UNIVERSITY OF SOUTHAMPTON

Applications and Enhancements of Aircraft Design Optimization Techniques

by

Stephen R. Powell

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Engineering and the Environment
School of Engineering Sciences

June 2012

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

Faculty of Engineering and the Environment

SCHOOL OF ENGINEERING SCIENCES

Doctor of Philosophy

Applications and Enhancements of Aircraft Design Optimization Techniques

by Stephen R. Powell

The aircraft industry has been at the forefront in developing design optimization strategies ever since the advent of high performance computing. Thanks to the large computational resources now available, many new as well as more mature optimization methods have become well established. However, the same cannot be said for other stages along the optimization process - chiefly, and this is where the present thesis seeks to make its first main contribution, at the geometry parameterization stage.

The first major part of the thesis is dedicated to the goal of reducing the size of the search space by reducing the dimensionality of existing parameterization schemes, thus improving the effectiveness of search strategies based upon them. Specifically, a refinement to the Kulfan parameterization method is presented, based on using Genetic Programming and a local search within a Baldwinian learning strategy to evolve a set of analytical expressions to replace the standard 'class function' at the basis of the Kulfan method. The method is shown to significantly reduce the number of parameters and improves optimization performance - this is demonstrated using a simple aerodynamic design case study.

The second part describes an industrial level case study, combining sophisticated, high fidelity, as well as fast, low fidelity numerical analysis with a complex physical experiment. The objective is the analysis of a topical design question relating to reducing the environmental impact of aviation: what is the optimum layout of an over-the-wing turbofan engine installation designed to enable the airframe to shield near-airport communities on the ground from fan noise. An experiment in an anechoic chamber reveals that a simple half-barrier noise model can be used as a first order approximation to the change of inlet broadband noise shielding by the airframe with engine position, which can be used within design activities. Moreover, the experimental results are condensed into an acoustic shielding performance metric to be used in a Multidisciplinary Design Optimization study, together with drag and engine performance values acquired through CFD. By using surrogate models of these three performance metrics we are able to find a set of non-dominated engine positions comprising a Pareto Front of these objectives. This may give designers of future aircraft an insight into an appropriate engine position above a wing, as well as a template for blending multiple levels of computational analysis with physical experiments into a multidisciplinary design optimization framework.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Stephen Robert Powell, declare that the thesis entitled *"Applications and Enhancements of Aircraft Design Optimization Techniques"* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: Powell and Sóbester [2010], Sóbester and Powell [2012], Powell et al. [2011], Powell et al. [2012a], Powell et al. [2012b].

Signed:................................ Date:................................

Acknowledgements

There are a great many people that I am indebted to in creating this thesis, none moreso than my supervisor, Dr. András Sóbester. Three years ago, he took me on under his wing, providing me with the guidance and encouragement for completing this thesis. He has gone far beyond the call of duty, giving financial and moral backing, and was always available, even if it was just for a chat.

A special thank you must go to Prof. Philip Joseph, my second supervisor, for his guidance in the engine noise study, as well as for checking the numerous revisions of the published papers. I must also thank Matieau Gruber and John Fithyan of ISVR for their advice and assistance in the design and undertaking of the noise experiment presented in this thesis.

Thanks should go to Ms Julie Chueng, Mr Stuart Jinks and Mr Alex Purdue for the support they have given me, as well as each other, often from breaks incited from the one word e-mail “coffee?”. And to Mr Matthew Palmer, who usually turned up, eventually.

Ms Rosalind Mizen deserves special thanks for her assistance in administrative tasks, but also for her kind and friendly attitude, not to mention for looking after everyone in the Computational Engineering and Design Group. Also, Dr David Toal, Dr Giles Endicott, Mr Sanjay Pant, Mr Moresh Wankheed, Mr James Parr, and Dr Athanasios Makrodimopoulos all contributed toward the friendly office atmosphere, and so all deserve special thanks. I must include in that group Mr James Meas, a regular traveller to the office. His boxes of broken biscuits were always appreciated.

Of course, no acknowledgement page would be complete without special thanks given to my Mother and Father. They have been supportive throughout my education, and will no doubt continue to be through the rest of my career. Thanks also go to my two brothers, Andrew and Richard, whose sarcastic comments of ‘have you not finished yet?’ seem to never get old.

Finally, I would like to thank Ms Kimberley Knowler for her support, understanding and for generally putting up with me whilst I wrote this thesis.

Nomenclature

Greek Characters

α	Angle of attack
Δ	Noise shielding metric
ϵ	Hyperparameter governing the rate of correlation decay
λ	Wave length
λ_r	Regression parameter
μ	Mean
σ^2	Variance
ζ	Slack variables

Roman Characters

a	Radius of duct
B	Bernstein polynomial
c	Chord
C	Class function
C_D	Drag coefficient
c_d	Cross-section drag coefficient
C_L	Lift coefficient
c_l	Cross-section lift coefficient
c_p	Pressure coefficient
c_y	Wing chord at the engine spanwise position
DC(60)	Distortion Coefficient, based on a 60^{o} sector angle

F	Fitness function
f_c	One-third octave center frequency
f	Frequency
$f(x)$	Objective function of $\mathbf{x}$
g	Equality constraint
h	Nacelle diameter
$\mathbf{I}$	Identity matrix
k	Free space wave number
L	Likelihood
$N(u)$	B-Spline basis function
M_{dd}	Drag divergence Mach number
$M_{dd,K}$	Drag divergence Mach number, according to the korn equation
N	Fresnel number
n	The number of design variables
$N1$, $N2$	Exponents of the Kulfan transformation
P	Penalty function
p	Pressure
$\mathbf{p}$	Vector of control points
$\bar{p}_c^2$	Mean square pressure recorded without aircraft in anechoic chamber
$\bar{p}_s^2$	Mean square pressure recorded with aircraft in anechoic chamber
P_t	Total pressure
q	Hyperparameter governing the degree of smoothness
$\mathbf{R}$	Correlation matrix
Re	Reynolds number
S	Shape function
t/c	Thickness-to-chord ratio
U	Knot vector

w	Weight element
$\mathbf{X}$	A matrix containing a set of input sets $\mathbf{x}$
x	Design variable
$\mathbf{x}$	Design variable set
y	Function output
$\mathbf{z}_a$	Approximated z-coordinate set
$\mathbf{z}_t$	Target z-coordinate set

Chapter 1

Introduction

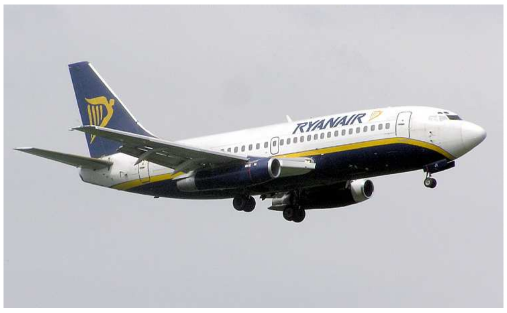
1.1 Optimization in Engineering Design

The current financial turmoil that has engulfed the world's most affluent nations, along with the acceptance of human contribution to climate change, has resulted in political leaders putting more emphasis on improving the economy and reducing the effect we have on the environment. Air travel affects, and is affected by, both issues; there is intense media coverage of the current rate of increased air travel as the world becomes more accessible to expanding multi-national companies, leading to increased fuel emissions that forces a greater influence on climate change. With the economic future uncertain for many countries around the world there is an increased need to reduce ticket costs. Due to the ever increasing cost of oil, and the fines issued by airports to aircraft that fail to meet their environmental emission targets, average ticket prices are set to increase further. This has led to aircraft manufactures having to find solutions to stricter aircraft design requirements in the areas of fuel emissions, efficiency, and noise reductions.

Solutions to these design drivers could be achieved by a radical rethink in the design, but we may also be able to achieve significant advances by making subtle changes to current configurations. These two mind-sets are visually seen in the design evolution of Aircraft and Formula 1 racing cars (Figure 1.1); major civil airline manufacturers have typically refined the tube-fuselage/wing concept, whilst Formula 1 has gone through numerous revolutions in the past 60 years, due to the fast introduction of new parts that vastly improve the car's performance. Formula 1 teams do not solely rely on the knowledge and flair of their engineers to create radically different parts. They too go through the process of iteratively improving the designs until the *'optimal'* solution is found. This fundamental difference in mind-set is not due to the lack of creativity of the airline manufacturer's engineers, but due to the differences in requirements between the two industries; Formula 1 companies are striving to win races, whereas airline manufacturers are looking to make a profit. Many of the new ideas in civil aircraft design that are

(a) Lotus 78 F1 car

(b) McLaren's 2010 F1 car


(c) Boeing 737-200

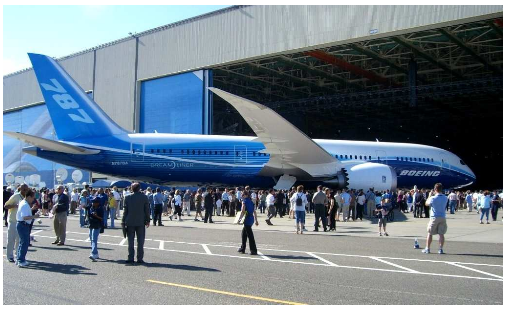

(d) Boeing 787 in 2007

FIGURE 1.1: The effect of increased use of computational modelling in design; the external shape of Formula 1 cars have changed drastically, whilst the influence on civil aircraft is more subtle.

publicised in aircraft journals and presented at conferences are considered too much of a financial risk to the airline manufactures, mainly due to the development costs, but also due to the probability of success.

This thesis presents and improves upon methods and tools used to optimize aircraft. The term 'optimization' is sometimes used to describe a structured process that we use to improve upon a current design. Until recently, the process involved was not always systematic, and generally centred around a trial and error approach. Nowadays, with the aid of modern supercomputing clusters and computational analysis tools, highly effective and time efficient optimization strategies exist that require little input from the design engineer. Many different optimization methodologies exist, all with their own strengths and weaknesses, and so the user must have an understanding of all available to him/her to be able to choose the most appropriate technique for the problem at hand. A weakness common in many optimization applications is that the time taken to find a solution can be long - we target this in the work presented in this thesis.

An important stage in the formulation of any optimization process is in the identification of the main variables (or parameters) in the problem and modelling them appropriately, which is termed the parameterization of the problem. Typical examples include Euclidean positions of components and their dimension values. A more difficult parameterization problem involves the definition of the shapes of the components, for example, wing cross sections or engine intake geometry. The complexity of these shapes can, at this point in time, only be modelled by methods that use a set of parameters. However, increasing the design space, that is, increasing the number of parameters, has a detrimental effect on the time taken to find a solution. For example, consider a parameter set with n parameters. If we wished to sample every parameter d times, we would need to evaluate d^n designs. If the analysis of each design took t hours to compute then the final solution would take td^n hours to find – the time taken to find a solution is of the order d^n. Optimization methods are often discounted due to computational budget constraints, and so the size of the parameter set n is an important factor in the setup of an optimization problem.

Along with the requirement to limit the number of parameters used to describe shapes, a parameterization model must also be able to take the form of a variety of different shapes. At worst, the model must be able to generate all existing designs of the component in question. However, the flexibility of the model must be restricted somewhat – we must ensure that for any given parameter set, the model produces physically viable shapes; for example, in most cases we cannot have surfaces crossing each other. In summary, optimization requires a shape parameterization method that can accurately approximate a variety of physically viable shapes whilst restricting the number of parameters. The limited research currently being published in this area and the potential performance improvements that can be made in this field, as discuss above, has been the motivation of part of the work presented in this thesis.

Within the many different fields of the aerospace industry, developments are being made to computational analysis tools that can be used within an optimization framework. The development of such methods has been assisted by an increase in computational power, as well as the ability to use parallel computing clusters. For example, RANS computational fluid dynamics (CFD) solvers are widely used today, whereas before lower fidelity potential solvers were the only realistic tools that could be applied. However, some of these prediction methods are still not of the required degree of fidelity, and so physical experiments still have their role to play in preliminary design. This causes a significant difficulty in a design optimization setting, with a physical experiment unable to be coupled within an automated optimization process.

Although numerical simulations have become more popular as design analysis tools, they can still significantly prolong the optimization process, with some simulations taking several hours to complete. This effect may prohibit the use of some optimization methods that require a large number of function evaluations. However, surrogate models provide a relatively cheap alternative for finding an optimal design: a surrogate essentially models the performance output for a given set of parameter inputs. The surrogate, built on a set of carefully chosen sample points where the expensive analysis was run, thus becomes a cheap substitute of the expensive numerical simulation. We then search this model for the optimal solution. Of course, we still require to perform a set of numerical analyses, but the amount is minute compared to that of, say, a global optimizer.

In 'real world' design problems the ability of a component to perform well is dependent upon multiple characteristics. For example, we may require a component to possess a high tensile strength whilst also being lightweight. Moreover, the important design characteristics are more often than not topics within different engineering disciplines. In these situations we must look towards using a Multidisciplinary Design Optimization (MDO) approach to find the most appropriate design solution. This involves the coupling of performance disciplines in an effort to find a balance between all competing goals, as opposed to a sequential iterative procedure between the disciplines. The difficulties in performing multi-disciplinary case studies of more than two disciplines, as well as the restrictions of using physical experiments in optimization, are addressed in this thesis using surrogate modelling.

1.2 Thesis Overview

The performance of an optimization procedure depends upon the performance of the various techniques applied at different stages along the optimization process. The overall performance can be categorised not only by the quality of the final solution, but also the time taken to find the solution. The goals tackled in this thesis concentrate on the latter, without hindering the former.

Firstly, we pursue the need for reducing the parameter space without the loss of flexibility in creating application-relevant shapes. For this we take inspiration from Kulfan's Class-Shape transformation, which is a relatively new parameterization model that meets the aforementioned parametric model attributes well. It essentially separates the problem by selecting a suitable analytical class function, then a parameterization model adds the specific detail. Kulfan suggests a general form class function that can be used for different applications (i.e. airfoils, dart, etc.) by modification of a pair of variables. However, if the class function chosen is not specifically applicable to the problem then it is likely that a large number of parameters will be required in the parameterization model to accurately define the existing set of designs. We therefore look into a possible methodology that can define application specific class functions to aid this parameterization process.

Secondly, we introduce an original aircraft optimization case study: we investigate the potential of an alternative aircraft configuration as an answer to the growing concern of aircraft noise emissions. Installing the engines so that they are positioned above the wing is considered to give the potential ability to use the airframe as a noise shield between the engine and on-the-ground communities. Initially, we investigate this by performing a shielding study based on a physical experiment to obtain the performance of each design. However, engine integration is dependent upon a number of other performance factors, making it a candidate for an MDO study. Therefore, we also investigate the aerodynamic performance as well as engine performance, and consider all three performance metrics in an MDO setting.

Chapters 2 and 3 give an overview of design optimization and parameterization respectively. These chapters are aimed to give a brief introduction to aircraft design optimization, giving a flavour of some of the most common methods, as well as introducing those used subsequently in this thesis.

The main aim of Chapter 4 is to present a simple optimization problem on a 2D airfoil that underpins the basic design optimization process. After introducing NASA's SC(2) airfoil family, and presenting a method that finds the design conditions of these airfoils, we optimize a member of the SC(2) family for flow conditions away from those that the airfoil was designed for.

Chapter 5 is dedicated to improving the Kulfan transformation parameterization via a method that generates application specific class functions. The significance of this method is shown by comparing the results of using the Kulfan class function and the generated class functions in an optimization procedure similar to the one presented in Chapter 4.

In Chapter 6 we present the over-wing engine installation case study, along with the noise experiment performed in the University of Southampton's large anechoic chamber. We find the optimal position of the engines above the wing in terms of noise shielding performance, and present a first order approximation model built around simple half-plane barrier theory. Chapter 7 presents the MDO study on the over-wing engine installation, based on generating surrogate models of the individual objectives and using them as a quick evaluation of each performance metric in the generation of the Pareto optimal solutions.

We complete this thesis in Chapter 8 by drawing conclusions on what has been achieved, as well as giving potential avenues of further work within the realm of aircraft design optimization.

Chapter 2

Optimization in Aircraft Design

2.1 A Review of Optimization

To correctly and efficiently use optimization strategies in aircraft design, one must not only understand the optimization methods, but every stage along the optimization process. Figure 2.1 presents a flowchart of a basic optimization procedure, showing the main stages in this process. The first stage is the parameterization of the problem. This is essentially the mathematical formulation used to define designs using a unique set of variables. Moreover, modification of these variables must generate a variety of different designs. It is very important to create an accurate parameterization model, as the performances of all the proceeding stages are dependent upon this first stage. This will be further discussed in Chapter 3. Some optimization techniques require a population of initial designs, with others only needing a single design to start the optimization loop. In the present discussion here we assume that only a single design is required in the first instance. However, we will discuss, later in this chapter, strategies that require numerous designs to initialise the optimization cycle, and we introduce methods that aid us in choosing them.

Once we have decided on the initial design, defined by its unique variable set, we can evaluate its performance. In preliminary design nowadays, designs are usually evaluated using a set of computationally expensive simulations that describe the physics of the problem. Examples include Computational Fluid Dynamics (CFD) and Finite Element Analysis (FEA). Physical experiments, such as wind tunnel testing, are rarely used in this respect primarily due to the inability to automate the optimization loop, as human interaction is needed to swap, calibrate, and test the models. Note that physical experiments are still essential, as a means to validate the idealised solutions of the numerical analyses.

For the optimizer to understand the difference between designs that perform well or poorly we must quantify the performance of designs using an objective function, which is based upon the results taken from the performance analysis module. From the value of the objective function the optimizer decides whether to change the parameter values, as well as what to change these values to, and iterate over the process again, or to regard the current solution good enough to be chosen as the final optimal design. This decision is usually made with reference to a convergence criterion of the optimization process that is usually some residual value from the optimizer, rate of change in the solution, or the time expired from when the optimization process began. It can also be some target value that the optimizer attempts to match.

We can describe the optimization procedure as

$$\begin{array}{lll} \textbf{min} & f(\mathbf{x}) & \\ \text{subject to} & g_i(\mathbf{x}) = c_i & \text{for } i = 1, \ldots, n \\ & h_j(\mathbf{x}) \leq d_j & \text{for } j = 1, \ldots, m. \end{array} \tag{2.1}$$

In words, it is the requirement to minimize (or maximize) an objective function ($f(\mathbf{x})$) by modifying a set of inputs ($\mathbf{x}$) within a set of inequality ($h(\mathbf{x})$) and equality ($g(\mathbf{x})$) constraints. These constraints will be discussed later, but their job is regarded as restricting the range of the input variables, either directly or indirectly, through using other functions.

We often describe the optimization process as a 'search' for the optimal solution; given a set of bounds to the inputs of a problem, we can assume that the optimal solution is out there within these variable bounds, and so the job of the optimizer is to search for the optimal solution within these bounds. There are a vast number of optimization techniques available, all with different capabilities and restrictions. There is no single technique that is superior for all situations, and so the user must have an underlying understanding of the different types so the one most suitable for the application is selected. Misusing these optimization methods can lead to a long wait for the final solution. Moreover, it can fool the user by returning an apparently optimal solution, when, in fact, better solutions exist. This could result in designers wrongfully dismissing

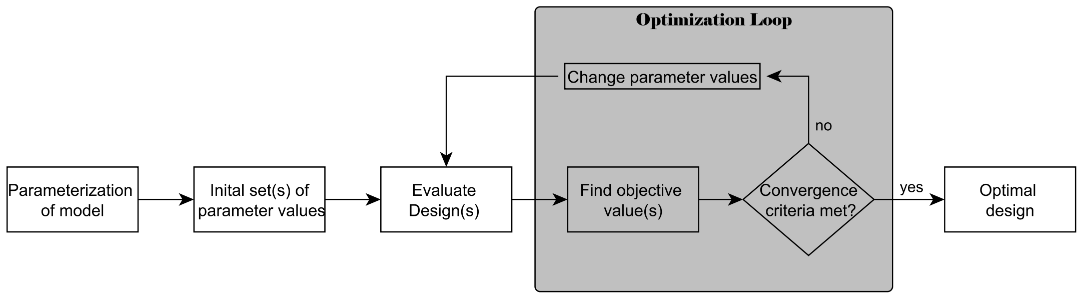


Figure 2.1: Flow chart of a typical optimization process.

significant design changes when in fact those changes, if optimized in the right manner, could lead to improved performance.

In this section we review the optimization techniques by grouping them into two different types: local and global. The optimization process is notoriously slow, and so the never ending desire to reduce the time from start to finish has led to the branch of optimization that uses surrogate models to approximate the objective function in a predefined variable space. We describe surrogate modelling as a type of machine learning, where a computational model learns the mapping between the design variables and the objective function. We also consider other techniques that belong to the machine learning family: Genetic Programming, which hails from a specific set of global optimizers, and Support Vector Classifiers, used to find optimal boundaries to separate members of different classes. We discuss other aspects in optimization that are as important to understand as the optimizers themselves in order to achieve the best solution. These include defining the objective function, the constraints in the problem, and penalty functions. We also discuss the situation where we have more than one objective, and have to trade the best solution of an objective for consideration of others.

2.2 Local Optimization

When discussing optimization with a non-specialist, the general description given is that the inputs to a system are altered in a systematic fashion to improve upon the outputs of that system. A slight alteration or 'tinkering' can be regarded as a *local optimization process*, whereby we search the neighbourhood of an initial set of inputs to improve upon the system outputs. Historically, local optimizers have used gradient information of the input/output relationship as a means to target the most appropriate direction to search. For multivariable optimization this involves finding the gradients of a function with respect to all the inputs when deciding the next step. The simplest technique is the steepest descent method [Press et al., 2007], where the new approximation can be found by:

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \lambda_i^* \nabla f(\mathbf{x}_i) \tag{2.2}$$

where $\nabla f(\mathbf{x}_i)$ is the gradient at the previous step $\mathbf{x}_i$, and λ_i^* is the optimal step length.

These first order optimization methods can be impossible to use if affordable gradient information is not available (finite difference gradients are expensive to compute and their cost rises with the dimensionality of the problem), and so zeroth order methods also exist that do not have the need to compute gradients. Methods such as Hooke and Jeeves [1960] and Nelder and Mead [1965] use a set of heuristics or rules to try to home in on the local optimum.

2.2.1 The Simplex Method

In Chapters 4 and 5 we use MATLAB's `fminsearch` command that utilises the Nelder and Mead [1965] algorithm. This method uses a simplex (a construction in n dimensional space that has $n+1$ vertices where, n equals the number of input variables) and modify it to search the local domain. There are four moves we can take to modify the simplex to home in on our target minimum;

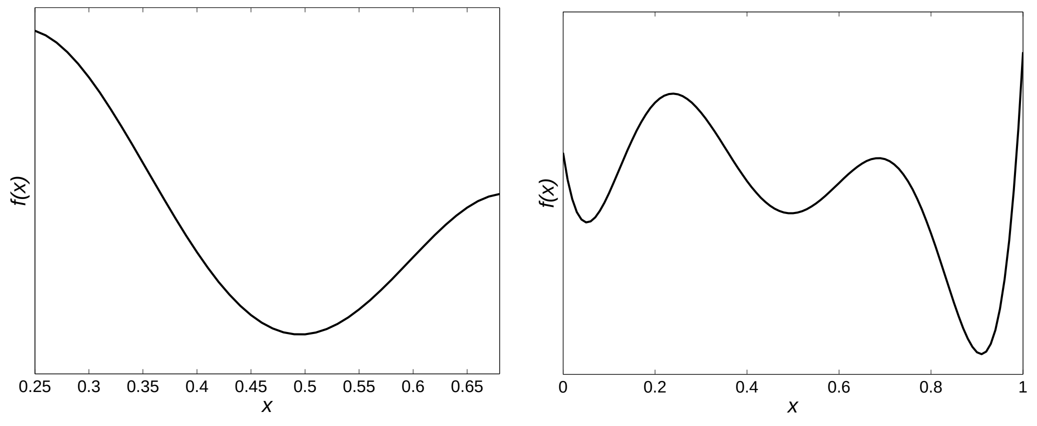
1. Reflection - The worst vertex is reflected through the face opposite.
2. Reflection and Expansion - The worst vertex is reflected through the opposite face but the point is placed further away from that face.
3. Contraction - The vertex is pulled toward the face opposite it, but not as far as to pull it through the face, reducing the volume.
4. Multiple contraction - A face is pulled toward the vertex opposite, reducing the volume.

To make these movements the algorithm must initially rank the vertices, from best to worst. It then compares them to a point opposite the worst point outside the simplex. Then the following tests are applied in order to decide which move to make;

1. if the solution at the trial point lies between the best and the next best it is accepted and the simplex is reflected.
2. if the solution is better than the best point then the worst point is not only reflected but expanded in this direction as well, unless the expansion fails and then it is just reflected.
3. if the solution is poorer at this point, the new trial point is found along the line joining the worst and the trial point. If this point is better than the worst vertex point it is accepted and the simplex is contracted, if not then all points apart from the best are contracted.
4. otherwise the trial point must be better than the worst so it replaces the worst and a contraction is also made.

The process terminates when the simplex reduces to a predefined minimum size or when the difference between the objective function values of the vertices is below a specified tolerance.

Local search methods are generally very efficient and accurate at finding a local solution from an initial starting point. Problems exist in using these techniques when we want



(a) A single trough of $f(x)$.

(b) A multimodal objective function containing multiple minima.

FIGURE 2.2: An objective function landscape for a one variable problem at different variable ranges.

to search a predefined range of inputs, where the objective function landscape may have multiple local minima. This inability is depicted in Figure 2.2; if we restrict the search to be between $x = 0.25$ and $x = 0.65$, as in Figure 2.2(a), a local search will be very effective in finding the optimal value. However, if we increase the range of our search, shown in Figure 2.2(b), a local search will likely become stuck in one of the local optima, with the performance of a local search being dependent on its start position. Of course we can use a sampling algorithm to give multiple starting points and run local searches at each point, but this may not be the most efficient technique. The performance of many local searches is also dependent on the step sizes they make between iterations: too large and the optimizer may jump over the minimal point into another minimum; too small, and the efficiency of the search decreases and we may get stuck in a minimal trough caused by noisy data.

2.3 Global Optimization

Due to the limitations of local searches on multi-modal optimization problems, we must look to use a different set of optimization techniques – ones that are specifically designed to search a global input space (Figure 2.2(b)). Global optimization techniques generally involve a stochastic approach, where the random nature of the algorithms can lead to different optimal solutions for different runs on the same problem. Conversely, the local methods discussed earlier usually have a deterministic nature and always give the same solution when using the same set of initial inputs.

Two of the most popular stochastic techniques, Simulated Annealing (SA) and Genetic Algorithms (GA), involve populating the input space with a collection of different variable sets, or designs, and using the sets that perform well to direct the search further. A new collection of variable sets are produced that have been generated from information gathered from the previous sets. This process is iterated over until the convergence criteria are met. In comparison to local searches, global techniques have the ability to leave local minima in search of the global minimum. However, the precision of the final solution can be low along with the convergence rate of the solution.

The disadvantages of a global stochastic method can be combated by a hybrid global-local search process; initially a stochastic method is used to find a set of minima, then a local search is used to home in on the exact minimum of each set. This increases precision and the convergence rate increases as we only require the global search to find minimal regions, and not the exact solution.

2.3.1 Genetic Algorithms (GAs)

GA's take their inspiration from the Darwinian theory of natural selection. That is, genetic information is passed on from parents to child, with the parents chosen in terms of "survival of the fittest"; an instance (or a design in our case) in a population is more likely to contribute to the next generation if it has good fitness, where the fitness is usually given by some scaled version of the objective function. An instance with a poor fitness is still given a chance to contribute, as it may have some underlying information that we may lose if we discount it. Using this process the new instances, or children, in a population are likely to attain characteristics similar to its parents. Thus, by encouraging the process to pick parents with characteristics that provide good fitness, the offspring should possess an overall better fitness than their parents.

A common and simple method for parent selection is the idea of a roulette wheel, whereby the selection is random but the largest area on the roulette is taken by the parent with the highest fitness. This leads to a prospective parent having a greater chance of being selected if it has a good fitness. Another method of selection is to create a tournament; a random set of potential parents are chosen from the current population and they compete with each other, with the winner being the one with the best fitness.

Once the parents are selected a procedure called cross-over is carried out. This is the mating stage, where genetic information is passed from parents to offspring. That is, the list of inputs of each parent is cut at a specific point. Then the left side of the first parent is attached to the right side of the second parent, and vice versa, creating two offspring to be assessed in the next generation.

Other evolutionary features may also be included in a GA:

- Mutation, whereby we randomly modify a small part of the design so that all regions in the design space can be reached. This has a specific probability of occurring on the offspring.

- Elitism, where the best design is guaranteed to survive.

GAs have a number of parameters that can have a significant effect on the final solution: the size of the population, the maximum number of generations, the probability of cross-over, and the probability of mutation all have to be carefully selected. It is difficult to make these selections *a priori*, and so a GA may be required to be run numerous times to ensure the solution found is indeed the optimal one. The GA used in this thesis is the so-called canonical implementation, due to Goldberg [1989].

2.4 Machine learning and Surrogate Modelling

Global optimization schemes are known to give satisfactory results but may do so by requiring a large number of objective function evaluations. They are therefore utilised when we have available a quick analytical procedure to analyse the performance of the designs. However, if each function evaluation involves using an analysis tool such as CFD or FEA, which could take hours or days to complete, the optimization process could take an inordinate amount of time to complete. One technique that requires fewer objective function evaluations, but still allows us to search globally, is *surrogate modelling* (also called meta-modelling).

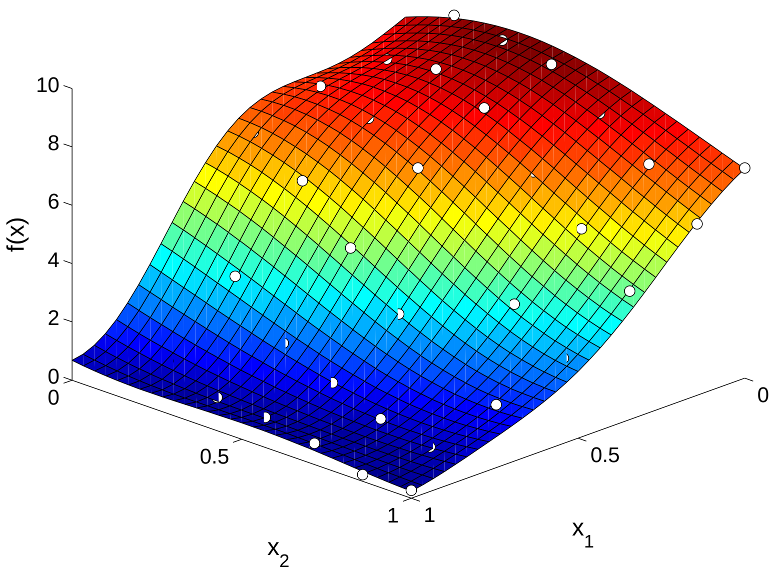


FIGURE 2.3: A simple example of a surrogate model generated using 36 training points in a two-dimensional input space.

The optimization strategies that have already been discussed concentrate their efforts primarily on finding the optimal solution. Surrogate modelling is a representative model of the whole search space that can be effectively searched by an optimization process. For example, Figure 2.3 shows a surrogate model that has been fitted to the responses of 36

(x_1, x_2) combinations. It is essentially a mathematical formulation designed to enable the *learning* of some response (objective function in our case) to a set of inputs. Surrogates belong to the *machine learning* family of algorithms. Mitchell [1997] describes machine learning as “any computer program that improves its performance at some task from experience”. Surrogates, after the initial model has been built, are often assessed through using update designs, which are also often used to improve the surrogate approximation. Any method that does slightly better than random guessing can be classed as a learning technique, albeit a weakly learnable strategy [Schapire, 1990]. The approach described here, that of guiding a learning process using the output variables of a system, is known as *supervised learning* [Hastie et al., 2009]. More specifically, surrogates use inductive inference, whereby any hypothesis found to approximate well over a large data set will also generalize well over the whole search space, including unobserved data; this is a main requirement of all surrogate models.

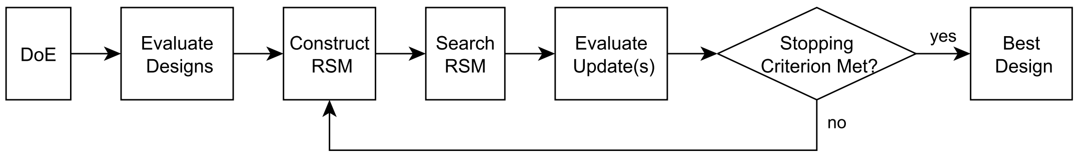


FIGURE 2.4: Flowchart, edited from Keane and Nair [2005], of a surrogate modelling approach.

The surrogate modelling procedure is shown in Figure 2.4. We first select a set of initial designs, with the aid of a Design of Experiment (DoE) technique, discussed in Section 2.5. In the formulation of a surrogate model we only need to analyse the performance of the initial population. A single evaluation of the surrogate takes a fraction of the time of a CFD or FEA computation, thus running the optimization process on the surrogate drastically reduces the time to find a solution. Once we have completed the expensive computations used to find the objective function of each design we fit the surrogate model. We must remember that a surrogate model is only an approximation to the objective function landscape, and the quality of the optimization procedure will be highly dependent upon the accuracy of the surrogate. Therefore, the optimum of the surrogate may not be the true optimum to the problem. To improve upon the accuracy of the surrogate we can perform an update or refinement procedure. We may wish to further explore the model to improve its global accuracy, or to concentrate on improving the accuracy in the region of the predicted optimum. Either way, the quality of the model can be evaluated by comparing the true solutions of newly chosen points to the approximation given by the model. If poor accuracy is met, the new points can be used to update the model. The positions of these update points can be intelligently placed using update strategies based on the prediction of the objective function, the probability of improvement, or the expected improvement over the function landscape. This process of choosing new points to evaluate, comparing with the current surrogate model, and then updating the model, can be iterated until a convergence (termination) criteria of the accuracy is met.

The choice of model used to fit the objective function points can be a difficult task. Keane and Nair [2005] list a number of models, including polynomial regression, Splines, Kriging, and Genetic Programming. All of the models need to be tuned to accurately model the data, whether it is the coefficients of the terms in a quadratic equation, or the hyper-parameters in the Kriging model. This tuning can be a simple task when using a quadratic as we may use a least squares solution on the problem. However, a quadratic may not be able to accurately describe complex surfaces, in which case a Spline or Kriging technique, known to perform better over multi-modal landscapes, may be more appropriate. We follow this description of surrogate modelling by describing three of the techniques: Polynomial Modelling, Kriging, and Genetic Programming. A selected class of machine learning techniques is that of classifiers. We shall use one such model in this thesis – its description is included in this chapter.

2.4.1 Polynomial Modelling and Least Squares

To construct a response surface through a set of data points, we can use a linear set of any specified functions of the inputs $\mathbf{x}$. For example, a linear combination of 1, x^2, and $\sin(x)$ is

$$y(x) = c_1 + c_2 x^2 + c_3 \sin(x), \tag{2.3}$$

where c_1, c_2 and c_3 are coefficients or weights of each function. We can produce a general form of this model, where $\mathbf{X}$ is the set of fixed functions of x, m is the number of functions, and c_k are the respective weights of each of these so called *basis functions*:

$$y(x) = \sum_{k=1}^{m} \mathbf{c}_k \mathbf{X}_k(x) \tag{2.4}$$

A set of equations that can be represented in this form are polynomial equations. Consider a problem where we have two input variables, x_1 and x_2 that are stored in vector $\mathbf{x}$, resulting in one output function $y(\mathbf{x})$. If we sample the input space to generate a training dataset, where $\{\mathbf{x}^{(i)}, y^{(i)}, i = 1, 2, \ldots, n\}$, we can fit a polynomial response surface through these points to come up with an approximation $\hat{y}$ to the output y. Keane and Nair [2005] write a quadratic response surface as

$$\hat{y} = c_0 + \sum_{1 \le j \le p} c_j x_j + \sum_{1 \le j \le p, k > j} c_{p-1+j+k} x_j x_k \tag{2.5}$$

c_0, c_j and $c_{p-1+j+k}$ are the coefficients we are trying to find to best fit the quadratic to the training data. Incorporating the training data, we are left with n equations to solve to find the coefficients $\mathbf{c} = [c_0, c_1, \ldots, c_{m-1}]$. These can then be used to construct our response surface approximations $\hat{\mathbf{y}}$ over the entire search space. We can write this series of equations in matrix form $\mathbf{Ac} = \mathbf{y}$ where $\mathbf{y} = [y^{(1)}, y^{(2)}, \dots, y^{(n)}]^T$, and

$$\mathbf{A} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & (x_p^{(1)})^2 \\ . & . & . & \dots & . \\ . & . & . & \dots & . \\ 1 & x_1^{(n)} & x_2^{(n)} & \dots & (x_p^{(n)})^2 \end{bmatrix}. \tag{2.6}$$

If we now multiply each side by the transpose of $\mathbf{A}$ to obtain the normal equations

$$\mathbf{A}^T\mathbf{Ac} = \mathbf{A}^T\mathbf{y}. \tag{2.7}$$

we can find the solution for the set of basis coefficients $\mathbf{c}$ by using a least-squares method. For example, the Moore-Penrose pseudo-inverse of $\mathbf{A}$, $\mathbf{A}^+$ ($\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$):

$$\mathbf{c} = \mathbf{A}^+\mathbf{y} \tag{2.8}$$

We can use the newly found polynomial coefficients with Equation 2.5 to plot our response of the model.

A requirement of the least squares approach is that more training points exist than coefficients in the model, $n > m$.

2.4.2 Kriging

Polynomial modelling is a very popular method used to model the response to a set of inputs, due to it being simple to understand and quick to implement. What it lacks is the flexibility and accuracy of other models in dealing with multimodal terrain. Polynomials are notoriously unwieldy surrogates of landscapes with multiple basins of attraction. Kriging (also known as Gaussian Process modelling) is a popular, highly flexible alternative for engineering design problems. It also enables accurate control of the amount of regression required. [1]

Consider the set of inputs $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n\}^T$, each of which has a corresponding objective function output $\mathbf{y} = \{y_1, y_2, \dots, y_n\}^T$. In the Kriging procedure we assume the funtion values to be the observed values of a stochastic process, which are normally distributed with a mean μ and variance σ^2. For any two input values, $\mathbf{x}_i$ and $\mathbf{x}_j$, we can assume that their corresponding performance, $y(\mathbf{x}_i)$ and $y(\mathbf{x}_j)$, will be close together in value if the "distance" between the inputs $|\mathbf{x}_i - \mathbf{x}_j|$ is small. Of course this is also assuming that the function is continuous. The correlation between the y values of any two random inputs is assumed to be of the form

[1] The interested reader is invited to read Forrester et al. [2008] for a more detailed description of the Kriging technique.

$$\mathrm{Corr}[y(\mathbf{x}_i), y(\mathbf{x}_j)] = \exp\left(-\sum_{l=1}^{d} \epsilon_l |x_{i_l} - x_{j_l}|^{q_l}\right) \tag{2.9}$$

where d is the number of control variables, ϵ_l determines the rate of correlation decay, and q_l the smoothness of the fit as we move in the lth coordinate direction. If $\mathbf{x}_i = \mathbf{x}_j$ then the correlation equals one, and if $|\mathbf{x}_i - \mathbf{x}_j| \to \infty$ then the correlation equals zero. The correlations between all the inputs $\mathbf{X}$ are stored in the correlation matrix

$$\mathbf{R} = \begin{pmatrix} \mathrm{Corr}[y(\mathbf{x}_1), y(\mathbf{x}_1)] & \cdots & \mathrm{Corr}[y(\mathbf{x}_1), y(\mathbf{x}_n)] \\ \vdots & \ddots & \vdots \\ \mathrm{Corr}[y(\mathbf{x}_n), y(\mathbf{x}_1)] & \cdots & \mathrm{Corr}[y(\mathbf{x}_n), y(\mathbf{x}_n)] \end{pmatrix}. \tag{2.10}$$

The values of ϵ and $\mathbf{q}$ are chosen in order to maximise the likelihood of the data having resulted from the postulated model $\hat{Y}$. This is achieved by using the likelihood function in logarithmic form,

$$\ln(L) = -\frac{n}{2}\ln(\sigma^2) - \frac{1}{2}\ln(|(\mathbf{R} + \lambda_\mathbf{r}\mathbf{I})|) - \frac{(\mathbf{y} - \mathbf{1}\mu)^T(\mathbf{R} + \lambda\mathbf{I})^{-1}(\mathbf{y} - \mathbf{1}\mu)}{2\sigma^2}. \tag{2.11}$$

We have included a regression constant, λ_r, to the diagonal of the correlation matrix, $\mathbf{R}$ to account for 'experimental error' associated with the performance y values. We must find values of λ_r, μ, σ, $\mathbf{q}$, and $\boldsymbol{\epsilon}$ to maximise the likelihood function in Equation 2.11. The optimal values of the mean μ and variance σ^2 can be made to be dependent upon the ϵ and $\mathbf{q}$ unknowns in the matrix $\mathbf{R}$, and λ_r, by taking derivatives of Equation 2.11 and setting them to zero. The result is:

$$\hat{\mu} = \frac{\mathbf{1}^T(\mathbf{R} + \lambda_\mathbf{r}\mathbf{I})^{-1}\mathbf{y}}{\mathbf{1}^T(\mathbf{R} + \lambda_\mathbf{r}\mathbf{I})^{-1}\mathbf{1}} \tag{2.12}$$

$$\hat{\sigma}^2 = \frac{1}{40}(\mathbf{y} - \mathbf{1}\hat{\mu})^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}) \tag{2.13}$$

where $\mathbf{1}$ is an $n \times 1$ column vector of ones. By substituting these optimal values back into Equation 2.11 to obtain the concentrated log-likelihood function

$$\ln(L) \approx -\frac{40}{2}\ln(\hat{\sigma}^2) - \frac{1}{2}\ln(|(\mathbf{R} + \lambda_\mathbf{r}\mathbf{I})|) \tag{2.14}$$

the likelihood is now only dependent on the regression constant $\lambda_\mathbf{r}$ and the correlation matrix $\mathbf{R}$, which itself is dependent on the hyperparameters ϵ, $\mathbf{q}$, from the correlation basis function in Equation 2.9. To find these parameters we must perform an optimization procedure to maximise Equation 2.14. Forrester et al. [2008] suggest the use of a global method, as the concentrated likelihood function is very quick to compute and it often features multiple local optima. Here we use a Genetic Algorithm.

Once we have found our optimal values of λ_r, ϵ and $\mathbf{q}$ from our optimization we can predict the value of y for any set of inputs $\mathbf{x}$, given by;

$$\hat{Y}(\mathbf{x}) = \hat{\mu} + \mathbf{r}^T(\mathbf{R} + \lambda_{\mathbf{r}}\mathbf{I})^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}). \tag{2.15}$$

The vector $\mathbf{r}$ contains the correlations of point $\mathbf{x}$ and the points used to produce the Kriging model using Equation 2.9.

