Gradient free optimisation in selected engineering applications.

Walton, Sean Peter

Swansea University
Prifysgol Abertawe

Gradient Free Optimisation in Selected Engineering Applications

Sean Peter Walton

Submitted to Swansea University in fulfilment
of the requirements for the degree of Doctor of Philosophy

College of Engineering

2013

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Declaration of Authorship

I, Sean Walton, declare that this thesis titled, 'Gradient Free Optimisation in Selected Engineering Applications' and the work presented in it are my own. I confirm that:

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed: _____

Date: 31/05/2013

This thesis is the result of my own investigations, except where otherwise stated. Where correction services have been used, the extent and nature of the correction is clearly marked in a footnote(s).

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed: _____

Date: 31/05/2013

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed: _____

Date: 31/05/2013

"The Gods give with one hand and take with the other."

George R.R. Martin

SWANSEA UNIVERSITY

Abstract

College of Engineering

Doctor of Philosophy

by Sean P. Walton

The aim of this thesis was to investigate the application of gradient free optimisation algorithms to a number of practical engineering applications. A modified cuckoo search algorithm was developed which outperformed a number of other optimisation algorithms in a variety of benchmark cases. The algorithm was applied to two selected practical examples. The first was that of aerodynamic shape optimisation, where the algorithm performed well when applied to both inverse design and aerofoil improvement problems. A limitation of gradient free algorithms is poor efficiency, in terms of number of function evaluations. When a single function evaluation is computationally expensive, this presents a problem. In an attempt to address this an interpolation scheme based on proper orthogonal decomposition was developed for unsteady fluid flow problems. This approach resulted in a significant reduction in the CPU time required to find new solutions, without a significant loss in accuracy. Modified cuckoo search was also applied to mesh optimisation. For this application reduced order mesh optimisation, was introduced. It was applied to the difficult problem of optimising meshes for use in co–volume techniques. Proper orthogonal decomposition was used to reduce the number of dimensions in the problem. This reduction allowed the expression of the optimisation problem globally, rather than locally as in other techniques. This technique performed better than other existing techniques at improving mesh quality in both two and three dimensional examples.

Acknowledgements

I would firstly like to thank my supervisors Prof. Oubay Hassan and Prof. Kenneth Morgan. Their support and guidance during my PhD has been invaluable. I cannot begin to list what I have learnt under their supervision.

I would also like to thank all my colleagues and friends in the College of Engineering, Swansea University. I thank Dr. Rowan Brown for helping me gain a better understanding of the field of optimisation. I also especially thank Rubén Sevilla, Ben Evans and fellow PhD students Rachel Pelley, Hannah Buckland and Bruce Jones, without whom, the last three years would have been much more difficult.

I acknowledge the Engineering and Physical Sciences Research Council for funding the project.

Contents

Abbreviations

CFD	**C**omputational **F**luid **D**ynamics
ROM	**R**educed **O**rder **M**odelling
CS	**C**uckoo **S**earch
MCS	**M**odified **C**uckoo **S**earch
POD	**P**roper **O**rthogonal **D**ecomposition
AI	**A**rtificial **I**ntelligence
GA	**G**enetic **A**lgorithm
DE	**D**ifferential **E**volution
PSO	**P**article **S**warm **O**ptimisation
NFL	**N**o **F**ree **L**unch
LHS	**L**atin **H**yper–cube **S**ampling
CVT	**C**entroidal **V**oronoi **T**essellation
EA	**E**volutionary **A**lgorithm
NURBS	**N**on-**U**niform **R**ational **B**-**S**plines
ALE	**A**rbitrary **L**agrangian–**E**ulerian
FOM	**F**ull **O**rder **M**odel
SVD	**S**ingular **V**alued **D**ecomposition
MAC	**M**arker **a**nd **C**ell

Dedicated to my parents and wife, Jo, for their continued love, support and patience throughout.

In memory of Dr Tudor Jenkins who inspired, and encouraged, me to pursure a career in science

Chapter 1

Introduction

1.1 General

Design is at the core of engineering. Many different definitions of the word design exist. The main focus of this thesis was the verbal meaning of the word design, i.e. the process of design. Many fields in engineering attempt to improve some part of this core process.

Design processes all begin with two main questions: what is it that the final design needs to achieve and what constraints are there on the design? A designer's role is to best meet the goals given by the first question, whilst satisfying the constraints identified in the second. This is often achieved during design cycles.

Before a design cycle can take place, an initial design is required. This initial design will almost certainly be formed using a designer's own experience and training, or perhaps even simply utilising a previous design from another project. Once an initial design is constructed, it will need to be analysed. Historically, this analysis is achieved through physical experiments. Such an example is the invention of the first commercially practical light bulb by Thomas Edison. Lacking the required theoretical background Edison undertook a trial and error search for the best material for the filament. This type of trial and error is sometimes called the Edisonian approach.

Using the evidence obtained through this analysis, the original design can be refined to better meet performance goals and constraints. As scientific theories became more sophisticated, it was increasingly possible to predict the performance of different designs using hand calculations. However, this is a laborious process and the models used would often contain approximations to make hand calculations possible.

To simplify the process of refining a design, parameterisations are often employed. This allows the size and shape of a design to be represented by a series of parameters. For

instance, a simple beam may be parameterised in terms of length, width and height. The goal of the design cycle is therefore to find a set of these parameters which result in the best performance whilst meeting the constraints. The designer must parameterise the design in an efficient and logical manner to make the process easier.

The initial design represents a first guess of the best set of parameters for the problem. The cycle of testing a design at a given set of parameters, readjusting the parameters, then retesting, could be repeated many times. As problems become more complex, with more parameters, a designer will want to keep a design in this cycle longer. The limiting factor in how long a design cycle can last, in reality, are deadlines specified by contractors, potential customers or competitors. The designer has no control over this time period, so the field of engineering seeks to make the design cycle increasingly efficient.

Computer simulation is making a measurable impact on the efficiency of design cycles across multiple disciplines. Sophisticated computer models now allow the construction of virtual experiments for many different potential designs. The primary example considered in this work was that of computational fluid dynamics (CFD). Before computational power and model validation made CFD a practical and reliable tool, wind tunnel testing and hand calculations were used to test designs in aerospace engineering. Wind tunnel testing requires expensive experimental equipment, with new models needed for each new geometry. This results in a very limited number of design cycles before a deadline is reached.

In 1980, Boeing constructed 77 different model wings for wind tunnel testing whilst developing the 767. This is compared to 11 wings for the development of the 787 in 2005. The reduction is due to advancements in CFD [4]. Compared to experimental and hand calculation analysis, computer simulations allow the, relatively, fast and cheap analysis of potential designs. Computer aided design applications also allow the automatic parametrisation and generation of new design geometries. The designer is faced with the task of using this potentially endless supply of data efficiently and effectively.

The requirements and constraints of a problem can first be used to define some kind of quality measure of a design. One example might be the lift to drag ratio of an aerofoil. Once the quality measure is defined, a computer simulation can be run for a set of sample designs, to obtain the quality measure for each. If only a single parameter defines the design, the quality measure can simply be plotted against this parameter on a simple line graph. By simply inspecting this graph this designer can determine the optimum design. When two parameters exist, the quality measure will have to be plotted as a contour map or a surface, which the designer may inspect. With three or more parameters it becomes difficult, if not impossible, to visualise and understand this data.

As the number of parameters increase, it becomes increasingly difficult to determine the optimum design by hand. Doing so relies on the intuition and experience of a skilled designer. As technology and development pushes the boundaries of current knowledge, this experience and intuition becomes less important. For example, in the design of micro air vehicles on the scale 15cm and below, classical aerodynamic design theory does not apply [5]. This could result in unexpected shapes representing optimum designs. In addition, the time taken to comprehend the effect of each parameter on the performance of the design will limit the number of possible design cycles.

The methods presented in this thesis all aimed to automate this searching process, while increasing its efficiency. There were two methods considered for increasing the efficiency of the process. The first was to reduce the number of sample designs needed to obtain the optimum design, this is the aim of the field of optimisation. The second was to reduce the time taken to obtain the quality measure for a given design, which is the role of reduced order modelling (ROM).

1.2 Objectives

The initial objective of this work was to apply reduced order fluid flow modelling and optimisation techniques to aerospace design problems. The goal was to develop techniques which would be capable of making large design changes. It was recognised that many local optimisation techniques exist which can make significant improvements to initial designs. However, local optimisation techniques are sensitive to the initial design supplied. The aim here was to develop techniques which are not sensitive to an initial design. An additional aim, was to write algorithms which were as general and automatic as possible.

To achieve this aim, an optimisation technique capable of global optimisation needed to be identified. In addition to this, a ROM technique was required to reduce the computational cost of numerical models used by the optimiser. Such a ROM technique would need to be able to predict responses to significant changes in the geometry and flow regime.

1.3 Layout of the Thesis

Chapter 2 provides a brief overview of optimisation. The concepts of local and global optimisation are discussed. This resulted in a motivation to use metaheuristic gradient free optimisation methods. The Cuckoo Search (CS) algorithm was selected as the

optimisation algorithm for this work. The algorithm and motivations for selecting it, are discussed in Section 2.4.1.4.

Chapter 3 discusses the modifications made to the CS algorithm. These modifications which became known as Modified Cuckoo Search (MCS) [1], are detailed. The modifications result in an algorithm which can minimise a function in a low number of evaluations, compared to other metaheuristic gradient free methods.

Further changes, discussed in Section 3.4, were made to the published MCS algorithm. The motivation for these changes originate from the results presented by Bhargava et al [6], who applied MCS to phase equilibrium modelling.

The MCS algorithm was applied to the problem of aerofoil shape optimisation, which is presented in Chapter 4. It was found that the method could improve the design of a given aerofoil. However the high CPU cost of evaluating the objective function would make the method impractical for use in an industrial setting. This led to an investigation into ROM techniques.

An overview of ROM is given in Chapter 5. A ROM technique based on Proper Orthogonal Decomposition (POD) and interpolation methods was deemed the most suitable for optimisation purposes. The existing method as it applies to steady fluid flow problems, discussed in Section 5.4, was extended, successfully, to unsteady flow problems [2]. The resulting unsteady ROM technique is presented in Section 5.5.

The ROM techniques considered are not suited to predicting changing flow regimes, which makes them unsuitable for general aerodynamic shape optimisation. However, POD was used to reduce the dimension of mesh optimisation problems. When coupled with MCS, the technique, Reduced Order Mesh Optimisation [3], is found to be very successful in otherwise difficult mesh optimisation problems. The reduced order mesh optimisation algorithm is presented in Chapter 6.

Finally the general findings of this thesis are discussed in Chapter 7, along with areas for future work.

Chapter 2

Overview of Optimisation

optimum

n. The most favourable or advantageous condition, value, or amount

adj. Best, most favourable, esp. under a particular set of circumstances

Oxford English Dictionary

2.1 Introduction

In Section 1.1, the design cycle was discussed. Typically, a design cycle is a series of decisions made by a designer or team of designers, based on experience and intuition. Decisions need to be made on which parameter of the design to adjust, and by how much. Another decision is when to stop making changes and finalise the design. The field of optimisation involves developing techniques to automatically make these kinds of decisions.

Modern optimisation techniques originate from methods developed to find roots of polynomials, or to solve systems of equations. One such example is the conjugate gradient method, which was designed to solve a system of linear equations [7]. The solution is found by minimising the residual of the system, which means that if a problem could be framed as a minimisation problem then this technique can potentially be applied. An early example of such an application is the minimisation of an energy function to numerically determine the electronic structure of molecules [8].

Computers were the key to the development of optimisation techniques. Little was known about how to approach the numerical optimisation of many variable functions before

1940. In the 1940s and 1950s, a new branch of research, named linear programming, was introduced. Essentially, this was the birth of linear optimisation. The kind of functions these early techniques could be applied to was very restricted. In the 1960s, it was still considered quite a challenge to optimise functions with 10 variables. Modern techniques are designed to optimise functions with hundreds of variables [9].

2.2 Fundamentals

Let $f(\mathbf{x})$ represent some measure of quality or performance of interest. The aim of optimisation is to find the set of inputs, or parameters in the case of design optimisation, $\mathbf{x}$ that either maximises or minimises the output of this function. The inputs of the function may be represented over any number of dimensions or degrees of freedom. The space on which $\mathbf{x}$ is represented is often referred to as the search or parameter space. Optimisation algorithms traverse the search space to find the optimum set of parameters by evaluating the objective function at different coordinates. The evaluation of the objective function, $f(\mathbf{x})$, may be expensive so it is desirable to minimise the number of evaluations required to find optimum value.

2.2.1 Categorising Optimisation Problems

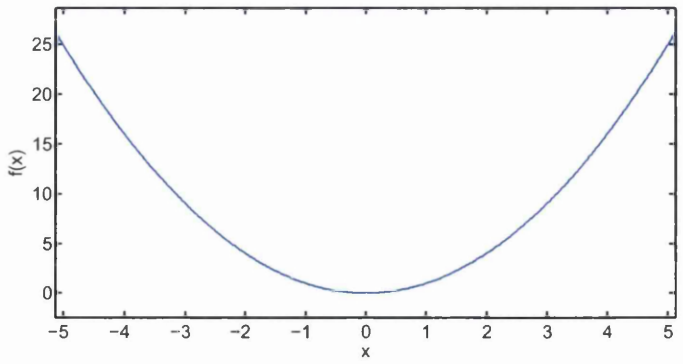
A wide range of optimisation methods exist. To select a suitable method, for a particular objective function, methods of classifying objective functions are required. In this section, a number of terms used to describe objective functions and optimisation problems are explained.

2.2.1.1 Smoothness

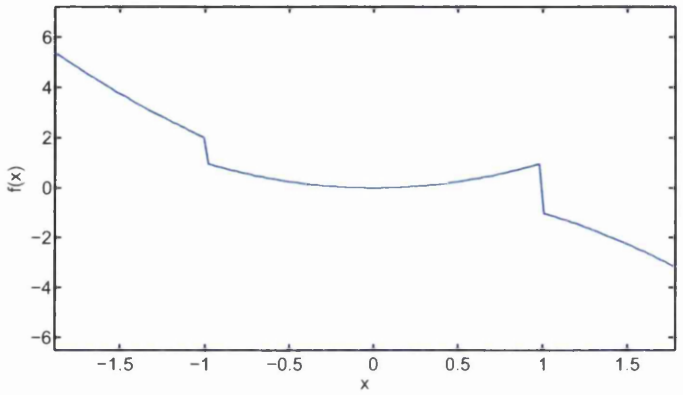
The de Jong function

$$f(\mathbf{x}) = \sum_{i=1}^{d} x_i^2 \qquad x_i \in [-5.12, 5.12] \tag{2.1}$$

with $d = 1$ is plotted in Figure 2.1(a). This is an example of a smooth objective function. Smooth functions have continuous second derivatives. Conversely, non–smooth functions do not have second derivatives and may contain discontinuities [10]. An example of a non–smooth function is plotted in Figure 2.1(b). Non–smooth functions provide a challenge, in particular, for optimisation techniques which require gradient calculations.



(a) Graph of de Jong's function showing an example of a smooth function



(b) Example of a non–smooth function

FIGURE 2.1: Smooth and non-smooth functions

2.2.1.2 Local–minima

The concept of local–minima is important in optimisation. Optimisation algorithms can be considered either local or global minimisers [11]. This is illustrated in Figure 2.2, which shows a plot of Schwefel's function given by

$$f(\mathbf{x}) = \sum_{i=1}^{d} \left[-x_i \sin(\sqrt{|x_i|}) \right] \qquad x_i \in [-500, 500] \tag{2.2}$$

where $d = 1$. Multi–modal functions have many local–minima, whereas uni–modal functions do not. Optimisation algorithms which determine search direction using gradients may become trapped in a local–minima, depending on the initial guess of the input parameters. This leads to the concept of local vs. global minimisers.

Local minimisers depend critically on the initial guess of the input parameters. Such optimisation algorithms will only converge to the global minimum, if the initial guess is close to it, otherwise they converge to the local–minimum closest to the initial guess.

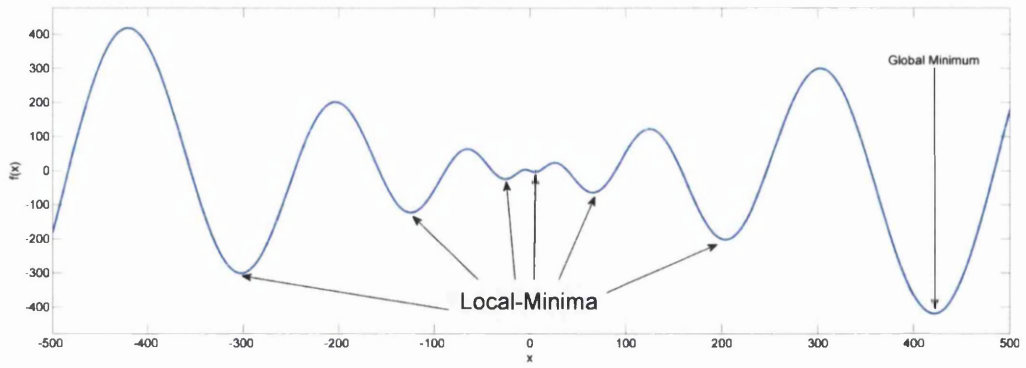


FIGURE 2.2: Graph of Schwefel's function which is an example of a multi–modal function

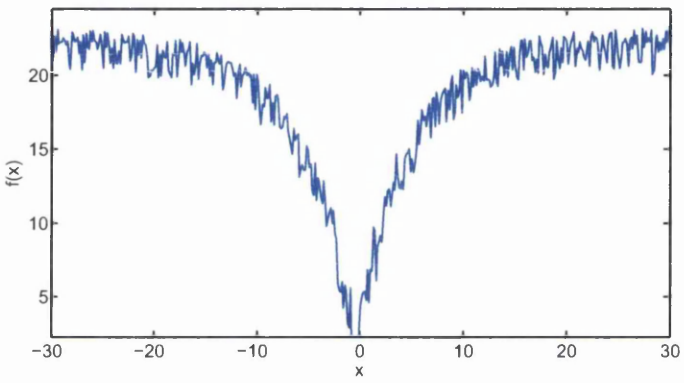


FIGURE 2.3: Example of a noisy function

True global minimisers, however, will find the global minimum in every case, regardless of the initial guess. In practice locating the global minimum is difficult, as is recognising it when it is found [10].

2.2.1.3 Noise

Figure 2.3 shows an example of a noisy function. These types of objective functions are often found when optimising a function based on data from either physical or numerical experiments. The noise is likely to represent experimental or numerical error. Noise will result in a function which is both non–smooth and multi–modal and, therefore, the difficulties discussed for those classes of functions apply here. A key strategy in dealing with noise is to attempt to find an underlying function and remove the noise.

2.2.1.4 Continuous and Discrete

All the objective functions plotted above are examples of continuous optimisation problems. The input variables have continuous values within a specified range. Some problems, however, contain inputs which have discrete states. For these problems, discrete optimisation strategies are required.

A defining feature of discrete optimisation problems is that the input parameters are drawn from a large, but finite, set. In continuous optimisation, the set of feasible input parameters can be considered infinite [10]. All applications considered in this thesis fell into the continuous category.

2.2.1.5 Constrained and Unconstrained

In unconstrained problems, the input parameters can take any value [9]. Constrained optimisation problems represent most real design problems, where constraints are imposed on the input parameters. These constraints may be simple bounds on the input parameters, such as size, or more complex constraints which themselves are a function of the input parameters, such as budgetary or structural constraints. Constraints are often expressed in the form of equalities

$$g_i(\mathbf{x}) = 0 \qquad i \in \iota$$

and of inequalities

$$g_i(\mathbf{x}) \geq 0 \qquad i \in \varepsilon$$

Here, g_i is the function representing the constraint i, and the sets ι and ε contain the equality and inequality constraints respectively [10].

2.2.1.6 Linear and non–linear

A linear optimisation problem has all linear objective and constraint functions, whereas non–linear optimisation problems do not. Linear programming aims to solve linear optimisation problems and non–linear programming tackles non–linear problems. During the initial development of optimisation techniques most problems, such as those in management, financial and economics, were formulated as linear problems. This is because linear problems are much easier to solve being uni–modal by nature. In physics and engineering the objective functions are almost always non–linear, which forced the development of non–linear programming techniques. This development has allowed more sophisticated non–linear formulations of financial and economic optimisation problems [10].

2.2.1.7 Robust Design Optimisation

Standard deterministic formulations of optimisation techniques rely on a static specification of operating conditions and do not account for the inevitable uncertainty in design parameters associated with the manufacturing process. The result is a design which will perform poorly in conditions outside of those defined in the optimisation. Robust design optimisation attempts to produce designs which perform well in a wide range of uncertain parameters (operating conditions, manufacturing errors etc) [12].

2.3 Gradient Based Algorithms

Gradient based algorithms have existed since the introduction of linear programming. Gradient based techniques are among the most used optimisation techniques in engineering, because of their high computational efficiency and well understood behavior. Examples of gradient based techniques include Newton–Rapshon [13], steepest descent [14] and the conjugrate gradient method [15].

All gradient based algorithms start with an initial set of input parameters $\mathbf{x}^{k=1}$. This initial set is a user's best guess at where the optimum may exist. Using information about the function at $\mathbf{x}^{k=1}$ the algorithm decides upon a new set of parameters $\mathbf{x}^{k+1}$ to inspect. The aim is for $f(\mathbf{x}^{k+1}) < f(\mathbf{x}^{k})$ [10] i.e. the value of the objective function to decrease each iteration. Often when describing these techniques the input parameters are referred to as solutions, probably due to the origin of these methods in equation solvers.

2.3.1 Gradient Based Optimisation Strategies

A number of different gradient based algorithms and strategies are now described. These techniques are commonly used and can be found in commercial packages such as DOT [16] or the MATLAB [17] Optimisation Toolbox.

2.3.1.1 Line Search Algorithms

Line search algorithms are optimisation techniques which undergo the following basic procedure each iteration

1. calculate a search direction $\mathbf{p}^{k}$,

2. find the value of α which minimises $f(\mathbf{x}^k + \alpha\mathbf{p}^k)$, and

3. $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha\mathbf{p}^k$.

Different algorithms result from different methods of determining $\mathbf{p}^k$. If the method of determining $\mathbf{p}^k$ is effective any value of $\alpha > 0$ should result in a reduction of the objective function. This means in practice it is not worth the effort to calculate the value of α which minimises $f(\mathbf{x}^k + \alpha\mathbf{p}^k)$ exactly. Instead approximate optimisation techniques can be used to find α [9].

In the steepest descent method

$$\mathbf{p}^k = -\nabla f(\mathbf{x}^k)$$

where ∇ is the gradient operator. This is the most obvious choice for $\mathbf{p}^k$, but it can result in slow convergence for complex problems. The slow convergence is due to the fact that $-\nabla f(\mathbf{x}^k)$ is always orthogonal to the contours of the function [10].

Conjugate gradient methods attempt to solve this problem by setting

$$\mathbf{p}^k = -\nabla f(\mathbf{x}^k) + \beta^k \mathbf{p}^{k-1}$$

where β^k is determined to ensure p^k and p^{k-1} are conjugate. Conjugate gradient methods are much more effective than steepest descent in terms of convergence rate [10].

The fastest convergence rates achieved by line search algorithms come from Newton direction descent methods. To find a Newton descent direction the second–order Taylor expansion

$$f(\mathbf{x}^k + \mathbf{p}) \approx f^k + \mathbf{p}^T \nabla f^k + \frac{1}{2}\mathbf{p}^T \nabla^2 f^k \mathbf{p}$$

is employed. The aim is then to find a direction, $\mathbf{p}^k$, which minimises this function. Taking the derivative with respect to $\mathbf{p}$ and setting it equal to zero results in

$$\mathbf{p}^k = -(\nabla^2 f^k)^{-1} \nabla f^k$$

for Newton descent methods. The primary drawback of Newton type methods is the requirement for a second derivative, which may not exist for non–smooth problems [10].

2.3.1.2 Trust Region Algorithms

Trust region algorithms use information gathered about f, during the optimisation process, to construct a model function m^k. The model function m^k will change each iteration and it resembles f close to the current optimisation point $\mathbf{x}^k$. For this reason, the search

for the minimum is restricted to a region close to $\mathbf{x}^k$ where the approximation m^k is trusted. The basic procedure followed by a trust region algorithm each iteration is

1. find $\mathbf{p}$ which minimises $m^k(\mathbf{x}^k + \mathbf{p})$ under the constraint $\|\mathbf{p}\| < \alpha^k$ where α^k is the trust–region radius for the current iteration k
2. $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{p}^k$
3. calculate the ratio
$$\frac{f(\mathbf{x}^k) - f(\mathbf{x}^{k+1})}{m^k(\mathbf{x}^k) - m^k(\mathbf{x}^{k+1})}$$
If this ratio is close to one then m^k is a good representation of f so increase α^{k+1}. Otherwise decrease α^{k+1}

The difference between trust region and line search algorithms is the order in which the direction and distance to $\mathbf{x}^{k+1}$ is determined. In line search algorithms, a direction is specified and the distance calculated by solving a minimisation problem. The reverse is true for trust region algorithms where a maximum distance is specified and the direction calculated using a minimisation [10].

2.3.1.3 Method of Feasible Directions

The method of feasible directions is a gradient based algorithm applied to constrained optimisation problems. The starting point is selected such that it satisfies all the constraints. A solution is said to be feasible if all constraints are satisfied. The value of α is selected to be small enough not to move outside a region of feasible solutions. In the method of feasible directions, $\mathbf{p}^k$ is selected such that a small move in that direction does not violate any constraints and also results in a reduction in the objective function. Calculating $\mathbf{p}^k$ for this method requires gradient information for both the objective function and the constraint functions [18].

2.3.1.4 Sequential Linear Programming

Originally introduced as the method of approximation programming, sequential linear programming is a technique to optimise non–linear problems. The technique uses a Taylor expansion to linearise the non–linear objective and constraint functions. Linear programming methods can then be applied to this linearised function. The linearised function is constructed using local gradients and is only valid in a given region. A distance is calculated, which represents the maximum distance the optimisation algorithm

can move from a solution before the function needs to be approximated again. This leads to solving a sequence of linear problems [19].

2.3.1.5 Sequential Quadratic Programming

Sequential quadratic programming was introduced in the late 1970s to solve non–linearly constrained optimisation problems. The technique works the same way as sequential linear programming except that the non–linear objective and constraint functions are approximated as quadratic problems, so that quadratic programming techniques can be applied. There are many rapid and accurate algorithms which belong to the family of quadratic programming techniques, which motivates the quadratic approximation in this method [20].

2.3.1.6 The Adjoint Approach

Jameson [21] introduced the idea of regarding a shape design optimisation problem as a control problem. This has become known as the adjoint method. In this method, the shape of the design is considered as the input to a control problem. Let $s(\mathbf{x})$ be a function representing the shape of a design. A variation in the surface δs results in a variation in the objective function δf. When the objective function is the output of some system, which is specified by variables and parameters w, such as Mach number or angle of attack, then the objective function is a function of these parameters and the shape of the surface s. This allows the expression

$$\delta f = \frac{\partial f^T}{\partial w}\partial w + \frac{\partial f^T}{\partial s}\partial s$$

The trick in the adjoint method is to eliminate the ∂w term by using a Lagrange multiplier Υ. If Υ is selected to satisfy the adjoint equation

$$\left[\frac{\partial R}{\partial w}\right]^T \Upsilon = \frac{\partial f^T}{\partial w} \tag{2.3}$$

where R is the governing equation for the model defining the objective function, it is shown that

$$\delta f = G\partial s$$

where

$$G = \frac{\partial f^T}{\partial s} - \Upsilon^T \left[\frac{\partial R}{\delta s}\right]$$

This means that δf is independent of δw and, hence, equation (2.3) only has to be solved once to calculate the objective function gradient for any number of input parameters [22].

Although Jameson has applied this technique to a number of real design problems, the development of adjoint techniques has not been rapid. A number of limitations are identified [23] which may explain this:

- in constrained optimisation an adjoint calculation is required for each constraint function.
- The benefit of the adjoint approach only becomes significant for large number of input parameters.
- The technique can be difficult to implement, despite the straight forward concept.
- Additional limitations of gradient based algorithms, which are discussed in detail below in Section 2.3.2.

2.3.2 Problems with Gradient Based Algorithms

The practicality of using gradient based optimisation techniques is reduced by the difficulty of automatically generating the derivatives of objective functions in real applications. When the objective function represents a complex numerical model, it is not possible to calculate derivatives analytically. It is possible to approximate derivatives numerically using, for example, finite difference methods, but this can be unreliable. In addition, it becomes increasingly difficult, and computationally expensive, to numerically calculate derivatives for large numbers of input parameters [10].

Gradient based techniques are particularly sensitive to noise in the objective function. Since many of the described techniques make use of the second derivatives, non–smooth functions pose challenges. Furthermore, for a number of non–linear engineering problems, the gradient will change throughout the search space and may not always point in the direction of the global minimum solution [12].

2.4 Gradient Free Algorithms

There are two broad categories of gradient free algorithms: metaheuristic and deterministic. Metaheuristic methods make use of random numbers to drive the optimisation [24], whereas deterministic techniques are, as the name implies, repeatable without randomness.

The Nelder–Mead Method [25] is an example of a deterministic technique which uses the concept of simplices to represent the agents in the search space. A simplex is a

collection of vertices which are connected to form a simple shape, for example a triangle in two dimensions. For an N dimensional problem, each simplex has $N + 1$ vertices. The objective function is evaluated once at each vertex and one of three operations then takes place to generate a new candidate vertex. These operations are reflection, contraction and expansion of the simplex. It has been shown that the method does not converge to a global minimum for general problems [26] but, despite this, it remains popular in industry, e.g. it has been widely used in the fields of chemistry, chemical engineering and medicine. This popularity is probably due to its simplicity and to the fact that it generally produces a significant improvement in the solution, within the first few iterations [26].

Broadly speaking deterministic techniques tend to be local optimisers, whereas metaheuristic techniques have the potential to be global optimisers [24]. For this reason deterministic techniques were not considered further.

Computer scientists investigated the possibility of applying the concepts of evolution as an optimisation tool for engineers during the 1950s and 1960s. This work gave birth to the genetic algorithm (GA), which is an example of an metaheuristic technique [27].

Although no rigorous definition exists, most metaheuristic algorithms contain the following elements:

- population of agents,
- selection according to fitness,
- crossover of agents, and
- random mutation of agents. [27]

These algorithms use large populations of agents to search the solution space. The objective function is calculated at the position of each agent and a series of rules are followed, with the aim to move the agents towards the global minimum.

An agent can be considered as a potential set of inputs to the objective function. These agents are given different names in specific algorithms, depending on the analogy employed. A selection scheme, based on the objective function, is constructed to be biased towards selecting the best individual agents to survive from one generation to the next. As the number of generations increases, the agents should approach the global optimum. This results in a number of gradient free methods based on principles of evolution.

In practice, testing for convergence in metaheuristic algorithms is not straightforward. This is due to most algorithms being designed to maintain diversity and avoid early

convergence to a local sub–minimum. Thus, algorithms of this type tend to simply be run for a prescribed time. Discussions with engineers in industry suggested that this would typically be overnight.

A successful metaheuristic technique finds a balance between exploring new areas of the search space and refining areas of the search space where current information suggests the minimum might lie. This balance is often determined by a set of parameters which are tuned to control the behavior of the agents.

To avoid the problem of sensitivity to the initial guess, gradient based techniques are often paired with metaheuristic techniques. For example, when applying optimisation techniques to training a neural network, Salimi et al [28] proposed a simple hybridisation between MCS [1] and a conjugate gradient method. They were training a neural network for classification and pattern recognition purposes. A neural network has an associated series of weights which governs its behaviour. Traditionally, the conjugate gradient method would be used to find the optimum weights. They found that this technique depends critically upon the initial guess, due to the multi–modal nature of the problem. The performance of the trained neural network increases when the global optimisation properties of the MCS were used to find an initial guess, which is then refined using a conjugate gradient solver.

2.4.1 Examples of Metaheuristic Algorithms

2.4.1.1 Genetic Algorithms

Among the earliest gradient free metaheuristic optimisation algorithms developed, GAs are based upon concepts of biological evolution. In most optimisation algorithms, the agents are represented by a vector of real numbers which refer to coordinates in the search space of interest. In GAs, the agents are, typically, represented by the strings of 1s and 0s that define the binary representation of the coordinates in the search space. Crossover and mutation operate on the string representation of each candidate solution, not on the physical, geometrical or real representations [27]. Representing a solution in this way can lead to the requirement for large numbers of degrees of freedom for complex problems. The fitness of each agent is evaluated and a number of the best candidates are selected to undergo the crossover and mutation processes to generate a new population. GAs have been applied successfully to many problems, including pattern recognition, robotics, electronic circuit design [29] and aerofoil shape optimisation [30].

2.4.1.2 Differential Evolution (DE)

In DE, the agents are simply represented by vectors of real numbers referring to co-ordinates in the search space. Given an initial population of agents, new agents are generated in the following manner:

1. A target agent, $\mathbf{x}_i$, is selected, with nth element denoted by $x_{n,i}$.
2. Three more randomly selected agents are selected, $\mathbf{x}_j$, $\mathbf{x}_k$, $\mathbf{x}_l$, where $i \neq j \neq k \neq l$.
3. A donor agent, $\mathbf{v}$, is generated using $\mathbf{v} = \mathbf{x}_j - \beta(\mathbf{x}_k - \mathbf{x}_l)$. Here, β is a mutation factor, which is a constant user selected parameter from the interval $[0, 2]$. For the studies presented in this thesis the value of $\beta = 0.5$ was employed.
4. A trial agent, $\mathbf{t}$, is generated by crossing the target and donor agents. Many different crossover strategies exist. The most competitive is binomial crossover [31], which was the strategy adopted in this thesis.
5. For each dimension, n, in the search space a random number r_n is generated from a uniform distribution between zero and one. If $r_n < \gamma$ then $t_n = v_n$ else $t_n = x_{n,i}$. The user specified parameter γ is known as the crossover probability. In the studies presented in this thesis the value $\gamma = 0.9$, which appears to be a commonly used value for this parameter [31], was used.
6. The fitness of $\mathbf{t}$ is evaluated and, if it is better than $\mathbf{x}_i$, then it replaces it.

This iteration scheme repeats until a suitable stopping criterion is met [32].

DE has been applied with success in the fields of electrical power systems, electromagnetic engineering, control systems and robotics, chemical engineering, pattern recognition, artificial neural networks and signal processing [31].

2.4.1.3 Particle Swarm Optimisation (PSO)

PSO is a metaheuristic technique used for global optimisation [33]. The method is based upon an analysis of how swarms of creatures behave in nature. Although these swarms are found to exhibit complex behaviour, each individual in the swarm follows a simple set of rules [34]. PSO attempts to mimic this behaviour by defining a swarm of individuals wandering in the design space [12]. Each individual, j, has a position, $\mathbf{x}_j$, defined by the design parameters, an associated fitness defined by the objective function, a local memory of its best location $\mathbf{p}_j$ and knowledge of the best position, $\mathbf{p}_g$, found globally by the swarm. In the process, the velocity, $\mathbf{v}_j$, of individual j changes, so that

the individual is drawn towards the best region of the design space, based upon the accessible information [34]. This is achieved by updating the velocities iteratively, using the equation

$$\mathbf{v}_j^{k+1} = \chi[w^k \mathbf{v}_j^k + c_j r_j(\mathbf{p}_j^k - \mathbf{x}_j^k) + c_g r_g(\mathbf{p}_g^k - \mathbf{x}_j^k)] \tag{2.4}$$

where k denotes the iteration number, r_j and r_g are random coefficients and χ, w^k, c_j and c_g are coefficients which can be selected to control the evolution of the algorithm. The coefficient χ determines how far a particle can move in an iteration step, c_j is the local memory coefficient, which controls the pull towards the best position in the memory of particle j, and c_g controls the pull to the best position found globally. The coefficients c_j and c_g and are always less than one and are typically given the value 0.75. PSO techniques are not dramatically influenced by the non–smoothness of the objective functions and are easy to implement. In addition, PSO can be readily parallelised [12].

Bratton and Kennedy [33] discuss the main improvements that have been made to the PSO procedure since its original introduction, e.g. the form adopted in equation (2.4) includes a small modification, involving the inertia weight parameter w^k, which affects the amount by which individuals overshoot known areas of high fitness. The lower the value of w^k, the greater the tendency of the swarm to cluster around the best solution found. This parameter is initially assigned a value greater than one and its value is then reduced, as the iterations proceed [33]. A problem with PSO is the large number of tuning parameters that are required. The effect of these parameters may not be immediately obvious.

PSO has been applied to training artificial neural networks, electrical power systems and biological systems [35].

2.4.1.4 Cuckoo Search (CS)

The CS algorithm is inspired by the breeding behaviour of cuckoos [36]. This behaviour is combined with an efficient flight strategy exhibited by many organisms. CS has been applied to a number of optimisation problems since its introduction [37–46]. In this section, the development of the original algorithm is described [36]. CS is discussed in more detail, since this is the algorithm which was modified and used in this thesis.

The Analogy In CS, the agents are represented by cuckoo eggs. Yang and Deb [36] aimed to emulate cuckoo breeding behaviour to drive their optimisation algorithm. They specify that an egg in a nest represents a solution and an egg represents a new potential solution. An equivalent representation, which is used throughout this thesis, is to let an egg represent any solution (new or old) and the nests simply being containers for the

eggs. Each egg carries two pieces of information, that is, the coordinates it has in the solution space and its fitness value. Using this analogy, the nests can be thought of as locations in an array where this information is stored. A nest may contain one or more eggs, although in most algorithms each nest contains only a single egg.

Cuckoo Breeding Behaviour The breeding behaviour of cuckoos is interesting because of its aggressive nature. This aggression motivates its application to optimisation algorithms. The key mechanism in cuckoo breeding is brood parasitism. This is the act of laying eggs in the nests of other birds, which may or may not be of the same species [47]. There are three types of brood parasitism, which are termed intraspecific, cooperative and nest takeover [48]. If the host bird discovers the cuckoo egg in her nest, she may destroy the egg or abandon the nest altogether. These mechanisms led to the evolution of cuckoo eggs which resemble eggs of locally found birds. Cuckoo eggs which resemble eggs of locally found birds are analogous to solutions with high quality fitnesses in CS.

Yang and Deb [47] attempt to model this behaviour by introducing the following three rules:

- Each cuckoo lays one egg at a time and deposits it into a random nest
- The best nests, which contain the highest quality eggs, carry over to the next generation
- The number of available host nests is fixed and there is always a probability that the cuckoo egg is discovered by the host. If an egg is discovered, the host bird throws it away. This effect is approximated by discarding a fraction of the eggs and replacing them at each generation.

Essentially, these rules provide a selection process for the optimisation algorithm, ensuring the best eggs survive from generation to generation. To complete the algorithm, a method of generating the eggs is required. This is where the Lévy flight is applied.

One of the most powerful features of CS is the use of Lévy flights to generate new eggs. It is this complex, random walk, strategy that makes CS stand out when compared with many other optimisation algorithms [49]. It should, however, be noted that this search pattern is applied in other optimisation techniques [50]. A Lévy flight is a random walk, characterised by a series of instantaneous jumps generated by a probability density function which has a power law tail. The Cauchy distribution, plotted in Figure 2.4, is often used for this purpose. The result is a path made up of many small steps and the

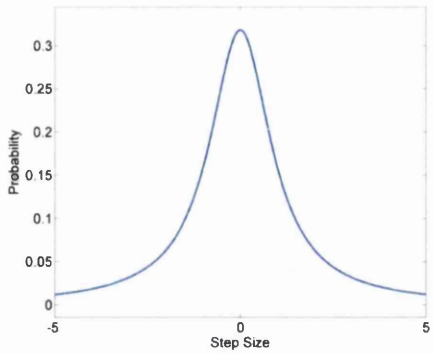


FIGURE 2.4: Cauchy probability distribution

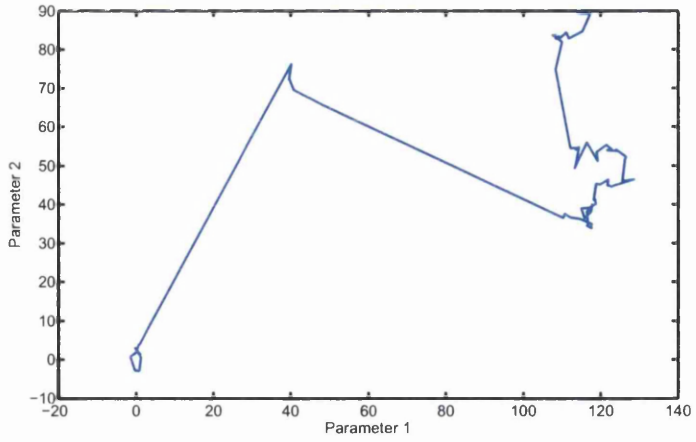
occasional large jump. A path of this type is frequently found in nature and is generally considered to represent the optimum random search pattern [51].

One of the defining characteristics of the Lévy flight search pattern is that it is scale free [47]. The scale free nature means that the pattern is the same in small scale searching and large scale searching. Small scale searching occurs locally in the solution space, and large scale searching occurs globally. This should lead to an automatic balance between exploration and refinement. However, the random nature of the flight means this can not be guaranteed. An example of a Lévy flight is shown in Figure 2.5. This is compared with a Gaussian walk, shown in Figure 2.6, where the random steps are generated using a probability function with a normal distribution. The lack of large jumps in the Gaussian walk means that it is less suited to large scale global searching.

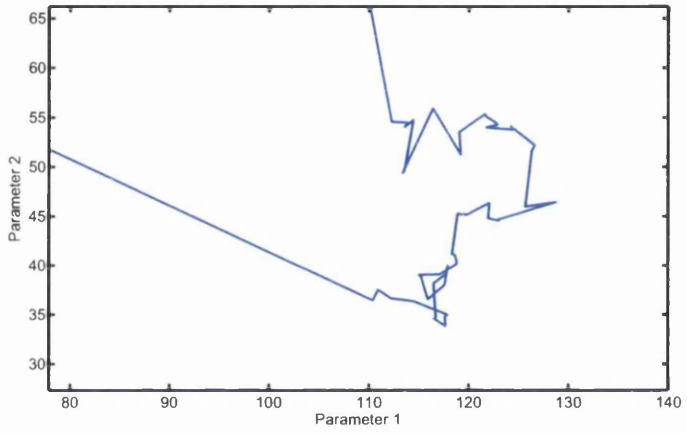
New eggs are generated by performing a Lévy flight from a randomly selected egg. The new egg is then represented by the coordinates reached at the end of the flight. Once the fitness of this new egg is evaluated a random nest is selected. If the fitness of the new egg is better than that of the egg already in the random nest, then it is replaced. To control the size of the Lévy flight a user specified coefficient α is defined. When a Lévy step is generated, using a random number generator, it is first multiplied by α before it is used to generate a new egg.

The CS algorithm is shown in its full form in Algorithm 1 [47].

To benchmark the CS algorithm, Yang and Deb [36] compared its performance to that of PSO and GA, by applying it to 10 standard optimisation benchmark functions. Each algorithm was run 100 times, to provide statistical significance to the test. Algorithms were terminated when the variation in the values of the objective function, throughout the population of eggs, was less than 10^{-5}. The percentage success rate at finding the global minimum, and the number of function evaluations needed before the stopping criteria was met, were recorded for each function for each algorithm. For all functions



(a) An example of an 100 step Lévy flight. Note the small localised searches connected by large jumps.



(b) A detailed section of the same Lévy flight, it shows the same characteristics as the full flight highlighting the scale free nature of this process.

FIGURE 2.5: Example of a Lévy Flight

considered, CS outperforms PSO and GA in terms of success rate and number of required objective function evaluations. Yang and Deb [36] state that the reason for this success is both a good balance of local and global searching and the small number of control parameters.

A major strength of this algorithm is its simplicity. There are only two parameters to be adjusted i.e. the fraction of eggs to be abandoned each generation and the flight step size. Furthermore they found that the convergence rate was not sensitive to the parameter values. In the examples presented in this thesis, the Lévy flight coefficient $\alpha = 1$ and the fraction of eggs to be discarded $p_a = 0.25$ were used, as suggested by Yang and Deb [47].

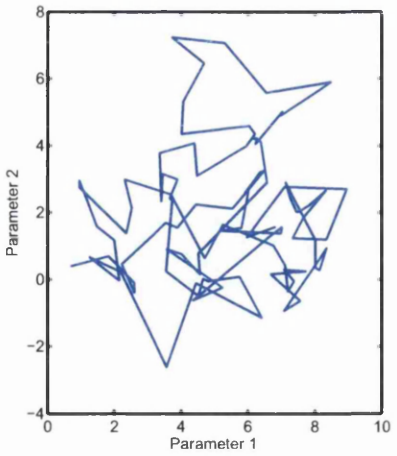


FIGURE 2.6: An 100 step Gaussian Walk with a step size of 1. Compared with the Lévy flights there are no large jumps accross the search space.

Algorithm 1 Cuckoo Search (CS)

Initialise a population of n host nests $\mathbf{x}_i$, $i = 1, 2, \cdots, n$
for all $\mathbf{x}_i$ **do**
  Calculate fitness $F_i = f(\mathbf{x}_i)$
end for
while $NumberObjectiveEvaluations < MaxNumberEvaluations$ **do**
  Generate a cuckoo egg ($\mathbf{x}_j$) by taking a Lévy flight from a random egg
  Calculate fitness $F_j = f(\mathbf{x}_j)$
  Choose a random nest i
  if $(F_j > F_i)$ **then**
    $\mathbf{x}_i \leftarrow \mathbf{x}_j$
    $F_i \leftarrow F_j$
  end if
  Abandon a fraction p_a of the worst eggs
  Build new eggs at new locations via Lévy flights to replace the lost eggs
  Evaluate the fitness of the new eggs and rank all solutions
end while

Applications A number of applications of CS have been published since its introduction. A selection of these applications, in which comparisons with alternative optimisation techniques have been made, are now be discussed.

Machine learning has become a popular application area for CS. For example, in the training of spiking neural networks [45]. The artificial neurons that make up a spiking neural network attempt to model the behaviour of biological neurons subjected to an input current. Such a network can be trained, based on the parameters associated with each neuron, to perform a variety of tasks. Vazquez [45] trained a network to perform a

pattern recognition task, using both CS and DE, and found that networks trained using CS perform slightly better than those trained with DE.

Selvi and Purusothaman [44] compared the performance of a number of heuristic algorithms for the task of breaking encrypted messages. Together with a number of heuristic algorithms, they also used a variety of ciphers to encrypt the messages. They found that CS performs well at breaking some of the ciphers, but is not as consistently successful as PSO.

A bloom filter is a simple data structure which is used to check if a string belongs to a particular database of strings. Bloom filters can be used as spam filters to detect unsolicited e–mails. A limitation of bloom filters is that there is always the risk of a false positive and, depending on parameters that define the filter's behaviour, different strings in the database have different probabilities of producing false positives. Natarajan and Subramanian [43] constructed an objective function which minimised the probability of producing false positives on the strings in a spam database which had the highest occurrence in spam mail. They found that filters constructed using CS outperform filters constructed in other ways.

CS has also been applied to the problem of optimising cutting parameters in milling operations [46]. In this example, the goal was to maximise the total profit rate by adjusting the cutting speed and feed rate of the milling process. A series of complex constraints were applied to these parameters, to ensure feasibility and sufficient product quality. It was found that, not only does CS find the maximum profit rates but that it also requires the smallest number of function evaluations compared to six other competing optimisation algorithms.

Another popular application of CS is for structural optimisation problems. These problems are highly nonlinear and involve large numbers of design variables with complex constraints. Problems of this type are difficult to solve globally using gradient based techniques. Gandomi et al [39] presented 13 different structural design problems, ranging from the simple design of an I–beam to the optimisation of a complete car structure subject to a side impact. They applied CS to the problems and compared its performance to a number of other state of the art optimisation algorithms. In almost every case, CS is better than the other methods used and, it was noted that, the sensitivity of the solution to the CS parameters is very small.

In the field of truss optimisation, Gandomi et al [40] applied a number of optimisation algorithms to five truss problems of increasing complexity. They were also able to show that CS outperforms many other state of the art algorithms. It should be noted that, in both papers [39, 40], different numbers of nests were used in each example

without comment. This implies that either this parameter needed tuning to get the best performance, which was at odds with their conclusions [39], or that this parameter was changed because of the increased CPU costs associated with some examples.

2.4.2 Problems with Metaheuristic Algorithms

In metaheuristic algorithms, the objective function needs to be evaluated for each agent, each generation, resulting in a large number of objective function evaluations. The computational cost of metaheuristic techniques is, therefore, often greater than gradient based methods [52]. The problem is particularly significant when considering applications where a single objective function evaluation represents a significant computational cost. This situation often arises in many engineering applications and the problem is often addressed by the use of cheap surrogate, or meta–models, to approximate the expensive objective function [53–56].

The long term goal is to produce a black-box metaheuristic optimisation algorithm which can be applied to any problem [57]. In an attempt to reach this goal, algorithms based on novel analogies are published on a regular basis. There is a possibility that this large number of different algorithms may cause confusion and hinder the application of these algorithms to real problems. In addition, when a new optimisation algorithm is introduced, there is generally an interest among academics in attempting to improve its performance. While this is obviously beneficial to the wider community, there might be a concern that this can lead to a large number of different classes of the same algorithm, which can cause difficulties when trying to compare individual algorithms. For example, it is not enough now to simply state that a PSO algorithm has been used, but the modifications which have been applied, to the base algorithm, must also be stated. Furthermore, it is valid to ask if these different analogies actually represent different approaches or can they be considered as specific examples of a more general algorithm [58].

The large number of associated tuning parameters may have also prevented the application of these algorithms to real problems. These tuning parameters effect the behavior of the agents in some way and often need to be tuned for a particular problem. CS was considered in this study, since it has only two such parameters, which is a small number compared with other algorithms.

Another potential problem with metaheuristic algorithms was raised with the publication of the "No Free Lunch" (NFL) theorem [57]. NFL states that, when averaged over all possible objective functions, the performance of all optimisation algorithms are the same. This can equivalently be stated by considering two optimisation algorithms A and B.

The result of the theorem is that for as many objective functions which A out performs B there are equally as many where this is not the case. The potential worrying outcome of NFL is that a completely random search will perform as well as any other algorithm averaged over all functions. In reality, no optimisation algorithm would be applied to all possible objective functions, only the subset of real problems of interest.

The NFL theorem highlights the care which must be taken during development and validation. New optimisation algorithms are often tested by applying them to benchmark problems, which have known analytical optimum finesses. Inevitably, a collection of popular benchmarking functions has emerged to which most algorithms were applied. For a new algorithm to be accepted by the research community, it will most likely have performed well on this set of functions. Over time, this could lead to optimisation algorithms which are engineered with these functions in mind [57].

The benchmark objective functions used in the optimisation literature have known minima, which allow the definition of an exact stopping criterion. Having a known minimum allows an easy comparison between two algorithms, which may be difficult in real applications. Adding the computational expense of the objective functions in real applications, leads to benchmark functions being the only practical way of comparing two algorithms. An open question is how well do benchmark functions actually model real problems? Using benchmarking functions to compare algorithms, and determine the sensitivity of tuning parameters, is a fruitful endeavour. However, wherever possible, real applications should be used to benchmark algorithms. Even if real applications are used, any comparisons may still be difficult, due to the random nature of these algorithms. Unlike gradient based optimisation, metaheuristic optimisation will not always result in the same answer for a single problem. Thus, in practice it is desirable to run the optimiser multiple times, which can be an expensive process.

2.5 Conclusion

A question may be asked which is better, gradient free metaheuristic or gradient based optimisation?

Gradient free metaheuristic algorithms belong to the field of soft computing. In this field, inexact solutions to difficult tasks are the goal. By contrast, the field of hard computing searches for the exact solutions to more simple problems. Gradient based algorithms fall into this category.

Is it better to have an exact set of inputs, which may be sub–optimal, or a set which are within the region of the global optimum? This highly depends on the situation. If the

objective function is the model of some physical system, then there will be errors both in terms of the inputs and outputs of the function. This makes the idea of finding an exact set of inputs less important.

The differences between metaheuristic and gradient based optimisation are so large that the question of which one is better was considered meaningless. It completely depends on the application and, ideally, both methods could be applied to the same problem.

The direction taken in this thesis was to investigate the application of gradient free metaheuristic optimisation, specifically CS. A long term goal is to tackle global optimisation problems as automatically as possible. The need for global optimisation points towards metaheuristic gradient free algorithms. CS itself has fewer tuning parameters than most algorithms of this type so lends itself to automation, since it would be expected that less tuning is necessary. Further to this initial studies into the performance of CS has shown it can perform better than other state of the art algorithms.

Chapter 3

Improving the Cuckoo Search Algorithm

3.1 Introduction

Yang and Deb [36] introduced CS in 2009. Their work shows that the CS algorithm has a much stronger ability to find true global minima solutions than a number of other metaheuristic algorithms. For this reason CS was applied to the problem of mesh optimisation, which is the subject of Chapter 6. It was initially found that the time taken to find optimum meshes using CS was prohibitively long. This was due to the large number of objective function evaluations CS needs to find an objective function's minimum.

To address this problem a number of modifications were made to the algorithm, to speed up convergence. The result of this work was MCS [1]. When applied to a mesh optimisation scheme [3] MCS is found to outperform existing techniques.

To present MCS to the wider community, the study presented in Section 3.2.2 was undertaken and published [1]. The study aimed to recreate the study performed by Yang and Deb [36] for CS. It is found that MCS outperforms CS in almost every case.

Since the publication of MCS, other modifications have been made to CS. For completeness a selection are briefly discussed in Section 3.3.

In a recent study, Bhargava et al [6] found that CS outperforms MCS when applying them both to a phase equilibrium modelling problem. This raised questions about the suitability of the validation studies performed both in CS [36] and MCS [1]. These questions are explored and a more detailed study presented in Section 3.4.

3.2 Modified Cuckoo Search (MCS)

Two modifications were made, with the goal of speeding up the convergence and, hence, reducing the number of objective function evaluations required to find the global minimum.

How well the CS algorithm modelled cuckoo behaviour was not considered important. In reality, cuckoo populations actually appear to be in decline [59]. Furthermore, it was apparent that many modifications to other optimisation algorithms are less concerned with modelling the original analogy and more concerned with improving performance.

The first modification was to the Lévy flight coefficient, α. In the CS, α is constant and typically the value $\alpha = 1$ is employed [36]. In MCS, the value of α was made to decrease as the number of generations increased. This was done for the same reason that the inertia constant might be reduced in PSO [33], which is to encourage more localised searching as the agents, in this case the eggs, get closer to the solution.

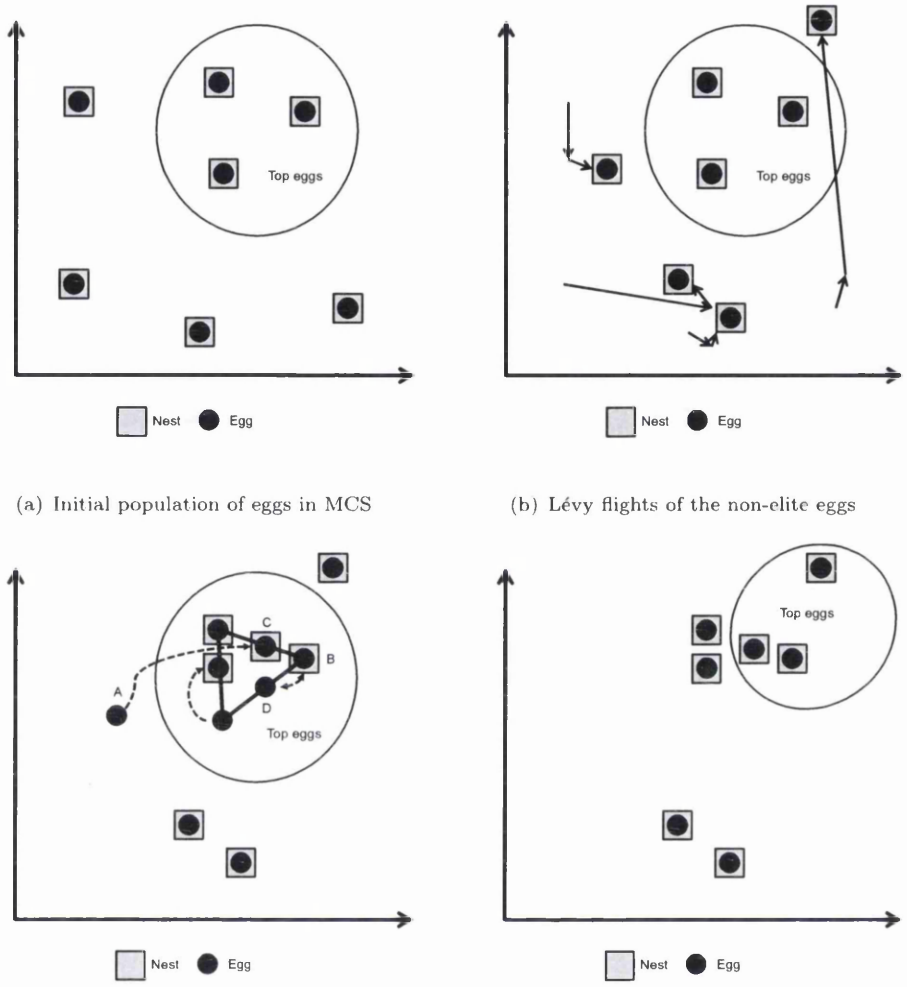
An initial value of the Lévy flight coefficient $\alpha = A = 1$ was selected and, at each generation, a new value is calculated using $\alpha = A/\sqrt{G}$, where G is the generation number. This exploratory search is only performed on those nests that are to be abandoned.

The unmodified CS algorithm could be considered as a series of independent Lévy flights, where the ineffective flights are reset each generation. The missing ingredient was crossover between the solutions. The second modification added this crossover. This was done by changing the structure of the algorithm, as illustrated in Figure 3.1.

The eggs are first ordered by fitness and a fraction of the best, or top, eggs are considered as the elite eggs. This initial population is shown in Figure 3.1(a). The first step in the algorithm models both host birds discarding eggs and random genetic mutation in the next generation of eggs. Starting at each non–elite egg, a Lévy flight is performed to generate a new egg. The fitness of these new eggs is calculated and they replace each of the non–elite eggs, as shown in Figure 3.1(b). The replacement is performed regardless of the relative fitnesses of the new and old eggs.

Crossover between cuckoos, which survive to the next generation, is modelled as follows:

1. Each of the top elite eggs randomly picks a second elite egg
2. A new egg is generated along the line which connects these two eggs and the fitness evaluated. The distance along this line at which the new egg is located is calculated, using the inverse of the golden ratio $\varphi = (1 + \sqrt{5})/2$, such that it is located closer to the egg with the better fitness. In the case that both eggs have



(a) Initial population of eggs in MCS

(b) Lévy flights of the non-elite eggs

(c) Crossover of elite nests

(d) The top eggs updated

FIGURE 3.1: Illustration of the Modified Cuckoo Search Algorithm

the same fitness, the new egg is generated at the midpoint. There is a possibility that, in the previous step, the same egg wass picked twice. In this case, a local Lévy flight search is performed, from the randomly picked nest, with Lévy flight coefficient $\alpha = A/G^2$.

3. A nest is picked at random, from all nests. The egg generated in the process above is compared with the egg in the random nest. If the newly generated egg is of better fitness than the egg already in that nest, then it replaces it, otherwise it is discarded.

This process is shown in Figure 3.1(c), where egg C is found to be better than egg A and so replaces it, but egg D is not better than egg B and so it does not replace it.

Once all eggs which are not in nests are discarded, the group of top eggs is updated, as shown in Figure 3.1(d), and the process iterates.

There is a possibility, particularly during crossover, that a new egg may be generated at the same location as an existing egg. To evaluate the objective function again at this location would be wasteful, so a check is made to see if the newly generated egg exists. If it is found that the newly generated egg does already exist in the population, a local Lévy flight search is performed. Ideally, it would be good to check if the location of a new egg has been visited previously by the algorithm, but this would require a potentially prohibitively large amount of storage, so it is considered unfeasible.

Algorithm 2 details the steps involved in the MCS technique. It was found through experience that abandoning 75% of the nests and placing the remaining 25% in the elite group produced the best results for a variety of test functions.

3.2.1 Population Initialisation

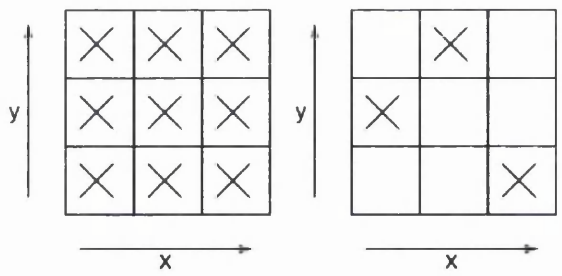
Most gradient free algorithms require that a starting population be specified by the user. Many optimisation algorithms show a bias to the starting population [33], which suggests that this is an aspect of these algorithms that needs to be carefully considered. Latin hyper–cube sampling (LHS) is a popular choice for generating initial populations for gradient free algorithms and this was the method adopted here.

The field of work which attempts to build a theoretical foundation for sampling a parameter space is experimental design. Figure 3.2 compares two contrasting techniques, full factorial sampling and LHS. In full factorial sampling each dimension is split into a number of bins where the same number of bins, n, is taken for each dimension. The total number of samples would be $N = n^d$ where d is the number of dimensions. Clearly, as n and d increase N will increase rapidly. An example of full factorial sampling is shown in Figure 3.2(a).

LHS attempts to sample the full design space, with a minimum number of samples. For LHS with N samples, each dimension in the design space is split into N bins. The N samples are randomly selected such that, if a projection is viewed along any single dimension, there is a sample in each bin [60]. An example of LHS is shown in Figure 3.2(b). The random nature of this sampling method means that a space filling sampling is not guaranteed. To account for this, in the examples presented here, 1000 LHS were generated iteratively and the sampling with the maximum minimum distance between sample points was selected.

Algorithm 2 Modified Cuckoo Search (MCS) [1]

$A \leftarrow MaxLévyStepSize$
$\varphi \leftarrow GoldenRatio$
Initialise a population of n eggs $\mathbf{x}_i (i = 1, 2, \cdots, n)$
for all $\mathbf{x}_i$ **do**
  Calculate fitness $F_i = f(\mathbf{x}_i)$
end for
Generation number $G \leftarrow 1$
while $NumberObjectiveEvaluations < MaxNumberEvaluations$ **do**
  $G \leftarrow G + 1$
  Sort eggs by order of fitness
  Select the eggs to be abandoned
  for all eggs to be abandoned **do**
    Current position $\mathbf{x}_i$
    Calculate Lévy flight step size $\alpha \leftarrow A/\sqrt{G}$
    Perform Lévy flight from $\mathbf{x}_i$ to generate new egg $\mathbf{x}_k$
    $\mathbf{x}_i \leftarrow \mathbf{x}_k$
    $F_i \leftarrow$ Calculate fitness $f(\mathbf{x}_i)$
  end for
  for all of the top eggs **do**
    Current position $\mathbf{x}_i$
    Pick another egg from the top eggs at random $\mathbf{x}_j$
    if $\mathbf{x}_i$=$\mathbf{x}_j$ **then**
      Calculate Lévy flight step size $\alpha \leftarrow A/G^2$
      Perform Lévy flight from $\mathbf{x}_i$ to generate new egg $\mathbf{x}_k$
      Calculate fitness $F_k = f(\mathbf{x}_k)$
      Choose a random nest l from all nests
      if $(F_k > F_l)$ **then**
        $\mathbf{x}_l \leftarrow \mathbf{x}_k$
        $F_l \leftarrow F_k$
      end if
    else
      $dx = |\mathbf{x}_i - \mathbf{x}_j| / \varphi$
      Move distance dx from the worst egg to the best egg to find $\mathbf{x}_k$
      Calculate fitness $F_k = f(\mathbf{x}_k)$
      Choose a random nest l from all nests
      if $(F_k > F_l)$ **then**
        $\mathbf{x}_l \leftarrow \mathbf{x}_k$
        $F_l \leftarrow F_k$
      end if
    end if
  end for
end while



(a) Full Factorial Sampling (b) Latin Hyper-Cube Sampling

FIGURE 3.2: Comparing sampling techniques. x and y represent two dimensions of the search space, and the symbol "X" indicates sample locations

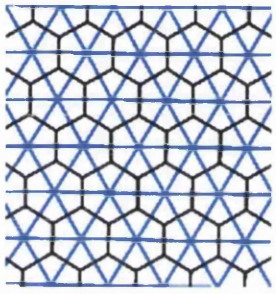


FIGURE 3.3: Example of a CVT. The vertices of the equilateral triangles are the Voronoi generators and the Voronoi vertices connect to form hexagonal Voronoi cells. In this case, the Voronoi generators lie at the centroid of the Voronoi cells, creating a CVT

Recently, Shatnawi and Nasrudin [61] attempted to address this issue by employing centroidal Voronoi tessellation (CVT) as a technique to improve the starting positions of the eggs. If each egg is considered as a Voronoi generator, then each egg is surrounded by a Voronoi cell in the solution space. Each Voronoi cell defines the area of space containing all points closest, in terms of Euclidean distance, to its generator. When the centroid of the Voronoi cell lies at the same point as the Voronoi generator, the set of points define a CVT. Figure 3.3 shows an example of a CVT.

Shatnawi and Nasrudin [61] argued that, if an iterative algorithm, such as the one presented by Du and Gunzburger [62], is used to move an initial random distribution of eggs close to a CVT, this results in faster convergence to the global minimum. This is because a CVT is space filling and results in an increased probability of one of the initial eggs being close to the global minimum. Their results show a marginal decrease in the number of objective function evaluations required to find the global minimum, when comparing CVT to a random sampling. However, they noted that the additional cost of generating a CVT may outweigh any benefit in function evaluation cost reduction. A CVT algorithm may also be difficult to implement for an arbitrary number of dimensions,

so it was not adopted as a technique in the applications presented in this thesis.

3.2.2 Benchmarking

In this section, MCS was compared to DE, PSO and CS. All algorithms were implemented in MATLAB [17]. To compare the relative performance of each method, a series of seven test functions, taken from those originally used by Yang and Deb [36], were used as objective functions.

For each function, 30 trials were performed for each method. Both unimodal and multimodal functions were selected and a range of dimensions were tested to gauge the robustness of each method. The initial population of 20 individuals were picked at random using LHS. Since many optimisation methods show a bias towards the initial population of samples [33], the same 30 sets of initial populations, were generated and used for benchmarking each algorithm.

To compare the relative success of each method, the Euclidean distance from the known global minimum to the coordinates of the member of the population with lowest objective function value was measured. Each data point represents the mean, taken from all 30 trials, of this distance at a given number of objective function evaluations and the error bars represent the standard deviation.

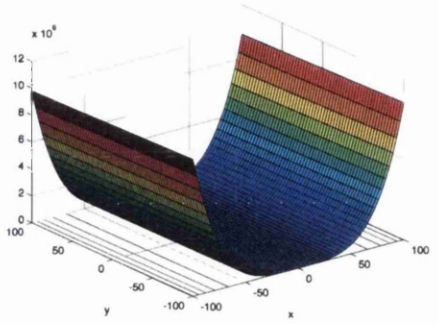
3.2.2.1 Rosenbrock's Function

Rosenbrock's function

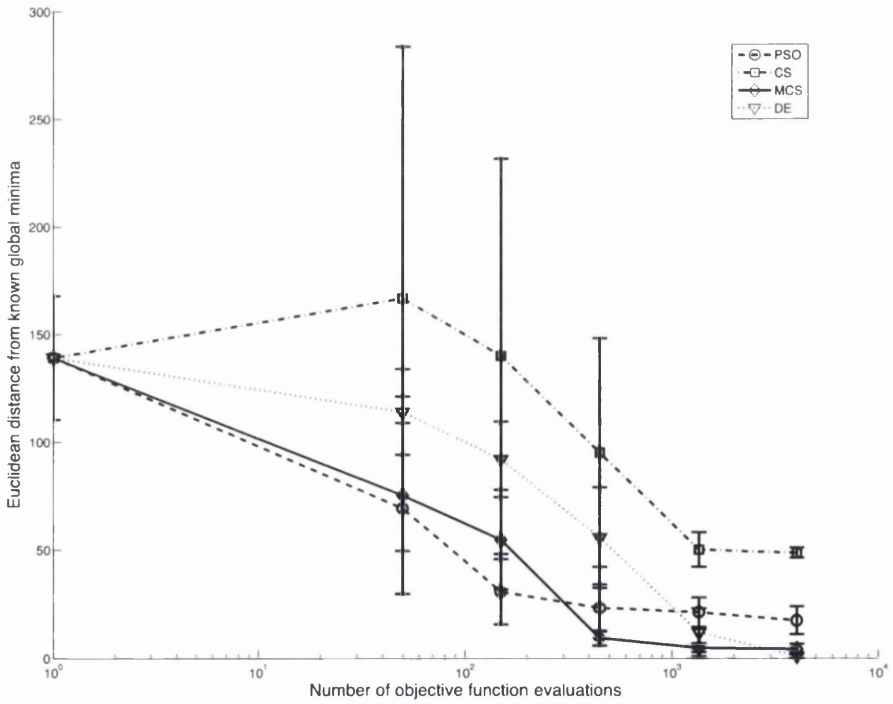
$$f(\mathbf{x}) = \sum_{i=1}^{d-1} \left[(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2\right] \qquad x_i \in [-100, 100] \tag{3.1}$$

with $d = 10$ was used in this example. The two dimensional version of this function is plotted in Figure 3.4(a). This has a unique global minimum of $f = 0$ which lies inside a long, narrow, parabolic shaped valley. The valley is easy to find, but finding the true global minimum inside this valley is difficult.

The results obtained for Rosenbrock's function are plotted in Figure 3.4(b). In this example, PSO and MCS showed an initial fast convergence towards the known minimum. As the number of objective function evaluations increased, MCS moved closer to the minimum than PSO. The DE had a slower initial convergence rate, but eventually matched the performance of MCS. All methods clearly outperformed the standard CS algorithm.

