Optimisation of Aspects of Rotor Blades using Computational Fluid Dynamics

Thesis submitted in accordance
with the requirements of
the University of Liverpool for
the degree of Doctor of Philosophy
by

Catherine Johnson

June 2012

Declaration

I hereby declare that this dissertation is a record of work carried out in the School of Engineering at the University of Liverpool during the period from November 2008 to June 2012. The dissertation is original in content except where otherwise indicated.

June 2012

.....

(Catherine Johnson)

Abstract

This work presents a framework for the optimisation of various aspects of rotor blades in forward flight. The literature survey suggests that the quest for such a method is generating much research as more performance is obtainable from current designs. With increasing computational power and efficient methods, this can be of practical use to the helicopter industry. The proposed method employs CFD in conjunction with metamodels such as artificial neural networks (ANNs) and kriging interpolation, and a non-gradient based optimiser, in the form of genetic algorithms (GAs), for optimisation. The approach is demonstrated using several cases, including the optimisation of linear twist of rotors in hover (a steady case) and the optimisation of rotor sections in forward flight (an unsteady case); other cases include transonic aerofoils, wing and rotor tip planforms. For rotor tip planforms, first a simple rectangular rotor in hover was optimised. Then the developed method was used to optimise the anhedral and sweep of the UH60-A rotor blade in forward flight while constraining its hover performance and the final rotor optimisation was for a BERP-like rotor in forward flight, also constraining hover performance. For each case, a parameterisation method was defined, a specific objective function created using the initial CFD data and the metamodel was used for evaluating the objective function during the optimisation using the GAs. The obtained results suggest optima in agreement with engineering intuition but provide precise information about the shape of the final lifting surface and its performance. The results were checked by comparison with the Pareto subset of data and the metamodels were also validated with high-fidelity CFD data. Neither was sensitive to the employed techniques with substantial overlap between the outputs of the selected methods. The main CPU cost was associated with the population of the CFD database necessary for the metamodel. To improve this further, the Harmonic Balance alternative for obtaining the CFD data (as opposed to Time Marching) was used to increase efficiency and reduce clock time for the BERP-like tip optimisation. The novelty of this method is the use of a metamodel in conjunction with high-fidelity CFD data so that high-resolution performance improvements can be captured efficiently using a non-gradient based method.

List of Publications

In Journals

- C.S. Johnson and G.N. Barakos, A Framework for Optimising Aspects of Rotor Blades, The Aeronautical Journal, vol.115, Issue 1165, p.147-161,15p, March 2011.
- C.S. Johnson, M. Woodgate and G.N. Barakos, Optimisation of Aspects of Rotor Blades in Forward Flight, International Journal of Engineering Systems Modelling and Simulation, vol.4, Issue 1/2, p.79-93, 2012.
- C.S. Johnson and G.N. Barakos, Development and Demonstration of a Framework for Optimising Aspects of Rotor Blades in Forward Flight, International Journal for Numerical Methods in Fluids (in review)

In Conference Proceedings

- C.S. Johnson and G.N. Barakos, Development of a Framework for Optimising Aspects of Rotor Blades, American Helicopter Society 66th Forum, Paper 377 [CDROM], Phoenix, Arizona, May 2010.
- C.S. Johnson and G.N. Barakos, Optimising Aspects of Rotor Blades in Forward Flight, AIAA Forum, Paper ID 895432, Orlando, Florida, USA, Jan 2011.
- C.S. Johnson and G.N. Barakos, Rotor Tip Optimisation in Forward Flight, 46th Symposium of Applied Aerodynamics, Aerodynamics of Rotating Bodies, Paper 32 in Conference Proceedings, Orléans, France, March 2011.
- C.S. Johnson and G.N. Barakos, Optimisation of Aspects of Helicopter Rotor Blades and Fuselage, 37th European Rotorcraft Forum, Paper 93, Milan, Italy, 13-15th September 2011.

Technical Notes

- Technical Note, TN10-013, Parameteric Mesh Generation for HMB.
- Technical Note, TN10-022, Metamodel and Optimisation Algorithms for HMB.
- Technical Note, TN10-, BERP Tip Grid Generation Technique in ICEMCFD.
- Technical Note, TN10-, Parameterisation Technique for the UH60-A blade.
- Technical Note, TN12-003, Actuator Disk Models in HMB.

Acknowledgements

I would like to thank my supervisor, Prof. George Barakos for being an exemplary mentor and a brilliant advisor for this project. His passion for research and his experiences have taught me a lot that I won't forget. I would also like to thank Dr. Steijl and Dr. Woodgate for their consistent support throughout the period of this project. I definitely could not have achieved so much without them. Thank you to my work colleagues who have also been a great support in and out the lab, both those present now and those who were here previously. I have enjoyed working with this group of people because of the quality of work produced and the diversity and character of the people that make it. I would also like to mention Alan Brocklehurst and thank him for his valuable advice and for his time. I learnt much from him in every conversation.

In addition, I would also like to thank my family who have been my unconditional support in every aspect. I would like to honour my father who is an inspiration to me and has sacrificed much for me in this time. A very special thank you goes to my sister and greatest friend, Caroline, for being my ever present help in times of need regardless of what it may be, even learning about my project with me. I'd like to also thank Sidra, my housemate, who also recently finished her PhD. It was great to have her as a friend on this journey from the first day as engineering undergraduates to the last day as PhD students.

To all at Grace Family Church, Liverpool, who have supported me in so many ways that I would have to bind my acknowledgements separately if I mentioned them all. Their friendship and concern was greatly valued. Last, but not least at all, I would like to thank God without whom, none of this would be possible. His work is in this project, the original inspiration that led to Genetic Algorithms and Neural Networks and every other aspect in this project. He has been my strength (Philippians 4:13) and guidance and most of all, I hope this work tries to reflect His standard of excellence.

Contents

List of Figures

List of Tables

Nomenclature

Latin

A	Area of rotor disk
AR	Aspect ratio
AoA	Angle of Attack
b	semi-wing span
c	chord
C_D	Coefficient of Drag
C_d	Sectional coefficient of Drag
C_L	Coefficient of Lift
C_l	Sectional coefficient of Lift
C_M, C_m	Coefficient of Moment
C_n	Coefficient of normal force
C_T	Coefficient of Thrust, Non-dimensional ratio of thrust to rotor disk area, density and velocity squared, $\frac{T}{\frac{1}{2}\rho S(R\Omega)^2}$
C_p	Coefficient of Pressure
C_P	Coefficient of Power
C_Q	Coefficient of Torque, Non-dimensional ratio of torque to rotor disk area, density, velocity squared and length, $\frac{Q}{\frac{1}{2}\rho SR(R\Omega)^2}$
dM/dt	Pitch translation oscillation
e	Internal energy of fluid element
E	Error, Total Energy per unit volume
E_{ell}	Difference in elliptic loading
f	Rotation frequency
FM	Figure of Merit, Ratio of power required to produce thrust, P to total power required, $P + P_o$ where P_o is the profile power to overcome aerodynamic drag of blades, $\frac{C_T^{3/2}}{2C_Q}$.
h	Hidden layer
k	Reduced frequency, $\frac{c\omega}{U_\infty}$
k	Turbulent kinetic energy in $k-\omega$ model
M	Mach number
M1, M2	Midchord 1 and 2 values
N_b	Number of blades
P	Power
p	Pressure
q	Heat flux
r	Radial station along blade
R	Length of blade, radius of rotor, Residual
Re	Reynolds number
S	Area
T	Temperature
t	time
U	Velocity in a generic direction
u	Flow velocity in x-direction
V_{tip}	Tip velocity
v	Flow velocity in y-direction
v_i	Induced velocity
w	Flow velocity in z-direction
X	A generic direction
x	x direction

y	y direction
z	z direction

Greek

α	Parameterisation coefficient, angle of attack
β	Coning angle
γ	Lock number
Δ	difference between maximum and minimum
μ	Advance Ratio, Ratio of forward velocity to blade tip velocity
ξ	Vorticity
ρ	Density
λ	Induced inflow ratio
κ	Induced power correction factor
τ	Viscous stress tensor
σ	Rotor solidity, Ratio of total blade area to total rotor-disk area, $\frac{N_b c}{\pi R}$. Typically between 0.05 and 0.12
η	Learning rate
θ_o	Collective angle
Ψ	Azimuth angle
Ω	Angular Velocity
∇	Gradient

Subscripts and Superscripts

avg	average
$\overline{C}$	Average value of C
c	cosine term
s	sine term
o	constant term
tip	blade tip conditions
∞	under freestream conditions

Acronyms and Definitions

AE	Algorithm Effort
AHS	American Helicopter Society
AIAA	American Institute of Aeronautics and Astronautics
ANN	Artificial Neural Networks
BET	Blade Element Theory - considering the span of the blade as separate strips.
Bézier curves	Using just one equation to get more and more complex curves leads to high degrees of polynomial. One solution is to create complex curves out of many simple curves by matching them point-to-point and gradient-to-gradient. Bézier curves are defined by 4 control points, knots - 2 are the end points and the other 2 effectively define the gradient at the end points.
B-splines	They are a more general form of Bézier curves. They allow local knots to have more weight in determining the course of the curve.
BVI	Blade Vortex Interaction
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy condition
Disk loading	Thrust divided by rotor-disk area.
DoF	Degrees of Freedom
EA	Evolutionary Algorithms
ERF	European Rotorcraft Forum
FM	Figure of Merit
GA	Genetic Algorithms
GDM	Gradient Descent with Momentum
HART	Higher harmonic control Aeroacoustics Rotor Test
HB	Harmonic Balance
HMB	Helicopter Multiblock Solver
Inverse design	Design of component starts with the required impact on the fluid e.g. blade geometry designed starting from pressure coefficients, lift coefficients etc. required.
JAXA	Japan Aerospace Exploration Agency
LE	Leading Edge
LMA	Levenberg-Marquardt Algorithm
MOO	Multi-objective optimisation
NLFD	Non-linear Frequency Domain
NS	Navier-Stokes
NURBS	Non-Uniform Rational B-Splines
OFO	Objective Function Optimisation
OFV	Optimisation Function Value
PFO	Pareto Front Optimisation
PIV	Particle Image Velocimetry
POD	Proper Orthogonal Decomposition
RAE	Royal Aircraft Establishment
RSA	Response Surface Area
RBF	Radial Basis Function
RPSO	Repulsive Particle Swarm Optimiser
SA	Simulated Annealing
SBD	Simulation Based Design

SBO	Surrogate-based Optimisation
Sensitivity	The partial derivative of a response with respect to a design variable.
SST	Shear Stress Transport
TE	Trailing Edge
TM	Time Marching
Vorticity	A measure of the rotation of fluid elements in fluid flow. Given as $\xi = \nabla \times V = 2\omega$

Chapter 1

Introduction

1.1 Motivation

The design of rotor blades is complex, in that it involves many disciplines of engineering such as aerodynamics, structural, dynamics, aeroelasticity and control systems. These disciplines do not just play an individual part in the design of the rotor blade, but are also coupled and hence have influence on each other; some more strongly than others[12] . Even within a single discipline such as the aerodynamics of the rotor, there are often conflicting design requirements; Forward flight tends to have opposite requirements to hover, the blade on the advancing side has opposite requirements to that on the retreating side of a forward moving helicopter and so on[13] . Therefore, defining an optimum blade tends to be a compromise between these various conditions. Hence, the optimum is determined by the objectives to be achieved by the rotor. While the initial concept design may be relatively easy to come up with, finding the optimum design parameters, while taking into consideration all the variables, is not an easy task.
Therefore, computer codes that aid helicopter designers have been, and are still being, developed and used in industry[14,15] . In the past, these methods were limited to using simple theories that modelled the aerodynamics of a rotor. This limitation was due to the high cost in obtaining sufficient data to make valid comparisons for a set of design parameters. This is obtained either by experimental data or by computational simulations. Both methods involve high costs. The computational costs, however, can be reduced by reducing the complexity of the models used to simulate the aerodynamics around a rotor. This, however, compromises the accuracy of the data. Nevertheless, over time, computing power capabilities have increased allowing more advanced simulation models to be used. This has gathered a lot of interest in the research of design and optimisation methods as seen in Section 1.3.

A variety of methods have been developed, and the majority of the applications have been for cases that are small in size (such as aerofoils) or simple in the simulation of the aerodynamics (such as cases where operation is optimised for a single static condition). The challenge now, is to apply these methods to a complex design such as a helicopter rotor blade, where the aerodynamics are complex and change, and the design space has a large number of dimensions, and to do this accurately and efficiently. The initial concept design is a well-established process and designers and engineers of rotor blades have a lot of experience to lean on, as well as the assistance of simple codes to aid them in obtaining a preliminary rotor design. The optimisation methods become more applicable when optimisation of an existing design is required to obtain even greater performance from a rotor.

Optimisation techniques usually require a starting design point, so the design stage is just as important as the optimisation. While it is possible for optimisation to lead to new designs, the time and effort involved would attract a high cost. Take the BERP tip blade as an example[16] . The BERP tip blade is not something that can easily be created simply by using optimisation methods. However, if the designer's ideas and experience in the field of aerodynamics was used to create

an initial design, then the dimensions and extent of BERP blade characteristics can be perfected to improve its performance greatly. When many characteristics of a rotor are 'tweaked' in such a way, a considerable amount of improvement can be made[13] . This is what makes optimisation so important and has led to the increase in the amount of research and papers written on this topic.

In addition to the added performance gained by using optimisation procedure, the use of numerical solutions for the optimisation problem removes some of the workload of obtaining the optimum design from the designer while still giving the designer flexibility. In the case of rotors specifically, there are many criteria and objectives that must be fulfilled simultaneously, what is known as multiobjective optimisation (MOO). Depending on the required performance, it is possible to program the optimiser to create a rotor tailored to its expectations in many diverse conditions.

The challenge of optimisation for helicopters has been summarised in the following quote from Celi[17] :
"Researchers in helicopter applications of optimisation face a complex multidisciplinary problem, with several possible choices of design variables, objective functions, behavior and side constraints, analysis models, sensitivity formulations, approximation concepts, optimisation algorithms, not to mention the many types of results that can be generated and presented."

1.2 Literature Search

The literature survey started off with very broadly used terms such as rotor optimisation, multiobjective optimisation, shape, aerodynamic optimisation, helicopter rotor design, etc. The papers that were relevant often led to other specific papers through the bibliography. Also, specific words and ideas that repeated themselves in many papers, such as 'shape optimisation', 'evolutionary algorithms', 'artificial neural networks', 'genetic algorithms', 'adjoint methods', 'surrogate models', 'metamodels', 'non-linear programming', etc. were then used in the same databases. This led to a greater search space as it covered other areas of engineering such as electric component engineering and so on. A lot of the literature was also obtained from conferences, especially the American Institute of Aeronautics and Astronautics (AIAA), American Helicopter Society (AHS) and European Rotorcraft Forum (ERF) and their corresponding journal papers as well as more specific conferences geared towards optimisation and Computational Fluid Dynamics (CFD).

1.3 Literature Review

The design of helicopters, in general, is a complicated task and requires a number of iterations before the final design is obtained. The University of Maryland has documented an upgrade of a helicopter[18] . This paper provides some basic insight into the design of helicopters, especially the rotor. For example, the preliminary sizing to determine the number of blades, the diameter, AR and solidity of rotors is to achieve a required cruise speed at low cost. This incorporates many variables such as aerofoil sections, taper, twist and blade tip design (which itself includes sweep, taper and anhedral).
NASA documented a design-from-scratch procedure for rotorcraft design and optimisation[1] . Figure 1.1(a) is an image from this document showing a simplified overall design process for helicopters. Figure 1.1(b) is an expanded version of a part of that process, which involves the design of the rotor. As can be seen, the design of the rotor is an iterative process between various disciplines. However, even within the aerodynamic discipline alone, the process requires a number of iterations that must fit in with constraints that may be due to sizing, material and other such limiting factors, which are not necessarily aerodynamic related. This narrows the aspects of a blade that can be optimised aerodynamically.

NASA also developed a more recent conceptual design tool[14] that designs rotorcraft for specific requirements and then analyses the performance for a set of flight conditions and missions. It has the ability to calculate the weight, power and size of typical configurations but allows flexibility in obtaining new concepts. Various components can be added and sized accordingly. The governing equations are simplified methods and most are analytical, although higher order methods such as CAMRAD have been coupled with them[19] .

The optimisation for rotor aerodynamics is typically one that is initiated from an already 'optimised' design, to tweak the values of certain design parameters so that more performance is obtained in hover and forward flight. Over time, many studies have been carried out on the effectiveness of various design parameters for rotor blades. Leishman's book[13] gives some insight in to the significance of each parameter. It also provides some basic theories and effects that the geometry of a rotor has on overall performance. In some cases, the significance is large, but the design parameter is optimised for a non-aerodynamic purpose primarily. For example, in the preliminary design of the rotor, the rotor diameter is usually specified according to non-aerodynamic purposes such as storage, gear box ratio, torque limits, stiffness or drooping of stationary blades and so on. However, the rotor diameter does affect the aerodynamic performance of the helicopter, since the bigger the diameter, the better the hover performance because of the lower disc loading and hence the lower the induced velocity and power. In addition, the autorotation capabilities are improved. Another characteristic sometimes affected by non-aerodynamic parameters mentioned is solidity, σ. Aerodynamically, reducing the rotor solidity, reduces the profile power but increases the disc loading and hence induced power. Since there is less lifting area, the angle of attack of the blades may need to be increased to maintain the same lift. Therefore, σ is limited by the stall margin of the blade. In forward flight, a higher solidity is required to maintain the same lift, but the greater the σ, the lower the Figure of Merit (FM).
In his book[13] , Leishman also gives a basic outline of how to select aerofoils. For hover,

$$FM = \frac{\frac{C_T^{3/2}}{\sqrt{2}}}{\frac{\kappa C_T^{3/2}}{\sqrt{2}} + \frac{\sigma C_{d_o}}{8}} \tag{1.1}$$

$$= \frac{1}{\kappa + \frac{2.6}{\sqrt{\sigma}} \left(\frac{C_L^{3/2}}{C_D} \right)^{-1}} \tag{1.2}$$

According to Equation 1.2, the aerofoils must have high $C_l^{3/2}/C_d$.
The general emphasis is typically on obtaining a high $C_{l_{max}}$. This will allow lower solidity and

high manoeuvre loads, a higher drag divergence Mach number permitting higher flight speeds without too much power loss and noise, a good lift-to-drag over a wide range of Mach numbers to maintain low profile power consumption and low autorotative descent and a low pitching moment to minimise blade torsion moments, vibrations and to keep loads on the control system small.

Usually thickness is kept small to accommodate a higher Mach number. To compensate for the lift, camber is added. In some cases, trailing edge tabs and reflex angles are added to highly cambered aerofoils so that high lift and zero moments can be obtained. On the retreating side, where stall is likely to occur, thicker and more cambered aerofoils stall more gently than thin aerofoils with sharp leading edges. To obtain the best performance, a compromise is required.

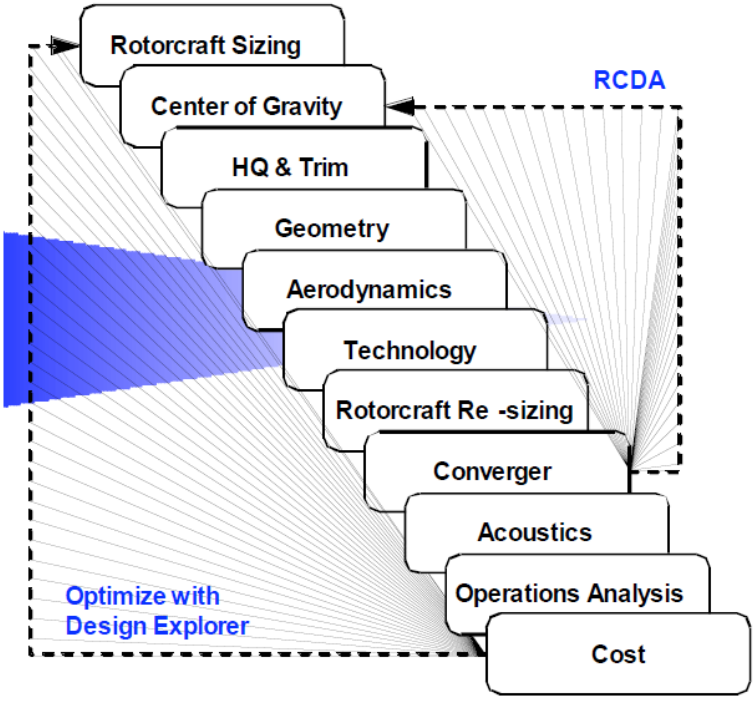
Martin and Leishman[20] performed a study on how different tip shapes (rectangular, sweep, taper and subwing tips) in hover affect wake geometry (which is a significant contributor to the performance of rotors - especially in hover[12] - as it can result in velocity gradients that cause stall on the following blade, high induced velocities and radial flow) and came up with a number of findings. For example, in hover, tip sweep appeared to decrease both radial and axial convection of the core while taper increased radial and decreased axial convection. Also, the primary effect of sweep is that the vortex core is pushed outward past the tip plane path because a swept tip does not conform to the circular streamlines, but is "sheared" so that it extends to outside r/R = 100%. The vortex peak swirl velocity (over tip speed) is a function of vortex strength, core circulation normalised with the farfield value and vortex radius. It is also a function of blade planform (twist, taper and sweep) since these parameters influence the peak bound circulation which influences the vortex strength.

Previously, Caradonna also carried out experiments on wakes and looked specifically at vortex strength[21] . His experiments used a hot-wire probe to find the wake trajectories and velocities. He used a method similar to that used by Cook[22] to separate the velocity induced by the wake and the velocity induced by the rest of the system. Even though there is quite a lot of variability between vortices, they all seem to have similar appearances to the classic Rankine vortex[23] . However, as rotor speed increases, the appearance departs from the Rankine vortex. The non-dimensional vortex strength remained, however, independent of tip speed. Furthermore, blade vortices seem to contain all the blade circulation.
Caradonna also wrote a second paper on the use of CFD in rotorcraft design[12] . Some of the compromises in the design of the main rotor - such as the tip speed - are pointed out. The compromise here is between transonic flow on the advancing side and stall on the retreating side. However, this is strongly dependent on the aerofoil section, twist, planform and propulsive requirements. For hover, the single most important flow issue is the wake because it is the primary determinant of induced power. For forward flight, it has slightly less significance because the wake is convected away from the blade faster; but its detailed prediction is still required for vibratory loadings, for example.

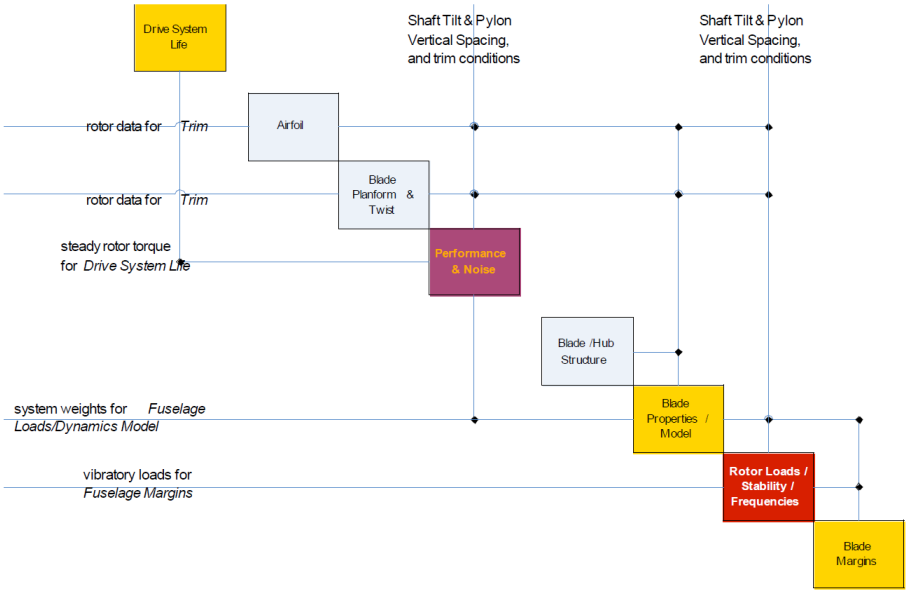
Very specific design parameters such as twist, aerofoil sections, tip sweep, anhedral and planform geometry start to emerge as significant contributors to rotor blade performance as more of these studies are looked at. Keys et al.[24] also carried out wind tunnel studies on the effect of twist on rotor blades. Twist improves hover performance because of the better distribution of loads, i.e. more uniform downwash in the far wake leads to lower induced power. However, very low and even negative angles on the tips of rotor blades can adversely affect forward flight performance and vibratory loads. Also, as twist is increased, there is more downwash inboard in hover, and so there is an increased downward push on the fuselage. In effect, one has a rotor disc that pushes the load (or fuselage) down in the middle but is trying to lift it with the tips. The effect of twist in forward flight is that more power is required above an advance ratio (μ) of 0.2 due to the increase in profile power on the advancing tip rather than the increase in Mach number resulting in a decrease in L/D. The effect on stall inception is minimal. Stall inception

was defined as the C_T/σ where the blade torsion or pitch link loads increase rapidly (C_m). Also, in general, increasing blade twist increases hub and blade vibratory loads. The paper also provide some suggested solutions such as the use of sweep and taper on the blade tip but also the use of live twist, so that there is azimuthal variation in the twist to obtain a reduction in twist on the advancing side. Some helicopters such as the UH60-A reverse the twist at the tip so that the loss is reduced in forward flight, while maintaining the performance in hover[13] .