2.4.3 Cross-Validation of a Surrogate Model

There are many ways to assess the accuracy of a surrogate model. Accuracy cannot be assessed solely by the error between the points fitted, as the purpose of a surrogate model is to generalise well over the whole search space. One technique is to not include a set of design points in the creation of the surrogate, then to use this set and compare their outputs to those given by the surrogate. However, surrogates are used where the objective function is expensive to compute, and so the set of design points tested will usually be small and so we would want to use all the results to generate the surrogate. Cross-validation uses only the design points that were used in the construction of the surrogate model. This involves splitting the design points into q usually equal subsets, then removing one of the subsets in turn and fitting the model to those subsets that remain. A loss function L_f can then be computed, defined as the root mean squared error (RMSE) between the predictor $\hat{\mathbf{f}}_q$ and the points in the subset left out of the model fitting $\mathbf{y}_q$:

$$L_f = \sqrt{\frac{\sum_{i=1}^{q}[\mathbf{y}_q - \hat{\mathbf{f}}_q]^2}{q}}. \tag{2.16}$$

To compare the accuracy of surrogate models from different experiments the responses are usually normalised (converted so that they range from [0, 1]) before cross-validation is performed. A special case of cross-validation is the leave-one-out cross-validation, where the number of subsets q equals the number of design points. This is a popular choice for assessing surrogate models, due to the limited number of design points used to create the models.

2.4.4 Genetic Programming

Genetic programming (GP) [Koza, 1992] uses the same mind-set as genetic algorithms. That is, a population is evolved using the Darwinian model of non-random survival of randomly varying replicators. They differ in the fact that a GP's genetic material encodes programs, rather than vectors of numbers. For the purpose of our explanation these programs are symbolic expressions representing a curve fit between data points. In other words, genetic algorithms modify the coefficients of a known function. In GP, the analytical form of the function is not known *a priori* and thus must be found along with the coefficients. GP-evolved symbolic algebra has previously been used for function approximation [Alvarez et al., 2001] and as a partial differential equation solver [Sóbester et al., 2008].

The implementation of a GA, where we create an initial population, evaluating each point, and generate a new population using cross-over, mutation and reproduction (see Section 2.3.1) is identical to the process used in GP. Added factors are required in GP to find the symbolic expressions; we can define the starting expression length, and the maximum length that we allow the expressions to become.

An important aspect of the genetic programming environment is the method used to create the initial population of random programs (in our case the programs are symbolic expressions). The traditional method uses a tree description, an example of which is shown in Figure 2.5. An expression is made up of two types of components: operators (the arithmetic operations and mathematical functions), and terminals (the variables and coefficients in the expressions). Note that terminal nodes, such as $b, 2, a$ in Figure 2.5, can be classed as the leaves of the tree, as no branches emanate off them. We start at the root node of the tree with a mathematical operator, randomly selecting operators and terminals for the nodes at the end of each branch. The branches are grown until terminated using a terminal node. In the creation of expressions using this method, we can produce illegal instances that cannot be evaluated. In this case we regenerate the expression until we achieve a valid expression. Using grammars [O'Neill and Ryan, 2001] instead of the tree approach ensures within the construction of the expressions that each will be syntactically correct, although the increased computational time from using variable length binary strings, converting them into integers, and then converting into the expression structure makes the method less attractive. The operators and terminals are then included after the expression structure has been created, meaning the binary string can be over double the size of the string used in the tree method.

Another important issue with GP is in the robustness of the symbolic expressions in accepting the input data: we must ensure that any argument in the expression returns real values. This involves, for example, using a special division operator that can accept 0 in the denominator and an operator that accepts negative arguments in square root terms.

If we have any prior knowledge about the type of symbolic expressions we are looking for, we can restrict the list of operators so we have a better chance of finding such an expression. For instance, if we know we are looking for a second order polynomial, we may include a square term in our list of operators and omit trigonometric operators. We must be careful however in deciding which operators to include and omit, as omitting useful operators may cause us to never find a good solution, and including less useful operators increases the size of the search space and thus could increase the time taken to find a solution.

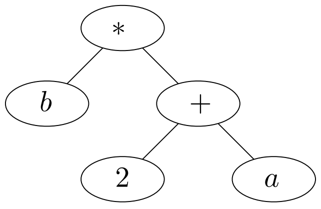


FIGURE 2.5: The symbolic expression $b(2 + a)$ depicted in tree form.

The main advantage of genetic programming over many other curve fitting methods, such as Kriging and polynomials, is that the GP result can give some insight into the underlying physics of the system, and so can be used for scientific exploration. For example, a portion of a sine wave can be modelled using a polynomial function. However, the sinusoidal nature of this system is lost by modelling it using this method. A GP search has the ability to identify the sinusoidal nature, as long as a sinusoidal term is present within the list of operators. The downside to GPs is that the discovery of an accurate regression curve is slow, and it is difficult to trade off good solutions with the complexity of the model.

2.4.5 Support Vector Classification

For some problems we do not require to model the objective function landscape, rather we only require to model the boundaries between different sets of data that are classified in a specific way. A typical example in the engineering community is that of determining the boundary between infeasible and feasible solutions (see Chapter 7). If the boundary between the two classes is seen to be linear we can often use a linear classifier, also called an optimal separating hyperplane. If a non-linear pattern is observed then we can use *Kernel* functions to map the problem to a feature space, where the problem can be separated using a hyperplane. Our discussion here gives only a brief overview of the method, with further details can be found in texts by Hastie et al. [2009] and Cristianni [2000]

Consider a training data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n\}$, each belonging to one of two sets of classes $\mathbf{y} = \{y_1, y_2, \dots, y_n\}$, $y \in \{-1, 1\}$. Support Vector classifiers (SVC) use support vectors in such a way so that two different classes are separated via a region rather than just a hyperplane. By viewing Figure 2.6 we can see that any point lying on the hyperplane between the two classes will satisfy $\mathbf{w}.\mathbf{x} + b = 0$, where $\mathbf{w}$ is the normal to the hyperplane and $b/|\mathbf{x}|$ is the perpendicular distance of the hyperplane to the origin. The hyperplane is said to be optimal when it separates the two sets of data without error and the distance between the hyperplane and the closest members of each class is maximal. We can define the distance between the two hyperplanes that define the margin between the two sets of data ($\mathbf{w}.\mathbf{x} + b = 1$ and $\mathbf{w}.\mathbf{x} + b = -1$) as $2/\|\mathbf{w}\|$, and so we wish to maximise this value. For mathematical convenience, we substitute $\frac{1}{2}\|\mathbf{w}\|^2$ for $2/\|\mathbf{w}\|$ and in turn make this into a minimisation problem in what follows.

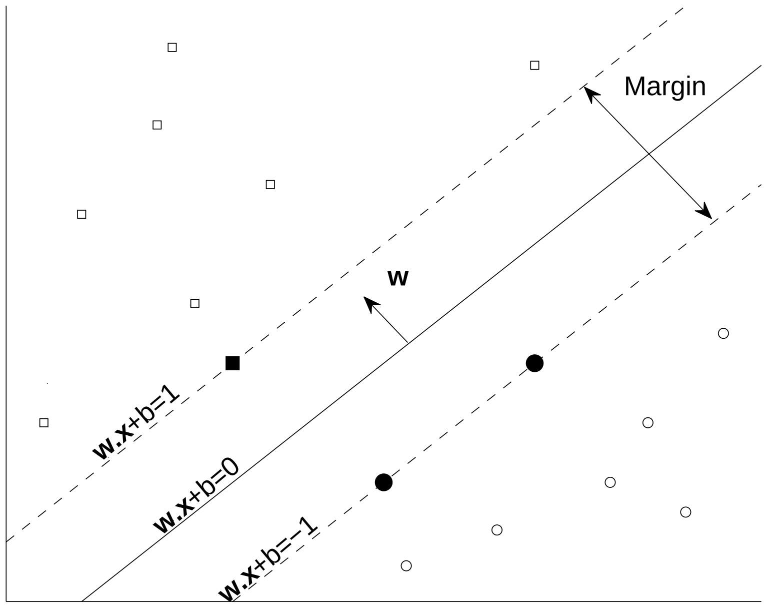


FIGURE 2.6: A separable problem of two classes (shown by □ and ○), showing the two support vectors (the dotted lines) and the boundary hyperplane (solid line).

To find the optimal separating hyperplane we must solve the following optimization problem

$$\begin{aligned} &\min && \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{subject to} && y_i(\mathbf{w}, x_i) \geq 1 - \xi_i \quad \text{for } i = 1, \dots, n \end{aligned} \tag{2.17}$$

where C is a constant that penalises for non-zero ξ_i, or the slack variables used to measure the degree of miss-classification. The slack variables become active when the training data is not linearly separated by the hyperplane. The solution to this optimization problem can be given by the saddle point of the Lagrangian functional. This leads to the creation of the dual problem:

$$\begin{aligned} &\min_{\alpha} && \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{n}\alpha_i \\ &\text{subject to} && 0 \leq \alpha_i \leq C \qquad \text{for } i = 1, \dots, n; \\ &&& \sum_{j=1}^{n}\alpha_j y_j = 0 \end{aligned} \tag{2.18}$$

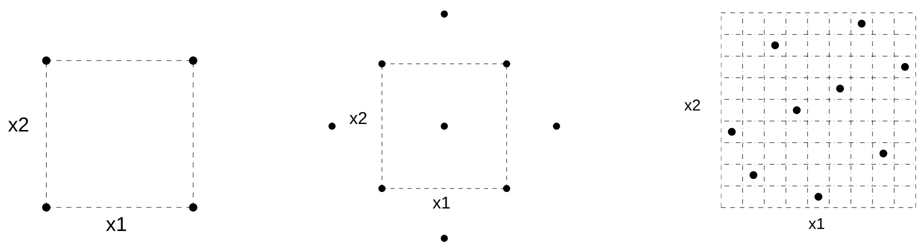
where α are the Lagrange multipliers and $K(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel function: a non-linear mapping of the input space to a high dimensional feature space, chosen *apriori* by the user – when a linear boundary is unable to classify between the two sets of data we can use a kernel function to map the input space into a higher dimensional feature space where a hyperplane can separate the two classes. There are many different types of kernel that can be used, many of which are similar to the geometry parameterization methods we discuss in Chapter 3 and used to generate surrogate models: polynomials, Gaussian Radial Basis Functions, and many different types of Splines can be used as Kernel functions. The value of C is now the upper bound of the Lagrangian multipliers, and is essentially chosen to reflect the noise in the data.

We can solve this optimization problem by using a quadratic programming method, for example, an interior-point solver [Vanderbei et al., 1997].

2.5 Design of Experiments for Computer Simulations

In general, the optimization process requires that we have some sort of initial guess of the solution to begin the optimization loop with. The local searches described earlier start from a single such point. However, many other methods require the performance information of a set of initial designs that are enclosed within the area bounded by the variable limits; the *design space*. The performance of the optimizers can be highly dependent upon the positions of the initial set of designs in the design space, and so random positioning of the initial design variable sets is ill-advised. Also important is the size of the design set; to densely fill the design space with different designs nullifies the use of the elegant optimizers under consideration. Moreover, it will lead to a disproportionate amount of time to find a solution. However, not sampling the design space enough can cause poor convergence rates, and poor accuracy in initial surrogate models. To aid us in positioning the initial set of designs we can make use of sampling plan techniques.

Two key attributes of sampling plan layouts (sometimes termed DoEs – short for Design of Experiment) are *a)* the *uniformity* with which the data points fill the design space and *b)* its *stratification*, which refers to the uniformity of the projections of the data points onto the design variable axes. To create an efficient optimization process we must also seek to limit the size of the initial set without jeopardising the quality of the optimization search. A large influence to the size of the sampling plan, if using a computational procedure to analyse the designs, is the computational resources available; we may be able to run all designs within the design set in parallel, and so the limitation would be how many parallel runs we are able to make. We may also be able to use multiple processor cores to speed up a single simulation. Also note that for algorithms, such as GAs, the size of the design set is evaluated for every generation and so $n \times g$ computations have to be completed, where n is the size of the design set and g is the number of generations. All of these requirements are unlikely to be achieved using a randomly generated set of designs, and so specific sampling plans exist for us to exploit.



(a) Two-level full factorial (b) Cicumscribed Central Composite (c) Latin Hypercube

FIGURE 2.7: A variety of sampling plans that can be used in a DoE.

Typical sampling plans include the Central Composite, Latin Hypercube, and Full Factorial methods (see Figure 2.7). The full factorial design shown in Figure 2.7(a) shows a two level plan, where the experiment points are positioned on the upper and lower bounds of the design space. This may be the simplest form of sampling plan, but is widely used for problems where discrete designs or operating conditions are available, or if we know beforehand that the response to the design changes will be linear. For a higher order quadratic response a sampling plan like the central composite design would be more appropriate (Figure 2.7(b)). With this plan four extra designs populate outside the bounds of the full factorial design, plus one in the middle. The outside points allow for curvature to be modelled in the response. The analysis of the middle design is often replicated in order to measure the experimental error.

The full factorial and central composite methods are popularly used for linear and quadratic problems respectively. However, for many engineering design problems the characteristics of the response are not known *a priori*, with many being found to be highly multimodal, as shown previously in Figure 2.2(b). We can increase the number of levels within the design space, but by doing so we increase the number of samples for the full factorial design to l^k, where l is the number of levels and k is the number of variables. Using large values of k is a very inefficient way to sample the search space [Alam et al., 2004] and thus a costly optimization process.

Sampling plans that fall into the class of *Latin Hypercubes* [McKay, 1992] have excellent stratification properties: in an attempt to limit the number of design points, the Latin hypercube methods require that there is only one sample in each row and each column within an $n \times n$ grid of the design space (Figure 2.7(c)). The positions of these sample points can be optimised, as done by Morris and Mitchell [1995], where the minimum distance between any two points is maximised within the design space, and so filling the design space well. Uniformity is maximised (as measured by the Morris and Mitchell [1995] criterion) only for when we have a square number of points.

2.6 Optimizer Selection

The type of optimizer we should use is highly dependent upon the problem at hand. Most notably, the time taken to find a solution is dependent on the optimization process; local optimizers are often quicker than global ones in finding a solution, with surrogate models used to speed up the global optimization process when it is computationally expensive to find the objective function values. This may force us to use local optimizers, but we must remember that they only search the space near the initial starting point in a multi-modal search space. If known to be multi-modal it would be more appropriate to use a global search, with or without a surrogate attached. Again, a surrogate may accelerate the process, but as it is an approximation to the real objective surface, our solution may not be as accurate when using a global model on its own.

Global optimizers are generally good at exploring the design space, finding the optimal regions. What they lack is the ability to exploit these regions and find the exact optimum within each region. This has led to researchers developing hybrid global/local optimizers; the global search is used to explore the search space, and the optimal regions found are searched by local optimizers to return a highly accurate optimal solution. Within the topic of hybrid searches is the choice of defining an instance and its performance; we can define an instance as the result of the global search, with its fitness being updated to the local search result, or define it as a result and fitness of the local search. We may initially think that this does not matter, as a deterministic local search will always find the same solution from the same starting point. It becomes more obvious when we consider the global optimizer, in that the next generation of instances in a population are dependent on the current generation. We may find that if we update the current instance with the result from the local search that we restrict our exploration of the search space, as many may be forced into the same optimal region, therefore obtaining the same solution. However, this result may make the solution converge faster than not updating the instance.

The process whereby we update the variables within the instances from the local improvement to be put back into the global search is referred as *Lamarkian learning*; characteristics acquired during the lifetime of the instance can be inherited. This type of learning goes against the Darwinian idea that is the basis of genetic algorithms. An alternative to Lamarkian learning is *Baldwinian learning*, named after the work of Baldwin [1896]; traits of the parents are inherited by the offspring, and not features generated from the environment. In our current setting this is equivalent to not updating the variables of the current instance by the results of the local search, but assigning them the 'as optimized' objective value. Once again, the type chosen for a study is highly dependent on the problem; if we have a computationally expensive function, we may see the throwing away of computational results from the local search as wasteful, and so a Lamarkian process may be used. Some see Lamarkian learning as unnatural and, to keep within the same mind-set as evolutionary theory, a Baldwinian approach must be used. Houck et al. [1996] gives a comparison of using these two different learning strategies using a GA/local hybrid, noting that a Lamarkian learning strategy led to a faster convergence rate. However, Whitley et al. [1994] found that, in his studies, the Lamarkian learning strategy converged to a local optimum, whereas the Baldwinian cases found the global optimum. A more detailed account of hybrid optimization is given by Sóbester [2003], where he delves more into the search time between the exploration of the global model and the exploitation given by the local improvement.

2.7 Objectives and Constraints

The type of optimization algorithm has been presented to be an important factor in the setup of an optimization study. Equally important is the definition of the inputs, objective function(s), and the constraints that need to be applied to find a valid solution. The inputs are generated from the parameterisation process, which is to be discussed in section 3, and so we will not focus on this here. We concentrate our attention to the importance of defining the objectives and constraints within an optimization process, giving examples for each.

The choice of objective function is decided directly from the aim of the optimization procedure. A general optimization strategy, in a design sense, is to use a performance metric as the objective function - the drag of an aircraft, for example. In some problems, we may not have such a single valued objective. Typical examples are minimizing the difference between pressure profiles in an inverse design case, or obtaining the input parameters to best fit a set of points in a curve fitting process. In these cases we must find a way to model the objective as a single value.

Consider an airfoil in an inverse design problem, where we are modifying the shape to match a predefined pressure profile. At first glance a reasonable definition of the objective function could be the summation of the difference between the target profile pressure coefficient $c_p^{(t)}$ with the airfoil profile pressure coefficient c_p at n points along the chord:

$$f(\mathbf{x}) = \sum_{i}^{n} \left[c_{p,i} - c_{p,i}^{(t)} \right] . \tag{2.19}$$

However, the summation of $c_{p,i} - c_{p,i}^{(t)}$ may not be a robust objective, as the value can be positive or negative: a positive difference between the c_p distributions in the summation can be negated by a negative difference at some other position on the airfoil. To alleviate this issue we use the sum of the squares of the difference between the pressure profiles:

$$f(\mathbf{x}) = \sum_{i}^{n} \left[c_{p,i} - c_{p,i}^{(t)} \right]^2 . \tag{2.20}$$

This method gives us an overall statistic at how well the pressure profiles match. What we do not gain is an insight into the actual shapes and whether they are similar. For instance, we might generally match the pressure profiles well, but a sharp peak may exist that does not follow the target profile. This result may not be noticeable from Equation (2.20) as the other input points may be defined well enough to cover the poor residual of this point. To counter this we can use a penalty term P that adds a factor a to the objective function every time an instance in the sum of our objective is larger than an amount g:

$$P = \begin{cases} a & \text{if} \quad \max\left(c_{p,i} - c_{p,i}^{(t)}\right) > g \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

The size of a and g can be constant (for instance within a certain tolerance), or it can be a certain percentage from the average of the other instances in the sum.

Consider the example of attempting to define a circular object. The difference of the y coordinates at the corresponding x coordinates between the target and our approximation would be a poor choice of objective function; the gradients at corresponding x-positions vary between zero and infinity, leading to different objective function sensitivities across the coordinate points. A more suitable objective would be to use the difference between the radius at pre-specified angles; the sensitivities at all angles will be similar, leading to an objective that is easier to optimize

'Real-world' problems often feature constraints, which restrict the search to parts of the design space. A simple constraint may be the upper and lower bounds of the input variables. However, constraints may also involve more complex relationships; for example, when optimizing an airfoil, we need to maintain a specific thickness value where the spars will be placed. The shape of the airfoil will generally be approximated by one of the parameterization methods to be discussed in Section 3, the inputs of which are unlikely to be directly linked to the thickness of the airfoil. If equality constraints exist ($h(\mathbf{x}) = c$) we must attempt to eliminate them by restructuring the optimization problem. In situations where this cannot be done, equality constraints can be transformed to a pair of opposed inequality constraints, $h(\mathbf{x}) \leq c$, $h(\mathbf{x}) \geq c$.

One of the many methods that we can use to invoke constraints into our formulation of a problem is to add a penalty function, such as the one in Equation 2.21. This penalty function is only applied if the search passes a constraint boundary and enters an *infeasible region.* Careful consideration must be made when formulating a penalty function. Inserting a single value to become active when a constraint is violated causes a severe cliff in the objective function landscape. Any solution on or near the constraint boundary can therefore be difficult to find. Moreover, if a search finds itself within the infeasible region, the contours of the objective function landscape may be in such a position to force the search away from the feasible region for where we want the solution to lie. Both of these issues are inherent to gradient-based optimization methods, with zeroth-order methods performing a little better. Sometimes it is appropriate to use a function, such as a linear penalty function, dependent upon the degree of violation, which may allow for the search to drop back into the feasible search space.

2.8 Multidisciplinary Design Optimization

For nearly all 'real' design problems a product must perform well over multiple disciplines for it to be regarded as a well posed solution. For example, for an aircraft, we do not only require that the aircraft has a high lift-to-drag ratio, but also that the airframe can withstand the loads acting upon it, as well as reducing the costs in manufacturing. Sobieszczanski-Sobieski and Haftka [1997] define Multidisciplinary Design Optimization (MDO) as the "methodology for the design of systems in which strong interaction between disciplines motivates designers to simultaneously manipulate variables in several disciplines". By simultaneously optimizing the different disciplines we can achieve a significantly better solution, as well as decreasing the time required to find a solution.

Historically, industry is very reluctant to utilise optimization processes because they do not integrate readily into legacy design processes. Many companies have groups that specialise in a single discipline, and so devising an automated process within an MDO environment is a difficult task in itself. The process is made easier by packages such as Isight and SIMULIA [2011] that can be used to couple analysis tools of different disciplines within a workflow. The development of these organisational and computational tools is an active area of research, a review of which is given in Sobieszczanski-Sobieski and Haftka [1997], with many example applications in Kodiyalam [1998] and Kodiyalam and Yuan [2000].

Each discipline within the MDO process may generate its own objective function. In some situations we may be able to develop an overall objective function. For example, we can use the Breguet range equation to combine the aerodynamic efficiency, $\frac{L}{D}$, and specific fuel consumption, SFC;

$$R = \frac{L}{D}\frac{V}{\mathrm{SFC}}\ln\left(\frac{W_e + W_f}{W_e}\right) \tag{2.22}$$

where R is the range of the aircraft, V the cruise velocity, W_e the empty weight of the aircraft, and W_e the weight of the fuel. However, these relationships do not occur often. If we are lucky the objectives of all the disciplines will converge on the same optimal solution, but often the optimal solutions for the individual disciplines are competing against each other. If this is the case then we require a *Multiobjective Optimization* (MO) strategy.

2.8.1 Multiobjective Optimization (MO)

There are many different techniques we can use within the field of MO to find a single, optimal solution. In situations where we can assign weighting factors to the relative importance of each objective, we can effectively run an optimization algorithm on a global objective function that is a function of the single objectives multiplied by their respective weightings. However, it is likely that we have no indication to the importance of either objective in relation to each other. *Multiobjective weight assignment techniques* can be used, that attempt to define the weightings in a repeatable and justifiable manner, but there is no scientific reasoning behind these techniques. A more in depth discussion of these methods is given by Keane and Nair [2005].

When we are unable to combine the multiple objectives into a single function, we have to identify a Pareto set of designs; in a Pareto set each member is non-dominated, which means that no other design has better performance on all objectives. The simplest way to generate a Pareto set is to do direct searches on the sum of the objectives, changing their respective weights each time. This may be inefficient as this process may have to be repeated many times to build the full Pareto set. A more efficient procedure uses a multiobjective evolutionary algorithm (MOEA), one of which is the non-dominated sorting genetic algorithm (NSGA-II), developed by Deb et al. [2002]. The non-dominated solutions at each generation of a GA are given a rank of one and the rest are given ranks of two or higher (with rank two designs being non-dominated if we were to remove the rank one set, and so on). The GA is then guided to explore through the design space by rewarding or penalising each individual in the population, based on its distance from the Pareto front (encouraging Pareto optimality) and other members (encouraging a uniform spacing of designs along the Pareto front).

We can visualise the Pareto set by plotting the designs against each objective function, as in Figure 2.8. In this example the aim is to minimise both objectives. We have also included the objective function results of numerous other designs that reside within the design space, as an intuitive illustration of the fact that all designs within the Pareto set are indeed optimal in some way. From the individuals belonging to the Pareto set, designers can actively seek a single solution that may stand out from the others - a Pareto-optimal design may possess a counterintuitive solution that may lead to a radical rethink in a design. Designers may also want to perform more analyses to the Pareto set of designs, and pick an optimal solution with regards to another objective.

2.9 Summary

A brief introduction to the optimization methods popular in design optimization has been presented in this chapter, where we have introduced a few of the different types,

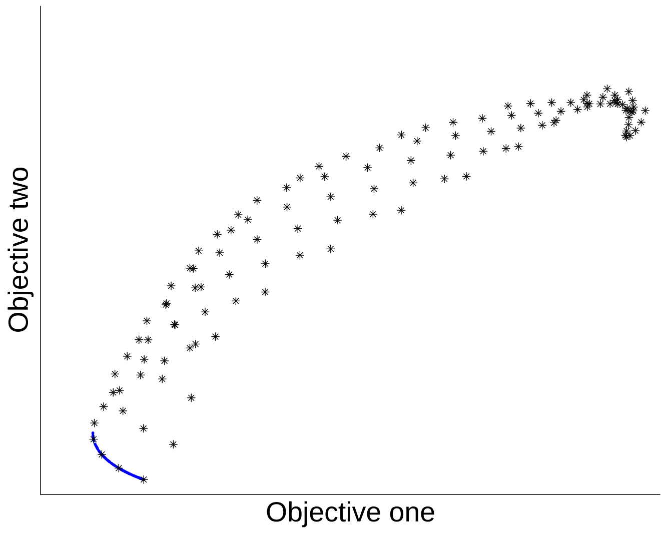


FIGURE 2.8: Two objective functions plotted against each other for numerous designs (the black stars), and the respective Pareto front (the line in the bottom left corner) generated using NSGA-II.

giving examples. More importantly, we have described the advantages and disadvantages of each type that is deemed critical for designers to know and understand when planning the optimization procedure. We have also shown how an optimization process can be converted to include multiple objectives, producing a number of optimal solutions within a Pareto set of designs.

Although the choice of optimization strategy is critical to the solution, also important are the other steps along the optimization cycle. We must incorporate constraints appropriately, either by redefinition of the problem statement or by altering the objective function in some way. The objective function must accurately define the performance of all designs within the design space, and be sensitive to changes in the input parameters. As important to the steps already discussed is the definition of the inputs, known as *parameterization* of the model. This is discussed in Chapter 3 within the realm of shape design optimization, where we describe its importance to the optimization procedure as well as the techniques already in existence.

Chapter 3

Geometry Parameterization Methods

In Chapter 2 we briefly discussed the role of the parameterization model in the optimization process. We mentioned that it is the formulation of the set of inputs that characterises the design to be optimized. The values of these input variables are unique for each design, and thus a change of any of the values causes a physical change in the design. For some geometries the input set is obvious (for example, the number of cross plys in a composite material, or the length and width of a beam). Providing a set of inputs that describe more elegant shapes, which can also be modified to provide an almost unlimited number of diverse shapes and thus designs, is not so intuitive. Many researchers have concentrated their efforts into this area, creating the field of *geometry parameterization*. In the context of this thesis, we define the shape of an object as the surface that encompasses that object. In 2D terms we can describe the surface as a curve, and we will see that we can define these curves using similar curve fitting techniques to those used in surrogate modelling methods discussed in Section 2.4.

In this chapter we firstly discuss the attributes a geometry parameterization model must possess if it is to be useful in an optimization procedure. We then describe some of the more popular models used in aircraft design. We finish by discussing the results published in the literature that compare some of the different techniques.

3.1 Parameterization Model Attributes

We have already mentioned a few of the attributes we require a parameterization model to possess:

- it must describe the shapes already used by existing designs for the application under consideration so that they can be considered in the optimization process;

- a single set of the input variables generated from the parameterisation model must be unique to that design, thereby any modification of the input values must lead to a physical alteration of the previous design.

However, other attributes also exist that go a long way to ensure the success of an efficient optimization procedure. *Geometric robustness* is a term given to the ability to repeatedly create "real" designs in regard to the application. The term "real" is given, in a geometry context, to any design that has finite volume and thickness everywhere, and is within specific constraints set by the user. Geometric robustness ensures that the designs being considered can also be easily manufactured and are valid within the constraints given by the designer. An example that ensures real designs is the control polygon of the Bezier and B-Spline methods; the curve of such a method is always within the control polygon of its control points.

When we discussed the selection of optimization method in Section 2.6, we stated that the computational time is an important consideration to the choice of the method. Due to their numerical formulations and the codes used to evaluate the designs, optimization procedures are notoriously slow at finding solutions, and so any advancements found to speed up the process are often seen as a good one. The parameterization model can aid in the efficiency of the optimization process by reducing the size of input set. However, we must be careful when doing this not to consequently diminish the ability to define an almost infinite number of different shapes. The significance of this can be seen in Figure 3.1, where the computational time increases exponentially with number of input dimensions; consider searching a single input d times. If we were to sample n different inputs to the same level of detail, we would have to evaluate d^n different input sets. This is known as the *curse of dimensionality*. It has a more significant effect than, say, the number of generations g in a genetic algorithm, which only increases the number of simulations by $p \times g$, where p is the population size. Unfortunately, many parameterisation techniques generate variables in the order of tens for a simple 2D airfoil problem and so by only reducing the input set by one will significantly speed up the optimization process.

The requirement "that any modification of the input values must lead to a physical alteration of the previous design" leads to the importance of the *geometric sensitivity*, defined by Samareh [2001] as "the partial derivative of a response with respect to a design variable". More simply it is the amount a geometry changes its shape with a small change to the input set. The ideal would be a proportionally small change to the shape with a small change of the parameter values. For example, a Lagrange polynomial is very poor at maintaining its structure when its control parameters are modified slightly (see Cottrell et al. [2009]), whereas a Bernstein polynomial maintains its general shape when a control point is perturbed (described later).

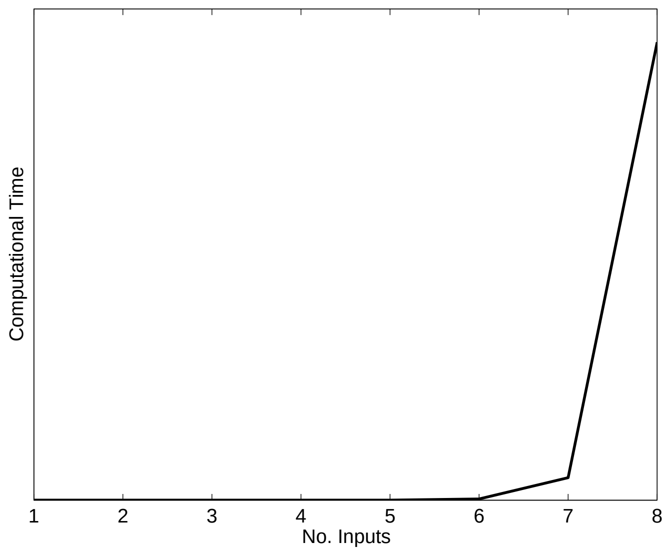


FIGURE 3.1: Dependence of computational time with number of variables.

We must also consider the sensitivity of the performance values, generated by some analysis code or experiment, to changes in the input set. For a computational fluid dynamics solution, we are not just concerned with the geometric sensitivity, as we approximate the shape using a grid of points called a mesh, as well as using a numerical procedure to find the flow characteristics. Therefore, the aerodynamic sensitivity is effectively the product of the geometric sensitivity, grid sensitivity to the geometry, and the aerodynamic sensitivity to the grid (see Samareh [2001]). Again, from an optimization point of view, it is important that only a proportionally small change in the aerodynamic characteristic occurs with a small change in the design variables, otherwise we may never find the most suitable design for our objective; a small change in the parametric variable leading to a drastic change in the aerodynamic characteristics results in us missing the aerodynamic relationship between this region, which may lead to missed local minima or constraint boundaries. When the design analysis procedure requires discretizing the physical space in or around the designs, we must take into account the grid or mesh approximation to the surface – the accuracy of this approximation is, of course, a function of mesh density. A similar reasoning can be employed when we export a CAD model - we can use a variety of export formats to do this, but many work on the assumption that the surfaces are linear interpolations between points on the surface, just like the assumption made by many grid generation codes. The transfer of geometric data from CAD to the discretization process is itself a topic of research within the field of *isogeometric analysis*(see Cottrell et al. [2009]).

We have listed many of the attributes we require of a parameterization model, however we may have other requirements that we may wish the parameterization to possess. Many design engineers work with the aid of Computational Engineering Design (CAD) systems, and generally wish to view the products they are designing rather than to trust a set of numbers. Shape parameterization is also used in the gaming and animation industries, where it allows fine detail and movement with minimal computational restrictions. The ability to use these techniques within a CAD environment has therefore become an important aspect. Many techniques that have not possessed this ability are therefore not favoured in industry. Splines are an example that have been incorporated within CAD models, as they can be manipulated using *affine transformations* so the geometry can be rotated, translated, and scaled. This ability has led to Splines becoming very popular within the design community.

As designers model more complex geometries, restrictions due to computer performance become common. If we were to define geometries in a CAD model as interpolations between points on the surfaces, we would require a large number of points to gain an accurate representation. For relatively large or complex geometries we may find that the number of points required exceed the amount that can be stored by the computer. In optimization simulations, it is too computationally expensive to describe a complex object by large numbers of discrete points, as it would take lifetimes to find the ideal solution. This is why original CAD systems used polygonal methods, as objects were essentially a collection of simple shapes that were easy to compute. We must be careful when developing such a geometric method, as simplifying a method to increase computational efficiency could in fact reduce the flexibility of the model itself. The use of discrete points and polygonal methods, as well as the Constructive Solid Geometry (CSG) technique that most CAD systems are now built on, are not described here, but the interested reader can find further information in Mortenson [2006]. We concentrate our discussion here on techniques widely used in the aircraft design domain to optimize aircraft and their components. We discuss a variety of techniques to give a flavour of those available to a designer, but it must be stressed that this is not an exhaustive list. A more definitive collection is given by Samareh [2001].

3.2 Parametric Shape Modelling

Suppose we wanted to approximate our geometry using a polynomial model. When discussing surrogate modelling in Section 2.4, we used a polynomial in its explicit form ($y = f(x)$) and found the coefficients using the least squares method. Defining the equation explicitly in Euclidean space brings a number of drawbacks, as stated by Mortenson [2006]:

- The points defining the shape are dependent on the coordinate system, when we are concerned with their dependence upon each other.
- Tangent lines may become parallel to principal axes, which may result in values of infinity, leading to undefinable curves.

- Non-planar and bounded curves are not easily represented by ordinary nonparametric functions.

This has led to the preferred way to represent geometric shapes using parametric equations; we split our explicit function $y = f(x)$ into two functions $x = x(u)$ and $y = y(u)$, both of which are dependent on a new parameter u. It must be stressed that we use the word 'parametric' here with its mathematical meaning, and not that of parameterization in geometric modelling. Usually we normalise the parameter space by setting $u \in [0, 1]$, so that the curve is bounded as it has two definite end points. In reference to a 3D case, we have the natural vector representation,

$$\mathbf{c}(u) = [x(u)\ y(u)\ z(u)] \tag{3.1}$$

We can represent the components of $\mathbf{c}(u)$ as any mathematical function. For example, we can define each component $c_i(u)$ as a set of basis functions:

$$c_i(u) = \sum_{j}^{n} \mathbf{A}_{i,j} \boldsymbol{\theta}_j(u) \tag{3.2}$$

where $\boldsymbol{\theta}(u)$ is a n sized vector of fixed functions of u, and $\mathbf{A}$ is a $n \times 3$ matrix comprising of the coefficients of $\boldsymbol{\theta}(u)$ for each parametric component $c_i(u)$. We can then modify the coefficients of $\mathbf{A}$ to create our required curve.

Mortenson [2006] gives a list of advantages of parametric equations:

- They allow separation of variables and direct computation of point coordinates
- Easy to express as vectors
- Each variable is treated alike
- More degrees of freedom to control curve shape.
- Transformations may be performed directly on them
- They accommodate all slopes without computational breakdown
- Extension or contraction into higher or lower dimensions is direct and easy without affecting the initial representation
- The curves they define are inherently bounded when the parameter is constrained to a specific finite value. This allows us to create a piecewise representation of a curve.
- The same curve can often be represented by many different parameterizations.

Parametric methods appear to be the ideal modelling techniques for all cases, but they do have their own set of drawbacks. For example, Du and Qin [2007] indicate that parametric techniques have trouble with shape bending and collision detection. Their disadvantages are thought to be outweighed by their advantages, which is evident from their popularity within the design community.

We describe here some of the most popular parametric methods used by the design community. We begin with the most simple polynomial methods and then evolve the process to create Bézier, B-Splines and finally NURBS curves. We only consider here these methods in their simplest form. More advanced routines, as well as the definition of surfaces, is given by Piegl and Tiller [1997].

3.2.1 Polynomial Approaches

Using polynomials to describe curves gives a large amount of flexibility in the curve definition, with using only a few design parameters. These are easily and efficiently computed, as they require little storage space and are robust to floating point and round-off errors. The simplicity of polynomial curves leads to their inability to represent all curves. We can combat this disadvantage by representing curves as a system of polynomials; an nth-degree power basis representation is given by

$$\mathbf{c}(u) = \sum_{i=0}^{n} \mathbf{a}_i u^i \qquad 0 \leq u \leq 1 \tag{3.3}$$

where $\mathbf{a}_i = (x_i, y_i, z_i)$ are the coefficient vectors of the power bases u^i. It can be seen that if $n = 1$ then $\mathbf{c}(u) = a_0 + a_1 u$, which is a straight line, and if $n = 2$ then $\mathbf{c}(u) = a_0 + a_1 u + a_2 u^2$, which is a parabolic arc. We can compute the points on the curve $\mathbf{c}(u)$ by using Horner's method, which finds the coordinates using only n multiplications and n additions. For example, the polynomial

$$\mathbf{c}(u) = \mathbf{a}_3 u^3 + \mathbf{a}_2 u^2 + \mathbf{a}_1 u + \mathbf{a}_0 \tag{3.4}$$

is computed in the form $\mathbf{c}(u) = [(\mathbf{a}_3 u + \mathbf{a}_2)u + \mathbf{a}_1]\, u + \mathbf{a}_0$.

This approach is the very basic form of curve generation using polynomials. The coefficients $\mathbf{a_3}, \mathbf{a}_2, \mathbf{a}_1, \mathbf{a}_0$ do not give any insight into the curve that is to be generated, and the method itself is prone to round-off errors. To achieve a greater intuitiveness of the curve structure we can define these coefficients in terms of the end points $\mathbf{c}(0)$ and $\mathbf{c}(1)$

and their tangent vectors:

$$\begin{aligned}\mathbf{c}(0) &= \mathbf{a}_0\\ \mathbf{c}(1) &= \mathbf{a_3} + \mathbf{a_2} + \mathbf{a_1} + \mathbf{a_0}\\ \mathbf{c}^u(0) &= \mathbf{a}_1\\ \mathbf{c}^u(1) &= 3\mathbf{a}_3 + 2\mathbf{a}_2 + \mathbf{a}_1\end{aligned} \tag{3.5}$$

where $\mathbf{c}^u$ defines the derivative of $\mathbf{c}$ with respect to u. Simultaneously solving this set of equations gives the coefficients:

$$\begin{aligned}\mathbf{a}_3 &= 2\mathbf{c}(0) - 2\mathbf{c}(1) + \mathbf{c}^u(0) + \mathbf{c}^u(1)\\ \mathbf{a}_2 &= -3\mathbf{c}(0) + 3\mathbf{c}(1) - 2\mathbf{c}^u(0) - \mathbf{c}^u(1)\\ \mathbf{a}_1 &= \mathbf{c}^u(0)\\ \mathbf{a}_0 &= \mathbf{c}(0)\end{aligned} \tag{3.6}$$

Substituting these values into Equation 3.4 gives

$$\mathbf{c}(u) = F_1(u)\mathbf{c}(0) + F_2(u)\mathbf{c}(1) + F_3(u)\mathbf{c}^u(0) + F_4(u)\mathbf{c}^u(1) \tag{3.7}$$

where

$$\begin{aligned}F_1(u) &= 2u^3 - 3u^2 + 1\\ F_2(u) &= -2u^3 + 3u^2\\ F_3(u) &= u^3 - 2u^2 + u\\ F_4(u) &= u^3 - u^2.\end{aligned} \tag{3.8}$$

are the *Hermite basis functions.* It is convenient to have shape controlling variables at the end points of the curve as, we will see when discussing B-Splines, it makes it easier to join curves together.

3.2.2 Bézier Curves

The power basis method and Hermite form give very little geometric insight into the shape of a curve. It is more desirable to have a method that can control a curve's shape in a predictable way using only a few parameters. One method that achieves this was developed by Pierre Bézier; a Bézier curve is defined as:

$$\mathbf{c}(u) = \sum_{i=0}^{n} B_{i,n}\mathbf{p}_i \qquad 0 \leq u \leq 1 \tag{3.9}$$

where $\mathbf{p}_i$ are the control points that are interpolated to create a *control polygon* and $B_{i,n}$ are the nth-degree *Bernstein polynomials*, defined as:

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!}u^i(1-u)^{n-i} \tag{3.10}$$

$n = 0$	1				
$n = 1$	$1 - x$	x			
$n = 2$	$(1 - x)^2$	$2(1 - x)x$	x^2		
$n = 3$	$(1 - x)^3$	$3(1 - x)^2x$	$3(1 - x)x^2$	x^3	
$n = 4$	$(1 - x)^4$	$4(1 - x)^3x$	$6(1 - x)^2x^2$	$4(1 - x)x^3$	x^4

FIGURE 3.2: Bernstein polynomial representation pyramid reproduced from. Kulfan and Bussoletti [2006]

Figure 3.2 shows clearly the patterns of the Bernstein polynomials as the degree n is increased. The properties of these basis functions, stated by Piegl and Tiller [1997], are:

1. Nonnegativity: $B_{i,n} \geq 0$ for all i, n and $0 \leq u \leq 1$
2. Partition of unity $\sum_{i=0}^{n} B_{i,n}(u) = 1$ for all $0 \leq u \leq 1$
3. $B_{0,n}(0) = B_{n,n}(1) = 1$
4. $B_{i,n}(u)$ attains exactly one maximum on the interval [0,1] at $u = i/n$
5. For any n, the set of polynomials $\{B_{i,n}(u)\}$ is symmetric with respect to $u = 1/2$
6. Recursive definition: $B_{i,n}(u) = (1 - u)B_{i,n-1}(u) + uB_{i-1,n-1}(u)$, where we define $B_{i,n}(u) \equiv 0$ if $i < 0$ or $i > n$
7. Derivatives:

$$B'_{i,n}(u) = \frac{dB_{i,n}(u)}{du} = n(B_{i-1,n-1}(u) - B_{i,n-1}(u))$$

   with $B_{-1,n-1}(u) \equiv B_{n,n-1}u \equiv 0$

These properties allow for the polynomials to be calculated simply, and for the easy differentiation of the curves at their end points, which are symmetric with each other.