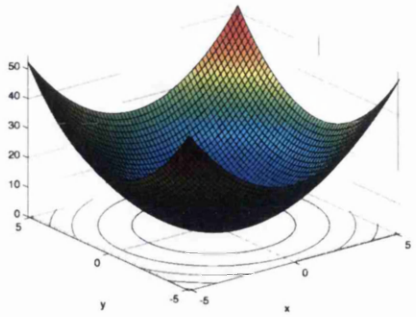


(a) Function Plot

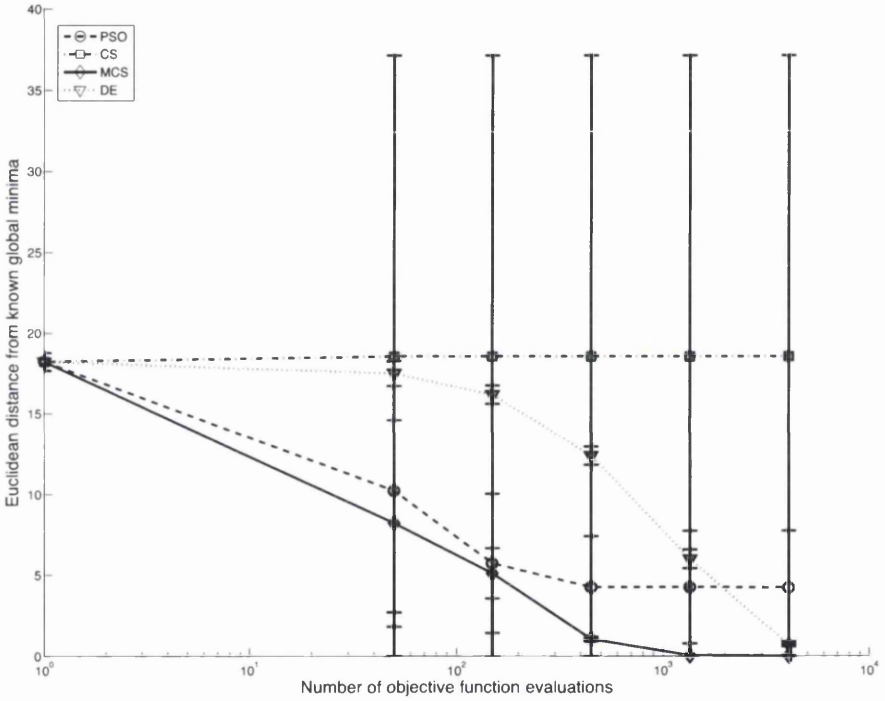


(b) Comparison of the performance of the different methods

FIGURE 3.4: Rosenbrock's function



(a) Function Plot



(b) Comparison of the performance of the different methods

FIGURE 3.5: de Jong's function

3.2.2.2 de Jong's Function

The de Jong function

$$f(\mathbf{x}) = \sum_{i=1}^{d} x_i^2 \qquad x_i \in [-5.12, 5.12] \tag{3.2}$$

with $d = 50$ plotted in Figure 3.5(a) for $d = 2$. This function has a unique global minimum of $f = 0$.

The results obtained for minimising this function with each optimisation algorithm are plotted in Figure 3.5(b). Clearly MCS had the fastest convergence rate, finding the global minimum after 10^3 objective function evaluations. Although slower, DE eventually found the global minimum. CS did not manage to succeed, in this relatively simple problem, within the maximum number of objective function evaluations, showing a wide range of obtained results. This highlights the lack of information exchange between nests in the standard algorithm, meaning CS could not take advantage of the symmetry of this function, as the other methods appeared to.

3.2.2.3 Rastrigin's Function

Figure 3.6(a) shows a plot of Rastrigin's test function

$$f(\mathbf{x}) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i)\right] \qquad x_i \in [-5.12, 5.12] \tag{3.3}$$

for $d = 2$. This function has a unique global minimum of $f = 0$, for benchmarking the function was used with $d = 100$.

For this high dimensional unimodal function, MCS outperformed both PSO and CS in all 30 trials across the full range of objective function evaluations. It also showed a much higher convergence rate than DE. These results are presented in Figure3.6(b).

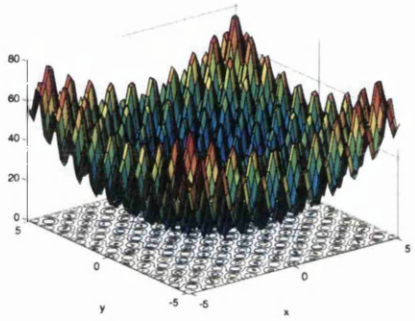
3.2.2.4 Schwefel's Function

Figure 3.7(a) shows a plot of Schwefel's test function

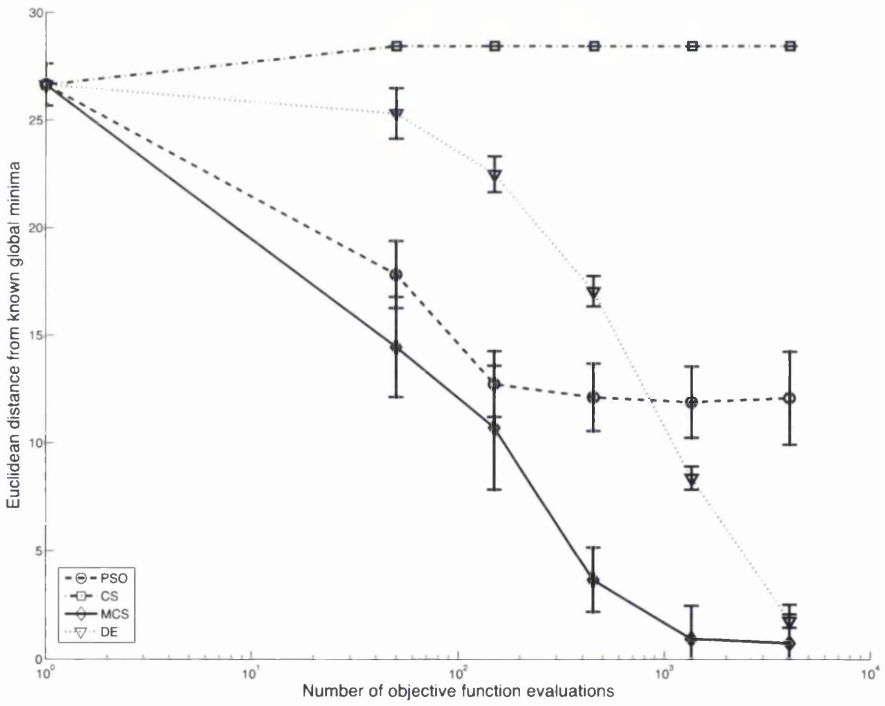
$$f(\mathbf{x}) = \sum_{i=1}^{d} \left[-x_i \sin(\sqrt{|x_i|})\right] \qquad x_i \in [-500, 500] \tag{3.4}$$

for $d = 2$. This is a multimodal function with a global minimum of $f = -418.9829d$. The 10 dimensional version of the function was used in the benchmark test.

For this case, Figure 3.7(b) shows that there was little difference between the performance of PSO, CS and MCS. Here DE significantly outperformed all other methods.

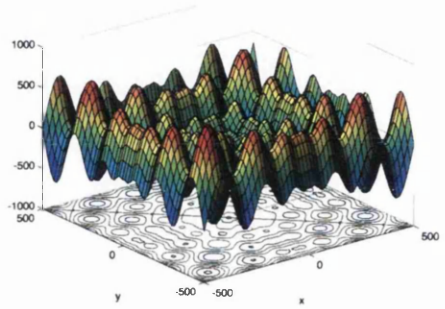


(a) Function Plot

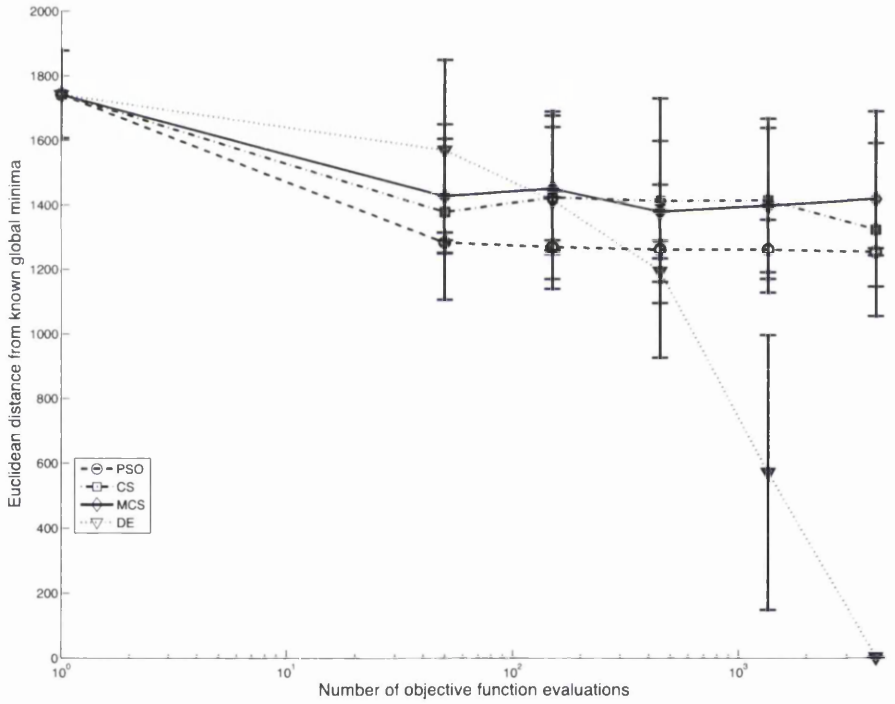


(b) Comparison of the performance of the different methods

FIGURE 3.6: Rastrigin's function

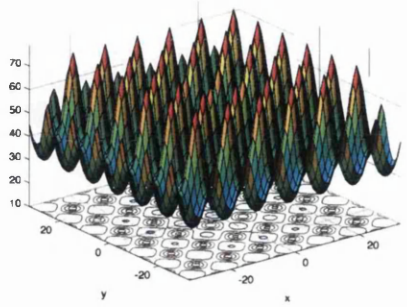


(a) Function Plot

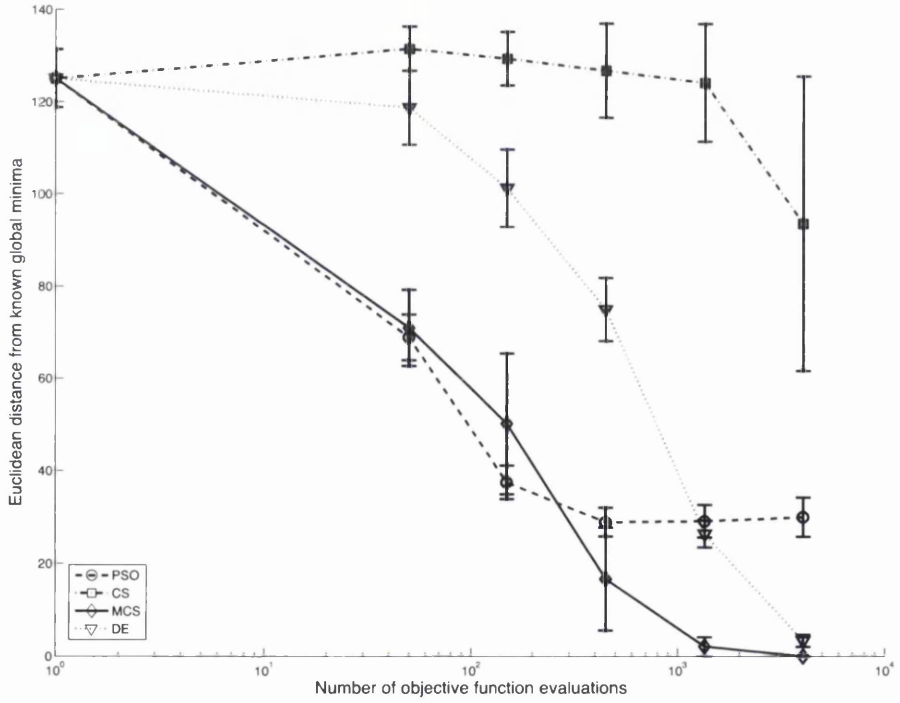


(b) Comparison of the performance of the different methods

FIGURE 3.7: Schwefel's function



(a) Function Plot



(b) Comparison of the performance of the different methods

FIGURE 3.8: Ackley's function

3.2.2.5 Ackley's Function

Figure 3.8(a) shows a plot of Ackley's function

$$f(\mathbf{x}) = -20\exp\left[-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right] - \exp\left[\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right] + (20+e) \qquad x_i \in [-32.768, 32.768] \tag{3.5}$$

with $d = 2$. This is a multimodal function with a global minimum of $f = 0$. 50 dimensions were used for this function in the benchmark.

MCS outperformed all other methods in this example, as shown in Figure 3.8(b).

3.2.2.6 Griewank's Function

Figure 3.9(a) is a plot of Griewank's test function

$$f(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{d}x_i^2 - \prod_{i=1}^{d}\cos(\frac{x_i}{\sqrt{i}}) + 1 \qquad x_i \in [-600, 600] \tag{3.6}$$

with $d = 2$. This function has many local minima, but a global minimum of $f = 0$. In this example $d = 100$ was used for benchmarking.

In this high dimensional multimodal problem, MCS outperformed all other methods in all 30 trials across the full range of objective function evaluations. The results are presented in Figure 3.9(b).

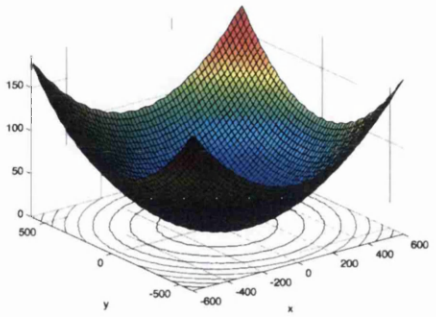
3.2.2.7 Easom's 2D Function

Figure 3.10(a) shows a plot of Easom's two-dimensional test function

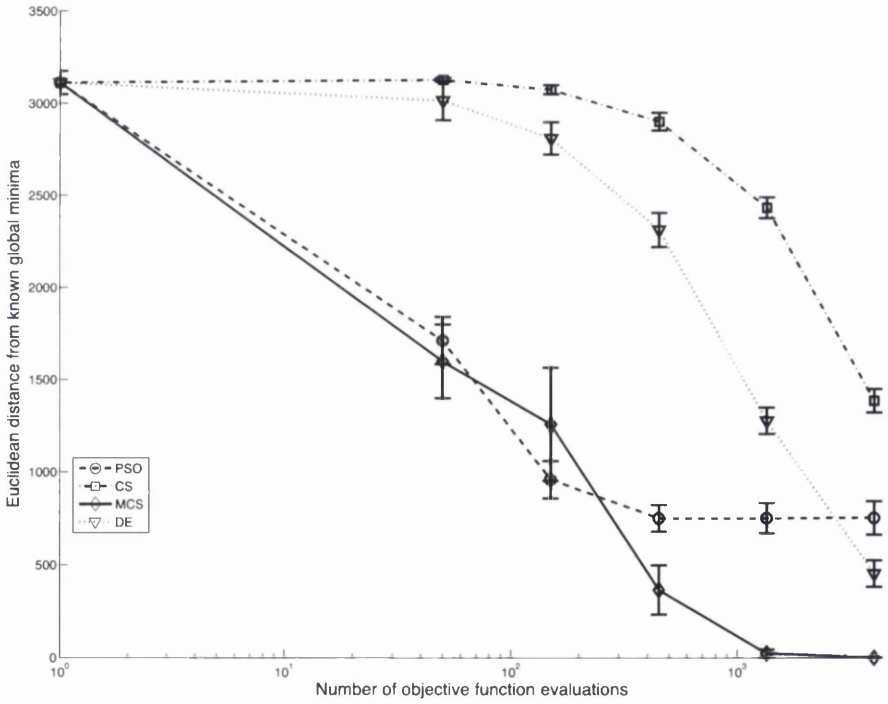
$$f(\mathbf{x}) = -\cos(x_1)\cos(x_2)\exp\left[-(x_1-\pi)^2 - (x_2-\pi)^2\right] \qquad x_i \in [-100, 100] \tag{3.7}$$

which has a very sharp minimum of $f = -1$.

The PSO performed better than CS and MCS at higher numbers of objective function evaluations. The results are presented in Figure 3.10(b). This performance difference could be due to the nature of how the particles move compared to the nests in CS and MCS. In MCS and CS, the nests jump around, searching from point to point via random walks. In PSO, the particles move smoothly with velocities influenced by local and global memory, which possibly increases the possibility of finding the very sharp minimum. DE

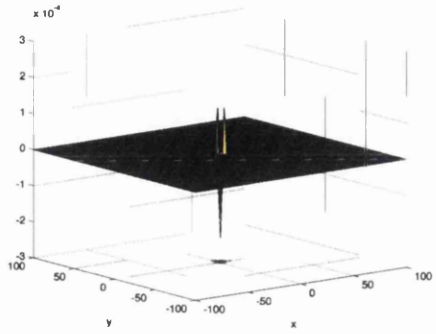


(a) Function plot

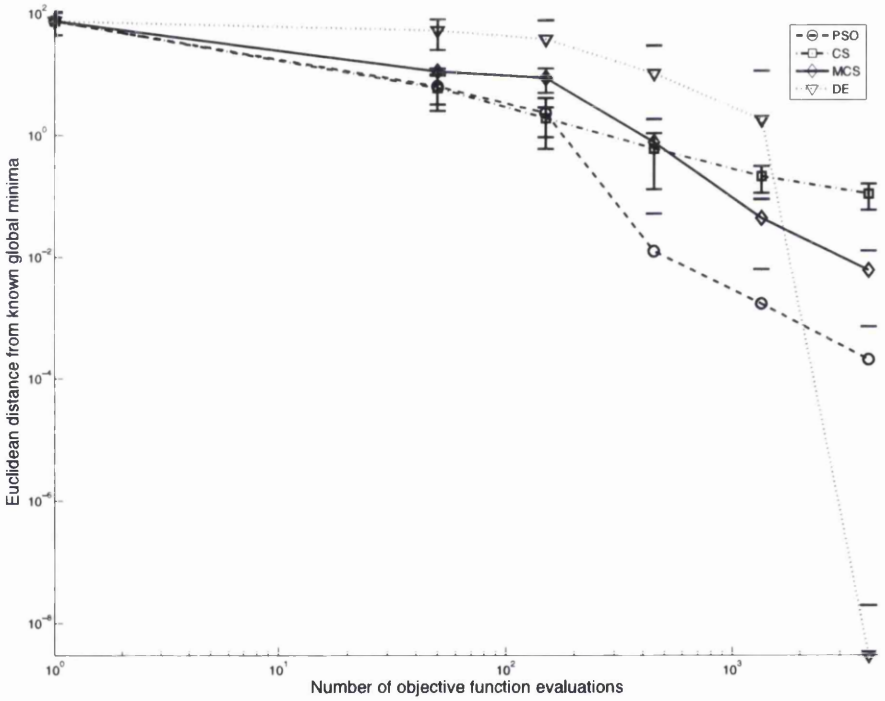


(b) Comparison of the performance of the different methods

FIGURE 3.9: Griewank's function



(a) Function Plot



(b) Comparison of the performance of the different methods

FIGURE 3.10: Easom's 2D function

came closest to the minimum at the maximum number of objective function evaluations. However, its initial convergence rate was slower than the other methods.

3.2.3 Conclusion

For all the test examples considered, MCS outperformed CS and performed as well as, or better than, PSO. In some examples, the MCS performed significantly better.

At low numbers of objective function evaluations, the differences between the methods tested were less significant but, in general the PSO and MCS outperformed CS in this region, which was one of the main goals of this work. This led to the conclusion that, in applications with computationally expensive objective functions, PSO and MCS would be the techniques of choice.

At high numbers of objective function evaluations, MCS outperformed PSO in a number of cases, and was matched elsewhere. It followed that MCS is the technique of choice, even when the CPU cost of the objective function is not prohibitive.

The results suggested that MCS is well suited to problems where the objective function has a high number of dimensions. In these situations, MCS significantly outperformed PSO and CS. This is probably due to the Lévy flight effectively reducing the size of the search space with the large jumps between steps, and the simultaneous refinement of local searches, which an unmodified CS did not achieve. A similar behaviour can be achieved in the PSO by using multiple swarms from an initial particle population [63]. However, MCS has the advantage of being simple to implement compared to sophisticated multiple swarm PSO algorithms.

In the majority of cases tested, MCS exhibited a much higher convergence rate than DE. Eventually, DE got as close to the minimum as MCS and in some cases closer. However, the success of DE came at the cost of many more objective function evaluations. The examples showed that MCS performed equally well over a variety of functions and dimensions, and can find a good minimum in a conservative number of objective function evaluations.

The caveat on all the findings detailed in this conclusion is that they were limited to the comparison of the optimisation algorithms considered. This means that when a statement such as 'MCS is the technique of choice...' was made, it was only valid when compared to PSO, DE and CS on this subset of problems. It would be impossible to compare MCS to all algorithms for all problems.

Since the publication of the original paper [1], a MATLAB implementation of MCS has been made available in the open source project `modified-cs` [64]. This has led to the use

of MCS in a number of applications, including the optimum design of steel frames [41] and as a diagnostics tool for the automatic detection of diabetes [65]. This, along with the applications presented as part of this thesis, continues to build a body of evidence showing that MCS can be successful for a wide range of problems.

3.3 Other Modifications

3.3.1 Emotional Chaotic Cuckoo Search (ECCS)

The modification suggested by Lin and Lee [66] includes no comparison of its performance with that of the original CS. The two goals of their work were to improve the quality of neighborhood searching, when generating new eggs, and to increase the likelihood of eggs escaping local minima.

To address the first goal, they identify an issue with the Lévy flight. The coefficient α which determines the size of the Lévy flight step size needs to be correctly selected, depending upon the scale of the problem. The scale may be unknown in some cases. This is also often difficult because the random Lévy flight steps do not have a well defined range of values. A Lévy flight with a fixed α does not display ergodicity, which means there is not an equal chance of an egg travelling through every point in the solution space. They attempted to address this issue by replacing α with a value taken from a chaotic sequence. This sequence is ergodic and, thus, increases the chance that every possible scale is used during the search. This, in turn, increases the chance of finding the correct minimum. This also removes the step size from the list of parameters that must be specified by the user.

The second goal was addressed by using an emotional model to change the simple egg replacement scheme of CS, where an egg replaces another if its fitness is better. The emotional model compares the fitnesses and, based on a logarithmic relationship, only replaces an egg when a certain threshold is exceeded. This resulted in an increased probability of moving across hills and valleys in the fitness landscape, which, it is claimed, reduced the chance of getting trapped in local–minima. However, without a detailed comparison to the original CS, it is hard to verify that these claims are actually justified.

3.3.2 PSO CS Hybrid

A common practice is to hybridise two gradient free algorithms, in an attempt to replace the weaknesses of one with the strengths of the other. Ghodrati and Lotfi [67] recognise

that the original CS lacks communication between the individual eggs. They add this communication, using a PSO update equation to move the eggs, after the Lévy flight step is performed and before the worst eggs are discarded.

The hybridisation with PSO results in double the number of parameters that require adjustment in the algorithm compared with CS. Over a range of benchmarking functions, they found that, in terms of convergence rate and closeness to the global minimum, the hybrid algorithm outperforms CS 80% of the time and PSO 85% of the time.

3.4 Dissecting the Cuckoo Search

Recently Bhargava et al [6] applied CS [36] to the problem of phase equilibrium modelling. This is of interest in the development of novel chemical industrial processes. The problem involves the minimisation of a free energy function which is highly non–linear, with many local sub–minima which have values close to the true global minimum.

As part of their work, the performance of CS is compared with that of MCS. They found that, although MCS has a higher success rate than CS at low numbers of iterations, it almost always became trapped in sub–minima. With more iterations, the performance of CS overpowers that of MCS, reaching a success rate of almost 100%.

These findings suggested that there may be a problem with the MCS performance study undertaken initially, because their conclusions were quite different. This motivated a more detailed investigation into the mechanisms of CS and MCS.

The benchmarking functions originally used to validate MCS [1] are selected from the set of functions used to validate the CS algorithm [36]. Issues can be identified with using these functions as models for real optimisation problems. In every function the global optimum occurs at a point where all the function inputs have the same value and, in most cases, that point is near the centre of the search space. With this in mind, it was deemed necessary to complete a parameter study of MCS on more appropriate benchmark problems. The results of this study are presented in this section, after which the findings of Bhargava et al [6] are discussed.

To provide a better model of a real optimisation problem, the point of the global optimum was shifted, using the mapping

$$F(\mathbf{x}) = f(\mathbf{x} - \mathbf{o}_{new} + \mathbf{o}_{old}) \tag{3.8}$$

where $f(\mathbf{x})$ is the non–shifted function, $\mathbf{o}_{old}$ is the global optimum of the non–shifted function, $\mathbf{o}_{new}$ is the new global optimum and $F(\mathbf{x})$ is the shifted objective function.

The new optimum position can have different values in each of its elements to better reflect real problems.

A series of studies are now presented which show the sensitivity of MCS's performance to the tuning parameters and various strategies used in the algorithm. Each study was performed on both shifted and non–shifted problems, to show that different conclusions can be reached if only one type of problem is considered. The new global optima in the shifted functions were randomly positioned within the domain of the problem. For each problem, the number of dimensions used were as in the original validation [1] and each data point represents the mean of 100 runs with 100 different sets of starting eggs. In the graphs, the distance from the global minimum represents the Euclidean distance of the best egg from $\mathbf{o}_{new}$ for the shifted functions and $\mathbf{o}_{old}$ for the non–shifted problems.

The studies resulted in a large set of data. To aid readability of the thesis, only a selection of the graphs, showing typical results, are presented. The remaining data is available on request.

3.4.1 Fraction of Discarded Eggs

The fraction of eggs to be discarded, p_a, controls how many eggs are replaced using a Lévy flight at each generation. The remaining eggs undergo a crossover scheme and are only replaced if an improvement is made. It would be expected that increasing the value of p_a would result in a more exploratory type search, with less chance of getting trapped in local–minima. To determine if this is the case, a diversity metric was defined. The metric is simply the standard deviation of egg position normalised by the distance between the pairs of eggs furthest apart in the current population.

The optimisation algorithm was run with 50 eggs, for varying values of p_a, with all other parameters kept constant at the values used in the original study [1]. An adaptive value for p_a was also tested, where the diversity was tracked every iteration and p_a set to one minus the diversity measure. The argument for doing this was that the more diverse the search, there may be benefit to refine the search, so that fewer eggs should be discarded.

3.4.1.1 Non–shifted Functions

Figures 3.11 to 3.14 show the results obtained for the non–shifted Ackley and de Jong functions. The general trend showed an increase in diversity as p_a was increased. This increase in p_a also, unexpectedly, resulted in a decrease in performance. The distance from the best egg to the global minimum increased as p_a was increased.

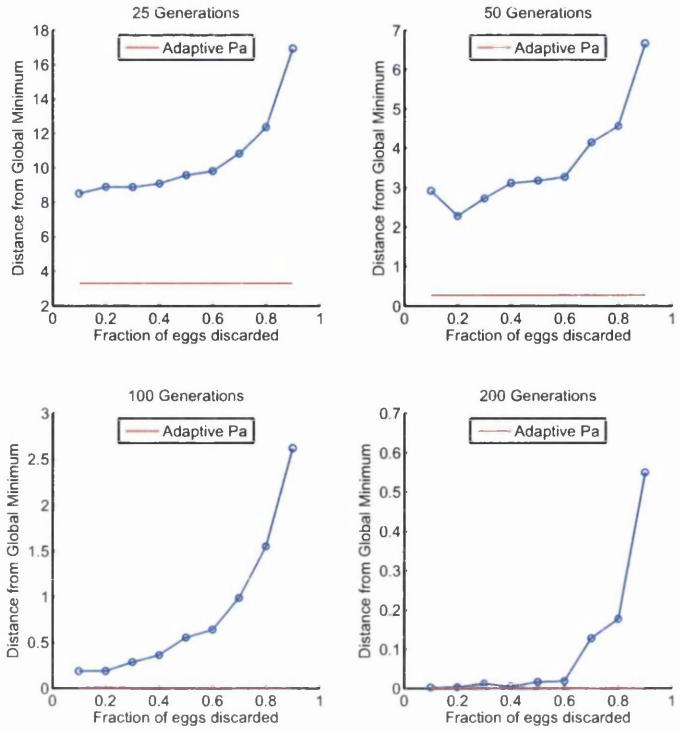


FIGURE 3.11: Fraction of discarded eggs study: Ackley's function, non–shifted, distance to minimum

In all cases the adaptive p_a performed significantly better than a constant p_a maintaining an almost constant diversity throughout the optimisation process.

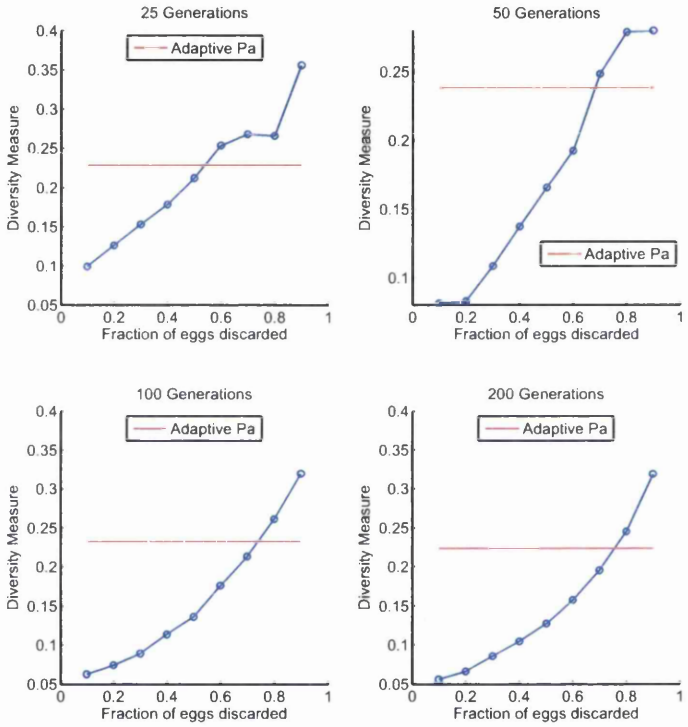


FIGURE 3.12: Fraction of discarded eggs study: Ackley's function, non–shifted, diversity measure

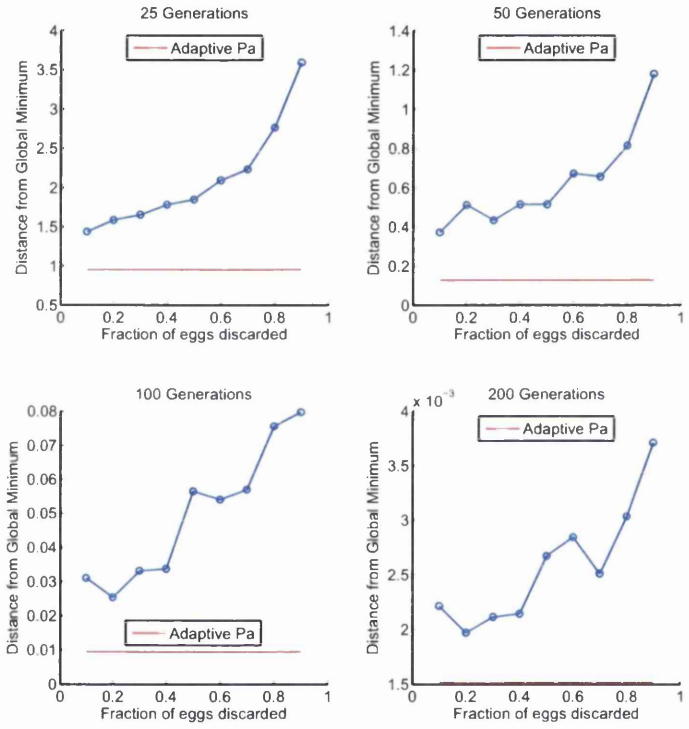


FIGURE 3.13: Fraction of discarded eggs study: de Jong's function, non–shifted, distance to minimum

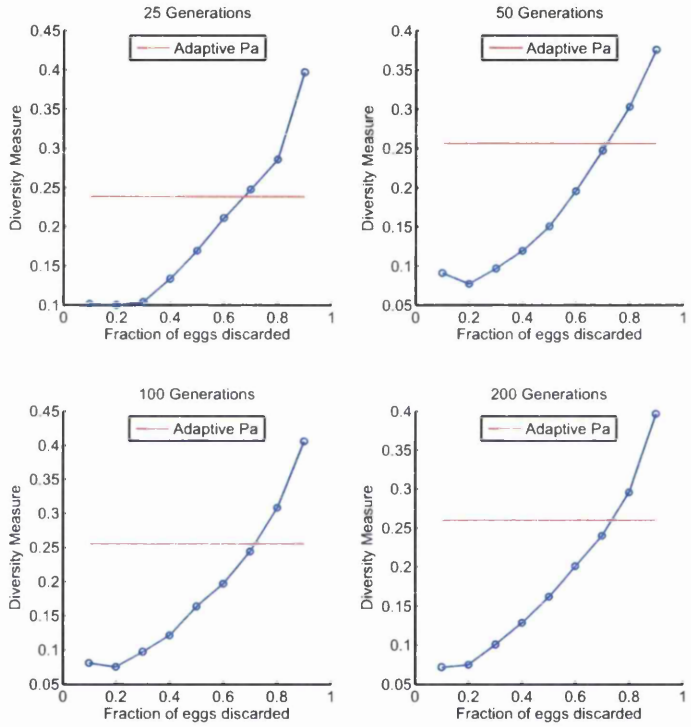


FIGURE 3.14: Fraction of discarded eggs study: de Jong's function, non–shifted, diversity measure

3.4.1.2 Shifted Functions

The results obtained for the shifted Ackley and de Jong functions are presented in Figures 3.15 to 3.18. The effect of increased p_a on diversity was the same as in the non–shifted cases, that is, increased p_a resulted in an increased diversity. However, the effect of p_a on performance was quite different. The general trend was that performance decreased as p_a was increased. The adaptive p_a strategy performed reasonably well, but not as well as some constant values for p_a, when applied to the shifted functions.

For low values of p_a, the dominating search mechanism is the crossover step, in which a new egg is generated at points along lines connecting the eggs. It is not difficult to imagine that, in the case that this crossover step is performed on a distribution of eggs repeatedly, the eggs would naturally move towards the centre of the search space. In the non–shifted functions, this will pull the eggs towards the global optimum whereas, in the shifted cases, this may pull the eggs away from the optimum.

Furthermore, in the current crossover strategy, there is no mechanism for searching different distances along different dimensions. The only mechanism for doing this is the Lévy flight. This is perhaps less important for non–shifted functions, since the global optimum positions have the same value along each dimension.

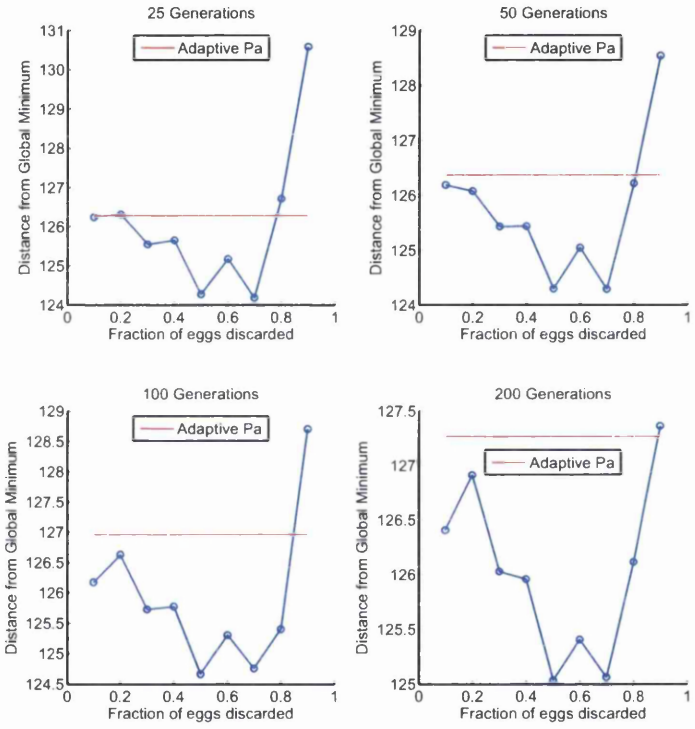


FIGURE 3.15: Fraction of discarded eggs study: Ackley's function, shifted, distance to minimum

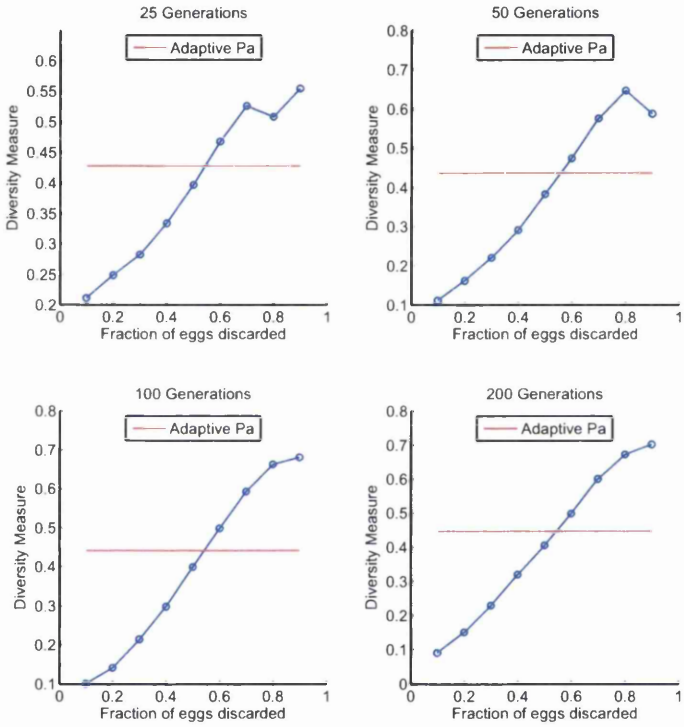


FIGURE 3.16: Fraction of discarded eggs study: Ackley's function, shifted, diversity measure

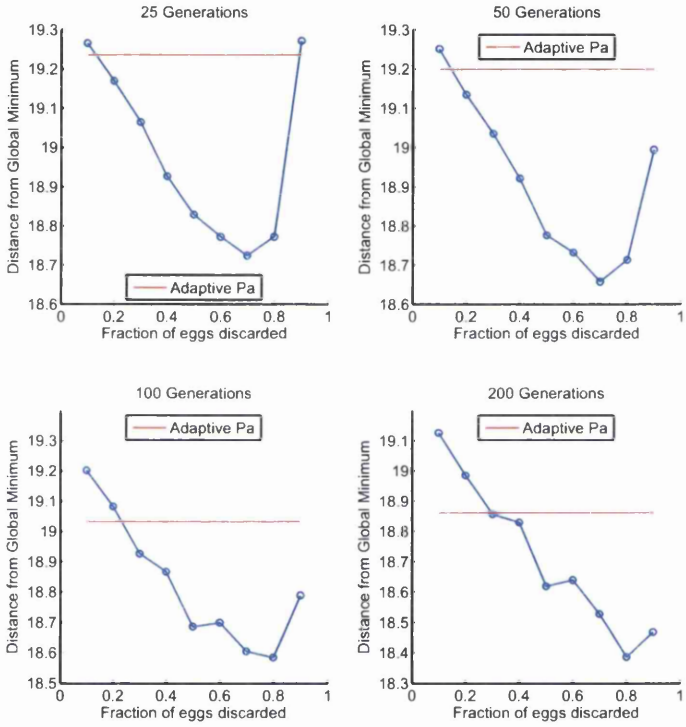


FIGURE 3.17: Fraction of discarded eggs study: de Jong's function, shifted, distance to minimum

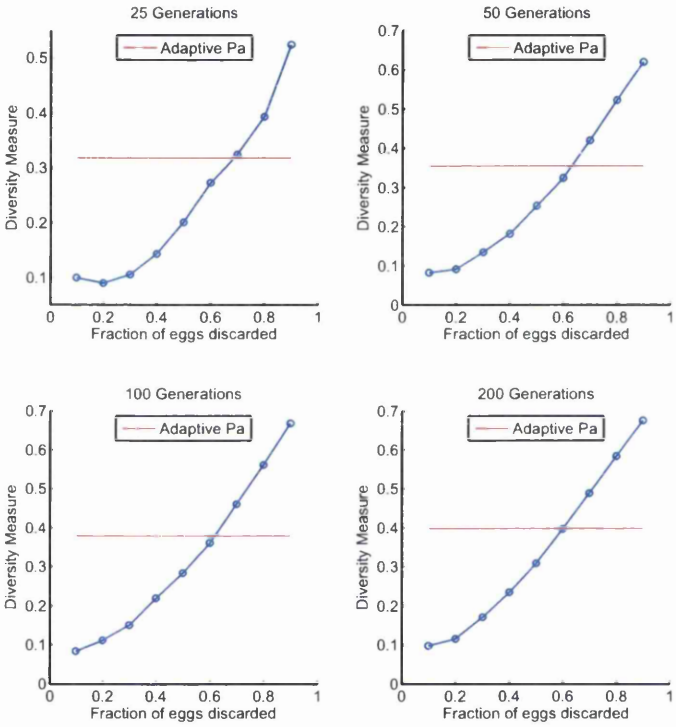


FIGURE 3.18: Fraction of discarded eggs study: de Jong's function, shifted, diversity measure

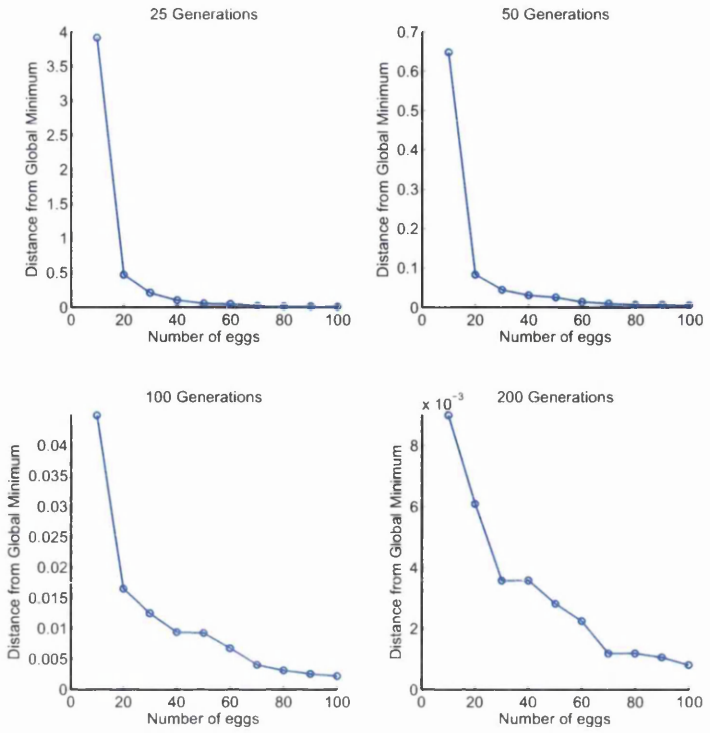


FIGURE 3.19: Number of eggs study: Easom's 2D function, non–shifted, distance to minimum

3.4.2 Number of eggs

The effect of the number of eggs used in MCS was investigated. All other parameters were kept constant to the values suggested by the original study [1]. It would be expected that an increased number of eggs would result in an increased performance, simply because the sampling of the solution space would be better.

3.4.2.1 Non–shifted Functions

The general trend for the non–shifted functions was that increasing the number of eggs increased performance. The results which showed this are presented, for Easom's and Rastrigin's functions, in Figures 3.19 and 3.20. Interestingly, the trend was more apparent in the 25 generation plot and became less significant as the number of generations increased.

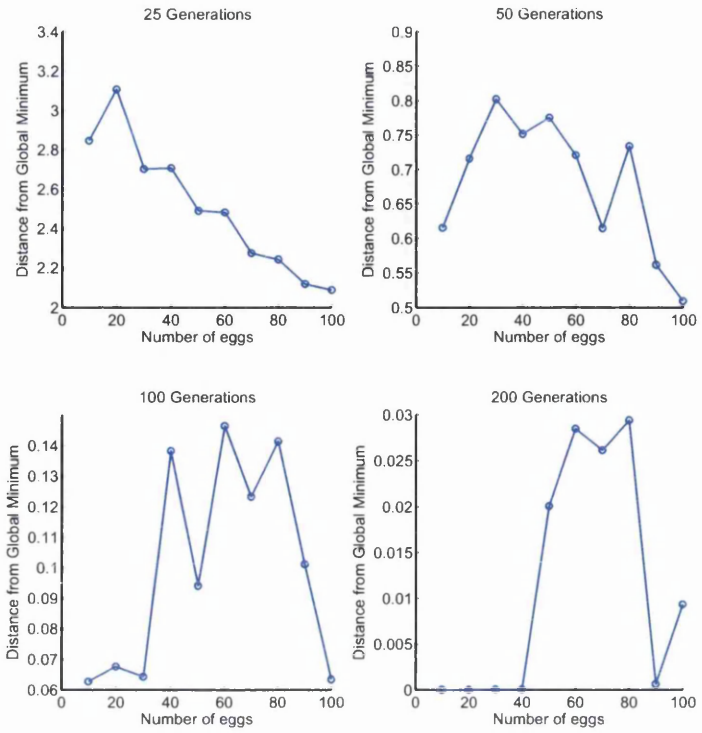


FIGURE 3.20: Number of eggs study: Rastrigin's function, non-shifted, distance to minimum

3.4.2.2 Shifted Functions

For the shifted Easom's and Rastrigin's functions, Figures 3.21 and 3.22 show the increase in performance with increased number of eggs was much more significant than in the non-shifted case. The trend remained apparent as the number of generations increased. This may be an indicator that the initial search space sampling, which can be directly controlled by the number of eggs, is more important in general functions.

3.4.3 Initial Step Size

In MCS, the coefficient, α, is reduced as the algorithm iterates. The value of the initial coefficient is investigated in this section.

It is stated in a number of CS papers [1, 36, 47] that the step size is dependent on the size of the problem. In the following study, this was reflected by expressing the initial step size, $\mathbf{A}$, as a fraction of the bounds of each dimension of the objective function. This was calculated using

$$\mathbf{A} = \frac{\mathbf{L}}{NF} \tag{3.9}$$

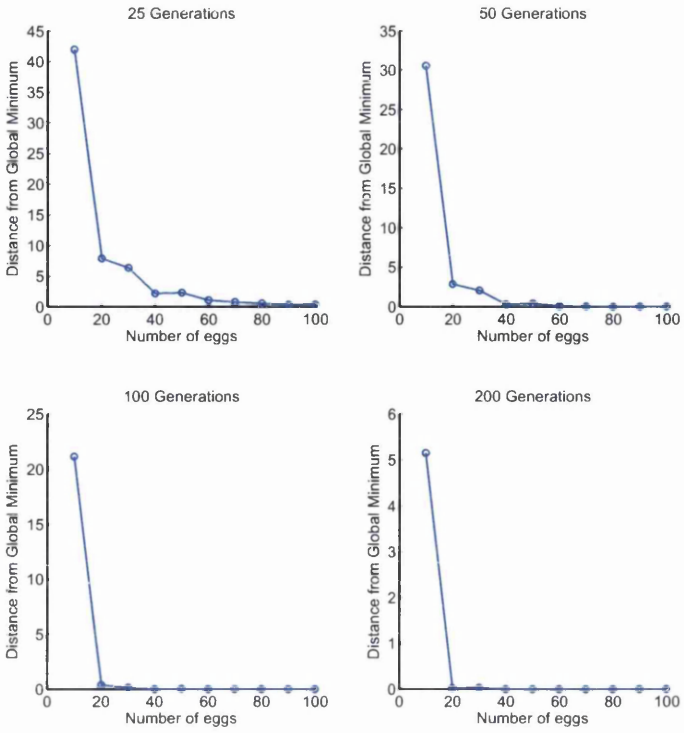


FIGURE 3.21: Number of eggs study: Easom's 2D function, shifted, distance to minimum

where NF is the step size normalisation factor and $\mathbf{L}$ is the range of each search space dimension here. $\mathbf{A}$ is a vector since some problems will have varying ranges for each dimension in the search space.

The step size controls the size of the Lévy flight taken when an egg is discarded. Since the Lévy distribution has an infinite mean, the value of NF needs to be large enough to stop the search diverging outside the search space, but small enough not to limit global exploration.

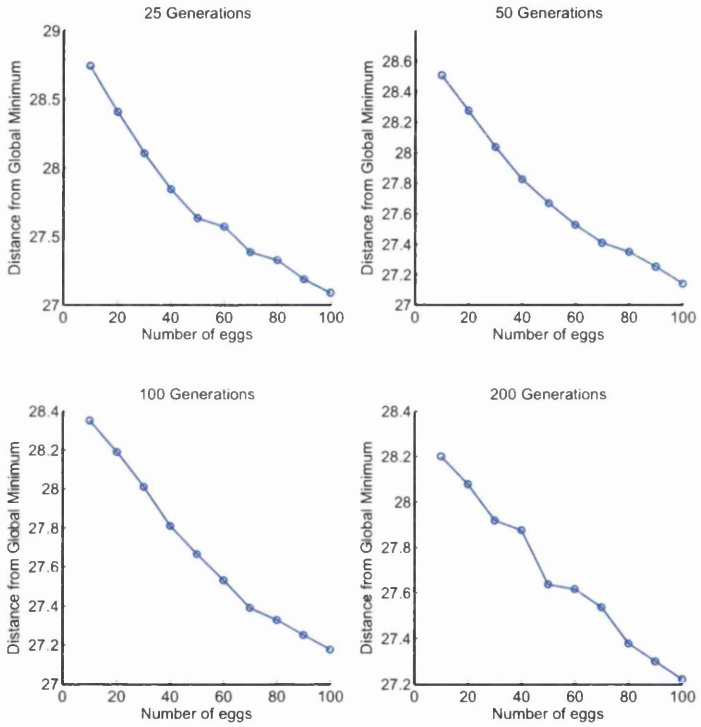


FIGURE 3.22: Number of eggs study: Rastrigin's function, shifted, distance to minimum

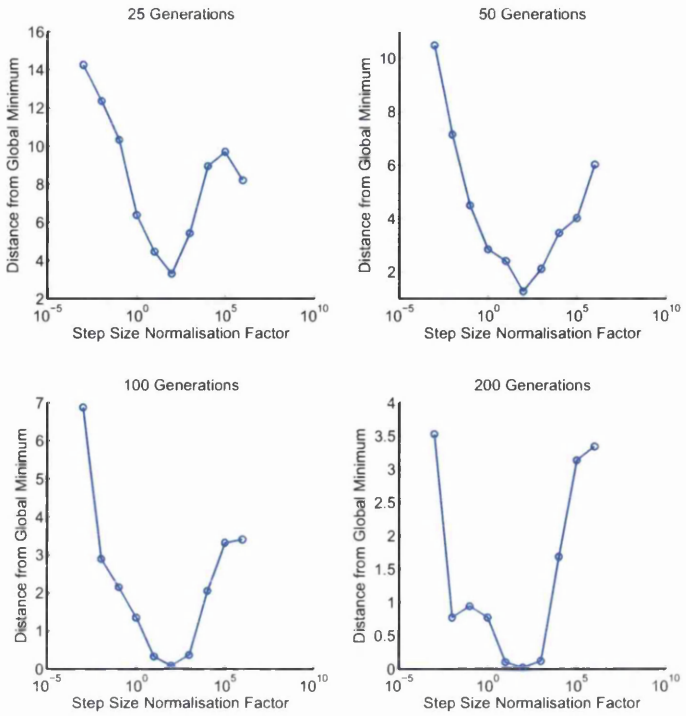


FIGURE 3.23: Initial step size study: Easom's 2D function, non–shifted, distance to minimum

3.4.3.1 Non–shifted Functions

The results for the non–shifted Easom's and Griwank's functions are presented in Figures 3.23 and 3.24. In general, the optimum value for NF appeared to lie between 10^2 and 10^3. In most cases, the performance did not change much as NF was increased beyond these values. However, for Easom's 2D function (Figure 3.23), which has a very sharp global optimum, it was clear that 10^2 was the best value for NF.

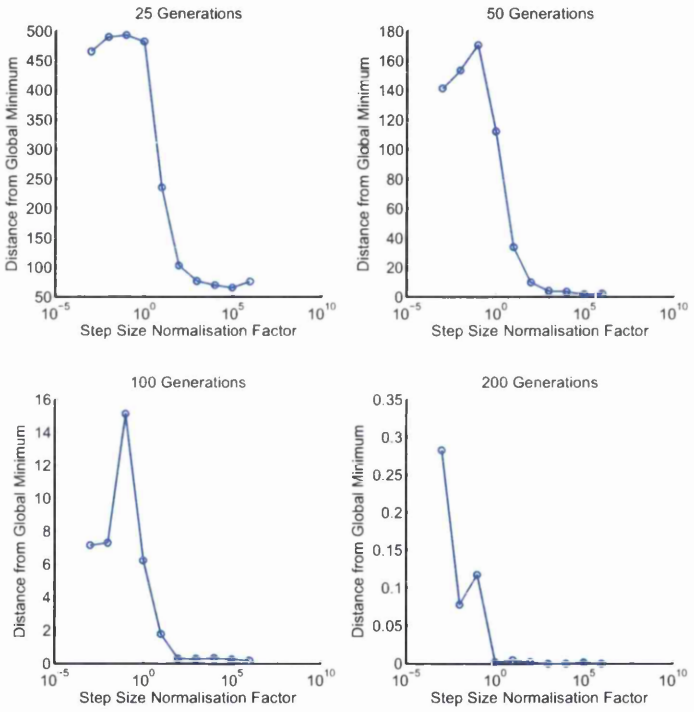


FIGURE 3.24: Initial step size study: Griewank's function, non–shifted, distance to minimum

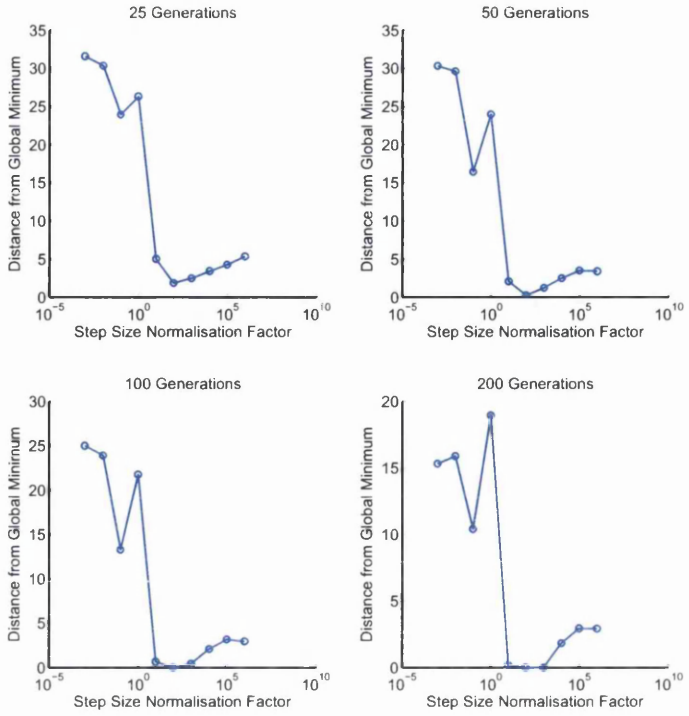


FIGURE 3.25: Initial step size study: Easom's 2D function, shifted, distance to minimum

3.4.3.2 Shifted Functions

Similar findings appeared when the shifted functions were considered. Results are presented for Easom's and Griwank's funtions in Figures 3.25 and 3.26. A slightly higher value of NF seemed to be better in the shifted cases, i.e. 10^3 rather than 10^2. This could be because, as discussed previously, the Lévy flight is the only mechanism which allows different searching along different dimensions. This is more important for the shifted functions than non–shifted. A larger value of NF allows finer searching along independent dimensions.

3.4.4 Step Size Reduction

Each generation in MCS the step size, α, is calculated using

$$\boldsymbol{\alpha} = \frac{\mathbf{A}}{G^{Pwr}} \tag{3.10}$$

where G is the generation number and $\mathbf{A}$ is defined in equation (3.9). This step is included as an attempt to speed up convergence, by encouraging local searching towards the end of the optimisation process. The value $Pwr = 0.5$ is used in the original study [1].

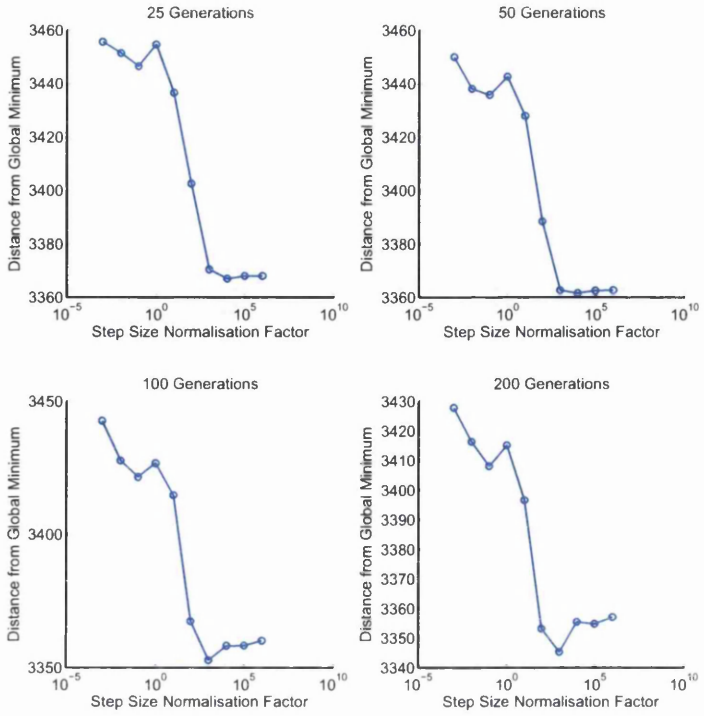


FIGURE 3.26: Initial step size study: Griewank's function, shifted, distance to minimum

The effect of changing this parameter was studied. Giving this too large a value may reduce global searching too quickly, resulting in trappings in local–minima. Specifying a too small value may not have the desired effect at all.

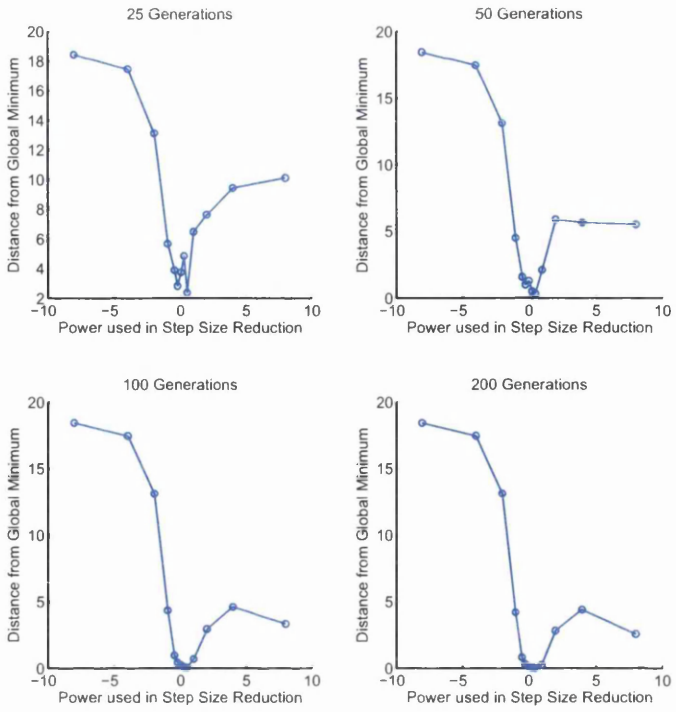


FIGURE 3.27: Step size reduction study: Easom's 2D function, non–shifted, distance to minimum

3.4.4.1 Non–shifted Functions

The majority of the results for non–shifted functions showed the same trend, sample results for Easom's and Rosenbrock's functions are presented in Figures 3.27 and 3.28. Negative values of Pwr, which corresponds to increasing α, showed the worst performance. The performance was almost identical for all negative values considered, there was a sudden increase in performance at $Pwr = 0$, corresponding to constant α, after which the performance did not change significantly. Results obtained for Easom's 2D function, shown in Figure 3.27, were slightly different in that $Pwr = 0.5$ seemed to be the optimum value after which there was a drop in performance.

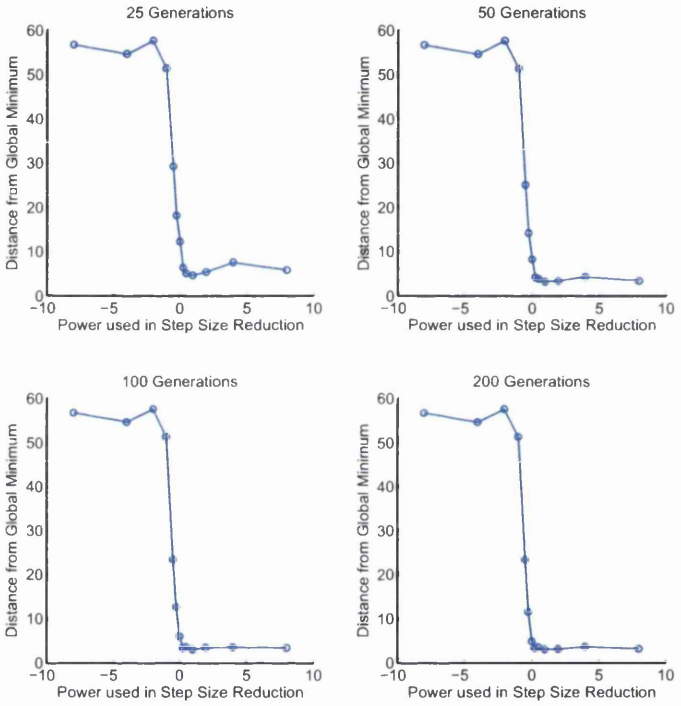


FIGURE 3.28: Step size reduction study: Rosenbrock's function, non–shifted, distance to minimum

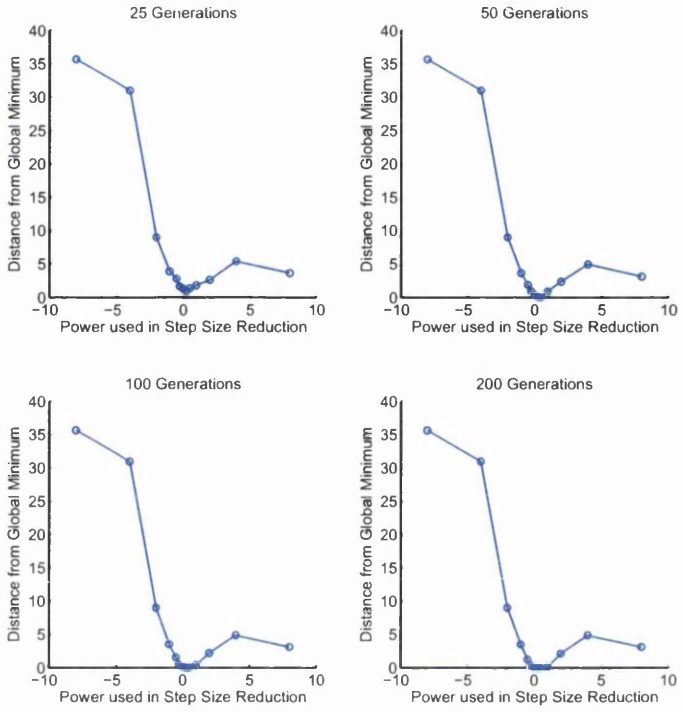


FIGURE 3.29: Step size reduction study: Easom's 2D function, shifted, distance to minimum

3.4.4.2 Shifted Functions

The shifted functions behaved much like Easom's 2D function in the non-shifted case. Results for Easom's and Rosenbrock's function are presented in Figures 3.29 and 3.30 $Pwr = 0.5$ appeared to be the optimum value. In all cases having $Pwr > 0$ was the best strategy, i.e. reducing α as the number of generations increases was better than a constant α.

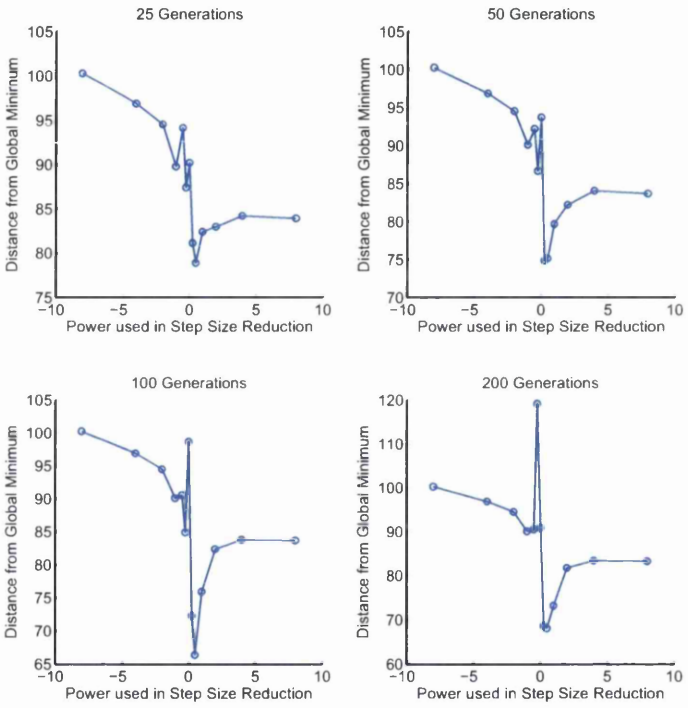


FIGURE 3.30: Step size reduction study: Rosenbrock's function, shifted, distance to minimum

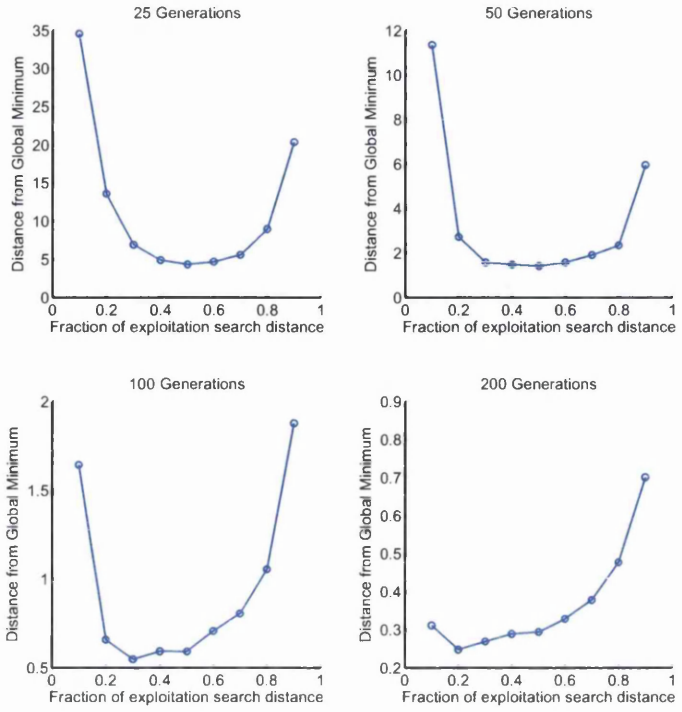


FIGURE 3.31: Crossover fraction study: Ackley's function, non–shifted, distance to minimum

3.4.5 Crossover Fraction

As previously discussed, MCS adds a step to model genetic crossover between the elite cuckoos [1]. For each egg in the elite set, a second egg is picked and a new egg is generated along the line which connects the two eggs. The fraction along the line where the new egg was positioned is taken as the inverse of the golden ratio $\varphi = (1 + \sqrt{5})/2$, such that it is closer to the egg with the best fitness.

In the following study various different fractions were used to see the effect this had on the algorithms performance.

3.4.5.1 Non–shifted Functions

Figures 3.31 and 3.32 show the results of this study for the non–shifted Ackley and de Jong objective functions. The general trend showed roughly equally good performance for the fraction taking values between 0.2 and 0.8, with decreased performance for the maximum and minimum values of the fraction. If the fraction was either too high or too low this would simply generate eggs close to existing eggs which slowed down convergence to the global optimum.

