choosing high γ. This trend can be observed in figures 6.4 (c) - 6.9 (c). This in turn leads to Pareto clouds which in turn adversely affect the convergence metric for high dimensional problems; see figure 6.7 (d), 6.8 (d) and 6.9 (d) Next we analyze the reasons for the better diversity observed for the Probabilistic NSGA-II method.

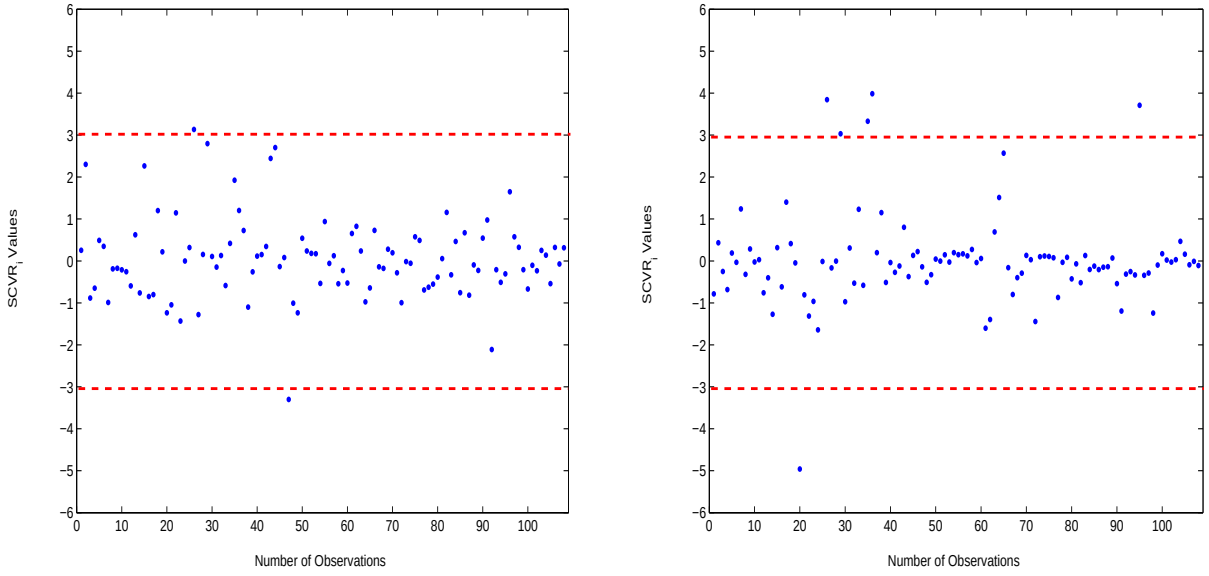
The basic concept of any GA including NSGA-II is to preserve and propagate solutions which are better than others. The working principles are similar to natural genetics and the algorithms try to artificially simulate processes such as *Reproduction* , *Crossover* and *Mutation*. These algorithms provide a powerful tool for optimization when used with true functions. However, when used with approximate models of the true function they can be misled due to the presence of errors in predictions. These errors can lead to selection of *offspring* which are sub-optimal in the new parent generation. Such sub-optimal designs would be propagated across populations and could take the search away from the true Pareto front. Since the existing Gaussian emulator assisted search algorithms does not consider the errors in prediction (posterior variance for Gaussian emulator), the approximate Pareto set may have a good predicted values but may also have high errors. A consequence could be that during update the search would be spending time in regions which are not of interest. This would also slow down convergence to the true Pareto front. In Probabilistic NSGA-II we take into account the posterior variance while ranking and selecting the new generation, hence ensuring that the solutions with less error are selected and propagated. Another point to be noted is that the Probabilistic NSGA-II algorithm, is at least as good as the Gaussian emulator assisted NSGA-II algorithm; i.e. if the surrogate model is a very good approximation of the true function the Probabilistic NSGA-II method would tend to simple predicted value comparison like the Gaussian emulator assisted NSGA-II.

6.5 Robustness against Erosion: Revisited

Firstly we revisit the robust design in the presence of erosion problem considered earlier in Chapter 4. Recall that, we conducted an inner and outer array based experiment, where the inner array was employed to estimate the statistics over the noise space (erosion param-

eters). Two Gaussian emulators were trained to predict the mean and standard deviation, respectively. Due to the small sample size employed for estimating the statistics over the noise variables, we are dealing with a dataset with noise in the output. Here we take a dataset of 108 points from that experiment (after 5 updates on the initial 50 points for the previous study), to train the surrogate models. As the data is noisy we employ the regression based Gaussian Stochastic Process model for this case. For regression we get one extra hyperparameter to tune to the dataset, and the lower the regression parameter the better the fit. Maximum likelihood estimation revealed that the regression hyperparameter of the surrogate for the mean is 0.000552 and that for the standard deviation is 0.231273. This is to be expected since we know from prior experience that the errors in predicting standard deviation is higher.

Another issue involved is that the standard deviation is always positive by definition. However, in this case the noise space is sparsely populated for predicting the statistics and as a consequence the surrogate model output may not be strictly positive. To manage this issue we train the Gaussian emulator to the log of the predicted standard deviation $\log(\sigma)$. We present the $SCVR_i$ tests on both the surrogate models in figure 6.10. This tests ensures



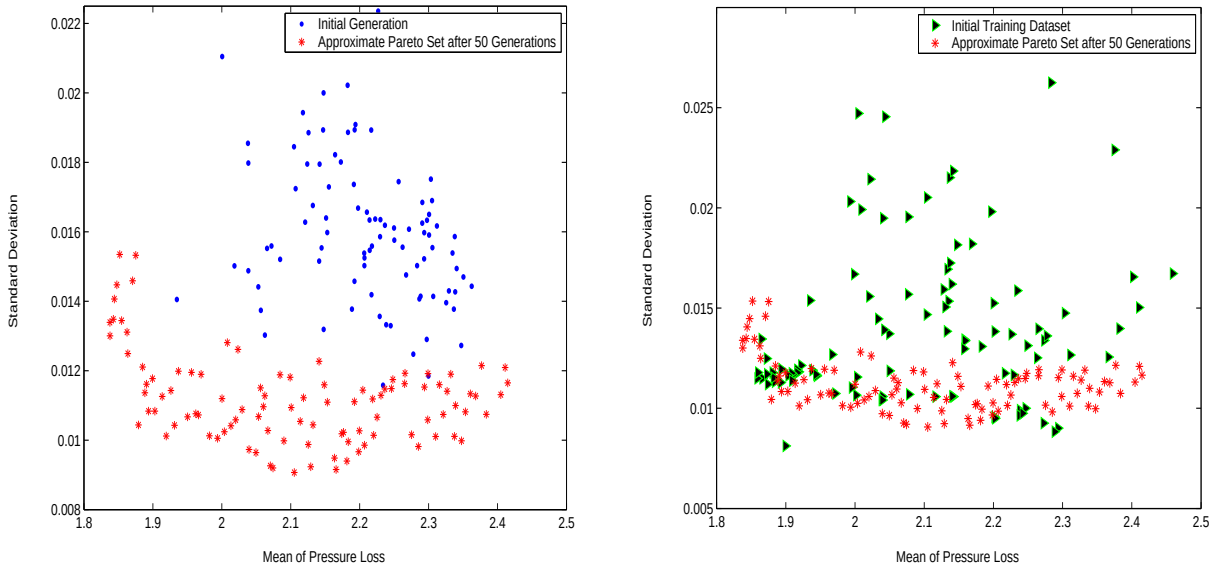
(a) (b)

FIGURE 6.10: $SCVR_i$ validation check on the Gaussian emulators for (a) Mean of Pressure Loss (b) Standard Deviation of Pressure Loss

that the Gaussian Process assumption is valid as most of the $SCVR_i$ values lie between

the -3σ and $+3\sigma$ range. It can be observed from figure 6.10 that the Gaussian emulator model for predicting mean is better than that for predicting the standard deviation. Next we conduct the robust design study in conjunction with the surrogate models.

We apply the Probabilistic NSGA-II method shown in Algorithm 1 to this problem. It is important to note that, we conduct probabilistic ranking on the objectives and their random nature is attributed here to the Gaussian emulator being employed to approximate them. We conduct a probabilistic NSGA-II run with population size $N = 100$ for 50 generations. The *Confidence Factor* (γ) is taken to be 0.7. The approximate Pareto set obtained is plotted in



(a) (b)

FIGURE 6.11: Approximate Pareto set after 50 generations using Probabilistic NSGA-II with (a) Initial GA generation (b) Initial training dataset

figure 6.11. The figures also show the initial generation from which the GA started the search as well as the initial dataset which was employed to train the surrogate models. Significant improvements have been made by the proposed algorithm from the initial generation. Recall that the initial dataset in figure 6.11 (b) was obtained after 5 updates during robust design studies using the method employed in Chapter 4. Even so the surrogate based Probabilistic NSGA-II algorithm predicts improvements with good diversity in the Pareto set. To verify the prediction we select 10 points using low crowding algorithm from the approximate Pareto set in figure 6.11 for verification. At each of these points a 15 point MCS is conducted over the noise variables (erosion parameters) to estimate the mean and standard deviation. The

results are shown with the initial training dataset in figure 6.12. It can be observed from

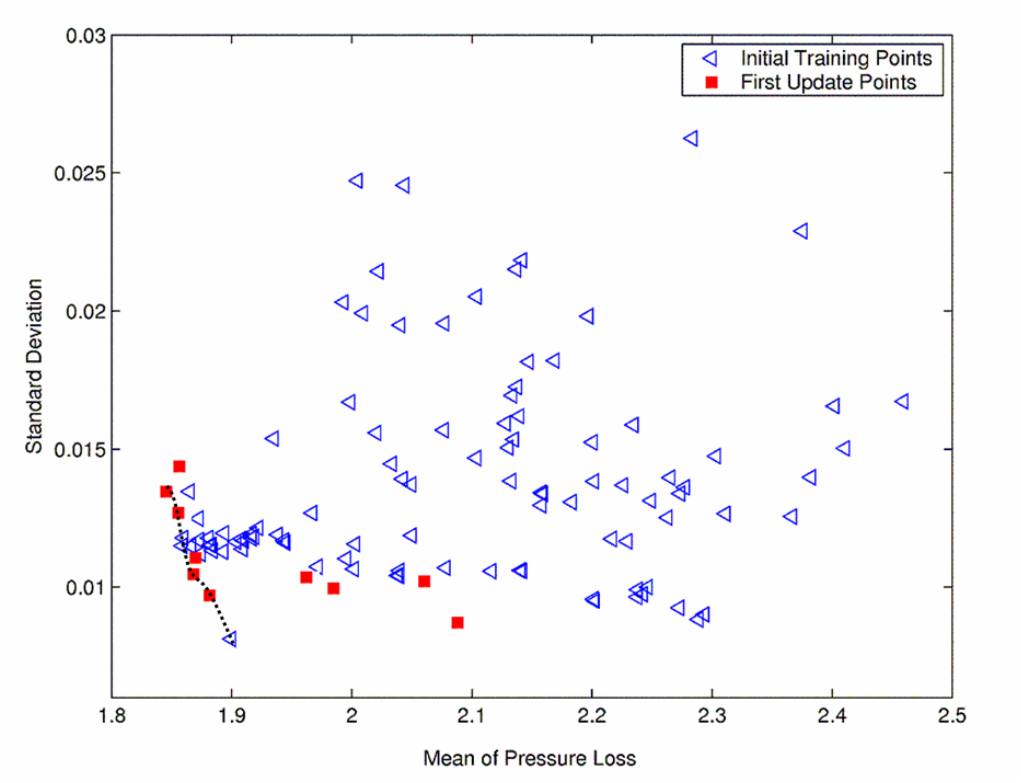


FIGURE 6.12: Approximate Pareto Set after appending the update points to the initial training dataset. The dotted line shows the approximate Pareto Front.

the figure that the Pareto front has improved and as expected the points do not move far from the predicted Pareto set after verification. This can be attributed to probabilistic ranking which ensures that designs with low errors in prediction are chosen in favour to high error prone designs. The results obtained here look promising, however we need to conduct many more updates and GA based searches on the subsequently updated surrogate models to understand the efficiency of this model. Another improvement that can be made for the erosion problem set up is to replace the MCS in the inner array (over erosion parameters) by BMCS to estimate the performance statistics. We have shown in Chapter 5, that BMCS gives far better convergence for predicting mean and standard deviation than MCS.