Bézier curves are more geometric than power basis curves in the fact that due to a variation diminishing property (if we were to draw a straight line across a Bézier curve the line will not intersect the curve more times than it intersects the control polygon) the control polygon (area encompassed by the control points) approximates the shape of the curve. Together with the convex hull property (curves are contained within the outermost lines connecting the control points, p), we have a more intuitive understanding to the characteristics of the curve and so the user can be more interactive. It is also less prone to round off error due to the *deCasteljau* algorithm being a repeated interpolation from the control points to find any point u along the length of the curve;

$$\mathbf{C}(u) = \mathbf{p}_{k,i}(u_0) = (1-u_0)\mathbf{p}_{k-1,i}(u_0) + u_0\mathbf{p}_{k-1,i+1}(u_0) \qquad \text{for} \begin{cases} k = 1,\ldots,n \\ i = 0,\ldots,n-k \end{cases} \tag{3.11}$$

for an nth-degree Bezier curve at point u_0. This method is depicted in Figure 3.3, with the white point on the curve found through using the deCasteljau algorithm (repeated interpolation of the control points, shown by the small black dots). This figure also shows the intuitiveness between the control points and the resulting curve shape. A more detailed explanation is given in Piegl and Tiller [1997]. Bézier methods are invariant under affine transformation, making them attractive in both CAD and optimization communities.

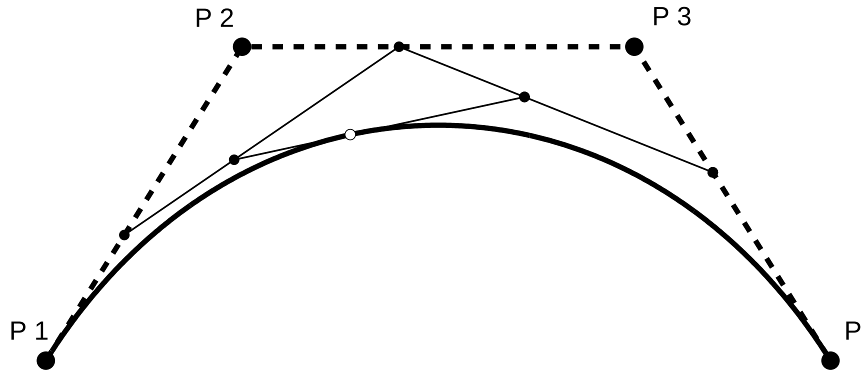


FIGURE 3.3: Repeated linear interpolations have been utilised by deCasteljau to find any point on a Bézier curve.

Using simple Bézier curves, it is difficult to represent curves such as circles, spheres, cylinders, cones etc. Rational functions - functions that are the ratio of two polynomials- alleviate this restriction; a rational Bézier curve can be defined as:

$$\mathbf{c}(u) = \frac{\sum_{i=0}^{n} B_{i,n} w_i \mathbf{p}_i}{\sum_{i=0}^{n} B_{i,n} w_i} \qquad 0 \le u \le 1 \tag{3.12}$$

where w_i are scalars, called the weights. A rational Bézier curve is invariant under perspective transformation, which is not true for non-rational Bézier curves.

3.2.3 B-Splines

Another drawback of Bézier curves is the fact that a $n-1$ degree Bézier curve is needed to pass through n data points, which at high n is very inefficient and can be numerically unstable. With this in mind it is also difficult to accurately define some complex shapes using Bézier curves, and they are difficult to control as they do not allow a lot of local control. To remedy these drawbacks curves and surfaces can be defined as piecewise polynomial, that is, consisting of several polynomial segments.

This is not as easy as adding multiple Bézier curves together, as the whole curve would not necessarily have the properties listed of Bézier curves, i.e. convex hull, variation diminishing, etc. Therefore the whole curve has to be represented as

$$\mathbf{c}(u) = \sum_{i=0}^{n} f_i(u)\mathbf{p}_i \tag{3.13}$$

where $f_i(u)$ are piecewise polynomial functions. As each function $f_i(u)$ is multiplied by its relative control point $\mathbf{p}_i$ then moving $\mathbf{p}_i$ only affects the curve shape where $f_i(u)$ is nonzero, giving local adaption of the curve only.

We would like to have a piecewise set of curves that have individual basis functions. We can do this by using a set of *knot values*. Assume that $U = u_o, ..., u_m$ is a *knot vector*, which is a non-decreasing sequence of real numbers, where u_i are called *knots*. The B-Spline basis function of degree p (order $p + 1$) is defined as

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u \leq u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{3.14}$$

It is clear to see that the zeroth degree basis functions are only involved in their left-open intervals $u\epsilon[u_i, u_{i+1})$ (u cannot take a value less than the current u), with the others being a linear combination of two (p-1) degree basis functions. The basis functions $N_{i,p}(u)$ are piecewise polynomials, and the knots may take the same value as its adjacent knot, so long that $u_i \leq u_{i+1}$. If a knot span is defined as

$$U = \{\underbrace{0, \ldots, 0}_{p+1}, \underbrace{1, \ldots, 1}_{p+1}\}$$

then the basis functions become Bernstein polynomials and thus the curve is a Bézier curve. Duplicating values in a knot sequence increases its multiplicity at that knot position. The multiplicity of a knot determines the number of times a basis function is continuously differentiable and thus the continuity of the base functions - $N_{i,p}(u)$ is $p - k$ times continuously differentiable, where k is the multiplicity of the knot. Multiple knots also reduce the span or number of intervals in which a basis function is non-zero – A zero basis function leads to a discontinuity in the curve.

The overall definition of a B-spline is

$$\mathbf{c}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{p_i} \tag{3.15}$$

$$U = \{\underbrace{a, \ldots, a}_{p+1}, u_{p+1}, \ldots, u_{m-p-1}\underbrace{b, \ldots, b}_{p+1}\}$$

where the degree p, number of control points, $n+1$, and number of knots, $m+1$ are related by

$$m = n + p + 1.$$

Figure 3.4 shows a set of B-Splines of different degree, and Figure 3.5 shows a Bézier curve with a B-Spline.

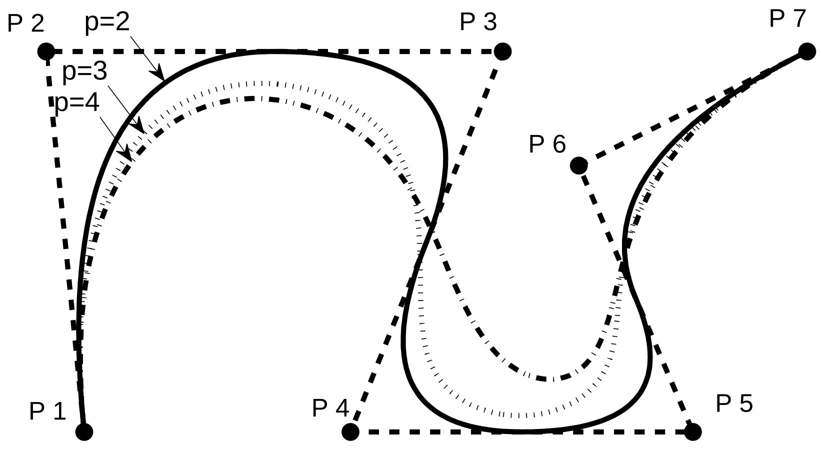


FIGURE 3.4: B-spline curves of different degree

3.2.4 NURBS

NURBS, or Non Uniform Rational B-Splines, are an amalgamation of the theory already discussed in this section. It is easy to see this when viewing the definition of a NURBS curve:

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u)w_i\mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u)w_i} \tag{3.16}$$

$$U = \{\underbrace{a, \ldots, a}_{p+1}, u_{p+1}, \ldots, u_{m-p-1}\underbrace{b, \ldots, b}_{p+1}\}.$$

It can be said that NURBS are the rationalisation of B-Splines by using a ratio of two polynomials, hence the name. This allows us to define complex curves like conics, and cylinders using B-Splines, allowing better control and flexibility than the rational Bézier

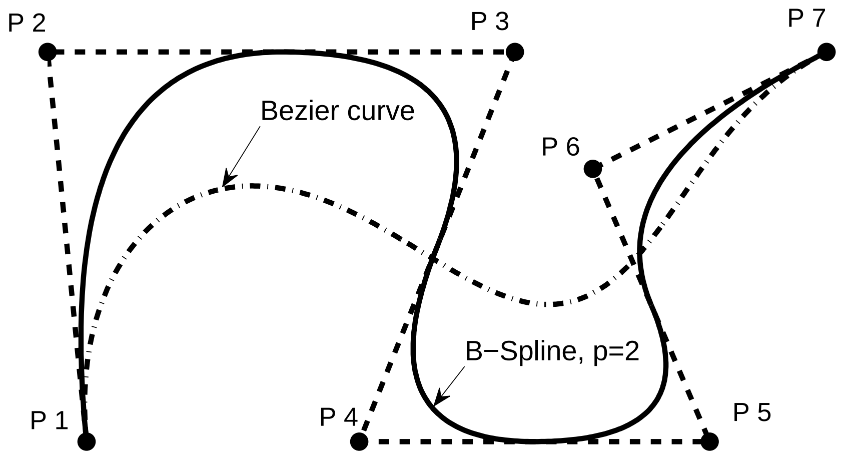


Figure 3.5: Bézier curve and a B-Spline curve.

curves. With the use of control point movement and weight modification we can modify the local shape and thus have a more interactive method to design shapes.

Polynomial techniques have been shown to be adequate when it comes to defining two-dimensional 2D curves. However, it must be noted that each control point has a combination of variables (x, y in 2D, x, y, z in three-dimensional 3D Cartesian coordinate system), increasing the number of input variables to the optimization process. Also, surfaces are defined by a multitude of curves and as such require a large amount of control points, that can result in solutions from an optimization possessing irregular or wavy geometries [Samareh, 2001].

3.3 Aero-Specific Modelling Techniques

The methods already described can be used for almost any application, and as such are available in most CAD software. Many other techniques exist that have been used or developed specifically for aerospace applications. Even though some of the methods described here may also be used in a more general setting, we only discuss the techniques when being used in an aero-specific capacity. The main application used in the literature to assess these methods is airfoil case studies, as the 2D characteristic enables simplicity in defining performance metrics, can be easily visualized, and the airfoil represents the cross-section of many external components on aircraft – if a method can accurately approximate an airfoil, likelihood is that it can be used for most external shape definitions of aircraft components. We first discuss the NACA series airfoils, which were initially characterised through a set of empirical equations (z=f(x)). We move on to parameterization techniques that can be used to accurately approximate many airfoil variants –

useful for when incorporated into an optimization procedure. Specifically, we describe the Hicks-Henne bump functions, the PARSEC method, and Kulfan's class/shape transformations. All these methods use, as inputs, the x-coordinates that span between the leading and trailing edges of the wing, normalised with the airfoil chord ($x \in [0, 1]$).

3.3.1 NACA Series Airfoils

In the 1930's the National Advisory Committee for Aeronautics (NACA)[1] developed a set of airfoils that were identified using a simple denotion, describing their shape: the first digit signified the maximum camber as a percentage of chord ($100m$); the second digit the distance of maximum camber from the leading edge ($10p$); the last two digits being the maximum thickness of airfoil as a percentage of chord ($100t$). This four digit identification led to the set of airfoils being defined as the NACA four-digit airfoils. The descriptive digits are inputs to a set of equations that generate the coordinate points for each airfoil:

1. The thickness distribution:

$$z = \frac{t}{0.2}\left(0.2969\sqrt{x} - 0.126x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4\right) \tag{3.17}$$

   provided that the position along the chord $x \in [0, 1]$.

2. The leading edge radius of the airfoil:

$$r_t = 1.1019t^2 \tag{3.18}$$

3. The mean camber line:

$$z_c = \begin{cases} m\frac{x}{p}(2p - x) & \text{if } 0 \leq x \leq p \\ m\frac{1-x}{(1-p)^2}(1 + x - 2p) & \text{if } p \leq x \leq 1 \end{cases} \tag{3.19}$$

4. Finally, the coordinates of the NACA four digit airfoils are given by

$$\begin{aligned} x_u &= x - z\sin\theta \quad , \quad z_u = z_c + z\cos\theta \\ x_l &= x + z\sin\theta \quad , \quad z_l = z_c - z\cos\theta \end{aligned} \tag{3.20}$$

   where $\theta = \arctan\left(\frac{dz_c}{dx}\right)$.

The five-digit airfoils are derived in a similar way, except the first digit, when multiplied by 0.15, gives the design lift coefficient c_l, the second and third digits give $2p$, and the fourth and fifth digits give $100t$. The six-series airfoils were based on the one-series aifoils, but were derived numerically from flow data rather than created then

[1]NACA was incorporated into the National Aeronautics and Space Administration (NASA) in 1958.

assigned their aerodynamic characteristics from analysis. The six-series were therefore characterised by their aerodynamic characteristics as well as their maximum thickness, and so no analytical expression was used to create them. A more detailed explanation of these airfoils, and other work of NACA on wing sections is given by Abbott and von Doenhoff [1959].

3.3.2 Hicks-Henne Bump Functions

As we delve further into the methods used to define airfoils, we will notice similarities in their definitions, with many using a basic airfoil shape as a starting point to be modified by a set of mathematical functions. A well-known analytical approach that does this is that of Hicks and Henne [1978] that was initially used to optimize a wing, modifying the airfoil sections at the root and at the mid-span. A baseline model $\mathbf{z}_{\text{basic}}$ is perturbed using a set of shape functions b_i which are controlled by participation coefficients,A_i:

$$\mathbf{z} = \mathbf{z}_{\text{basic}} + \sum_{i=1}^{n} b_i \tag{3.21}$$

where n is the number of shape functions. The ith shape function is defined by Sobester and Keane [2002] as:

$$b_i(x) = A_i \left[\sin\left(\pi x^{-\frac{\ln 2}{\ln x_{p,i}}} \right) \right]^{t_i}, \qquad x \in [0,1] \tag{3.22}$$

where x_p is the location of the peak of the bump and t is a parameter that controls the width of the bump (a large t value corresponds to sharp bumps). The participation coefficients are the design variables, that are initially set to zero to obtain the characteristics of the original structure, but are then modified to improve the chosen characteristics.

The Hicks-Henne functions have been popular in the academic community [Kim et al., 2000], [Wu et al., 2003], [Castonguay and Nadarajah, 2007], as they allow local smooth modifications to a geometry that is difficult to achieve using methods such as the discrete point method.

3.3.3 Orthogonal Basis Functions

A method that follows a similar approach to Hicks-Henne functions is the orthogonal basis function representation presented by Robinson and Keane [2001]. Here, once an initial approximation to NASA's SC(2) airfoil family was created by a least squares fitting, airfoil basis functions were derived from the residuals of the preceding fit. It was noted that acceptable accuracy was gained by only using four base functions. Chang et al. [1995] and Catalano et al. [2008] have also used orthogonal basis functions to model airfoils. Suppose we had a set of functions (f). Two functions are said to be orthogonal if their inner product equals zero, where the inner product is defined as

$$(f_i, f_j) = \int_0^1 f_i(x) f_j(x) dx \qquad \text{for } i \neq j \tag{3.23}$$

and $(f_i, f_i) \neq 0$. We can create a set of orthogonal basis functions $(h(x))$ using the Gram-Schmidt orthogonalization process, as we know that the functions are linearly independent:

$$h_i(x) = \begin{cases} f_1 & \text{if } i = 1 \\ f_i(x) - \sum_{j=1}^{i-1} a_{ij} h_j(x) & \text{otherwise} \end{cases} \tag{3.24}$$

where $a_{ij} = (f_i, h_j)/(h_j, h_j)$ is the projection of f_i in the direction of h_j. We now normalize the function set $h(x)$ to find the the orthanormal set $g(x)$:

$$g_i(x) = h_i(x)/(h_i, h_i) \tag{3.25}$$

Orthogonal basis functions have been used to reduce the number of parameters to improve the efficiency of an optimization process. Chang et al. [1995] managed to define the transonic Korn airfoil using ten shape functions, with the maximum residual being 5.7×10^{-4}. A subsonic NACA airfoil was modelled with a maximum error of 1.8×10^{-5} using only four shape functions. Robinson and Keane [2001] showed that only three base functions are needed to accurately model 9 of NASA's SC(2) supercritical airfoils. Catalano et al. [2008] undertook a gradient based search using four base functions, showing that the computational time is 6-10 times lower than a similar search using a Bézier curve representation.

3.3.4 PARSEC Method

Orthogonal methods may well reduce the number of parameters that need optimizing, but these parameters do not give us an intuitive understanding of the physical characteristics of the airfoils. Sobieczky [1998] defined a parameterization method using 11 parameters that were physical characteristics of the airfoil:

- Leading edge radius (r_{le})
- Upper crest position ($x_{\text{up}}, z_{\text{up}}$)
- Upper crest curvature ($z_{xx_{\text{up}}}$)
- Lower crest position ($x_{\text{lo}}, z_{\text{lo}}$)
- Lower crest curvature ($z_{xx_{\text{lo}}}$)
- Trailing edge direction (α_{te})

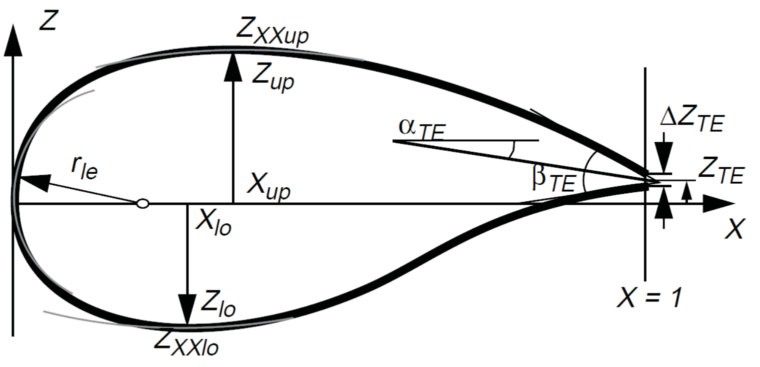


FIGURE 3.6: Variables used in the PARSEC method [Sobieczky, 1998].

- Trailing edge wedge angle (β_{te})
- Trailing edge offset (z_{te})
- Trailing edge thickness ((Δz_{te})

These parameters, included within the vector **e**, are included into the airfoil definition via a sixth order polynomial that independently defines the upper and lower surfaces:

$$\mathbf{z} = \sum_{i=1}^{6} b_i(\mathbf{e})\mathbf{x}^{i-\frac{1}{2}} \tag{3.26}$$

where b_i coefficients depend on the 11 parameters above.

The generation of this technique was from the knowledge and experience of its creator, and is restricted by some of the assumptions it uses so that an airfoil can be defined via its physical characteristics. For example, any shape that does not have a round leading edge cannot be modelled using the PARSEC method. This is a severe limitation, especially when applied to a global optimization strategy. The curvature of the leading edge can easily be modified by alteration of the leading edge exponent, as we will experience in the next technique that we discuss.

3.3.5 Kulfan's Class/Shape Transformation Method

Using the PARSEC method it is difficult to define an airfoil that has a shape significantly different to convention. For example, the parameter that defines the curvature at the leading edge brings difficulty in modelling airfoils that do not have a smooth gradient at the leading edge, such as NASA's supercritical and Natural Laminar Flow (NLF) airfoils. Kulfan [2008] has proposed that a baseline airfoil shape, defined by a simple analytic function called a class function, can be modified using a specific shape function

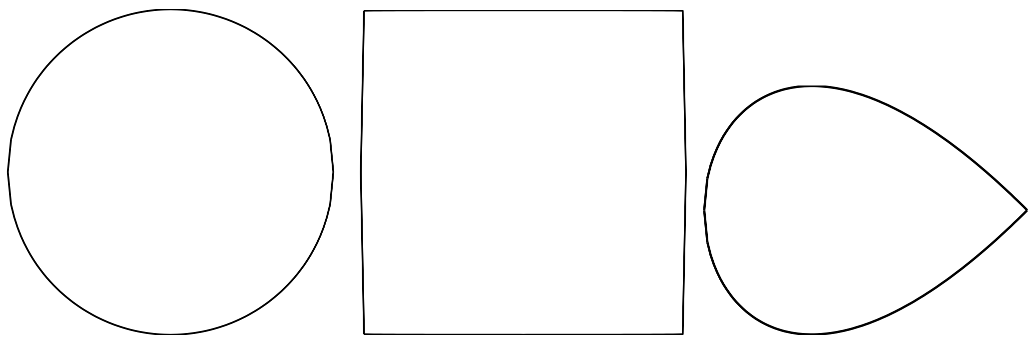

(a) $N1$=0.5,$N2$=0.5 (b) $N1$=0.001, $N2$=0.001 (c) $N1$=0.5, $N2$=1

FIGURE 3.7: The exponents to the class function ($N1$, $N2$) must agree with the shape being modelled.

to gain the airfoil's "uniqueness". The class function is defined as

$$C_{N2}^{N1}(x) = x^{N1}[1-x]^{N2} \tag{3.27}$$

where $N1$ and $N2$ are exponents that determine the basic shape. For a round nose and sharp trailing edge, characteristic of an airfoil, we define $N1$ and $N2$ as 0.5 and 1 respectively. Modifying the values of $N1$ and $N2$ allows for the production of other basic shapes, giving us a very simple and flexible method to define geometry (see Figure 3.7).

The purpose of the shape function is to model the quotient between the target airfoil $z_t(x)$ and the class function:

$$S(x) = \frac{z_t(x)}{x^{N1}[1-x]^{N2}} \tag{3.28}$$

This can be approximated, by a limited number of parameters, using Bernstein polynomials (Equation 3.10). The overall approximation is

$$z(x) = C_{N2}^{N1}(x)S(x) = x^{N1}[1-x]^{N2}S(x). \tag{3.29}$$

We can then use the least squares solution for the coefficients of the Bernstein polynomials to obtain $z(x)$.

3.4 Modifying Existing Geometry

Once we have successfully modelled a design we may wish to edit the shape, for example to either create an animation or to obtain finer control over the surface. These methods enable us to make changes in the shape without having to recreate the object, which may be time consuming and often difficult to perform.

We may require a local refinement within a region of a surface, or we may need a global deformation to change the whole characteristic of the object. Here we summarise two of the methods, one that can be used to refine, and the other used to fully deform a solid structure.

3.4.1 Hierarchical B-Spline Refinement

Due to the global influence of control points to each piecewise polynomial in a parametric curve it becomes difficult to perform local refinement within a piecewise curve. We must attempt to find a mapping of this region to a curve defined by many more control points that produces an exact re-representation of the curve. The increased number of control points will give us greater local control in this local region.

Forsey and Bartels [1988] developed a method whereby a surface patch is split into multiple patches. This, however, can only be performed if the original surface used basis functions that "can be re-expressed as a linear combination of one or more smaller basis functions" [Forsey and Bartels, 1988], i.e.

$$S(u,v) = \sum_{i=0}^{n}\sum_{i=0}^{m} B_{i,n}(u)B_{j,m}(v)\mathbf{P_{i,j}} \tag{3.30}$$

with the basis functions $B_i(u), B_j(v)$ defined using a number of smaller basis functions:

$$\begin{aligned} B_{i,n}(u) &= \sum_{r} \alpha_{i,n}(r)N_{r,n}(u) \\ B_{j,m}(u) &= \sum_{s} \alpha_{j,m}(s)N_{s,m}(v) \end{aligned} \tag{3.31}$$

The new surface patches do not replace the original surface, as the original surface is created with fewer control points. The replacement vertices saved are those that are used in the modification of the patch, and not all that were used to define that patch. In a bicubic B-Spline representation, once we have refined a surface patch into a representation having 16 patches, if we only modify the central vertex we localize the editing to this one patch, as the derivatives of the boundaries between the original patches remain unchanged. This one patch representation is known as an overlay. This method has been given the name *Hierarchical B-Spline Refinement* as the process can be iterated applying the technique to refined patches to further increase the resolution of the refinement.

3.4.2 Free-Form Deformation

The words “free-form” suggests a surface can be shaped like a piece of clay in a sculptor’s hands, and therefore planes, quadrics and tori are generally not considered to be free-from [Sederberg and Parry, 1986]. The clay metaphor suggests that this technique is for solids, but it is also applicable to surfaces. Many CAD models define shapes through planar surfaces and therefore are not free-form.

Barr [1984] presented an $R^3 \rightarrow R^3$ mapping, showing a useful way of bending, twisting and tapering solid objects. He used tangent and normal vectors; tangent vectors to construct the local geometry, and normal vectors to obtain surface orientation. However, this was not classed as a free-form technique.

FFD was initially described by Sederberg and Parry [1986], where they explain that it can sculpt solids bounded by any analytical surface. It can also be applied locally or globally to an object, and can have constraints to control the degree of volume changes. Sederberg and Parry [1986] define FFD by assuming a number of objects are contained in a cubic shape (parallelepiped). The cubic can be discretized by points in the different coordinate directions. Then a deformation function defined by trivariate tensor product Bernstein polynomials can be used to deform the cube:

$$\hat{\mathbf{x}} = \sum_{i=0}^{l} \sum_{j=0}^{m} \sum_{k=0}^{n} B_i^l(u) B_j^m(v) B_k^n(w) \mathbf{p_{ijk}} \tag{3.32}$$

where $\hat{\mathbf{x}}$ is the new position of the nodes, and $\mathbf{p_{ijk}}$ are the grid point positions (control points):

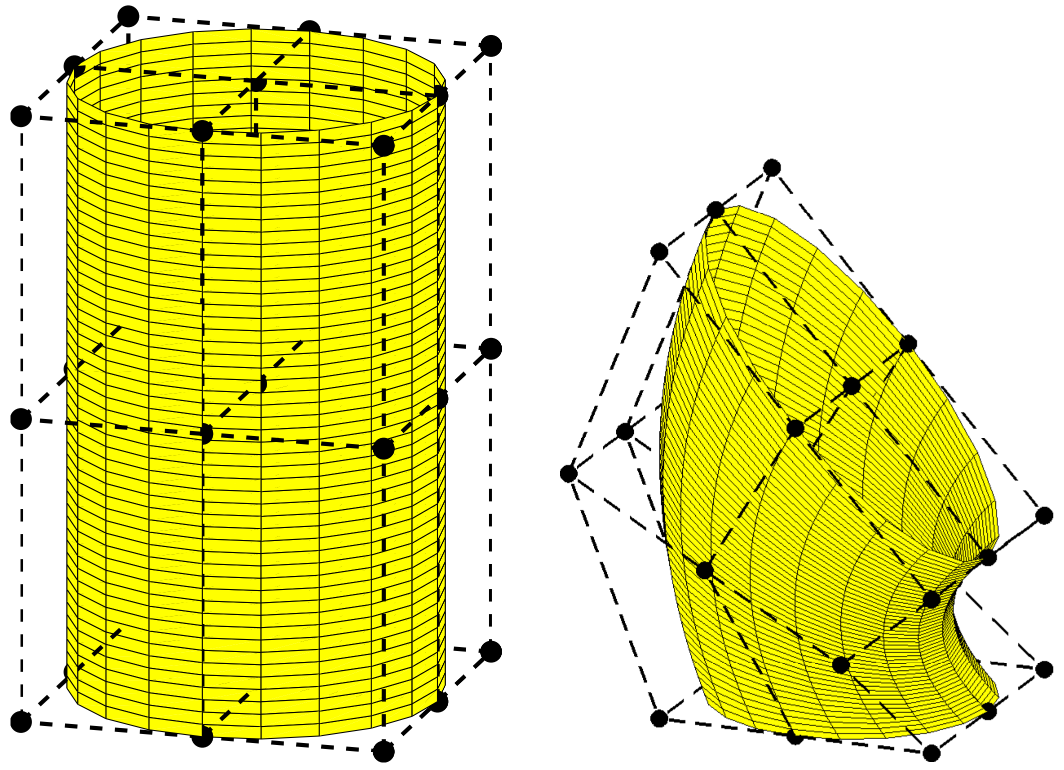
$$\mathbf{p_{ijk}} = \mathbf{x}_0 + \frac{i}{l} S + \frac{j}{m} T + \frac{k}{n} U \tag{3.33}$$

l, m, n are the number of grid points in the S, T, U directions respectively, with $0 \leq i \leq l$, $0 \leq j \leq m$, $0 \leq k \leq n$. $\mathbf{x_0}$ is the starting point of the coordinate system. When the cube is deformed the objects inside the cube deform with it, as shown in Figure 3.8(b) where the cylinder is deformed from its original shape shown in Figure 3.8(a).

Figure 3.8(b) also shows the control points, which are actually the coefficients of the Bernstein polynomials, with the edges of the parallelepiped defined as Bezier curves controlled by the points that initially were positioned on the edges. According to Lamousin and Waggenspack Jr. [1994] there are four main stages in performing FFD:

1. Construct a parametric solid; the objects to be deformed must be enclosed by the parallelepiped with control points and corresponding basis functions. Each point within the solid can be mapped to a parametric grid (u, v, w)

2. Embed the object within the solid. The parametric coordinates are determined for each point (x, y, z) describing the object.


(a) A parallelepiped encompassing a cylinder before deformation.

(b) The parallelepiped and cylinder after deformation.

FIGURE 3.8: An example of free form deformation.

3. Deform the parametric solid by displacing the vertices of the parallelepiped or lattice.
4. Evaluate the effect of the deformation on the embedded object. The parametric coordinates found in Step 2 are used with the deformed lattice control points to evaluate the new positions of the embedded point set and the topology, or connectivity, of the original object is used to reconstruct the object.

In the example in Figure 3.8 a parallelepiped is used to give a basic understanding of the method, as is the uniformly spaced gridding. Bernstein approximations also give a simplified feel to the method as other, more advanced methods can be used. Lamousin and Waggenspack Jr. [1994] used NURBS based FFDs to model a leg, in which control points were concentrated in areas of high detail.

3.5 Comparison of the Different Techniques

There have been a variety of reports that have studied various parametric techniques to be used in aerospace applications. Wu et al. [2003] compared the ability of the PARSEC method, Hicks-Henne bump functions, and the mesh point method to be used in inverse design problem applied to three different turbomachinery cross-sections. The PARSEC method was found not to be suitable for turbine blade optimization, as in some cases it produced unphysical shapes. This is not surprising as its formulation is based on conventional airfoils. The Hicks-Henne method reached an optimum faster, but the mesh-point method reached a higher accuracy, due to its higher order of design variables.

Song and Keane [2004] compared the Robinson and Keane [2001] Orthogonal Basis Function method with the B-Spline method for three different airfoils. Not surprisingly, the orthogonal method could not accurately find the leading and trailing edge shapes, as the basis functions were derived using the NASA family of supercritical airfoils, which have special characteristics in these regions (see section 4.1.1). The B-Spline method was more accurate in its definition, but still had trouble in approximating the leading edge. This may have been improved if an analytical term, similar to the leading term of the Kulfan [2008] class function, had been used.

Castonguay and Nadarajah [2007] compared four techniques: Hicks-Henne bump functions, B-Spline curves, mesh-point, and the PARSEC method. This was done in the context of an adjoint based shape optimization. Again, the PARSEC method was not suitable in inverse design problems as, although the shock moved to a similar position to the target pressure distribution, the leading edge was not modified to obtain the exact pressure distribution. The accuracy of the Hicks-Henne method was lower than the B-Spline and mesh point methods, but produced similar results to the B-Spline method in drag minimization. Mousavi et al. [2007], reported the results of a similar study. This time they compared the Kulfan [2008] transformation with mesh point and B-Spline methods. In the conclusions they report that Kulfan's transformation could model an ONERA M6 airfoil but to a lower level of accuracy, and that it failed to attain the same level of accuracy as the B-Spline method in a 3D drag minimization. It is difficult to understand from this paper if the Kulfan transformation can be regarded as having poorer accuracy, as the accuracy is dependent on the shape function used, in this case the Bernstein polynomials. They conclude that higher orders of polynomial developed high frequency, lower amplitude oscillations. This may have been remedied by using a similar method as was done for Bézier curves in Section 3.2.2; that is, to represent it as piecewise polynomial. This may allow greater accuracy in the initial definition as well as a higher order to be used in the optimization algorithm, which should help in the optimization to reduce drag. This idea has been presented by Straathof et al. [2008], however no detailed comparisons between using different shape functions were made.

3.6 Summary

In this chapter we have discussed the many attributes we require from a shape parameterization model for it to be used effectively in an optimization procedure. We have introduced a few of the many techniques available, highlighting their strengths and weaknesses. These were emphasised by the review of articles that compare the performances of some of the techniques.

We have now given an overview of all the stages within an optimization procedure, introducing a variety of different methods that can be used at each stage. To cement the main processes involved in planning a design optimization study, we present a simple example in the next chapter. Note that we are only concerned in the next chapter with the setup of the optimization problem, and so little discussion is given to the optimal solution found in the study.

Chapter 4

Application - Optimization of a NASA SC(2) Airfoil

In chapters 2 and 3 we have given an overview of the main aspects in an optimization procedure, giving advantages and disadvantages of many of the techniques used at different stages along the optimization process. We now present an application of these methods on a simple 2D airfoil problem that underpins the basic design optimization process. More specifically, we optimize the shape of an airfoil belonging to NASA's supercritical family, for a target lift coefficient c_l and thickness-to-chord ratio t/c different to that airfoil's design specification. The aim of the optimization procedure is to also highlight the applications of the parameterization procedure developed by Sóbester and Powell [2012] that maps the design identifiers of NASA's supercritical family (design c_l and thickness-to-chord ratio t/c) to the parametric values of the Kulfan [2008] transformation, allowing the generation of an airfoil with any set of c_l, t/c combinations within a specific range.

Of course, the first task before we even begin to plan the optimization process is to understand the problem we are attempting to solve: we initially give an overview of NASA's Supercritical airfoil family, describing the design philosophy as well as the probable design methodology, something of which is not obvious from the available literature. Accurately describing the design methodology is essential, and so we go on to verify this process using CFD analysis on airfoils generated by the Sóbester and Powell [2012] re-parameterization, mentioned above. After verification of the design process, we give a full description of the optimization problem and procedure, including the parameterization of the model, choice of objective function (including penalties), and the convergence criteria. After presenting the optimization results we only briefly compare the aerodynamic differences between the performance of the optimization solution and the performance of an airfoil generated by the Sóbester and Powell [2012] re-parameterization, as the main focus here is on the optimization process, and not on the results.

4.1 NASA's Supercritical Airfoils

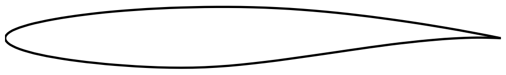
The SC(2) Family of Supercritical Airfoils is the result of research conducted by NASA starting in the 1960s aimed at the development of "practical airfoils with two-dimensional transonic turbulent flow and improved drag divergence Mach numbers while retaining acceptable low-speed maximum lift and stall characteristics" [Harris, 1990]. They trace their lineage back to the work of Whitcomb and Clark [1965], who noted that a three quarter chord slot between the upper and lower surfaces of a NACA 64A series airfoil gave it the ability to operate efficiently at Mach numbers greater than its original critical Mach number - hence the term 'supercritical', or 'SC' for short. Critical Mach number is defined as the free stream Mach number where local sonic velocities develop around the airfoil. Therefore this type of airfoil was extremely useful in conditions of transonic flow, reducing the drag experienced at these speeds.

The development of these airfoils continued into the 1970s and 1980s resulting in three phases of designs. A variety of experiments were performed with these airfoils in the Advanced Technology Airfoil tests (ATAT) [Ladson and Ray, 1981] where a number of airfoil concepts were tested over a wide range of Reynolds numbers. They are identified using the designation 'SC' followed by the development phase for which that particular airfoil was developed, in brackets. It is proceeded by two characteristics of the airfoil; the design lift coefficient (multiplied by ten), and the maximum thickness to chord ratio (as a percentage). Thus, the SC(2)-0410 indicates a second series airfoil with a thickness-to-chord ratio of 10% that has been designed for a lift coefficient of 0.4.

4.1.1 Design Philosophy

The design philosophy of the supercritical airfoils was initially described by Whitcomb [1974] in 1974. A more complete account was given by Harris [1990], many years after their conception. The airfoils were designed to improve supercritical flow, whilst maintaining a design lift coefficient c_l. This was achieved by delaying the drag rise on the upper surface through a reduction in the severity of the airfoil curvature in the middle region, reducing the flow acceleration and thus the local Mach number. This, in turn, reduces the severity of the adverse pressure gradient there and thus the associated shock is moved rearward with reduced strength.

The region of the flow above the airfoil can be modelled by a set of expansion waves (characteristic lines) emanating from the leading edge. These waves reflect back off the sonic line (a line that separates the sonic flow from the subsonic flow) as compression waves onto the airfoil surface, where they again reflect in the form of expansion waves. Expansion waves have the effect of increasing the velocity of the flow, whilst compression waves decrease the velocity. The design goal in the generation of these airfoils was to balance out these competing waves so that a flat top pressure distribution was


(a) RAE2822 airfoil

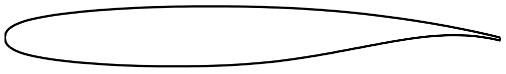

(b) SC(2)-0612 airfoil

FIGURE 4.1: The SC(2) airfoils have a number of characteristics different to other airfoils, notably the leading edge curvature and the cusp of the aft lower surface.

achieved forward of the shock position, that helps to stabilize the boundary layer so it does not separate through the shock. Geometrically, this was achieved through a large leading edge curvature (creates strong expansion waves), and by the flat mid-chord region (reduces the accelerations that would have needed to be overcome by the reflected compression waves) [Whitcomb and Clark, 1965].

Theory suggests that the pressure coefficient should return to stagnation conditions at the trailing edge, but at this point the boundary layer separates and the pressure rise is less. It was noted that, in boundary layer theory, if the pressures had to rise from the levels before the shock to the usual positive pressures at the trailing edge, separation would occur. Therefore, to make the pressure at the trailing edge only slightly positive, and thus reduce the pressure recovery, the lower surface has the same slope at the trailing edge as the upper surface, making the trailing edge finite. The thick finite trailing edge increases the amount of lift with very little increase in the subsonic drag.

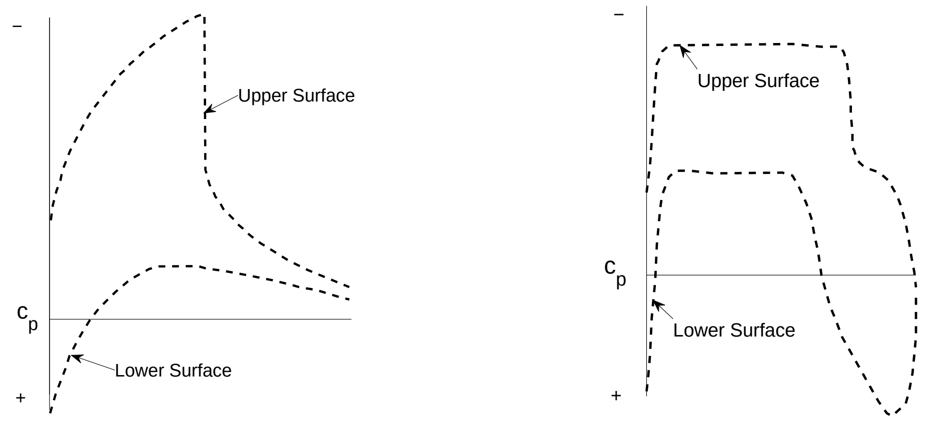
The discussed characteristics reduce the circulation on the airfoil, which results in a loss of lift. To remedy this, the rear portion of the lower surface was curved in such a way to increase the circulation there and thus maintain the lift generated. A consequence of this is an aft loaded pressure distribution, which has the effect of reducing the angle of attack required to achieve the design lift coefficient.

The physical features discussed were, at the time of creation, unique to these airfoils and not apparent in any other airfoil. The RAE2822, shown in Figure 4.1(a), is an example of a transonic airfoil used at this time. Compared to the SC(2)-0612 airfoil in Figure 4.1(b), we can easily see that the leading edge radii are dissimilar, as well as the curvature on the lower surfaces and the thickness of the trailing edges.

4.1.2 Design Process

Although the design philosophy is discussed by Whitcomb [1974], no indication was given as to the procedure used to design the airfoils for a specific design lift coefficient. The design criteria were discussed in Harris [1975a] and Harris [1975b], in the form of three principal guidelines:

- Off-design sonic plateau criterion; at some incremental normal-force coefficient below the design normal-force coefficient the pressure distribution on the upper



(a) Pressure profile of a NACA64-A airfoil. (b) Pressure profile of a Supercritical airfoil.

FIGURE 4.2: The typical pressure coefficient plots for a NACA64-A series airfoil and for the Supercritical airfoils [Harris, 1990].

and lower surfaces is to be flat with the upper surface pressures just below the sonic value.

- The gradient of the aft pressure recovery is to be gradual enough to avoid separation problems for lift coefficients and Mach numbers up to the design values.
- The airfoil must have sufficient aft camber so that at design conditions the angle of attack (α) should be about zero.

None of the guidelines listed above suggest any flow conditions at which the airfoils were designed, apart from the vague description that at design conditions $\alpha \approx 0$. Whitcomb [1974] noted that due to the concavity on the lower surface, which provides the circulation needed to maintain lift, the α required to achieve the design lift coefficient is a slightly negative value of around -0.5^{o}. However, this paper was presented before the second phase of airfoils were developed.

Harris [1990] reported that initially the airfoils were designed using experimental methods, with the first series of integral airfoils being defined empirically by several equations (These equations were never published due to the airfoils still being in their development stage). From around 1975 a “viscous airfoil analysis program” developed by Bauer et al. [1975] was utilised that designed airfoils by solving an inverse design problem – the program attempts to find the body which generates a set of pre-specified flow characteristics. The computer program uses “the hodograph transformation and analytic continuation into the complex domain” [Bauer et al., 1975]. This analysis program is a modified version of that described by Bauer et al. [1972], which was the basis for the potential flow program, VGK [ESDU96028, 2004]. According to Harris [1990], during the design of the supercritical airfoils using this method, the Mach number was allowed to float to achieve the generalized pressure distributions at the design conditions seen in

Re	30×10^6
M	float
α	≈ 0
$\mathrm{d}c_d/\mathrm{d}M_{dd}$	0.1

Table 4.1: Airfoil Design Conditions.

Figure 4.2 and at the sonic plateau. The Reynolds number (Re) for design conditions were suggested to be 30×10^6 for airfoils with a thickness-to-chord ratio of .06 and above, with $Re = 10 \times 10^6$ for ratios below this value. This is contradicted in a paper by Jenkins [1989], that used Re = 40×10^6 for the design conditions of the SC(2)-0714 airfoil. It is not clear if the drag divergence Mach number (M_{dd}), defined as the Mach number where the gradient of drag coefficient with respect to M equals 0.1, is part of the design conditions as it is only given a mention by Harris [1990].

In summary, the literature indicates that NASA's SC(2) airfoils were created by iteratively modifying the surface nodes in the potential flow code created by Bauer et al. [1972] to gain a target pressure distribution. No definitive design procedure for the creation of these airfoils has been found, but from the evidence we have discussed, we can assume the airfoils were designed to meet a specified target c_l, where M was left to float until the drag divergence Mach number M_{dd} was found. If the airfoil achieved a pressure profile dissimilar to that shown in Figure 4.2(b) the angle of attack α was perturbed and a new search for the M_{dd} at that angle was performed. This procedure continued until the pressure profile agreed with Figure 4.2(b). The design conditions are summarised in Table 4.1.