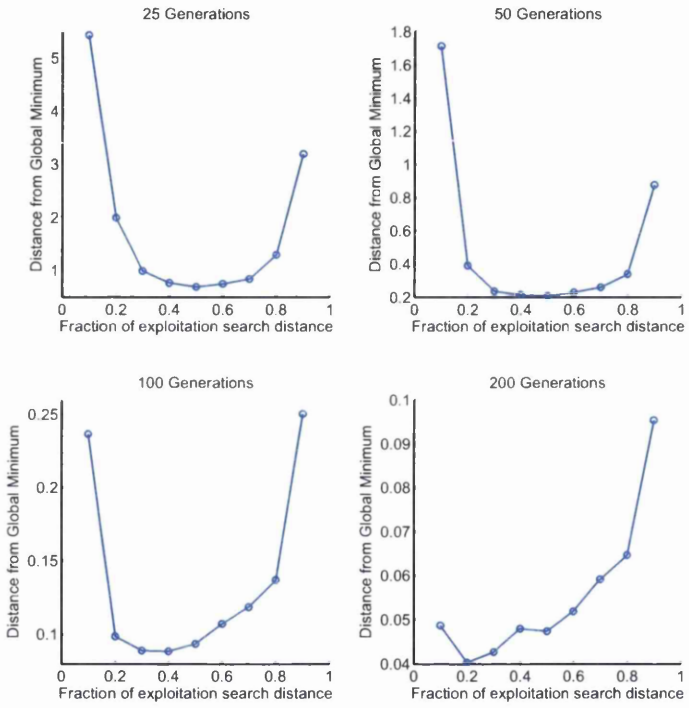


FIGURE 3.32: Crossover fraction study: de Jong's function, non–shifted, distance to minimum

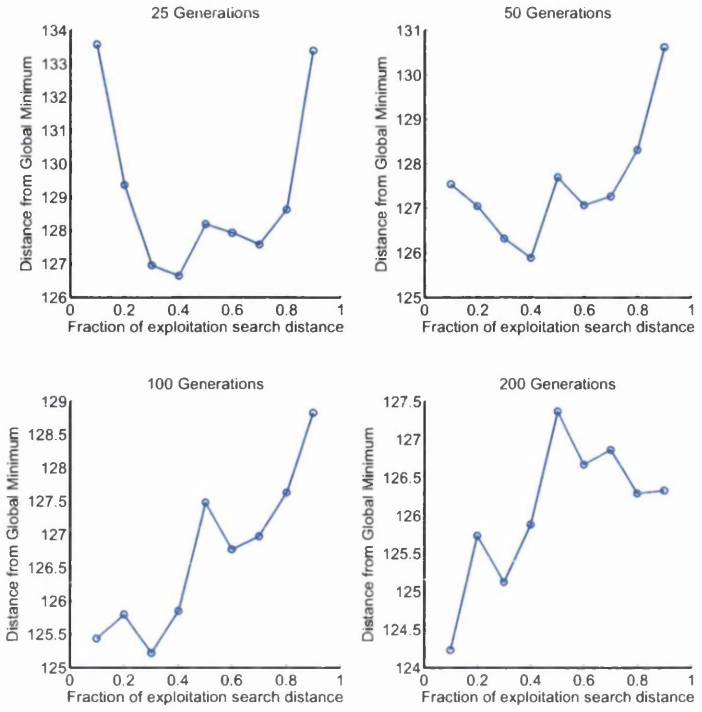


FIGURE 3.33: Crossover fraction study: Ackley's function, shifted, distance to minimum

3.4.5.2 Shifted Functions

Figures 3.33 and 3.34 show the results of this study for the shifted Ackley and de Jong objective functions. The trend here was much less clear. Some functions, such as Ackley's and Griewanks's for example, showed a similar trend to the shifted functions initially, but others did not. As the number of generations increased, the effect of this fraction became even less clear. As previously discussed, independent searching along each different direction is more important in the shifted functions, this may explain the behavior. It may be fruitful to consider different crossover strategies, which allows independent searching along each dimension.

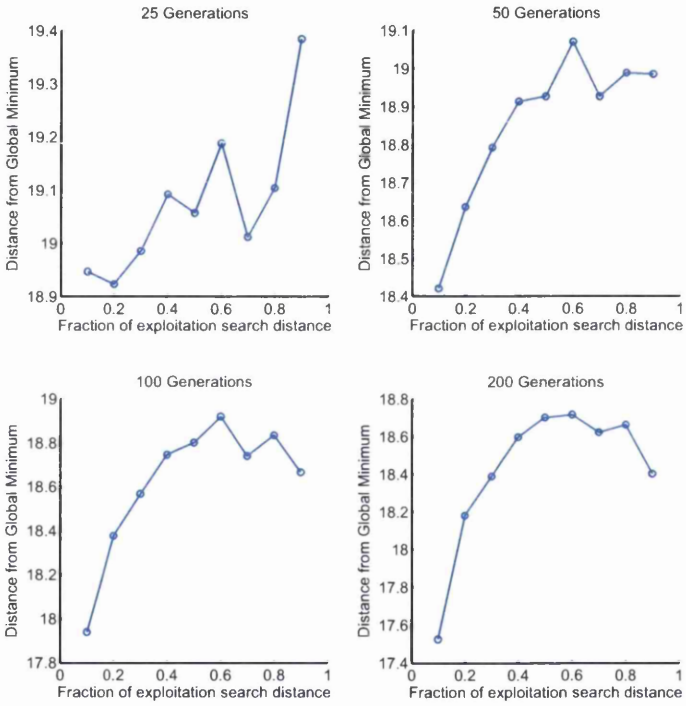


FIGURE 3.34: Crossover fraction study: de Jong's function, shifted, distance to minimum

3.4.6 Crossover Strategy

The crossover strategy in the original MCS algorithm can be considered a weighted mean between two eggs, the fraction considered in the last section being what determines the weight given to each egg [1]. There are many other possible crossover strategies used in various different optimisation algorithms.

A successful crossover scheme was discussed in Section 2.4.1.2 and is used in the DE optimisation algorithm [32].

In MCS the weighted mean is calculated for each dimension in the search space. These means are all entered into the new trial egg. In DE, this is not the case. Once a donor agent, which is equivalent to the weighted mean in MCS, is generated, each element of the agent enters the trial agent with a probability γ, called the crossover probability. Crossover of this type is called binomial crossover and considered to be the most successful strategy in DE [31].

Using the DE crossover terminology, the original MCS can be considered a mean crossover with $\gamma = 1$. The unmodified CS is a mean crossover with $\gamma = 0$ since no crossover takes place. Having a crossover probability less than one means that there is the opportunity for different searching in different dimensions, which may be beneficial for the shifted functions.

In the following study, such a scheme was tested. The algorithm was structured as follows:

1. a random egg from the elite group, considered the target vector, is selected,
2. the weighted mean between that egg and a second random elite egg is calculated to form a donor vector,
3. the target vector is duplicated to make a trial vector, and
4. elements from the donor vector replace elements in the trial vector with a probability γ.

A second strategy similar to that used in DE was also tested. In this strategy, the donor vector was calculated as

$$\mathbf{x}_d = \mathbf{x}_{best} + (\mathbf{x}_{randElite1} - \mathbf{x}_{randElite2})$$

where $\mathbf{x}_{best}$ is the position of the best known egg, $\mathbf{x}_{randElite1}$ and $\mathbf{x}_{randElite2}$ are the positions of two randomly selected elite eggs and $\mathbf{x}_d$ is the donor vector. This strategy is one of the most competitive DE strategies [31].

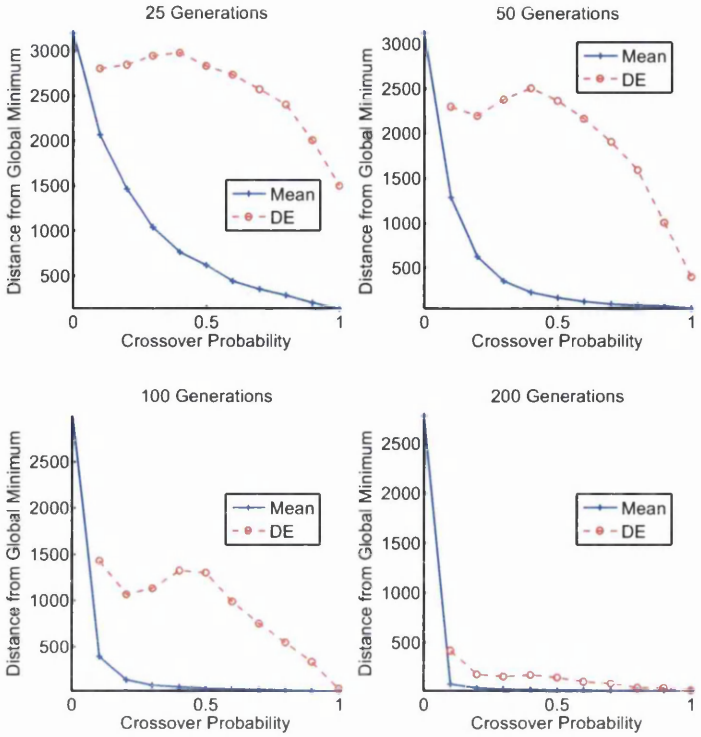


FIGURE 3.35: Crossover strategy study: Griewank's function, non–shifted, distance to minimum

All other parameters were kept constant to the values suggested in the original study [1], and the algorithm was run with 50 eggs.

3.4.6.1 Non–shifted Functions

Figures 3.35 and 3.36 show the results of this study for the non–shifted Griewank and Rastrigin functions. For each function the weighted mean was a much more successful strategy, and as γ increased the performance increased. The best value was $\gamma = 1$. This reflects the statements made earlier that, for the non–shifted functions, independent searching along each direction is not important. In fact, it seemed that there was a disadvantage with the independent searching.

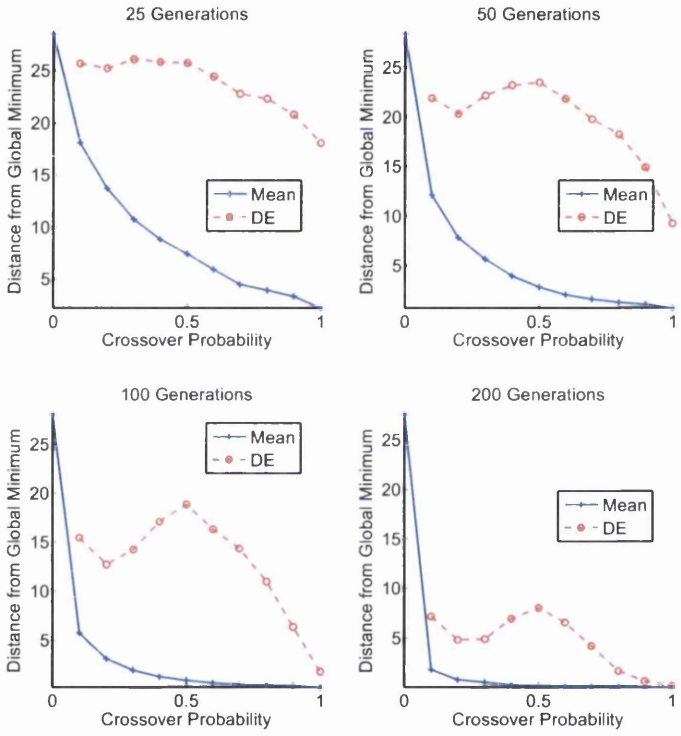


FIGURE 3.36: Crossover strategy study: Rastrigin's function, non–shifted, distance to minimum

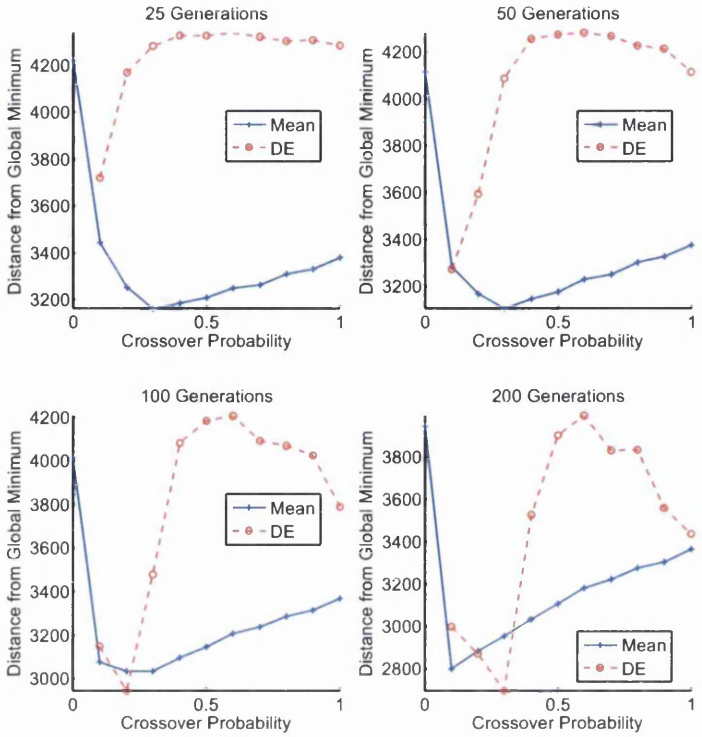


FIGURE 3.37: Crossover strategy study: Griewank's function, shifted, distance to minimum

3.4.6.2 Shifted Functions

The shifted functions showed a different result to the non–shifted functions. Figures 3.37 and 3.38 show the results for Griewank's and Rastrigin's functions. It can be seen that, for most functions, the weighted mean crossover strategy still worked best. The difference between the shifted and non–shifted functions was that, in the shifted functions, there seemed to be some advantage to having $0.2 < \gamma < 0.8$. This, again, was not surprising, considering that the shifted functions have global optima at random positions in each search dimension.

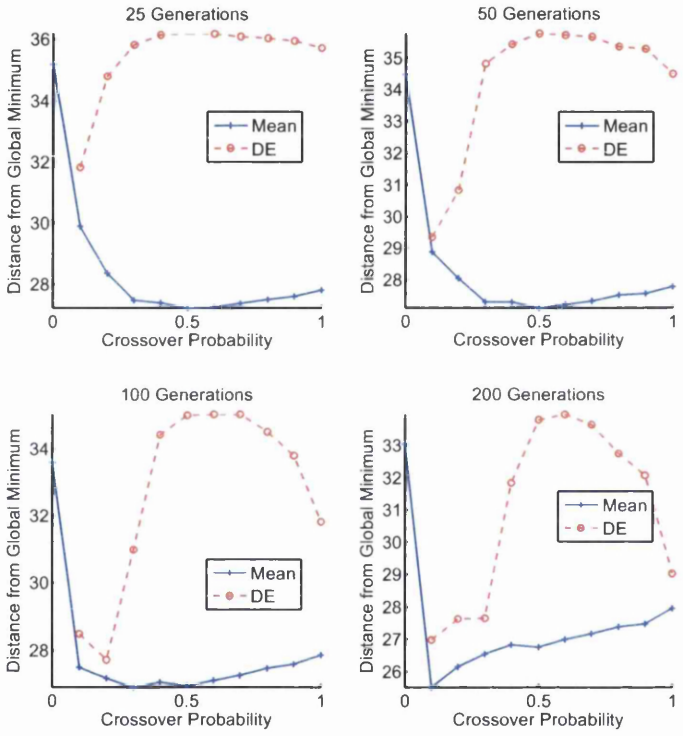


FIGURE 3.38: Crossover strategy study: Rastrigin's function, shifted, distance to minimum

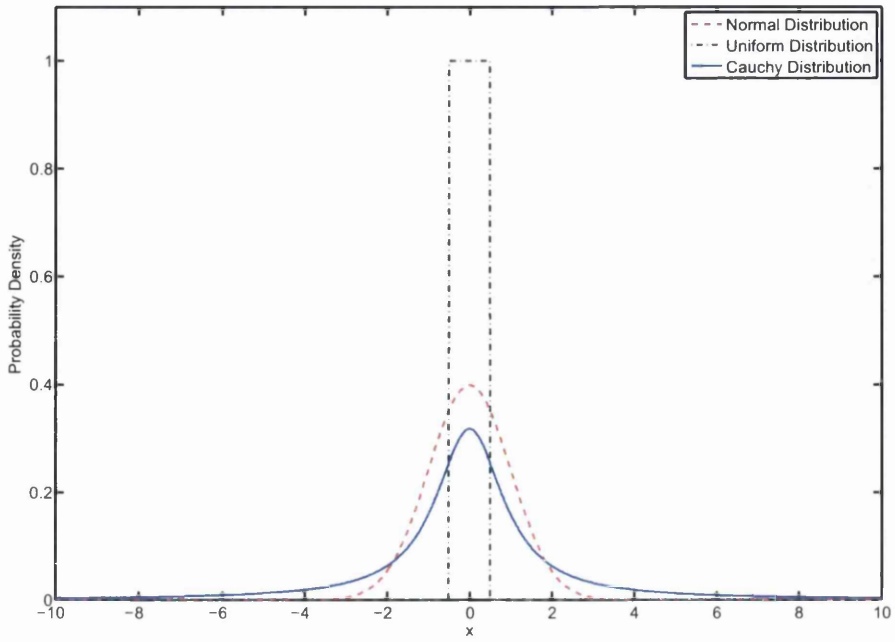


FIGURE 3.39: Probability density plotted for normal, uniform and Cauchy distributions

3.4.7 Random Walk Strategy

In MCS, a Lévy flight is used to move each of the discarded eggs. As discussed in Section 2.4.1.4, the Lévy flight random walk strategy makes CS stand out compared with other optimisation algorithms [49]. The Lévy flight is not the only possible random walk strategy.

A number of strategies were compared. All other tuning parameters remained constant, as in the other studies. Five different random walk strategies were compared. The first strategy was the Lévy flight, as implemented in Yang and Deb's [36] MATLAB [17] CS implementation which is available [47]. Their implementation uses Mantegna's algorithm for simulating a Lévy flight [68]. Cauchy and Gaussian walks are random walks where each steps is taken from Cauchy and Gaussian distributions respectively. These were generated using a stable random number generator [69]. The Cauchy walk results in a Lévy flight. The Normal distribution was a Gaussian walk in which each step was taken from the MATLAB [17] implementation of a Gaussian distribution. The uniform distribution was a random walk where each step was taken from the MATLAB [17] implementation of a uniform distribution. Figure 3.39 compares the probability density functions of the three different distributions. In the figure, the x axis can be thought of as representing the size of a step in a random walk. Note how the Cauchy distribution has a probability of large steps, whereas the normal and uniform distributions do not.

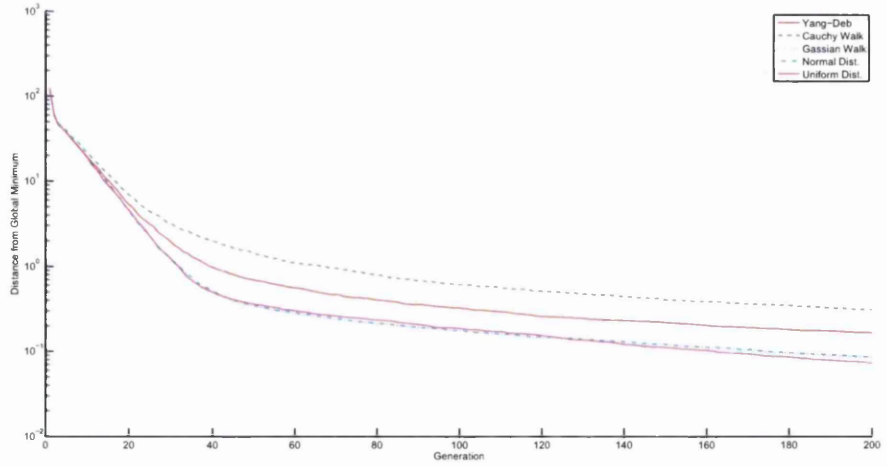


FIGURE 3.40: Random walk strategy study: Ackley's function, non–shifted, distance to minimum

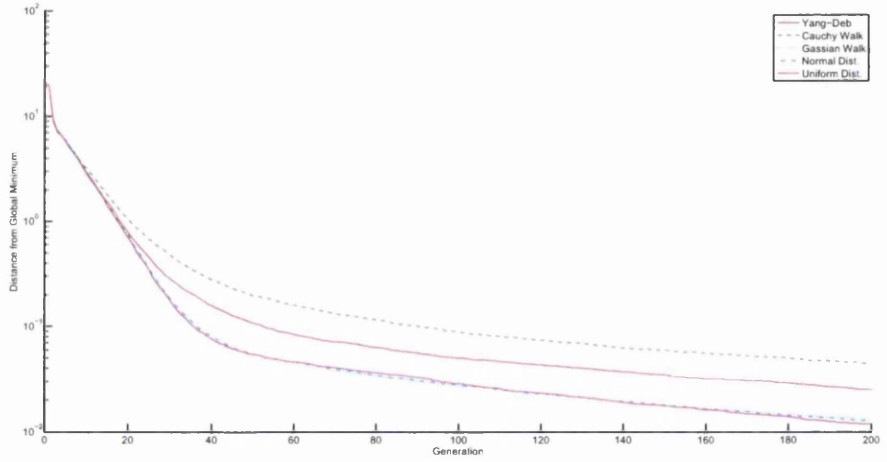


FIGURE 3.41: Random walk strategy study: de Jong's function, non–shifted, distance to minimum

3.4.7.1 Non–shifted Functions

Sample results, for this study, are plotted in Figures 3.40 to 3.43. In almost every case, the two Gaussian walks and the uniform distribution performed the best for non–shifted functions. This is surprising since it is often stated that the Lévy flight is an optimum search pattern [51]. Perhaps this result was because the non–Cauchy walks have a higher likelihood of searching the same distance in each direction, since the probability distributions are symmetrical. This could be an advantage, when the objective functions have the same optimum value for each search dimension.

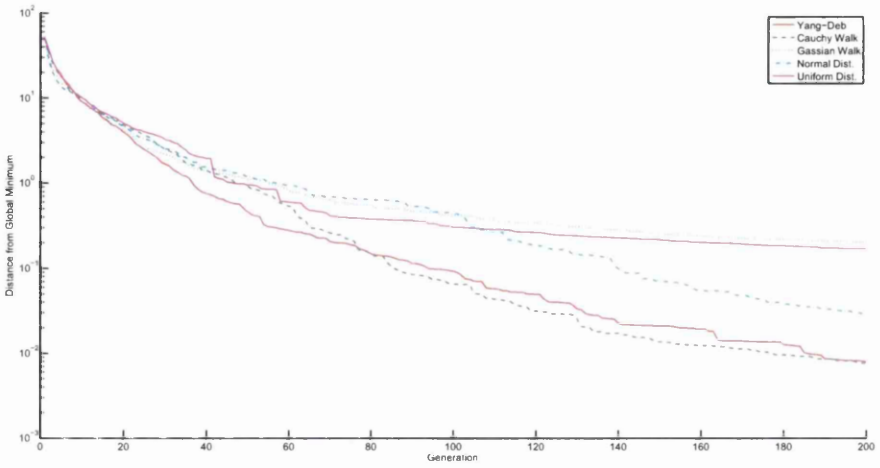


FIGURE 3.42: Random walk strategy study: Easom's 2D function, non–shifted, distance to minimum

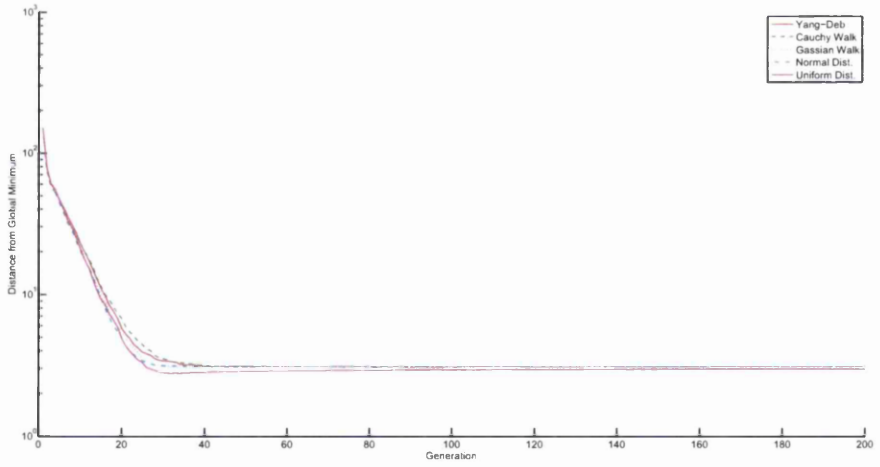


FIGURE 3.43: Random walk strategy study: Rosenbrock's function, non–shifted, distance to minimum

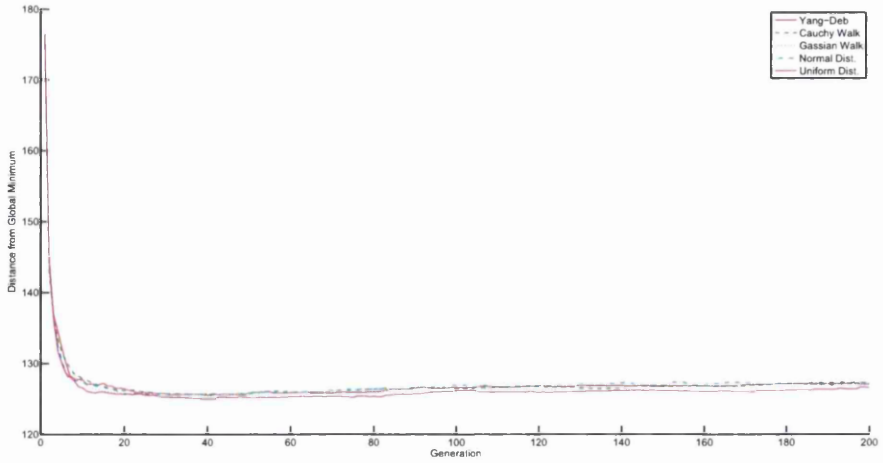


FIGURE 3.44: Random walk strategy study: Ackley's function, shifted, distance to minimum

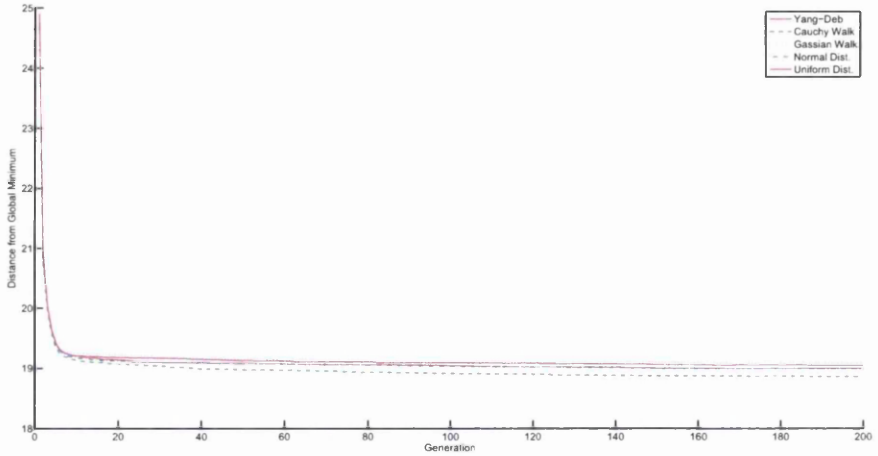


FIGURE 3.45: Random walk strategy study: de Jong's function, shifted, distance to minimum

3.4.7.2 Shifted Functions

Figures 3.44 to 3.47 show sample results of the flight strategy study for the shifted functions. In these cases, it was harder to tell the difference between each strategy. Easom's 2D function, results plotted in Figure 3.46, and Rosenbrock's function, results plotted in Figure 3.47, had the most apparent differences. In these cases, the Cauchy walks showed the better performance. This further supported the conclusions made when considering the results from the non-shifted functions in this study.

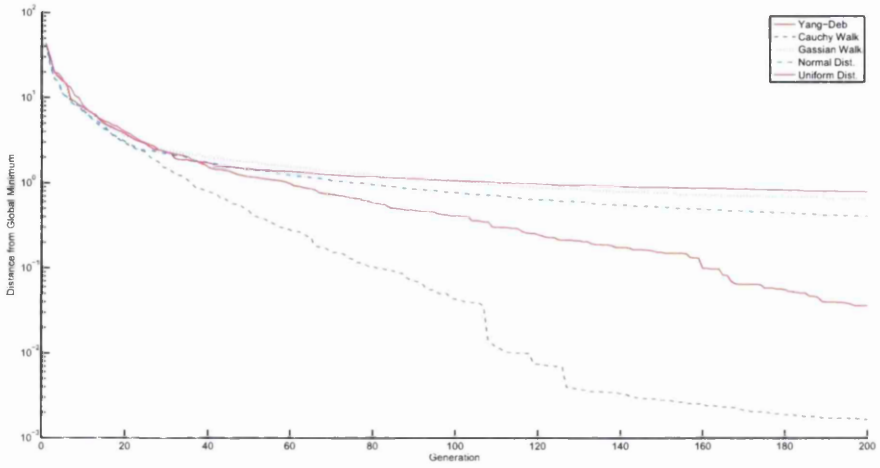


FIGURE 3.46: Random walk strategy study: Easom's 2D function, shifted, distance to minimum

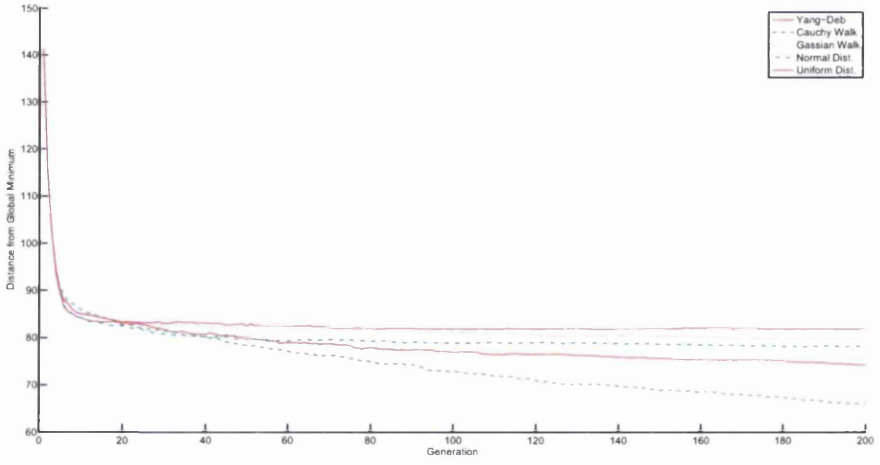


FIGURE 3.47: Random walk strategy study: Rosenbrock's function, shifted, distance to minimum

3.4.8 Conclusions

Conclusions reached by using the two different types of function, shifted and non–shifted, gave very different outcomes. It can be easily argued that shifted functions are a more suitable benchmark and, thus, the conclusions made in the original MCS study [1] using non–shifted functions come under question. This could explain the different findings presented by Bhargava et al [6], a point which will be discussed in more detail later in this chapter. With this in mind, it was considered appropriate to use the results from the shifted function studies to tune MCS such that it is better suited to more general functions. In all further analysis, only shifted functions will be considered as benchmarks.

When considering the value of p_a, the results from the shifted functions suggested that a value of p_a of around 0.75 seems to be the best. The adaptive p_a did not perform well enough, for general functions, to be used.

The number of eggs has a direct effect on the number of function evaluations required. More eggs results in more evaluations in each generation. The results from the shifted function showed that a larger population of eggs is better. The trend between the number of eggs and performance remained almost unchanged as the algorithm is run longer. There is a possibility that this increase in performance was due to a better sampling of the search space. This raised the question: is it possible to start with a large population and then reduce the number of eggs each generation?

A study was carried out in which the starting population of eggs was 50. At each generation, a number of the worst eggs were eliminated from the search so that the population was decreased. The population was decreased by different amounts each generation, to ascertain the effect on performance. When the population reaches 10 eggs, no further reduction is made.

Figures 3.48 and 3.49 show sample results of this study. Although there was a significant drop in performance if 10 eggs are eliminated each generation, there was not a significant decrease if only 1 egg is eliminated each generation. Even after the algorithm was left to run for 200 generations, the difference in the performance was very small between the algorithm with no reduction and the one where a single egg is eliminated each generation. The savings in number of function evaluations will be significant, so it is worth including this step in the algorithm.

The studies into the step size was very clear. The values of $NF = 10^3$ and $Pwr = 0.5$ were clearly the best options.

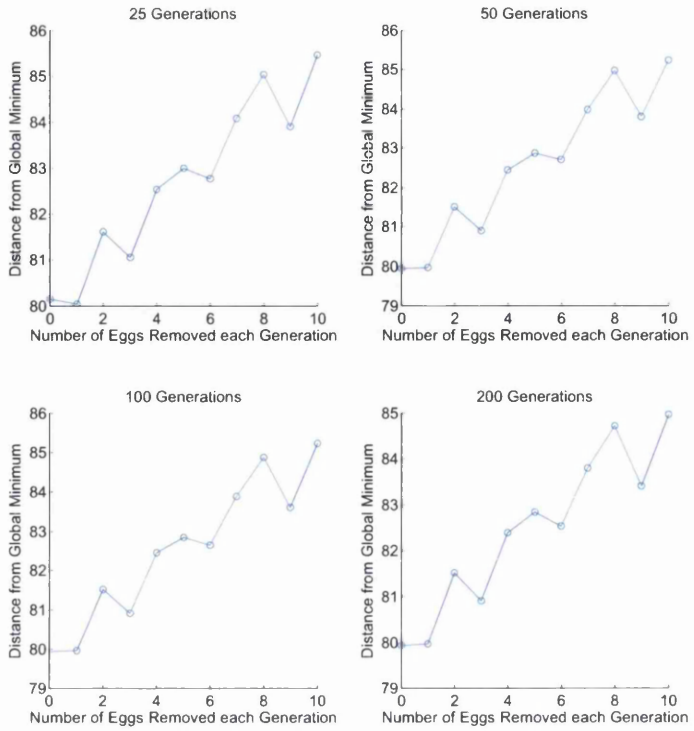


FIGURE 3.48: Decreasing population strategy study: Ackley's function, shifted, distance to minimum

When considering different crossover fractions and strategies, there was a clear difference between the shifted and non-shifted functions. The value of the crossover seemed to be unimportant, as long as it was between 0.2 and 0.8. Since the global optima of general functions do not necessarily have the same ordinate value in each dimension, there is an advantage to having a crossover rate less than one.

Rather than use the Binomial crossover strategy from DE [32], to further encourage independent searching along each dimension in the weighted mean operation, a different weight is now applied to each dimension. This is done using a different random number between 0 and 1 as the crossover fraction for each dimension.

The best choice for the random walk strategy was the Lévy flight implementation in Yang and Deb's MATLAB CS code, which offered best performance in terms of the optimisation and, it turns out, CPU cost of generating a Lévy flight.

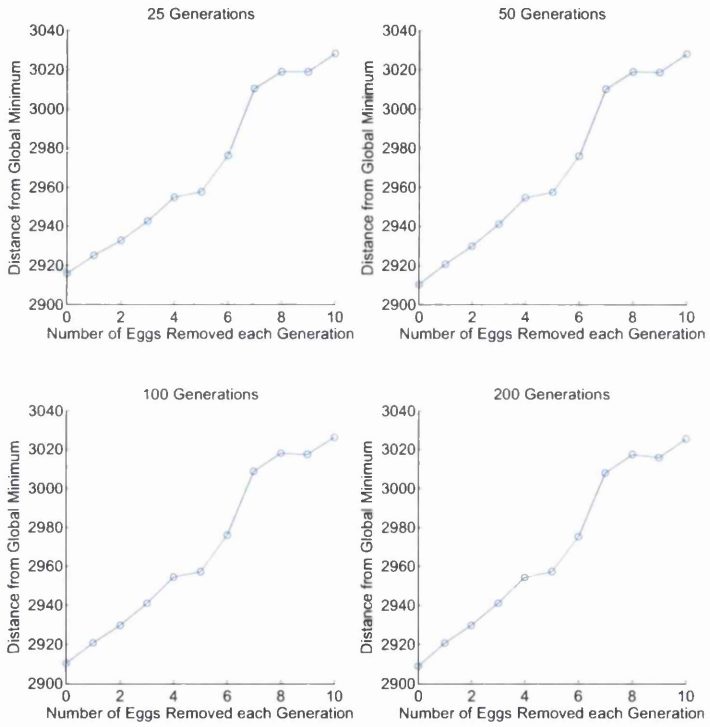


FIGURE 3.49: Decreasing population strategy study: Griewank's function, shifted, distance to minimum

In summary, for best performance

- Set $p_a = 0.7$
- Use as many eggs as can be afforded in the first generation then remove the worst egg each generation down to a minimum of 10 eggs
- Set $NF = 10^3$
- Set $Pwr = 0.5$
- In the weighted mean calculation take a random fraction for each search dimension
- Use Yang and Deb's Lévy flight

3.5 Adding a Biased Random Walk

The study by Bhargava et al [6] clearly showed that CS outperforms MCS in their application. This appeared surprising since, at the most basic level, MCS is CS with some extra steps. No insight was provided into why this difference occurs, but it is stated that it is the MATLAB implementation of CS developed by Yang and Deb [47] which is employed.

The MATLAB implementation of CS has three main sub–routines that are performed at each generation, viz. `get cuckoos`, `get best nest` and `empty nests`. The sub–routine `get cuckoos` generates new eggs by performing a Lévy flight step from the previous location of each egg. The sub–routine `get best nest` calculates the objective function at each of these new locations and determines the best solution found so far. Once this is done the sub–routine `empty nests` is applied. The sub–routine essentially takes random pairs of eggs, calculates the vector between them, multiplies this by a random number and adds it to the eggs originally marked as candidates for discarding. It was clear that this process is very similar to the DE [32] strategy detailed in Section 2.4.1.2. The process is not mentioned in the original CS paper [36] and is only touched upon in the second CS paper [47], where is is referred to as being a biased random walk. No process of this type is included in MCS [1].

To investigate the effect of the inclusion of a biased random walk, a number of different algorithms were tested. The published CS implementation [47] was used without modification, as was the MCS implementation. In addition, the CS implementation was modified to create two new algorithms. The first incorporated only the `get cuckoos` sub–routine, meaning that new eggs were generated by Lévy flights only and not by a

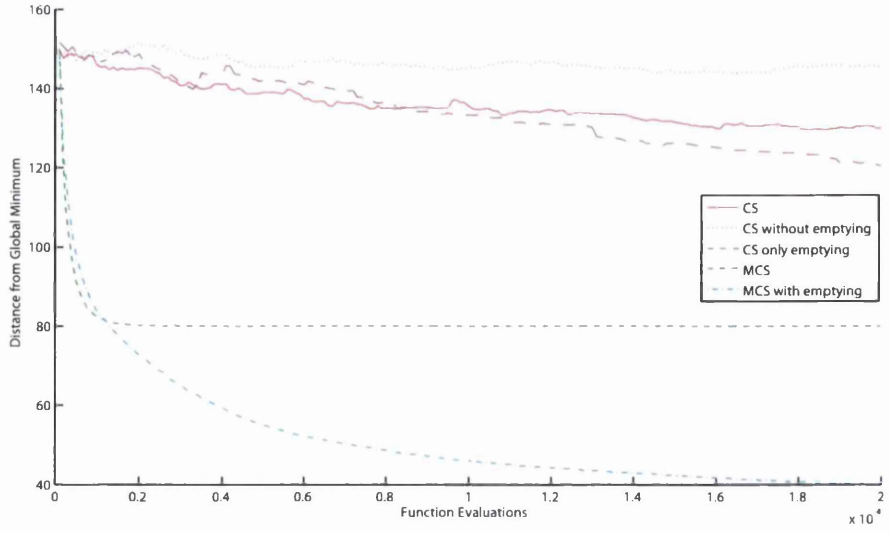


FIGURE 3.50: Different cuckoo search algorithms: Ackley's function, shifted, distance to minimum

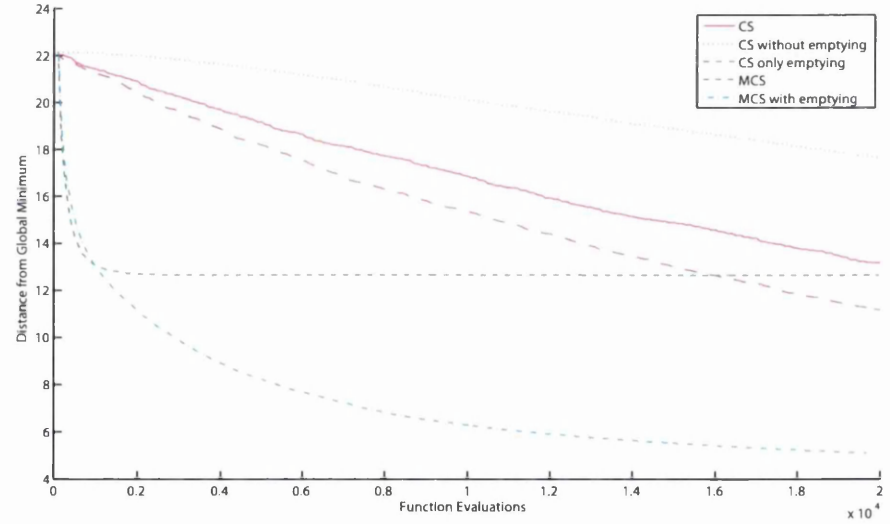


FIGURE 3.51: Different cuckoo search algorithms: de Jong's function, shifted, distance to minimum

biased random walk. The second incorporated the `empty nests` routine only, meaning that new eggs were only generated using the biased random walk. A final algorithm was the MCS implementation with the `empty nests` sub-routine added as an additional step at the end of each generation. The number of eggs used for each algorithm was 50. Results are presented, in Figures 3.50 to 3.55.

These results gave insight into the possible reasons behind the findings of Bhargava et al [6]. Firstly, MCS without the `empty nests` sub-routine had a fast initial convergence,

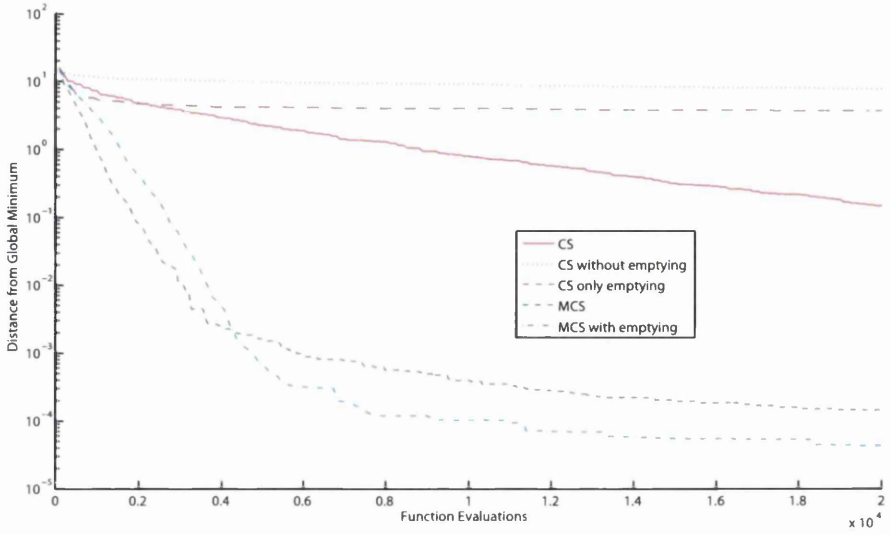


FIGURE 3.52: Different cuckoo search algorithms: Easom's 2D function, shifted, distance to minimum

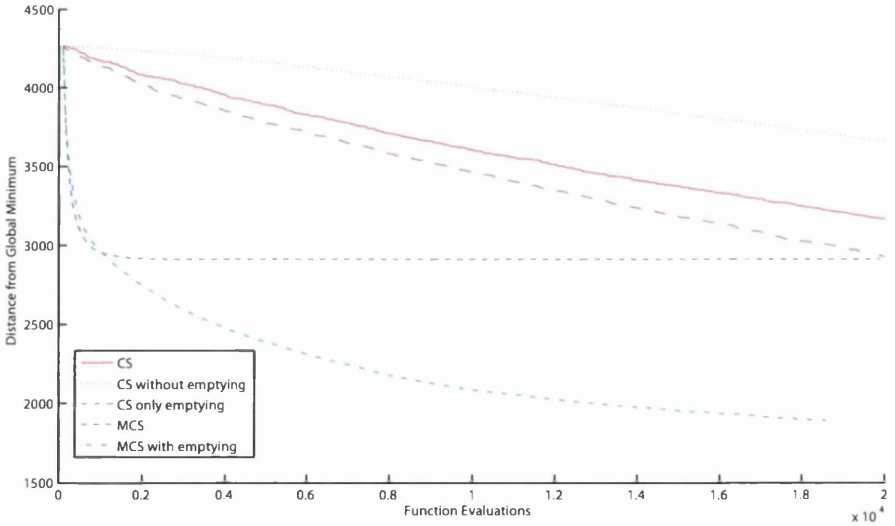


FIGURE 3.53: Different cuckoo search algorithms: Griewank's function, shifted, distance to minimum

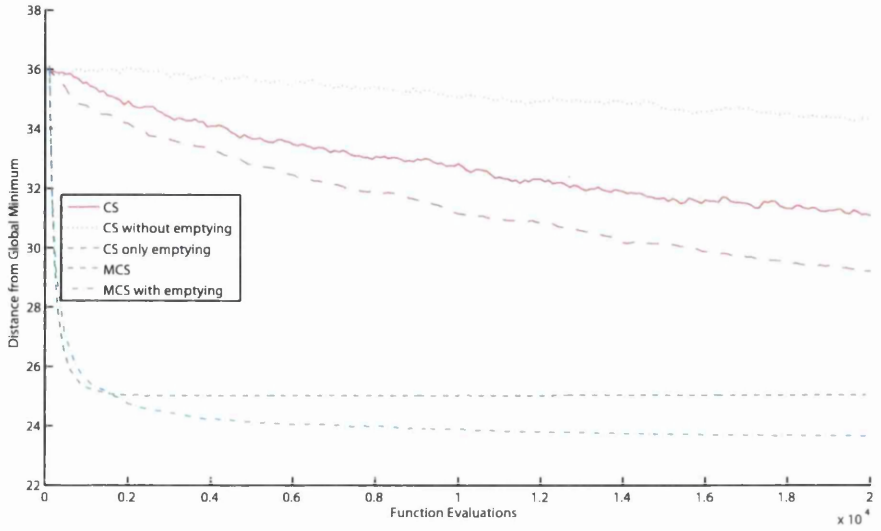


FIGURE 3.54: Different cuckoo search algorithms: Rastrigin's function, shifted, distance to minimum

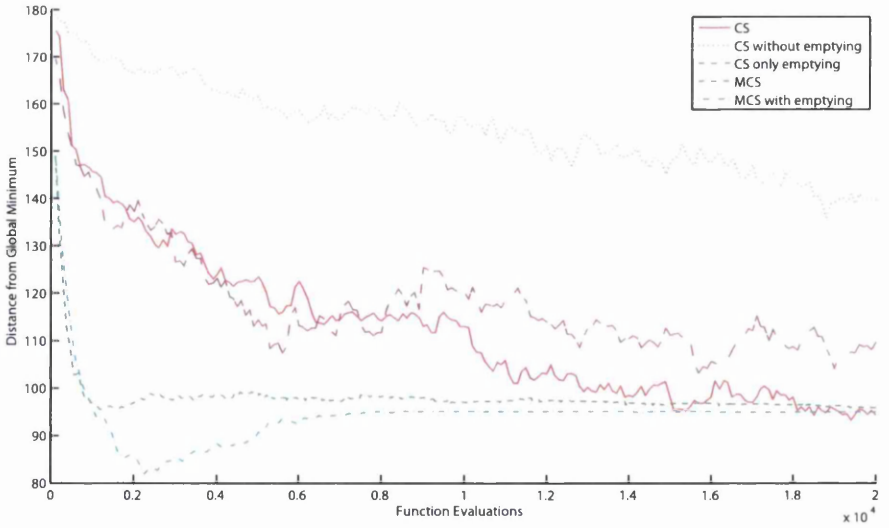


FIGURE 3.55: Different cuckoo search algorithms: Rosenbrock's function, shifted, distance to minimum

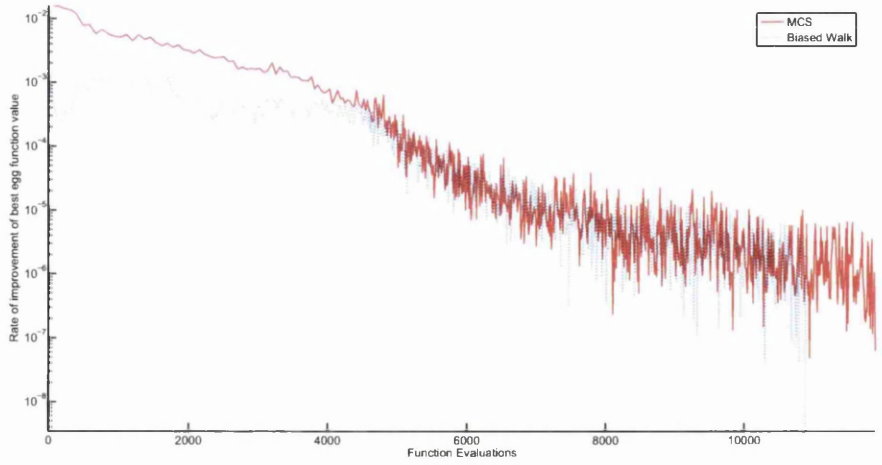


FIGURE 3.56: The relative effect of the MCS and biased walk algorithms: Ackley's function, shifted

but did get trapped in local–minima. The addition of the emptying sub–routine fixed this problem and the modified method had a fast initial convergence and found the best solution in every case. For Rosenbrock's function, Figure 3.55, CS also performed well. What was really interesting is that when the `empty nests` sub–routine was used alone it performed better than CS in every case, apart from Rosenbrock's function. Simply using Lévy flights alone resulted in a very poor performance.

These results showed that, in Yang and Deb's implementation [47], the most influential routine was `empty nests`. The CS implementation used in the original MCS validation does not include this `empty nests` routine, which almost certainly explains the findings of Bhargava et al [6].

As previously mentioned, the `empty nests` routine could be considered a specific example of a DE strategy, which makes it easy to explain why the sub–routine improves both CS and MCS so much.

Figures 3.56 to 3.61 are an attempt to quantify the effect of the biased random walk on overall performance. As described previously, each generation the algorithm applied the standard MCS followed by a random biased walk. The reduction in objective function value along with the number of function evaluations for each step was recorded each generation. The improvement was divided by the number of function evaluations to yield an objective function value reduction rate. This reduction rate was then plotted against the number of function evaluations performed for the standard MCS and the biased random walk. The graphs show that, initially, MCS yielded the fastest reduction in function value, with the biased walk becoming more significant as the number of function evaluations increases.

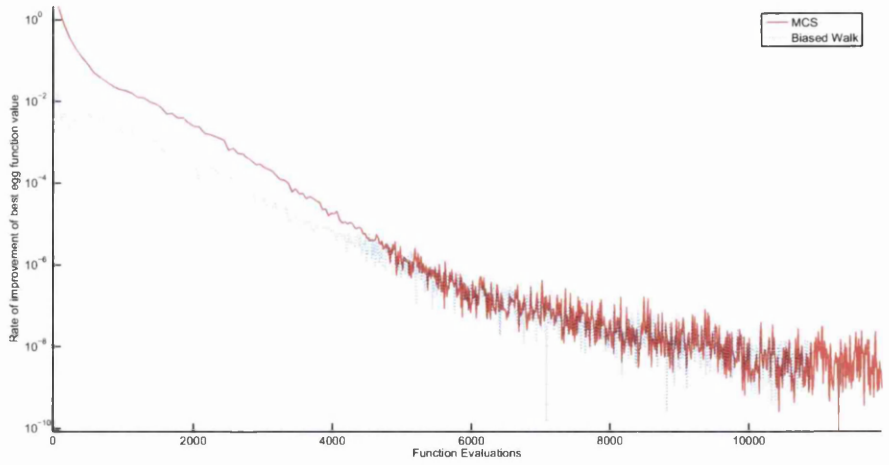


FIGURE 3.57: The relative effect of the MCS and biased walk algorithms: de Jong's function, shifted

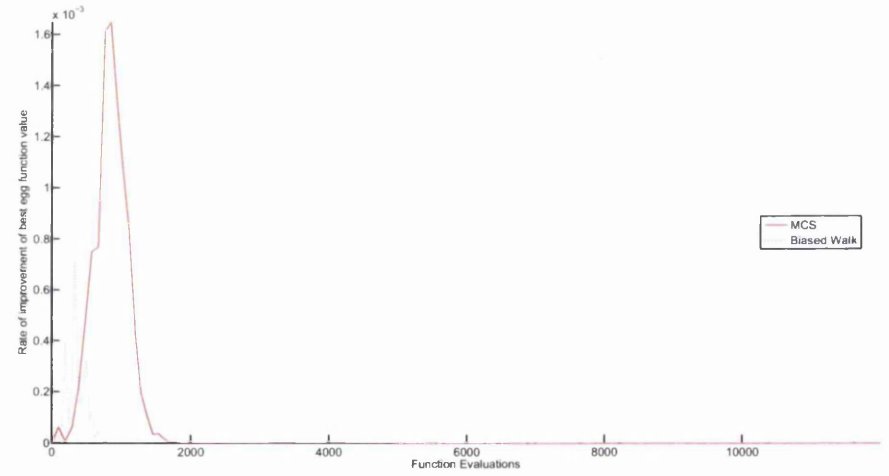


FIGURE 3.58: The relative effect of the MCS and biased walk algorithms: Easom's 2D function, shifted

These findings are very recent and the mesh optimisation application of MCS presented in this thesis does not include the `empty nests` routine. However, the open source project `modified-cs` [64] has been updated to include this step.

3.6 Further Comparison with Existing Algorithms

Up to this point in the thesis, algorithms have been compared by measuring the distance from the coordinates of the lowest found objective function value to the known global minimum. This measure of success is the most frequently used in optimisation literature.

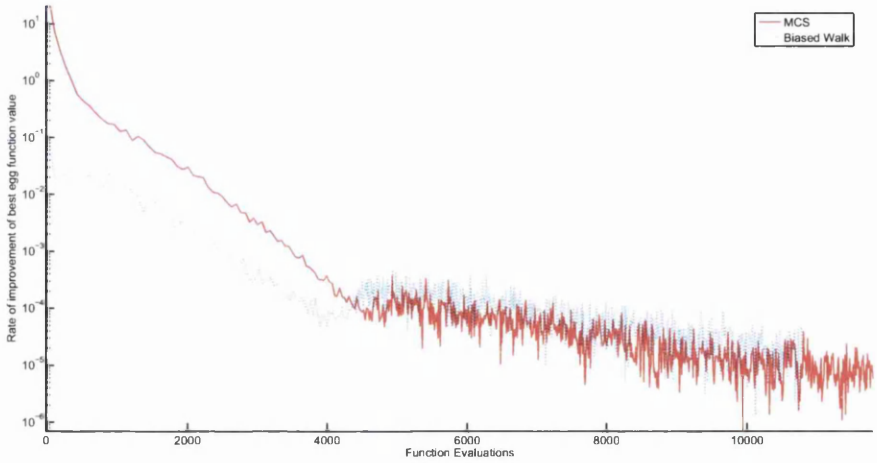


FIGURE 3.59: The relative effect of the MCS and biased walk algorithms: Griewank's function, shifted

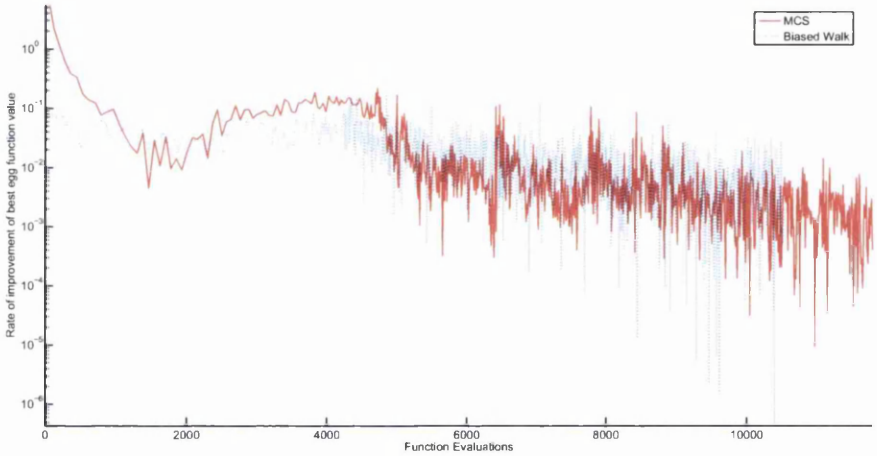


FIGURE 3.60: The relative effect of the MCS and biased walk algorithms: Rastrigin's function, shifted

It was selected as a measure in order to publish an initial MCS validation [1] reflecting the validations presented for CS [36, 47].

A question can however be raised as to how suitable it was to compare algorithms this way. Consider again Schwefel's function in one dimension shown in Figure 3.62. The lowest local–minimum occurs at around $x = -300$, and is in fact the furthest local–minimum away from the global minimum. Using distance from the global minimum as a measure of success, an algorithm which found the $x = -300$ local–minimum would be considered worse than one which found, say, the sub–minimum at $x = 200$. If this function represented a real value, such as mesh quality or drag coefficient, then would it

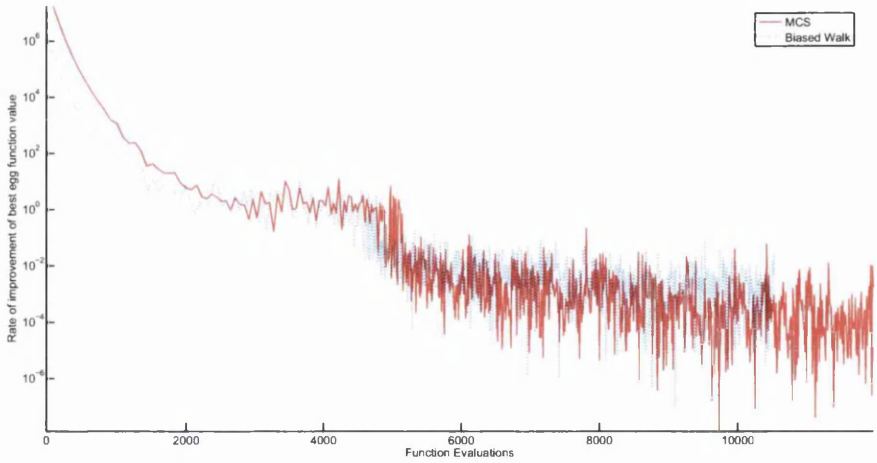


FIGURE 3.61: The relative effect of the MCS and biased walk algorithms: Rosenbrock's function, shifted

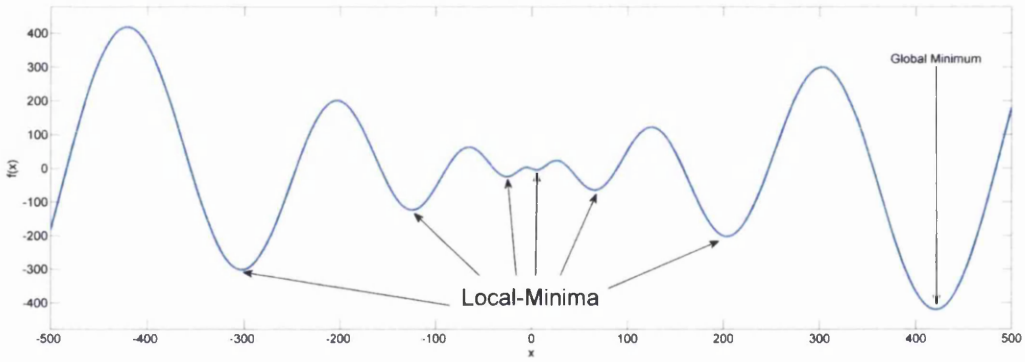


FIGURE 3.62: Schwefel's function in one dimension

be better to have a lower value further away from the global minimum or a higher value closer to the global minimum? In the context of the applications to be presented in this thesis, it is better to have a lower value.

In this section, the latest version of MCS was compared with the latest version of CS, including the biased random walk, DE and PSO. Each of the algorithms had user specified parameters set to the values used previously in other studies. For each objective function used in previous studies, 100 sets of starting populations, each with 100 agents, were generated. To compare performance, the lowest objective function value obtained was plotted against the number of function evaluations.

The study was carried out, initially, on the shifted objective functions used in previous studies. It was then carried out on rotated versions of the objective functions. Most benchmark objective functions are linearly separable. This means that each dimension

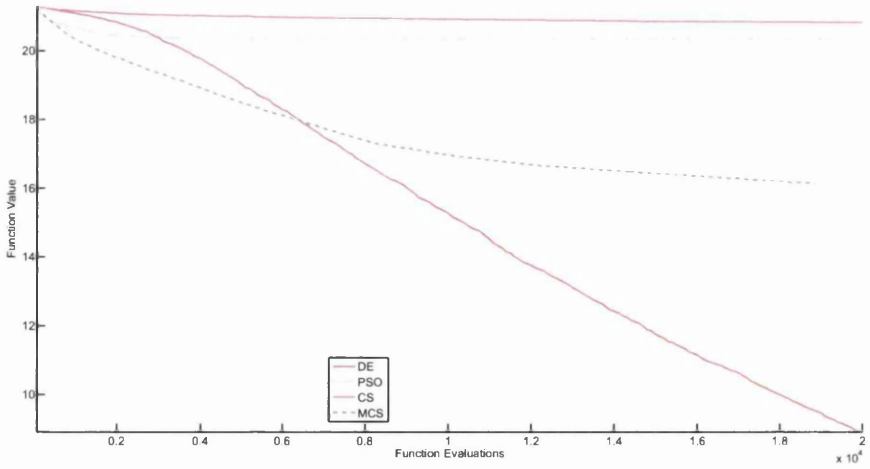


FIGURE 3.63: Comparing DE, PSO, CS and MCS algorithms using objective function value: Ackley's shifted function

can be optimised independently of any other. Many real applications are not linearly separable, since there are interdependencies between the input variables. This kind of interdependency can be introduced into analytical benchmark problems by rotating the coordinate system of the test function [70].

3.6.1 Shifted Objective Functions

The results of the study using shifted objective functions are plotted in Figures 3.63 to 3.68. When applied to Ackley's function, MCS initially reduced the value of the objective function faster than any other method. However, as more function evaluations were performed, DE found a lower value. Similar behavior was found when applying the algorithms to de Jong's, Griewank's and Rosenbrock's functions. For Easom's 2D function, PSO found the minimum in the fewest number of objective function evaluations, with MCS the second fewest followed by DE. MCS was the best performing algorithm at any number of objective function evaluations, when applied to Rastrigin's function in 100 dimensions. These findings further suggested MCS would be a good algorithm to apply to high dimensional problems, where the number of objective function evaluations might be limited.

3.6.2 Rotated Objective Functions

The same study was performed using rotated versions of the benchmarking functions. A random orthogonal rotation matrix was generated for each function and applied to

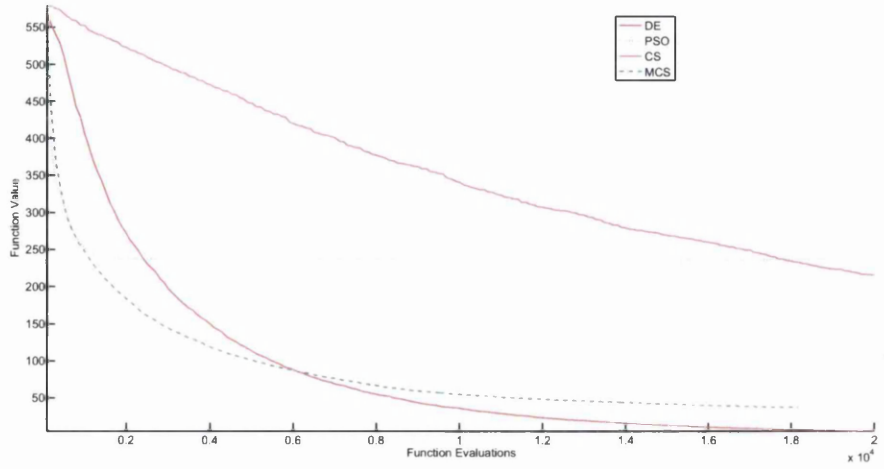


FIGURE 3.64: Comparing DE, PSO, CS and MCS algorithms using objective function value: de Jong's shifted function

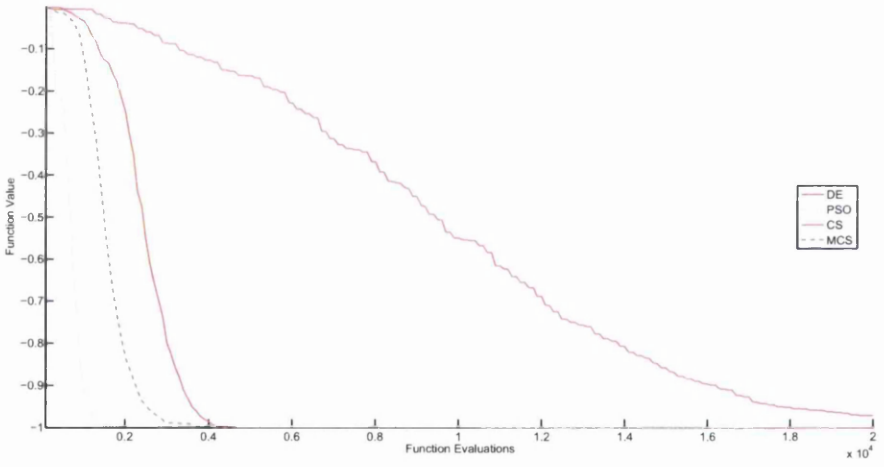


FIGURE 3.65: Comparing DE, PSO, CS and MCS algorithms using objective function value: Easom's 2D shifted function

the inputs before evaluation. This was achieved by generating a $d \times d$ matrix of random numbers, then applying singular valued decomposition to this matrix. The result was an orthogonal rotation matrix that will introduce interdependencies between the different input parameters.

The results of the study were plotted in Figures 3.69 to 3.74. The rotated Easom's 2D function results were similar to the non–rotated case.

For almost all other rotated functions, a similar pattern emerged. Initially PSO appeared to perform best, but in every case PSO was overtaken by MCS after the first 2000 or so function evaluations. PSO seemed to get trapped in sub–minima on a number of

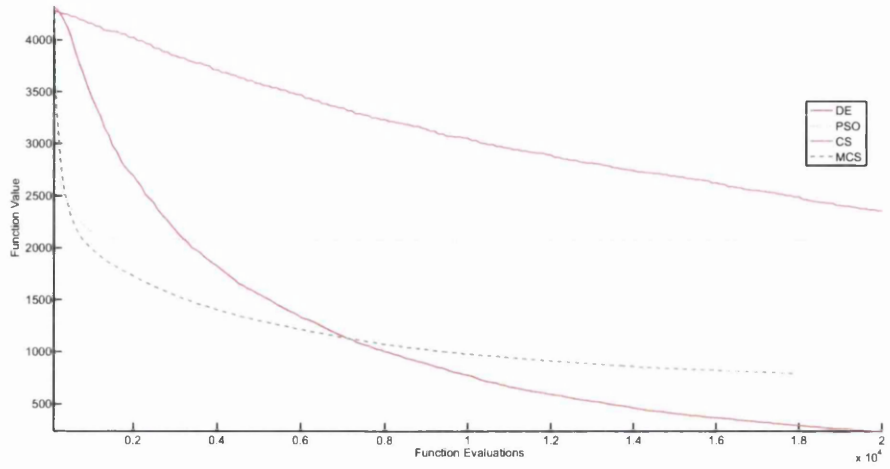


FIGURE 3.66: Comparing DE, PSO, CS and MCS algorithms using objective function value: Griewank's shifted function

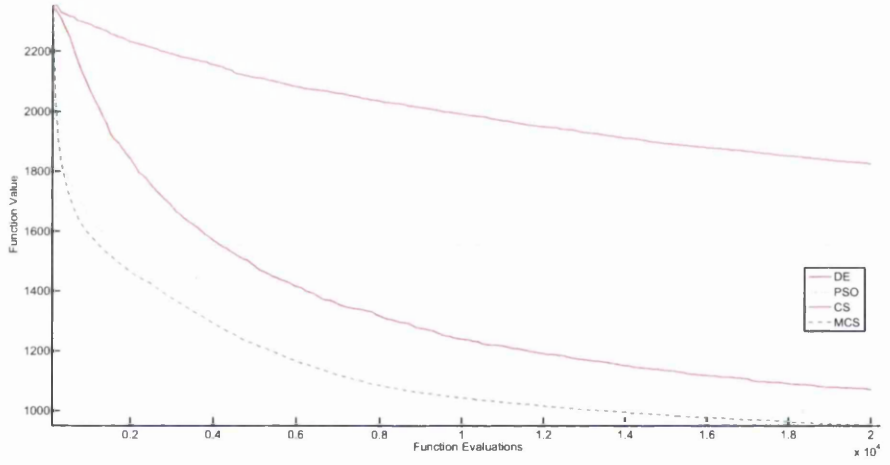


FIGURE 3.67: Comparing DE, PSO, CS and MCS algorithms using objective function value: Rastrigin's shifted function

occasions, whereas MCS did not. DE and CS did not appear to cope well with the rotation, apart from for Rosenbrock's function where DE did eventually outperform MCS.