The proposed algorithm can also be applied to the manufacturing uncertainty problem discussed in Chapter 5, however the implementation is more complex. Recall that we conducted a *Combined Array* experiment for that study and trained a Gaussian Process emulator to the observed dataset. This Gaussian emulator was then exploited to perform a BMCS over the noise variables (manufacturing uncertainty) at each design point. As shown in sec-

tion 5.2.1, this provided us with an estimate of the statistics of interest as Gaussian random variable which are given by equations 5.2 and 5.3.

$\langle \mathcal{M} \rangle$ in equation 5.3 can be interpreted as an estimate of the mean of the quantity of interest $Y(\boldsymbol{\xi})$ (which is pressure loss in our case), while $\sigma^2_{\mathcal{M}}$ is an estimate of the variance in the prediction of the mean. This provides us our first random objective (Gaussian) for probabilistic ranking, i.e. $\mathcal{F}_1 \sim \mathcal{N}(\langle \mathcal{M} \rangle, \sigma^2_{\mathcal{M}})$. Note that $\sigma^2_{\mathcal{M}}$ is not the estimate of the variance of the response which would be our second random objective. Using the same line of thought, we can train the Gaussian Process emulator for $Y^2(\boldsymbol{\xi})$ values and evaluate the estimates of variance as Gaussian random objectives to get the second random objective $\mathcal{F}_2 \sim \mathcal{N}(\langle \mathcal{M}' \rangle, \sigma^2_{\mathcal{M}'})$, where $\langle \mathcal{M}' \rangle$ is the estimate of the standard deviation of pressure loss and $\sigma^2_{\mathcal{M}'}$ is an estimate of the variance in the prediction of the standard deviation. Subsequently, we can conduct probabilistic dominance check over both the objectives during the Probabilistic NSGA-II search method. Next we summarize the major ideas presented in this chapter.

6.6 Summary

In this chapter we set out to alleviate the concerns raised in the previous chapter regarding lack of diversity in the Pareto set and wasting of computational effort in regions which may not be of interest to the designer. We developed a probabilistic framework for comparing solutions during the multiobjective optimization search. Probabilistic definitions for *Dominance* and *Pareto Set* were formulated and employed to rank random objectives in multiobjective optimization. An existing algorithm (NSGA-II) was extended to include probabilistic ranking and tested on a suite of test problems. The performance of the Probabilistic NSGA-II was compared to direct NSGA-II and the Gaussian process emulator assisted NSGA-II optimization method. The proposed algorithm was shown to have better performance in diversity as compared to a naive strategy where the Gaussian process emulator errors are ignored. The convergence metric for the Gaussian emulator assisted NSGA-II algorithm was better than probabilistic NSGA-II algorithm, as we observed Pareto clouds which adversely affected the convergence metric. To circumvent this issue we can add one more stage to the algorithm. After each iteration in Algorithm 1, we could separately rank (non-probabilistic dominance ranking) all the true function evaluations and employ the Pareto set thus obtained to evaluate the convergence metric. As there would be no more uncertainty in prediction, it would give us a better idea of the convergence. This would also ensure that we eventually get rid of

the Pareto cloud and get a final Pareto front for the designer to choose from. This method would be part of our future work.

Finally, the robust design against erosion problem was revisited and the Probabilistic NSGA-II optimization method was applied to it. The application of the proposed methodology to the manufacturing uncertainty problem was also discussed. It is important to mention the limitations of the proposed method. Probabilistic NSGA-II suffers from the curse of dimensionality; i.e. for high dimensional problems (e.g. $d > 12$), the algorithm would be very slow and the performance would also deteriorate. This can be attributed to the use of Gaussian Process emulators to approximate the objectives. A possible solution for this issue could be to employ local surrogates during the optimization search (Ong *et al.*, 2003). Sensitivity analysis methods, as discussed in section 5.3.3 could also be employed to help in managing this concern. They could be used to understand the impact of the design variables on the performance and the variables which have low influence could be dropped. This effectively would reduce the dimensionality of the problem and aid in improving the performance of the proposed algorithm. It should be noted however that, the Probabilistic NSGA-II method performs better than the simple Gaussian process assisted NSGA-II method for high dimensional problems as well.

Another point worth mentioning is that here we employed probabilistic ranking in conjunction with Gaussian Process emulators and NSGA-II. The idea of probabilistic dominance and Pareto set are not limited to use with these methods. It is a very general idea and could be employed to expedite multiobjective optimization in many other scenarios. We will elaborate on a few of these ideas in the future work section of the next chapter.

Chapter 7

Conclusions and Further Areas of Research

This chapter concludes the thesis with a brief synopsis of the main conclusions and contributions of the present research. Some directions for future research are also outlined.

7.1 Research Summary

The main focus of the present research was to seek compressor blades that are robust in the presence of erosion and manufacturing uncertainty. Novel parametric geometry methods for modeling eroded blades and simulating manufacturing uncertainty were presented. Probabilistic analysis on the baseline blade was conducted and a case for robust design was established for both the problems. Efficient multiobjective robust design methods based on multiobjective optimization and Gaussian Process emulators were proposed to seek blades with reduced sensitivities in the presence of geometric uncertainty. The main conclusions and contributions are summarized below.

Surrogate based Probabilistic Analysis of Compressor Blade

In this study we employed Gaussian Process emulators to expedite the probabilistic analysis of compressor blades in the presence of geometric uncertainty. We employed DOE techniques in conjunction with the parametric geometry models, automatic grid generator and high fidelity CFD solver to create training datasets. The Gaussian Process emulators were trained to the dataset and then verified using *Leave-one-Out* and *SCVR* validation techniques and if

needed improved by using additional data. These emulators were then used for probabilistic analysis using Monte Carlo and Bayesian Monte Carlo Simulation. The studies showed that blade performance may deteriorate significantly in presence of uncertainty due to erosion and manufacturing variations. High *Mean shift* and variation in performance in the presence of the above-mentioned uncertainty was observed. Main effect analysis was performed to understand the effect of each noise parameter on the design performance. This analysis could be very useful in understanding the critical *location, depth* and *width* of the erosion patterns on the blade. Sensitivity analysis was also conducted for the manufacturing uncertainty problem and the relative importance of each variable was quantified. The understanding of relative importance of variable can be very useful for reducing the dimension of the problem or perhaps performing a higher resolution analysis for the more important variables. The limitations of employing simplified geometry models and using small sample size for estimating the statistics in the erosion problem were discussed. These studies quantified the effect of uncertainty on performance, reemphasized the need for considering robustness during design and provided a framework for conducting robust design studies.

Robust Design Methodology

We first proposed an efficient multiobjective robust design methodology to seek compressor fan blades that are less sensitive to geometric uncertainty due to environment variables like erosion. An elitism based non-dominated sorting genetic algorithm was employed in conjunction with the Gaussian emulators to search the design space. A Pareto-optimal set was identified for trade off between the mean and standard deviation of the pressure loss. The predicted approximate Pareto-optimal set suggested by the NSGA-II, was verified using CFD simulations and few points were selected from the verified Pareto front for further analysis. Monte Carlo Simulation based on the surrogate models was executed for the selected blades and the results were found to be considerably better and more robust than the baseline design. The performance of the robust design was also compared to an optimal design obtained from a deterministic optimization method. A probabilistic study was conducted on a deterministically optimized blade and robust blade to compare their performance in presence of erosion. The deterministic optimal blade showed considerably larger deterioration in performance as compared to the robust blade. There was considerably lower shift in the mean performance for the robust geometry as compared to the deterministic optimal geometry.

For the manufacturing uncertainty problem we formulated a Bayesian Monte Carlo Simulation based robust design method. In this method we employed a combined array experiment over the design (control factors) and noise factors to create the initial training dataset. A Gaussian Process emulator was trained and verified using the standard validation techniques. The validated surrogate model was then exploited to conduct BMCS over the noise variables to estimate the performance statistics in the presence of manufacturing uncertainty. This model was then used in conjunction with NSGA-II to perform multiobjective optimization for seeking robust designs. A robust design from the converged Pareto set was chosen for comparison with deterministic optimal design. The robust design had better performance against manufacturing uncertainty than deterministic optimal design in all respects covering *Mean, Standard Deviation, Mean Shift* and *Worst Case Performance*. Significant computational savings for the robust design studies gained by employing efficient Gaussian emulators were also reported.

Finally, the issues involved in employing multiobjective optimization directly to Gaussian Stochastic Process model based surrogates were discussed. A probabilistic framework for dealing with random objectives during multiobjective search was developed. The definitions of dominance and Pareto optimality were extended to consider stochastic objectives and the concepts were implemented. A new algorithm, Probabilistic NSGA-II, was proposed and then tested on a suite of test functions. Comparison with the NSGA-II based direct search and Gaussian emulator assisted search methods were also performed and the trends showed that the proposed algorithm ensured better diversity. There was a need to improve the algorithm for better convergence and more analysis was required to understand the choice of *confidence factor*, which would make an important part of our future work. The proposed algorithm was then applied to the erosion problem and the manufacturing uncertainty problem was also briefly revisited.

Other Contributions

The research work reported in this thesis has led to the following publications

1. A. Kumar, A.J. Keane, P.B. Nair and S. Shahpar "Robust Design of Compressor Blades against Erosion" *Journal of Mechanical Design*, Volume 128, Issue 4, pp. 864-873, July 2006.