4.2 Design Space Dimensionality Reduction through Physics-based Geometry Re-Parameterization

As an example of applying low dimensionality design parameterization in a design context, we briefly outline here an algorithm described by Sóbester and Powell [2012]. The algorithm uses the Kulfan transformation shape parameterization technique discussed in Section 3.3.5, applied to the SC(2) airfoils, and re-parameterizes it terms of the design features used to identify these airfoils: that is, thickness-to-chord ratio (t/c) and design lift coefficient (c_l). This "re-parameterization" allowed for the creation of an infinite number of airfoils that could be said to belong to the SC(2) family (they have the same design features) with all having different combinations of c_l and t/c [1].

This parameter reduction was achieved by noticing that the half-thickness distributions of the SC(2) airfoils are identical for family members with the same maximum thickness.

[1]Of course the range of these design designations were restricted by the range of the training examples used to generate the models.

The camber was found to vary with both the design lift coefficient and thickness, and so a Kriging model was used to map c_l to the Kulfan transformation variables and thus completing the mapping

$$(t/c, c_l) \rightarrow [z(x)].$$

Once this mapping was achieved it allowed for other airfoils to be defined that fell between those in the Supercritical family.

4.3 Verifying the Design Process of the SC(2) Airfoils

The newly defined airfoils from the work above were used to verify our interpretation of the aerodynamic design procedure, by proving that these airfoils do indeed possess the same aerodynamic features at design conditions as those for the SC(2) family as well as agree with their design c_l for which we use for their identification (t/c is a geometrical feature and so is of no concern in this aerodynamic procedure). We applied a Reynolds-Averaged Navier-Stokes (RANS) approach, using a density-based implicit solver and $k - \epsilon$ turbulence model in the CFD package FLUENT. This setup had been validated to accurately model experimental data from the ATAT program [Ladson and Ray, 1981], with the results discussed in Appendix A.

As mentioned in Section 4.1.2, from what we understand, the design conditions for the SC(2) airfoils are found by allowing M and α to float whilst finding the drag divergence Mach number(M_{dd}) and a flat upper surface pressure plateau at the design c_l, as shown in Figure 4.2. We obtained an intial estimate of M_{dd} from the Korn Equation [Mason, 2009]:

$$M_{dd,K} + \frac{c_l}{10} + \frac{t}{c} = 0.95 \tag{4.1}$$

By viewing the relationship of drag variation with M at transonic speeds in a book by Anderson [2001] we approximated the region encompassing drag divergence as a second order polynomial function, by computing CFD runs at three Mach numbers; $M_{dd,K}$, $M_{dd,K} + 0.001$, $M_{dd,K} - 0.001$. Differentiation of the resulting polynomial yielded the drag divergence Mach number (where $\mathrm{d}c_d/\mathrm{d}M = 0.1$).

The flow field was then solved at this new condition, with the result added to the existing solutions and a polynomial fitted once more. This pattern was continued until the position of the drag divergence Mach number remained constant over consecutive iterations. If this was not achieved after three iterations a local search was used. A summary of the method is shown below:

1. Using the Korn Equation find the initial Mach number $M_{dd,K}$ and run a CFD simulation at this Mach number.

2. Run a further two CFD simulations at $M_{dd,K} + 0.001$, $M_{dd,K} - 0.001$.

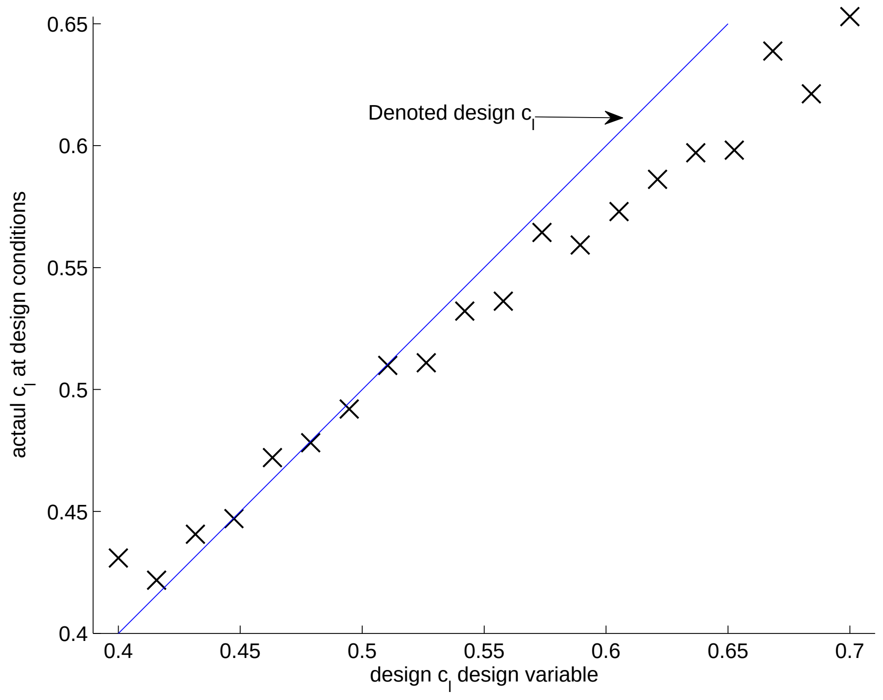


FIGURE 4.3: Actual c_l at design conditions versus 'c_l' design variable for 20 airfoils created using the method of Sóbester and Powell [2012].

3. Fit a polynomial curve to the c_d/M relationship, differentiate the polynomial, and find the Mach number when $\mathrm{d}c_d/\mathrm{d}M = 0.1$. Run a CFD simulation at this Mach number.

4. Fit another polynomial using the three closest values to M_{dd}, found by using linear gradients between points.

5. Repeat the previous step.

6. Do a local search until the gradient c_d/M equals 0.1

The resultant pressure coefficient graph at this position was compared to the ideal graph of Figure 4.2, with the above heuristic repeated at a new α if the graphs were dissimilar.

Using the methodology above we found the design conditions for 20 airfoils created using the re-parameterization [Sóbester and Powell, 2012] . The strong correlation in Figure 4.3 indicates that the generated airfoils do indeed reflect the design trends of the SC(2) airfoils, although the linear relationship has a slightly lower gradient than the desirable denoted values, indicated by the straight line.

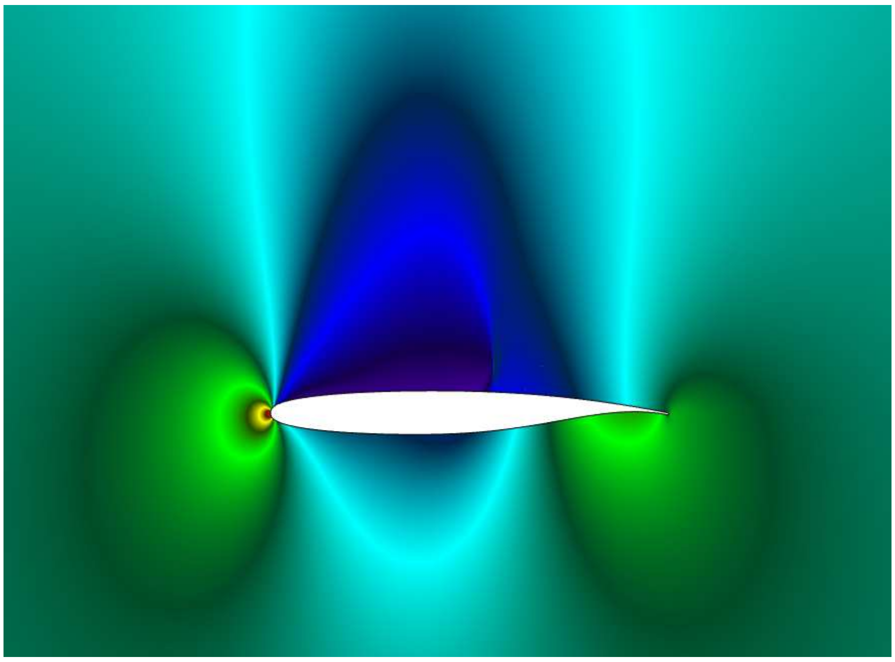


FIGURE 4.4: The pressure pattern around the $SC(2)_s - 0511$ airfoil generated by the Sóbester and Powell [2012] mapping.

4.4 Optimization of the SC(2)-0412 airfoil

Consider a situation where we require an airfoil, with the supercritical characteristics of the SC(2) airfoils, but for a design c_l of 0.5 and t/c of 11%, which are not possessed by any airfoil belonging to the SC(2) family. Of course, we can now exploit the Sóbester and Powell [2012] re-parameterization for such a problem: we have generated an SC(2) type airfoil with a design c_l of 0.5 and a thickness-to-chord ratio of 11%. We have given this airfoil the designation SC(2)$_s$-0511, following the system used for the original SC(2) airfoils, where the subscript s indicates it was generated by the Sóbester and Powell [2012] re-parameterization. Using the method in Section 4.3, the SC(2)$_s$-0511 design conditions were found:

- angle of attack, $\alpha = 0.3$
- Mach number, $\mathrm{M} = 0.7732$
- Reynolds number, $\mathrm{Re} = 30 \times 10^6$.

The resulting drag coefficient c_d of the SC(2)$_s$-0511 was 0.00129. Figure 4.4 shows the pressure contours around the SC(2)$_s$-0511 airfoil, which provides an excellent illustration of the flow regimes inherent to all the SC(2) airfoils; a shock present on the upper surface aft of the mid-chord region, and the increased circulation generated by the lower surface cusp near the trailing edge that recovers the lift lost by the reduced curvature of the upper surface.

However, prior to the technique described by Sóbester and Powell [2012], a natural solution would have been to optimize the shape of an SC(2) close to the specified characteristics. In the optimization procedure presented here we did just that: we optimized the shape of the SC(2)-0412 airfoil at the design conditions of the $\mathrm{SC(2)}_s - 0511$, in an attempt to find an airfoil that has a $c_d \leq 0.00129$ at a c_l of 0.5. The t/c was allowed to reduce to a limit of 11%. This airfoil was chosen, not only because it is one of the existing airfoils that closely matches the requirements, but also as it allowed the optimization to begin within a feasible region ($t/c > t/c_{\mathrm{SC(2)_s-0511}}$), and was able to reach an α that gave a c_l of 0.5 at the flow conditions of this study, without any instabilities in the CFD simulation. We knew that by using an optimization procedure the time spent on finding a similar airfoil will be significantly longer and so the aim of this exercise was to find the time-savings afforded by the re-parameterization approach.

Below we give a rationale into the choices we made for each stage in the optimization process, and present the results of the optimization process.

4.4.1 The Optimization Procedure

To assess the performance of each airfoil generated by the optimizer we use the same FLUENT setup as in Section 4.3, the formulation of which is detailed in Appendix A. This method frequently took up to three hours to find the performance of a single airfoil, as the angle of attack of each airfoil had to be perturbed to meet the target c_l value of 0.5. Such a burden to the computational efficiency would be unsuitable in a global optimization procedure, as the cost of running multiple Fluent calculations, and the limited computational resources we had at the time, would have led to an unsatisfactory lapse to achieve the final solution. Only having one design to initialise the optimization process also led to us having to use a local search, even though we were unsure of the modality of the solution.

To ensure we could compare the results here with those of the $\mathrm{SC(2)}_s - 0511$ airfoil, it was critical to use the same parameterization technique; the Kulfan transformation used by Sóbester and Powell [2012] had 16 variables that defined the upper and lower surfaces, trailing edge thickness, and a leading edge parameter of an airfoil.

It was necessary to constrain the maximum thickness of the airfoils created within the optimization process, as we knew that the profile drag desires a low frontal area, and thus thin airfoils. This constraint was adopted into the optimization process via a step penalty function incorporated into the objective function. The objective function was therefore:

$$f(\mathbf{x}) = \mathrm{c_d} + P \tag{4.2}$$

where

$$P = \begin{cases} 1 & \text{for t/c} < \text{t/c}_{\text{max}} \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

Using this step function would make it difficult for a gradient method to find solutions close to the constraint boundary, and so a zeroth order method was used here: the Nelder and Mead [1965] simplex search (see Section 2.2.1). Finally, we used the c_d value of the $\text{SC}(2)_s - 0511$ airfoil as the termination criteria, and limited the search to 50 objective function evaluations, as the aim of this work was to assess the time-savings of using the re-parameterization of Sóbester and Powell [2012] rather than to do such an optimization study presented here. 50 function evaluations equated to over 6 days computational time – a significant amount of time compared to the seconds it takes to generate the re-parameterization airfoil.

4.4.2 Results

The history of the Simplex search is shown in Table 4.2. Iteration zero is the evaluation of the starting design, i.e. the SC(2)-0412 airfoil. Intuitively we would assume that this airfoil returned a significantly larger objective function value than the c_d value found for the $\text{SC}(2)_s - 0511$ airfoil (c_d =0.0129), as the starting airfoil was designed for a lower c_l and had a larger maximum thickness (the profile drag would be greater). Remember from Section 2.2.1 the algorithm has to first rank the vertices, or variables, to decide the best move to make for the search. This is shown here by the 16 function evaluations made in the first iteration. After the first iteration the search only improved upon the current best four more times until the maximum number of iterations was reached, where it did not find an airfoil that improved upon the SC(2)$_s$-0511 characteristics. Figure 4.5 shows that the best airfoil from the optimization process produced a shock at the same position as for the SC(2)-0511 airfoil. However, the low pressure region that occurs mid-chord on the lower surface reduces the c_l of the airfoil, and had to be recovered via the leading edge of the upper surface by increasing α. This effect can also be described by comparing the pressure contours in Figures 4.4 and 4.6, where the lower pressure region on the lower surface in Figure 4.6 (depicted as shades of blue) is not present in Figure 4.4.

Note in Table 4.2 that a total of 50 function evaluations were made, and so reached the maximum number of function evaluations. This total elapsed time for optimization process was over six days. Moreover, the optimization failed to find an airfoil that improves upon the characteristics of the SC(2)$_s$-0511. If we alternatively deployed the SC(2)$_s$-0511 as the starting airfoil in the same optimization problem, that is, improve drag at a c_l of 0.5 constrained by $t/c \geq 11\%$, then we can effectively save at least 6 days of computational time.

Iterations	Function count	min f(x)
0	**1**	**0.0155923**
1	**17**	**0.0150771**
2	18	0.0150771
3	**20**	**0.0140839**
4	21	0.0140839
5	22	0.0140839
6	23	0.0140839
7	24	0.0140839
8	25	0.0140839
9	26	0.0140839
10	27	0.0140839
11	**29**	**0.013633**
12	30	0.013633
13	31	0.013633
14	**33**	**0.0135157**
15	34	0.0135157
16	35	0.0135157
17	36	0.0135157
18	37	0.0135157
19	38	0.0135157
20	39	0.0135157
21	40	0.0135157
22	41	0.0135157
23	42	0.0135157
24	43	0.0135157
25	**45**	**0.0134534**
26	46	0.0134534
27	47	0.0134534
28	48	0.0134534
29	49	0.0134534
30	50	0.0134534

Table 4.2: Simplex performance history of the search to find an optimal shape for a 11% thick airfoil with a design c_l of 0.5. The starting airfoil was a SC(2)-0412.

4.5 Summary

The purpose of this chapter was to present an optimization procedure on a simple 2D airfoil problem. We have described the rationale for the decisions used to setup the optimization procedure, and presented the results of the Nelder and Mead [1965] search.

We have also presented here a process that finds the design conditions of NASA's SC(2) airfoils, and have used this to determine if the airfoils generated by the Sóbester and Powell [2012] re-parameterization technique agree with their designations.

Now that we have given an overview of aircraft design optimization theory, and applied it to a simple problem, the reader should be sufficiently prepared to not only follow the

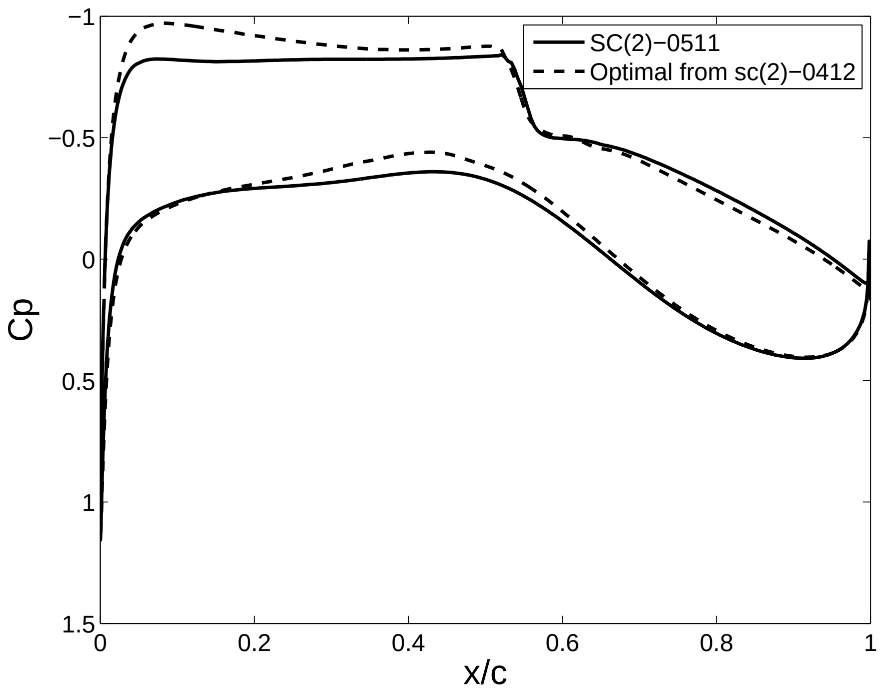


FIGURE 4.5: The c_p profiles around 1) the airfoil generated by the Sóbester and Powell [2012] mapping and 2) the resulting airfoil from the local optimization process starting from the SC(2)-0412 airfoil.

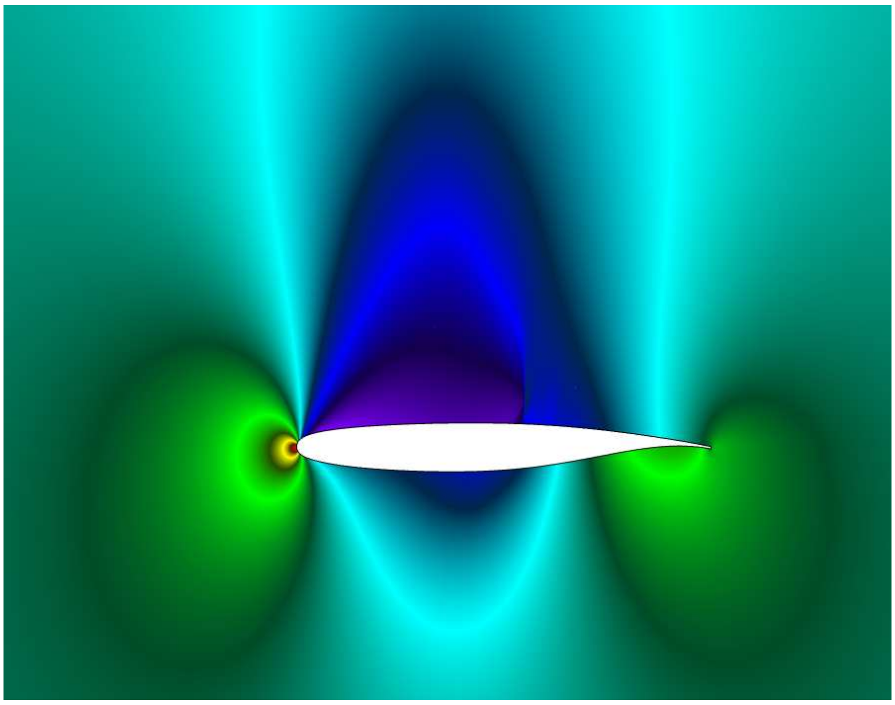


FIGURE 4.6: The pressure pattern around the airfoil generated by the optimization process.

work in the next chapters, but to also understand the reasoning behind the research and where it can be applied. In the next chapter we modify the Kulfan parameterization technique used here. Although we know that the Kulfan technique can provide appropriate accuracy when modelling the SC(2) airfoils, it does so with a significantly large number of variables (we used 16 variables in the case in this chapter). What we present in the next chapter is a modification that allows us to describe the SC(2) airfoils with a smaller number of variables without inhibiting the accuracy, in an effort to reduce the curse of dimensionality and thus allow for a more efficient optimization procedure.

Chapter 5

Application-Specific Class Functions for The Kulfan Transformation of Airfoils

In Chapter 3 we described the difficult task of choosing a geometry representation technique for design optimization. This difficulty is emphasised for geometries such as airfoils where a vast array of techniques exist. We expressed the importance of the selected technique to possess a number of attributes. The curse of dimensionality is a problem that significantly affects the optimization search, as increasing the number of variables exponentially increases the size of the design space. Having a large variable set can also lead to a high proportion of physically unrealistic designs, making the search highly time intensive. We can combat the creation of these designs by restricting the ranges of the variables but, due to the highly complex nature of the interaction between variables, this may not be effective. Conversely, reducing the number of variables in a parametric technique often tends to restrict the range of valid designs that can be created. We must choose a geometry representation technique that minimizes the number of variables and allows flexibility in the shapes it can create whilst maintaining geometric robustness (generating a high proportion of physically viable shapes).

We presented in Section 3.3.5 the Kulfan [2008] transformation technique that achieves a certain measure of geometric robustness by initially classifying a component using an analytical equation[1]. This maintains the underlying geometric shape of the component whilst allowing local adjustments using the coefficients of a parameterized model, the so-called shape function. Kulfan uses Bernstein polynomials for the shape function as, for any order, their sum is equal to one. Any alteration in the Bernstein coefficient values from one alters the shape from the class function. Note that increasing the polynomial

[1]Less fortunate choices of shape function coefficients can still yield ‘unphysical’ shapes

order by one adds an extra parameter to be searched and thus exponentially increases the design space via the curse of dimensionality, as discussed previously.

Since its introduction, the Kulfan transformation approach has been thoroughly examined due to its simple yet powerful ability to produce accurate shapes using a limited number of parameters. Specific characteristics of the method have been discussed by Ceze et al. [2009] whilst comparisons with other methods in MDO have been studied by Mousavi et al. [2007]. More recently, Lane and Marshall [2009] applied the Kulfan transformation to optimize components that incorporated discontinuous surfaces by using a new surface parameterization when noticing a disproportionate change in surface curvature. The method has also been reviewed by Bogue and Crist [2008] for use in 2D and 3D transonic optimization.

For subsonic airfoils Kulfan and Bussoletti [2006] found that the Kulfan transformation could represent the NACA0012 airfoil to within wind tunnel tolerances using a fourth order Bernstein polynomial as the shape function for both the upper and lower airfoil surfaces. However, similar accuracy was not achieved for the transonic RAE2822 and NASA's SC(2)-0714 airfoils. The difficulty lay at the leading edge, where, as discussed in Section 4.1.1, these supercritical (SC) airfoils are heavily modified from the uniform curvature of the subsonic airfoils to produce strong expansion waves to be reflected back at the sonic line as compression waves [Harris, 1990], and to reduce curvature in the mid-chord region to prevent premature shocks forming. Increased polynomial orders were needed to accurately define the leading edge for both supercritical airfoils, which is undesirable for optimization purposes. Bogue and Crist [2008] had similar difficulty when optimizing the RAE 2822 airfoil, with the optimal solution being achieved using a polynomial order of six for upper and lower surfaces. Higher polynomial orders only improved the solution by one drag count, which was not considered optimal due to the minimal benefit being achieved with increased dimensionality of the search.

To allow the Kulfan transformation to be used in transonic airfoil design using a similar number of parameters to the subsonic case we can attempt to find a new class function to better represent the family of supercritical airfoils, or pre-modify the airfoil curvature so that it fits the class function better; this modification must then be put back after the Kulfan transformation has been performed. The latter of the two has already been considered (see, for example, Sóbester [2009]), through using an additional leading edge term, and the former has been studied by Lane and Marshall [2009], but with the class function only being modified by the choice of exponents used for the original class function.

If we use the approach of Lane and Marshall [2009] that attempted to find optimized values of the exponents, we end up with a different class function for each airfoil. This may be useful in proving that the general class function can be modified to produce accurate results with a low number of parameters but it does not give us a universal method to describe the supercritical family.

With this in mind we present a method of finding a new class function to be used in the Kulfan transformation to approximate the SC(2) family of airfoils. We require the new class function to be simple in terms of the number of unary operators $(\sin, \cos, \exp)$ and the length of the expression. This is so that we have some intuitive feel for the shape of the object being designed and to maintain the advantages of using an analytical class function as opposed to defining one that is an interpolation of an array of coordinates. Once we show the aspects that make it difficult for SC(2) airfoils to be represented by the Kulfan transformation, we compare two methods for finding an appropriate class function for a single member of the SC(2) supercritical airfoil family; by using standard Genetic Programming(GP), discussed in Section 2.4.4, and by using GP combined with a local search. We then use GP to find a class function that accurately describes a selection of SC(2) supercritical airfoils, specifically those designed for civil transports and business jets. We assess the practicality of using such a class function in an optimization context by applying it to an airfoil optimization problem, and comparing the results to that of using the general airfoil class function.

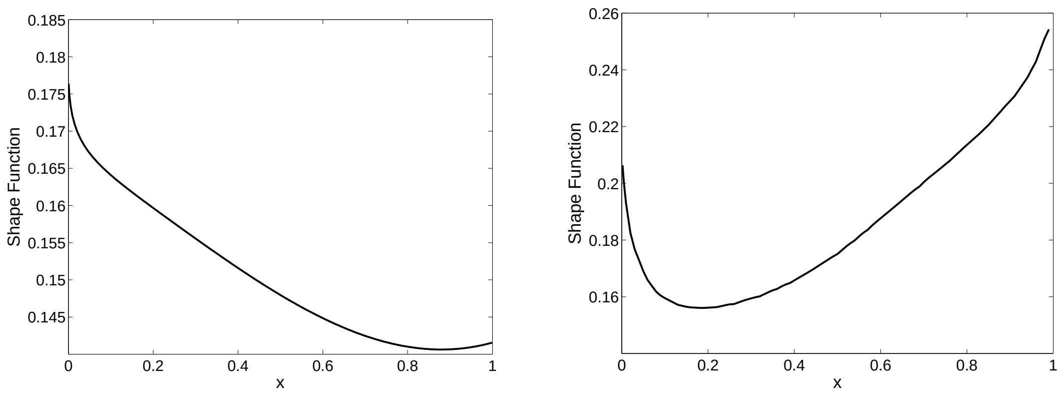
5.1 Kulfan Transformation Performance in Describing Supercritical Airfoils

The Kulfan transformation attempts to approximate 2D and 3D arbitrary shapes by defining the geometry within a specific class. The general class function as defined by Kulfan [2008] is

$$C(x) = x^{\mathrm{N1}}(1-x)^{\mathrm{N2}} \qquad , \qquad x \in [0,1]. \tag{5.1}$$

For the case study in Chapter 4 of a supercritical airfoil we used the exponents N1 and N2 associated with Kulfan's airfoil class, that is 0.5 and 1 respectively as we discussed in Section 3.3.5. This yields a round leading edge and a sharp trailing edge. The class function is chosen to give the general shape of an airfoil, with the unique characteristics of the airfoil being modelled using a parametric representation called the shape function.

Figure 5.1 shows the respective shape functions for the upper surfaces of the subsonic NACA0012 airfoil and the transonic SC(2)-0714 airfoil when using the class function in Equation (3.27). Figure 5.1(a) shows the complex curvature of the subsonic airfoil has been significantly simplified by the class function. However, Figure 5.1(b) shows no real reduction in curve complexity when the class function is applied to the transonic airfoil, suggesting we will have difficulty in finding an accurate approximation to the shape function. This is corroborated by the poorer error residuals of the SC(2)-0714 airfoil to the NACA0012 airfoil found by Kulfan and Bussoletti [2006]. We may, therefore, require



(a) NACA0012 shape function

(b) SC(2)-0714 shape function

FIGURE 5.1: Graphical representation of the shape functions of the upper surfaces for a subsonic and transonic airfoil.

a different class function from that of Equation (3.27) to define NASA's Supercritical airfoils to the same degree of accuracy as the subsonic airfoils. A method of finding such a class function is presented in the following sections.

5.2 Using Genetic Programming to Evolve a New Class Function

To produce a new class function we use a process that can modify the terms in an analytical equation as well as the coefficients that weight each term. For example, we have previously optimized the coefficients of a set of basis functions in Chapter 4. Here, we would also alter the basis functions themselves.

We can evolve a set of class functions by using the genetic programming method discussed in Section 2.4.4. Remember that this method does not only modify a set of numbers to obtain an optimal solution, but also the terms within symbolic expressions. The modifications of the GP method described in this section are based on the observation that the fitness of a symbolic expression may be highly dependent upon its constants, where the constants are numerical terminals in the symbolic expressions (for example, the terminal *2* in Figure 2.5). If we assume otherwise we could dismiss an expression as being inappropriate for the problem when, to the contrary, it could be the best solution given the correct constant values.

Koza [1992] suggested the use of an *ephemeral random constant* to be included in the terminal set that randomly generated a constant in the initial generation of the symbolic expressions. New constants could then only be created using combinations of these constants with arithmetic operators and other constant values included in the list of terminals. Many variations of this method have been devised and are discussed by Dempsey et al. [2007]. They also discuss the use of *concatenation constant creation*, whereby a list of digits are randomly put together with an option of including a decimal point.

The constant creation methods discussed so far do not take any account of the fitness function. A more useful method for our study is to optimize the fitness function subject to the coefficients of each expression in the population using a local simplex search, so the ability of the structure of each expression to produce good results is assessed. This is a type of Baldwinian learning (see Section 2.6), whereby each individual has to learn from a given situation; that is, we use ephemeral random constants in the initial generation of the expressions and append the fitness solution of the expression that has been locally optimized[2]. Thus we judge an expression by its potential ability to find a good solution to our problem. A similar approach has been used by Topchy and Punch [2001], although they used a gradient descent method and only run the local optimizer for a few iterations. Here the local search is run to convergence.

Within our local search GP method we include a few further alterations from the standard GP search. We ensure that if any two expressions within a population have the same structure their coefficients are different. We therefore give the local search several starting points within the population to find the best solution. We also ensure that each term in the expressions includes a coefficient that can be used in the optimization.

5.3 Applying the Fitness Function to GP

Once we have created our initial population of random symbolic expressions we must assess their suitability to being class functions for the Kulfan transformation. To do this we can compare their ability, when incorporated into the Kulfan transformation, to approximate existing airfoil designs. For example, if we choose an airfoil with coordinates $(\mathbf{x}_t, \mathbf{z}_t)$ as our target airfoil to be approximated, and we replace the general class function (Equation 3.27) with a symbolic expression generated by the GP algorithm, we can then use the Kulfan transformation described in Section 3.3.5 to approximate the target airfoil, finding the approximated z coordinates $\mathbf{z}_a$ at the same x positions as the $\mathbf{z}_t$ coordinates. Now, if we define the difference of the z coordinates as the error of our approximation, we can state our fitness function as being the sum of the squares of the error:

$$F = \frac{\sum_{i=1}^{n}(z_a(i) - z_t(i))^2}{n} \tag{5.2}$$

where n is the total number of coordinate points. Our aim is to find a class function that returns the lowest fitness value F possible, i.e. the class function that gives the lowest error. We can help the GP in finding valid solutions by noticing that any class function will have to go through the points (0,0) and (1,0). We therefore multiply each symbolic

[2]We do not insert the optimized constants back into the expression as each expression learns different optimal constants.

expression candidate by $x(1-x)$ to ensure that the proposed class function does go through these points. We also note from experience that the Kulfan transformation can find it difficult to accurately approximate regions around the leading and trailing edges of airfoils. We counter this problem by giving extra weighting of the accuracy of the approximation at the ten points nearest to the leading and trailing edges when computing the fitness function:

$$F = \frac{\sum_{i=1}^{n}(z_a(i) - z_t(i))^2}{n} + \frac{\sum_{i=1}^{10}(z_a(i) - z_t(i))^2}{10} + \frac{\sum_{i=n-9}^{n}(z_a(i) - z_t(i))^2}{10} \tag{5.3}$$

5.4 Applying the GP Method to Find a New Class Function

The Kulfan transformation can approximate the upper or lower surface of subsonic airfoils to within machine tolerances using a shape function polynomial of order 4 [Kulfan and Bussoletti, 2006]. We now attempt to define a NASA SC(2) supercritical airfoil using the same shape function characteristics by creating a new class function. We conduct two GP searches to find the most appropriate class function for the upper surface of the SC(2)-0612 airfoil: a canonical GP search, and a GP using a local search to assess each equation's potential ability to find an accurate solution. We include the general class function of Equation (3.27) in the initial population to give the GP the opportunity to build upon it. We use different starting lengths of the symbolic expressions for the initial population and run the GP algorithms ten times at each starting length. We restrict the maximum length of the expressions to 15 characters, not including the size of the constants. The results shown in Table 5.1 are for a population size of 500 for the traditional GP and 100 for the GP combined with a local search.

The GP with a local search of the constants outperforms the canonical GP as, even though the canonical GP code has a much larger population size to work with, it produces a best fitness value that is ten times greater than the worst achieved using a GP search combined with a local search. From studying the results in Table 5.1(b) we see that we produce similar expressions when using starting lengths of 4 and 6. We can choose either solution for our class function as, although the best solution is found using a starting expression length of 8, the expression is considered too complex for such a small gain in accuracy. Reducing the size of the constants without a large change in fitness, our class function for the upper surface of the SC(2)-0612 airfoil becomes:

$$x(1-x)[3.1x^{\cos(1.9x-8.396)} - 0.68] \tag{5.4}$$

The lower surface of the SC(2) airfoils have a shape reminiscent of a sine wave aft of the midpoint. In Section 4.1.1, we explain that this exists to improve circulation

(a) GP (Population: 500)

Eqn length	fitness ($\times 10^{-7}$)	$x(1-x)*$Expression
4	9.286	$\sqrt{5/x}$
6	7.896	$2-\sqrt{7/x}$
8	0.3757	$4/((\sqrt{x}+x)+x)$
10	0.3685	$(\sqrt{(7)}/x)^{\sqrt{7}}$
12	0.326	$\sqrt{\frac{6}{(x^x)^x}}$

(b) GP with local search (Population: 100)

Eqn length	Fitness ($\times 10^{-7}$)	$x(1-x)*$Expression
3	0.3568	$1.7244x^{-2.6412}$
4	0.0335	$1.1266x^{\sin(1.9061x-0.54194)} - 0.24814$
6	0.0335	$3.095x^{\cos(1.9061x-8.3959)} - 0.68194$
8	0.0295	$0.47273x + 1.3498x^{4.9726x+1.2588\cos(8.2779+2.7833x)}$

Table 5.1: Results for the upper surface of the SC(2)-0612 airfoil using the canonical GP search and the GP search with local optimization of the constants after 20 generations.

around the trailing edge so that the lift lost due to the flattening of the upper surface can be recovered [Harris, 1990]. If we use Equation 5.4 as our class function to find the Kulfan transformation of the lower surface, we do not make any improvement from using the general class function, implying another class function needs to be developed for the lower surface. If we take the same approach to find the class function for the lower surface we find the GP search produces large and complicated expressions with fitness values that are twice as large as those for the upper surface. If we compare the expressions in Table 5.2(a) to those in Table 5.1(b) we see that $x^{\text{trigonometric term}}$ is common in the best results, and so we may be able to use the newly found class function for the upper surface in the solution to finding the class function for the lower surface.

A method we can use to incorporate the class function of the upper surface to define the lower surface class function is gradient boosting. Gradient boosting in GP involves fitting a model to the data, then fitting another model to the residuals of the first model, and so on, with the final model made up of the sum of these partial models [Sóbester et al., 2008]. Table 5.2(b) shows that the boosting strategy halves the size of the fitness values whilst building more concise and intuitive equations. The final class function for the lower surface of the SC(2)-0612 airfoil is:

$$x(1-x)[3.1x^{\cos(1.9x-8.396)} - 0.014\sin(16x+8.5) + 0.76x - 0.7025] \tag{5.5}$$

(a) GP with local search (Population: 100)

Eqn length	Fitness ($\times 10^{-7}$)	$x(1-x)*$Expression
3	0.1613	$\frac{1.7599}{x^{1.7703-0.1685/(-2.0094x-0.33052)}}$
4	0.0557	$1.6205x^{2.9373x+0.5083\cos(9.7451-0.18347\cos(7.969x))}$
6	0.0693	$0.40489x.^{1.8279x-0.668\cos(\cos(10.3031+5.06x-3.45x^2))}$
8	0.0733	$0.41886x-(x+0.007816x^2).^{\sin(x^2-2.6272)}+1.2397\sin(1.98+1.356x)$

(b) GP with local search and boosting (Population: 100)

Eqn length	Fitness ($\times 10^{-7}$)	$x(1-x)*(3.1x^{\cos(1.9x-8.396)}-0.68+$Expression$)$
3	0.0345	$0.18033\sin(\sin(-108.8157/(6.2567x-14.6484)-6.3164)+2.2462)$
4	0.0355	$0.77243x+0.014784\cos(15.805x-15.1137)$
6	0.0325	$-0.014213\sin(8.5023+16.037x)-0.0245+0.76156x$
8	0.0343	$0.2009\sin(x(\sin(11.4195x^2+12.717x)+1.3724x))$

TABLE 5.2: Results for the lower surface of the SC(2)-0612 airfoil using the GP with local search and with the addition of a boosting procedure at different starting expression lengths after 10 generations.

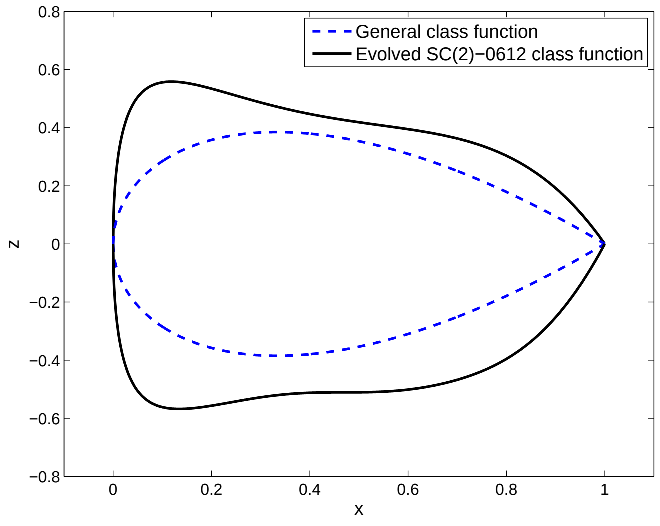
The graphical representation of the class functions for the upper and lower surfaces of the SC(2)-0612 airfoil are shown in Fig. 5.2(a). The shapes of the evolved class functions are significantly different from the general class function, but the accuracy depicted by the residuals in Fig. 5.2(b) proves their greater capability in defining the SC(2)-0612 airfoil.

5.5 Finding a Universal Class Function for the Supercritical Airfoils

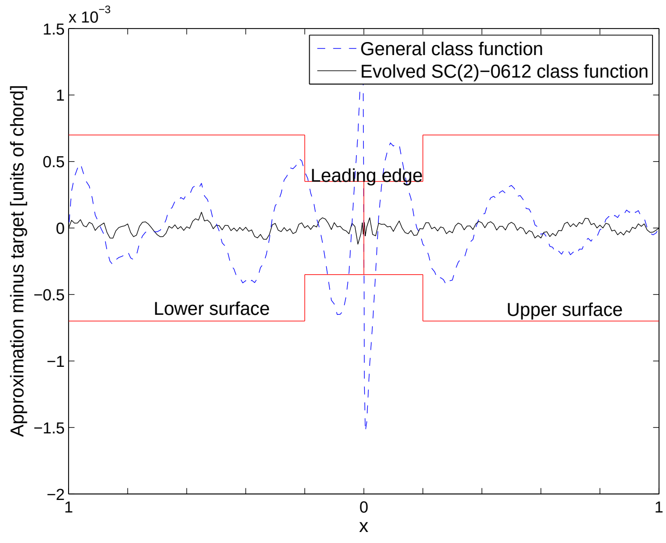
We have shown that the GP/Simplex combination works successfully in finding a class function that can be used in the Kulfan transformation to accurately approximate the SC(2)-0612 airfoil using only 4 parameters. To find an expression for the class function that can be used to describe all civil transport members of the SC(2) airfoils we must define a fitness function based on several of the SC(2) airfoils. We have chosen 6 out of the 21 members of NASA's supercritical family to be included in the fitness function: SC(2)-0410, SC(2)-0610, SC(2)-0710, SC(2)-0412, SC(2)-0612, SC(2)-0712. These are a sample of the SC(2) airfoils designed for civil transports and business jets. To assess the quality of each individual in our population of candidate class functions we could use the fitness function of Equation (5.3) and apply it to each airfoil in our set, then divide by the number of airfoils tested. This may give the best average value but there could exist an unacceptable fitness result for one of the airfoils that is not expressed in the average, as the fitness values of the other airfoils may cancel out the influence of the poor result.

It is apparent that we must reduce the error over all airfoils, and not only the average of all the objectives. A solution is to give the worst approximated airfoil an extra weighting in the average of the airfoil fitness values. For example we can increase the influence of an outlier by adding an extra instance that has the same value as that outlier. In our case we add an extra 'imaginary' airfoil that has the same fitness value as the airfoil approximation with the worst fitness. Now the influence of the worst approximated airfoil dramatically increases the fitness value the further it is from the real average. We have also added a linear penalty function in an attempt to maintain similar error residual across the entire chord. This penalty function is effective for error residuals between the wind tunnel tolerance at the leading edge (3×10^{-4}) and (8×10^{-4}), which is slightly larger than the wind tunnel tolerance for the rest of the airfoil approximation. If the worst error residual across the airfoil chord is within this range then the penalty function is activated. If the largest error residual is larger than 8×10^{-4} then the penalty function is its value at 8×10^{-4}, which is 0.4[3]. Our penalty function is mathematically described as

[3] This value was used as it is approximately double the fitness value corresponding to being within the wind tunnel tolerances at the leading edge.



(a) The shape of the general class function and the evolved class functions.



(b) Error Residuals between the target airfoil and the Kulfan transformation approximations using different class functions. The horizontal lines show the typical wind tunnel tolerances, with tighter tolerance within 20% of the leading edge.

FIGURE 5.2: Graphical comparisons between the general class function and the evolved class function for the SC(2)-0612 airfoil.

$$P_i = \begin{cases} 800[\max(\mathbf{z}_i - \mathbf{z}_t) - 3 \times 10^{-4}] & \text{if} \quad 3 \times 10^{-4} < \max(\mathbf{z}_i - \mathbf{z}_t) < 8 \times 10^{-4} \\ 0.4 & \text{if} \quad \max(\mathbf{z}_i - \mathbf{z}_t) > 8 \times 10^{-4} \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{z_t}$ is the vector containing the target z coordinates and $\mathbf{z_i}$ is the vector containing the target z coordinates of the Kulfan transformation approximation.

