Inexpensive Uncertainty Analysis for CFD Applications

Devendra Ghate

St Hugh's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

2014

ABSTRACT

The work presented in this thesis aims to provide various tools to be used during design process to make maximum use of the increasing availability of accurate engine blade measurement data for high fidelity fluid mechanic simulations at a reasonable computational expense.

A new method for uncertainty propagation for geometric error has been proposed for fluid mechanics codes using adjoint error correction. Inexpensive Monte Carlo (IMC) method targets small uncertainties and provides complete probability distribution for the objective function at a significantly reduced computational cost. A brief literature survey of the existing methods is followed by the formulation of IMC. An example algebraic model is used to demonstrate the IMC method. The IMC method is extended to fluid mechanic applications using Principal Component Analysis (PCA) for reduced order modelling. Implementation details for the IMC method are discussed using an example airfoil code. Finally, the IMC method has been implemented and validated for an industrial fluid mechanic code HYDRA.

A consistent methodology has been developed for the automatic generation of the linear and adjoint codes by selective use of automatic differentiation (AD) technique. The method has the advantage of keeping the linear and the adjoint codes in-sync with the changes in the underlying nonlinear fluid mechanic solver. The use of various consistency checks have been demonstrated to ease the development and maintenance process of the linear and the adjoint codes. The use of AD has been extended for the calculation of the complete Hessian using forward-on-forward approach. The complete mathematical formulation for Hessian calculation using the linear and the adjoint solutions has been outlined for fluid mechanic solvers. An efficient implementation for the Hessian calculation is demonstrated using the airfoil code.

A new application of the Independent Component Analysis (ICA) is proposed for manufacturing uncertainty source identification. The mathematical formulation is outlined followed by an example application of ICA for artificially generated uncertainty for the NACA0012 airfoil.

Keywords: Uncertainty analysis, Monte Carlo, Adjoint error correction, manufacturing uncertainty, Automatic Differentiation, Independent Component Analysis

ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to Prof. Mike Giles who has patiently supervised my work throughout the thesis. It has been a long winding journey for me which would not have been possible without his guidance and encouragement. It has been an extraordinary opportunity to work under his guidance.

I also thank Dr. Mihai Duta for making my introduction to HYDRA and PADRAM effortless and also for numerous discussions regarding numerical analysis in general. My sincere gratitude to Dr. Leigh Lapworth and Dr. Shahrokh Shahpaur (from Rolls-Royce Plc.) for their prompt and extensive help with the queries concerning HYDRA and PADRAM.

I am grateful to EPSRC, Rolls-Royce Plc. and Clarendon Trust for supporting me financially during these years. I am also indebted to St. Hughs College for making my stay at Oxford memorable.

I would also like to thank my examiners Prof. Jens-Dominik Müller and Prof. Endre Süli for taking the time to read the thesis and many insightful comments on the work.

Finally, I wish to thank my parents and my brother for their constant support through these years.

LIST OF SYMBOLS

ROMAN SYMBOLS

A	Matrix with each column containing the leading modes of PCA in the decreasing order of variance
a_k	k^{th} column of A representing the k^{th} leading mode of PCA
c	Computational cost of one evaluation of the objective function
C	Computational cost of one iterative nonlinear fluid mechanic solution
C_p	Coefficient of pressure on the airfoil surface
C_x	Covariance matrix of the matrix with centred error vectors corresponding to blade measurements ($\tilde{X}$)
D	Mixing matrix in the ICA model
f	Nonhomogeneous term of the linear flow equation
g	Nonhomogeneous term of the adjoint flow equation
J	Nonlinear objective function
L	Disrete linearised flow equations
m	Dimension of the fluid mechanic solver
n	Number of blade/airfoil measurements
n_r	Number of leading modes used in the reduced order model (based on PCA) of the geometric uncertainty
N	Number of sample points in a Monte Carlo simulation
p	Number of mesh points used for the discretisation in the fluid mechanic solver
R	Vector of discrete nonlinear residuals of the flow equations
S	Matrix with each column representing an independent source of manufacturing variability
x^0	Vector of nominal(ideal) surface blade geometry
$\tilde{x}_i$	Vector of centred errors corresponding to the i$^{\text{th}}$ blade measurement
x	Vector of linear perturbations in the mesh Coordinates

X	Vector of mesh coordinates
$\tilde{X}$	Matrix with i$^{\text{th}}$ column containing the centred set of error vector corresponding to the i$^{\text{th}}$ set of blade measurements
u	Vector of linear perturbations in the flow variables
U	Vector of flow variables
v	Vector of adjoint variables over the entire mesh domain
z, Z	Vector of baseline solution that includes mesh coordinates as well as flow solution

GREEK SYMBOLS

α	Vector of independent design variables
α_i	The i^{th} sample point of the design variable for the Monte Carlo simulations
$\alpha_{k,s}$	The s^{th} sample point of the k^{th} design variable for the Monte Carlo simulations
σ	Standard deviation of a probability distribution
μ	Mean of a probability distribution

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 BACKGROUND

The effect of uncertainty on the performance of an aircraft engine has been a topic of study for the last couple of decades. Uncertainty is defined here as any deviation from the baseline design conditions even if the deviation is predictable and quantifiable. Uncertainty may be introduced in the form of manufacturing variability, in-service wear-and-tear and the operational conditions. Every stage in the life cycle of an engine blade introduces uncertainty in the design process.

A baseline design (also referred to as ideal design) is arrived at by optimising for the baseline design conditions. An engine design cycle typically involves optimisation for the baseline design conditions followed by an estimate of the performance variation in the neighbourhood of the baseline design. This method of engine design is termed as the *deterministic design process*. These estimates are mainly arrived at by

- empirical formulas,
- rules of thumb accumulated over the years, and
- selective experimentation.

The first two approaches are low fidelity and the last approach is prohibitively expensive and time consuming for a detailed analysis. Since rigorous bounds on the performance variation of the baseline design are not available, large safety factors are used in the design process resulting in overall suboptimal performance.

In recent years, the trend has been to move from deterministic design to *robust design* and *reliability-based design*. Robust design optimises for a measure of the performance over a range of input parameters as opposed to optimising the performance for baseline conditions. A variety of different designs can be achieved depending on the measure of performance used for optimisation. The two benefits of robust design over the traditional deterministic design are – a better overall performance of engine over its life cycle and a more quantitative approach in defining the deterioration in performance away from the baseline conditions. Reliability based design algorithms are a special case of robust design algorithms in which the

measure of performance is defined in a way as to guarantee a minimum/maximum performance with a given statistical probability[1].

However, optimisation algorithms in general, and robust design algorithms in particular are computationally expensive. Typically robust design algorithms are used with low fidelity fluid mechanic models. A single high fidelity simulation is performed on the design obtained at the end of a robust design algorithm. Such design processes are termed here as the *mixed fidelity design* processes. Until recently, the limitations on the available computing power made it impossible to use high fidelity simulations throughout the design process. Such design processes introduce modelling uncertainty in the design process due to the various approximations and assumptions introduced by low fidelity simulations[2].

1.2 MOTIVATION

There is a need to move towards the use of *high fidelity* simulation techniques throughout the design process to reduce modelling uncertainty. One of the major initiatives in this direction is the research carried out by the NASA Multi-disciplinary Optimisation group[3] which looked at methodologies to introduce a systematic and high fidelity design optimisation approach in the aerospace industry. Some of the key drivers behind the increasing acceptance of this approach are affordable computing power and high fidelity measurement technologies.

1.2.1 *Computing Power*

It is no longer difficult to use high fidelity simulation techniques routinely in a design process with the steady rise of affordable computing power. All the major research institutes and aerospace companies have extensive computing power at their disposal. In addition to the exponential growth in the floating-point operations per second (FLOPS) of computers, the data storage capacity of the computing systems has also improved considerably. The improved large storage capacity is critical in maintaining large databases of design data for engineers. Availability of affordable, scalable and

[1] The concept of six-sigma design is a good example of reliability-based designs.

[2] A classic example of mixed fidelity approach in fluid mechanics is the use off Navier-Stokes equations for the baseline design followed by Euler simulations for the robust design algorithms. This introduces modelling uncertainty in the design process in the form of ignored viscous effects

[3] `mdob.larc.nasa.gov`

energy efficient computing is set to accelerate even further with the recent advent of Graphics Processing Unit (GPU) computing. Because of the availability of these computing facilities, large scale simulations are being carried out on a scale not possible before. The availability of increased computing power has lead to the development of software tools for efficient management, access and allocation of large-scale distributed heterogeneous hardware resources. On the systems software side, intelligent database management tools that can handle terabytes of data on a distributed system have become important. One such recent initiative is the Grid Enabled Optimisation and Design Search Engineering (GEODISE)[4] project undertaken at the Southampton University. The culmination of all these advances in the computing technology has put the onus back on the design community to develop design algorithms that make effective use of this computing power to systematically incorporate high fidelity simulations in the design process.

1.2.2 *Measurement Technology*

Presently, most measurement techniques for engine blades are only designed to avoid a few key defects in the manufactured components. Ultrasonic [90] and infrared scanning techniques are used to measure a few key locations on the blade surface. It is found that the exact geometry of blades is not one of the most important features. Häcker [49] reported that blades are measured at 30 to 40 points and inspected with automated visual inspection techniques [83] to check their fulfilment of certain tolerances. Similarly, Thakur et al. [89] reported the use of blade measurements where each blade measurement consisted of 18 ultrasonic minimum wall thicknesses, six each across the tip, mid-span and root cross-sections. As the geometry measurements are defined using only a few parameters (low fidelity measurement data), it is not possible to do a meaningful high fidelity fluid mechanics analysis.

However, with the increasing availability of cheaper and more accurate measurement devices, a large set of measurement data is expected to become available for designers. Very high quality measurements can be taken with the introduction of new technologies. Currently there are three major techniques for workpiece (engine blade) surface acquisition: range imaging, coordinate measuring machines (CMM), and computer tomography (CT). Range imaging with laser sensors [50] can acquire three-dimensional surface coordinate data quickly, while touch probe CMM [14] can

[4] www.geodise.org

acquire three dimensional surface coordinate data accurately. Both techniques can only capture the external surface. On the other hand, CT can non-destructively capture both external and internal surface data [96]. Thakur et al. [90] (§5.1) have pointed out that high density nickel alloy blades require highly powered X-ray CT scanners. These are presently expensive and the measurement data needs to be post-processed to filter out the scanning errors.

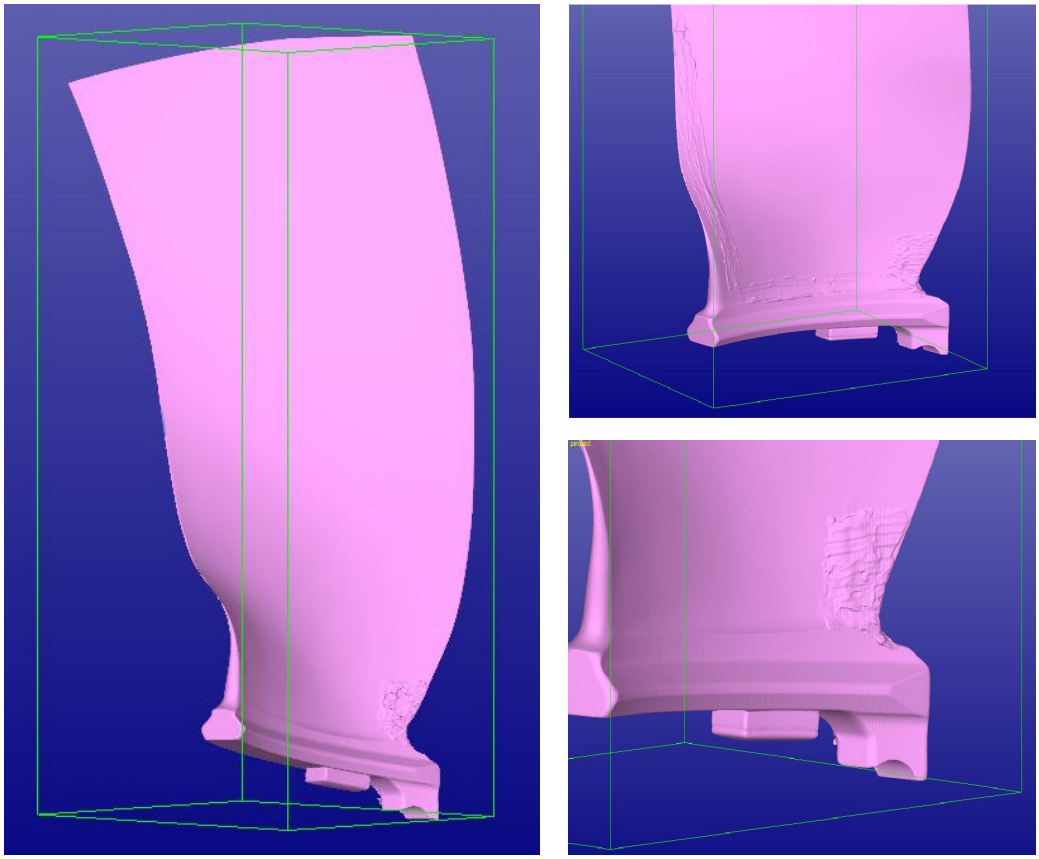

Figure 1.1 Laser scanned BR710 fan blade visualised using DeskArtes. The surface definition consists of triangulation using 542,886 mesh points defined in STL format.

These methods allow accurate measurement of the position of points on the complex surface of engine blades. A large number, 1000 or more, of sample measurements can be acquired in a reasonable time.

Figure 1.1 shows a laser scanned BR710 fan blade (provided by Rolls Royce Plc.) visualised using DeskArtes®. The surface definition consists of triangulations with 542,886 points provided in the stereolithography (STL) format[5]. The surface definition

is accurate enough to clearly show the extra material left behind after the casting process. This gives a flavour for the high fidelity measurements that will be available in large datasets in the future.

As the scanning devices become more affordable, large numbers of high fidelity measurements are expected to become readily available across the organisation for designers. For example, Garzon [30, 31] used a large set of aircraft engine rotor blade measurements from one of the major engine manufacturers for the work presented in his dissertation.

It should be possible in the near future to keep a complete measurement log of a single engine blade from manufacturing to the end of its life span. These measurements can be used effectively at the design stage to develop robust aircraft engines with minimal performance variability to geometric perturbations. The availability of this data has made it possible to perform a more accurate design analysis, and assessment of the effects of geometric perturbations.

It should be noted here that an engine design cycle involves application of multiple disciplines like structural mechanics, fluid mechanics and acoustics. Hence the uncertainty analysis has to be performed for all the different disciplines. The work presented in this dissertation specifically targets fluid mechanics applications. However, many of the ideas demonstrated using fluid mechanics solvers may find applications and can be extended to other disciplines.

1.3 UNCERTAINTY PROPAGATION

Detailed surveys [63, 93] have been carried out concerning the various uncertainties associated with fluid mechanics. Among others, Luckring et al. [63] proposed a rigorous methodology to quantify uncertainty at various stages of a fluid mechanics design. A very good survey and implementation guide of various methods for uncertainty propagation in fluid mechanics has been reported by Walters and Huyse [93]. The article discusses the effect of modelling uncertainty, geometric uncertainty, various numerical errors introduced in the solution process and the uncertainty introduced by boundary condition implementations. The article also discusses uncertainty propagation methods like Monte Carlo simulations and the moment methods.

[5] The STL surface measurements can be easily imported in MATLAB® for further analysis.

A large amount of research has also been done at the Aerospace Systems Design Laboratory (ASDL)[6], Georgia Institute of Technology to develop a framework for quantifying, managing and propagating uncertainty through the engine design cycle. In particular, they have developed a method [25] for cheap Monte Carlo simulations that propagates the entire probability distribution function (PDF) through the process. In the case of nonlinear functions, it is not always enough to look at the mean and the standard deviation. The shape and the spread of the entire probability distribution may be of interest in certain situations. This is elaborated in more detail in chapter two of this dissertation.

Though all the above studies are of generic nature, the work presented in this dissertation intends to look at methods to account for geometric uncertainty introduced by manufacturing variability.

1.3.1 *Geometric Uncertainty*

Gumbert et al. [48] have reported a study on the effect of geometric uncertainty on the wing design. The work reports the use of first order moment methods to propagate the geometric uncertainty through fluid mechanics solvers for optimisation. They have concluded that geometric uncertainty is a significant factor in deciding the active constraints during the optimisation process.

Garzon [30] proposed a method for complete probabilistic design and optimisation considering manufacturing variability in the aircraft engine blades. Garzon optimised a set of blade airfoils for design conditions and another set of airfoils for minimum loss variability under geometric uncertainty. A successful application of Principal Component Analysis (PCA) [57] is demonstrated to reduce the number of geometry design variables to a manageable level without sacrificing much variability. 30 to 40% reduction in the variability of polytropic efficiency of multi-stage compressor models can be achieved using the second set of airfoils under geometric uncertainty. Though all simulation models used in this study are low fidelity, the effect is expected to be of the similar scale for production grade high fidelity simulation models. Häcker [49] has also reported a framework for robust aerothermal design of engine blades using Monte Carlo simulations.

In his thesis, Kumar [59] proposed a robust design method for optimisation of compressor blades using parametrisation of geometric uncertainty arising from

[6] www.asdl.gatech.edu

erosion and manufacturing variability. Different sets of parametrisation are used for the two different types of uncertainty sources because of their different leading modes. Kumar has provided a detailed study of various sampling techniques, Monte Carlo simulation models and surrogate modelling methods. The study conclusively proves that geometric uncertainty, if unaccounted for, may lead to significant deterioration and even failure at off design points. On the other hand, compressor blades designed using robust design methodologies lead to a much better over-all performance.

Thakur et al. [90] reported use of 1052 blade measurements to study the effects of manufacturing variability on the Intermediate Pressure (IP) engine blade life. Each blade measurement consisted of 18 ultrasonic minimum wall thicknesses, six each across the tip, mid-span and root cross-sections. After removing measurement error using various techniques, free form deformations are used to reconstruct a set of representative engine blade geometries that capture the measured manufacturing variability. These representative geometries are then used for life and stress calculation using finite element analysis.

Bui-Thanh et al. [10] have demonstrated a method to propagate input geometric uncertainty through a two dimensional unsteady aerodynamic flow solver for small geometric perturbations. The analysis quantifies the effects of small variations in the blade geometry on the blade forced response for a subsonic rotor blade. PCA based reduced order model for input geometric uncertainty is used in conjunction with a reduced order model for the unsteady flow solver based on Petrov-Galerkin projection. This reduced order model is then used to carry out a Monte Carlo simulation for small geometric variations. The reduced order model thus derived is problem specific and has to be regenerated for every configuration.

It can be safely concluded from the above studies that geometric uncertainties significantly affect the engine performance and should be incorporated in the industrial aircraft engine design cycle. Also, the most popular approaches for geometric uncertainty propagation are Monte Carlo simulations and moment methods.

The Monte Carlo method using high fidelity simulations is computationally expensive. The estimator of the mean converges as $\mathcal{O}(\frac{1}{\sqrt{n}})$ for Monte Carlo methods with random sampling where n is the number of samples. Hence, a large number of samples are required for a satisfactory estimate. Various sampling strategies and quasi Monte Carlo methods can be introduced to reduce the number of sampling points. However, the computational cost is still prohibitively high. This problem is managed for

fluid mechanics simulations using two strategies. A low fidelity model can be used instead of a high fidelity model thereby reducing the computational cost. A standard example of this approach is the use of Euler equation solutions for Monte Carlo simulations instead of Navier-Stokes models. Another approach is to use coarse mesh calculations instead of fine mesh calculations that are better able to resolve fine flow features. This move from high fidelity simulation to low fidelity simulations result in the loss of essential flow characteristics.

The second strategy is to use a surrogate model [59] or a reduced order model [10] of the underlying high fidelity fluid mechanic solution. This surrogate model is then sampled for the Monte Carlo simulations. The number of high fidelity fluid mechanic simulations used to generate the surrogate model can be fine-tuned depending on the required accuracy for the surrogate model or the degree of nonlinearity in the design space.

1.4 AREAS OF INTEREST

The three key areas of interest for uncertainty propagation are high fidelity fluid mechanic simulations, reliable gradient information and reduced order modelling.

1.4.1 *High Fidelity Simulations*

Most of the studies [10, 25, 30, 49, 60] have used academic low fidelity fluid mechanic simulations for design and optimisation. Once the robust design has been obtained a single high fidelity analysis is performed to validate the design. However, it is preferable to use high fidelity fluid mechanics solvers including the complex phenomenon like turbulence modelling in all industrial design exercises.

The other alternative is to use stochastic partial differential equations [70, 97] which can handle large uncertainties and high nonlinearities. They are accurate and can handle large non-linearities if a converged solution is obtained. However, they are also relatively difficult to implement and all the effort invested may not be justified for smaller uncertainties.

The work presented in this dissertation proposes a new method for inexpensive Monte Carlo (IMC) simulations using high fidelity fluid mechanics code for relatively small uncertainties as introduced by manufacturing error. This approach is justified because of the ever decreasing variance of the geometric uncertainty due to the

increasing sophistication of the manufacturing processes. A review by Lamb [60] of manufacturing variability in the state-of-art point/flank milling processes for the engine blade manufacturing suggests the manufacturing tolerances are small.

1.4.2 *Gradient Information*

One of the key obstacle in geometric uncertainty propagation is the difficulty in obtaining reliable gradient information for a highly complex industrial fluid mechanics solver[7]. Gradient information is not only key to any design process but is essential to implement all gradient based optimisation algorithms. Particularly in fluid mechanics, a large body of research [68, 81] has been undertaken to address this issue. This issue becomes particularly relevant for large industrial codes which are under continuous development. It is becoming increasingly difficult to keep the linear code for gradient calculation consistent with the changes in the underlying nonlinear code.

The work presented in this dissertation proposes a systematic methodology for gradient calculation using automatic differentiation (AD) [45] software and the incorporation of various automatic validation checks for consistent and automatic generation of gradients. Furthermore, a method to calculate Hessian matrix of an objective function with respect to the design variables at a significantly lower cost is also introduced. The use of AD to automate the whole process of maintaining linear, adjoint and Hessian codes consistent with the original nonlinear code is also demonstrated.

1.4.3 *Reduced order modelling*

Lastly, it is not always straightforward to select the design parameters describing geometric uncertainty for optimisation studies. Kumar [59] has reported the use of different parametric definitions to model geometric uncertainty due to erosion and manufacturing errors for compressor blades. A recent study by Lamb [60] aimed at identifying the important geometric parameters that affect the performance of compressor blades and their dependence on the manufacturing tolerances and design methodology. This interesting study concludes that manufacturing precision and tolerances affect the best geometric indicators to be used in a design study. In other words, no fixed set of design variables would define the design space efficiently for different manufacturing geometric uncertainties. Lamb reports that the leading

[7] Refer to Peter and Dwight [77] for a good survey on the state of the art sensitivity analysis techniques.

indicator of the performance of an engine blade changes (for example from the leading edge thickness to the chord length) depending on the uncertainty introduced by the manufacturing process. Hence, the choice of design parameters defining the design space is dependent on the manufacturing tolerances. In a complex design process (with a large number of design variables), it is not efficient to use the same set of design variables irrespective of the uncertainty introduced by the manufacturing process. It is not possible to intuitively select a best set of design parameters in a complex design process. Instead, there is a need for a methodology to chose an efficient set of design parameters dependent upon the manufacturing uncertainty. Hence it is important to find an appropriate reduced order model to characterise the geometric uncertainty.

Garzon [30] introduced the key idea of using PCA for reduced order modelling to bring down the number of independent design variables defining an arbitrary geometry perturbation in the engine design process for geometric uncertainty. PCA allows for the originally large dimensional design space to be constrained in the best possible leading dimensions in some sense. This topic is discussed in more detail in chapter three of this thesis. Thakur et al. [90] have demonstrated the successful use of PCA for reduced order modelling and eliminating noise from the measurement data of IP turbine blades.

Furthermore, with the increasing availability of blade measurements, it should in-principle be possible to identify the specific sources of variability provided the manufacturing process is sophisticated enough to minimise purely random variations. For example, the seasonal floor-shop temperature variation will follow a cyclic variation over a period of one year and hence the systematic manufacturing variability introduced by this variation should be a separate identifiable source. By their very nature, all these sources are statistically independent of each other and hence, given sufficient measurement data, it should be possible to identify them. A new application of an existing technique called Independent Component Analysis (ICA) [54] is proposed in this thesis for this purpose. ICA is extensively used in signal and image processing communities. ICA has already been used for diesel engine noise source separation [62], biomedical signal processing [64], automobile manufacturing variability analysis [100], and reduced order modelling in structural damage detection [98].

To conclude, deterministic design ignores operating uncertainties from various sources and the real world performance is significantly inferior to the design objectives. On the other hand, low fidelity robust design models can over-estimate the performance

deterioration and may result in sub-optimal designs. Hence an approach to systematic high fidelity robust design analysis and optimisation is required.

1.5 CONTRIBUTIONS

The work presented in this thesis sets out to provide mathematical tools for uncertainty analysis and robust design using fluid mechanic solvers. In particular, the work presented in this thesis addresses the analysis of the manufacturing variability of the aircraft engine blades using fluid mechanic solvers. Key contributions made in this thesis are listed below.

- A mathematical formulation for inexpensive Monte Carlo simulations using the linear and adjoint solutions is presented. The use of adjoint corrected linear extrapolation instead of the nonlinear Monte Carlo is proposed to perform the inexpensive Monte Carlo simulations. The formulation is presented keeping in mind the ever decreasing manufacturing tolerances and nonlinear nature of the underlying fluid mechanic solvers. An argument for the use of at least second order approximation of the objective function during the design process is presented. The performance of the proposed methods is compared with the nonlinear Monte Carlo and the various moment methods for a model algebraic problem. The mathematical formulation is extended to the systems of PDEs for the fluid mechanic solvers and a successful implementation of the method is presented for a two dimensional airfoil code and three dimensional industrial CFD solver HYDRA.

- A methodology for automatic and consistent code development for the linear and adjoint codes for fluid mechanics solvers is presented in this thesis. An efficient implementation of the linear and adjoint code is presented using automatic differentiation. A methodology to ensure consistency between the linear and adjoint solvers is presented. An example two dimensional airfoil code is used to demonstrate the concepts.

- A mathematical formulation (based on previous work as outlined in section 5.1) for computationally efficient calculation of the Hessian of the objective function using linear and adjoint solutions is presented. The implementation details are demonstrated with the help of a two dimensional airfoil code. A good agreement is demonstrated between the Hessian matrix thus calculated

and the finite difference approximation of the Hessian matrix for the airfoil code.

- A conceptual argument for the separation of sources of manufacturing variability using a large set of measurement data using ICA techniques is also presented in the thesis. An introduction to the key assumptions, limitations and algorithms used by the ICA techniques is presented. The use of ICA is demonstrated using artificially generated geometric uncertainty for an airfoil. However, it should be noted that this is very much an conceptual argument without any experimentation with the actual engine blade measurements.

1.6 OUTLINE

The thesis is divided in five further chapters. Chapter two outlines the basic formulation behind the proposed inexpensive Monte Carlo simulations using simple test problems. This chapter also discusses the key issues in extending this formulation to fluid mechanics solvers laying the foundation for the subsequent chapters.

Chapter three discusses the reduced order modelling issues with a detailed formulation for PCA. The basic motivation behind the use of ICA analysis is presented after that. The ICA problem statement pertaining to the manufacturing variability source identification is presented followed by a brief survey of available ICA algorithms. The chapter concludes with a proof-of-concept model problem demonstration. Various tests carried out to test the performance of a few leading ICA algorithms are also presented.

Chapter four discusses the methodology for efficient, consistent and automated development and maintenance of linear/adjoint codes.

The next chapter presents the complete formulation and efficient implementation details for Hessian calculation. Both the chapters include results for a simple two dimensional Euler solver to demonstrate the validity of the proposed methods.

Chapter six presents a comparison of the inexpensive Monte Carlo simulation methodology with full nonlinear Monte Carlo simulations. A test case of a NACA0012 airfoil introducing three artificial modes of geometric perturbations has been discussed. Chapter seven extends the proposed inexpensive Monte Carlo simulation methodology to a three dimensional fluid mechanic solver HYDRA.

The last chapter summarises the key aspects of the work presented in this thesis followed by a brief discussion of the possible future areas of work.

2 INEXPENSIVE MONTE CARLO

Various Monte Carlo methods are the most popular tool for uncertainty propagation during robust design and optimisation. Theoretically, Monte Carlo methods can generate arbitrarily accurate estimates of mean and variance. However, the convergence is slow and many engineering scenarios do not justify the computational expense demanded by the nonlinear Monte Carlo methods.

The proposed Inexpensive Monte Carlo (IMC) method is a cost efficient alternative to Monte Carlo method under certain assumptions. Judicious use of the IMC method can greatly speed-up uncertainty propagation for robust design algorithms. This chapter outlines the motivation behind the proposed IMC method. Probable application areas are identified along with a survey of various alternative methods available presently. The basic formulation of the IMC method is presented. The implementation details and the effectiveness of the methods are explored using a simple algebraic model problem. The limitations of the IMC method are also demonstrated in a clear fashion using the model problem. Finally, performance of the proposed IMC methods is compared with the more traditional moment methods (MM) for a model problem.

2.1 BACKGROUND

A designer is interested in propagating the uncertainty in a design variable α to the objective function $J(\alpha,z)$ where z is the intermediate variable which is defined by an intermediate implicit function $R(\alpha,z) = 0$. In fluid mechanics problems, these are typically nonlinear fluid mechanic PDEs, and the intermediate variables are the mesh coordinates and the flow solution. There are two major classes of methods for propagating uncertainty — Monte Carlo methods and moment methods (MM).

Monte Carlo Methods

Monte Carlo methods propagate uncertainty to obtain a probability distribution function for the objective function. A *nonlinear Monte Carlo simulation* refers to Monte Carlo simulations using the exact solution of the intermediate implicit function R (in the form of fluid mechanic PDEs) at each sampling point (α_i) to calculate the objective function. No assumptions are made on the probability distributions

of the independent variables as well as the objective function. Nonlinear Monte Carlo methods can handle large uncertainties in the independent variable α as well as large nonlinearities in the implicit function R. Monte Carlo methods are fairly straightforward to implement and are extensively used in various fields of engineering. The popularity of Monte Carlo methods springs from the fact that the rate of convergence is independent of the dimension of the problem.

However, evaluation of the objective function at each sampling point requires a computationally expensive iterative solution of nonlinear PDEs (e.g. Reynolds Averaged Navier-Stokes equations). This is prohibitively expensive for production grade applications. The estimator of the mean of the objective function converges as $\mathcal{O}(\frac{1}{\sqrt{n}})$ for Monte Carlo methods with random sampling where n is the number of samples. Hence, a large number of samples are required for a satisfactory estimate. Various sampling strategies are available to improve the order of convergence. However, the computational cost of the simulations can still be prohibitive for a quick turn around during a design cycle.

Moment Methods

Moment methods use the Taylor series expansion of the objective function J with respect to the independent variable α to propagate uncertainty. Uncertainty in the objective function is represented by the mean and variance of the function. This is achieved by assuming a Normal distribution for the objective function[8]. Mean and variance of the objective function is calculated as a function of the moments of independent variable α and the derivatives of the objective function with respect to the independent variable. Mean and variance of the objective function can be calculated to any order of accuracy by reducing the truncation error of the Taylor series[9]. However, only the first order accurate moment method is mostly used for nonlinear fluid mechanic solvers as higher order derivatives of the objective function are difficult to obtain.

Putko et al. [81] and Taylor III et al. [88] demonstrated the use of moment methods to propagate uncertainty through a fluid mechanic solver for quasi one dimensional Euler equations. The authors demonstrated accurate calculation of mean and

[8] Higher order moments (skew, kurtosis etc.) of the Normal distribution are zero by definition. Assuming that a probability distribution function (PDF) is defined completely using only the mean and variance is equivalent to assuming a Normal distribution.

[9] Refer to section 2.2 for the mathematical formulation of moment methods as well as a more complete description

variance of the objective function for geometric as well as flow uncertainties for relatively small input uncertainties.

Another variation of the moment method for propagating uncertainty in the form of mean and variance of the objective function is proposed by Padulo et al. [75]. Univariate quadrature method based on sigma point methods is used to calculate mean and variance of the objective function to the accuracy of $\mathcal{O}(\sigma_\alpha^2)$ for symmetric and $\mathcal{O}(\sigma_\alpha^3)$ for the non-symmetric probability distributions of the independent variables. The method scales linearly $(2n+1)$ with the number of independent variables n. This method requires the knowledge of second, third and fourth order moments of the input variables. The higher order moments are difficult to obtain for real life geometric uncertainty parameters (which come in the form of geometry measurements) as they are highly sensitive to outliers. However, if reliable higher order moment estimates are available then the mean and variance of the objective function can be calculated to at least second order accuracy with reasonable computational cost.

The next section presents the mathematical formulation for various moment methods followed by a simple algebraic example to illustrate the performance of moment methods in different scenarios.

2.2 MOMENT METHODS

2.2.1 *Mathematical Formulation*

Given a smooth nonlinear function f,

$$J \;=\; f(\alpha),\;\; \forall \;\; \alpha, J \;\in\; \mathbb{R}.$$

If the probability density function $p(\alpha)$ for α is known, then the n^{th} order moment of J is defined as

$$\mathbb{E}^n(J) \;=\; \int_{-\infty}^{+\infty} (f(\alpha) \;-\; f(\mu_\alpha))^n \; p(\alpha) \; d\alpha,$$

where μ_α is the mean of α. Moment methods approximate $\mathbb{E}^n(J)$ using the Taylor series expansion of J. The Taylor series expansion of J about μ_α is,

$$\begin{aligned} J \;=\; f(\mu_\alpha) \;+\; f'(\mu_\alpha)(\alpha-\mu_\alpha) \;+\; \frac{1}{2}f''(\mu_\alpha)(\alpha-\mu_\alpha)^2 \\ +\; \frac{1}{6}f'''(\mu_\alpha)(\alpha-\mu_\alpha)^3 \;+\; \mathcal{O}((\alpha-\mu_\alpha)^4), \end{aligned}$$

where the primes denote derivatives with respect to α.

By considering various orders of the Taylor series expansion and taking moments, the mean and variance of output J can be approximated in terms of its derivatives evaluated at μ_α. After taking moments of various orders of approximations for J, the mean μ_J and variance σ_J^2 are approximated by the following expressions in increasing order of accuracy.

First order moment method:

$$\begin{aligned} \mu_J = &\quad f(\mu_\alpha) \quad + \mathcal{O}(\sigma_\alpha^2) \\ \sigma_J^2 = &\; f'(\mu_\alpha)^2\sigma_\alpha^2 + \mathcal{O}(\sigma_\alpha^3) \end{aligned} \tag{2.1}$$

Second order moment method:

$$\begin{aligned} \mu_J = &\quad f(\mu_\alpha) \quad + \frac{1}{2}f''(\mu_\alpha)\sigma_\alpha^2 + \mathcal{O}(\sigma_\alpha^3) \\ \sigma_J^2 = &\; f'(\mu_\alpha)^2\sigma_\alpha^2 + f'(\mu_\alpha)f''(\mu_\alpha)S(\alpha)\sigma_\alpha^3 \\ &\qquad + \frac{1}{4}f''(\mu_\alpha)^2(K(\alpha)-1)\sigma_\alpha^4 + \mathcal{O}(\sigma_\alpha^4) \end{aligned} \tag{2.2}$$

Third order moment method:

$$\begin{aligned} \mu_J = &\quad f(\mu_\alpha) \quad + \frac{1}{2}f''(\mu_\alpha)\sigma_\alpha^2 + \frac{1}{6}f'''(\mu_\alpha)S(\alpha)\sigma_\alpha^3 + \mathcal{O}(\sigma_\alpha^4) \\ \sigma_J^2 = &\; f'(\mu_\alpha)^2\sigma_\alpha^2 + f'(\mu_\alpha)f''(\mu_\alpha)S(\alpha)\sigma_\alpha^3 \\ &\qquad + \left(\frac{1}{4}f''(\mu_\alpha)^2(K(\alpha)-1) + \frac{1}{3}f'(\mu_\alpha)f'''(\mu_\alpha)K(\alpha)\right)\sigma_\alpha^4 \\ &\qquad + \frac{1}{6}f''(\mu_\alpha)f'''(\mu_\alpha)(P(\alpha)-S(\alpha))\sigma_\alpha^5 \\ &\qquad + \frac{1}{36}f'''(\mu_\alpha)^2(Q(\alpha)-S(\alpha)^2)\sigma_\alpha^6 \end{aligned} \tag{2.3}$$

where $S(\alpha)$ is the skewness, $K(\alpha)$ is the Kurtosis, and $P(\alpha)$ and $Q(\alpha)$ are higher order moments given by

$$S(\alpha) = \frac{\mathbb{E}^3(\alpha-\mu_\alpha)}{\sigma_\alpha^3}, \quad K(\alpha) = \frac{\mathbb{E}^4(\alpha-\mu_\alpha)}{\sigma_\alpha^4},$$
$$P(\alpha) = \frac{\mathbb{E}^5(\alpha-\mu_\alpha)}{\sigma_\alpha^5}, \quad Q(\alpha) = \frac{\mathbb{E}^6(\alpha-\mu_\alpha)}{\sigma_\alpha^6}.$$

The expressions presented above have been derived from the basic statistical definitions of mean and variance [28]. Typical values of these parameters for the

Normal distribution are $S = 0$, $K = 3$, $P = 0$ and $Q = 15$. Putko [81] and co-workers have used the first and second order moment methods for uncertainty propagation. As all the input variables are assumed to have Normal distributions, all the terms involving skewness $S(\alpha)$ are ignored in their work. The above expression is generic with no assumptions being made on the distribution of the input variables.

It should also be noted that in equation (2.2) the error in the variance for the second order moment is $\mathcal{O}(\sigma_\alpha^4)$ instead of $\mathcal{O}(\sigma_\alpha^5)$. This is because the third order moment method (equation (2.3)) reveals that the σ_α^4 term is incomplete in the second order moment expression (equation (2.2)). Hence, it is not clear *a priori* as to whether this fourth order term should be included in the expression. This depends on the magnitude of the coefficients of this term. In general, the order of error is not known *a priori* unless expressions for the higher order moment methods are derived to ascertain that all the coefficient terms for a particular order are captured.

2.2.2 *Simple Example*

A set of simulations are presented with simple functions $J = \sin(\alpha)$ and $J = \cos(\alpha)$ to assess the performance of moment methods with respect to the Monte Carlo simulations. Monte Carlo simulations have been carried out in a direct way by sampling α according to its distribution and evaluating J at each sample point. Monte Carlo simulations are performed for increasing values of σ_α and the predictions from the first, second and third order moment methods are compared.

Figure 2.1 shows the performance of various methods as σ_α is increased progressively from 0 to $\pi/8$. α is assumed to have a Normal distribution with $\mu_\alpha = 0$. A sample size of 50,000 is used for each Monte Carlo simulation.

It can be observed here that first order moment methods are only valid for small values of σ_α. The advantage obtained by moving to a higher order method solely depends on the values of the higher derivatives at the mean of the input. For example, moving from first to second order moment method does not improve the variance estimate for $\sin(\alpha)$.