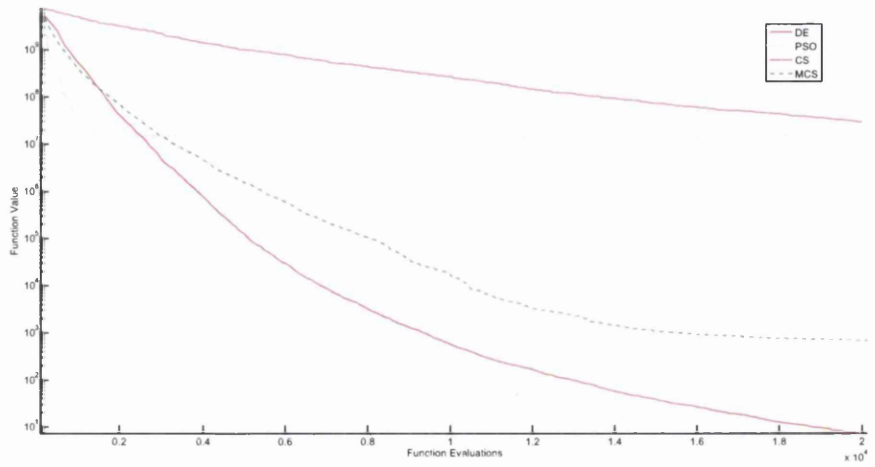


FIGURE 3.68: Comparing DE, PSO, CS and MCS algorithms using objective function value: Rosenbrock's shifted function

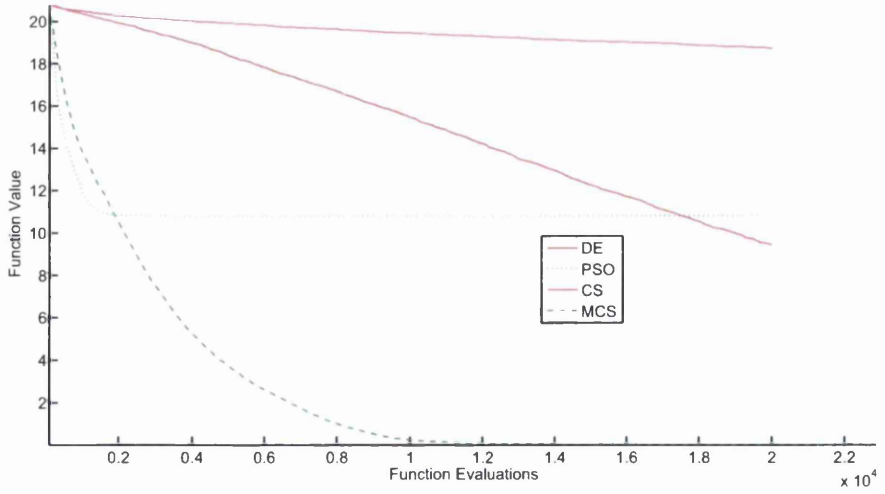


FIGURE 3.69: Comparing DE, PSO, CS and MCS Algorithms using objective function value: rotated Ackley's function

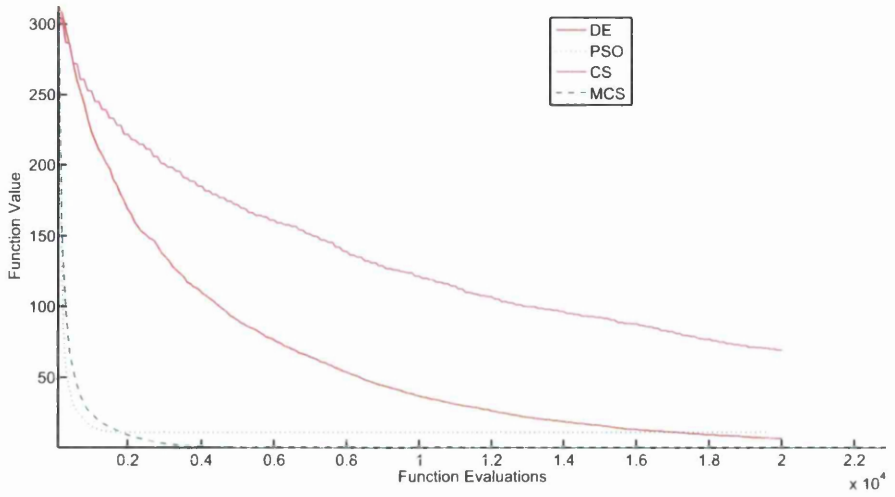


FIGURE 3.70: Comparing DE, PSO, CS and MCS Algorithms using objective function value: rotated de Jong's function

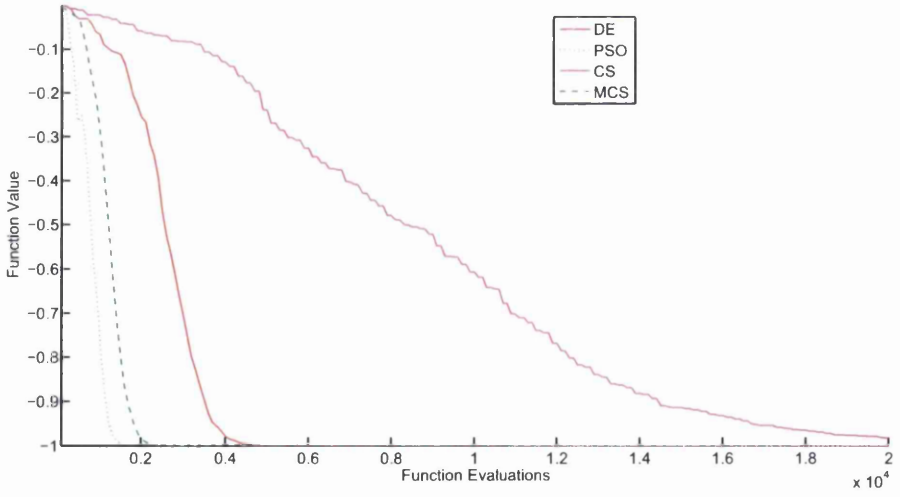


FIGURE 3.71: Comparing DE, PSO, CS and MCS Algorithms using objective function value: rotated Easom's 2D function

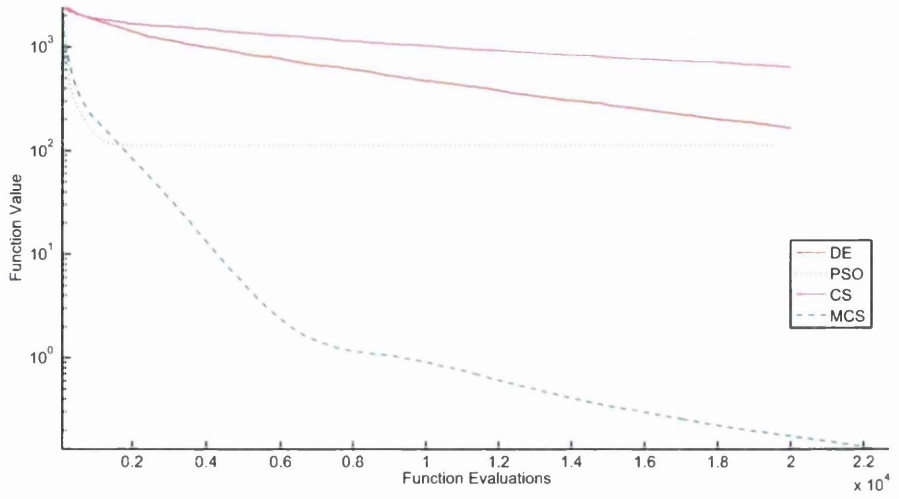


FIGURE 3.72: Comparing DE, PSO, CS and MCS Algorithms using objective function value: rotated Griewank's function

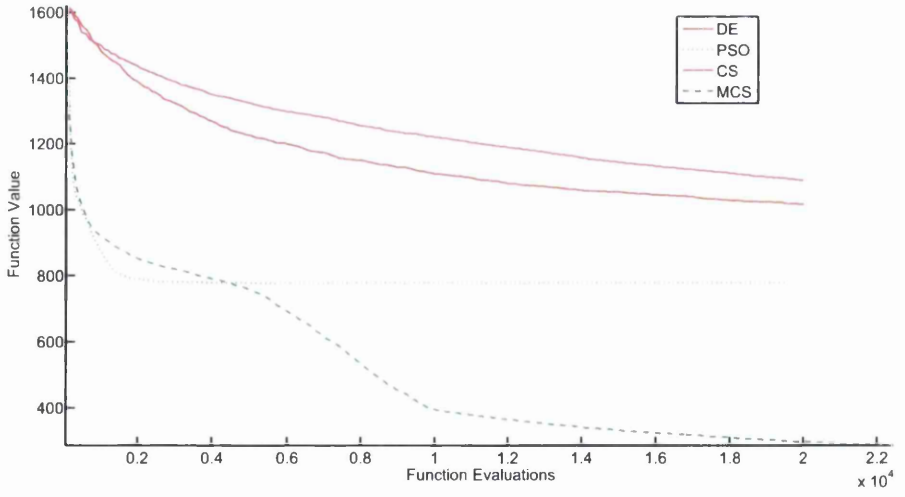


FIGURE 3.73: Comparing DE, PSO, CS and MCS Algorithms using objective function value: rotated Rastrigin's function

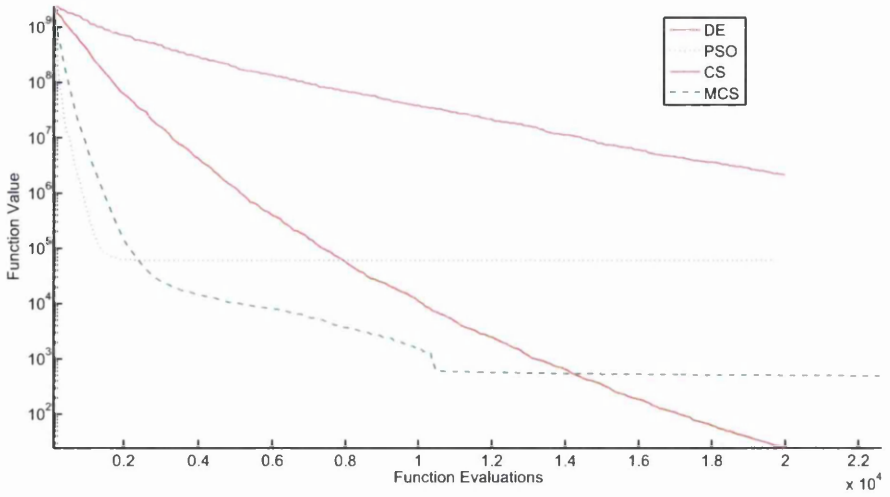


FIGURE 3.74: Comparing DE, PSO, CS and MCS Algorithms using objective function value: rotated Rosenbrock's function

3.7 Conclusion

In this chapter, it was shown that CS has many desirable features for use in optimisation applications. Its performance was increased by the introduction of MCS, which has since been refined further. The final study using rotated functions showed that MCS is largely unaffected by the addition of parameter interdependency when compared to DE, PSO and CS. This suggested that MCS could perform well when applied to real problems. The focus of the remainder of this thesis, was to ascertain how MCS performs when applied to real problems.

Chapter 4

Aerofoil Shape Optimisation

4.1 Introduction

The use of CFD solvers continues to increase in industrial settings. Predominantly, these solvers are used, along with experimental data, during a user led design cycle. As computational power continues to increase, both in terms of clock speed and increased parallelisation, the possibility of using CFD to automatically determine the optimum design for a given application becomes practical [71].

The area of work which is attempting to reach this goal is shape optimisation. Aerodynamic shape optimisation is typically the search for a body with minimum resistance to a surrounding moving fluid [72]. The optimisation process will also be subject to a number of constraints, such as the level of lift that is required, or an allowable body thickness. Once an objective function for the desired application is formulated, two additional ingredients are required for shape optimisation, viz. a shape parameterisation and an optimisation technique.

In the work presented in this chapter, the shape parameterisation technique introduced by Kulfan and Bussoletti [73] was coupled with MCS [1] to perform aerofoil shape optimisation.

There is a long history of applying classical gradient based optimisation methods to shape optimisation [74]. One of the most popular techniques is to express the optimisation problem in terms of control theory and to apply an adjoint formulation to find the optimum design [22]. The primary advantage of these methods is their high computational efficiency, i.e. their ability to converge in a short time [75]. This high rate of convergence has resulted in a capability to make an impact on real design problems [76]. Despite this success, it is well understood that these methods are highly dependent upon

the assumed initial design [75], which motivated the consideration of the application of metaheuristic gradient free techniques. These techniques could, of course, potentially enhance gradient based techniques by providing this initial design.

A large body of the work published on the application of metaheuristic techniques to shape optimisation focuses on attempts to address the issue of computational cost using meta-models [53–56]. The focus of the work, presented in this chapter, was to illustrate the capability of metaheuristic techniques. Meta models were not considered at this stage of assessment, as these may have had unforeseen effects on the performance of the optimisation.

Mäkinen et al [77] applied GA to aerofoil shape optimisation. The flow past the aerofoil was modelled using the full potential equations. They are able to construct a shape which matches a target velocity distribution, to eliminate shocks in the pressure field on the aerofoil surface, using their technique.

GA has also been applied with success to multidisciplinary shape optimisation, both minimising aerodynamic drag and radar cross section. It was found, however, that due to excessive CPU cost (around 85 hours for a single optimisation run) it is difficult to tune the algorithm to the problem [78].

Evolutionary algorithms (EA) are able to decrease the drag on an aerofoil by 10% at a Mach number of 0.8 [79]. EAs were also applied to the robust shape optimisation of aerofoils. A drag reduction of 30 – 40% is achieved in 24 hours [80].

Two types of test problem were considered here. The first was the inverse design of a transonic RAE2282 aerofoil. Inverse design attempts to find a shape whose surface pressure matches one specified by the designer. This leads to a faster convergence rate, when compared to optimising the drag or lift coefficients alone [81]. The principle drawback of inverse design is the requirement of a surface pressure distribution, which, if incorrectly specified, may not represent the optimum distribution. This raised questions about the suitability of inverse design in real applications, but it was useful as a validation exercise.

The second problem considered was the drag minimisation of a shape at a target lift coefficient which is a common application in aerofoil design.

4.2 Methods

Implementing the shape optimisation process required the coupling of shape parameterisation, geometry meshing and a fluid solver with the optimiser. The optimiser itself

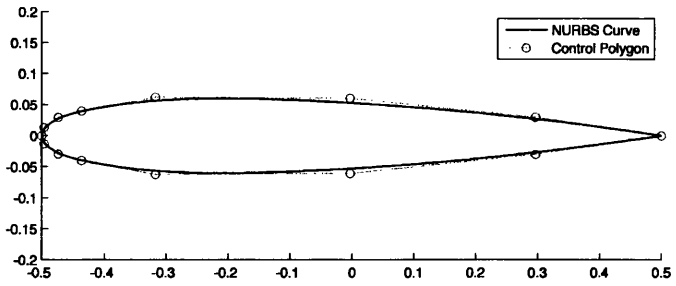


FIGURE 4.1: NURBS definition of NACA0012

should treat the application as a black box objective function. It provides a set of inputs and expects an objective function value in return. In this section, the parameterisation and geometry meshing techniques employed are discussed.

The two dimensional version of the unstructured finite volume compressible inviscid fluid solver developed at Swansea University, FLITE [82], was used to calculate pressure distributions on the aerofoils. An edge–based, node–centered finite volume approach is adopted in the solver, to discretise the Euler equations. This discretisation is performed on an unstructured triangular mesh. All techniques developed in this thesis were designed to be independent of the solver.

MCS was used to perform the optimisation. The specifics on how the various tuning parameters are set is discussed at the end of this section.

4.2.1 Shape Parameterisation

A number of different strategies can be employed for parameterising aerofoil shapes. The most basic parameterisation is to consider the ordinate of each surface node as a parameter. This would result in a prohibitively large number of parameters in two dimensions, and even more in three dimensions. This number of parameters could be reduced by considering spline representations of a curve, such as non-uniform rational B-Splines (NURBS). Figure 4.1 shows a NURBS representation of a NACA0012. The vertices of the control polygon define the surface shape entirely. By moving these nodes the shape can be changed. Although this could be a suitable parameterisation, initial

testing showed that this often resulted in shapes with high frequency wiggles, or bumps, along the surface. With this in mind, a parameterisation that resulted in more realistic shapes was sought.

Initially, it was considered appropriate to use the definitions associated with various aerofoil series as the optimisation parameters. For example, the four digits defining a NACA aerofoil represent three degrees of freedom, which specify the shape. This would almost always result in valid realistic shapes. However, this form of parameterisation suffers from a lack of flexibility, in that only standard shapes will emerge from the optimisation process. Since the key motivation in using metaheuristic gradient free methods was to remove the dependence on initial designs, this form of parameterisation was deemed to be unsuitable.

The parametrisation technique, which balances shape flexibility with number of parameters, introduced by Kulfan and Bussoletti [73] was adopted in this work.

Kulfan and Bussoletti [73] define analytical shape, $S(x/c)$, and class, $C(x/c)$, functions to represent an aerofoil, where c is the chord length. Using these functions, the y coordinate of the aerofoil surface is defined to be

$$y(x/c) = C(x/c) * S(x/c) \tag{4.1}$$

In practice, both C and S are defined in terms of two separate functions, one defining the upper surface and one the lower surface. The class function is given by

$$C(x/c) = (x/c)^{N1}(1 - x/c)^{N2} \tag{4.2}$$

where $N1$ and $N2$ are exponents which define the general shape class. Most aerofoils have values $N1 = 0.5$ and $N2 = 1.0$, which lead to the class function form illustrated in Figure 4.2. There is still ongoing work in determining whether or not these exponents should be considered as free parameters. In the inverse design example presented, they were held constant at $N1 = 0.5$ and $N2 = 1.0$, but, in the RAE2822 optimisation example, they were allowed to change, in order to gain higher shape flexibility.

It follows, from equations (4.1) and (4.2), that the class function acts as a constraint, ensuring that the aerofoil is closed at both the leading and the trailing edges. This enables a large degree of flexibility in the shape function.

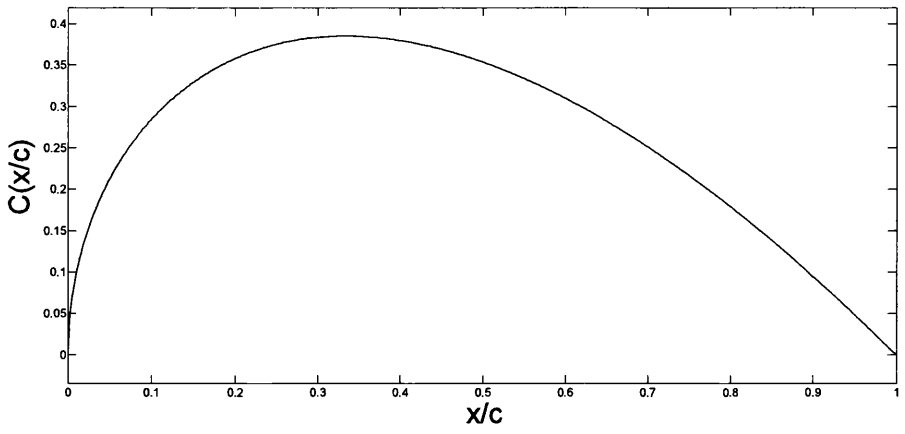


FIGURE 4.2: The class function for $N1 = 0.5$ and $N2 = 1.0$

Kulfan and Bussoletti [73] suggest using Bernstein polynomials as shape functions, leading to the expression

$$S(x/c) = \sum_{r=0}^{n} a_r * K_{r,n} * (x/c)^r (1 - x/c)^{n-r} \tag{4.3}$$

when $K_{r,n}$ is a binominal coefficient, n is the order of the Bernstein polynomial and a_r is a member of the set of n control parameters which define the shape. Here, a Bernstein polynomial order 5 was used to represent each surface, which adds 10 degrees of freedom to the parameterisation. Most aerofoils can be represented using this number of degrees of freedom with this technique [73]. Given more time, it would have been advantageous to investigate the effect increasing and decreasing the order of this polynomial has on the optimisation process.

From now on, $\boldsymbol{a}$ is regarded as a vector containing the degrees of freedom which defines a particular shape and the notation $\mathbf{a}^{RAE2822}$, for example, as representing the particular set of coefficients which produce the RAE2822 aerofoil is employed. There is no easy way to calculate $\boldsymbol{a}$ for a general shape. In order to determine $\mathbf{a}$, a minimisation problem was constructed as the square of the difference between the actual shape and $y(x/c)$. To find values of $\boldsymbol{a}$ MCS was applied to solve the minimisation problem.

Like most gradient free optimisation algorithms, MCS will require an initial population of shapes for the optimisation process. This is almost always done using Latin hypercube sampling, as discussed in Section 3.2.1. To generate a population in this way, limits on the value of each parameter were needed. In the parameterisation adopted here, the

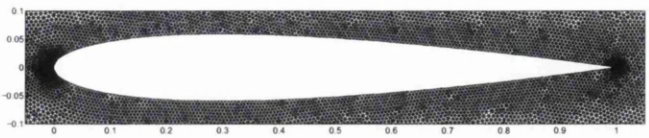


FIGURE 4.3: The initial NACA 0012 mesh

coefficients which make up $\boldsymbol{a}$ have no physical meaning, and thus, appropriate limits on their values were not immediately apparent.

A sample of 25 existing aerofoils were selected to aid determine limits to be applied on the parameters. For each aerofoil considered, values of $\boldsymbol{a}$ were determined using MCS. The minimum and maximum values of each parameter in this group of 25 aerofoils was then determined to give a range for each. For each parameter, the range was also extended by 10% in each direction, in an attempt to avoid the limitations of using a finite sample of aerofoils. These final extended ranges were then used as limits for LHS in the generation of the initial population.

4.2.2 Mesh Movement

Generating a new mesh each time an objective function evaluation is required would be inefficient. A Delaunay mesh generation technique [83], with automatic node creation, was used to create an initial mesh around a NACA0012 geometry. This mesh contained $33,278$ triangular elements, connecting $16,821$ nodes, and an enlargement near the aerofoil is plotted in Figure 4.3.

Each time a new objective function evaluation is required, the NACA0012 geometry was moved to the new aerofoil shape defined by equation (4.1). The mesh was moved using a so called Delaunay graph method [84].

The course Delaunay triangulation, used for mesh movement in the following examples, is shown in Figure 4.4. Once a course Delaunay triangulation is defined the steps in the remeshing method are as follows:

1. locate the unmoved mesh points in the course Delaunay triangulation,

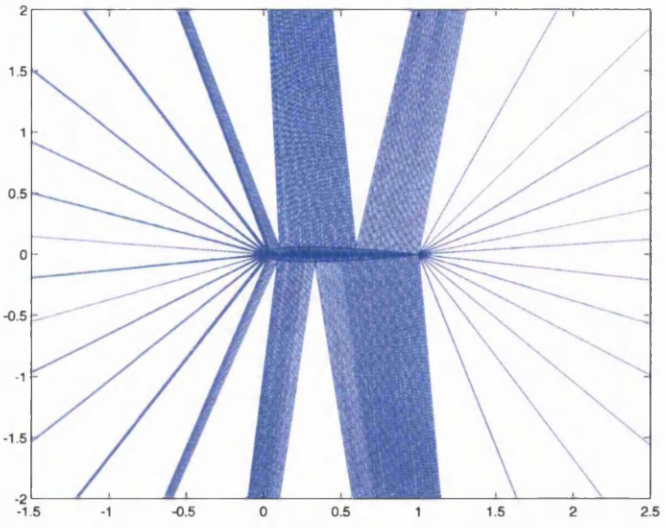


FIGURE 4.4: Course Delaunay triangulation of NACA0012

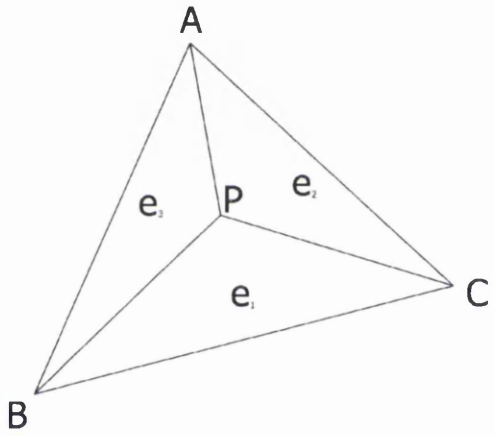


FIGURE 4.5: Barycentric coordinates of P

2. calculate the barycentric coordinates of each mesh point within an element of the course Delaunay triangulation,

3. move the Delaunay triangulation according the boundary change, and

4. transform the mesh from the barycentric coordinates, calculated in the second step, into physical space using the new location of the course Delaunay triangulation.

Consider the course Delaunay element ABC, shown in Figure 4.5, that contains the mesh point P. The physical coordinates of P can be defined in terms of the coordinates of points A, B and C and the three area ratio coefficients e_1, e_2 and e_3. These are calculated using

$$e_i = \frac{S_i}{S} \qquad i = 1, 2, 3 \tag{4.4}$$

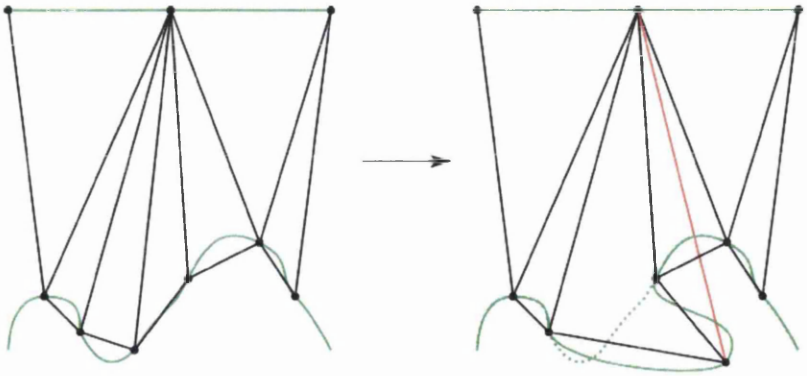


FIGURE 4.6: A mesh movement invalidating the Delaunay mapping due to inverted elements

where S is the area of the Delaunay element ABC and S_1, S_2 and S_3 are the areas of the triangles PBC, PAC and PAB respectively. These are essentially the barycentric coordinates of P in ABC. Given this information, the physical coordinates of P, (x_P, y_P), can be calculated using the equations

$$x_P = \sum_{i=1}^{3} e_i x_i \tag{4.5}$$

$$y_P = \sum_{i=1}^{3} e_i y_i \tag{4.6}$$

where $(x_i, y_i), i = 1, 2, 3$ are the nodal coordinates of the Delaunay element.

An implementation of this technique needs to identify which Delaunay element each mesh point lies in and the area coefficients associated with that point. Calculating this mapping is the largest computational cost of this remeshing technique. However, this calculation only needs to be performed once. The transformation from barycentric to physical coordinates will remain valid, as long as the boundary movement does not cause the inversion, or flipping, of any elements in the Delaunay graph, such as illustrated in Figure 4.6. As long as this condition is not violated the resulting mesh will be valid. If this problem occurs, the procedure will need to be performed at an intermediate mesh movement to ensure a valid mesh.

Once the mapping had been generated for the unmodified NACA0012, for a new objective function evaluation the boundary was moved to the new aerofoil shape defined by equation (4.1). Then, for all the mesh points inside a Delaunay element for which one or more vertex coordinate has changed, the new coordinates were found by applying equations (4.5) and (4.6) using the new coordinates of A, B and C.

This method offers high quality fast mesh reconstruction and also results in a fast check for shape validity. If any of the mesh elements are inverted in the mesh movement this indicates that the new aerofoil surface intersects itself, which is invalid. Further examples are shown in Figures 4.7 to 4.11 to give an indication of the range of achievable shapes using the parametrisation and meshing techniques.

4.2.3 MCS Parameters

The parameters used for MCS in the optimisation were:

- $p_a = 0.7$
- 50 initial eggs were generated using LHS with the limits found for the parameters in the initial study
- $NF = 10^2$
- $Pwr = 0.5$

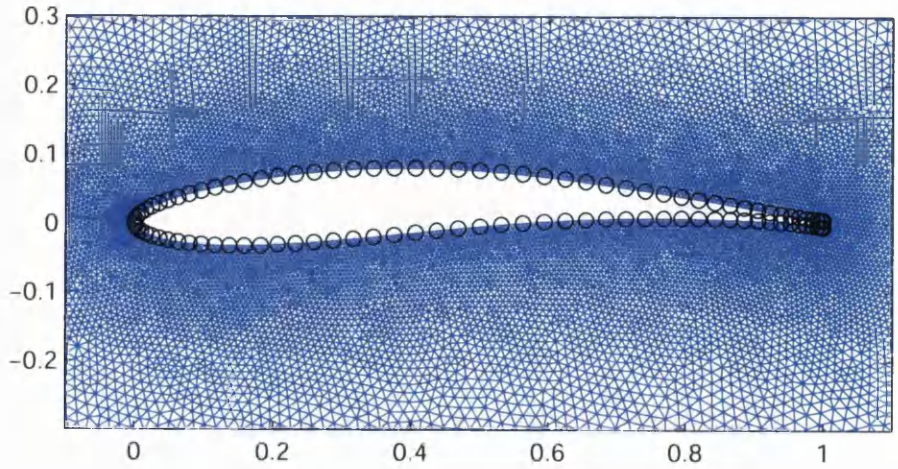
A constraint was placed on the optimisation in regards to the shape of the aerofoil. If a set of parameters resulted in a boundary which is self-intersecting, it was considered invalid and the solution was ignored during the optimisation. A CPU time limit of 100 hours was placed on each optimisation, rather than a generation number limit.

4.3 Examples

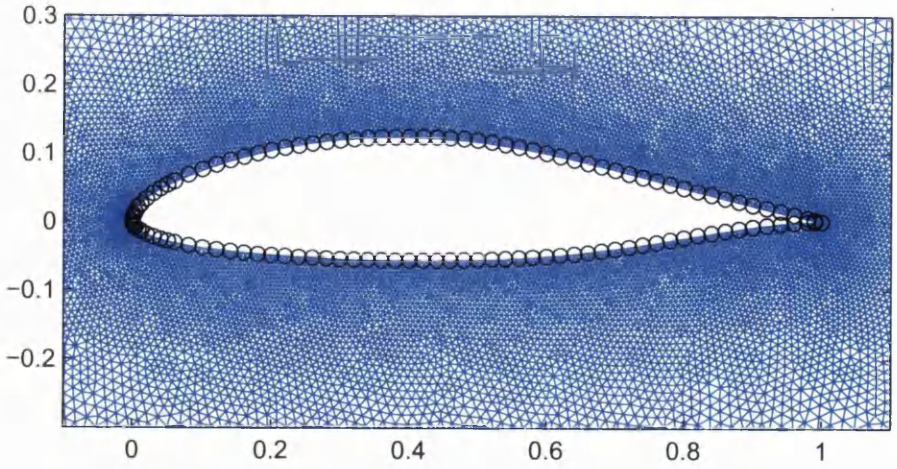
For the examples in this chapter, the calculations were performed on a single 1400 MHz AMD Opteron 240 processor. In the results presented, pressure is given as a dimensionless pressure normalised by ρU_∞^2, where ρ is density and U_∞ is the free-stream velocity norm.

4.3.1 Inverse Design of RAE2822

To define the objective function in this example, the surface pressure on the $\boldsymbol{a}^{RAE2822}$ mesh was first calculated and stored. The objective function was simply the squared difference between the pressure on the surface of current aerofoil and that for the target RAE2822. Since an optimum solution for this problem exists, and is known, it provided a good benchmark for determining the validity of this approach.

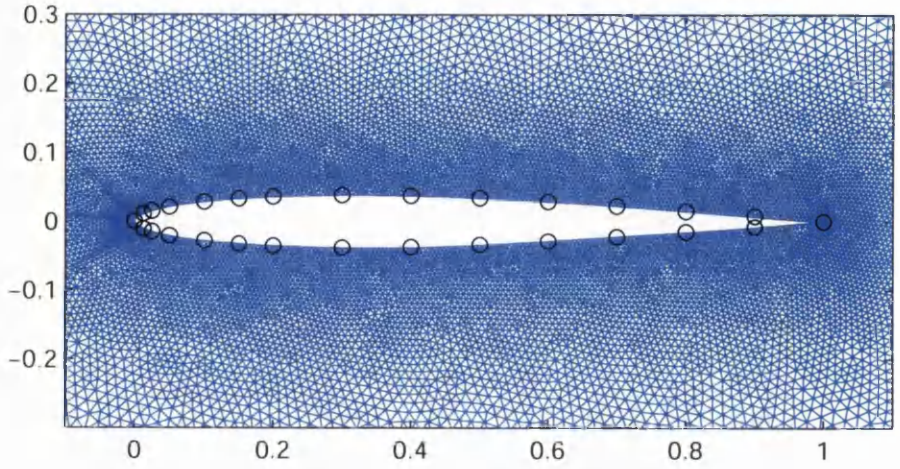


(a) OAF102

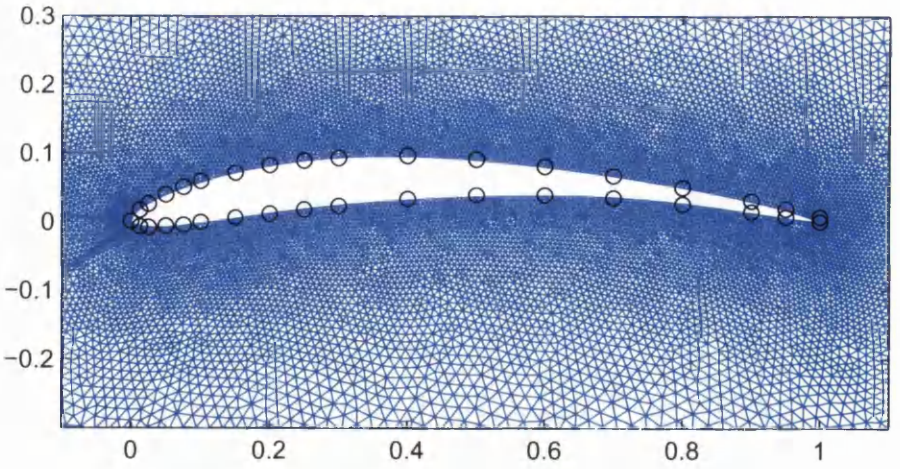


(b) R1080

FIGURE 4.7: Sample aerofoils taken from a study to investigate the values of $\boldsymbol{a}$. The black circles are the original aerofoil geometry, the mesh shows the closest fit found using the parameterisation

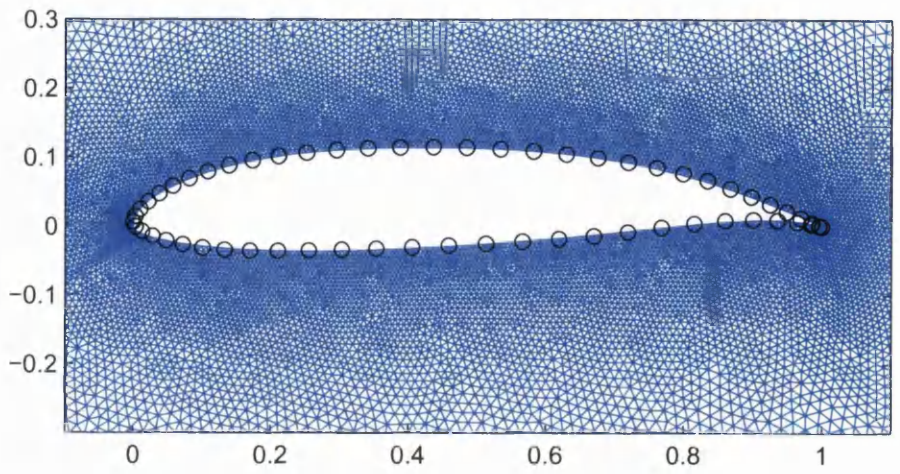


(a) RAF30MD

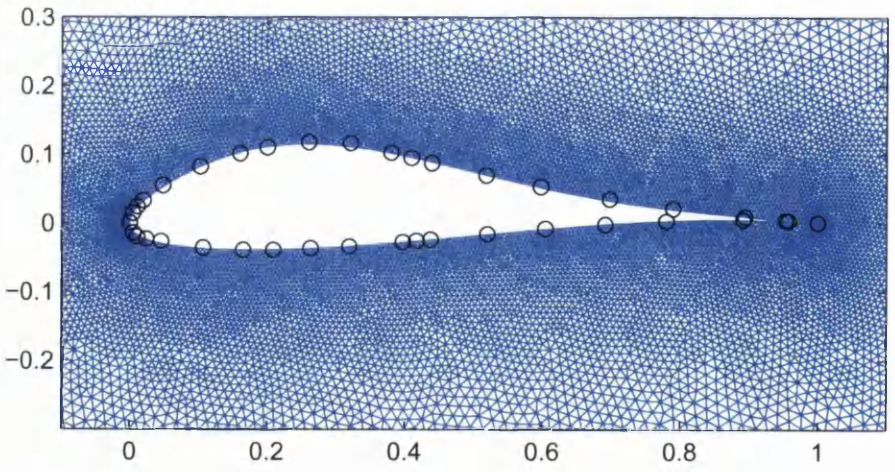


(b) Sokolov

FIGURE 4.8: Sample aerofoils taken from a study to investigate the values of $\boldsymbol{a}$. The black circles are the original aerofoil geometry, the mesh shows the closest fit found using the parameterisation

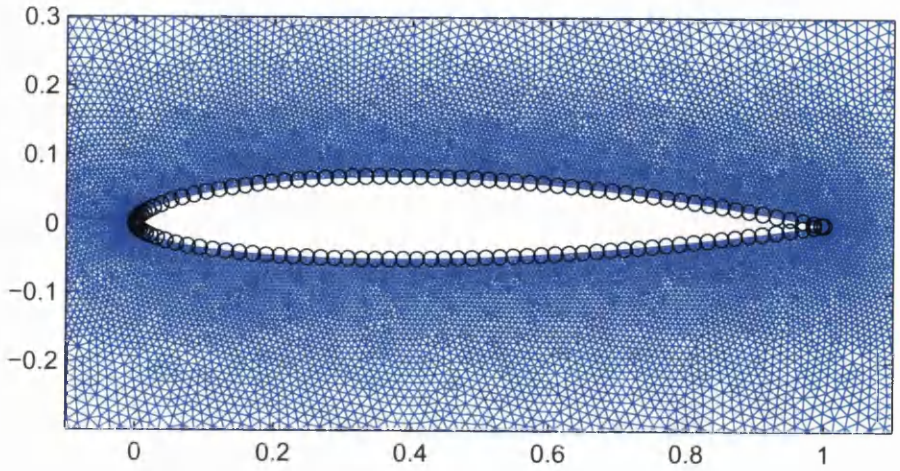


(a) STE87391

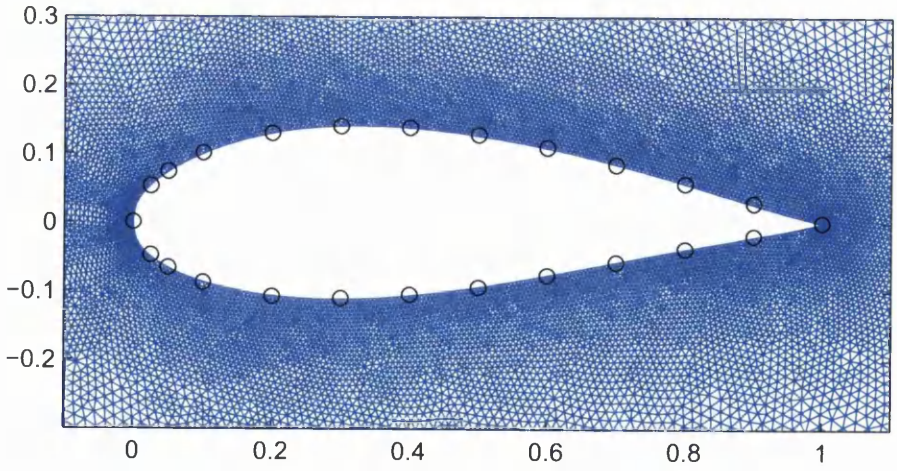


(b) Strand

FIGURE 4.9: Sample aerofoils taken from a study to investigate the values of $\boldsymbol{a}$. The black circles are the original aerofoil geometry, the mesh shows the closest fit found using the parameterisation

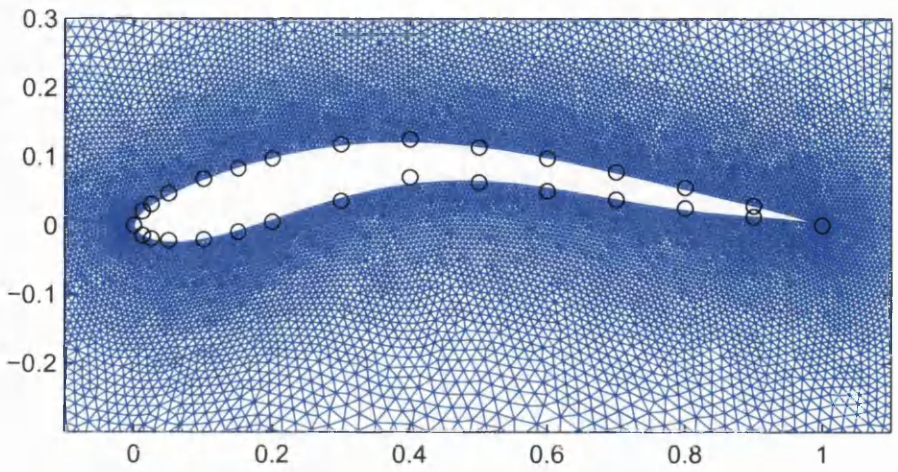


(a) Tempest2

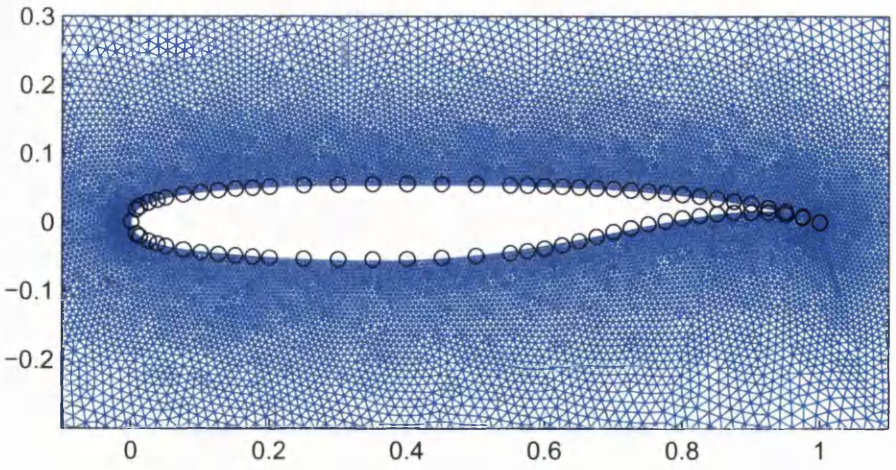


(b) RAF89

FIGURE 4.10: Sample aerofoils taken from a study to investigate the values of $\boldsymbol{a}$. The black circles are the original aerofoil geometry, the mesh shows the closest fit found using the parameterisation



(a) Saratov



(b) Whitcomb

FIGURE 4.11: Sample aerofoils taken from a study to investigate the values of $\boldsymbol{a}$. The black circles are the original aerofoil geometry, the mesh shows the closest fit found using the parameterisation

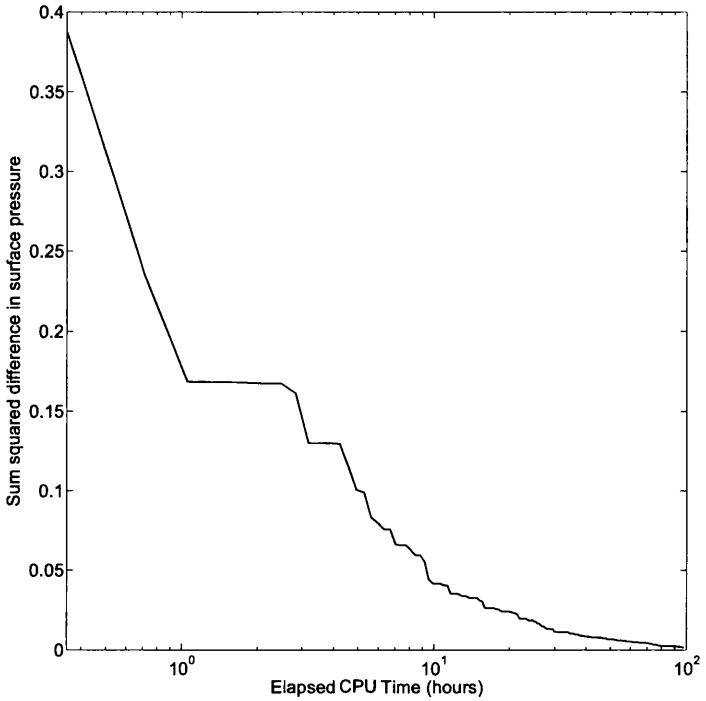


FIGURE 4.12: Convergence history for inverse design of a RAE2822 aerofoil at Mach 0.5

The basic implementation of MCS was used, with tuning parameters suggested in the original validation [1]. An example was performed for two Mach numbers, 0.5 and 0.75, and zero angle of attack. In this example the parameter values were constrained to within the limits found in the previously discussed study.

Mach 0.5 Figure 4.12 shows the convergence history for the Mach 0.5 inverse design example. After 277 generations of MCS, the sum squared difference between the target RAE2822 aerofoil surface pressure and the current best design decreased, by two orders of magnitude, from 0.3870 to 0.0016.

A sum squared difference of 0.0016 represents a small difference in the pressure distribution. This difference is shown in Figure 4.13. The solution obtained after 12 hours is also plotted to show the improvement achieved. This agreement in the surface pressure coincided with a close match to the target geometry, as shown in Figures 4.14 and 4.15. The solution obtained after 12 hours is also plotted to show the improvement achieved.

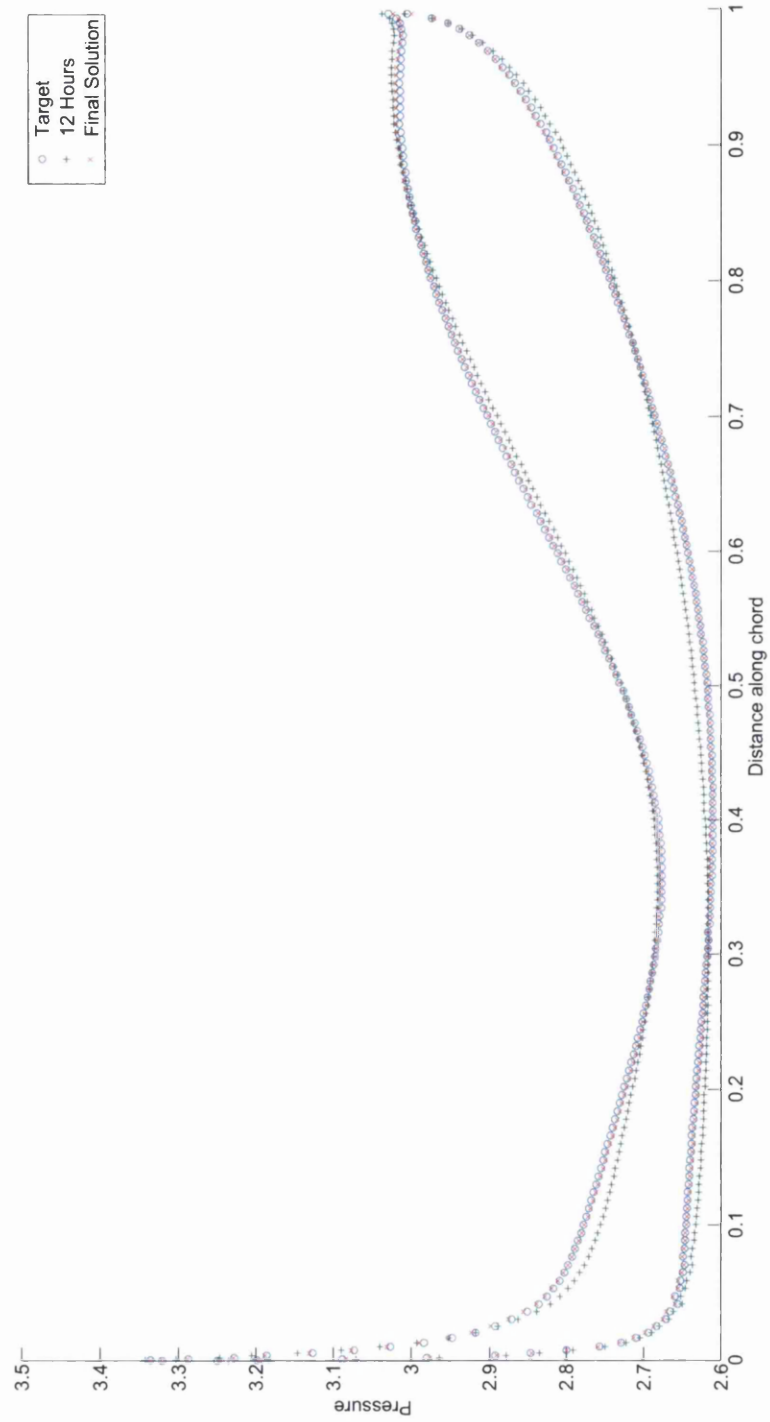


FIGURE 4.13: Comparing the dimensionless surface pressure distribution results for the inverse design example, Mach 0.5.

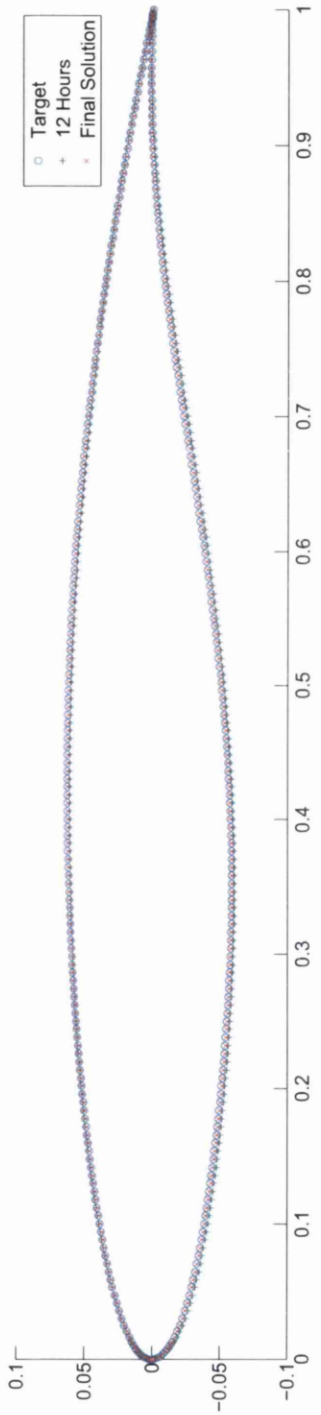


FIGURE 4.14: Comparing the physical surface, during the optimisation process, for the inverse design example, Mach 0.5.

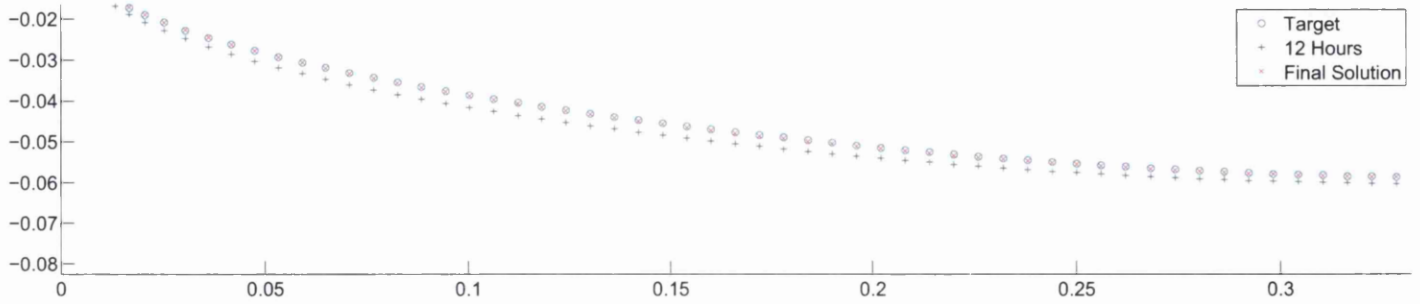


FIGURE 4.15: Comparing the detail of the physical surface, during the optimisation process, for the inverse design example, Mach 0.5.

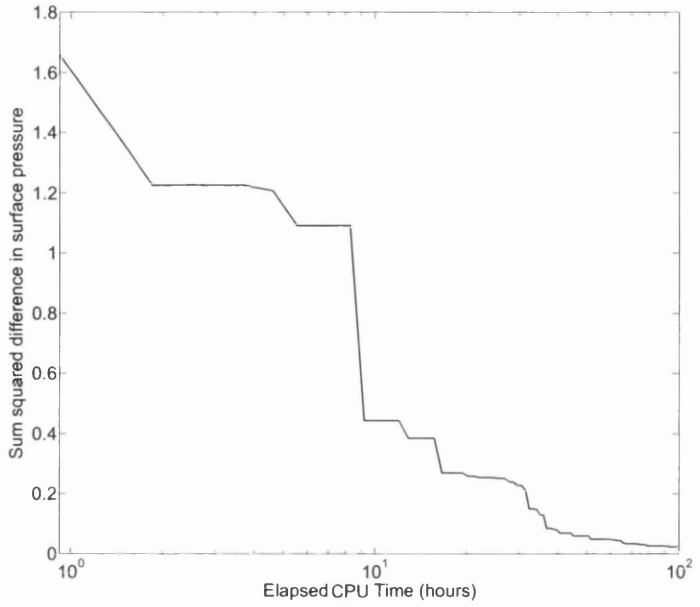


FIGURE 4.16: Convergence history for inverse design example, Mach 0.75.

Mach 0.75 The convergence history for the Mach 0.75 inverse design example is shown in Figure 4.16. The sum squared difference between the target RAE2822 aerofoil surface pressure and the current best design decreased from 1.6580 to 0.0172 in 134 generations. This represented a decrease of two orders of magnitude. Figure 4.17 shows the differences between the surface pressures of the final design, the design after 12 hours and the target. A shock can clearly be seen in this figure. This has had no negative effect on the convergence of MCS. Figures 4.18 and 4.19 show how closely the final design geometry matched the target RAE2822.

4.3.2 Aerofoil Improvement

In this example, the aim was to improve an existing aerofoil by reducing the drag coefficient without reducing the lift. As with the previous example, the RAE2822 was used as a target. The target lift coefficient, C_l^{target}, for this geometry was calculated and stored. The objective function was defined as

$$f(\boldsymbol{a}) = max(0, (C_l^{target} - C_l)) + abs(C_d) \tag{4.7}$$

where C_l and C_d are the lift and drag coefficients of the aerofoil, and $abs(C_d)$ indicates that the magnitude of the drag coefficient was minimised. Here the flow was inviscid so the physical meaning of C_d is not immediately obvious. Truly shock free inviscid flow

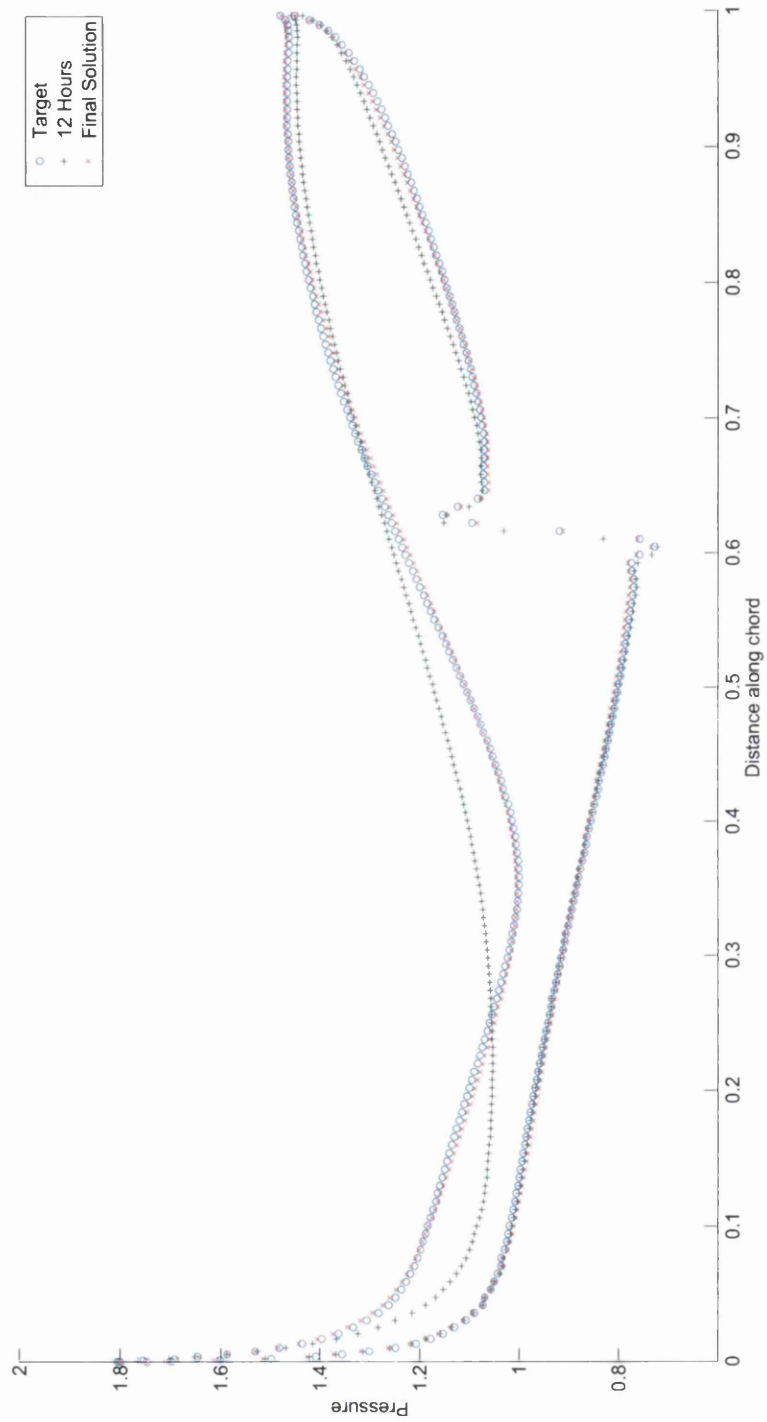


FIGURE 4.17: Comparing the dimensionless surface pressure distribution results for the inverse design example, Mach 0.75.

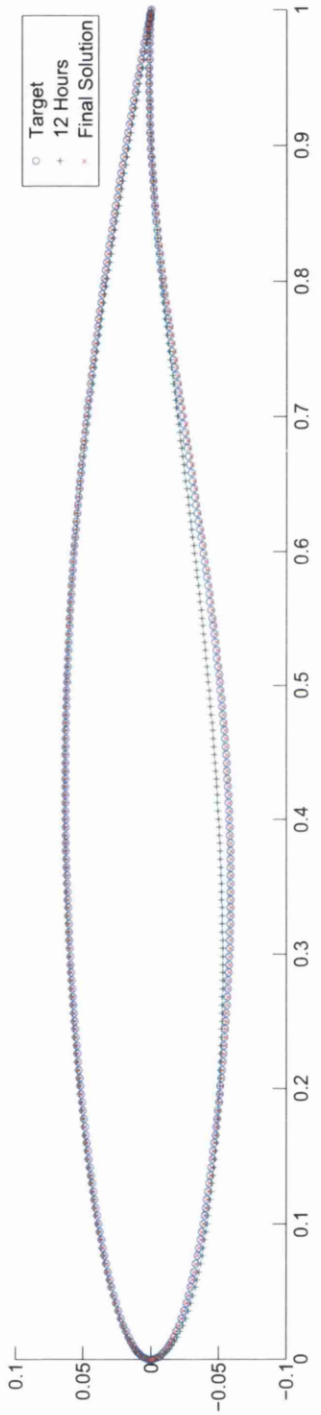


FIGURE 4.18: Comparing the physical surface, during the optimisation process, for the inverse design example, Mach 0.75.

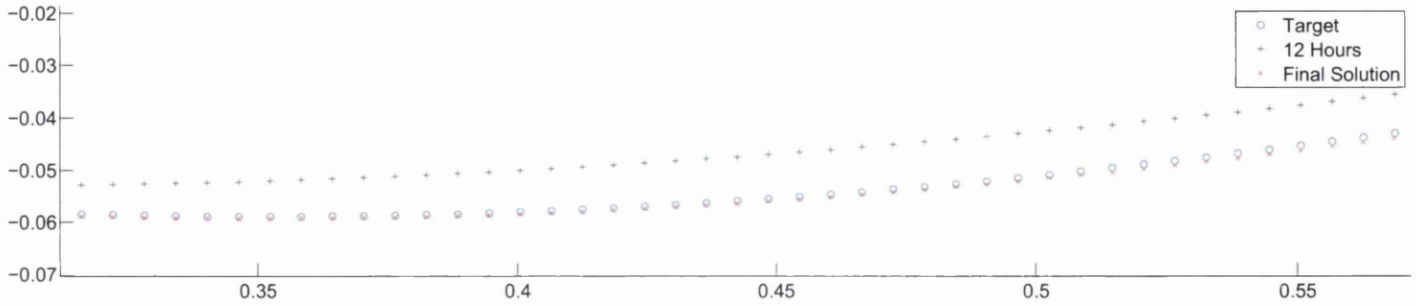


FIGURE 4.19: Comparing the detail of the physical surface, during the optimisation process, for the inverse design example, Mach 0.75

TABLE 4.1: Limits of the parametrisation defined in the vector $\boldsymbol{a}$, used in the constrained examples. Notation from Section 4.2.1 has been employed

Parameter	Minimum	Maximum
$N1$	0.1927	0.6669
$N2$	0.5542	1.4730
Angle of Attack (degrees)	-5	5
Top Surface		
a_1	0.0975	0.3090
a_2	-0.0205	0.5708
a_3	0.0118	0.4853
a_4	-0.0208	0.5074
a_5	0.0572	0.3078
Bottom Surface		
a_1	0.0442	0.2709
a_2	-0.2817	0.3854
a_3	-0.6530	0.4805
a_4	-0.3646	0.3300
a_5	-0.2998	0.3829

has $C_d = 0$. Minimising C_d in inviscid flow is often used to design shock free aerofoils. Studies of this kind have even reported negative values of C_d, which simply means the flow field is not shock free [85]. To reflect the aim to eliminate shocks the free-stream Mach number selected for this example was 0.8.

The parameter vector $\boldsymbol{a}^{RAE2822}$ was included as a starting egg, which means that equation (4.7) ensures that any improvement in drag would only be accepted if the lift coefficient was equal or higher to that of the initial design.

Together with the 10 degrees of freedom defining the shape function, and the 2 exponents defining the shape function, the angle of attack was included as a degree of freedom in these examples. The initial RAE2822 lift and drag coefficients corresponded to an angle of attack of zero. These values are shown in Table 4.2. Two studies were performed, one in which the optimisation was constrained and one without constraints.

In the constrained optimisation the values of each parameter in the vector $\boldsymbol{a}$ was restricted to within the limits determined during the study detailed in Section 4.2.1. The limits are presented in Table 4.1.

TABLE 4.2: Initial Design Lift and Drag Coefficients

RAE 2822		
Mach Number	C_l	C_d
0.8	0.4783	2.309×10^{-2}

TABLE 4.3: Final Design from Unconstrained MCS Design Lift and Drag Coefficients

Unconstrained MCS Design		
Mach Number	C_l	C_d
0.8	0.4783	1.562×10^{-3}

TABLE 4.4: Final Design from Constrained MCS Design Lift and Drag Coefficients

Constrained MCS Design		
Mach Number	C_l	C_d
0.8	0.7056	-1.3646×10^{-8}

To aid readability, all the results are presented in Tables 4.3 and 4.4. These results are then discussed in detail in the remainder of this section. For both examples, sample geometries are plotted, to show the evolution of the shapes. The samples selected represent the large jumps in design quality which seems to be characteristic of MCS optimisation.

4.3.2.1 No Constraints

In this example, the coefficient of lift for the final design, shown in Figure 4.20, did not change compared with the original RAE2822. However, the coefficient of drag was reduced by an order of magnitude to $C_d = 1.562 \times 10^{-3}$ compared with the RAE2822. The surface pressures plotted in Figure 4.21 explain this. The RAE2822 had two shocks in the flow on the surface at Mach 0.8 and the optimisation process managed to remove these shocks. The final angle of attack for this design was -1.1598 degrees.

The convergence history is shown in Figure 4.22. The evolution of the shape of this design is shown in Figure 4.23.

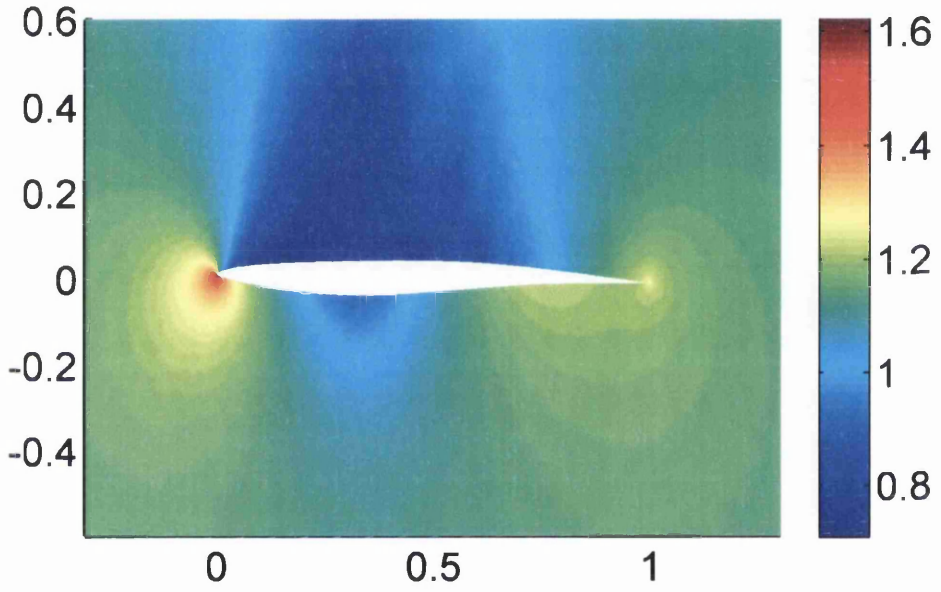


FIGURE 4.20: Dimensionless pressure around the final design for the Mach 0.8 aerofoil unconstrained optimisation

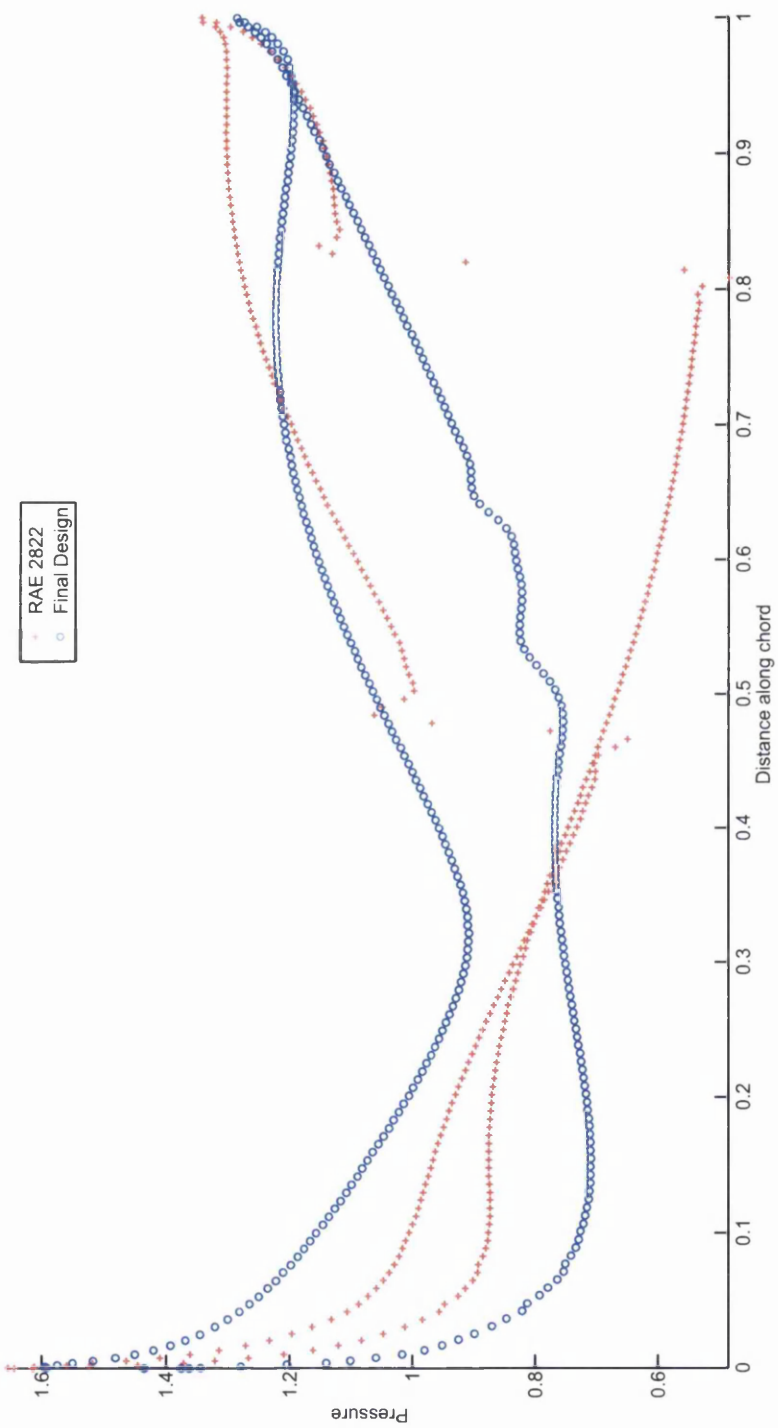


FIGURE 4.21: Comparing the dimensionless surface pressure around the final design to the RAE2822 for the Mach 0.8 aerofoil unconstrained optimisation

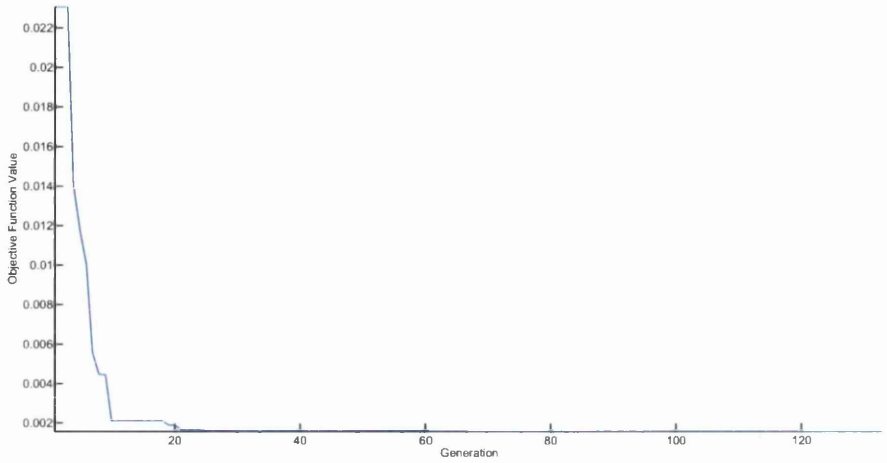
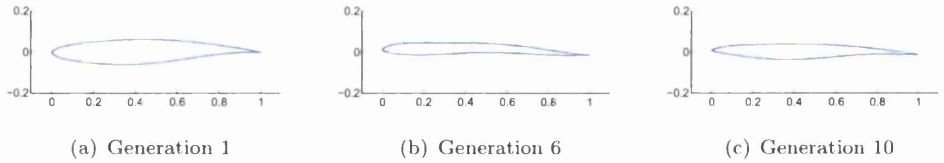
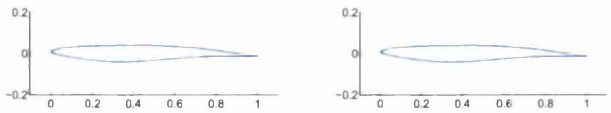


FIGURE 4.22: Convergence history for the Mach 0.8 aerofoil unconstrained optimisation



(a) Generation 1 (b) Generation 6 (c) Generation 10



(d) Generation 21 (e) Generation 26

FIGURE 4.23: Evolution of the best egg: Mach 0.8, Unconstrained MCS

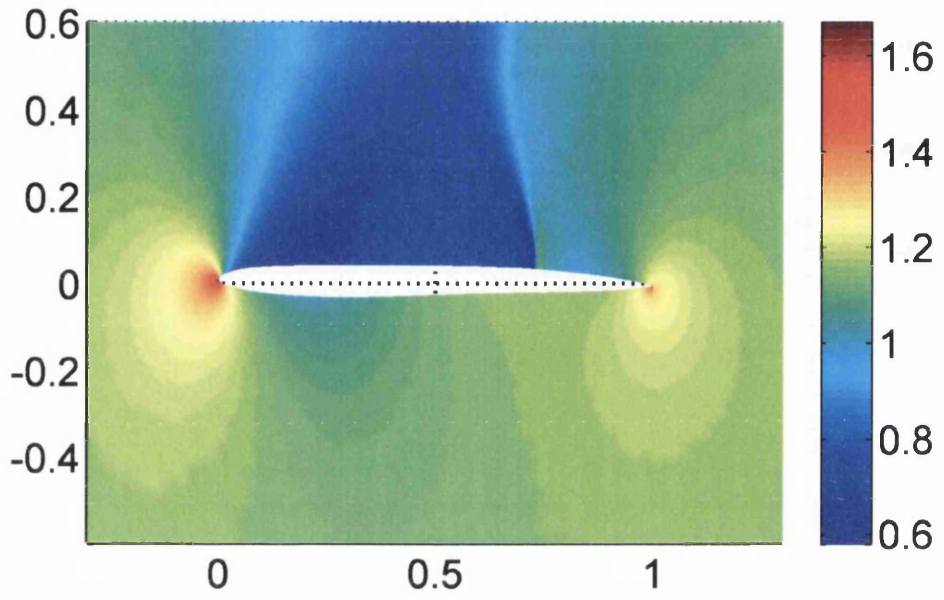


FIGURE 4.24: Dimensionless pressure around the final design for the Mach 0.8 aerofoil constrained optimisation

4.3.2.2 Constrained

Figure 4.24 shows the dimensionless pressure field around the final design in the constrained Mach 0.8 example. Unlike the unconstrained case, the shock was not eliminated, as shown in Figure 4.25. The final angle of attack was -0.8317 degrees. At this Mach number, the drag coefficient achieved by the constrained MCS, -1.3646×10^{-8} was much lower than that found by unconstrained MCS.