These parameters, which have relatively large influence on the design, have become the focus of design and optimisation and are modified to suit the requirements of the rotor. Figure 1.2 shows some of the blade planforms that have been developed.



(a)



(b)

Figure 1.1: *(a) General design process for helicopters, (b) The design process involving rotors. (Courtesy of Stanley A. Orr and Robert P. Narducci, Framework for Multidisciplinary Analysis, Design, and Optimization With High-Fidelity Analysis Tools, NASA/CR-2009-215563, Feb 2009[1]).*

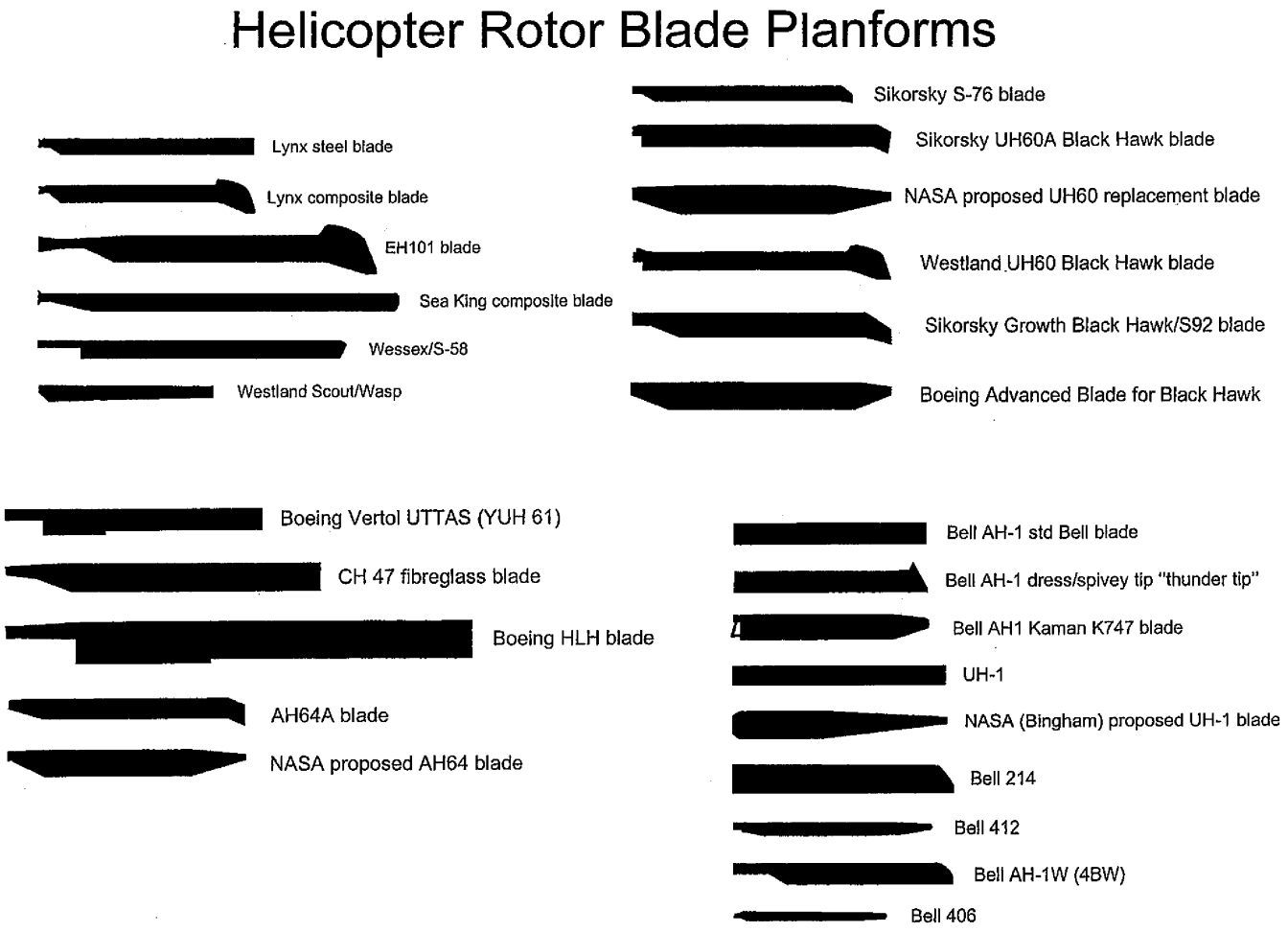


Figure 1.2: *Various blade planform profiles.*

1.3.1 High-fidelity and Low-fidelity Models

Rotor blade optimisation is becoming an increasingly important part of the design process as engineers push the boundaries of rotorcraft efficiency and performance further. However, the development of helicopter rotor optimisation specifically has been slow compared to other rotors such as turbines and compressor blades. This is due to the increased complexity and unsteady nature of the flow around a rotor blade[17] . Also, the problem is a multi-disciplinary problem involving aerodynamics, aeroelastics, dynamics and flight control coupled to each other. This has lead to the development of comprehensive codes that are able to take into consideration all these disciplines[17] by combining modules of codes for each discipline. A recent example of this is the work by Rajmohan et al.[25] . Here, GT-Hybrid, a CFD solver was coupled with a dynamics code, DYMORE to assess a manoeuvre for a UH60-A rotor. The effects of the blade geometry and wake model fidelity were explored in steady flight comparing loose and tight coupling of the codes. GT-Hybrid was used as a high-fidelity CFD code employing Navier-Stokes equations near the blade and wake models further away. DYMORE uses lifting line theory and an auto-pilot algorithm to perform a fully-trimmed aeroelastic simulation. Differences in the aerodynamics, such as stall on the retreating side, were found to exist even due to the options of loosely and tightly coupling the two codes, suggesting that the coupling of disciplines will be a key part in developing good optimisation tools in the future, when computational power increases and CPU-time is reduced. Biedron and Lee-Rausch[26] also used an unstructured flow solver (FUN3D) loosely coupled with with CAMRAD II in comparing experimental and CFD data and how they are integrated for the UH60-A rotor.
Celi also suggests that the optimisation method should be decoupled from the analysis so that it is usable with different analysis tools, hence allowing the optimisation technique to use the best version of the analysis code at any time.
Also, the development of optimisation techniques has progressed quite quickly in the structural aspect of rotor design for various objectives such as vibration reduction[27,28] . This was not the case on the aerodynamic front due to the lack of modelling techniques[17] that are both efficient and accurate, although there has been significant improvement. Nevertheless, several authors[2,17,29,30] have attempted to devise a variety of successful optimisation techniques. However, each method is typically limited either by the efficiency of the method or the accuracy of the results. The reason for this is that high-fidelity CFD simulations are necessary to accurately capture the effects of design changes, especially for rotor aerodynamics but a number of these CFD solutions are required for the process and each calculation can take a long time at a significant computational cost. For example, it was found by Walsh[2] , who initially optimised a blade without including a wake model for the analysis, that including one had significant contributions to the design of the final optimum rotor blade. Even in the case of fixed-wing aircraft for a single point optimisation problem, the use of higher-fidelity solvers is desirable such as in the work by Rallabhandi and Mavris for sonic boom optimisation[31] .

To overcome the computational cost problem, a lower fidelity model or metamodel can be used. A metamodel is a model of a model. For this project, it would be a lower fidelity model of the high-fidelity CFD model used to obtain the performance of a number of designs. The metamodel is able to predict the performance of unknown design points a lot faster but with the same accuracy as a higher fidelity model. It is able to do this because it is given a reduced number of parameters relative to the CFD method and uses only these parameters to find patterns in the change of the output required. It does this accurately because the data it is provided with, are based on high-fidelity aerodynamic data obtained from CFD.

Some work has also been done in using both techniques dynamically such as the work by Kolencherry et al.[32] where a kriging metamodel was constantly updated with promising high-fidelity data as the optimisation was carried out. Collins et al.[30] also used a combination of low and high-fidelity models to optimise a scaled BO-105 rotor in hover and forward flight for noise and power. An initial set of designs was obtained and used to obtain Pareto optimal designs, a few

of which were selected for validation using the high-fidelity solver and these results were used to update the lower order model.

CFD Methods

In the first attempts at optimisation of rotors, the CFD methods used were simplified. For example Walsh[2] attempted to optimise a rotor in hover while maintaining the performance in forward flight. *HOVT* (a strip theory momentum analysis with no wake model in the initial analysis) was used to obtain the hover horsepower and *CAMRAD*, a comprehensive code, for the forward flight horsepower and manoeuvre and trim conditions. Even at this early stage, the idea of using CFD methods for optimisation was attractive. CAMRAD was used because it could be coupled with CFD methods that could be used to obtain better models of effects like transonic effects[33] . Only some change in the geometry of the blade was captured and hence only a slight improvement was obtained. The conclusions were that further validation and testing would be required.
In the work done by Bohorquez et al.[34] for optimisation of small rotors where one of the key objectives was cost-effectiveness, the computational cost of the solutions was further reduced by using incompressible steady calculations with one-equation models of turbulence.

The optimisation by Le Pape and Beaumier[35] was coupled with the solver *elsA* developed by ONERA. Steady RANS was used for the hover calculations. Computations were performed on a coarse grid during optimisation and a fine grid once the minimum was found. In addition, the HOST (Helicopter Overall Simulation Tool) was coupled with elsA, to include aeroelastic and dynamic disciplines in the analysis (i.e. soft blade), since blade torsion has a significant influence on the aerodynamics of the rotor. So at each optimisation step, the HOST code computed the blade deformation (flap bending and torsion) and the rotor angles (flap and lead-lag) and a new grid was built around the deformed blade and the CFD computed. However, no recomputation of trim was done, making the coupling slightly inaccurate, especially if forward flight was being considered.

Le Pape[36] later extended his work to forward flight. elsA was only used for hover optimisation since forward flight calculations would take a lot more time and a trimming model would be required as well. HOST was used as the aerodynamic model for the forward flight cases as well as in hover coupled with elsA. It was used as a lifting-line model with a prescribed wake model for forward flight. However, these types of solvers are not very capable of determining the performance of complex blade shapes in forward flight.

Imiela[37] optimised the aerodynamic properties of rotor blades considering both hover and forward flight using CFD, whilst constraining structural and aeroelastic deformation as calculated by CSD. The solutions were based on high-fidelity CFD/CSM analyses that had weak fluid-structure coupling (i.e. each discipline calculated alternately) combined with an optimisation algorithm. FLOWer was used for the aerodynamic simulation and HOST for blade dynamics and elasticity, although the optimisation itself was focused on aerodynamics. The structural model was not modified during the optimisation.

It can be seen that using a high-fidelity model directly for optimisation of rotor blades is either limited to simpler cases such as hover, or the model's fidelity is reduced for more complex cases such as forward flying rotors. To be able to model forward flight efficiently, a lower order model is required, but its accuracy still needs to be maintained, especially because of its complexity. Metamodels have this ability.

Metamodels

Metamodels have been used extensively in the optimisation of turbomachinery blades[38,39] .
Mengistu and Ghaly[38] used an artificial neural network (ANN) as a metamodel, along with a Genetic Algorithm (GA) optimiser to optimise turbomachinery blades. They used high-fidelity CFD

data to train the ANN to predict the performance of interpolated design points. The prediction weights were updated using an optimisation algorithm to minimise the error between the network output and the training data (obtained using CFD). The training strategy was enhanced using a GA and gradient-based optimisation. There was no rule for determining the number of nodes in the layer of neurons, but a good initial guess was the average of the input and output nodes. This number was increased to create an optimal-trained network. If the number of nodes was too small, underfitting occured, where high error values were obtained and too much generalisation occurred, and vice versa.

In terms of helicopter rotors, optimisation using metamodels has been performed mostly with the aim of vibration and noise reduction such as the work done by Glaz, Friedmann and Liu[27,28]. Here, it is suggested that for the particular case of vibration reduction, kriging has proven to be the best surrogate method. However, it brings out the factors that affect the choice of approximation method - such as sampling density, scheme used to select the points and the parameters of the metamodel. It is suggested that since no single metamodel is generically the best, it may be useful to use a combination of metamodels instead. Three metamodels were used viz. polynomial response surface, kriging and radial-basis neural networks. These were combined by a weighted average surrogate algorithm. The conclusions drawn were that the most accurate method depended on sample size and the error evaluation method and that the most accurate metamodel did not necessarily produce the best design. Also, in the work done by Liu et al.[40], it was shown that for more complicated cases, kriging out-performs RBFs whereas for simpler cases, RBFs are more accurate and efficient. Celi[17] also mentioned that the "connection between predictions and accuracy of the optimisation may be more subtle than appears at first glance."
Imiela[37] used DACE[41], a MATLAB kriging based metamodel for the optimisation of the rotor in forward flight. Kriging is similar to radial basis function interpolation (RBFs). It is a method of interpolation but the weights involved are driven by the data rather than arbitrary functions. Usually, the weights are functions of the distances of points from the required data point. This means that an estimate of the error is also available. The aim is to determine the weights such that the variance of the estimator is low.[42]
Kriging is becoming increasingly popular. One reason is because it has few assumptions; in fact the only real assumption is that the design space or equation is continuous[43]. For example, Glaz et al.[44,45] used it to predict the unsteady lift, drag and moment under dynamic stall for rotor sections on a helicopter blade. The Navier-Stokes equations were used to compute the calculations and the dynamic stall was modelled as a function of time and time history. This non-linear mapping function that is numerically found, linking the input to the output, was modelled using kriging at each time step. In effect, the surrogate model had two dimensions: the kriging for each time step, and the SBRF (surrogate based recurrence framework) to complete the prediction including the time histories. This worked well, with error values falling within 3%. There are a number of types of kriging apart from ordinary kriging, such as universal, co-kriging and blind kriging[43]. Kriging, however, has been found to be less accurate when the data is sparse[43].
Marcelet and Peter[46] compared the performance of four different metamodels. It was found that kriging and ANN were two of the most accurate methods, and that kriging was more efficient. Ahmed and Qin[47] compared RBFs and a number of kriging methods as metamodels to optimise hypersonic spiked blunt bodies. They explain how kriging works and that it can be as good a metamodel if not better than an RBF. Rendall and Allen[48] did some work on improving the efficiency of RBFs. Their paper describes a version of RBFs called greedy RBFs whereby the prediction of a point is dependent on a smaller selection of points from the entire design space thereby avoiding the calculation of each point's prediction in relation to the required output. It also lists a number of commonly used basis functions used in RBFs and suggests that the greedy RBF will be useful in compressing data and improving efficiency. This was not verified by other researchers.

In the work done by Sun et al.[49], Response Surface Models (RSMs) were used to obtain the performance of interpolated design variables for rotor sections in pitching motion. However, they

also included a regression model to give it more flexibility in fitting the surface, similar to kriging. In addition, they used t-statistics to determine the significance of each variable to the objective function and hence determine the accuracy of the response surface.
Hajela[50] also discusses a number of metamodels used in rotor optimisation such as Immune Network modelling, where the design and performance parameters are simulated as antibodies and their corresponding antigens. Their relation is quantified by a matching function, which works similarly to the fitness value in a Genetic Algorithm (a non-gradient based optimiser).
Other methods include the use of the Fourier transform to predict the flow solver data, as in the work done by Nadarajah and Tatossian[51] . Their goal was to introduce a Mach number variation into an existing non-linear frequency domain (NLFD) framework and then introduce a time-varying cost function through an adjoint boundary condition to redesign a NASA rectangular supercritical wing. They state that there is a need to develop cost-effective optimal control techniques for unsteady aerodynamics and that an efficient method may be the use of periodic methods such as the Linearised Frequency Domain. However, these methods become highly inaccurate when there are strong non-linearities although further validations with high fidelity software show that this can be overcome[52] . The efficiency of the NLFD approach is dependent on the periodicity of the flow. In the case of rotor aerodynamics, representing the variations of loading in the frequency domain can lead to a large number of modes required to represent the data as compared to other methods, such as the Proper Orthogonal Decomposition method (POD). The POD method as an interpolation technique was used by Oyama et al.[53] to optimise transonic aerofoils for lift/drag ratio. The aerofoils were represented as splines with the design parameters being the control points, and a POD technique was used to obtain the lift-to-drag ratio for new aerofoils that existed on the Pareto front.

A modified version of the POD technique called the Gappy POD method was developed by Willcox et al.[54,55] for aerodynamic applications. This method is used to interpolate for incomplete or "gappy" data by solving a linear system of equations given a set of POD modes[54] . Such a method could be used to fill in for incomplete experimental data, or for unknown conditions and even inverse design. In the work done by Gunes et al.,[56] a comparison is made of kriging and POD based interpolation to reconstruct the unsteady flowfield around a cylinder. It was found that the POD method is more accurate when there are many snapshots and smaller gaps in the field. When this was not the case, kriging provided a more accurate solution.

Metamodels can be very useful in reducing computational cost and expanding databases as long as they can give accurate predictions that the optimiser can rely on. The usual way of testing the accuracy is to validate an unknown point in the database with its actual value as obtained from the high-fidelity solver. In the work done by Tahara et al.[57] , where two optimisers were being compared, they introduced an error value which was the difference between the measured and expected improvement by the optimiser. In a similar way, this can be applied to metamodels, where the difference between the predictions and actual values are used to validate the model. If the cost of running an additional set of data is high, another method of testing the prediction accuracy of a metamodel is to use cross-validation[43] . Here, the design space is divided into subsets, and each one is removed and the metamodel trained without it. It is then reintroduced and compared with its prediction. The average of these is the error. The problem with metamodels is that they fit the existing data but the accuracy of their predictions outside of this trained area is questionable.

The gain in computational cost through the use of metamodels increases with the number of objectives but the accuracy decreases[58] . The reasons for this are stated by Karakasis and Giannakoglou[58] who suggest that even in the comparison of the case of single and multi objective optimisation, within the first generation, the single objective optimsation individuals cluster around the optimum region and the metamodel has sufficient points to reliably evaluate them whereas in the case of a Pareto optimisation technique, the available information is spread over a front. To counter this, a number of these can be combined with weighted formulae.

In Karakakis[58] , a metamodel was built for each new member of the population and was limited to a local vicinity of approximately 100 to 150 individuals. It was found that this worked better than building an overall global metamodel in terms of better approximations.

1.3.2 Sampling Methods

Before a metamodel can be used, the sampling of points in the design space must be determined. This is what is known as 'Design of Experiments', (DOE). The DOE depends on many factors such as the metamodel to be used, the cost of running the high-fidelity software, the dimensions of the problem and the shape of the design space. Usually Latin hypercube sampling (LHS), and its variants, minimum bias design and orthogonal arrays are preferred[59] . Crowley et al.[60] used a nested LHS method, where there were LHS of increasing resolution within each other.

Mackman et al.[61] compared kriging and RBF methods for various cases including aerodynamic coefficients for aircraft manoeuvres. They compared a number of sampling methods and using error estimations, determined which were the most accurate and the quickest to converge by dedicating a portion to exploration and some to refinement of non-linearities.

Other common sampling methods are the factorial methods including full and fractional factorial[62] . Some less common ones are also available, such as the Delauney sampling method[63] . This method treats the design space like an unstructured grid, dissecting it using the Delauney triangulation method into simplexes that are repeatedly refined to the optimal refinement for the objective. The Morris-Mitchell criterion is another method that uses a criterion that best satisfies an exchange technique iteratively[43] .

1.3.3 Optimisation and its Challenges

There are several difficulties involved in engineering optimisation, some of which are[64] :

- *Unevaluable and infeasible points/regions*: Not all points in the space are legitimate designs - inevaluable points cause the simulation to crash and infeasible points are not physically realisable designs. This is addressed via constraints.
- *Expensive evaluation*: Simulators are usually designed for accuracy more than efficiency and therefore they can take a non-negligible amount of time to evaluate a point. A variety of methods, efficient and more accurate, should be used to counteract this. For example the use of both low order models and high order models or Time-marching and Harmonic Balance techniques.
- *Badly structured space*: The structure of the space of good designs may be difficult to search. Therefore, the use of global optimisation methods should be used.
- *Multiple local optima*: The search space may have a large number of local optima and this can trap gradient-based local optimisers and even global optimisers. They may be true local optima or apparent local optima such as those caused by noise, (e.g. due to round-off errors, approximations in the model or in the grid, inexact solvers and so on). Engineering intuition and the use of methods such as Genetic Algorithms counter this.
- *Non-smoothness*: This often results from table look-up or numerical problems which usually appear as discontinuities in objective functions. Metamodels can be used to smooth surfaces leading to better results.

The second point is very true of rotor optimisation. Hence, work on rotor optimisation has tended to lean towards gradient-based optimisation techniques since the number of solutions required for non-gradient based optimisation is high, which increases the total computational cost of optimising helicopter rotors. For example, Le Pape and Beaumier's[35] attempt to maximise Figure of Merit (FM) of a rotor by modifying the geometry of the blade in an automated routine used

CONMIN[65] , a gradient-based optimiser developed by Vanderplaat.
However, non-gradient based methods, especially evolutionary type methods, have been used more for other types of rotors such as compressor and turbine blades, and usually for a single point condition.
For example, in Zhao[66] , the optimisation of turbines is performed using a neural network trained model and a GA. The original data was obtained using a Navier-Stokes solver. Bézier curves were used to define the geometry and their control points were used as design parameters. In Samad[39] , a Navier-Stokes RANS code was coupled with a multi-objective GA to find the optimum blade shape of a transonic axial compressor rotor to improve adiabatic efficiency and total pressure ratio. Again, Bézier curves were used for the definition of the blade geometry. In this case, the GA was used to find a global optimum region and then a refined search was made using Newton's method, which is gradient-based. In addition, clustering was used to find the Pareto-optimal solution or front. Mengistu and Ghaly[38] also used a GA to optimise turbomachinery blades. In their case, the performance is measured by the adiabatic efficiency and its total pressure coefficient. The design variables include the back pressure and the shape parameters. The first term in the optimisation function attempts to maximise the efficiency or minimise the total pressure loss coefficient, while the second term would eliminate large differences in the above between the design and off-design points. The last term is a penalty term including geometric and aerodynamic constraints such as the exit flow angle, spacing to chord ratio and stall margin. The weightings of the penalty coefficient are set by the user. They were successful in obtaining an optimised blade.

Evolutionary methods have also become attractive for fixed wing aircraft. For example, Watanabe[67] uses Euler simulations with a GA to optimise a passenger plane wing for shock waves, that is the lift-to-drag ratio. Constraints were applied such as the thickness at the root, the reference area, the minimum volume of the wing or fuel cells and the range of angle of attack. Steady state CFD calculations of rotors also attract evolutionary optimisation methods. For example, Liu et al.[68] used an extended compact genetic algorithm (ECGA) to optimise Horizontal Axis Wind Turbines (HAWT) rotor blades for maximising power by modifying twist and chord.

In terms of helicopter rotors, evolutionary type optimisation was applied by Glaz et al.[27,28] in the optimisation of forward-flying rotors for vibration and noise reduction. Sun et al.[49] used GAs to optimise rotor sections in pitch-translation motions for forward flight simulation.
Allen, Rendall and Morris[29] started off with aerofoils and fixed wings, but applied their methods to optimise rotor twist in hover. Their method directly modifies the design shape and its meshing and one of the key features of their method is their parameterisation technique, which is linked to RBF interpolation allowing local and global control of the grid coordinates. The optimisation itself is based on a gradient-based parallel optimiser. The differences between having local and global control over the mesh and shape deformation is highlighted. The process, even for hover of a relatively coarse grid, was lengthy. Application of such a method to forward flight might render it impractical.
Bohorquez et al.[34] tried to reduce the computational load by using a combination of blade element momentum theory (BEMT) coupled with lookup tables and experimental data to optimise a small rotor in hover for better power loading.

Typically, the rotor blade design problem is also complicated by conflicting performance requirements between hover and forward flight[35] . Le Pape's[36] method was a very versatile method, optimising for anhedral, sweep, twist, chord and aerofoil distribution of rotor blades, where these variables could be optimised for the whole blade or only part of it. In hover, collective pitch was an added variable so that thrust could be varied and maximum FM could be achieved. In the multi-objective optimisation process, one flight condition was optimised for whilst the other acted as a constraint. For example, for the 7A ONERA rotor, twist was optimised in hover, whilst constraining performance in forward flight.

Imiela[37] used a Genetic Algorithm to optimise the rotor and the MATLAB DACE kriging toolbox as a metamodel. Hover and forward flight were both considered whilst constraining structural integrity within boundaries. A number of optimisation algorithms were tested - CONGRA, a conjugate-gradient based method (which works by using gradients as well as finding new search directions based on former iterations), SUBPLEX - a non-gradient based method - and EGO, which uses a metamodel in a GA. The conclusions were that CONGRA does not reach the optimum fast enough, and that it was dependent on the step size. SUBPLEX had difficulties with the tip optimisation, almost producing a rectangular blade. This may be due to issues with accuracy. EGO was the most efficient and the least-error prone, which suggests that metamodels are promising for rotor optimisation.

What to optimise for?