Our new fitness function is now:

$$F = \frac{\sum_i^m \mathbf{f} + \max(\mathbf{f})}{m+1} + \sum_i^m P_i \tag{5.6}$$

where $\mathbf{f}$ is the vector of the individual fitness values for each airfoil i calculated by Equation (5.3), m is the number of real airfoils chosen for our search

Using this method maintains the order of the individual fitness values so that we can still compare them with the results in Section 5.4.

The aim of our investigation is to find simple and intuitive symbolic expressions to be used as class functions in the Kulfan transformation technique. We define simple expressions as being under a certain length and having a limited number of repeated single branch operators within a term (for instance sin(cos(exp(...)))). We can force the GP to only find expressions with a single branch operator in each term, and limit the maximum expression length as we have done in the previous experiments.

We follow the same approach as when finding the class function for the SC(2)-0612 airfoil; that is, we find the class function for the upper surface, then use the upper surface class function in the boosting strategy to find the class function for the lower surface.

Table 5.3(a) shows the fitness values being approximately 10 times those for a single airfoil shown in Table 5.1(b). This was expected, with the GP having to find a general solution for all airfoils. It is interesting to note that the solution starting with an expression length of 3 is similar to that starting with a length of 8, and that neither includes a trigonometric operator like the solution for the SC(2)-0612 airfoil in Equation (5.4) did. The solutions starting with expression lengths of 4 and 6 do include the trigonometric term, but are more complex than the other solutions with no overall improvement in fitness. Due to these comments we have chosen the class function for the upper surface of SC(2) airfoils to be:

$$x(1-x)[1.265x - 1.4x^{1.28x-0.533}]. \tag{5.7}$$

Using this equation in the boosting strategy for the lower surface produces the results in Table 5.3(b). The best result, from starting with an expression length of 8, is considered too large to be used as a class function. However, using a starting length of 6 introduces 2 extra terms to the upper class function as the x term in the solution is combined with the x term for the upper surface. Our class function for the lower surface of the SC(2) airfoils is:

$$x(1-x)(-1.4x^{1.28x-0.533} - 11.335x + 0.0399\sin(18.8x+1.3) - 1.2) \tag{5.8}$$

(a) Upper surface

Eqn	Fitness for each airfoil ($\times 10^{-7}$)						
len	0410	0610	0710	0412	0612	0712	Equation
3	0.1972	0.0847	0.2213	0.1128	0.1333	0.0598	$x(1.265 - 1.4x.^{1.28x-1.53})$
4	0.2247	0.1232	0.3223	0.0858	0.1535	0.0672	$0.80607x - 0.8793x.^{\cos(16.716+1.4194x)}$
6	0.2200	0.2167	0.4350	0.0783	0.1067	0.1250	$1.1588x^{-0.54057+18.1719x(x\sin(0.38593x-6.7395))}$
8	0.2540	0.0973	0.2482	0.1185	0.1502	0.1007	$1.477x^{1.27x-0.533} - 1.328x$

(b) Lower surface using boosting

Eqn	Fitness for each airfoil ($\times 10^{-7}$)						
len	0410	0610	0710	0412	0612	0712	Equation
3	0.4553	0.2300	0.4122	0.4473	0.1827	0.1475	$0.42835\sin(3.2813x - 4.5017)$
4	0.4008	0.1732	0.2535	0.4842	0.1475	0.1323	$6.8446\sin(-1.9696x - 0.16424)$
6	0.2750	0.2562	0.3957	0.2698	0.1658	0.1827	$0.039977\sin(18.8572x + 1.2872) - 12.5841x - 1.2257$
8	0.1623	0.1135	0.1835	0.1800	0.1302	0.1178	$-x(0.60966x - 4.33x\sin(5.664x - 5.3886) - 0.7307x^2 + 7.32517)$

TABLE 5.3: The GP results using a population of 200 and 10 generations for the upper and lower surfaces of the SC(2)-0410, SC(2)-0610, SC(2)-0710, SC(2)-0412, SC(2)-0612, SC(2)-0712 airfoils.

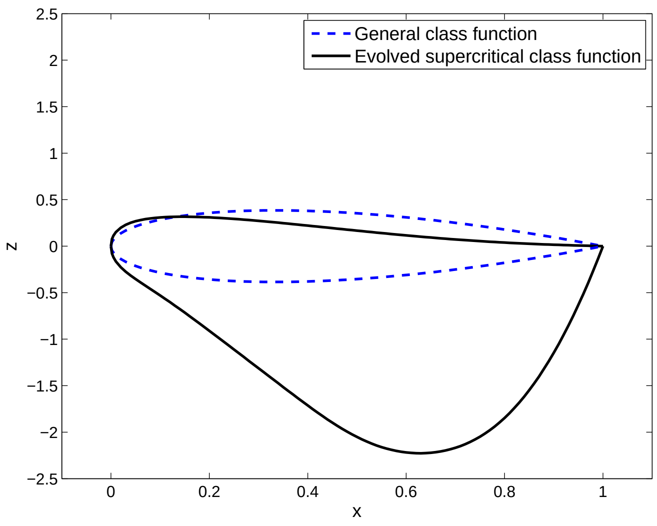
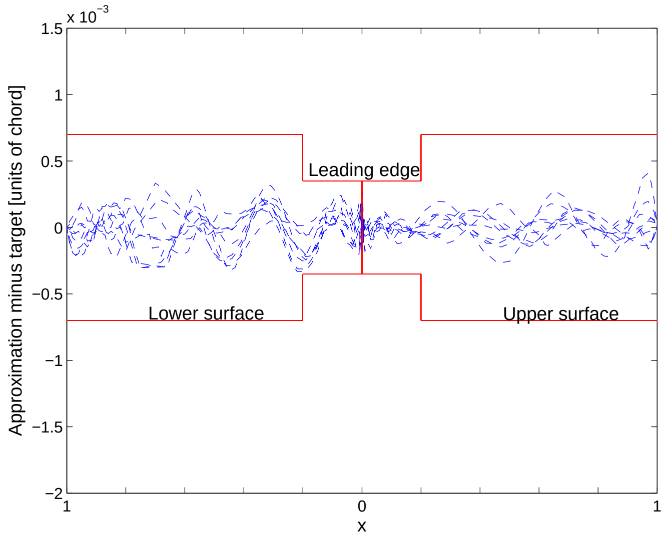


FIGURE 5.3: Graphical comparison between the general class function and the evolved class functions for supercritical airfoils designed for civil transports and business jets.

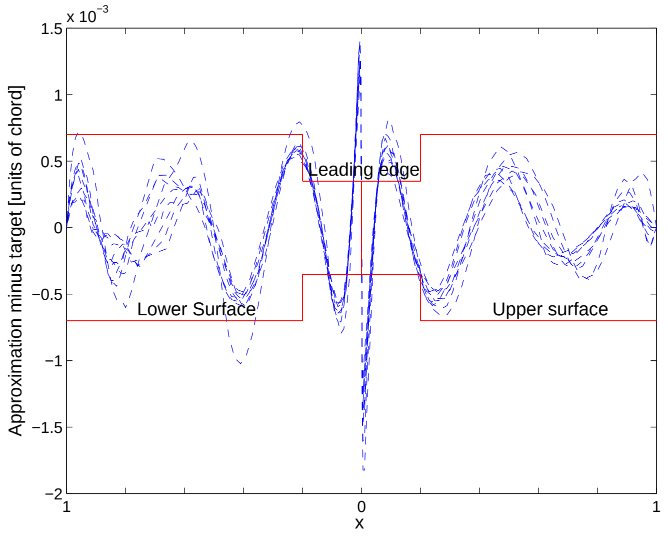
The graphical representation of the class function in Figure 5.3 suggests an unintuitive shape for the class function. However, we note that from Figure 5.4(b) the general class function finds it difficult to describe the leading edge, and so the evolved class function shows significant modification in this region. Also we note that the general class function in the Kulfan transformation finds it difficult to cope with the sinusoidal cusp associated with the lower surface of the supercritical airfoils. The evolved lower surface class function attempts to negate the influence of the cusp by modelling it using a sinusoidal function. The residuals in Figure 5.4(b) prove that the evolved class functions give a better approximation to the supercritical airfoils that were designed for civil transports and business jets. The lower surface is not as well approximated as the upper, mainly due to the cusp being characteristically different for each airfoil.

5.6 2D Airfoil Optimization

The advantages of modifying the class function can be assessed by applying it to a design optimization problem. We have set up a local optimization to modify the shape of the SC(2)-0610 airfoil to reduce the drag at constant lift. From studying the design requirements from Harris [1990] we find that, when using the potential flow solver VGK [ESDU96028, 2004], the SC(2)-0610 airfoil produces the design pressure distribution at $M = 0.7728$, Re $= 30 \times 10^6$, and angle of attack$= -0.43$. This gives a c_d of 0.0077 and a design c_l of 0.63, which we keep constant for every instance in the optimization process by altering the angle of attack, α. We use the Nelder and Mead [1965] Simplex search to reduce the drag at these design conditions, with spar thickness constraints of $0.095c$ and $0.0739c$ at 25% and 65% chord respectively, where c is the length of the



(a) Evolved supercritical class functions.



(b) General class function.

FIGURE 5.4: Error residuals using the evolved supercritical class functions and the general class function for the SC(2)-0410, SC(2)-0610, SC(2)-0710, SC(2)-0412, SC(2)-0612, SC(2)-0712, SC(2)-0414, SC(2)-0614, SC(2)-0714 airfoils.

chord[4]. We also constrain the trailing edge to be finite. We use the coefficients of the Bernstein polynomials as our variables to be optimized, and obtain flow characteristics using VGK. The results using the general class function and the evolved class functions with different orders of Bernstein polynomials are shown in Figure 5.5(a).

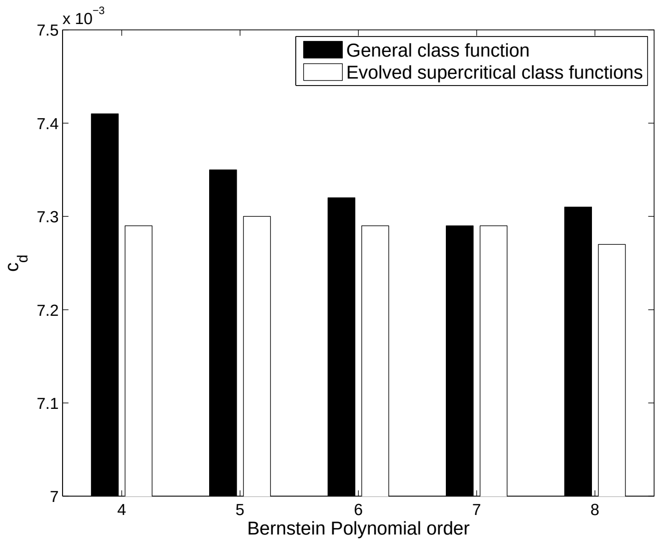
We find that we improve upon the drag of the SC(2)-0610 airfoil for every class function and polynomial order, even though we know that when using the general class function the SC(2)-0610 airfoil is not very well approximated using low Bernstein polynomial orders. When using the evolved class functions there is no dependence on Bernstein polynomial order, and performance at minimising drag is better than when using the general

[4]These values were taken from the SC(2)-0610 geometry.

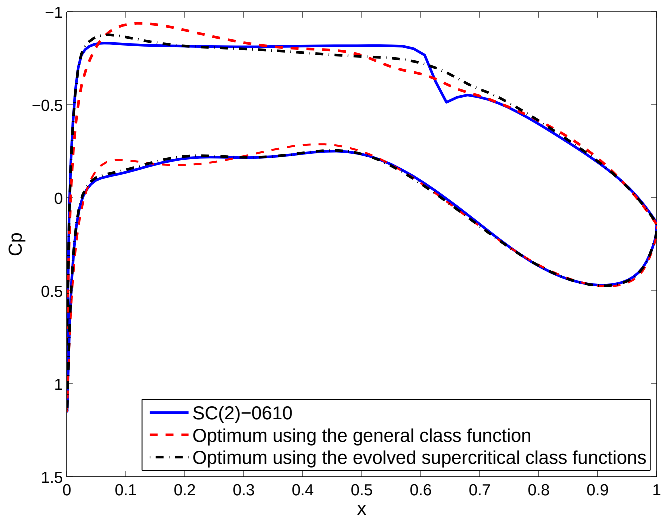
class function in the Kulfan transformation. When using the general class function, the Kulfan transformation attains improved performance with increasing polynomial order up to a polynomial order of 7. Figure 5.5(b) shows the inability of the Kulfan transformation to accurately approximate the leading edge curvature when using the general class function, as the pressure coefficient (c_p) is significantly different at the leading edge for the optimised SC(2)-0610 airfoil. This displacement is less significant when using the evolved class functions. Figure 5.5(b) does indicate that when using either the general or the evolved class functions the reduction in drag is found by losing the shock on the upper surface. The c_p of the lower surface when using the general class function is much more perturbed from the original compared to using the evolved class functions, which is again likely to be due to the poor approximation of the SC(2)-0610 airfoil as a starting point.

5.7 Summary

Using a GP algorithm along with a local search of the constants in the symbolic expressions, we have found an alternative class function for the Kulfan transformation technique to be used in approximating the SC(2) class of airfoils, specifically those designed for civil transports and business jets. The evolved class functions model the unique characteristics of the SC(2) airfoils better than the general class function, thereby reducing the complexity of the resulting shape function to be modelled using Bernstein polynomials. The use of a specific class function can be advantageous in reducing the dimensionality of an optimization search, as shown by the 2D optimization presented, but care must be taken as over specific class functions can lead to restricting the design space making it impossible to find the optimal solution. The ability to find new class functions is not limited to airfoil geometries; the same method can be applied to find other class functions where perhaps the Kulfan class function cannot produce an accurate approximation within a required design space size, or in a three-dimensional application. There is also potential to use this method to find other baseline shapes to be used by other geometry representation techniques. The experiments presented here could be repeated with the leading edge shaping term used by Sóbester [2009] included in both the original and GP based formulation. It is expected that such an investigation would yield a different class function.



(a) Drag improvement with shape function polynomial order for the SC(2)-0610 airfoil using the general class function and the evolved supercritical class functions in the Kulfan transformation.



(b) Pressure coefficient plots of the SC(2)-0610 airfoil and the optimized airfoils using the general class function and evolved supercritical class functions in the Kulfan transformation with fourth order Bernstein polynomials.

FIGURE 5.5: Results of the Simplex optimization for the SC(2)-0610 airfoil using the general class function and the evolved supercritical class functions in the Kulfan transformation.

Chapter 6

The Over-Wing Engine Configuration: Using Surrogates to Understand Noise Shielding Performance

The optimization procedure used in Chapter 4, and that to assess the quality of the application-specific class functions in Chapter 5, involved a local search of the parametric variables. The choice of a local search was motivated by the time associated in finding an optimum solution, even though we expected the optimization landscape to be multimodal in both cases. Now consider a problem where there are only two design variables: the curse of dimensionality is not as paramount as it was previously, and so the computational expense to search the design space is substantially reduced, allowing us to consider a global optimization strategy.

Also consider the situation where a numerical performance evaluation procedure is either not available, or is available but does not provide a high degree of accuracy. In such a case a physical experiment has to be devised where specialist equipment may be highly expensive and only be available for a limited period. Moreover, the setup of the experimental procedure may inhibit the ability to perform, say, an accurate set of experiments on newly generated designs from an optimization process. It would therefore be impossible to use a standard global procedure like a Genetic Algorithm. However, if the initial set of experiments densely fills all regions of the design space equally, we may be able to use a surrogate modelling process based on an experimental plan derived to ensure uniform sampling of the design space.

In this chapter we present a solution to a problem similar to the scenario above, by using surrogate modelling, without updates, on a set of designs selected in such a way that they suitably fill the design space. Before presenting this method we discuss the problem at hand: that of designing aircraft to reduce their noise impact on the communities that surround airports. We suggest a possible solution whereby the engines of civil transports are installed in a position above their respective wings where the airframe can deflect the noise generated by the engines in such a way that it is not incident to the ground. We set up an optimization procedure to explore the potential of this aircraft configuration in achieving this objective.

6.1 Tackling Aircraft Noise

Since the advent of the civil jet airliner, aircraft noise has been a significant issue, especially for those living in the close proximity of airports. Although measures such as land planning restrictions have been applied to reduce the impact of noise, the rise in air traffic, as well as public awareness, has increased the importance of designing aircraft that reduce the noise experienced by on-the-ground observers. Figure 6.1 shows that the perceived noise level (PNL) has been declining with time, but the improvements made due to the high bypass ratios of the second generation turbofans seem to have reached their limit [Crichton et al., 2007]. The ACARE2020 [2001] noise target aimed at reducing the average perceived noise by one half is looking increasingly unlikely to be achieved using current aircraft configurations and engine technology.

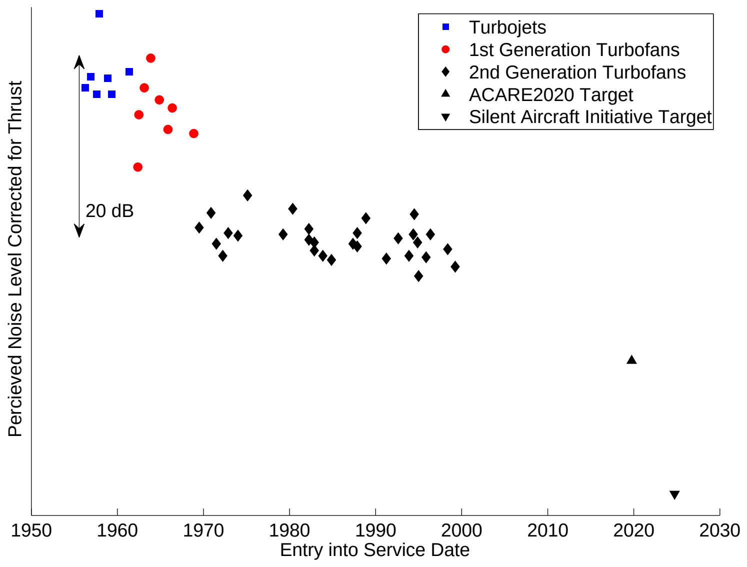


FIGURE 6.1: Reduction of perceived noise level corrected for thrust over time (graph reproduced from Crichton et al. [2007]).

New aircraft concepts designed with the potential to drastically reduce aircraft environmental noise, amongst other environmental design factors, have yet to become reality.

The radical difference in layout from conventional aircraft, apparent with many of these designs, may lead to anxiety amongst the travelling public. Moreover, the large development costs incorporated with such unconventional designs, combined with the risk of failure, mean airframe manufacturers are unlikely to pursue such ambitious projects. To alleviate these pressures, NASA have begun a research effort with the goal of developing three designs (denoted *N+1, N+2, N+3*), each to be operational at different periods from between 2012 and 2035, that gradually evolve away from the current aircraft configurations (denoted as *N*) we see today. The *N+1* design is to be similar to current 'tube and wing' aircraft, using new technologies that are at a high technology readiness level, leading up to a drastic change for the *N+3* aircraft. An example of the latter is the Blended Wing Body (BWB), which promises to greatly improve aerodynamic efficiency as well as noise reduction compared to current configurations [Hileman et al.]. However, due to the aircraft being constrained to remain conventional, the *N+1* configuration concentrates on the powerplant and its placement with respect to the wing [Berton et al., 2009], in order to alleviate the aforementioned problems of public acceptability and development costs.

Increased efficiency and reduced noise can be achieved through increased powerplant diameter, allowing for higher by-pass ratios, and the use of geared fan drives. However, on conventional under-the-wing installations the size of the engine is constrained by ground and taxiway light clearance requirements. Issues such as landing gear failure and foreign object ingestion from runways have to also be taken into account.

6.1.1 The Over-Wing Configuration

All of the issues listed above are eliminated by installing the engines above the wing, allowing for higher turbofan bypass ratios without affecting the ground clearance of the aircraft, its roll clearance and the location of passenger escape zones [Berry, 1994]. Above the wing, engine size will only be limited by structural and attachment fitting limitations [Berry, 1994]. Kinney and Hahn [1997] noted that the absence of engine exhaust below the wing would eliminate the need for thrust gates between the flaps, allowing for a single flap that would reduce noise created by the tip vortices and give larger effective flap area. There may also be an exhaust interaction with the wing upper surface, delaying separation due to the Coanda affect. Further, with the height of the aircraft from the ground independent of the engines, the size of the landing gear only depends on tail scrape angle and the magnitude of rotation at take-off. Other advantages that Kinney and Hahn [1997] identified with over-wing engines are the increased flexibility of landing gear placement on the wing, greater safety in water ditching and wheels-up landing scenarios, and, of most interest to us here, the benefit of reduced noise due to shielding of the wing.

The disadvantages of an over-wing configuration include maintenance issues (the engine not being accessible from the ground) and a possible increase in cabin noise[Berton, 2000, Ronzheimer et al., 1996]. A wholesale re-design of the wing structure may be necessary, incurring significant development costs (though, of course, not as significant as those of the blended wing-body configuration). From an aerodynamic standpoint, altering the upper surface of the wing may incur a decrease in the lift to drag ratio, and may also cause issues with trim drag and stability, depending on the position of the engine on the wing.

6.1.2 Acoustic Benefits

Installing an engine above the wing gives us the opportunity to use the airframe to shield engine noise from observers on the ground. NASA began investigating the advantages of over-wing mounted engines in the early 1970s, in order to reduce aircraft noise affecting communities near airports[Reshotko and Olsen, 1972]. This work was published after the first prototype of the Fokker VFW 614 aircraft took to the air on its maiden flight in 1971 [Jordan, 2010], the only civil jet airliner to have over-wing mounted engines. The main considerations that led to this design were: maximizing payload flexibility, minimizing foreign object matter ingestion, and satisfying fuselage ground clearance requirements. Noise shielding (leading to a 4 EPNdB noise reduction at approach) and undivided landing flaps were not amongst the main considerations for this design, rather they were serendipitous effects of the over-wing configuration. Noise shielding was not a primary consideration three decades later for the Honda Business Jet either; the engines were located above the wing, as the alternative was to install them on the rear fuselage, which would have reduced the cabin volume due to the structural supports[Fujino and Kawamura, 2003].

Studies by Agarwal and Dowling [2007] and Agarwal et al. [2007] have highlighted the noise reducing potential of using parts of the airframe as a shield, albeit applied to a blended wing body aircraft design. Berton [2000] analysed the potential of over-wing engines on a large, long haul, four engine airliner. He reported that aircraft noise experienced by local communities can be reduced by up to 34.5 cumulative EPNdB using the wing planform as a shield, and that the 95EPNdB noise footprint can be reduced from 0.96 to 0.57 square miles. Core and fan discharge noise were the noise components that saw significant reductions, with no effect on fan inlet noise as the engine was forward of the leading edge of the wing.

NASA's Leavitt and Smith [2010] also agree that an over-wing engine configuration looks promising, and is an area that they intend to explore further. However, NASA is currently concentrating on investigating the aft radiated noise of a hybrid wing body (HWB) configuration [Thomas et al., 2010], and on the noise generated by new engine concepts [Van Zante and Breeze-Stringfellow, 2010]. Another study on installation affects is an on-going project considering open rotor noise on both a HWB and a conventional configuration (see Thomas et al. [2010]).

In terms of the streamwise positioning of over-wing engines on jet aircraft a clear distinction can be made between nacelles positioned forward or aft of the leading edge. The Honda business jet, the VFW 614, and the Be-200 have their engines located on the aft portion of the wing, whereas the aforementioned noise shielding studies of Reshotko and Olsen [1972] and Berton [2000] had the engine located forward of the leading edge in an attempt to shield ground-based observers from the jet and core noise. In the aft position there is the potential to significantly reduce the fan inlet noise that propagates to the ground, a contribution to the total engine noise of which Berton's forward mounted over-wing engine was not able to reduce [Berton, 2000].

In the remainder of this chapter we investigate the broadband noise, generated by interaction of the blade tip with the casing boundary layer, rotor-wake-turbulence [Groenweg et al., 1995], and rotor-self noise. An experiment was performed whereby a broadband noise source was used to simulate the fan inlet noise, suspended above a scaled aircraft model. The degree of noise shielding was then measured using an array of microphones positioned below the aircraft model. The experimental setup allowed for the vertical movement of the airframe and the chordwise movement of the 'engine', permitting numerous combinations of chordwise and vertical nacelle positions to be tested with respect to the wing. 40 tests were performed, corresponding to different nacelle positions. Statistical analysis tools were used to find the optimal position of the nacelle in terms of noise shielding performance. The results were also compared to half-plane diffraction theory in order to better understand the measured behaviour.

6.2 Experimental Setup

The main objective of this study is to investigate the variation of the spatially averaged noise diffracted below a wing due to variations in position of the nacelle inlet above it. In this investigation the relative location of the nacelle was varied in both the chordwise x direction from the wing leading edge and in the vertical z direction above the wing (see Figure 6.2). The nacelle, represented by a long duct, was kept fixed in the spanwise y direction. One end was fitted with an inlet model and was directed into the anechoic chamber (see Figure 6.3). The other end was situated within the adjacent reverberation chamber, which was excited by a loudspeaker driven by incoherent white noise signals (see Figure 6.4). The noise propagating along the duct and radiating from the inlet was therefore broadband and multimodal with well-defined characteristics, as discussed below.

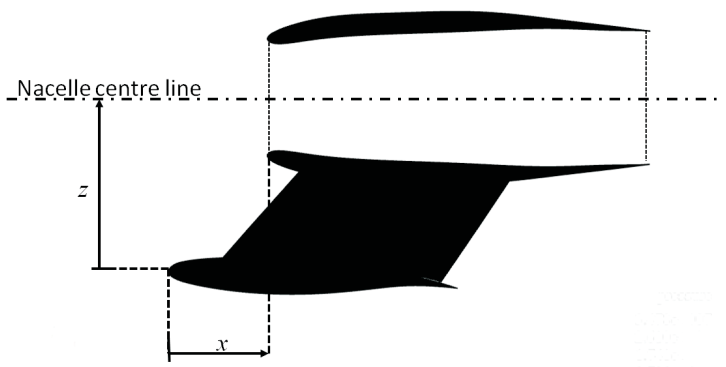


FIGURE 6.2: Diagram of the wing/pylon/nacelle cross sections at their intersections, x and z being the design variables of the study described.

6.2.1 Description of the Aircraft Model

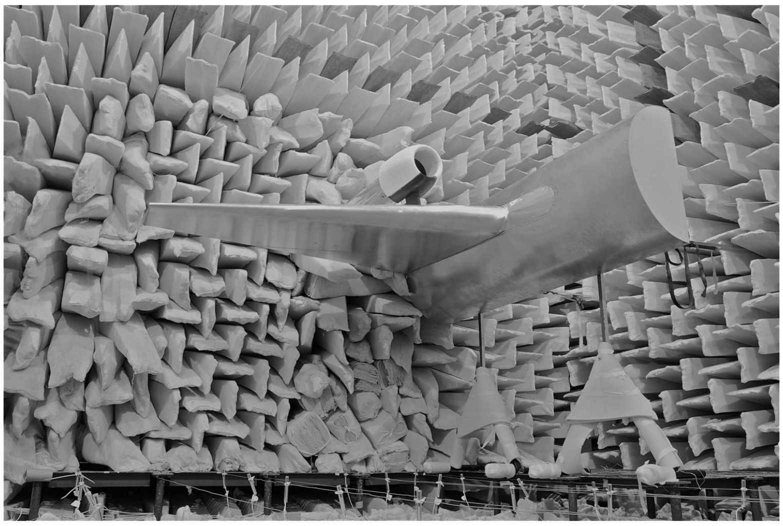


FIGURE 6.3: Setup of our DLR-F6 aircraft half model in the University of Southampton's ISVR Large Anechoic Chamber.

The starboard half of a 12.2% scale-model, representative of the DLR-F6 aircraft model used in the American Institute of Aeronautics and Astronautics (AIAA) drag prediction workshops [Laflin et al., 2005], was installed in the University of Southampton's Large Anechoic Chamber (see Figure 6.3)[1]. The fuselage structure was made of plywood. Model-quality foam was used for the wing with aluminium sheeting wrapped around it to ensure that the noise reflectivity at the surface was more representative of a passenger airliner. Since we were interested in the fan inlet noise radiation from the front portion of the wing, the trailing edge of the wing was modified so that it could be positioned flush to the back wall of the anechoic chamber to prevent trailing edge noise from reaching the microphones.

[1]The dimensions of the chamber are $7.33 \times 7.33 \times 5.5\text{m}^3$

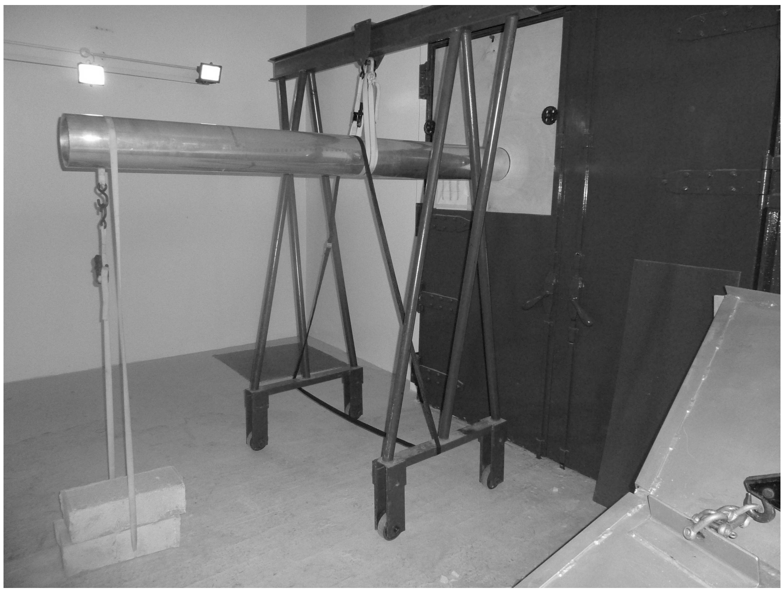


FIGURE 6.4: The reverberation chamber, with the nacelle duct leading into the University of Southampton's ISVR Large Anechoic Chamber.

The fuselage model extended 0.9m beyond the root of the leading edge and spanned the whole of the root chord. Due to manufacturing constraints, the leading edge sweep was slightly reduced from 27.1^o on the DLR-F6 wing to 25^o in the scale model. The nacelle was constructed from a 0.193m internal diameter aluminium pipe. At the end of the pipe, located in the anechoic chamber, was attached a nacelle inlet model constructed from ABS plastic (via a stereolithography process), with the leading edge portion of the pylon attached to the bottom surface made from the same material. The other side of the duct was located within the adjacent reverberation chamber, where a loudspeaker driven by a broadband white noise signal up to 20kHz (i.e., at 1.25kHz at full scale) provided the acoustical excitation. This upper frequency corresponds to a non-dimensional frequency ka of approximately 38 (i.e. between the 1st and 2nd blade passing frequencies for a modern turbofan engine), where k is the free space wave number, and a is the duct radius. Note that the resulting spectrum of the noise radiated from the engine duct, as shown in Figure 6.6, is a property of the radiation characteristics of the loud speaker and of the end of the pipe and is therefore not representative of the noise spectrum of the broadband noise due to the various turbulence sources listed in Section 6.1.2. Diffuse field excitation of the pipe within the reverberation chamber ensured mutually incoherent modes within the duct, whose amplitudes are such that the total acoustic energy is shared equally amongst the modes. The far field directivity due to this distribution of modes has been investigated by Joseph and Morfey [1999] and shown to provide a distribution of mean square pressure in the forward arc given by $\bar{p^2} \propto \cos\theta$, where θ is the polar angle measured from the duct axis. For values of θ greater than $\pi/2$, i.e., in the rear arc, the redirected pressure was found to decay substantially with increasing θ.

The noise diffracted around the wing leading edge to below the wing was measured by an array of 20 microphones distributed on the anechoic wedges on the floor. The microphone outputs were acquired simultaneously using a National Instruments PXI-1042 data acquisition system at a sampling frequency of 50kHz in each channel and fed into a computer for post-processing. The time histories were then converted to Power Spectral Densities (PSD), with each signal being corrected for variations in their distances from the centre of the duct, assuming the inverse square law for spherical spreading. Here, the window length was 0.02048s and the total measurement time was 28.67s. In order to quantify the effect of shielding by the wing the noise at the microphone array was measured with and without the wing present, denoted by s and c respectively ('shielded' and 'control', respectively). The differences in these measurements were then used to compute a shielding metric, in decibels

$$\Delta = 10\log_{10}\left[\frac{\sum_{\theta,\phi}\bar{p_c^2}(\theta,\phi)}{\sum_{\theta,\phi}\bar{p_s^2}(\theta,\phi)}\right], \qquad \text{(dB)} \tag{6.1}$$

where $\bar{p^2}(\theta,\phi)$ is the mean pressure and (ϕ,θ) is the angular position of the microphones (shown in Figure 6.7) with respect to the centre of the duct. The variation of Δ in 1/3 octave bands was investigated for the different nacelle positions relative to the wing.

6.2.2 Experiment Plan

The position of the front of the nacelle was varied between the leading edge of the wing at $x = 0$ to $x = 0.7c_y$, where c_y is the wing chord length measured at the pylon position. The vertical displacement of the nacelle z was varied within $1.5h$, where h was the diameter of the nacelle, measured from the leading edge point of the wing to the nacelle centre line (see Figure 6.2). In total, the noise from 40 nacelle positions was measured relative to the leading edge of the wing at the spanwise position of the engine.

In order to accurately find the optimal position of the engine for maximum noise shielding the 40 nacelle positions were carefully chosen within the range of x and z stated above. As the optimal position of the engine is unlikely to be one of the 40 tested, a *surrogate model* (see Section 2.4) was developed to relate the noise shielding Δ to any arbitrary vertical and chordwise engine position x and z. Of the variety of available surrogate modelling formulations, here we use Kriging (also known as Gaussian Process modelling), chiefly because it enables accurate tuning and control of the amount of regression required and experience shows it to be well suited to this class of objectives.

The accuracy of surrogate models is highly dependent upon the positions of the "training data", i.e. of the 40 tested nacelle positions. We stated in Section 2.5 that sampling plans are traditionally used: the primary aim of designing such training data layouts is to ensure that the surrogate model will be accurate at predicting the performance metric Δ at untested nacelle positions. Sampling plans that fall into the class of *Latin Hypercubes*

[McKay, 1992] have excellent stratification properties, but they only maximize uniformity (as measured by the Morris and Mitchell [1995] criterion) for square numbers of points. To satisfy these criteria in our experiment we have chosen an *Orthogonal Array Latin Hypercube* [Tang, 1993] sampling plan, comprising 36 points, with four additional points testing the vertices of the design space. Figure 6.5 shows the resulting pattern of nacelle position, which has the serendipitous advantage of facilitating the re-positioning of the rig between consecutive experiments, due to the appropriate ordering of its values along the z axis.

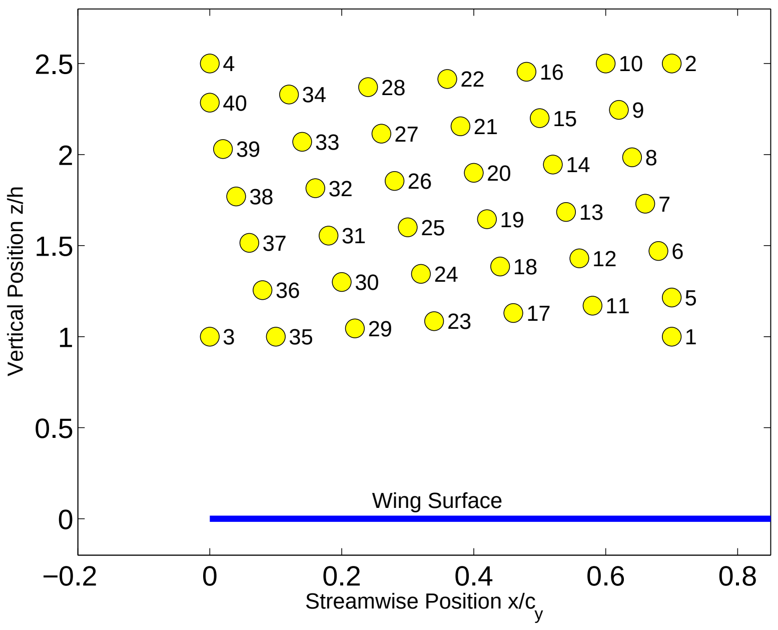
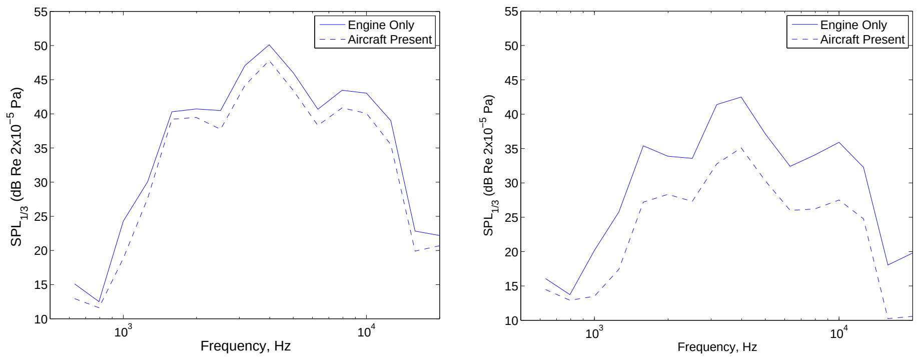


FIGURE 6.5: The tested nacelle positions in the investigations, with their associated experiment number.

6.3 Results

Prior to the experiments designed to measure the shielding properties of the wing a control experiment was performed where measurements were made of the noise emanating from the intake, in the absence of the wing and fuselage. Figure 6.6 shows typical one-third octave sound pressure levels (SPL), with the aircraft present and absent, at two microphones when the nacelle is at the position $x = 0.28x/c_y$, $z = 1.85z/h$ (Experiment 26 in Figure 6.5). The SPL in Figure 6.6(a) is that recorded by microphone 10, positioned directly in front of the nacelle with direct ray path between the duct and the microphones (i.e., in the 'light zone'). Microphone 7 was positioned directly beneath the wing, the SPL of which is shown in Figure 6.6(b), and is therefore in the shadow zone of the wing. Figure 6.6(b) shows that the sound attenuation of 6 dB is significantly higher in this region. In the illuminated zone, little, if any, benefit was present in the SPL below 500Hz, where the background noise dominates.



(a) SPL recorded from microphone 10, which was in the light zone.

(b) SPL recorded from microphone 7, which was in the shadow zone.

Figure 6.6: SPLs recorded from two different microphones for the nacelle position $x = 0.28$m, $z = 1.85z/h$ (Experiment 26 in Figure 6.5).

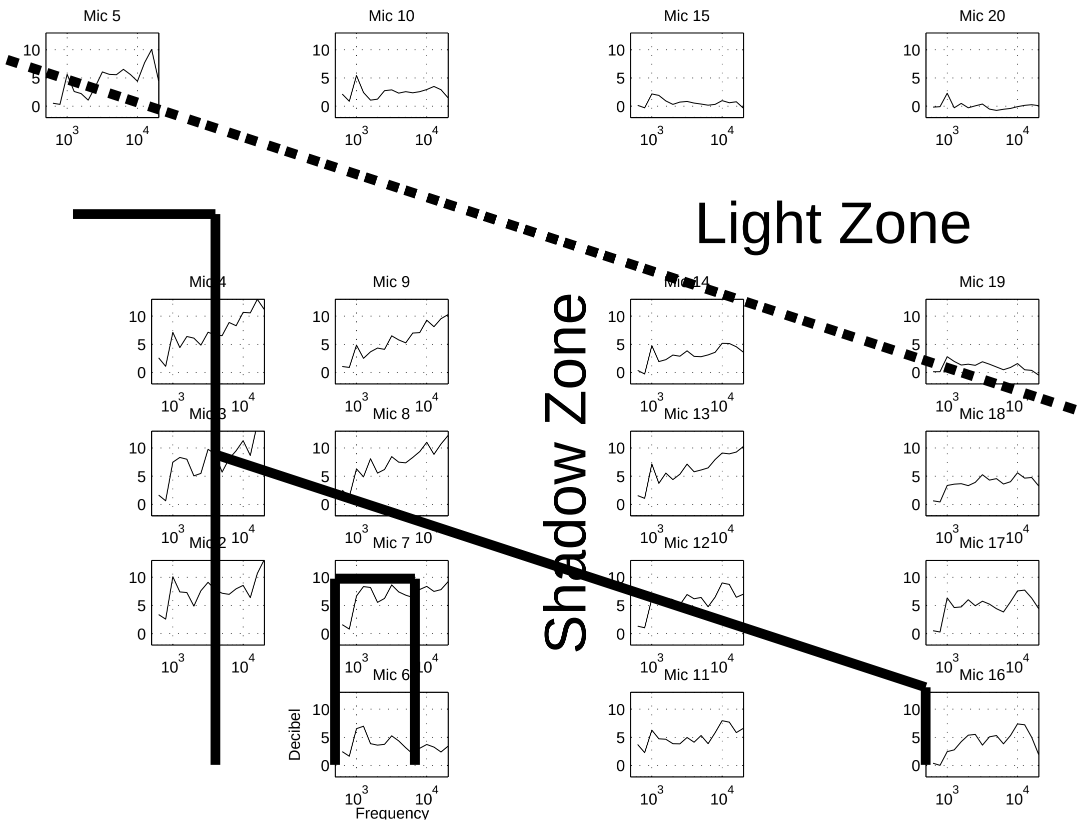


Figure 6.7: Differences of the SPL for when the aircraft is, and is not, present for each microphone, shown in their respective positions on the floor, for when the nacelle is at $x = 0.28x/c_y$, $z = 1.85z/h$ (Experiment 26 in Figure 6.5). The thick line indicates the aircraft model outline, with the thick dashed-line indicating the shadow outline of the wing leading edge on the floor.

Figure 6.7 shows the differences in the noise spectrum at each of the 20 microphones for the case of the nacelle positioned at $x = 0.28x/c_y$, $z = 1.85z/h$. Also shown is the projection of the wing and the nacelle on the floor, in relation to the microphones. To assist in the explanation of the results we have also projected the outline of the aircraft model onto the floor, as seen from a point source at the centre of the duct. This helps in determining which microphones are in the shadow zone, and which are in the light zone. In the shadow region the noise reductions are significantly higher compared with those in the light zone. Note that the noise reduction in the shadow zone generally increases linearly with the logarithm of frequency, suggesting that the reduction in mean square pressure follows a frequency power law. This is consistent with classical diffraction theory, discussed in greater detail in Section 6.4.1. Outside of the shadow zone, where the duct is in direct line of sight of the microphones, the SPLs indicate very little noise attenuation, suggesting that noise shielding is confined to the shadow zone at high frequencies.