Also note that the first order method has completely missed the behaviour of the mean and variance of the cosine function. Performance criteria like lift and efficiency are expected to behave more like the cosine function with peak values at the input mean and deterioration away from the design point. This stresses the need for at least second order methods for uncertainty propagation. It should also be noted that

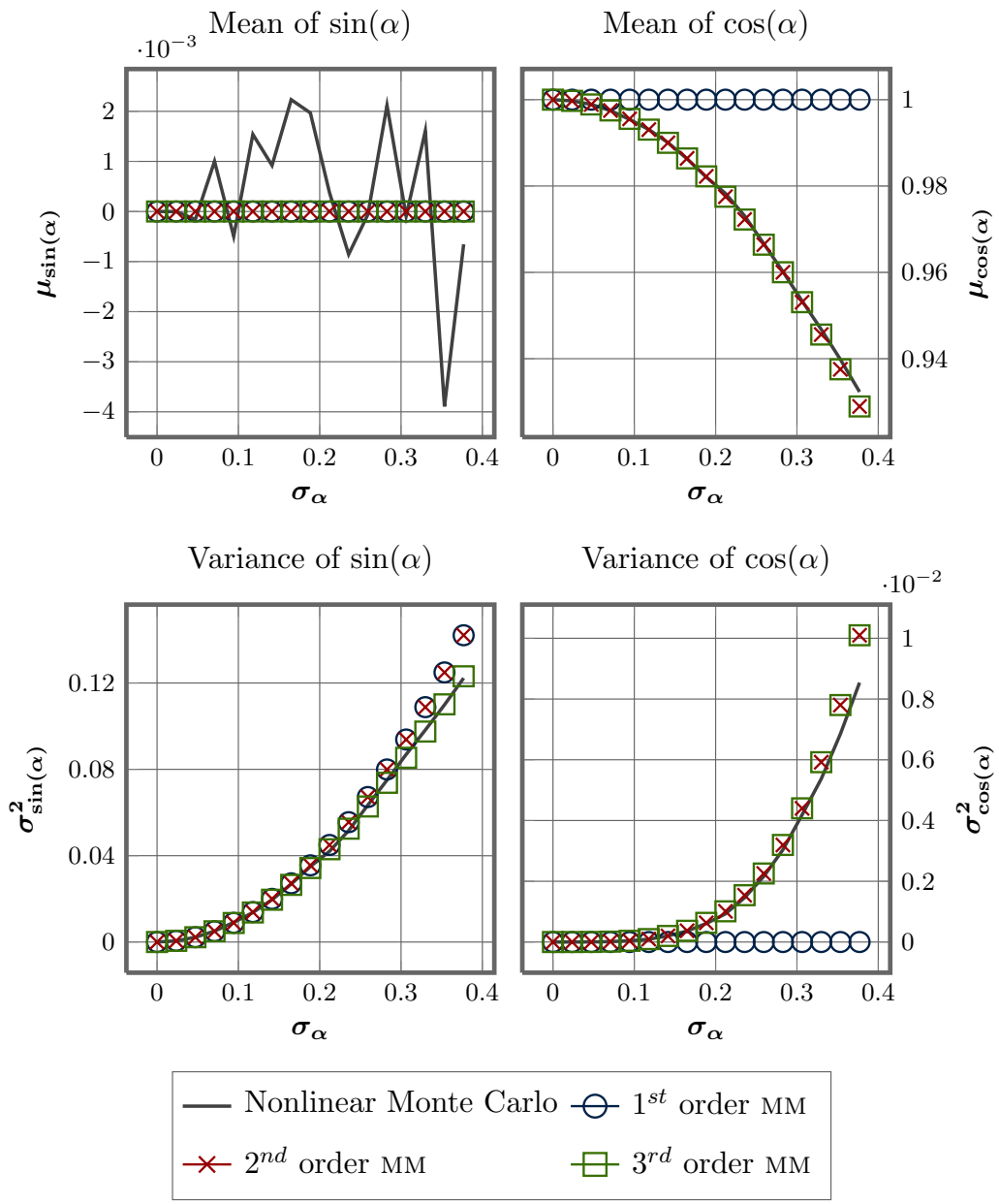

Figure 2.1 Prediction of μ_J and σ_J^2 with increasing σ_α for $\mu_\alpha = 0$

the algebra becomes increasingly complex for higher order approximations though only univariate distributions are considered here.

A comment on the actual implementation of moment methods for production grade fluid mechanic solvers is due here. Though the mathematical formulation presented above extends effortlessly to cases where the objective function J is a solution of a set of fluid mechanic PDEs, the implementation becomes complex. Firstly, it is difficult

to calculate second order derivatives for complex production grade fluid mechanic codes even using the recent advances in automatic differentiation (AD) technology[10]. Hence, for most production grade fluid mechanic codes, only the first order moment method can be implemented. As demonstrated above, the first order moment method might not be adequate for some objective functions. Secondly, the number of (first and higher order) derivative calculations is directly proportional to the number of uncertain input variables. If the number of uncertain input variables is not limited, then the computational cost of such calculations becomes prohibitively expensive. However, this issue can be circumvented using some form of parametrisation or reduced order modelling as discussed in chapter 3 of this thesis.

The next section presents a general discussion on the various uncertainty propagation methods available presently in light of the more thorough understanding of moment methods obtained in this section.

2.3 MOTIVATION

Figure 2.2 shows the classification of major uncertainty propagation methods based on

- the representation of uncertainty in the objective function,
- the amount of uncertainty in the independent variables (α), and
- the degree of nonlinearity of the function $J = f(\alpha)$ defining the objective function.

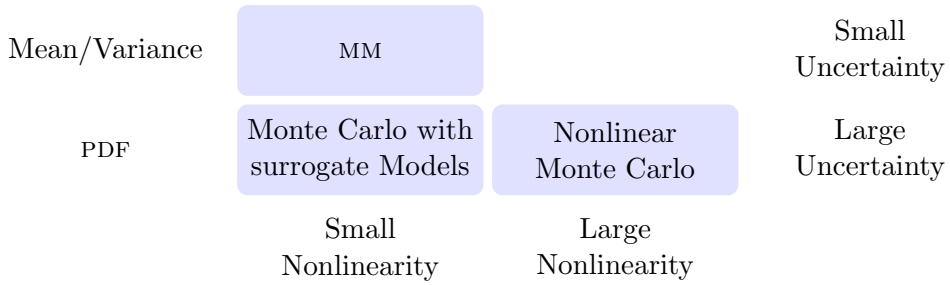

Figure 2.2 A landscape of various uncertainty propagation methods

As pointed out in the previous section, nonlinear Monte Carlo methods can handle large nonlinearities/uncertainties and obtain the complete PDF of the objective function. Hence they are the most accurate form of uncertainty propagation method.

[10] Refer to chapter 4 for a detailed exploration of issues involved in derivative calculations using AD.

However, it is wasteful to perform a nonlinear Monte Carlo simulation if the underlying nonlinearity is small enough in magnitude as to warrant the use of linear or quadratic approximation. Various global/local surrogate models of the objective function can be created [25, 59, 94, 95] to alleviate the computational cost of the Monte Carlo simulations significantly. Since the global surrogate models use fluid mechanic simulations from the entire design space, they can handle large uncertainties in the independent variables. However, the computational advantage of using surrogate models for Monte Carlo simulations quickly vanishes with the increasing nonlinearity of the objective function (J).

On the other hand, moment methods can only handle relatively small nonlinearities and represent the uncertainty in the objective function in the form of mean/variance. Since higher order sensitivity information for the fluid mechanic solvers is hard to come by, moment methods are typically restricted to only the first order derivatives and hence can only handle small nonlinearities. Since moment methods use only the gradient information at the baseline design, they cannot be used for scenarios with large uncertainties in the independent variables even if the objective function has small nonlinearities.

The distinction between the amount of uncertainty (in the independent variables) and the order of nonlinearity in the objective function should be noted here. Typically, literature on the uncertainty propagation methods treats these two quantities synonymously.

This distinction can be highlighted by considering the relationship between the angle of attack (AOA) and the lift of an airfoil. For low angles of attack, lift is almost a linear function of AOA. Hence, even a large uncertainty in the AOA in this region will have small nonlinearity. Mean/Variance calculations from the moment methods should be an accurate indicator of the uncertainty in the lift. However, at high angles of attack, the relationship between AOA and lift becomes nonlinear. Even a small uncertainty in AOA can not be handled adequately by the first order moment method.

2.3.1 *Uncertainty Representation*

Moment methods represent uncertainty in the form of mean and variance of the objective function. On the other hand, various Monte Carlo methods obtain the complete PDF of the objective function. It is often not clear as to which quantities best represent uncertainty. If the mean of the objective function is the quantity of

interest then representing uncertainty in the form of mean/variance is an adequate representation. For example, while estimating the pressure loss across a compressor stage under geometric uncertainty, mean and variance of the pressure loss represent the uncertainty adequately.

On the other hand, consider turbine life prediction under high temperatures during a design process. A designer needs to ensure that the predicted turbine life does not fall below a certain level specified by design objectives under uncertain temperature variations. Hence, though the mean of the turbine life is useful, the tail part of the distribution is of critical importance. A good estimate of the tail part of the PDF is desirable in such cases. PDF is also required for the calculation of the cumulative density function (CDF) which is one of the popular choices [65] for an objective function in robust optimisation. Moment methods are inadequate in these scenarios. There is a need for a method that produces the tail of the PDF with adequate accuracy with reasonable computational cost.

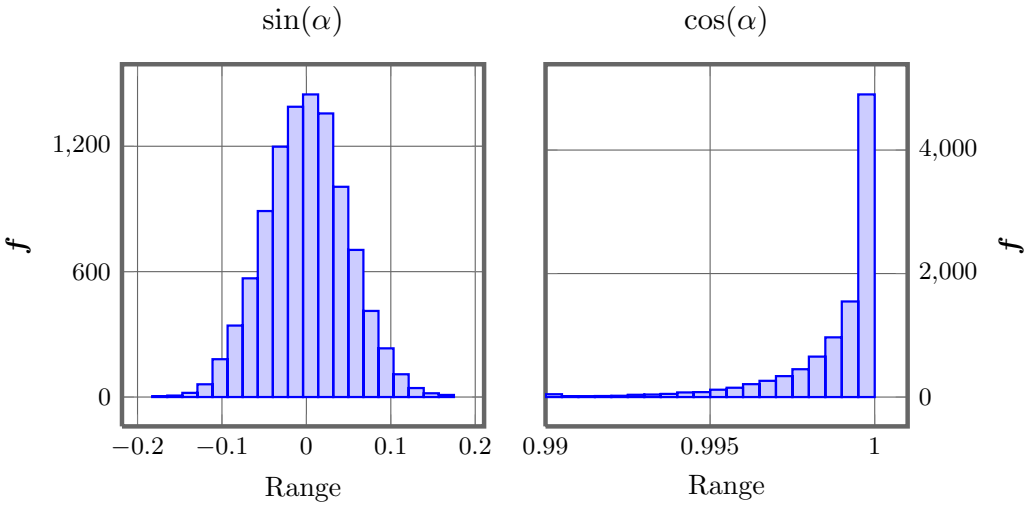

Figure 2.3 Frequency distribution for $\sin(\alpha)$ and $\cos(\alpha)$ for $\mu_\alpha = 0$, $\sigma_\alpha = \frac{\pi}{64}$

Even if the quantity of interest is only the mean of the objective function, this might not be adequate because of the highly nonlinear nature of the flow equations. This point can be illustrated rather dramatically using simple functions like $J = \sin(\alpha)$ and $J = \cos(\alpha)$. Figure 2.3 shows the output frequency distribution of these function for Normally distributed α with $\mu_\alpha = 0$ and $\sigma_\alpha = \frac{\pi}{64}$. This comparison highlights the fact that even for small uncertainties, distributions of the objective function may not have any resemblance to the input distributions for non-linear functions.

Consequently, mean and variance calculations might not be the best way to represent uncertainty.

2.3.2 *Geometric Uncertainty*

This thesis aims specifically at exploring methods for efficient uncertainty propagation methods for manufacturing geometric uncertainties. With the ever increasing precision of the manufacturing processes, it is expected that the geometric uncertainty due to manufacturing error is small and ever diminishing. Hence, there is a need for an uncertainty propagation method for

- small uncertainties, for
- high fidelity simulations, which is
- computationally cheap, and that generates
- the complete PDF of the objective function (subsection 2.3.1).

Moment methods cannot be used as the entire PDF of the objective function is required. Also, first order moment methods are likely to be inadequate as outlined in subsection 2.3.1. Nonlinear Monte Carlo method and Monte Carlo simulations with surrogate models are computationally expensive to justify their use for the small geometric uncertainties from manufacturing variation.

The next section outlines the proposed IMC simulation method which partially addresses some of the issues raised in this section.

2.4 INEXPENSIVE MONTE CARLO

As pointed out by Giles et. al. [34], the purpose of numerical computations in mathematical modelling of fluid flow is to approximate an objective function of the analytical solution to a differential equation, rather than to obtain pointwise accurate values of the solution itself. In other words, an engineer is interested in the values of lift and drag coefficients (which are integrals of the flow solution) over an airfoil instead of the flow solution at individual mesh points in the computational domain. Hence, the solution of PDEs (like Euler equations) governing fluid flow is an intermediate step. Though an engineer is interested in the accuracy of the flow solution itself, he/she is more concerned about the accuracy of the objective functions (like lift coefficient). An appreciation of this key concept is necessary for understanding the rationale behind the IMC method.

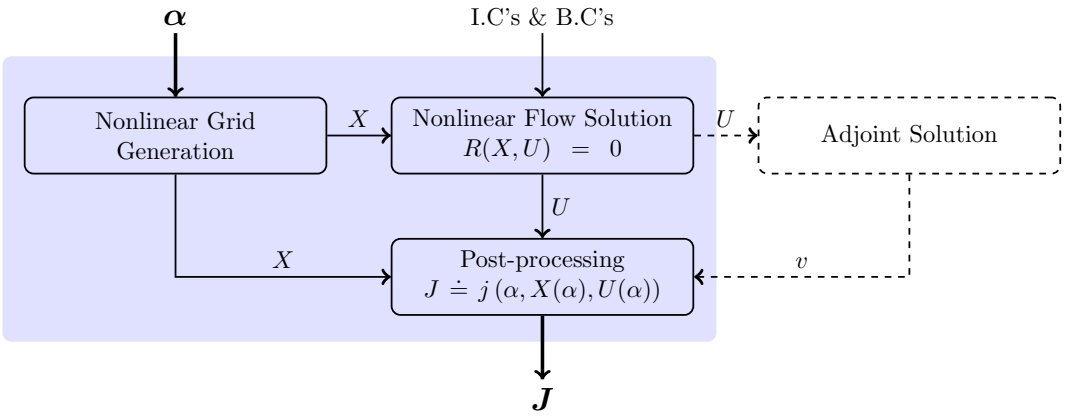

Figure 2.4 A typical fluid mechanic solution workflow

To elaborate this point further, consider a typical fluid mechanic solution workflow outlined in Figure 2.4. The design variables α are provided by a designer, an design & optimisation algorithm or a Monte Carlo sampling algorithm. In each scenario, the expected output is an objective function J. Grid generation process generates mesh X that along with the initial conditions (ICs) and boundary conditions (BCs) are input to the nonlinear flow solver. Flow solution U thus obtained is processed further to obtain an objective function J. Typically, J is an integral of some function of the flow solution over the entire or a part of the domain. The flow solution is typically the most expensive computational unit in this workflow[11].

This fact was first noticed in elasticity theory by Babuška et al. [3, 4] who introduced *a posteriori* error estimates using the dual (or adjoint) problem solution for finite element methods. The adjoint solution is used to improve the order of accuracy of the objective function. This approach was extended to the fluid mechanics problems by Giles and Pierce [40] as the adjoint correction method.

2.4.1 *Adjoint Error Correction*

Various aspects of the adjoint error correction have been studied in detail by Giles and co-workers. The theory has been introduced and extended in a series of publications. The property of the superconvergence of the objective function interest is first established in Giles et al. [34] for the finite element method. Superconvergence means that the order of accuracy of the objective function increases twice as quickly as the order of the finite element polynomial function.

[11] As explained later in this section, if the flow solution is computationally as cheap as the post-processing step, then the IMC approach is not effective.

The property of superconvergence can be shown to exist in the case of finite volume methods also. An approximate analytic function is constructed from the approximate finite volume solution. The product of the approximate residual function and the approximate adjoint solution can then be used to obtain a superconvergent objective function. This means that for a second order accurate finite volume converged solution, a fourth order accurate objective function can be obtained by applying the adjoint error correction.

This idea has been demonstrated for quasi-1D Euler equations (with and without shocks) by Giles and Pierce [39, 40]. Property of superconvergence is demonstrated numerically in this work by using the closed form solutions of the quasi-1D Euler equations. The work has been extended to two dimensional Euler equations in [43, 79].[12]

2.4.2 *Mathematical Formulation*

IMC is based on the idea of adjoint error correction as demonstrated by Giles and Pierce [40]. The formulation is first developed for simple nonlinear algebraic equations. It can then be extended to the discretisations of PDEs.

Given input independent variable $\alpha \in \mathbb{R}$, the output dependent variable $z \in \mathbb{R}$ is the solution of a set of implicit nonlinear algebraic equations

$$R(z(\alpha),\alpha) = 0. \tag{2.4}$$

The corresponding linear equation with respect to α is

$$\left(\frac{\partial R}{\partial z}\right) \frac{dz}{d\alpha} = -\left(\frac{\partial R}{\partial \alpha}\right). \tag{2.5}$$

Let $J(z(\alpha), \alpha) \in \mathbb{R}$ be the objective function which is at least twice differentiable continuous function of u and α. The discrete adjoint equation corresponding to this objective function is

$$\left(\frac{\partial R}{\partial z}\right)^T v = \left(\frac{\partial J}{\partial z}\right)^T, \tag{2.6}$$

where $v \in \mathbb{R}$ is the adjoint variable. Let z^* and v^* be some approximations to z and v respectively. The first order Taylor expansion of equation (2.4) around z^* is

[12] Refer to [42, 44] for the most complete reference on adjoint error correction.

$$R(z,\alpha) \approx R(z^*,\alpha) - \frac{\partial R}{\partial z}(z^* - z) + \mathcal{O}\left(\|z^* - z\|^2\right), \tag{2.7}$$

with all the higher order error terms being ignored and a leading error of second order. Similarly the first order Taylor expansion of the objective function around z^* is

$$J(z,\alpha) \approx J(z^*,\alpha) - \frac{\partial J}{\partial z}(z^* - z) + \mathcal{O}\left(\|z^* - z\|^2\right).$$

Using the adjoint equation (2.6),

$$J(z,\alpha) \approx J(z^*,\alpha) - v^T \frac{\partial R}{\partial z}(z^* - z) + \mathcal{O}\left(\|z - z^*\|^2\right).$$

Equation (2.7) then can be written as,

$$J(z,\alpha) \approx J(z^*,\alpha) - v^T R(z^*,\alpha) + \mathcal{O}\left(\|z^* - z\|^2\right).$$

The order of the error still remains the same. Finally if an approximate solution v^* is used instead of the exact adjoint solution v, then

$$J(z,\alpha) \approx J(z^*,\alpha) - v^{*T} R(z^*,\alpha) + \mathcal{O}\left(\|z^* - z\|^2, \|z^* - z\| \, \|v^* - v\|\right) \tag{2.8}$$

Equation (2.8) provides a computationally cheap approximation to the objective function J which forms the basis for the new IMC method.

Given the probability distribution for input random variable α with mean μ_α and standard deviation σ_α, Monte Carlo simulations are performed to calculate the resulting probability distribution for J accurately. This entails a large number of iterative solutions of equation (2.4) at the sampling points.

If the input uncertainty in α is sufficiently small then equation (2.8) can be used to approximate the objective function at the sampling points. This is termed as the inexpensive Monte Carlo (IMC) simulations. As the function $R(z^*,\alpha)$ only has to be evaluated at each sample point (as opposed to a nonlinear solution which will require expensive iterations), this method is computationally cheaper than the full nonlinear Monte Carlo simulations. The solution of the nonlinear equations, linear equations and the adjoint equations are only required for the baseline.

Suppose the baseline solutions $z(\mu_\alpha)$ and $v(\mu_\alpha)$, as well as their linear sensitivities $\frac{dz}{d\alpha}$, and $\frac{dv}{d\alpha}$ are available at $\alpha = \mu_\alpha$.

Three different IMC approaches are presented depending on the choice of approximate solutions z^* and v^*.

- IMC–1: Use baseline values,

$$z^* = z(\mu_\alpha),$$
$$v^* = v(\mu_\alpha).$$

  This approach requires only the calculation of the baseline nonlinear solution and the adjoint solution, and results in the overall leading error of second order. Based on the leading order of error, IMC–1 should perform as well as the first order moment method.

- IMC–2: Linear extrapolation to estimate z^* and baseline value for v^*

$$z^* = z(\mu_\alpha) + \frac{dz}{d\alpha}(\alpha - \mu_\alpha),$$
$$v^* = v(\mu_\alpha)$$

  This approach requires baseline nonlinear solution, adjoint solution as well as the linear solution. In general, if $\alpha \in \mathbb{R}^n$ then n linear sensitivity calculations along with the baseline solution is required. The IMC–2 has the leading error of third order. Based on the leading order of error, IMC–3 should perform at least as good as the second order moment method.

- IMC–3: Linear extrapolations for both z^* and v^*.

$$z^* = z(\mu_\alpha) + \frac{dz}{d\alpha}(\alpha - \mu_\alpha),$$
$$v^* = v(\mu_\alpha) + \frac{dv}{d\alpha}(\alpha - \mu_\alpha)$$

  This is the most accurate approach with the leading error of the fourth order. However, this requires calculation of linear sensitivities for both z and v which are difficult to obtain for fluid mechanic solvers. If $\alpha \in \mathbb{R}^n$ then along with the baseline solution, adjoint solution and n linear solutions, this approach will require n linear sensitivity calculations for the adjoint approach.

2.4.3 *Computational Cost*

Method	Computational Cost
Nonlinear Monte Carlo	NC
1^{st} order MM	$(1+\gamma_a)C$
2^{nd} order MM	$(1+\gamma_a+n\gamma_l)C$
IMC – 1	$(1+\gamma_a)C+Nc$
IMC – 2	$(1+\gamma_a+n\gamma_l)C+Nc$
IMC – 3	$(1+\gamma_a+n\gamma_l+n\gamma_{al})C+Nc$

Table 2.1 Comparison of the computational cost of various Monte Carlo methods assuming that there are n independent design variables, N sampling points for the Monte Carlo methods, C is the computational cost of the nonlinear iterative solution of the equation of state at baseline conditions, and c is the cost of one evaluation of the objective function.

A comparison of the computational cost of the various methods of uncertainty propagation is presented in this section for the fluid mechanics application. Let the computational cost of one nonlinear iterative solution of the state equation (2.4) at the baseline be C. Furthermore, let $\gamma_l C$ and $\gamma_a C$ be the cost of one linear (equation (2.5)) and adjoint (equation (2.6)) solution respectively. The factors of multiplication γ_l and γ_a have the typical values ranging between 2 and 4 depending on the implementation of the linear and adjoint solutions for fluid mechanic solvers [17, 37, 74]. Let c represent the computational cost of a single evaluation of the objective function. Since fluid mechanic solvers require large number of iterations for a solution, $C \gg c$. Table 2.1 shows the computational cost for various approaches of uncertainty propagation.

If N sample points are used for the Monte Carlo simulations then the computational cost quickly becomes prohibitively expensive for the nonlinear Monte Carlo simulations. Moment methods are considerably cheaper than the nonlinear Monte Carlo simulations. First order moment method only requires a single iterative nonlinear solution and the solution of the adjoint equation (computational cost $\gamma_a C$) for the calculation of the Jacobian matrix. The computational cost of the second order moment method also includes the cost of Hessian calculation $n\gamma_l C$ where n is the dimension of the independent variable α. Though moment methods are considerably cheaper than the nonlinear Monte Carlo simulations, they only calculate the mean and variance of the objective function.

IMC methods calculate the complete PDF of the objective function (like nonlinear Monte Carlo simulations) at roughly the same computational cost as the moment methods. IMC–1 method requires the iterative solution of the baseline nonlinear equation along with the adjoint solution like the first order moment method. In addition, N cheap evaluations of the objective function at the sampling points are required with the resulting computational cost of Nc. Since $C \gg c$ for the fluid mechanic applications, the overall computational cost of the IMC simulations is significantly lower than the nonlinear Monte Carlo simulations. In addition, IMC–2 approach requires the calculation of n linear solutions. IMC–3 approach requires calculation of n linear sensitivities of the adjoint solution $n\gamma_{al}C$ as well[13].

2.4.4 *Model Problem*

The use of these approaches is demonstrated using a simple model problem:

$$\begin{aligned} R(z(\alpha),\alpha) &= z + z^3 - \alpha, \\ J(z(\alpha),\alpha) &= z^2. \end{aligned} \tag{2.9}$$

The input variable α is treated as a random variable with Normal distribution. The estimates for mean and standard deviation for J using the moment methods and IMC methods are compared with the full nonlinear Monte Carlo simulations over a range of σ_α. Equation (2.9) is solved using simple Newton iterations to machine accuracy. Analytic expressions (as given in equation (2.10)) for the calculation of v, $\frac{dz}{d\alpha}$ and $\frac{dv}{d\alpha}$ are evaluated at $\alpha = \mu_\alpha$. A sample size of 50,000 is used for all Monte Carlo simulations.

$$\begin{aligned} \frac{dz}{d\alpha} &= \frac{1}{1+3z^2} \\ v &= \frac{2z}{1+3z^2} \\ \frac{dv}{d\alpha} &= \frac{2-6z^2}{(2+3z^2)^4} \end{aligned} \tag{2.10}$$

Figure 2.5 shows the comparison between various methods for the model problem (equation (2.9)) at $\mu_\alpha = 5$. The top left hand side figure plots the mean of the objective function (μ_J) as standard deviation of the input variable α is increased from 0 to 1. Similarly, the figure on the top right hand side plots the variance of

[13] Please refer to section 6.1.1 for further details on the implementation of linearisation of adjoint equations for fluid mechanic solvers using AD.

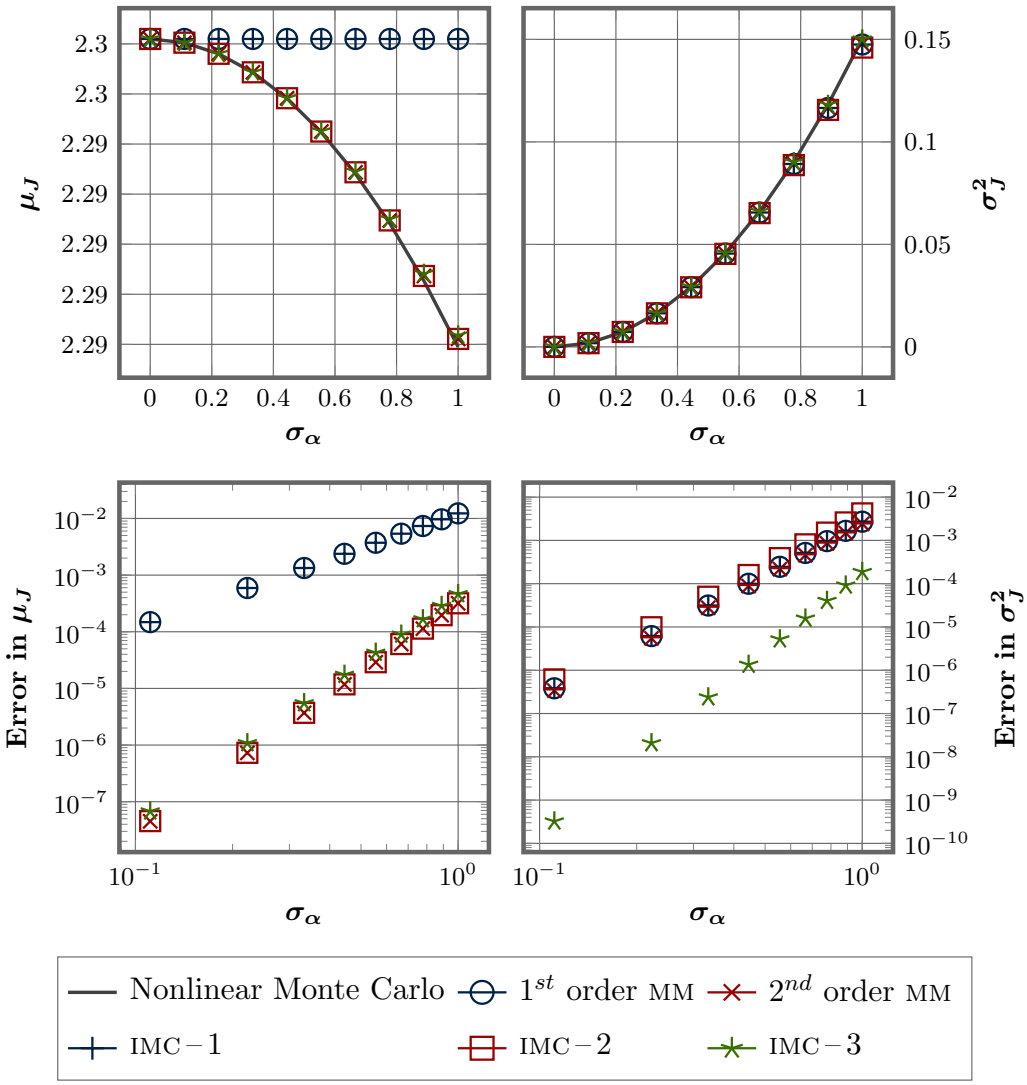

Figure 2.5 Performance of different methods for the model problem for $\mu_\alpha = 5$

the objective function σ_J^2 with increasing standard deviation of the input variable α. Both the plots compare the performance of the nonlinear Monte Carlo method with various moment methods and the proposed IMC methods.

The bottom left figure plots the difference between the mean of the objective function using various moment methods and the IMC methods and, the nonlinear Monte Carlo method with the change in σ_α.

As expected IMC – 1 and the first order moment method, and IMC – 2 and the second order moment method show similar behaviour. The trend for the variance of the objective function is captured adequately by all the methods. IMC – 1 method misses the trend for the variation of mean completely because the distribution of the input

variable α is symmetric. This suggests that IMC – 1 methods are not adequate for application to real life problems. Hence, higher order IMC – 2 and IMC – 3 methods are desirable.

Figure 2.6 shows a similar comparison for $\mu_\alpha = 0$. In this case the error increases rapidly with σ_α. Higher order effects become significant sooner in this case as opposed to for $\mu_\alpha = 0$ (note the difference in the range of the horizontal axis). It can be clearly observed that the IMC – 3 method follows the actual behaviour longer than the rest of the methods.

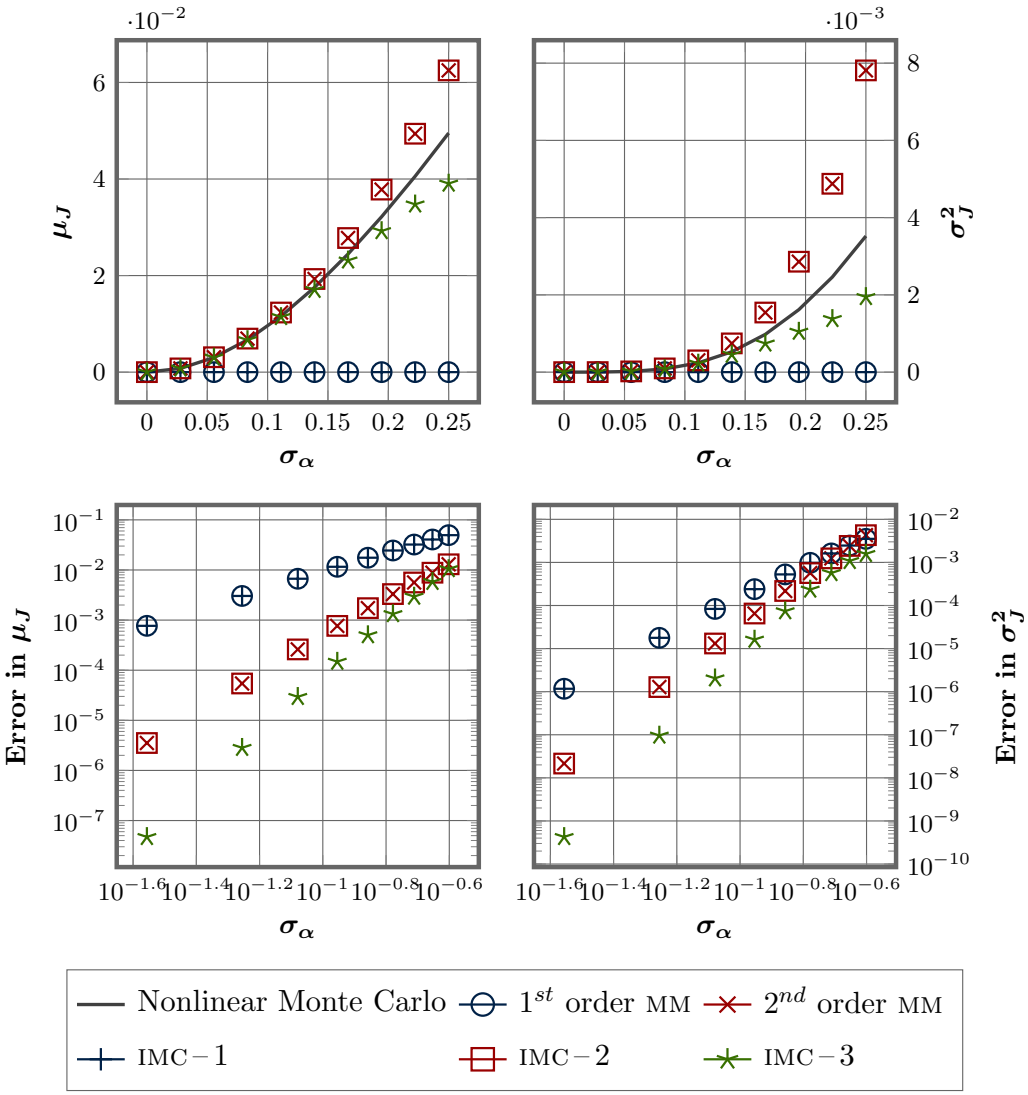

Figure 2.6 Performance of different methods for the model problem for $\mu_\alpha = 0$

This case illustrates the problem in defining the validity range for these methods *a priori*. In other words, it is not possible in real life to define *a priori* what are "small

enough" perturbations to justify these kinds of second order approximations. The explanation for the bad performance of IMC methods for $\mu_\alpha = 0$ can be found by looking at Figure 2.7. It shows the variation of $\frac{dv}{d\alpha}$ with α. Because of a rapid change in adjoint variables away from $\alpha = 0$, the error increases rapidly with increasing standard deviation. But at $\alpha = 5$ as the adjoints are not changing significantly, use of the linear extrapolation or no extrapolation of the adjoint variables is justified for a relatively long range. This kind of analysis is not always possible for real life applications. Linearisation of the adjoint code is possible using automatic differentiation as illustrated in [17, 66] for the fluid mechanic solvers. Hence, the gradient of the adjoint solution can be obtained which gives an idea regarding the behaviour of the adjoint variables near the baseline solution. If the gradient has a high value, then it is expected that the IMC methods will have a limited range of validity.

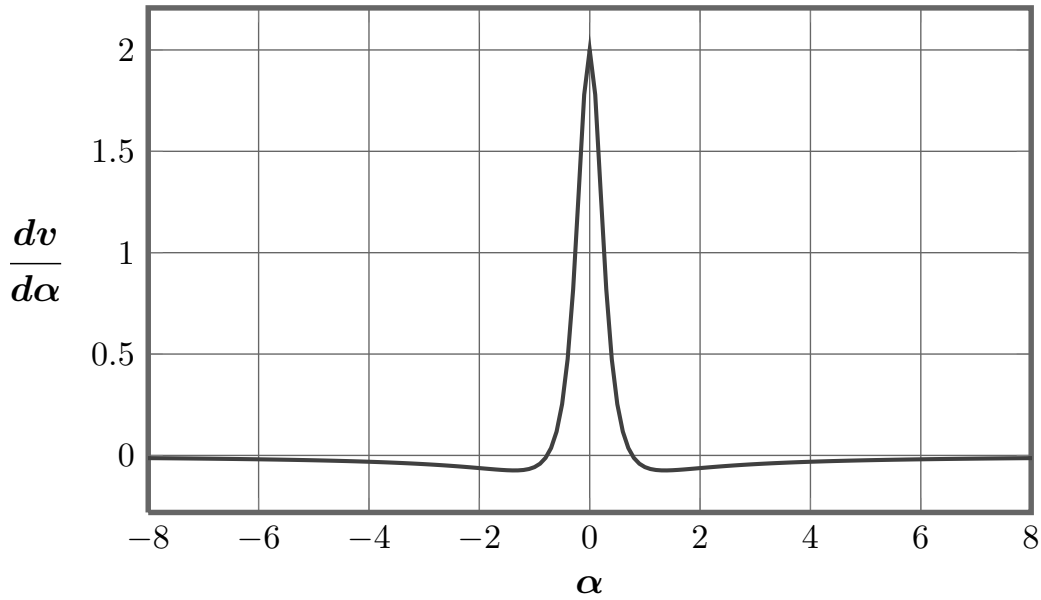

Figure 2.7 Variation of derivative of adjoints with respect to α

Though the IMC approach has been introduced here for simple algebraic equations, it can be easily extended to the discretised PDE equations. The governing flow equations in the discretised form can be represented in the form of equation (2.4). Typically the objective function for the fluid mechanic applications would be quantities like lift, drag and pressure loss. The rest of the analysis in section 2.4 also applies to the systems of equations arising from the discretisation of PDEs. The extra issues that need to be addressed are related to the implementation details. Conceptually, this analysis extends seamlessly to discretised PDEs.

2.5 CONCLUSION

The motivation and the basic idea behind the proposed IMC simulations is presented in this chapter. Performance of the IMC methods with respect to the moment methods of various orders has been compared for a simple model problem. It has been shown that IMC methods can compute the entire probability distribution of the objective function at roughly the same computational cost as the moment methods. The next chapter introduces the concept of reduced order modelling for implementing IMC methods for fluid mechanic solvers. The implementation of the IMC methods for a two dimensional Euler solver is discussed and demonstrated in chapter 6.

3 REDUCED ORDER MODELLING

The basic formulation for IMC simulations has been introduced in the previous chapter using a model problem. An additional complexity is introduced in using IMC for fluid mechanic codes in the form of large number of input random variables defining the surface geometry. These could range from thousands to millions depending on the complexity of the geometry, the accuracy of the surface definition and the physical model employed in the fluid mechanic code.

Complex geometries encountered in turbomachinery typically require a large number of grid points to define the surface boundaries. Surface definition requires even more grid points to incorporate localised geometric deviations due to manufacturing variability.

Even with simple and ideal geometries, modelling of complex flow phenomenon like heat transfer and turbulence require clustering of grid points for greater resolution of flow properties in the boundary layer region of the flow. As the fluid mechanic simulations try to capture real life flows to greater accuracy, this trend is expected to continue.

The cumulative effect of these contributing factors has resulted in a steady increase in the number of parameters defining the geometry in a typical fluid mechanic simulation.

If the number of input random variables (geometry variables) is not restricted, then the computational cost would increase linearly due to the large number of linear sensitivity calculations as shown in the previous chapter. To reduce this extra computational cost, a reduced order model of the geometric uncertainty is necessary. Once the number of independent geometry variables is reduced to a manageable level through a reduced order model, IMC simulations can be performed with reasonable computational cost.

3.1 BACKGROUND

Reduced order modelling is a highly complex topic in itself and has found application in various fields of engineering. A complete survey of reduced order modelling

methods is out of scope here. Only a few representative examples of reduced order modelling in the area of fluid mechanics are mentioned.

Typically reduced order modelling methods can be categorised as parametric and non-parametric.

Parametric methods require *a priori* knowledge about the system to be reduced. For example, A. Kumar [59] used piece-wise cubic functions and Hicks-Henne functions for parametric modelling of the localised effect of erosion on compressor blades using three parameters. His thesis also demonstrates the use of Hicks-Henne functions along with splines for modelling manufacturing variations. Once a model is selected based on the knowledge of the geometric uncertainty, the exact coefficients can be calculated using the measurement data. Parametric methods are typically more efficient (in terms of the number of independent input variables required to capture the variation) because of the use of deep knowledge of the problem under consideration. As the geometries and simulations under consideration become more complex, this knowledge and intuition is hard to come by.

The non-parametric method of reduced order modelling does not require any prior knowledge of the system under consideration. One of the most important and by far the most commonly used method is Principal Component Analysis (PCA) [57, 80]. PCA is also known by the names of Proper Orthogonal Diagonalisation (POD), the Hotelling transform and the discrete Karhunen-Loéve transform (KLT).

PCA essentially tries to estimate a meaningful basis (in some sense) for a set of data. This is achieved by identifying mutually uncorrelated basis vectors in the diminishing order of their importance. PCA only uses second order statistical information (variance and covariance). All the higher order information is discarded. This makes this analysis computationally efficient.