2. A. Kumar, P.B. Nair, A.J. Keane and S. Shahpar "Robust Design using Bayesian Monte Carlo" (*International Journal for Numerical Methods in Engineering* - Submitted for Review.

3. A. Kumar, P.B. Nair and A.J. Keane. "Probabilistic Multi-objective Optimization Algorithm with Gaussian Process Emulators" (In Preparation).

4. A. Kumar, A.J. Keane, P.B. Nair, and S. Shahpar. "Robust Design for Compressor Fan Blades against Manufacturing Variations" *In Proceedings of ASME IDETC/CIE*, DETC2006-99304, Pennsylvania, Sept 2006

5. A. Kumar, A.J. Keane, P.B. Nair and S. Shahpar. "Efficient Robust Design for Manufacturing Process Capability" *In Proceedings of 6th ASMO/ISSMO International Conference*, pp. 242-250, Oxford, July 2006

6. A. Kumar, A.J. Keane, P.B. Nair and S. Shahpar. "Efficient GA based Robust Design Method for Compressor Fan Blades" *In Proceedings of ASME IDETC/CIE*, DETC2005-85042, Long Beach (CA), September 2005.

7. A. Kumar, P.B. Nair, A.J. Keane and S. Shahpar. "Probabilistic Performance Analysis of Eroded Compressor Fan Blades" *In Proceedings of ASME Power Conference*, PWR2005-50070, Chicago, April 2005.

7.2 Future Research

Future research would involve improving the robust design methodologies proposed in this research. This would encompass improvements in uncertainty quantification and propagation using measured data and realistic geometry models. Enhancements to the multiobjective framework and exploitation of surrogates are also considered. Some directions for future work are outlined below

- In the present study for the erosion problem, Monte Carlo Simulation was employed for estimating the statistics over the noise variables. In the future work, it is proposed to replace it by Bayesian Monte Carlo Methods. The BMCS estimates can then be employed to train Gaussian Process emulators for multiobjective robust design search. It would be interesting to investigate the behaviour of the random objectives obtained, by approximating the BMCS estimates by the Gaussian Process emulators. The ideas

of probabilistic optimization can also be similarly extended to robust design of blades against manufacturing uncertainty.

- Improvements can also be made to the Probabilistic NSGA-II algorithm proposed in this study. A multiobjective optimization method was proposed, which takes into account the randomness in objectives due to approximating them by surrogate models. However, the objectives also contain uncertainty due to noise variables (erosion and manufacturing uncertainty) and the choice of design and noise samples. Methods for quantifying such uncertainties and exploiting them in a probabilistic framework for robust designs should be addressed. Future work could include development of a generic probabilistic multiobjective robust design framework for dealing with systems with different types of uncertainty models.

- Recall that we employed global surrogate models to replace the high fidelity CFD solver throughout this study. It is worth noting that global surrogates suffer from the curse of dimensionality and the computational cost incurred in the training phase scales as $(\mathcal{O}(N^3))$, where N is the number of points in the training dataset. Hence, construction of surrogate models can become computationally expensive for more than say a few thousand training points. It would be of interest to investigate how local surrogate models, that are built *on-the-fly* for each design point of interest, can be used to reduce the computational cost of surrogate model construction while increasing prediction accuracy.

- In the final chapter we observed that the choice of γ (*confidence factor*) had a very important role to play in the Probabilistic NSGA-II Algorithm. It would be important to carry out more numerical experiments to understand the effect of γ on the convergence metric and diversity of the approximate Pareto set. A good line of action to enhance the algorithm would be to add one more stage in each GA iteration, where all the true function evalauted points can be ranked. This would give a better indication of convergence and lead to Pareto fronts rather than Pareto clouds.

- A major limitation of the present work was the use of simplified models for uncertainty representation. Any methodology is as good as the inputs and an important area for future work could be improvements in uncertainty representation. There is therefore a pressing need to develop methods for assimilating data obtained through measurements to enhance probabilistic models used for representing the input uncertainties.

Appendix A

Program for Modeling Eroded Blades

The parametric eroded blade code was written in C programming language. The model was added as a module to the Rolls-Royce proprietary software PADRAM. We present here only the snippets of codes which were added during the implementation of the model. The changes were made to the code **ss04_convert.c** and the code is presented here.

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <signal.h>
#include <stdlib.h>
#include <math.h>
#include "ss04/ss04_om.h"
#include "ss04/ss04_aero.h"
#define SPLINE_ spline_
#define SEVAL_ seval_

extern OM ogvi;
/* Variables added for erosion model */
int index_min, index_max;
OM A,head,tail,mid,ogvf,ogvi_transrot,ogvi_dummy,ogvi_upper,
OM ogvi_lower, ogvi_norm, ogvi_new,LE_Points;

void convert(double *x2, double *y2, int iogv, int np, int npn1, int npn2)
{
   int              i, j, ii, ile, ite, ile2,ile3, ite2, ite3, npn;
   double          *snew, *snew2;
   /* Could have taken this as an external but
         np can be different for different DT6 sections */
   double           alfa, coe1, coe2, abet, eta;
   double           xle, xte, yle, yte, ss;
   double           xle2, xte2, yle2, yte2;
   double           xle3, xte3, yle3, yte3,ang1b,ang2b;
   double           value, value1, value2,*s, *spls, *splx, *splxb, *sply,
                   *splyb;
   int              m, npnb, nparc1s, nparc3s;
   int              imid1,imid2;
```

```
    double          arc0, arc1, arc2,rtod;
    double         *bufx, *bufy;
    AERO            ogv;
    FILE           *fptr = NULL;
    double         *s3,ds1,ds2;
    /* This is used for 3 arc segments for the middle arc */
    int                                          interp_type;
    /* flag for whether to use spline or linear interp*/
    double          chord,alf,mdash;
    int             npnb1,npnb2,npnb3,ifail,ifail1,ifail2,ifail3,index;
    double          arc01,arc02,arc03,*s31,*s32,*s33,penny_le_arc,penny_te_arc;
    double          x3,y3,xx1,yy1,xx2,yy2,xx,yy,cc,marc;
    double          maxs,mids1,mids2,mids3;
    int             npn1new, npn2new;
    FILE            *fp; /* fro erosion model*/
    FILE            *fptr_fod1, *fptr_shape; /*added for erosion model*/



    /* variables needed for erosion model */
   double peak_position, fod_width, location, peak_height, width, a[4], b[4];
   double *alpha_upper, *alpha_lower,theta,LE_Center_X,LE_Center_Y;
   double x1u,y1u,x2l,y2l,x3l,y3l,m1,m2,xc,yc,x0,y0,m0,min_x_lower,min_x_upper,max_x_lower;
   double max_x_upper,*theta_n,pi;
   int start_pt_index, end_pt_index, index_diff, af, temp_fod, FLAG_FOD, NPOINTS, r_points, nhh;
   int index_te_upper,index_te_lower,nlower,nupper,index_dummy,nhh_manf;

/* Read the input variables for erosion model */
 fptr_fod1 = fopen("INPUT_FOD.DAT","r");
                        fscanf(fptr_fod1, "%d \n",&FLAG_FOD);
        fscanf(fptr_fod1, "%d \n",&r_points);
                                fscanf(fptr_fod1, "%lf \n",&peak_height);
        fscanf(fptr_fod1, "%lf \n",&width);
        fscanf(fptr_fod1, "%lf \n",&location);
        fscanf(fptr_fod1, "%lf \n",&peak_position);
        fscanf(fptr_fod1, "%lf \n",&fod_width);
        fscanf(fptr_fod1, "%lf \n",&a[0]);
        fscanf(fptr_fod1, "%lf \n",&a[1]);
        fscanf(fptr_fod1, "%lf \n",&a[2]);
        fscanf(fptr_fod1, "%lf \n",&a[3]);
        fscanf(fptr_fod1, "%lf \n",&b[0]);
        fscanf(fptr_fod1, "%lf \n",&b[1]);
        fscanf(fptr_fod1, "%lf \n",&b[2]);
        fscanf(fptr_fod1, "%lf \n",&b[3]);
  fclose(fptr_fod1);

/* ..... Original code continues  ..... */
/* ..................................... */

/* Erosion model */

  if(FLAG_FOD == 1)
  {
        fp = fopen("ogvi_initial","w");
        for(i=1;i<=npn1+npn2;i++)
        {
                fprintf(fp,"\n %3.5f %3.5f",ogvi.x[i-1],ogvi.y[i-1]);
        }
        fclose(fp);

        /* Get the indices of the start and end points on the airfoil in the ogvi array,
        select af = 1 for upper airfoil and af =2 for lower airfoil*/
        af = 1;
        NPOINTS = 12 + r_points;
```

```
        return_indices(npn1, npn2, location, width, af, &start_pt_index, &end_pt_index);

        /* first decompose the whole ogvi array into three parts head, mid and tail*/
        decompose_ogvi(npn1, npn2, start_pt_index, end_pt_index);

        /* prepare the mid array and perturbed according to the Hick Hennes function*/
        introduce_fod(npn1, npn2, end_pt_index, start_pt_index, peak_height, &
        & peak_position, fod_width,a, NPOINTS);

        /* Now merge the three parts head mid and tail into a final ogvf array*/
        merge_ogvi(npn1, npn2, start_pt_index, end_pt_index);
        printf("\n start_pt_index = %d npn=%d end_pt_index=%d peak_h = %f", &
        & start_pt_index,ogvi.np,end_pt_index,peak_height);

        /* introduce changes in lowe surface with peak height = o */
        index_diff = end_pt_index - start_pt_index;
        npn1new = npn1-((end_pt_index - start_pt_index)+1)+NPOINTS;
        npn2new = npn2;
        af = 2;

        return_indices(npn1new, npn2new, location, width, &
        & af, &start_pt_index, &end_pt_index);
        peak_height=0.00000;
        start_pt_index = end_pt_index;
        end_pt_index = start_pt_index + index_diff;
        decompose_ogvi(npn1new, npn2new, start_pt_index, end_pt_index);
        introduce_fod(npn1new, npn2new, end_pt_index, start_pt_index, peak_height, &
        & peak_position, fod_width,b, NPOINTS);
        merge_ogvi(npn1new, npn2new, start_pt_index, end_pt_index);

        /* write out the eroded geometry */
        fp = fopen("ogvifinal","w");

        for(i=1;i<=ogvi.np;i++)
        {
                fprintf(fp,"\n %3.5f %3.5f",ogvi.x[i-1],ogvi.y[i-1]);
        }
        fclose(fp);

  }

/* ..... original code begins again ..... */
/* ..................................... */

free(snew), free(snew2);
  if (interp_type==0){ free(s);}
  else{ free(spls);free(splx);free(splxb);free(sply);free(splyb);}

   free(ogv.x);

   return;

/* end of main function */
}

/* functions added for erosion model */

/* returns indices for start and end of erosion */
void return_indices(int npn1, int npn2, double location, double width, int af, int *startindex, int *endindex)
{
        int i,wr,j,counter_start,counter_end;
        double pt[2],xchord,ychord,cm,temp,D,pwl,pcl;
        *startindex=0;
        *endindex=0;
```