The design objectives are usually specified as an objective function containing the relevant performance parameters as components that are weighted. In some cases, the best compromise is found between all the performance parameters and a design is selected from this subset. One example is the Pareto front method. Other methods also include goal programming[69], where instead of finding a point that is the maximum or minimum of an objective function, a solution is found that satisfies a goal or reaches it with minimum distance and satisfying constraints. The paper by Deb[69] describes this in more detail, including a method independent of weight factors.
Where objective functions are used, they should have the following qualities[4]:

- Complete, so that all pertinent aspects of the decision problem are presented, i.e. all the objectives are captured for example, stall on the retreating side, compressibility effects on the advancing side, pitching moments and so on.
- Operational, in that they can be used in a meaningful manner i.e. the optimiser must be able to access the performance and make a decision based on that.
- Decomposable, if disaggregation of objective functions is required or desirable. For example, the rotor with the best advancing side performance should be obtainable by adding or removing components from the objective function.
- Non-redundant, so that no aspect of the decision problem is considered twice. Sometimes it is possible that two components of an objective function capture the same performance metric in addition to others. It is important to make sure that this does not happen as it would result in one performance parameter being weighted more than intended.
- Minimal, such that there is no other set of objective functions capable of representing the problem with a smaller number of elements. This avoids the above problem as well. For example, moment curves on a blade over a revolution can capture pitching moment performances as well as stall and shock effects. Therefore, the use of, for example, the drag curve to capture the stall effects is unnecessary and would result in redundancies and added unnecessary workload.

K. and S. Hall[70] attempted to find the optimum radial and azimuthal distribution of blade element thrust that would give minimum rotor induced drag, similar to reducing downwash for fixed-wing aircraft where elliptic loading is the most ideal. With increasing forward speed, the induced power (in the form of induced drag or torque), first decreases and then increases sharply because the rotor must remain trimmed (especially in roll) even though the region of low speed and flow reversal on the retreating side increases. Their aim was to find the optimal distribution for low induced power by optimising the circulation distribution independent of the rotor geometry and also reducing the gaps in the wake on the retreating side at moderate advance ratios (μ). In their paper, they present the theory and a computational model for finding minimum power and the circulation distribution for that minimum power.

In the work done by Le Pape[36], maximising the Figure of Merit (FM) was the objective in

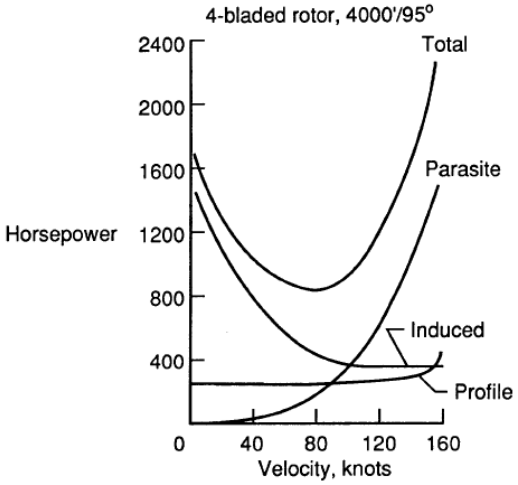
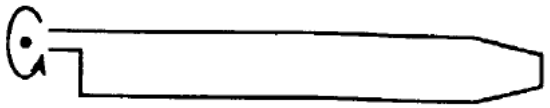


Figure 1.3: Figure showing power as a function of velocity of forward flight[2] .



Blade by Walsh[2]

Blade by Le Pape[35]

Figure 1.4: *Figure comparing blades optimised for different objectives.*

hover. In forward flight, torque coefficient (C_Q) was optimised for while constraining the thrust coefficient (C_T). The same objectives were used in Imiela's work[37] , with the addition of constraining the propulsive force in forward flight.

In Bohorquez et al.[34] , a small rotor was optimised in hover for power loading rather than FM. This parameter was actually the maximum power loading envelope obtained at various disk loadings so that the results were independent of disk loadings.

Hajela and Lee[71] performed multidisciplinary optimisation, and for forward flight, the objective was to avoid stall while maintaining trim conditions and power used i.e. torque. In addition, they had limitations on blade frequencies, pitching and rolling moments, hub shears, aeroelastic stability margins, autorotation index and other material constraints.

In the Blue-Edge design[72] , the objective was to actively reduce BVI noise while constraining forward flight performance. This also happened to be one of the first cases where high-fidelity CFD was used in the optimisation loop to obtain better hover performance by modifying the anhedral and twist of the blade.

The objective function should encompass all the objectives as well as the constraints to obtain the required performance. For example, the objective in Walsh's paper[2] was to reduce the required power in hover without compromising forward flight, i.e. for the total power curve of Figure 1.3 to be lowered. The resulting optimised blade had very different characteristics from that obtained by Le Pape[35] , for which the objective was the increase of FM and only considered hover as shown in Figure 1.4.

Gradient-based Optimisation

Gradient-based methods use the gradient of a performance function in order to find the best direction that leads to the optimum solution. One of the more commonly used pieces of software is *CONMIN*, developed by Vanderplaat[65] at NASA. It has been used by a number of authors[35] . The memorandum by Walsh[2] states that the CONMIN optimiser was used to optimise the aerodynamic performance of rotor blades by selecting the point of taper initiation, root chord, taper ratio and maximum twist, which minimise hover horsepower while not degrading forward flight

performance. CONMIN was also used by Celi[17] for aerodynamic and vibration reduction optimisation.

CONMIN can optimise constrained linear and non-linear problems. It uses finite difference methods to find the gradient and can also use conjugate direction methods by Fletcher and Reeves[73] for unconstrained minimisation problems. The optimisation can also be performed without any conditions of constraint. Also, in order to test other designs that may not otherwise be found, constraints can be added after an optimum is found and continued from that point. Such an option may be desirable when minimising functions for which one or more constraints are difficult to evaluate.
It works by iteratively updating the design so that at iteration i, $\bar{x}^i = \bar{x}^{i-1} + \alpha \bar{s}^i$ where $\bar{s}$ is a vector direction and α is the distance in that direction. $\bar{s}$ is determined so that for an arbitrary small α, the optimisation function is reduced (since one is trying to find minimum) the most, i.e. steepest descent, and no constraints violated. If it violates a constraint, a direction $\bar{s}$ is found to overcome the constraint with minimal increase in the optimisation function[3]. Figure 1.5 is a summary how the CONMIN method works for three points (A, B and C) under various constraint scenarios.
CONMIN works efficiently if there are few constraints and relatively smooth varying data. In the case of total aerodynamic rotor optimisation, where the constraints are much more and the data uneven, it may be less efficient and may fail to reach the global optimum. There is a high probability of the solution getting trapped in local optima.

The tendency of gradient-based methods to stagnate near local optimum regions of the de-

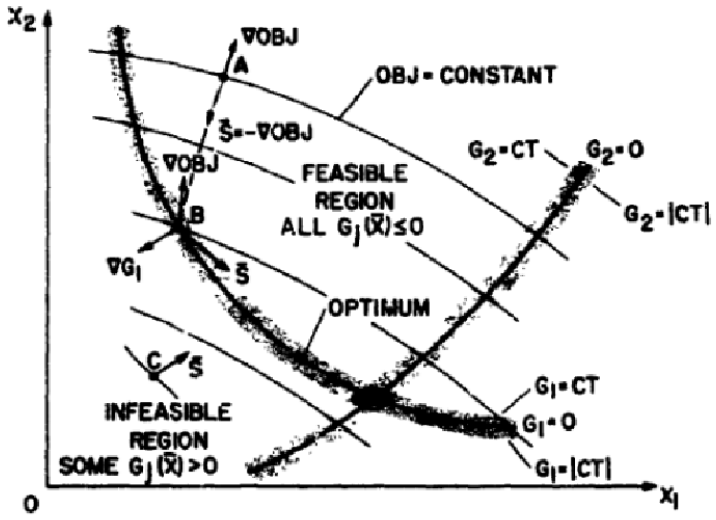


Figure 1.5: *Figure showing how the gradient-based optimiser, CONMIN operates. Three points, A,B and C are shown in various locations with regard to the constraints. At A, steepest descent is used to reach the minimum, at B, a constraint is encountered and the point attempts to move along the constraint contour and at C, the point attempts to overcome the constraint.*[3]

sign space is a major problem. This is why in the work carried out by Walsh[2], it was found that the starting design point had a significant role in the final geometry obtained. Therefore, it was necessary to start the optimisations from various points in the search space so that a number of optima are obtained and the best of these was selected.

Schwabacher et al.[74] worked to overcome this problem. They used gradient-based methods to optimise various shapes such as aircraft and yachts, which used a learning algorithm based on previous optimisation procedures to determine where the best initial design point should be. Similarly, Chen and Lee[75] tried to improve gradient based methods by using a 'gradient-forecasting

search method (GFSM)' to dynamically adjust prediction steps so that it can overcome local optima.

Mohammadi and Pironneau[76] used a gradient-based optimisation method to optimise supersonic aircraft for sonic boom. They make the suggestion that in the case of multi-criteria optimisation, shape optimisation is conflicting. Therefore, the use of the Pareto optimality method attempts to improve all aspects of the body by minimising a convex combination of all criteria. However, they suggest that the argument in such linear combinations led to problems such as the existence of many sub-optima, requiring global optimisation tools such as Genetic Algorithms (GAs), which are simple but very slow and hence propose that the solution may well be a combination of gradient and evolutionary methods.
Lehner et al.[77] also propose a hybrid-optimisation technique for multi-disciplinary optimisation of a subsonic aircraft. The optimiser was a genetic algorithm non-gradient based one, but at smoother points in the design space and where the data is continuous, a gradient based method was used.

Non-gradient based optimisation is expensive especially for cases where each solution is costly, such as forward flight rotor optimisation. This is why Le Pape[36] for example, used a GA for hover optimisation but used CONMIN for the forward flight optimisation. This again re-iterates the importance of using lower order metamodels to improve non-gradient based optimisation efficiency.

Another option is to use codes such as ADIFOR that can find derivatives analytically and semi-analytically and hence considerably reduce computational time[17] .

Non-gradient based Optimisation

These methods do not rely on the computation of gradients of a function and hence are not trapped in local optima. This is their greatest advantage and they therefore belong to a group termed global optimisation techniques. Although the convergence to a global optimum is not guaranteed, the probability of it occurring is very high. Also, new designs in the search space are easily introduced, increasing the chances of a truly global optimum design for the required objective.

Typically, these methods consist of a performance metric and a selection phase based on this metric. The added advantage is that sample points can be created and combined with other points in the population so new designs are easier to create than with gradient-based methods. Some examples are statistical methods, genetic algorithms, evolutionary programming, simulated annealing (although it does involve some discretised determination of gradients), branch and bound etc.

Genetic Algorithms (GAs) simulate the evolutionary process involved in natural selection i.e. that an initial population exists from which newer generations are created, with each new generation increasing in 'fitness' by the processes of crossover and mutation within the individuals.

Another method similar to GAs is Simulated Annealing (SAs)[64] . SAs are based on an analogy with the way metals cool and anneal. They start from an initial point and move from one point to another until they reach an optimum. At each iteration, a new point replaces the current point dependent on a probability, P. If the new point is better than the current point, P=1, if not, it has a lower value. This still gives it the chance of trying new paths and subsequently escaping local optima. At the start, the probability of moving to a worse point is high, but decreases as the optimisation progresses.
In comparison, SAs differ in that they follow a path, whereas GAs combine points from various regions in the search space to obtain new solutions. GAs also have a high inertia in switching attention to a new area compared to SAs. The best method is dependent on the problem. GAs

maintain a population of points rather than one, and they hence are beneficial in that way. SAs are useful where the optimisation problem is large and has many local optima around the global optimum[78].

Badyrka et al.[79] carried out a study comparing three optimisers; Genetic Algorithms (GAs), Repulsive Particle Swarm Optimiser (RPSO) and a direct geometric search algorithm (VTDirect). The RPSO method is relatively new and was developed by Mishra[80]. In the PSO, the data points simulate the individuals (or particles) of a swarm. Each has a position and velocity in the swarm and keeps in memory the overall best position it has been in. The individuals communicate the global or local best position to all members. Generally, it is known that the PSO methods suffer from premature convergence[81]. Therefore modified versions of this method tend to be used. In the repulsive version, it allows the particles to have a wider local search ability and prevents it from getting stuck in local optima. Particles are allowed to learn from a pre-determined number of random individuals[80]. Furthermore, particles are not allowed to move into areas outside the constraints specified. In the Regrouping version, once premature convergence is detected, the particles are automatically regrouped around the optimum found but with a smaller design space and the process is repeated[81].

The VTDirect method works by systematically subdividing the design space into quadrants. The division is such that the objective function value (OFV) is kept at the centre and the decision of how to divide is based on the size of the quadrant and the OFV. A quadrant is most likely to be divided if it is large and has a small OFV.
Badyrka et al.[79] came to the conclusion that the RPSO method has a lot of promise and can have similar performance levels to GAs. It has less control parameters than the GA. They also suggest that for more complex optimisations, it may be good to couple the RPSO with a GA.

Vytla et al.[82] employed a hybrid GA-PSO optimiser to optimise a train nose for reduced drag. They found the combined method to be more efficient and robust than either optimiser on its own. Similarly, Poloni et al.[83] used a hybrid GA and gradient-based method along with an ANN to optimise the keel of a yacht for high lift and low drag, parameterised by Bézier curves. They describe a number of Pareto-based selection methods. One of the novel methods described is a version of a multi-directional crossover. They also suggest that replacing individuals rather than whole generations is more efficient. This makes it easier to parallelise the whole procedure. Dulikravich et al.[78] also used a hybrid method that combined a number of optimisation techniques including GAs, SAs, gradient methods and the Nelder-Mead simplex method (NM) to create a more robust optimiser. The method switches between these techniques where they work best; for example the gradient method is employed when the variance from the GA's output is small and the NM method is used when the random behaviour of the GA is prevalent. This is controlled by a set of rules which can be found in further detail in the paper. The rules determine which method is the most efficient and accurate based on aspects such as the diversity of the selection and so on.
Other methods are population based incremental learning, Ant Colony Optimisation etc.

Régnier et al.[84] state that there are three classifications of non-gradient based optimisation:

- Apriori - Decide and Search: the decision maker decides what is required and the solution is found e.g. the use of a weighting system. Only one Pareto solution is found.
- Progressive and Sequential - Decide and Search: Optimisation and decision-maker are intertwined. The preferences are sequentially updated. Multiple runs are required to obtain Pareto solution.
- Aposteriori - Search and Decide: A single optimisation run provides a set of solutions that the decider can choose from e.g. Evolutionary Algorithms (EAs).

The last one is of particular interest to this project since no target performance is required. Rather, the best design within the constraints on performance is required. Both the weighted sum and the Pareto method were used. The advantage of using a weighted sum is that it selects a small region in the database where the optimum can be found. The design variables do not change much in this area. However, this reduces the flexibility of finding another such region unless the weights are changed. Therefore, the Pareto method is used to find the best compromise of the performance parameters and typically, the results of the weighted sum method should lie somewhere on the Pareto front[85]. The advantage of using a Pareto method is it allows the designer to see the best compromise before making the decision. However, this can still be a difficult task if the number of design variables is high and if the properties of the Pareto subset are not properly analysed. More information about this can be found in the paper by Daskilewicz and German[86].

Some other search methods such as Tabu, Nelder-Mead and Powell's method are given in Hajela's paper[50]. Also, the paper by Sobieski et al.[87], although relatively dated, has some useful insights into the various methods used in multi-disciplinary optimisation.
The cost of running high-fidelity software in order to find optimum solutions usually prevents the use of global optimisation techniques. Therefore, rotor optimisers usually consist of an approximate model or metamodel mentioned in Section 1.3.1 (known as Surrogate-based Optimisation (SBO)), which is used to predict the behaviour of flow without the use of the high-fidelity software, such as a highly accurate flow solver. Once the optimal solution has been reached, the high-fidelity solver is used to validate the optimum solutions.

Methods involved in Genetic Algorithms

The performance metric (mentioned in Section 1.3.3) for GAs, is called a *fitness* value. The success of a GA depends on its ability to perform a balanced amount of exploration as well as exploitation of the promising regions. A good GA usually favours exploration in the beginning of the search and gradually shifts to exploitation[64]. In general, the GA has a number of stages. It begins with a population of individuals from which a selection of two parents are made based on their fitness value. These parents are crossed over to create offspring that have a new combination of the design parameters from their parents. Mutation might occur, which alters the offspring. Based on the fitness value of the offspring, it survives into the next generation.

The fitness value is determined by the objective of the optimisation and can be obtained based on a number of different methods, some of which are given in Tan et al.[4]. They performed a survey of various evolutionary methods for MO (Multi-objective) optimisation and compare them quantitatively and qualitatively. A list of existing methods is given in Table 1 of that report. This table is included here as Table 1.1.

Evolutionary Methods for MO	MO Handling Techniques	Other Operators Applied
Charnes and Cooper (1961)	Goal programming.	
Ijiri (1965)	Goal programming.	
Jutler(1967)	Weighted min-max approach.	
Solich (1969)	Weighted min-max approach.	
Fourman (1985)	Lexicographic ordering, starting with the most one and proceeding to the order of importance of objectives.	
Schaffer (1985): VEGA	Main population was divided into sub-populations and selection was performed according to each objective function in each sub-population.	Proportional selection.
Goldberg and	Simple Pareto domination scheme.	Sharing on whole population.

Continued on Next Page...

Table 1.1 – Continued

Evolutionary Methods for MO	MO Handling Techniques	Other Operators Applied
Richardson (1987)		Proportional Selection
Allenson (1992)	Sex was used to distinguish between two objectives and was assigned at birth.	
Chen et al. (1992)	Transformation of qualitative relationships between objectives into quantitative attributes for an appropriate weight of each objective in a way similar to linguistic ranking methods. The weight generated can be used with an aggregating approach or Pareto-ranking.	
Hajela and Lin (1992): HLGA	Weighted-sum method was used for fitness assignment. To search for multiple solutions in parallel, the weights were not fixed but encoded in the genotype instead.	The diversity of the weight combinations was promoted by phenotype fitness sharing. Mating restriction was employed for faster convergence and better stability.
Jakob et al. (1992)	Linear combination of objective functions.	
Fonseca and Fleming (1993): MOGA	Pareto domination scheme. It was extendable for single set of goals and priorities.	Sharing on whole population. Mating restriction was employed for faster convergence and better stability.
Wilson and Macleod (1993)	Goal attainment.	
Adeli and Cheng (1994)	Use of Penalty function.	
Horn et al. (1994): NPGA	Use of Penalty function, Randomly-selected comparison set was used to determine the winner of the competitors.	Phenotype sharing was applied if the competitors end in a tie in domination. Tournament selection.
Ritzel et al. (1994)	Minimizing one objective function while considering other objective functions as constraints.	Reduction to a single objective.
Srinivas and Deb (1994): NSGA	Several layers of classification of the individuals according to domination were applied.	Sharing on dummy fitness value in each layer. Proportional Selection
Sandgren (1994)	Goal programming.	
Murata and Ishibuchi (1995): MIMOGA	The weights that are attached to the MO functions were not constant but randomly specified for each selection.	A tentative set of Pareto-optima was stored and updated at every generation. A number of individuals were randomly selected from the set and used as elite individuals.
Vemuri and Cedeno (1995)	Individuals were ranked for each objective. Total ranking for each individual was then determined by the sum of all the ranking numbers.	Similarity among individuals was used during selection and replacement.
Coello (1996): Monte Carlo method 1	Space was explored twice, first searching for ideal vector and then searching for min-max optimum.	Mating Restriction was applied. Contraints were handled by death penalty method.

Continued on Next Page...

Table 1.1 – Continued

Evolutionary Methods for MO	MO Handling Techniques	Other Operators Applied
Coello (1996): Monte Carlo method 2	Space was explored only once, and Pareto set was generated while searching for ideal vector. Then this set was analysed to check min-max optimum.	Mating restrictions were applied. Sharing was used to overcome high selection pressure. Min-max tournament selection was applied.
Greenwood et al. (1996)	Compromise between no preference information (in the case of pure Pareto rankings) and aggregation methods like the weighted-sum to perform imprecise ranking of attributes.	Sharing was employed to distribute the Pareto-front. Tournament selection.
Kita et al. (1996)	Use of concepts of entropy and temperature, combined with Pareto-based ranking technique, in selection.	
Sakawa et al. (1996)	Fuzzy goals of objective functions were qualified by eliciting linear membership functions.	
Viennet et al. (1996)	Each function as separately optimized. The populations from each run were processed and set Pareto-optima was obtained via elimination of Pareto inefficient points.	Elitist selection was employed when optimizing each function separately.
Bently and Wakefield (1997): SWR	Linear combination of objection functions which have been converted into ratios by using the best and worst solution in the current populations.	
Bently and wakefield (1997): SWGR	Sum of weighted global ratios.	
Bently and Wakefield (1997): WAR	Weighed average ranking.	Non-generational selection. Fitness of an individual was counted increasingly. Crossover and mutation to produce new individual which substituted the worst individual.
Lis and Eiben (1997): MSGA	Generalized version from (Allenson 1992) where the sex was not restricted to male and female.	Multi-parent crossover was applied for recombination, requiring one parent from each sex. A solution was represented by a string, like in classical GA, and the sex market.
Marcu (1997)	Adaptation of goal and use of goal values in Pareto-ranking to direct the search towards the middle region of the trade-off.	
Fujita et al. (1998)	Multiple functions were unified into scalar function so that 1 for every Pareto optima while other non-Pareto optima has value equal to how far it was from a set of Pareto optima.	Sharing was employed to the scalar function. The algorithm limited the crossover between a pair of similar solutions.
Jaszkiewicz(1998)	Weights are randomly selected. Then a temporary population is created composed of best known solution on the current weights.	Each individual in the population is optimized locally according to the randomly selected weight function.

Continued on Next Page...

Table 1.1 – Continued

Evolutionary Methods for MO	MO Handling Techniques	Other Operators Applied
Laumanns et al. (1998)	Predators were applied to chase the prey (candidate solution) according to one of the objectives. As there are several predators with different selection criteria, those prey individuals, which are best with respect to all objectives, are able to produce more.	
Voget and Kolonko (1998)	The method is similar to goal attainment, except that membership functions are used to express goals in vague terms.	
Cvetkovic and Parmee (1999)	Transformation of qualitative relationships between objectives into quantitative attributes for an appropriate weight of each objective in a way similar to linguistic ranking methods. The weight generated can be used with a aggregating approach or Pareto-ranking.	
Hiroyasu et al. (1999)	Simple Pareto domination scheme.	Population was divided into several islands where simple genetic operations were performed in each island. After certain generations, migration was performed. When the size of the frontier solution exceeds a criterion, sharing was performed.
Knowles and Corne (1999): PAES	Simple Pareto domination scheme.	Based on (1+1) evolution strategy. Local search was used from a population of one but using a reference archive of previously found solutions to store Pareto-optima and to identify dominance ranking of candidate solutions. Tracking the degree of crowding in different regions of solution space to spread reference archive.
Romero and Manzanares (1999): MOAQ	A family of agents for each objective and each family tried to optimize an objective considering the solutions found for other objectives.	
Sait et al. (1999)	Fuzzy goal-based cost computation measure combined with fuzzy allocation scheme was applied. It used fuzzy rules and membership functions to combine multiple objectives and added controlled randomness in placing a cell on an empty location within a fuzzy window.	
Tagami and Kawabe (1999)	Based on a Pareto neighbourhood search method on the basis of distribution in objective space divided into pre-specified regions.	
Tan et al. (1999): MOEA	Pareto domination scheme, extendable for soft/hard goals and priorities and even set of goals and priorities.	Dynamic sharing on whole population. Switching Preserved Strategy (SPS) for elitism. Tournament selection.
Zitzler and Thiele (1999):SPEA	Simple Pareto domination scheme. Fitness of solutions was determined only from solutions stored in the external non-dominated set.	Use of clustering to reduce the number of non-dominated solutions stored. Non-dominated solutions found so far were store externally. Binary tournament selection.

Continued on Next Page...

Table 1.1 – Continued

Evolutionary Methods for MO	MO Handling Techniques	Other Operators Applied
Andrzej and Stanislaw (2000)	Simple Pareto domination scheme.	Constraint tournament selection where functions are evaluated only for feasible solutions.
Knowles and Corne (2000): M-PAES	Simple Pareto domination scheme.	Based on PAES (Knowles and Corne 1999), but uses a population of solutions and employs crossover. Besides the main population, two archives were required for elitism.
Mariano and Morales (2000): MDQL	Each agent proposes a solution for its corresponding objective function. Solutions were then evaluated using non-dominated criterion and solutions in the final Pareto set were rewarded.	
Rekiek et al. (2000)	Use of preference ranking organisation method for enrichment evaluation (Brans et al. 1986).	
Sefroui and Periaux (2000)	Based on non-cooperative game theory to find Nash equilibria.	
Khor et al. (2000): IMOEA	Pareto domination scheme, extendable for soft/hard goals and priorities and even set of goals and priorities.	Dynamic population size based upon on-line discovered Pareto-front for desired population distribution density and Dynamic local fine-tuning to achieve broader neighborhood.
Khor et al. (2001): EMOEA	Pareto domination scheme, extendable for soft/hard goals and priorities and even set of goals and priorities.	Tabu list and Tabu constraint were used for individual examination and preservation. Lateral interference for uniform distribution.