Garzon [30] has reported the use of PCA to reduce the geometry model of high-pressure compressor integrally-bladed rotor (IBR) blades to lower order. His work shows that a successful reduced order model can be derived using PCA for real fan blade measurements without any prior knowledge of the sources or the characteristics of the variation in the data. Furthermore the leading modes resulting from the analysis were found not to correspond in general to customary geometric design and tolerancing parameters of known aerodynamic importance. This suggests that real life geometric variability due to manufacturing may not be best captured by parametric reduced order modelling.

Several studies have found that a small number of PCA modes can capture manufacturing variability in blades accurately [8, 10, 31, 86]. Garzon [30] has reported the use of only 5 leading eigenmodes to capture 99% of the scatter energy (equation (3.4)) which is a measure of the geometric variability. Similarly, Bui-Thanh et al. [10] have reported the use of only two leading modes of PCA model to capture the variation in subsonic rotor blade measurements. Recently, Thakur et al. [90] successfully used PCA to eliminate noise from a set of 1052 turbine blade measurements.

The next section outlines the mathematical formulation of PCA for the geometric measurement data[14].

3.2 PCA

3.2.1 *Mathematical Formulation*

PCA is inspired by the idea of reducing redundancy and noise in a given set of measurement data. The basic assumption behind PCA is that the measurement process is accurate enough as to have low noise levels. In other words, directions with greatest variance represent the features of interest. This can be achieved by diagonalising the covariance matrix of the measurement data. The directions that diagonalise the covariance matrix are known as the principal components. A reduced order model is derived by selecting a few leading principal components for the data.

Let the nominal surface geometry be defined by p coordinate points $x_i^0 \in \mathbb{R}^m,\ i = 1, ..., p$ where m is the dimension of the data. Suppose n sets of measurements $\{\hat{x}_{i,j} \in \mathbb{R}^m \mid i = 1,...,p\};\ j = 1,...,n$ are available. Here the index j uniquely identifies a specific instance of measurement while index i identifies the measurement corresponding to a unique nominal coordinate point.

[14] PCA based Reduced order modelling has also been used to approximate the behaviour of nonlinear flow equations for a specific problem. This obviates the need for expensive iterative solutions of nonlinear PDEs thereby reducing the computational requirements dramatically.
Ravindran [82] reported the use of proper orthogonal diagonalisation (POD) to reduce a fluid mechanical system (using Navier-Stokes equations) for the optimisation and control of unsteady flows. Discretised Navier-Stokes equations were reduced to a few leading components using POD. Another popular approach for reduced order modelling of the fluid mechanics equations is the use of various surrogate models [25, 59].

The error from the nominal geometry is given as

$$x'_{i,j} = \hat{x}_{i,j} - x_i^0. \tag{3.1}$$

This error includes both the manufacturing variability as well as the measurement error. It will be assumed henceforth that the measurement error is negligible as compared to the manufacturing variability and hence $x'_{i,j}$ effectively represents the manufacturing variability.

Subtracting from the error vectors their mean gives a centred set of m-dimensional vectors,

$$\tilde{x}_{i,j} = x'_{i,j} - \bar{x}_i \mid i = 1,...,p; j = 1,...,n. \tag{3.2}$$

where, the mean of these error vectors is calculated as

$$\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x'_{i,j}, i = 1,...,p.$$

This set of centred error vectors can be written in a matrix form $\tilde{X}$ with the j^{th} column $\tilde{X}_j = [\tilde{x}_{1,j}^T,...,\tilde{x}_{p,j}^T]^T$. Here $\tilde{X}_j$ contains the centred set of error vector for the j^{th} set of measurement. Generally, $\tilde{X}$ would be a dense $mp \times n$ matrix[15]. The covariance matrix of $\tilde{X}$ is given by

$$C_x = \frac{1}{n-1}\tilde{X}\tilde{X}^T$$

As stated earlier, PCA identifies statistically meaningful new bases for the data. A PCA algorithm identifies directions that are mutually uncorrelated to each other and identify the directions of greatest variance of the data in some sense. This is equivalent to finding a transformation matrix to diagonalise the covariance matrix C_x. One method of achieving this is to solve the eigenvalue problem for the covariance matrix C_x. This is one way of solving for the principal components. The more popular and a more general approach to identifying the principal components is to perform a Singular Value Decomposition (SVD) of matrix $\tilde{X}$.

The SVD of $\tilde{X}$ is given by

$$\tilde{X} = M\Sigma N^T. \tag{3.3}$$

[15] It is should be noted that $\tilde{x}_{i,j}$ are column vectors while these are row vectors in Garzon's [30] treatment. Also, the $\tilde{X}$ matrix here is the transpose of $\tilde{X}$ matrix as defined by Garzon.

Here, M is a $mp \times mp$ orthonormal matrix while N is a $n \times n$ orthonormal matrix. Σ is $mp \times n$ diagonal matrix with diagonal entries ordered in decreasing value.

A reduced order model based on PCA is obtained by selecting the first few principal components to approximate the original variability in matrix $\tilde{X}$. A good measure of this variability is the total scatter energy E given by

$$\begin{aligned} E &= tr(\tilde{X}\,\tilde{X}^T) \\ &= tr(M\,\Sigma\,N^T\,N\,\Sigma\,M^T) \\ &= tr(M\,\Sigma\Sigma\,M^T) \\ &= \|M\,\Sigma\|_F^2\,, \end{aligned} \tag{3.4}$$

where $\|.\|_F^2$ is the Frobenius norm. Since the Frobenius norm is invariant under unitary multiplication,

$$E = \|M\,\Sigma\|_F^2 = \|\Sigma\|_F^2 = tr(\Sigma\,\Sigma^T) = \sum_{i=1}^{mp} \lambda_i^2,$$

where λ_i is the i^{th} diagonal entry of Σ. By selecting the first few modes from the matrix M and looking at the corresponding eigenvalues, it is possible to predict how much of the scatter of the original data is captured. This is the basic idea behind the PCA based reduced order modelling.

One of the key assumptions behind the use of PCA is that the directions (of the principal components) with the greatest variance are the important ones. This assumption is only valid for data with low noise levels. With the availability of accurate measurement techniques (like laser scanning) for the manufactured blades, this assumption is justified.

Another reason behind the suitability of PCA for reduced order modelling is the ease of implementation and computational efficiency of the PCA algorithm. Singular value decomposition is a popular and convenient way of implementing PCA. Efficient implementations of SVD are available in MATLAB and the NAG® library[16]. This is primarily the result of the assumption of linearity imposed while searching for the principal components. This assumption is imposed by requiring that the principal components are orthogonal to each other.[17]

[16] Function `svd()` and `svds()` in MATLAB and `f02wec`, `f02wgc` and `f02xec` in the NAG library among others.

The next section discusses the derivation of a reduced order model based on the principal components obtained in equation (3.3).

3.2.2 *PCA based model*

Defining $A = \frac{1}{\sqrt{n}} M\Sigma$ and $W^T = N^T\sqrt{n}$, equation (3.3) can be written as

$$\tilde{X} = A\,W^T. \tag{3.5}$$

A is a $mp \times n$ matrix while W is a $n \times n$ square matrix. The columns of A represent the principal components in diminishing strength. The centred error matrix $\tilde{X}$ can be retrieved exactly from equation (3.5).

As N is an orthonormal matrix, if w_i is the i$^{\text{th}}$ column of W then

$$\frac{1}{n}\sum_{i=1}^{n} w_{ij}^2 = 1, \text{and} \tag{3.6}$$

$$\frac{1}{n}\sum_{i=1}^{n} w_{ij}\,w_{ik} = 0, \forall\ j \neq k. \tag{3.7}$$

This suggests that w_{ij} can be approximated as an independent normal random variable $\mathcal{N}(0,1)$. Using equation (3.5), a reduced order model for the geometric uncertainty can be written as

$$x_r = x^0 + \bar{x} + \sum_{k=1}^{n_r} a_k\,\mathcal{N}(0,1) \tag{3.8}$$

where $n_r \leq mp$ is the number of leading order modes selected and a_k is the k^{th} column of A. $\mathcal{N}(0,1)$ is a random variable with zero mean and unit variance. The number of leading order modes can be selected depending on the total scatter energy to be captured. As n_r increases the total scatter of x_r approaches that of $\hat{x}$.

Garzon [30] has used a similar approach but with a different normalisation. In his presentation the eigenvectors are of unit norm and the amplitude of the perturbation

[17] PCA fails for cases where some of the underlying assumptions behind the analysis limit its search. For example, consider a set of points $x = \cos(\theta)$ and $y = \sin(\theta)\ \forall\ \ \theta \in [0,2\pi]$. Clearly, this can be reduced to a single component. As θ is a nonlinear function of the present basis x and y, PCA will not be able to reduce this dataset. To address these scenarios, nonlinear PCA algorithms are being developed that are known as kernel PCA. Kernel PCA employs *a priori* knowledge (if available) about the data to introduce nonlinearities in the form of kernel are applied to the data to transform the data to a more appropriate naive basis.

is given by the diagonal entries of Σ. In this derivation, the eigenvectors show a varying degree of perturbation in decreasing order and therefore signifying their importance.

If the complete model is used, mp linear calculations along with an adjoint calculation will be required for the IMC−2 approach. But in the reduced order model, only n_r linear calculations would be required affording huge computational savings without significant loss of accuracy.

Next section introduces the concept of Independent Component Analysis (ICA) as an extension of the already discussed PCA. Possible use of ICA techniques to identify different sources of manufacturing variability is also explored along with a discussion of its limitations.

3.3 EXTENSION OF PCA

3.3.1 *Motivation*

At present some measurement data is available for the engine blades, though it is not clear if this data is used by the design community except for calibrating the tolerances. High quality measurement methods like laser scanning are increasingly being used regularly at the manufacturing sites. The main drivers behind this are the falling costs and ease-of-use of these devices. The entire life cycle of an engine blade is expected to be monitored in the future with routine high fidelity measurements.

A large set of blade measurements makes the data more amenable to statistical analysis. If additional statistical information about the sources of variability is extracted, it should help engineers to identify possible sources of variability in the manufacturing process and investigate ways of reducing it. If it is not economically viable to eliminate or reduce some of the sources of this variability, then their knowledge might help in off-setting their effect on the engine performance at the assembly stage.

Secondly, with the ever increasing precision of manufacturing facilities, it is expected that only a finite number of independent sources would produce variations in the blade geometry. Two sources of variability are independent of each other if they originate from mutually independent physical sources. Any blade measurement will show us the cumulative effect of the blade perturbations introduced by all

the independent sources. Two examples of independent sources of variations are discussed below.

Flank milling is used extensively in the engine blade manufacturing process. Once a new machine tool has been introduced during the milling process, it wears down gradually in a continuous fashion. It is replaced with a new tool after it wears down outside a certain tolerance range, and the same cycle is repeated. This is one of the systematic sources of variability in the engine blade manufacturing process. If this is one of the few sources of manufacturing variability, then this will have a distinct non-normal distribution. In fact, if the machine tool wears down linearly then the variation mode introduced by it in the blade geometry will have a uniform distribution. In the general case, leading mode of variation in the blade geometry introduced by the machine tool wear will have a distinct non-normal distribution.

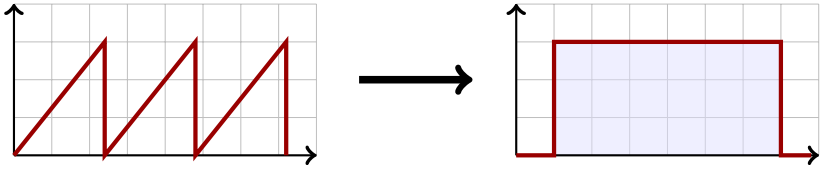

Figure 3.1 Expected distribution for the error introduced in the blade geometry during the flank milling process if the machine tool wears down linearly

Another example of a source of variability is the periodic shop floor temperature variation over a cycle of one year. The manufacturing variations introduced by it will have a distinct leading mode with a distinct distribution. Intuitively it can be seen that these two sources of variability are independent of each other because of their separate physical causation. Hence, the distributions of their modes of perturbation are expected to be statistically independent.

Assuming that such measurement data is available then independent component analysis (ICA) [22] can be used to separate out these sources of variations. Consider n independent sources of signals that are mixed together in a black-box. The mixing is linear (which means a multiplication by a mixing matrix D). The mixed signals are then observed by m separate measurement devices. ICA addresses the issue of separating these signal sources using only the measurements[18]. There are a number

[18] This is known as the cocktail problem in the ICA community. Given n speakers in a cocktail party and m observations from different locations (of sufficient duration), estimate the original voice signals from each of the speaker given the m observations.

of assumptions made before an ICA algorithm can separate out these sources.

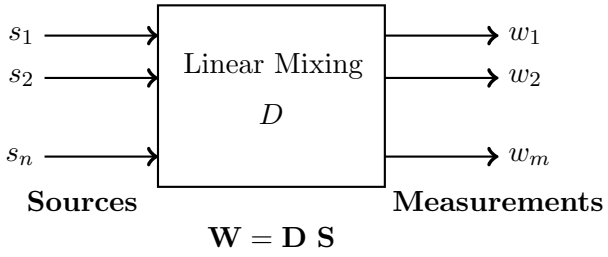

Figure 3.2 Conceptual ICA model

In the similar fashion, the signal sources can be considered as n independent sources of manufacturing variability. The mixing box can be replaced by the manufacturing process and the measurement data comes from the geometric measurements of the blades. In principle, ICA techniques allow for the separation of independent sources of manufacturing variations.

3.3.2 *ICA Background*

ICA is a well established tool used for signal and image processing. It is also known as blind signal separation (BSS) in the signal processing community. The same ICA algorithms can be used for feature extraction while processing images. The fundamental idea behind both applications is to separate out statistically independent components from a mixture. Aapo Hyvärinen et al. [54] have written an excellent introductory book explaining basic ICA concepts, various ICA algorithms and applications. A good recent survey of various ICA algorithms can be found by Tichavský et al. [91].

ICA has found applications in many areas. A few applications of interest are discussed below which are similar to the present manufacturing variability source identification.

Li et al. [62] demonstrated the use of ICA algorithms in separating out the various acoustic signals of a typical diesel engine. These separated signals when studied individually can be used to precisely identify the important sources of noise in the diesel engines.

In another application, Zhang et al. [100] have proposed the use of a Bayesian BSS algorithm for identifying the sources of variability at the car assembly stage. Spatial patterns arising from the routine measurements on the manufactured door of a car were separated into a few leading components using ICA. Precise identification of the

source of variation has been reported using the knowledge of the assembly process and the separated sources.

3.4 ICA

3.4.1 *Mathematical Formulation*

A basic formulation for ICA as can be applied to the manufacturing variability source separation is presented here. For a more complete formulation with rigorous mathematical treatment refer to Hyvärinen [54].

Suppose there are n_r independent geometric variability sources and n observations (or geometry measurements of turbine blades). Let S be a $n_r \times n$ matrix whose each column $s_i \in \mathbb{R}^{n_r}$ represents an independent source of manufacturing variability. Also, W is an $n_r \times n$ matrix consisting of one instance of measurement of a blade geometry per column. Hence, W represents n set of observations in the reduced order model derived in the previous section. Assuming that each geometry measurement is a linear combination of all the sources of error, ICA model can be written as

$$W = D\ S, \tag{3.9}$$

where D is an $n_r \times n_r$ unknown mixing matrix. Each column s_i represents a separate source while every column w_i represents an instance of measurement. The distributions for each measurement w_i are known in the form of discrete approximations of them with zero mean and unit variance.

Given W, ICA method estimates the mixing matrix D such that s_i are as mutually independent as possible.

The following aspects of the ICA problem statement should be noted here.

- The number of leading modes of geometric variability is assumed to be equal to the number of sources to simplify the presentation of the ICA concept without any loss of generality. However, for the application to the manufacturing variability source identification, one can always select leading modes from the PCA analysis equal to the number of sources to be identified.

- This is an ill posed problem and hence exact solution is not possible. As both D and S are unknown, for any scalar β,

$$W = (\beta D)(S/\beta)$$

is also a solution. This means that the variances of the independent components cannot be determined. In other words, the relative magnitude (energies) of various sources of manufacturing variability cannot be determined. This ambiguity is removed by requiring that each s_i has unit variance. However, this still leaves the sign and renumbering ambiguity.

It should also be noted here that the ICA is not performed on the original matrix of centred measurements of geometric variability $\tilde{X}$, but is performed on the reduced order model W derived using PCA. ICA can also be performed on the measurement matrix directly but this approach has some advantages.

Firstly, since W is an orthogonal matrix, the number of unknown variables D_{ij} are reduced from $n_r \times n_r$ to $\frac{n_r(n_r-1)}{2}$. This greatly reduces the complexity of the problem. Also, since W is the reduced order model of the original measurement dataset, the dimensionality of the problem is greatly reduced. Since the reduced order model captures most of the scatter energy of the original data, there is no loss of generality. This also has the added benefit of reducing the measurement noise [54].

The original measurements can always be obtained by reversing the linear transformation applied by PCA (equation (3.8)).

To get a better feel for the problem let's take a look at a model two dimensional problem. Consider two independent sources s_1 and s_2 each with a uniform distribution with zero mean and unit variance. Figure 3.3(a) shows the scatter plots for the original sources using a sample size of 500. They are then mixed using the mixing matrix[19]

$$D^x = \begin{pmatrix} 1 & 2 \\ 0.5 & 4 \end{pmatrix}, \tag{3.10}$$

to produce measurements x_1 and x_2 also shown in Figure 3.3(b). The next two scatter plots show transformation of the measurements x_1 and x_2 to w_1 and w_2 using PCA followed by the application of ICA to obtain estimates s_1^{est} and s_2^{est} for the original independent sources. The estimated[20] mixing matrix in this case is

[19] Matrix D^x is different from the mixing matrix D as defined equation (3.9). D^x mixes the original matrix containing a centred set of measurements $\tilde{X} = D^x S$.

[20] The matrix is estimated using MATLAB implementation of RADICAL ICA algorithm [61] available at their website `http://people.cs.umass.edu/~elm/ICA`.

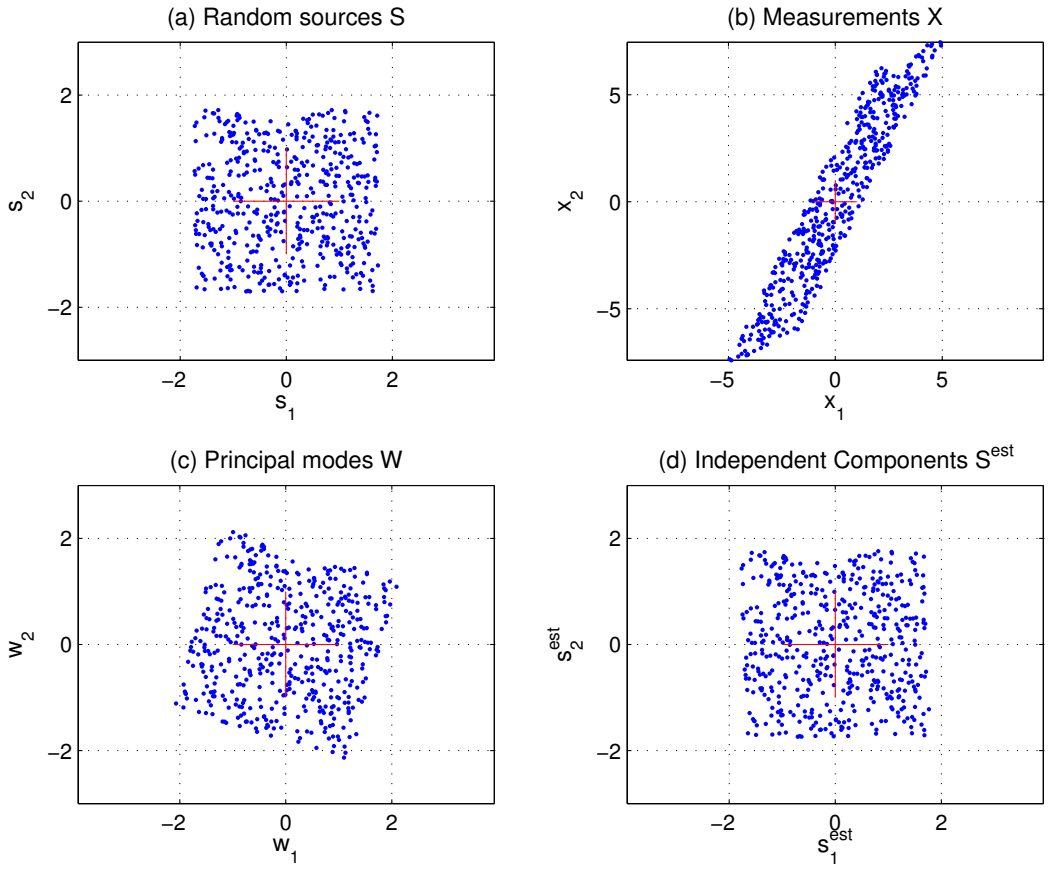

Figure 3.3 Application of ICA for two sources with uniform distribution

$$D^x_{est} = \begin{pmatrix} 0.56 & 1.16 \\ 0.24 & 2.31 \end{pmatrix} = 0.56 \begin{pmatrix} 1 & 2.07 \\ 0.43 & 4.13 \end{pmatrix}.$$

PCA applied on the observations X gives uncorrelated leading modes W. ICA algorithms can then be applied to identify the closest approximation to the mixing matrix providing the distributions of the original independent sources. However, it should be noted here that the exact mixing matrix cannot be identified. If the sources are considered as time series signals then the exact source signals cannot be recovered. But often the probability distributions of the source signals can be estimated to a good accuracy subject to the size of measurement data, underlying source distribution and the ICA algorithm. This is the basic rationale behind using ICA techniques for manufacturing variability source recognition.

The rest of the chapter introduces the assumptions behind ICA analysis, followed by the limitations of the ICA analysis. The basic formulation for independence and its various approximations are elaborated in the next section followed by the

introduction of a few prominent ICA algorithms. The chapter ends with a set of tests performed to assess the performance of a few prominent ICA algorithms.

3.4.2 *Assumptions*

The fundamental assumption underlying all ICA algorithms is the independence of the sources. In statistical terms,

$$\mathbb{E}[f(s_i)g(s_j)] = \mathbb{E}[f(s_i)]\ \mathbb{E}[g(s_j)] \tag{3.11}$$

where operator $\mathbb{E}$ denotes the expectation operator for any analytic functions f and g. This is a very strong condition. Many measures of independence can be derived from this basic definition. If the underlying sources are not independent then ICA algorithms cannot be used for source separation. Hence, only mutually independent sources of manufacturing variability can be separated using ICA algorithm. As only mutually independent sources of manufacturing variability are of interest, ICA can be used to separate them. However, actual measurement data is needed to test this assumption. If two or more sources of manufacturing variability are correlated and are not independent, then the ICA algorithms will not be able to separate them.

Secondly, it is assumed that the source signals combine with each other in a linear fashion to produce each observation. This is a key underlying assumption in almost all ICA algorithms. With the ever increasing precision of the manufacturing process, it is expected that the variations due to various sources will be small in magnitude. Hence, the combination of variability due to various sources is expected to be modelled effectively as a linear combination.

Garzon [30] has reported the use of actual blade measurement data for integrally-bladed rotor blades fabricated via flank-milling process. The average measured geometry of the blade at the mid-span section is compared with the nominal geometry (refer to Garzon [30] (Table 2.1)) in terms of six parameters, namely, maximum thickness, maximum camber, leading edge radius, trailing edge angle, chord and the area of the airfoil section. It has been reported that except for the leading edge radius, the average deviation in the measured blades from the nominal geometry is less than one percent. Average deviation for the leading edge radius is around 18%. If the geometric variation due to manufacturing error is of this order then the linearity assumption in the ICA model will be satisfied by the actual measurement data with high probability. Although it is not known at present if such tight tolerances are a norm or an exception for the typical engine blade manufacturing processes. Blade

measurements on a large scale for a variety of different manufacturing processes are required to verify if this assumption is satisfied.

Thirdly, only one of the independent components can have a Gaussian distribution. Consider a simple example with two sources s_1 and s_2 with Gaussian distribution with zero mean and unit variance. If they are mixed using the mixing matrix as given by equation (3.10) then Figure 3.4 shows the results of ICA. It can be seen intuitively that once the signals are uncorrelated, then there is no extra information available to separate the independent sources. It can be seen that the joint probability density of W is completely symmetric. For example, if the mixing matrix D is orthonormal then the joint distribution of X is

$$p(x_1,x_2) = \frac{1}{2\pi} \exp\left(-\frac{s_1^2 + s_2^2}{2}\right).$$

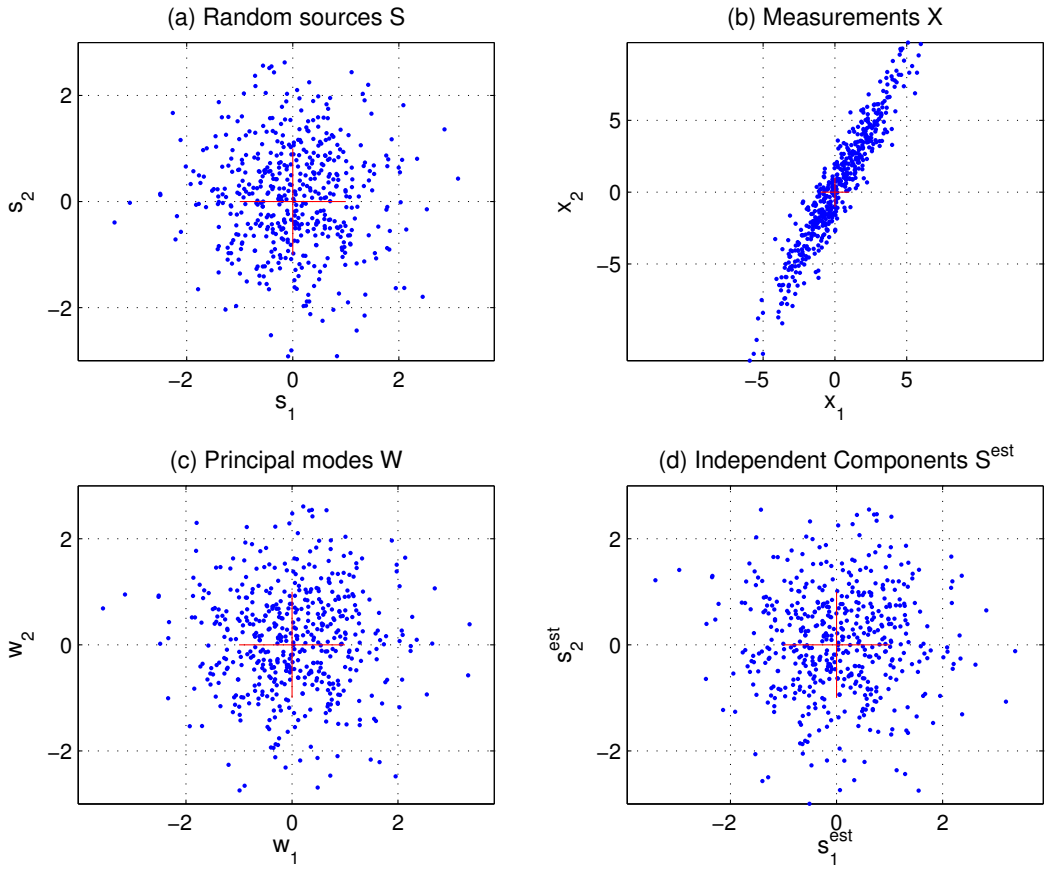

Figure 3.4 Application of ICA for two sources with Gaussian distribution

Therefore, in the case of two Gaussian sources, the ICA model is effectively same as the PCA model. No further information can be extracted. More rigorous mathematical

treatment of this subject can be found in [6, 21]. For manufacturing variability source identification, it is expected that the sources of interest will have non-Gaussian and unique distributions. Actual high fidelity data in statistically significant amounts is required to ascertain this condition.

Lastly, the mixing matrix D has to be full rank. This means that each of the observations has to be an independent observation with no two observations having identical values. This is expected to be trivially true for the manufacturing measurement data.

The next section introduces the basic algorithm for ICA.

3.4.3 *Maximisation of Non-Gaussianity*

A more convenient form of the ICA model given by equation (3.9) is presented here. It should be noted that each row of the matrix S represents an independent source of manufacturing variability. Hence, a $n_r \times 1$ column vector s^r can be constructed using S where each element s_i^r of the s^r vector is a centred random variable with unit variance representing a source of manufacturing variability.

Similarly, a $n_r \times 1$ column vector w^r can be constructed from W with each element w_i^r representing a random variable corresponding to a leading mode of measurement as given by the PCA model. A discrete approximation to the PDF of the each measurement random variable can be obtained by the n instances of measurements in the row vectors of the W matrix.

The ICA model can then be expressed in statistical form as

$$w^r = D\, s^r. \tag{3.12}$$

Here, each element of the w^r vector (equation (3.6) and equation (3.7)) and s^r (by definition) vectors is a random variable with zero mean and unit variance.

A consequence of the Central Limit Theorem is that a combination of two or more independent random variables (under certain conditions) is more Gaussian than any of the individual distributions [54]. This is the basic principle behind all the ICA algorithms. Consider y_i defined as

$$y_i \;=\; s_i^{est} \;=\; e^T w^r \;=\; e^T D s^r \tag{3.13}$$

where e is a $n_r \times 1$ transformation column vector. Now as s_i^r are independent random variables, therefore y_i is in general more Gaussian than any of the sources s_i^r. So it becomes least Gaussian only when it is in fact equal to one of the sources. If we have a quantitative measure of Gaussianity then we can optimise to find e in such a way as to find a least Gaussian y_i. In practice, we will use W to calculate y_i and optimise over e. The simplest measure of Gaussianity is kurtosis as presented in the next section.

3·4·4 *Kurtosis*

The simplest measure of Gaussianity is kurtosis defined for a random variable y_i as

$$\text{kurt}(y_i) \;=\; \mathbb{E}[y_i^4] \;-\; 3\mathbb{E}[y_i^2]^2.$$

Kurtosis is zero for the Gaussian distribution while it is positive for super-Gaussian distributions (e.g. Laplacian distribution) and negative for sub-Gaussian distributions (e.g. Uniform distribution).

So the optimisation problem can be posed as:

Find y_i that

$$\begin{aligned} \text{minimises} \qquad & -|\text{kurt}(y_i))| \\ \text{with respect to} \qquad & e \\ \text{such that} \qquad & \|e\|^2 = 1 \;\;\&\;\; y_i = e^T w^r. \end{aligned} \tag{3.14}$$

Using

$$\text{kurt}(\alpha_1\, x + \alpha_2\, y) = \alpha_1^4\, \text{kurt}(x) + \alpha_2^4\, \text{kurt}(y),$$

we get,

$$\text{kurt}(y_i) = \sum_j e_j^4\, \text{kurt}(w_j^r).$$

The constraint on the transformation vector e follows as a consequence of the equation (3.12). In two dimensional space (with two unknown sources of manufacturing variability), it can be seen that this optimisation problem is equivalent to finding the minimum kurtosis on a unit circle. The various local minima of this optimisation problem yield the various sources. Theoretically, if we have n_r sources then this problem has $2n_r$ local minima [21].

One issue with this approach is the fact that the value of kurtosis is very sensitive to outliers. Hence, it requires a large number of sample points to obtain a smooth behaviour of kurtosis. Kurtosis is presented here as a simplest measure of Gaussianity.

3.4.5 *ICA Algorithms*

Most ICA algorithms derive their objective function using the Kullback-Leibler (KL) divergence. The KL divergence between the probability density functions $f(s)$ and $g(s)$ is defined as [21]

$$KL(f|g) = \int^{s} f(s) \log \frac{f(s)}{g(s)} ds$$

KL is a measure of distance between the two probability distributions in some sense. It should be noted here that KL divergence is not symmetric. Using this as a measure, a contrast function $J(Y)$ is defined as

$$J(y) = p(y_1,...,y_{n_r}) \log \frac{p(y_1, ... ,y_{n_r})}{p(y_1)\, p(y_2) \ldots p(y_{n_r})}\, d\mu \tag{3.15}$$

where, y is a $n_r \times 1$ vector of random variables, $p(.)$ is the PDF of a random variable, and $d\mu = dy_1\, dy_2 \ldots dy_{n_r}$. It can be seen here that the contrast function is zero if and only if y_i are mutually independent random variables. Hence, this contrast function can be regarded as a direct measure of the mutual independence. Many ICA algorithms start the formulation by defining the contrast function $J(y)$ as outlined above and then make various assumptions to simplify the formulation.

Using equation (3.13), the contrast function can be written as [21]

$$J(y) = \sum_{i=1}^{n_r} H(y_i) - H(w_1^r, ... ,w_{n_r}^r) - \log(|E|)$$

where $y = E\, w^r$ and,

$$H(f) = \int_{s} f(s)\, \log f(s)\, ds$$

is the differential entropy of a random variable s with probability density $f(s)$ [54]. Here, E is a $n_r \times n_r$ matrix made of e^T as the row vectors.

Since E is an orthonormal matrix (equation (3.14)), the minimisation of the contrast function $J(y)$ reduces to finding

$$J_1(y) = \sum_{i=1}^{n_r} H(y_i). \tag{3.16}$$

Robust, Accurate, Direct ICA algorithm (RADICAL) uses an entropy estimator based on order statistics [61] for calculating the differential entropy. The estimator has the desirable property of being computationally efficient ($\mathcal{O}(n_r \log(n_r))$. Since the estimator of the marginal entropy used in the RADICAL algorithm is computationally inexpensive to calculate, an exhaustive search is used for optimisation. Since E is orthonormal, this involves Jacobi rotations for finding the minima. The exhaustive search methodology used for optimisation makes RADICAL computationally expensive for higher dimensions (number of sources). However, for the application of ICA for identifying the sources of manufacturing variability, this is not a limitation.

The FastICA algorithm uses a slightly modified version of the objective function (equation (3.16)). It uses negentropy [54] given by

$$J_2(y) = \sum_{i=1}^{n_r} H(y_{gauss_i}) - \sum_{i=1}^{n_r} H(y_i) \tag{3.17}$$

where, y_{gauss_i} is a Gaussian random variable with variance equal to that of y_i. Since the Gaussian random variable has the maximum entropy than a random variable of equal variance for all distributions, this objective function is always positive.

Algorithm	Objective Function	Optimisation Method
FastICA (1997) [55]	Approximations of Negentropy	Fixed point Iterations
JADE(1999) [16]	Entropy estimates using higher order cumulants	Jacobi Optimisation
KernelICA (2002) [5]	Kernel approximation of Marginal Entropy	Steepest descent method
RADICAL(2003) [61]	Estimate of marginal entropy based on order statistics	Exhaustive Search

Table 3.1 Prominent ICA algorithms with their objective function and optimisation approach

Hyvärinen developed various approximations to negentropy [52] based on the maximum-entropy principle. The objective function used in FastICA has the form

$$J_2(y) \approx \sum_{i=1}^{p} k_i[\mathbb{E}(G_i(y)) - \mathbb{E}(G_i(\nu))]^2$$

where, k_i are some positive constants, and ν is a Gaussian variable of zero mean and unit variance. The functions G_i are some nonquadratic functions. By taking $G(y) = y^4$, this reduces to the objective function based on kurtosis. An example of the nonquadratic function used by FastICA is $G(y) = \frac{1}{a_1} \log \cosh (a_1\, y)$. FastICA uses an efficient fixed-point iteration technique for optimisation [53].

The JADE algorithm [16] assumes near Gaussian distributions for the independent sources and derives an approximation to the differential entropy (equation (3.16)). Using higher order cumulants (variance and kurtosis) of near Gaussian distributions, an efficient estimate of the objective function can be obtained. This is then optimised using joint diagonalisation of a set of cumulant matrices using efficient algebraic techniques like the Jacobi algorithm [15]. The use of Jacobi algorithm makes JADE the most computationally efficient ICA algorithm. However, the assumption of near Gaussian distribution for the error sources makes this algorithm less robust.

Finally, KernelICA use contrast functions based on canonical correlations in a Reproducing Kernel Hilbert Space [5]. A detailed mathematical presentation of this algorithm is beyond the scope of this thesis. KernelICA uses the $\mathcal{F}$- correlation between independent random variables to define a measure of independence. Steepest descent method is used for optimisation. KernelICA algorithm is slower than the rest of the algorithms since the objective function is computationally expensive.

Table 3.1 summarises various ICA algorithms along with the objective function and the optimisation algorithm employed by them.

3.5 RESULTS

3.5.1 *Performance Plots*

Two tests are carried out to compare the performance of prominent ICA algorithms as listed in Table 3.1. MATLAB codes for various algorithms are used as provided by the authors. All the codes are written in MATLAB and are available from the websites of the authors.

A MATLAB code is written to generate a mixing matrix D using random Jacobi rotations. Sources were selected from a set of nineteen different distributions. The

set of distributions contain unimodal and multimodal distributions with a mixture of Sub-Gaussian and Super-Gaussian distributions. Each selected distribution is sampled to generate the source matrix S. The number of samples is equal to the desired number of measurements for the test. The matrix of measurements W is then generated using the mixing matrix D and the source matrix S. These measurements are then input to the various ICA algorithms to obtain an estimate of the mixing matrix D_{est}.

The performance at each point is defined by Amari distance (or Amari error) [1] which is a well known measure of performance in the ICA community. Given a mixing matrix D and its estimate D_{est}, Amari distance is defined as

$$\mathcal{A}(D,\, D_{est}) = \frac{1}{2n}\sum_{i=1}^{n}\left(\frac{\sum_{j=1}^{n} \parallel c_{i,j} \parallel}{\max_j \parallel c_{i,j} \parallel} - 1\right) + \frac{1}{2n}\sum_{j=1}^{n}\left(\frac{\sum_{i=1}^{n} \parallel c_{i,j} \parallel}{\max_i \parallel c_{i,j} \parallel} - 1\right) \tag{3.18}$$

where, $C = D^{-1}\, D_{est}$.

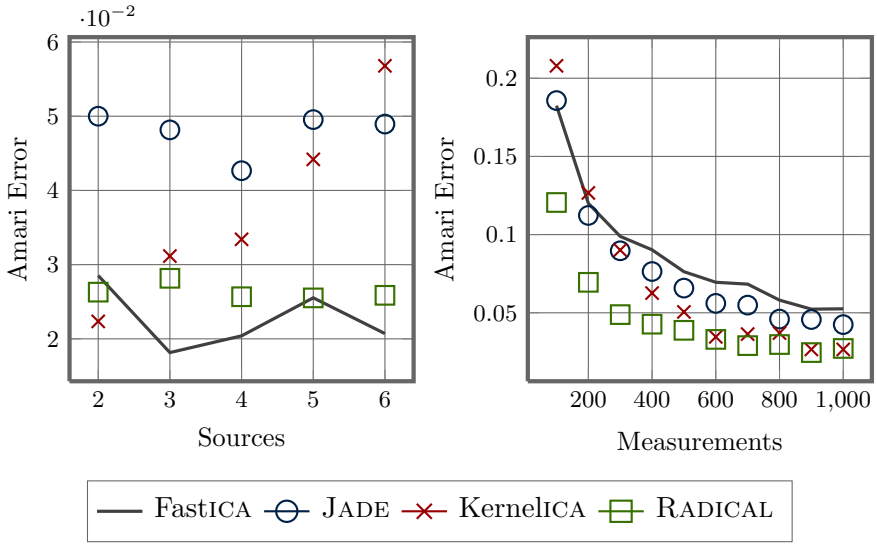

Figure 3.5 Performance of various ICA algorithms for increasing number of sources (left) and number of measurements (right)

Figure 3.5 (right) plots the performance of four ICA algorithms as the number of measurements are increased progressively from 100 to 1000. The number of sources is fixed at two for all the simulations. A set of 1000 simulations is performed at each point to get a statistically significant mean of the Amari error. The sources are selected randomly from the set of nineteen distributions for each simulation.