```
        cm = (ogvi.y[npn1]-ogvi.y[0])/(ogvi.x[npn1]-ogvi.x[0]);
        pcl=location/100;
        xchord = (1-pcl)*ogvi.x[0] + pcl*ogvi.x[npn1];
        ychord = (1-pcl)*ogvi.y[0] + pcl*ogvi.y[npn1];

        D = fabs(ogvi.y[0] - ychord + (ogvi.x[0] - xchord)/cm);
        temp = D;
        pwl=width/100;
        if(af == 1)
        {
                counter_start = 1;
                counter_end = npn1;
        }
        else
        {
                counter_start = npn1;
                counter_end = npn1+npn2;
        }
        D = fabs(ogvi.y[counter_start - 1] - ychord + (ogvi.x[counter_start - 1] - xchord)/cm);
        temp = D;

        for(i=counter_start;i<counter_end;i++)
        {
                D = fabs(ogvi.y[i] - ychord + (ogvi.x[i] - xchord)/cm);
                if (D-temp < 0)
                {
                        temp = D;
                        *startindex = i;
                        fprintf(stderr,"\n D = %f start = %d",D,*startindex);
                }
        }
        pcl = pcl+pwl;
        xchord = (1-pcl)*ogvi.x[0] + pcl*ogvi.x[npn1];
        ychord = (1-pcl)*ogvi.y[0] + pcl*ogvi.y[npn1];

        D = fabs(ogvi.y[counter_start-1] - ychord + (ogvi.x[counter_start-1] - xchord)/cm);
        temp = D;
        for(j=counter_start;j<(counter_end);j++)
        {
                D = fabs(ogvi.y[j] - ychord + (ogvi.x[j] - xchord)/cm);
                if(D < temp)
                {
                        temp = D;
                        *endindex = j;
                        fprintf(stderr,"\n D = %f end = %d",D,*endindex);
                }
        }

}

/* merges different arrays of coordinates */
void merge_ogvi(int npn1, int npn2, int startindex, int endindex)
{
        int i,count;
        ogvf.np = head.np + mid.np + tail.np;

        ogvf.x = (double *)malloc(ogvf.np*sizeof(double));
        ogvf.y = (double *)malloc(ogvf.np*sizeof(double));

        count = 0;

        for(i=0;i<head.np;i++)
        {
                ogvf.x[count] = head.x[i];
```

```
                ogvf.y[count] = head.y[i];
                count ++;
        }
        for(i=0;i<mid.np;i++)
        {
                ogvf.x[count] = mid.x[i];
                ogvf.y[count] = mid.y[i];
                count ++;
        }
        for(i=0;i<tail.np;i++)
        {
                ogvf.x[count] = tail.x[i];
                ogvf.y[count] = tail.y[i];
                count ++;
        }

        free(ogvi.x);
        free(ogvi.y);

        ogvi.x = (double *)malloc(ogvf.np*sizeof(double));
        ogvi.y = (double *)malloc(ogvf.np*sizeof(double));

        ogvi.np = ogvf.np;

        for(i=1;i<=ogvi.np;i++)
        {
                ogvi.x[i-1] = ogvf.x[i-1];
                ogvi.y[i-1] = ogvf.y[i-1];

        }

        free(ogvf.x);
        free(ogvf.y);

}

/* decompose the airfoil into parts */
void decompose_ogvi(int npn1, int npn2, int startindex, int endindex)
{
        int i,length_A;

        A.np = endindex-startindex+1;
        A.x = (double *)malloc(A.np*sizeof(double));
        A.y = (double *)malloc(A.np*sizeof(double));

        for(i=0;i<A.np;i++)
        {
                A.x[i] = ogvi.x[startindex+i];
                A.y[i] = ogvi.y[startindex+i];
        }

        head.np = startindex;
        head.x = (double *)malloc(head.np*sizeof(double));
        head.y = (double *)malloc(head.np*sizeof(double));

        for(i=0;i<head.np;i++)
        {
                head.x[i] = ogvi.x[i];
                head.y[i] = ogvi.y[i];
        }

        tail.np = (npn1+npn2)-(endindex)-2;
        tail.x = (double *)malloc(tail.np*sizeof(double));
        tail.y = (double *)malloc(tail.np*sizeof(double));
```

```
        for(i=0;i<tail.np;i++)
        {
                tail.x[i] = ogvi.x[endindex+i+1];
                tail.y[i] = ogvi.y[endindex+i+1];
        }
}

/* create inputs for modeling erosion in a given range */
void linspace( double start, double end, int numpoints, double *x)
{
        double delta_x;
        int    i;
        delta_x = (end-start)/(numpoints-1);
        for(i=0;i<numpoints;i++)
                x[i] = start+i*delta_x;
}

/* Hich-Henne function */
double hickhenns(double x,  double peak_height, double peak_position, double fod_width)
{
        double y, f1,f2,pi;

        pi = 4*atan(1.0);
        f1 = log(0.5)/log(peak_position);
        f2 = sin(pi* pow(x, f1));
        y = peak_height*pow(f2,fod_width);
        return y;
}

/* function for finding the coefficients of any cubic polynomial given 4 conditions */
void calculate_coefficients_for_cubic_polynomial(int start_pt_index, int end_pt_index, double *a)
{
        double *Coeff_Matrix;
        int *ipivCoeff_Matrix;
        double x1, x2, y1, y2, m1, m2,num,den;
        int err;

        x1 = A.x[0];
        y1 = A.y[0];
        x2 = A.x[A.np-1];
        y2 = A.y[A.np-1];
        m1 = (A.y[1]-A.y[0])/(A.x[1]-A.x[0]);
        m2 = (A.y[A.np-1]-A.y[A.np-2])/(A.x[A.np-1]-A.x[A.np-2]);


        Coeff_Matrix = (double *)malloc(16*sizeof(double));
        ipivCoeff_Matrix = (int *)malloc(4*sizeof(double));
        Coeff_Matrix[0] = 1;
        Coeff_Matrix[1] = x1;
        Coeff_Matrix[2] = x1*x1;
        Coeff_Matrix[3] = x1*x1*x1;

        Coeff_Matrix[4] = 1;
        Coeff_Matrix[5] = x2;
        Coeff_Matrix[6] = x2*x2;
        Coeff_Matrix[7] = x2*x2*x2;

        Coeff_Matrix[8] = 0;
        Coeff_Matrix[9] = 1;
        Coeff_Matrix[10] = 2*x1;
        Coeff_Matrix[11] = 3*x1*x1;

        Coeff_Matrix[12] = 0;
```

```
        Coeff_Matrix[13] = 1;
        Coeff_Matrix[14] = 2*x2;
        Coeff_Matrix[15] = 3*x2*x2;

        // Preparing the RHS
        a[0] = y1;
        a[1] = y2;
        a[2] = m1;
        a[3] = m2;

        num = 2*m2*(x2-x1) - 2*(y2-y1) - (x1-x2)*(m1-m2);
        den = (x2-x1)*(pow(x2,2)-2*x1*x2+pow(x1,2));

        a[3] = num/den;

        num = (m1-m2)-(3*a[3]*(pow(x1,2)-pow(x2,2)));
        den = 2*(x1-x2);

        a[2] = num/den;

        num = (y2-y1) - a[2]*(pow(x2,2)-pow(x1,2)) - a[3]*(pow(x2,3)-pow(x1,3));


        a[1] = num/(x2-x1);

        a[0] = y1 - (a[1]*x1) - (a[2]*pow(x1,2)) - (a[3]*pow(x1,3));

        free(Coeff_Matrix);
        free(ipivCoeff_Matrix);
}


/* introduce erosion in the airfoil */
void introduce_fod(int npn1,int npn2, int end_pt_index, int start_pt_index, double peak_height,...&
&...double peak_position, double fod_width, double *a, int NPOINTS)
{
        double slope1, slope2, m1, m2;
        double cx, cy, r0, pi, cm, LOC;
        double inpoint_x[NPOINTS], inpoint_y[NPOINTS],Xnew[NPOINTS],Ynew[NPOINTS],mnew[NPOINTS],theta[NPOINTS];
        double theta_min, theta_max, angle, r[NPOINTS];
        int    i;
        FILE *fp;

        //calculate_coefficients_for_cubic_polynomial(start_pt_index, end_pt_index, a);
        pi = 4*atan(1);
        // Calculating the slopes of the tangents
        slope1=(A.y[1]-A.y[0])/(A.x[1]-A.x[0]);
        slope2=(A.y[A.np-1]-A.y[A.np-2])/(A.x[A.np-1]-A.x[A.np-2]);

        // Find the new X
        linspace(A.x[0],A.x[A.np-1], NPOINTS, Xnew);

        // Calculate the Ynew and slope of the respective normals using the cubic polynomial
        for (i=0; i<NPOINTS; i++)
        {
                Ynew[i] = a[0] + a[1]*Xnew[i] + a[2]*pow(Xnew[i],2) + a[3]*pow(Xnew[i],3);
                mnew[i] = a[1] + 2*a[2]*Xnew[i] + 3*a[3]*pow(Xnew[i],2);
                theta[i] = atan(-1/mnew[i]);
        }

        //find the hickhenne deviations and deviate the cubic polynomial in the normal directions
        linspace(0, 1, NPOINTS, inpoint_x);
        mid.np = NPOINTS;
        mid.x = (double *)malloc(mid.np*sizeof(double));
```

```
        mid.y = (double *)malloc(mid.np*sizeof(double));

        for (i=0; i<NPOINTS; i++)
        {
                inpoint_y[i] = hickhenns(inpoint_x[i],  peak_height, peak_position, fod_width);
                mid.x[i] = Xnew[i] - fabs(inpoint_y[i]*cos(theta[i]));
                mid.y[i] = Ynew[i] - fabs(inpoint_y[i]*sin(theta[i]));
        }

}

/* Find the TE cusp and split the aiirfoil in two halves */
void split_airfoil(int n1, int n2)
{
        int i;
        double a, b, theta;
        double x,y;
        ogvi_transrot.x = (double *)malloc((n1+n2)*sizeof(double));
        ogvi_transrot.y = (double *)malloc((n1+n2)*sizeof(double));
        ogvi_upper.x = (double *)malloc(n1*sizeof(double));
        ogvi_upper.y = (double *)malloc(n1*sizeof(double));
        ogvi_lower.x = (double *)malloc((n2)*sizeof(double));
        ogvi_lower.y = (double *)malloc((n2)*sizeof(double));

        /* Added by apu 20th Nov 2005 */

        a = ogvi.x[0];
        b = ogvi.y[0];
        theta = fabs( atan( (b - ogvi.y[n1-1])/ (a - ogvi.x[n1-1]) ) );