The convergence history and shape evolution are shown in Figures 4.26 and 4.27 respectively. The shape was much flatter than the unconstrained case and it was almost approaching a flat plate.

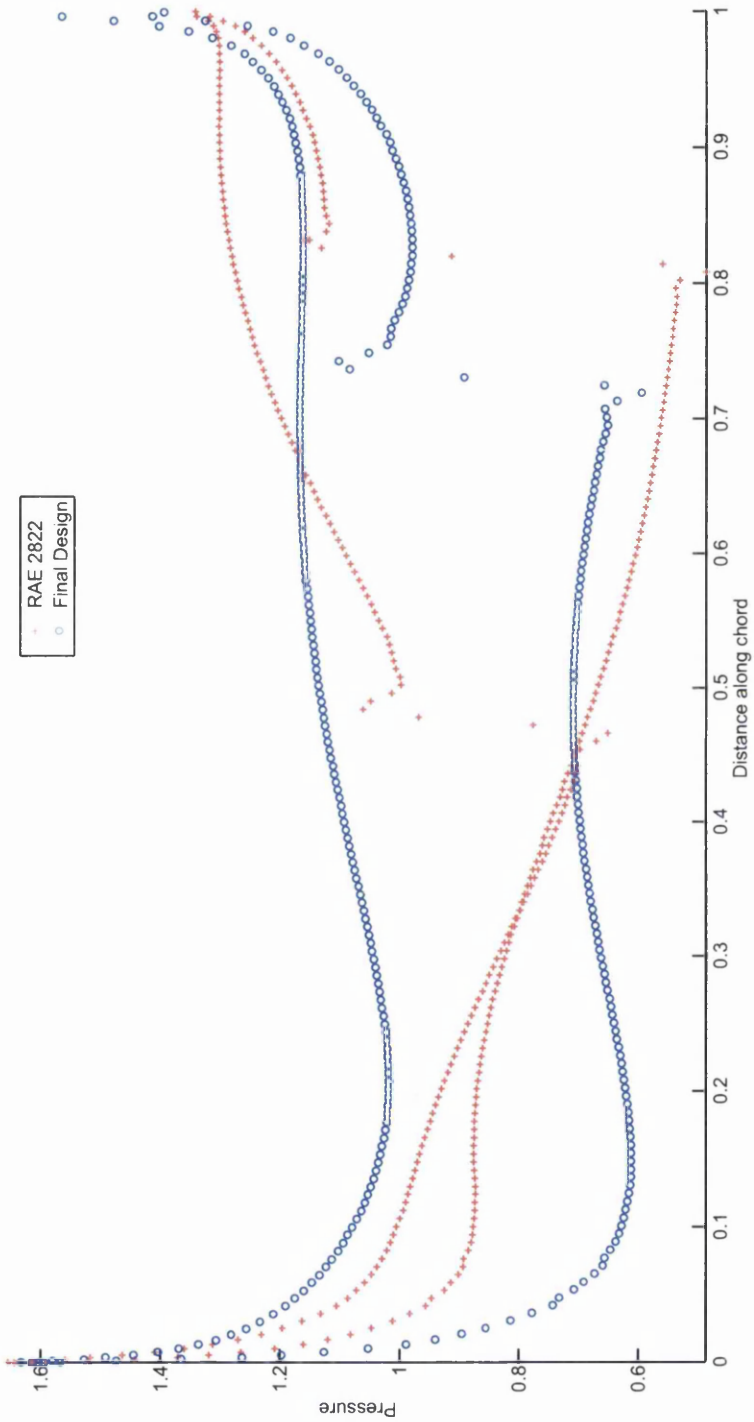


FIGURE 4.25: Comparing the dimensionless surface pressure around the final design to the RAE2822 for the Mach 0.8 aerofoil constrained optimisation

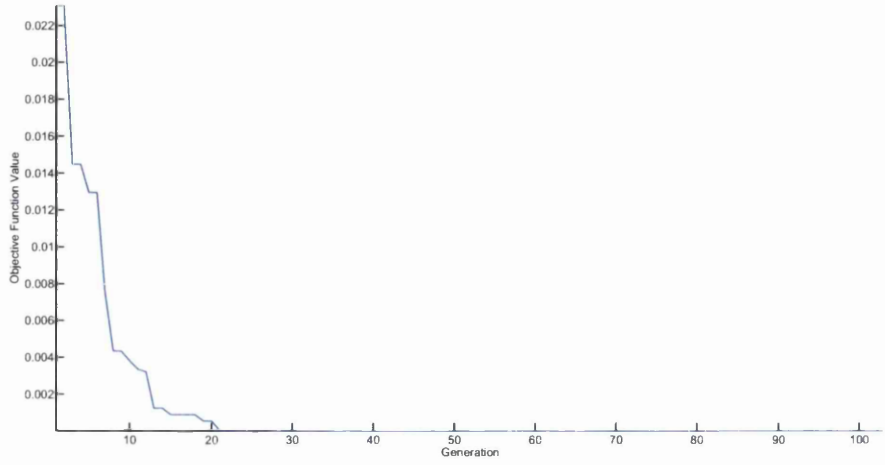
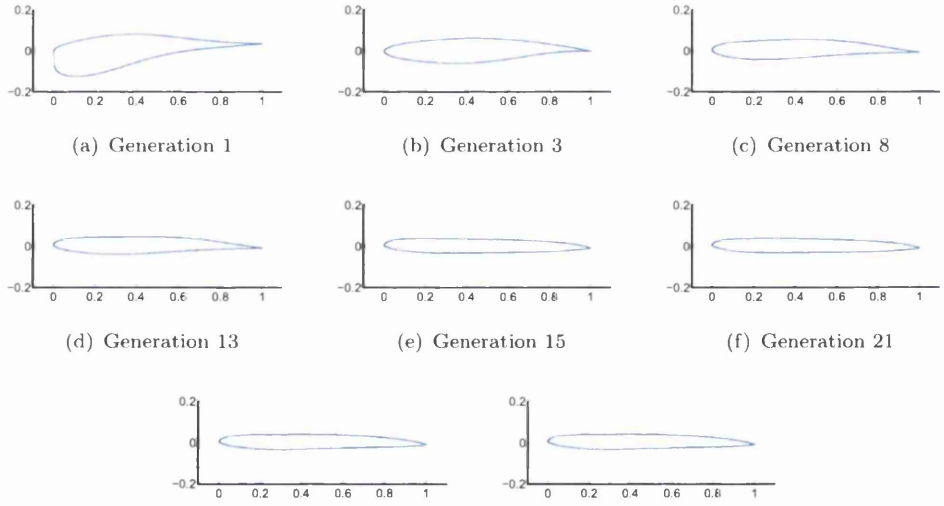


FIGURE 4.26: Convergence history for the Mach 0.8 aerofoil constrained optimisation



(a) Generation 1 (b) Generation 3 (c) Generation 8

(d) Generation 13 (e) Generation 15 (f) Generation 21

(g) Generation 29 (h) Generation 35

FIGURE 4.27: Evolution of the best egg: Mach 0.8, Constrained MCS

4.4 Discussion

The results presented in this chapter show the potential for applying a CS based algorithm to the problem of shape optimisation. Validation of the approach has been achieved here by using an inverse design problem with a known solution. The ability of the method to improve an existing design has also been shown. The examples considered are still very much toy problems and, in the future, it will be necessary to add more complexity by improving the flow physics and moving to three dimensions. In addition structural constraints, such as minimum thickness, will need to be considered. This will inevitably lead to increased CPU costs, which may limit the practicality of using the approach. This difficulty could be alleviated if a suitable method of approximating the objective function could be devised. In the next few chapters, a potential method of achieving this is explored.

Chapter 5

Reduced Order Modelling

5.1 Introduction

The equations describing fluid dynamics have a wide application in engineering design and analysis. The non–linearity of the equations mean analytical solutions only exist once simplifications are made. These simplifications limit their application to a small number of flow situations [86]. The field of CFD has arisen to solve this problem by advocating numerical solutions of the governing equations.

The nature of the equations that govern fluid flow has meant the development of such code is difficult and computationally expensive [87]. Typically, the solution of equations with millions of degrees of freedom is required for accurate Navier–Stokes simulations. This can result in calculations which take in excess of 24 hours to run, even on modern parallelised hardware. These kinds of problems, which have high numbers of degrees of freedom, are examples of high-dimensional problems.

Calculation time needs to be reduced to make CFD an effective practical addition to the design process. As discussed in previous chapters, such a process may require a large number of unsteady solutions for a geometry over a large number of parameters [88]. The excessive CPU cost of running CFD codes is particularly prohibitive when considering applications in metaheuristic gradient free optimisation. ROM originates from this need to solve these kinds of complex nonlinear multiparametric problems in a computationally efficient way [89, 90].

The fundamental aim of ROM is to reduce the number of degrees of freedom representing a system, whilst retaining an acceptable level of accuracy [91]. This is done by developing dimension reduction techniques [92]. ROM can be thought of as an example of data

compression, from high-dimensional data to low-dimensional data, which is why many ROM techniques originate from the field of data compression [88].

The number of degrees of freedom representing a system in a typical CFD solver is large, this is primarily caused by the governing equations being discretised in space. Each degree of freedom in this case is an unknown variable on a mesh node. The key step in ROM is to generate a set of spatial modes, each containing information about every mesh node. These modes can be used to, essentially, remove the spatial dependence of the equations. The modes provide key spatial ingredients for a ROM. By changing the mixture of these modes, the structures of the full order model (FOM) will recreated [93]. A solution can be represented by a sum of these modes and the modal coefficients can be considered the degrees of freedom of the system of equations. These coefficients can be considered a function of either time, control parameters or a combination of both. The vast majority of ROMs are applied to time-dependent problems, where the exact governing, time-dependent, equations of a system are projected onto a series of spatial modes [94]. The time-dependent coefficients of these modes become the unknowns of the system.

Each member of the set of spatial modes, sometimes called basis functions, has a different energy contribution to the system of interest. The term energy is used throughout the field of ROM to describe the importance of each mode. In a number cases this energy may not be defined in terms of the physical meaning of the word, but the total energy in a system is defined as the sum of energy contribution from each mode. The energy of each individual mode is normally expressed as a percentage of the total energy. Different methods will have different techniques of calculating that contribution. The basis may sometimes be truncated, such that the low energy modes are discarded.

There are two contrasting families of techniques to calculate these spatial modes, viz. data driven and equation driven techniques. In equation driven methods, such as non-linear normal modes [95], the modes are constructed by applying mathematical reduction techniques directly to the equations themselves. Data driven methods, such as POD, use sample stored calculated solutions or experimental data to generate the spatial modes [88]. These sample solutions are often referred to as snapshots.

For the optimisation applications considered in this thesis, the important question was how well do these methods deal with parameter changes, such as shape or angle of attack? The work presented in this chapter aimed to answer this question. Initially, a number of different methods for generating the spatial modes are discussed. The selected method, POD, is discussed in more detail. Existing methods using POD applied to parametric steady problems were studied, and a technique was developed to apply POD to unsteady parametric problems.

5.2 Methods for Generating the Spatial Modes

5.2.1 Proper Orthogonal Decomposition

POD (also known as Karhunen-Loève decomposition, and the Hotelling transform [96, 97]) was first proposed in 1901 as a tool to analyse complex systems [97]. The goal of POD is to identify an optimal coordinate system to represent an ensemble of snapshots by removing redundant information [98]. This is equivalent to finding the smallest possible linear set of basis functions which represent the ensemble [90, 97]. When restricted to a finite dimensional case, and truncated after a few terms, POD can be considered equivalent to principle component analysis [96].

Performing POD can be computationally expensive, so the technique was virtually unused until the middle of the 20th century [96]. POD has since been applied to a wide range of problems, including signal analysis, pattern recognition, image reconstruction [91] and weather prediction [93].

The most striking property of POD is its optimality. It is very efficient at capturing dominant components [93]. In fact, when truncated at any number of modes, it captures more relative energy, with those modes, than any other decomposition method truncated in the same way. POD is also an unbiased technique, since it acts on the data itself and requires no prior information about the problem of interest [99].

When applied to flow problems, the linear POD procedure results in a number of orthogonal modes [91, 92]. The POD mode which carries the dominant percentage of energy typically represents the mean flow structure in the example being studied. All other modes represent an optimum basis to decompose the flow [99]. The resulting projection of high–dimensional data into lower dimensional space can reveal important, perhaps even unexpected, structures hidden in the data [96]. Today, one can find many examples of its use in flow analysis [92, 99, 100]. It has been applied with success to laminar flows, both compressible and incompressible, and in particular it has been used to successfully extract large-scale organized motions from turbulent flows [97].

One drawback of constructing a ROM, using a POD basis, is that the basis, and hence the model, only contains information that was present in the set of snapshots used to create it [98]. Intelligent selection of snapshots is, therefore, the key to constructing a successful POD basis.

5.2.2 Nonlinear Normal Modes

NNM constructs and defines the ROM from specific properties of the dynamic system. It does this by adapting the mathematical reduction techniques, of the centre manifold theorem and normal form theory directly into the governing equations. Not requiring any snapshot information NNM results in a reduced system of equations to represent the FOM [95].

Amabili et al [95] provide a comparison between POD and NNM as methods of producing a ROM for non–linear vibrations of fluid filled shells. One advantage NNM has over the, essentially linear, POD method is that it is non–linear. The result is that fewer modes, and therefore less computational cost, is required to capture the dynamics of a system by a NNM based ROM than by a POD based ROM. Amabili et al [95] found that both methods perform equally well compared to a FOM in most cases. This is not the case in situations where the driving vibration amplitude is large, resulting in chaotic behaviour. In this case, the POD method has the advantage, since the snapshot data means that, unlike NNM, it is a global method and can capture complex behaviour, as long as the snapshots are well selected.

5.2.3 CVT Reduced-Order bases

CVT was discussed in Section 3.2.1 as a method of moving an initial set of geometric points to better sample a given space. CVT in the context of ROM is a data compression technique, whose starting point is a set of snapshots as in POD. The Voronoi tessellation takes place in the abstract multi–dimensional space defined by the snapshot vectors. Each snapshot is assigned a weight, which defines a density function throughout this space. A set of points is sought, where the mass, as defined by the density function, centroid of the Voronoi vertices coincides with the point which generated them. These points become the basis functions, which can be used to form ROMs [88]. The weighting allows different significance to be given to different snapshots, which is not possible in POD.

Burkardt et al [88] compared POD to CVT, using a so called T-cell example. This is an incompressible, viscous flow problem in a T-shaped region with a parameterised time-varying inflow boundary. They used 500 snapshots, created at different time steps using a finite element model, to generate the POD and CVT basis functions. These functions were used to find solutions for various inflow parameters and compared to the full order finite element model. They found that both ROMs are very effective in approximating the FOM in low dimensions. They could not find an advantage with either technique,

both the computational cost and the errors are of the same order. It was stated that CVT is less tested in the literature than POD and that differences may be found with further research. Burkardt et al [88] suggested combining the two techniques by using CVT to group the snapshots into clusters and performing POD on each group of snapshots.

5.3 Proper Orthogonal Decomposition

POD was picked as the method of choice for two reasons. Firstly it is a data driven technique which can be treated, in the most part, as a general black box. Equation driven methods, however, may require problem specific formulations. The other reason was that POD appeared to be well tested and popular in the literature.

Consider some spatial and parameter-dependent value $\theta(\mathbf{x}, \boldsymbol{\alpha})$, defined over a domain Ω, where $\boldsymbol{\alpha}$ represents some parameters, which may or may not include time, and $\mathbf{x}$ represents space. A snapshot, $\tilde{U}_k(\mathbf{x})$, of $\theta(\mathbf{x}, \boldsymbol{\alpha})$ at a particular set of parameters $\boldsymbol{\alpha}_k$ is obtained as

$$\tilde{U}_k(\mathbf{x}) = \theta(\mathbf{x}, \boldsymbol{\alpha}_k) \tag{5.1}$$

POD attempts to find a set of spatial modes $\boldsymbol{\Phi}(\mathbf{x})$ which best represent the characteristic structure of an ensemble of these snapshots. The field data is then approximated using these modes as

$$\theta(\mathbf{x}, \boldsymbol{\alpha}) = \sum_{i=1}^{M} \Phi_i(\mathbf{x}) T_i(\boldsymbol{\alpha}) \tag{5.2}$$

where M is the number of POD modes, Φ_i is a typical member of $\boldsymbol{\Phi}$ and $T_i(\boldsymbol{\alpha})$ are coefficients which depend on the parameters. In the above expression, Φ_i is given as a function of $\mathbf{x}$ since it is a spatial mode. The practical implication is that $\Phi_i(\mathbf{x})$ is a vector, containing a list of unknown variables, one for every mesh node in the model of interest. The same is true for $\tilde{U}_k(\mathbf{x})$. To construct a ROM, using these modes, methods for determining $T_i(\boldsymbol{\alpha})$ are required.

5.3.1 Calculating the POD Modes

Specifically, the aim is for the POD modes to describe a typical member of the snapshot set better than any other basis [93]. The terms characteristic structure and typical member point towards the use of an averaging operation. With this in mind, the POD modes are constructed mathematically by maximising the mean of the square of the projection of $\boldsymbol{\Phi}(\mathbf{x})$ onto each snapshot, i.e. the expression $\left\langle \left| \left(\tilde{U}_k, \boldsymbol{\Phi} \right) \right|^2 \right\rangle / ||\boldsymbol{\Phi}||^2$ is maximised. Throughout this thesis the symbols $\langle * \rangle$ refer to the mean, $||*||$ is the norm and

$(*,*)$ is an inner product. The mean is maximised to reflect the goal of representing a typical member of the snapshots and the normalisation factor $||\Phi||^2$ is included to prevent Φ increasing without limit.

The maximisation problem described would result in a single POD mode, but other critical points of the function have physical significance and, together, they correspond to the desired basis [93]. To find these critical points, under the condition $||\Phi||^2 = 1$, the problem is reformulated as seeking the maximum of

$$R = \left\langle \left| \left(\tilde{U}_k, \Phi \right) \right|^2 \right\rangle - \lambda(||\Phi||^2 - 1) \tag{5.3}$$

where λ is a Lagrange multiplier. This can be shown [96] to lead to the integral eigenvalue problem

$$\int_\Omega \left\langle \tilde{U}_k(\mathbf{x}) \tilde{U}_k(\mathbf{x}') \right\rangle \Phi(\mathbf{x}') d\mathbf{x}' = \lambda \Phi(\mathbf{x}) \tag{5.4}$$

where $\left\langle \tilde{U}_k(\mathbf{x}) \tilde{U}_k(\mathbf{x}') \right\rangle$ is the spatial autocorrelation function. This function compares a spatial field to itself and is a mathematical tool for finding repeating patterns.

It is assumed that the modes can be expressed in terms of a linear combination of a set of M snapshots sampled from the FOM, i.e.

$$\Phi = \sum_{i=1}^{M} b_i \tilde{U}_i \tag{5.5}$$

where the values of b_i are to be determined. Substituting equation (5.5) into equation (5.4) yields the eigenvalue problem [91]

$$Lb_i = \lambda_i b_i \qquad i = 1...M \tag{5.6}$$

where L is the $M \times M$ matrix form of the spatial autocorrelation function, with typical entry

$$L_{ij} = \frac{1}{M} (\tilde{U}_i, \tilde{U}_j) \tag{5.7}$$

The eigenvalues λ_i of L are real and positive and represent the relative energy contained in each POD mode. The corresponding eigenvectors b_i are mutually orthogonal and are termed the POD basis vectors or modes Φ_i, $i = 1, \ldots, M$ [91]. Constructing the POD basis using snapshots implies that any ROM based on this basis will intrinsically contain all linearly invariant properties of the snapshot set, e.g. the incompressibility condition, if applicable. This follows from the fact that the basis has been computed by performing only linear operations on the snapshot set [101].

The problem of solving equation (5.6) can be stated, equivalently, as finding the eigenvalues and eigenvectors of the matrix UU^T, where U is the $D \times M$ matrix of snaphots [100], where D is the number of degrees of freedom in the system of interest. Each column of U is a vector length D which contains all the degrees of freedom representing a particular snapshot. A solution to this problem may be achieved by employing singular value decomposition (SVD) [88, 96, 100]. Any real matrix can be decomposed as

$$U = \Phi \Sigma S^T \tag{5.8}$$

where Φ is a $D \times D$ matrix, whose columns contain the left singular vectors of U, S is a $M \times M$ matrix, whose columns contain the right singular vectors of U, and Σ is a pseudo–diagonal $D \times M$ matrix, whose diagonal elements are the singular values s_i of U. It can be shown [100] that the matrix UU^T can be decomposed as

$$UU^T = \Phi \Sigma^2 \Phi^T \tag{5.9}$$

It follows that the POD modes are the left singular vectors of U, given by $\mathbf{\Phi}_i = \Phi_{*,i}$ [88], and that the corresponding eigenvalues are given by $\lambda_i = s_i^2$ [100]. Reliable algorithms have been developed to compute the SVD and can be easily applied to POD applications [96]. In the examples presented in this thesis the algorithm presented by Demmel and Kahan [102] is used to perform SVD.

The ability of a POD basis to approximate the FOM relies entirely on the information contained in the set of snapshots used to generate the basis itself [88]. Ravindran [103] gives three different strategies for generating snapshots:

- a set of steady–state solutions corresponding to several different sets of parameters, e.g. angle of attack, Reynolds number,
- an unsteady solution for a fixed set of parameters, evaluated at different instants in time, and
- a combination of the above, in the form of multiple unsteady solutions of varying sets of parameters, each evaluated at different instants in time.

POD can be thought of as an interpolation or an extrapolation technique, with the set of snapshots supplying information about the system to the resulting ROM. It is difficult to perform a general error analysis of a ROM based on POD, because of the dependency on the quality of the snapshot selection. For this reason, in practical applications, a POD based ROM generally needs to be assessed for each new problem.

To use the POD modes as part of a ROM, methods need to be developed for generating solutions outside of the snapshot set [104]. The choice of method for generating these snapshots and solutions outside the set largely depends on the problem to be tackled, in respect to the number and types of parameters which make up $\boldsymbol{\alpha}$. Two types of problems were considered here, steady problems where, $\boldsymbol{\alpha}$ does not contain time but any number of other parameters, and unsteady problems, where $\boldsymbol{\alpha}$ contains time and any number of additional parameters.

5.4 Reduced Order Modelling of Steady Fluid Problems

Two contrasting techniques were considered for this problem, a residual reduction method and a POD interpolation technique.

5.4.1 Residual Reduction Method for Steady Flows

Alonso et al [89] introduced a residual reduction method for using a POD basis for steady flows. The governing equations for the flow of interest are assumed to consist of m partial differential equations, in a domain Ω, and n boundary conditions, on a boundary Γ. These equations will include flow control parameters, such as, for example, the Reynolds number. A FOM is used to generate snapshots with different parameter values and the snapshots are used to calculate a set of POD modes for each fluid variable in the ways discussed above. A minimisation problem is then formulated with the aim of minimising the residuals associated with the governing equations and the boundary conditions, where the inputs or variables for the problem are the POD mode coefficients. Such a problem could be solved using optimisation techniques.

In their study, Alonso et al [89] use a GA to find the POD coefficients. This step has the potential to be computationally expensive, since using a GA requires the residual to be calculated many times and over all nodes. They overcome this difficulty by limiting the number of nodes, over which the residual is calculated, to a small area in the domain. This area is called the projection window, this modification is justified by noting that, since the calculations involved are not exact, the number of nodes used in the residual calculation needs only to be larger than the total number of unknown amplitudes, but not necessarily equal to the total number of mesh points. The total number of mesh points is, after all, a property of the CFD model and not of the POD approximation.

To test their method, 25 snapshots of a non–isothermal flow past a backward facing step were used, with 6 POD modes retained for each fluid variable when constructing the

ROM. The method is shown to work well, with the location of the projection window having only a small effect on the accuracy of the ROM when compared to the FOM. When the window is placed in a region where the most flow topology existed, they found the highest accuracy. The accuracy of solution is virtually independent of the number of nodes used in the residual calculation, provided the number used is greater than the number of unknown amplitudes. The results show a striking accuracy throughout the domain, despite only performing calculations on a small fraction of it.

This technique was not deemed suitable at this stage. Experience gained through work on this thesis has shown that metaheuristic algorithms do require tuning and the same solution for the same problem is not guaranteed. The consequence is that if a ROM constructed in this way was applied to an optimisation problem, the same point in the parameter space may have slightly different objective function values.

5.4.2 POD Based Interpolation

The interpolation approach is described by My–Ha et al [105], who considered the shape of the water plume on the surface following a set of underwater explosions. The shape depends on parameters defining the explosions, and particular shapes can be used to obstruct aerodynamic objects near the surface. The implementation rapidly predicts the parameters needed to achieve a free surface plume of any shape.

A similar technique was also used by Qamar et al [106], who tested the ability of POD to predict flow fields. The example of steady high-speed flow past an axisymmetric triangular surface of various heights, mounted on a spherical nosed body was considered. They found that, when interpolating within the snapshot parameter space, the root mean square error is less than 1% over the full flow field and even smaller when just considering the variables on the body surface.

Using POD as a basis for interpolation is very simple. First, M snapshots U_k, $k = 1, \ldots, M$ are calculated using a FOM off–line. The term off–line is used to indicate the processes of building the ROM, while the term on–line is used for the process of using the ROM. The snapshots are sampled throughout the parameter space, with corresponding parameter coordinates $\boldsymbol{\alpha}$, using the FOM and are then used to calculate the POD basis. Each snapshot can be reconstructed as

$$U_k = \sum_{l=1}^{M} T_l(\boldsymbol{\alpha}_k)\Phi_k \tag{5.10}$$

For each parameter vector, a set of coefficients, considered as functions of $\boldsymbol{\alpha}$, are calculated, using an inner product, as

$$T_l(\boldsymbol{\alpha}_k) = (\Phi_k, U_k) \tag{5.11}$$

for $l = 1, \ldots, M$. In the on–line process, a set of coefficients, $\mathbf{T}(\boldsymbol{\alpha})$, which best reproduce $U(\boldsymbol{\alpha})$ using the POD basis, is calculated using interpolation techniques. These coefficients are used to reconstruct a solution at the new set of parameters. Depending on the capability of the interpolation technique adopted, the snapshots may be sampled uniformly or non-uniformly throughout the parameter space.

This family of POD coefficient interpolation techniques, which appear to have been introduced originally by Ly and Tran [97], have not been widely adopted. When all POD modes are retained, these techniques are exactly equivalent to nodewise interpolation of the solution field. To illustrate this, consider the simple case of two snapshots, in a one dimensional parameter space, at coordinates α_1 and α_2, where

$$U_1 = \sum_{l=1}^{M} T_l(\alpha_1)\Phi_l \tag{5.12}$$

and

$$U_2 = \sum_{l=1}^{M} T_l(\alpha_2)\Phi_l \tag{5.13}$$

Using linear interpolation, $T_l(\alpha)$ can be obtained, for any α, as [107]

$$T_l(\alpha) = T_l(\alpha_1) + \frac{T_l(\alpha_1) - T_l(\alpha_2)}{\alpha_1 - \alpha_2}(\alpha - \alpha_1) \tag{5.14}$$

for $l = 1, \ldots, M$. Since

$$U(\alpha) = \sum_{l=1}^{M} T_l(\alpha)\Phi_l \tag{5.15}$$

substituting from equations (5.12), (5.13) and (5.14), and applying simple summation rules, equation (5.15) can be expressed as

$$U(\alpha) = U_1 + \frac{\alpha - \alpha_1}{\alpha_1 - \alpha_2}(U_2 - U_1) \tag{5.16}$$

This represents a simple nodewise interpolation. These findings are also discussed by Bouhoubeiny and Druault [108], who applied a similar technique to interpolate experimental data in time. Their proof can be generalised to higher dimensions and other interpolation techniques, as discussed by Zimmermann and Görtz [104].

If this method is simply equivalent to nodewise interpolation, the question needs to be

asked, are there any advantages to POD coefficient interpolation over nodewise interpolation? A number of benefits were identified, which result from the reduced number of interpolation operations. It can be measured that, even when including an SVD of the snapshot matrix, POD coefficient interpolation is orders of magnitudes faster than nodewise interpolation. Fewer values requiring interpolation also means it becomes feasible to inspect the smoothness of these values with respect to parameter variation and to apply adaptive interpolation schemes [109]. These adaptive schemes may make it possible, with future investigation, to identify and capture discontinuities in non smooth parameter spaces.

Zimmermann and Görtz [104] compared interpolation techniques to a residual reduction technique and found that the interpolation method provides a better match to the FOM, when considering aerodynamic coefficients. However, the residual reduction method is better at capturing flow features in the solution field itself. Since the majority of optimisation applications will mainly be considering the aerodynamic coefficients, this method was deemed most suited to the problems considered here.

5.4.3 Examples

The performance of this technique was assessed by applying it to the steady turbulent compressible flow past an RAE2822 aerofoil. The Reynolds number remained constant at 6.5×10^6 and both a one and a two-dimensional parameter space were considered.

5.4.3.1 1D Parameter space: Angle of Attack

In the following examples the angle of attack, α, was considered the parameter of interest. A sample of snapshots were generated, between $\alpha = -5$ and $\alpha = 5$ degrees, using the 2D version of the FLITE solver for viscous flow [82]. In all the meshes considered, layers of structured quadratic elements were generated at the aerofoil boundary to correctly capture the boundary layer.

Each solution had a lift coefficient converged to four orders of magnitude. Five sets of snapshots were considered, each containing snapshots at different sample spacings of α, these are shown in Table 5.1.

Each set of snapshots resulted in a ROM which could predict a flow field for a given value of α. To test the prediction capability, the FOM was used to generate solutions between 5 and -5 degrees sampled every 0.2 degrees. The sum percentage error, compared to the FOM, of the density field at each sample point was then calculated. This represented a CPU cost, on an 1400 MHz AMD Opteron 240 processor, of around 40 hours using the

No. of Snapshots, N	Values of α
3	-5, 0, 5
4	-5, -1.8, 1.4, 5
5	-5, -2.8, -0.4, 2, 5
6	-5, -3, -1, 1, 3, 5
11	-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5

TABLE 5.1: Snapshot sets for angle of attack parameter sweep

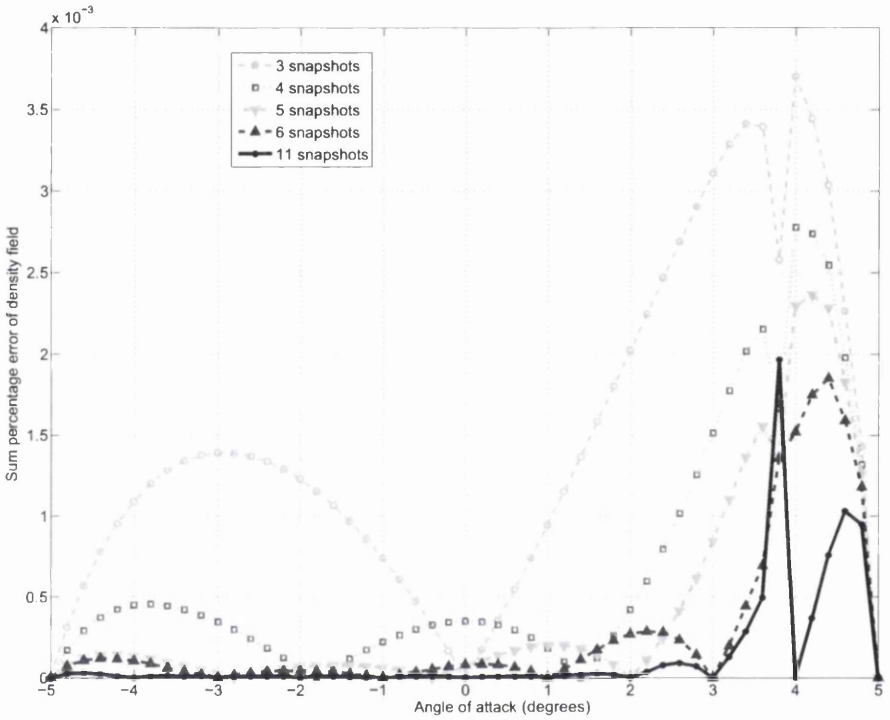


FIGURE 5.1: Percentage error of density field at various angles of attack at Mach 0.5

FOM and 5 seconds using the 11 snapshot ROM. The study was repeated, for different Mach numbers, to gauge the effect of the appearance of shocks in the snapshot sets. In these examples, a cubic spline was used to interpolate the POD coefficients.

Mach 0.5 Figure 5.1 shows how the percentage error varied with the angle of attack for each snapshot set. Even for the 3 snapshot set, all errors were below $4 \times 10^{-3}\%$. As this was a sum of all errors over the entire field, this was a very good result. Looking at the graph in Figure 5.1 in detail, the results between 3 and 5 degrees stand out as being higher than the other errors, although they were still acceptably low.

The result at 3.8 degrees for the 11 snapshots set was very close to a known solution from the snapshot set, yet it represented the maximum error over the sampled angles.

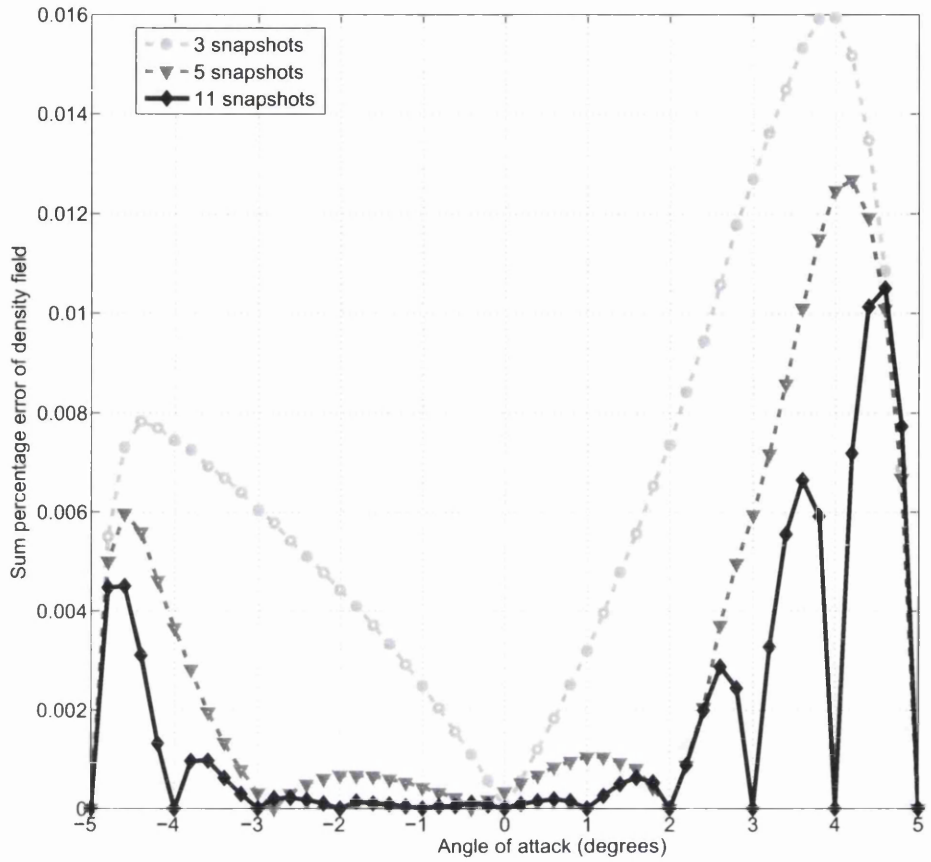
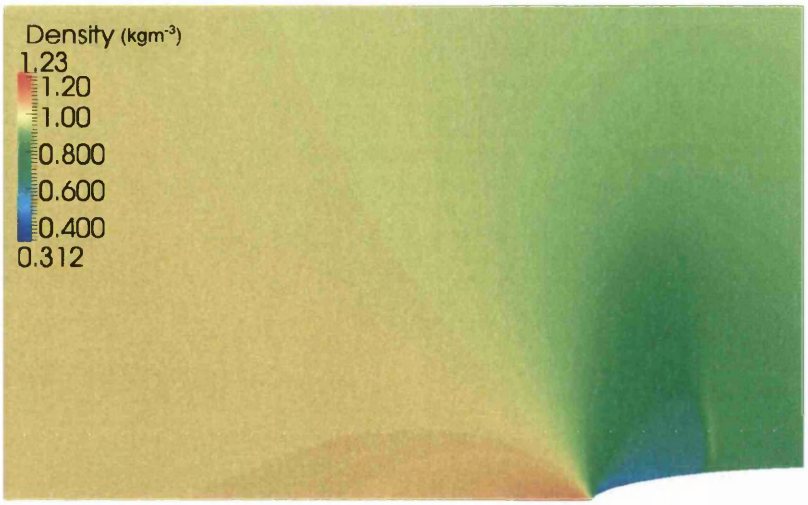


FIGURE 5.2: Percentage error of density field at various angles of attack at Mach 0.65

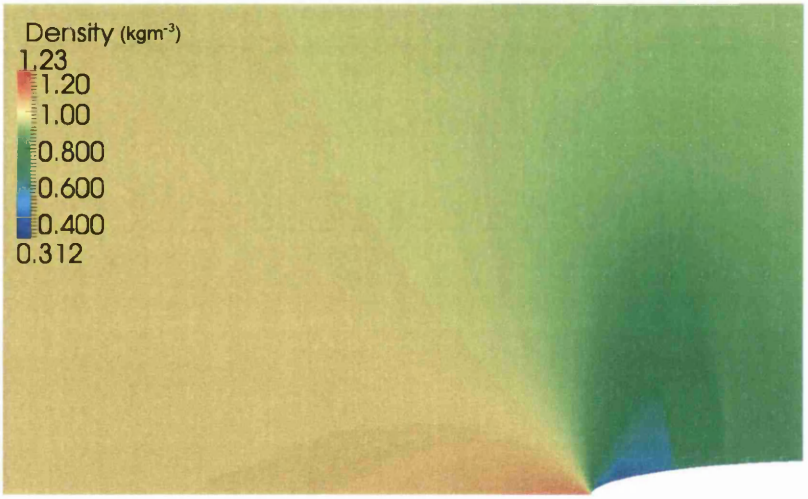
An error in the density may have impacted the accuracy of the lift and drag calculations. At $\alpha = 3.8$ the FOM gave lift and drag coefficients of 0.71570 and 0.010063 respectively. Performing the same calculations using the fields obtained from the 11 snapshot ROM, lift and drag coefficients of 0.71362 and 0.010065 were found. These were still very close to the FOM.

Mach 0.65 The challenge in this example was predicting the onset of weak shocks as the angle of attack increased. No shock appeared between -3 and $+3$ degrees. However, for larger angles, shocks began to appear. Despite this, the percentage error remained below 0.01% as shown in Figure 5.2.

The maximum error in the 11 snapshots ROM occurred at 4.6 degrees, which corresponded to a lift coefficient of 0.94288 and drag coefficient of 0.017332. The solutions are plotted for this case in Figures 5.3(a) and 5.3(b). The ROM failed to fully capture the shock as seen in the FOM. However, when calculating the lift and drag coefficients,



(a) FOM



(b) 11 snapshot ROM

FIGURE 5.3: Density field at $\alpha = 4.6$ degrees, Mach 0.65

using the solution obtained with the ROM, the values were still close to the FOM values of 0.94137 and 0.017296 respectively.

Even though these results still showed close agreement between the ROM predictions and the FOM, it did suggest a possible weakness in POD based interpolation methods to predict changes that result in a significant change in flow regime. For example, it is unlikely that this technique would be able to predict the appearance of shocks in the solution field.

Mach 1.0 Figure 5.4 shows the error plots constructed the same way as in previous examples with free steam Mach number of 1.0. Even when constructing a ROM with

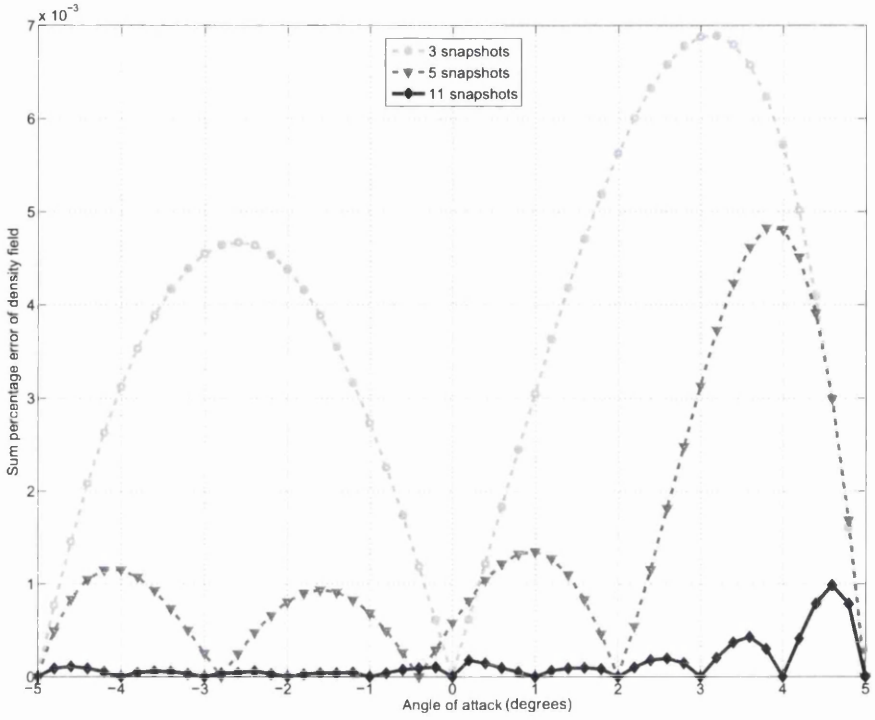


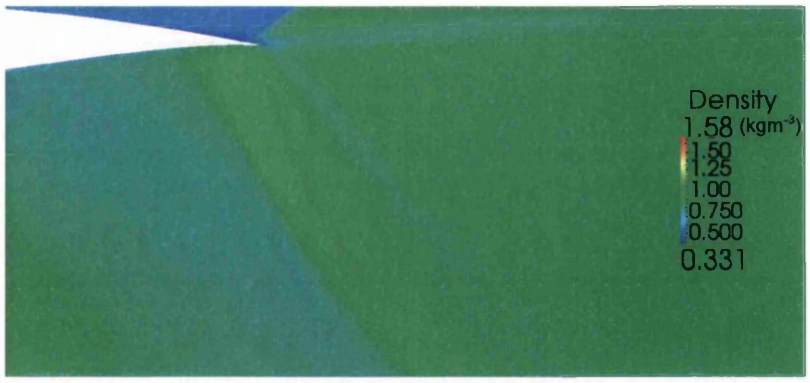
FIGURE 5.4: Percentage error of density field at various angles of attack at Mach 1.0

only 3 snapshots, all errors were less than $7 \times 10^{-3}\%$

The highest error for the 11 snapshot set was at 4.6 degrees. At this angle, the lift and drag coefficients using the FOM were 0.40630 and 0.13953 respectively. Figures 5.5(a) and 5.5(b) show the density fields from the FOM and ROM respectively. The ROM failed to correctly capture the shock, as found in the previous example. Despite this, the lift and drag coefficients of 0.40615 and 0.13958 respectively, calculated using the ROM, were still very close to the FOM values.

Mach 1.25 The solutions in this example were all in the supersonic flow region and contain bow shocks. Figure 5.6 shows that the error in reproducing these solutions using the ROM was less than 0.01% in all cases. This was not as good as the subsonic cases, but was still a good result.

The worst solution in the 11 snapshot set occurred at $\alpha = 4.6$, as shown in Figure 5.7(a) and 5.7(b). It can be seen that the bow shock was not correctly represented in the ROM. This had little effect on the lift and drag calculations. The lift coefficient was 0.37004 using the FOM and 0.36994 using the ROM, while the drag coefficient was 0.12265 and 0.12266 using the FOM and ROM respectively.



(a) FOM

(b) 11 snapshot ROM

FIGURE 5.5: Density field at 4.6 degrees, Mach 1.0

Set	*Range of* M_∞	*Range of* α	*No. of Snapshots*
A	0.1 to 0.3	−1.79 to 1.79	12
B	0.1 to 0.4	−2.79 to 2.79	24
C	0.1 to 0.5	−3.79 to 3.79	40
D	0.1 to 0.6	−4.79 to 4.79	60
E	0.1 to 0.7	−5.79 to 5.79	84

TABLE 5.2: Snapshot sets for angle of attack and Mach number parameter sweep

5.4.3.2 2D Parameter space: Angle of Attack and Mach number

To gauge the ability of POD to predict the flow field when two parameters are changed, both the Mach number and angle of attack were varied. A total of 84 snapshots were calculated for the initial example using the FOM, at angles of attack, α, ranging from −5.79 to +5.79 degrees in 1 degree steps, and at a free-stream Mach numbers, M_∞, ranging from 0.1 to 0.7 in steps of 0.1.

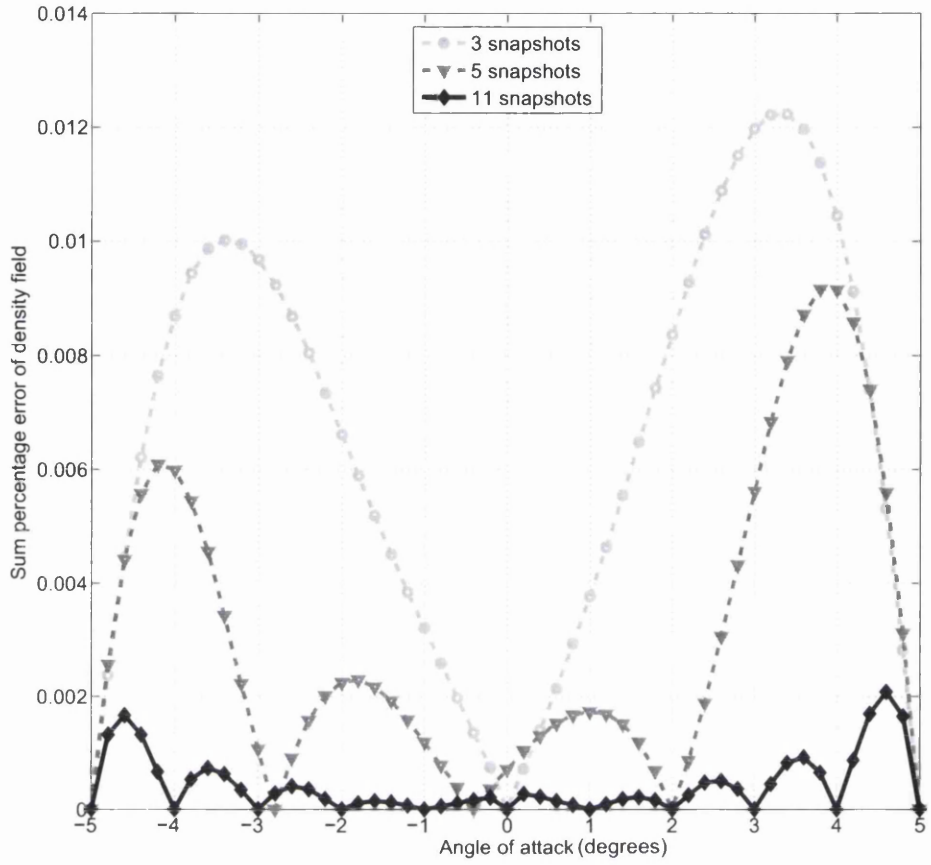
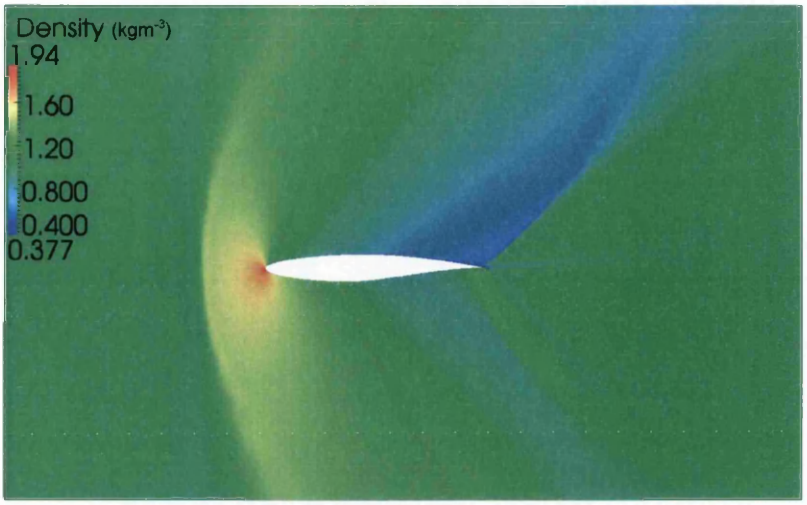


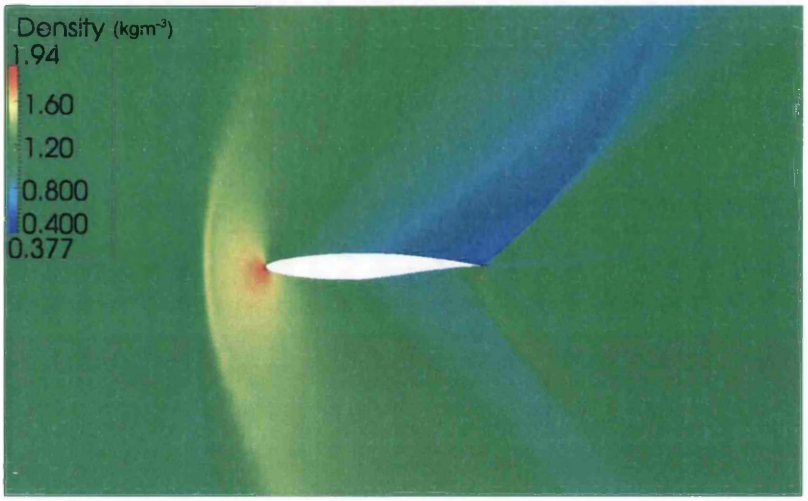
FIGURE 5.6: Percentage error of density field at various angles of attack at Mach 1.25

The snapshots were grouped into 5 sets, as shown in Table 5.2, to determine the effect of shocks appearing in the snapshot set. No shocks were present in sets A, B and C, weak shocks start to appear in set D and strong shocks can be found in set E. Each set was used to calculate a set of POD modes and these were employed to reconstruct density fields at $M_\infty = 0.2$ to $M_\infty = 0.3$ in steps of 0.01, all with $\alpha = 0$. Unlike the 1D parameter space example, the different sets did not refine the parameter space, but they expanded it. Essentially, this was a study to gauge the effect of snapshot set quality on the solution produced. In these examples, bicubic interpolation was used to interpolate the coefficients.

Figure 5.8 compares the FOM to solutions reconstructed using snapshot sets A and E. In this example, the effect on the number of POD modes retained in a solution was also investigated. Figure 5.9 shows how the error decreased, as the number of POD modes used to reconstruct a solution increased for snapshot set C. This trend was the same for



(a) FOM

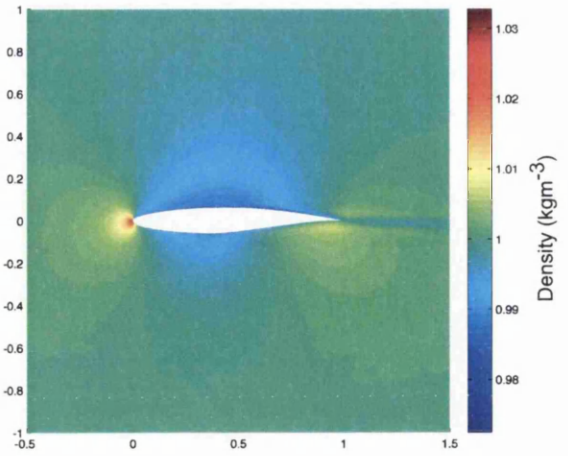


(b) ROM

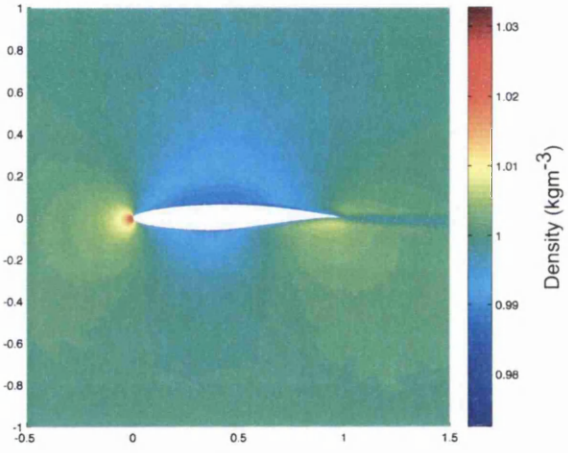
FIGURE 5.7: Density field at 4.6 degrees, Mach 1.25

all snapshot sets. The graph also shows how the error converged. In this study, only around half the total number of the modes were needed to reach the minimum error.

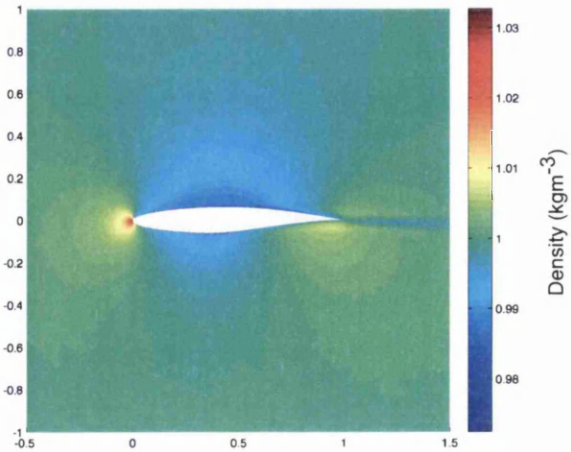
Figure 5.10(a) shows that, as the ROM moves away from solutions in the snapshot set, its error increased. This effect was more significant as the quality of the snapshot set decreases. The relative quality of each snapshot set is shown in Figure 5.10(b), by taking the mean percentage error over all solutions reconstructed by each set. It illustrated two factors that ought to be considered when selecting snapshots: the first, intuitively, was the number of snapshots and the second was the information contained in the snapshots. The ability of the POD reconstruction to approximate the FOM relies entirely on the information contained in the set of snapshots used to generate the POD modes [88]. As



(a) Full order model



(b) Snapshot set A



(c) Snapshot set E

FIGURE 5.8: Comparing full order (a) and reconstructed density fields (b) 12 snapshots and (c) 24 snapshots, for $M_\infty = 0.25$, $\alpha = 0$.

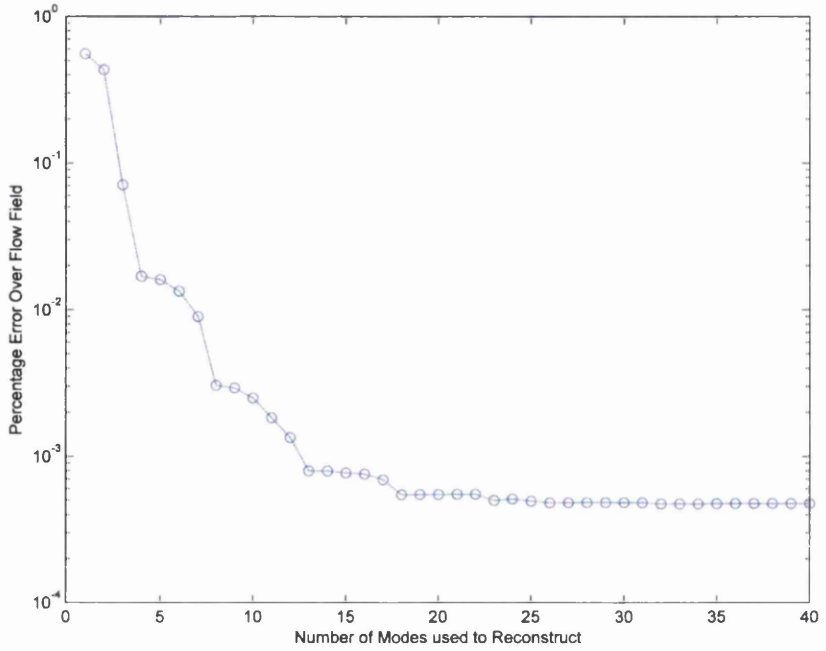


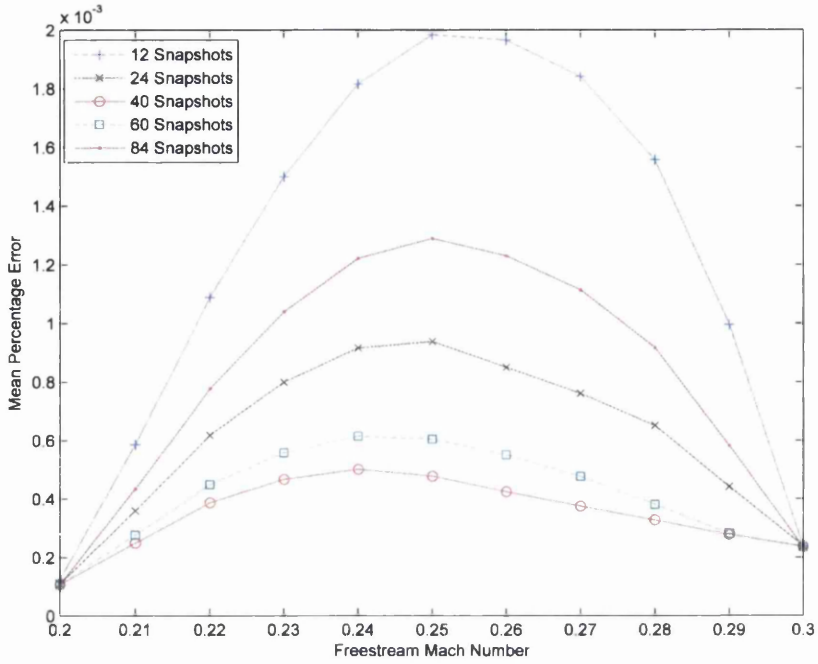
FIGURE 5.9: Percentage error plotted against number of POD Modes used to construct the solution, for $M_\infty = 0.25$, $\alpha = 0$.

the number of snapshots increased the error decreases, up to the point where shocks start appearing in the snapshot set. The more shocks in the snapshot set, the bigger effect they had on the solution quality.

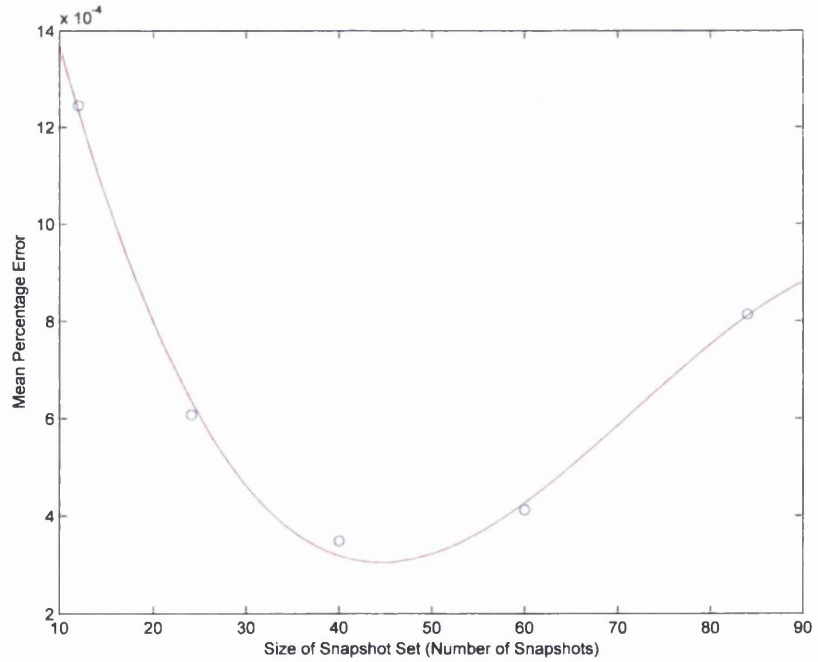
Results presented here showed that this method was able to successfully reproduce solutions with errors of the order of $10^{-4}\%$. This was an excellent result, when considering the corresponding reduction in CPU time that was achieved, from around 30 minutes on an 1400Mhz AMD Opteron 240 to less than a few seconds on a 2.4GHz Intel Core 2 Duo.

5.4.4 Discussion

The ability of the POD coefficient interpolation methodology to deal with parametric variation in steady flow was investigated. It was found that this method, in some cases, could reduce the computational cost of parametric studies significantly, with losses of accuracy much less than 1%. It was not able to when the parametric variation caused significant change in the flow characteristics, for example the onset of shocks in transonic flow.

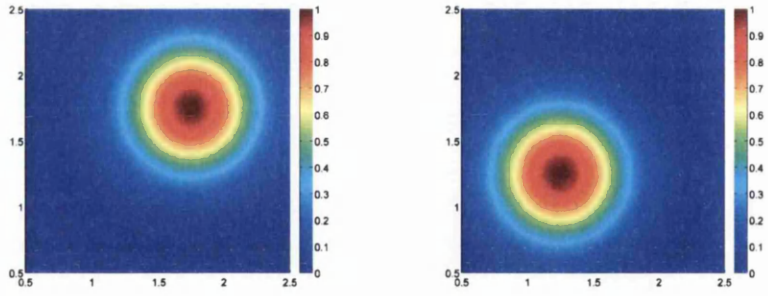


(a) Change of error with Mach number



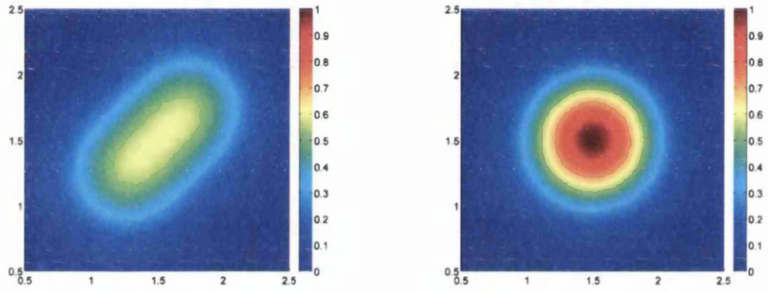
(b) Change of error with number of snapshots

FIGURE 5.10: Investigating the changing error in the 2D parameter space example



(a) Snapshot 1 (b) Snapshot 2

FIGURE 5.11: Two-dimensional Gaussian peaks at two different centres



(a) ROM (b) FOM

FIGURE 5.12: Comparing the FOM and ROM in the Gaussian peak example

The root of this problem is that the method is equivalent to independent interpolation of the nodal values. Interpolation in this way will not move features, like shocks, but simply diffuse them. This concept can be easily shown through a simple two-dimensional Gaussian function. Consider the two Gaussian peaks plotted in Figures 5.11(a) and 5.11(b). Both features have the same radius and magnitude but have different locations. Using the two peaks as snapshots to generate POD modes and interpolating the resulting POD coefficients results the field plotted in Figure 5.12(a). If this was a moving feature, then the expected result might be the field plotted in Figure 5.12(b) which could represent any scaler variable of interest. Since the POD interpolation techniques see the model as a black-box, and only acts on the data, it will always result in the surface shown in Figure 5.12(a). In situations where the flow regime changes, or problems with moving shocks or features, this could be a significant problem.

5.5 Reduced Order Modelling of Unsteady Fluid Problems

In this section, problems where $\boldsymbol{\alpha}$ contains time are considered. Galerkin projection is the most frequently used method for generating solutions for an unsteady problem using POD. It was decided that Galerkin projection was not suited to the examples considered in this thesis. Since Galerkin projection is so popular, justification was needed to explain why it was not being considered for application here.

In Galerkin projection, the exact time dependent governing equations are directly projected over the POD basis [89]. A ROM is generated which predicts solutions at varying time, t, and at a single set of flow control parameters i.e. $\boldsymbol{\alpha} = t$. Suppose the time dependent flow field state vector, $\mathbf{u}(\mathbf{x}, t)$, is represented by a linear combination of the POD modes, Φ_i, as

$$\mathbf{u}(\mathbf{x}, t) = \sum_{i=1}^{M} T_i(t)\Phi_i(\mathbf{x}) \tag{5.17}$$

where $T_i(t)$ denotes the time dependent amplitude for the POD mode Φ_i. It is assumed that the governing equation for the system is defined, in general form, as

$$\frac{\partial \mathbf{u}}{\partial t} = \Lambda_1 \mathbf{u} + \Lambda_2(\mathbf{u}, \mathbf{u}) + \Lambda_3(\mathbf{u}, \mathbf{u}, \mathbf{u}) \tag{5.18}$$

where Λ_1 is a linear operator, Λ_2 is a quadratic operator and Λ_3 is a cubic operator. Projecting this equation onto the POD basis, using an inner product, results in the equation

$$\left(\frac{\partial \mathbf{u}}{\partial t}, \Phi_i\right) = (\Lambda_1 \mathbf{u}, \Phi_i) + (\Lambda_2(\mathbf{u}, \mathbf{u}), \Phi_i) + (\Lambda_3(\mathbf{u}, \mathbf{u}, \mathbf{u}), \Phi_i) \tag{5.19}$$

and substituting the expansion of equation (5.17) into this equation gives

$$\frac{\mathrm{d}T_k}{\mathrm{d}t} = \sum_l T_l(\Phi_k, \Lambda_1(\Phi_l)) + \sum_{l,m} T_l T_m(\Phi_k, \Lambda_2(\Phi_l, \Phi_m)) + \sum_{l,m,n} T_l T_m T_n(\Phi_k, \Lambda_3(\Phi_l, \Phi_m, \Phi_n)) \tag{5.20}$$

This is the ROM for equation (5.18) using Galerkin projection onto a POD basis [90].

Either the discrete representation of the FOM equations, or the continuous FOM equations, can be used in the projection. Using the continuous equations has the advantage that the ROM does not need to be made for a specific FOM. This means that any solution algorithm can be used to generate the snapshots [90]. It must be remembered that the discrete representation might contain unphysical stabilisation terms, which would also need to be accounted for in the governing equations which the POD modes are projected upon [89].

In simulations involving several different physical variables, the choice of which inner product to use is not obvious [110]. Standard inner product definitions may not yield physically meaningful results when acting on solution vectors. Barone et al [90] and Rowley et al [110] both detail how the numerical stability of the Galerkin projection based ROM depends upon the proper selection of inner product definition. They provide detailed formulations of how to define the inner product for any set of governing equations. This decreases the generality of the Galerkin projection method, since a new set of equations will have a new condition for the inner product. However, it is noted that, if coded intelligently, only subroutine for calculating the inner product would need changing, as this is the only part of the projection which is not general [110].

The solution of equation (5.20) requires the evaluation of integrals over the entire domain for each mode. This could potentially be computationally expensive [89]. To reduce this cost, the series of POD modes Φ_i, making up the POD basis $\mathbf{\Phi}$, is often truncated. Suppose $\mathbf{\Phi}^{\tilde{M}}$ represents a POD basis that has been truncated to contain $\tilde{M}$ modes. This truncation can significantly reduce the computational cost, but may lead to instabilities in the ROM. An example in which this is shown to be particularly important is ocean flow models. Here, with high characteristic Reynolds numbers, energy transfer between large and small scale flow structures is very important. If the value of $\tilde{M}$ is selected to be too small, this transfer is inhibited and a projection type ROM based on $\mathbf{\Phi}^{\tilde{M}}$ is unstable [91].