Figure 3.5 (left) shows the performance of various algorithms as the number of sources are increased progressively. The number of measurements available for each

experiment is kept constant at 1000. A set of 1000 simulations at each points is performed to get a statistically significant mean of the Amari error. This Amari error for four different ICA algorithms is plotted as the number of sources are increased progressively from 2 to 6. The plot does not show any significant difference in the performance of various ICA algorithms.

It should be noted here that the error trends in these plots are only average performances of various algorithms. It was observed that the performance of each algorithm depends strongly on the probability distribution of the source terms. The actual preference for an algorithm for manufacturing variability source identification can be decided only after several experiments with actual blade measurements. RADICAL has been used for the numerical experiments presented in the next section for the artificial geometric variability introduced for an airfoil geometry.

3.5.2 *Airfoil Uncertainty*

This section presents an example application of PCA and ICA to NACA0012 airfoil. Three artificial modes of perturbation are introduced to the baseline airfoil. The three modes of perturbation are thickness perturbation (equation (B.3)), angle of attack (AOA) perturbation (rotation of the airfoil about the trailing edge) and leading edge shape perturbation (equation (B.4)). Refer to Appendix B for the exact expressions used for introducing these perturbations. three modes of perturbations for $\pm 10\sigma$. A set of 1000 artificial geometric perturbations were generated for the baseline NACA0012 airfoil by randomly introducing the three modes of perturbations. A uniform random distribution is used for the thickness perturbation, a normal random distribution is used for AOA perturbation, and a beta distribution is used for the leading edge shape perturbation. The resultant set of 1000 measurements of the airfoil geometry are used for the PCA and ICA as outlined in this chapter.

The PCA was performed using the svd function in MATLAB. The three leading eigenvectors thus obtained were used for reduced order modelling and ICA was performed using RADICAL ICA algorithm.

Figure 3.6 shows the probability distributions for the original modes of perturbations (left) and the estimated probability distributions of the three modes of perturbations. As can be seen from the histograms, the probability distributions of the original modes of perturbation are obtained by ICA to a good degree of accuracy. However, there remains an ambiguity regarding the order of the modes of perturbation.

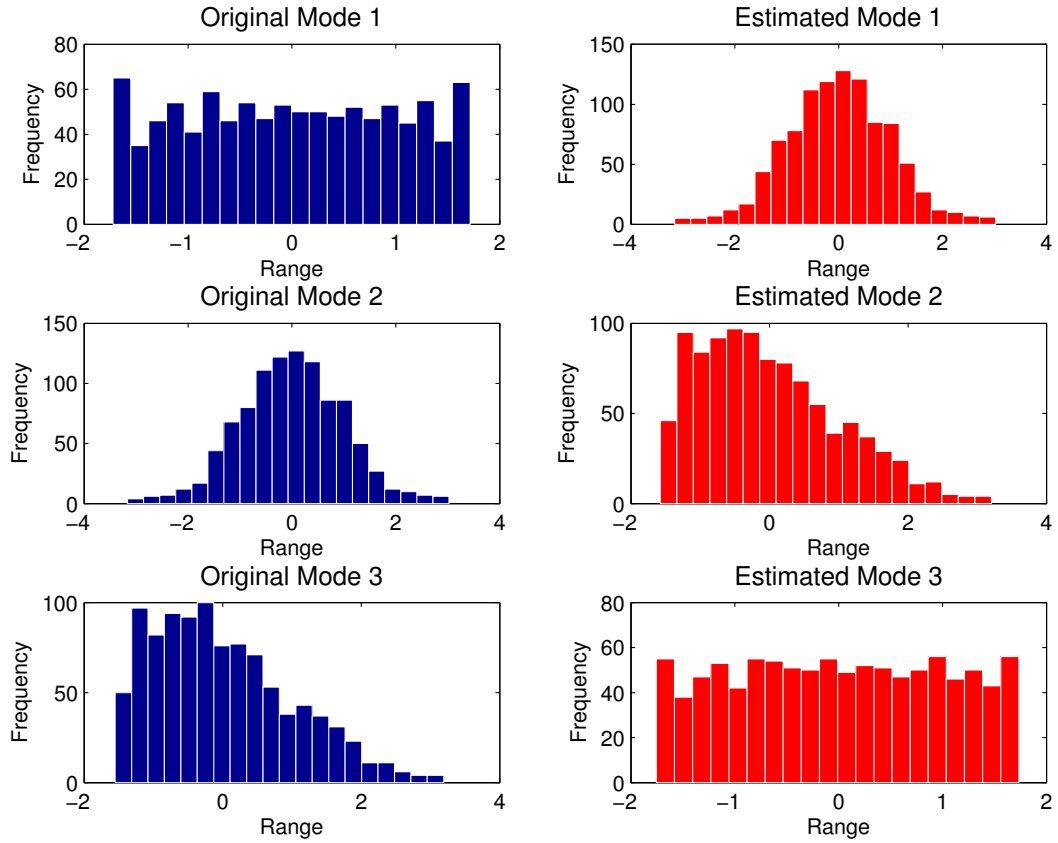

Figure 3.6 Application of ICA to the artificially generated set of geometry perturbations for NACA0012. The original probability distributions of the three artificial modes of geometric perturbations (left) and the estimates of these probability distributions obtained using RADICAL ICA algorithm (right) are shown.

3.6 CONCLUSION

This chapter presented a reduced order model based on PCA to reduce the number of independent variables for the IMC simulations. In the PCA, scatter energy associated with the direction of each eigenvector is proportional to the corresponding eigenvalue. The eigenvector corresponding to the highest eigenvalue represents the direction along which the scatter is maximized. The eigenvector corresponding to the next eigenvalue maximises the scatter in the direction perpendicular to the first eigenvector and so on. Hence, if the scatter is not completely uncorrelated, PCA based reduced order model captures most of the scatter using a few leading eigenmodes. This is the basis of the proposed reduced order model based on the PCA.

In principal, the ICA can be used to identify the sources of manufacturing variability under certain assumptions. A brief mathematical formulation of the ICA has been

presented in this chapter. An example use of the ICA for source identification was presented for artificially generated geometric uncertainty for an airfoil. Although a successful application of ICA algorithms is demonstrated for separating the independent sources of artificial uncertainty introduced to the airfoil geometry, it does not prove the suitability of the ICA algorithms for the application in mind. The artificial geometric uncertainty in the airfoil was generated such that all the underlying assumptions of the ICA formulation (section 3.4.2) are satisfied. It is not known at present if all the assumptions underlying ICA algorithms will be satisfied for the actual measurement data for engine blades. The assumption of the independence of the sources of uncertainty is fundamental to all the ICA algorithms. If the underlying sources of uncertainty are not independent then the ICA algorithm will not be able to separate the different sources of error. If two or more sources of manufacturing error are correlated with each other, the ICA algorithm will identify the joint probability distribution of these correlated sources of error as an independent source of manufacturing error. It is not clear at this point whether ICA algorithm is useful in such scenarios.

Another key assumption behind most of the ICA algorithms is that the independent sources combine linearly with each other. If the manufacturing process is not accurate and the variations in the geometry are not small enough, then the assumption of linear combination will not hold true. ICA algorithms cannot be applied for the stated purpose in this scenario. Hence, it is important to ensure that the geometric variation due to manufacturing error is small enough to justify the linearity assumption.

4 CODE DEVELOPMENT

4.1 INTRODUCTION

The basic formulation for IMC simulations is introduced in chapter 2. Chapter 3 introduced the mathematical formulation of PCA for reduced order modelling required to apply IMC to fluid mechanic solvers. Another and perhaps the most important component for successful IMC simulations is the availability of accurate linear and adjoint solutions.

Linear and adjoint solutions thus obtained can then also be used for sensitivity analysis, gradient optimisation algorithms and surrogate modelling in the design process. All the above design steps are typically limited by and tailored to the availability of accurate gradient information. Another limiting factor is the human effort required to maintain and update the linear/adjoint codes with the corresponding changes in the nonlinear solver. This might entail updating the linear/adjoint codes multiple times every year for a large fluid mechanic code (for example, the HYDRA suite of fluid mechanic solvers developed by Rolls-Royce in cooperation with multiple university technology centres across the globe). It is desired that minimum (and possibly nil) human effort is required to keep the linear/adjoint code versions in sync with the original nonlinear code with feedback being generated about any discrepancies at every update.

A two dimensional Euler flow solver is used in the next three chapters as an example code to illustrate various ideas. For further implementation details refer to Appendix A. This chapter discusses the development of linear and adjoint codes using a two dimensional Euler airfoil code as an example. The objective of this activity is to understand the difficulties and challenges involved in developing consistent linear/adjoint codes using AD. Systematic validation and testing methods for the development of the linear and adjoint codes are also explored. The next section reviews various methods of sensitivity propagation for fluid mechanic solvers.

4.2 SENSITIVITY PROPAGATION

A note on the general fluid mechanic flow solution model adopted in this thesis

is useful here. In any design process, the designer is interested in the gradients of an objective function $J \in \mathbb{R}^m$ with respect to the design variables $\alpha \in \mathbb{R}^n$. Figure 2.4 shows a typical flow calculation in the design process. Mesh generation usually involves third party CAD packages for surface definitions followed by iterative algorithms to generate the volume mesh X. Conceptually, the mesh generator can be treated as an implicit nonlinear function solution requiring iterative solution. Since the source code of the mesh generator is usually not available, it is treated as a black-box throughout this thesis.

The flow solver computes the flow solution U by solving the nonlinear governing flow equations which can be written in discrete form as

$$R(U, X) \;=\; 0.$$

The flow solver can be treated as an implicit nonlinear function requiring iterative solutions.

Post-processing usually involves an explicit surface or volume integral to be calculated based on the flow solution U. The objective function is therefore

$$J(\alpha) \;\doteq\; J(U(\alpha), X(\alpha), \alpha).$$

The perturbation in the design variable α is to be propagated along the entire chain to the objective function $J(\alpha)$. It is assumed that the source code for the flow solver and post-processing units is available for modification. Since the flow solution is by the far most computationally expensive step, the cost of one entire flow solution cycle is considered to be equal to one iterative solution of the nonlinear flow solver.

The most straightforward method for sensitivity propagation is central finite difference approximation. If $f(\alpha)$ is a real analytic function then

$$\frac{df}{d\alpha} \;=\; \frac{f(\alpha + \Delta\alpha) - f(\alpha - \Delta\alpha)}{2\Delta\alpha} \;+\; \mathcal{O}(\Delta\alpha^2) \tag{4.1}$$

Computationally this will require $2n$ iterative solutions of the nonlinear flow solver. It is straightforward to implement using all the modules being used as black boxes. However, because of the finite precision arithmetic, the subtractive cancellation error associated with the finite difference becomes increasingly important as the step size $\Delta\alpha$ gets smaller. Hence, experimentation with the step size is required to balance between the truncation error and the subtractive cancellation error for accurate gradients. A much better modification of this method is the Complex Variable Trick (CVM) as discussed in section 4.6.1.

The other method is to develop linear or adjoint versions of the original nonlinear modules. The linear and adjoint equations then have to be solved iteratively. Propagating a perturbation from α to J is known as the forward mode (in AD community). Propagation of adjoint variables from J to α is known as the reverse (or adjoint) mode. Complete forward propagation of first order sensitivities will be computationally $\mathcal{O}(n)$ while the backward propagation will result in $\mathcal{O}(m)$ computational cost. Various hybrid versions of this sensitivity propagation method are possible by partial forward and reverse mode propagations along the chain from α to J.

Though this approach is computationally more efficient, it presupposes the availability of the original nonlinear code. More importantly, even if the nonlinear code is available, development and maintenance of the linear/adjoint codes is a complex task. The next section presents the basic formulation of the linear and adjoint solutions.

4·3 FORMULATION

Considering a weak implementation of all the boundary conditions[21], the discretised form of flow equations can be written as

$$R(U,X) = 0\,, \tag{4.2}$$

where U is a vector of flow variables and X is a vector of coordinates of mesh points used for the discretisation of the equations. It should be noted that this is possible for any generic Reynolds averaged Navier-Stokes equations. These equations are solved iteratively for the flow solution U. This constitutes the nonlinear solver. In the example airfoil code, two dimensional Euler equations have been solved using a predictor-corrector algorithm. Here, U and X are $4p \times 1$ and $2p \times 1$ vectors respectively with p being the number of mesh points used for discretisation.

The linearised version of equation (4.2) with respect to α is

$$\frac{\partial R}{\partial U}u + \frac{\partial R}{\partial X}x = 0\,, \tag{4.3}$$

[21] Weak formulation refers to the weak imposition of boundary conditions at the solid walls. This entails specification of zero mass flux through faces on the wall surface for the Euler equations. If there are mesh points on the wall then the wall boundary condition is imposed by forcing the normal component of the velocity to be zero. This is known as the strong boundary conditions. Refer to Giles et al. [37] for the formulation of adjoint solvers with strong boundary conditions.

where

$$u = \frac{dU}{d\alpha}, \text{ and } x = \frac{dX}{d\alpha}.$$

The variables u and x are known as the perturbation variables. The above equation can be rewritten as,

$$Lu = f, \tag{4.4}$$

where,

$$L = \frac{\partial R}{\partial U}, \text{ and } f = -\frac{\partial R}{\partial X}x.$$

Typically equation (4.4) is solved iteratively to obtain the linear flow perturbations u in the linear solver. The perturbation in the objective function is

$$\frac{dJ}{d\alpha} = g^T u + \frac{\partial J}{\partial \alpha} \tag{4.5}$$

where,

$$g^T = \frac{\partial J}{\partial U}, \text{ and } \frac{\partial J}{\partial \alpha} = \frac{\partial J}{\partial X}x$$

Feeding in the flow and mesh perturbation values, the desired gradient is calculated. Multiple linear solutions are required to calculate the overall perturbation in the objective function for multiple design variables.

The adjoint solver is developed along similar lines to the linear solver. The discrete approach [41] is used here as opposed to the continuous approach explored by Jameson *et al* [56]. This means that the nonlinear flow PDEs are discretised first and then the adjoint formulation is obtained from the discretised PDEs. Though mathematically both approaches should give equally accurate results, the discrete approach has many advantages at the implementation level. In particular, the whole process can be automated with the use of AD tools.

The linear perturbation in the objective function as given by equation (4.5) can also be calculated as

$$\frac{dJ}{d\alpha} = v^T f + \frac{\partial J}{\partial \alpha}, \tag{4.6}$$

where v is the vector of adjoint variables which satisfy

$$L^T v = g, \tag{4.7}$$

and f is as defined in equation (4.4). The equivalence of these two formulations can be shown as

$$v^T f = v^T L u = (L^T v)^T u = g^T u.$$

Equation (4.6) is independent of the linear perturbations u. Hence, only one adjoint solution for every objective function is required irrespective of the number of design variables. This has led to a widespread use of adjoints in design and optimisation [2, 27].

4.4 FIXED POINT ITERATIONS

The linear equations thus obtained are solved using the fixed point iteration method used for the nonlinear code, though any method can be used independent of the solution methodology used for the original nonlinear solver. A two step predictor-corrector method is used for the present two dimensional Euler nonlinear equations. The time integration scheme for the n^{th} time-step and i^{th} cell for the linear solver is

$$\begin{aligned} u_i^{int} &= u_i^n - P_i((Lu^n)_i - f_i) \\ u_i^{n+1} &= u_i^n - P_i((Lu^{int})_i - f_i)\,, \end{aligned} \tag{4.8}$$

where P is a diagonal matrix with the i^{th} diagonal value being equal to the local time-step divided by the cell area. The convergence rate of these iterations is equal to the asymptotic rate of convergence of the original nonlinear equations. It should be noted here that the matrix L and vector f remain constant throughout the fixed point iterations. Hence, the execution time can be greatly reduced by storing them at the start of the solution.

The central issue in the development of an adjoint solver is the computation of $L^T v$. Consider a face between the i^{th} and j^{th} cells. Let the nonlinear flux through this face be R_{ij} defined by equation (A.2). Also consider the L matrix with contributions from only this face. Then

$$\begin{aligned} (Lu)_i &= \frac{\partial R_{ij}}{\partial U_i} u_i + \frac{\partial R_{ij}}{\partial U_j} u_j\,, \\ (Lu)_j &= -\frac{\partial R_{ij}}{\partial U_i} u_i - \frac{\partial R_{ij}}{\partial U_j} u_j\,. \end{aligned}$$

By considering the L matrix with only the contribution from this face and then transposing the matrix, it can be quickly seen that the corresponding adjoint flux contributions to these cells are

$$(L^T v)_i = \frac{\partial R_{ij}}{\partial U_i}^T (v_i - v_j),$$
$$(L^T v)_j = \frac{\partial R_{ij}}{\partial U_j}^T (v_i - v_j).$$

It should be noted here that the above expressions are valid for boundary conditions also. In the case of weak boundary conditions, the discrete adjoint approach does not require the derivation of the adjoint boundary conditions as needed by the continuous adjoint approach. Correct adjoint boundary conditions emerge simply by properly transposing the L matrix. Though this would not have been a big issue in the present case, it might be difficult for more complex production grade solvers.

The correctness of the L^T matrix can be checked using the following simple identity

$$v^T(Lu) = u^T(L^T v). \tag{4.9}$$

As this identity holds for any values of u and v, it can be used a check for the correct implementation for an individual cell in the interior and at the various boundaries. This greatly speeds up the debugging process. After the construction of the L^T matrix, the system of linear equations (4.7) is solved in an explicit manner. The predictor-corrector fixed point iteration scheme for the linear solver given by equation (4.8) can be rewritten as

$$u^{n+1} = u^n - Q(Lu^n - f), \tag{4.10}$$

where $Q = P(I - LP)$ and I is the identity matrix. The adjoint time marching scheme can be written as

$$v^{n+1} = v^n - Q^T(L^T v^n - g) \tag{4.11}$$

But since P is a diagonal matrix

$$Q^T = (I - P^T L^T)P^T = (I - P L^T)P = P(I - L^T P).$$

Hence, the same predictor-corrector algorithm can be used for the adjoint solution. Though the time integration scheme used here is self-adjoint, this is not true for most other methods like Runge-Kutta family of schemes. In general, more effort is required to find the adjoint time integration operator Q^T [35].

It should also be noted here that the rate of convergence for the linear and adjoint solvers is identical as $I - QL$ and $I - Q^T L^T$ have identical eigenvalues.

The following approach is used to ensure that the adjoint solver gives exactly the same sensitivities as the linear solver (and thereby use all the efforts put in the validation of the linear code). If both the linear and adjoint solutions start with the initial conditions of $u^0 = 0$ and $v^0 = 0$ then it can be shown that both methods give exactly the same perturbation for the objective function J with equal number of iterations.

The following lemmas provide the key results to prove the final theorem.

Lemma 4.1

If $u^0 = 0$, then for the time marching scheme given by equation (4.10) *the iterate after the n^{th} step is given by*

$$u^n = \sum_{m=0}^{n-1} (I - QL)^m \, Q \, f$$

Proof

Proof by induction. Supposing that the lemma is true for $n = i - 1$, then

$$u^{i-1} = \sum_{m=0}^{i-2} (I - QL)^m \, Q \, f \, .$$

Then for $n = i$,

$$\begin{aligned} u^i &= u^{i-1} - Q(Lu^{i-1} - f) \\ &= (I - Q\,L)u^{i-1} + Q\,f \end{aligned}$$

Using the equation for u^{i-1},

$$\begin{aligned} u^i &= \sum_{m=1}^{i-1} (I - QL)^m \, Q \, f + Q \, f \\ &= \sum_{m=0}^{i-1} (I - QL)^m \, Q \, f \end{aligned}$$

Hence true for $n = i$.

For $n = 1$, since $u_0 = 0$,

$$\begin{aligned} u^1 &= u_0 - Q(L\,u_0 - f) \\ &= Q\,f \\ &= \sum_{m=0}^{1-1} (I - QL)^m\,Q\,f \end{aligned}$$

Hence proved by induction. □

Lemma 4.2

If $v_0 = 0$, then for the time marching scheme given by equation (4.11) *the iterate after the n^{th} step is given by*

$$v^n = \sum_{m=0}^{n-1} (I - Q^T\,L^T)^m\,Q^T\,g$$

The proof runs on the similar lines as the previous one.

Theorem 4.1

Given lemmas 4.1 *and* 4.2*, then $(v^n)^T\,f = g^T\,(u^n)$.*

Proof

Using the two lemmas,

$$\begin{aligned} g^T\,u^n &= g^T \left(\sum_{m=0}^{n-1} (I - Q\,L)^m \right) Q\,f \\ &= g^T\,Q \left(\sum_{m=0}^{n-1} (I - L\,Q)^m \right) f \\ &= g^T\,Q \left(\sum_{m=0}^{n-1} (I - (L\,Q)^T)^m \right)^T f \\ &= \left(\sum_{m=0}^{n-1} (I - Q^T\,L^T)^m\,Q^T\,g \right)^T f \\ &= (v^n)^T\,f \end{aligned}$$

□

Since $\frac{\partial J}{\partial X}\frac{\partial X}{\partial \alpha}$ is the same for both methods, it follows that the two methods give identical values for the gradient. Consequently, full machine precision agreement between the gradient of the objective function given by the linear and adjoint codes can be achieved at every iteration.

It is noted by Christakopoulos et al. [18] that machine precision agreement between the gradient of the objective function obtained by linear and adjoint solvers for equal number of iterations is only relevant during the validation process. During the routine use of the adjoint solver, it is not essential to make sure the iterative equivalence of the gradient of the objective function. Hence, once the adjoint code is validated against the linear solver, there is no need to use the adjoint time integration operator Q^T for the adjoint solution.

It is proposed that a simple self-adjoint time integration operator (like predictor-corrector method) can be used for validation of the adjoint code by ensuring the iterative equivalence of the gradient of the objective function. After validation, any suitable time integration scheme can be used for the adjoint solvers. This obviates the need for the derivation of the transpose of the time integration operator for advanced fixed point iterations. It should be noted that the above statement is only valid for a contractive Jacobian. However, the iterative process needs to be appropriately checkpointed and reversed in the case of dual time-stepping.

4.5 USE OF AD

The development of the linear and adjoint solvers by hand coding is difficult for production grade fluid mechanic solvers (e.g. Navier-Stokes equations with various turbulence models). The linear matrices given in Appendix A for a simple two dimensional Euler code illustrate the difficulty of hand-coding for linear/adjoint solvers. Human errors introduced during the development stage are undesirable. Also if the core nonlinear modules are being continuously modified by various people, it is practically impossible to keep the linear and adjoint versions of the code consistent with it using hand-coding. Hence AD is the only way forward.

The linear and adjoint solutions obtained using AD can be used to calculate the entire Jacobian matrix. However, in practice, only the Jacobian-vector product (directional derivatives) are required for most applications like gradient based optimisation. The

Jacobian matrix thus calculated using AD is accurate to machine precision [45]. The Jacobian matrix is accurate in the sense that it approximates the difference of the discrete flux vector of the original nonlinear solver for a given perturbation. However, the accuracy of the Jacobian matrix thus obtained in approximating the analytic Jacobian of the PDEs depends on the truncation errors introduced in approximating the PDEs over the domain using a discrete flux vector and, the convergence of the original nonlinear solution.

Giles et al. [37] have reported 20–30% greater memory requirements for the adjoint code as compared to the nonlinear code depending on the mesh. The CPU cost per iteration is reported to be only 10–20% greater than for the nonlinear code. Such efficiencies can be achieved by careful hand-coding followed by meticulous validation tests. However, the human effort required for the development and the continuous maintenance of the linear/adjoint code makes it unattractive for the industrial environment. Additionally, a similar (or slightly inferior) performance can be achieved by intelligent restructuring of the nonlinear code and selective application of AD as outlined in the subsequent section. The penalty in the performance is more than compensated by the automation of the linear/adjoint code development.

Much work has gone into the development of adjoint codes over the years [2, 36, 37, 71]. The idea of using AD as a tool to automate the process is relatively new with the first application being the development of the adjoint version of an ocean circulation model developed at MIT [33]. Mohammadi et al. [71] reported one of the earliest AD applications in turbomachinery. Courty et al. [23] presented one of the earliest demonstrations of the partial usage of AD for developing adjoint solver for use in optimal design problems.

Christianson [20] presented the key ideas behind the use of reverse mode AD (reverse accumulation) for calculating the gradient of a linear objective function that requires the solution of a linear implicit function. The algorithm is then extended to the case of nonlinear implicit functions and to general functions that use implicit functions serially or nested in their construction. This key idea is used by all the present AD implementations of the adjoint solver which consider the solution of the nonlinear flow solution as an implicit function constructed out of a series of implicit nonlinear functions.

Recently Christakopoulos et al. [17, 18] have reported the development of a methodology for efficient usage of AD for linear/adjoint solvers. They have compared the computational cost of the black-box application of AD with an alternative efficient

algorithm for adjoint solution termed as the primal time-stepping (PTS) adjoint algorithm. The work also presents a detailed explanation of various issues related to the efficient use of AD for adjoint code development for production grade fluid mechanic solvers. Implementation details for incorporating multigrid acceleration and pre-conditioning are also discussed. They have reported the computational cost of one adjoint solution at 2.6 times the computational cost of one nonlinear solution using the PTS adjoint algorithm. The test solver includes multigrid acceleration and the block Jacobi pre-conditioner.

AD can be used at various levels of user interaction for the linear and adjoint solver development. The black-box application of the AD in reverse mode is not feasible for the fluid mechanic solvers as the tape-size of the data stored by the AD software in the first pass would be huge because of the fixed point iteration loop. In order to make the tape-size manageable, the values of the intermediate variables would have to calculated again in the reverse pass making the algorithm computationally expensive. The black-box approach can be made more efficient (for example as demonstrated by [17]), by using pragma definitions (in case of TAPENADE, `$AD II-LOOP`) to help the AD software identify iteratively independent loops.

However, for the application of fluid mechanic solvers, these approaches are still not suitable as demonstrated by Christakopoulos [17]. Hence, to reduce the CPU and memory usage, a selective use of AD is required. However the selective use of AD requires careful structuring of the original nonlinear code to allow for efficient use of AD.

To illustrate this point, the next sections summarise the structure of the nonlinear code and the key aspects of differentiating the code for linear/adjoint versions.

4.5.1 *Nonlinear Solver*

All the nonlinear subroutines need to be developed in a modular way so as to facilitate the use of AD tools. The following key nonlinear modules are developed:

- `time_cell` Computes the local Adt for a cell (equation (A.3))
- `flux_face` Computes flux across an internal face (equation (A.2))
- `flux_wall` Computes flux for a wall face
- `lift_wall` Computes lift for a wall face (equation (A.4))

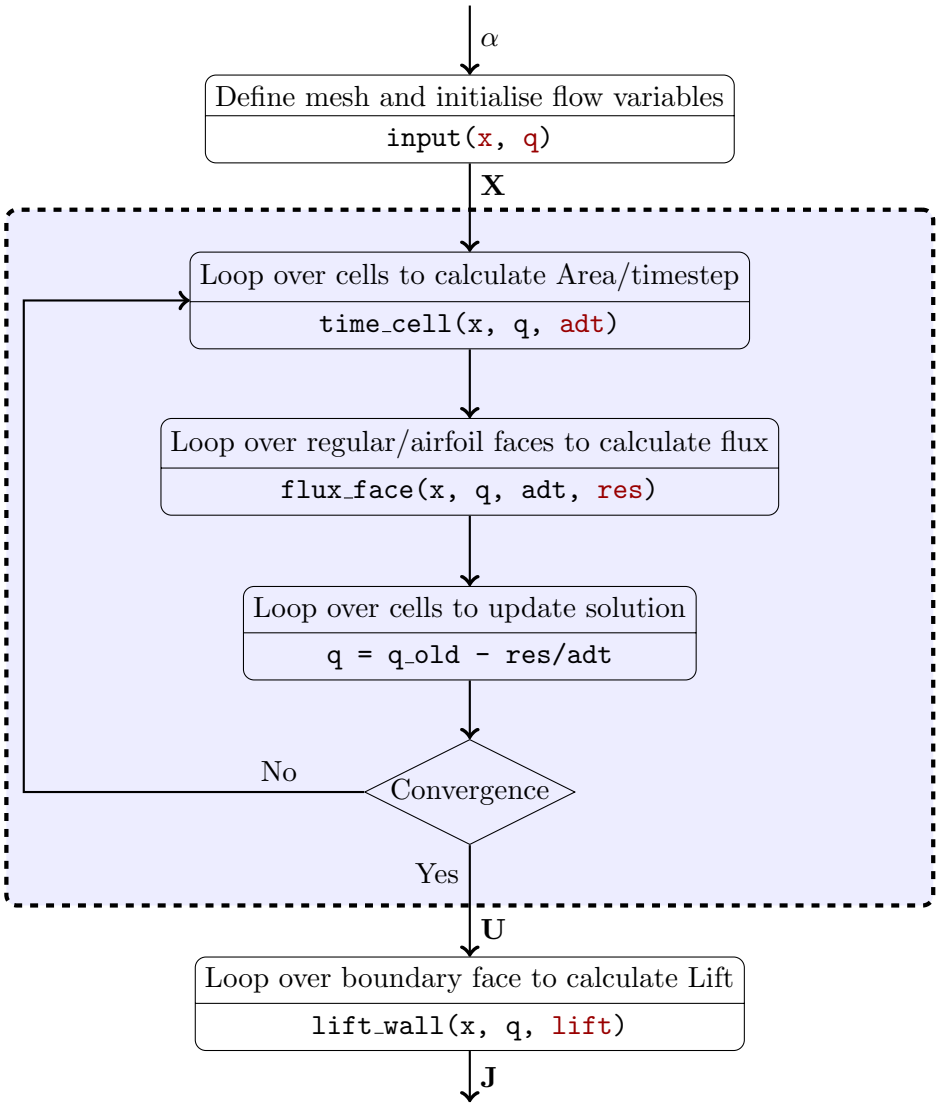

Figure 4.1 Schematic for the nonlinear solver

The input to these functions are of the form

```
func(geometry parameters, flow parameters, output)
```

.

This makes it easy to specify the input and output variables for the AD software. Figure 4.1 gives the algorithm for the nonlinear solver. It should be noted that all the nonlinear routines are separated out as small modular functions which can then be passed on to the AD software.

4.5.2 *Linear Solver*

TAPENADE[22] developed by Laurent Hascoët and others at INRIA [24, 51] is used in this work. This tool can differentiate FORTRAN codes using source transformation (as opposed to operator overloading as used for C++). TAPENADE generates linear perturbations (forward mode AD) and adjoint perturbations (reverse mode AD) for a code.

The linear code is generated by applying TAPENADE to the core nonlinear routines instead of the complete nonlinear solver. Hence, the outer fixed point iteration loop is maintained similar to the original nonlinear code and only the nonlinear subroutines are replaced with the appropriate linear subroutines generated by AD.

Each subroutine has to be differentiated twice: once to differentiate with respect to the flow variables (to generate the elements of L matrix) and then differentiated with respect to the mesh perturbations (to generate the elements of the f vector). For the `flux_face(x, q, res)` subroutine with `x` as the mesh input, `q` as the flow variable input and `res` as the residual output, the following two linear versions are produced after processing it with the AD software,

- Differentiation with respect to the flow variables

  ```
  flux_face_d(x, q, qd, res, resd)
  ```

- Differentiation with respect to the geometry variables

  ```
  flux_face_dx(x, xd, q, res, resd)
  ```

Here all the variables with `d` suffix are the forward perturbation variables.

The linear subroutines thus generated can be tested and validated for a single cell using the methods outlined in section 4.6. Finally, a main program for the linear solver is written that calls all these linear versions of the nonlinear subroutines. Figure 4.2 shows the schematic for the linear solver. Different colours are used to highlight the active input (blue) and output (red) variables. Linear solver thus obtained may use a different fixed point iteration scheme than the nonlinear solver. But it is more convenient to use the same time integration scheme as the original nonlinear solver.

---

[22] `http://www-sop.inria.fr/tropics`

It should also be noted that since the right hand side of the linear equation (equation (4.4)) does not change during the fixed point iteration, it is calculated outside the iteration loop and used during each iteration reducing the computational cost.

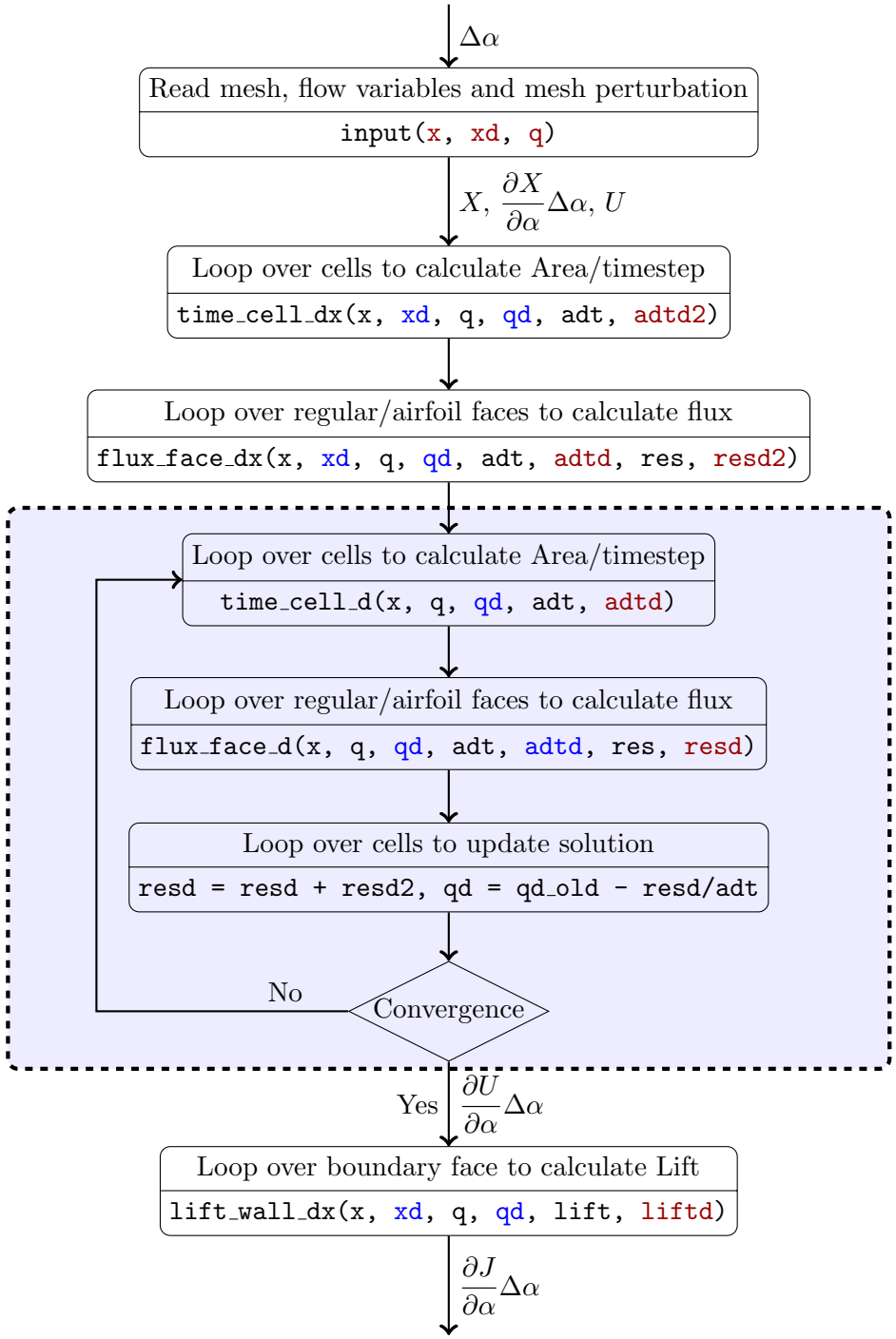

Figure 4.2 Schematic for the linear solver

4.5.3 *Adjoint Solver*

The development of the adjoint solver runs on the similar lines, though it is not as intuitive as the linear solver. The nonlinear routines are differentiated twice in the reverse mode (adjoint mode). Again looking at the `flux_wall` function, if `q` is the active input variable and `res` is the output variable then TAPENADE generates

`flux_wall_b(x, q, qb, res, resb)`.

Here `resb` is the input perturbation which is propagated to `qb` and not the other way round. All the variables with `b` suffix are called the adjoints of the corresponding original variables. `qb` is essentially $\left(\frac{\partial J}{\partial q}\right)^T$. All the subroutines need to be called in the reverse order compared to the linear solver. Figure 4.3 shows the schematic of the adjoint solver.

A unit perturbation in the objective function J is propagated to calculate the right hand side g (equation (4.7)) of the adjoint equation. Since g does not change during the fixed point iterations, it is calculated only once outside the fixed point iteration loop and used thereafter. Inside the iteration loop, adjoint residual calculation is carried out by first calculating the flux residuals followed by the adjoint time step calculation. After each adjoint residual calculation, the right hand side g is subtracted from the residuals. It should be noted here that the adjoint variables ν are stored in the `resb` variable while the residuals are stored in `qb` variable in the adjoint code generated by AD.

Once a converged solution of the adjoint variables is obtained, it is used to calculate the linear perturbation in the objective function for any mesh perturbation. The airfoil code used to demonstrate the ideas uses a simple rotation of the mesh to demonstrate the idea. However, in general, any mesh perturbation corresponding to a design variable can be provided to obtain the corresponding perturbation in the objective function.

The whole process of generating the linear and adjoint codes is automated using a Unix `Makefile`. A `Makefile` is written to generate all the linear/adjoint versions of the original nonlinear subroutines. Each directive in the `Makefile` specifies the active input and output variables for the TAPENADE and the mode of differentiation. Various linear/adjoint versions of the original nonlinear subroutines can be obtained by changing the active variables

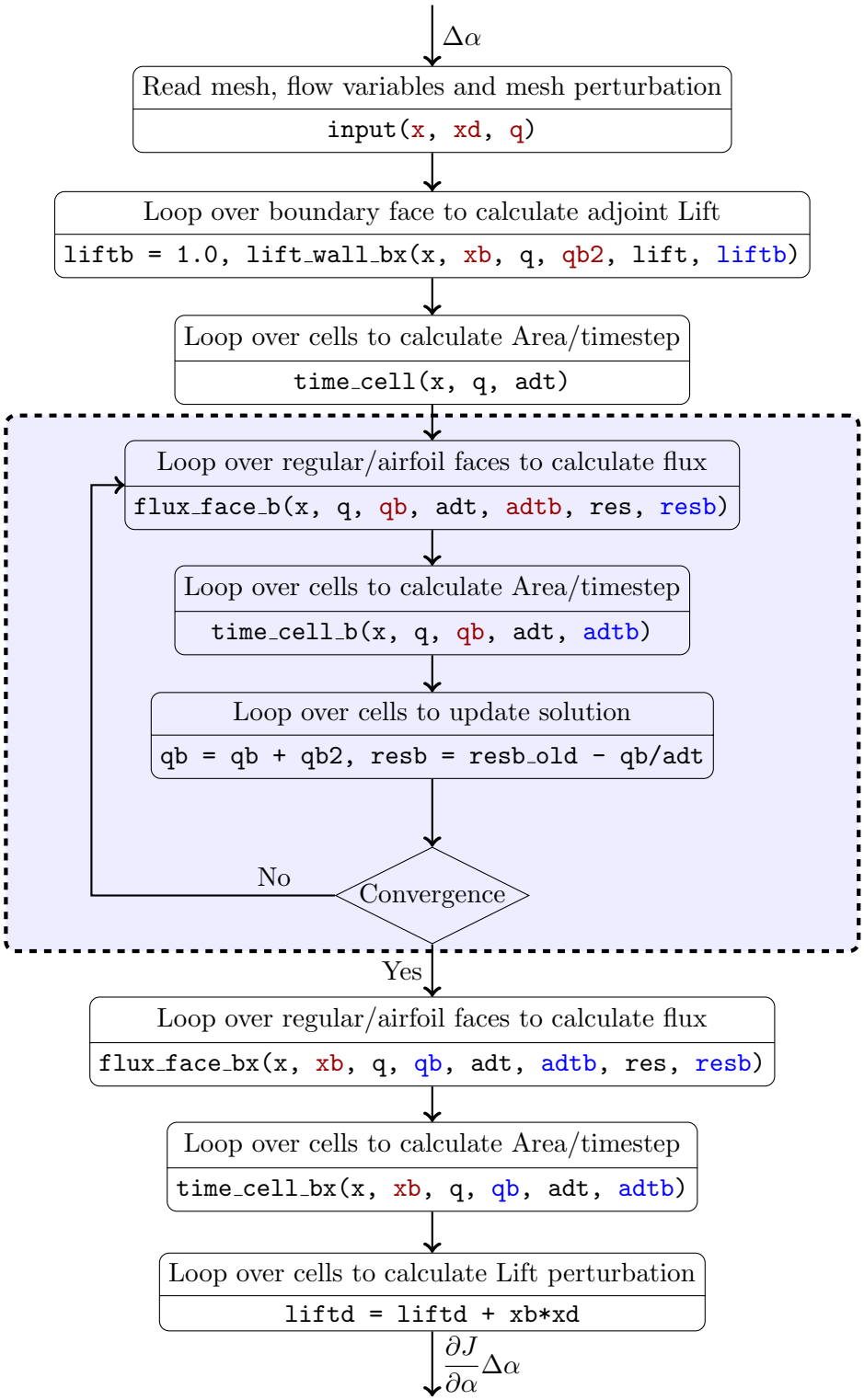

Figure 4.3 Schematic for the adjoint solver

Refer to Jones et al. [58] for a generic and scalable method of handling the creation of linear/adjoint codes using `Makefile` suitable for industrial fluid mechanic solvers. The features that are addressed include differentiating derived datatypes, the use of hand differentiated subroutines along with the AD generated subroutines, and selective differentiation of certain nonlinear subroutines by using user-defined macros.