Table 1.1: *Summary of existing evolutionary methods compiled by Tan et al.*[4]

For each method in Table 1.1, a comparison is made for the type of method used. Figure 1.6 from this report, is a comparison of the trends in each of these methods. As can be seen, the most popular schemes used over the last few years (up to 2001) are Pareto, ranking, and weights. Overall, the report states that methods that used Pareto, ranking, goals and preference, preserved important non-dominant individuals via the method of switching preserved strategy. Also, the distribution along the Pareto front and the robustness in a noisy environment was more or less the same. It was also observed that elitism was found to be an important strategy for obtaining good performance. Elitism allows high performing individuals to enter the next generation without the selection, crossover or mutation operations. These methods are still the most popular methods used today in Genetic Algorithms. A few newer methods that are as robust and have good performance have been developed. An example is the conservation technique called crowding described in the work by Hirsh[64] . Crowding takes into consideration other as well as fitness values, such as diversity preservation etc.

A number of methods exist for selecting the parents from a population. The aim of the selection process should be to select individuals who are generally fitter so that the pressure is directed towards better performing individuals and hence generations. One of the simplest and most straightforward methods is the roulette-wheel method where the probability of selection is proportional to the fitness of the individual. Another method is the rank-based method which is more useful when there is a wide range of fitness values. Here the individuals are ranked and their

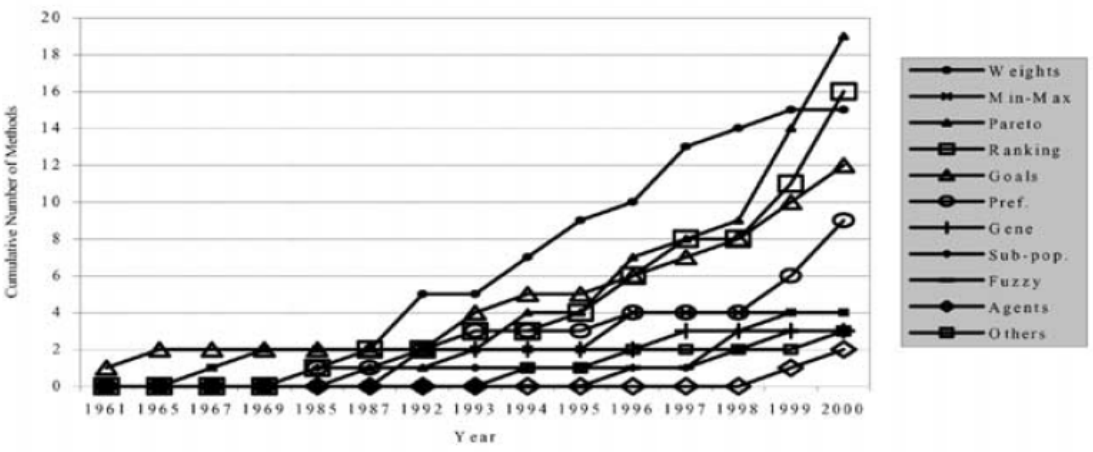


Figure 1.6: *Comparison of the methods used in optimisation techniques developed in the last 50 years. It can be seen that ranking, Pareto and weights are the techniques that have become most popular.*[4]

probability of selection is proportional to their rank rather than their fitness value.
There are a number of crossover methods as well. Usually the design parameters are converted to binary format (the gene) and then a random point is selected where the gene is split and part of it swapped with the other parent. Floating point methods also exist where any one design parameter is swapped with that of the other parent. Other methods include heuristic crossover, linear crossover, arithmetic crossover and so on[64]. In the case of the optimisation of the turbomachinery blades, Mengistu and Ghaly[38] used two methods of crossover - arithmetical where two parents are linearly combined, and heuristic which uses the fitness of the parents to determine the search direction and then creates the new offspring.
Similarly, there are numerous ways that mutation can be applied. The simplest is to randomly switch a random bit of the binary format of the design parameter. Other methods are more progressive where the probability of mutation occurring drops as the optimisation matures[64]. Mutation can create new designs or reintroduce old designs in the gene pool and improve the diversity of the design search space or gene pool. This is important because after many generations, selection could drive all the bits into 1 position (using the example of binary coding - to either a 1 or a 0). Mutation allows you to reintroduce the missing bit. Also, it allows retesting. However, the number of mutations must be kept to a minimum to reduce randomness[88]. Sometimes, binary representation can mean that there are big changes in the binary form for small changes in parameter. Therefore, in these types of cases, the *Gray code* or *reflected binary code*[89] is sometimes used where two successive parameter values differ in a single bit in their binary form, i.e. instead of counting up from 7 to 8 for example, which results in a binary change of 4 bits (0111 to 1000), this code allows a conversion of only one bit (0111 to 0101) which translates to counting from 7 to 5.

Constraints can be applied to the resulting offspring. One approach is to treat the gene as a faulty gene and try to repair it[50]. A simpler approach is the penalty method where the fitness value is diminished based on the constraint that has been violated and by how much it is violated. In some cases, if a constraint is violated, the offspring is removed from the new population i.e. a hard constraint, otherwise its fitness value is compromised, but it is allowed to join the new population based on its new fitness value i.e. a soft constraint[50].

Hirsh created a design optimiser called GADO (Genetic Algorithm Design Optimiser)[64] used for engineering design. The document describes a method of improving the search for the optimum function in a genetic algorithm (GA) using GADO.

These are some of its features:

- *Screening Module (SM)*
  The idea of the SM is to predict the merit of a proposed design before the simulator is used to evaluate it.
  The SM selects a point that is promising (using extrapolation and not simulation). Before a point is selected, the SM finds the K nearest neighbours (usually two, but this is dependent on the sharpness of the optimum) and checks if it is above a certain threshold value before adding it to the new sample population.

- *The Diversity Maintenance Module*
  This ensures that the selected points are not very similar to each other such that it does not represent the search space. It prevents being trapped in local optima. If at a certain point it is found that the points are quite similar, the module rebuilds the population using previous points.

- *Guided Crossover*
  - It forms search directions without computing gradients, done by joining pairs of points, ranking these directions and taking a step in the best direction. Directions are ranked by the ratio of the difference in the fitness to the distance between them. It works like this:

    – One point is selected, c1.

    – For every other point in the population, a mutual fitness is calculated as below:

    $$Mutualfitness(A,B) = \frac{(fitness(A) - fitness(B))^2}{Euclideandistance(A,B)^2} \tag{1.3}$$

    A choice for the second point, c2 is made by maximising the mutual fitness.

    – c1 and c2 are swapped if necessary so that c1 has the highest fitness.

    – The result of the crossover is a point along the line joining c1 and c2 selected at random from the small region around c1, the better point as follows:

    $$Result = L \times c1 + (1 - L) \times c2 \tag{1.4}$$

    where L is a uniformly distributed random number in the interval *[1-0.2x,1+x]* and $x = 0.75 \times (1 - n) + 0.25$ where n is number of iterations. Nevertheless, this should not be used as the only crossover operator[64] . There are many other methods such as the heuristic method:
    If the parents are $\bar{X}$ and $\bar{Y}$ such that $\bar{Y}$ is fitter than $\bar{X}$, then the new born is $\bar{Y} + r \cdot (\bar{Y} - \bar{X})$ where r is a random value selector between 0 and 1.
    Other methods include point, random (which produces a lot of diversity), arithmetic and linear.

- *Dynamic Penalty*
  This GA uses an 'adaptive penalty' approach for handling constraints. When the GA focuses too much on feasibility it reduces the penalty coefficient and vice versa.

A number of selection strategies are described in Hirsh[64] two of which are:

- Fitness proportional (roulette wheel) selection
  The probability of an individual being selected is dependent on its fitness.

- Rank-based selection
  Each individual's probability of being selected depends on its fitness rank rather than its actual fitness. The most commonly used methods of carrying out this selection are:

- Tournament selection: Two candidates selected and the fitter of the two is selected for reproduction.
- Weight series selection: Each individual is assigned a weight dependent on fitness rank (weights usually taken to be a decreasing arithmetic or geometric series). Proportional selection is then performed using the weights rather than the actual fitness.

Rank-based is considered to be more appropriate for use in domains where the fitness range is extremely wide.

In steady state GAs, each new individual replaces an old individual, so the population remains constant. In generational GAs, the entire population or a portion of it is replaced simultaneously. Steady state converges faster than generational.

Two mutation operators are described:
Uniform mutation: It replaces each component of a solution vector with a random value uniformly selected from the component range.
Non-uniform mutation: At the beginning of the optimisation it acts like uniform mutation. It then becomes more and more conservative about the amount of change it makes to a vector component as the optimisation progresses.

Toffolo and Benini[90] also did some work on improving MO optimisation by improving the diversity of the gene pools created by GAs. In their version (GeDEM), genetic diversity was a real objective in itself, measured as variable or optimisation function distance to neighbours. Benini also used GeDEM to optimise calibration of centrifugal pumps[91] . One of the aspects of the GA used was in the selection of parents. A reproduction pressure was included that selected fitter individuals to crossover, but this pressure varied with the generation so that at the start of the optimisation, the pressure allowed more diversity, but towards the end, more exploitation of the best genes rather than exploration of the design space took place.

Another advantage of GAs is that they can be easily parallelised to reduce the time that it can take[92] . González et al.[93] created a parallel computing modified GA called the hierarchical asynchronous parallel evolutionary algorithm (HAPEA) which proved to use slave machines more efficiently. It also created a number of populations at each generation instead of a single population[93] .

Other Optimisation Methods

Hassan and Charles[94] used a reverse-design method to design aerofoil blades for a hovering helicopter. Here the pressure distribution was specified by the user and the aerofoil geometry that best produced this distribution was created for each radial station along the blade span. CAMRAD/JA was used to compute the rotor trim state as well as the far-wake-induced incidence and the actual modification and analysis of the geometry was carried out by the solver RFS2 which had a built-in ability to generate new grids of the geometry created at each iteration. This concept could work well, but it is necessary for a three dimensional approach to be taken rather than the aggregation of many 2D sections to form a blade.

Another method of optimisation or design that has generated a lot of interest is Adjoint-Base Optimisation. This method computes the sensitivity of the shape to the objective. Jameson et al.[95] use an adjoint approach where a target pressure distribution is specified for a fixed wing and the Navier-Stokes equations are used to iteratively correct the shape function to obtain the required distribution by reducing a constrained cost function by gradient-based optimisation. Mani et al.[96] developed an adjoint based method to optimise aerofoils for dynamic stall. The objective was to reduce the peak pitching moment while maintaining lift. This was successfully carried out

with considerably less computational effort than performing a time-dependent simulation for the same amount of flexibility in the design. Nielsen et al.[97] applied an adjoint based method for the design of rotors in hover using Navier-Stokes equations with the aim of creating a general time-dependent adjoint-based optimisation method for rotorcraft. The results were comparable with those of independent approaches and showed improvement in Figure of Merit.

Liu et al.[98] also used a reverse engineering method to design rotor blades for HAWTs. They modified the twist, chord and aerofoil in the form of Bézier curves for each section along the blade and merged these 2D designs smoothly along the span. The strip theory was used taking into account hub and tip losses, cascade effects etc. A good blade design was obtained that had an overall increase in the output power. However, for a more accurate prediction and hence a more validated result, it is important to optimise taking into account the effects of the flow field over the entire 3D region. Inverse design methods have been explored for a long time. Lekoudis and Sankar[99] used an inverse potential flow method to prescribe the skin friction coefficient and obtain a pressure distribution from it for a wing. Also, a similar method was used to design aerofoils and nacelle inlets for a target pressure distribution.

Drayna et al.[100] created a direct nonlinear sensitivity solver to optimise a scramjet inlet. It can be incorporated into an existing solver without having to rewrite any of the original solver code, which is what would be required in the case of an adjoint sensitivity solver. It is more accurate than a linearised solver and also requires no new boundary conditions.

Fuselage Optimisation

A small part of this project was dedicated to fuselage optimisation. This is because a significant contributor to the inefficiency of helicopter flight is the fuselage, and research is dedicated to this area specifically. For example, Garavello et al.[101] used GAs to optimise parts of the fuselage of the ERICA tiltrotor using CFD. Changes to the wing-fuselage junction, the air intake and engine exhaust led to improvements in lift, drag and pressure loss. Therefore although the main focus of this project is the optimisation of rotor blades, a part of this study focuses on a method to parameterise and optimise a simplified fuselage geometry.

Flow around a helicopter component is complex. However, it becomes even more complex when the components are close to each other due to their induced effects on each other. Changes in the positioning of these components can result in large changes in the aerodynamic performance of the overall helicopter. Sa et al.[102] carried out CFD tests on the HARTII rotor as an isolated rotor as well as combined with the fuselage to determine the effects the fuselage has on the aerodynamics of the rotor. It was found that improvements were made on the capture of the blade-vortex interactions on the advancing and retreating sides and the phase difference at the front of the disk. Stronger upwash was also observed which which affected the cyclic trim angles and blade deformation. A common feature that also has a significant effect due to rotor/fuselage interactions is the distance between the rotor and the fuselage. This can cause significant variation in the inflow and dynamic loads[103]. There is a large amount of documentation on work done on rotor-body interactional aerodynamics, one of the more popular and early ones being the work done by NASA on the ROBIN (ROtor Body INteraction) body[104–106].

The ROBIN body was created by smooth transitions of ellipsoidal equations resulting in a very streamlined body, much more than expected on a typical helicopter fuselage. A more realistic modification of it using the same parameterisation method is the ROBIN-mod7[10].

The Japan Aerospace Exploration Agency (JAXA) recently also developed their own simplified version of a helicopter fuselage, one that has closer drag characteristics to a typical helicopter fuselage[107]. This fuselage however, does not have a parameterisation method. The aim therefore, is to apply the parameterisation technique of the ROBIN to the JAXA body and use it as a

demonstration of the optimisation and parameterisation method for simplified fuselage bodies.

Another example of an experimental fuselage body is the ROTEST stand used in the HART (Higher harmonic control Aeroacoustics Rotor Test) project[108] , although it was not specifically built for fuselage modelling and testing.

1.3.4 Parameterisation Techniques

Parameterisation is essentially numerical coding of the design variables to obtain a general form of representing a particular shape. It is a key part of the optimisation process. Samareh[109] states that a good parameterisation technique for multidisciplinary shape optimisation should be:

- automated
- provide consistent geometry changes across all disciplines
- provide sensitivity derivatives
- fit into product development time
- have a direct connection to the CAD system for design
- produce a compact and effective set of design variables

However, the initial main focus in this case will be the last point made. This is because the optimisation method desired is one that will not require grid regeneration until the end of the process. The coupled method has been used by a number of authors such as Le Pape[36] where for the elsA solver, an in-house analytic grid-generator was used to create a grid for each evaluation. However, in this project, the optimisation technique is to be decoupled from the high-fidelity CFD model and linked to it through a lower fidelity model.
Hàjek describes a number of techniques used to parameterise aerofoils in the light of aerodynamic optimisation[110] . This paper states that the choice of technique strongly influences the optimisation. Similarly, Castonguay and Nadarajah[111] also performed a study on four different parameterisation method for the optimisation of aerofoils using an inverse method. Below are some of the methods described in both these papers:

Mesh point approach: Here eac mesh point can be independently moved. This is one of the easiest to implement and allows a lot of flexibility in the design but can lead to discontinuous geometries and possibly too many control variables[111] .

Joukowski transformation: Consists of transforming a circle in the complex plane via the transformation

$$z = \epsilon + \frac{1}{\epsilon} \tag{1.5}$$

The circle should pass through the point $\epsilon = 1$. The aerofoil shape is controlled by varying the centre of the circle. This method was especially advantageous in the past because it enabled the plane potential flow to be analytically solved.

Splines: These use piece-wise polynomial approximations of curves. B-splines use the same idea but are slightly more complex in that they are built on linear combinations of base functions. Non-Uniform Rational B-Splines (NURBS) are a further development. Ghaly and Mengistu[112] show that Bézier curves were used because of the simplicity of their implementation. However, they are global and hence a change in a single control point changes the shape of the entire blade and so the designer has less control over local regions. To accommodate this, B-splines were used. They work just like Bézier curves but have more complex interpolation functions. These allow more local control, but are still not accurate in describing conic sections such as rounded leading edges etc. Therefore, non-ration B-splines (NURBS) were used since these allowed local control

and could also reproduce conic sections as accurately as required by industrial standards.

Hick-Henne 'bump' functions[49]: Modelling small or moderate perturbations of 'baseline' airfoil shapes used especially in inverse design. The perturbation is expressed as a linear superposition of 'bump' functions of the form

$$y = sin^t(x^\beta) \tag{1.6}$$

where β is used to control the maximum of the bump function, located at $x = \pi/2^{1/\beta}$ and t controls the width of the function (typically $t = 3$). It allows specific regions to be refined thereby reducing the number of variables and their nature ensures continuous gradients of the shapes. However, they are not orthogonal and hence are incapable of representing the full set of continous functions[111].

PARSEC[5,111]: used mainly for subsonic and transonic aerofoils. The aerofoil shape is expressed as an unknown linear combination of suitable base functions and selecting 12 important geometric characteristics as the control variables in such a way that the shape can be determined by solving a linear system with these variables. The variables are shown in Figure 1.7. The advantage of

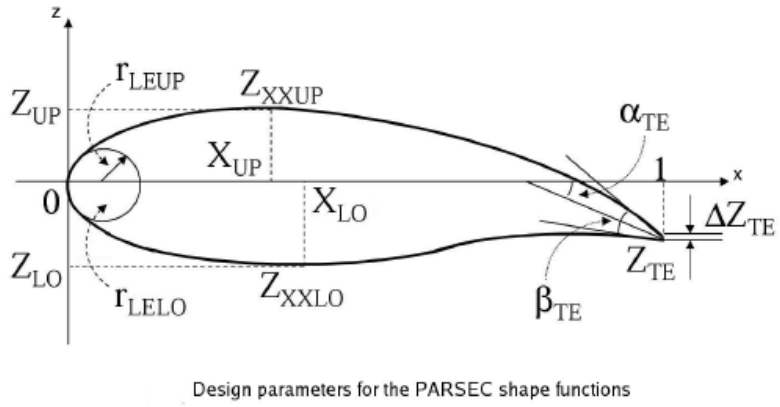


Figure 1.7: *PARSEC variables of an aerofoil for parameterisation*[5].

PARSEC is that no baseline shape is needed, a wide range of aerofoil shapes can be generated, constraints are easy to impose and the impact of individual PARSEC design parameters on aerodynamic properties can be easily predicted. The disadvantage though, is that it cannot cover as wide a range as spline curves.

B-splines, NURBS: For complex geometries, the use of Bézier curves, splines and NURBS is quite popular, especially in the optimisation of compressor blades[39,66] mainly because they allow a greater degree of freedom with the use of fewer variables and can produce smooth, discontinuity free curves. However, the implementation of constraints is not as straightforward as using numerical representations of physical measurements. Philip Schneider gives a detailed explanation of how NURB curves work and his article.[113] Joh[114] used NURBS with 36 points to fully describe the M6 wing. This shows the versatility of the technique but also the high number of parameters used to describe a simple body. In Mengistu and Ghaly's paper[112], they describe a method for finding the minimum number of control points to represent turbomachinery blades using NURBS within a specified tolerance. An objective function to be minimised is a representation of the error and the weights and control point coordinates are the parameters being optimised for. In this way, coordinate point data curves can be converted to NURB curves and vice versa. In the case of Mengistu and Ghaly[112], simulated annealing (SA) was used to optimise for the minimum

number of points.
In Samad[39] , cubic spline interpolation was used to define the blade (i.e. a separate cubic spline exists for each interval - these are popular because they produce smooth curves and are easy to implement). In total 20 variables were brought down to 6 variables.
Le Pape[36] used Beziér curves to parameterise all the distributions of the variables in rotor optimisation except for the aerofoils.
The use of splines. Bézier curves and NURBS allow localised control and reduced number of variables and the ability to represent continuous curves as well as discontinuities in the geometry. However, the link between the values and the design variables are not intuitive to the user.

Kulfan and Bussoletti[115] also carried out some work on parameterisation. They divided the parameterisation into two classes, one for the shape function such as aerofoils etc. and the other for body cross-sections such as fuselages, nacelles etc. For an aerofoil, the leading edge and trailing edge are specified and the curve in between these extremes is what is modified. Berstein polynomials were used to do this. This technique was adapted to parameterise aerofoils and wings[116] .

Vanderplaat also describes a parameterisation method in the appendix of his paper[3] where the shape of an aerofoil is defined with an equation:

$$t = \delta \left(A\sqrt{x} + a_1 x + a_2 x^2 + ...a_n x^n \right) \tag{1.7}$$

where t is the thickness of the aerofoil and δ is the thickness to chord ratio for the initial aerofoil. The square root term yields a parabolic LE term. The coefficients A, a_1, a_2,..., a_n are the variables perturbed by the optimisation function.

Other parameterisation methods include Fourier series, piece-wise polynomial, and orthogonal polynomial. NACA has in effect a parameterisation system of 4, 5 and 6-digit sections. In general, most helicopter optimisation problems rarely exceed 10 - 30 design variables[17] . The aim in terms of parameterisation, would be to maintain a low number of design parameters to optimise, while giving the designer as much flexibility in the specific design variables required as possible.

1.4 Summary of Aims

For this project, the broad aim is to be able to initiate the optimisation process from an existing blade that is already some-what optimised for a specific objective to a certain extent. From the literature survey, it seems that between non-gradient and gradient based methods, the former seems to be gaining popularity for more complex cases such as rotor blade design. The limiting factor seems to be the computational effort in attempting to use this method with high-fidelity simulation data. However, several lower order models can be used to accurately predict interpolated data given a set of high-fidelity data. They are able to maintain accuracy and increase efficiency at the same time by orders of magnitude, because they are given only a limited set of inputs that are actually the objectives of the designer. Based on the patterns in changing these input parameters, they make the link between the input and output. This seems to be more in line with the objective of fine-tuning the aerodynamic design of a blade.
The adjoint based method offers a lot of flexibility to the design and is more suitable for designing from scratch or from highly unoptimised designs i.e. cases where a big design change is expected. Therefore it was not developed for this project at this time.
The Genetic Algorithm is a popular non-gradient based method that provides a simple efficient way of implementation. Other non-gradient based methods can be used as well. However, as a starting point, this method is selected. Some of the features explained in Section 1.3.3 will be used to create the GA.
A number of metamodels are of interest. This part of the optimisation process is quite important, as it is the part that overcomes the problem of efficiency with non-gradient based methods. The kriging and Neural Network methods particularly show a lot of potential for accurate interpolation and ease of coupling with the optimiser.
The design parameters and objectives are also key components of the optimisation process. The literature survey has provided important objectives in selecting the techniques used to parameterise designs and offered good points to take into consideration when creating an objective function. These two parts are highly dependent on the user and their experience. This method however, is aimed at assisting, rather than replacing the designer. Therefore, some of it is reliant on user-experience.

The process will be demonstrated for a number of aerodynamic bodies ranging from aerofoils to rotors in order to validate the process. Metamodels will be employed with Navier-Stokes CFD as the high-fidelity flow model. The novelty of the work is the combination of simple methods that result in efficient and accurate optimisation of well-defined parameters of rotor blades, which are already designed with specific objectives in mind, in order to obtain the best performance out of the design. This is obtained by decoupling the optimisation process directly from the high-fidelity solver and linking it through a lower-order model that relies on the CFD model for its accuracy.

1.5 Thesis Outline

Chapter 2 is a description of the high-fidelity solver, the Helicopter Multi-Block solver (HMB) used for all the cases in this project.
Chapter 3 describes the optimisation technique and all its components in detail. The chapters that follow show the application of the method to a variety of cases ranging from aerofoils in steady state, to rotors in forward flight. Some parts of these results are included in Chapter 3 to explain the optimisation method better.
Chapter 4 describes the application of the method to a transonic aerofoil.
Chapter 5 describes the application of the method to the optimisation of rotor sections in forward flight. This case was used to develop the initial code.
Chapter 6 applies the method to a wing planform.
Chapter 8 describes the application of the method to the UH60-A in forward flight motion. Hover performance was also optimised.

Chapter 10 describes the application of the method to the BERP-like rotor tip in forward flight and hover.
Chapter 11 describes the summary, conclusions and future work that can be carried out from this project.

For each case, a unique aspect of the optimisation technique was developed. The fully developed method was used on the final case, the BERP-like tip rotor. What follows after that is the conclusion and a summary of further developments that can be made on this project. The appendices were used to present the optimisation method and raw data.

Chapter 2

CFD Methods

2.1 Helicopter Multiblock Solver

The CFD Solver used for this work is the Helicopter Multiblock Solver (HMB). For all the test cases, the Reynolds Averaged Navier-Stokes (RANS) method was used with the $\kappa - \omega$ turbulence model. HMB solves the RANS equations in integral form and discretises using a cell-centred finite volume approach on structured multiblock grids. Temporal integration is done using an implicit dual-time stepping method. The details of the theory behind this method and the derivation of its equations can be found in Anderson[23] .
The Navier-Stokes equations defines the flow by mathematically stating the conservation laws of physic:

- Conservation of mass (Continuity equation):

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho U_i)}{\partial X_i} = 0 \tag{2.1}$$

  where ρ is the density, U_i is the velocity in the i^{th} direction which is X_i.