The appropriate selection of a value for $\tilde{M}$, as described by Hung et al [97], can be assisted by interpreting the eigenvalues λ_i of equation (5.6) as representing the relative energy contained in each POD mode, Φ_i. These eigenvalues give the relative importance of each POD mode. As previously discussed in the field of ROM this relative importance is usually termed the energy of each mode, but may not be a physically meaningful measure of energy. The total energy is captured by the non-truncated basis $\mathbf{\Phi}$ and a suitable measure of this is the sum of the eigenvalues $\sum_{i=1}^{M} \lambda_i$. An important step during POD is arranging the eigenvalues and eigenvectors in ascending order, such that λ_1 is the largest eigenvalue and λ_M is the lowest. This allows the calculation of the relative energy captured by the truncated basis function $\mathbf{\Phi}^{\tilde{M}}$ as $\sum_{i=1}^{\tilde{M}} \lambda_i$. The value of $\tilde{M}$ is then selected such that

$$\sum_{i=1}^{M} \lambda_i \approx \sum_{i=1}^{\tilde{M}} \lambda_i \tag{5.21}$$

The percentage of energy captured by $\mathbf{\Phi}^{\tilde{M}}$ can be easily calculated as $(\sum_{i=1}^{\tilde{M}} \lambda_i / \sum_{i=1}^{M} \lambda_i) \times 100\%$. The POD method has the property of rapid convergence of the energy associated with each POD mode, which results in a small value of $\tilde{M}$ for most simulations [100].

Despite the rapid convergence of energy, truncating the basis function may still be a risky process. If each POD mode is considered a dimension, in the space which represents the solutions of a given set of equations, truncating removes one of these dimensions. ROMs based on Galerkin projection are frequently unstable, due to the truncation of the POD basis which neglects high order modes [89].

Further disadvantages of Galerkin projection are detailed by Lucia et al [111]. It is observed that the boundary conditions are not explicitly accounted for, so the basis function used for the projection needs to be formed to meet them. Non–linearities, which couple unknown variables, cause the number of terms in a Galerkin projection to increase rapidly. These terms can contain of the order of M^3 components, making them difficult to evaluate [110, 111].

In addition to these problems, it is also well understood that a particular set of snapshots will result in POD modes which are only appropriate in a limited parameter space. For example, Lieu et al [112] applied Galerkin projection to a complete F–16 aircraft model, which depended only on the angle of attack and free stream Mach number, M_∞. Snapshots were taken from an unsteady finite element FOM by inducing vibrations in the system at a particular M_∞. The result is a POD basis which only applies to a ROM for the flow at that M_∞ value. This posed a serious obstacle when considering applying these methods to optimisation problems, in which a parameter space is to be explored. An attempt to overcome this difficulty is made [112] by performing a subspace angle interpolation of the POD modes, in order to retain orthogonality between each mode, and then projecting these onto the governing equations.

To circumvent this problem, and others introduced by projection methods, it was worth considering POD as a basis for interpolation, as detailed for the steady case in section 5.4.2. However, it has already been shown that POD interpolation techniques turn out to be equivalent to a simple nodewise interpolation. In this case, was it worthwhile pursuing them further? To answer this question, information on how direct interpolation of the solution field compared to projection based ROM techniques was required. It was difficult to find comparisons of this nature but, in an early POD/ROM paper, Pettit and Beran [113] speculated that using a ROM should be better than interpolation, but stated that this needed to be tested. Degroote et al [114] provide a comparison between Kriging interpolation of the solution field, and ROMs projected onto basis functions interpolated with respect to the parameter space. They found that direct interpolation of the solution field gives errors of similar order to those achieved by the projection ROMs.

Recently, Wang et al [107] also compared POD interpolation and POD projection techniques. In one of their examples, with a four dimensional parameter space, they found that the interpolation technique produces errors which are orders of magnitude lower

than those produced with the projection technique. However, in another example, using a six-dimensional parameter space, they find the opposite result to be true. Based upon these results, they argue that POD interpolation was ineffective. However, it should be noted that only linear interpolation is used in their examples.

When considering the stability problems [111, 113, 115] and boundary condition violations [111] which are reported when applying POD projection methods to simple two dimensional flow problems, interpolation appeared an attractive alternative. The success of a direct interpolation technique does however rely on a smooth variation of the unknowns with respect to the parameters. In applications, such as electromagnetic scattering [116], where a small variation in parameters can result a sudden change in the unknowns, at a fixed point in space, the sampling resolution required to capture parametric variation may be so high as to make direct interpolation fruitless. In these cases, POD projection techniques may perform better.

Considering the applications of interest in this thesis it made sense to extend the POD interpolation technique to unsteady problems. An unsteady POD interpolation technique was developed [2] as part of this thesis and is detailed in the remainder of this section.

5.5.1 The Unsteady Full Order Model

In previous examples, the steady 2D FLITE solver was used as the FOM. In this section, the 3D unsteady inviscid compressible flow past a moving boundary was considered. The solver used is a 3D version of FLITE, using an arbitrary Lagrangian–Eulerian (ALE) formulation to handle the moving boundaries. The spatial domain is discretised using an unstructured tetrahedral mesh and the Euler equations are approximated using a cell-vertex finite volume method in space. Stabilisation and discontinuity capturing is achieved by the explicit addition of artificial viscosity. A fully implicit three-level second-order method is adopted for the time discretisation. At each time step, the implicit equation system is solved by explicit iteration, with multigrid acceleration [82, 117, 118].

For the examples included in this section, the explicit solution process continued within each time step until the residual had been reduced by five orders of magnitude.

5.5.2 POD and Changing Meshes

In the examples considered in this section, and for many other examples of industrial interest [86], meshes may change as the parameters defining the problem of interest change. This could be between time steps, in an unsteady problem, or between different

designs, for shape optimisation techniques. This introduces a difficulty when constructing the POD basis function using snapshots, since the number of degrees of freedom in the model could be different on different meshes. Furthermore, the POD construction acts on the computational space only and includes no information on the spatial location of each node.

Anttonen et al [119] attempted to apply POD to a deforming grid used to model unsteady flow past an oscillating cylinder. The grid was deformed in a manner that retained connectivity between nodes, but the position of each node in physical space changed with time. Applying the same procedure in both cases, they find a loss of accuracy when comparing POD on a deforming mesh to POD on a constant mesh. To alleviate this problem, a series of POD basis functions were constructed for several meshes, with the model selecting the most appropriate POD basis from the mesh nearest to the current mesh.

An alternative solution has been developed by Fang et al [91, 120], who employed a POD inverse model for an adaptive mesh ocean model. A fixed grid was constructed to interpolate solutions to and from the adaptive mesh so that, when the snapshots were interpolated onto this fixed mesh, the normal POD procedure could be followed. The fixed reference mesh was constructed to be as fine as the finest adapted mesh used in the simulation. Their procedure for interpolation is as follows: for each node in the fixed mesh, the element of the adapted mesh in which the node lies is identified; using least squares, fit a local high–order polynomial to the patch of nodes around the element; use this polynomial to calculate the interpolated values. Excellent results are achieved using this method and errors in the velocity are reduced by half by increasing the polynomial order from linear to quadratic.

The Delaunay graph concept [84], discussed in Section 4.2.2, was used to generate moved meshes from an original mesh in this work, which resulted in the same nodal connectivities on each mesh. Sufficient accuracy was obtained, when applying POD, without the need for more sophisticated techniques.

5.5.3 Unsteady POD Interpolation

The simplest way to extend the steady POD interpolation method to unsteady problems would be to include t in $\boldsymbol{\alpha}$ as an additional dimension to interpolate over. This was not the approach adopted here for the following reasons:

- For most optimisation applications, there is no interest in interpolating in time, only in the rest of the parameter space

- It was identified during the steady interpolation study that this technique is unable to correctly interpolate moving features. Since, in the vast majority of cases, features will move with respect to time it is not suitable to interpolate along this dimension
- The approach adopted resulted in significantly less interpolation operations, and, hence, increased CPU cost savings, than simply adding t to the list of parameters.

Suppose that, for a particular problem, M snapshots are computed, using the FOM for M different values, $\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_M$, of the flow parameter set on a mesh with D nodes. If each snapshot involves N time steps, the output from the FOM is a total of $N \times M$ solution vectors, $\boldsymbol{p}^n(\boldsymbol{\alpha}_k)$, where $k = 1, \cdots, M$ and $n = 1, \cdots, N$. These solution vectors can be used to form the columns of a snapshot matrix

$$\boldsymbol{P} = \begin{bmatrix} p_1^1(\boldsymbol{\alpha}_1) & p_1^2(\boldsymbol{\alpha}_1) & \cdots & p_1^N(\boldsymbol{\alpha}_1) & p_1^1(\boldsymbol{\alpha}_2) & \cdots & p_1^N(\boldsymbol{\alpha}_M) \\ \vdots & \vdots & \ldots & \vdots & \vdots & \ldots & \vdots \\ p_D^1(\boldsymbol{\alpha}_1) & p_D^2(\boldsymbol{\alpha}_1) & \cdots & p_D^N(\boldsymbol{\alpha}_1) & p_D^1(\boldsymbol{\alpha}_2) & \cdots & p_D^N(\boldsymbol{\alpha}_M) \end{bmatrix} \tag{5.22}$$

where $p_I^n(\boldsymbol{\alpha}_k)$ denotes component I of the vector $\boldsymbol{p}^n(\boldsymbol{\alpha}_k)$. This matrix can be factorised, using SVD, as

$$\boldsymbol{P} = \boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{V}^* \tag{5.23}$$

where the columns of $\boldsymbol{\Phi}$, denoted by Φ_j for $j = 1, \cdots, N \times M$, are the left singular vectors of $\boldsymbol{P}$ and these will be used as the POD modes. In addition, $\boldsymbol{\Sigma}$ is the diagonal matrix containing the singular values of $\boldsymbol{P}$, which indicate the relative importance of each mode, and the columns of $\boldsymbol{V}^*$ are the right singular vectors of $\boldsymbol{P}$. The vectors Φ_j are mutually orthogonal and can be regarded as a basis for an $N \times M$ dimensional space in which each element represents a different solution vector for the fluid problem under consideration. It follows that each column of $\boldsymbol{P}$ can be reconstructed, using the POD modes, as

$$\boldsymbol{p}^n(\boldsymbol{\alpha}_k) = \sum_{j=1}^{N \times M} T_j^n(\boldsymbol{\alpha}_k)\Phi_j \tag{5.24}$$

For each set of parameter coordinates, the vector $\mathbf{T}^n(\boldsymbol{\alpha}_k)$ of coefficients in this reconstruction can be viewed as a path through the coordinate system $\boldsymbol{\Phi}$. The goal was to predict a path that the system takes through $\boldsymbol{\Phi}$ for a set of parameter values not included in the original sampling. This was achieved by considering each member of $\boldsymbol{\Phi}$ in

turn and creating new $N \times M$ snapshot matrices, $\boldsymbol{S}_j$, for $1 \leq j \leq N \times M$, where

$$\boldsymbol{S}_j = \begin{bmatrix} T_j^1(\boldsymbol{\alpha}_1) & \cdots & T_j^1(\boldsymbol{\alpha}_M) \\ \vdots & \cdots & \vdots \\ T_j^N(\boldsymbol{\alpha}_1) & \cdots & T_j^N(\boldsymbol{\alpha}_M) \end{bmatrix} \tag{5.25}$$

Each column in $\boldsymbol{S}_j$ consists of the values of the coefficients, at different times, for a given POD mode, Φ_j, for a given snapshot. A SVD of $\boldsymbol{S}_j$ produces the matrix $\boldsymbol{\Psi}^j$ of left singular vectors, whose columns Ψ_i^j, where $i = 1, \cdots, M$, will be used as POD modes. Again, each matrix $\boldsymbol{\Psi}^j$ can be viewed as a coordinate system, where a point represents the path taken by a given set of parameters along a given axis of $\boldsymbol{\Phi}$. These paths can be reconstructed using

$$\boldsymbol{T}_j(\boldsymbol{\alpha}_k) = \sum_{i=1}^{M} Q_i^j(\boldsymbol{\alpha}_k)\Psi_i^j \tag{5.26}$$

where $\boldsymbol{T}_j$ is the vector whose nth component is T_j^n. Values of $\boldsymbol{Q}^j(\boldsymbol{\alpha}_k)$ can be readily calculated, when the POD modes are all generated. When this process was completed, the solution vector $\boldsymbol{p}^n(\boldsymbol{\alpha})$, at a new set of parameter coordinates $\boldsymbol{\alpha}$, was determined. Interpolation was required to find the values of $\boldsymbol{Q}^j(\boldsymbol{\alpha})$ and these interpolated values may be used to obtain $\boldsymbol{T}_j(\boldsymbol{\alpha})$, as

$$\boldsymbol{T}_j(\boldsymbol{\alpha}) = \sum_{i=1}^{M} Q_i^j(\boldsymbol{\alpha})\Psi_i^j \tag{5.27}$$

The ROM approximation to the solution vector follows as

$$\boldsymbol{p}^n(\boldsymbol{\alpha}) = \sum_{j=1}^{N \times M} T_j^n(\boldsymbol{\alpha})\Phi_j \tag{5.28}$$

This process resulted in fewer interpolation operations than nodewise interpolation. The number of interpolation operations undertaken to generate a single solution without using POD would be $D \times N$. Using the unsteady POD interpolation scheme developed here, the number of interpolation operations was $M \times N$. The use of $M < D$ will result in a computational saving using POD based interpolation. This result is independent of the interpolation method implemented.

It may first seem that this condition will almost always be fulfilled since, in industrial applications, D is very large. However, since the interpolation is applied to each node independently, the nodes used to generate the POD modes need only be in the area of interest in the design or optimisation process. For example, the nodes on boundary surfaces may be the only nodes of interest. In these cases, D may be smaller, which may

lead to nodewise interpolation being more efficient. It also depends on how many snapshots are required to satisfactorily sample the parameter space, which is also problem dependent.

It is also worth considering how many interpolation operations would be required if the steady POD interpolation technique was extended by simply adding time as an extra interpolation dimension. In that case, each timestep would need to be interpolated separately leading to $N \times M \times N$ operations, which is N times more than required using the technique developed here.

5.5.3.1 Initial Validation

The examples which were considered initially involve the periodic oscillation of a wing, with the FOM solver used to simulate ten complete oscillation cycles. The results of the final cycle were used to form a snapshot for one prescribed set of the flow parameters. Different values of the parameter sets lead to different snapshots. These snapshots were used to create a ROM which could be employed to efficiently produce the flow solution for different parameter values. For illustration, it was only the pressure values of the final cycle that were used to form the snapshots.

In all examples a motion was initially applied to the tip of the wing. The deformation at the tip was then interpolated along the wing, such that there was zero deformation at the root, and applied to each boundary node. Pitch oscillation refers to a rotation of the tip, measured in degrees, and heave oscillation refers to a vertical translation of the tip, measured in the dimensionless coordinate system employed.

Example 1 is included to demonstrate the validity of the approach that has been described. It involved inviscid flow over an oscillating ONERA M6 wing at a free stream Mach number of 0.2. In the dimensionless coordinates employed here, the wing chord was 10 units and the wing span was 14 units. The wing tip followed a prescribed sinusoidal pitch oscillation, with a maximum amplitude of ± 3 degrees, and the root of the wing was held fixed. The amplitude of the pitch oscillation varied linearly between the fixed root and the tip. The only flow parameter that was varied was the reduced frequency, f, of the oscillation.

An initial mesh with 44 573 nodes was generated and, to allow for the complete movement of the wing, a further 31 meshes were obtained from this mesh by mesh movement. The simulations were performed using 32 time steps per cycle, with each mesh representing the geometrical configuration at one time level. With this approach, oscillations of different reduced frequencies could be simulated by simply altering the size of the physical time interval between successive meshes.

Identifier	Reduced Frequency	Time Step Size
$f1$	0.1563	0.2
$f2$	7.813×10^{-2}	0.4
$f3$	5.208×10^{-2}	0.6
$f4$	3.906×10^{-2}	0.8
$f5$	3.125×10^{-2}	1.0

TABLE 5.3: Reduced Frequencies and time step sizes used in Example 1

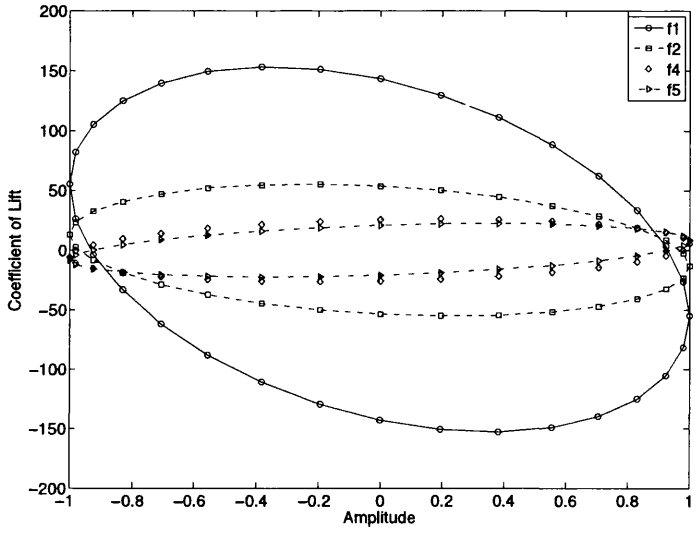


FIGURE 5.13: The lift coefficient as a function of pitch amplitude for each snapshot employed in Example 1. In this graph the units of amplitude are dimensionless, normalised by the maximum pitch amplitude

Five FOM simulations were performed, with the reduced frequency values employed, designated $f1$ to $f5$, and the corresponding time step sizes, shown in Table 5.3. To illustrate the effectiveness of the proposed process, the solutions obtained with the reduced frequencies $f1$, $f2$, $f4$ and $f5$ were used to create snapshots and the ROM was utilised to predict the solution at the reduced frequency $f3$. The lift history for each of these snapshots is shown in Figure 5.13.

The snapshot pressure fields were used to create the snapshot matrix $\boldsymbol{P}$ of equation (5.22) and the POD modes, $\boldsymbol{\Phi}_j$, where $j = 1, \cdots, 32 \times 4$, were generated from this matrix. The first four modes on the initial mesh, obtained in this fashion, are plotted on the upper surface of the wing in Figure 5.14.

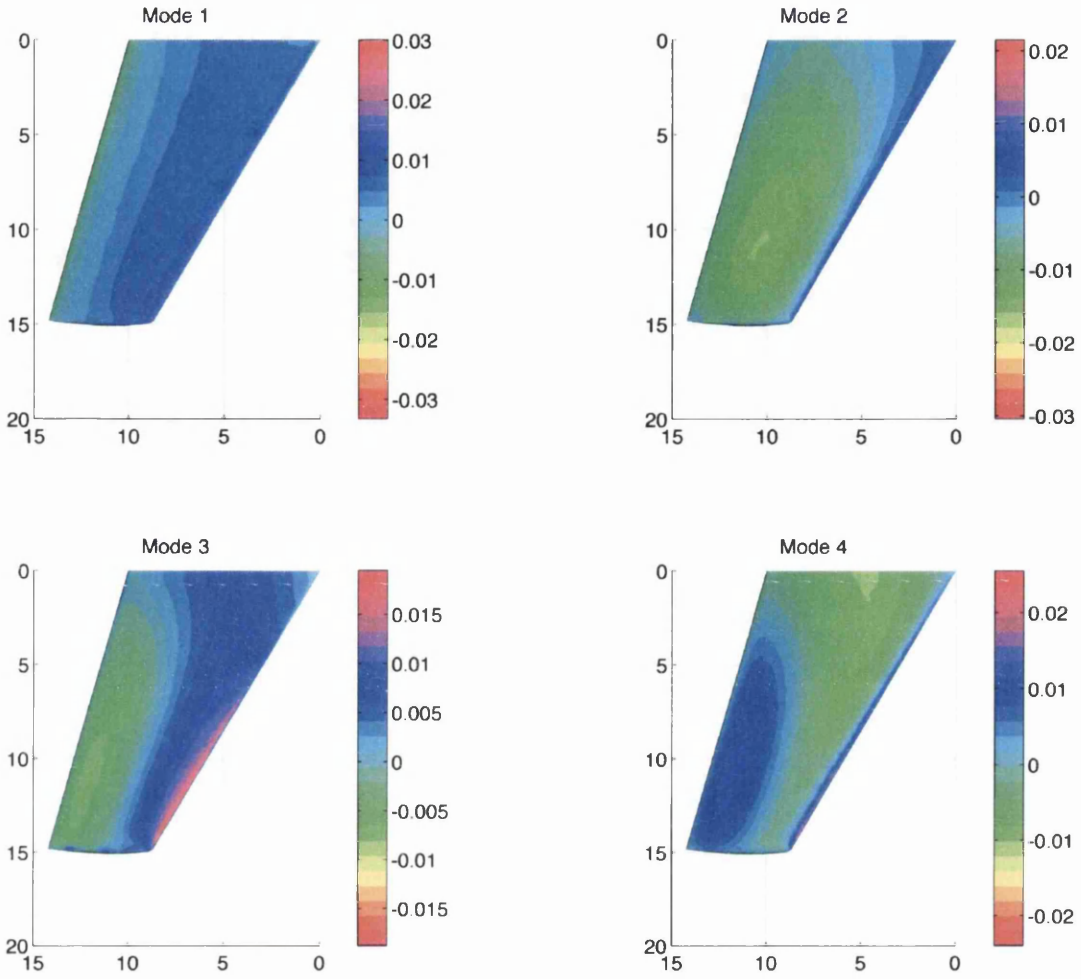


FIGURE 5.14: The first four $\mathbf{\Phi}_j$ modes of the pressure coefficient on the upper surface of the wing of Example 1

A set of coefficients was calculated, for each mode for each snapshot, and Figure 5.15 shows the variation in time of the coefficient of the first mode, $\mathbf{\Phi}_1$, for each snapshot.

The time variation of the first spatial modes were used to generate the snapshot matrix $\boldsymbol{S}_1$, as shown in Equation (5.25). The POD modes $\mathbf{\Psi}_i^1$, where $i = 1, 2, 3, 4$ are obtained by applying SVD to this matrix. These temporal modes are plotted in Figure 5.16.

A coefficient was calculated for each of these modes, such that a linear combination of the modes reconstructs one of the snapshots in Figure 5.15, as shown in equation (5.26). These coefficients are a function of the flow parameter, which in this case was the reduced frequency of the oscillation. One dimensional linear, cubic spline and Hermite interpolation were used to determine the values of the coefficients in the model appropriate for the reduced frequency $f3$. This was accomplished using the MATLAB [17]

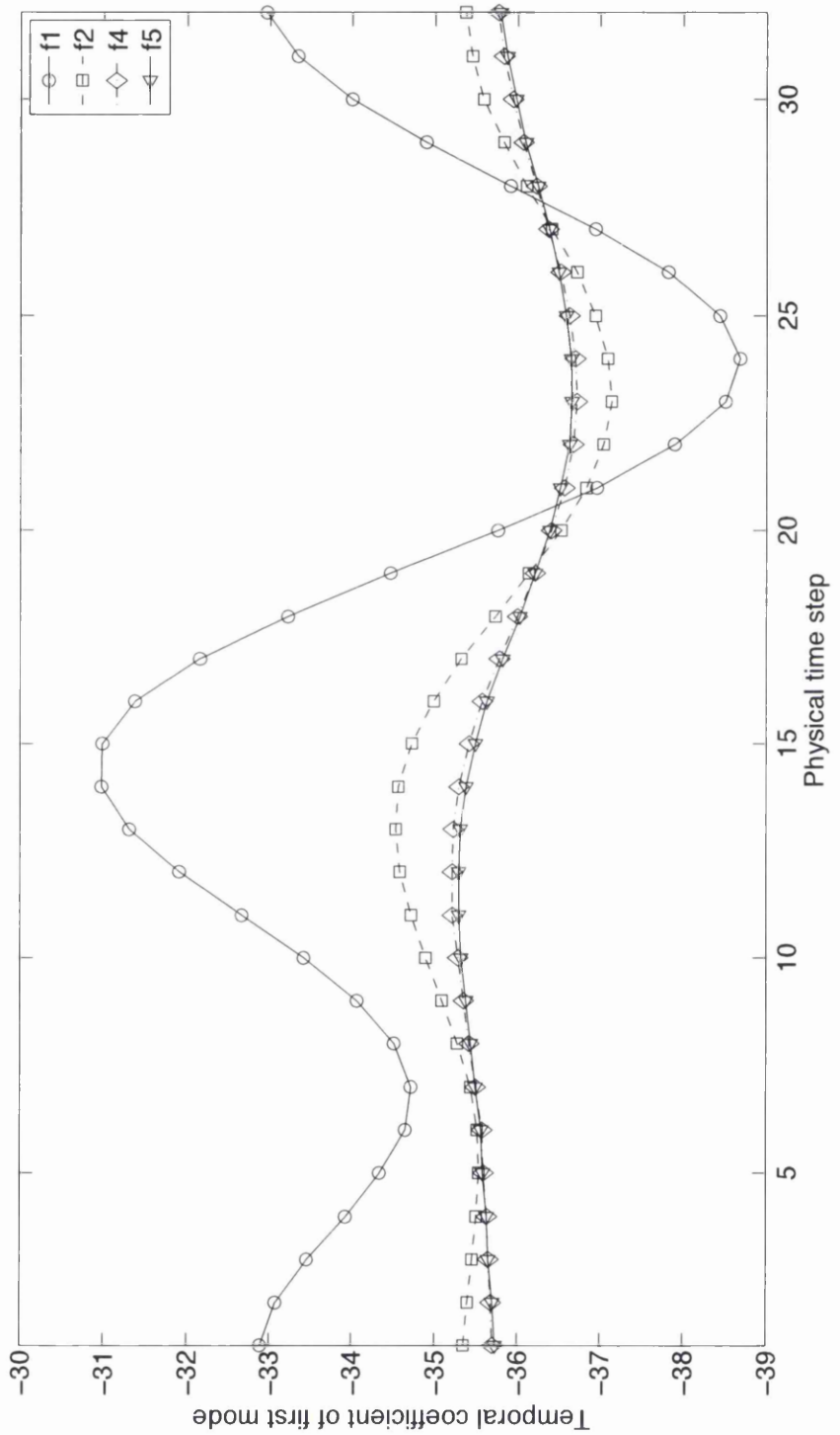


FIGURE 5.15: Time variation of the coefficient of the first mode for each snapshot of Example 1. The y-axis of each graph gives the value of the coefficient.

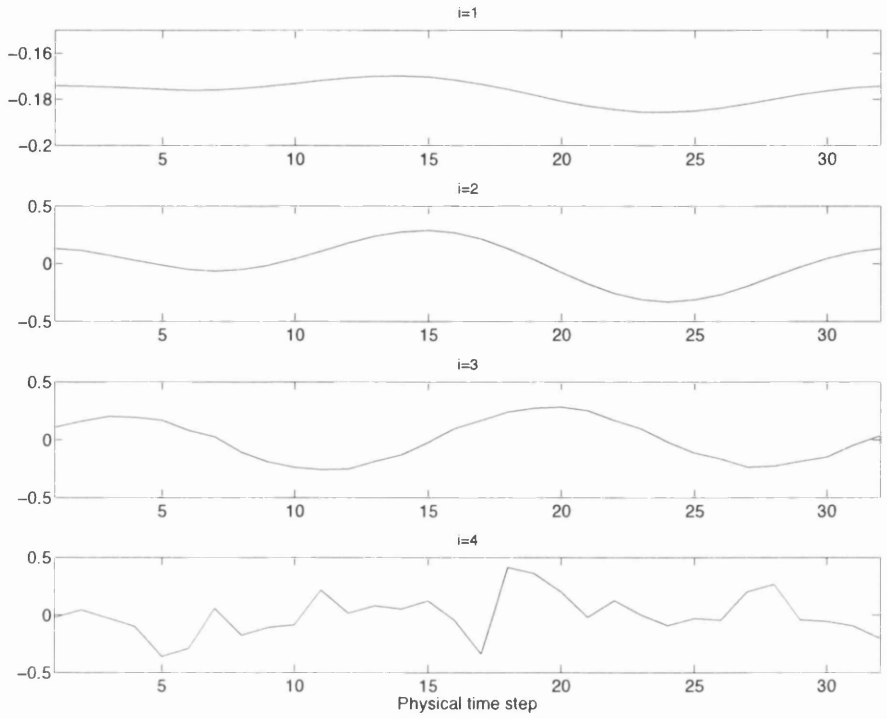


FIGURE 5.16: Temporal modes Ψ_i^1, for $i = 1, 2, 3, 4$, for Example 1

function `interp1`. The resulting ROMs were used to predict the lift polar for the reduced frequency $f3$ and the predicted lift polars are compared to that obtained from the FOM solution for this reduced frequency in Figure 5.17.

The pressure coefficients on the upper surface of the wing, at the 15th time step, predicted by these ROMs are compared to the distribution obtained from the FOM at this time in Figure 5.18.

The root mean square error in the pressure, Er_p, and in the lift coefficient, Er_{C_ℓ} were determined and expressed as a percentage of the range of p and C_ℓ respectively. The results enabled a quantitative comparison to be made between the performance of the different methods. The use of piecewise cubic Hermite interpolation produced the lowest values of $Er_p = 0.2242\%$ and $Er_{C_\ell} = 1.5505\%$.

All ROM calculations in this chapter were performed on a standard desktop machine, with an Intel Core i5–2500 running at 3.30GHz, while the FOM calculations used a cluster of 4 1400 Mhz AMD Opteron 240 processors run in parallel. The typical CPU time for a single FOM simulation was 13 500 s. On average, the total CPU time used to complete all the tasks in the ROM for this example was 7 s, which included generating the

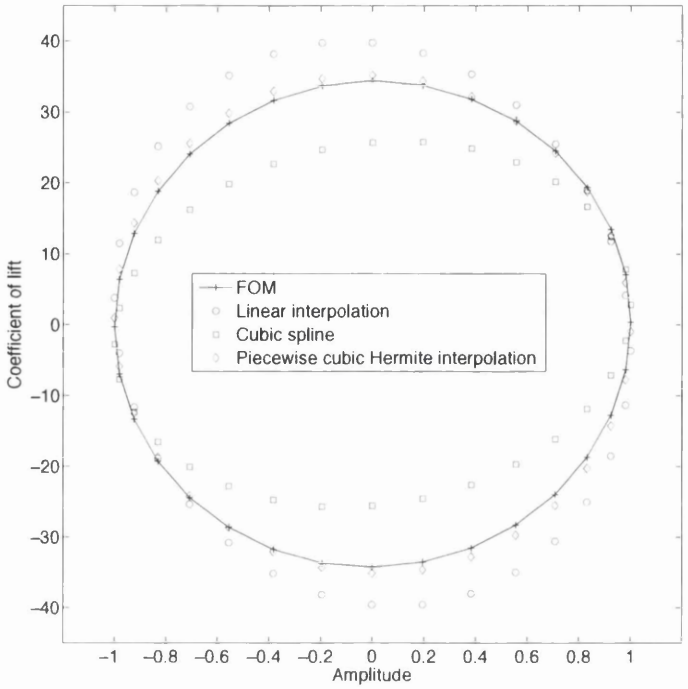


FIGURE 5.17: Comparison between the lift obtained using the ROMs constructed for Example 1 with that obtained from the FOM

modes and calculating the coefficients. Subsequent predictions of the solution at different values of the frequency had negligible CPU cost, since the modes and coefficients are stored. To put this into perspective, plotting the three-dimensional solutions involved a larger CPU cost than generating the solutions using the ROM on the desktop machine.

5.5.3.2 Computational Effectiveness of the ROM

The ROM was developed with the objective of reducing the cost associated with using a computational simulation method as part of a design or optimisation process. The results presented for Example 1 illustrate the potential of the technique, but it is worth considering how this technique might be employed within the design process and how the resulting CPU costs might be estimated.

Since the technique was based upon interpolation, it would be naive to think that satisfactory accuracy could be achieved with the same number of snapshots for every example. The accuracy that can be achieved will depend upon the number of snapshots and also upon the location of the snapshots in parameter space. This means that validation will

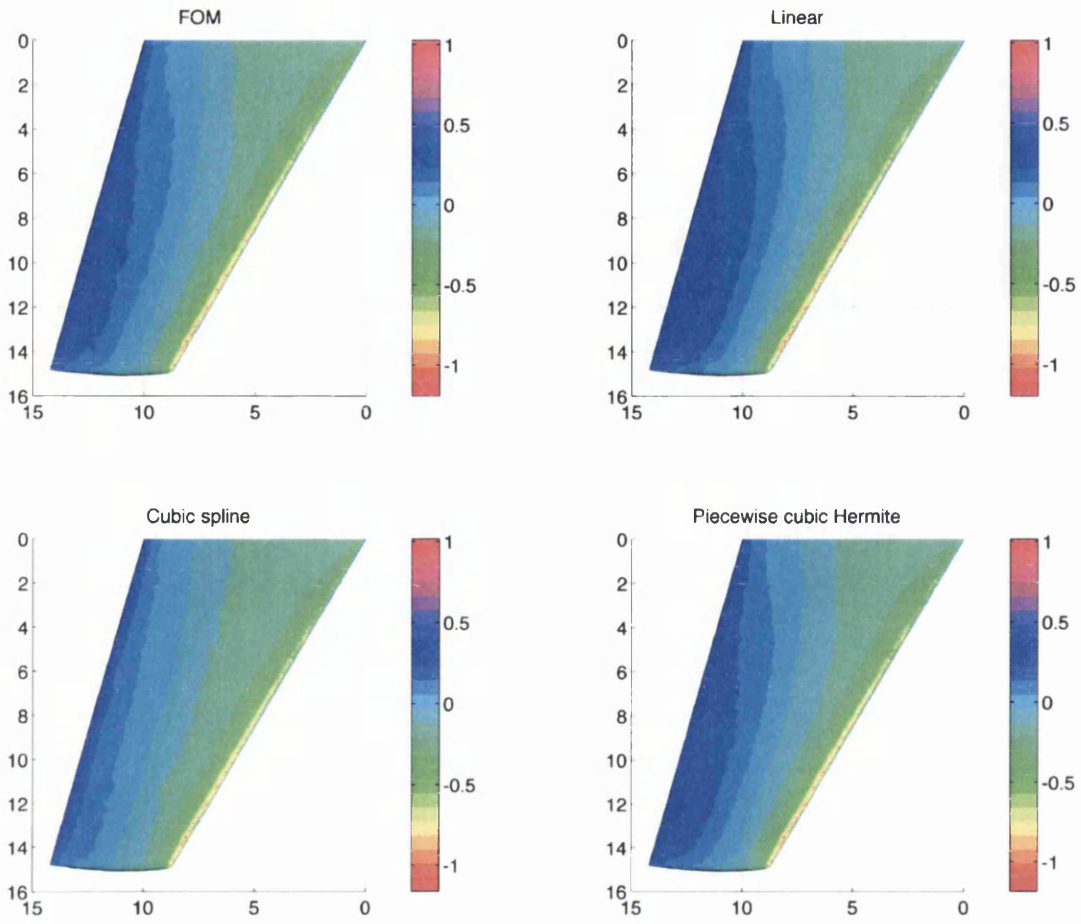


FIGURE 5.18: The pressure coefficients on the upper surface of the wing, at a prescribed time, computed for Example 1 with the FOM and predicted by the different ROMs

be necessary for each new example. Fortunately, however, this is actually quite a simple process.

As part of the normal procedure, the parameter space of interest will be sampled using the FOM and these samples are then available for use as snapshots for producing a ROM. A series of ROMs can be created by simply omitting each snapshot, in turn, during the ROM generation process and measuring the errors that are obtained when using the ROM to predict the omitted solution. If the error calculated in this manner is too large, more FOM solutions can be calculated and more snapshots generated. These additional FOM solutions can be targeted towards those areas of the parameter space in which the errors were originally found to be the largest. Of course, this still provides nothing more than an estimate of the error and thus the ROM will need to be continuously refined during the design or optimisation process.

To investigate the CPU costs involved in validating the ROM in this way, suppose that the number of solutions required in a particular parameter space, for a particular design

cycle, is M_T and suppose that the cost of a single FOM evaluation is C_{FOM}, expressed as seconds of wall clock time. It follows that the total CPU cost that would be incurred by using only the FOM in the design cycle would be

$$C_{TotalFOM} = M_T \times C_{FOM} \tag{5.29}$$

The total CPU cost when the ROM in used to perform the same task can be expressed as

$$C_{TotalROM} = M_T \times C_{ROM} + C_{Val} + C_{Build} + M \times C_{FOM} \tag{5.30}$$

where C_{Build} is the cost of building the ROM, but not including evaluation of the snapshots, C_{Val} is the cost of validating the ROM, C_{ROM} is the cost of evaluating a solution of the ROM and $M \times C_{FOM}$ represents the cost of calculating the M snapshots. The cost of validating the ROM, using the technique that has been described, can be expressed as

$$C_{Val} = M \times (C_{Build} + C_{ROM}) \tag{5.31}$$

It follows that the requirement for the process to be worthwhile, in terms of CPU saving, is that

$$C_{TotalFOM} > C_{TotalROM} \tag{5.32}$$

and, using the above expressions, this requires that

$$M_T > \frac{M \times (C_{Build} + C_{ROM} + C_{FOM}) + C_{Build}}{C_{FOM} - C_{ROM}} \tag{5.33}$$

Since it is reasonable to assume that

$$C_{Build} \ll M \times (C_{Build} + C_{ROM} + C_{FOM}) \tag{5.34}$$

the requirement of equation (5.33) may be expressed in the form

$$\frac{M_T - M}{M} > \beta - 1 \tag{5.35}$$

where

$$\beta = \frac{C_{FOM} + C_{Build} + C_{ROM}}{C_{FOM} - C_{ROM}} \tag{5.36}$$

The minimum number of new solutions which would need to be calculated using the ROM, so as to result in a CPU saving, can then be determined, in terms of β, as

$$(M_T - M)_{min} = M(\beta - 1) \tag{5.37}$$

When $C_{ROM} + C_{Build}$ is much smaller than C_{FOM}, the value of β will be very close to unity. In this case, it would always be worth using this technique provided a ROM of

satisfactory accuracy can be achieved for $M < M_T$. In the remainder of this chapter, the value of $(M_T - M)_{min}$ will be used as an indication of the CPU saving that can be achieved by using the ROM.

5.5.3.3 Interpolation Using Radial Basis Functions

As the dimension of the parameter space increased, the process of interpolation for the required coefficients became considerably more complicated. In this case, an attractive alternative was to employ interpolation using radial basis functions (RBF). To illustrate this process, suppose that the entries in the $L \times 1$ vector $\boldsymbol{f}$ are the values, $f_\ell, \ell = 1, \cdots, L$, of an unknown function $F(\boldsymbol{x})$ at a set of data points, $\boldsymbol{x}_\ell, \ell = 1, \cdots, L$ in $\in \mathbb{R}^d$. The RBF approximation to F is the function

$$F_{RBF}(\boldsymbol{x}) = \sum_{\ell=1}^{L} w_\ell \Theta(|\boldsymbol{x} - \boldsymbol{x}_\ell|) \tag{5.38}$$

where Θ is defined, employing a multiquadratic form, as

$$\Theta(|\boldsymbol{x} - \boldsymbol{x}_\ell|) = \left(|\boldsymbol{x} - \boldsymbol{x}_\ell|^2 + c^2\right)^{1/2} \tag{5.39}$$

Here, c is a scalar parameter that effects the radius of influence of the data points. Suppose that the entries in the $L \times 1$ vector $\boldsymbol{w}$ are the values of the unknown coefficients, $w_\ell, \ell = 1, \cdots, L$ and let

$$W_{m\ell} = \Theta(|\boldsymbol{x}_m - \boldsymbol{x}_\ell|) \qquad 1 \leq \ell, m \leq L \tag{5.40}$$

It follows, that the unknown coefficients can be obtained by solving the matrix equation

$$\boldsymbol{W}\boldsymbol{w} = \boldsymbol{f} \tag{5.41}$$

where $\boldsymbol{W}_{m\ell} = W_{m\ell}$. When the value of c is specified, the unknown coefficients can be obtained, provided that $\boldsymbol{W}$ is non–singular.

It is known that the accuracy of the RBF approximation depends upon the value that is adopted for the parameter c [121]. Rippa [122] introduced an algorithm to find the optimum value for a particular data set but, for all the cases considered here, the additional expense incurred in performing an analysis of this type did not prove to be justified. Instead, the value of c was simply calculated to be the mean distance between any two data points.

The primary advantage of using RBFs was its meshless nature. Other interpolation techniques require gradient information at each data point, which may result in the need for a triangulation, or possibly, restrictions on the structure of the points. Using RBFs removed the need to consider the number of dimensions and allowed easy refinement by the addition of data points.

It is almost certainly the case that in the examples considered here, the advantages of RBFs were diminished due to the small number of parameters. However, when considering practical applications, such as shape optimisation where the parameter space may contain many more than three dimensions, the advantages may become more apparent. In addition, in a practical setting, the number of parameters may change through the development/design process. This can be achieved seamlessly with an RBF implementation.

When compared with 31 competing interpolation techniques on a range of functions, it is found that RBFs perform best in terms of accuracy for a variety of functions [123]. Interpolation techniques based on triangulations have the disadvantages of a large auxiliary storage required for the triangulation [123]. It is also found that the accuracy of techniques which require derivative information at the data points, such as Hermite interpolation, depends highly on the accuracy of the derivative estimates. Franke [123] states that extending triangulation and derivative based techniques to problems with more than two dimensions is extremely difficult and, in some cases, impossible. The primary disadvantage of RBF interpolation is, as the number of data points exceeds 100, the cost of calculating the weights becomes significant [123].

To validate this interpolation approach, Example 1 was reconsidered, and a ROM was again based upon FOM snapshots calculated with oscillation frequencies of $f1$, $f2$, $f4$ and $f5$. The pressure distribution on the upper surface of the wing, predicted by the RBF based ROM, was compared with corresponding results from the FOM in Figure 5.19. When RBF interpolation of the coefficients was employed, the magnitude of the errors produced when the ROM was used to predict the solution with the oscillation frequency $f3$ were $Er_p = 0.4345\%$ and $Er_{C_\ell} = 6.4293\%$. Although these were less accurate than the results obtained previously using Hermite interpolation, it was deemed appropriate to perform the analysis using RBFs, which will be the most desirable interpolation technique to use in real applications.

5.5.3.4 Results

ONERA M6 The numerical performance of the approach was demonstrated for three additional examples, involving additional complexity. For each of these examples, the

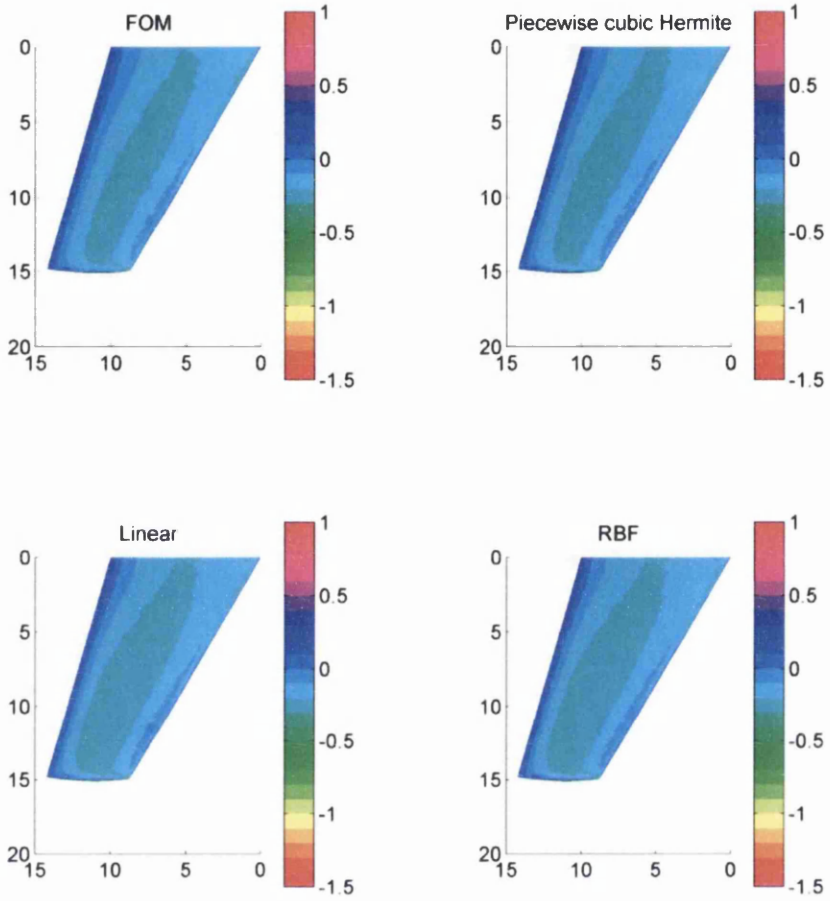


FIGURE 5.19: The pressure coefficients on the upper surface of the wing computed for Example 1 by the FOM and the ROM based upon RBF, Hermite and Linear interpolation at the time corresponding to the maximum deflection of the tip

RBF method was used to interpolate the values of the coefficients.

Example 2 involved transonic inviscid flow over an oscillating ONERA M6 wing at a free-stream Mach number of 0.84. The wing followed a prescribed sinusoidal pitch oscillation together with an oscillatory vertical heaving motion. The reduced frequency of the heave oscillation was prescribed to be $f = 2.778 \times 10^{-2}$, while the reduced frequency of the pitch oscillation was taken to be equal to $2f$. The wing was held fixed at the root and the values of both the pitch and the heave at any instant varied linearly between their values at the tip and at the root. The parameters of interest were the amplitude, a_p, of the pitch at the tip and the amplitude, a_h, of the heave at the tip, for the range of values $0 \leq a_p \leq 8^o$ and $0 \leq a_h \leq 1.6$.

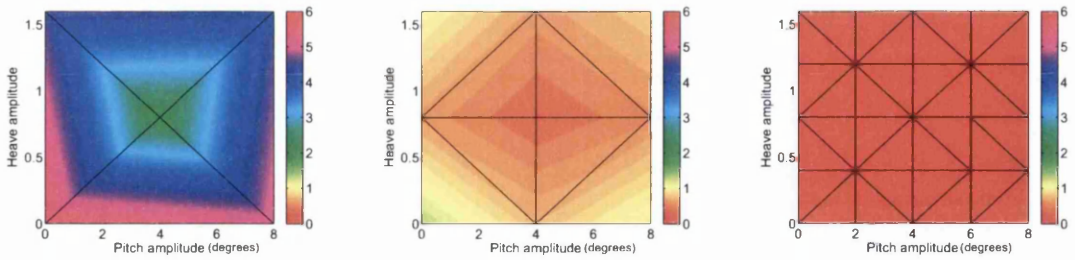


FIGURE 5.20: Variation of Er_p over the parameter space for Example 2 for (a) the 5 snapshot set; (b) the 9 snapshot set; (c) the 25 snapshot set. For each plot, the horizontal axis denotes the pitch amplitude and the vertical axis the heave amplitude.

Snapshots	Er_p	Er_{C_ℓ}
5	5.1563	24.807
9	1.1613	1.7277
25	0.4253	0.1413

TABLE 5.4: Maximum percentage errors calculated during the validation procedure for Example 2 using ROMs with different numbers of snapshots

An initial mesh was generated and 47 additional meshes were produced, by mesh movement, to describe the wing geometry during each stage of the heave oscillation. Three ROMs were constructed, employing 5, 9 and 25 snapshot sets. All snapshot sets included the case where both a_p and a_h were equal to zero, which represents a steady state solution. The performance of each of the three ROMs was tested in the proposed manner, with each snapshot, in turn, being omitted during the construction of the ROMs. This enabled the ROM error Er_p to be estimated at each snapshot location in turn.

The distribution of Er_p is plotted, as a function of $\boldsymbol{\alpha} = (a_p, a_h)$ in the parameter space, in Figure 5.20 for each of the three ROMs. Each vertex of the triangulation shown in these figures indicates the location of one of the snapshots. The triangulation was only used to interpolate, for plotting, between the computed values of Er_p at the vertices. It was interesting to note that the ROMs constructed using the 9 and the 25 snapshot sets were able to successfully predict the steady solution, using only unsteady snapshots. With the ROM based upon the 25 snapshot set, the steady state value of the lift coefficient was predicted to within 0.5268% of the actual value. Table 5.4 lists the maximum percentage error measurements for the unsteady solutions, calculated using the validation procedure discussed in Section 5.5.3.2. The maximum error for the 25 snapshot set occurred for the case $\boldsymbol{\alpha} = (8^o, 0)$. This point was located on the boundary

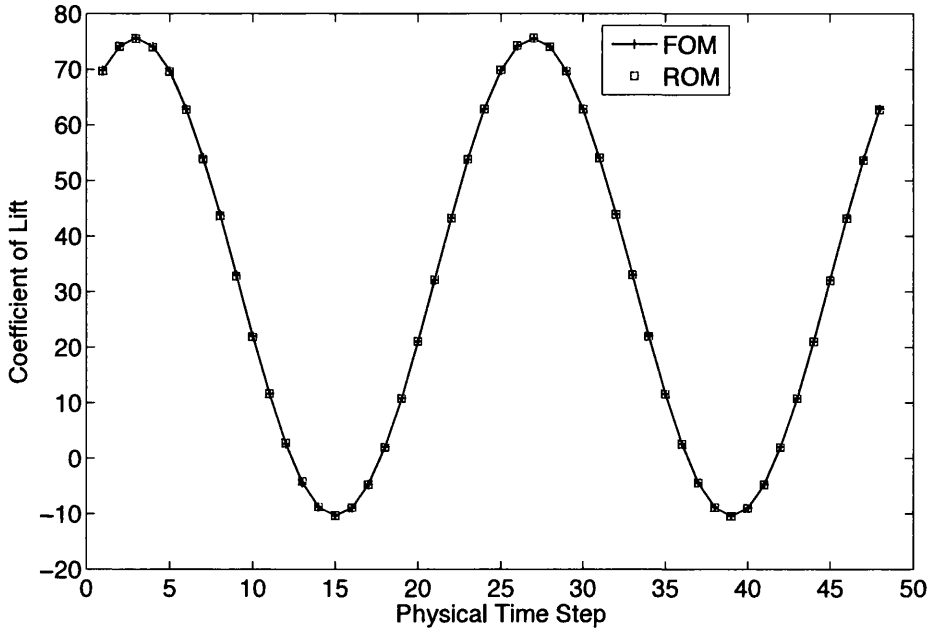


FIGURE 5.21: Comparison of the variation in the lift coefficient computed with the FOM with that obtained from the ROM using 25 snapshots for Example 2, at the point $\alpha = (8^o, 0)$ in the parameter space corresponding to the highest error.

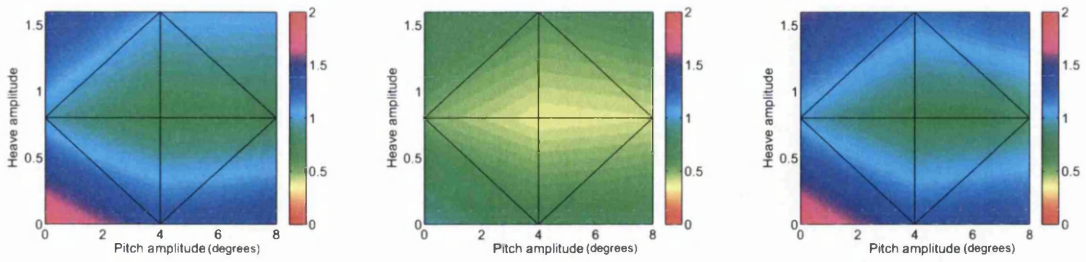
Snapshots	C_{Build}	C_{ROM}	$(M_T - M)_{min}$
5	1.7644	4.6279	0.0001
9	4.8053	8.6116	0.0004
25	115.6546	36.8045	0.0105

TABLE 5.5: CPU costs for Example 2

of the parameter space, where a high error would be expected. Figure 5.21 compares, at this point in the parameter space, the variation in C_ℓ computed by the FOM with the variation obtained from the ROM.

During the validation process, the values of C_{Build} and C_{ROM} were measured and this enabled the calculation of $(\beta - 1)$ for each set of snapshots. In the evaluation of β, the value of C_{FOM} was taken as 442 800s, which was a typical time for the FOM calculations. The resulting values are shown in Table 5.5. It can be observed that, for the largest snapshot set, the value of $(M_T - M)_{min}$ was 0.0105, which means that there would almost always be a net saving in CPU time.

The specification for Example 3 was exactly the same as that for Example 2, apart from the fact that the reduced frequency f was included in the list of flow parameters



(a) (b) (c)

FIGURE 5.22: Variation of Er_p over the parameter space for the ROM used in Example 3 for a frequencies (a) 4.167×10^{-2}; (b) 2.778×10^{-2}; (c) 2.083×10^{-2}. For each plot, the horizontal axis denotes the pitch amplitude and the vertical axis the heave amplitude.

of interest, so that $\boldsymbol{\alpha} = (a_p, a_h, f)$. The reduced frequency range was defined to be $2.083 \times 10^{-2} \leq f \leq 4.167 \times 10^{-2}$. Snapshots at different reduced frequencies were computed, on the same set of 48 meshes, by changing, as in Example 1, the size of the physical time step. For the purpose of illustration, the reduced frequency values 2.083×10^{-2}, 2.778×10^{-2} and 4.167×10^{-2} were selected for the snapshots, together with the nine values of (a_p, a_h) used in Example 2. This leads to a set of 27 snapshots. The resulting ROM was again tested, by omitting one snapshot in turn, and the resulting distribution of Er_p is plotted in Figure 5.22. The highest error in lift coefficient was 2.5045%, which occurred at the point $\boldsymbol{\alpha} = (0^o, 1.6, 2.083 \times 10^{-2})$. The time variation of the lift coefficient obtained from the ROM at this point is compared with the results of the FOM in Figure 5.23. For this case, $(M_T - M)_{min} = 0.0135$, which represents a significant potential saving in CPU time using the method.

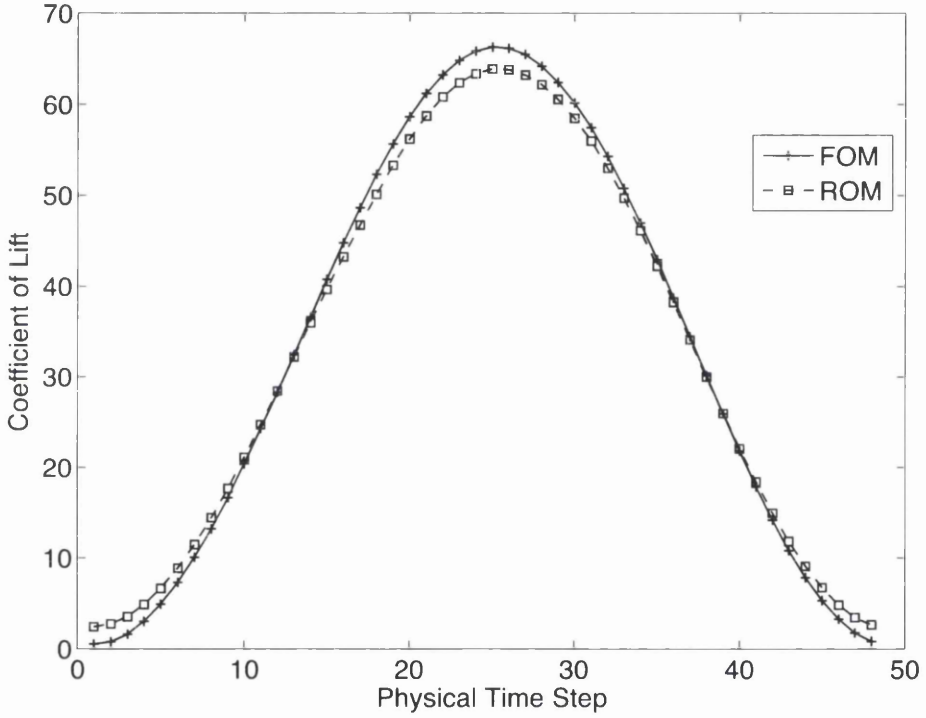


FIGURE 5.23: Comparison of the variation in the lift coefficient computed with the FOM with that obtained from the ROM for Example 3 at the point $\boldsymbol{\alpha} = (0^o, 1.6, 2.083 \times 10^{-2})$ in the parameter space corresponding to the highest error.

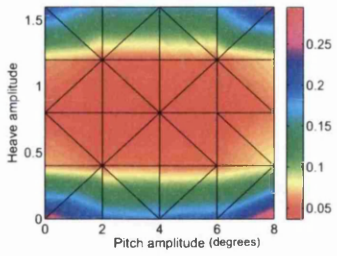


FIGURE 5.24: Variation of Er_p over the parameter space for the ROM used in Example 4. The horizontal axis denotes the pitch amplitude and the vertical axis the heave amplitude.

Snapshots	Er_p	Er_{C_ℓ}
5	3.9271	30.259
9	0.8944	3.6152
25	0.2954	0.3301

TABLE 5.6: Maximum error measurements calculated during validation procedure for Example 4

The specification for Example 4 was exactly the same as that for Example 2, apart from the fact that the values of the amplitudes of both the pitch and the heave at any instant varied quadratically between their values at the tip and at the root. The quadratic variation adopted was such that the values at the semi–span are a quarter of the values at the tip. The solution procedure followed the approach adopted for Example 2, and the computed values of Er_p, obtained with the 25 snapshot set, are plotted, as a function of the amplitude of the pitch at the tip and the amplitude of the heave at the tip, in Figure 5.24. The maximum percentage errors in the unsteady solutions constructed using 5, 9 and 25 snapshots are shown in Table 5.6, where the maximum Er_{C_ℓ} value was 0.3301% and the maximum Er_p value was 0.2954%. Despite the increased complexity of the motion, these error values were seen to be of a similar order to those occurring in Example 2. The maximum value of Er_p again occurred for the case $\boldsymbol{\alpha} = (8^o, 0)$.

For the case $\boldsymbol{\alpha} = (8^o, 1.6)$, the time variation of the lift coefficient obtained from the 25 snapshot ROM is compared to the results produced with the FOM in Figure 5.26. The surface pressure distribution computed with this ROM is compared, at two different times, with the results obtained with the FOM in Figure 5.25. The quadratic deformation of the wing surface is clearly apparent in the figure. For this ROM, it was found that $(M_T - M)_{min} = 0.0366$.

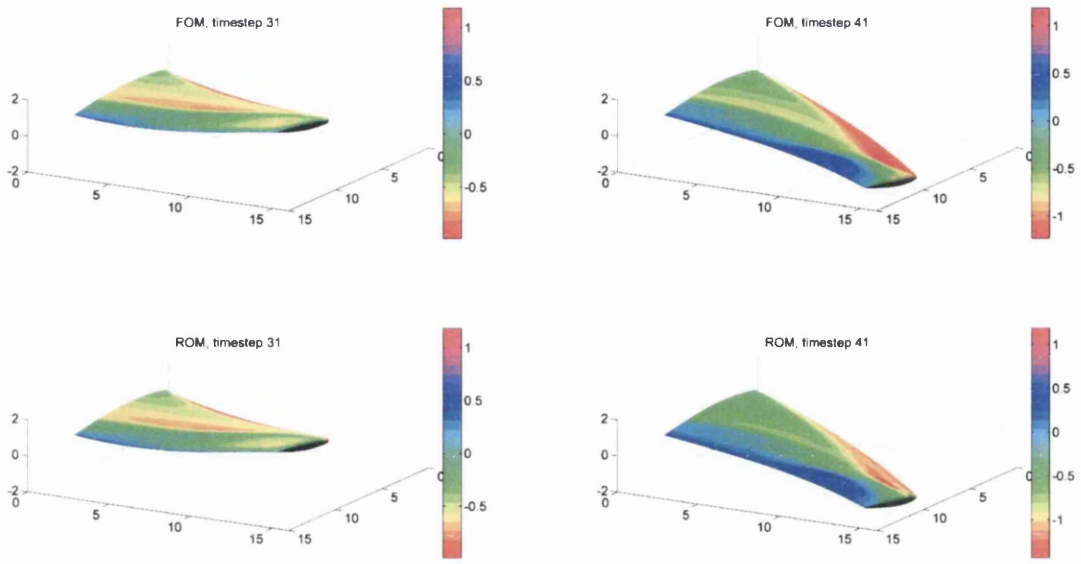


FIGURE 5.25: The pressure coefficients on the upper surface of the wing for Example 4 computed at the point $\boldsymbol{\alpha} = (8^o, 1.6)$ in the parameter space with the FOM and the 25 snapshot ROM at two different times

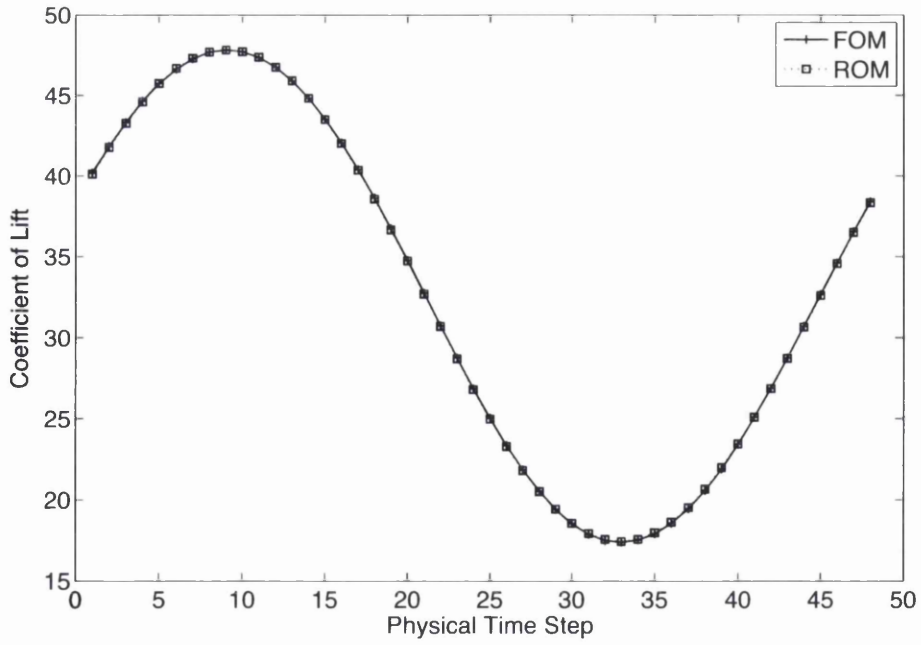
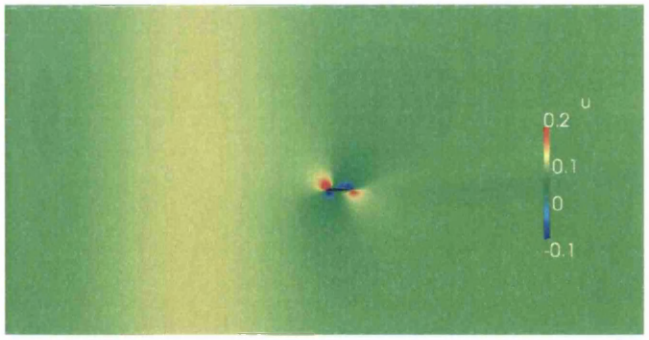
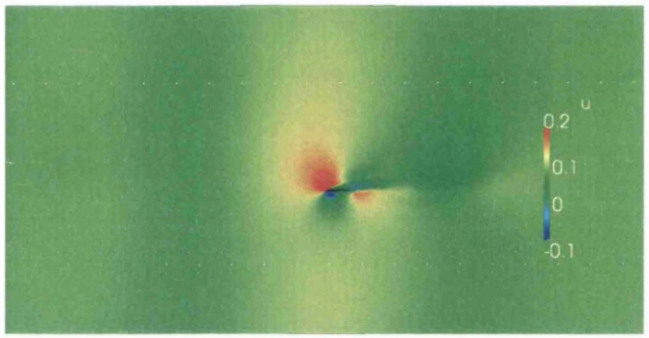


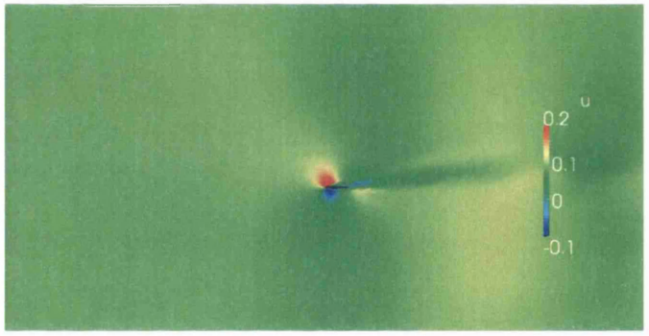
FIGURE 5.26: Comparison of the variation in the lift coefficient computed with the FOM with that obtained from the 25 snapshot ROM for Example 4 at the point $\boldsymbol{\alpha} = (8^o, 1.6)$ in the parameter space.



(a) Initial dimensionless y-velocity showing the sharp-edge gust downstream of the NACA 0012.



(b) The y-velocity after 12 timesteps, as the gust crosses the aerofoil.



(c) The y-velocity after 26 timesteps, after the gust has passed over the aerofoil.

FIGURE 5.27: Evolution of a sharp-edge gust past the NACA 0012.

Sharp Edged Gust past a NACA 0012 In this example compressible, inviscid flow at free stream Mach number 0.25 past a NACA0012 aerofoil was considered. A sharp-edge gust [124] was applied upstream of the aerofoil, by adding a constant magnitude of 0.1 to the velocity in the y-direction in an artificial box over a prescribed gust length in the x-direction. This velocity was added in the initial timestep only, then the freestream flow moved the gust towards and past the aerofoil. To avoid introducing shocks, the magnitude was decreased linearly in the y-direction from 0.1 at the edge of the box to zero at the north and south farfield boundaries. Figure 5.27 shows an example of the evolution of such a gust, here the velocity in the y-direction is plotted.

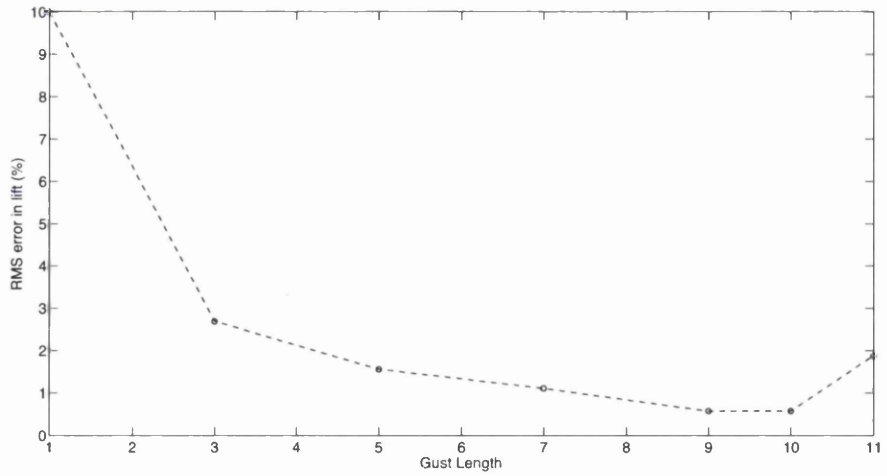


FIGURE 5.28: Changing error of the ROM with gust length, for the gust example

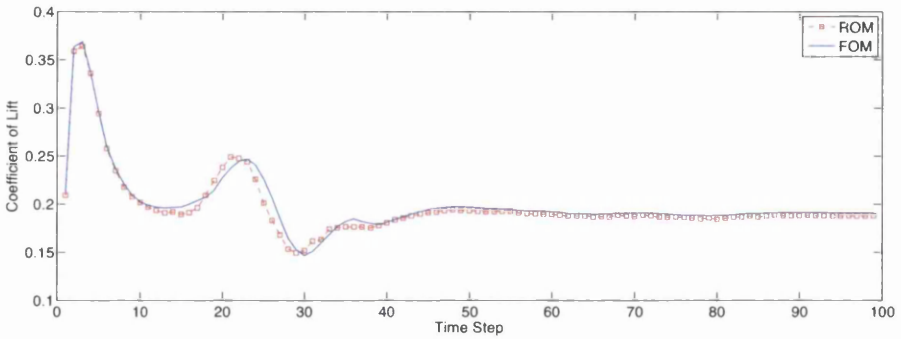


FIGURE 5.29: Comparing the lift of the FOM to ROM for the worst case in the gust example

Gust modeling itself is an interesting area of research, where a number of different modelling techniques exist. The interested reader is directed to [124] for more details, which are beyond the scope of this thesis.

The parameter space to be sampled was the gust length, measured in chords of the aerofoil, and 7 snapshots were taken from 1 chord length to 11 chord lengths. Each simulation was run for 100 timesteps on FLITE, which required a CPU cost of over 24 hours. The percentage error in lift was calculated at each snapshot, in the fashion described above, and the error calculated is plotted against gust length in Figure 5.28. The error was high at the extremities of gust length since these points represent extrapolations when removed from the snapshot set. Not counting these points, the highest error in lift was 2.6927%, at a gust length of 3 chord lengths. Figure 5.29 compares the time histories of lift of the ROM to the FOM for this case. The CPU cost to evaluate

this ROM, on an Intel Core i5-2500 running at 3.30GHz, was 15 seconds.

5.6 Applying POD Interpolation to Optimisation

A pilot study applying these POD interpolation type methods to the shape optimisation example presented previously has been performed and results are presented here. The inverse design example shown in Section 4.3.1 was performed at three values of the free-stream Mach number, viz. 0.25, 0.5 and 0.75. Instead of the evaluating the objective using the FOM, a ROM was constructed in the way discussed in Section 5.4.2 using RBFs to interpolate the POD mode coefficients.