The only code that is written by hand is the nonlinear solver and the main programs for the linear and adjoint solvers that call all the differentiated subroutines and set up the fixed point iteration loop. Again these solvers have the properties stated by Theorem 4.1.

4.6 CODE VALIDATION

There are two steps of code validation: firstly validating linear code against the nonlinear code and secondly the adjoint code against the linear code. The most straightforward way for the linear code validation is by checking the perturbation in the objective function with the finite difference perturbations. In other words, for a small enough α

$$J(X+\alpha x,U_0+\alpha u)-J(X,U_0)\approx\alpha\left(\frac{\partial J}{\partial U}(U_0)\,u+\frac{\partial J}{\partial X}(U_0)\,x\right), \tag{4.12}$$

where U_0 is the converged solution of the nonlinear equations.

If equation (4.12) is not satisfied then the two components on the right hand side have to be investigated separately for possible errors. However, even if equation (4.12) is satisfied, the level of agreement varies with the function of interest and the step size [69]. Hence, the level of accuracy to which equation (4.12) should be satisfied is not known *a priori*. This necessitates other methods of verification.

The following tests are carried out to make sure that the linear solution is consistent with the nonlinear solution.

- Ensure that the nonlinear solution is completely converged

$$R(X,U)\approx 1e^{-16}$$

- Ensure that the linear solution is converged

- Ensure that the linearisation with respect to the mesh perturbation is correct

$$R(X + \alpha x, U) - \alpha x \frac{\partial R}{\partial X} = \mathcal{O}(\alpha^2)$$

- Ensure that the linearisation with respect to the flow perturbation is correct

$$R(X, U + \alpha u) - \alpha u \frac{\partial R}{\partial U} = \mathcal{O}(\alpha^2)$$

- Ensure that the overall implementation is consistent

$$R(X + \alpha x, U + \alpha u) - \alpha x \frac{\partial R}{\partial X} - \alpha u \frac{\partial R}{\partial U} = \mathcal{O}(\alpha^2)$$

The following tests may be implemented for a single cell in the interior and on the various boundaries. This provides an effective sensor to quickly and automatically identify any inconsistencies.

4.6.1 *Complex Variable Method*

As noted earlier in equation (4.1), central finite difference approximation of a derivative suffers from the subtractive cancellation error and hence an optimal step size $\Delta\alpha$ cannot be specified *a priori*. However, if the complex Taylor series expansion of function J is used, Complex Variable Method (CVM) [87] approximation follows. Again, if f is real-valued and analytic, the complex Taylor series expansion of f about $x + \mathrm{i}\Delta x$ is

$$f(x + \mathrm{i}\Delta x) \;=\; f(x) \;+\; \mathrm{i}\Delta x \frac{df}{dx} \;+\; \mathcal{O}(\Delta x^2)$$

where i denotes the imaginary part of a complex quantity. By comparing the imaginary parts of both sides of this equation, following approximation for the derivative is obtained.

$$\frac{df}{dx} = \frac{\Im(f(x + \mathrm{i}\Delta x))}{\Delta x} \;+\; \mathcal{O}(\Delta x^2) \tag{4.13}$$

where $\Im$ represents the imaginary part of its arguments. As the subtraction of two quantities of similar magnitude is not involved in the above expression, there is no loss of accuracy due to finite precision arithmetic. Also, the convergence to limiting value is second order in Δx meaning that for $\Delta x < 10^{-8}$, a double precision accuracy is achieved. In practice, a value of $\Delta x = 10^{-20}$ is used.

For programming languages that support complex datatypes, it is relatively easy to modify the code and get the gradient information. All the double precision real

variables are declared as double precision complex variables. Since CVM is only valid for analytic functions in the complex plane, some of the functions used in the fluid mechanics code need to be redefined to make them analytic in the complex plane. The airfoil code required only the redefinition of absolute value function abs(x) to

$$\text{abs}(x) = \text{sign}(\mathbb{R}(x))x \tag{4.14}$$

CVM can be efficiently implemented using compiler preprocessors. However, it is a much more involved task while using languages like C that do not support complex variable type.

For the adjoint code validation, Theorem 4.1 is used. If the adjoint code is consistent with the linear code and the initial conditions for the linear and adjoint codes are $u^0 = 0$ and $v^0 = 0$ then the perturbation in the objective function J should be equal to machine precision for both the solvers. If this is not the case then equation (4.9) can be used to pin-point the cell where L^T matrix is not being calculated consistently with the linear solver. This simple identity acts as a sensor throughout the domain.

These methods are a powerful tool when AD is used for the linear/adjoint code generation. Once set up as different modules these tests can automatically pin-point the region of error in an environment where the original nonlinear code is being changed on a daily basis followed by its automatic differentiation to generate the linear code.

4.7 IMPLEMENTATION AND RESULTS

The mesh generator is implemented in MATLAB and gives a 16-digit accurate ASCII mesh file. All the solver routines are implemented using FORTRAN language with a C preprocessor being used to generate the nonlinear, linear and adjoint versions. The f and g vectors are stored initially as they remain constant throughout the iterations. It should be noted that the L matrix does not change throughout the linear and adjoint solution process. Hence, a computationally efficient but memory intensive approach is to store it throughout the solution process. However, the present implementation effectively calculates the L matrix at each iteration.

The example two dimensional Euler source code complete with documentation and `Makefile` is available at the website[23]. All the validation tests described in this

[23] `http://people.maths.ox.ac.uk/gilesm/airfoil/bangalore05.tar`

chapter are implemented in a function called `testlinadj`. This code demonstrates all the key ideas described in this chapter.

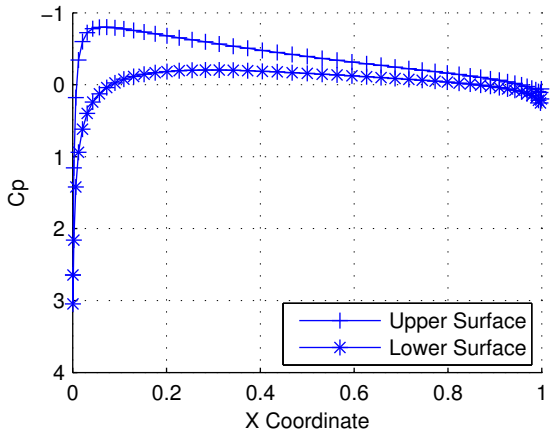

Figure 4.4 Plot of Cp distribution on the airfoil surface

The operating conditions (in SI units) for the validation tests are:

- Pressure = 1 N/m^2
- Density = 1 kg/m^3
- Mach = 0.4
- Flow angle = 3^o
- Angle of Attack perturbation = 0.01^o

Figure 4.4 shows the Cp distribution on the airfoil surface for $p_\infty = 1$, $\rho_\infty = 1$, Mach = 0.4 and $\alpha = 3^o$ with Cp defined as

$$Cp_i = \frac{2(p_i - p_\infty)}{\rho_\infty(u_\infty^2 + v_\infty^2)},$$

where the quantities with subscript infinity are far-field conditions. The upper and the lower surfaces of the airfoil are plotted separately. Note here that the direction of the Y-axis has been changed which is a standard practice in the aerospace community. This plot only illustrates that the nonlinear code is behaving reasonably in the subsonic range. The idea behind this exercise is to develop linear/adjoint codes consistent with the original nonlinear code. Hence it is not claimed in any way that this is an optimal implementation for good quality solution of two dimensional Euler solutions.

Figure 4.5 shows the behaviour of the last test as described in section 4.6 for all the four residuals over the entire domain as the perturbation angle α is successively increased. As expected all the residuals show quadratic behaviour which confirms the consistency between the linear and nonlinear solvers. Similar plots can be obtained for the interior cells and different boundary conditions to ensure complete consistency.

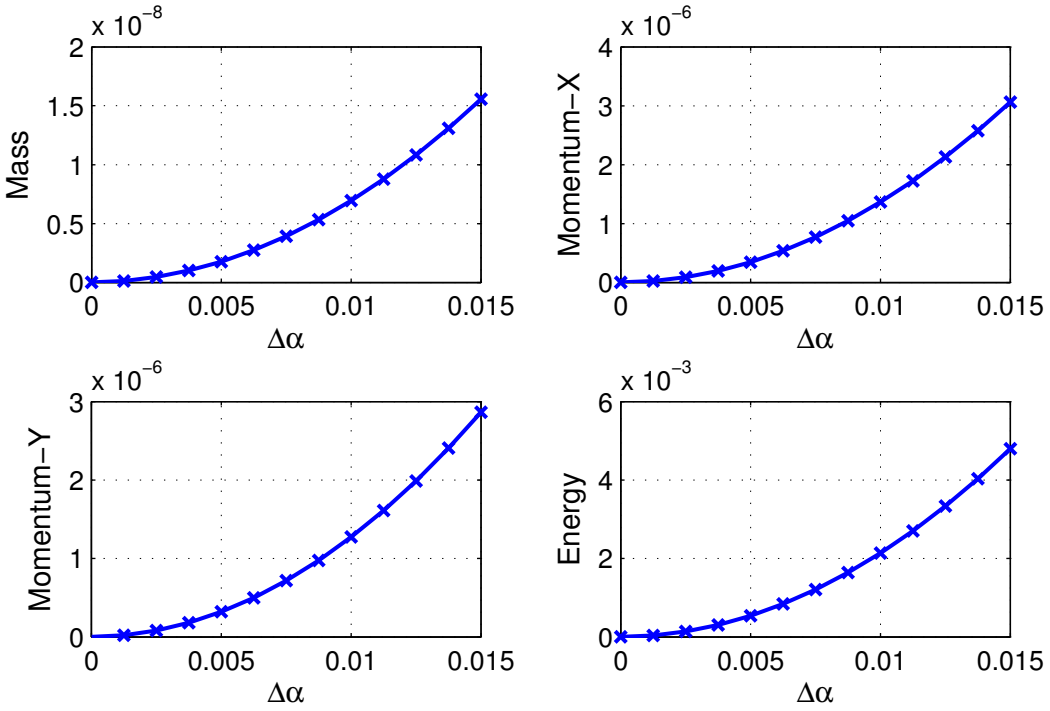

Figure 4.5 Quadratic behaviour for all the residuals over the entire domain

Figure 4.6 shows the relative error in calculating the sensitivity of lift with decreasing step-size for central finite difference. As the stepsize for α is reduced, the truncation error reduces proportionally. However, the subtractive cancellation error of equation (4.1) takes over below a certain step-size. Hence, there is no accuracy advantage gained by reducing the step-size any further. On the contrary, the accuracy deteriorates as the subtractive cancellation error becomes more prominent as the step-size is further reduced. The optimum step-size for the airfoil code was found to be $\Delta\alpha = 10^{-4}$.

As a next level of validation, Table 4.1 shows the comparison of the perturbations in lift obtained by three approaches. The first is the central finite difference of the nonlinear solutions, the second is the perturbation from the linear solver and the last is from the adjoint solver. The nonlinear, linear and adjoint codes are

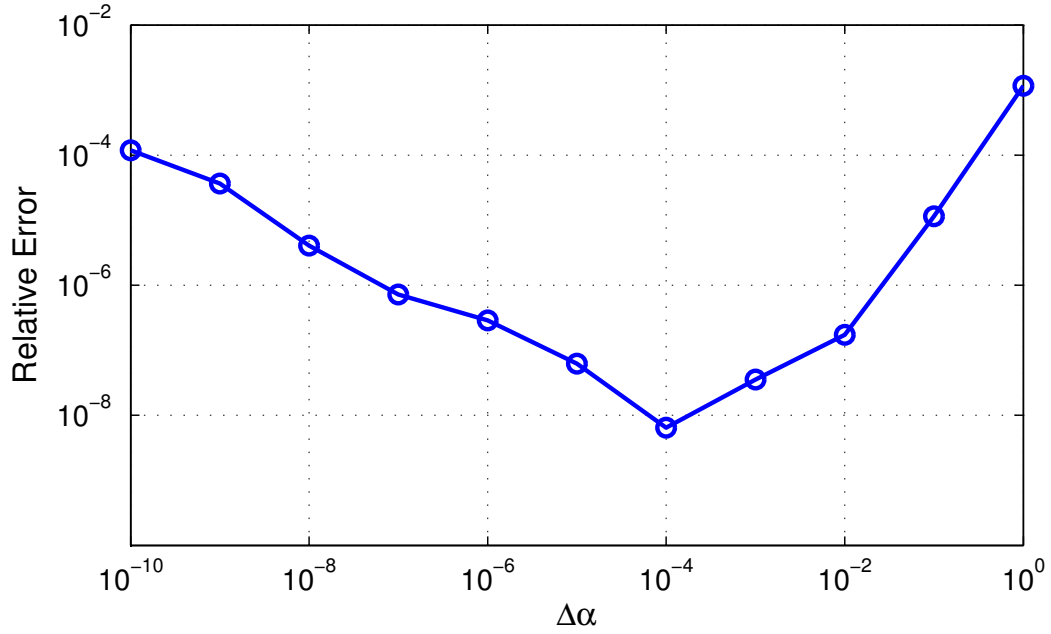

Figure 4.6 Relative error in finite difference sensitivity as a function of the step size $\Delta\alpha$

	Lift Perturbation
FD	**1.2377**4e+1
Linear	**1.2377**08778491253e+1
Adjoint	**1.2377**08778491255e+1

Table 4.1 Comparison of lift perturbations from various methods for $\delta\alpha = 0.01^o$

converged to machine precision. An agreement of up to five digits can be seen between the FD and the linear/adjoint codes. This shows that the linear/adjoint codes are consistent with the nonlinear solver. Also it should be noted that the linear and adjoint solutions give exactly the same perturbation (up to machine precision) in the objective function, since both codes are run for an equal number of iterations. This completes the entire validation process.

4.7.1 *Computational Cost*

This section compares the computational cost of the linear and the adjoint solvers to the original nonlinear solver. The calculations are performed on a Sony VAIO laptop with Intel i3-380M processor 2.53 GHz (3 MB L3 Cache) running Archlinux

operating system. TAPENADE (version 3.9) is used to differentiate the code. All the codes are compiled using GCC compilers (version gcc-4.8.2) with the highest level of optimisation.

The CPU time is measured using the native Linux command `time` and the user CPU time is reported. 1000 iterations were performed for all the codes. A set of 10 simulations were performed for each solver and the average value of time (in seconds) is reported in Table 4.2. Since the airfoil code calculates only inviscid fluxes and simple fixed point iterations, an acceptable computational overhead is observed for the linear and adjoint code. However, more efforts for code optimisation will be required when dealing with production grade CFD solvers [18, 58].

Solver	CPU time (sec)	Ratio
Nonlinear	8.3	1.0
Linear	14.94	1.8
Adjoint	15.15	1.83

Table 4.2 Comparison of the compuatational cost of the adjoint and the linear solvers with the original nonlinear airfoil code for 1000 iterations

4.8 CONCLUSION

This chapter demonstrated a step-wise approach for generating linear/adjoint codes from the original nonlinear solver using AD. The automation achieved using AD is a valuable tool in maintaining production grade codes which are constantly being modified by various teams working across the globe. Also, a well established foolproof method of validation checks have been described that ensures consistency between the nonlinear and linear codes, and the linear and adjoint codes. This formulation and code development methodology is extended to Hessian calculation in the next chapter.

5 HESSIAN CALCULATION

The effective use of AD to automate the process of Jacobian calculation has been demonstrated in the previous chapter. The Jacobian thus obtained is accurate to machine precision [45]. Selective and efficient use of AD is extended in this chapter for accurate and computationally efficient computation of the Hessian. Also, AD helps in keeping the linearised version of the nonlinear codes in-sync with the continuous changes made in the nonlinear code. The proposed method for Hessian calculation is a natural extension of this.

The Hessian thus obtained has various applications in the fields of optimisation algorithms, Monte Carlo simulations, surrogate modelling and uncertainty analysis.

5.1 BACKGROUND

The idea of calculating Hessians using AD is not new and the AD community has been addressing the issue of calculating higher order derivatives for a number of years. In one of the earliest papers, Christianson [19] describes an algorithm for Hessian calculation using reverse accumulation. Several strategies can be applied [85, 88] depending on the method of gradient calculation employed. The work reports that the forward-on-forward approach has a computational cost that is quadratic in the number of independent variables, while the rest – forward-on-reverse, reverse-on-forward and reverse-on-reverse are linear for the Hessian-vector product. The last approach is difficult to implement using AD (because of the `PUSH/POP` routines introduced in the reverse mode AD) while the reverse-on-forward approach does not present any computational advantage over forward-on-reverse. Hence, the two commonly used methods for calculating Hessians using AD are: forward-on-forward and forward-on-reverse. Forward-on-forward is a straightforward double application of AD to the original code in forward mode. If there are n independent variables, then the computational cost of this approach is $\mathcal{O}(n^2)$. Similarly, in forward-on-reverse mode, the code is differentiated first in the reverse mode and then in the forward mode. The computational cost for an objective function of dimension m and n independent variables is $\mathcal{O}(m \times n)$[24].

Presently, most AD tools — ADOL–C [46, 47], ADIFOR [7, 29], TAPENADE [51] and OpenAD [92] can be used to calculate the Hessian using forward-on-forward or forward-on-reverse modes. In addition, ADOL–C provides in-built driver routines that calculate the Hessian or Hessian-vector products. The black box application of AD for the calculation of the Hessian matrix using forward-on-forward mode will entail $\mathcal{O}(n^2)$ iterative solutions. Some of the cost can be alleviated by identifying the iteratively independent loops (`$AD II_LOOP` in TAPENADE). However, it is still computationally expensive [17, 23].

The implementation of the forward-on-reverse mode Hessian calculation requires the differentiation of the adjoint code. In the case of TAPENADE, this will necessitate the differentiation of the subroutines implementing `PUSH/POP` directives required for the reverse mode AD. Martinelli and Duvigneau [67] and Christakopoulos [17] have discussed these issues and have demonstrated a successful implementation of the forward-on-reverse method. The principal motivation behind favouring the forward-on-reverse mode Hessian algorithm is the fact that the computational cost is linear in the number of input independent variables for Hessian-vector product.

Martinelli et al. [66, 67] have explored the computational cost of Hessian calculation for various strategies using TAPENADE. A detailed computational cost analysis is presented for both the approaches. It is shown in [66] that the forward-on-reverse approach is cheaper than the forward-on-forward approach for Hessian-vector product calculations with large number of independent variables n. The exact threshold depends on the number of iterations required for the linear and adjoint solutions along with the overheads related to the AD implementation of the linear and adjoint calculations. The performance of the forward-on-forward approach is compared with forward-on-reverse for increasing number of input independent variables by using an artificial algebraic state function and objective function. Martinelli et al. [66] (section 4) have reported that the computational cost of the forward-on-reverse mode is independent of the number of independent variables n, while the computational cost of forward-on-forward approach is still linearly proportional on n. This computational cost analysis assumes calculation of the Hessian-vector product (directional derivative) instead of the calculation of the entire Hessian matrix. However, the computational cost for the calculation of the entire Hessian matrix is presented in section 5.6.

[24] The mathematical formulation presented in this chapter only considers Hessian calculation for a single objective function (m=1)

Some work on parallel calculation of Hessian in ADIFOR using OpenMP has been reported by Martin Bücker et al. [9]. This work reports use of ADIFOR to differentiate the entire code in the forward-on-forward mode. They have also taken the advantage of the fact that the $\frac{n\times(n+1)}{2}$ Hessian calculations can be independently performed. A module is developed to automatically insert OpenMP directives to parallelise the Hessian calculation. However, this black-box application of AD is not computationally efficient as still $\mathcal{O}(n^2)$ iterative solutions are required.

Taylor et al. [85, 88] were the first to investigate Hessian calculation using various algorithms. The computational cost and the implementation using AD is presented in their work. The publication does not explore the implementation details for a generic fluid mechanics code. This thesis demonstrates the implementation method in complete detail with the help of a 2D airfoil code. The order of computational cost, efficient AD implementation and the mathematical background behind the idea are presented.

Since the publication [32] of the work presented here on Hessian calculation, a considerable amount of work [17, 76, 84, 99] has been done on Hessian calculation using AD.

The next section presents the basic formulation for the algorithm for Hessian calculation.

5.2 BASIC FORMULATION

Following the approach as outlined in [85, 88], we want to calculate the Hessian of an objective function $\mathcal{J}(\alpha) \doteq J(\alpha, Z(\alpha))$, $\mathcal{J} \in \mathbb{R}^m$ with respect to independent variables $\alpha \in \mathbb{R}^n$ such that $Z(\alpha) \in \mathbb{R}^p$ satisfies the state equation

$$R(\alpha,Z) = 0. \tag{5.1}$$

is to be calculated. Henceforth Z will be referred to as the intermediate variable. To take an example from the fluid mechanic code,

$$Z = [X,U],$$

where X are the grid variables which change according to the design variables, and U are referred to as the state variables which are obtained by solving the state equation, e.g. the discretised Navier-Stokes equations.

$R(\alpha, Z)$ refers to such state equations augmented by the grid generation equations. Typically equation (5.1) is a set of nonlinear equations and is solved using some fixed point iteration method which is computationally expensive.

For simplicity consider that $\mathcal{J}$ is uni-dimensional, *i.e.* $m = 1$. The derivative of $\mathcal{J}$ with respect to one individual component of α is given by

$$\frac{\partial \mathcal{J}}{\partial \alpha_i} = \frac{\partial J}{\partial \alpha_i} + \frac{\partial J}{\partial Z}\,\frac{\partial Z}{\partial \alpha_i}. \tag{5.2}$$

Differentiating equation (5.2) again gives us

$$\begin{aligned}\frac{\partial^2 \mathcal{J}}{\partial \alpha_i \partial \alpha_j} = \frac{\partial^2 J}{\partial \alpha_i\, \partial \alpha_j} + \frac{\partial^2 J}{\partial \alpha_i \partial Z}\left(\frac{\partial Z}{\partial \alpha_j}\right) + \frac{\partial^2 J}{\partial \alpha_j \partial Z}\left(\frac{\partial Z}{\partial \alpha_i}\right) \\ + \frac{\partial^2 J}{\partial Z^2}\left(\frac{\partial Z}{\partial \alpha_i}\,\frac{\partial Z}{\partial \alpha_j}\right) + \frac{\partial J}{\partial Z}\left(\frac{\partial^2 Z}{\partial \alpha_i\, \partial \alpha_j}\right)\end{aligned} \tag{5.3}$$

The above equation tells us that the calculation of the Hessian requires the linear sensitivities of Z. Also the last term on the right hand side of the equation requires the second order sensitivity of Z. The computational cost of this calculation is

- One baseline nonlinear solution Z,
- $\mathcal{O}(n)$ linear solutions of $\frac{\partial Z}{\partial \alpha_i}$,
- $\mathcal{O}(n^2)$ second derivatives of $\frac{\partial^2 Z}{\partial \alpha_i \alpha_j}$, and
- $\mathcal{O}(n^2)$ evaluations of the right-hand side of equation (5.3).

If the intermediate variables Z are an explicit function of the design variables, then this is a simple task using AD. The original routines can be differentiated twice in the forward mode to propagate the second order sensitivities. The calculation of the linear and second order sensitivities of the intermediate variables will be computationally inexpensive.

However, for fluid mechanics problems, the intermediate variables are an implicit function of the design variables and they require an iterative procedure for the solution. This makes the calculation of the linear and second order sensitivities of the intermediate variables computationally expensive. A different formulation is presented henceforth to reduce this computational cost.

Equation (5.3) can be rearranged as

$$\frac{\partial^2 \mathcal{J}}{\partial \alpha_i \partial \alpha_j} = \frac{\partial J}{\partial Z}\,\frac{\partial^2 Z}{\partial \alpha_i \partial \alpha_j} + D^2_{i,j} J, \tag{5.4}$$

where

$$D^2_{i,j}J = \frac{\partial^2 J}{\partial\alpha_i\ \partial\alpha_j} + \frac{\partial^2 J}{\partial\alpha_i \partial Z}\left(\frac{\partial Z}{\partial\alpha_j}\right) + \frac{\partial^2 J}{\partial\alpha_j \partial Z}\left(\frac{\partial Z}{\partial\alpha_i}\right) + \frac{\partial^2 J}{\partial Z^2}\left(\frac{\partial Z}{\partial\alpha_i}\ \frac{\partial Z}{\partial\alpha_j}\right). \tag{5.5}$$

Differentiating the state equation (5.1) gives

$$\frac{\partial R}{\partial\alpha_i} + \frac{\partial R}{\partial Z}\ \frac{\partial Z}{\partial\alpha_i} = 0. \tag{5.6}$$

Differentiating again,

$$\frac{\partial R}{\partial Z}\ \frac{\partial^2 Z}{\partial\alpha_i\partial\alpha_j} + D^2_{i,j}R = 0, \tag{5.7}$$

where $D^2_{i,j}R$ is similarly defined as $D^2_{i,j}J$ in equation (5.5). Substituting for $\frac{\partial^2 Z}{\partial\alpha_i\partial\alpha_j}$ in equation (5.4) from equation (5.7),

$$\begin{aligned}\frac{\partial^2\mathcal{J}}{\partial\alpha_i\partial\alpha_j} &= -\frac{\partial J}{\partial Z}\left(\frac{\partial R}{\partial Z}\right)^{-1} D^2_{i,j}R + D^2_{i,j}J \\ &= D^2_{i,j}J - v^T D^2_{i,j}R.\end{aligned} \tag{5.8}$$

Here v is the adjoint solution associated with the objective function J defined by the adjoint equation

$$\left(\frac{\partial R}{\partial Z}\right)^T v - \left(\frac{\partial J}{\partial Z}\right)^T = 0. \tag{5.9}$$

Note this is the same adjoint solution defined in the previous chapter. Equation (5.8) is used to calculate the complete Hessian. The entire formulation presented here is also valid for a multi-dimensional objective function. Various methods for calculation of the adjoint solutions are available [37].

Now let us look at the computational cost for calculating the entire Hessian. The solution of the state equation (5.1) gives the baseline value of the intermediate variables Z^0 corresponding to the values of the design variables α^0 where the Hessian is to be calculated. It is clear from equations (5.5, 5.8) that a single adjoint solution $v(Z^0)$ corresponding to J is also required along with n linear solutions $\frac{\partial Z}{\partial\alpha_i}(Z^0)$ corresponding to the n independent variables α_i. If these solutions are available then only an evaluation of $D^2_{i,j}J$ and $D^2_{i,j}R$ is required for each element of the Hessian, *i.e.* these functions are evaluated for each pair of α_i and α_j.

Hence the total computational cost of the entire Hessian calculation is:

- Single baseline nonlinear solution Z^0,
- $\mathcal{O}(n)$ linear flow $\frac{\partial Z}{\partial \alpha_i}(Z^0)$,
- Single adjoint solution $v(Z^0)$,
- $\mathcal{O}(n^2)$ evaluations of $D^2_{i,j}J$, $D^2_{i,j}R$, and
- $\mathcal{O}(n^2)$ dot products for $v^T D^2_{i,j}R$.

As a single evaluation is much cheaper than an iterative solution, the cost of the last two items is negligible and hence the computational cost of calculating the entire Hessian is approximately $\mathcal{O}(n)$.

The entire argument presented above for a single dimensional objective function J also holds true for the general case of m dimensions. The net computational cost would be of the order $\mathcal{O}(n+m)$ because of the $\mathcal{O}(m)$ adjoint solutions corresponding to all the objective functions.

The basic concept behind the Hessian calculation for a general case has been explained in this section. But the mathematical formulation used in this implementation is slightly different and is presented in the next section.

5.3 FORMULATION FOR FLUID MECHANICS

We are interested in the Hessian of an objective function

$$\mathcal{J}(\alpha) = J(\alpha,X(\alpha),U(\alpha)), \quad \mathcal{J} \in \mathbb{R}^m$$

with respect to the independent variables $\alpha \in \mathbb{R}^n$ such that $X(\alpha)$ and $U(\alpha)$ satisfy the state equation

$$R(\alpha,X(\alpha),U(\alpha)) = 0. \tag{5.10}$$

α are the design variables and the Hessian of the objective function with respect to these design variables is to be calculated.

X are the grid variables which change according to the design variables. U are referred to as the flow variables.

For simplicity consider that $\mathcal{J}$ is uni-dimensional, i.e. $m = 1$. The derivative of $\mathcal{J}$ with respect to one individual component of α is given by

$$\frac{\partial \mathcal{J}}{\partial \alpha_i} = \frac{\partial J}{\partial \alpha_i} + \frac{\partial J}{\partial X}\ \frac{\partial X}{\partial \alpha_i} + \frac{\partial J}{\partial U}\ \frac{\partial U}{\partial \alpha_i}. \tag{5.11}$$

Differentiating equation (5.11),

$$\frac{\partial^2 \mathcal{J}}{\partial \alpha_i \partial \alpha_j} = \frac{\partial J}{\partial U}\ \frac{\partial^2 U}{\partial \alpha_i \partial \alpha_j} + D^2_{i,j} J \tag{5.12}$$

where,

$$\begin{aligned} D^2_{i,j} J = & \frac{\partial^2 J}{\partial \alpha_i\ \partial \alpha_j} + \frac{\partial J}{\partial X}\ \frac{\partial^2 X}{\partial \alpha_i \partial \alpha_j} + \\ & \frac{\partial^2 J}{\partial \alpha_i \partial U}\left(\frac{\partial U}{\partial \alpha_j}\right) + \frac{\partial^2 J}{\partial \alpha_j \partial U}\left(\frac{\partial U}{\partial \alpha_i}\right) + \frac{\partial^2 J}{\partial U^2}\left(\frac{\partial U}{\partial \alpha_i}\ \frac{\partial U}{\partial \alpha_j}\right) + \\ & \frac{\partial^2 J}{\partial \alpha_i \partial X}\left(\frac{\partial X}{\partial \alpha_j}\right) + \frac{\partial^2 J}{\partial \alpha_j \partial X}\left(\frac{\partial X}{\partial \alpha_i}\right) + \frac{\partial^2 J}{\partial X^2}\left(\frac{\partial X}{\partial \alpha_i}\ \frac{\partial X}{\partial \alpha_j}\right). \end{aligned} \tag{5.13}$$

Similarly, the entire process can be repeated for the state equation (5.10) to give

$$\frac{\partial R}{\partial U}\ \frac{\partial^2 U}{\partial \alpha_i \partial \alpha_j} + D^2_{i,j} R = 0, \tag{5.14}$$

where, $D^2_{i,j} R$ is similarly defined as $D^2_{i,j} J$ in equation (5.13). Substituting for $\frac{\partial^2 U}{\partial \alpha_i \partial \alpha_j}$ in equation (5.12) from equation (5.14),

$$\begin{aligned} \frac{\partial^2 \mathcal{J}}{\partial \alpha_i \partial \alpha_j} &= -\frac{\partial J}{\partial U}\left(\frac{\partial R}{\partial U}\right)^{-1} D^2_{i,j} R + D^2_{i,j} J \\ &= D^2_{i,j} J - v^T D^2_{i,j} R. \end{aligned} \tag{5.15}$$

where v is the adjoint solution associated with the objective function J defined by the adjoint equation

$$\left(\frac{\partial R}{\partial U}\right)^T v - \left(\frac{\partial J}{\partial U}\right)^T = 0. \tag{5.16}$$

Equation (5.15) is used to calculate the complete Hessian. It should be noted here that only the second derivative of flow variables is replaced here by the flow adjoint solution. Typically, the grid generation process is computationally much cheaper than the iterative flow solutions. Also, because of the involvement of the CAD packages in the grid generation process, it is difficult to develop adjoint codes for the grid generation process. Hence in this implementation grid adjoint solutions are not used. The linear and second derivatives of the grid variables $X(\alpha)$ are calculated in the forward mode and then passed on to the routines in the flow solver.

However if the grid adjoint solution capability exists or is desirable then the earlier more generic formulation can be used. The entire formulation presented here is also valid for a multi-dimensional objective function. The order of the computational cost remains the same as the earlier generic formulation. The next section discusses some of the related implementation issues.

5.4 IMPLEMENTATION

The process of an efficient implementation of the Hessian calculation is a natural extension of the methods discussed in the previous chapter for the linear and adjoint codes. All the work on the proper structuring of the original nonlinear code is also required and proves useful for the Hessian implementation.

The entire nonlinear code has to be written in a modular fashion with all the nonlinear bits separated from the time integration loop. Each of these functions containing nonlinear bits are then double differentiated in the forward mode using the AD software. For example the original nonlinear wall flux calculation subroutine is

```
flux_wall(x1, x2, q, res),
```

where, `x1` and `x2` are the nodes defining the wall edge, `q` is the state vector of the interior cell and `res` is the residue vector. This subroutine is differentiated in forward mode using an AD software with `x1`, `x2` and `q` as the independent variables and `res` as the dependent variable. The differentiated subroutine is

```
flux_wall_d( x1,  x1d,
             x2,  x2d,
              q,   qd,
            res, resd),
```

where all the variables appended with d are the perturbation variables. This subroutine is again differentiated in the forward mode with `x1`, `x1d`, `x2`, `x2d`, `q` and `qd` as the independent variables while `res` and `resd` are the dependent variables.

```
flux_wall_d2( x1,  x1d0,  x1d,  x1dd,
              x2,  x2d0,  x2d,  x2dd,
               q,   qd0,   qd,   qdd,
             res, resd0, resd, resdd)
```

Effectively each of the original variable gets linearised twice along with the second order perturbation in the variables appended by `dd`. These variables correspond to the complete second order derivative of the original variable, *i.e.* `resdd` is the complete second order derivative given by an expression similar to equations (5.12) and (5.13) if `x1d`, `x2d`, `qd` are initialised with perturbations with respect to α_i and `x1d0`, `x2d0`, `qd0` are initialised with perturbations with respect to α_j. By setting `qdd` $= 0$, the $D^2_{i,j}$ operator can be evaluated applied to `res`. These perturbations have to be carried forward throughout the solver code in a similar fashion to calculate the complete $D^2_{i,j}R$ over the entire grid. $D^2_{i,j}J$ is evaluated in a similar fashion.

5.5 VALIDATION CHECKS

Although AD by definition removes all the user intervention and associated errors, it is always good practice to introduce validation checks to ensure the correctness of the implementation. Validation checks for Hessian calculation are developed on the similar lines as discussed in the previous chapter for the linear and adjoint code development [38]. Unfortunately, it is not so straightforward as the linear and adjoint code development. Still some simple checks can be introduced.

Given fully converged nonlinear U and linear $\frac{\partial U}{\partial \alpha_i}$ solutions the following equations should be satisfied in the entire domain to machine precision

$$R(X,U) = 0, \text{ and } \frac{\partial R}{\partial X}\frac{\partial X}{\partial \alpha_i} + \frac{\partial R}{\partial U}\frac{\partial U}{\partial \alpha_i} = 0.$$

These checks ensure that the converged nonlinear and linear solutions are being correctly introduced in the Hessian code. Also, it is desirable to ensure that the adjoint solution is consistent with the linear solutions using the checks discussed previously.

Finally, it should be tested that the calculated Hessian is symmetric, *i.e.*

$$\frac{\partial^2 \mathcal{J}}{\partial \alpha_i \partial \alpha_j} = \frac{\partial^2 \mathcal{J}}{\partial \alpha_j \partial \alpha_i}.$$

This identity should be satisfied to machine precision. This calculation would require n^2 evaluations of the $D^2_{i,j}J$ and $D^2_{i,j}R$ functions. Once this has been validated then only the upper diagonal Hessian matrix can be calculated requiring $\frac{n(n+1)}{2}$ evaluations.

However the final validation has to be the comparison with finite difference results. These are illustrated in the next section.

5.6 COMPUTATIONAL COST

This section compares the computational cost of the forward-on-forward approach (using adjoint solution as presented in this chapter) and the other popular alternative, forward-on-reverse method for the calculation of the entire Hessian[25]. Calculation of Hessian vector products (as required during the optimisation algorithms) is computationally cheaper than the calculation of the entire Hessian. This section only provides the comparison for the calculation of the entire Hessian.

Following the convention used in section 2.4.3, let C be the cost of one baseline nonlinear solution, $\gamma_a\, C$ be the cost of one adjoint solution and $\gamma_l\, C$ be the cost of one linear solution. γ_{al} is the computational slowdown for the forward differentiated adjoint code while γ_{ll} is the computational slowdown for the forward differentiation forward mode code. A comparison between the computational cost for entire Hessian calculation is presented for forward-on-forward and forward-on-reverse mode in Table 5.1 for m objective functions and n independent input variables.

Method	Computational cost
Forward-on-Forward	$(1 + m\,\gamma_a + n\,\gamma_l)C + m\,(\frac{n(n-1)}{2}\gamma_{al})\,c$
Forward-on-Reverse	$(1 + m\,\gamma_a + n\,\gamma_l + mn\,\gamma_{al})C + m\,(n\gamma_{ll})\,c$

Table 5.1 Comparison of the computational cost for the calculation of the entire Hessian matrix using two Hessian algorithms for m objective functions with respective to n independent variables

Both the approaches require calculation of the baseline nonlinear solution, n linear solutions and m adjoint solutions. However, the forward-on-reverse mode requires the iterative solution for $m \times n$ forward differentiated adjoint variables. As $C \gg c$ and assuming few input independent variables, if the second term is neglected, then the computational cost of forward-on-forward approach is $\mathcal{O}(m + n)$ while the forward-on-reverse approach is $\mathcal{O}(m \times n)$.

On the other hand, for large number of input independent variables n (where the second term cannot be neglected), the computational cost of the forward-on-forward

[25] A detailed mathematical formulation for the forward-on-reverse method of Hessian calculation is given in [17, 66]

approach is $\mathcal{O}(n^2)$, whereas forward-on-reverse approach still remains $\mathcal{O}(m \times n)$. It should be noted here that for a single objective function $m = 1$, forward-on-reverse approach is only $\mathcal{O}(n)$.

Please refer to [66] for an analysis of the computational cost of Hessian-vector product calculation using forward-on-forward and forward-on-reverse approaches.

5.7 AIRFOIL CODE

The airfoil code used to demonstrate the ideas of linear and adjoint code development using AD is extended to calculate the Hessian. More details on the nonlinear code can be found in Appendix A.

Three modes of artificial perturbation are introduced to the baseline geometry. Figure B.III shows the three modes of perturbation namely: thickness, twist (angle-of-attack) and leading edge shape. Please see Appendix B for the exact expressions for the three modes of perturbation.