- Conservation of momentum (Newton's 2^{nd} law):

$$\frac{\partial (\rho U_i)}{\partial t} + \frac{\partial (\rho U_i U_j)}{\partial X_j} = \partial f_i + \frac{\partial p}{\partial X_i} - \frac{\partial \tau_{ij}}{\partial X_j} \tag{2.2}$$

  where f_i are the body forces (such as gravity, coriolis effect and so on) in the i^{th} direction, p is the pressure and τ_{ij} is the Newtonian stress tensor given as,

$$\tau_{ij} = \mu \left[\left(\frac{\partial U_i}{\partial X_j} + \frac{\partial U_j}{\partial X_i} \right) - \frac{2}{3} \delta_{ij} \frac{\partial U_k}{\partial X_k} \right] \tag{2.3}$$

  where μ is the molecular viscosity and δ_{ij} is the Kronecker delta given as

$$\delta_{ij} = \left\{ \begin{array}{l} 1 \text{ if i=j} \\ 0 \text{ otherwise} \end{array} \right\} \tag{2.4}$$

- Conservation of energy (1^{st} law of thermodynamics):

$$\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial X_j} \left[U_i \left(\rho E + p \right) \right] + \frac{\partial}{\partial X_j} \left(U_i \tau_{ij} + q_j \right) = 0 \tag{2.5}$$

  where E is the total energy (internal and kinetic) per unit volume and q_i is the heat flux with respect to the i^{th} direction. Both are given as:

$$E = e + \frac{1}{2} U_i^2 \tag{2.6}$$

$$q_i = -k_T \frac{\partial T}{\partial X_i} \tag{2.7}$$

$$\tag{2.8}$$

  where e is the internal energy, k_T is the heat transfer coefficient and T is the temperature.

For fluids, all of the above are functions of the three velocity components in each perpendicular direction, the density, the pressure and the internal energy. They can be expressed in matrix form for a fluid element fixed in space (i.e. conservation form of the equations). So the N-S Equations can be expressed as

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = J \tag{2.9}$$

where Q is the vector of conserved variables, F, G and H are the flux vectors in the x, y and z directions and J contains the source terms as shown:

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho(e + \frac{U^2}{2}) \end{bmatrix} \tag{2.10}$$

$$F = \begin{bmatrix} \rho u \\ \rho u^2 + p - \tau_{xx} \\ \rho uv - \tau_{xy} \\ \rho wu - \tau_{xz} \\ \rho \left(e + \frac{U^2}{2}\right) u + pu - k\frac{\partial T}{\partial x} - u\tau_{xx} - v\tau_{xy} - w\tau_{xz} \end{bmatrix} \tag{2.11}$$

$$G = \begin{bmatrix} \rho v \\ \rho uv - \tau_{yx} \\ \rho v^2 + p - \tau_{yy} \\ \rho wv - \tau_{yz} \\ \rho \left(e + \frac{U^2}{2}\right) v + pv - k\frac{\partial T}{\partial y} - u\tau_{yx} - v\tau_{yy} - w\tau_{yz} \end{bmatrix} \tag{2.12}$$

$$H = \begin{bmatrix} \rho w \\ \rho uw - \tau_{zx} \\ \rho vw - \tau_{zy} \\ \rho w^2 + p - \tau_{zz} \\ \rho \left(e + \frac{U^2}{2}\right) w + pw - k\frac{\partial T}{\partial x} - u\tau_{zx} - v\tau_{zy} - w\tau_{zz} \end{bmatrix} \tag{2.13}$$

$$J = \begin{bmatrix} 0 \\ \rho s_x \\ \rho s_y \\ \rho s_z \\ \rho(us_x + vs_y + ws_z) + \rho q \end{bmatrix} \tag{2.14}$$

i.e. the first rows of each matrix are part of the continuity equation, the 2nd, 3rd and 4th represent the conservation of momentum and the last row represents the conservation of energy[23]. The source terms, s_x, s_y, s_z, $us_x + vs_y + ws_z$ and q are typically zero except in the case of a hovering rotor where a Froude condition is applied as explained later.

The N-S equations completely describe turbulent flow, however, at high Reynolds numbers, there are many turbulent scales that vary with time and space. Therefore, the number of turbulent scales are reduced by time averaging the Navier-Stokes equations to give the Reynolds-Averaged Navier-Stokes equations (RANS). This results in additional unknowns (called Reynolds stresses) which must be modelled with a turbulence model. This is described in more detail later on in this chapter.

As mentioned, the HMB flow solver uses a cell-centred finite volume approach combined with an implicit dual-time stepping method. The governing equations are applied to each cell in turn. The spatial discretisation of Eqn 2.9 leads to a set of ordinary differential equations in time,

$$\frac{d}{dt} Q_{i,j,k} \vartheta_{i,j,k} + R_{i,j,k} = 0 \tag{2.15}$$

where $\vartheta_{i,j,k}$ is the volume of the cell and $R_{i,j,k}$ is the flux residual.
The solution marches in pseudo-time iterations for each real time-step to achieve fast convergence. The following system of equations is then solved in the implicit scheme during the time integration process

$$\frac{\Delta VQ_{i,j,k}^{m+1} - \Delta VQ_{i,j,k}^{m}}{\Delta V \Delta \tau} + \frac{\Delta VQ_{i,j,k}^{n+1} - \Delta VQ_{i,j,k}^{n}}{\Delta V \Delta t} = R_{i,j,k}^{n+1} \tag{2.16}$$

where n represent real time step iterations, m represents pseudo time step iterations, $\Delta\tau$ is the pseudo time-step increment and Δt is the physical time step increment. ΔV is the cell volume change. The term $Q_{i,j,k}^{n+1}$ is obtained when the pseudo time steps converge (first part of Eqn 2.16) to within a user-defined tolerance. An implicit scheme is used for the pseudo time integration and $R_{i,j,k}^{n+1}$ is approximately given as

$$R_{i,j,k}^{n+1} \approx R_{i,j,k}^{n} + \frac{\partial R_{i,j,k}^{n}}{\partial Q_{i,j,k}^{n}}(Q_{i,j,k}^{n+1} - Q_{i,j,k}^{n}) \tag{2.17}$$

The pseudo time integration is typically carried out till a convergence of 0.001. To ensure this is reached, for most cases, the maximum number of pseudo time steps was set to between 100 and 200.
Combining Eqns 2.16 and 2.17, the linear system becomes,

$$\left(\frac{1}{\Delta t} + \left(\frac{\partial R_{i,j,k}}{\partial Q_{i,j,k}}\right)^{n}\right)\left(Q_{i,j,k}^{n+1} - Q_{i,j,k}^{n}\right) = -R_{i,j,k}^{n} \tag{2.18}$$

The convective fluxes are resolved using Osher's scheme[117]. Second-order accuracy is provided using the Monotone Upstream-centred Schemes for Conservation Laws (MUSCL) variable extrapolation method[118] and any spurious oscillations across shock waves is removed using the Van Albada limiter. Central differencing spatial discretisation is used to solve for the viscous terms. The linearisation results in a set of non-linear equations. This is solved by integration in pseudo-time using a first-order backward difference. A Generalised Conjugate Gradient (GCG)[119] method is then used in conjunction with a Block Incomplete Lower-Upper (BILU)[119] factorisation as a pre-conditioner to solve the linear system of equations, which is obtained from a linearisation in pseudo-time.

Turbulent flow causes aerodynamic structures that vary with time and space quite frequently. To run the full Navier-Stokes equations to obtain all of these structures would require a high resolution in time and space. However, with increasing Reynold's number, it is found that the larger scale turbulent structures carry more of the energy than the small ones. To avoid the high computational cost, therefore, the variables can be split into two components: an average value and a fluctuating component. So if the conservation equations are time averaged, then a number of additional terms, Reynolds stress terms, appear in the equations (denoted with '^{R}' in the following equations):

$$\frac{\partial(\rho U_i)}{\partial t} + \frac{\partial(\rho U_i U_j)}{\partial X_j} = \partial f_i + \frac{\partial p}{\partial X_i} - \frac{\partial}{\partial X_j}\left(\tau_{ij} + \tau_{ij}^{R}\right) \tag{2.19}$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial X_j}\left[U_i\left(\rho E + p\right)\right] + \frac{\partial}{\partial X_j}\left(U_i(\tau_{ij} + \tau_{ij}^{R}) + q_j^{R}\right) = 0 \tag{2.20}$$

where q_j^R is the turbulent heat flux. The turbulence model is then employed to model these Reynolds stress terms. A number of turbulence models are available in HMB. For the cases presented here, the two-equation $\kappa - \omega$ model was used[11]. In a few cases Menter's Shear Stress Transport (SST)[120] blending was also applied as it improves the performance where adverse pressure gradients exist. For this model, two transport equations are used: the turbulent kinetic energy, κ, and the κ-specific dissipation rate, ω which is a function of the length. The eddy viscosity is obtained as:

$$\mu_T = \rho\frac{\kappa}{\omega} \tag{2.21}$$

The full turbulent transport equations used in the formulation of the $\kappa - \omega$ model are given as

$$\frac{\partial}{\partial t}(\rho\kappa) + \frac{\partial}{\partial x_j}(\rho U_j \kappa) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_T}{\sigma_\kappa}\right)\frac{\partial \kappa}{\partial x_j}\right] + \rho\left(P_\kappa - \beta^* \omega \kappa\right) \tag{2.22}$$

$$\frac{\partial}{\partial t}(\rho\omega) + \frac{\partial}{\partial x_j}(\rho U_j \omega) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_T}{\sigma_\omega}\right)\frac{\partial \kappa}{\partial x_j}\right] + \rho\left(\frac{\alpha}{\upsilon_t} P_\omega - \frac{\beta}{\beta^* \omega^2}\right) + \rho S_l \tag{2.23}$$

The values for the coefficients are given in Table 2.1 for the model. The flow solver was used in parallel for the large grid cases. Message Passing Interface (MPI) was used for the communication between the processors in parallel.

α	β	β^*	σ_κ	σ_ω	S_l
$\frac{5}{9}$	$\frac{3}{40}$	$\frac{9}{100}$	2	2	0

Table 2.1: *Table of coefficients for the $\kappa - \omega$ turbulence model from Wilcox*[11] .

2.2 Harmonic Balance Method

The time-marching method, even when parallel computing is used, can take a few days of clock time for a rotor to be fully analysed. An alternate method that can be used to obtain the performance of the rotors to the same accuracy (provided a sufficient number of modes is used), is the Harmonic Balance Method (HB)[121] . With HB, the time taken to do the same calculation is about an order of magnitude less than the time taken using the time-marching method. This greatly improves the efficiency of the optimisation process, making it a more usable technique in the design of rotors. The method is demonstrated for the UH60-A rotor in forward flight and the results obtained using 4 modes are in fair agreement with flight test data and the time marching results. This is a positive outcome suggesting that either harmonic balance or time marching methods can be used with this framework. A higher number of modes would be needed if higher harmonic vibratory performance parameters are required.

If the solution can be assumed to be fully developed and periodic, then the governing equations can be represented in the frequency domain. This is the basis of the HB method. It obtains the Fourier coefficients of the variables at each point in the mesh. Eqn 2.24 (which is the same as Eqn 2.15 but with terms removed for clarity) represents the governing equation where $Q(t)$ is the solution and $R(t)$ is the residual which are assumed to be periodic,

$$F(t) = \frac{dQ(t)}{dt} + R(t) = 0 \tag{2.24}$$

$R(t)$ is the residual and is a function of the grid, the grid velocity and the variables, $Q(t)$. The grid motion is known, therefore, $Q(t)$ can be obtained from Eqn 2.24. As a Fourier series with a fixed number of modes, N_H, Eqn 2.24 becomes

$$Q(t) = \hat{Q}_o + \sum_{n=1}^{N_H}\left(\hat{Q}_c \cos(\omega n t) + \hat{Q}_s \sin(\omega n t)\right) = \sum_{k=-N_H}^{N_H} \hat{Q}_k e^{ik\omega t} \tag{2.25}$$

$$R(t) = \hat{R}_o + \sum_{n=1}^{N_H}\left(\hat{R}_c \cos(\omega n t) + \hat{R}_s \sin(\omega n t)\right) = \sum_{k=-N_H}^{N_H} \hat{R}_k e^{ik\omega t} \tag{2.26}$$

$$F(t) = \hat{F}_o + \sum_{n=1}^{N_H}\left(\hat{F}_c \cos(\omega n t) + \hat{F}_s \sin(\omega n t)\right) = \sum_{k=-N_H}^{N_H} \hat{F}_k e^{ik\omega t} \tag{2.27}$$

where k is the wave number. This system of equations is substituted into Eqn 2.24. The Fourier terms are then given as

$$\hat{F}_o = \frac{\omega}{2\pi}\int_o^{2\pi/\omega} F(t)dt = \hat{R}_o \tag{2.28}$$

$$\hat{F}_c = \frac{\omega}{\pi}\int_o^{2\pi/\omega} F(t)\cos(wnt)dt = \omega n\hat{Q}_s\hat{R}_c \tag{2.29}$$

$$\hat{F}_s = \frac{\omega}{\pi}\int_o^{2\pi/\omega} F(t)\sin(wnt)dt = -\omega n\hat{Q}_c\hat{R}_s \tag{2.30}$$

This results in equations for each mode, n, giving a system of N_T equations, where $N_T = 2N_H + 1$, for the Fourier series coefficients:

$$\hat{R}_o = 0 \tag{2.31}$$

$$\omega n\hat{Q}_s\hat{R}_c = 0 \tag{2.32}$$

$$-\omega n\hat{Q}_c\hat{R}_s = 0 \tag{2.33}$$

or as

$$i\omega k\hat{W}_k + \hat{R}_k = 0 \tag{2.34}$$

This is expressed in matrix form as:

$$\omega M\hat{Q} + \hat{R} = 0 \tag{2.35}$$

where M is an $N_T \times N_T$ matrix.
As mentioned earlier, $R(t)$ is a function of $W(t)$ and it is non-linear. Therefore each coefficient, $\hat{R}_k$ depends on all the coefficients $\hat{W}_k$ and hence must be solved iteratively. There are a number of ways that this can be carried out.
In the pseudo-spectral method, Eqn 2.34 is transformed back to the time domain and the period is split into N_T equal discrete time intervals, represented as W_{hb} and R_{hb}. Then Eqn 2.35 is decomposed to form,

$$\omega DW_{hb} = R_{hb} \tag{2.36}$$

where $D = E^{-1}ME$, E being a transformation matrix transforming $\hat{W}$ and $\hat{R}$ to W_{hb} and R_{hb}. The diagonal of D is 0. Pseudo time marching can then be applied to the harmonic balance equation

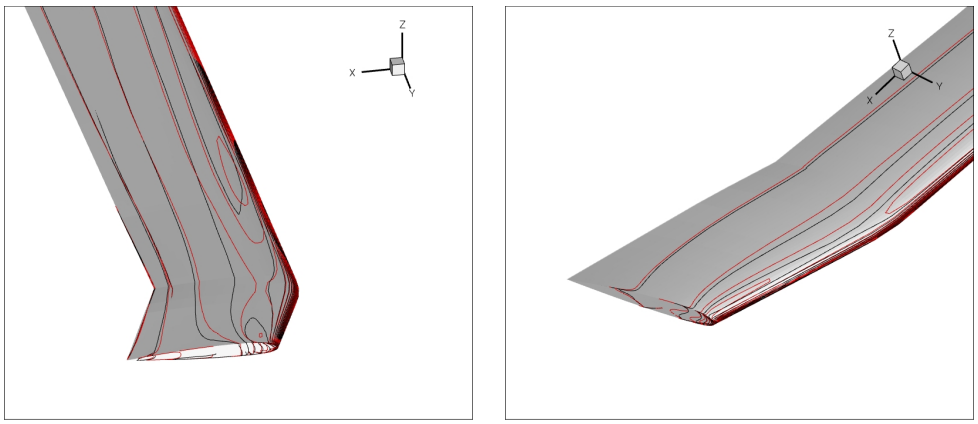
$$\frac{dW_{hb}}{dt} + \omega DW_{hb} + R_{hb} = 0 \tag{2.37}$$

This is the standard way of solving for the solution. However, other methods exist that use less memory, which can be a limiting factor in the use of the HB method. More details are given in Woodgate et al.[121,122] and Jang et al.[123] .

Figure 2.1 shows a qualitative comparison of a UH60-A blade solution using TM and HB (2 and 4 modes). That the results are similar especially when more modes are used, suggests that both data sets could be included in the CFD database. Figure 2.2 shows the lift and moment coefficient comparison between the analysis of the TM method and the HB method for the same rotor. Four modes were used to construct the solution using the HB method. i.e. a solution exists for every 10 degrees of azimuth.

2.3 Grid Generation

In all cases, multiblock structured grids were used and were generated using ICEM-CFD[124] . Different types of grids were used for different topologies. For aerofoil sections and blocking around the blade of a rotor, a C-grid was used as shown in Figure 2.3. For rotors, this C-grid is wrapped in an H-grid. For helicopter fuselage bodies, the geometry is enclosed in an O-grid, which is wrapped in an H-grid. The mesh spacing normal to the solid surfaces was always 1×10^{-5} chords



2 modes 4 modes

Figure 2.1: *Pressure contours of the UH60 rotor using 2 and 4 modes for the Harmonic Balance method (red lines) compared to the solution from the time-marching method (black lines). The lines represent isobars at the same pressure values at an azimuth of 90 degrees, which represents the worst case.*

($y^+ < 1$) to resolve the turbulent boundary layer. The spacing was then transitioned smoothly i.e. the ratio of spacing from one cell to the next was never greater than two.

For a hovering rotor, the wake is assumed to be steady and so the calculation is a steady-state and assumed to be periodic. Therefore the flow is solved for only a $1/\mathrm{N_{blade}}$ segment of the full flow field with a periodic boundary conditions as shown in Figure 2.4(a) on the left which is for a 4-bladed rotor. The boundary conditions for the inflow and outflow are 'potential sink/Froude' condition to suppress recirculation in the flow domain and were placed at the rotor origin[125]. These conditions must be prescribed at a distance of at least 5 rotor radii[125]. An axial velocity was prescribed at the outflow boundary based on the rotor thrust and outflow radius. Also, for all rotors, the blade is not connected to the hub directly so that root vortices can also be captured, as seen on the right of Figure 2.4(a). Figure 2.4(b) shows the root on the left and tip on the right with typical wall spacing and number of cells.
Also for hovering rotors, the hub is modelled as a cylinder, whereas for forward flight, it is modelled as a simplified smooth elliptic hub.

For the forward flying grid, the full rotor is required. Therefore, a $1/\mathrm{N_{blade}}$ segment is built (as in Figure 2.4(a)) and then copyrotated at the symmetry planes to form the full rotor and flow field. The typical number of cells used is also shown in Figure 2.4. Rotor grid sizes were approximately eight million cells. It is assumed that the rotor blades are rigid and are connected to the hub by a set of three hinges. The flap, lead-lag and pitch centre positions and order are user-defined. In all the cases presented here, this was the order used. The flapping, lead-lag and pitch angles were defined using a set of harmonics with cosine and sine components as follows:

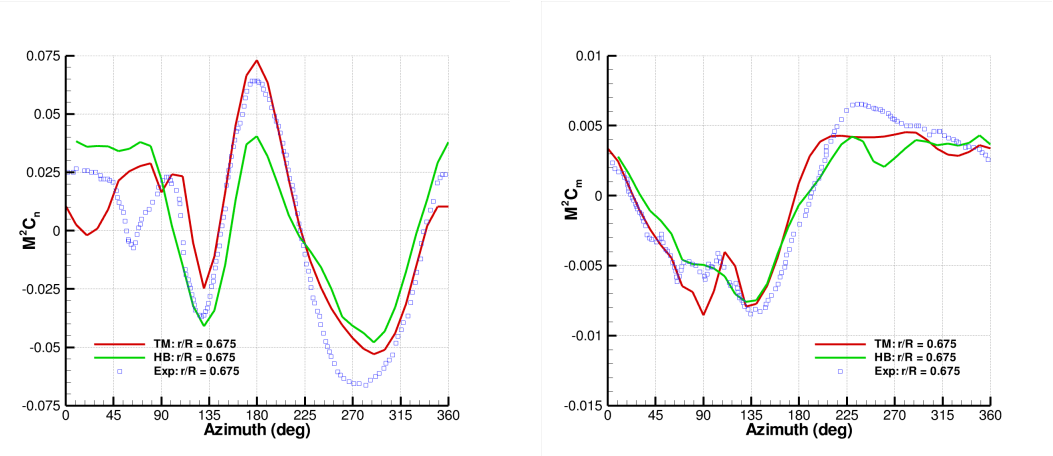
$$\beta(\Psi) = \beta_o - \beta_{1s}\sin(\Psi) - \beta_{1c}\cos(\Psi) - \beta_{2s}\sin(2\Psi) - \beta_{2c}\cos(2\Psi) - ... \quad (2.38)$$

$$\delta(\Psi) = \delta_o - \delta_{1s}\sin(\Psi) - \delta_{1c}\cos(\Psi) - \delta_{2s}\sin(2\Psi) - \delta_{2c}\cos(2\Psi) - ... \quad (2.39)$$

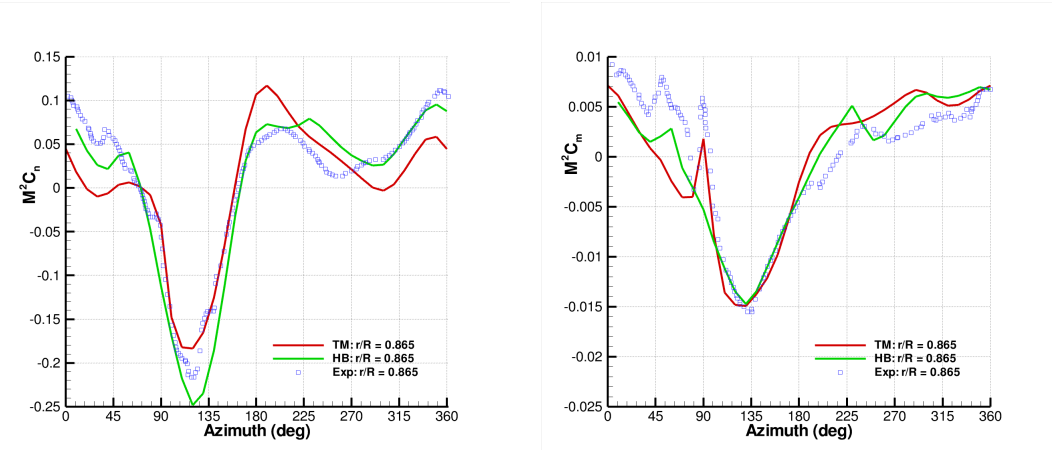
$$\theta(\Psi) = \theta_o - \theta_{1s}\sin(\Psi) - \theta_{1c}\cos(\Psi) - \theta_{2s}\sin(2\Psi) - \theta_{2c}\cos(2\Psi) - ... \quad (2.40)$$

where β is the flap angle, δ is the lead-lag angle and θ is the pitch angle. The way that these values are prescribed in the frame of reference of the helicopter is detailed in Steijl et al.[125] and Leishman[13].

For wing optimisation cases, the grid was generated automatically using the ICEM-CFD scripting language. Here, the position of the aerofoil coordinates could be modified and using this as a reference, the entire geometry was built around it. The blocking could then be automatically snapped to named points and curves. An example of these replay (*.rpl*) files can be found in



r/R = 0.675



r/R = 0.865

Figure 2.2: *Experimental data in comparison to CFD by time marching (TM) and Harmonic Balance (HB) methods. The mean value has been removed.*

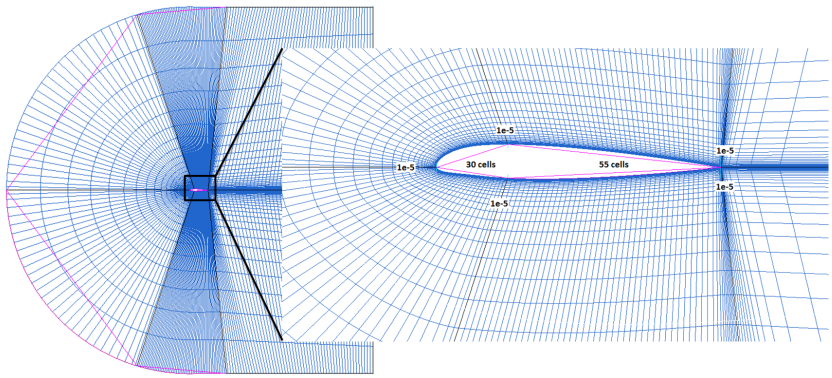
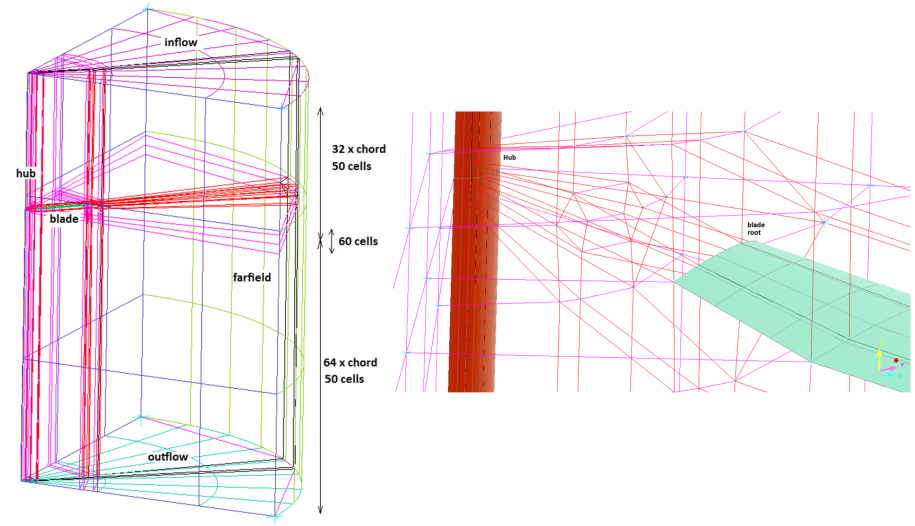


Figure 2.3: *C grid for aerofoils.*

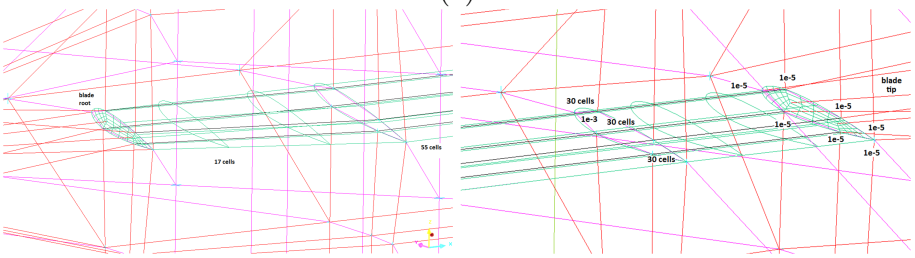
Appendix B.10.1. A similar, but only partly automated, method was used for BERP-like rotor tip shapes.