Using the results recorded at the 40 nacelle positions, and the Kriging model fitted to these, a surrogate contour map was obtained (Figure 6.8) showing the dependence of the performance metric Δ on streamwise (x) and vertical (z) nacelle displacement. The white dots indicate actual nacelle positions as part of the Orthogonal Array Latin Hypercube (see Section 6.2.1), as well as the four vertices in our variable range. The stars denote experiment positions that are also part of the Latin Hypercube, but their readings were not included in the surrogate generation, due to higher than background microphone pressures in the results between 0-500Hz. As expected, Figure 6.8 shows

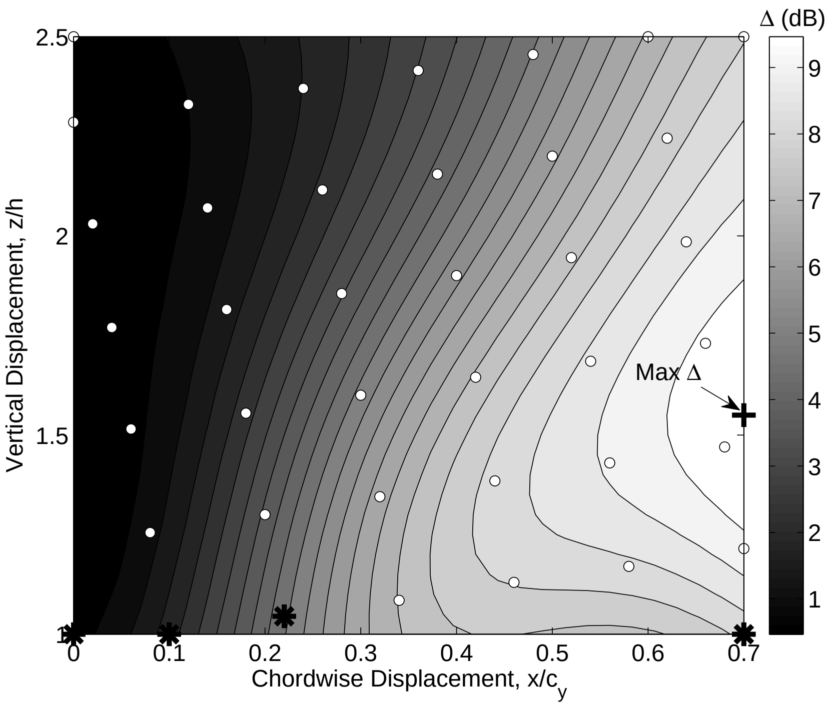


FIGURE 6.8: A Kriging surrogate model of the overall sound shielding metric Δ with the control variables. The white circles indicate the nacelle positions used to create the model, with the stars being nacelle positions that were tested but their results had to be discarded. The '+' indicates the maximum shielding position.The surrogate leave-one-out cross-validation RMSE yields 0.026.

that, in general, the further back the engine is positioned from the leading edge, the greater the shielding. Note that, for streamwise positions forward of the mid-chord, greater noise reductions are obtained when positioned vertically closer to the wing. However the greatest shielding is achieved when the nacelle is in the fully aft position and 1.5 nacelle diameters above the wing.

6.4 Analysis of Results

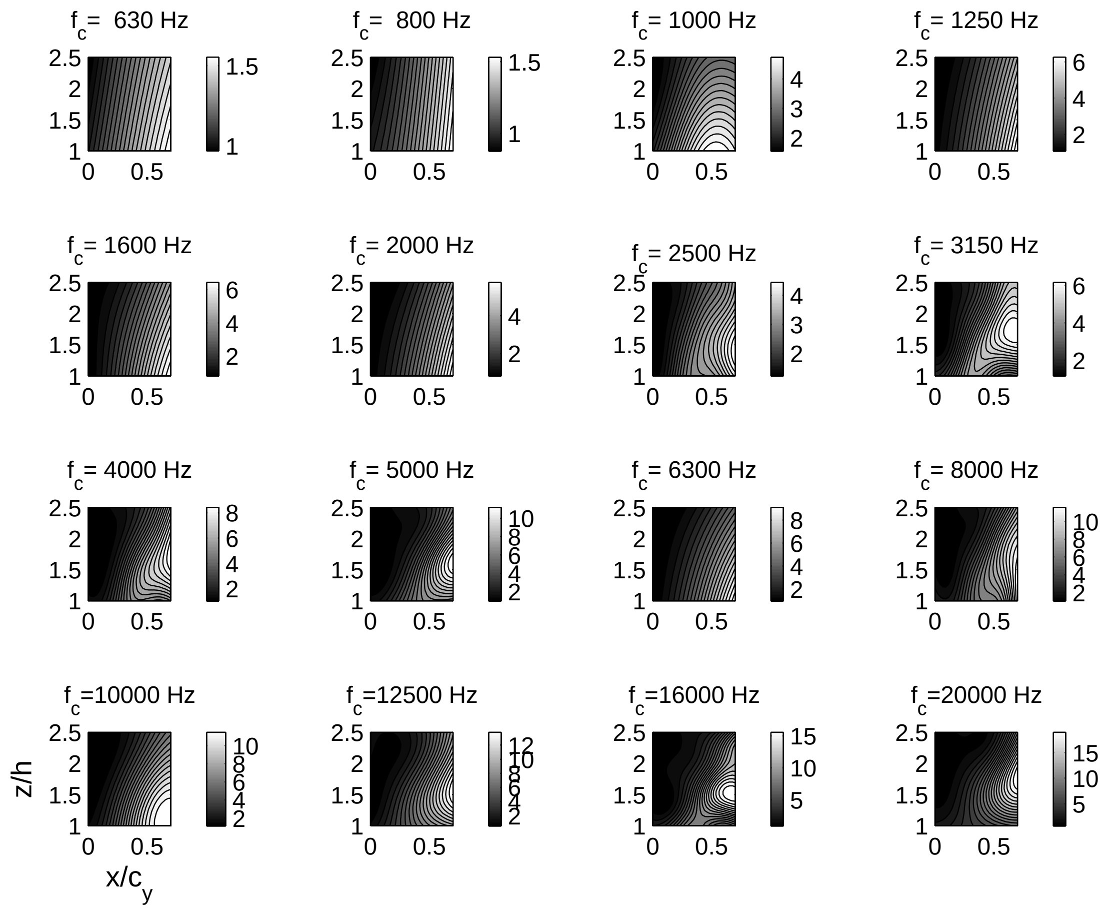


FIGURE 6.9: Kriging surrogate models of the noise shielding metric $\Delta(f_c)$ for each one-third octave band.

For a more detailed analysis of the variation of Δ with x and z in Figure 6.8, we applied the Kriging procedure to the frequency dependent shielding metric:

$$\Delta(f_c) = 10\log_{10}\left(\frac{\sum_{\theta,\phi}\bar{p_c^2}(f_c,\theta,\phi)}{\sum_{\theta,\phi}\bar{p_s^2}(f_c,\theta,\phi)}\right) \tag{6.2}$$

where $\bar{p^2}(f_c,\theta,\phi)$ is the 1/3 octave band mean square pressure. This process was performed for each octave band, generating the 16 contour plots shown in Figure 6.9. In general, $\Delta(f_c)$ increases as the engine is moved further aft of the wing, and increases as z decreases at frequency bands below 2.5kHz. Note, however, that at and above 2.5kHz an optimal vertical position generally exists near 1.5 z/h, resembling the optimum nacelle position found for the overall sound shielding metric Δ in Figure 6.8. To further understand this relationship of maximum shielding position we performed the Kriging procedure on the shielding metric dependent on each octave and microphone position:

$$\Delta(f_c, \theta, \phi) = 10\log_{10}\left(\frac{\bar{p_c^2}(f_c, \theta, \phi)}{\bar{p_s^2}(f_c, \theta, \phi)}\right). \tag{6.3}$$

The contour plots in Figure 6.10 show the surrogate models of Equation 6.3 for four microphones at the high frequency of 16 kHz (see Figure 6.7 for the microphone positions respective to the aircraft model). Microphone 7, positioned directly underneath the wing, shows little dependence of $\Delta(f_c, \theta, \phi)$ on vertical position, with the shielding linearly increasing the further back the nacelle is placed across the whole plot. Microphones 8, 14, and 20, all positioned forward of the leading edge, are ordered according to their distances away from the leading edge of the wing, with microphone 8 being the closest. All of these plots show an irregular pattern when the nacelle is placed at its furthest aft position. As indicated by the behaviour of the frequency dependent shielding metric (Figure 6.9), and with the evidence from Figure 6.10 , the nacelle position that achieves maximum shielding varies with frequency at all microphones positioned ahead of the leading edge of the wing for frequencies at and above 2.5kHz. The corresponding wavelength is 0.136m, which approximately coincides with the radius of curvature of the wing leading edge at the spanwise station of the engine. This indicates that the thickness of the blunt leading edge may have a role to play here, though further experiments would be needed to establish a definite link.

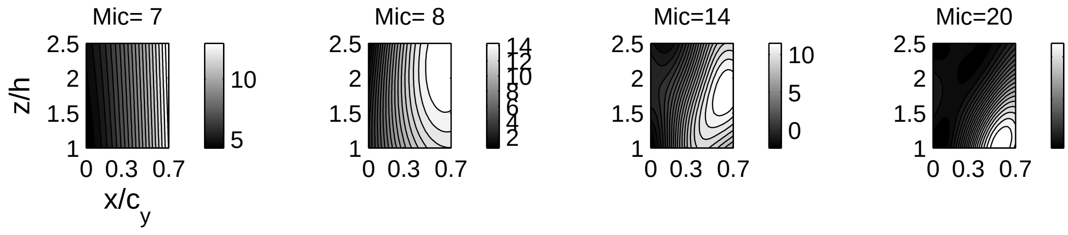
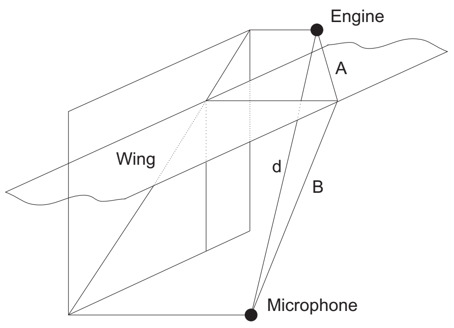


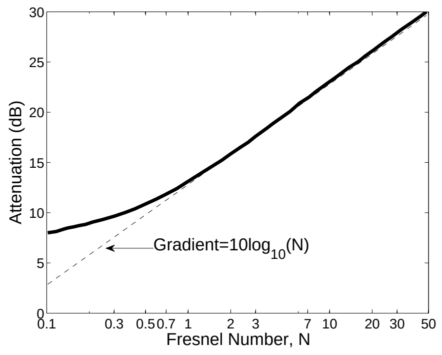
FIGURE 6.10: Kriging surrogate models of the sound shielding metric $\Delta(f_c, \theta, \phi)$ for four microphones at a centre frequency of 16 kHz.

6.4.1 Diffraction Theory

Further insights into the shielding performance of the various designs can be gained by considering the diffraction of the engine noise around the wing by comparison with simple empirical half-plane diffraction theory, as described by Maekawa [1968]. Maekawa showed that the level of noise reduction obtained by diffraction around an infinite half plane can be predicted solely from the Fresnel Number, N. With reference to Figure 6.11(a), the Fresnel number is defined as the difference between the shortest path around



(a) Diagram reproduced from Ref Bies and Hansen [1988] showing the variables used to calculate the Fresnel Number, N.



(b) Sound attenuation by a semi-infinite screen in free space (reproduced from Ref.Koyasu and Yamashita [1973]).

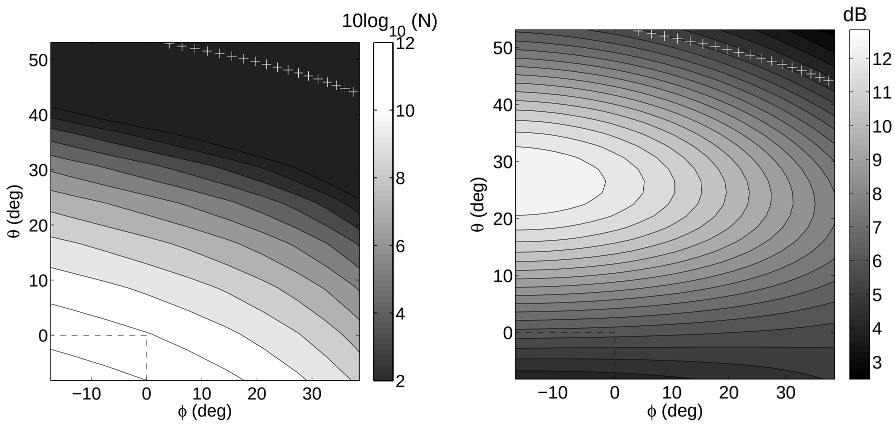
FIGURE 6.11: Variables used in the calculation of the Fresnel number and its relationship with noise attenuation.

the barrier from the source to the receiver (lengths A and B) and the geometric shortest distance from source to receiver (length d), compared to the acoustic wave length

$$N = (2/\lambda)(A + B - d). \tag{6.4}$$

Maekawa found that the rate of attenuation, defined as the sound pressure level difference with and without the acoustic barrier present, increases linearly with $\log_{10}(N)$ above $N = 1$. This relationship can be seen in Figure 6.11(b), reproduced from Koyasu and Yamashita [1973]. Contours of Fresnel number under the wing for each design (an example is shown in Figure 6.12) can be computed for any arbitrary nacelle position. Comparing the contours of $\log_{10}(N)$ with contours of $\Delta(f_c, \phi, \theta)$ at a third octave frequency we find that, close to the leading edge region, both experimental and analytical contours follow the projection of the leading edge. However, there are obvious differences between the theoretical and experimental results.

To further explore the relationship between half-plane diffraction theory and our measurements of Δ, for any experiment and microphone, $\Delta(f_c, \phi, \theta)$ can be plotted against N in order to establish a relationship between them. In Figure 6.13 we show these plots for three nacelle positions at three microphone positions. To aid us in the discussion, we use negative N values to indicate the results for when a microphone is in direct sight of the source, with the positive N values indicating a microphone situated in the shadow zone where diffraction theory is expected to apply. Indeed, the plots show that $\Delta(f_c, \phi, \theta)$ is negligible when in the light zone, and in some cases there is even a small increase in noise. As we move into the shadow zone we begin to see that $\Delta(f_c, \phi, \theta)$ increases almost linearly with $\log_{10}(N)$. In the shadow region, Figure 6.13(a) shows a gradual increase in the gradients of the linear regression lines. Close to, and above a Fresnel number of 1, the results show strong agreement between $\Delta(f_c, \phi, \theta)$ and $10\log(N)$. We found insignificant differences in the trends between spanwise and streamwise microphone positions.



(a) Fresnel Contour. (b) Performance metric Δ.

FIGURE 6.12: Fresnel Number and shielding metric $\Delta(f_c, \phi, \theta)$ contours across the angular range of the microphones at a frequency band of 6300Hz for Experiment 10 (see Figure 6.5). The source is located at angle (0,0) and the + indicates the angular position of the leading edge.

Figure 6.14 provides an alternative perspective on the relationship between Fresnel number and the measurements of Δ; we plot the results for every tested nacelle position against N at microphone 8 and at specific one-third octaves. This figure shows that in a single frequency band, the variation of $\log_{10}(N)$ against $\Delta(f_c, \phi, \theta)$ is fairly flat until near $N = 1$, where $\Delta(f_c, \phi, \theta)$ follows the $10\log(N)$ relationship . The example results from microphone 8 (Figure 6.14) show that for $N \geq 1$ the designs follow the $10\log_{10}(N)$ relationship. For $N < 1$, $\Delta(f_c, \phi, \theta)$ deviates from this relationship and at low frequencies tends to zero. At higher frequencies there is also some shielding when microphone 8 is in the direct line-of-sight of the nacelle.

6.5 A Possible Analytical Model to Assess Noise Shielding

We have compared in the previous section the experimental results to half-plane barrier theory, showing in Figure 6.13 that $\Delta(f_c, \theta, \phi)$ is proportional to $10\log_{10}(N)$;

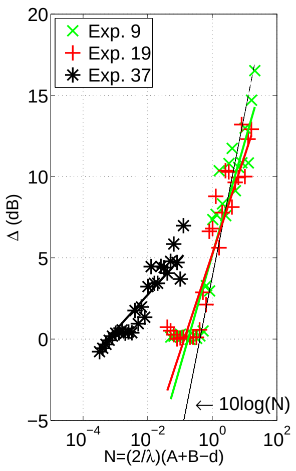
$$10\log_{10}\left[\frac{\bar{p_c^2}(f_c,\theta,\phi)}{\bar{p_s^2}(f_c,\theta,\phi)}\right] \propto 10\log_{10}(N(f_c,\theta,\phi)) \tag{6.5}$$

for $N > 1$. If we assume

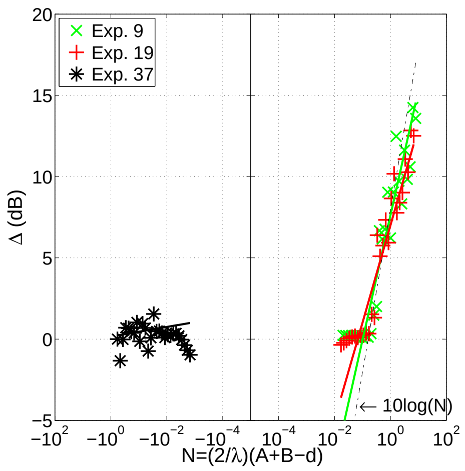
$$\frac{\bar{p_c^2}(f_c,\theta,\phi)}{\bar{p_s^2}(f_c,\theta,\phi)} = \begin{cases} N(f_c,\theta,\phi) & \text{for} \quad N(f_c,\theta,\phi) \geq 1 \\ 1 & \text{for} \quad N(f_c,\theta,\phi) < 1 \end{cases} \tag{6.6}$$

a model of total noise shielding Δ can be obtained that is dependent only on $N(f_c, \theta, \phi)$;

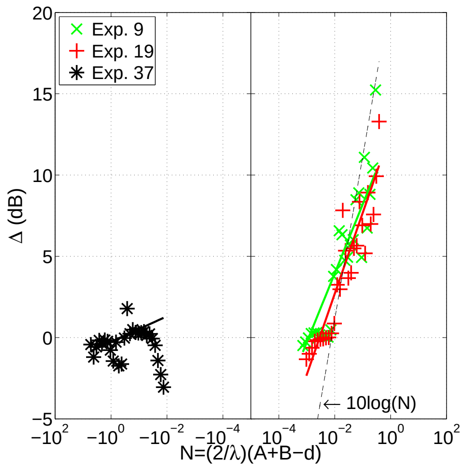
$$\hat{\Delta}_N = 10\log_{10}\left[\frac{\sum_{f_c}\sum_{\theta,\phi}\bar{p_c^2}(f_c,\theta,\phi)N(f_c,\theta,\phi)\Delta_f(f_c)}{\sum_{f_c}\sum_{\theta,\phi}\bar{p_c^2}(f_c,\theta,\phi)\Delta_f(f_c)}\right] \tag{6.7}$$



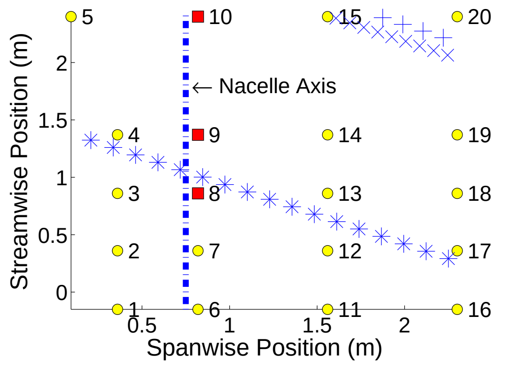
(a) Microphone 8



(b) Microphone 9



(c) Microphone 10

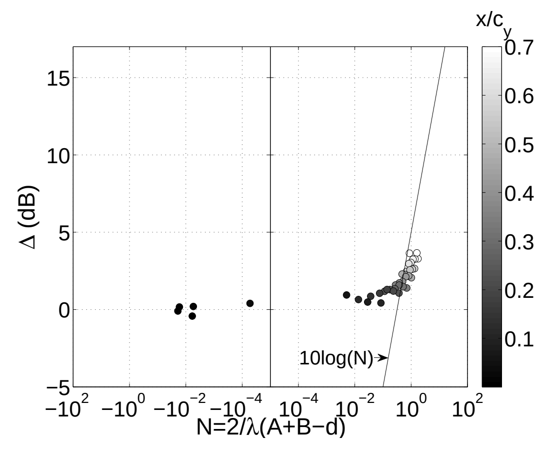


(d) Positions of the microphones in relation to the aircraft model, with the leading edge shadow lines shown on the ground

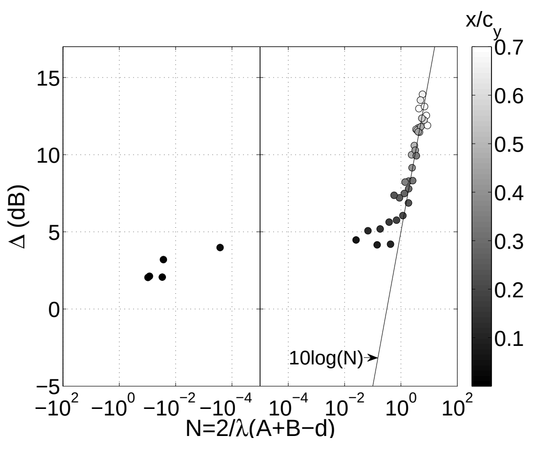
FIGURE 6.13: $\Delta(f_c, \phi, \theta)$ against Fresnel Number N for experiments 9 (x=0.62x/c_y, z=2.245z/h), 19 (x=0.42x/c_y, z=1.645z/h), 37 (x=0.06x/c_y, z=1.515z/h) at microphones 8, 9, and 10, with their respective linear regression lines.

where $\Delta_f(f_c)$ is the band width of the one-third octave f_c. Contours of $\hat{\Delta}_N$ computed from Equation 6.7 are shown in Figure 6.15. Close to the leading edge of the wing the contours of $\hat{\Delta}_N$ are similar to the overall shielding Δ contours shown in Figure 6.8. However, non-linearity at aft nacelle positions, leading to the maximum shielding position around 1.56z/h above the wing, is not captured by the Fresnel model.

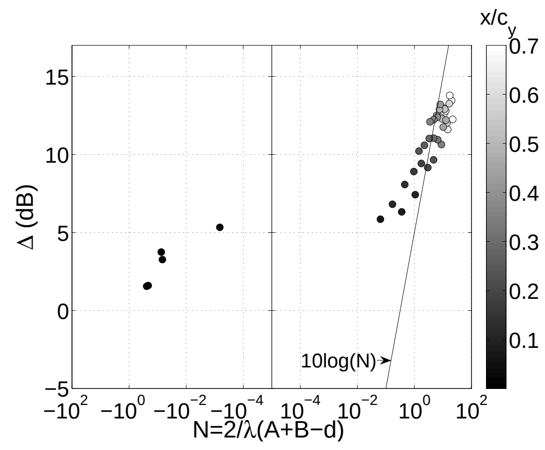
Simple diffraction theory is therefore useful as a first order (conceptual) design tool but detailed design requires experimental studies or more sophisticated prediction schemes. Possible reasons for the discrepancy between the measured values of Δ and those predicted by classical half-plane diffraction theory are:



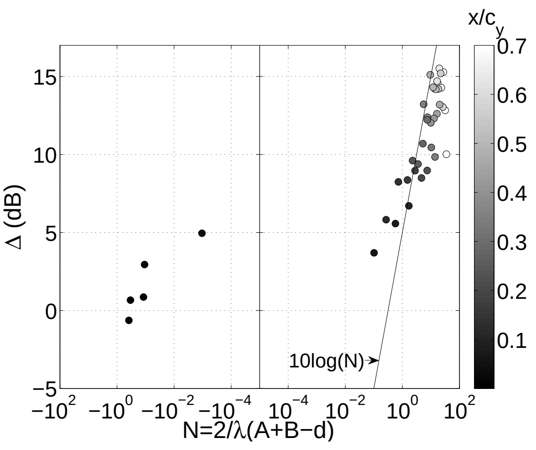
(a) f_c = 800 Hz



(b) f_c = 4 kHz



(c) f_c =10 kHz



(d) f_c =16 kHz

FIGURE 6.14: $\Delta(f_c, \phi, \theta)$ against Fresnel Number N plotted at various frequencies at microphone 8 with all designs plotted, with their linear regression lines.

- **Duct sound directivity** - The distribution of mean square pressure in the forward arc is given by $\bar{p}^2 \propto \cos\theta$, where θ is the polar angle measured from the duct axis[Joseph and Morfey, 1999]. The half-plane diffraction theory assumes a point source.

- **Curvature of wing leading edge** - The Fresnel model assumes a flat plate with a sharp edge; the actual wing has a smooth, round leading edge, the radius of curvature of which reduces linearly along the span.

- **Influence of the fuselage** - The Fresnel model does not take into account the influence of the fuselage.

- **Assuming $N = 1$ for $N < 1$** - For the Fresnel model used to find $\hat{\Delta}_N$, the assumption was made that as N is not proportional to Δ below $N = 1$, and the attenuation at these positions are comparably small to those above $N = 1$, it was thought not to alter the contours of Figure 6.15 considerably.

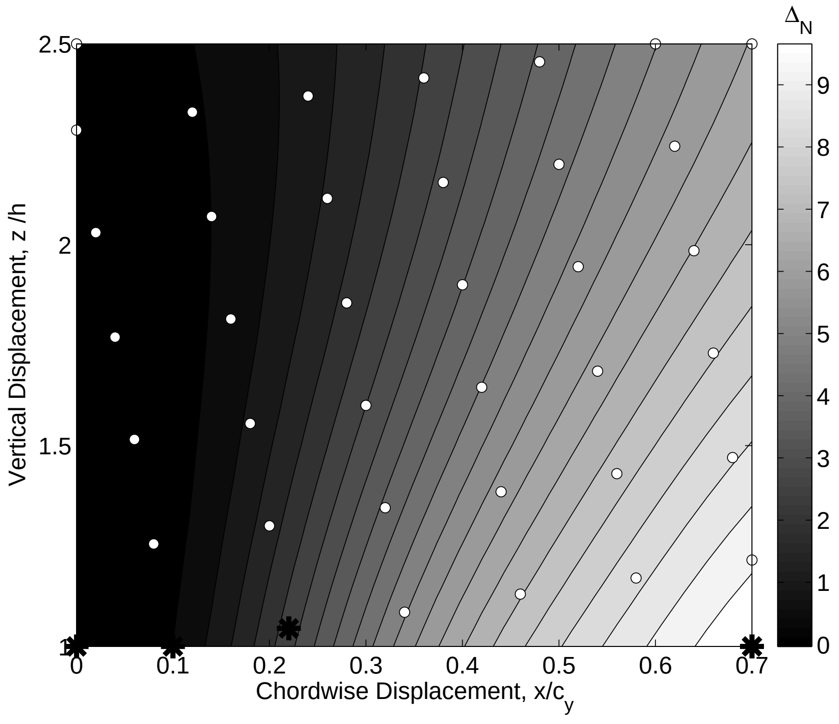


FIGURE 6.15: Contour plot of the sound shielding metric Δ_N.

6.6 Summary

The study described here was motivated by the strict community noise targets facing the airline industry and encouraged by earlier experimental indications (Such as Berton [2000]) that smart design can use the airframe of an aircraft as a shield against broadband noise emanating from the inlet. We have conducted a series of noise shielding experiments on a configuration that requires the smallest departure from existing designs: a conventional tube-and-wing airliner with over-the-wing engines.

The analysis presented here highlights the potential of this layout to create a noise shadow zone under a climbing airliner passing over communities near an airport. Our parametric study has also shown that, while basic diffraction theory can provide a first order estimate of the shielding performance for a given design, the phenomena involved are sufficiently subtle to warrant a more sophisticated experiment, such as the one presented here. Finally, we have demonstrated the viability for airframe acoustic design of modern surrogate modelling techniques as parametric design investigation tools, which are particularly useful in cases where measuring the performance of a design is costly and therefore limits the size of the design space.

Noise is, of course, just one of the engineering constraints that drive the design of a modern airliner and it is therefore clear that further analysis of the configuration discussed here is necessary. We take a step in this direction in the next chapter where we present an aerodynamic study on the same over-wing engine installation. More specifically, we consider the following question: how does the relative position of the engine with respect to the airframe influence the overall drag of the aircraft, as well as the characteristics of the flow entering the engine?

Chapter 7

The Over-Wing Engine Configuration: An MDO Approach

We presented in Chapter 4 an optimization procedure that finds the optimal solution with respect to a single objective. We used a similar single objective study to assess the quality of the application-specific class functions in Chapter 5. However, we discussed in Chapter 2 that, in general, complex engineering design problems include multiple performance requirements from a range of disciplines. One such problem is that of the over-wing engine installation, considered in Chapter 6.

We consider here the effect of engine placement on the over-wing engine installation to the aerodynamic performance of the aircraft. To evaluate the aerodynamic performance we conducted steady state RANS simulations using the Fluent [2011] flow solver at cruise flight conditions, from which we obtained an overall drag coefficient value C_D. In order to assess the installation effects on the engine itself, for each nacelle position, we extracted total pressure values at the fan face, enabling us to compute a fan face pressure distortion criterion [Seddon and Goldsmith, 1999]. We performed these analyses on the 40 designs tested in the noise study, presented in Chapter 6, generating surrogate models from the performance values, as we did with the noise shielding. We use these surrogate models to find the non-dominated set of designs that comprise the so-called Pareto surface of these three objectives.

7.1 Aerodynamic Studies on the Over-Wing Engine Configuration

The aerodynamic performance implications of installing the engines above the wing have been studied, but the trade-offs between noise generation and aerodynamic performance are less well understood. This is the motivation behind the work presented here, where we consider aircraft configurations that use a pylon to attach the engines to the wing, and exclude those that incorporate the nacelle within the wing geometry (as in Kinney and Hahn [1997] and Hill et al. [2009]). Ronzheimer et al. [1996] investigated the aerodynamic characteristics of the Fokker VFW 614 aircraft, using an Euler code. They found that the flow inboard of the pylon was characteristically different from the outboard section of the wing, due to the channel created by the fuselage, wing, pylon and nacelle. Fujino and Kawamura [2003] comment that over-wing engines may cause a strong shock wave to occur that reduces drag divergence Mach number. However, they used the interaction of the wing with the nacelle and pylon to create a design that has better drag-rise characteristics compared to that of the clean wing configuration. Yoneta et al. [2010] conducted a similar study, showing that the performance of an over-wing engine configuration is highly dependent on the placement of the engines with respect to the wing surface. Both of these studies idealize the engines as flow through nacelles, as Fujino and Kawamura [2003] report insignificant differences to the wave drag when modelling the jet from the engine. No indication was given to the quality of flow entering the engine.

7.2 Aerodynamic Investigation

To determine how variations in nacelle position affect the aerodynamic efficiency of both the engine and the whole aircraft the flow around the 40 designs of the Latin Hypercube was analysed using the Reynolds Averaged Navier-Stokes (RANS) solver Fluent [2011]. The geometry was, once again, that of the DLR-F6 research aircraft model that was also used in the American Institute of Aeronautics and Astronautics (AIAA) drag prediction workshops [Brodersen and Sturmer, 2001], with its FX2B fairing around the connection between the wing and the fuselage. The engine characteristics were modelled assuming a CFM-56 type engine was installed, with flow characteristics found using analytical equations from basic engine parametric cycle analysis [Mattingly, 1996] and assuming a capture area ϵ of 0.75 enters the engine:

$$\epsilon = \frac{A_\infty}{A_f} \tag{7.1}$$

A_∞ is the area of the streamtube entering the engine at a position far upstream of the engine, and A_f is the fan area.

The same flight conditions were used as defined in the third AIAA drag prediction workshop [Laflin et al., 2005]: namely, Mach number $M = 0.75$ and Reynolds number $\mathrm{Re} = 5 \times 10^6$. The angle of attack was continually altered to obtain a target C_L of 0.5. The pylon cross section was the symmetrical SC(2)-0012 airfoil. The sweep angle of the pylon for each design was determined by a mutual connection point of the pylon leading edge on the wing and nacelle surfaces for each design[1].

7.2.1 Description of the CFD Setup

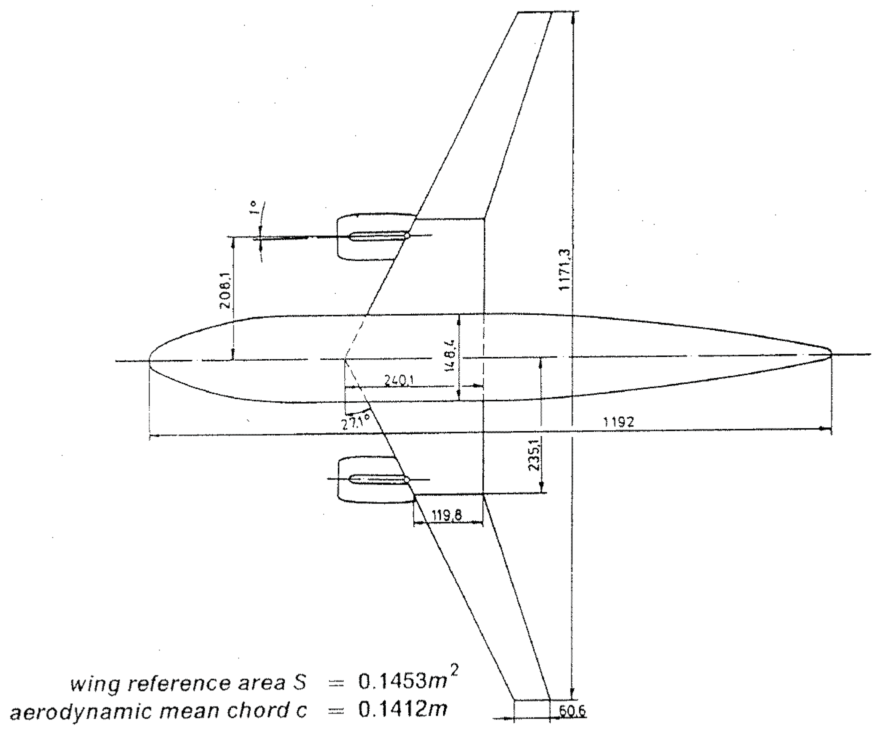
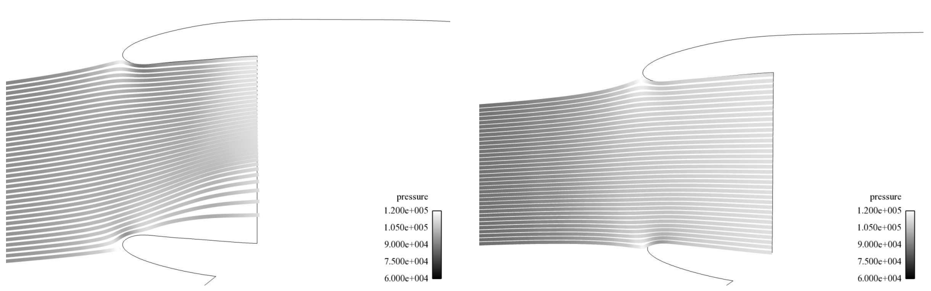

1°
208.1
1171.3
148.4
240.1
27.1°
1192
235.1
119.8
60.6
wing reference area S = 0.1453m^2
aerodynamic mean chord c = 0.1412m

FIGURE 7.1: Diagram of the DLR-F6 research model presented in Rossow and Ronzheimer [1992], and Brodersen and Sturmer [2001] (with dimensions in mm). Courtesy of DLR.

The geometry used in this investigation had the dimensions of the DLR-F6 wind tunnel model [Rossow and Ronzheimer, 1992, Brodersen and Sturmer, 2001], as shown in Figure 7.1. To generate the meshes for the CFD analysis we used the octree mesher Harpoon [2011]. The geometry of the fuselage and wing was split into sections using CATIA, allowing each section to have a unique mesh size. A similar process was achieved with the pylon and nacelle, but these geometries were generated in MATLAB to provide automatic redefinition of nacelle position and pylon curvature integration with the wing and the nacelle. Stereolithography (STL) files were generated whereby the end points of the pylon STL file were integrated with the wing STL file that was created in CATIA.

[1]The trailing edge connection with the nacelle was allowed to float so that we obtained the same gradient on the leading and trailing edges of the pylon. This did result in the pylon trailing edge finishing aft of the nacelle outlet for some nacelle positions.



(a) $x = 0x/c_y$, $z = 1z/h$, Unconverged solution.

(b) $x = 0.51x/c_y$, $z = 1.93z/h$, Fully converged solution.

FIGURE 7.2: Streamlines, coloured by pressure, of the flow entering the engine. Pressure values are in Pascals.

This improves Harpoon's ability to produce an accurate mesh of this junction. The meshes comprised of between 7-8 million cells for each model. The RANS equations were solved in Fluent [2011] using the density based implicit solver and the Spalart-Allmaras turbulence model. This setup was validated against the results from the AIAA drag prediction workshop [Vassberg et al., 2007] for the DLR-F6 aircraft model. The CFD simulations were run on the University of Southampton's Iridis 3 supercomputer, which has 1,008 Intel Nehalem compute nodes, with each having 22 GB of RAM. Each run used 8 nodes and was run for around 6,000 iterations, taking just under 28 hours to complete. A more detailed description of the CFD process is given in Appendix B.

To define the aerodynamic efficiency of the aircraft we used the total C_D calculated via the RANS solve. As a measure of engine performance, we used the distortion coefficient at the fan face [Seddon and Goldsmith, 1999], based on a 60^o sector angle:

$$\mathrm{DC}(60) = \frac{P_{t,f} - P_{t,60}}{P_{t,f}} \tag{7.2}$$

where $P_{t,f}$ is the mean total pressure on the fan face, and $P_{t,60}$ is the mean total pressure in the sector of the fan face that provides the lowest pressure.

7.2.2 Inlet Efficiency Results

For a subset of engine positions a separation bubble emerged on the lower part of the nacelle inlet (see Figure 7.2(a)). These designs were clustered in a single region of the design space: at forward and low nacelle positions, where the oncoming flow is highly deviated due to its interaction with the wing leading edge. For the purposes of this study we have simply postulated these designs to be infeasible.

To determine the boundary of the infeasible region in the design space we performed a classification learning procedure. Here, all tested feasible designs were allocated a value of one, with infeasible designs receiving a value of minus one. We performed a Support Vector Classification process [Gunn, 1998], where a quadratic programming method was

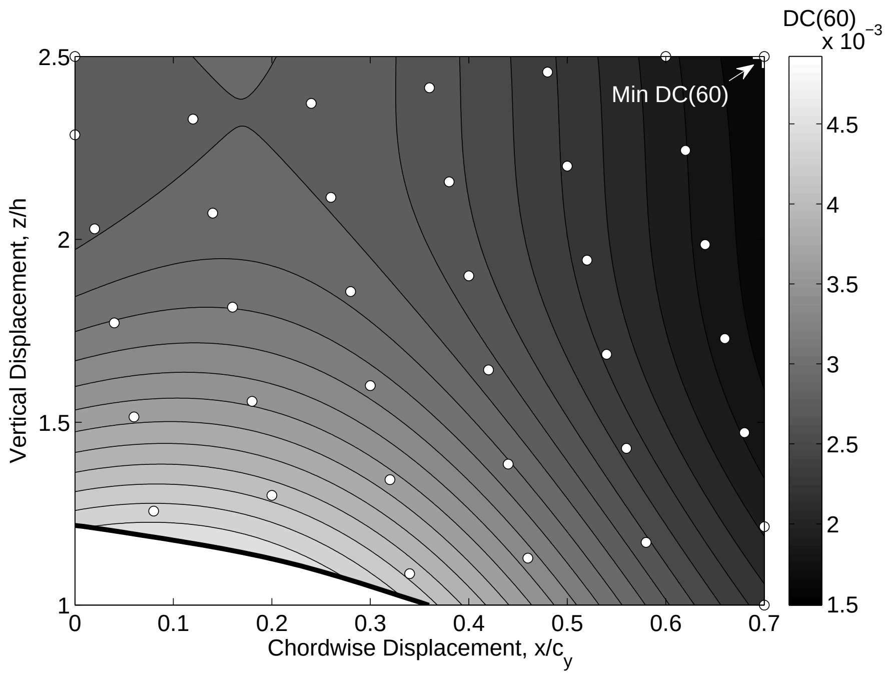


FIGURE 7.3: Surrogate plot of the pressure distortion at the fan face, with the white dots indicating the values of the tested designs. The surrogate leave-one-out cross-validation RMSE yields 0.071.

used to maximise the margin of the closest points either side of the feasible boundary, where each point on the feasible boundary had a value of zero. This usually involves fitting a hyperplane through the two sets of values. However, it was uncertain if our classification boundary would be linear. We therefore used a linear Spline kernel to map the input space to a higher-dimensional feature space, where we fitted a hyperplane through the two sets. This resulted in the infeasible design region shown by the white space in Figure 7.3.

The Kriging model shown in Figure 7.3 was generated using the feasible tested designs only. As expected, when the nacelle is positioned close to the feasible/infeasible boundary, we obtain high pressure distortion. The lowest pressure distortion is obtained for when the engine is positioned high and aft of the wing, i.e. furthest away from any deviation of the freestream flow.

7.2.3 Aerodynamic Efficiency Results

Figure 7.4 shows contours of the Kriging model fitted to the C_D values for each nacelle position. Again, the white dots indicate the nacelle positions that were tested. Minimal drag was achieved at the most aft and lowest engine position. Maximum drag was found for engine positions around $x/c_y = 0.15$, with the drag generally lower for further forward engine positions.

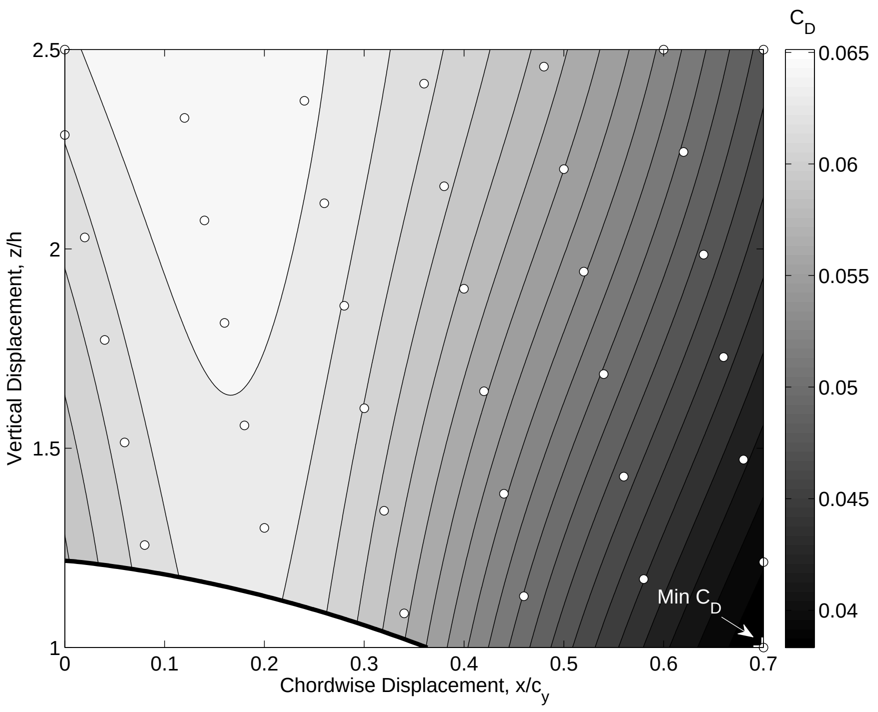


Figure 7.4: Surrogate plot of the drag objective, with the white dots indicating the values of the tested designs. The surrogate leave-one-out cross-validation RMSE yields 0.040.