Hessian Element	Finite Difference	Direct
1 - 1	$-\mathbf{3.11120}3625702499E-07$	$-\mathbf{3.11120}3862791910E-07$
1 - 2	$-\mathbf{2.09760}0811599999E-06$	$-\mathbf{2.09760}0748629300E-06$
1 - 3	$-\mathbf{9.95922}301885120E-07$	$-\mathbf{9.95922}312828186E-07$
2 - 1	$-\mathbf{2.09760074}7610895E-06$	$-\mathbf{2.09760074}8629318E-06$
2 - 2	$-\mathbf{2.15968742}3428786E-04$	$-\mathbf{2.15968742}4802269E-04$
2 - 3	$-\mathbf{1.74653785}7481162E-04$	$-\mathbf{1.74653785}9860203E-04$
3 - 1	$-\mathbf{9.95922}2904915369E-07$	$-\mathbf{9.95922}3212828262E-07$
3 - 2	$-\mathbf{1.74653781}1210817E-04$	$-\mathbf{1.74653785}9860204E-04$
3 - 3	$-\mathbf{1.97093703}6569875E-05$	$-\mathbf{1.97093703}4187627E-05$

Table 5.2 Comparison between direct Hessian calculation and finite difference calculation for Lift with respect to the three modes (thickness, twist and leading edge) of airfoil perturbation. (Bold digits are matching digits)

A `Makefile` is written to calculate all the relevant linear, nonlinear and adjoint versions of the code using TAPENADE. A separate program `air_hes` is written which calls all the appropriate double differentiated subroutines to calculate the $D^2_{i,j}R$ and $D^2_{i,j}J$ functions, and finally the entire Hessian with respect to these three modes. Despite the fact that the Hessian is symmetric, all nine components are

calculated. The Hessian thus calculated is compared with the central finite difference approximation given by

$$\frac{\partial^2 \mathcal{J}}{\partial \alpha_i \partial \alpha_j}(\alpha^0) \approx \frac{1}{2\Delta\alpha_j}\left(\frac{\partial \mathcal{J}}{\partial \alpha_i}(\alpha^0 + \Delta\alpha_j) - \frac{\partial \mathcal{J}}{\partial \alpha_i}(\alpha^0 - \Delta\alpha_j)\right). \tag{5.17}$$

This expression is used instead of the second order finite difference of the nonlinear solution to minimise the sensitivity to the step-size. Also, the linear perturbations calculated using a linear solver are accurate to machine precision.

Table 5.2 shows the comparison between the Hessian calculated directly and the finite difference calculations using appropriate stepsize. Good agreement between these two is the final verification of the correctness of the Hessian code implementation.

Solver	CPU time (sec)	Ratio
Nonlinear	75.6	1.0
Hessian	173.5	2.3

Table 5.3 Comparison of the compuatational cost between 1000 evaluations of the objective function using the original nonlinear code and the 1000 evaluations of the Hessian code (that reads the nonlinear, linear and adjoint solutions and propagates the second order perturbations)

Table 5.3 shows the comparison of the computational cost between the original nonlinear solver and the Hessian driver module. The nonlinear solver reads the input data, performs one fixed point iteration, calculates the objective function (lift for the airfoil code) and exits. This is roughly equivalent to the c as defined in section 5.6. Hessian driver module reads the nonlinear, linear and adjoint solutions and calculates an element of the Hessian matrix. Both the codes are run for 1000 times using a shell script and the computational cost is reported. The specifics of the hardware on which these tests were performed are given in section 4.7.1.

It is observed that one Hessian evaluation is 2.3 times more expensive than one evaluation of the objective function. The slowdown (γ_{ll} as defined in section 5.6) factor is on the lower side since the underlying nonlinear airfoil code is not complex. For the production grade fluid mechanic codes with viscous fluxes, larger mesh size and complex datatypes this value is expected to rise. Martinelli et al. [66] have

reported a value of 4.9 for a three dimensional Euler solver. Since the fixed point iteration scheme does not play a role in the calculation of the residuals, adding multigrid acceleration, preconditioning and higher order time integration algorithms should not affect the computational cost of one Hessian evaluation.

5.8 EXTRAPOLATION

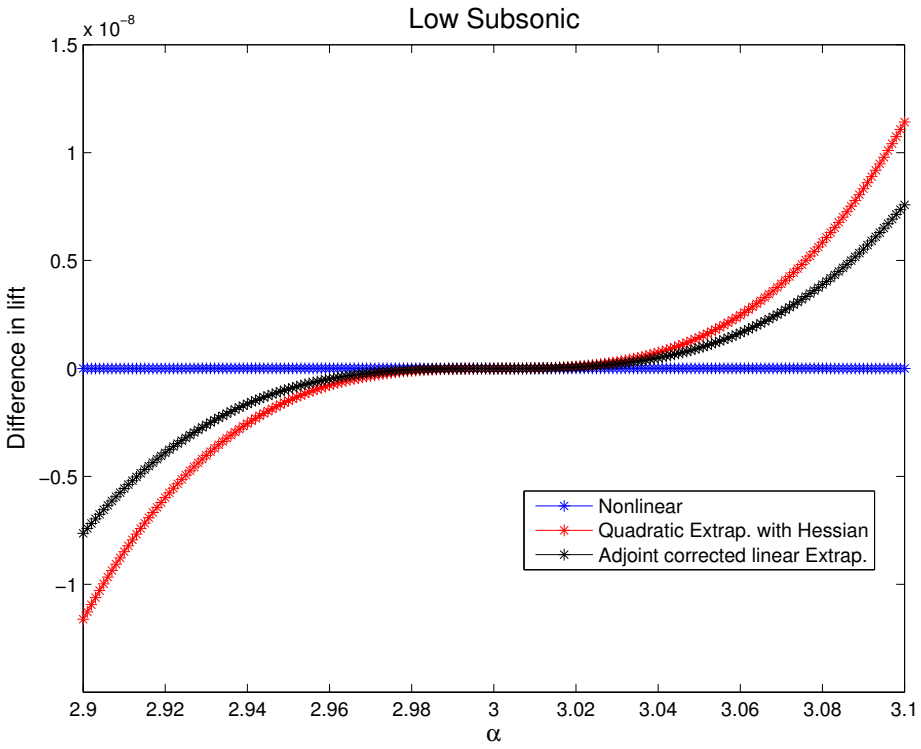

Figure 5.1 Comparison between quadratic and adjoint corrected linear extrapolation for low subsonic case (Mach = 0.4 and AOA = 3^o)

One of the major applications of the Hessian thus obtained is in extrapolation. It is interesting to compare the performance of the quadratic extrapolation using linear and Hessian solutions with the linear extrapolation using adjoint correction [40]. For the sake of completeness, the two expressions are given here. The quadratic extrapolation is given by

$$\mathcal{J}_\alpha = \mathcal{J}_{\alpha_0} + \frac{\partial \mathcal{J}}{\partial \alpha}(\alpha - \alpha_0) + \frac{1}{2}\frac{\partial^2 \mathcal{J}}{\partial \alpha^2}(\alpha - \alpha_0)^2,$$

while adjoint corrected linear extrapolation is

$$\mathcal{J}_{\alpha} = \mathcal{J}_{\alpha_0} + \frac{\partial \mathcal{J}}{\partial \alpha}(\alpha - \alpha_0) - v(\alpha_0)^T R\left(X(\alpha), U(\alpha_0) + \frac{\partial U}{\partial \alpha}(\alpha - \alpha_0)\right).$$

The adjoint corrected linear extrapolation thus obtained has a third order leading error (i.e. $\mathcal{O}\left((\Delta\alpha)^3\right)$). Extrapolation about a base angle-of-attack (AOA) is carried out using the second mode of perturbation. A set of nonlinear simulations converged to full machine precision is carried out by varying α from 2.9^o to 3.1^o in the steps of 0.001^o. The nonlinear lift thus obtained is compared with the two methods of extrapolation described above. A polynomial of third degree is fitted to the nonlinear lift calculations in the least square sense with angle of attack as the independent variable. Figure 5.1 plots the difference between these extrapolations and the cubic fit of the nonlinear lift calculations for the low subsonic range (Mach = 0.4 and AOA = 3^o) against the angle-of-attack. This difference is plotted instead of the actual lift plots as to normalise the data and focus on the variation in the predictions. Both the approaches look equally accurate in this case. As the perturbation in α is small,there is relatively small error with respect to the cubic fit of the nonlinear lift.

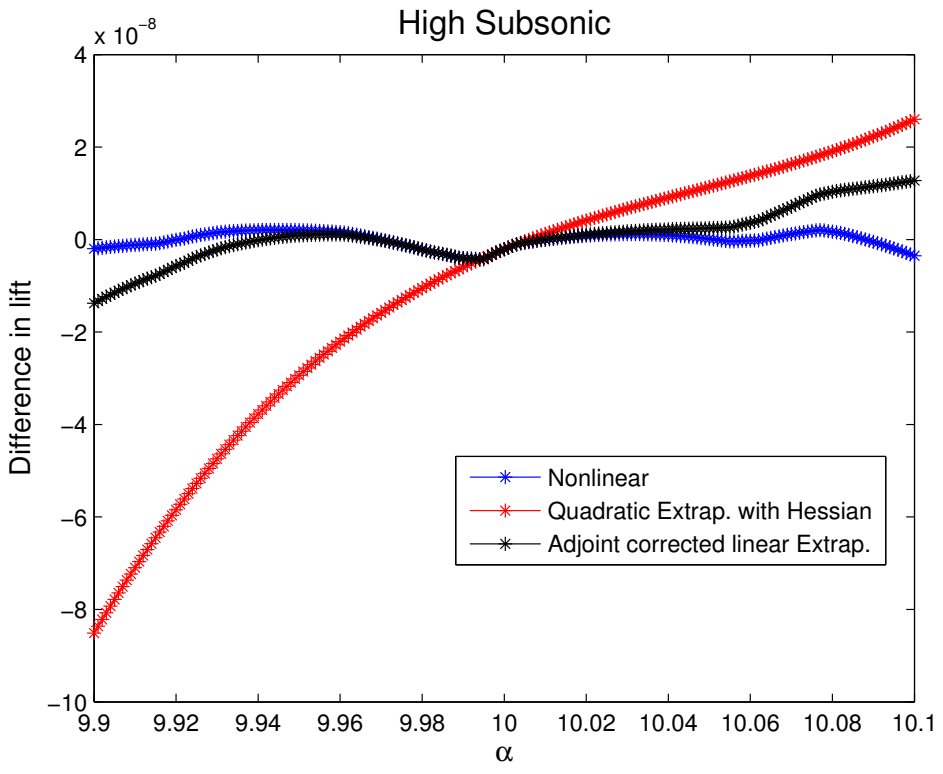

Figure 5.2 Comparison between quadratic and adjoint corrected linear extrapolation for high subsonic case (Mach = 0.65 and AOA = 10^o)

Similarly, Figure 5.2 shows comparison for a higher subsonic test case with Mach =

0.65 and AOA = 10^o. The behaviour of the nonlinear lift is not smooth with some pronounced kinks. Further investigation of this curious behaviour revealed that these kinks are arising because of the fundamental non differentiability of the underlying function. It was found that the numerical instability arose because of a key quantity Adt (a variable storing the ratio of the area of a cell and the local time-step) in the nonlinear calculations (see equation (A.3) in Appendix A).

$$Adt = \frac{\sum_i |u\ dy_i - v\ dx_i| + c\ \sqrt{dx_i^2 + dy_i^2}}{CFL}, \tag{5.18}$$

Here, u and v are the velocity components in x and y directions respectively, while dx and dy are the projections of the length of an edge. CFL is the Courant-Friedrichs-Lewy number. The summation is over the four edges forming a cell. Since the calculation of Adt involves a term with the absolute function, this variable is C^0 continuous, but is not C^1 continuous.

To elaborate on the problem more clearly, consider a generic function

$$g(x) = |f(x)|$$

with $f(x)$ being infinitely differentiable.

Now $g(x)$ is continuous on the entire real line with a discontinuity in $g'(x)$ at all x with $f(x) = 0$ and $f'(x) \neq 0$. Investigation of a relatively large kink in Figure 5.2 at $\alpha = 9.995^o$ revealed that $|u\ dy - v\ dx|$ changes its sign in multiple cells between $\alpha = 9.994$ and $\alpha = 9.995$. These small perturbations accumulated over multiple cells account for the kink in the nonlinear lift value. If the Hessian is calculated about such a sensitive point then it introduces error.

It should be noted here that these errors are extremely small and in comparison to the convergence and discretisation errors in the real-life applications, these are not significant. However, such errors accumulated from multiple sources for highly complex codes may create significant errors. The current example of a two dimensional inviscid solver is too simplistic to assess the true extent of errors that might be introduced.

Also it should be noted that the adjoint corrected linear extrapolation performs better than the quadratic extrapolation in this case. It closely follows the trend of the underlying nonlinear solution.

To confirm that this is the only source of error for the non-smooth behaviour, the time-step calculation is modified slightly to avoid the sign change. The time-step calculation as given in equation (5.18) is modified to

$$Adt = \frac{1}{CFL} \sum_i c \; ds_i \left(\sqrt{(m_x \frac{dy_i}{ds_i} - m_y \frac{dx_i}{ds_i})^2 + \epsilon^2} \; + 1 \right),$$

where $ds_i = \sqrt{dx_i^2 + dy_i^2}$, $m_x = \frac{u}{c}$, $m_y = \frac{v}{c}$ and $\epsilon = 0.1$. ϵ is chosen to ensure that the derivative of Adt is always defined. The entire simulations are carried out again and the results were presented in Figure 5.3. The smooth behaviour of the nonlinear lift confirms the hypothesis.

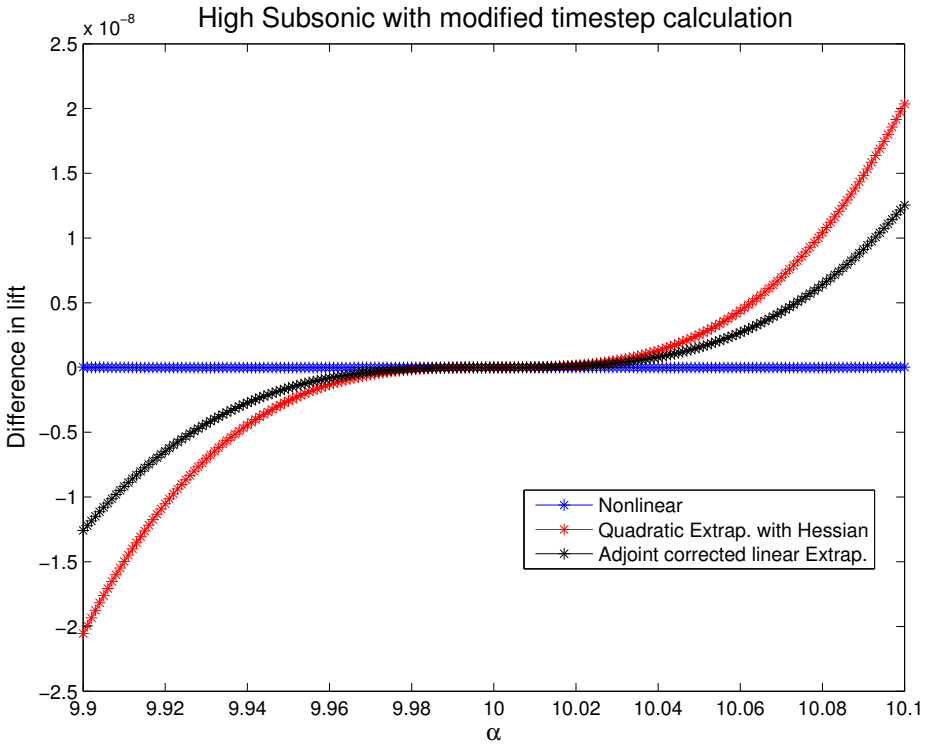

Figure 5.3 Comparison between quadratic and adjoint corrected linear extrapolation for high subsonic case (Mach = 0.65 and AOA = 10^o) with the modified time-step calculation

5.9 CONCLUSION

Black-box use of AD on fluid mechanics codes for Hessian calculation is computationally inefficient. An alternative computationally cheaper formulation for Hessian calculation has been described. Successful use of AD has been demonstrated for generating all the necessary double differentiated routines. A `Makefile` can be generated to automatically keep in-sync with changes in the original nonlinear code. A set of checks have been introduced which at least point out any obvious inconsistencies in the nonlinear, linear and adjoint solutions used during the Hessian calculation.

A conscious effort is required while developing any nonlinear solvers to avoid inherently non-differentiable component functions. It will be increasingly difficult to track down these problems in complex codes. Adjoint corrected linear extrapolation seems to perform at least as well as the quadratic extrapolation. In non-smooth regions, adjoint corrected linear extrapolation closely follows the nonlinear trend in contrast to the quadratic extrapolation using the Hessian which may introduce relatively large errors. This is because the quadratic extrapolation requires more regularity than the adjoint corrected linear extrapolation.

6 IMC FOR AIRFOIL APPLICATION

This chapter presents the implementation details of the IMC method (as outlined in chapter two of this dissertation) for fluid mechanic applications. A simple two dimensional airfoil code (see Appendix A) is used to demonstrate the concept. The idea of reduced order modelling (see chapter three) using PCA is used for the extension of IMC for the present fluid mechanic application. Finally, comparison between the nonlinear Monte Carlo simulations and the various IMC methods is presented for an artificially generated geometric uncertainty for the airfoil code.

Conceptually, the formulation of IMC method presented in section 2.4 can be extended in a straightforward manner to systems of discretised PDEs representing Euler equations. However, the implementation details get more involved. For the sake of completeness, the entire formulation for IMC as extended to fluid mechanics problem is presented here.

6.1 MATHEMATICAL FORMULATION

Given input design variable $\alpha \in \mathbb{R}^n$, let

$$\mathcal{J}(\alpha) \doteq J(X(\alpha), U(\alpha), \alpha), \quad \mathcal{J} :\rightarrow \mathbb{R}$$

be the objective function such that $X(\alpha) \in \mathbb{R}^p$ and $U(\alpha) \in \mathbb{R}^p$ satisfy the state equation

$$R(X(\alpha), U(\alpha), \alpha) = 0. \tag{6.1}$$

Here, X are the mesh coordinates and U is the vector of flow solutions at the mesh points. There are two formulations possible here. To follow the mathematical formulation presented in section 2.4, we can define

$$Z \doteq [X, U],$$

and the rest of the formulation extends naturally. However, this will require the solution of two adjoint equations, namely the flow adjoint equation

$$\left(\frac{\partial R}{\partial U}\right)^T v = \left(\frac{\partial J}{\partial U}\right)^T, \tag{6.2}$$

where $v \in \mathbb{R}^p$ is the flow adjoint variable, and the adjoint equation corresponding to the mesh perturbations,

$$\left(\frac{\partial R}{\partial X}\right)^T v_x = \left(\frac{\partial J}{\partial X}\right)^T. \tag{6.3}$$

If the solution of adjoint equations (6.2) and (6.3) is available, then the IMC formulation can be used as it is.

A different formulation is presented henceforth which does not require the solution of mesh adjoint equations. Here we observe that mesh generation algorithms are typically orders of magnitude cheaper than the flow solution algorithms. The calculation of exact mesh coordinates at each sampling point is not prohibitively expensive. Hence, the exact mesh coordinates can be used at each sampling point of the IMC algorithm thereby obviating the need for its effect to be approximated by Taylor series expansion.

The first order Taylor expansion of equation (6.1) around U^* is

$$R(X,U,\alpha) \approx R(X,U^*,\alpha) - \frac{\partial R}{\partial U}(U^* - U) + \mathcal{O}\left(\|U^* - U\|^2\right), \tag{6.4}$$

with all the higher order error terms being ignored and a leading error of second order. Similarly the first order Taylor expansion of the objective function around U^* is

$$J(X,U,\alpha) \approx J(X,U^*,\alpha) - \frac{\partial J}{\partial U}(U^* - U) + \mathcal{O}\left(\|U^* - U\|^2\right).$$

Using the adjoint equation (6.2),

$$J(X,U,\alpha) \approx J(X,U^*,\alpha) - v^T \frac{\partial R}{\partial U}(U^* - U) + \mathcal{O}\left(\|U - U^*\|^2\right).$$

Using equation (6.4) this can be written as,

$$J(X,U,\alpha) \approx J(X,U^*,\alpha) - v^T R(X,U^*,\alpha) + \mathcal{O}\left(\|U^* - U\|^2\right).$$

The order of the error still remains the same. Finally if an approximate solution v^* is used instead of the exact adjoint solution v, then

$$J(X,U,\alpha) \approx J(X,U^*,\alpha) - v^{*T} R(X,U^*,\alpha) + \mathcal{O}\left(\|U^* - U\|^2, \|U^* - U\| \, \|v^* - v\|\right) \tag{6.5}$$

Hereafter, the three IMC approaches as outlined in subsection 2.4.2 can be used in fluid mechanic applications. IMC–1 approach only requires the adjoint solution while

the IMC – 2 approach requires linear as well as the adjoint solution. However, the IMC – 3 approach also requires the linear sensitivities of the adjoint solution with respect to the independent variables $\frac{dv}{d\alpha_i}$. The next section briefly discusses issues related to the calculation of sensitivities of the adjoint variable with respect to the independent variables for fluid mechanic solvers using AD.

6.1.1 *Sensitivity of the adjoint variables*

The mathematical formulation and the implementation details (using AD) of the linear sensitivities of the adjoint variables are presented in [17, 66]. The linear sensitivities of the adjoint variables thus calculated are then used for the Hessian calculation using the forward-on-reverse approach. However, the same sensitivities can also be used for the implementation of the IMC – 3 approach.

Differentiating the adjoint equation (equation (4.7)) with respect to an independent variable α_i and following the formulation presented in Christakopoulos [17], it can be seen that $\frac{dv}{d\alpha_i}$ is a solution of

$$\frac{\partial R}{\partial U}^T \frac{dv}{d\alpha_i} = \hat{J}_u - \hat{R}_u, \tag{6.6}$$

where,

$$\hat{J}_u = \frac{\partial}{\partial \alpha_i}\left(\frac{\partial J}{\partial U}^T\right) + \frac{\partial}{\partial U}\left(\frac{\partial J}{\partial U}^T\right)\frac{\partial U}{\partial \alpha_i},$$

$$\hat{R}_u = \frac{\partial}{\partial \alpha_i}\left(\frac{\partial R}{\partial U}^T\right) + \frac{\partial}{\partial U}\left(\frac{\partial R}{\partial U}^T\right)\frac{\partial U}{\partial \alpha_i}.$$

The left hand side of equation (6.6) is same as the left hand side of the adjoint equation. Hence, for a given iterative solution procedure, equation (6.6) and the adjoint equation will have the same rate of convergence.

However, the construction of the right hand side is not straightforward. For example, the construction of a subroutine to calculate $\frac{\partial}{\partial U}\left(\frac{\partial J}{\partial U}^T\right)$ requires forward mode differentiation of a reverse mode differentiated code. In case of TAPENADE, this requires the forward mode differentiation of the `PUSH/POP` functions introduced by TAPENADE in the reverse mode differentiated code. Martinelli et al. [66] and Christakopoulos [17] have reported the successful calculation of the sensitivities of the adjoint variables using this strategy. A computationally efficient implementation

can be obtained by precomputing and reusing various terms of the linearised adjoint equations outside the fixed point iteration loop.

The IMC – 3 approach is not implemented in the present work. The description of the issues related to the calculation of $\frac{dv}{d\alpha_i}$ are only presented for the sake of completeness. We now move to the implementation details for IMC – 1 and IMC – 2 methods which only require the linear and the adjoint solution.

6.2 IMPLEMENTATION

This section discusses the various implementation details and options for the two IMC methods (IMC – 1 and IMC – 2). Figure 6.1 shows the algorithm for the IMC – 2 approach to propagate geometric uncertainty through fluid mechanic solvers in general, and the airfoil code in particular. Though the flowchart shows a sequence of steps, it does not imply that this sequence needs to be followed. Moreover, many of the steps in the flowchart can be carried out in parallel. However, all the steps as outlined in this flowchart need to be carried out in one form or another. Given the baseline geometry of the airfoil x^0, a baseline volume mesh X^0 is generated. This is followed by the nonlinear flow solution $U(X^0)$ and the adjoint solution $v(X^0)$ corresponding to the objective function J. These are computationally expensive calculations since they require iterative methods for the solution.

As outlined in chapter 3 of this thesis, a PCA based reduced order model is derived for a set of geometry measurements $\hat{x}_i$, i = 1,...,n. This has the advantage that if the first n_r leading modes a_k of the PCA model are used, then only n_r linear solutions $\frac{dU}{da_k}(X^0, U(X^0))$ are required at the baseline. This dramatically reduces the number of linear flow calculations required for the IMC – 2 approach from mp to n_r without significant loss of accuracy. Garzon [30] has reported the use of only 5 leading eigenmodes to capture 99% of the variability from the actual measurement data for engine blades.

There are two possible choices after this step. Either mesh generation can be carried out for the actual measured airfoil geometries or a Monte Carlo sampling strategy can be used to obtain sample points of PCA based reduced order model for the IMC simulations. The first approach uses only the measurement data for the IMC simulations. This can result in suboptimal performance of the Monte Carlo simulation algorithms depending on the distribution of the measurement data in the range of interest.[26]

For the airfoil code, the original coefficients from the PCA analysis were used to perform Monte Carlo simulations.

However, an accelerated convergence rate for Monte Carlo simulations can be achieved by using various sampling strategies. For example, Wang [94] has demonstrated the use of sensitivity information obtained from the adjoint solution to accelerate the convergence rate of Monte Carlo simulations using importance sampling. The use of a Monte Carlo sampling strategy will have no effect on the leading order of error for the IMC simulations[27].

The next step is to generate volume mesh for all the M sample points. As mesh generation algorithms are typically computationally cheaper than the corresponding nonlinear flow solution, it is expected that this step will result in only modest computational expense. In the case of the airfoil code, the mesh generation uses a Joukowski transformation and is computationally inexpensive. However, if the mesh generation process is computationally expensive and the solution of the mesh adjoint equation (6.3) is available, then it can be included in the IMC algorithm. If the mesh adjoint solution is used, then the computational expense of M nonlinear mesh generation algorithms is replaced by a single iterative solution of the mesh adjoint equation. It is expected that the mesh generation algorithms will remain sufficiently computationally inexpensive as to make this option unattractive.

Finally, M IMC simulations are carried out using the baseline nonlinear equation, the baseline adjoint equation, n_r linear equations and M volume meshes. A separate module needs to be written to evaluate equation (6.5) at each sampling point. The implementation details for this module are presented in the next section.

This section outlined the algorithm for IMC – 2 simulations for the fluid mechanic applications. Implementation details for the IMC – 1 approach are simpler than this

[26] Suppose, 10000 blade measurements are available, but only 1000 measurements are used for the Monte Carlo simulations. Now these 1000 measurements will sample the distribution of measurement data (obtained from 10000 samples) more poorly than various sampling strategies (like stratified sampling). Hence Monte Carlo simulations using actual measurement data might be inferior to Monte Carlo using sampled data points.

[27] It should be noted here that the improvement in performance by using various sampling strategies is dampened by the fact that the evaluation of the objective function at each sampling point is computationally cheap in IMC methods. One can always increase the sample size to achieve desired accuracy without a significant computational penalty. Sampling strategies have a critical impact on nonlinear Monte Carlo simulations, since each evaluation of the objective function requires an expensive iterative solution.

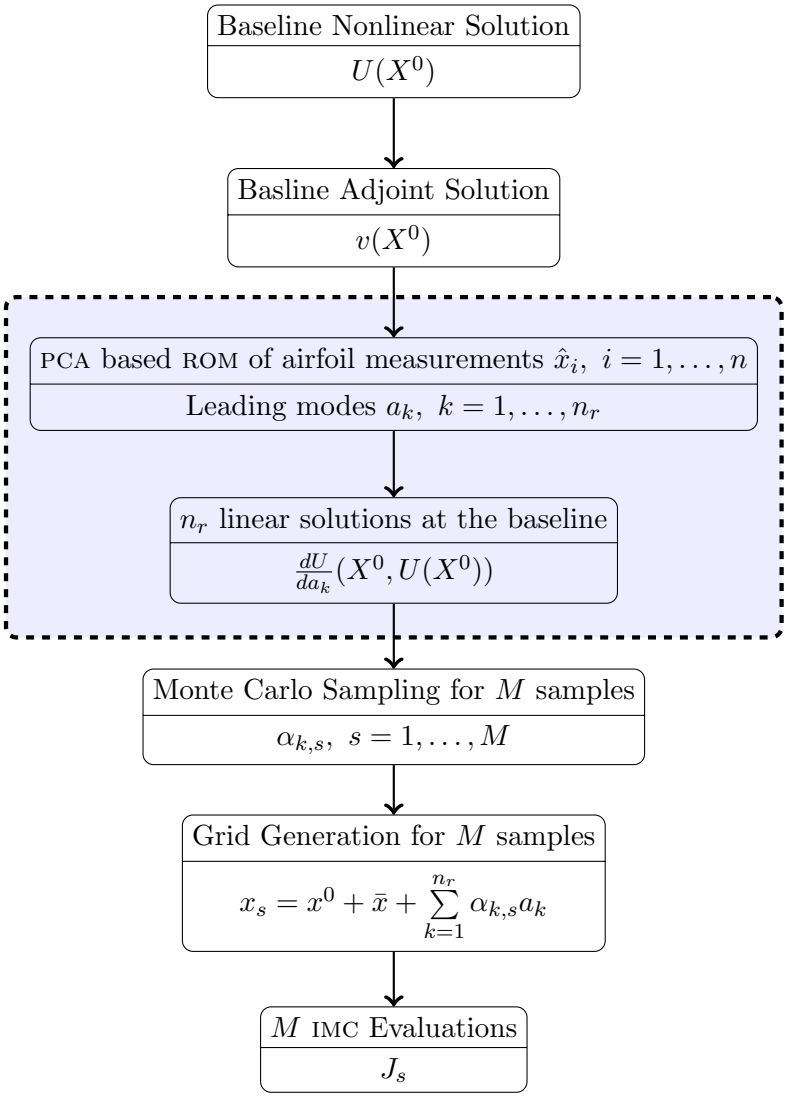

Figure 6.1 Flowchart for the implementation of IMC–2 simulations for fluid mechanic solvers

approach. Since no linear sensitivities are used in the IMC–1 approach, there is no need for reduced order modelling. After the solution of baseline nonlinear and adjoint solution, the geometry measurement points can be used as sampling points for IMC–1.

6.3 CODING

IMC methods need to be implemented as a separate function for fluid mechanic applications. However, it will be shown in this section that the implementation is fairly straightforward if the nonlinear code is written in a modular fashion.

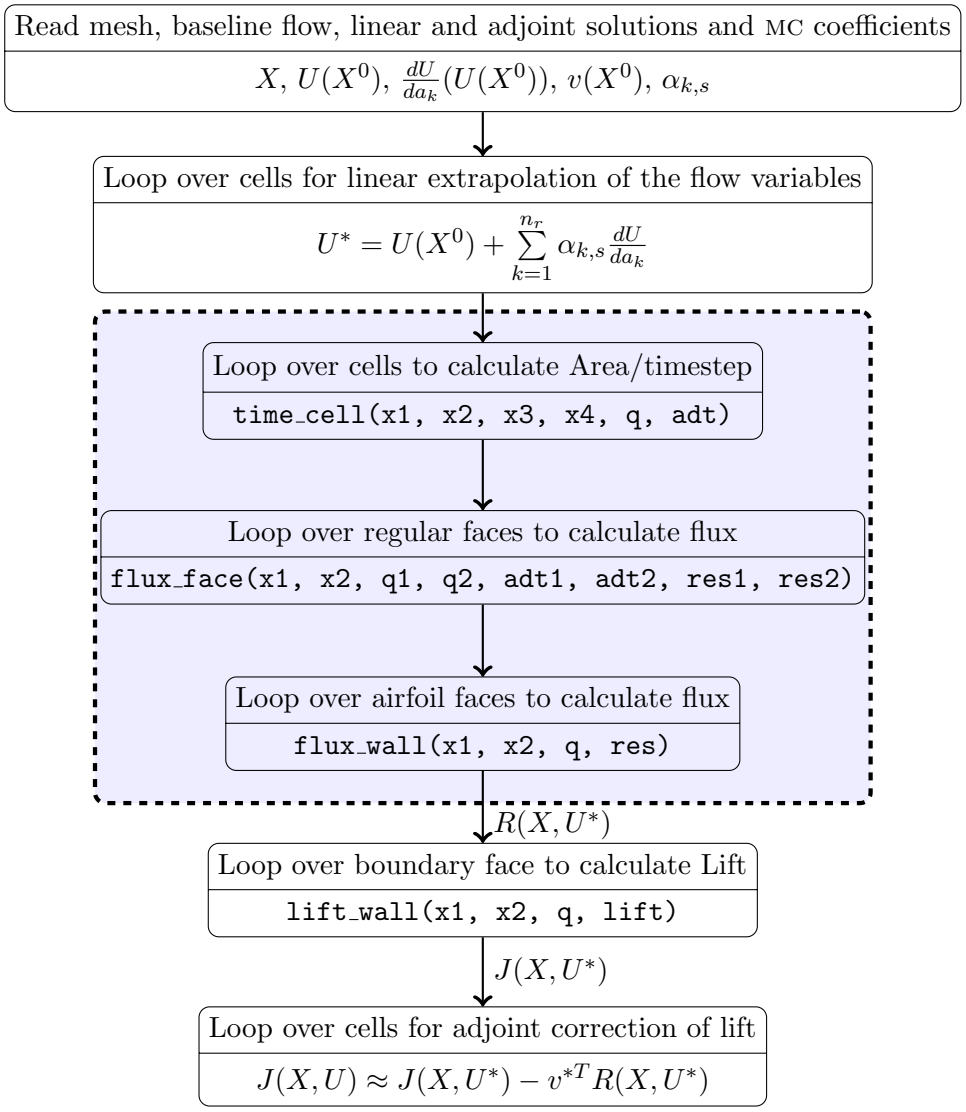

Figure 6.2 Algorithmic representation of the IMC–2 implementation for the airfoil code

Figure 6.2 shows the algorithm for IMC–2 as implemented for the airfoil code. After reading the nonlinear, linear and the adjoint solutions at the baseline, the flow variables are extrapolated using the MC coefficient. The mesh is not extrapolated and nonlinear mesh generation is carried out at each sampling point. The residual

in the entire domain in calculated (as highlighted by the dotted rectangle). This residual is then used for adjoint correction of the objective function. Note that the calculation of $J(X, U^*)$ only involves values of U^* at the airfoil surface. However, since the approximate residual $R(X,U^*)$ is non-zero in the entire mesh (and not just the airfoil surface), the adjoint correction $v(x_0)^T R(X,U^*)$ has to be performed in the entire mesh. However, both of these are simple evaluations which are much cheaper than the iterative nonlinear solutions. IMC–1 implements adjoint correction without linear extrapolation while the linear extrapolation implementation does not perform adjoint correction.

6.4 COMPUTATIONAL COST

Assuming there are n_r leading modes and one objective function, the cost of IMC–2 is:

- One nonlinear solution on baseline geometry,
- One adjoint solution at baseline,
- n_r linear solutions for $\frac{dU}{da_k}$
- M inexpensive Monte Carlo simulations (which are equivalent to M residual evaluations)

This is much cheaper than the full nonlinear Monte Carlo simulations involving M nonlinear iterative solutions.

6.5 RESULTS

6.5.1 *Validation*

An O-mesh is generated for the NACA0012 airfoil using the Joukowski transformation (refer to Appendix B for more details). A coarse mesh is generated with 99×100 vertices. The mesh is rotated through angles of -0.5^o to 0.5^o with the step size of 0.1^o in order to generate artificial mesh perturbations, mimicking the angle of attack uncertainty for an airfoil. The far-field conditions in non-dimensional units are:

- Pressure = 1,
- Density = 1,
- Mach = 0.4, and
- Flow Angle = 3^o.

The lift is calculated using full nonlinear calculations at each point. The adjoint and linear solutions are calculated at the baseline solution. The lift predictions are also obtained using simple linear extrapolation as well as IMC−1 and IMC−2. All the calculations are converged to machine precision. All these simulations are carried out on a 16 node LINUX cluster using a queueing system. A script is generated in MATLAB to automatically execute all these jobs. Figure 6.3 shows the error in the predictions and the actual nonlinear solutions with respect to the angle of perturbation. As expected, the error in the linear extrapolation and IMC−1 shows quadratic behaviour while the error for IMC−2 is cubic. This completes the validation of the implementations for the linear extrapolation, IMC−1 and IMC−2 methods.

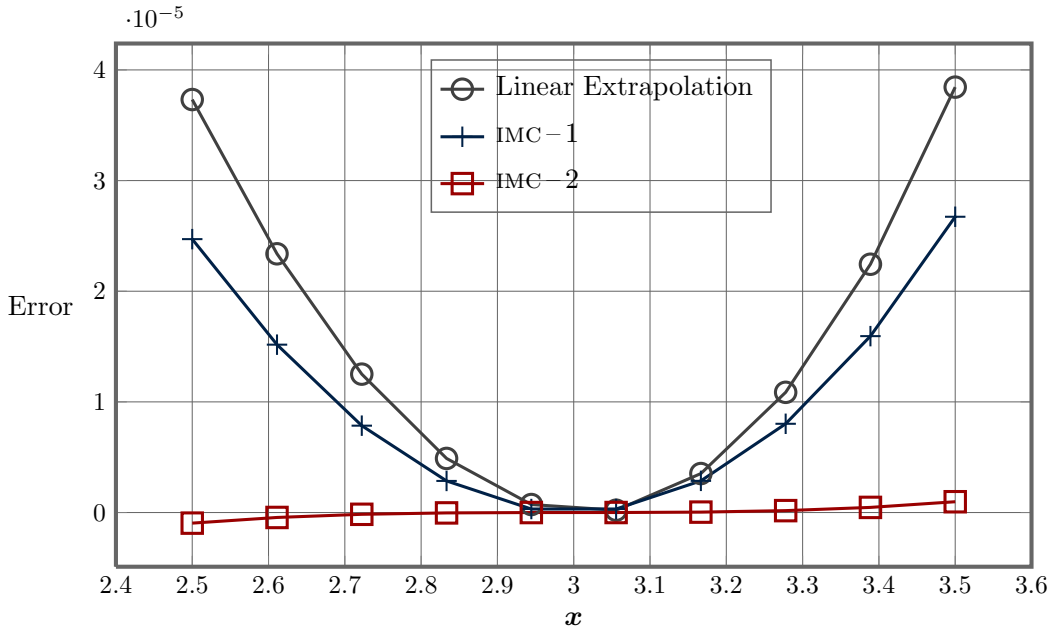

Figure 6.3 Error plot for linear extrapolation, IMC−1 and IMC−2

6.5.2 *IMC Simulations*

After the validation, the IMC methods are tested on a set of airfoils with artificial uncertainty. The same three modes (see Appendix B for more details) are used for IMC simulations as used for the demonstration of the ICA algorithms in section 3.5.2.

Stratified sampling is used to generate the sample set for nonlinear Monte Carlo simulations. A sample size of 1000 is used. After the generation of the airfoils, an internal mesh needs to be generated for these airfoils. For this, mesh perturbations corresponding to the three modes (thickness, twist and leading edge perturbation) are stored initially and then depending on the coefficients generated by the sampling method, these perturbations are added to the base mesh to obtain the perturbed mesh. In the future, mesh perturbation methods need to be implemented for complex three dimensional meshes to save the computational cost of nonlinear mesh generation.

Each nonlinear flow calculation took around 17 minutes. The full set took eighteen hours on a Linux cluster of 16 nodes. All the solutions are converged to machine accuracy. IMC simulations along with linear extrapolation are also carried out using the procedure explained above. As the three modes are already available, it is not necessary to perform PCA in this case. These modes are used as they are. The IMC – 3 approach with extrapolation of the adjoints is not implemented here because of the difficulty in computing the gradients of adjoints $\frac{dv}{da_k}$.

	Mean	Variance
Nonlinear	$\mathbf{3.720894}e-02$	$\mathbf{1.513875}e-06$
Linear Extrap.	$\mathbf{3.721927}e-02$	$\mathbf{1.500570}e-06$
IMC – 1	$\mathbf{3.721417}e-02$	$\mathbf{1.509853}e-06$
IMC – 2	$\mathbf{3.720903}e-02$	$\mathbf{1.512292}e-06$

Table 6.1 Comparison of Mean and Variance predictions for the non-dimensionalised lift parameter

Table 6.1 shows the mean and variance predictions for lift using the above methods. IMC – 2 performs better than the linear extrapolation and IMC – 1 methods. However, in this case, linear and IMC – 1 methods are certainly good enough for engineering purposes.