2.4 Procedure for dM/dt

To simulate the aerodynamics of a section of a rotor blade, a harmonic pitching-translation motion was used. The aerofoil is hinged at its quarter-chord point. The translation uses 1 harmonic and



(a)



(b)

Figure 2.4: *Grid for hovering rotor.*

can be represented as:

$$x = x_o + x_s \sin(2kt) + x_c \cos(2kt) \tag{2.41}$$

where x_o is the initial position, k is the reduced frequency and x_s and x_c are the coefficients of the sine and cosine components.
Since the blade tip Mach number and the free stream Mach numbers are known, the local Mach number is given by:

$$M = M_{tip}\frac{r}{R} + M_\infty \sin(\Psi) \tag{2.42}$$

$$= M_{ref} + M_\infty \sin(\Psi) \tag{2.43}$$

where Ψ is the azimuth angle, given by ωt. Since $\mu = \frac{M_\infty}{M_{tip}}$,

$$M = M_{ref} + \mu M_{tip} \sin(\Psi) \tag{2.44}$$

$$\frac{M}{M_{ref}} = 1 + \mu\frac{R}{r}\sin(\Psi) \tag{2.45}$$

Ψ is ωt and ω is represented as a non-dimensional value as

$$\omega = \frac{2kU_\infty}{c} \tag{2.46}$$

So the translational motion can be fully defined by knowing the advance ratio, the tip Mach number, the station along the blade, the reduced frequency and the free stream Mach number.
Similarly, the angle of attack is also represented as a single harmonic motion. x_c and x_s can be used to specify the amplitude of oscillation. For a rigid blade, the amplitude would be the same throughout the span.

2.5 Wake Visualisation and Vortex Criterion

There are several ways of visualising the wake of a rotor depending on the definition of a vortex. In this project, the Q and λ_2 criteria have been used to do this. Using the eigenvalues of the velocity gradient tensor, ∇u, it has been proposed that a vortex core is a region with complex eigenvalues of ∇u. Complex eigenvalues imply that the local streamline pattern is closed or spiral in a reference frame moving with the point[126].
Q is the second invariant of ∇u. If it is positive and the pressure is lower than the ambient pressure, then an eddy exists according to Hunt[126]. Q represents the balance between shear strain rate and vorticity magnitude. Q vanishes at a wall unlike vorticity strength.

Haller[127] analyses a few definitions, including the Q and λ criteria. Q is defined as

$$Q = \frac{1}{2}[|\Omega|^2 - |S|^2] > 0 \tag{2.47}$$

and must satisfy the condition in Equation 2.47 for there to be a vortex. Here, Ω is the norm of the vorticity tensor (or the spin tensor) and S is the strain rate i.e. Ω must exceed S.
On the other hand, the λ criterion states that

$$\lambda_2(S^2 + \Omega^2) < 0 \tag{2.48}$$

for a vortex to exist. Jeong and Hussain[126] start from the idea that a pressure minimum is not a characteristic unique to vortices and that pressure minimums that do exist in vortices can be removed by viscous effects. Therefore, they break down the velocity gradient tensor and obtain S and Ω and define a vortex as a region where the $S^2 + \Omega^2$ has two negative eigen values. More details on the formulation of these definitions can be found in Haller[127] and in the original paper by Jeong and Hussain[126].

Chapter 3

Optimisation Method

As seen in Chapter 1, there has been substantial research and interest in the use of evolutionary algorithms (EAs) in the optimisation of rotors. However, application and development in the past has mostly occurred in the design of compressors, turbines or electric rotors, where the ratio of the blade size to the actual rotor is quite low[38,39,66] . In the case of helicopter and other rotorcraft rotors, the more popular methods used were gradient-based since they have a lower demand for computational effort, time and cost. Using non-gradient based methods is expected to obtain global optima. However, the high computational effort rendered these methods impractical for rotor optimisation.

Nevertheless, there are a number of ways that have recently been developed that could reduce the computational effort in terms of time as well as storage and with the advances made in computing power, the use of these methods in rotor optimisation is becoming more feasible. The use of metamodels or surrogate models can reduce the calculation time and is expected to counteract the extra optimisation time required with the use of non-gradient based optimisation techniques whilst still maintaining global optimisation. Figure 3.1 is a map of the stages involved in the optimisation. It is an aposteriori type method[84] . The work described here aims to demonstrate a framework allowing different aspects of rotors and fuselages to be aerodynamically optimised given an existing design as a starting point. For real helicopter rotors, the initial designs would already be near optimum and the aim is to capture the aerodynamic effects of any design changes and adjust the variables of the design to find an optimum that will lead to better rotor performance.

3.1 Optimisation Framework

To improve the efficiency of using a non-gradient based method, the high-fidelity solver is decoupled from the optimisation loop. It is available to the optimiser through a database. Therefore the first step in the process is to create a database or population of designs. It is important for the parameters to be clearly defined as well as the boundaries of the database and the constraints. These must be user-defined and are based on experience and the requirements of the optimisation case. A parameterisation technique must be created if it does not already exist for the case. A few have been developed in this project for cases where the parameterisation technique was not clearly defined or it did not adequately describe the design for the optimisation required. The boundaries of these parameters are then set, based on engineering expertise of the problem and the case in question. Once these have been determined, a sampling method is used to generate a number of design points to create the database or design space using the high-fidelity solver. For each case, the geometry is modified and run using the high-fidelity solver.
The next step is to determine the performance parameters that capture the objectives of the optimisation and to then combine these into a single value that the optimiser can use to determine the overall performance of an individual design. This again, is dependent on the experience of the user, but is also guided by the performance of the designs in the database and validated using other optimisation criteria such as the Pareto Front method[53,86] .

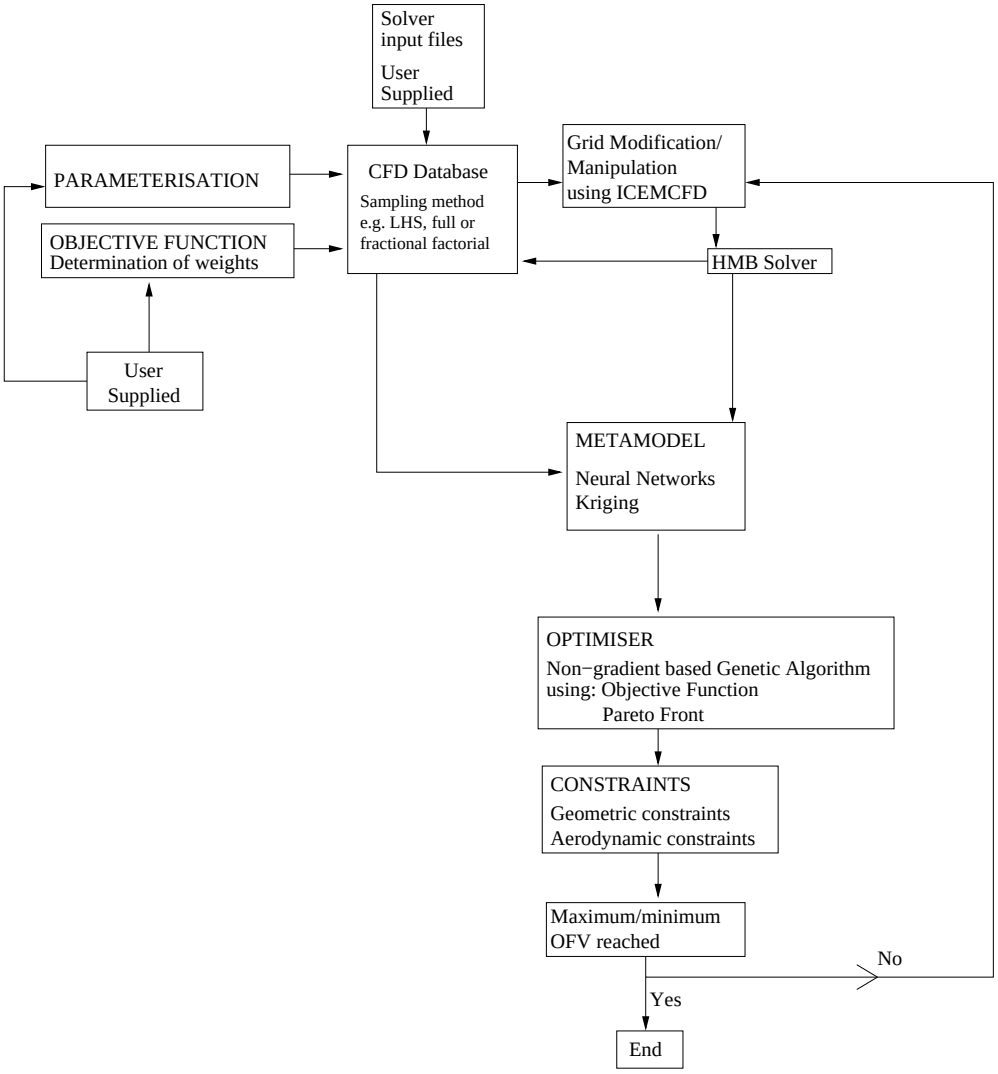


Figure 3.1: *Map of the processes involved in the optimisation of rotors.*

The optimiser now uses the database (which represents the design space) to determine the performance of a design point. However, it may need to evaluate the performance of a design point that may not exist in the database and whose solution would need to be obtained using the solver in order to obtain its performance. This would incur a large computational cost and time as such requirements tend to occur often. To overcome this, a metamodel is used to predict the performance of unknown design points based on the performance of the existing points in the database by some interpolation technique. The accuracy or the tolerance value of the error in these predictions is what determines the resolution of the design parameter values in the database. The metamodel can be validated using a small number of additional design points obtained using the high-fidelity solver. A selection of different metamodels were tested in this work including artificial neural networks (ANNs), kriging, polynomial fit and proper orthogonal decomposition (POD). Of these, the former two were the most successful in terms of accuracy and efficiency. The optimiser can then employ the predictions of the metamodel to obtain the performance of new designs quickly. After a number of iterations, the designs are expected to converge to an optimum or a cluster of

optimal designs. The optimiser is able to exercise constraints specified by the user. The final new design is then validated using the high-fidelity solver.

This method was tested for different cases including:

- sections along a rotor blade,
- transonic aerofoils,
- wing planforms,
- rotors in hover,
- rotors in forward flight,
- simplified helicopter fuselages.

For the rotors, three different test cases were used:

- a simple rectangular NACA 0012 blades in hover,
- the UH60-A rotor in hover and forward flight and
- a BERP-like rotor in hover and forward flight applied to a typical fast flying, manoeuvrable rotor.

Most of the method was created using the first case, the sections along a rotor blade, and a number of metamodels were compared as well as the outcome in comparison to the Pareto front optima. In terms of parameterisation techniques, a parameterisation method for curves was developed using the transonic aerofoil test case. For the fuselage body, a modified version of the ROBIN body parameterisation technique[105] was employed. For the BERP-like rotor, the planform was defined using three different curves that defined the sweep and notch geometry.
In most of the cases, the sampling method used was a full-factorial method due to the small number of points in the database. However, for the optimisation of a multi-segment fixed wing planform, other sampling methods were tested, and the one used was a variation of the fractional-factorial method. Also, the grid generation was automated for this case which allowed for a much faster way of developing the initial database since there were five design parameters. This automation was also applied to the fuselage and BERP-like rotor cases. The Latin Hypercube Sampling (LHS) method was also tested in the transonic aerofoil test case. For the rotor cases, the overall optima in both flight conditions were obtained for the UH60-A and the BERP-like rotor. It was during this time that further development of the overall optimisation method took place. The method was also applied to a simplified fuselage body for drag reduction.

The rest of this chapter describes each stage of the optimisation in more detail and the following chapters describe the application and results obtained for each test case.

3.2 Parameterisation Techniques

Parameterisation plays a key part in increasing the efficiency of the optimisation process. The fewer the design variables used to define a shape (for a fixed amount of flexibility), the smaller the initial population of high-fidelity data (since the number of dimensions of the database is reduced), the less training time required for the metamodels, the more accurate their predictions are given a fixed number of samples and the faster the output of the optimiser. For the cases analysed in this work, there already exists a number of parameterisation techniques, some that can only be applied to specific cases and some that are more generic.

For example, for the rotor section test case, NACA aerofoils were used. NACA aerofoils have their own parameterisation technique that defines aspects such as thickness, camber, position of maximum camber and so on using independent numerical values in their naming system. The advantage of this method is that values are easily interpreted into the design i.e. they are intuitive to the actual shape of the design. However, the disadvantage is that they can only be used for a

particular class of aerofoils and cannot be generically applied to define all aerofoils. Also, for more flexibility in the design, more design variables would be required, which multiplies the number of initial designs.

There are other techniques that can be used to reduce the number of parameters. These techniques, however, can reduce the intuitiveness of the shape representation or reduce the flexibility of the design changed. For example, the use of splines and NURBS can be used to represent the data quite efficiently for the amount of flexibility available, but the movement of control points and weights is not directly intuitive to how the shape is being changed. However, this method is very useful for the design of compressor blades for example, as these are anyway typically defined as splines. Also, for the objectives of this project, perhaps the amount of flexibility that such a technique provides is not required, as the aim is to fine-tune specific design parameters rather than change a shape completely.
Another method uses equations, where the coefficients of the equations are varied to change the design and the equations are then matched at their end points. This technique allows a lot of flexibility since any number of equations can be used to define a shape. The matching can be used to reduce the number of independent variables. With experience, these variables can become some-what intuitive to the change in the shape. This is useful in planform design and it has also been used in this project for the BERP-like tip and fuselage parameterisation.
Overall, the parameterisation method should be tailored to what best works for each case. Kulfan[115] discusses a number of methods used to parameterise aerofoils and also discusses a new method of parameterisation. A number of desirables of an ideal parameterisation method is also listed, specifically for aerofoils as follows[115] , although the characteristics can be applied to other cases too:

1. Well behaved: produce smooth and realistic shapes.
2. Mathematically efficient and a numerically stable process that is fast, accurate and consistent.
3. Require relatively few variables to represent a large enough design space to contain optimum aerodynamic shapes for a variety of design conditions and constraints.
4. Allows specification of key design parameters such as leading edge radius, boat-tail angle, aerofoil closure (examples specific to aerofoils).
5. Provide easy control for designing and editing the shape of a curve.
6. Intuitive - Geometry algorithm should have an intuitive and geometric interpretation.
7. Systematic and Consistent - The way of representing, creating and editing different types of geometries must be the same.
8. Robust - The represented curve will not change its geometry under geometric transformations such as translation, rotation and affine transformations.

For most of the cases, the parameterisation was simply the use of the design variable itself. For example, for the hovering rotor, the twist parameter was simply the linear twist angle, for the UH60-A rotor planform optimisation, the angle of anhedral and sweep at the tip were the parameters used. However, for the optimisation of some of the cases, such as the RAE 2822 aerofoil, the fuselage and the BERP-like rotor, specific techniques were developed and these are described below. The choice of these methods was mainly due to simplicity specific to the case. For example, the Chebyshev method for the transonic aerofoil allowed smooth design with specific parameters for the design variables optimised and is described further in Section 3.2.1. The fuselage parameterisation is a well established method, and with modifications (explained in Section 3.2.3), allows a single parameter to vary all other parameters for a smooth transition from one cross-section to another. Similarly, this was the case for the BERP-like blade equations. Each design variable required was independently defined but allowed a smooth curve to be created and the process was automated as explained in Section 3.2.2.

3.2.1 Curve Parameterisation

To parameterise the RAE 2822 aerofoil, a different approach was adopted as it does not have a direct parameterisation method that is flexible enough like the NACA digit sections. The method is based on the Chebyshev function[128] , such that a curve can be fully defined with a small number of coefficients. A function, f(x) is used to represent the aerofoil as follows:

$$f(x) = \sum_{k=0}^{N} \alpha_k D_k \tag{3.1}$$

where α_k represents the coefficients, k is the coefficient number up to N, D_k is given by $T_k - T_{k+2}$ where

$$\begin{aligned} D_k &= T_k - T_{k+2}, & (3.2)\\ T_k &= cos(k\gamma(x)), & (3.3)\\ \gamma(x) &= cos^{-1}(2\sqrt{x} - 1) & (3.4) \end{aligned}$$

x is an array of points along the chord of the aerofoil for which the corresponding y points will be found. The error as a function of the coefficients is

$$\begin{aligned} E(\alpha) &= \sum_{n=1}^{N_p} (f(x_n) - y_n)^2 & (3.5)\\ E(\alpha) &= \sum_{n=1}^{N_p} \left(\sum_{k=0}^{N} \alpha_k D_k(x_n) - y_n \right)^2 & (3.6) \end{aligned}$$

where N_p is the number of co-ordinate points. To minimise E, the gradient must be equated to 0. Therefore differentiating Equation 3.6,

$$\nabla E = \frac{\partial E}{\partial \alpha_k} = 2 \sum_{n=1}^{N_p} \left(\sum_{k=0}^{N} \alpha_k D_k(x_n) - y_n \right) D_k(x_n) \tag{3.7}$$

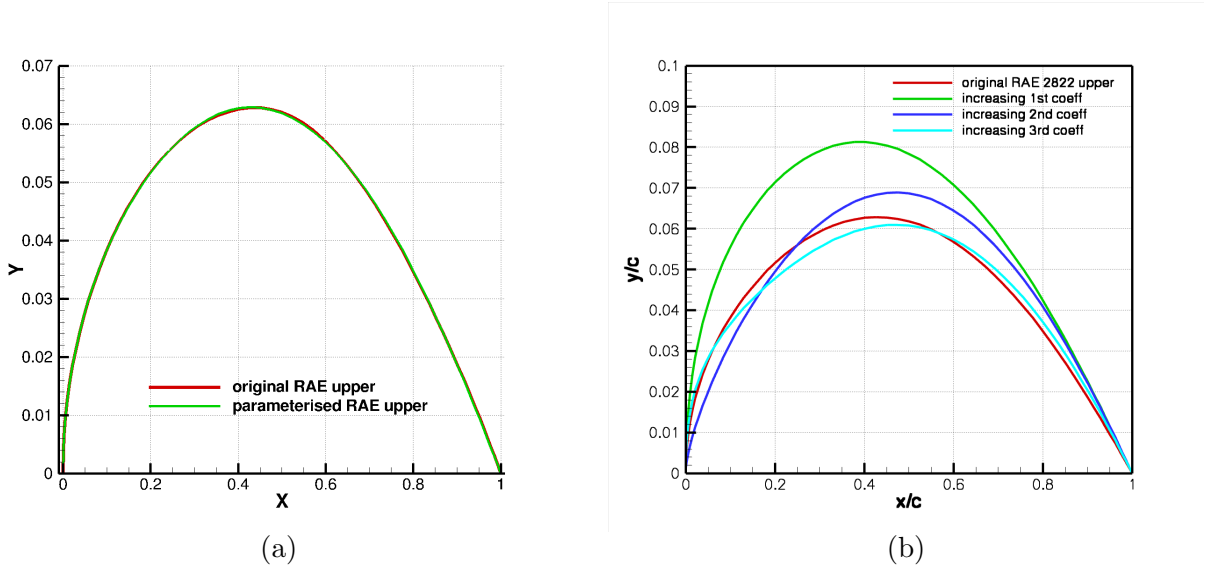
And the Hessian is

$$H = \frac{\partial^2 E}{\partial \alpha_k^2} = 2 \sum_{n=1}^{N_p} D_k(x_n) D_k(x_n) \tag{3.8}$$

So, the new α_k can be found as follows

$$\begin{aligned} E(\alpha_{k_{new}}) &= E(\alpha_k) + \nabla E(\alpha_{k_{new}} - \alpha_k) + \frac{1}{2}(\alpha_{k_{new}} - \alpha_k) H (\alpha_{k_{new}} - \alpha_k) & (3.9)\\ \alpha_{k_{new}} &= \alpha_k + H^{-1} \nabla E & (3.10) \end{aligned}$$

where ∇E, H are the gradient and Hessian matrices. The RAE 2822 upper surface was defined using six coefficients, the first three of which were used for the optimisation. The error for convergence, $E(\alpha_{k_{new}})$ was 0.0074. Modifying the coefficients has different effects on the thickness and curvature of the aerofoil shape as shown in Figure 3.2.



(a) (b)

Figure 3.2: *(a) The original upper curve compared to the parameterised upper curve of the RAE 2822. Error convergence was 0.0074. (b)The effect of increasing the 3 coefficient values to the upper surface curve. The original surface's six coefficients are* $\alpha_1 = 0.009149$, $\alpha_2 = 0.001444$, $\alpha_3 = -0.000325$, $\alpha_4 = -0.000266$, $\alpha_5 = -0.000060$, $\alpha_6 = -0.000050$.

This parameterisation technique was coded in C and can be found in Appendix B.7.

The method can also be used to parameterise a planform of a wing. For the parameterisation of a complete 3D wing, the program in Appendix B.7.1 was used. This is a simple model of a wing only used to demonstrate this parameterisation technique on a wing planform. Figure 3.3 shows the changes that occur to the leading edge if its first parameterisation coefficient is changed. It results in a change in the chord distribution of the wing. The change that occurs is not linear along the span of the wing which is more evident when the changes are bigger.
Figure 3.4 shows the changes that occur to the leading edge of the wing when both the first and second coefficients are changed. The 2^{nd} coefficient has the effect of moving the maximum chord towards the tip and also slightly modifying the maximum chord. This slight change in the maximum chord can be compensated for using the first coefficient.
The 3^{rd} coefficient appears to produce a 'jump' in the size of the chord or a 'kink' in the leading edger curve at the root and the 4^{th} coefficient does the same but in the opposite direction. Both cases are shown in Figure 3.5. So, if both are used and modified, a smooth curve can be obtained as shown in Figure 3.5. This occurs because the addition of two more coefficients adds an additional mode to the equation i.e. it gives an additional degree of freedom or increases its order.
Figure 3.6 shows that with more coefficients, more flexible curves can be obtained. On the other hand, the lack of a direct and independent link between the coefficients and established aerofoil parameters (chord, thickness, camber etc.) is a limitation of this method.