To further study the effect of the pylon and nacelle on the aerodynamic efficiency, and to find the causes of the behaviour in Figure 7.4, the drag from the individual components was identified. Figure 7.5 shows contours of the Kriging models fitted to the C_D contributions of the different parts of the aircraft: the wing and fuselage, the nacelle, and the pylon. The relationship of total drag with nacelle position is dominated by the pressure drag, with the viscous drag only contributing little in comparison. Thus the trends shown in Figure 7.5 can be taken to be due to differences in pressure on the aircraft geometry. Pressure contours for a low drag and a high drag nacelle position are shown in Figures 7.6(a) and 7.6(b), respectively. In both figures we see the presence of two shocks, either side of the pylon. In the high drag case the outboard shock spans the pylon, aft of the pylon mid-chord. The inboard shock emanates from the interconnection of the nacelle and the pylon, close to the pylon leading edge, and meets the interconnection between the pylon and wing at the pylon mid-chord. This is similar to the low drag case. However, the outboard shock follows the same relationship to the inboard case, unlike the outboard shock in the high drag case. This difference can be attributed to the deflection of the flow direction as a consequence of the sweep on the wing leading edge, which returns to be tangential to the freestream flow at the trailing edge.

Using the information from the pylon pressure contours with the surrogate model in Figure 7.5(c), we can determine that the large outboard shock causes a significant drag increase on the pylon, which is increased due to the height of the pylon. Low drag on

Objective	x/c_y	z/h	Optimal Value
Δ	0.7	1.55	9.9 dB
C_D	0.7	1	0.038
DC(60)	0.7	2.5	0.0016

Table 7.1: The optimal position of the engine for each performance objective.

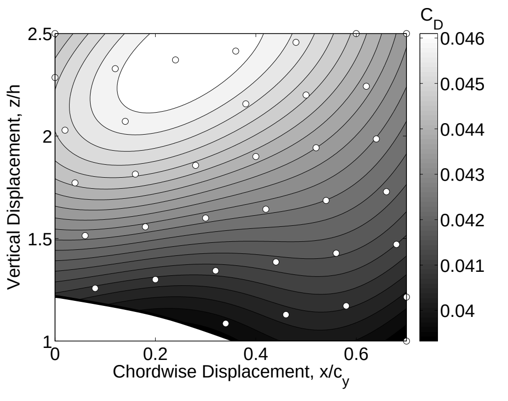
the pylon is found when the shock is close to the pylon leading edge, and where its propagation distance is low.

Figure 7.5(a) shows the maximum drag on the wing/fuselage to be for a nacelle position near the maximum thickness of the wing, and at the maximum vertical position, where the shock on the pylon can propagate further along the wing (see Figure 7.6(b)). Minimum drag on the wing corresponds to the furthest aft position of the engine, close to the wing, where the pylon shock propagates across the aft portion of the wing, as shown in Figure 7.6(a). A higher angle of attack ($\alpha \approx 0.5 - 2^o$) is needed to meet the design C_L compared to the clean wing ($\alpha \approx -0.4^o$), due to the reduced lift-producing surface area of the wing and due to the area of high pressure where the pylon leading edge meets the wing. The long spanning shocks created by the powerplant geometry at high nacelle positions help to reduce this effect, but consequently cause a drag increase that is greater than the induced drag caused by changing the angle of attack at other nacelle positions.

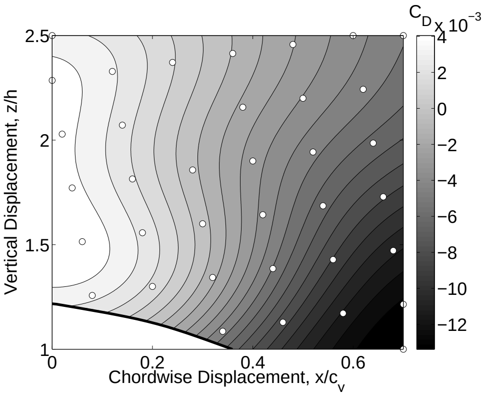
Maximum nacelle drag, as seen in Figure 7.5(b), is generated at a forward nacelle position, where the pylon is heavily unswept, allowing a strong shock to form at the nacelle-pylon intersection. Minimum nacelle drag occurs at the lowest and aft nacelle position, at maximum sweep. The negative drag value at this and the surrounding nacelle positions is due to the high pressure field generated by the forward thrust at the engine exhaust interacting with the rear wall of the nacelle, causing the pressure integral aft of the nacelle to be greater than at the front.

7.3 Trade-Off Study

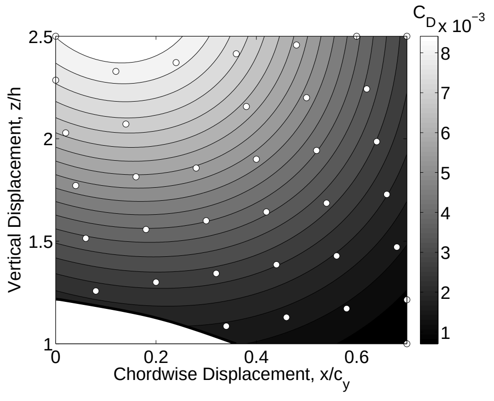
We have presented here and in Chapter 6 the results of the noise and CFD experiments, producing surrogate models of the three performance objectives. The optimal position of the engines for each objective is shown in Table 7.1. These optimum positions, and the relationships seen in the plots of the surrogate models shown in Figures 6.8, 7.4 and 7.3, indicate that the best streamwise position for the engine is in an aft position on the wing. Vertically however, each objective requires a different engine position for optimal performance. This leads to a very difficult yet common conundrum in design engineering, where the multi-objective analysis yields several different solutions across the search space.



(a) Drag from Wing and Fuselage



(b) Drag from Nacelle

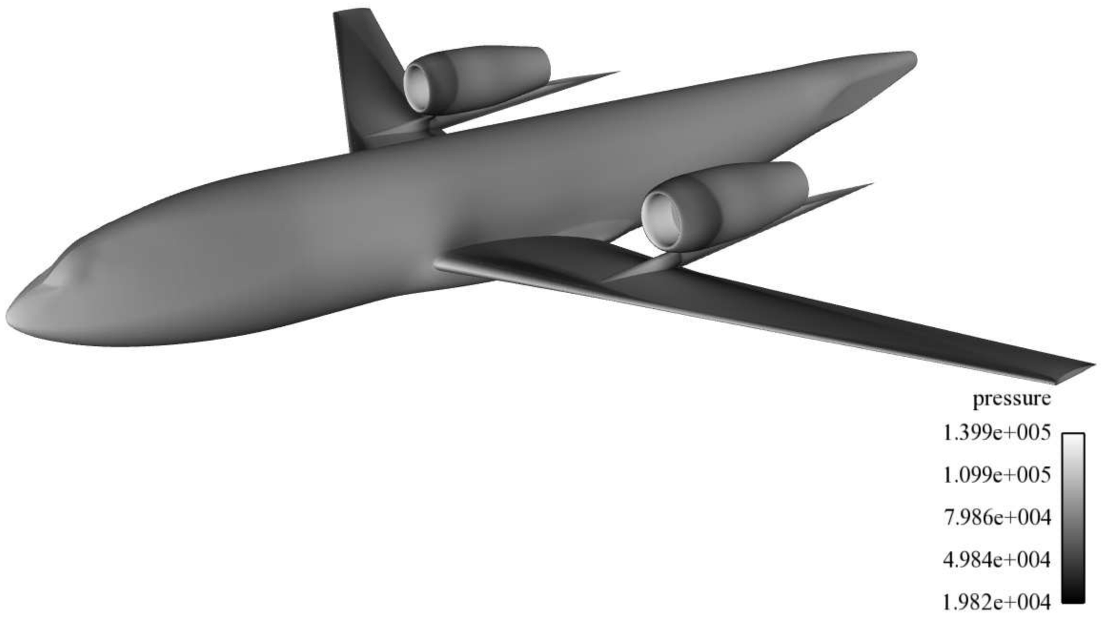


(c) Drag from Pylon

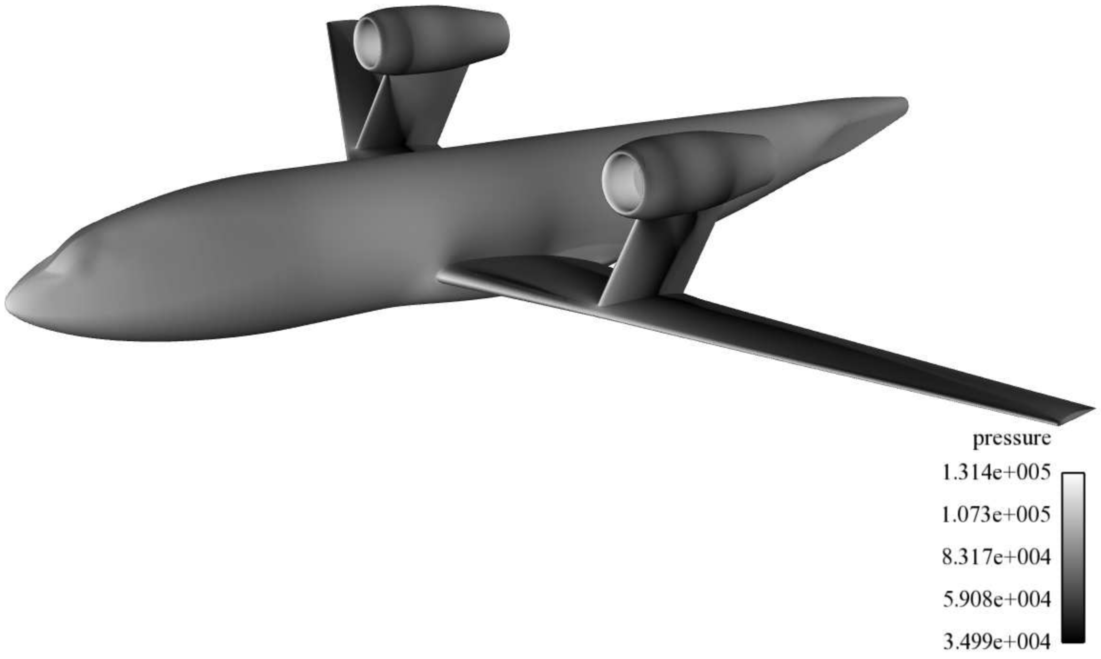
FIGURE 7.5: Contour plots of the Krig models fitted to the C_D contributions of different aircraft parts. Pressure values in Pascals.

The surrogate models of the performance metrics allow us to find an approximation to the performance of any nacelle position within the design space. To find relationships between the different performance objectives we have re-sampled the feasible region of the design space at the nodes of a uniform grid of 2474 points and plotted the corresponding objective values against the three objectives in Figure 7.7. An optimal solution would feature minimum DC(60) and C_D, with maximum Δ (i.e. the nearest and lowest point in Figures 7.7(b) and 7.7(a)). From the performance of the engine positions that form the convex curvature in this optimum region, we can conclude that no single nacelle position exists that is optimal for all objectives. These engine positions that make up the curve in the optimum region are likely to be optimal in some way, with any improvement in one objective reducing the performance of either of the other objectives. If this is true, these engine positions are classed as being non-dominated, and collectively comprise the Pareto set of solutions.

In the absence of a practical means of establishing the relative importance of each objective, such Pareto surfaces can be used to highlight non-dominated designs. For an even more accurate identification of the Pareto front we also ran a multi-objective Genetic Algorithm on the three surrogates.

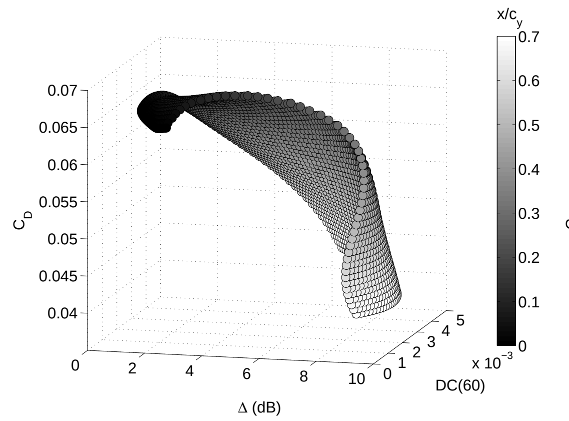


(a) $x = 0.7x/c_y$, $z = 1z/h$, (Low drag case)

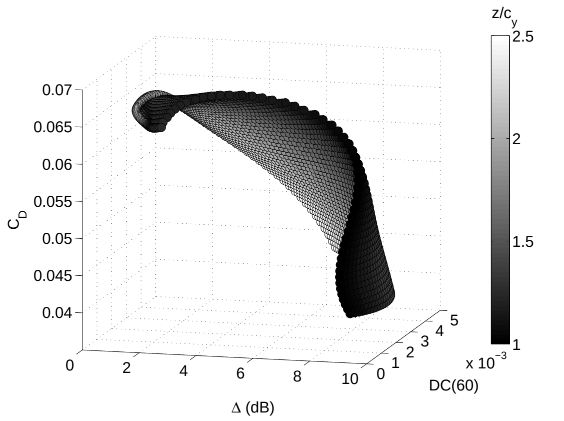


(b) $x = 0.12x/c_y$mm, $z = 2.33z/h$, (High drag case)

FIGURE 7.6: Pressure contours on the surface the DLR-F6 at two different nacelle positions. Pressure units are in Pascals.



(a) Coloured by engine streamwise position.



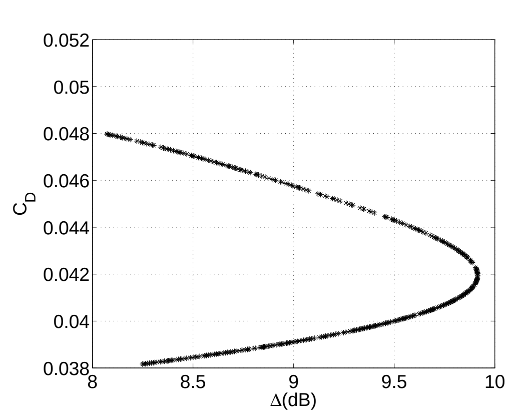
(b) Coloured by engine vertical position.

FIGURE 7.7: 2474 designs plotted against all objectives, with the objective values taken from the each performance metric's surrogate model.

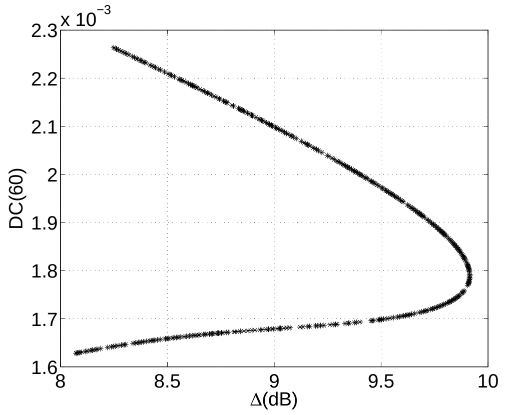
Specifically, we used the Non-dominated Sorting Genetic Algorithm (NSGA) II [Deb et al., 2002]. This uses a genetic algorithm (GA) to search for the non-dominated designs, by allocating the non-dominated solutions a rank of one, with the rest given ranks of two or higher (with rank two designs being non-dominated if we were to remove the rank one set, and so on). The GA is then guided through its exploration of the design space by the ranking system and rewarding or penalising each individual in the population, based on its distance from the Pareto front (encouraging Pareto optimality) and other members (encouraging a uniform spacing of designs along the Pareto front).

Figure 7.8 shows the Pareto set found using NSGA-II for each performance metric plotted against each other. Figures 7.8(a) and 7.8(b) are similar in that they show a plateau point; part of the Pareto front (above the plateau in the both figures) both performance metrics improve. On the other side of the plateau, the metrics conflict, where any improvement in C_D or DC(60) will be detrimental to the noise shielding Δ. Figure 7.8(c) shows that DC(60) always conflicts with C_D. The nacelle positions (x/c_y, z/h) of the Pareto set, shown in Figure 7.8(d), show that the optimal nacelle position is indeed at the aft end of the chordwise position range. The optimal vertical position is less evident; the purpose of the Pareto analysis here is to inform a wider analysis process, where other factors (e.g. structural weight) might be included within the design selection.

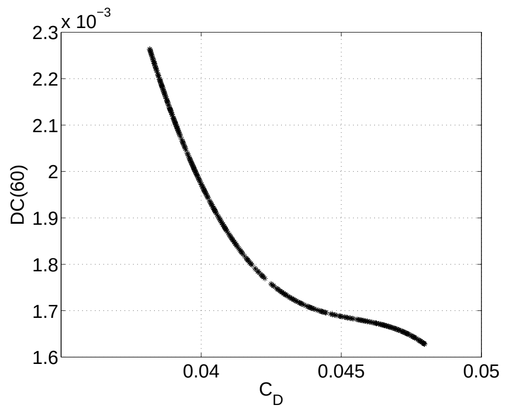
We have also conducted a CFD simulation for the case of a conventional design, where the engine is located underneath the wing in a position typical of airliners belonging to this class forward of the wing leading edge (see Figure 7.9). For this case we obtained a distortion coefficient of 0.0014 and a drag coefficient of 0.036. Note from Table 7.1 that the optimal positions for the over-wing case, although worse compared to the underwing design, are relatively close to those results. However, these optimal values represent three different designs, and so the chosen optimal compromise will be even poorer compared to the underwing case. If we consider the underwing case to give 0dB noise shielding



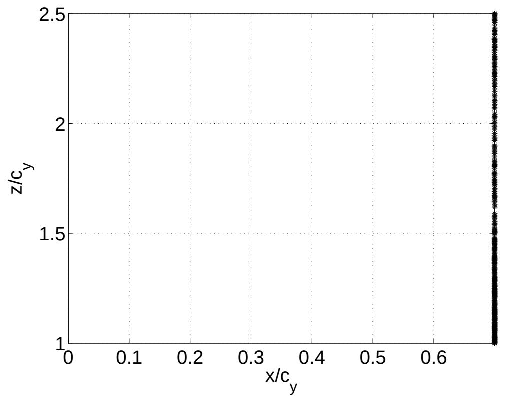
(a) Noise shielding Δ against Aero efficiency C_D.



(b) Noise shielding Δ against pressure distortion at fan DC(60).



(c) Aerodynamic shielding C_D against pressure distortion at fan DC(60).



(d) The nacelle positions of the Pareto set – all furthest aft, but covering a broad vertical position range

Figure 7.8: The NSGA II found Pareto set created using the Kriging models shown in Figures 6.8, 7.4 and 7.3.

(no aircraft structure is present to act as a barrier to the engine noise), we find that for slightly poorer aerodynamic efficiency and engine airflow quality we can gain significant noise reductions experienced by near airport communities.

7.4 Summary

In this chapter we have presented two single objective optimization investigations aimed at finding the ideal nacelle position in terms of overall airframe aerodynamic efficiency and pressure distortion at the engine fan face. The optimal nacelle position was found to be different for each performance metric, including the noise shielding presented in Chapter 6, and so a Pareto set of optimal nacelle positions was generated using the NSGA II algorithm on the surrogate models for each performance metric. The most appropriate position for the nacelle was found to be as far aft of the leading edge of the wing as possible. However, the Pareto set spanned most of the vertical design space, creating a difficult dilemma for choice of this design variable.

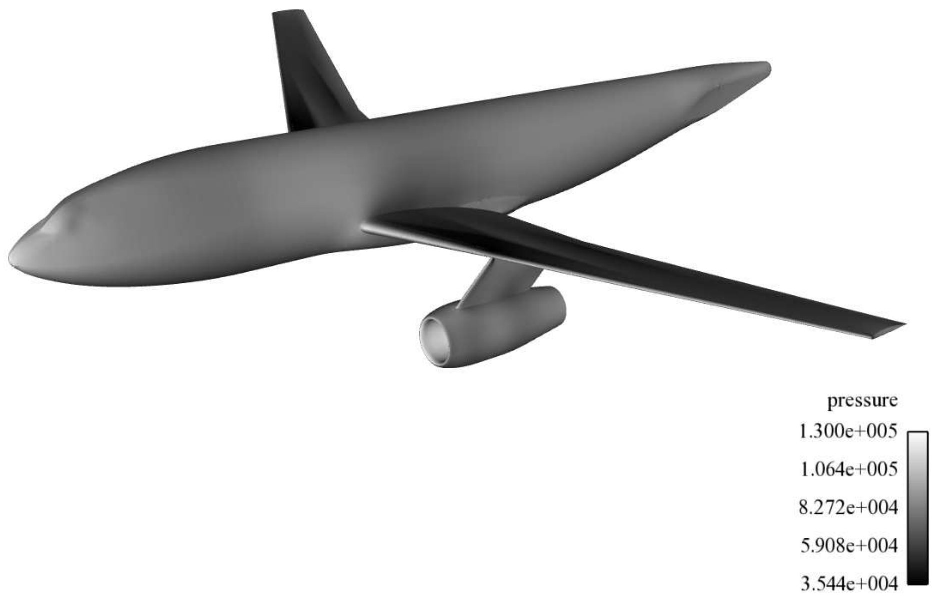


Figure 7.9: Pressure contours, in Pascals, for a design where the engines are located underneath the wing.

It must be noted that the three performance metrics chosen in this study are not the only ones to be considered to assess the over-wing aircraft configuration. Moreover, they may not be the most important. Flight dynamics and structural considerations may play a very important role too. Future studies should assess aircraft stability, structural supports, and flutter. The flexibility of the geometry could be increased too, so that the shape of the pylon and its interconnections can be optimized, which will have a significant effect on the shocks generated in the CFD computations and thus alter the C_D contours in Figure 7.4.

Chapter 8

Conclusions and Recommendations for Further Work

8.1 Conclusions

It is becoming standard practice to use optimization methods within an aircraft design context. This is evident by reading recent aviation journal publications, where it is rare to find the word ‘design’ without ‘optimization’ or ‘optimal’ in the article titles. The design community has also realised that the optimization process must consider all performance factors, whether it involves including a set of constraints to the problem, or some sort of multidisciplinary optimization process. In many published articles design optimization is deliberately restricted to a single performance objective when, in reality, the design is dependent upon many more. Another feature in the recent design literature is the bias in favour of improving the efficiency of the optimization heuristics, as opposed to making the problems themselves more tractable by improving their parameterization.We have discussed in this thesis the massive impact the parameterization procedure can have on the computational burden of the optimization process, with the parameterization of a full aircraft geometry for an aerodynamic optimization study still being some way away, mainly because the number of parameters would be in the hundreds. Granted, the recent advancements by the surrogate modelling community allow for more efficient global optimization, but design parameterization still has a major role to play in improving optimization efficiency.

We noted in Chapter 3, which is devoted to the parameterization stage of the optimization process, that the Kulfan transformation uses an analytical class function to model the general shape of an object, then uses a parameterized shape function to achieve an accurate model. In Chapter 5 we stated that, for some applications, the class function suggested by Kulfan does not suitably simplify the problem. For example, for the SC(2) airfoils, we showed that the resulting shape function is not significantly simplified by the airfoil class function. This was mainly due to the physical features of the SC(2) airfoils, namely the high gradient of curvature of the leading edge, and the lower surface cusp, being characteristically different to other airfoils. We developed a procedure to generate a new class function using genetic programming, together with a local search of the constants for each member of the population along the evolutionary process, within a Baldwinian learning procedure. By restricting the growth of the members, we found suitably sized class functions specific to a collection of the SC(2) airfoils. These new class functions (one for the upper surface, and one for the lower surface) were then compared to Kulfan's airfoil class function, via deploying them in an optimization procedure, with the results indicating the Kulfan transformation performs significantly better using the newly found class functions. Moreover, the results when using the new class functions were independent of Bernstein polynomial orders above four, indicating that by reducing the number of parameters we have not inadvertently reduced the flexibility of the model.

While most design optimization work to date is underpinned almost exclusively by computational analysis, this should not necessarily be the case. We have shown that physical experimentation can also be integrated into the process. We performed an experiment in the University of Southampton's Large Anechoic Chamber designed to determine the potential of over-wing engines in reducing the perceived noise to ground observers. Forty combinations of streamwise and vertical positions of the engine with respect to the wing were tested, whilst keeping the spanwise position of the engine constant. In this thesis we have presented the results of this study and compared them to simple half-plane noise barrier theory, showing that this analytical theory can be used as a first order approximation to the noise shielding performance of any over-wing engine position. Moreover, by using a surrogate modelling strategy, the optimal position of the engine was found that maximised the noise shielding potential.

We explained in Chapter 7 that the performance of the over-wing configuration cannot be solely quantified by its noise shielding potential, as installing the engine above rather than below the wing has an influence on other performance metrics. With this in mind, we performed a CFD study on the engine positions that were tested in the noise study, extracting the drag at the design c_L, as well as the total pressure values at the fan face. Using the same surrogate modelling strategy as in the noise study, we were able to model the objective function landscapes of the aerodynamic efficiency (the drag at design c_L) and the quality of airflow entering the engine (pressure distortion at the fan face), thus finding the optimal position of the engine in respect to each single performance metric. It was found that the optimal engine position for each performance metric was different, and so a Pareto set of solutions was found using the NSGA II algorithm.

To summarise, the major contributions of this thesis are listed below.

- A strategy to find the design conditions of NASA's SC(2) airfoils has been developed, whilst verifying that the airfoils generated using the re-parameterization method of Sóbester and Powell [2012] do indeed follow the SC(2) design characteristics.
- The time-savings of using the Sóbester and Powell [2012] re-parameterization model to generate an SC(2) airfoil with specific c_l and t/c requirements have been investigated, by performing an optimization procedure.
- A class function generation procedure has been developed for the Kulfan Transformation that improves the parameterization for specific applications.
- An experiment was conducted in the University of Southampton's Large Anechoic Chamber, to determine the potential of the over-wing engine installation in shielding engine inlet noise to people situated at ground level. The results of this were found to agree with simple half-plane barrier theory that can now be used as a first order model for the amount of noise shielding at a specific engine position.
- Using Surrogate models the optimal position of an engine above the wing of a civil aircraft has been found, in terms of:
  - Engine inlet noise shielding to observers situated on the ground;
  - Drag at cruise c_L;
  - Pressure distortion at the fan face, also finding a feasible/infeasible boundary using Support Vector Classifiers;
- A Pareto set of designs for engine placement above an aircraft wing that are non-dominated by either, noise shielding, aerodynamic efficiency, or quality of airflow entering the engine has been found.

8.2 Further Work

8.2.1 A Set of Class Functions for Specific Applications

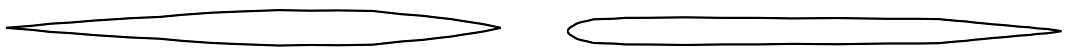
The work presented in Chapter 5 concentrated on generating specific class functions for the SC(2) airfoils following from Kulfan's difficulty in accurately modelling these shapes with the Kulfan transformation. It is highly likely that the Kulfan transformation would have similar difficulties with other aircraft geometries that are characteristically different to a standard airfoil. For example, fan blade cross sections or the stators used to direct the flow between the different fan stages in an engine. Using the method described in Chapter 5 we can find sets of class functions that can be used for the specific application. Of course, we also need to find a set of examples from each application to form a training dataset, which can be a difficult task in itself with many of these geometries being propriety property of the respective companies, and thus few are available for such a study.

8.2.2 New Shape Function Techniques for the Kulfan Transformation

In the Kulfan Transformation shape parameterization technique, the class function can be regarded as the starting design that is to be modified by the shape function. In her introductory paper on this technique, Kulfan [2008] only suggests the use of Bernstein polynomials to be used as the shape function, with there being little restrictions to the method we can use. Granted, the coefficients of the Bernstein polynomials are quickly found using a least squares approach, but the high frequency oscillations reported by Mousavi et al. [2007] restrict the quality at higher dimensions, if we dare perform an optimization with such a large parameter set. In Chapter 3 we stated that Bernstein polynomials, used by Kulfan as the shape function, are also the bases of Bézier curves. Due to their multimodal behaviour at high dimensions B-Spline methods were developed and so a similar process may alleviate the problem here. An approach similar to this one has been considered by Straathof et al. [2008], but no performance comparisons were made. We also presented in Chapter 3 the Free Form Deformation technique that has the ability to change the shapes of existing objects. It would be interesting to assess the performance of FFD in modifying the class function, however this would not be classed to be within the framework of the Kulfan transformation, as FFD would not be applied in the same way as a shape function. Performing a comparison study using different techniques for the shape function, along with the FFD method just discussed, may give further insights into the ability of this method, and we may find an increase in performance.

8.2.3 Increase the Number of Objectives for the Over-Wing Engine Installation Case Study

In Chapter 7 we state that we must consider a variety of design factors when optimizing the over-wing engine position, as optimizing for a single objective (noise shielding, for example) may lead to unacceptable performance in a different discipline. To address this issue we found the aerodynamic performance of the over-wing configuration, generating a Pareto set that comprises of a set of optimal designs in terms of noise shielding, drag, and airflow quality into the engine inlet. However, other major performance metrics exist that need to be accounted for in this design problem: the stability of the aircraft may be influenced by the engine position, and the structural integrity of the support struts in the pylon may have a significant effect. Without studying these performance metrics we cannot fully evaluate the over-wing configuration as a viable option for future aircraft.



(a) From a VFW-614 aicraft (over-wing engine configuration)

(b) From a DLR-F6 aircraft (under-wing engine configuration)

Figure 8.1: Sketches of two pylon cross-section profiles.

8.2.4 A More Detailed Parameterization of the Over-Wing Engine Installation Case Study

To simplify the noise shielding study presented in Chapter 6 of this thesis we only considered two parametric variables; streamwise and vertical position of the engine. This was an acceptable simplification in the noise study, as the shielding performance was unlikely to be affected by the outer curvature of the pylon or nacelle. However, the shape of the pylon and the curvature of the nacelle was found by Yoneta et al. [2010] to be highly influential to the aerodynamic efficiency of the aircraft. Lynch and Intemann [1994] stated that the wing has to be "designed accounting for the presence of the engine/nacelle/pylon installation and the nacelle/pylon is designed accounting for the presence of the wing". Therefore, the SC(2)-0012 airfoil section used as the pylon cross-section, and the CFM56 engine nacelle, must be parameterized and included within the optimization process to determine the full aerodynamic potential.

The difference in the flow conditions above and below the wing is apparent from comparing the respective shapes of the pylon geometry. Figure 8.1(b) shows that a typical underwing pylon has a round leading edge, and tapers from approximately three-quarters of the pylon chord to the tip. Figure 8.1(a), a sketch of the Fokker VFW 614 pylon, is characteristically different with regards to the pointed leading edge and highly curved mid-chord region. If we return to using the Kulfan Transformation, both of these geometries can be considered in an optimization process by including the first class function exponent $N1$ as a parameter. If we decide to include the structural factors into this optimization procedure, as discussed in Section 8.2.3, a closely coupled system between the aerodynamic and structural modules would have to be achieved as the pylon outer surface must incorporate the pylon struts.

Appendix A

CFD Replication of Results from the ATAT Program

In what follows we discuss CFD experiments aimed at replicating the results obtained by Jenkins and Hill [1988] on the SC(2)-0714 airfoil. The purpose of the work discussed here was to ensure we had an accurate process to be able to replicate the design conditions and thus determine the 'real' characteristics of each NASA supercritical airfoil. However, we first had to modify the results obtained from Jenkins and Hill [1988] of the SC(2)-0714 airfoil so that they could be compared with the results of the CFD simulations.

The NASA SC(2)-0714 airfoil was tested by Jenkins and Hill [1988] at a variety of Mach numbers , Reynolds numbers and angles of attack. Only one data set was taken here for comparison: Reynolds number and Mach number remained constant at 40 million and 0.6 respectively, and angle of attack was varied between -1.98 and 5 degrees. A problem with using the data from Jenkins and Hill [1988] was that no correction of the data was made for three-dimensional (3D) or blockage effects in the windtunnel. Blockage correction tables in Jenkins and Adcock [1986] suggest that the effective Mach number for an initial flow at M=0.6 is M=0.587, and that a correction factor of 1.015 must be multiplied to the force and moment coefficients. However the 3D flow effects in the windtunnel also demand that an upwash correction is applied to the angle of attack, which was not available. Barnwell [1978] gives an angle of attack correction procedure where the amount of upwash correction is a function of the lift coefficient:

$$\Delta\alpha = \frac{-c_l c}{8(1+j)h} \times \frac{180}{\pi} \tag{A.1}$$

where c is the chord (6 in.), h is the tunnel's semi-height (12 in.), and $j = aK/h$ (a is the slot spacing (4 in.) and K is the semiempirical constant (3.2)). Note that Equation A.1 is only valid for the 0.3m TCT tunnel. After integrating the experimental pressure coefficient graphs from Jenkins and Hill [1988] to find the lift coefficient values a table of the new effective attack angles was produced (See Table A.1).

α	Corrected α
-1.98	-2.4724
-0.96	-1.6742
0.01	-0.9168
0.50	-0.5334
1.02	-0.1307
1.54	0.2780
2.00	0.65221
2.49	1.0217
2.99	1.4354
3.98	2.2034
5.00	2.9994

TABLE A.1: Corrected angles of attack, α due to upwash.

A.1 Method

Using the procedure developed by Kulfan [2008] a geometric approximation was generated of the SC(2)-0714 airfoil using a Bernstein Polynomial order of eight. The polynomial coefficients were found using the least squares method, with the resulting profile being accurate to within manufacturing tolerances. The mesh was generated in GAMBIT, which was then imported into Fluent to solve the RANS equations. This process was automated, allowing us to perform multiple CFD computations with varying angles of attack without any input from the user.

A.1.1 Pre-Processing

We used an unstructured mesh to discretize the external region of the airfoil out to the farfield. We determined the thickness of the boundary layer ,T, using thin-plate theory [Schlichting, 1979];

$$T = 0.37x(\frac{Ux}{\nu})^{-0.2} \tag{A.2}$$

where U is the freestream velocity, ν is the kinematic viscosity, and x is the distance from the leading edge. The first cell size within the boundary layer was taken to be

$$y^1 = \frac{y_1^+ \nu}{u_0}, \tag{A.3}$$

where

$$\frac{u_0}{U} = 0.0296(\frac{Ux}{\nu})^{-0.2}. \tag{A.4}$$

A near wall model was used in the CFD formulation and so $y_1^+ \approx 1$. Note that Equation A.3 only gave an approximation of the y_1 required, thus the value was modified within numerous CFD computations until the appropriate y^+ value was achieved.

A "C" mesh around the airfoil was generated in GAMBIT, with the farfield extended 12.5m upstream, 20m downstream and 12.5m above and below the airfoil. An unstructured tetrahedral mesh was used (See Figure A.1), as it allowed automated regeneration, an essential requirement for it to be used in an optimization procedure.

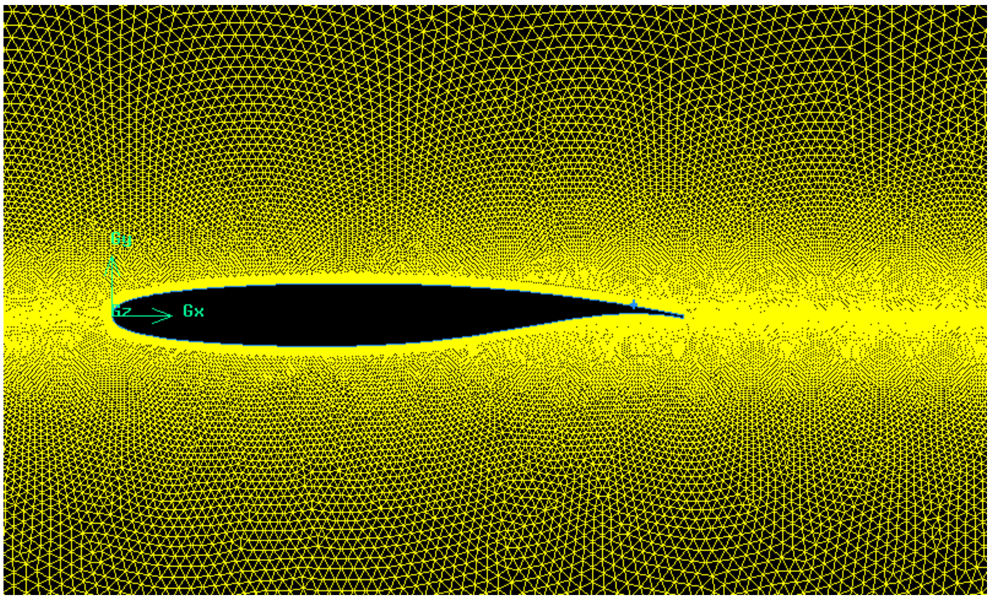


FIGURE A.1: Near field view of the airfoil mesh.

A.1.2 Solver

In Fluent we used the density based implicit solver with the $k-\epsilon$ turbulence model (initial results showed insignificant difference in pressure coefficient values when using different turbulence models). Enhanced wall treatment was selected with pressure gradient wall functions and the energy equation was selected to be on. The medium was set to be air with the density given to be an ideal gas and viscosity set to the Sutherland algorithm[Fluent, 2011], defined as

$$\frac{\mu}{\mu_0} = \left(\frac{T}{T_0}\right)^{3/2} \frac{T_0 + 110}{T + 110}. \tag{A.5}$$

To find the pressure at the farfield we used the Mach number

$$M = \frac{U}{k} = \frac{U}{(\gamma RT)^{1/2}} \tag{A.6}$$

to find the velocity, U_∞ and then the Reynolds number

$$Re = \frac{\rho U_\infty l}{\mu} \tag{A.7}$$

to find the density, ρ, with the viscosity μ found using Equation A.5. We then used the ideal gas law to find the pressure, P;

$$P = \rho RT. \tag{A.8}$$

This value of pressure was set in the boundary conditions for the pressure farfield along with the other initial conditions. Turbulent intensity was set to 4%, turbulent viscosity ratio to 10, and the temperature to 300K. The operating pressure was set to zero. Initial Courant number was set to 5, which would be increased to 20 during the simulation. All discretizations were set to second order with an under-relaxation turbulent kinetic energy of 0.8. During the solves the mesh would be adapted (refined) using the adaptation capability in Fluent.

A.2 Discussion

Figure A.4 compares our CFD pressure coefficient results with experimental data found in Jenkins and Hill [1988] at different angles of attack for the SC(2)-0714 Airfoil. Figure A.2 compares the lift values obtained from the pressure coefficient plots in Figure A.4. At angles between -0.5^o and 1.5^o there is good correlation with the experimental results. At low angles the CFD solution tends to predict an initial peak forward of that of the experimental data. The peak is created on the lower surface of the airfoil, which can be seen in Figure A.3 on the lower surface, near the leading edge. At larger angles of attack, where a shock develops on the top of the airfoil, the CFD simulation finds it difficult to accurately predict the position of the shock compared to the experimental results.

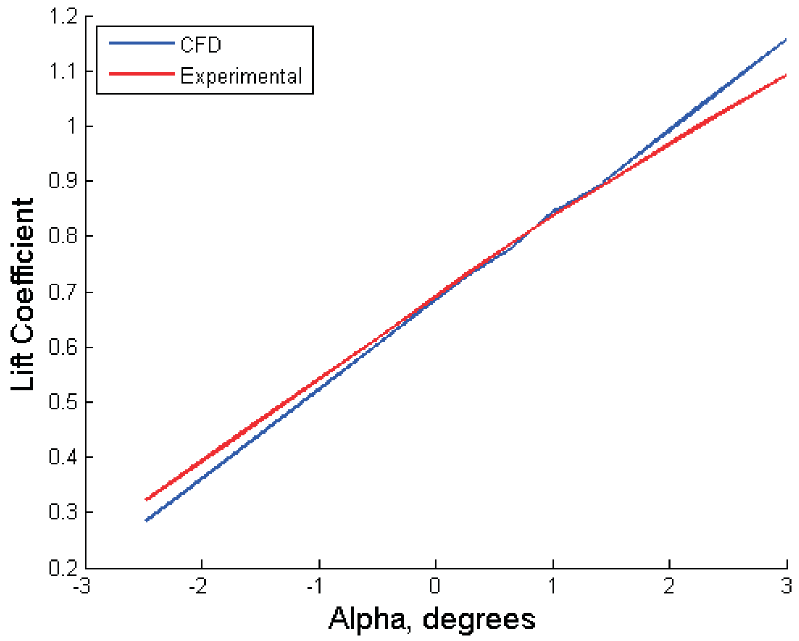


FIGURE A.2: Comparison between CFD and experimental c_l values with different angles of attack, α.