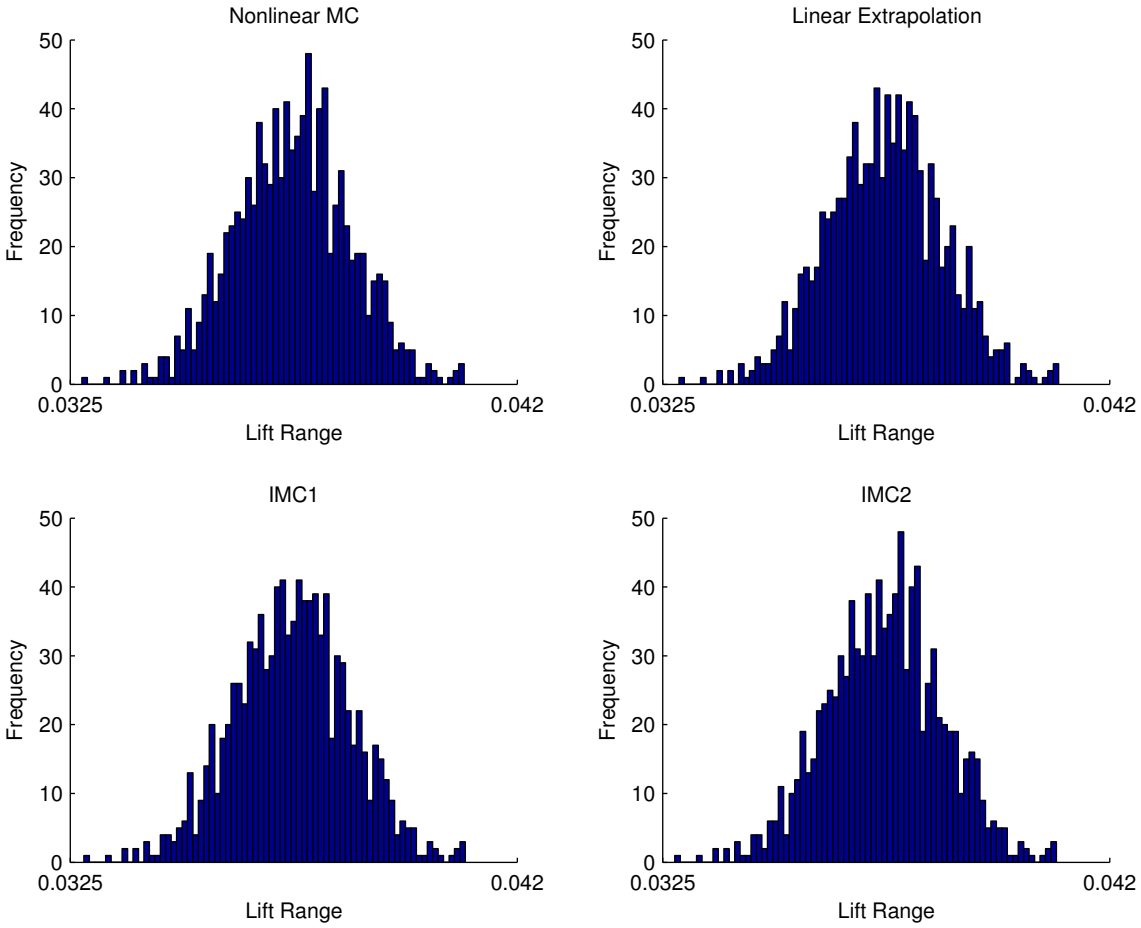

Figure 6.4 Histograms for Lift perturbations showing comparison between nonlinear Monte Carlo simulations, linear extrapolation, IMC – 1 and IMC – 2 methods

Figure 6.4 shows the histograms for the four methods discussed above. As can be seen, the IMC methods closely capture the actual histograms (and consequently PDF) of the distribution. As can be tentatively seen from the histograms, the IMC – 2 method performs slightly better than the other two methods.

Table 6.2 shows the comparison between the computational cost of the various implementation of the IMC methods. The CPU time for 1000 evaluations for each IMC method is given in seconds and also as a ratio of 1000 evaluations of the residuals and the objective function. Each evaluation of the residuals and the objective function is performed by excluding the output subroutine (that writes the flow solution to the disk) and the update subroutine (fixed point iteration), and setting the number of iterations to 1 in the original airfoil code. This modified airfoil code (for the residual and objective function calculation) and the driver programs for the various

IMC methods are run 1000 times using a shell script and the CPU time is measured using the native Linux utility `time`. The specifics of the hardware on which these tests were performed are given in section 4.7.1.

It is observed from the data that IMC – 1 approach which only involves reading one adjoint solution, one vector-vector product after the residual evaluation for calculating the objective function is nearly as expensive as one evaluation of the residual and the objective function. On the other hand, since the linear extrapolation and IMC – 2 involve reading three linear solutions (corresponding to the three modes of perturbation to the airfoil code), they are considerably slower than the IMC – 1. Also, the linear extrapolation and IMC – 2 approach involve extrapolation of the flow variables using the linear solutions. Since the majority of the time is spent in reading files from the disk (rather than calculating residuals), the cost of the IMC – 2 approach is expected to rise linearly with the number of input independent variables. On the other hand, the computational cost of the IMC – 1 approach is independent of the number of input independent variables.

For production grade fluid mechanic solvers, this trend is set to continue with the input operation dominating the computational cost of the IMC methods. Since the fixed point iteration (including various preconditioners and multigrid acceleration) is not required for the residual calculation (and consequently not required for the evaluation of the objective function), reading various solutions is going to take up the majority of the CPU time.

	CPU time (sec)	Ratio
Objective function evaluation	75.6	1.00
Linear Extrapolation	115.5	1.52
IMC – 1	84.5	1.12
IMC – 2	115.9	1.53

Table 6.2 Comparison of the computational cost between the Monte Carlo simulations using linear extrapolation, IMC – 1 and IMC – 2. 1000 evaluations for each Monte Carlo simulation is compared with 1000 evaluations of the objective function using the original nonlinear code.

It should be noted here that the computational cost analysis presented in this section does not include the computational cost of the iterative solution of the nonlinear, linear and the adjoint solver. Only the performance of the driver routines was

compared here to compare the efficiency of the implementation of the driver routines for the IMC methods.

6.6 CONCLUSION

The mathematical formulation of the IMC methods is extended from the algebraic equations to the fluid mechanic applications in this chapter. The algorithm for the IMC simulations for the fluid mechanic applications is outlined. A schematic of the extra module to implement the IMC – 2 method is also presented. A successful implementation of the proposed IMC methods has been demonstrated for a simple two dimensional inviscid airfoil code. Though the methods have performed satisfactorily for this synthetic case, real life measurements for the manufacturing uncertainty are required to assess the suitability of the method.

There is no *a priori* method of determining the range of validity for the IMC – 1 and the IMC – 2 methods. However, in the neighbourhood of the optimum design point, the objective function is likely to show quadratic behaviour (with deterioration of the objective function away from the optimum) necessitating the use of IMC – 2 method. On the other hand, if the tail part of the PDF is being investigated using IMC methods, then IMC – 1 may be sufficient.

Given the availability of the linear and the adjoint solutions, the Hessian matrix can be calculated as outlined in the previous chapter. An alternative to the IMC – 2 method is to use Hessian based quadratic extrapolation as a surrogate model for the Monte Carlo simulations. Since the adjoint corrected linear extrapolation has the leading error of third order (section 2.4.2), the two approaches should result in comparable results. Hence the performance of the adjoint corrected linear extrapolation (used for IMC – 2 method) and the quadratic extrapolation based on the Hessian is compared in section 5.8. As argued in that section, both the extrapolation methods give equally accurate estimates in smooth regions. However, adjoint corrected linear extrapolation (and consequently IMC – 2) closely follows the underlying nonlinear trend than the quadratic extrapolation using Hessian in non-smooth regions.

7 IMC SIMULATIONS USING HYDRA

The previous chapter discussed the extension of IMC methods to fluid mechanic applications using a simple airfoil code. In this chapter, an example application of IMC method is presented using PADRAM as the mesh generator and HYDRA as the fluid mechanic solver. Since HYDRA is an industrial fluid mechanic solver, this adds a lot more complexity in the mesh generation, code maintenance and code validation. This especially highlights the usefulness of various techniques of code validation detailed in chapter 4. Another significant difference is the fact that PADRAM is treated as a third party industrial mesh generator for which the source code is not available. The first section presents a general overview of PADRAM and HYDRA.

7.1 OVERVIEW

7.1.1 *PADRAM*

PADRAM is used for creating the geometry and for the subsequent volume mesh generation. PADRAM is a rapid prototyping and mesh generation tool developed at Rolls Royce Plc., typically used in conjunction with the optimisation package SOPHY. A multi-domain structured mesh is generated by PADRAM which is then stored in the unstructured format to be used with HYDRA. PADRAM is especially suited for parametrised mesh generation during an optimisation cycle. All the aspects of the mesh generation are controlled through a set of input files without any requirement of user input. PADRAM has a rich and complete set of parameter definitions for engine blade geometry to allow a fine grained control over every aspect of the generated volume mesh.

PADRAM is capable of creating two and three dimensional, single and multi-passage meshes suitable for viscous and inviscid flow calculations. PADRAM uses C-O-H topology to generate multi-block structured grids. The O type grid is used to refine the region near the airfoil surface for better resolution of boundary layer. H type grid blocks are employed near the far-field boundaries where stretched cells are required. PADRAM uses transfinite as well as elliptic mesh generation algorithms to

generate the volume grid. The algorithm used by the mesh generator internally can be controlled by the user via input parameters.

PADRAM is used in this study as industrial software without the availability of the source code and, consequently, without the ability to modify the algorithm in any way.

7.1.2 *HYDRA*

HYDRA is a suite of fluid mechanic codes [26, 72, 78] originally developed at the Rolls-Royce University Technology Centre (UTC) of Oxford University by Prof. Giles and co-workers over a period of years. Since then it has become the central tool for aerodynamic analysis across Rolls Royce research centres and is being actively developed at various other UTCs. The components of HYDRA used in this dissertation are the nonlinear solver and the harmonic linear and adjoint solvers.

The nonlinear solver solves the unsteady Reynolds Averaged Navier-Stokes equations coupled with the Spallart-Allmaras one-equation turbulence model using an edge-based spatial discretisation on unstructured hybrid grids and a pseudo-time marching scheme. Characteristic smoothing is applied for inviscid fluxes in the spatial discretisation, and a five-stage Runge-Kutta algorithm to evolve the solution to the steady state. Multigrid [73] is used to accelerate convergence. The nonlinear solver also supports distributed-memory parallel computations using Oxford Parallel Library (OPlus) [11] framework for unstructured grid solvers which is based on the message passing interface (MPI) library.

Based on this nonlinear solver, a harmonic linear and adjoint solver (developed by Duta [26]) is also available in the HYDRA suite. The linear and adjoint solvers follow the discrete approach for adjoint code development. This entails that the discretised nonlinear flow equations are linearised and the linear harmonic operator thus obtained is used to form the corresponding adjoint equations. These are used in the present demonstration of the IMC simulations in this thesis. The harmonic linear and adjoint solvers inherit the data structures, preconditioning algorithms, multigrid method, spatial discretisation scheme and the time integration schemes from the nonlinear solver.

Each of the harmonic solvers has three distinct stages: pre-processor, harmonic solver and the post-processor. The pre-processor generates the right hand side of the linear solver (equation (4.4)) and the adjoint solver (equation (4.7)). The main harmonic

solvers use the non-homogeneous terms output by the pre-processor to calculate the linear/adjoint solution. Finally, the post-processor uses the linear/adjoint solutions to output the sensitivities of the objective functions. Hence, each run of linear/adjoint solver consists of essentially three separate runs of the pre-processor, main solver and the post-processor.

An additional complexity is added in solving the linear and adjoint equations for three dimensional viscous applications. If the nonlinear solution is not fully converged and is trapped in a low level limit cycle oscillation, then the linear and adjoint solvers were unstable. A capability of generalised minimal residual method (GMRES) is available in HYDRA to tackle such situations. A GMRES iteration [12, 13] is wrapped around the standard multigrid solver to stabilise such instabilities.

The source code of HYDRA is available giving full access to internal data structures. Hence modules can be written to calculate arbitrary functions of the mesh and flow variables.

7.2 WORKFLOW

Apart from the two main components of the fluid mechanic suite, various other helper pre/post processing programs from HYDRA are used. Figure 7.1 shows the workflow for a single nonlinear fluid mechanic simulation using HYDRA.

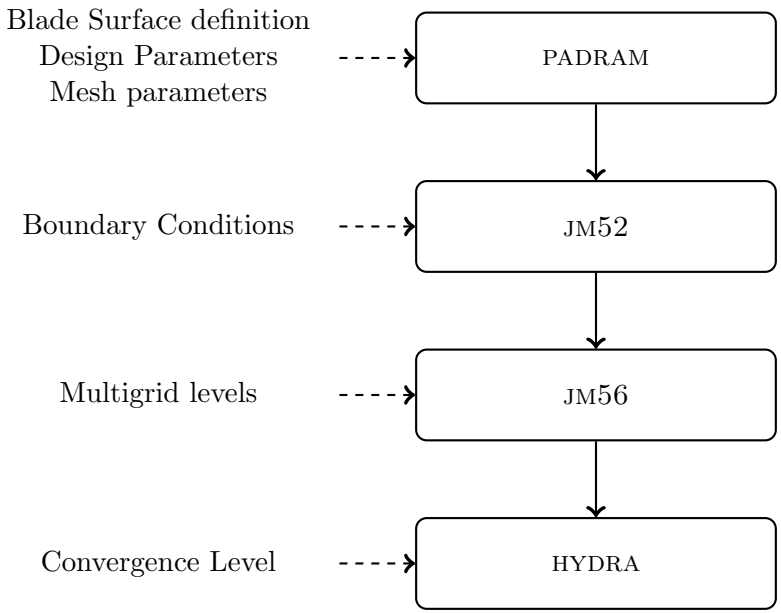

Figure 7.1 Workflow for a single fluid mechanic simulation using HYDRA

Engine blade geometry is input to PADRAM along with a configuration file which specifies all aspects of the mesh generation. A multi-block structured mesh is generated by PADRAM and stored in a single precision ASCII PLOT3D format[28]. The mesh is converted into an unstructured format by JM52 module which also generates input files specifying the initial and boundary conditions for HYDRA. The JM52 also calculates weights for edges to be used by the flow solver. The input files are converted into binary advanced data format (ADF) format at this stage and all the HYDRA modules henceforth use double precision binary ADF format. The JM56 module processes the single block unstructured mesh to generate various coarse meshes for multi-grid calculations. Fluid mechanic analysis using the nonlinear, linear and adjoint HYDRA modules can be carried out after this.

The whole process of mesh generation, preprocessing and nonlinear solution can be automated using any scripting language (MATLAB is used in the present application) since all the software components listed do not need any user intervention. All the input parameters for various components can be supplied through input files.

An additional step is introduced for the linear solver in the form of calculation of the grid perturbations. Some aspects of this are discussed in the next section.

7.3 IMC IMPLEMENTATION

Implementation of IMC methods (as discussed in section 6.3) extends seamlessly from the two dimensional example airfoil code to HYDRA. An extra module is written by calling various functions from the HYDRA code. Since HYDRA is written in a modular fashion as detailed in chapter 4, it is a relatively simple task to write a separate module for IMC methods following the algorithm outlined in Figure 6.2.

7.4 TEST CASE

Since actual geometry measurements are not available, NASA-rotor-37 is used as the baseline blade along with a design variable to perturb the blade surface. The ensuing results show the performance of the IMC method for a single variable. They can be extended to situations with multiple design variables (as demonstrated in Chapter 6 for the airfoil code).

[28] Since the work presented in this chapter has been carried out in 2007-08, some of the details presented here regarding PADRAM may be out of date.

The skew design variable as provided by PADRAM version 3.10.1 is used to perturb the geometry. Skew is defined as the rotation of the rotor blade about the radial axis (Z axis) passing through mid chord. The range of skew is chosen such that the maximum y movement at the tip of the blade is not more than 1% of the tip radius. The range of skew variation is chosen as $0^o \pm 0.5^o$.

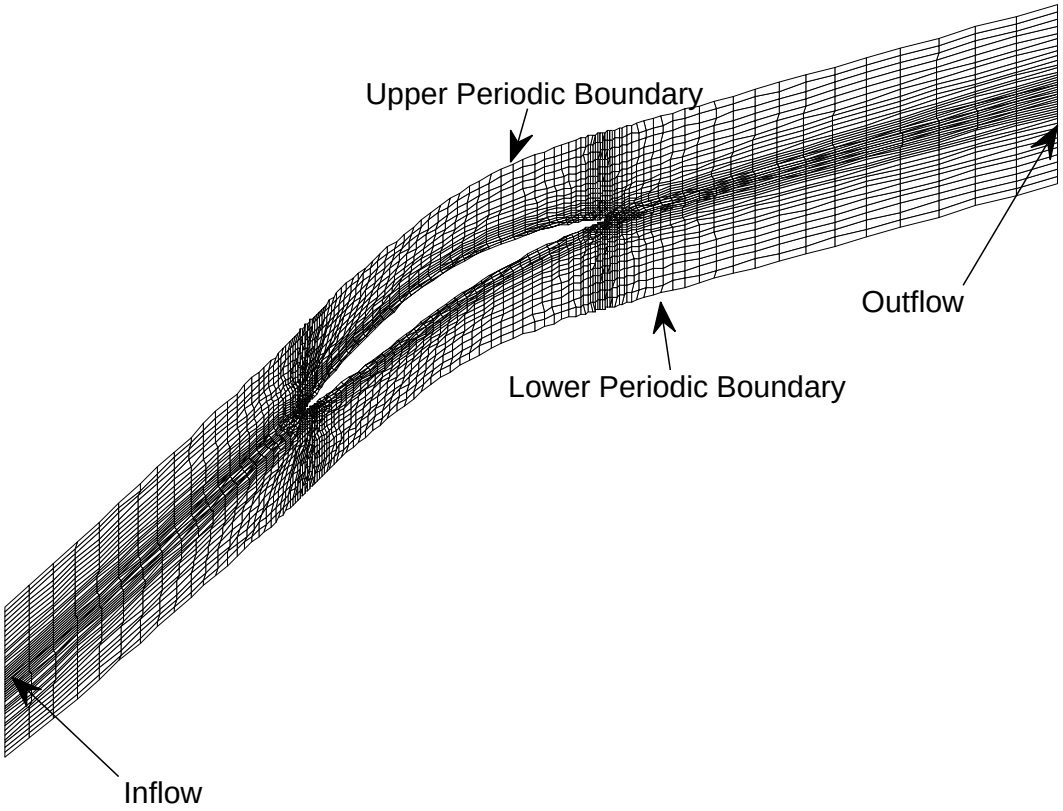

Figure 7.2 Mesh for the hub viscous wall boundary condition highlighting the two dimensional features of the mesh.

The volume mesh consists of 318798 nodes for a single passage calculation. Figure 7.2 shows the mesh for the hub wall. The inter-blade phase angle (IBPA) is set to zero for a single passage calculation. A step size of $s = 0.01^o$ is chosen for grid perturbation for the linear calculations. HYDRA version 3.41 is used for all the simulations on a 16 node Linux cluster used at the Oxford University Computing Laboratory.

Figure 7.3 shows the comparison of linear static pressure perturbation (bottom figure) for the outflow boundary condition with the nonlinear static pressure perturbation. The nonlinear perturbation is obtained as the difference between the pressure profiles obtained by nonlinear solutions at the perturbed point ($s = 0.01^o$) and the baseline

($s = 0^o$). The two profiles show good qualitative agreement. Table 7.1 shows the comparison of the sensitivity of efficiency using the central difference scheme with skew perturbation of $\pm 0.00001^o$ and the sensitivity derived from the linear solver for skew perturbation. They show good agreement thereby confirming the choice of step size for the linear solver.

Method	Step size	Sensitivity
Finite Difference	0.00001^o	$(84.45853874 - 84.45852439)/0.00002 = 0.717202$
Linear Solver	0.01^o	$(7.085526192e-3 + 8.514637778e-5)/0.01 = 0.717067$

Table 7.1 Comparison of the sensitivity of isentropic efficiency using central finite difference with step size of $\pm 0.00001^o$ for skew and the sensitivity obtained by the linear solver using the step size of 0.01^o.

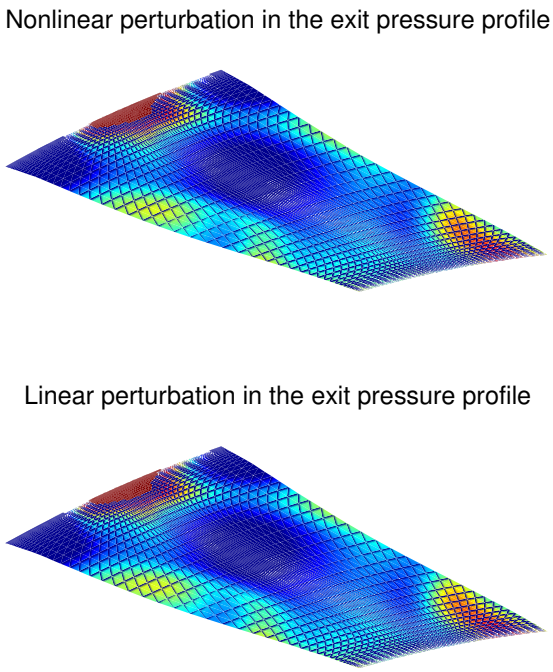

Figure 7.3 The static pressure profile of the outflow boundary condition. The top figure shows the difference between the static pressures from two nonlinear solutions (one for the perturbed grid with skew of 0.01^o and the baseline solution (s = 0^o). The second figure shows the linear perturbation in the static pressure with the skew step size of 0.01^o.

7.5 MESH PERTURBATION

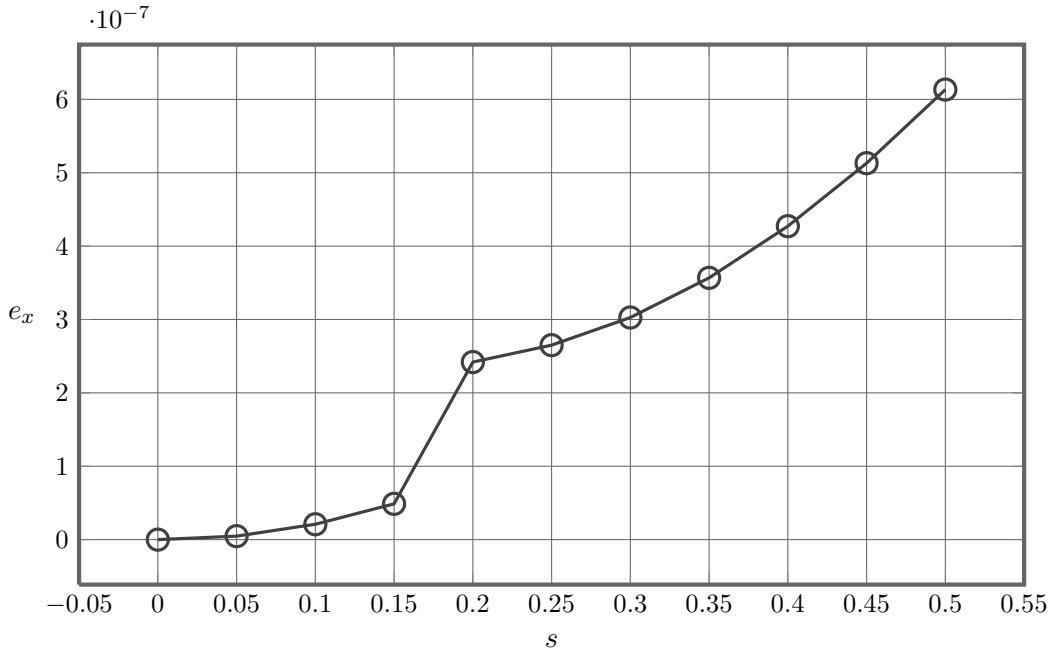

Figure 7.4 Figure plots the L^2 norm of the difference between nonlinear grid and linear grid perturbation with skew. A clear discontinuity can be seen between $s = 0.15^o$ and $s = 0.2^o$.

Since PADRAM is treated as an industrial mesh generator whose source code and internal data structures are not available for manipulation, grid perturbations can not be generated in a straightforward manner (as for the airfoil code). A mesh generation algorithm is a nonlinear function with design parameter α as the input and the mesh coordinates X as the output. Let

$$X = g(\alpha)$$

where, g is the nonlinear mesh generation function in the discrete form. The degree of the nonlinearity of this function depends on the algorithm employed by the mesh generator to calculate the mesh coordinates. If the mesh generator is considered as a black box, then the finite difference method is the only option for generating the volume mesh perturbations which can be represented as,

$$\frac{dX}{d\alpha} = \frac{g(\alpha + \Delta\alpha) - g(\alpha - \Delta\alpha)}{2\Delta\alpha} + \mathcal{O}(\Delta\alpha^2). \tag{7.1}$$

As discussed in section 4.2, this necessitates experimentation to find an appropriate step size for the perturbation.

Figure 7.4 plots baseline geometry over the entire domain with increasing skew (s).

$$e_x = \sum_i (X_i(s + \Delta s) - X_i(s) - \Delta s \frac{dX_i}{ds})^2 \tag{7.2}$$

where e_x is the error and s is the skew variable. It is expected that the error is of second order. However, it can be observed clearly from the plot that there is a sudden jump in the error between $s = 0.15^o$ and $s = 0.2^0$.

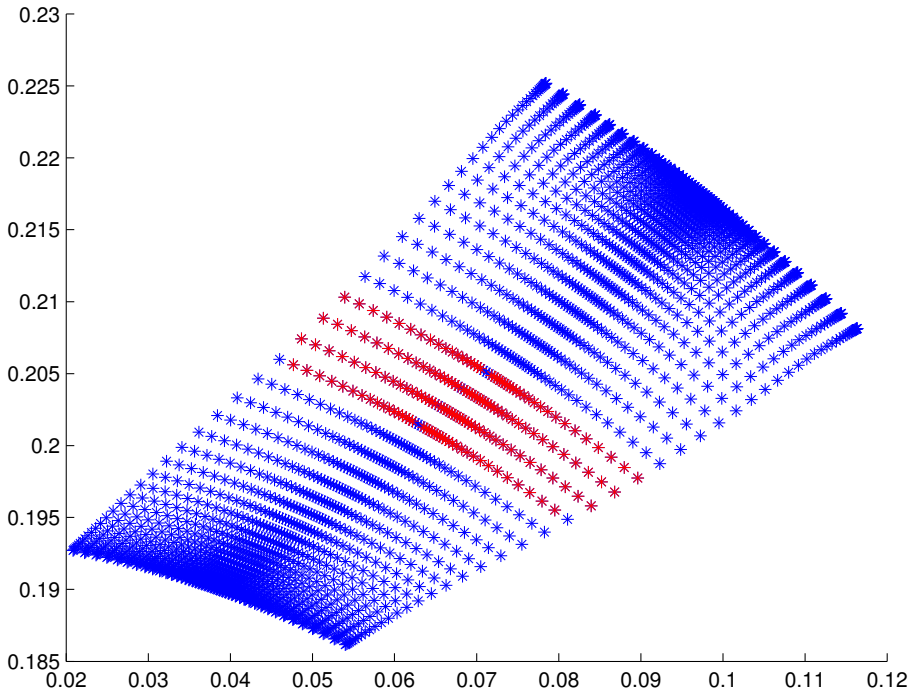

Figure 7.5 Outflow boundary of the single passage mesh with some mesh points not showing the leading error of second order.

Further investigation revealed that this jump occurs at $s = 0.1884^o$. The error (e_x) was plotted at various internal mesh points and the boundary conditions to investigate the source of this jump. Figure 7.5 plots the outflow boundary of mesh domain. The mesh points that do not show the quadratic behaviour for the L^2 error norm e_x are highlighted in red color. It was found that there is a sudden switch in the internal PADRAM mesh generation algorithm at $s = 0.1884^o$ at the outflow boundary plane. The exact reason for this behaviour is not investigated since the source code for PADRAM is not available and it is treated as a black box.

Hence, PADRAM could not be used for nonlinear mesh generation for IMC simulations. To address this issue, a volume mesh perturbation is generated using skew perturbation of 0.01^o and added to the baseline volume mesh to generate mesh for various skew values. JM52 and JM56 are run after adding the volume mesh generation to the baseline mesh to add boundary conditions and the multigrid mesh.

7.6 IMC VALIDATION

The skew variable s in the range of $\pm 0.5^o$ (as outlined in the previous section) is used as the design variable. The performance of the IMC methods is compared for three different objective functions: isentropic efficiency, outlet massflow and the pressure loss. A baseline nonlinear solution followed by the linear harmonic solution for the skew perturbation of $s = 0.01^o$ was performed. Similarly, the adjoint solutions are obtained for the three objective functions at the baseline.

The methodology outlined in the previous section is used to generate the volume mesh for the nonlinear flow solutions. The nonlinear flow solutions are obtained for various points in the skew range of $-0.5^0 \leq s \leq 0.5^o$. The error between the objective functions obtained using IMC–1 and IMC–2 methods compared to the nonlinear objective functions is evaluated. The performance of IMC methods is also compared with the objective functions obtained using linear flow extrapolation. This requires the addition of the flow and mesh perturbations to the baseline solutions followed by the calculation of the objective function.

Figure 7.6 shows such comparison for the linear extrapolation, IMC–1 and IMC–2 methods with skew variation with the isentropic efficiency as the objective function. The error for linear extrapolation and IMC–1 method is quadratic while IMC–2 is more accurate. Figure 7.7 and Figure 7.8 show similar plots for the massflow and pressure loss objective functions. This completes the validation process for the implementation of the IMC methods for HYDRA.

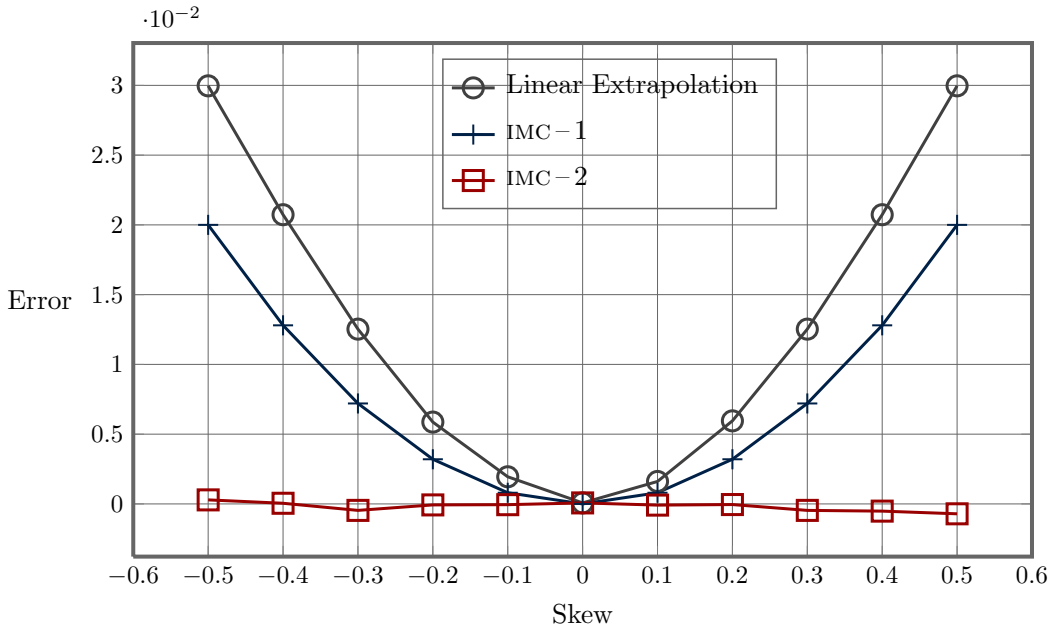

Figure 7.6 Error plot for isentropic efficiency showing the difference of values obtained by linear extrapolation, IMC − 1 and IMC − 2 methods compared to the nonlinear solution

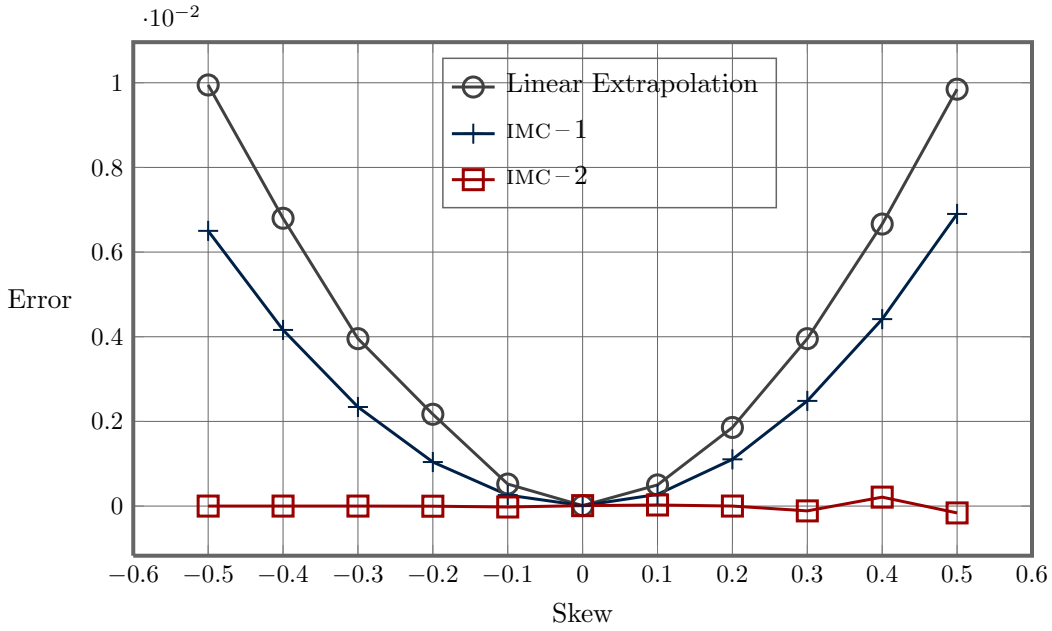

Figure 7.7 Error plot for massflow showing the difference of values obtained by linear extrapolation, IMC − 1 and IMC − 2 methods compared to the nonlinear solution

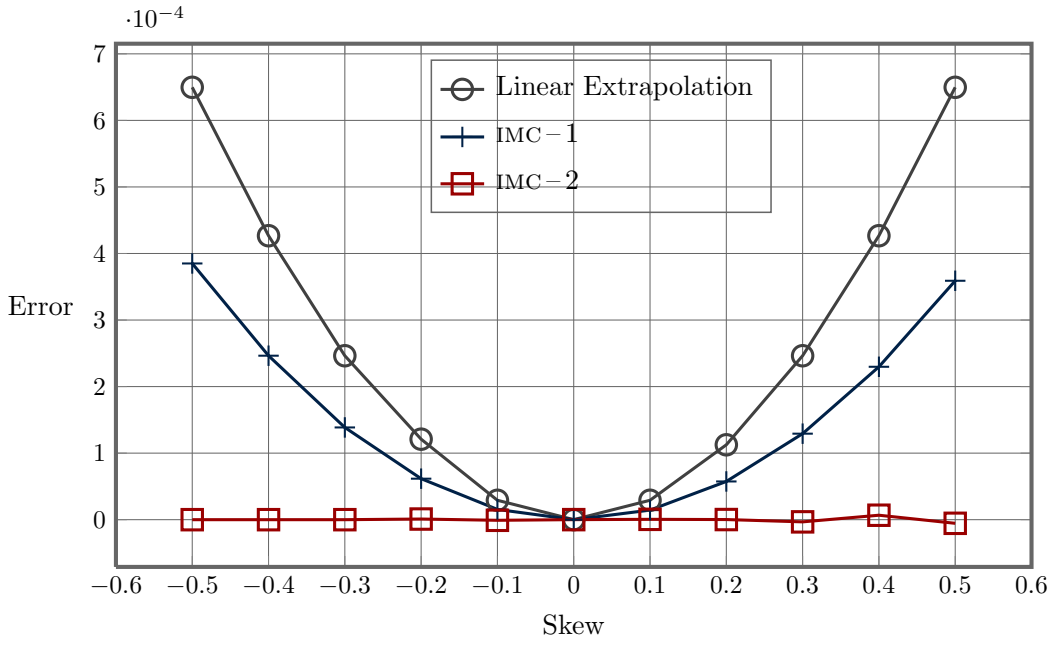

Figure 7.8 Error plot for pressure loss showing the difference of values obtained by linear extrapolation, IMC−1 and IMC−2 methods compared to the nonlinear solution

7.7 CONCLUSION

A successful implementation of the IMC method is demonstrated for an industrial flow solver in this chapter. If accurate linear and adjoint solutions are available, it is a relatively straightforward process to implement IMC methods. It is found that as the complexity of the mesh generator and the flow solver increases, it is increasingly important to perform systematic checks at each stage of IMC simulations.

The aim of the work presented in this chapter was to demonstrate the successful implementation of IMC methods for a complex industrial flow solver. Hence, the design variable (skew) was perturbed in a relatively narrow range. Though the validation process includes only a single design variable, the IMC simulations can be extended to multiple design variables as demonstrated for the example airfoil code in chapter 6.

8 CONCLUSION

This chapter summarises the work presented in this dissertation. It once again outlines the motivation behind the research reported in this thesis. This is followed by a summary of the work with a particular emphasis on the original contributions made during the work. The chapter concludes with a few remarks on the possible directions for future research.

The primary motivation behind the work presented in this thesis has been the increasing availability of the measurement data for aircraft engine blades. This trend is set to continue in the future with the ready availability of affordable and accurate measurement machines. New possibilities have opened up before the design engineers to perform various analyses with the available data with the accompanying engineering challenges.

The IMC method (based on the concept of adjoint error correction) is proposed in this thesis to make a better use of the blade measurement data to yield a complete probability distribution for the objective function at a reasonable computational cost. The present alternatives are the nonlinear Monte Carlo method which is computationally expensive, and the moment methods which only provide the mean and the variance of the objective function. It has been shown in chapter 2 that the proposed IMC simulation method yields the complete probability distribution of the objective function at a marginally larger computational cost than the first order moment method. The key ideas of the IMC are outlined in chapter 2. A comparison between the IMC methods, moment methods and nonlinear Monte Carlo method is presented using an example algebraic equation.

Chapter 6 extends the IMC formulation to the fluid mechanic solvers with the use of an example airfoil code. The complete implementation details of the IMC methods are presented in this chapter along with the use of reduced order modelling. Three modes of artificial geometric uncertainty are added to NACA0012 airfoil to demonstrate the use of IMC methods. It has been shown that for a small uncertainty, IMC simulations are as effective as the nonlinear Monte Carlo methods in obtaining the probability distribution of the objective function.

An application of IMC simulations has been extended to an industrial flow solver HYDRA in chapter 7. The aim of this activity was to demonstrate the ease of implementation of IMC simulations for a large production grade fluid mechanic code. PADRAM has been used as an example industrial mesh generator along with HYDRA. Artificial geometric uncertainty is introduced in the blade geometry and propagated through PADRAM and HYDRA to show successful implementation of the IMC simulations.

This completes the development of IMC methods from the mathematical formulation to the successful implementation for an industrial code (HYDRA).

Another avenue for the use of the blade measurements is the possibility of a meaningful statistical analysis of the measurements themselves to glean more information about the possible sources of manufacturing variability. A new application of the ICA method has been proposed to provide extra statistical information regarding the sources of manufacturing geometric variability. The rationale behind this argument and an artificial example application has been presented in chapter 3.

Another motivation behind the work presented in this thesis has been the recent interest in the use of high-fidelity simulations throughout the design process. Routine use of high-fidelity simulations has been partly possible because of the increasing availability, and the decreasing cost of the computing resources. This switch to high-fidelity simulations has also thrown its unique challenges at the design community.