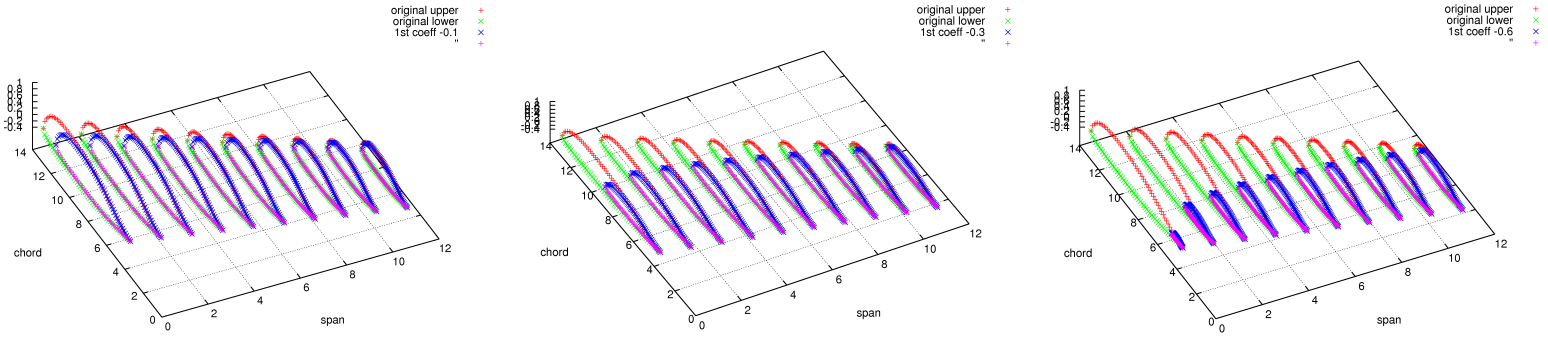


Figure 3.3: *Effect of modifying the first coefficient to the leading edge curve.*

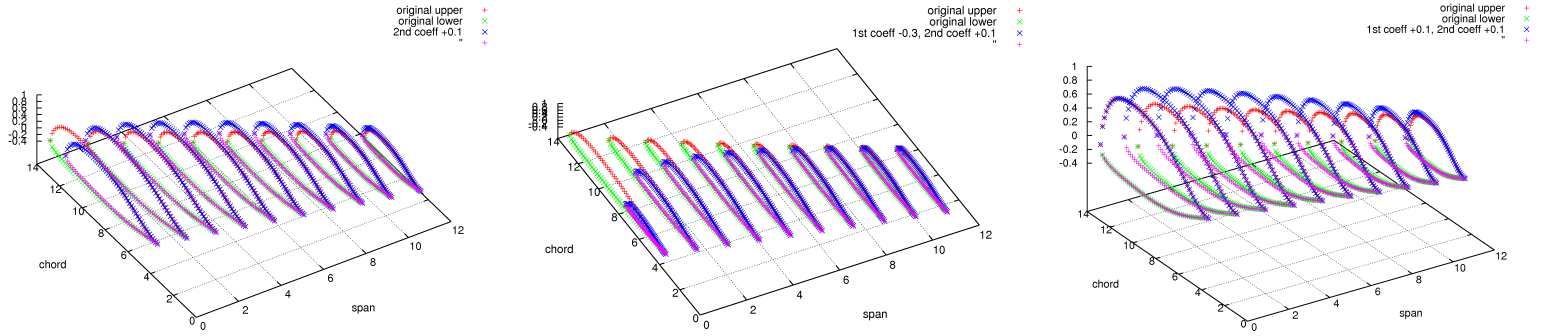


Figure 3.4: *Effect of modifying the first and second coefficients to the leading edge curve.*

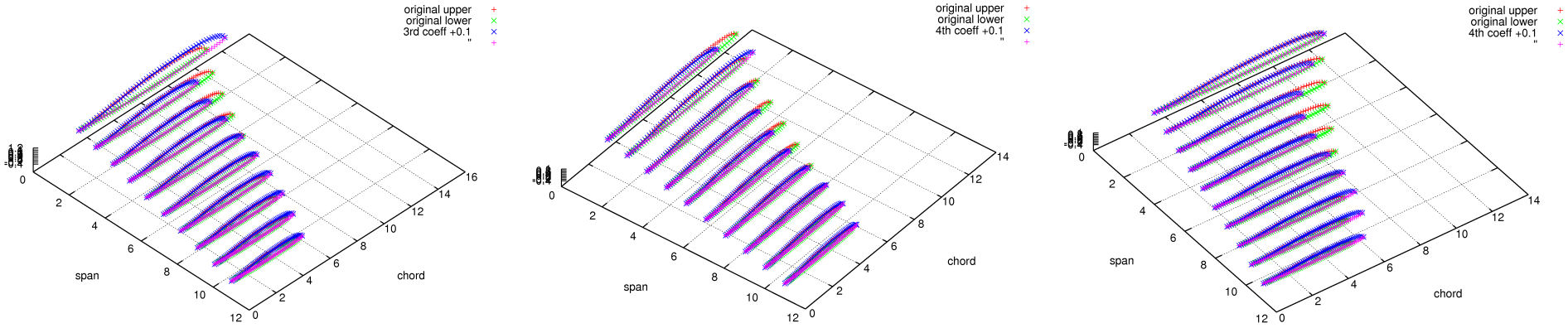


Figure 3.5: *Effect of modifying the third and fourth coefficients to the leading edge curve.*

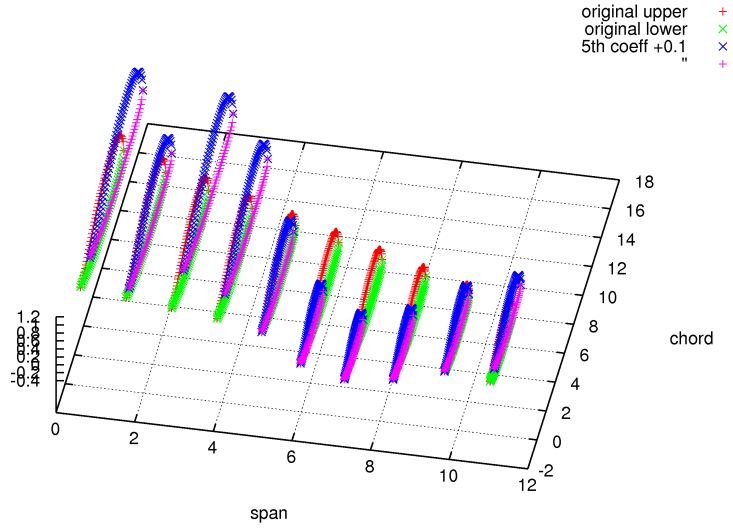


Figure 3.6: *Effect of adding more coefficients to the leading edge curve.*

3.2.2 BERP-like Rotor Tip Parameterisation

This parameterisation technique allows for the following design features to vary: (i) the sweep angle, (ii) the gradient of the BERP notch, (iii) the spanwise position of the notch. A schematic showing the variables used to modify these parameters is shown in Figure 3.7. To do this, both the leading and trailing edges of the BERP tip are modified. Referring to Figure 3.8, the leading edge is defined by three equations and the trailing edge by two.
For the leading edge, the first part is defined by a sigmoid curve that represents the notch region (Figure 3.8(a)). The sigmoid equation is given as:

$$y = \frac{\Delta y}{1 + e^{-g(x - x_o + \Delta x/2)}} \tag{3.11}$$

where Δy is the notch height i.e. the notch length in the chord-wise direction, Δx is the total width of the notch i.e. the notch length in the span-wise direction, g is the gradient of the notch and x_o is where the notch starts from. The x coordinate of the notch maximum is defined by the user and is kept constant except when the notch position needs to be varied. The g value is varied to change the gradient.

The second part is used to define the sweep (Figure 3.8(b)). It represents the part of the leading edge after the notch as a parabola:

$$y = -a(x - x_1)^2 + \Delta y + y_{add} \tag{3.12}$$

where a is the gradient of the parabola used to alter the sweep, x_1 corresponds to the notch end and the beginning of sweep, Δy is the notch height and y_{add} is an additional y offset value to ensure that the y ordinate of the parabola starts at the same position as the notch height. The value of y_{add} is computed automatically, once x_1 and Δy are known.

The third part describes the delta tip which joins to the trailing edge (Figure 3.8(c)). It is represented as a polynomial of order 2.5:

$$y = -b(x + c)^{2.5} - \Delta y' \tag{3.13}$$

where b is the gradient of the delta tip, c is the centre where the gradient of the curve becomes 0 (used to match the gradient of the curve to the previous parabola) and $\Delta y'$ is the additional y displacement required to match the curve to the previous parabolic curve. See Figure 3.8.

The two parameters, g and a can be changed independently and the rest of the parameters appearing in the equations are automatically adjusted so that the curves match at the point and the gradient level. These are the values of Δx, $\Delta y'$, c. The initial x co-ordinate, x_o is modified with the gradient of the parabola so that the tip point occurs at the same place for a required sweep. This is why for different notch positions, different sweep parameters are used to obtain the same sweep distribution. The gradient b is dependent on the trailing edge curve as well. Therefore the trailing edge must be determined first. The gradient of the trailing edge curve can also be modified independently of the sweep gradient of the leading edge. This allows the tip point of the blade to move in the y-direction which inherently modifies the chord distribution as well.

The trailing edge is defined first by a linear curve that has the same gradient as the leading edge sweep parabola or a scaled value of it, if required, and then by a polynomial of order 3.5 that is matched to the point and gradient of the sweep curve that comes before. The trailing edge curve must be specified before hand, as the tip point is required to find the gradient of the delta polynomial so that the leading and trailing edge curves meet at a single point. So the first curve for the trailing edge is given as

$$y = -a(x - x_1) + \Delta y + y_{add_{TE}} \tag{3.14}$$

And the latter part of the trailing edge is given by

$$y = -b_{TE}(x + c_{TE})^{3.5} - \Delta y'_{TE} \tag{3.15}$$

where all the values and constants correspond to the trailing edge parameters except for the sweep parameter, a which is exactly the same as the leading edge sweep. The gradient of the trailing edge can be increased or decreased relative to the leading edge sweep gradient by scaling it with a factor. Figure 10.6 show the examples used to build the design space for the optimisation.

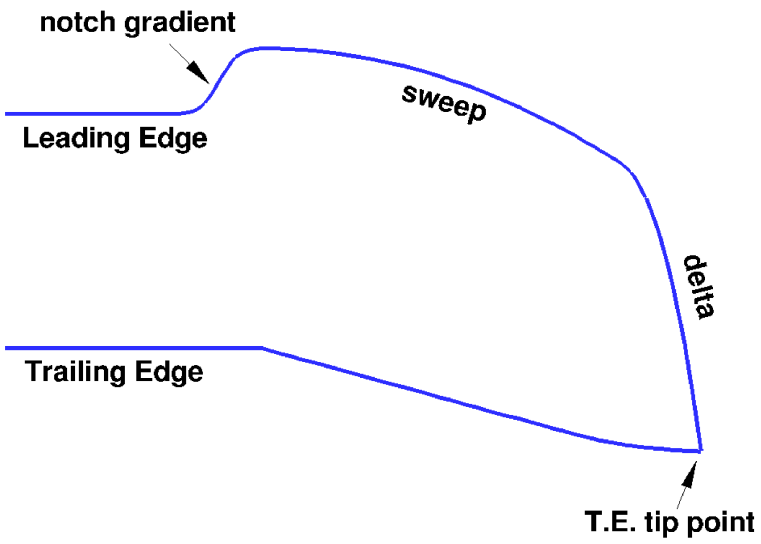


Figure 3.7: *BERP-like tip Schematic.*

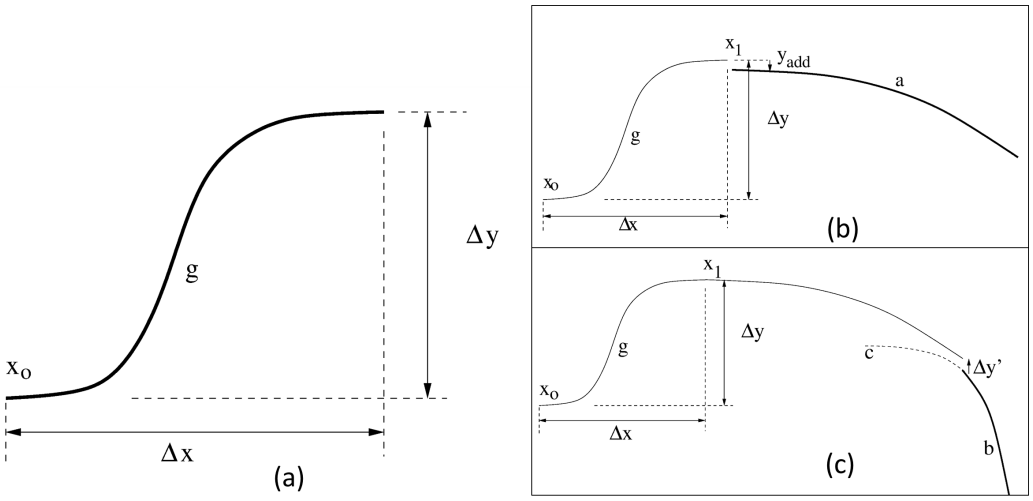


Figure 3.8: *(a) Notch gradient, (b) sweep and (c) delta parameter equation definitions for the BERP planform.*

3.2.3 Fuselage Parameterisation

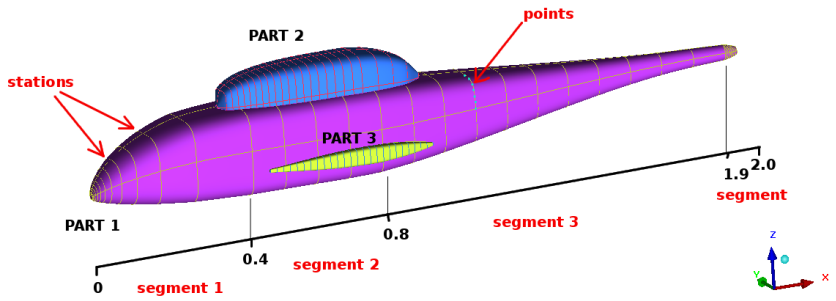


Figure 3.9: *ROBIN fuselage with additional parts.*

There are two parts to the parameterisation of the fuselage. First the parameters must be found to define the shape of the original fuselage design. The second part is to then determine which parameters to optimise and how the rest of the parameters are associated with the selected parameters. The method described here is with reference to the JMRTS fuselage developed by JAXA[107]. The parameterisation method is based on the super-ellipse equations used on the ROBIN body[105]. These equations are as follows:

$$\left(\frac{x+x_o}{A}\right)^n + \left(\frac{y+y_o}{B}\right)^m = C \tag{3.16}$$

The simplest form of this equation and one that is commonly known, is that of a circle where m = n = 2, A = B = 1, C is the radius squared and x_o, y_o is the coordinate of the centre. i.e.

$$(x-x_o)^2 + (y-y_o)^2 = r^2 \tag{3.17}$$

For this parameterisation, the longitudinal axis of the fuselage is always specified on the x-axis. The fuselage is then prescribed as a number of cross-sections along this axis, called stations (see Figure 3.9. Therefore, the varying co-ordinates of these stations are the y and z co-ordinates. The y and z coordinates at each station are obtained from the centre of the station (Y, Z), its height and width (H, W) and the curvature of the section at its corners (N). This latter parameter has the ability to make the cross-section vary from a diamond shape, through to a circle and to the other extreme of a square. These five parameters are defined as a function of x i.e. they are varied along the x-axis using the super-ellipse equations so that at each station (x coordinate), there is a specific, H, W, Y, Z and N from which its y and z coordinates can be obtained. For generalisation purposes, let the five parameter be called y for now. Then if Equation 3.16 is rearranged so that y is given in terms of x, it becomes:

$$y = B\left[C - \left(\frac{x+x_o}{A}\right)^n\right]^{1/m} - y_o \tag{3.18}$$

To represent y in polar co-ordinate form, the following must be true,

$$y + y_o = rcos\phi \tag{3.19}$$
$$x + x_o = rsin\phi \tag{3.20}$$

where r is the radius of the polar circle. Substituting these into Equation 3.18,

$$(y+y_o)^m = B^m\left[C - \left(\frac{x+x_o}{A}\right)^n\right] \tag{3.21}$$
$$(rcos\phi)^m = B^m\left[C - \left(\frac{rsin\phi}{A}\right)^n\right] \tag{3.22}$$

which after some re-arrangement becomes,

$$r^m A^n cos^m\phi + r^n B^m sin^n\phi = A^n B^m C \tag{3.23}$$

For this to be in polar co-ordinate form, m = n and C = 1. Therefore,

$$r^n \left(B^n cos^n\phi + A^n sin^n\phi\right) = A^n B^n \tag{3.24}$$

Therefore,

$$r = \left[\frac{A^n B^n}{B^n cos^n\phi + A^n sin^n\phi}\right]^{1/n} \tag{3.25}$$

where now, A = radius vertically = H/2, B is the radius horizontally = W/2 and n is the power of the ellipse which is N, the curvature at the corners. Therefore r and the y and z coordinates are given by,

$$r = \left[\frac{\left(\frac{H}{2}\frac{W}{2}\right)^N}{\left(\frac{W}{2}cos\phi\right)^N + \left(\frac{H}{2}sin\phi\right)^N}\right]^{1/N} \tag{3.26}$$

$$y = rsin\phi + y_o \tag{3.27}$$

$$z = rcos\phi + z_o \tag{3.28}$$

In this way, a number of parts can be obtained for a helicopter fuselage such as the main body, doghouse, sponsons, fin and tail plane and so on. Figure 3.11 shows an example of this on a ROBIN fuselage. The points are obtained for one side and then mirrored in the vertical plane. To do this, a systematic method has been developed. Figure 3.9 shows the terminology used to describe the parameters of a fuselage using the ROBIN body with sponsons added. The hierarchy is shown in Figure 3.10.

The first division in the hierarchy is the part e.g. the main body, the dog house or pylon,

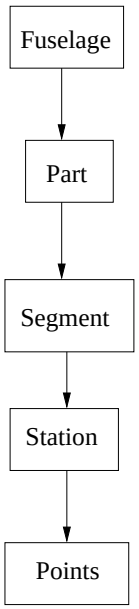


Figure 3.10: *Fuselage Parameterisation hierarchy.*

sponson 1 etc. Each of these parts is defined by segments. A segment has a set of super-ellipse coefficients that are used over that segment. In Figure 3.9, the main body is made up of four segments. Each segment consists of a number of stations, each with its own x co-ordinate and

finally, each station is made up of a number of points with y and z co-ordinates. This method was used for the parameterisation and optimisation of the JMRTS fuselage in Chapter 9. An additional parameter, N_b was added so that the curvature on the bottom surface can be different to the upper surface. The code simply reads an extra line of coefficients and applies it to when ϕ corresponds to the lower part of the geometry. Figure 3.12 summarises the process of generating a fuselage. More details can be found in the HMB Technical Note[129] . The related programs can be found in Appendix B.8.1 and B.8.2 and an example file can be seen in Appendix B.8.3 and B.8.4.

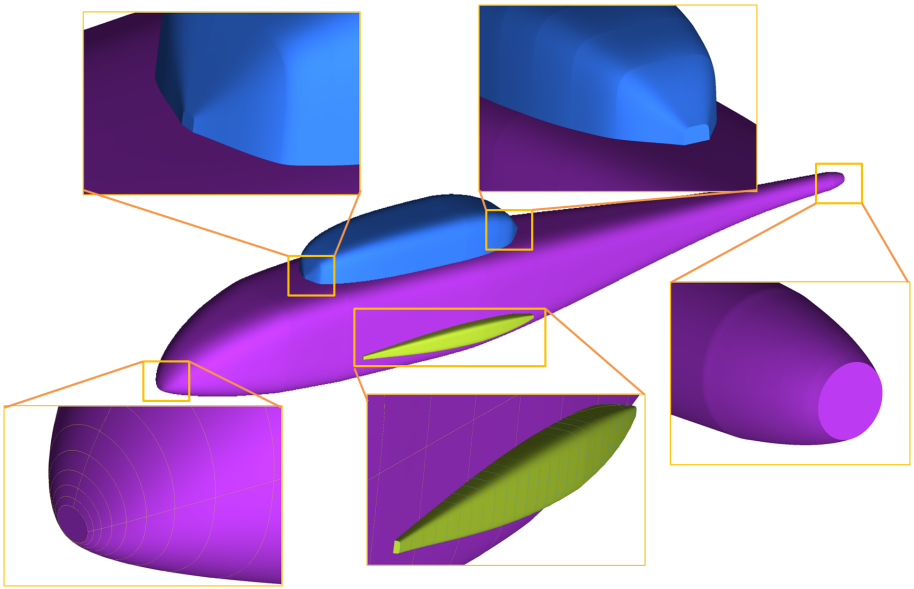


Figure 3.11: *Example of parts of a fuselage created using the parameterisation method.*

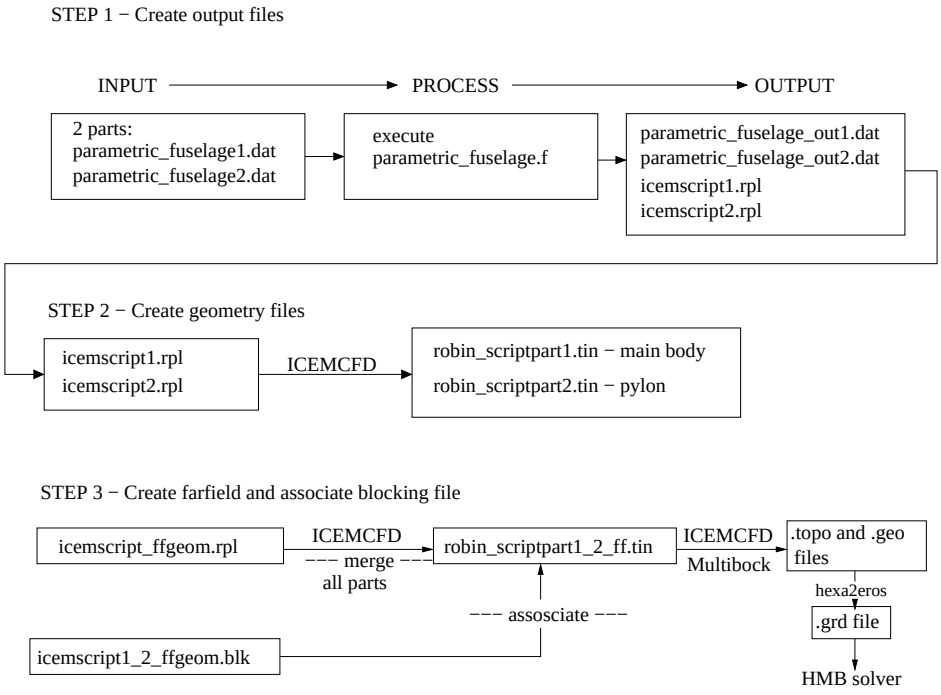


Figure 3.12: *Schematic of the mesh generation process.*

3.3 Sampling the Design Space

There are a number of ways that the database can be populated[130] . In this case three types of methods have been used dependent on the size of the optimisation case: the full factorial, fractional factorial and Latin hypercube sampling (LHS). These methods were used loosely and not in their strictest sense in order to achieve more accurate predictions from the metamodels.

The full factorial method analyses all the other variables for each design variable. Therefore it has the highest number of design points. The fractional factorial method uses every other point of the full factorial method i.e. it is a sparser version of the full factorial database. For this project, the points selected using this method were such that there was a higher concentration where the optimum was heading towards. This was found by repeating the optimisation a number of times with additional points in the database each time. This is also called adaptive sampling or updating[43] .

The LHS method attempts to use a design parameter value only once or as few times as possible in populating the database. So for example, if there are two parameters with equal number of design parameters, the LHS will select the design points along the diagonal line (or hyperplane for higher number of parameters) of the database. If there are more variables for one design parameter than another, then a random additional point is created for the component that has fewer variables. Now in the cases in the work where the LHS was used, in addition to the LHS points, points along the boundaries of the database were also included. This is why it was stated that some of the sampling methods used were not used in their strictest sense for various reasons as will be explained for each case individually.

The more points that exist in the database, the more accurate the metamodel predictions will be. This means that the full factorial method will always produce the most accurate results. However, for cases where the computation time and cost of obtaining the performance of each design point is high, it is important to reduce the number of such calculations by selecting as few points as possible as in the case of fractional factorial and LHS methods. These methods can produce good results if the points are selected carefully and if some additional points on the boundary of the database is included. Cameron et al.[62] used an automated adaptive sampling method to optimise laminar flow aerofoils. In his method, the Pareto front was updated after a set number of generations and the mean square error of the kriging metamodel was used to calculate the probability that a new sample added at any given point would dominate members of the existing Pareto front. This was automated in the genetic algorithm used, NSGAII[62] and repeated until a set sample size was obtained. Forrester et al.[131] also describe an optimal LHS method based on adaptive sampling. In this project, however, since the case is larger than an aerofoil, the process was carried out manually for more control over the selection points.

3.4 Objective Function

The objective function is a key part that determines the outcome of the optimisation. Ideally it should be[4] :

1. Complete so that all aspects of the decision problem are presented
2. Operational i.e. they can be used in a meaningful manner
3. Decomposable if disaggregation of the function is required
4. Non-redundant so that no aspect of the decision is considered twice
5. Minimal so that the function considers the minimum required for a decision

In line with these properties, the objective function is therefore presented as the summation of weighted components that quantify the objectives. However, the actual values of each component that makes up the objective function may vary by orders of magnitude, meaning that the objective

function would be biased based on the range of values, rather than the sensitivity of the value to performance. Therefore, each performance parameter is scaled with the corresponding value of a reference design which is usually the original design. So a generic objective function can be presented as:

$$OFV = \sum_{i=0}^{n} w_i \frac{P_i}{P_i^r} + C \tag{3.29}$$

where OFV is the objective function value, w_i is the weight assigned to the i$^{\text{th}}$ performance parameter, P_i is the i$^{\text{th}}$ performance parameter value for the design being evaluated, P_i^r is the performance parameter value for the reference design and C is a constant value added so that if the reference design was the design being analysed, the OFV would be 0 i.e. $C = -\sum_{i=0}^{n} w_i$.

The weights for the function are guided by the initial CFD data. Say for example, there are two performance parameters, P_1 and P_2, for an optimisation problem and the objective is primarily to improve P_1 and secondarily to improve P_2. For each design point a ratio can be found between P_1 and P_2. Assume the average of all these ratios is 1:1.5 for P_1:P_2. Then the limiting weight to weigh P_1 more than P_2 in the objective function is $1.5/(1+1.5) = 0.6$ or $w_1 = P_2/(P_1+P_2)$, i.e. on average, the weight of P_1 must be ≥ 0.6. However, this should only be used as a guidance value, as individually, the deviation from this ratio can be large. This method is a simple way to obtain some guidance in determining the weights, but its complexity can be increased for example, with the use of standard deviation and other attributes.