        /* Perform Translation and Rotation */
        for(i=0;i<(n1+n2); i++)
        {
                x = ogvi.x[i];
                y = ogvi.y[i];

                ogvi_transrot.x[i] = ((x-a)*cos(theta) - ((y-b)*sin(theta)));
                ogvi_transrot.y[i] = ((x-a)*sin(theta) + ((y-b)*cos(theta)));
        }


        for (i=0; i<n1; i++)
        {
                ogvi_upper.x[i] = ogvi_transrot.x[i];
                ogvi_upper.y[i] = ogvi_transrot.y[i];
                /*ogvi_upper.x[i] = ogvi.x[i];
                ogvi_upper.y[i] = ogvi.y[i];*/
        }
        for (i=n1; i<n1+n2; i++)
        {
                ogvi_lower.x[i-n1] = ogvi_transrot.x[i];
                ogvi_lower.y[i-n1] = ogvi_transrot.y[i];
                /*ogvi_lower.x[i-n1] = ogvi.x[i];
                ogvi_lower.y[i-n1] = ogvi.y[i];*/
        }

        free(ogvi_transrot.x);
        free(ogvi_transrot.y);
}

/* find the extrema using search method */
void search_extrema(double *x, int n, double *min, double *max)
{
        int i;
```

```
        *min = x[0];
        index_min=1;
        *max = x[0];
        index_max=1;
        for(i=1;i<n;i++)
        {
                if(x[i]<(*min))
                {
                        *min = x[i];
                        index_min = i;
                }
                if(x[i]>(*max))
                {
                        *max = x[i];
                        index_max = i;
                }
        }

        printf("\n I am in search extrema ");
        printf("\n index_min = %d min_value = %E ", index_min, *min);
        printf("\n index_max = %d max_value = %E ", index_max, *max);
        getchar();
}

/* normalize the airfoil*/

void normalize_airfoil(OM a, int n)
{
        int i;
        double min_x, min_y, max_x, max_y;
        search_extrema(a.x,n,&min_x,&max_x);
        ogvi_norm.x = (double *)malloc(n*sizeof(double));
        ogvi_norm.y = (double *)malloc(n*sizeof(double));