One of the most important areas of research has been the development of the methodologies to calculate accurate and consistent gradient information for the fluid mechanics codes. As outlined in chapter 4 of this thesis, traditional finite difference methods are not adequate for this purpose. On the other hand, it is not feasible to maintain a linear and an adjoint version of the ever changing nonlinear flow solver. A relatively new tool: automatic differentiation is used to address this issue. Though AD tools are getting increasingly sophisticated and computationally efficient, it is not feasible to use them as a black-box application for the nonlinear flow solvers. Considerable amount of work has already been done by various groups on the efficient use of AD for gradient calculation[18]. The work presented in this thesis outlines a step-by-step consistent methodology (refer to chapter 4) to use AD technology efficiently to automate the development and maintenance of the linear and the adjoint versions of the original nonlinear code. The methodology explored in

this thesis can keep the linear and the adjoint versions of the code in-sync with the changes in the nonlinear flow solver. Additional tests have also been developed to debug any programming errors during the development of the linear and the adjoint codes. An example airfoil code has been developed to illustrate all the key ideas presented here.

The use of AD for Jacobian calculation is extended to Hessian calculation in the chapter 5. Efficient and systematic use of AD is demonstrated for the Hessian calculation based on the ideas presented in [85, 88]. The method and algorithm is demonstrated using an example airfoil code. Good agreement between the Hessian and its finite difference approximation is achieved.

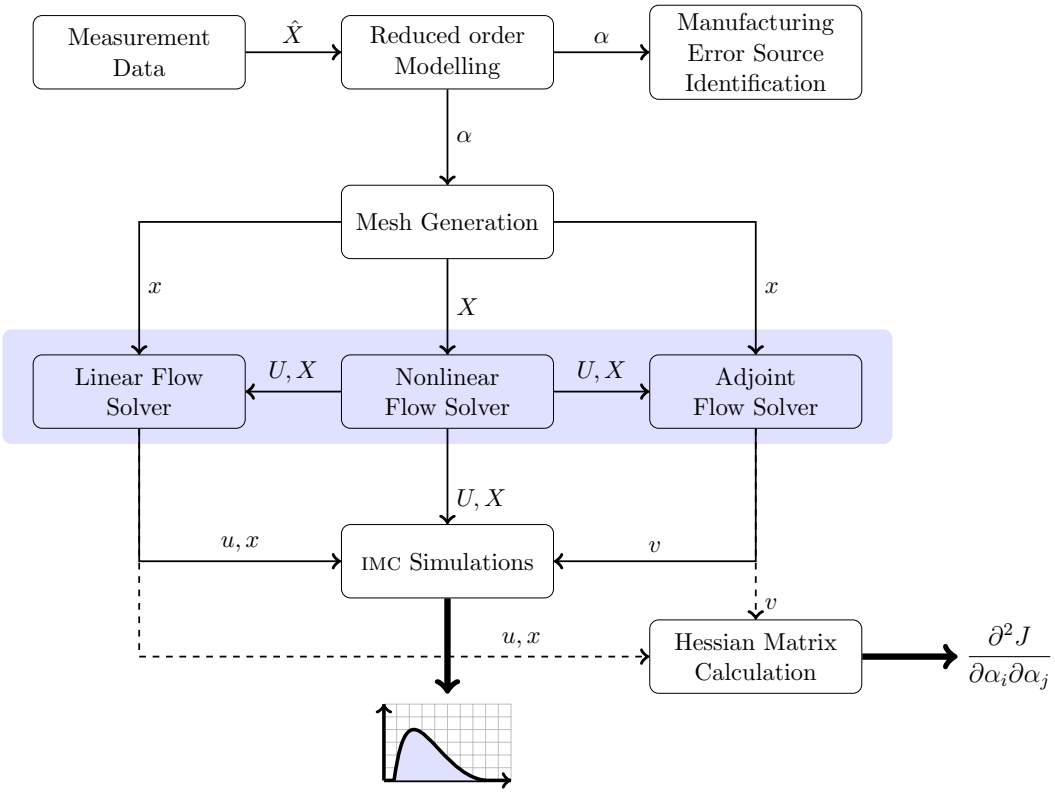

Figure 8.1 Various aspects of the manufacturing uncertainty analysis addressed by the present thesis

This concludes the summary of the work presented in this thesis. Figure 8.1 highlights all the different areas of design process in which original contributions have been reported in this thesis.

- A novel method of IMC simulation based on the idea of adjoint error correction (as proposed by Giles and Pierce) has been reported in this

thesis. The mathematical formulation of the IMC method is followed by the implementation details for the example airfoil code and HYDRA.

- A systematic method for the development of the linear and the adjoint codes based on the nonlinear flow solver has been outlined in this thesis. The systematic method can keep the linear and the adjoint solvers in-sync with the changes in the nonlinear flow solver. The implementation details have been demonstrated for an example airfoil code.

- Efficient use of AD technology is demonstrated for the development of Hessian calculation using the airfoil code. A complete mathematical formulation for the Hessian calculation for fluid mechanics solvers is presented (based on the ideas presented by Putko et al. [85, 88]) followed by the implementation details for a typical fluid mechanics solver.

- A new application of the ICA is proposed for identifying the sources of manufacturing variability. A brief introduction to various ICA algorithms is presented followed by the demonstration of the ICA algorithm on artificial geometric uncertainty introduced to an airfoil. However, since actual blade measurement data was not available, the effectiveness of ICA has not been established in the present work. Some of the possible limitations of the application of the ICA to manufacturing variability source separation are discussed. A discussion on the applicability of the ICA algorithms to actual blade measurement data is also presented.

Future Work

The work reported in this thesis has outlined the mathematical formulation, algorithms and implementation details for the IMC methods. The implementation of the IMC–1 and IMC–2 simulations has been demonstrated for a two dimensional airfoil code and a three dimensional fluid mechanic solver HYDRA. However, IMC–3 simulation that requires the sensitivity of the adjoint variables with respect to the independent input variables has not been demonstrated. A future direction of study might include calculation of the sensitivity of the adjoint variables using AD (as reported by [17, 66]) and the implementation of the IMC–3 simulation. Another area for exploration is to assess the validity of the IMC simulations for blade measurement data. It will be interesting to investigate if the geometric uncertainty due to manufacturing variability is small enough as to require only IMC–2 simulation. Once the blade measurement data is available, the performance of the IMC simulations

can be compared at various engine design points. It is expected that the ability of the IMC simulations to capture the trend of the nonlinear Monte Carlo methods will depend on the design point at which the simulation is performed.

Similarly, the proposed idea of using ICA for manufacturing variability source identification (as outlined in chapter 3) needs to be tested using blade measurement data. There are multiple lines of enquiries to be pursued here. It remains to be seen if the basic assumptions behind ICA are satisfied by the blade measurement data. Firstly, it needs to be ascertained that there are indeed only small number of independent sources of manufacturing variability. Secondly, another key assumption for the ICA analysis is the linear combination of the sources of manufacturing variability. If the sources do not combine in a truly linear fashion then it needs to be investigated if the magnitude of geometric uncertainty is small enough to justify a linearity assumption. Performance of various ICA algorithms can then be compared using the blade measurement data. Comparison of the performance of various ICA algorithms for the artificially generated geometric uncertainty for airfoil does not show clear advantage for any ICA algorithm over others.

A EULER SOLVER

A.1 NONLINEAR SOLVER

The nonlinear two dimensional code solves Euler flow around an airfoil using an unstructured grid. Unsteady inviscid flow is governed by the following Euler equations:

$$\frac{\partial U}{\partial t} + \frac{\partial F(U)}{\partial x} + \frac{\partial G(U)}{\partial y} = 0 \tag{A.1}$$

where

$$\begin{aligned} U &= \left(\rho, \rho v_x, \rho v_y, \rho E\right)^T, \\ F(U) &= \left(\rho v_x, \rho v_x^2 + p, \rho v_x\, v_y, \rho v_x\, H\right)^T, \text{ and} \\ G(U) &= \left(\rho v_y, \rho v_x\, v_y, \rho v_y^2 + p, \rho v_y\, H, \right)^T. \end{aligned}$$

Here,
ρ is the density,
v_x is the velocity in x-direction,
v_y is the velocity in y-direction,
p is the static pressure,
$E = \frac{p}{\rho(\gamma-1)} + \frac{v_x^2+v_y^2}{2}$ is the total energy, and
$H = E + \frac{p}{\rho}$ is the total enthalpy.

These equations are solved using a cell centred finite volume discretisation in conservative variables. A simple centred averaging method with constant smoothing is used to calculate flux across the faces. As the code is only used for subsonic flow conditions the solution is not expected to have any discontinuities in the form of shocks. Hence, there is no need for a more sophisticated discretisation method like upwind schemes. Consider a face between the i^{th} and j^{th} cells, the flux across this face is given by

$$R_{ij} = \frac{(F_i + F_j)}{2} dy - \frac{(G_i + G_j)}{2} dx + S \tag{A.2}$$

where, dx and dy are projections of the length of the face in the x and y directions respectively. The smoothing term S is given by

$$S = \epsilon \frac{(Adt_i + Adt_j)}{2}(U_i - U_j),$$

where ϵ is the smoothing coefficient and Adt is the area over the local time-step. A small value of $\epsilon = 0.05$ is used to minimise the first order error introduced by this smoothing term. Adt for a cell is calculated as

$$Adt = \frac{\sum_i |u\ dy_i - v\ dx_i| + c\ \sqrt{dx_i^2 + dy_i^2}}{CFL}, \tag{A.3}$$

where c is the speed of sound, CFL is the Courant-Friedrichs-Lewy number and the loop is over all the neighbouring faces. Fixed point iterations are performed using the predictor-corrector method which may be stated for the n^{th} iteration and the i^{th} cell as

$$U_i^{int} = U_i^n - \frac{R(U_i^n)}{Adt_i^n}$$

$$U_i^{n+1} = U_i^n - \frac{R(U_i^{int})}{Adt_i^n}$$

The nonlinear solver has been kept as simple as possible for the ease of linearisation. It is understood here that this is not the most efficient and accurate way to solve the Euler equations.

A.1.1 *Boundary Conditions*

All the boundary conditions are implemented in the weak form. The wall boundary condition is implemented using a dummy cell to ensure zero mass flux through the boundary faces. In practice a 4×4 transformation matrix can be calculated which when multiplied by the flux vector of the interior cell gives zero net flux through the wall faces.

The objective function lift is calculated as

$$J = \sum_i p_i \left(\cos\alpha\ dx_i + \sin\alpha\ dy_i\right), \tag{A.4}$$

where α is the flow angle and the summation is carried out on the airfoil wall faces.

A.2 LINEAR SOLVER

The linear solver uses the predictor-corrector time integration. The matrix L is calculated by summing over all the edges and differentiating equation (A.2) which gives

$$\frac{\partial R_{ij}}{\partial U_i} = \frac{(A_f dy - A_g dx)}{2} + A_s \,,$$

where,

$$A_f = \begin{pmatrix} 0 & 1 & 0 & 0 \\ v_x^2 + \frac{\gamma-1}{2}v^2 & (3-\gamma)v_x & -(\gamma-1)v_y & (\gamma-1) \\ -v_x v_y & v_y & v_x & 0 \\ -v_x(\gamma e - (\gamma-1)v^2) & \gamma e - \frac{\gamma-1}{2}(v^2 + 2v_x^2) & -(\gamma-1)v_x v_y & \gamma v_x \end{pmatrix}$$

$$A_g = \begin{pmatrix} 0 & 0 & 1 & 0 \\ -v_x v_y & v_y & v_x & 0 \\ v_y^2 + \frac{\gamma-1}{2}v^2 & -(\gamma-1)v_x & (3-\gamma)v_y & (\gamma-1) \\ -v_y(\gamma e - (\gamma-1)v^2) & -(\gamma-1)v_x v_y & \gamma e - \frac{\gamma-1}{2}(v^2 + 2v_y^2) & \gamma v_y \end{pmatrix}$$

where all the flow variables are from the i^{th} cell. Similar, matrices will have to be added from the j^{th} cell. Also, A_s is the linearisation matrix for the smoothing term S. This matrix will include the linearisation of the Adt term with respect to U. These terms are so messy that it is not feasible to give the exact expressions here. Similar expressions can be derived for $\frac{\partial R_{ij}}{\partial U_j}$ and $\frac{\partial R_{ij}}{\partial X}$, the linearisation with respect to the geometry.

Though the process of linearising these equations is straightforward, it can be seen that it gets tedious even for two dimensional Euler equations. They become increasingly difficult for three dimensional Navier-Stokes equations. These equations are only presented here to emphasise the ease with which AD can perform this linearisation with little effort.

B GRID GENERATION

A single block grid is generated for a NACA0012 airfoil using the Joukowski transformation which is defined as

$$z = \zeta + \frac{1}{\zeta} \tag{B.1}$$

where z and ζ are complex variables. The inverse transformation is defined as

$$\zeta = \frac{-z \pm \sqrt{z^2 - 4}}{2} \tag{B.2}$$

This transformation maps a line segment from $z = -2$ to $z = 2$ to a circle in the ζ plane centred at the origin and with unit radius. If the airfoil is mapped so that x varies from -2 to 2 then this transformation will give nearly a circle in the ζ plane. This near circular shape can be transformed into a circle using a simple algebraic transformation after which, the curve in the ζ plane can be expanded progressively and using equation (B.1) the interior grid can be generated. This will generate an O-type grid around the airfoil which is nearly orthogonal to the airfoil surface. Figure B.I shows a sample grid for NACA0012 with zero angle of attack. A standard grid size of 99×100 is used for all the calculations.

B.1 ARTIFICIAL MODES OF PERTURBATION

Three artificial modes of perturbation are introduced to the ideal airfoil geometry. These modes are used by the proof-of-concept testing of the ICA method, Hessian calculation and the IMC simulations for the airfoil code. The three principal modes of perturbation introduced are

- thickness,
- angle of attack, and
- leading edge shape.

The thickness perturbation is introduced using

$$\delta y = 0.04\ t\ x^b (1 - x)^c\,, \tag{B.3}$$

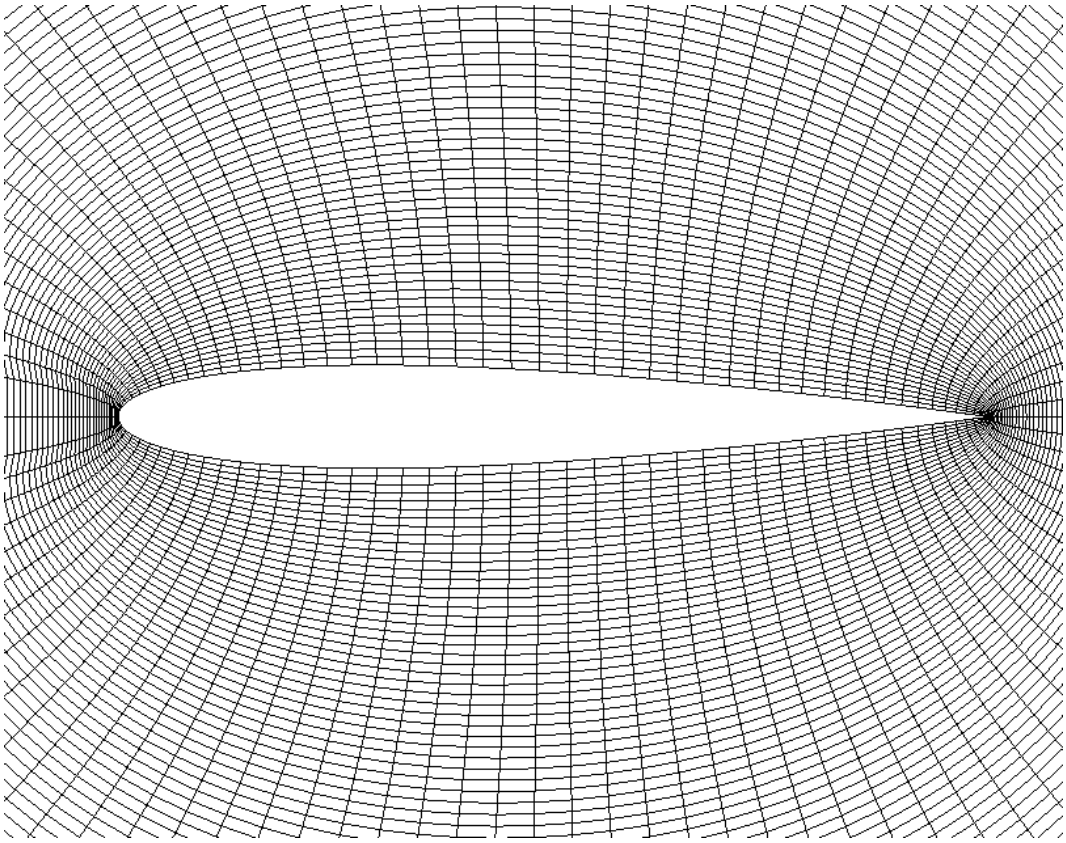

Figure B.I Sample grid for NACA0012 with $\alpha = 0$ and 99×100 grid points

where $b = 3$, $c = 3$ and t is a Normal random variable with zero mean and unit variance. This ensures that the thickness perturbation at one standard deviation is equal to 4% of the chord length. The coefficients b and c control the variation of the thickness perturbation along the chord.

The angle of attack perturbation is introduced in the form of a rotation of the airfoil about the trailing edge. A variation of $\pm 0.1^o$ is introduced with Normal distribution.

The leading edge shape perturbation is introduced using

$$\delta y = 0.02\ l\ x^d (1-x)^e \tag{B.4}$$

where $d = 0.5$, $e = 6$ and l is a random variable with beta distribution. Coefficients for the distribution are selected in such a way as to introduce a bias towards leading edge sharpness. Though all the errors introduced are artificial, they represent the actual geometric errors for engine blades [30]. Figure B.II shows the various airfoil surfaces generated during the Monte Carlo simulations.

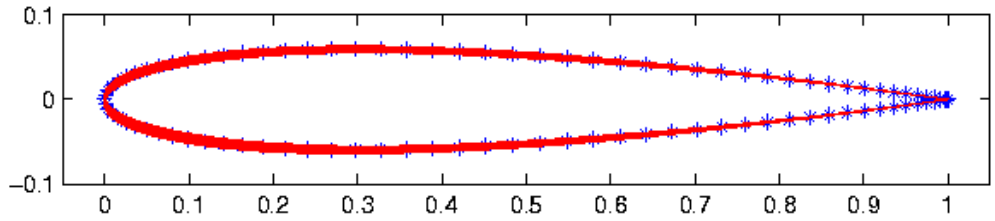

Figure B.II Sample airfoils generated with the artificial uncertainty

Though Figure B.II gives a sense of the magnitude of actual perturbations, it is not clear how the three modes of perturbations look like. Figure B.III shows the three modal perturbations for $\pm 10\sigma$.

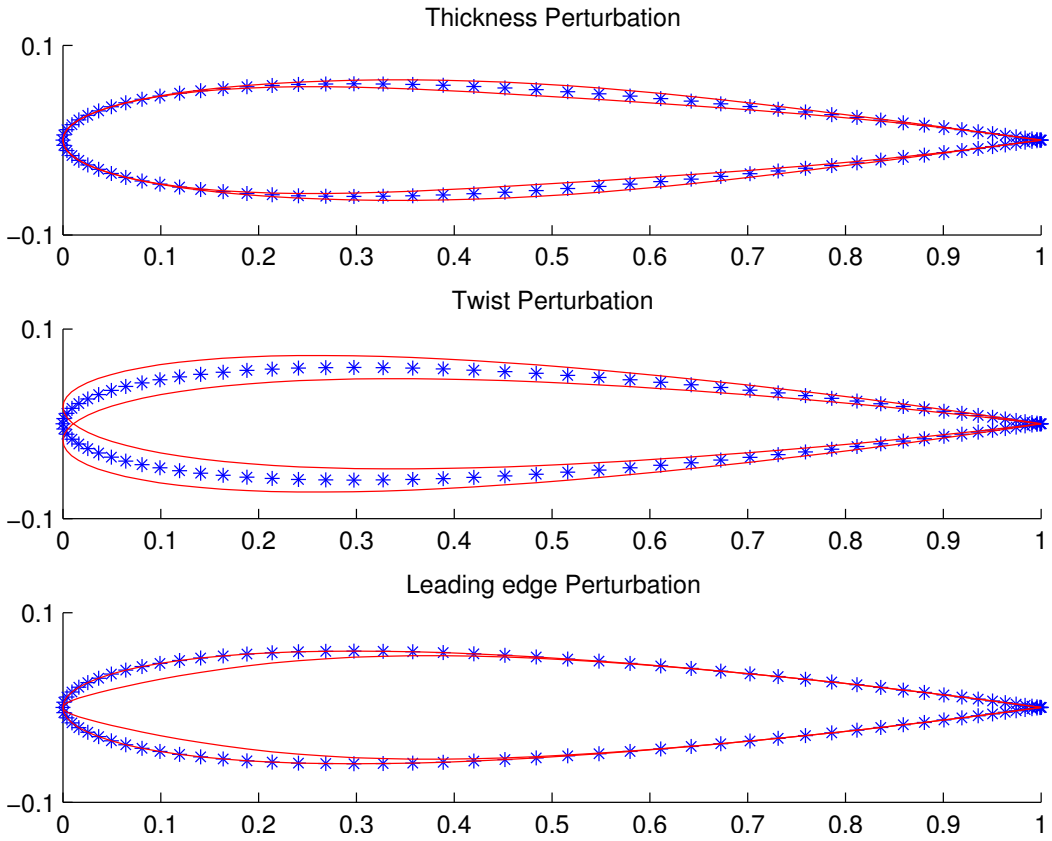

Figure B.III Three principle perturbations corresponding to $\pm 10\,\sigma$

BIBLIOGRAPHY

[1] S. Amari, A. Cichocki and H. H. Yang. A new learning algorithm for blind signal separation. In M. C. Mozer D. S. Touretzky and M. E. Hasselmo, editors, *Advances in Neural Iinformation Processing Systems, 8*, 1996. MIT Press.

[2] W. K. Anderson and D. L. Bonhaus. Airfoil design on unstructured grids for turbulent flows. *AIAA Journal*, 37(2):185-191, 1999.

[3] I. Babuška and A. Miller. The post-processing approach in the finite element method - Part 1: Calculation of displacements, stresses and other higher derivatives of the displacements. *International Journal of Numerical Methods in Engineering*, 20:1085-1109, 1984a.

[4] I. Babuška and A. Miller. The post-processing approach in the finite element method - Part 2: The Calculation of stress intensity factors. *International Journal of Numerical Methods in Engineering*, 20:1111-1129, 1984b.

[5] F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1-48, 2002.

[6] A. Benveniste, M. Goursat and G. Ruget. Robust identification of a nonminimum plase system: blind adjustment of a linear equalizer in data communications. *IEEE Transactions on Automatic Control*, 25(3):385-399, 1980.

[7] C. Bischof, A. Carle, P. Khademi and A. Mauer. ADIFOR 2.0: Automatic Differentiation for FORTRAN 77 programs. *IEEE Computational Science and Engineering*, 3(3):18-31, 1996.

[8] J. Brown, J. Slater and R. Grandhi. Probabilistic analysis of geometric uncertainty effects of blade model response. In *Proceedings of ASME Turbo Expo*, pages 247-255, Atlanta, Georgia, USA, 2003.

[9] H. M. Bücker, A. Rasch and A. Vehreschild. Automatic generation of parallel code for hessian computations. In *Proceedings of the 2005 and 2006 International inproceedings of OpenMP shared memory parallel programming*, pages 372-381, 2008. Springer-Verlag Berlin, Heidelberg, Germany.

[10] T. Bui-Thanh, K. Willcox and O. Ghattas. Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications. *AIAA Journal*, 46(10):2520-2529, 2008.

[11] D. A. Burgess, P. I. Crumpton and Mike Giles. A parallel framework for unstructured grid solvers. *Programming Environments of Massively Parallel Distributed Systems*, pages 97-106, 1994.

[12] Sergio Campobasso and Mike Giles. Effect of flow instabilities on the linear analysis of turbomachinery aeroelasticity. *AIAA Journal of Propulsion and Power*, 19(2), 2003.

[13] Sergio Campobasso and Mike Giles. Stabilization of linear flow solver for turbomachinery aeroelasticity by means of recursive projection method. *AIAA Journal*, 42(9):1765-1774, 2004.

[14] M. Cardew-Hall, J. Cosmas and M. Ristic. Automated proof inspection of turbine blades. *The International Journal of Advanced Manufacturing Technology*, 3:67-68, 1988.

[15] Jean-Fran Cardoso. Infomax and maximum likelihood for blind source separation. *IEEE Signal processing letters*, 4:112-114, 1997.

[16] J. F. Cardoso. High order contrasts for independent component analysis. *Neural Computation*, 11(1):157-192, 1999.

[17] Faidon Christakopoulos. *Sensitivity computation and shape optimisation in aerodynamics using the adjoint methodology and Automatic Differentiation.* PhD thesis, School of Engineering and Materials Science, Queen Mary University of London, 2012.

[18] Faidon Christakopoulos, Dominic Jones and Jens-Dominik Müller. Pseudo timestepping and verification for automatic differentiation derived CFD codes. *Computers & Fluids*, 46(1):174 - 179, 2011. 10th ICFD Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010).

[19] Bruce Christianson. Automatic hessians by reverse accumulation. *IMA Journal of Numerical Analysis*, 12:135-150, 1992.

[20] B. Christianson. Reverse accumulation and implict functions. *Optimization Methods and Software*, 9(4):307-322, 1998.

[21] P. Comon. Independent component analysis - a new concept?. *Signal Processing*, 36:287-314, 1994.

[22] Pierre Comon and Christian Jutten. *Handbook of Blind Source Separation.* Academic Press, 2010.

[23] F. Courty, A. Dervieux, B. Koobus and L. Hascoët. Reverse automatic differentiation for optimum design: from adjoint state assembly to gradient computation. *Optimization Methods and Software*, 18(5):615-627, 2003a.

[24] F. Courty, A. Dervieux, B. Koobus and L. Hascoet. Reverse automatic differentiation for optimum design: from adjoint state assembly to gradient computation. *Optimisation Methods and SoftwareOptimisation Methods and Software*, 18(5):615-627, 2003b.

[25] Dan DeLaurentis and Dimitri Mavris. Uncertainty modeling and management in multidisciplinary analysis and synthesis. In *38th AIAA Aerospace Sciences Meeting and Exhibit*, pages 1-12, 2000. AIAA-2000-0422.

[26] Mihai C. Duta. *The use of the Adjoint method for the minisation of forced vibration in Turbomachinery.* PhD thesis, Numerical Analysis Group, Computing Laboratory, 2002.

[27] J. Elliott and J. Peraire. Practical 3D aerodynamic design and optimization using unstructured meshes. *AIAA Journal*, 35(9):1479-1485, 1997.

[28] M.J. Evans and J.S. Rosenthal. *Probability and Statistics: The Science of Uncertainty.* W. H. Freeman, 2004.

[29] M. Fagan and A. Carle. Adifor 3.0 overview. Technical Report CAAM-TR00-03, Department of Computational and Applied Mathematics, Rice University, 2000.

[30] Victor Garzon. *Probabilistic Aerothermal Design of Compressor Airfoils.* PhD thesis, Dept. of Aeronautics and Astronautics, MIT, 2002.

[31] Victor E. Garzon and David L. Darmofal. Impact of geometric variability on axial compressor performance. *Journal of Turbomachinery*, 125(4):692-703, 2003.

[32] Devendra Ghate and Michael B. Giles. Efficient hessian calculation using automatic differentiation. In *25th AIAA Applied Aerodynamic Conference*, 2007.

[33] R. Giering and T. Kaminski. Recipes for adjoint code construction. *ACM Transactions on Mathamatical Software*, 24(4):437-474, 1998.

[34] Michael Giles, Mats G. Larson, J. Mårtsen Levenstam and Endre Süli. Adaptive error control for finite element approximations of the lift and drag coefficients in viscous flow. Technical Report, Oxford University Computing Laboratory, 1997. NA-97/06.

[35] Michael B. Giles. On the use of Runge-Kutta time-marching and multigrid for the solution of adjoint equations. Technical Report, Computing Laboratory, University of Oxford, 2000. Report No NA00/10.

[36] Michael B. Giles. *Discrete adjoint approximations with shocks.* Hyperbolic Problems: Theory, Numerics, Applications, editors T. Hou and E. Tadmor, Springer-Verlag, 2003.

[37] Michael B. Giles, M. C. Duta, J. D. Muller and N.A. Pierce. Algorithm developments for discrete adjoint methods. *AIAA Journal*, 42(2):198-205, 2003.

[38] Michael B. Giles, Devendra Ghate and Mihai Duta. Using automatic differentiation for adjoint CFD code development. In *Indo-French Workshop*, 2005. Also available as NA05/25.

[39] M. B. Giles and N. A. Pierce. Improved lift and drag estimates using adjoint Euler equations. In *14^th AIAA Ccomputational Fluid Dynamics Conference*, 1999. AIAA-1999-3293.

[40] Michael B. Giles and N. A. Pierce. Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM Review*, 42(2):247-264, 2000b.

[41] Michael B. Giles and N. A. Pierce. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 65(3-4):393-415, 2000a.

[42] M. B. Giles and N. A. Pierce. Adjoint error correction for integral outputs. In *Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics, Lecture Notes in Computational Science and Engineering*, pages 47-96, 2002. Springer-Verlag.

[43] M. B. Giles, N. A. Pierce and Endre Suli. Progress in adjoint error correction for integral functionals. *Computing and Visualisation in Science*, 6:2-3, 2004.

[44] M. B. Giles and Endre Suli. Adjoint methods for pdes: a posteriori error analysis and postprocessing by duality. *Acta Numerica*, pages 145-236, 2002.

[45] Andreas Griewank. *Evaluating Derivatives.* SIAM, Frontiers in Applied Mathematics, 2000.

[46] A. Griewank, D. Juedes, H. Mitev, J. Utke and O. Vogel et al.. ADOL-C: a package for the automatic differentiation of algorithms written in C/C++. *ACM TOMS*, 22(2):131-167, 1996a.

[47] A. Griewank, D. Juedes, H. Mitev, J. Utke and O. Vogel et al.. ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software*, 22(2):131-167, 1996b.

[48] C. R. Gumbert, P. A. Newman and G. J.-W. Hou. Effect of random geometric uncertainty on the computational design of 3-D wing. In *AIAA Applied Aerodynamics Conference*, 2002. AIAA-2002-2806.

[49] Jens Häcker. Statistical analysis of manufacturing deviations and classification methods for probabilistic aerothermal design of turbine blades. Master's thesis, Department of Aeronautics and Astronautics, Universität Stuttgart, 2000.

[50] Omer L. Hageniers. Recent advances in laser triangulation-based measurement of airfoil surfaces. In *Proceedings of SPIE - The International Society for Optical Engineering*, pages 2-22, Bellingham, WA, USA, 1995a.

[51] Laurent Hascoët and V. Pascual. Tapenade 2.1 user's guide. Technical Report 0300, INRIA, 2004.

[52] A. Hyvärinen. New approximations of differential entropy for independent component analysis and projection pursuit. *Advances in Neural Information Processing Systems*, 10:273-279, 1998.

[53] Aapo Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *Neural Networks, IEEE Transactions on*, 10(3):626–634, 1999.

[54] A. Hyvärinen, J. Karhunen and E. Oja. *Independent Component Analysis*. John Wiley and Sons, 2001.

[55] A. Hyvärinen and E. Oja. A fast fixed point algorithm for independent component analysis. *Neural Computation*, 9(7):1483-1492, 1997.

[56] Antony Jameson. Aerodynamic design via control theory. *Journal of Scientific Computation*, 3:233-260, 1998.

[57] I. T. Jollife. *Principal Component Analysis*. Springer, 2002.

[58] Dominic Jones, Jens-Dominik Müller and Faidon Christakopoulos. Preparation and assembly of discrete adjoint CFD codes. *Computers & Fluids*, 46(1):282 - 286, 2011. 10th ICFD Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010).

[59] Apurva Kumar. *Robust Design Methodologies: Application to Compressor Blades*. PhD thesis, Faculty of Engineering Science and Mathematics, University of Southampton, United Kingdom, 2006.

[60] Caroline Marie Lamb. Probabilistic performance-based geometric tolerancing of compressor blades. Master's thesis, Massachusetts Institute of Technology, 2005.

[61] E. Learned-Miller and J. W. Fisher III. ICA using spacing estimates of entropy. *Journal of Machine Learning Research (JMLR)*, 4:1271-1295, 2003.

[62] W. Li, F. Gu, A. D. Ball, A. Y. T. Leung and C. E. Phipps. A study of the noice from diesel engines using the independent component analysis. *Mechanical Systems and Signal Processing*, 15(6):1165-1184, 2001.

[63] J. M. Luckring, M. J. Hemsch and J. H. Morrison. Uncertainty in computational aerodynamics. In *AIAA Aerospace Sciences Meeting*, 2003. AIAA-2003-0409.

[64] D. Mantini, K. E. Hild II, G. Alleva and S. Comani. Performance comparison of independent component analysis algorithms for fetal cardiac signal reconstruction: a study on synthetic FMCG data. *Physics in Medicine and Biology*, 51:1033-1046, 2006.

[65] George Mantis. *Quantification and Propagation of Disciplinary Uncertainty via Bayesian Statistics*. PhD thesis, Georgia Institute of Technology, USA, 2002.

[66] Massimiliano Martinelli, Alain Dervieux and Laurent Hascoët. Strategies for computing second-order derivatives in CFD design problems. In *West-East High Speed Flow Field Conference*, 2007.

[67] M. Martinelli and R. Duvigneau. On the use of second-order derivatives and metamodel-based monte-carlo for uncertainty estimation in aerodynamics. *Computers and Fluids*, 39(6), 2010.

[68] J. R. R. A. Martins, I. M. Kroo and J. J. Alonso. A method for sensitivity analysis using complex variables. In *AIAA Aerospace Sciences Meeting & Exhibit*, 2000. AIAA-2000-0689.

[69] Joaquim R. R. A. Martins, Peter Sturdza and Juan J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, 2003.

[70] Lionel Mathelin, M. Y. Hussaini, T. A. Zang and Francoise Bataille. Uncertainty propagation for turbulent, compressible flow in a quasi-1D nozzle using stochastic methods. In *AIAA Computational Fluid Dynamics Conference*, 2003. AIAA-2003-4240.

[71] B. Mohammadi and O. Pironneau. Mesh adaption and automatic differentiation in a CAD-free framework for optimal shape design. *International Journal of Numerical Methods in Fluids*, 30(2):127-136, 1999.

[72] Pierre Moinier. *Algorithmic developments for an unstructured viscous flow solver*. PhD thesis, Numerical Analysis Group, Computing Laboratory, 1999.

[73] P. Moinier, J.-D. Muller and Mike Giles. Edge-based multigrid and preconditioning for hybrid grids. *AIAA Journal*, 40(10):1954-1960, 2002.

[74] J.-D. Müller and P. Cusdin. On the performance of discrete adjoint CFD codes using automatic differentiation. *International Journal for Numerical Methods in Fluids*, 47(8-9):939–945, 2005.

[75] Mattia Padulo, M. Sergio Campobasso and Marin D. Guenov. Novel uncertainty propagation method for robust aerodynamic design. *AIAA Journal*, 40(3):530-543, 2011.

[76] D. I. Papadimitriou and K. C. Giannkoglou. Aerodynamic design using the truncated newton algorithm and the continuous adjoint approach. *International Journal for Numerical Methods in Fluids*, 68:724-739, 2012.

[77] Jacques E.V. Peter and Richard P. Dwight. Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers and Fluids*, 39(3):373-391, 2010.

[78] N. A. Pierce. *Preconditioned multigrid methods for compressible flwo calculations on stretched meshes*. PhD thesis, Numerical Analysis Group, Computing Laboratory, 1997.

[79] N. A. Pierce and M. B. Giles. Adjoint and defect error bounding and correction for functional estimates. *Journal of Computational Physics*, 200:769-794, 2004.

[80] R. W. Preisendorfer. *Principal Component Analysis in Meteorology and Oceanography*. Elsevier, 1998.

[81] M. M. Putko, P. A. Newmann, A. C. Taylor III and L. L. Green. Approach for uncertainty propagation and robust design in CFD using sensitivity derivatives. In *AIAA CFD Conference*, 2001. AIAA-2001-2528.

[82] S. Ravindran. A reduced-order approach for optimal control of fluids using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids*, 34:425-448, 2000.

[83] R. D. Rosemau, S. Nawaz, A. Niu and W. G. Wee. Aircraft engine blade cooling hole detection and classification from infrared images. In *SPIE Conference on Nondestructive Evaluation of Aging Aircraft, Airports, and Aerospace Hardware III*, 1999.

[84] Markus P. Rumpfkeil and Dimitri J. Mavriplis. Efficient hessian calculations using automatic differentiation and the adjoint method. In *40th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2010.

[85] Laura L. Sherman, Arthur C. Taylor III, Larry L. Green and Perry A. Newman. First- and second-order aerodynamic sensitivity derivatives via automatic differentiation with incremental iterative methods. *Journal of Computational Physics*, 129:307-331, 1996.

[86] A. Sinha, B. Hall, B. Cassenti and G. Hilbert. Vibratory parameters of blades from coordinate measurement machine (CMM) data. In *Proceedings of 9th National Turbine Engine High Cycle Fatigue(HCF) Conference*, 2004.

[87] W. Squire and G. Trapp. Using complex variables to estimate derivatives of real functions. *SIAM Review*, 10(1):110-112, 1998.

[88] A. C. Taylor III, L. L. Green, P. A. Newmann and M. M. Putko. Some advanced concepts in discrete aerodynamic sensitivity analysis. In *AIAA CFD Conference*, 2001. AIAA-2001-2529.

[89] Nikita Thakur, A. J. Keane and P. B. Nair. Estimating the effect of Manufacturing Variability on Turbine Blade Life. In *Proceedings of 4th International Workshop on Reliable Engineering Computing (REC 2010)*, pages 311-322, 2010a.

[90] Nikita Thakur, A. J. Keane, P. B. Nair and A. Rao. Probabilistic life assessment of gas turbine blades. *Journal of Mechanical Design*, 132(12):121005–[9pp], 2010b.

[91] Petr Tichavský and Zbyněk Koldovský. Fast and Accurate Methods of Independent Component Analysis. *Kybernetika*, 47(3):426-438, 2011.

[92] Jean Utke, Uwe Naumann and Andrew Lyons. OpenAD/F: User Manual. Technical Report, Mathematics and Computer Science Division, Argonne National Laboratory, 2012.

[93] R. W. Walters and Luc Huyse. Uncertainty analysis for fluid mechanics with applications. Technical Report, ICASE, NASA Langley, 2002. ICASE-2002-1.

[94] Qiqi Wang. *Uncertainty Quantification for Unsteady Fluid Flow using Adjoint-based Approaches*. PhD thesis, The Institute for Computational and Mathematical Engineering, Stanford University, 2008.

[95] Qiqi Wang, Parviz Moin and Gianluca Iaccarino. A rational interpolation scheme with superpolynomial rate of convergence. *SIAM J. Numerical Analysis*, 47(6):4073-4097, 2010.

[96] W. Wang, L. Han, X. Wu and W. G. Wee. Three dimensional CT data processing for inspection. In *Proceedings of SPIE - The International Society for Optical Engineering*, pages 98-109, 1999a.

[97] Dongbin Xiu and George Em Karniadakis. Modeling uncertainty in flow simulations via generalized polynomial chaos. *Journal of Computational Physics*, 187:137-167, 2003.

[98] C. Zang, M. I. Friswell and M. Imregun. Structural damage detection using independent component analysis. *Structural Health Monitoring*, 3(1):69-83, 2004.

[99] T. Zervogiannis, D.I. Papadimitriou and K.C. Giannakoglou. Total pressure losses minimization in turbomachinery cascades using the exact hessian. *Computer Methods in Applied Mechanics and Engineering*, 199(41–44):2697-2708, 2010.

[100] Feng Zhang, Bani Mallick and Zhujun Weng. A bayesian method for identifying independent sources of non-random spatial patterns. *Statistics and Computing*, 15:329-339, 2005.