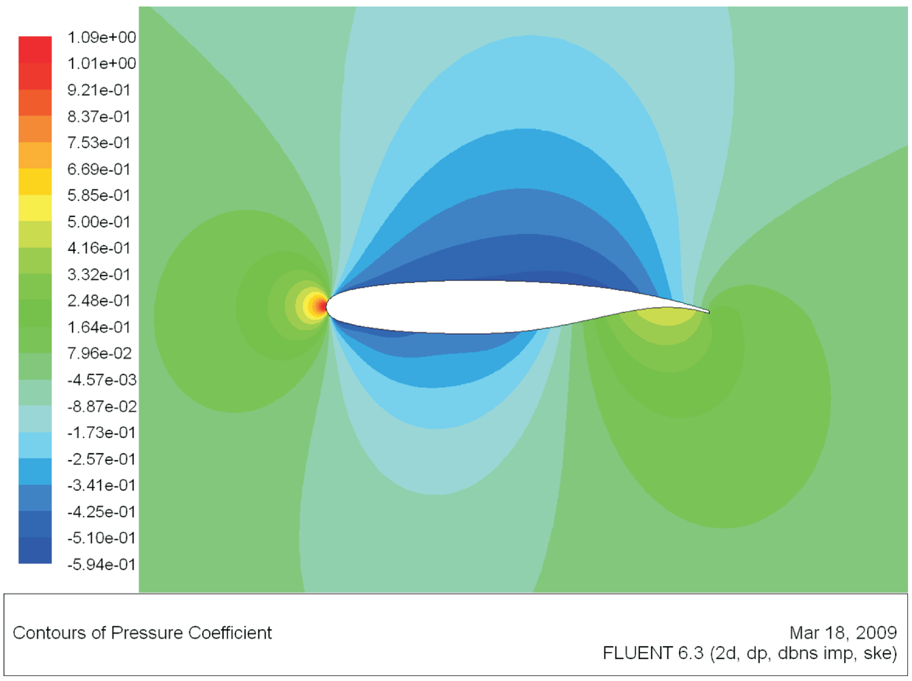
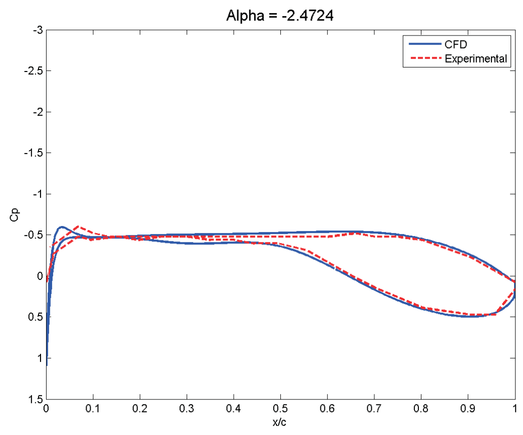
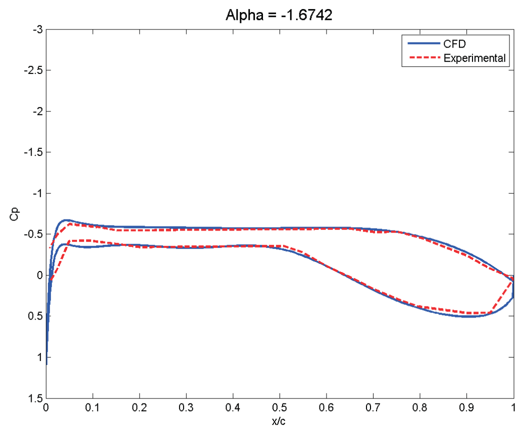
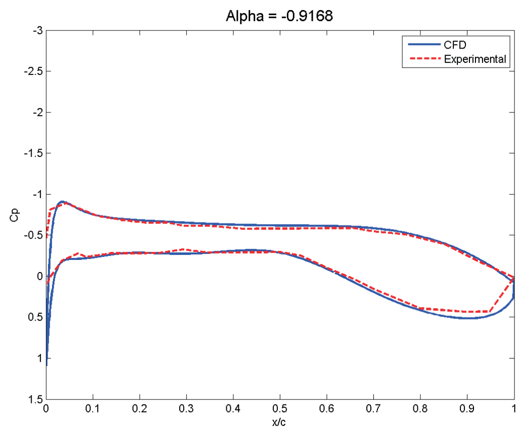
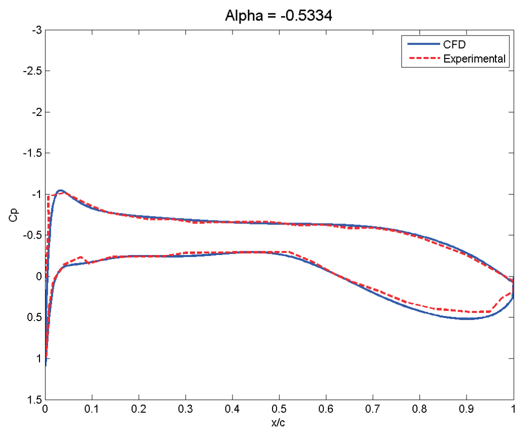


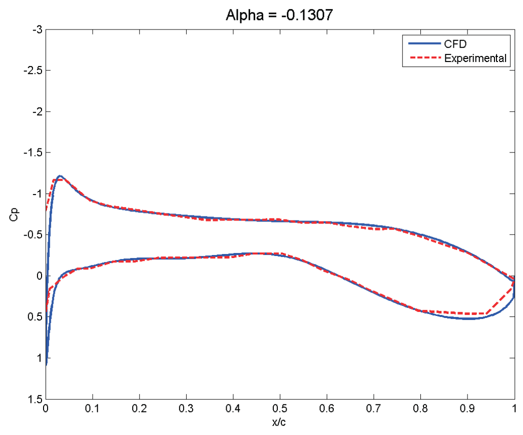
FIGURE A.3: Contours of pressure coefficient around an SC(2)-0714 airfoil at $\alpha = -2.4724^{\circ}$.

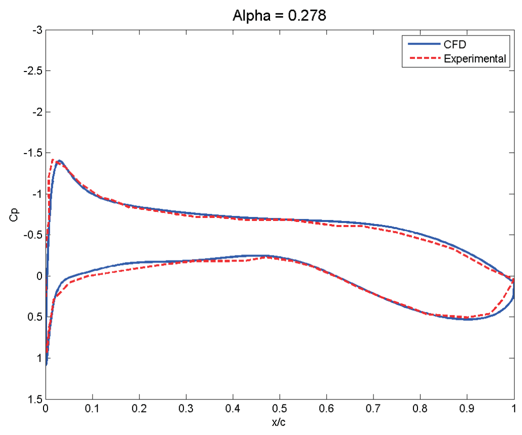

Alpha = -2.4724
CFD
Experimental
Cp
x/c


Alpha = -1.6742
CFD
Experimental
Cp
x/c


Alpha = -0.9168
CFD
Experimental
Cp
x/c


Alpha = -0.5334
CFD
Experimental
Cp
x/c


Alpha = -0.1307
CFD
Experimental
Cp
x/c


Alpha = 0.278
CFD
Experimental
Cp
x/c

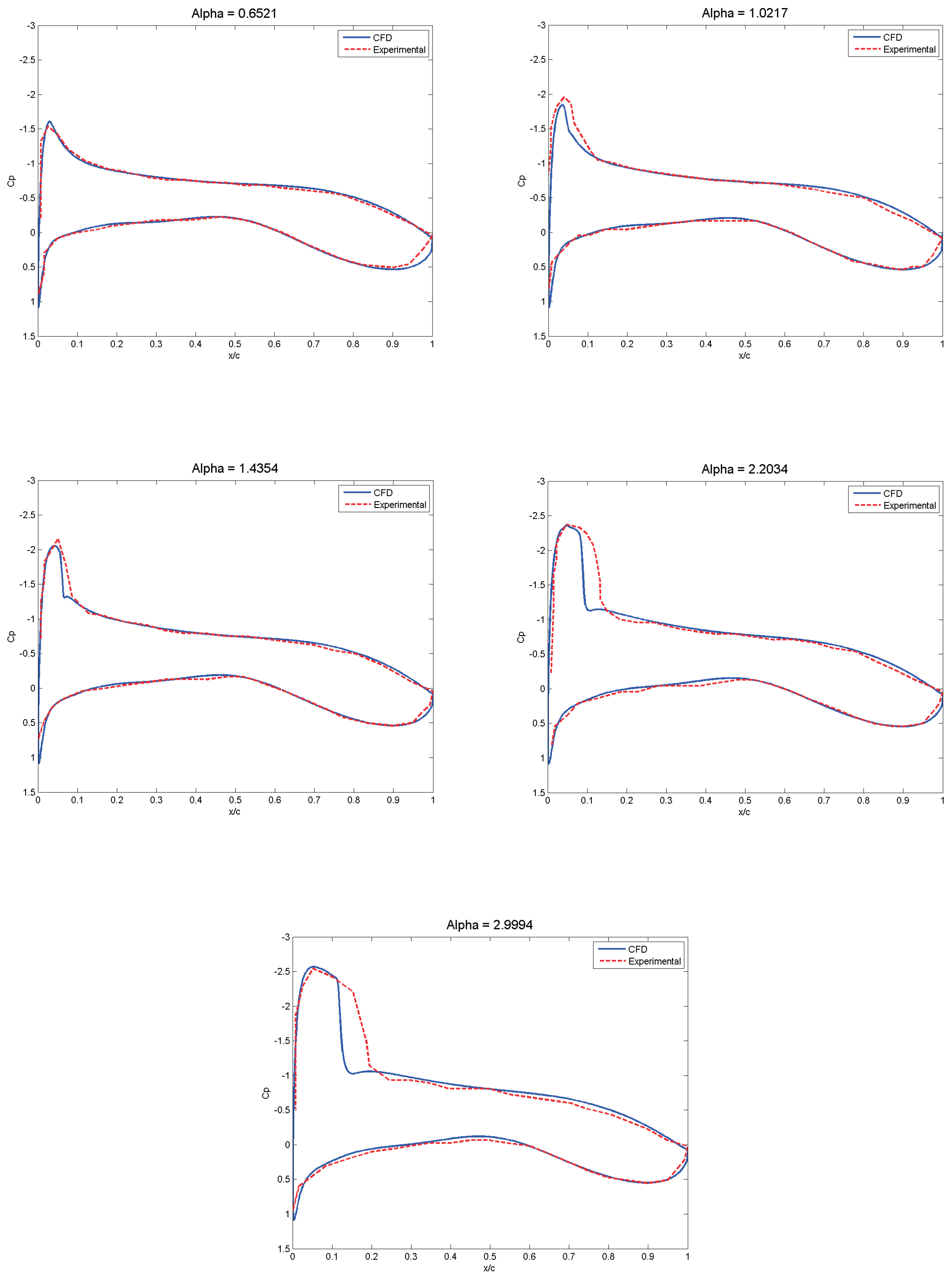


FIGURE A.4: Pressure coefficient plots of CFD and experimental investigations at various angles of attack, α.

Appendix B

Overwing Engine Installation – The CFD Setup

In Chapter 7 we presented the results of a CFD study where the engine was positioned in different places above the wing to find the optimal position in terms of aerodynamic efficiency and engine airflow quality. This chapter explains the CFD process of that study, from the creation of the geometry through to the exportation of the results. The process is validated against the results gathered from the third AIAA drag prediction workshop.

B.1 Geometry Generation

As noted in Chapter 7, the DLR F6 aircraft model, presented in Rossow and Ronzheimer [1992], and Brodersen and Sturmer [2001], was the aircraft under investigation in this study. The geometry used was obtained from the second American Institute of Aeronautics and Astronautics (AIAA) drag prediction workshop website [Frink, 2009a], which had nacelles typical to enclose a CFM56 type engine. Included in the model is the FX2B fairing around the connection between the wing and the fuselage, the geometry of which was obtained from the third AIAA drag prediction workshop website [Frink, 2009b]. These parts never alter between the different designs. Parts that do change between each design are the nacelle and pylon. Note that the change in nacelle is a simple translation above the wing and is therefore generated first. The pylon however must also be modified to maintain the attachment between the wing and the nacelle.

The nacelle and pylon geometries were created in MATLAB. This allowed quick and automated generation of each design. The code ensured that the pylon geometry sat flush against both the wing and the nacelle. This was achieved by first positioning the SC(2)0012 airfoil, used as the pylon cross section, so that its leading edge point sat on the wing surface at a specified position. The cross-section was then extruded to where the leading edge of the pylon meets the specified position on the nacelle surface. The cross-sections of the wing and nacelle at these positions were used to give a rough approximation of the curvature needed on the pylon at the respective connections. Note that the programme Harpoon [2011] was used to create the mesh of each of the designs. This required the parts to be exported as stereoligraphy files[1], which uses triangular facets to describe each part. To ensure a flush join between each part, the pylon points that connect to the wing/nacelle surface must lie within and be planar to the respective facet on the joining surface. This was achieved by importing the stereoligraphy file of the upper surface of the wing into MATLAB, and by assuming the pylon was linearly swept so that the respective joining facet on the surface of the wing/nacelle could be identified for each pylon connecting point. The quality of the meshes generated by Harpoon is very dependent on the facets that lie on the surface, especially at the connections between the different parts. In an effort to improve mesh quality the nodes of the pylon that join to the respective surface (the wing or the nacelle) were included in the stereoligraphy file of that surface, by using Delaunay triangulation to re-create all of the facets.

B.2 Pre-Processing

As mentioned, the mesh of each design was created using Harpoon [2011]. This creates fully automated hex-dominated meshes. A number of refinement regions were inserted around the geometry and in the wake. These refinement regions were parameterized with respect to the position of the engine and pylon. To create these regions, as well as to import the unique parts for each design, the configuration files that contain the commands to create the meshes were automatically generated from within MATLAB when the geometry was also created. The refinement regions, as well as the resulting mesh, of one of the designs can be seen in Figure B.1. The bounding box enclosed a region that spanned 8m from the centre of the aircraft model in all directions, excluding the direction normal to the symmetry plane. The initial boundary layer was defined using Equation A.3, and was then modified to achieve a y-plus between 30 and 100. The resulting meshes had between 7.8-9.3 million elements.

B.3 Engine Calculations

We created a crude turbofan model in the CFD package by using pressure outlet and inlet boundary conditions for the fan inlet and low speed, low temperature jet respectively. We did not include the high temperature core jet. We used the ideal parametric cycle

[1] The student version of the meshing programme Harpoon can only import one type of generic file – Stereoligraphy files.

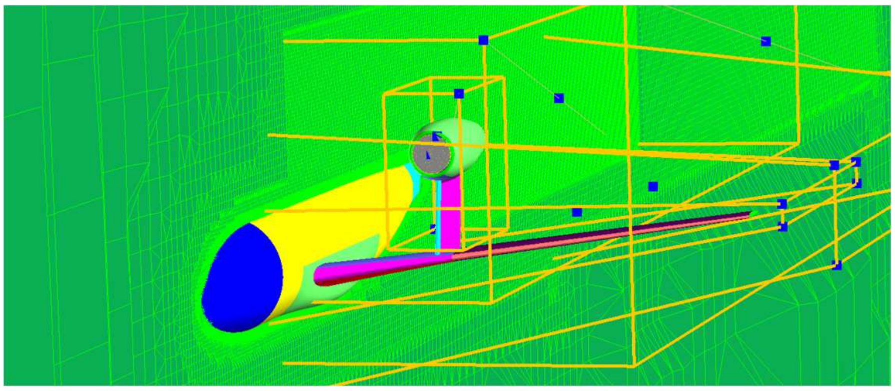


FIGURE B.1: An image of a cut through of the mesh created for when the nacelle is positioned at $x = 0$, $z = 2.5$.

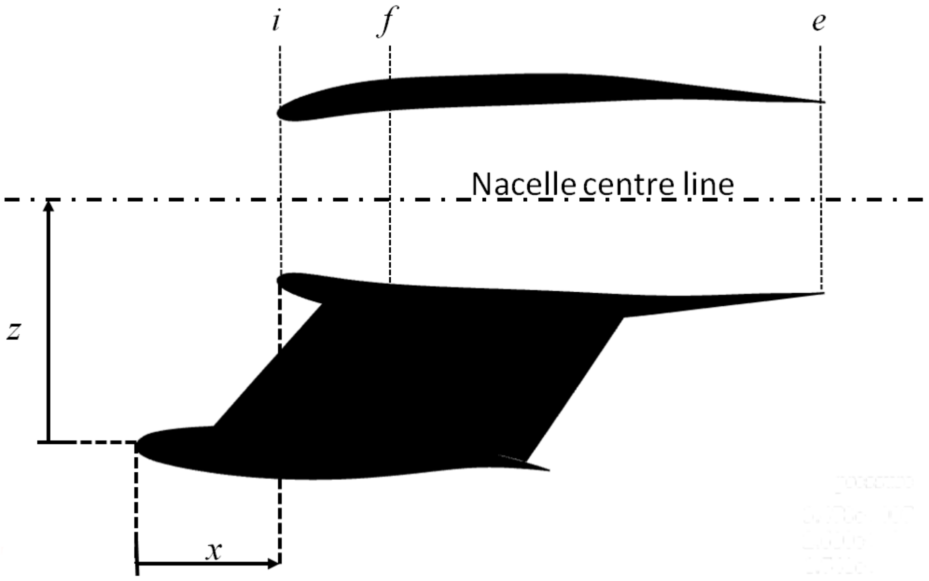


FIGURE B.2: Nacelle cross-section showing design parameters **x** and **z** and the station numbers i(inlet), f(fan) and e(exit)

analysis of a turbofan given in Mattingly [1996] along with the isentropic flow equations and perfect gas law to find the initial flow conditions at the fan inlet and low temperature jet. Table B.1 lists the characteristics of a CFM-56 type engine that we used for our calculations. The mass flow rate was then modified to ensure a capture area ϵ of 0.75 enters the engine:

$$\epsilon = \frac{A_\infty}{A_f} \tag{B.1}$$

A_∞ is the area of the streamtube entering the engine at a position far upstream of the engine, and A_f is the fan area.

B.3.1 Exhaust Conditions

From standard atmospheric tables the temperature, T, at a cruise altitude of 35,000 ft is 218 K. Our cruise velocity, U_∞, at a Mach number of 0.75 is therefore

$$U_\infty = M\sqrt{\gamma RT} = 221.97 \tag{B.2}$$

where γ is the ratio of specific heats (equals 1.7 for air) and R is the universal gas constant (equals 287 for air). Using the Sutherland law from Appendix A the viscosity is found to be $1.43{\times}10^-5$. We can now find the density ρ via the Reynolds number and characteristic length $L = 0.1412$

$$\rho = \frac{\mathrm{Re}\ \mu}{U_\infty L} = 1.3682 \tag{B.3}$$

and thus the pressure, p, from the ideal gas law

$$p = \rho RT = 85600.4 \tag{B.4}$$

If we assume that the total pressure entering the engine is equal to the total pressure at the fan then

$$P_{t_i} = P_{t_f} = P_\infty \left(1 + \frac{\gamma - 1}{2} M_\infty^2\right)^{\gamma/(\gamma-1)} = 124289 \tag{B.5}$$

The fan pressure ratio, π_f is used to calculate the total pressure at the fan exit:

$$\pi_f = \frac{P_{t_e}}{P_{t_f}} = 1.7 \tag{B.6}$$

Therefore, $P_{t_e} = 1.7 * 124289 = 211291.3$, and

$$P_e = P_{t_e} \left(1 + \frac{\gamma - 1}{2} M_e^2\right)^{-\gamma/(\gamma-1)} = 114238.5571 \tag{B.7}$$

with the exit Mach number assumed to be $M_e = 0.98$. The total temperature can be found using the isentropic relation

$$\left(\frac{P_e}{P_{t_e}}\right)^{\frac{\gamma-1}{\gamma}} = \frac{T_e}{T_{t_e}} = 0.83887 \tag{B.8}$$

and assuming that for an ideal engine $T_e = T_\infty = 218$;

$$T_{t_e} = T_e/0.83887 = 259.87. \tag{B.9}$$

Full Scale		ModelScale	
Characteristic	Value	Characteristic	Value
$\dot{m}_r$	310 m/s		
Fan Diameter	1.549 m	Fan Diameter	0.0542 m
Fan Pressure Ratio	1.7		
Cruise altitude	35,000 ft		
ρ_r	0.38 kg/m^3	ρ	1.3682 kg/m^3
T_r	218 K	T	218 K
M_r	0.8	M	0.75
		Re	3×10^6

TABLE B.1: Engine information taken from Jackson [2009], Niskode et al. [2011], and CFM [2011] for the CFM56-7B20 turbofan engine with its cruise conditions, together with information for our computational model

B.3.2 Inlet conditions

We have found cruise and off-design conditions for the GE-90 engine[Cantwell, 2011], whose mass flow rate changes from 1350kg/s @ take-off to 576 kg/s @ cruise. Using this ratio we obtain a cruise mass flow of $\dot{m} * 576/1350 = 132.267$kg/s. We can therefore find the velocity and thus the Mach number entering the engine:

$$U_{r_f} = \frac{\dot{m}_r}{\rho_r A_{r_f}} = \frac{132.267}{0.38 * \pi(1.549/2)^2} = 184.7\text{m/s} \tag{B.10}$$

$$M_{r_f} = \frac{U_{r_f}}{a_r} = \frac{184.7}{\sqrt{1.4 * 287 * 218}} = 0.6227 \tag{B.11}$$

Note that these are approximations as we have used the air properties in the freestream and not those at the fan face. If we use the same temperature as we have previously ($T = 218$k) then we can assume $U_{r_f} = U_f$ to determine the mass flow in our scaled engine, again using the freestream fluid properties:

$$\dot{m} = \rho_\infty A_f U_f = 1.3682 * \pi * \left(\frac{0.054}{2}\right)^2 * 184.7 = 0.5825\text{kg/s} \tag{B.12}$$

Assuming total pressure is the same as the freestream then $M_{r_f} = M_f$ and

$$P_f = P_{t_\infty}\left(1 + \frac{\gamma - 1}{2} M_f^2\right)^{-\gamma/(\gamma-1)} = 95697.4 \tag{B.13}$$

We can find the temperature by using the compressible equation relations;

$$\left(\frac{P_f}{P_{t_f}}\right)^{\frac{\gamma-1}{\gamma}} = \frac{T_f}{T_{t_f}} = 0.928 \tag{B.14}$$

Therefore $T = 0.928 * 242.4 = 224.95$ for the fan inlet. The mass flow rate specified above is modified to give capture area ϵ of 0.75. For an isolated engine without any aircraft geometry present a CFD simulation achieved this for a mass flow rate of 0.4375kg/s. This value was then used in all subsequent analyses.

B.4 Solver Setup

FLUENT was used to solve the Reynolds-averaged Navier-Stokes equations. As mentioned in Chapter 7, the flight conditions were set to a Mach number of 0.75 and a Reynolds number of 3 million (calculated using the mean aerodynamic chord of 0.1412m). The density based implicit solver was selected in FLUENT. The Spalart-Allmaras turbulence model was chosen, as results gathered in the third AIAA drag prediction workshop [Vassberg et al., 2007] show it to give reasonably accurate results for this case, and it is quicker to solve than the other popular two equation models. Farfield boundary conditions were assigned to the bounding box. The engine was modelled using a pressure outlet boundary condition with an associated mass flow rate for the inlet, with the engine exhaust modelled using a pressure inlet (See Section B.3 for the conditions at these boundaries). The Courant number was initially set to two, and the solver equations set to first order. During the simulation the equations were changed to second order, with the Courant number altered to five. The angle of attack was also changed during the simulation, to achieve the desired C_L of 0.5. The simulation was run until the average pressure at the fan face converged to within 1×10^{-4}.

B.5 Validation of CFD Setup

The CFD process detailed above was validated against the results obtained from the third AIAA drag prediction workshop [Vassberg et al., 2007], where the FX2B fairing was used at a fixed C_L of 0.5. The nacelle was not included in this study. The results using the above procedure gave a C_D value of 0.02719 at a C_L of 0.5. This is within the range of the results obtained from Vassberg et al. [2007] ($C_D = 0.02527 - 0.02727$), but higher than the average value of 0.02636. To further validate the setup we can compare pressure contours of the wing surface. Figure B.3 shows similar contours to those shown in Vassberg et al. [2007] of the pressure coefficient on the wing surface.

B.6 Summary

The CFD process described here has been validated against the results gathered from the third AIAA drag prediction workshop. The results from the test indicate that the setup is suitable and can be used for the over-wing design study detailed in Chapter 7.

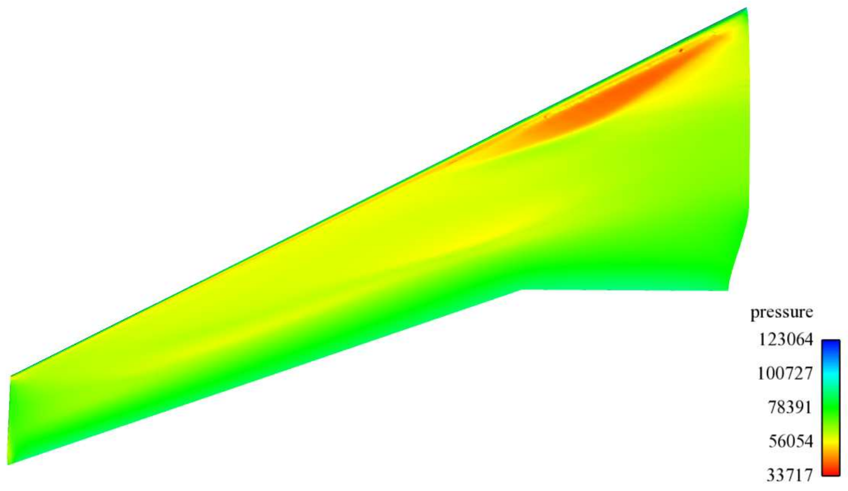


Figure B.3: Pressure contours, in Pascals, on the wing of a ‘clean’ configuration used in the validation case study

Bibliography

I.H. Abbott and A.E. von Doenhoff. *Theory of Wing Sections.* Dover Publications, 1959.

ACARE2020. European Aeronautics: A Vision for 2020,. Advisory Council for Aeronautics Research in Europe, 2001.

A. Agarwal and A.P. Dowling. Low-Frequency Acoustic Shielding by the Silent Aircraft Airframe. *AIAA Journal*, 45(2):358–365, 2007.

A. Agarwal, A.P. Dowling, H-C. Shin, and W. Graham. Ray-Tracing Approach to Calculate Acoustic Shielding by a Flying Wing Airframe. *AIAA Journal*, 45(5):1080–1090, 2007.

F.M. Alam, K.R. McNaught, and T.J. Ringrose. A comparison of experimental designs in the development of a neural network simulation metamodel. *Simulation Modelling Practice and Theory*, 12:559–578, 2004.

L. Alvarez, V. Toropov, D. Hughes, and A. Ashour. Approximation Model Building using Genetic Programming Methodology: Applications. *AIAA-200101628*, 2001.

J.D. Anderson. *Fundamentals of Aerodynamics.* McGraw Hill, 3rd edition, 2001.

J.M. Baldwin. A new factor in evolution. *American Naturalist*, 30:441–451, 1896.

R. W. Barnwell. Design and Performance Evaluation of Slotted Walls for Two-Dimensional Wind Tunnels. Technical Report NASA TM-78648, 1978.

A. H. Barr. Global and Local Deformations of Solid Primitives. *SIGGRAPH Comput. Graph.*, 18(3):21–30, 1984. ISSN 0097-8930. doi: http://doi.acm.org/10.1145/964965.808573.

F. Bauer, P. Garabedian, and Korn D. *Supercritical Wing Sections.* Springer-Verlag, lecture Notes in Economics and Mathematical Systems edition, 1972.

F. Bauer, P. Garabedian, and Korn D. *Supercritical Wing Sections II.* Springer-Verlag, lecture Notes in Economics and Mathematical Systems edition, 1975.

D.L Berry. The Boeing 777 Engine/Aircraft Integration Aerodynamic Design Process. *ICAS-94-6.4.4*, pages 1305–1320, 1994.

J. J. Berton, E. Envia, and C. L Burley. An Analytical Assessment of NASA's N1 Subsonic Fixed Wing Project Noise Goal. *15th AIAA/CEAS Aeroacoustics Conference*, 11-13 May 2009.

J.J. Berton. Noise Reduction Potential of Large, Over-the-Wing Mounted, Advanced Turbofan Engines. Technical Report TM-2000-210025, NASA, 2000.

D.A. Bies and C.H. Hansen. *Engineering Noise Control.* Unwin Hyman, 1988.

D. Bogue and N. Crist. CST Transonic Optimization Using Tranair++. *AIAA 2008-321*, 2008.

O. Brodersen and A Sturmer. Drag prediction of engineairframe interference effects using unstructured navierstokes calculations. *AIAA-2001-2414*, 2001.

B. J. Cantwell. The GE90 - An Introduction. Stanford course material `http://www.stanford.edu/~cantwell/AA283_Course_Material/GE90_Engine_Data.pdf`, Accessed Aug 2011.

P. Castonguay and S. Nadarajah. Effect of Shape Parameterization on Aerodynamic Shape Optimization. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, number AIAA-2007-59, Reno, Nevada, Jan 8-11 2007.

L.A. Catalano, A. Dandone, and V.S.E Daloiso. On the use of Orthogonal Functions in Fluid-Dynamic Gradient-Based Optimization. *International Journal for Numerical Methods in Fluids*, 56:1175–1184, 2008.

M. Ceze, M. Hayashi, and E. Volpe. A Study of the CST Parameterization Characteristics. *AIAA 2009-3767*, 2009.

CFM. In the wings with the CFM56 engines. CFM56 Technical Data, `http://www.cfm56.com/pdf/cfm-technical-data.pdf`, Accessed March 2011.

I.C. Chang, F.J. Torres, and C. Tung. Geometric Analysis of Wing Sections. *NASA TM 110346*, 1995.

J.A. Cottrell, T.J. Hughes, and Y. Bazilevs. *Isogeometric Analysis. Toward Integration of CAD and FEA.* Wiley, 2009.

D. Crichton, E. Blanco, and T. R. Law. Design and Operation for Ultra Low Noise Take-Off. *AIAA 2007-456*, 45th AIAA Aerospace Sciences Meeting and Exhibit, 8-11 January, Reno, Navada 2007.

N Cristianni. *An introduction to Support Vector Machines: and other kernel-based learning methods.* Cambridge University Press, 2000.

K. Deb, A. Pratap, S. Agarwal, and T.; Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197, 2002.

I. Dempsey, M. O'Neil, and A. Brabazon. Constant Creation in Grammatical Evolution. *International Journal of Innovative Computing and Applications*, 1(1):23–38, 2007.

H. Du and H. Qin. Free-Form Geometric Modeling by Integrating Parametric and Implicit Pdes. *IEEE Transactions on Visualization and Computer Graphics*, 13(3): 549–561, 2007.

ESDU96028. VGK Method for Two-Dimensional Aerofoil Sections. Data Unit 96028, ESDU, 2004.

Fluent. *Fluent User's guide.* Fluent Inc., 2011.

A. Forrester, A. Sobester, and A Keane. *Engineering Design via Surrogate Modelling.* Wiley, 2008.

D. R. Forsey and R.H. Bartels. Hierarchical B-Spline Refinement. *Computer Graphics*, 22(4), 1988.

N.T. Frink. The Second AIAA Drag Prediction Workshop Website. `http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/Workshop2/workshop2.html`, Accessed Oct 2009a.

N.T. Frink. The Third AIAA Drag Prediction Workshop Website. `http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/Workshop3/`, Accessed Oct 2009b.

M Fujino and Y. Kawamura. Wave-Drag Characteristics of an Over-the-Wing Nacelle Business-Jet Configuration. *Journal of Aircraft*, 40:1177–1184, 2003.

D.E Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, 1989.

J.F. Groenweg, T.G. Sofrin, E.J. Rice, and P.R. Gliebe. *Aeroacoustics of Flight Vehicles: Noise Sources*, volume 1, chapter Turbomachinery Noise. Acoustical Society of America, 1995.

S. Gunn. Support vector machines for classification and regression. Technical report, School of Electronics and Computer Science, University of Southampton, 1998.

Harpoon. *Harpoon Manual.* Sharc, 2011.

C. D Harris. Aerodynamic Characteristics of the 10-percent-thick NASA Supercritical Airfoil 33 Designed for a Normal-Force Coefficient of 0.7. Technical report, NASA TM X-72711, 1975a.

C. D Harris. Aerodynamic Characteristics of a 14-percent-thick NASA Supercritical Airfoil Designed for a Normal-Force Coefficient of 0.7. Technical report, NASA TM X-72712, 1975b.

C. D. Harris. NASA Supercritical Airfoils – A Matrix of Family-Related Airfoils. Technical report, NASA TP-2969, 1990.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2 edition, 2009.

R. M. Hicks and P. A Henne. Wing Design by Numerical Optimization. *Journal of Aircraft*, 15(7):407–412, July 1978.

J.I. Hileman, Z.S. Spakovszky, and M. Drela. Aerodynamic and Aeroacoustic Three-Dimensional Design for a Silent Aircraft. *AIAA 2006-241*.

G. A. Hill, O.A Kandil, and A.S. Hahn. Aerodynamic Investigations of an Advanced Over-the-Wing Nacelle Transport Aircraft Configuration. *Journal of Aircraft*, 46(1): 25–35, 2009.

R. Hooke and T. A. Jeeves. "Direct Search" Solution of Numerical and Statistical Problems. 1960.

C.R. Houck, J.A. Joines, and M.G. Kay. Utilizing Lamarckian Evolution and the Baldwin effect in hybrid genetic algorithms. 1996.

Isight and SIMULIA. Simulia website. `http://www.simulia.com`, Accessed Aug 2011.

P. Jackson, editor. *All the World's Aircraft 2009-2010.* Jane's Information Group, 2009.

R. V. Jenkins and J. B. Adcock. Tables for Correcting Airfoil Data Obtained in the Langley 0.3-Meter Transonic Cryogenic Tunnel for Sidewall Boundary-Layer Effects. Technical report, NASA TM-87723, 1986.

R.V. Jenkins. NASA SC(2)-0714 Airfoil Data Corrected for Sidewall Boundary-Layer Effects in the Langley 0.3-Meter Transonic Cryogenic Tunnel. Technical report, NASA TP-2890, 1989.

R.V. Jenkins and A.S. Hill. Aerodynamic Performance and Pressure Distributions for a NASA SC(2)-0714 Airfoil Tested in the Langley 0.3-Meter Transonic Cryogenic Wind Tunnel. Technical report, NASA TM-4044, 1988.

P Jordan. The VFW 614 Fleet. `http://www.vfw614.de/flotte_e.html`, Accessed Mar 2010.

P. Joseph and C.L. Morfey. Multimode Radiation from an Unflanged Semi-Infinite Circular Duct. *Journal of the Acoustical Society of America*, 105(5):2590–2600, 1999.

A.J. Keane and P.B. Nair. *Computational Approaches for Aerospace Design - The Pursuit of Excellence.* John Wiley and Sons, 2005.

S Kim, J. Alonso, and A. Jameson. Two-Dimensional High-Lift Aerodynamic Optimization using the Continuous Adjoint Method. *AIAA 20004741*, 2000.

D.J. Kinney and P.A. Hahn, A.S. Gelhausen. Comparison of Low and High Nacelle Subsonic Transport Configurations. *AIAA-97-2318*, 1997.

S Kodiyalam. Evaluation of Methods for Multidisciplinary Design Optimization (MDO), Part I. *NASA/CR-1998-208716 Evaluation*, 1998.

S Kodiyalam and C Yuan. Evaluation of Methods for Multidisciplinary Design Optimization (MDO), Part II. *NASA / CR-2000-210313*, 2000.

M. Koyasu and M. Yamashita. Scale Model Experiments on Noise Reduction by Acoustic Barrier of a Straight Line Source. *Applied Acoustics*, 6(3):233–242, 1973.

J Koza. *Genetic Programming- On the Programming of Computers by Means of Natural Selection*. Cambridge, MA:MIT Press, 1992.

B. M. Kulfan. Universal Parametric Geometry Representation Method. *Journal of Aircraft*, 45(1):142–158, 2008. DOI: 10.2514/1.29958.

B. M. Kulfan and J.E. Bussoletti. Fundamental Parametric Geometry Representations for Aircraft Component Shapes. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2006.

C. L. Ladson and E. J. Ray. Status of Advanced Airfoil Tests in the Langley 0.3 meter Transonic Cryogenic Tunnel. In *Advanced Aerodynamics- Selected NASA Research*, pages 37–59. NASA CP 2208, 1981.

K.R Laflin, S.M. Klausmeyer, T. Zickuhr, J.C. Vassberg, R.A. Wahls, J.H. Morrison, O.P. Broderson, M.E. Rakowitz, E.N. Tinoco, and J. Godard. Data Summary from the Second AIAA Computational Fluid Dynamics Drag Prediction Workshop. *Journal of Aircraft*, 42(5):1165–1178, 2005.

H. J. Lamousin and W. N. Waggenspack Jr. NURBS-based Free-Form Deformations. *IEEE Comput. Graph. Appl.*, 14(6):59–65, 1994. ISSN 0272-1716. doi: http://dx.doi.org/10.1109/38.329096.

K. Lane and D. Marshall. A Surface Parameterization Method for Airfoil Optimization and High Lift 2D Geometries Utilizing the CST Method. *AIAA 2009-1461*, 2009.

L. Leavitt and S. Smith. NASA's ERA Vehicle Systems Integration Sub-Project. *Royal Aeronautical Society Aerodynamics Conference*, 2010.

F.T. Lynch and G.A. Intemann. The Modern Role of CFD in Addressing Airframe/Engine Integration Issues for Subsonic Transports. *ICAS-94-6.4.3*, 1994.

Z. Maekawa. Noise Reduction by Screens. *Applied Acoustics*, 1:157–173, 1968.

W.H. Mason. Configuration Aerodynamics. *Lecture Notes from Virginia Tech University*, 2009.

J.D. Mattingly. *Elements of Gas Turbine Propulsion.* McGraw Hill, 1996.

Michael D. McKay. Latin hypercube sampling as a tool in uncertainty analysis of computer models. In *Proceedings of the 24th conference on Winter simulation*, WSC '92, pages 557–564, New York, NY, USA, 1992. ACM.

T.M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

M.D. Morris and T.J. Mitchell. Exploratory Designs for Computational Experiments. *Journal of Statistical Planning and Inference*, 43:381–402, 1995.

M.E. Mortenson. *Geometric Modelling.* Industrial Press, 2006.

A. Mousavi, P. Castonguay, and S. Nadarajah. Survey of Shape Parameterization Techniques and its Effect on Three-Dimensional Aerodynamic Shape Optimization. In *18th AIAA Computational Fluid Dynamics Conference*, Miami, FL, June 25-28 2007.

M. Nelder and R. Mead. A Simplex Method for Function Minimization. *Computer Journal*, 7:308–313, 1965.

P.M. Niskode, R. Stickles, B. Allmon, and R. DeJong. CLEEN Consortium Open Session. `http://www.faa.gov/about/office_org/headquarters_offices/apl/research/aircraft_technology/cleen/2010_consortium/media/GE%20-%20FAA%20CLEEN` url%20Consortium%202010%20-%20Unlimited%20Rights.pdf, Accessed March 2011.

M. O'Neill and C. Ryan. Grammatical Evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):77–90, 2001.

L. Piegl and W. Tiller. *The NURBS Book,*, volume 2. Springer, 1997.

S. Powell, A. Sóbester, and P. Joseph. Performance and Noise Trade-Offs on a Civil Airliner with Over-the-Wing Engines. *AIAA-2011-266*, 2011.

S.R. Powell and A. Sóbester. Application-Specific Class Functions for the Kulfan Transformation of Airfoils. *AIAA-2010-9269*, 2010.

S.R. Powell, A. Sóbester, and P. Joseph. Fan Broadband Noise Shielding for Over-Wing Engines. *Submitted to the Journal of Sound and Vibration*, 2012a.

S.R. Powell, A. Sóbester, and P. Joseph. Performance and Noise Trade-Offs for an Over-Wing Engine Configuration. *Submitted to the Journal of Aircraft*, 2012b.

W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing.* Cambridge University Press, 3 edition, 2007.

M. Reshotko and W. A. Olsen. Preliminary Noise Tests of the Engine-Over-The-Wing Concept II. 10^o - 20^o Flap Position. Technical Report TM X-68104, NASA, 1972.

G.M. Robinson and A.J. Keane. Concise Orthogonal Representation of Supercritical Airfoils. *Journal of Aircraft*, 38(3):580–583, 2001.

A. Ronzheimer, R. Rudnik, O. Broderson, and C.C. Rossow. Interference Phenomena of Upper-Wing-Mounted Engines. *Mitteilung-Deutsche Forschungsanatalt fur Luft und Raumfahrt*, pages 12.1 – 12.14, 1996.

C.C Rossow and A. Ronzheimer. Investigation of Interference Phenomena of Modern Wing-Mounted High-Bypass-Ratio Engines by the Solution of the Euler-Equations. *AGARD Conference Preceedings*, CP 498, 1992.

J. A. Samareh. Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization. *AIAA Journal*, 39(5), May 2001.

R. Schapire. The Strength of Weak Learnability. *Machine Learning*, 5:197–227, 1990.

H. Schlichting. *Boundary Layer Theory.* McGraw-Hill, 1979.

J. Seddon and E.L. Goldsmith. *Intake Aerodynamics.* AIAA Eduction Series, 2 edition, 1999.

T. W. Sederberg and S. R. Parry. Free-form Deformation of Solid Geometric Models. *SIGGRAPH Comput. Graph.*, 20(4):151–160, 1986. ISSN 0097-8930. doi: http://doi.acm.org/10.1145/15886.15903.

A. Sóbester. *Enhancements to Global Design Optimization Techniques.* PhD thesis, University of Southampton, UK, 2003.

A. Sóbester. Exploiting Patterns in the Kulfan Transformations of Supercritical Airfoils. *AIAA-2009-6951*, 2009.

A. Sobester and A. J. Keane. Empirical Comparison of Gradient-Based Methods on an Engine-Inlet Shape Optimization Problem. In *9th AIAA/ISSMO Symposioum on Multidisciplinary Analysis and Optimization*, number AIAA 2002-5507, Atlanta, Georgia, 4-6 September 2002.

A. Sóbester and S. Powell. Design space dimensionality reduction through physics-based geometry re-parameterization. *accepted into the Journal of Optimization and Engineering*, 2012.

A Sóbester, P. Nair, and A. Keane. Genetic Programming Approaches for Solving Elliptic Partial Differential Equations. *IEEE Transations on Evolutionary Computation*, 12(4):469–478, August 2008.

H. Sobieczky. Parametric Airfoils and Wings. *Notes on Numerical Fluid Mechanics*, pages 71–88, 1998.

J. Sobieszczanski-Sobieski and T. Haftka, R. Multidisciplinary aerospace design optimization: survey of recent developments. *Structural Optimization*, 14:1–23, 1997.

W. Song and A. J. Keane. A Study of Shape Parameterisation Methods for airfoil Optimization. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, number AIAA-2004-4482, Albany, New York, Aug 2004.

M.H. Straathof, M.J.L van Tooren, and M. Voskuijl. Aerodynamic Shape Parameterisation and Optimisation of Novel Configurations. *Proceedings of the RAeS Aerodynamic Shape Parameterisation and Optimisation of Novel Configurations Conference*, 2008.

B. Tang. Orthogonal Array-Based Latin Hypercubes. *Journal of the American Statistical Association*, 88(424):1392–1397, 1993.

R.H. Thomas, C.L. Burley, and E.D. Olson. Recent Progress in Propulsion Airframe Aeroacoustics. *Royal Aeronautical Society Aerodynamics Conference*, 2010.

A. Topchy and W.F. Punch. Faster Genetic Programming based on Local Gradient search of Numeric Leaf Values. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 155–162, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann. ISBN 1-55860-774-9. URL `http://garage.cse.msu.edu/papers/GARAGe01-07-01.pdf`.

D.E. Van Zante and A. Breeze-Stringfellow. The NASA/GE Collaboration on Open Rotor Testing. *Royal Aeronautical Society Aerodynamics Conference*, 2010.

Robert J. Vanderbei, David, and David F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1997.

J.C. Vassberg, E.N. Tinoco, M. Mani, O.P. Broderson, B. Eisfeld, R.A. Wahls, J.H. Morrison, T. Zickuhr, K. Laflin, and D.J. Mavriplis. Summary of the Third AIAA Drag Prediction Workshop. *AIAA 2007-0260*, 2007.

R. T. Whitcomb. Review of NASA Supercritical Airfoils. In *The Ninth Congress of the International Council of the Aeronautical Sciences*, volume ICAS 74-10, 1974.

R. T. Whitcomb and L. R. Clark. An Airfoil Shape for Efficient Flight at Supercritical Mach Numbers. Technical report, NASA TM X-1109, 1965.

D. Whitley, V. Gordon, and K. Mathias. Lamarckian evolution, the Baldwin effect and function optimization. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Mnner, editors, *Parallel Problem Solving from Nature PPSN III*, volume 866 of *Lecture Notes in Computer Science*, pages 5–15. Springer Berlin / Heidelberg, 1994. ISBN 978-3-540-58484-1.

H. Y. Wu, S. Yang, and F Liu. Comparison of Three Geometric Representations of Airfoils for Aerodynamic Optimization. In *16th AIAA Computational Fluid Dynamics Conference*, number AIAA-2003-4095, Orlando, Florida, June 23-26 2003.

R. Yoneta, D. Sasaki, and K. Nakahashi. Aerodynamic Optimization of an Over-the-Wing-Nacelle-Mount Configuration. *AIAA-2010-1016*, 2010.