        for (i=0; i<n; i++)
        {
                ogvi_norm.x[i] = (a.x[i]-min_x)/(max_x - min_x);
                ogvi_norm.y[i] = a.y[i];
        }
}
```

Appendix B

Program for Modeling Manufacturing Uncertainty

The manufacturing uncertainty modeling code was written in *C* programming language. The model was added as a module to the Rolls-Royce proprietary software PADRAM. We present here only the snippets of codes which were added during the implementation of the model. The changes were made to the code **ss04_convert.c** and the code is presented here.

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <signal.h>
#include <stdlib.h>
#include <math.h>
#include "ss04/ss04_om.h"
#include "ss04/ss04_aero.h"
#define SPLINE_ spline_
#define SEVAL_ seval_

extern OM ogvi;

/* Variables added for manufacturing model */
OM A,ogvf,ogvi_transrot,ogvi_dummy,ogvi_upper, ogvi_lower, ogvi_norm;

void convert(double *x2, double *y2, int iogv, int np, int npn1, int npn2)
{
   int              i, j, ii, ile, ite, ile2,ile3, ite2, ite3, npn;
   double          *snew, *snew2;
   /* Could have taken this as an external but
            np can be different for different DT6 sections */
   double           alfa, coe1, coe2, abet, eta;
   double           xle, xte, yle, yte, ss;
   double           xle2, xte2, yle2, yte2;
   double           xle3, xte3, yle3, yte3,ang1b,ang2b;
   double           value, value1, value2,*s, *spls, *splx, *splxb, *sply,
                   *splyb;
   int              m, npnb, nparc1s, nparc3s;
   int              imid1,imid2;
   double           arc0, arc1, arc2,rtod;
```

```
    double          *bufx, *bufy;
    AERO             ogv;
    FILE            *fptr = NULL;
    double          *s3,ds1,ds2;
    /* This is used for 3 arc segments for the middle arc */
    int                                          interp_type;
    /* flag for whether to use spline or linear interp*/
    double           chord,alf,mdash;
    int              npnb1,npnb2,npnb3,ifail,ifail1,ifail2,ifail3,index;
    double           arc01,arc02,arc03,*s31,*s32,*s33,penny_le_arc,penny_te_arc;
    double                        x3,y3,xx1,yy1,xx2,yy2,xx,yy,cc,marc;
    double           maxs,mids1,mids2,mids3;
    int                                    npn1new, npn2new;
    FILE                                   *fp; /* for manufacturing uncertainty model*/
    FILE                                   *fptr_manf; /*added for manufacturing uncertainty model*/



  /* variables needed for manufacturing uncertainty model */
  double *alpha_upper, *alpha_lower,theta,LE_Center_X,LE_Center_Y;
  double x1u,y1u,x2l,y2l,x3l,y3l,m1,m2,xc,yc,x0,y0,m0,min_x_lower,min_x_upper,max_x_lower;
  double max_x_upper,*weights_manf,slope_tangent,slope_normal,*theta_n,pi;
  int r_points, nhh, FLAG_SHAPE,FLAG_MANF;
  int index_te_upper,index_te_lower,nlower,nupper,index_dummy,nhh_manf;

  /* Read the input variables for manufacturing uncertainty model */
                                FLAG_MANF=1;
                                        fptr_manf = fopen("INPUT_MANF.DAT","r");
                                        fscanf(fptr_manf,"%d \n",&nhh_manf);
                                        weights_manf = (double *)malloc(nhh_manf*sizeof(double));

                                        for(i=0;i<nhh_manf;i++)
                                                                fscanf(fptr_manf,"%lf \n",&weights_manf[i]);
                                        fclose(fptr_manf);


/* ................ Original code continues  ................. */
/* ........................................................... */

/* manufacturing uncertainty model */

  /* Model manufacturing variations (including LE and Chord) using combination of Hicks-Henne functions*/
        fp = fopen("ogvi_initial","w");
        for(i=1;i<=npn1+npn2;i++)
        {
                fprintf(fp,"\n %3.5f %3.5f",ogvi.x[i-1],ogvi.y[i-1]);
        }
        fclose(fp);

  if(FLAG_MANF == 1)
  {
        pi = 4*atan(1.0);
        x1u = ogvi.x[1];
        y1u = ogvi.y[1];
        x2l = ogvi.x[npn1+npn2-3];
        y2l = ogvi.y[npn1+npn2-3];
        x3l = ogvi.x[npn1+npn2-4];
        y3l = ogvi.y[npn1+npn2-4];

        m1 = (y1u-y2l)/(x1u-x2l);
        m2 = (y1u-y3l)/(x1u-x3l);

        xc = (((x1u+x3l)/(2*m2)) - ((x1u+x2l)/(2*m1)) - ((y2l-y3l)/2))*(m1*m2/(m1-m2));
        yc = ((-1/m2)*(xc - ((x1u+x3l)/2))) + ((y1u+y3l)/2);
```

```
printf("\n The Centre of the leading edge circle is (%3.5f, %3.5f)",xc,yc);

/* Align the co-ordinate system such that origin is at (x0,y0)
and the new x-axis is in the direction of the radius */
x0 = ogvi.x[0];
y0 = ogvi.y[0];

m0 = (y0-yc)/(x0-xc);
theta = fabs(atan(m0));

ogvi_transrot.x = (double *)malloc((npn1+npn2)*sizeof(double));
ogvi_transrot.y = (double *)malloc((npn1+npn2)*sizeof(double));

fp = fopen("ogvitransrotdata","w");
for(i=0; i< (npn1+npn2-1) ; i++)
{
        ogvi_transrot.x[i] = ((ogvi.x[i]-x0)*cos(theta)) - ((ogvi.y[i]-y0)*sin(theta));
        ogvi_transrot.y[i] = ((ogvi.x[i]-x0)*sin(theta)) + ((ogvi.y[i]-y0)*cos(theta));
        fprintf(fp,"%3.6f %3.6f\n",ogvi_transrot.x[i],ogvi_transrot.y[i]);
}
fclose(fp);
/* Place a dummy value at the last element of the array */
ogvi_transrot.x[npn1+npn2-1] = 0.0;
ogvi_transrot.y[npn1+npn2-1] = 0.0;


/* It is known that in the new coordinate system the trailing edge is parallel to y' */
for(i=npn1-5;i<(npn1+5);i++)
{
        printf("%E\n",fabs(ogvi_transrot.x[i]-ogvi_transrot.x[i+1]));
        //getchar();
        if(fabs(ogvi_transrot.x[i]-ogvi_transrot.x[i+1]) <= 2*pow(10,-6) )
        {
                index_te_upper = i;
                index_te_lower = i+1;
                break;
        }
}

/* Create a dummy ogvi_dummy which contains 0<=x<=1 and y=0 and its length is npn1+npn2-1 */

ogvi_dummy.x = (double *)malloc((npn1+npn2-1)*sizeof(double));
ogvi_dummy.y = (double *)malloc((npn1+npn2-1)*sizeof(double));

linspace(0,1,npn1+npn2-1,ogvi_dummy.x);
for(i=1;i<=npn1+npn2-1;i++)
        ogvi_dummy.y[i-1] = 0.0;

/*Call shape to insert combination of nhh hickshennes functions in ogvi_dummy */
shape(ogvi_dummy,npn1+npn2-1,nhh_manf,weights_manf);
fp = fopen("ogvi_dummy","w");
for(i=0; i< (npn1+npn2-1) ; i++)
{
        fprintf(fp,"%3.6f %3.6f\n",ogvi_dummy.x[i],ogvi_dummy.y[i]);
}
fclose(fp);

theta_n = (double *)malloc((npn1+npn2-1)*sizeof(double));
index_dummy = 0;
for(i=0;i<=index_te_upper;i++)
{
/*Find the direction of tangent and normal at the point and make deviations along the normal*/
        if((ogvi_transrot.x[i]-ogvi_transrot.x[i+1])==0)
                theta_n[i] = theta_n[i-1];
```

```
        else
        {
                slope_tangent = (ogvi_transrot.y[i]-ogvi_transrot.y[i+1])/...&
                &....(ogvi_transrot.x[i]-ogvi_transrot.x[i+1]);

                if(slope_tangent == 0)
                        theta_n[i] = theta_n[i-1];
                else
                {
                        slope_normal = -1/slope_tangent;

                        if (slope_tangent < 0)
                                theta_n[i] = atan(slope_normal);
                        else
                                theta_n[i] = pi + atan(slope_normal);
                }
        }
        ogvi_transrot.x[i] = ogvi_transrot.x[i] + ogvi_dummy.y[index_te_upper-i]*cos(theta_n[i]);
        ogvi_transrot.y[i] = ogvi_transrot.y[i] + ogvi_dummy.y[index_te_upper-i]*sin(theta_n[i]);

        index_dummy ++;
}
index_dummy=0;
for(i=index_te_lower;i<=npn1+npn2-2;i++)
{
/*Find the direction of tangent and normal at the point and make deviations along the normal*/
        if((ogvi_transrot.x[i]-ogvi_transrot.x[i+1])==0)
                theta_n[i] = theta_n[i-1];
        else
        {
                slope_tangent = (ogvi_transrot.y[i]-ogvi_transrot.y[i+1])/....&
                &...(ogvi_transrot.x[i]-ogvi_transrot.x[i+1]);

                if(slope_tangent == 0)
                        theta_n[i] = theta_n[i-1];
                else
                {
                        slope_normal = -1/slope_tangent;

                        if (slope_tangent < 0)
                                theta_n[i] = atan(slope_normal);
                        else
                                theta_n[i] = pi + atan(slope_normal);
                }
        }
        ogvi_transrot.x[i] = ogvi_transrot.x[i] - ....&
        &...ogvi_dummy.y[npn1+npn2-2-index_dummy]*cos(theta_n[i]);
        ogvi_transrot.y[i] = ogvi_transrot.y[i] - ....&
        &...ogvi_dummy.y[npn1+npn2-2-index_dummy]*sin(theta_n[i]);

        index_dummy ++;
}

fp = fopen("ogvitransrotdata_manf","w");
for(i=0; i< (npn1+npn2-1) ; i++)
{
        fprintf(fp,"%3.6f_%3.6f\n",ogvi_transrot.x[i],ogvi_transrot.y[i]);
}
fclose(fp);

/* Untranslate and Unrotate to get back to original coordinate system */

for(i=0; i< (npn1+npn2-1) ; i++)
{
```

```
                ogvi.x[i] = x0 + ((ogvi_transrot.x[i]*cos(theta)) + (ogvi_transrot.y[i]*sin(theta)) );
                ogvi.y[i] = y0 + ((ogvi_transrot.y[i]*cos(theta)) - (ogvi_transrot.x[i]*sin(theta)) );
        }
                        ogvi.x[318] = ogvi.x[0];
                        ogvi.y[318] = ogvi.y[0];

        fp = fopen("ogvi_manf","w");
        for(i=0; i<= (npn1+npn2-1) ; i++)
        {
                fprintf(fp,"%3.5f %3.5f\n",ogvi.x[i],ogvi.y[i]);
        }
        fclose(fp);


        free(theta_n);
        free(ogvi_dummy.x);
        free(ogvi_dummy.y);
}

/* ..... original code begins again ..... */
/* ..... */

free(snew), free(snew2);
  if (interp_type==0){ free(s);}
  else{ free(spls);free(splx);free(splxb);free(sply);free(splyb);}

   free(ogv.x);

   return;

/* .....end of main function..... */
}


/* functions added for manufacturing uncertainty model */


/* decompose the airfoil into parts (lower surface and upper surface) */
void decompose_ogvi(int npn1, int npn2, int startindex, int endindex)
{
        int i,length_A;

        A.np = endindex-startindex+1;
        A.x = (double *)malloc(A.np*sizeof(double));
        A.y = (double *)malloc(A.np*sizeof(double));

        for(i=0;i<A.np;i++)
        {
                A.x[i] = ogvi.x[startindex+i];
                A.y[i] = ogvi.y[startindex+i];
        }

        head.np = startindex;
        head.x = (double *)malloc(head.np*sizeof(double));
        head.y = (double *)malloc(head.np*sizeof(double));

        for(i=0;i<head.np;i++)
        {
                head.x[i] = ogvi.x[i];
                head.y[i] = ogvi.y[i];
        }

        tail.np = (npn1+npn2)-(endindex)-2;
        tail.x = (double *)malloc(tail.np*sizeof(double));
```

```
        tail.y = (double *)malloc(tail.np*sizeof(double));

        for(i=0;i<tail.np;i++)
        {
                tail.x[i] = ogvi.x[endindex+i+1];
                tail.y[i] = ogvi.y[endindex+i+1];
        }
}

/* create inputs in a given range */
void linspace( double start, double end, int numpoints, double *x)
{
        double delta_x;
        int    i;
        delta_x = (end-start)/(numpoints-1);
        for(i=0;i<numpoints;i++)
                x[i] = start+i*delta_x;
}

/* Hich-Henne function */
double hickhenns(double x,  double peak_height, double peak_position, double fod_width)
{
        double y, f1,f2,pi;

        pi = 4*atan(1.0);
        f1 = log(0.5)/log(peak_position);
        f2 = sin(pi* pow(x, f1));
        y = peak_height*pow(f2,fod_width);
        return y;
}

/* function for finding the coefficients of any cubic polynomial given 4 conditions */
void calculate_coefficients_for_cubic_polynomial(int start_pt_index, int end_pt_index, double *a)
{
        double *Coeff_Matrix;
        int *ipivCoeff_Matrix;
        double x1, x2, y1, y2, m1, m2,num,den;
        int err;

        x1 = A.x[0];
        y1 = A.y[0];
        x2 = A.x[A.np-1];
        y2 = A.y[A.np-1];
        m1 = (A.y[1]-A.y[0])/(A.x[1]-A.x[0]);
        m2 = (A.y[A.np-1]-A.y[A.np-2])/(A.x[A.np-1]-A.x[A.np-2]);


        Coeff_Matrix = (double *)malloc(16*sizeof(double));
        ipivCoeff_Matrix = (int *)malloc(4*sizeof(double));
        Coeff_Matrix[0] = 1;
        Coeff_Matrix[1] = x1;
        Coeff_Matrix[2] = x1*x1;
        Coeff_Matrix[3] = x1*x1*x1;

        Coeff_Matrix[4] = 1;
        Coeff_Matrix[5] = x2;
        Coeff_Matrix[6] = x2*x2;
        Coeff_Matrix[7] = x2*x2*x2;

        Coeff_Matrix[8] = 0;
        Coeff_Matrix[9] = 1;
        Coeff_Matrix[10] = 2*x1;
        Coeff_Matrix[11] = 3*x1*x1;
```

```
        Coeff_Matrix[12] = 0;
        Coeff_Matrix[13] = 1;
        Coeff_Matrix[14] = 2*x2;
        Coeff_Matrix[15] = 3*x2*x2;

        // Preparing the RHS
        a[0] = y1;
        a[1] = y2;
        a[2] = m1;
        a[3] = m2;

        num = 2*m2*(x2-x1) - 2*(y2-y1) - (x1-x2)*(m1-m2);
        den = (x2-x1)*(pow(x2,2)-2*x1*x2+pow(x1,2));

        a[3] = num/den;

        num = (m1-m2)-(3*a[3]*(pow(x1,2)-pow(x2,2)));
        den = 2*(x1-x2);

        a[2] = num/den;

        num = (y2-y1) - a[2]*(pow(x2,2)-pow(x1,2)) - a[3]*(pow(x2,3)-pow(x1,3));


        a[1] = num/(x2-x1);

        a[0] = y1 - (a[1]*x1) - (a[2]*pow(x1,2)) - (a[3]*pow(x1,3));

        free(Coeff_Matrix);
        free(ipivCoeff_Matrix);
}



/* Find the TE cusp and split the aiirfoil in two halves */
void split_airfoil(int n1, int n2)
{
        int i;
        double a, b, theta;
        double x,y;
        ogvi_transrot.x = (double *)malloc((n1+n2)*sizeof(double));
        ogvi_transrot.y = (double *)malloc((n1+n2)*sizeof(double));
        ogvi_upper.x = (double *)malloc(n1*sizeof(double));
        ogvi_upper.y = (double *)malloc(n1*sizeof(double));
        ogvi_lower.x = (double *)malloc((n2)*sizeof(double));
        ogvi_lower.y = (double *)malloc((n2)*sizeof(double));

        /* Added by apu 20th Nov 2005 */

        a = ogvi.x[0];
        b = ogvi.y[0];
        theta = fabs( atan( (b - ogvi.y[n1-1])/ (a - ogvi.x[n1-1]) ) );

        /* Perform Translation and Rotation */
        for(i=0;i<(n1+n2); i++)
        {
                x = ogvi.x[i];
                y = ogvi.y[i];

                ogvi_transrot.x[i] = ((x-a)*cos(theta) - ((y-b)*sin(theta)));
                ogvi_transrot.y[i] = ((x-a)*sin(theta) + ((y-b)*cos(theta)));
        }
```

```
        for (i=0; i<n1; i++)
        {
                ogvi_upper.x[i] = ogvi_transrot.x[i];
                ogvi_upper.y[i] = ogvi_transrot.y[i];
                /*ogvi_upper.x[i] = ogvi.x[i];
                ogvi_upper.y[i] = ogvi.y[i];*/
        }
        for (i=n1; i<n1+n2; i++)
        {
                ogvi_lower.x[i-n1] = ogvi_transrot.x[i];
                ogvi_lower.y[i-n1] = ogvi_transrot.y[i];
                /*ogvi_lower.x[i-n1] = ogvi.x[i];
                ogvi_lower.y[i-n1] = ogvi.y[i];*/
        }

        free(ogvi_transrot.x);
        free(ogvi_transrot.y);
}

/* find the extrema using search method */
void search_extrema(double *x, int n, double *min, double *max)
{
        int i;

        *min = x[0];
        index_min=1;
        *max = x[0];
        index_max=1;
        for(i=1;i<n;i++)
        {
                if(x[i]<(*min))
                {
                        *min = x[i];
                        index_min = i;
                }
                if(x[i]>(*max))
                {
                        *max = x[i];
                        index_max = i;
                }
        }

        printf("\n I am in search_extrema ");
        printf("\n index_min = %d min_value = %E ", index_min, *min);
        printf("\n index_max = %d max_value = %E ", index_max, *max);
        getchar();
}

/* normalize the airfoil*/

void normalize_airfoil(OM a, int n)
{
        int i;
        double min_x, min_y, max_x, max_y;
        search_extrema(a.x,n,&min_x,&max_x);
        ogvi_norm.x = (double *)malloc(n*sizeof(double));
        ogvi_norm.y = (double *)malloc(n*sizeof(double));

        for (i=0; i<n; i++)
        {
                ogvi_norm.x[i] = (a.x[i]-min_x)/(max_x - min_x);
                ogvi_norm.y[i] = a.y[i];
        }
}
```

Appendix C

Estimation of Probability using erfc

We need to evaluate the probability that a Gaussian random variable Z is greater than zero ($\mathscr{P}(\mathcal{Z} > 0)$). This is graphicaly shown in the figure C1. The shaded area can be evaluated