The number of snapshots, generated using LHS, was varied and the optimisation was repeated 10 times for each different number of snapshots. Figure 5.30 shows how the percentage difference between the target RAE2822 and the best design surface pressure decreases as the number of snapshots increases. A limitation of the technique was highlighted when comparing the results for the different Mach numbers. As the flow regime become transonic, the percentage difference increased. This was because the ROM failed to predict the movement of shocks, unlike the FOM. This was deemed a severe limitation of the POD technique and limits its application to shape optimisation. It may be possible to improve the technique with adaptive sampling, by increasing the number of snapshots in areas of the parameter space where the flow regime changes. An alternative technique, which has been shown to perform well in wave scattering problems, proper generalised decomposition [125], may be more suited to these types of problems.

A point worth noting, becomes apparent when the pressure field is the unknown being interpolated. Interpolating the pressure, and using these interpolated values to calculate lift and drag coefficients is equivalent to interpolating the lift and drag values themselves. This is because the lift and drag coefficients are simply a linear sum of surface pressures. It follows that, in the case where the objective function is made up of lift and drag only, it is more efficient to simply interpolate these values alone.

Without further development and investigation, the problems identified limit the application of the POD interpolation techniques to predict flow physics for optimisation algorithms. Some applications do not require such significant changes in flow physics, and require more information than just lift and drag coefficients. For such applications, POD interpolation techniques offer a good solution to reduce the cost of the objective function.

However the POD formulation can be used alone to reduce the dimension of optimisation problems which do not involve prediction. Such an example is presented in Chapter 6.

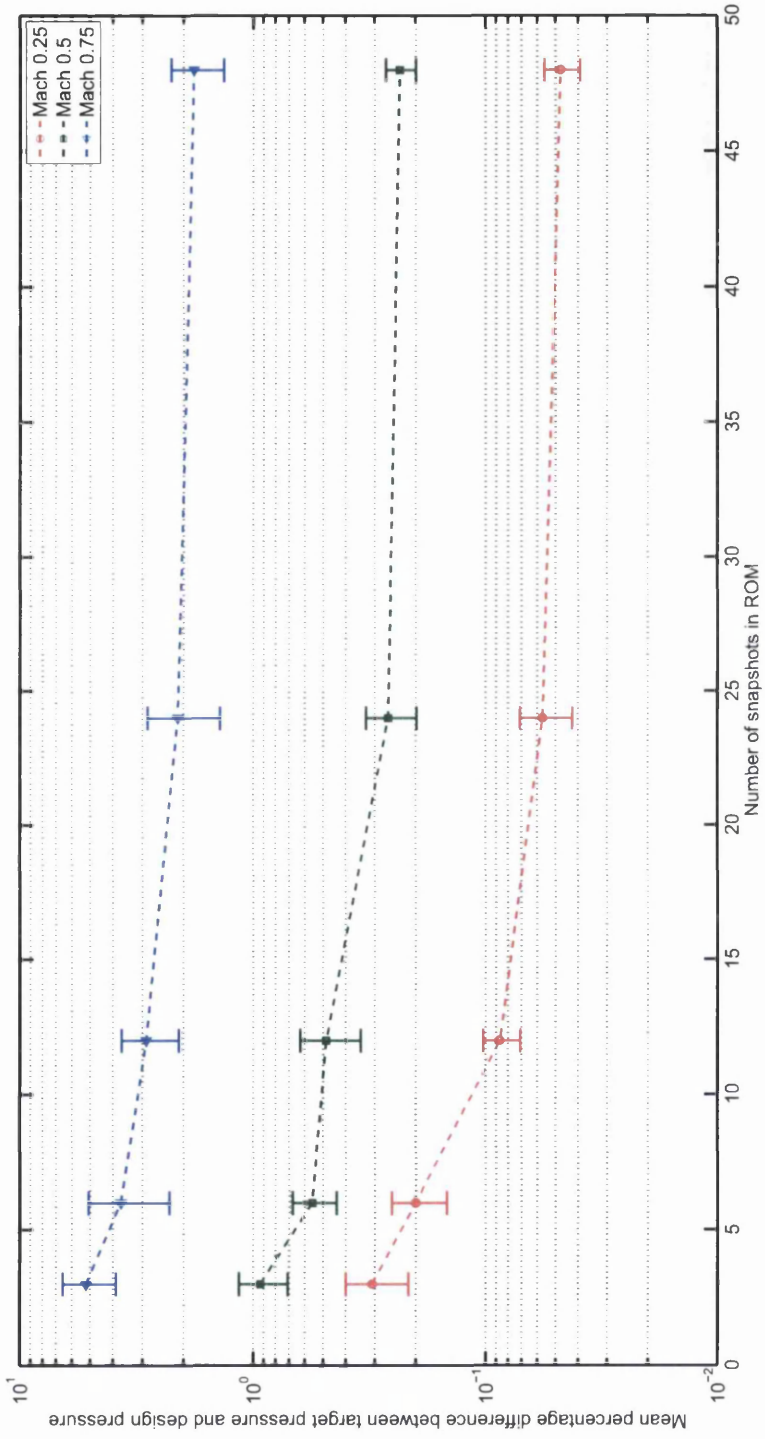


FIGURE 5.30: Using a POD interpolation ROM to approximate the objective function for the inverse design example.

5.7 Conclusion

The aim of this chapter was to attempt to address the problem of the large costs that are normally encountered when computational methods for unsteady flow are employed as part of a design process. The ROM techniques that have been proposed acted upon the computational domain and require no knowledge of the system or of the solver used to generate the snapshots. The methods were completely general and could be applied, without modification, to general time-dependent or time-independent problems. For the examples presented, the CPU cost of the ROM was several orders of magnitude lower than that for the FOM. Convergence of the error, as the number of snapshots increases, has been shown and this increased confidence in the usefulness of the technique. The nature of the method means that the CPU cost of the ROM will almost always be much lower than that for the FOM. But, it cannot be assumed that the loss of solution accuracy will be small in every case. However, generating the snapshots necessary for building the ROM would already be part of a normal design process. The methods could seamlessly fit into a parameter sampling cycle, starting with a coarse sampling of snapshots and iteratively validating the resulting ROM each time the sampling is refined. If an acceptable error in the ROM was reached, significant saving could be achieved by employing the ROM for the remaining parameter sweep.

Chapter 6

Reduced Order Mesh Optimisation

6.1 Introduction

Traditionally ROM techniques are applied to find the solutions of differential equations. However, these techniques can be applied wherever it is useful to decrease the number of degrees of freedom in a system. With this in mind, POD was applied as a dimension reduction technique to enable the use of MCS in the high dimensional problem of mesh optimisation [3]. Specifically, the problem of interest was the optimisation of an unstructured Delaunay primal mesh, and its orthogonal Voronoi dual, for use with co-volume solution algorithms. In this chapter, the technique is presented and a number of test cases are included to demonstrate its ability.

Ensuring the quality of the mesh employed is essential for a successful numerical simulation of a physical problem. This is of a particular importance when considering unstructured mesh implementations of co–volume schemes, such as the marker and cell (MAC) algorithm for the solution of the Navier Stokes equations [126] or the Yee algorithm for the solution of Maxwell's equations [127]. Co–volume methods are classically staggered in time and implemented on a pair of staggered orthogonal meshes in space, resulting in a discretisation that is second order accurate in space and time on uniform meshes [128]. Such algorithms exhibit a high degree of computational efficiency, in terms of a low operation count and low storage requirements. However, a successful implementation depends critically upon the generation of the pair of high quality mutually orthogonal meshes for the problem [129]. For simulations involving complex geometries, generating such a mesh can cause great difficulties, as current standard unstructured

mesh generation and enhancement techniques normally only focus on the quality of the primal triangulation.

The problem of constructing a good quality dual orthogonal mesh for complex geometries can be approached by developing a stitching technique [130]. In this method, a local high quality triangulation of the region near the boundaries is stitched to an ideal equilateral triangular mesh covering the remainder of the computational domain. The local triangulation is constructed using a modification of the advancing front technique [131] in which, after splitting the boundary curve into edges, nodes are placed to enable the construction of a layer of near ideal elements adjacent to the boundary. A number of layers are generated in this fashion, before stitching to the ideal mesh is attempted. Although this technique proves effective at producing dual orthogonal meshes in two dimensions, the approach has not been successfully extended to general three dimensional problems.

In the work presented in this chapter, a Delaunay mesh generation technique, with automatic node creation, was employed to generate primal unstructured meshes in both two and three dimensions [83]. This method starts with a set of the nodes defining the boundaries. Once a Delaunay triangulation of these boundary nodes is constructed, nodes are inserted sequentially at the centroids of the existing elements and a new Delaunay mesh of the nodes is produced at each stage. The process continues until the node density in the mesh meets the requirements specified by a user–defined mesh control function [132]. For the implementation of co–volume solution algorithms, the Voronoi diagram provides the natural choice for the dual orthogonal mesh. In this chapter, the standard convention of referring to the points in the Delaunay triangulation as nodes, and the Voronoi points as vertices is used.

The motivation of this work was to take these automatically generated meshes as a starting point and to attempt to improve them, using appropriate optimisation techniques, to make them suitable for use with co–volume solution algorithms. Co–volume solution algorithms can be one to two orders of magnitude faster than standard finite element techniques [128] in terms of CPU time. Considering this, the CPU cost of any mesh optimisation procedure was considered less of an issue. The primary motivating factor in all of this work was to attempt to make a mesh suitable for co–volume techniques. It was assumed that once a suitable mesh is generated, the subsequent saving in time due to the efficiency of co–volume techniques would make the increase in mesh preprocessing time become negligible.

In the present context, MCS [1] was utilised as an optimisation strategy, because the objective function, which represents a measure of the element quality in a mesh, could become non–smooth [133]. In the present application, the objective function was the

weighted sum of a quality measure over all elements. The weight was given a value of either zero or one for each element. An objective function, constructed in this way, is highly likely to contain discontinuities due to the jump of the weights, motivating the use of gradient free techniques.

When accurate simulations are being attempted, the number of nodes in a mesh can quickly reach the order of millions. In mesh optimisation, the dimensionality of the global optimisation problem is equivalent to the number of nodes multiplied by the number of dimensions. Most gradient free algorithms are only tested on objective functions with up to 300 dimensions [134]. Clearly, this presents a significant problem to be overcome when considering the application of MCS to mesh optimisation.

In one approach, PSO was applied to hexahedral mesh smoothing [135]. This is shown to be a promising technique, but there are significant issues created by the large numbers of degrees of freedom associated with the meshes typically employed for industrial simulations. To reduce the number of degrees of freedom, a two part process of dividing the domain into various sub–domains, and the exploitation of any available symmetry, is advocated. By dividing the domain, the problem becomes more localised.

In the work presented here, the dimensinality of the problem was reduced by applying POD as a dimension reduction technique [96, 97]. The technique was used to reduce the number of degrees of freedom in the mesh optimisation problem from millions to hundreds. This made the use of global gradient free optimisation techniques possible, without the need to divide the domain.

The approach proposed uses a combination of a gradient free optimisation method, the MCS, and a dimension reduction technique, POD, to tailor an unstructured mesh for use with a co–volume solution algorithm. The targeted mesh requirements are outlined first and then possible alternative techniques for solving the problem are discussed. The proposed approach is explained and its effectiveness was measured by applying it to a number of examples in two and three dimensions.

6.2 Methodology

6.2.1 Mesh requirements for co-volume techniques

Co–volume methods are implemented on a pair of mutually orthogonal meshes. For an unstructured mesh implementation, the most obvious mesh choice is to use a primal Delaunay mesh and its Voronoi dual [130]. To ensure second order accuracy of the solution algorithm, each Delaunay edge and its corresponding Voronoi edge must be mutually

perpendicular and mutually bisecting. In addition, if a Voronoi vertex lies outside the corresponding Delaunay element, any integral over the element would be approximated in terms of values from outside the element. This results in a loss of accuracy [128]. Elements which do not include their corresponding Voronoi vertex are referred to as bad elements and the goal of the optimisation technique, that will be presented, was to remove the bad elements from a mesh generated using standard techniques.

The bad elements were removed at the expense of a loss of local second order accuracy, caused by the relaxation of the condition that the Delaunay and Voronoi edges should be mutually bisecting. This local loss of accuracy is not seen globally as illustrated in the examples presented in [128]. The goal of removing all bad elements is different from other mesh optimisation techniques, which commonly seek to improve the poorest elements in a mesh (such as in [136]). Attempting to improve the poorest elements in a mesh is a problem which can be expressed locally, whereas removing all bad elements is a global problem and should be treated as such. Many experiments have shown that, after a series of local optimisation operations, some poor quality elements will still remain in a mesh [137], which further motivated the use of a global optimisation technique.

6.2.2 Mesh smoothing for co–volume techniques

6.2.2.1 Existing smoothing algorithms

Laplacian smoothing is a widely used approach for improving mesh quality. In its simplest form, each node is moved towards the mean position of its neighbouring nodes [138]. This has the effect of increasing the regularity of the mesh. This may be of benefit in two dimensions, where the ideal mesh for co–volume techniques is equilateral triangles [130], but perhaps not in three dimensions, where the ideal mesh consists of non-perfect tetrahedra [139–141].

Another algorithm, which initially appears to be well–suited to the problem of interest here, is Lloyd's algorithm [137, 142]. This is a CVT scheme that begins with an initial mesh, then moves the generated nodes to the mass centroids of their corresponding Voronoi cells. The iteration process continues until all nodes are sufficiently close to their corresponding centroids. Lloyd's algorithm has already been used in an attempt to improve mesh quality for use with co–volume solution techniques. However, it was discovered that, for general meshes, around 10% of the elements are still not of sufficient quality at the end of the process, due to boundary constraints [128].

An alternative iterative algorithm was designed to eliminate all bad elements in a two dimensional triangular mesh, while retaining the connectivity, by moving the interior

nodes to reduce a global energy function [143]. It is found that, in a situation where a node has fewer than five neighbours, it is not possible to repair all the bad elements. A triangle subdivision technique was introduced which split elements around such nodes into a number of smaller elements until no bad elements remained. It is advantageous to avoid, if possible, the requirement for such a subdivision as, in some practical applications, the use of a uniform element size is desirable [130].

The failure of these smoothing techniques for the current application is due to two reasons; the smoothing acts locally both in terms of position in the mesh and in terms of a local sub–optimum, and the improvement is not specifically targeted at removing bad elements.

6.2.2.2 Optimising nodal coordinates

The number of degrees of freedom, representing the coordinates in a mesh, is equal to the number of nodes N (not including the boundary nodes, which are fixed) multiplied by the number of dimensions D. When N is large, it is difficult to optimise this high dimensional problem. The approach adopted was to split this global $N * D$–dimensional problem into N local D–dimensional problems, by considering each node in turn. Since $D \leq 3$, these local optimisations were computationally cheap when applying the MCS.

To calculate the fitness of a single node, the function

$$F(k) = \sum_{i=1}^{E_k} W_i \frac{\|\boldsymbol{C}_i - \boldsymbol{V}_i\|}{\delta_i} \tag{6.1}$$

was used in the optimisation process. In this expression, k is the node number, E_k is the number of elements which include node k, the index i points to the elements including node k, $\boldsymbol{C}_i$ is the position vector of the centroid of element i, $\boldsymbol{V}_i$ is the position vector of the Voronoi vertex of i and δ_i is the mean edge length of element i. The quantity W_i is equal to zero if V_i is inside element i and equal to one otherwise. For the optimisation process, the argument of the objective function was a vector, $\mathbf{x}$, which was added to the coordinates of k. After this addition had taken place, equation (6.1) was evaluated at the new coordinates of k and a fitness returned to the optimiser. When an optimum value of $\mathbf{x}$ was found, the coordinates of k were updated as

$$\mathbf{x}_k = \mathbf{x}_k + \lambda \mathbf{x} \tag{6.2}$$

where $0 < \lambda \leq 1$ is a relaxation parameter that was used to slow down the movement of the nodes. The objective function of equation (6.1) is plotted for some sample

nodes in Figure 6.1. The plots show the objective function was clearly non–smooth and multimodal highlighting the need for a gradient free approach.

Each local optimisation was run for 5 generations using the MCS. It was found that 5 generations provided a good balance between CPU cost and the improvement achieved by the optimisation. All of the algorithms developed as part of this work were heuristic in nature (see the work by Klinger and Shewcuck [144] for further discussion on this point). There were a number of parameters which will effect the optimisation process. For example, it may have been possible to optimise the value of λ, but this turns out to be very problem specific. In practice, any algorithm should be automatic without the need for significant user input, so it was deemed best to pick a value of λ which works well on a variety of examples and to keep it constant between examples. If λ is set too high, the smoothing is too aggressive and the mesh can become tangled. This is of particular importance in 3D examples. This is illustrated in Figure 6.2, where the optimisation procedure was carried out on the internal mesh of a sphere with different values of λ. It was found, through trial and error, that setting $\lambda = 0.25$ produced the best performance in both 2D and 3D for a variety of cases and this was the value used in all examples presented.

Once the coordinates of k were updated, the next node to be optimised was selected at random and the process repeated until all nodes had been treated. After one complete sweep of the mesh, a new triangulation of the node set, constrained by the boundary connectivity, was generated. This eliminated problems, reported in [143], which can be created when nodes have insufficient neighbours. Adjusting the positions of the nodes along the boundary would clearly improve the performance of the algorithm, by adding more degrees of freedom. However, in practice, this is undesirable. The boundary nodes define the geometry of interest in an example. In three dimensional examples, in particular, it is undesirable to carry the geometric definitions, which would be required to move nodes along the boundary, throughout the meshing process. Since the aim was to make the technique as general as possible, the boundary node positions were maintained.

6.2.2.3 Comparison of local node optimisation techniques

To justify the use of MCS, a comparison was performed between a number of node optimisation techniques. In all cases, the techniques were applied to an unstructured mesh of triangles, which has been generated in the region around a NACA0012 aerofoil. The mesh contains a total of 32460 triangular elements. To compare performance of the different techniques, the number of bad elements in the mesh was plotted against the

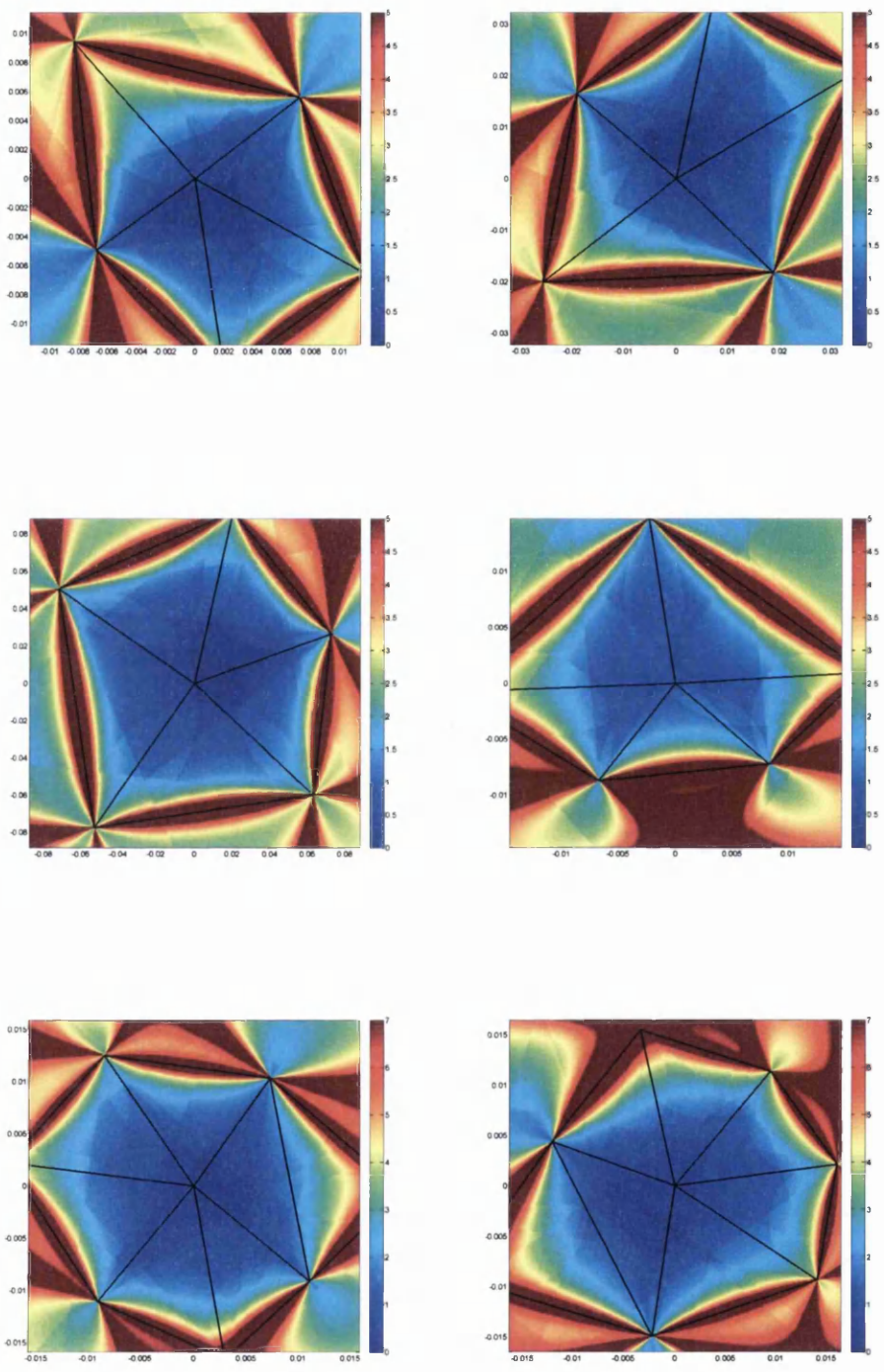


FIGURE 6.1: Dual orthogonal mesh objective function plots for sample nodes. Each plot shows a single sample node from a 2D mesh along with its attached elements. The colour map shows the value of the objective function (6.1) if the node was moved to that position in space.

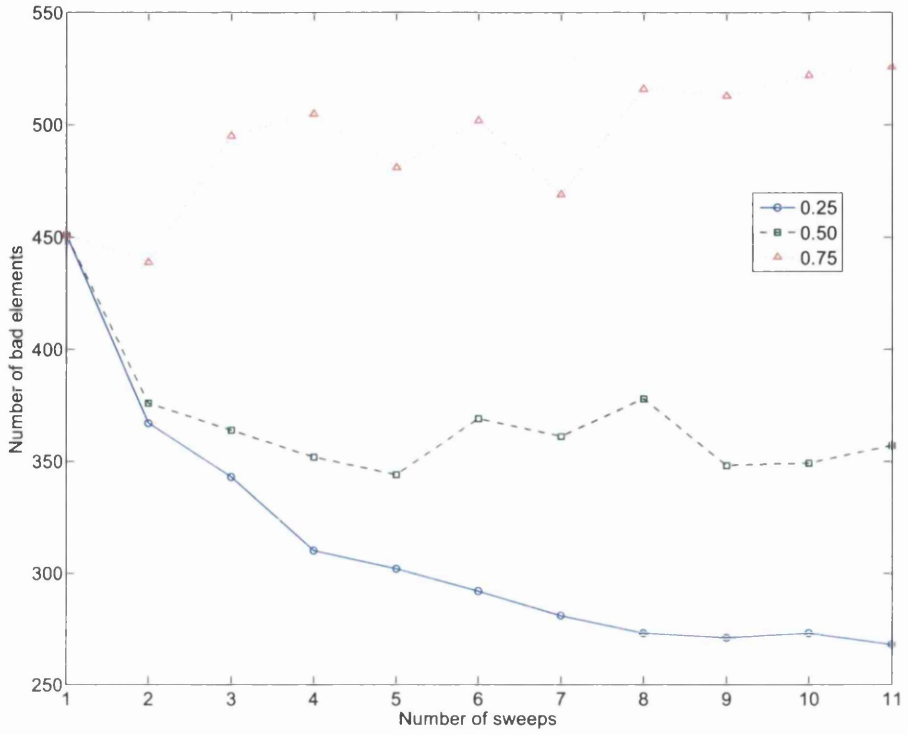


FIGURE 6.2: Local nodal mesh optimisation applied to a mesh of the region inside a sphere with various values of λ

CPU time in seconds. The CPU times quoted in this chapter all refer to the use of a 3.30GHz Intel Core i5 processor, unless stated otherwise.

Figure 6.3 compares the performance of Laplacian smoothing and Lloyd's algorithm to the MCS smoothing technique described above. In the first 10 seconds, Lloyd's algorithm and Laplacian smoothing outperformed MCS. However, over time, MCS significantly outperformed both the other algorithms which started to level off. This showed the benefit of constructing an objective function targeted at the mesh quality which needs to be improved.

To show that MCS was the best choice of optimiser, MCS was replaced by two other optimisation techniques applied to the same objective function, in the same way, as described above. The implementations `fminunc` and `fminsearch` from the MATLAB [17] optimisation toolbox were used, to replace MCS with gradient based and simplex gradient free optimisers respectively. The simplex based optimisation is the Nelder-Mead method [25] and is similar to the technique used by Freitag and Plassmann [145] for mesh untangling. Figure 6.4 shows how MCS outperformed both techniques, in terms of fast reduction of bad elements at low CPU cost, and in long term maximum reduction of bad

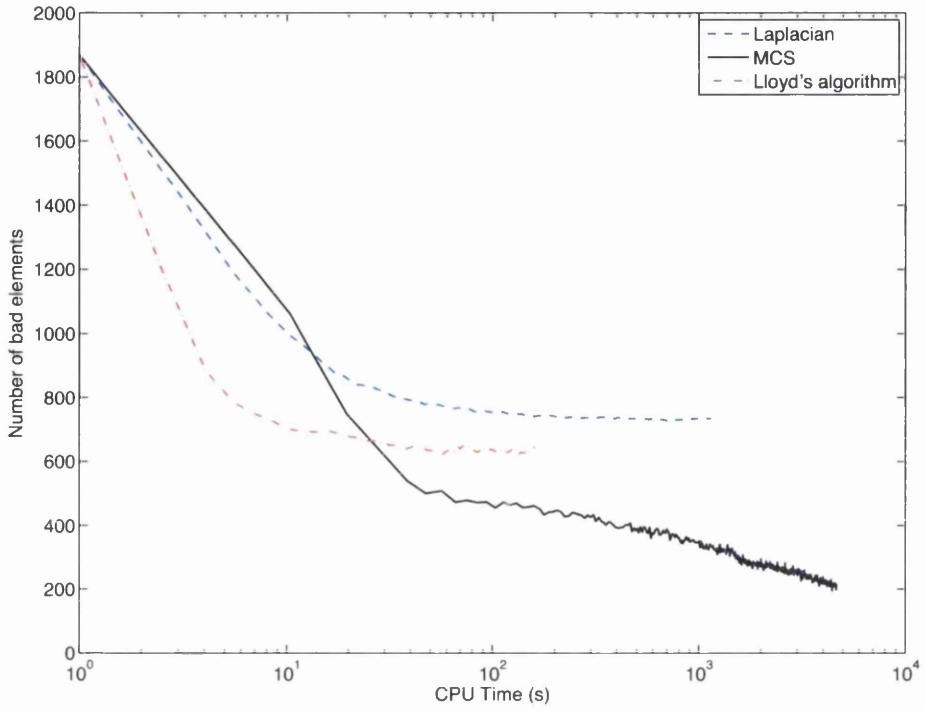


FIGURE 6.3: Comparing the performance of Laplacian smoothing and Lloyd's algorithm to the local MCS node optimisation

elements. This result further supported the previous findings regarding the behaviour of MCS [1]. Despite the strong performance this local node optimisation was unable to fix all the bad elements. To address this, a method of globally optimising the position of the Voronoi vertices was introduced.

6.2.3 The weighted Voronoi power diagram

An alternative method for eliminating bad elements is to move the Voronoi vertices, while retaining orthogonality, by using the weighted Voronoi power diagram. To allow for a greater level of flexibility, this method relaxes the requirement that the Voronoi edges and their corresponding Delaunay edges should be mutually intersecting. The requirement that these edges should be mutually perpendicular is retained. Figure 6.5(a) illustrates a Voronoi vertex O located at the circumcenter of a bad element, ABC. This vertex can be located at the intersection of the common chords of the circles, each with the same radius as the circumcircle of ABC, centred at each of the vertices A, B and C. It can be observed that the common chords are perpendicular bisectors of their associated Delaunay edges. In Figure 6.5(b), the radius of the circle centred at B has been reduced and, using the modified common chords, the Voronoi vertex is pulled inside the element,

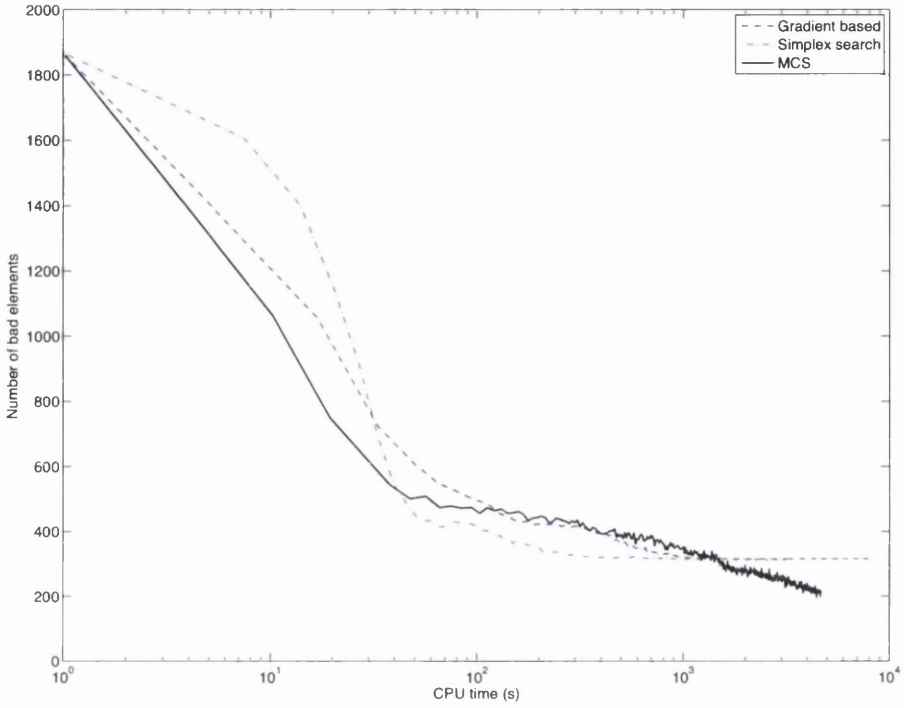
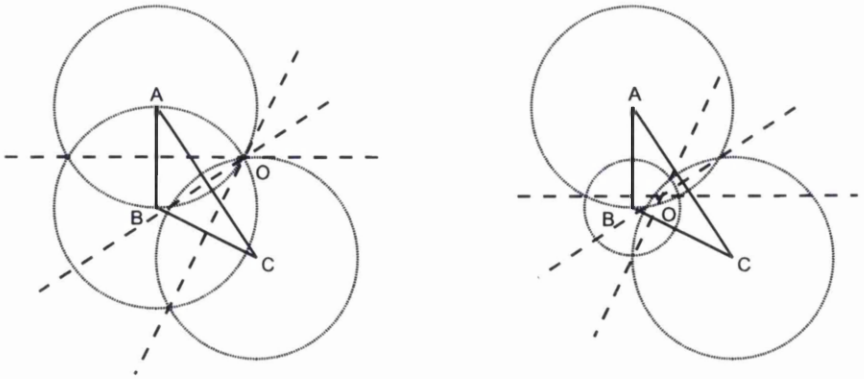


FIGURE 6.4: Comparing the performance of replacing MCS with a gradient based optimiser and gradient-free simplex search method



(a) The bad element ABC and corresponding Voronoi vertex O

(b) Effect of applying a negative weight to node B

FIGURE 6.5: Moving a Voronoi vertex to repair a bad element

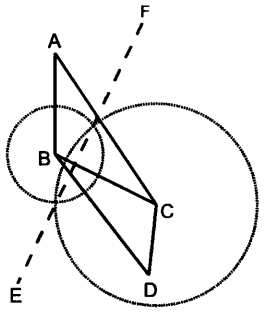


FIGURE 6.6: Illustration of the constraints on the weight optimisation process

while the chords are still perpendicular to the edges. Given the change in the radius δr_K, at each node K, the coordinates of O can be readily obtained by solving the equations

$$(x_1^O - x_1^A)^2 + (x_2^O - x_2^A)^2 - \delta r_A^2 = (x_1^O - x_1^B)^2 + (x_2^O - x_2^B)^2 - \delta r_B^2 \tag{6.3}$$

$$(x_1^O - x_1^A)^2 + (x_2^O - x_2^A)^2 - \delta r_A^2 = (x_1^O - x_1^C)^2 + (x_2^O - x_2^C)^2 - \delta r_C^2 \tag{6.4}$$

where a two dimensional Cartesian (x_1, x_2) coordinate system has been adopted. An identical process can also be applied to tetrahedral elements in three dimensions [128].

6.2.3.1 Optimising node weights

Considering again the configuration illustrated in Figures 6.5(a) and 6.5(b), each node was assigned an associated weight, which is equal to the change in radius of the circle, used to define the Voronoi vertex, centred at that node. It is possible to improve the quality of the dual mesh, by optimising the position of the Voronoi vertices, and this may be achieved by considering the weight of each node when constructing the dual. The challenge was to optimise these weights, such that the number of bad elements globally was reduced.

If a bad element is considered independently of all other elements, it is a relatively straightforward three dimensional optimisation problem to minimise the distance between the Voronoi vertex of the element and its centroid. Figure 6.5(b) provides a good example of this. However, difficulties occur when considering the elements connected to this initially bad element. It is possible that the optimisation may turn another, initially good element, into a bad element, such as the element shown in Figure 6.6. The weights were optimised in a similar fashion to which the coordinates were optimised. Each node

was considered in turn, in a random order. If the value of the objective function at a node was greater than zero, an optimisation was performed. This procedure can be thought of as a local optimisation of the Voronoi vertices, using the node weights. Each weight was optimised using MCS applied to the objective function of equation (6.1).

A similar local weight optimisation, using gradient based techniques in place of MCS, is applied to a mesh generated for the problem of scattering of an electromagnetic wave, by a perfect electrical conductor with two pairs of rear fins [128]. A detailed view of the discretised surface and of the discretisation on a cut through the volume is shown in Figure 6.7, along with the solution field calculated using a co-volume technique. For the mesh shown in Figure 6.7, the percentage of bad elements was reduced from 16% to 4.8% using the local optimisation.

The results of the co-volume technique are compared to an explicit nodal low order finite element time domain scheme [146, 147], and a multi-level fast multipole method, in Figure 6.8. The distribution obtained from the co-volume scheme is in excellent agreement with the fast multipole results [128].

Local optimisation is unable to fix all the bad elements in a general mesh. However, in the work presented here, it is shown that the combination of MCS with reduced order modelling techniques proved much more effective than local optimisation.

6.2.4 Reduced order optimisation

6.2.4.1 Proper orthogonal decomposition

In the other applications considered in this thesis, model reduction was used to reduce the CPU time required to calculate solutions. In the present application, the major motivation for using POD was to enable the formulation of a global optimisation problem with a low number of degrees of freedom. This then enabled the use of MCS. The method of snapshots was used to construct a POD basis [96, 97].

The snapshots, $\mathbf{y}^i; i = 1, \ldots, M$, represent different configurations of weights throughout the mesh of interest. Each snapshot is a vector, containing N degrees of freedom and, if an $N \times M$ matrix whose columns are the snapshot vectors is constructed, the POD modes, $\boldsymbol{\phi}_j; j = 1, \ldots, M$, can be calculated by applying a SVD to this matrix. The POD modes are the left singular vectors of the snapshot matrix [96, 97]. Each of the snapshots used to generate the POD modes can be fully reconstructed, by a linear combination of these POD modes, as

$$\mathbf{y}^i = \sum_{j=1}^{M} a_j^i \boldsymbol{\phi}_j \tag{6.5}$$

FIGURE 6.7: Scattering by a PEC fin/body configuration: view of the computed distribution of the E_3 component of the scattered electric field on the surface of the body and an a cut through the volume mesh [128]

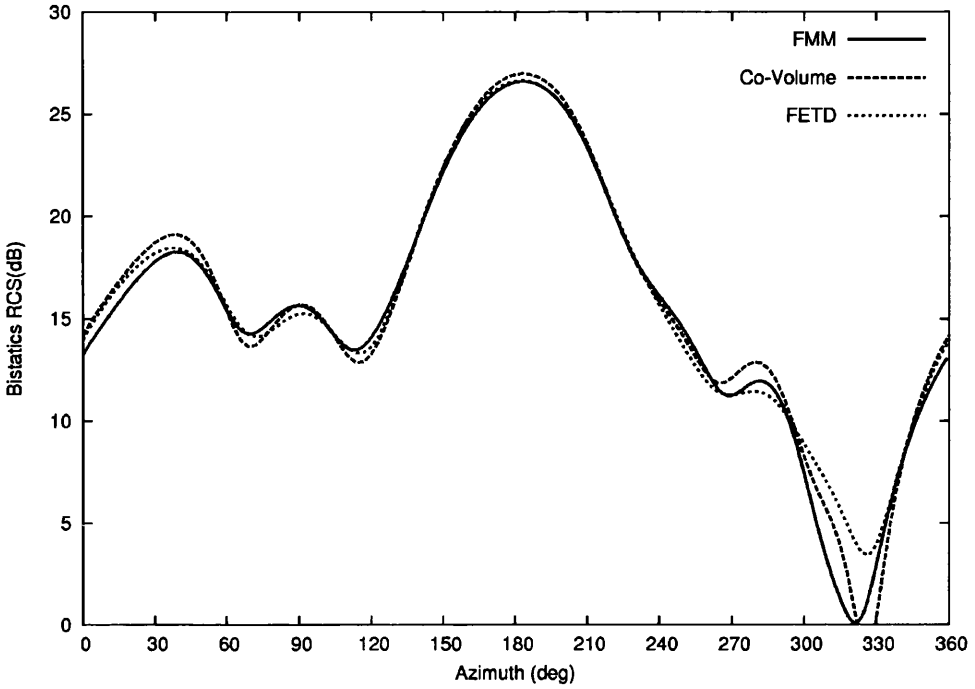


FIGURE 6.8: Scattering by a PEC fin/body configuration of electric length 6λ: comparison between the computed RCS distributions and the distribution obtained from a fast multipole method in the plane $\phi = \pi/2$ [128]

and the coefficient sets calculated, using the dot product, as

$$a_j^i = (\boldsymbol{\phi_j}, \mathbf{y}^i) \qquad \text{for } i = 1, \ldots, M \tag{6.6}$$

since $\boldsymbol{\phi_j} \perp \boldsymbol{\phi_{i \neq j}}$. This allows the different configurations of weights to be represented by M degrees of freedom and, if the number of snapshots is selected such that $M \ll N$, the number of degrees of freedom will have been successfully reduced. By picking new sets of coefficients, a_j^i, which do not belong to the snapshot set used to construct the basis, new configurations of weights can be reconstructed. Essentially, POD provides the ability to search between, and around, the original set of snapshots in a low dimensional space. Since the mesh domain is not divided into subsections, any objective function based on a POD representation is truly global. In essence, the POD provides a coordinate system on which the optimiser can operate. The coordinate system is generated numerically, using SVD, and is highly dependent on the snapshots used to construct it. This means it would be very difficult to infer an objective function's smoothness, which further motivates the use of MCS.

6.2.4.2 The weight optimisation procedure

As the POD basis only contains information that is present in the snapshots, intelligent selection of snapshots is the key to constructing a successful POD basis [98]. Clearly, the resulting set of weights, calculated from the process described above, depends heavily on the order in which the node weights are optimised. However, testing indicated that the behaviour of the optimisation process was independent of the choice of the initial node. Initially, each node is assigned a weight of zero, which produces the standard Voronoi diagram. This configuration of weights is represented by the vector **M0**. This is transformed into a new configuration **M1** by following the process of optimising node weights described above. Following the same procedure, starting with **M0**, results in a third configuration of weights **M2**. The configurations **M1** and **M2** will be different, if the order of elements treated in the process is different. Since both configurations are a transformation of **M0**, there is the chance that they may be quite similar, as the quality improvement process always starts in areas of **M0** in which there are bad elements. Employing a group of similar snapshots is undesirable, since this will limit the diversity of solutions which can be represented by the POD basis functions. Thus, if the procedure is applied to either the configuration **M1** or **M2** instead of always applying it to **M0**, there is a greater chance of introducing diversity to the snapshot set.

The procedure of generating snapshots, for a mesh with starting weight configuration **M0**, may be described as follows:

1. add **M0** to the list of snapshots;
2. pick a random configuration of weights, **MR**, from the list of snapshots and apply the process outlined in Section 6.2.3.1, resulting in a new configuration, **MR***;
3. add **MR*** to the list of snapshots;
4. if the number of configurations in the list of snapshots is the desired number then stop, otherwise go to step 2.

These solutions can be used as snapshots to construct POD basis functions representing element weights. This allows a reduction in the number of degrees of freedom, from the number of nodes to the number of POD modes. Following this reduction, global optimisation of the weights is feasible, with the POD mode coefficients as the free variables. In this process, the objective function is of the form indicated in equation (6.1), summed over all elements.

6.3 Examples

A number of examples were employed to demonstrate the performance of the proposed method. Unless otherwise stated, all meshes were generated using the method developed by Weatherill and Hassan [83]. The optimisation procedure implemented in MATLAB [17] was as follows; firstly (unless stated otherwise) Lloyd's algorithm was performed upon the mesh, until no reduction in the number if bad elements was achieved. The optimiser then switched to Laplacian smoothing, until no further improvement was seen. Once these algorithms could no longer improve the mesh, four sub–iterations of the node coordinate optimisation were performed, with $\lambda = 0.25$. Following this, 30 snapshots of potential weights were generated as described above. These snapshots were used to generate a POD basis on which MCS was performed with the 15 best snapshots as the starting eggs for 30 generations. The optimiser switched between these two techniques, until all the bad elements were repaired. The results were presented by plotting the number of bad elements against CPU wall time. These were recorded every iteration.

6.3.1 Mesh quality measures in two dimensions

For present purposes, the primary measure of the quality of a mesh was the number of bad elements it contains. A number of alternative mesh quality parameters are now defined, which were used to gauge the other effects that this procedure has on the meshes. All the quality measures considered are for isotropic meshes applied to co–volume techniques, but it is stressed that the number of bad elements is the only quality measure this procedure was designed to improve.

6.3.1.1 Delaunay element quality

An attempt to move a dual Voronoi vertex may result in a decrease in the quality of the primal Delaunay elements. To measure this effect, the Delaunay element quality parameter, q_α, defined by

$$q_\alpha = 3 - \frac{6}{\pi}\alpha_{max} \tag{6.7}$$

was used for each element in the mesh, where α_{max} is the maximum angle in the element being considered. This is a quality measure typically employed when analysing co–volume meshes [130]. For equilateral triangles, $q_\alpha = 1$ and for right angled triangles $q_\alpha = 0$. The ideal mesh for a co–volume technique contains elements with $q_\alpha = 1$. A negative value of q_α indicates that a triangle is obtuse, which implies that the corresponding Voronoi vertex lies outside the element. Thus, an acceptable element has $q_\alpha > 0$.

6.3.1.2 Distance between a Voronoi vertex and a corresponding element centroid, q_β

In an ideal equilateral triangular mesh, the dual Voronoi vertex and the centroid of the corresponding primal element are co–located. For a general mesh, the distance between these two points was computed for each element and normalised by the inner radius of the corresponding element. The result, q_β, was expressed as a percentage and $q_\beta < 100\%$ is acceptable.

6.3.1.3 Voronoi edge length

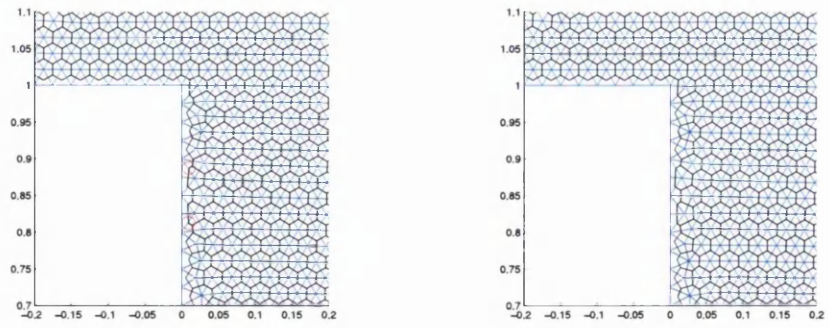
The magnitude of the time step in an explicit co–volume solution algorithm is restricted by the minimum Voronoi edge length [130]. Hence, an important mesh quality measure is the length of each Voronoi edge. The edge length was normalised with the value $\delta/\sqrt{3}$, which corresponds to the Voronoi edge length for a primal mesh of equilateral triangles of side length δ. Expressing the result, q_γ, as a percentage, provides the value for each edge where $q_\gamma > 10\%$ is considered acceptable.

6.3.1.4 Distance between the Delaunay edge midpoint and the Voronoi edge

By changing node weights, the requirement that a dual edge be a bisector of the Delaunay edge has been relaxed. A significant deviation from this property can lead to a loss of solution accuracy when the co–volume discretisation is applied [128]. The magnitude of this deviation was measured, for each Delaunay edge, by calculating the perpendicular distance between the Delaunay edge midpoint and the Voronoi edge. This value, q_δ, was normalised by division by half the Delaunay edge length and expressed as a percentage.

$$q_\delta = \frac{200 \times d_{DV}}{\delta_D} \tag{6.8}$$

where d_{DV} is the perpendicular distance between the Delaunay edge midpoint and the Voronoi edge, and δ_D is the Delaunay edge length. A value of zero indicates bisection, while any value less than 100% indicates that the line defined by the segment of the Voronoi edge intersects the Delaunay edge and is considered acceptable.



(a) Before optimisation

(b) After optimisation

FIGURE 6.9: Detail of the backward facing step example, with red circles indicating Voronoi vertices which lie outside their associated Delaunay element.

6.3.1.5 Distance between the Voronoi edge midpoint and the Delaunay edge

This quality measure q_ϵ is the reverse of the previous measure and was calculated in a similar manner, as

$$q_\epsilon = \frac{200 \times d_{VD}}{\delta_V} \tag{6.9}$$

where d_{VD} is the perpendicular distance between the Voronoi edge midpoint and the Delaunay edge, and δ_V is the Voronoi edge length. A value of zero indicates bisection, while any value less than 100% indicates that the line defined by the segment of the Delaunay edge intersects the Voronoi edge and is considered acceptable.

6.3.2 2D Backward facing step

The first example was the geometry of a simple backward facing step. The generated primal mesh contains 53 427 elements, with mean Delaunay element quality $q_\alpha = 0.9260$, and there are 60 bad elements. This is a small percentage of bad elements, occurring mainly near the boundaries, making optimisation of the mesh quality a relatively straightforward process. Figure 6.9(a) shows part of the mesh before optimisation and Figure 6.9(b) shows the same part after the process has taken place. Voronoi vertices which lie outside their associated Delaunay element are shown circled in red. The mean CPU cost of one iteration was 93 seconds.

All 60 bad elements were repaired by the optimisation scheme. The history and CPU cost are shown in Figure 6.10. Table 6.1 shows the quality measures before and after the optimisation, with both q_α and q_β slightly improved. The minimum value of q_γ was increased by two orders of magnitude, from 0.3901% to 11.14%, resulting in all Voronoi edge lengths being acceptable. The maximum value of q_δ had increased, as expected,

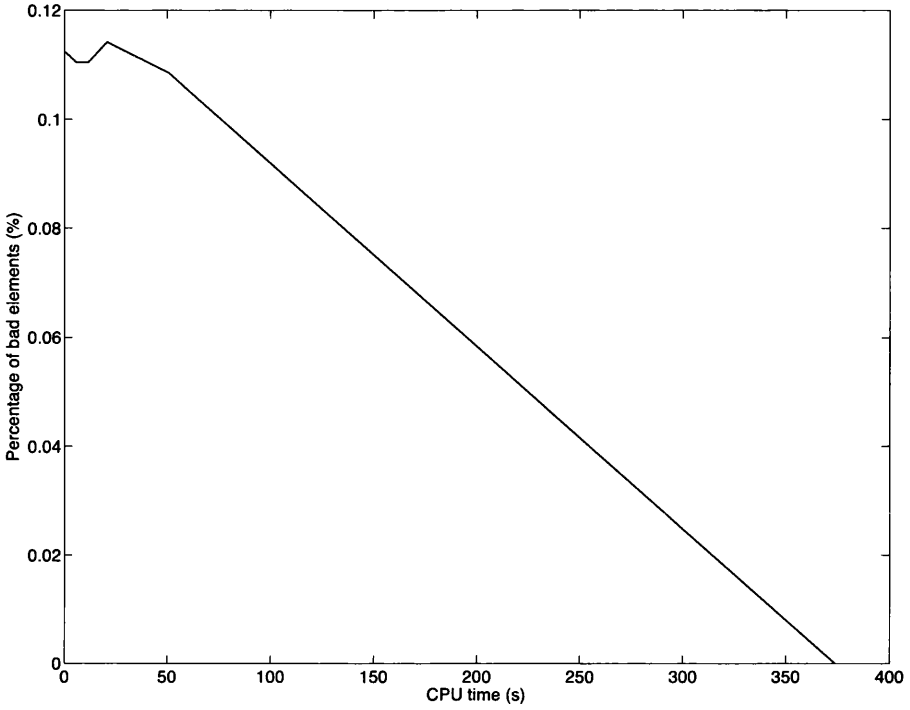


FIGURE 6.10: Optimisation history for backward facing step example

to 48.70% but, since this value was less than 100%, all edges remained acceptable. An improvement could also be seen in the values of q_ϵ, from 99.925% acceptable edges to all the edges being acceptable.

TABLE 6.1: Quality measures for the mesh for the backward facing step.

Delaunay element quality, q_α

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	99.88	0.9260	−0.6272	1.0000
After	99.89	0.9239	−0.4034	1.0000

Distance between Voronoi vertex and centroid, q_β

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	99.94	5.456	0.0015	185.9
After	99.99	5.590	0.0013	101.0

Voronoi edge length, q_γ

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	99.99	100.4	0.3901	301.7
After	100.0	100.4	11.14	344.1

Distance between the Delaunay edge midpoint and the Voronoi edge, q_δ

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	100.00	0.0000	0.0000	0.000
After	100.00	0.0259	0.0000	48.70

Distance between the Voronoi edge midpoint and the Delaunay edge, q_ϵ

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	99.93	3.880	0.0000	15230
After	100.00	3.280	0.0000	100.0

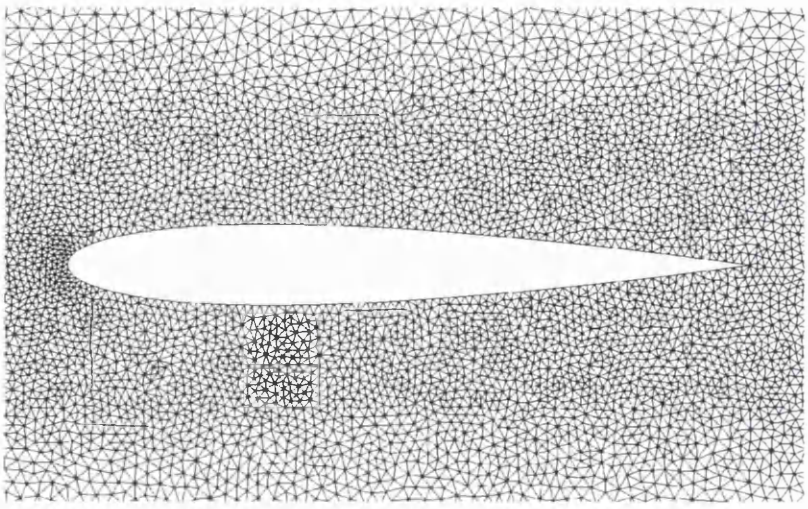


FIGURE 6.11: Mesh, englarged near the aerofoil, for the NACA0012 aerofoil

6.3.3 NACA0012 aerofoil

The second example involved a triangular mesh generated in the region bounded internally by a NACA0012 aerofoil and externally by a circular far–field boundary. A detail of the mesh in the vicinity of the aerofoil is shown in Figure 6.11. The mesh consists of 32 460 elements, with initial mean Delaunay quality $q_\alpha = 0.5278$. However, 1 867, or almost 6%, of these elements are bad. The rate at which the number of bad elements decreases to zero is shown in Figure 6.12. The mean CPU cost of one iteration was 12 seconds. Figure 6.13(a) shows a detail of the mesh, in the vicinity of the trailing edge of the aerofoil, before optimisation and Figure 6.13(b) shows the same area of the mesh after optimisation. The mean Delaunay element quality for the optimised mesh is $q_\alpha = 0.6392$, which means the quality of the elements was improved during the optimisation process. Considering the value of q_α alone, the percentage of acceptable elements was increased by around 4%. The full set of quality measures is presented in Table 6.2. It can be seen that there was an increase, of around 3%, in the percentage of acceptable elements when considering q_β. The minimum value of q_γ was increased by two orders of magnitude from 0.0160% to 3.308% and the number of acceptable Voronoi edge lengths had increased slightly. As in the previous example, the maximum value of q_δ was increased to 77.11%. The number of edges with an acceptable value of q_ϵ was significantly increased from 96.310% to 100%.

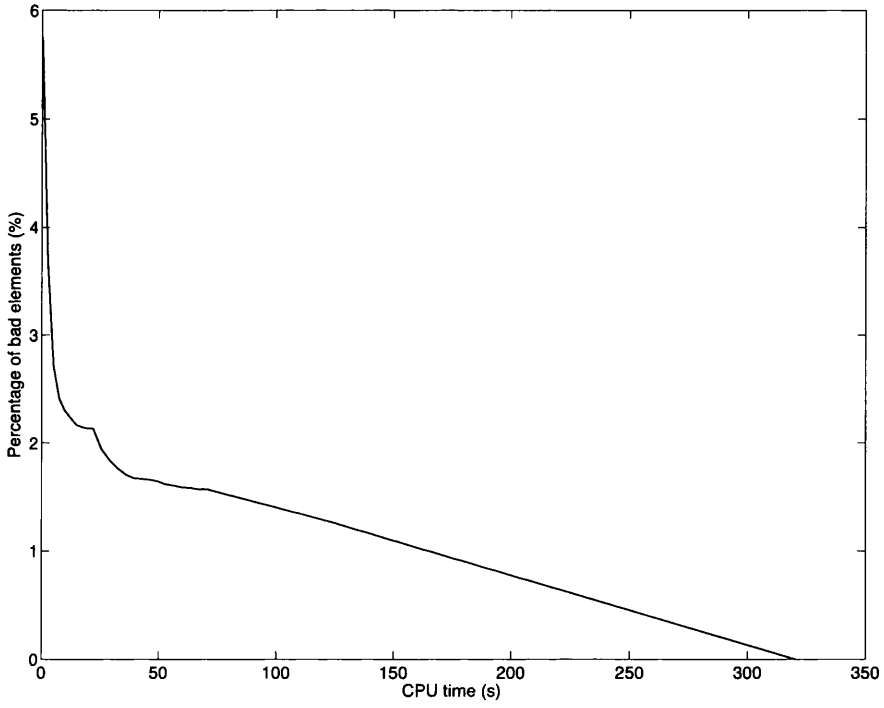
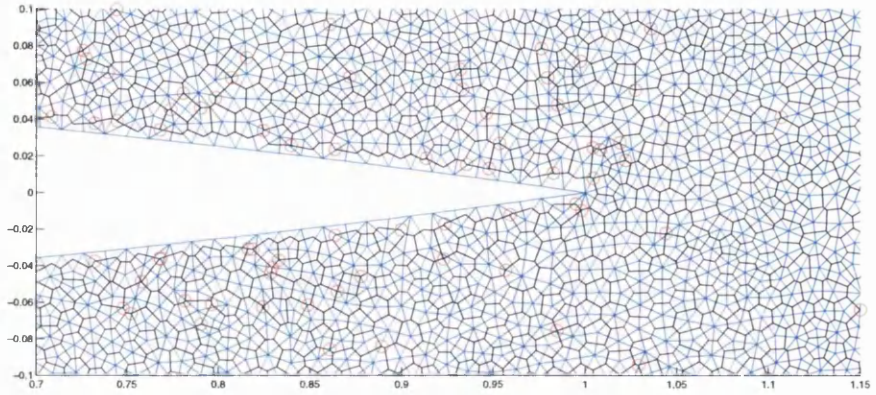
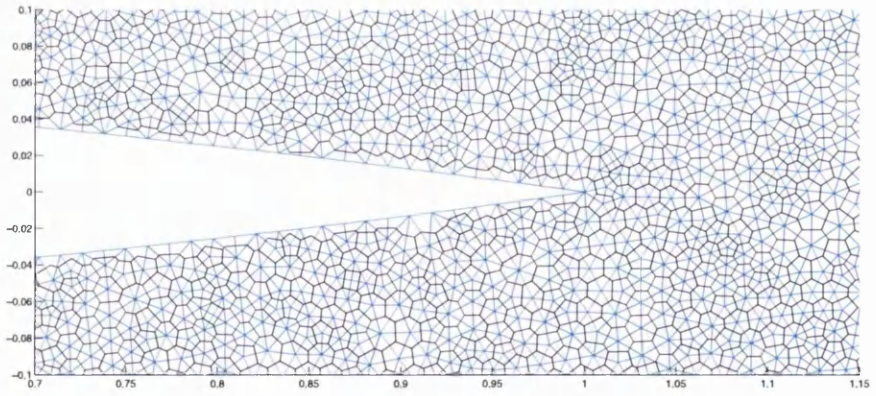


FIGURE 6.12: Optimisation history for the NACA example



(a) Before optimisation



(b) After optimisation

FIGURE 6.13: Detail of the NACA example, with red circles indicating Voronoi vertices which lie outside their associated Delaunay element.

TABLE 6.2: Quality measures for the mesh for the NACA0012 aerofoil.

Delaunay element quality, q_α

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	94.25	0.5278	-1.625	0.9974
After	98.72	0.6392	-0.5017	0.9992

Distance between Voronoi vertex and centroid, q_β

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	96.81	37.40	0.2046	730.10
After	99.99	26.84	0.0541	116.7

Voronoi edge length, q_γ

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	98.90	107.0	0.0160	369.1
After	99.99	103.9	3.308	419.9

Distance between the Delaunay edge midpoint and the Voronoi edge, q_δ

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	100.0	0.0000	0.0000	0.0000
After	100.0	0.4251	0.0000	77.11

Distance between the Voronoi edge midpoint and the Delaunay edge, q_ϵ

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	96.31	43.33	0.0012	2.192×10^5
After	100.0	19.56	5.842×10^{-4}	99.99

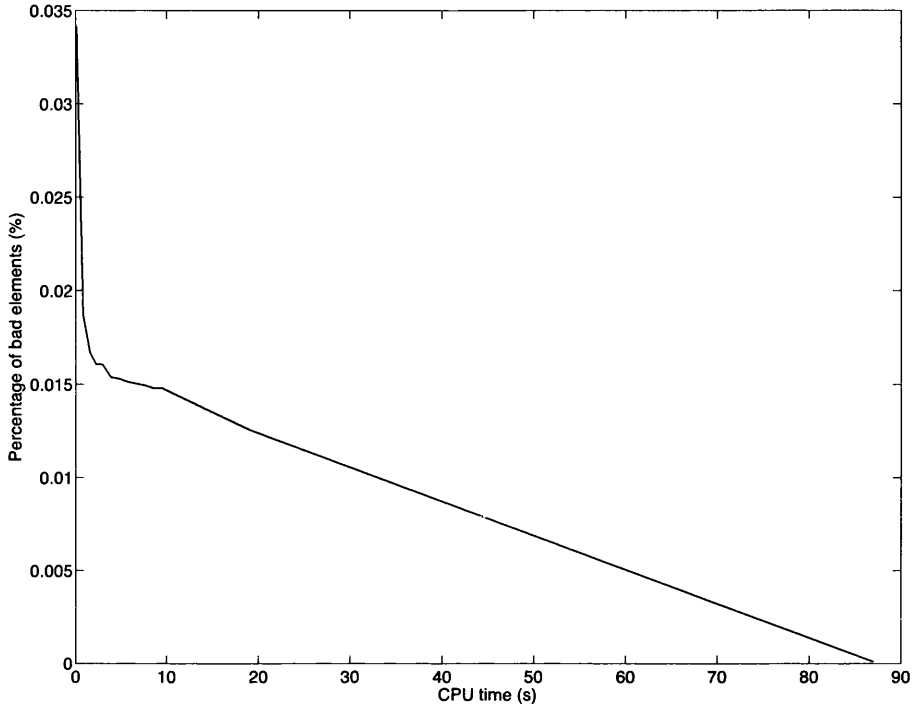
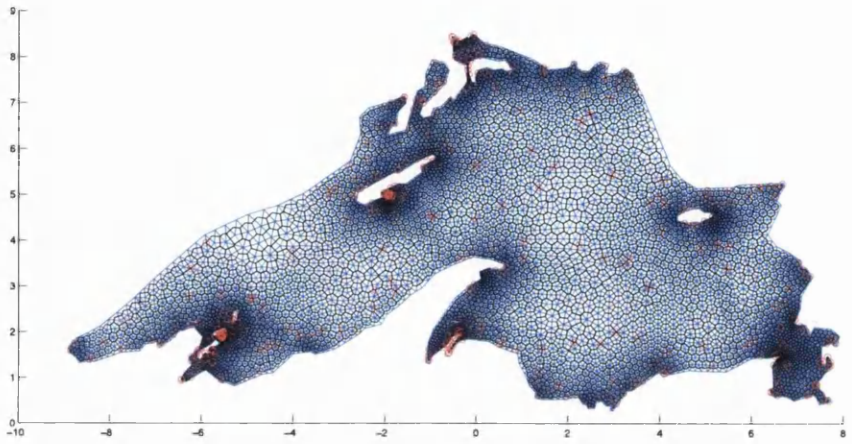


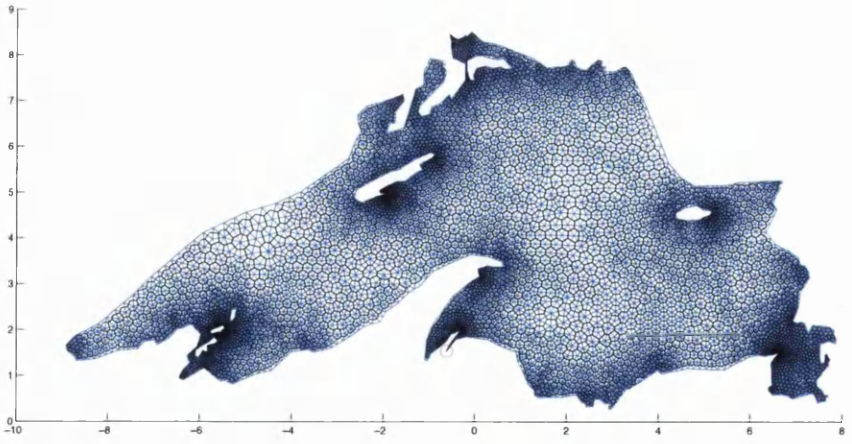
FIGURE 6.14: Optimisation history for the Lake superior example

6.3.4 2D Lake Superior

The next example involved a mesh generated to represent the geometry of Lake Superior. The initial mesh consists of 10 850 elements, with mean Delaunay element quality $q_\alpha = 0.6186$, and contains 402 bad elements. The optimisation history is shown in Figure 6.14. The mean Delaunay element quality increased to $q_\alpha = 0.6697$ and only one bad element remained. Further iterations were unable to repair this element. The initial and optimised meshes are shown in Figures 6.15(a) and 6.15(b) respectively. A detail of the region of the mesh which includes this bad element, before and after optimisation, is shown in Figure 6.16. The nodes of the remaining bad element all lie on the boundary and were unable to move. Since the snapshot generation process is a local optimisation, and only one bad element exists, the MCS could not find a set of weights to fix the element. The only way to fix this would be to move the boundary nodes along the boundary, which as discussed in the methodology is undesirable for practical reasons. The mean CPU cost of one iteration was 6 seconds for this example. The mesh quality measures are presented in Table 6.3. The results showed the same trends as in the previous examples. The percentage of acceptable values of q_β increased, the mean value of q_γ remained close to the acceptable value of 100% and the minimum value increased by

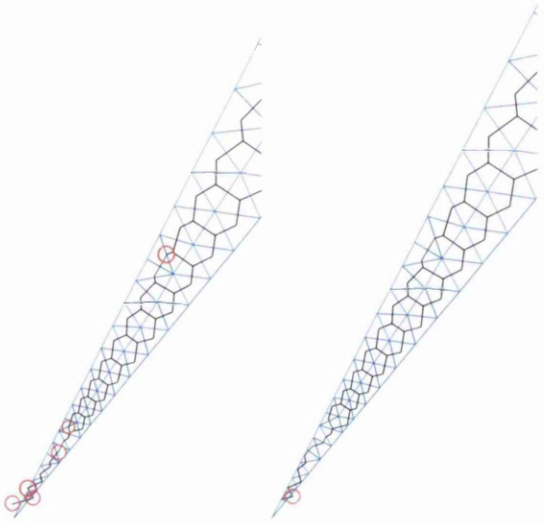


(a) Before optimisation



(b) After optimisation

FIGURE 6.15: The mesh for Lake Superior with red circles indicating Voronoi vertices which lie outside their corresponding Delaunay element.



(a) Mesh before optimisation (b) Mesh after optimisation

FIGURE 6.16: Detail of the mesh for Lake Superior with red circles indicating Voronoi vertices which lie outside their corresponding Delaunay element.

two orders of magnitude. The maximum value of q_δ increased to 66.77% from zero and all values were acceptable. The number of acceptable values of q_ϵ increased to 100%.

TABLE 6.3: Quality measures for mesh for Lake Superior

Delaunay element quality, q_α

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	96.30	0.6186	−1.162	0.9972
After	98.64	0.6697	−1.162	0.9969

Distance between Voronoi vertex and centroid, q_β

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	98.33	29.18	0.1849	499.3
After	99.95	24.66	0.3446	256.9

Voronoi edge length, q_γ

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	98.94	102.8	0.0584	656.5
After	99.98	102.1	6.046	360.0

Distance between the Delaunay edge midpoint and the Voronoi edge, q_δ

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	100.0	0.0000	0.0000	0.0000
After	100.0	0.3696	0.0000	66.77

Distance between the Voronoi edge midpoint and the Delaunay edge, q_ϵ

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	97.47	31.11	1.855×10^{-4}	66200
After	100.00	18.17	0.0020	99.99

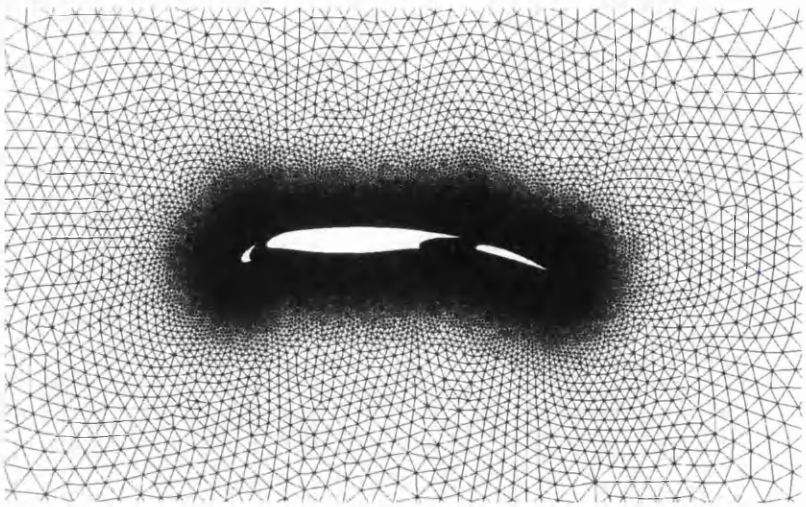


FIGURE 6.17: Mesh for the multi–component aerofoil

6.3.5 Multi–component aerofoil

This example involved a triangular mesh generated in the region bounded internally by a multi–component aerofoil and externally by a circular far–field boundary. The initial mesh consists of 166 294 elements, which was the largest two dimensional example considered. This example was different from the previous examples in that the initial mesh contains a significant variation in the element size, as is apparent from Figure 6.17 which shows a view of the mesh. In spite of the complexity of the geometry, the initial mesh only contains 158 bad elements. This variation in element size had no apparent impact on the effectiveness of the optimisation technique, as Figure 6.18 shows all the bad elements were repaired. This was the most expensive 2D example, with a mean CPU cost of one iteration equal to 232 seconds. Although the nodes were moved during optimisation, the mean Delaunay element quality remained largely unchanged. Table 6.4 shows the full set of quality measures for this example. The maximum value of q_β was more than halved, since all the bad elements were eliminated from the mesh. The mean value, and the percentage of elements with an acceptable value, of q_β were not significantly changed due to the small number of bad elements in the initial mesh. The mean value of q_γ remained unchanged, but the minimum value was increased by an order of magnitude. The maximum value of q_δ was increased from zero to 70.57%, so that all the edges were acceptable. The number of acceptable values of q_ϵ increased to 100%.