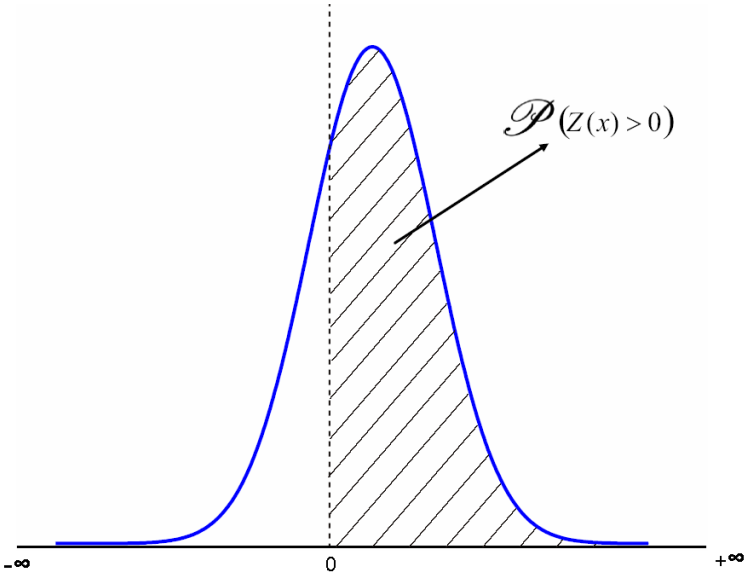


FIGURE C.1: Shaded area under the probability density function is the probability to be evaluated

to give us the desired probability, this can be expressed as

$$\mathscr{P}(\mathcal{Z} > 0) = \int_0^\infty \frac{1}{\sigma_\mathcal{Z}\sqrt{2\pi}} e^{-\frac{(\mathcal{Z}-m_\mathcal{Z})^2}{2\sigma_\mathcal{Z}^2}} \mathrm{d}\mathcal{Z}. \tag{C.1}$$

After substituting $\frac{\mathcal{Z}-m_\mathcal{Z}}{\sigma_\mathcal{Z}\sqrt{2}} = t$, the right hand side of the above equation can be written as

$$\begin{aligned}\int_0^\infty \frac{1}{\sigma_{\mathcal{Z}}\sqrt{2\pi}} e^{-\frac{(\mathcal{Z}-m_{\mathcal{Z}})^2}{2\sigma_{\mathcal{Z}}^2}} \mathrm{d}\mathcal{Z} &= \int_{\frac{-m_{\mathcal{Z}}}{\sigma_{\mathcal{Z}}\sqrt{2}}}^\infty \frac{1}{\sqrt{\pi}} e^{-t^2} \mathrm{d}t \\ &= \tfrac{1}{2}\mathrm{erfc}(\tfrac{-m_{\mathcal{Z}}}{\sigma_{\mathcal{Z}}\sqrt{2}}).\end{aligned} \quad , \tag{C.2}$$

where erfc(x) is defined as

$$\mathrm{erfc(x)} = \int_x^\infty \frac{2}{\sqrt{\pi}} e^{-t^2} \mathrm{d}t. \tag{C.3}$$

We can now numerically estimate the integral in equation C.2. This function is available in most of the standard scientific tool boxes like Matlab.

References

Alexandrov, N.M., Dennis, J.E., Lewis, R.M. & Torczon, V. (1998). A Trust region framework for managing the use of approximation models in optimization. *Structural Optimization*, **15**, 16–23.

Allen, M. & Maute, K. (2004). Reliability-based design optimization of aeroelastic structures. *Structural and Multidisciplinary Optimization*, **27**, 228–242.

Athan, T. & Papalambros, P. (1996). A note on weighted criteria methods for compromise solutions in multi-objective optimization. *Engineering Optimization*, **27**, 155–176.

Audze, P. & Eglais, V., eds. (1977). *Problems of Dynamics and Strength*, vol. 35, 104–107. (in Russian).

Balan, C. & Tabakoff, W. (1984). Axial flow compressor performance deterioration. *AIAA 84-2108*.

Ben-Haim & Elishakoff, I. (1990). *Convex Models of Uncertainty in Applied Mechanics*. Elsevier Science.

Ben-Tal & Nemirovski, A. (1997). Robust truss topology design via semidefinite programming. *SIAM Journal of Optimization*, **7**, 991–1016.

Bothe, D. (2001). *Measuring Process Capability*. Landmark Publishing, Cedarburg.

Box, G. (1988). Signal-to-noise ratios, performance criteria, and trnaformations (with discussions). *Technometrics*, **30**, 1:17.

Box, G. & Draper, N. (1987). *Emperical model building and response surfaces*. Wiley, New York.

Box, G. & Jones, S. (1992). Designing products that are robust to the environment. *Total Quality Management*, **3**, 265–282.

Boyles, R.A. (1991). The taguchi capability index. *Journal of Quality Technology*, **23**, 17–26.

Bragg, M.B. (1986). An experimental study of the aerodynamics of a naca0012 airfoil with simulated glaze ice. *AIAA-85-0484*.

Bragg, M.B. & Khodadoust, A. (1989). Effect of simulated glaze ice on a rectangular wing. *AIAA-89-0750*.

Chan, K., Saltelli, A. & Tarantola, S. (1997). Sensitivity analysis of model output: can the variance-based methods make the difference? In D.H.W. S. Andradottir K. J. Healy & B.L. Nelson, eds., *1997 Winter Simulation Conference*, 261–268, WSC, Atlanta.

Chandu, S. & Grandhi, R. (1995). General purpose procedure for reliability based structural optimization under parametric uncertainty. *Advanced Engineering Software*, **23**, 7–14.

Chase, K.W. & Parkinson, A.R. (1991). A survey of research in the application of tolerance analysis to the design of mechanical assemblies. *Research in Engineering Design*, **3**, 23–37.

Chen, W., Jin, R. & Sudjianto, A. (2005). Analytical variance-based global sensitivity analysis in simulation-based design under uncertainty. *ASME Journal of Mechanical Design*, **127**, 875–884.

Cressie, N. (1998). Spatial prediction and ordinary kriging. *Mathematical Geology*, **20**, 405–421.

Das, I. (2000). Robustness optimization for constrained nonlinear programming problems. *Engineering Optimization*, **32**, 585–618.

Deb, K. (1995). *Optimization for engineering design*. Prentice-Hall, New delhi.

Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test functions. *Evolutionary Computation*, **7**, 205–230.

Deb, K. & Jain, S. (2002). Running performance metrics for evolutionary multi-objective optimization. In *Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning*, 13–20, SEAL'02), Singapore.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions of Evolutionary Computation*, **6**, 182–197.

Dennis, J.E. & Torczon, V. (1997). Managing approximation models in optimization. In N.M. Alexandrov & N. Hussaini, eds., *Multidisciplinary Design Optimization: State-of-the-Art*, 330–347, SIAM, Philadelphia, PA.

Drela, M. (1998). Pros and cons of airfoil optimization. *Proc.Frontiers of computational fluid dynamics, World Scientific*.

Egorov, I. (1992). Optimization of a multistage axial compressor. stochastic approach. *ASME 92-GT-163*.

Egorov, I. & Kretinin, G. (1993). Optimization of gas turbine engine elements by probability criteria. *ASME 93-GT-191*.

Emmerich, M.T.M., Giannakoglou, K.C. & Naujoks, B. (2006). Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, **10**, 421–439.

Fonseca, C. & Fleming, P. (1993). Genetic algorithm for multi-objective optimization: Formulation, discussion and generalization. In S. Forrest, ed., *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA.

Fonseca, C.M. & Fleming, P.J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, **3**, 1–16.

Garzon, V.E. (2003). *Probabilistic Aerothermal Design of Compressor Airfoils*. PhD Thesis, Massachusetts Institute of Technology.

Garzon, V.E. & Darmofal, D.L. (2001). Using computational fluid dynamics in probabilistic engineering design. *AIAA 2001-2526*.

Garzon, V.E. & Darmofal, D.L. (2003). Impact of geometric variability on axial compressor performance. *ASME Turbo Expo GT2003-38130*.

Gentle, J. (1998). *Random Number Generator and Monte Carlo Methods*. Springer, Berlin.

Girard, A. (2004). *Approximate Methods for Propagation of Unceratainty with Gaussian Process Models*. PhD Thesis, University of Glasgow.

Giunta, A., Wojtkiewicz(Jr), S.F. & Eldred, M. (2003). Overview of modern design of experiments methods for computational simulation. *AIAA 2003-0649*.

Hamed, A., Tabakoff, W. & Singh, D. (1998). Modeling of compressor performance deterioration due to erosion. *International Journal of Rotating Machinery*, **4**, 243–248.

Harry, M. (1997). *The nature of Six Sigma Quality*. Motorola University Press, Shaumburg, IL.

Haylock, R. & O'Hagan, A. (1996). On inference for outputs of computationally expensive algorithms with uncertainty on the inputs. In J. Bernardo, ed., *Bayesian Statistics*, vol. 5, 629–37, Oxford University Press, London.

Hedayat, A., Sloane, N. & Stufken, J. (1999). *Orthogonal Arrays: Theory and Applications*. Springer, New York.

Helton, J., Johnson, J. & Oberkampf, W.L. (2004). An exploration of alternative approaches to the representation of uncertainty in model predictions. *Reliability Engineering and System Safety*, **85**, 39–71.

Hicks, R.M. & Henne, P. (1978). Wing design by numerical optimisation. *Journal of Aircraft*, **15**, 407–412.

Horn, J., Nafploitis, N. & Goldberg, D. (1994). A niched pareto genetic algorithm for multi-objective optimization. In Z. Michalewicz, ed., *Proceedings of the First IEEE Conference on Evolutionary Computation*, 82–87, Piscataway, NJ.

Hurtado, J.E. & Barbat, A.H. (1998). Monte carlo techniques in computational stochastic mechanics. *Arch. Comput. Meth. Engrg*, **5**, 3–30.

Huyse, L. (2001). Solving problems of optimization under uncertainty as statistical decision problem. *AIAA 2001-1519*.

Huyse, L. & Lewis, R. (2001). Aerodynamic shape optimization of two-dimensional airfoils under uncertain operating conditions. Technical Report NASA/CR-2001-210648, NASA Langley Research Center, Hampton, VA.

Huyse, L., Padula, S., Lewis, R.M. & Li, W. (2002). Probabilistic approach to free-form airfoil shape optimization under uncertainty. *AIAA Journal*, **40**, 1764–1772.

Jin, Y. & Branke, J. (2003). Tradeoff between performance and robustness: An evolutionary multiobjective approach. In *Evolutionary Multi-Criterion Optimization*, LNCS 2632, 237–251, Berlin, Germany: Springer Verlag.

Jin, Y., Olhofer, M. & Sendhoff, B. (2003). A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, **6**, 481–494.

Jones, D.R., Schonlau, M. & Welch, W.J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, **13**, 455–492.

Keane, A. (2003a). Wing optimization using design of experiment, response surface, and data fusion methods. *Journal of Aircraft*, **40**, 741–750.

Keane, A. (2006). Statistical improvement criteria for use in multiobjective design optimization. *AIAA Journal*, **44**, 879–891.

Keane, A.J. (2003b). The options design exploration system. Reference Manual and User Guide, at http://www.soton.ac.uk/ ajk/options.ps.