Incompressible laminar viscous flow, at a Reynolds Number Re=10 000, over this multi–component aerofoil was modelled by solving the Navier–Stokes equations using a unstructured mesh implementation of the co–volume marker and cell method [148]. Although a solution is obtained on the optimised mesh, it proved to be impossible to obtain a stable

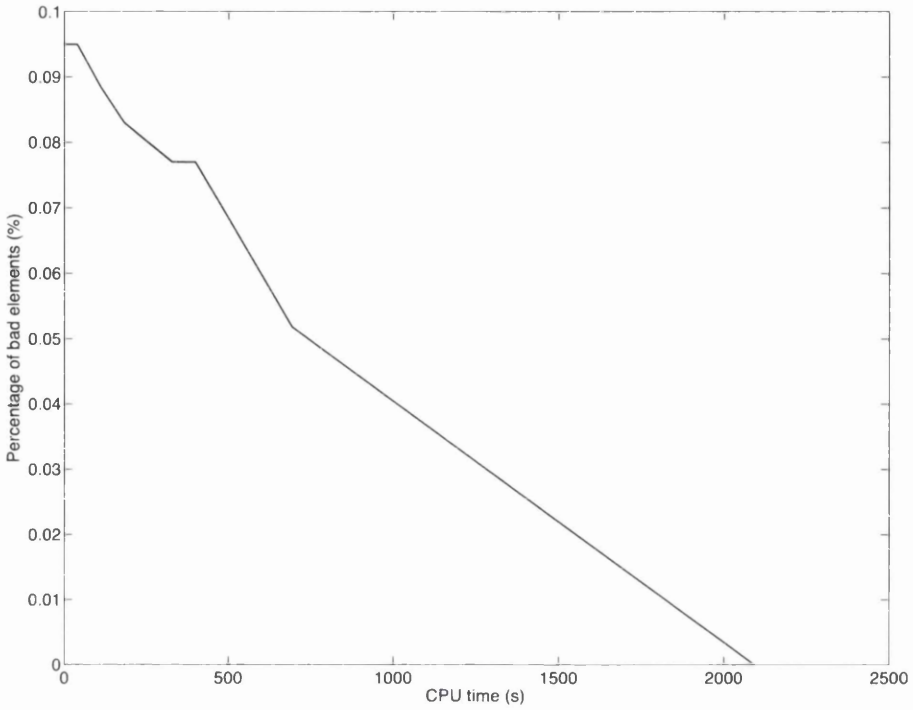


FIGURE 6.18: Optimisation history for the multi–component aerofoil example

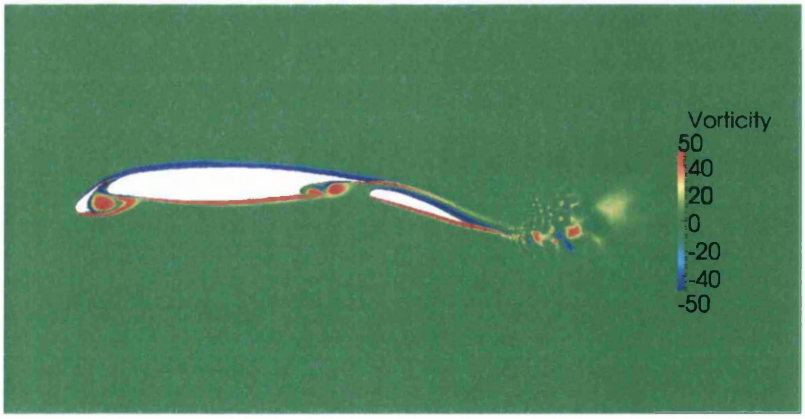


FIGURE 6.19: Plot of the vorticity for incompressible viscous flow over the multi–component aerofoil [148]

solution on the initial mesh. A detail of the computed distribution of vorticity, in the vicinity of the aerofoil on the optimised mesh, is shown in Figure 6.19.

TABLE 6.4: Quality measures for the mesh for the multi–component aerofoil

Delaunay element quality, q_α

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	99.90	0.8584	−0.7744	0.9999
After	99.95	0.8619	−0.4834	0.9999

Distance between Voronoi vertex and centroid, q_β

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	99.95	10.52	0.0041	216.8
After	100.0	10.21	0.0026	97.94

Voronoi edge length, q_γ

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	99.99	100.9	0.3665	298.7
After	99.99	100.9	7.111	322.2

Distance between the Delaunay edge midpoint and the Voronoi edge, q_δ

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	100.00	0.0000	0.0000	0.0000
After	100.00	0.0307	0.0000	70.57

Distance between the Voronoi edge midpoint and the Delaunay edge, q_ϵ

	Percentage acceptable	*Mean*	*Minimum*	*Maximum*
Before	99.94	7.485	0.0000	9853
After	100.0	6.986	5.451×10^{-6}	99.99

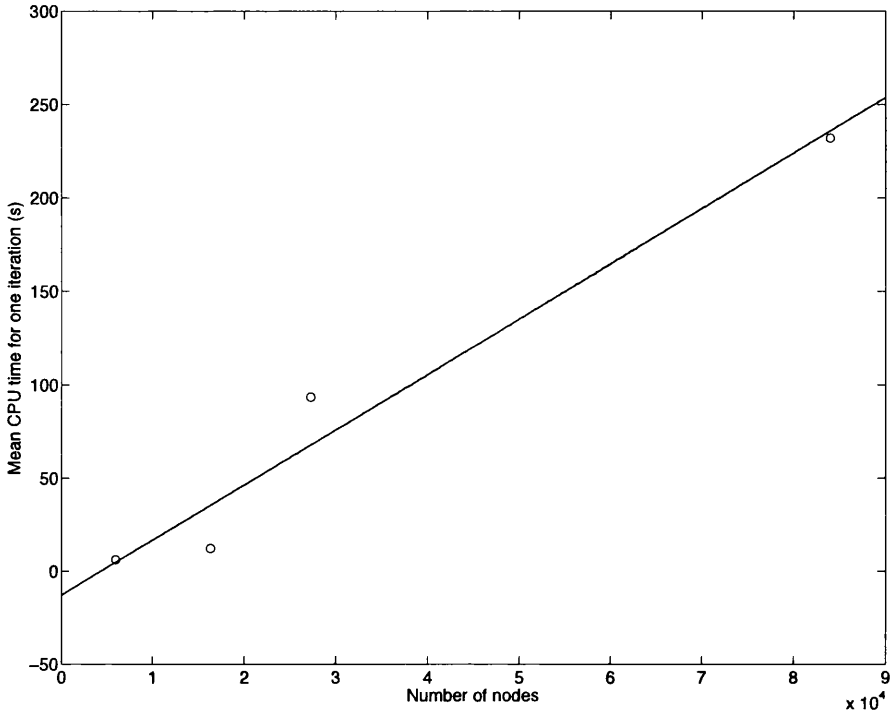


FIGURE 6.20: Cost analysis, number of nodes plotted against mean iteration cost

6.3.6 Cost analysis

Using the two dimensional examples a simple cost analysis of the scheme was performed. Since most operations involve looping over nodes, it would be expected that the cost of this scheme would be $O(N)$. Figure 6.20 shows that this was indeed the case. The mean CPU cost of an iteration was plotted against N for the above examples, with the solid line illustrating the relation was close to linear.

FIGURE 6.21: The sphere mesh before optimisation, the left hand plot shows all the elements and the right hand plot shows just the bad elements

6.3.7 Sphere

Primarily, the focus of this chapter was to use two dimensional examples to explore a new strategy for mesh optimisation. A number of examples were also performed which illustrate the potential for this method to work in three dimensions. In the three dimensional examples, only the number of bad elements was considered as a quality metric, since this was the prime motivation of the technique. For the first example, an initial tetrahedral mesh, plotted in Figure 6.21, was generated for the region inside a sphere. Compared with the 2D examples, there was a much higher concentration of bad elements throughout the mesh. The mesh contains 2 775 elements, of which 451, or 16%, are bad. Figure 6.22 shows how the percentage of bad elements decreased to zero, during the optimisation process.

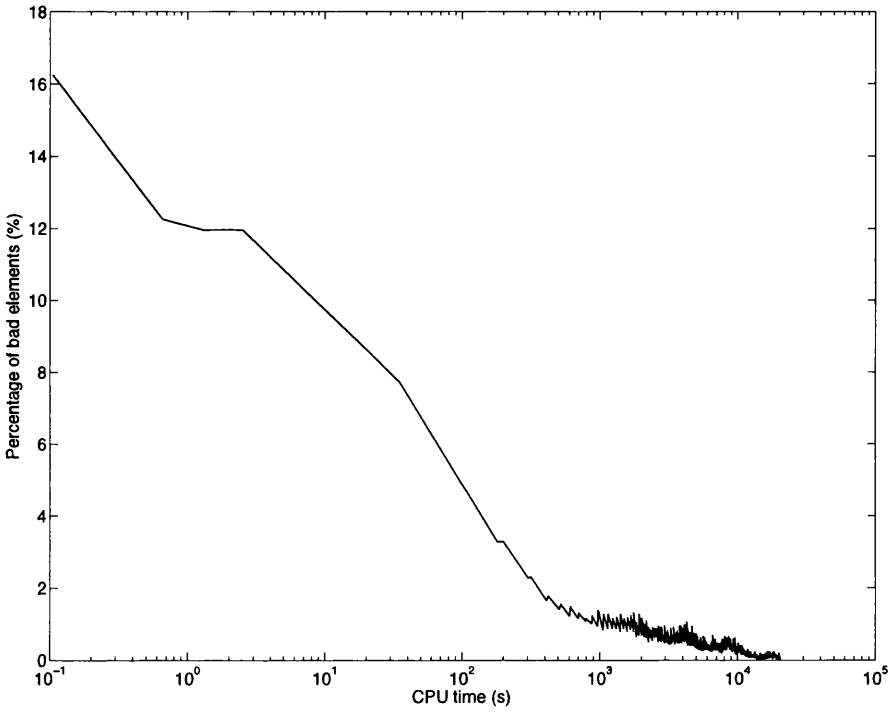


FIGURE 6.22: Optimisation history for the 3D sphere example

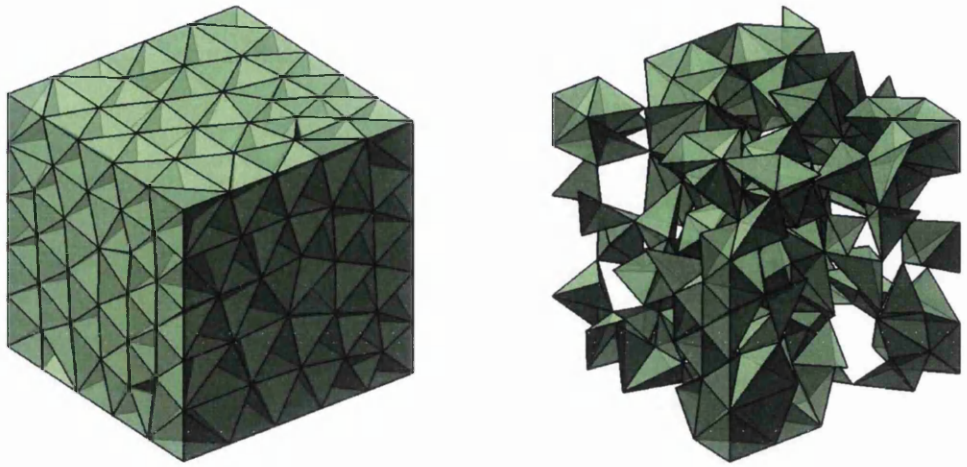


FIGURE 6.23: The cube mesh before optimisation, the left hand plot shows all the elements and the right hand plot shows just the bad elements

6.3.8 Cube

This example, used by Klinger and Shewchuk [144], is a mesh of a cube generated by NETGEN [149]. The mesh is plotted in Figure 6.23. Despite this being quite a small mesh of only 1184 elements, there is a large percentage of bad elements initially (287 in total). There are also a number of situations at the corners of the cube where an element is entirely made up of boundary nodes. In this case the only way to fix the element is by optimisation of the weights. Despite this difficulty all bad elements were fixed by the optimisation process. The cost and history of convergence is shown in Figure 6.24.

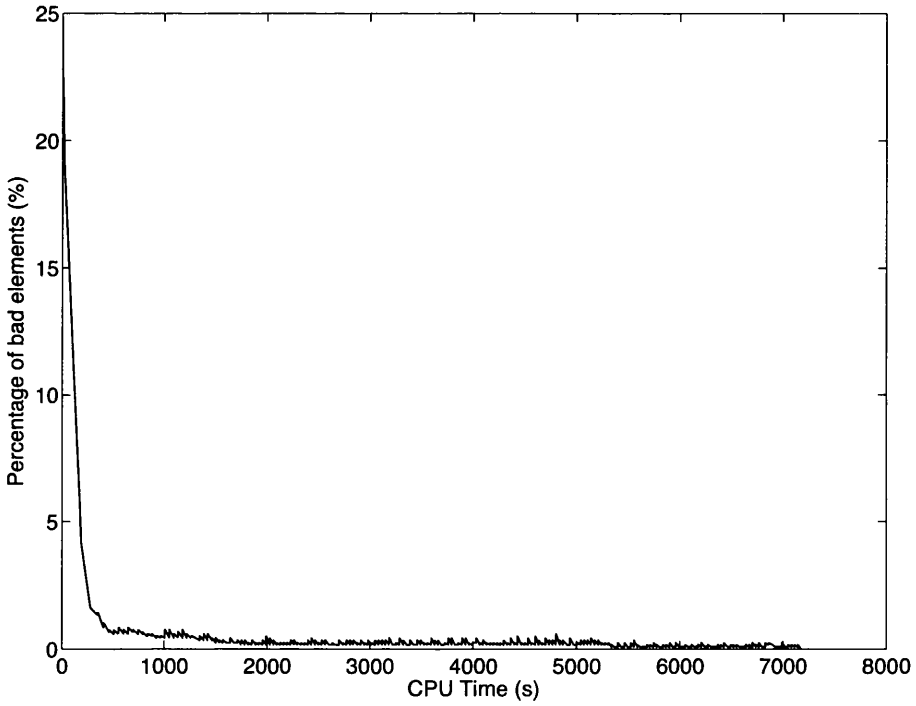


FIGURE 6.24: Optimisation history for the 3D cube example

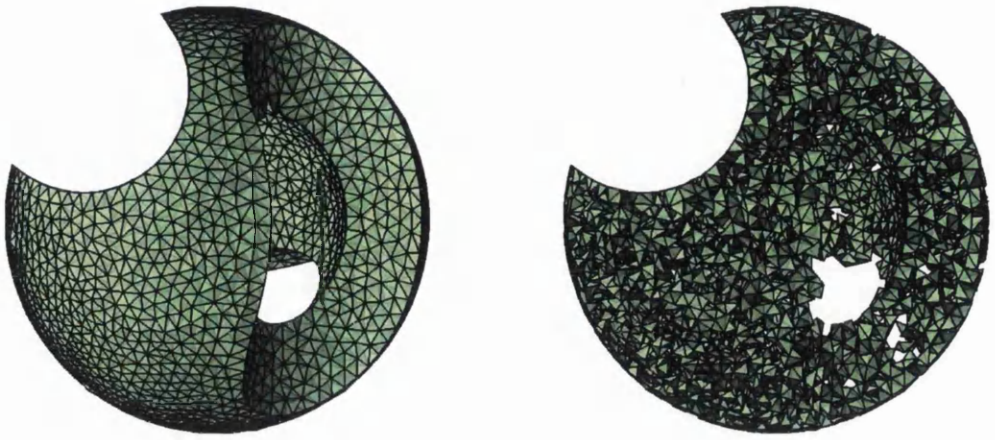


FIGURE 6.25: The St Gallen mesh before optimisation, the left hand plot shows all the elements and the right hand plot shows just the bad elements

6.3.9 St Gallen

Figure 6.25 shows the St Gallen mesh before optimisation. This example is also used by Klinger and Shewchuk [144]. It has many curved boundaries, and was generated by Alliez et al. [150]. The mesh contains 50391 tetrahedral elements, of which 15.3% are bad initially. The number of boundaries in this problem posed a challenge for the optimisation scheme, and the technique was unable to fix all elements, within an acceptable time. Despite this, the technique was significantly better than Lloyd's algorithm, as shown in Figure 6.26. Lloyd's algorithm reduced the percentage of bad elements to 10.8%, whereas a combination of local coordinate optimisation, using MCS, and global weight optimisation, using POD and MCS, was able to reduce the percentage of bad elements to 0.788%. In this example, Lloyd's algorithm and Laplace smoothing were not employed prior to the node and weight optimisation.

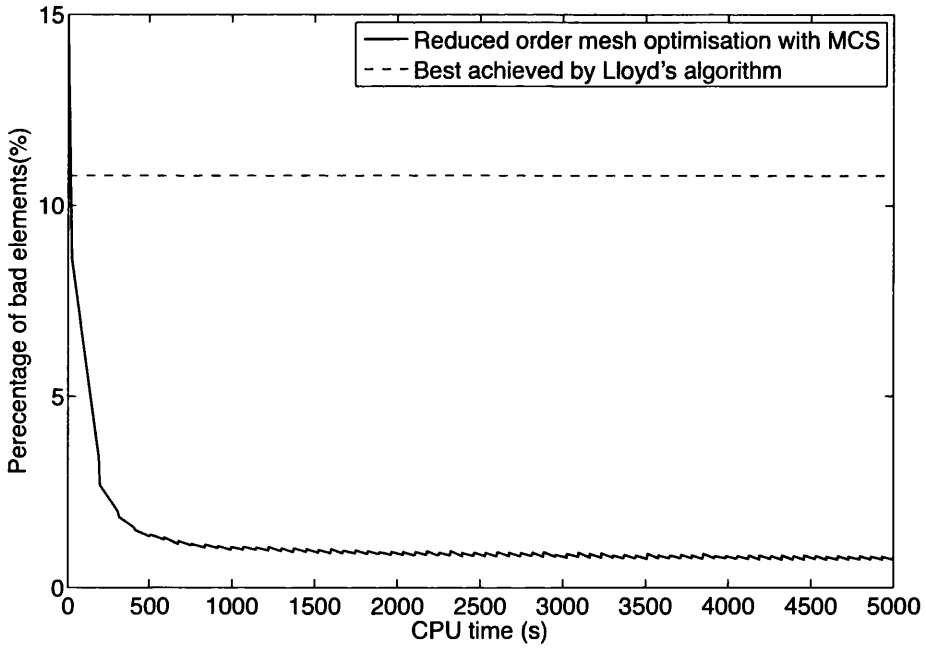


FIGURE 6.26: Optimisation history for the St Gallen example

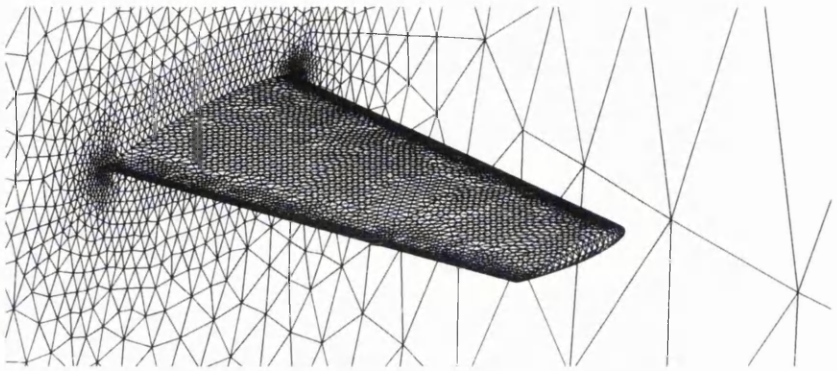


FIGURE 6.27: The ONERA M6 mesh before optimisation

6.3.10 ONERA M6

An ONERA M6 wing meshed externally to a spherical far field was the final example performed. Part of the surface mesh is plotted in Figure 6.27. This example was the largest mesh considered, with 245421 elements, of which 42.6% are bad initially. With the largest percentage of bad elements initially, this example was the most difficult considered. As with the St Gallen example, the technique was unable to fix all of the elements within an acceptable timeframe. However, the technique significantly outperformed Lloyd's algorithm. When just applying Lloyd's algorithm, the lowest number of bad elements achieved was 35.4%. Figure 6.28 shows the optimisation history for a combination of node and weight optimisation, without the initial application of Laplace smoothing or Lloyd's algorithm. After 5000 seconds of CPU time, the number of bad elements was reduced to 7.19% by the technique introduced. In both the St Gallen and ONERA M6 examples, boundary constraints limit the number of bad elements which were able to be fixed. With further research, this problem may be alleviated by moving the boundary points, or by inserting points on the surface or into the volume mesh. The possibility of merging Delaunay elements which have small Voronoi edges, as described in [128], has been neglected. This may also result in a reduction in the number of bad elements.

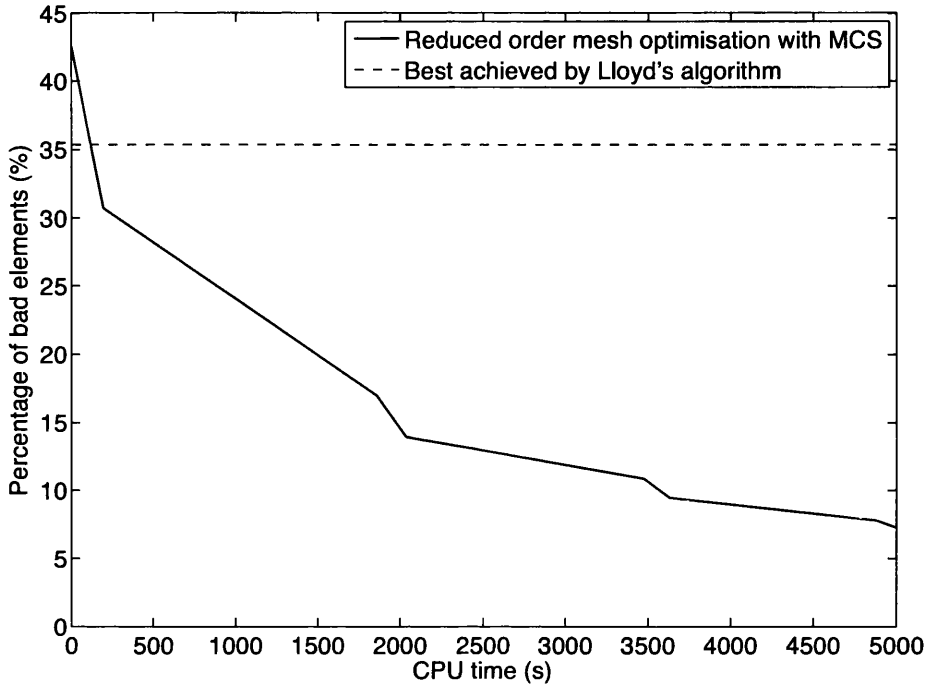


FIGURE 6.28: Optimisation history for the ONERA M6 example

6.4 Discussion

An optimisation technique for the problem of generating high quality dual orthogonal unstructured meshes has been presented. It has been shown that, by considering the weighted Voronoi power diagram, it was possible to find a coordinate/weight distribution which results in a mesh in which all, or almost all, the Voronoi vertices lie within their associated element. The positive effect of this improvement, on co–volume solution methods, has been discussed [128–130, 148]. In addition, it has been demonstrated that the optimisation technique did not have a negative effect on the spatial quality of the primal Delaunay mesh in two dimensions. Indeed, in some cases, the spatial quality of the primal Delaunay mesh was also improved. The maximum distance between any Voronoi vertex and the centroid of its corresponding element was reduced in all the examples considered and, unexpectedly, the optimisation process significantly increased the minimum Voronoi edge length in all two dimensional cases. This will have positive implications to the magnitude of time step size that can be adopted in explicit co–volume simulations. The cost of the process was that a Voronoi edge may no longer bisect the corresponding Delaunay edge. This will have certain implications on the accuracy of simulations. In the future, this could be addressed by considering the possibility of multi–objective function optimisation, i.e. attempting to optimise the mesh based on

multiple quality measures, such as adding a penalty for deviations from the Delaunay edge bisection.

The method is general and has been applied successfully in two simple three dimensional examples. It was unable to fix all bad elements in two of the more complex three dimensional examples presented. Despite being unable to fix all bad elements, the technique still performed well compared with standard techniques. In both these examples, the optimisation was still reducing the number of bad elements when the procedure was interrupted due to excessive CPU time. The rate at which the final $5 - 10\%$ of bad elements were being fixed was not rapid enough to be practical. This needs to be improved with further development of the technique. It is felt that the results presented here do show this novel technique has real potential moving forward.

The method did not require the subdivision of elements, by the addition of nodes, to repair all elements in a mesh, which is a requirement of some methods as discussed in Section 6.2.2.1. The aspects which led to the success of the method were the use of POD to reduce the dimensionality of the problem and enable the application of MCS. Although this method was applied specifically to optimising meshes for use with co–volume schemes, the method could be readily adapted for other applications, by simply changing the mesh quality objective function to reflect the new requirements. It would also be possible to apply model reduction to the node coordinates themselves.

Chapter 7

Conclusion

The objective of this thesis was to apply reduced order modelling and optimisation techniques to the problem of aerodynamic design. Only techniques where big design changes are possible were of interest. This objective can be split into two parts, the optimisation algorithm and the reduced order modelling techniques.

The overview given in Chapter 2 led to the consideration of metaheuristic gradient free optimisation techniques. Cuckoo search was selected as a potential algorithm and modified to improve performance in Chapter 3.

It was found, in Chapter 4, that, when not coupled with reduced order modelling techniques, the technique showed potential when applied to aerodynamic shape optimisation.

The literature study presented in Chapter 5 shows that most reduced order modelling techniques were only valid for fixed flow parameters. One technique which showed promise was the proper orthogonal decomposition interpolation technique discussed in Section 5.4.2. This technique was tested for steady problems in Section 5.4, where it was deemed unsuitable for many general optimisation problems.

The contributions and findings of the thesis are now summarised, and then suggestions for future work are presented.

7.1 Contributions and Findings of the Thesis

7.1.1 Modified Cuckoo Search [1]

As part of this work, the optimisation algorithm modified cuckoo search was developed. This algorithm is a modification of cuckoo search. The standard algorithm was modified

by the addition of information exchange between the top eggs (best solutions). Standard optimisation benchmarking functions were then used to test the effects of these modifications. It was found that modified cuckoo search performs as well or better than the standard cuckoo search and a particle swarm optimisation and differential evolution algorithm in most cases.

An implementation of the algorithm was published in the open source project `modified-cs` [64]. The code has been downloaded around 1,000 times and has already been used in a range of applications which were discussed in Chapter 3.

7.1.2 Reduced Order Modelling of Unsteady Fluid Flow using Proper Orthogonal Decomposition and Radial Basis Functions [2]

A technique was developed for interpolating unsteady solutions to parameterised fluid flow problems, using a combination of proper orthogonal decomposition and radial basis functions. The technique was validated by considering simulations involving three dimensional unsteady compressible inviscid flow over an oscillating ONERA M6 wing. It was demonstrated that the approach can result in a large reduction in the CPU time required to find solutions, at new parameter values, without a significant loss in accuracy.

The technique developed is much simpler than other unsteady reduced order modelling techniques, based on proper orthogonal decomposition, and is less prone to stability problems and boundary condition violations. If the limitations of this technique, i.e. that it is equivalent to nodewise interpolation, are recognised and accepted it could be very useful in a number of applications.

7.1.3 Reduced Order Mesh Optimisation using Proper Orthogonal Decomposition and a Modified Cuckoo Search [3]

A new mesh optimisation scheme, reduced order mesh optimisation, was introduced. The technique uses proper orthogonal decomposition to reduce the number of dimensions in a mesh optimisation problem. This reduction in dimensionality allows the expression of the optimisation problem globally rather than the more traditional local mesh optimisation or smoothing algorithms. To perform the optimisation, modified cuckoo search was applied. The effectiveness of the algorithm was shown by considering the problem of optimising meshes for use in co–volume techniques. Co–volume techniques require the existence of two mutually orthogonal meshes. This is achieved by utilising the Delaunay–Voronoi dual. A combination of considering the problem globally, and the use of a

gradient free technique, results in a scheme which significantly outperforms previous methods.

7.1.4 Applying Metaheuristic Algorithms to Engineering Applications

An aim of the thesis was to investigate the use of metaheuristic gradient free algorithms for engineering applications. A number of applications have been presented, in which the use of gradient free methods proved successful. However, three key problems can been highlighted, which may deter the use of metaheuristic algorithms for practical optimisation problems. An attempt to address these concerns is now presented.

7.1.4.1 Poor Computational Efficiency

The large number of objective function evaluations required by metaheuristic algorithms results in poor computational efficiency. This problem can be approached from several angles.

One approach is to attempt to improve the convergence behaviour of the optimisation algorithm itself, thus reducing the number of generations required to obtain a good solution. This often comes down to algorithm and tuning parameter selection, which is discussed in Section 7.1.4.2 below.

For a defined optimisation algorithm, two further strategies remain to increase computational efficiency, viz. parallelisation and meta modeling. Parallelisation appears promising and relatively straightforward, as each agent can simply evaluate the objective function on its own CPU. An alternative use of multiple CPUs is discussed in Section 7.1.4.3 below.

Reduced order modelling is an active field of research and, as reduced order models continue to improve, it can be expected that the use of meta models to approximate the objective function will become more viable.

Of course, the efficiency problem will also become less significant as computational power continues to increase.

7.1.4.2 Tuning Parameters and the Range of Available Algorithms

The choice of which algorithm to use may appear to be difficult. Experience indicates that, given enough experimental time, any algorithm can give good results for a particular problem. The range of successful applications, discussed in this thesis and elsewhere,

supports this statement. If this is the case, then the two main aspects that should be considered, when selecting an algorithm, are the ease of implementation and the number of parameters which need to be tuned.

The nature of metaheuristic algorithms means that it is very unlikely that a truly black–box optimiser, with no free parameters, will ever exist. A number of algorithms, including cuckoo search, claim low sensitivity to tuning parameters and this may be considered as a benefit. Conversely, a lack of tuning ability may mean that an algorithm shows reasonable convergence to most problems, but that it cannot be tuned to give excellent convergence for a specific problem of interest. For example, when applying modified cuckoo search to mesh optimisation, a single set of tuning parameters was used for every example [3], but, if time was spent tuning the parameters for each new example, better results probably could have been achieved. Problem specific tuning is, however, unlikely to be practical for real applications.

7.1.4.3 Randomness of Metaheuristic Algorithms

People can often be deterred by the random nature of these algorithms. This is an understandable concern, as a gradient based algorithm, run multiple times, always produces the same result. This is not guaranteed for gradient free techniques. Thus, in practice, it may be desirable to run the optimiser multiple times, which can be an expensive process. There is an argument that the best way to parallelise these algorithms is to simply use spare nodes to run completely independent optimisers, possibly even different algorithms.

The randomness should not be seen as a problem. The evidence presented in this thesis and elsewhere shows these algorithms work in real problems. The issue can be summed up in the following question, posed in the context of using an optimiser to improve a certain design. Does it matter if different designs are obtained with different runs, if they are all better than the starting design?

7.2 Future work

7.2.1 Cuckoo Search

In Chapter 3, a number of different modifications, made by various researchers, to the cuckoo search algorithm were detailed. A comparison between the performances of these different modifications is needed. Currently cuckoo search development has branched

into a number of different directions. It may be beneficial to merge some of these branches, by combining different modifications.

Parallisation strategies need to be investigated to improve computational efficiency. As previously discussed, there are many different strategies which will need to be compared.

Further validation of cuckoo search and modified cuckoo search is needed. Initially this could be done on a wider range of shifted and rotated test functions, but the ultimate goal should be to test the algorithms on real problems. Comparisons between other gradient free and gradient based algorithms should be made where possible.

7.2.2 Aerodynamic Shape Optimisation using Modified Cuckoo Search

Using modified cuckoo search for aerodynamic shape optimisation requires much more investigation. Initially, it is important to increase the complexity of the flow physics to make the problem more realistic. It will be important to consider robust design optimisation going forward, which should result in more realistic and practical designs which are less sensitive to flow conditions.

Two parameter studies need to be performed. The first with regards to the number of degrees of freedom associated with the shape parameterisation itself. More degrees of freedom would result in increased flexibility, but possibly at an increased CPU cost to effectively sample the design space. The second parameter study should investigate the sensitivity on the modified cuckoo search tuning parameters. Comparing the performance of modified cuckoo search to other optimisation algorithms is also needed.

Finally, more work into developing a suitable reduced order modelling technique for this application will need to be undertaken. At present, the number of objective function evaluations required to reach a solution will be prohibitive for industrial applications. However, from a purely scientific point of view it would be interesting to investigate the limits of shape optimisation using this technique. Is it possible to start a design cycle with a circle and end up with an aerofoil? Further to this it would be interesting to apply this to more non–standard problems. For example, work has already begun on applying modified cuckoo search to the design of the intake duct for the BLOODHOUND super sonic car. The geometry for the car has already been fixed, but it would be interesting to see if the optimisation algorithm reaches the same conclusion as the aerodynamic engineers.

7.2.3 Unsteady Proper Orthogonal Decomposition Interpolation Techniques

The most important study which needs to be performed is to compare the proper orthogonal decomposition interpolation techniques developed in this thesis, for unsteady problems, to Galerkin projection based reduced order models. The literature suggests that Galerkin projection has a number of stability and implementation issues which are not exhibited by proper orthogonal decomposition interpolation. A direct comparison, however, is still required.

As with the shape optimisation example, it is important to increase the complexity of the flow physics to three dimensional viscous flow. If the technique cannot perform well in these more complicated situations, it may not be worth developing further.

If the results of the above two studies are positive for the proper orthogonal decomposition interpolation technique, further investigation into which interpolation and sampling techniques are most efficient needs to be performed. In addition, these techniques may be enhanced by coupling them with the residual reduction method discussed in Section 5.4.1. Proper orthogonal decomposition interpolation could provide an initial guess solution for the optimisation process required by residual reduction methods. Furthermore, it may be beneficial to use modified cuckoo search in place of the genetic algorithms currently used for the optimisation.

Another interesting line of investigation would be to find a way to measure the ability of a particular set of modes to recreate new solutions. It should be possible to calculate the best achievable accuracy for a set of modes to reproduce a solution. This would then give a very clear measure of how successful a particular technique for calculating mode coefficients is.

One area where the technique could be applied immediately is in an educational setting. For simple problems, a database of snapshots could be generated to allow students to interactively explore a parameter space in real time. Instead of just seeing how the aerodynamic coefficients change, students would be able to inspect a reasonable estimation of the full flow field.

7.2.4 Reduced Order Mesh Optimisation

Reduced order mesh optimisation combines most of the techniques considered in this thesis and is a major contribution. It has the potential to be applied to many different forms of mesh optimisation. For example, it would be interesting to apply the technique

to the optimisation of meshes to be used for high order techniques, which currently presents a challenge. The work presented in the thesis may only scratch the surface of what this technique has to offer.

A weakness of the current algorithm is the computational cost required to improve tetrahedral meshes. Parallisation strategies, along the lines of those used by Doorly and Peiró [30], involving subpopulations of meshes on different CPUs evolving semi–independently may be beneficial. Another interesting avenue might be to assign different partitions of the mesh to different CPUs during local optimisation routines.

To strengthen the conclusions presented in this thesis, further comparisons with alternative mesh optimisation techniques is required. In addition, the benefit of global mesh optimisation compared to node wise mesh optimisation needs to be quantified. Further to this it would be interesting to see how far the idea of a global objective function can be pushed. For instance, what would be the effect of using vectors containing not only node weights, but node coordinates, as snapshots?

An ultimate goal is to fully automate the generation and optimisation of a mesh for any arbitrary geometry. The problem is that mesh optimisation, and generation, is highly problem specific. It would be impossible to test and tune a technique for every conceivable geometry. One way of solving this problem might be to take advantage of the always on–line nature of modern computers. If a mesh optimisation toolbox was made available for general use, it could be developed to report performance statistics, along with tuning parameters, back to a central server. This could be done in such a way to avoid issues with confidential geometries, by only sending mesh quality information. Over time this would build up a much bigger sample than any single research group could hope to achieve. The hope would be that an optimum set of tuning parameters would emerge.

Chapter 8

Publications Resulting from this Thesis

Journal Articles

1. S. Walton, O. Hassan, K. Morgan, and M. R. Brown. Modified cuckoo search: a new gradient free optimisation algorithm. *Chaos, Solitons & Fractals*, 44(9): 710–718, 2011

2. S. Walton, O. Hassan, and K. Morgan. Reduced order mesh optimisation using proper orthogonal decomposition and a modified cuckoo search. *International Journal for Numerical Methods in Engineering*, 93(5):527–550, 2013. ISSN 1097-0207. doi: 10.1002/nme.4400. URL `http://dx.doi.org/10.1002/nme.4400`

3. S. Walton, O. Hassan, and K. Morgan. Reduced order modelling for unsteady fluid flow using proper orthogonal decomposition and radial basis functions. *Applied Mathematical Modelling*, 2013. ISSN 0307-904X. doi: 10.1016/j.apm.2013.04.025

4. S. Walton, O. Hassan, and K. Morgan. Selected engineering applications of gradient free optimisation using cuckoo search and proper orthogonal decomposition. *Archives of Computational Methods in Engineering*, 20(2):123–154, 2013. ISSN 1134-3060. doi: 10.1007/s11831-013-9083-7. URL `http://dx.doi.org/10.1007/s11831-013-9083-7`

5. S. Walton, M. R. Brown, O. Hassan, and K. Morgan. Comment on "Cuckoo search: A new nature-inspired optimization method for phase equilibrium calculations" by V. Bhargava, S. Fateen, A. Bonilla-Petriciolet [Fluid Phase Equilibria 337 (0) (2013) 191 - 200]. *Fluid Phase Equilibria*, 2013. doi: 10.1016/j.fluid.2013.05.011

Book Chapters

1. S. Walton, O. Hassan, K. Morgan, and M. R. Brown. The development and applications of the cuckoo search algorithm. In X-S. Yang, Cui. C., R. Xiao, A. H. Gandomi, and M. Karamanoglu, editors, *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, pages 257–271. Elsevier, September 2013. doi: http://dx.doi.org/10.1016/B978-0-12-405163-8.00011-9

Conference Presentations

1. S. Walton, O. Hassan, and K. Morgan. Reduced order modelling of compressible flow past an airfoil using proper orthogonal decomposition. In A. Zervos, editor, *Proceedings of ACME2010: the 18th Annual Conference of the Association of Computational Mechanics in Engineering*, 2010, *was awarded Best Paper by a Postgraduate Student*

2. S. Walton, O. Hassan, and K. Morgan. Using proper orthogonal decomposition to reduce the order of optimization problems. In W. A. Wall and V. Wall, W. A., editors, *Proceedings of the 16th International Conference on Finite Elements in Flow Problems*, 2011

3. S. Walton, O. Hassan, and K. Morgan. Interpolating unsteady paramaterised CFD solutions using radial basis functions and proper orthogonal decomposition. In *European Congress on Computational Methods in Applied Sciences and Engineering*, 2012

Bibliography

[1] S. Walton, O. Hassan, K. Morgan, and M. R. Brown. Modified cuckoo search: a new gradient free optimisation algorithm. *Chaos, Solitons & Fractals*, 44(9): 710–718, 2011.

[2] S. Walton, O. Hassan, and K. Morgan. Reduced order modelling for unsteady fluid flow using proper orthogonal decomposition and radial basis functions. *Applied Mathematical Modelling*, 2013. ISSN 0307-904X. doi: 10.1016/j.apm.2013.04.025.

[3] S. Walton, O. Hassan, and K. Morgan. Reduced order mesh optimisation using proper orthogonal decomposition and a modified cuckoo search. *International Journal for Numerical Methods in Engineering*, 93(5):527–550, 2013. ISSN 1097-0207. doi: 10.1002/nme.4400. URL `http://dx.doi.org/10.1002/nme.4400`.

[4] D.N. Ball and H.H.P.C. Center. Contributions of CFD to the 787-and future needs. In *IDC HPC User Forum*, 2008.

[5] R. C. Michelson. *Overview of Micro Air Vehicle System Design and Integration Issues.* John Wiley & Sons, Ltd, 2010. ISBN 9780470686652. doi: 10.1002/9780470686652.eae401. URL `http://dx.doi.org/10.1002/9780470686652.eae401`.

[6] V. Bhargava, S.E.K. Fateen, and A. Bonilla-Petriciolet. Cuckoo search: A new nature-inspired optimization method for phase equilibrium calculations. *Fluid Phase Equilibria*, 337(0):191 – 200, 2013. ISSN 0378-3812. doi: 10.1016/j.fluid.2012.09.018.

[7] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[8] I. Stich, R. Car, M. Parrinello, and S. Baroni. Conjugate gradient minimization of the energy functional: A new method for electronic structure calculation. *Phys. Rev. B*, 39:4997–5004, Mar 1989. doi: 10.1103/PhysRevB.39.4997. URL `http://link.aps.org/doi/10.1103/PhysRevB.39.4997`.

[9] R. Fletcher. *Practical methods of optimization; (2nd ed.)*. Wiley-Interscience, New York, NY, USA, 1987. ISBN 0-471-91547-5.

[10] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, August 2000. ISBN 0387987932.

[11] F. Van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, 2006.

[12] Matteo Diez and Daniele Peri. Robust optimization for ship conceptual design. *Ocean Engineering*, 37(11–12):966 – 977, 2010. ISSN 0029-8018. doi: 10.1016/j.oceaneng.2010.03.010.

[13] E. Weisstein. "newton's method." from mathworld–a wolfram web resource., . URL `http://mathworld.wolfram.com/NewtonsMethod.html`.

[14] E. Weisstein. "method of steepest descent." from mathworld–a wolfram web resource., . URL `http://mathworld.wolfram.com/MethodofSteepestDescent.html`.

[15] N. Black, S. Moore, and E. Weisstein. "conjugate gradient method." from mathworld–a wolfram web resource. URL `http://mathworld.wolfram.com/ConjugateGradientMethod.html`.

[16] Inc. Vanderplaats Research & Development. DOT - design optimization tools, 2013. URL `http://www.vrand.com/DOT.html`.

[17] The MathWorks Inc. MATLAB version 7.8.0, 2009.

[18] S. S. Rao. *Engineering Optimization: Theory and Practice, 4th Edition*. John Wiley & Sons, New Jersey, 2009. ISBN 978-0-470-18352-6.

[19] R. E. Griffith and R. A. Stewart. A nonlinear programming technique for the optimization of continuous processing systems. *Management Science*, 7(4):pp. 379–392, 1961. ISSN 00251909. URL `http://www.jstor.org/stable/2627058`.

[20] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4:1–51, 0 1995. ISSN 1474-0508. doi: 10.1017/S0962492900002518. URL `http://journals.cambridge.org/article_S0962492900002518`.

[21] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3:233–260, 1988. ISSN 0885-7474. doi: 10.1007/BF01061285. URL `http://dx.doi.org/10.1007/BF01061285`.

[22] A. Jameson. Efficient aerodynamic shape optimization. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2004.

[23] M. B. Giles and N. A. Pierce. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 65:393–415, 2000. ISSN 1386-6184. doi: 10.1023/A:1011430410075.

[24] M. J. D. Powell. Direct search algorithms for optimization calculations. *Acta Numerica*, 7:287–336, 0 1998. ISSN 1474-0508. doi: 10.1017/S0962492900002841. URL `http://journals.cambridge.org/article_S0962492900002841`.

[25] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965. doi: 10.1093/comjnl/7.4.308. URL `http://comjnl.oxfordjournals.org/content/7/4/308.abstract`.

[26] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9:112–147, 1998.

[27] M Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, sixth edition, 1999.

[28] H. Salimi, D. Giveki, M. A. Soltanshahi, and J. Hatami. Extended mixture of MLP experts by hybrid of conjugate gradient method and modified cuckoo search. *International Journal of Artificial Intelligence & Applications*, 3, 2012.

[29] N. Chaiyaratana and A.M.S. Zalzala. Recent developments in evolutionary and genetic algorithms: theory and applications. In *Genetic Algorithms in Engineering Systems: Innovations and Applications, 1997. GALESIA 97. Second International Conference On (Conf. Publ. No. 446)*, pages 270 –277, sep 1997. doi: 10.1049/cp:19971192.

[30] D.J. Doorly and J. Peiró. Supervised parallel genetic algorithms in aerodynamic optimisation. In *Artificial Neural Nets and Genetic Algorithms*, pages 229–233. Springer Vienna, 1998. ISBN 978-3-211-83087-1. doi: 10.1007/978-3-7091-6492-1_50. URL `http://dx.doi.org/10.1007/978-3-7091-6492-1_50`.

[31] S. Das and P.N. Suganthan. Differential evolution: a survey of the state–of–the–art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, feb. 2011. ISSN 1089-778X. doi: 10.1109/TEVC.2010.2059031.

[32] R. Storn and K. Price. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimisation*, 11: 341–359, 1997.

[33] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 120 –127, april 2007. doi: 10.1109/SIS.2007.368035.

[34] C. Praveen and R. Duvigneau. Low cost PSO using metamodels and inexact pre-evaluation: application to aerodynamic shape design. *Computer Methods in Applied Mechanics and Engineering*, 198:1087–1096, 2009.

[35] Eberhart and S. Yuhui. Particle swarm optimization: developments, applications and resources. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 81 –86 vol. 1, 2001. doi: 10.1109/CEC.2001.934374.

[36] X.-S. Yang and S. Deb. Cuckoo search via Lévy flights. In *Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009, India)*, pages 210–214. IEEE Publications, 2009.

[37] E. R. Speed. Evolving a Mario agent using cuckoo search and softmax heuristics. In *Proceedings of Games Innovations Conference (ICE-GIC)*, pages 1–7, 2010. DOI 10.1109/ICEGIC.2010.5716893.

[38] K. Choudhary and G. N. Purohit. A new testing approach using cuckoo search to achieve multi–objective genetic algorithm. *Journal of Computing*, 3(4):117–119, 2011.

[39] A. H. Gandomi, X-S Yang, and A. Alavi. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering with Computers*,

29:17–35, 2013. ISSN 0177-0667. doi: 10.1007/s00366-011-0241-y. URL `http://dx.doi.org/10.1007/s00366-011-0241-y`.

[40] A. H. Gandomi, S. Talatahari, X-S Yang, and S. Deb. Design optimization of truss structures using cuckoo search algorithm. *The Structural Design of Tall and Special Buildings*, pages n/a–n/a, 2012. ISSN 1541-7808. doi: 10.1002/tal.1033. URL `http://dx.doi.org/10.1002/tal.1033`.

[41] A. Kaveh and T. Bakhshpoori. Optimum design of steel frames using cuckoo search algorithm with lévy flights. *The Structural Design of Tall and Special Buildings*, pages n/a–n/a, 2011. ISSN 1541-7808. doi: 10.1002/tal.754. URL `http://dx.doi.org/10.1002/tal.754`.

[42] A. Kaveh, T. Bakhshpoori, and M. Ashoory. An efficient optimization procedure based on cuckoo search algorithm for practical design of steel structures. *International Journal of Optimization in Civil Engineering*, 2:1–14, 2012.

[43] A. Natarajan and S. Subramanian. Bloom filter optimization using cuckoo search. In *Proceedings of the 2012 International Conference on Computer Communication and Informatics*, India, 2012.

[44] G. Selvi and T. Purusothaman. Cryptanalysis of simple block ciphers using extensive heuristic attacks. *European Journal of Scientific Research*, 78:198–221, 2012.

[45] R. A. Vazquez. Training spiking neural models using cuckoo search algorithm. In *2011 IEEE Congress on Evolutionary Computation*, 2011.

[46] AliR. Yildiz. Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. *The International Journal of Advanced Manufacturing Technology*, 64:55–61, 2013. ISSN 0268-3768. doi: 10.1007/s00170-012-4013-7. URL `http://dx.doi.org/10.1007/s00170-012-4013-7`.

[47] X.-S. Yang and S. Deb. Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*, 1:330–343, 2010.

[48] R. B. Payne, M. D. Sorenson, and K. Kiltz. *The Cuckoos.* Oxford University Press, 2005.

[49] P. Civicioglu and E. Besdok. A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. *Artificial Intelligence Review*, pages 1–32, 2011. ISSN 0269-2821. doi: 10.1007/s10462-011-9276-0. URL `http://dx.doi.org/10.1007/s10462-011-9276-0`.

[50] I. Pavlyukevich. Lévy flights, non–local search and simulated annealing. *Journal of Computational Physics*, 226:1830–1844, 2007.

[51] G. M. Viswanathan. Lévy flights and superdiffusion in the context of biological encounters and random searches. *Physics of Life Reviews*, 5:133–150, 2008.

[52] D. Quagliarella and A. Vicini. Viscous single and multicomponent airfoil design with genetic algorithms. *Finite Elements in Analysis and Design*, 37:365–380, 2001.

[53] A. L. Marsden, M. Wang, J. E. Dennis Jr., and P. Moin. Suppression of vortex–shedding noise via derivative–free shape optimization. *Physics of Fluids*, 16, 2004.

[54] K. C. Giannakoglou, D. I. Papadimitriou, and I. C. Kampolis. Aerodynamic shape design using evolutionary algorithms and new gradient–assisted metamodels. *Computer Methods in Applied Mechanics and Engineering*, 195:6312–6329, 2006.

[55] P. I. K. Liakopoulos, I. C. Kampolis, and K. C. Giannakoglou. Grid enabled, hierarchical distributed metamodel–assisted evolutionary algorithms for aerodynamic shape optimization. *Future Generation Computer Systems*, 24:701–708, 2008.

[56] J. Periaux, D. S. Lee, L. F. Gonzalez, and K. Srinivas. Fast reconstruction of aerodynamic shapes using evolutionary algorithms and virtual Nash strategies in a CFD design environment. *Journal of Computational and Applied Mathematics*, 232:61–71, 2009.

[57] D. Wolpert and W. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82, 1997.

[58] D. Weyland. A rigorous analysis of the harmony search algorithm—how the research community can be misled by a "novel" methodology. *International Journal of Applied Metaheuristic Computing*, 1–2:50–60, 2010.

[59] N. Saino, D. Rubolini, E. Lehikoinen, L.V. Sokolov, A. Bonisoli-Alquati, R. Ambrosini, G. Boncoraglio, and A.P. Møller. Climate change effects on migration

phenology may mismatch brood parasitic cuckoos and their hosts. *Biology Letters*, 5(4):539–541, 2009.

[60] M. J. Mifsud, S. T. Shaw, and D. G. MacManus. A high–fidelity low–cost aerodynamic model using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids*, 63:468–494, 2010.

[61] M. Shatnawi and M. F. Nasrudin. Starting configuration of cuckoo search algorithm using centroidal Voronoi tessellations. In *11th International Conference on Hybrid Intelligent Systems*, Melacca, 2011.

[62] M. F. Q. Du and M. Gunzburger. Centroidal Voronoi tessellations: applications and algorithms. *SIAM Review*, 41:637–676, 1999.

[63] R. Brits, A. P. Engelbrecht, and F. van den Bergh. Locating multiple optima using particle swarm optimization. *Applied Mathematics and Computation*, 189: 1859–1883, 2007.

[64] S. Walton. Open source project; http://code.google.com/p/modified-cs/, 2011. URL `http://code.google.com/p/modified-cs/`.

[65] D. Giveki, H. Salimi, G. Bahmanyar, and Y Khademian. Automatic detection of diabetes diagnosis using feature weighted support vector machines based on mutual information and modified cuckoo search. arXiv:1201.2173, Jan 2012.

[66] J-H. Lin and H.-C. Lee. Emotional chaotic cuckoo search for the reconstruction of chaotic dynamics. In N. Mastorakis, V. Mladenov, and Z. Bojkovic, editors, *Latest Advances in Systems Science & Computational Intelligence*. WSEAS Press, 2012.

[67] A. Ghodrati and S. Lotfi. A hybrid CS/PSO algorithm for global optimization. In J-S. Pan, S-M. Chen, and N. Nguyen, editors, *Intelligent Information and Database Systems*, volume 7198 of *Lecture Notes in Computer Science*, pages 89–98. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-28492-2. doi: 10.1007/978-3-642-28493-9_11. URL `http://dx.doi.org/10.1007/978-3-642-28493-9_11`.

[68] R. N. Mantegna. Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes. *Phys. Rev. E*, 49(5):4677–4683, May 1994. doi: 10.1103/PhysRevE.49.4677. URL `http://dx.doi.org/10.1103/PhysRevE.49.4677`.

[69] J. Huston McCulloch. Stabrnd.m stable random number generator. Ohio State University Econ. Dept., 1996. URL `http://dirart.googlecode.com/svn/trunk/support_programs/edge_perserving_filters/toolbox_nlmeans/toolbox/stabrnd.m`.

[70] A. W. Iorio and X. Li. Solving rotated multi-objective optimization problems using differential evolution. In *In AI 2004: Advances in Artificial Intelligence: 17th Australian Joint Conference on Artificial Intelligence*, pages 861–872. press, 2004.

[71] A. Jameson, J.J. Alonso, J.J. Reuther, L. Martinelli, and J. C. Vassberg. Aerodynamic shape optimization techniques based on control theory. *Control Theory, CIME (International Mathematical Summer School)*, pages 21–27, 1998.

[72] K. C. Giannakoglou. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Progress in Aerospace Sciences*, 38:43–76, 2002.

[73] B. M. Kulfan and J. E. Bussoletti. "Fundamental" parametric geometry representations for aircraft component shapes. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, number AIAA—2006–6948, 2006.

[74] G. Renner and A. Ekárt. Genetic algorithms in computer aided design. *Computer Aided Design*, 35:709–726, 2003.

[75] M. Harbeck and A. Jameson. Exploring the limits of shock–free transonic airfoil design. In *AIAA 43rd Aerospace Sciences Meeting and Exhibition*, 2005.

[76] J. C. Vassberg and A. Jameson. Aerodynamic shape optimization of a Reno race plane. *International Journal of Vehicle Design*, 28:318–338, 2002.

[77] *Shape design optimization in 2D aerodynamics using Genetic Algorithms on parallel computers*, 1996. Elsevier. ISBN 978-0-44-482322-9.

[78] R. A.E. Mäkinen, J. Periaux, and J. Toivanen. Multidisciplinary shape optimization in aerodynamics and electromagnetics using genetic algorithms. *International Journal for Numerical Methods in Fluids*, 30(2):149–159, 1999. ISSN 1097-0363. doi: 10.1002/(SICI)1097-0363(19990530)30:2<149::AID-FLD829>3.0.CO;2-B.

[79] D-S Lee, J. Periaux, J. Pons-Prats, G. Bugeda, and E. Onate. Double shock control bump design optimization using hybridised evolutionary algorithms. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8, july 2010. doi: 10.1109/CEC.2010.5586379.

[80] D-S Lee, J. Periaux, E. Onate, L. F. Gonzalez, and N. Qin. Active transonic aerofoil design optimization using robust multiobjective evolutionary algorithms. *Journal of Aircraft*, 48(3):1084–1094, May 2011. URL `http://eprints.qut.edu.au/43952/`.

[81] T. R. Barrett, N. W. Bressloff, and A. J. Keane. Airfoil design and optimization using multi–fidelity analysis and embedded inverse design. In *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pages 1–21, 2006.

[82] O. Hassan, K. Morgan, and Weatherill N.P. *FLITE SYSTEM Version 4 Theoretical Manual.* Swansea University, Singleton Park, Swansea, SA2 8PP, UK, 2009.

[83] N. P. Weatherill and O. Hassan. Efficient three–dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. *International Journal for Numerical Methods in Engineering*, 37:2005–2040, 1994.

[84] X. Liu, N. Qin, and H. Xia. Fast dynamic grid deformation based on delaunay graph mapping. *Journal of Computational Physics*, 211:405–423, 2006.

[85] A. Jameson and J. Vassberg. Further studies of mesh refinement - are shock-free airfoils truly shock free? In *20th AIAA Computational Fluid Dynamics Conference.* American Institute of Aeronautics and Astronautics, 2011. doi: doi:10.2514/6.2011-3983. URL `http://dx.doi.org/10.2514/6.2011-3983`.

[86] J.H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics.* Springer, third edition, 2002.

[87] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics The Finite Volume Method second edition.* Pearson Education Limited, 2007.

[88] John Burkardt, Max Gunzburger, and Hyung-Chun Lee. POD and CVT–based reduced–order modeling of Navier–Stokes flows. *Computer Methods in Applied Mechanics and Engineering*, 196:337–355, 2006.

[89] D. Alonso, A. Velazquez, and J.M. Vega. A method to generate computationally efficient reduced order models. *Computer Methods in Applied Mechanics and Engineering*, 198:2683–2691, 2009.

[90] M. F. Barone, I. Kalashnikova, D. J. Segalman, and Heidi K. Thornquist. Stable Galerkin reduced order models for linearized compressible flow. *Journal of Computational Physics*, 228:1932–1946, 2009.

[91] F. Fang, C.C. Pain, I.M. Navon, G.J. Gorman, M.D. Piggott, P.A. Allison, P.E. Ferrell, and A.J.H. Goddard. A POD reduced order unstructured mesh ocean modelling method for moderate Reynolds number flows. *Ocean Modelling*, 28: 127–136, 2009.

[92] Xiaoning G., J. P. Dunyak, D. A. Smith, and Fuqiang W. Using projection pursuit and proper orthogonal decomposition to identify independent flow mechanisms. *Journal of Wind Engineering and Industrial Aerodynamics*, 92:53–69, 2004.

[93] P. Holmes, J.L. Lumley, and G. Berkooz. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry.* Cambridge University Press, 1996.

[94] P. G.A. Cizmas, B. R. Richardson, T. A. Brenner, Thomas J. O'Brien, and Ronald W. Breault. Acceleration techniques for reduced-order models based on proper orthogonal decomposition. *Journal of Computational Physics*, 227:7791–7812, 2008.

[95] M. Amabili and C. Touzé. Reduced-order models for nonlinear vibrations of fluid-filled circular cylindrical shells: Comparison of POD and asymptotic nonlinear normal modes methods. *Journal of Fluids and Structures*, 23:885–903, 2007.

[96] G. Kerschen, J-C. Golinval, A. F. Vakakis, and L. A. Bergman. The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: An overview. *Nonlinear Dynamics*, 41:147–169, 2005.

[97] H. V. Ly and H. T. Tran. Modeling and control of physical processes using proper orthogonal decomposition. *Mathematical and Computer Modelling*, 33:223–236, 2001.

[98] M. Bergmann and L. Cordier. Optimal control of the cylinder wake in the laminar regime by trust–region methods and POD reduced–order models. *Journal of Computational Physics*, 227:7813–7840, 2008.

[99] M. V. Tabib and J. B. Joshi. Analysis of dominant flow structures and their flow dynamics in chemical process equipment using snapshot proper orthogonal decomposition technique. *Chemical Engineering Science*, 63:3695–3715, 2008.

[100] Y. Utturkar, B. Zhang, and W. Shyy. Reduced–order description of fluid flow with moving boundaries by proper orthogonal decomposition. *International Journal of Heat and Fluid Flow*, 26:276–288, 2005.

[101] J. Rambo and Y. Joshi. Reduced–order modeling of turbulent forced convection with parametric conditions. *International Journal of Heat and Mass Transfer*, 50: 539–551, 2007.

[102] J. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM Journal on Scientific Computing*, 11:873–912, 1990.

[103] S.S. Ravindran. Optimal boundary feedback flow stabilization by model reduction. *Computer Methods in Applied Mechanics and Engineering*, 196:2555–2569, 2007.

[104] R. Zimmermann and S. Görtz. Improved extrapolation of steady turbulent aerodynamics using a non–linear POD–based reduced order model. *The Aeronautical Journal*, 116(1184):1079–1100, 2012.

[105] D. My-Ha, K.M. Lim, B.C. Khoo, and K. Willcox. Real–time optimization using proper orthogonal decomposition: free surface shape prediction due to underwater bubble dynamics. *Computers & Fluids*, 36:499–512, 2007.

[106] A. Qamar and S. Sanghi. Steady supersonic flow–field predictions using proper orthogonal decomposition technique. *Computers & Fluids*, 38:1218–1231, 2009.

[107] Y. Wang, B. Yu, Z. Cao, W. Zou, and G. Yu. A comparative study of POD interpolation and POD projection methods for fast and accurate prediction of heat

transfer problems. *International Journal of Heat and Mass Transfer*, 55(17–18): 4827 – 4836, 2012. ISSN 0017-9310. doi: 10.1016/j.ijheatmasstransfer.2012.04.053.

[108] E. Bouhoubeiny and P. Druault. Note on the POD–based time interpolation from successive PIV images. *Comptes Rendus Mécanique*, 337:776–780, 2009.

[109] T. J. Mackman and C. B. Allen. Investigation of an adaptive sampling method for data interpolation using radial basis functions. *International Journal for Numerical Methods in Engineering*, 83:915–938, 2010.

[110] C. W. Rowley, T. Colonius, and R. M. Murray. Model reduction for compressible flows using POD and Galerkin projection. *Physica D*, 189:115–129, 2004.

[111] D. J. Lucia and P. S. Beran. Projection methods for reduced order models of compressible flows. *Journal of Computational Physics*, 188:252–280, 2003.

[112] T. Lieu, C. Farhat, and M. Lesoinne. Reduced–order fluid/structure modeling of a complete aircraft configuration. *Computer Methods in Applied Mechanics and Engineering*, 195:5730–5742, 2006.

[113] C. L. Pettit and P. S. Beran. Application of proper orthogonal decomposition to the discrete Euler equations. *International Journal for Numerical Methods in Engineering*, 55:479–497, 2002.

[114] J. Degroote, J. Vierendeels, and K. Willcox. Interpolation among reduced–order matrices to obtain parameterized models for design, optimization and probabilistic analysis. *International Journal for Numerical Methods in Fluids*, 63:207–230, 2010.

[115] J. S. R. Anttonen, P. I. King, and P. S. Beran. Applications of multi–POD to a pitching and plunging airfoil. *Mathematical and Computer Modelling*, 42:245–259, 2005.

[116] P. D. Ledger, J. Peraire, K. Morgan, O. Hassan, and N. P. Weatherill. Parameterised electromagnetic scattering solutions for a range of incident wave angles. *Computer Methods in Applied Mechanics and Engineering*, 193:3587–3605, 2004.

[117] O. Hassan, K. Morgan, and N. Weatherill. Unstructured mesh methods for the solution of the unsteady compressible flow equations with moving boundary components. *Philosophical Transactions of the Royal Society A*, 365:2531–2552, 2007.

[118] K. A. Sørensen, O. Hassan, K. Morgan, and N. P. Weatherill. A multigrid accelerated time–accurate inviscid compressible fluid flow solution algorithm employing mesh movement and local remeshing. *International Journal for Numerical Methods in Fluids*, 43:1207–1229, 2003.

[119] J. S. R. Anttonen, P. I. King, and P. S. Beran. POD–based reduced–order models with deforming grids. *Mathematical and Computer Modelling*, 38:41–62, 2003.

[120] F. Fang, C. C. Pain, I. M. Navon, M. D. Piggott, G. J. Gorman, P. E. Farrell, P. A. Allison, and A. J. H. Goddard. A POD reduced-order 4D–Var adaptive mesh ocean modelling approach. *International Journal for Numerical Methods in Fluids*, 60:709–732, 2009.

[121] P. Mokhasi and D. Rempfer. Nonlinear system identification using radial basis functions. *International Journal for Numerical Methods in Fluids*, 63:121–162, 2010.

[122] S. Rippa. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Advances in Computational Mathematics*, 11:193–210, 1999.

[123] R. Franke. Scattered data interpolation: Tests of some method. *Mathematics of Computation*, 38(157):181–200, 1982.

[124] L. Nguyen V. Golubev, T. Hollenshade and M. Visbal. Parametric viscous analysis of gust interaction with sd7003 airfoil. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, number AIAA 2010-928, 2010.

[125] F. Chinesta, A. Ammar, A. Leygue, and R. Keunings. An overview of the proper generalized decomposition with applications in computational rheology. *Journal of Non-Newtonian Fluid Mechanics*, 166(11):578 – 592, 2011. ISSN 0377-0257. doi: 10.1016/j.jnnfm.2010.12.012. URL `http://www.sciencedirect.com/science/article/pii/S0377025711000061`.

[126] F. H. Harlow and J. E. Welch. Numerical calculation of time–dependent viscous incompressible flow of fluid with free surface. *The Physics of Fluids*, 8:2182–2189, 1965.

[127] K. Yee. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, 14:302–307, 1966.

[128] Z. Q. Xie, O. Hassan, and K. Morgan. Tailoring unstructured meshes for use with a 3d time domain co-volume algorithm for computational electromagnetics. *International Journal for Numerical Methods in Engineering*, 87(1-5):48–65, 2011. ISSN 1097-0207. doi: 10.1002/nme.2970. URL `http://dx.doi.org/10.1002/nme.2970`.

[129] I. Sazonov, O. Hassan, K. Morgan, and N. P. Weatherill. Smooth Delaunay–Voronoi dual meshes for co–volume integration schemes. In P. P. Rebay, editor, *Proceedings of the 15th International Meshing Roundtable*, pages 529–541, Berlin, 2006. Springer.

[130] I. Sazonov, D. Wang, O. Hassan, K. Morgan, and N. Weatherill. A stitching method for the generation of unstructured meshes for use with co–volume solution techniques. *Computer Methods in Applied Mechanics and Engineering*, 195:1826–1845, 2006.

[131] J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz. Adaptive remeshing for compressible flow computations. *Journal of Computational Physics*, 72:449–466, 1987.

[132] K. Morgan, J. Peraire, and J. Peiro. Unstructured grid methods for compressible flows. In *Report 787: Special Course on Unstructured Grid Methods for Advection Dominated Flows*, pages 5.1–5.39, Paris, 1992. AGARD.

[133] J. Park and S. M. Shontz. Two derivative-free optimization algorithms for mesh quality improvement. *Procedia Computer Science*, 1:387–396, 2010.

[134] Z. Cui, X. Cai, J. Zeng, and G. Sun. Particle swarm optimization with FUSS and RWS for high dimensional functions. *Applied Mathematics and Computation*, 205: 98–108, 2008.

[135] A. Yilmaz and M. Kuzuoglu. A particle swarm optimization approach for hexahedral mesh smoothing. *International Journal for Numerical Methods in Fluids*, 60: 55–78, 2009.

[136] S. P. Sastry, S. M. Shontz, and S. A. Vavasis. A log-barrier method for mesh quality improvement. In *Proceedings of the 20th International Meshing Roundtable*, 2011.

[137] Q. Du and D. Wang. Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations. *International Journal for Numerical Methods in Engineering*, 56:1355–1373, 2003.

[138] Y. Ohtake, A. Belyaev, and I. Bogaevski. Mesh regularization and adaptive smoothing. *Computer–Aided Design*, 33:789–800, 2001.

[139] D. J. Naylor. Filling space with tetrahedra. *International Journal for Numerical Methods in Engineering*, 44:1383–1395, 1999.

[140] D. Eppstein, J. M. Sullivan, and A. Üngör. Tiling space and slabs with acute tetrahedra. *Computational Geometry: Theory and Applications*, 27:237–255, 2004.

[141] I. Sazonov, O. Hassan, K. Morgan, and N. P. Weatherill. Generating the Voronoi–Delaunay dual diagram for co–volume integration schemes. In C. M. Gold, editor, *4th International Symposium on Voronoï Diagrams in Science and Engineering*, pages 199–204. IEE CPS, 2007.

[142] S. Lloyd. Least square quantization in the PCM. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[143] E. VanderZee, A. Hirani, D. Guoy, and E. Ramos. Well-centered planar triangulation – an iterative approach. In *Proceedings of the 16th International Meshing Roundtable*, pages 121–138. SpringerLink, 2008. Session 1B.

[144] B. M. Klingner and J. R. Shewchuk. Aggressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable*, 2007.

[145] L. A. Freitag and P. Plassmann. Local optimization–based simplicial mesh untangling and improvement. *International Journal for Numerical Methods in Engineering*, 49:109–125, 2000.

[146] K. Morgan, O. Hassan, and J. Peraire. An unstructured grid algorithm for the solution of Maxwell's equations in the time domain. *International Journal for Numerical Methods in Fluids*, 19:849–863, 1994.

[147] K. Morgan, O. Hassan, and J. Peraire. A time domain unstructured grid approach to the simulation of electromagnetic scattering in piecewise homogeneous media. *Computer Methods in Applied Mechanics and Engineering*, 134:17–36, 1996.

[148] R. Pritchard, O. Hassan, and K. Morgan. An efficient marker and cell solver for unstructured hybrid meshes. In W. A. Wall and V. Gravemeier, editors, *Proceedings of the 16th International Conference on Finite Elements in Flow Problems*, page 127, Munich, 2011.

[149] J. Schöberl. NETGEN: An Advancing Front 2D/3D-Mesh Generator Based on Abstract Rules. *Computing and Visualization in Science*, 1:41–52, 2007.

[150] P. Alliez, M. Cohen-Steiner, D.and Yvinec, and M. Desbrun. Variational tetrahedral meshing. *ACM Transactions on Graphics, special issue on Proceedings of SIGGRAPH*, 24:617–625, 2005.

[151] S. Walton, O. Hassan, and K. Morgan. Selected engineering applications of gradient free optimisation using cuckoo search and proper orthogonal decomposition. *Archives of Computational Methods in Engineering*, 20(2):123–154, 2013. ISSN 1134-3060. doi: 10.1007/s11831-013-9083-7. URL `http://dx.doi.org/10.1007/s11831-013-9083-7`.

[152] S. Walton, M. R. Brown, O. Hassan, and K. Morgan. Comment on "Cuckoo search: A new nature-inspired optimization method for phase equilibrium calculations" by V. Bhargava, S. Fateen, A. Bonilla-Petriciolet [Fluid Phase Equilibria 337 (0) (2013) 191 - 200]. *Fluid Phase Equilibria*, 2013. doi: 10.1016/j.fluid.2013.05.011.

[153] S. Walton, O. Hassan, K. Morgan, and M. R. Brown. The development and applications of the cuckoo search algorithm. In X-S. Yang, Cui. C., R. Xiao, A. H. Gandomi, and M. Karamanoglu, editors, *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, pages 257–271. Elsevier, September 2013. doi: http://dx.doi.org/10.1016/B978-0-12-405163-8.00011-9.

[154] S. Walton, O. Hassan, and K. Morgan. Reduced order modelling of compressible flow past an airfoil using proper orthogonal decomposition. In A. Zervos, editor, *Proceedings of ACME2010: the 18th Annual Conference of the Association of Computational Mechanics in Engineering*, 2010.

[155] S. Walton, O. Hassan, and K. Morgan. Using proper orthogonal decomposition to reduce the order of optimization problems. In W. A. Wall and V. Wall, W. A., editors, *Proceedings of the 16th International Conference on Finite Elements in Flow Problems*, 2011.

[156] S. Walton, O. Hassan, and K. Morgan. Interpolating unsteady paramaterised CFD solutions using radial basis functions and proper orthogonal decomposition. In *European Congress on Computational Methods in Applied Sciences and Engineering*, 2012.