Keane, A.J. & Nair, P.B. (2005). *Computational Approaches for Aerospace Design*. John Wiley and Sons.

Koehler, J. & Owen, A. (1996). Computer experiments. In S. Ghosh & C. Rao, eds., *Handbook of Statistics*, 261–308, Elsevier.

Koski, J. (1985). Defectiveness of weighting method in multicriterion optimization of structures. *Communications is Applied Numerical Methods*, **1**, 333–337.

Kotz, S. & Johnson, N.L. (2002). Process capability indices - a review. *Journal of Quality Technology*, **34**, 2–19.

Kotz, S. & Lovelace, C. (1998). *Introduction to Process Capability Indices*. Arnold, London.

Kumar, A., Keane, A.J., Nair, P.B. & Shahpar, S. (2006). Robust design method of compressor fan blades againt erosion. *ASME Journal of Mechanical Design*, **128**, 864–873.

Lamb, C. (2005). *Probabilistic Performance-Based Geometric Tolerancing of Compressor Blades.*. Master Thesis, Massachusetts Institute of Technology.

Li, W., Huyse, L. & Padula, S. (2002). Robust airfoil optimization to achieve drag reduction over a range of mach numbers. *Structural and Multidisciplinary Optimization*, **24**, 38–50.

Li, W., Krist, S. & Campbell, R. (2006). Transonic airfoil shape optimization in preliminary design environment. *Journal of Aircraft*, **43**, 639–651.

MacKay, D.J.C. (1997). Guassian processes - a replacement for supervised neural networks. Tutorial lecture Notes for NIPS, at http://www.inference.phy.cam.ac.uk/mackay/abstracts/gp.html.

Mackay, D.J.C. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

Madsen, H., Krenk, S. & Lind, N. (1986). *Methods of Structural Safety*. Prentice-Hall, Inc, Englewook Cliffs, NJ.

Matheron, G. (1963). Principles of geostatistics. *Economic Geology*, **58**, 1246–1266.

Mavris, D. & Bandte, O. (1997). A probabilistic approach to multivariate constrained robust design simulation. *SAE-975508, Society of Automative Engineers*.

Mavris, D., Macsotai, N. & Roth, B. (1998). A probabilistic design methodology for commercial aircraft engine cycle selection. *SAE-985510, Society of Automative Engineers*.

Mckay, M.D., Conover, W.J. & Beckman, R.J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, **21**, 239–245.

Meckesheimer, M., Booker, A.J., Barton, R.R. & Simpson, T. (2002). Computationally inexpensive metamodel assessment strategies. *AIAA Journal*, **40**, 2053–2060.

Metwally, M., Tabakoff, W. & Hamed, A. (1995). Blade erosion in automotive gas turbine engine. *Journal of Engineering for Gas Turbine and Power*, **117**, 213–219.

Moinier, P. & Giles, M.B. (1998). Preconditioned euler and navier-stokes calculation on unstructured grids. *6th ICFD Conference on Numerical Methods for Fluid Dynamics, Oxford, UK.*.

Moinier, P., Muller, J.D. & Giles, M.B. (1999). Edge-based multigrid and preconditioning for hybrid grids. *AIAA-99-3339*.

Myers, R., Khuri, A. & Vining, G. (1992). Response surface alternatives to the taguchi robust parameter design approach. *The American Statistician*, **46**, 131–139.

Myers, R.H. & Montgomery, D.C. (2002). *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley Series in Probability and Statistics, John Wiley Sons, New York.

Nair, P.B., Choudhary, A. & Keane, A.J. (2001). A bayesian framework for uncertainty analysis using deterministic black-box simulation code. *AIAA-2001-1676*.

Nair, V.N. (1992). Taguchi's parameter design: A panel discussion. *Technometrics*, **34**, 127–161.

Neal, R.M. (1996). *Bayesian Learning for Neural Networks*. Springer.

Oberkampf, W., Diegert, K., Alvin, K. & Rutherford, B. (1998). Variability, uncertainty and error in computational simulation. *AIAA/ASME Joint Thermophysics and Heat Transfer Conference, ASME-HTD*, **357**, 259–272.

O'Hagan, A. (1987). Monte carlo is fundamentally unsound. *The Statistician*, **36**, 247–249.

O'Hagan, A. & Oakley, J. (2004). Probability is perfect, but we can't elicit it perfectly. *Reliability Engineering and Safety System*, **85**, 239–248.

O'Hagan, A., Kennedy, M. & Oakley, J. (1999). Uncertainty analysis and other inference tools for complex compute codes (with discussion). *Bayesian Statistics*, **6**, 503–524.

Ong, Y., Nair, P., Keane, A. & Zhou, Z. (2004). Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In Y. Jin, ed., *Knowledge Incorporation in Evolutionary Computation*, Studies in Fuzziness and Soft Computing, Springer Verlag.

Ong, Y.S., Nair, P.B. & Keane, A.J. (2003). Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, **40**, 687–696.

Pan, J., Loth, E. & Bragg, M.B. (2003). RANS simulation of airfoils with ice shapes. *AIAA 2003-0729*.

Pande, P., Neuman, R. & Cavanagh, R. (2000). *The Six Sigma way: how GE, Motorola, and other top companies are honing their performance*. McGraw-Hill, New York.

Pareto, V. (1896). *Cours D'Economie Politique*, vol. I and II. F. Rouge, Lausanne.

Parkinson, D. (1997). Robust design by variability optimization. *Quality and Reliability Engineering International*, **13**, 97–102.

Phadke, M.S. (1989). *Quality Engineering using Robust Design*. Prentice Hall, New Jersey.

Putko, M., Newman, P., III, A.T. & Green, L. (2001). Approach for uncertainty propogation and robust design in cfd using sensitivity derivatives. *AIAA 2001-2528*.

Rao, S. & Chen, L. (1998). Numerical solution of fuzzy linear equations in engineering analysis. *International Journal for Numerical Methods in Engineering*, **43**, 391–408.

Rasmussen, C. (2003). Gaussian processes to speed up hybrid monte carlo for expensive bayesian integrals. In M.B. J.M. Bernardo & M. West, eds., *Bayesian Statistics*, vol. 7, 651–659, Oxford University Press, London.

Rasmussen, C. & Ghahramani, Z. (2003). Bayesian monte carlo. In S.T. Suzanna Becker & K. Obermayer, eds., *Advances in Neural Information Processing Systems*, vol. 15, 489–496, MIT Press, Boston.

Roberts, W.B. (1984). Axial compressor performance restoration by blade profile control. *AIAA-84-GT-232, ASME*.

Robinson, D.G. (1998). A survey of probabilistic methods used in reliability, risk and uncertainty analysis: Analytical Techniques I. Technical Report SAND98-1189, Sandia National Laboratories, Livermore, California.

Sacks, J., Welch, W.J., Mitchell, T.J. & Wynn, H.P. (1989). Design and analysis of computer experiments. *Statistical Sciences*, **4**, 409–435.

Saltelli, A., Andres, T. & Homma, T. (1993). Sensitivity analysis model output: an investigation of new techniques. *Computational Statistics and Data Analysis*, **15**, 211–238.

Sanchez, S. (2000). Robust design: Seeking the best of all possible worlds. *Institute of Electrical and Electronic Engineers; Pisctaway, NJ*, in Proceedings of the 1994 Winter Simulation Conferece.

Schonlau, M. (1997). *Computer Experiments and Global Optimization*. PhD Thesis, University of Waterloo.

Shahpar, S. & Lapworth, L. (2003). Parametric design and rapid meshing systems for turbomachinery optimisation. *ASME-GT2003-38698*.

Shoemaker, A., Tsui, K. & Wu, C. (1991). Economical experimentation methods for robust parameter design. *Technometrics*, **33**, 415–427.

Sobol', I.M. (1993). Sensitivity Analysis for nonlinear mathematical models. *Mathematical modeling and Computational Experiment*, **1**, 407–414.

Sobol, I.M. (1994). *A Primer for the Monte Carlo Method*. CRC Press, Boca Raton.

Sobol', I.M. (2001). Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*, **55**, 271–280.

Sobol, I.M. & Stanikov, R. (1981). *The Choice of Optimal Parameters in Multicriteria Problems*. Nauka, Russia.

Statnikov, R.B. & Matusov, J.B. (2002). *Multicriteria Analysis in Engineering.*. Kluwer Academic Publishers.

Taguchi, G. (1986). *Introduction to Quality Engineering*. UNIPUB/Krauss International, New York.

Taguchi, G. & Wu, Y. (1980). *Introduction to Off-Line Quality Control*. Central Japan Quality Control Association, Nagoya, Japan.

Tata, M. & Thornton, A. (1999). Process capability database usage in industry: Myth vs reality. *Proceedings of ASME Design Engineering Technical Conference DEC99/DFM-8968*.

Trosset, M.W. (1996). Taguchi and Robust Design. Technical Report 31, Department of Computational and Applied Mathematics, Rice University, Houston, Texas.

Trosset, M.W., Alexandrov, N.M. & Watson, L.T. (2003). New methods for robust design using computer simulations. In *Proceedings of the Section on Physical and Engineering Science*, 4287–4291, American Statistical Association, Alexendria, VA.

Tsui, K. (1994). Avoiding unnecassary bias in robust design analysis. *Computational Statistics and Data Analysis*, **18**, 535–546.

Tsui, K. (1996). A critical look at taguchi's modelling approach for robust design. *Journal of Applied Statistics*, **23**, 81–95.

Tsutsui, S. & Ghosh, A. (1997). Genetic algorithms with a robust solution searching scheme. *IEEE Transactions on Evolutionary Computation*, **1**, 201–208.

Tsutsui, S., Ghosh, A. & Fujimoto, Y. (1996). A robust solution searching scheme in genetic search. In *Proceedings of Parallel Problem Solving in Nature*, 543–552.

Vapnik, V. (1998). *Statistical Learning Theory*. Wiley, New York.

Welch, W. & Sacks, J. (1991). A system for quality improvement via computer experiments. *Communications in Statistics-Theory and Methods*, **20**, 477–495.

Welch, W., Yu, T., Kang, S. & Sacks, J. (1990). Computer experiments for quality control. *Journal of Quality Technology*, **22**, 15–22.

Wu, H., Yang, S. & Liu, F. (2003). Comparison of three geometric representation of airfoils for aerodynamic optimization. *AIAA 2003-4095*.

Zang, T., Hemsch, M., Hilburger, M., Kenny, S.P., Luckring, J., Maghami, P., Padula, S. & Stroud, W. (2002). Needs and Opportunities for Uncertainty-Based Multidisciplinary Design Methods for Aerospace Vehicles. Technical Report NASA/TM-2002-211462, NASA Langley Research Center, Hampton, VA.

Zingg, D.W. & Elias, S. (2006). Aerodynamic optimization under a range of operating conditions. *AIAA Journal*, **44**, 2787–2792.

Zitzler, E., Deb, K. & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Emperical results. *Evolutionary Computation Journal*, **8**, 125–148.