The objective function method differs from what is known as the Pareto method of optimisation. The Pareto method tries to find the best compromise in performance for the designs, i.e. in the Pareto subset, an increase in one performance parameter will result in a decrease in another. Therefore it creates a boundary or front of design points. The objective function method however, tends to concentrate the optimum designs to a cluster in the design space as opposed to spreading the optimum design along a front i.e. it selects designs in a region of the Pareto front. In this project, the Pareto front was found using the GA where the elite members of the population were the ones that fulfil the Pareto conditions, as opposed to a selection of the fittest individuals for the weighted method. In this work, both methods were used and the results show that the selected optima using the weights method were also members of the Pareto front. Both the Objective Function and Pareto Front GAs can be found in Appendix B.6.1 and B.6.3.

3.5 Metamodels

The cost of running the high-fidelity solver to analyse every new design created by the optimiser can be quite high in terms of computational effort and time. To avoid this, a lower fidelity approximation of the performance parameters based on the high fidelity data in the database can be used until the final result is obtained. This allows the optimiser to access the accuracy that comes with the high-fidelity model with the efficiency of the low-fidelity model or metamodel (model of a model). Four metamodels are described here.

3.5.1 Artificial Neural Networks

An ANN interpolates based on patterns obtained from a set of data, similar to how the biological brain learns[132]. Figure 3.13 is a schematic of the structure of a multilayer feed-forward ANN. It consists of a number of neurons connected to every other neuron in the next layer, from input to output[6]. The layers between the input and output are known as hidden layers and make up what is known as the perceptron. It is here that 'learning' takes place. Each neuron is associated with a weight and an activation function. The weight determines how much influence a neuron has on the output and the activation function keeps the values within bounds and gives the ANN the ability to be differentiable so that error corrections can be made using, for example as in this

case, a gradient descent method. The activation case used most commonly and in all the cases presented here is the sigmoidal function[132] generically shown in Equation 3.30 and in Figure 3.14,

$$y = 1/(1 + e^{-x}) \tag{3.30}$$

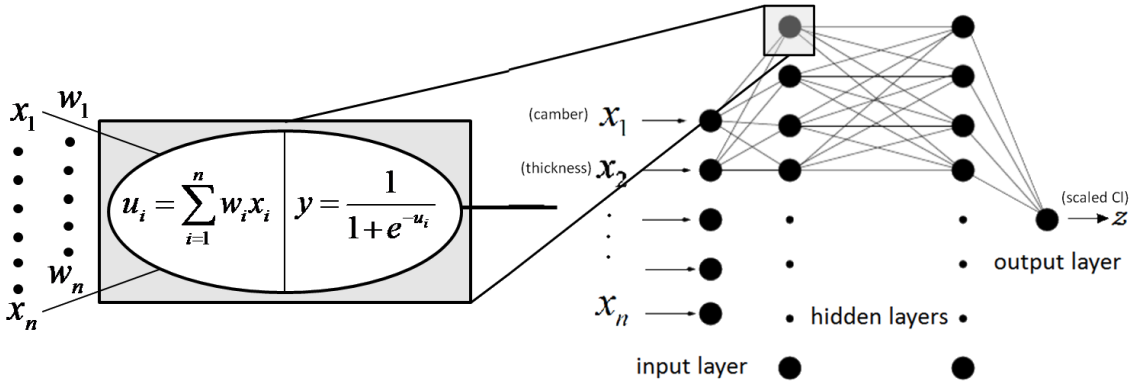


Figure 3.13: *An example of a neural network trained to receive inputs such as camber and thickness to obtain the lift coefficient (adapted from Spentzos[6]).*

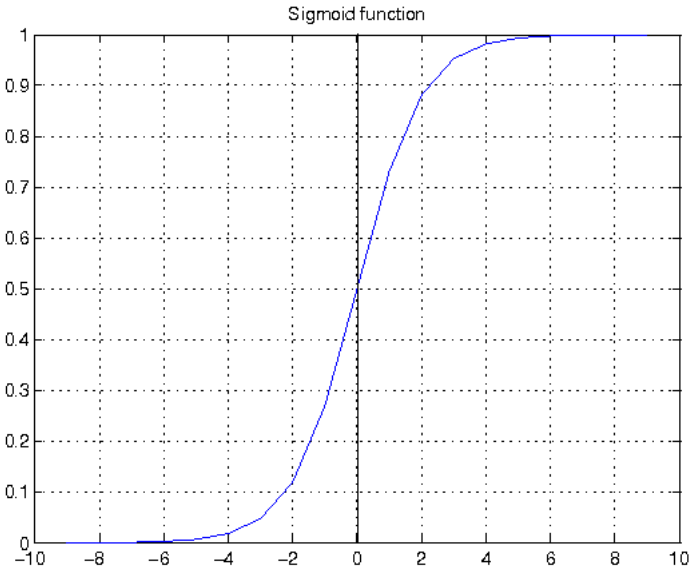


Figure 3.14: *Typical sigmoid function ($y = 1/(1 + e^{-x})$ from x=-10 to 10.*

There are two phases for ANNs: training phase and predicting mode. In the training phase, a training data set is available to the ANN, that is, both input and output. The weights of the neurons are randomly chosen and the ANN makes a prediction. The error between this predicted output and the target output is then fed-back through the layers and the weights are adjusted accordingly. The full set of data, known as an epoch, is fed in repeatedly and the error back-propagated until the error converges to a pre-set value. In this work, this is done via a conjugate gradient descent method.

For example, let's assume that there are n initial inputs of the ANN (i=1,2,...,n). If the hidden layers have J nodes each, then for the first layer, these are all connected to the n inputs through neurons with weights w_{ij} (i=1,2,...,n, j=1,2,...,J). Each input is first weighted with the

weight for that input and that neuron:

$$u_j = \sum_{i=0}^{n} x_i \times w_{ij}, i = 1, 2, ..., n \text{ and } j = 1, 2, ..., J \tag{3.31}$$

This is then passed through the activation function to get the output from that neuron:

$$y_j = \frac{1}{1 + e^{-u_j}}, j = 1, 2, ..., J \tag{3.32}$$

This is repeated at each node at every hidden layer until the final output layer is reached for each pattern. The optimum number of hidden layers depends on the complexity of the problem. If there are more dependencies between input parameters, then more layers are required to deal with these inter-dependencies i.e. for the ANN to be 'smarter'. In general, increasing the number of layers makes an ANN smarter and increasing the number of neurons per layer makes an ANN more accurate[6] . Also, it is critical that during every epoch, the patterns are introduced at a random order. This ensures faster learning, avoids 'memorising' and increases the capability of the ANN to tackle situations it has not been trained for[6] .

In the training phase, the difference between the target output and the predicted output (the error value), is used to determine the changes in the weights of the neurons through a back-propagation technique based on chain differentiation. For the ANN used in this project, the error was corrected after all patterns had been through the system, i.e. one epoch rather than after each pattern. The mean squared error can be found as

$$E = \sum_{i=1}^{K} |z_i - t_i| \text{ or as } \frac{1}{2}\sum_{i=1}^{K}(z_i - t_i)^2 \tag{3.33}$$

where K is the total number of outputs, t_i is the i$^{\text{th}}$ target output, z_i is the i$^{\text{th}}$ output obtained from the final output layer. The aim now is to find the output that minimises E. The method used is the gradient-descent with momentum (GDM) method[6] which works as follows.

The error change due to a change in the output is found by differentiating Eqn.3.33 and is given by

$$\frac{\partial E}{\partial z_i} = z_i - t_i \tag{3.34}$$

Therefore, the change in the error due to the change in the input to the node that created z_i is

$$\frac{\partial E}{\partial u_j} = \frac{\partial E}{\partial z_i}\frac{\partial z_i}{\partial u_j} = (z_i - t_i)(z_i(1 - z_i)) \tag{3.35}$$

where $\partial z_i/\partial u_j$ is found by differentiating Eqn.3.32 where z_i is the equivalent of y_j for the output layer. Similarly, the change in E due to a change in the hidden layer node input (which is similar to x in Eqn. 3.32 is

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial z_i}\frac{\partial z_i}{\partial u_j}\frac{\partial u_j}{\partial x_j} \tag{3.36}$$

$$= w_{ij}(z_i - t_i)(z_i(1 - z_i)) \tag{3.37}$$

This is repeated all the way to the first input and the total error calculated. Then the weights are changed by a small amount in proportion to the error calculated and the input to each neuron. So w_{ij} can be updated as:

$$\Delta w_{ij} = \eta\frac{\partial E}{\partial x_j} + \alpha\Delta w_{ij} \tag{3.38}$$

$$w_{ij} = w_{ij} + \Delta w_{ij} \tag{3.39}$$

In Eqn.3.39, α is termed the learning momentum as it scales up the change in the weight at each iteration (it is multiplied by the previous weight change), and η is termed the learning rate as it scales up the movement along the gradient. The higher these values are, the lower the training time required. However, too high values can cause instability and a failure to converge. To further improve the speed of training the ANN, the ANN is allowed to skip a number of steps periodically before the weights are modified as a ratio of the last written weights. A further improvement is the use of an adaptive learning rate. This means that the learning rate changes according to the change or gradient in error for each epoch or set of epochs which allows the training to occur faster when the ANN is learning quickly and slower if it is learning slowly. The learning rate, η, is multiplied by a fraction less than 1 if the ratio of the present error to the previous error falls above a certain value and greater than 1 if it falls below a certain value. In this way, η is constantly adapting to the ANN's learning ability.

Another important parameter to consider in training the ANN is the error convergence. This value must not be smaller than the error resolution of the data points used for training. For example, if the CFD data is accurate to within 0.01 of its value, then the ANN training must not force the difference between the predicted and target value to be less than 0.01. If so, the ANN prediction trends bend and flex to try and pass through each point inside that tolerance resulting in over-fitting and large training times. Figure 3.15 shows a comparison of ANN predictions when trained to different convergence values for the rotor section test case. This case is explained in more detail in Chapter 5. However, for the purpose of demonstrating the effects of the ANN parameters, it is explained briefly here. The aim was to optimise the camber and thickness values of a NACA 5-digit rotor sections for a blade in forward flight using the moment, lift and drag parameters. These loads are compared against the camber parameter here. As can be seen, the best compromise is a convergence value of about 0.01 in that the trend is captured. A smaller convergence value over-fits the data as it falls within the error tolerance of the training data itself. The length of time for training required increased exponentially with smaller convergence values. The smallest convergence took about 25 times longer than the intermediate which took approximately 6 times longer than the biggest convergence value. Also, using a fixed η took the ANN about 1.25 times longer to train.
Using the adaptive η results in a different weights file, but the differences in the overall predictions are negligible as shown in Figure 3.15.

The number of hidden layers (hl) and neurons (n) required to get accurate predictions is determined by the physics of the test case, the number of inputs and outputs[6] . It was found that if more inputs and output variables are used, then a 'smarter' ANN is required i.e. one which allows more degrees of freedom, therefore an increased number of layers and neurons are required. Having too many layers for a small number of inputs and outputs can cause over-fitting of the data. Figure 3.16 shows the predictions with different numbers of layers and neurons for a number of outputs belonging to the rotor section test case. The C_l and C_d curves show that when the output variables are increased, an increase in the number of layers or neurons has approximately the same accuracy as a single output prediction with less layers or neurons. The green curve is the most inaccurate because it has a large number of outputs and the least number of neurons and hidden layers. Increasing the number of neurons increases the accuracy of the predictions. This is seen more clearly in Figure 3.16(c) for the average C_m. For a single output, the ANN with 2 layers and 21 neurons has a more accurate curve fit than the ANN with 2 layers and 15 neurons. For the 7 output case for the same output in Figure 3.16(d), a higher degree of freedom is required and hence an increase in the number of layers is required to get a smooth curve fit through the data similar to the single output prediction. Increasing the number of neurons does not improve the prediction accuracy.

This shows the importance in validating the metamodel before using it in the optimiser. In this project, the metamodel is always validated with additional high-fidelity data unknown to the

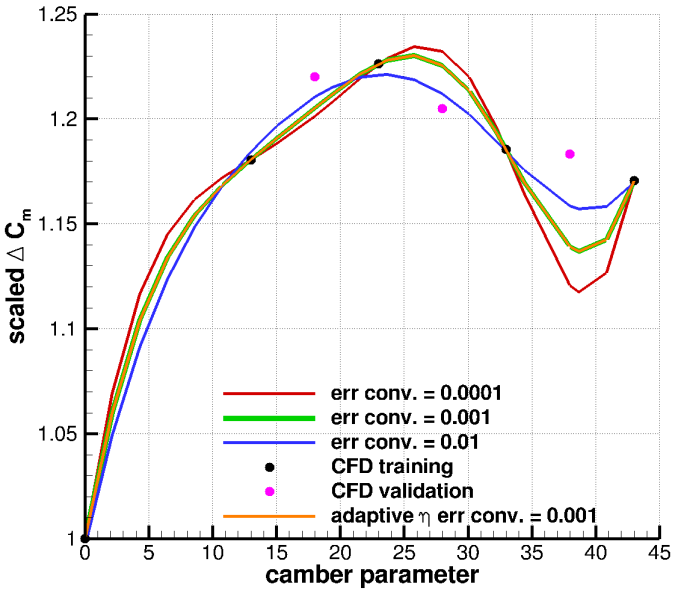


Figure 3.15: *Error convergence comparison for an example ANN. This is for the optimisation of a NACA 5-digit rotor section on a blade in forward flight. The x-axis is the camber value of the section and the y-axis is the peak-to-peak moment of the section over a full revolution. The various curves and the legend correspond to the training of the ANN to converge at different error values as well as a comparison with the ANN that had an adaptive learning rate. The black dots represent the data used for training and the pink dots are the accurate CFD data not used in the training set. i.e. validation data. The full test case analysis can be found in Chapter 5.*

metamodel. Once the ANN is trained, it can then be used to accurately predict the output for any input within the limits trained with.

The ANN was coded in fortran and can be found in Appendices B.3.2 - B.3.5. It is a modification of the work done by Spentzos[6] . For large values along the x-axis of the activation function i.e. for large input values, the resolution of the output is not good (see Figure 3.14). Hence, for more accurate results, the inputs are normalised to between 0 and 1 where the resolution is much better. To do this, the program *norma.f* (see B.3.2 in Appendix B) is used. If input data that is not within the range of the original data is required for the prediction stage (i.e. extrapolated of data), then these inputs must be included in the file to be normalised, so that *all* data is always between 0 and 1. These inputs for which the outputs are unknown, can be deleted from the data file in the training stage and re-introduced in the prediction stage. However, it has been found that the ANN is highly inaccurate in predicting extrapolated data, therefore it is suggested that the outputs of the input parameters on the boundaries of the design space are always known values. Once the inputs have been normalised, the normalised data are used to train the ANN using the program *nn.f* (see B.3.3 in Appendix B). Training can continue until a set epoch value or until the values converge to a pre-defined error value between the predicted and target results. In this case, training is always carried out until convergence to a specified error margin.

After training the ANN, the final weights are contained in a file. This file is used to predict the output for a normalised set of inputs using the program *predict.f* (see Appendix B.3.4) and then the predictions are denormalised using the *denorma.f* program (see Appendix B.3.5) to obtain the predicted results.

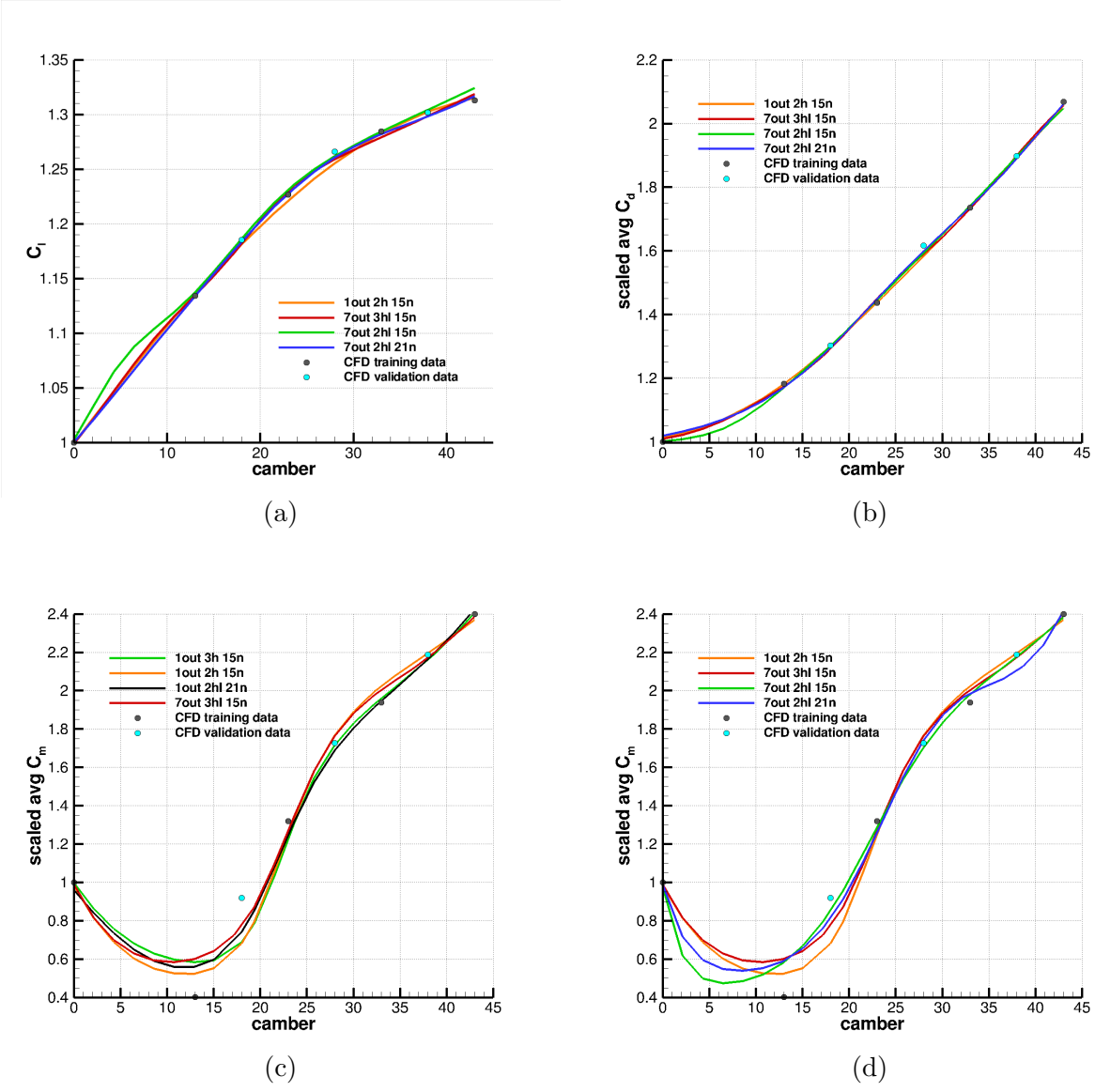


Figure 3.16: *Comparison of ANN prediction for scaled average C_l, C_d and C_m with different numbers of outputs (out), hidden layers (hl) and neurons (n). The number of inputs is constant at two. This is for the optimisation of a NACA 5-digit rotor section on a blade in forward flight. The x-axis is the camber value of the section and the y-axis is the average moment, lift and drag of the section over a full revolution. The various curves and the legend correspond to the training of the ANN with different numbers of neurons and layer. The black dots represent the data used for training and the cyan dots are the accurate CFD data not used in the training set. i.e. validation data. The full test case analysis can be found in Chapter 5.*

3.5.2 Polynomial Fit

Given a number of points, the aim is to fit a polynomial curve through them or create a polynomial equation that satisfies all the points given. Hence, the more known points there are, the more reliable the predictions are and the greater the degree of the polynomial can be. For example, if only two points existed, only a straight line could be used to predict data that falls anywhere on the plane. If three points were known, then a quadratic curve could be fitted through the points and hence the predictions are more accurate and so on. The maximum degree of the polynomial is dependent on the number of points available, the limit being due to the number of unknowns in a set of simultaneous equations. If (x_1,y_1) (x_2,y_2) and (x_3,y_3) are three points on a plane and a quadratic polynomial is to be fitted through these points, then,

$$Ax_1^2 + Bx_1 + C = y_1 \tag{3.40}$$
$$Ax_2^2 + Bx_2 + C = y_2 \tag{3.41}$$
$$Ax_3^2 + Bx_3 + C = y_3 \tag{3.42}$$

Using Gaussian elimination or LU decomposition or any other such technique to solve simultaneous equations, the coefficients A, B and C can be found.
It is possible to use a polynomial to fit nonlinear data. However, if the data do not follow a polynomial trend, having too high a degree of polynomial can result in over-fitting the curve. Therefore, normally, the degree of polynomial is limited to 6 or less[133] .

Figure 3.17 shows a comparison of a polynomial fit for the same data used to analyse the ANN in Section 3.5.1. With a polynomial of order 4, the data are not as accurate as the ANN, but with an order of 10, the polynomial fits closer, although it is less smooth and still not quite as accurate. The advantage of the polynomial technique is that it does not require training time as the ANN does. However, for the sake of accuracy, as it is not likely that more than 4 or 5 known points will exist for a variable, which limits the order of the polynomial, the ANN will be selected in preference to this technique.

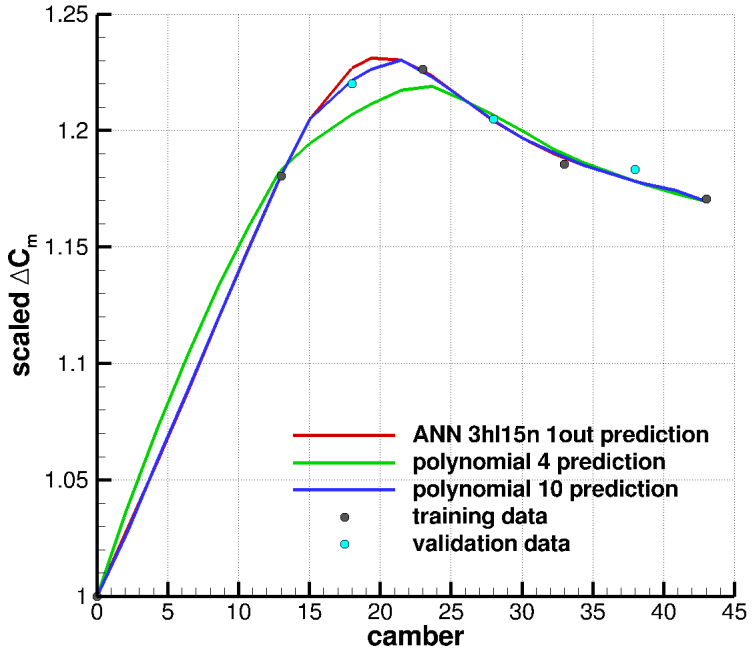


Figure 3.17: *Comparison of ANN prediction accuracy with polynomial of order 4 and 10 for* ΔC_m. *3 hidden layers (hl) and 15 neurons (n) were used for the ANN. Number of inputs is kept constant at 2. The solver training and prediction comparison is included.*

The polynomial fit method was coded with MATLAB and can be found in Appendix B.5. The method can be very effective if a law exists for the variation of the output. For example if C_d versus C_l follows a parabola, then polynomial fit can be used.

3.5.3 Kriging

The kriging approximation method is a more complex version of the polynomial fit method. It uses sample data points to build a model that can be used to predict the output or performance of interpolated design points by fitting a low-order polynomial through the data points but allowing the predictions along these polynomials to deviate based on a correlation model such as a Gaussian distribution of all the existing points. The Gaussian distribution's characteristics are based on the correlation between the sample points i.e. on their proximity to each other. This allows the kriging parameters to change with the prediction point giving it more flexibility while still maintaining accuracy[41] . Assume the output or performance, P is represented as:

$$P(x, y) = f(x, y) + Z(x, y) \tag{3.43}$$

where $f(x, y)$ is the polynomial and $Z(x, y)$ is the correlation model that is based on the Gaussian distribution for all the cases in this project. The correlation between all the sample points with each other and the correlation between the required point and all the points is found as the Gaussian distribution of the distances between all these points with a 'roughness' parameter, θ for each design parameter. So, Z is actually a function of θ as well as x and y i.e. $Z(\theta, x, y)$. All the parameters and values are normalised and then denormalised at the end so that the mean of $Z(x, y)$ is 0. Also, the data are normalised in a scalar way over each parameter between 0 and 1. So the correlation between P and F can be described as:

$$F\beta \approx P \tag{3.44}$$

where β can be approximated as:

$$\beta = \left(F^T Z^{-1} F\right)^{-1} F^T Z^{-1} P \tag{3.45}$$

and the variance can be estimated as:

$$\sigma^2 = \frac{1}{m} \left(P - F\beta\right)^T Z^{-1} \left(P - F\beta\right) \tag{3.46}$$

where m is the number of initial data points. The matrix Z, and β and σ^2 depend on θ. θ is effectively a width parameter that determines how far the influence of a data point extends[43]. It is similar to a weight put on each input depending on its influence on the output. A low value means there is a high correlation and the influence of each point affects many other points. A large value means there is less correlation and hence it has less influence on another point. In this way, θ can also be used to find out the design parameters that have the most influence on the performance parameters.
The optimum value of θ is defined as the maximum likelihood estimator, the maximiser of

$$-\frac{1}{2}\left(mln\sigma^2 + ln\left|Z\right|\right) \tag{3.47}$$

where $|Z|$ is the determinant of Z[41]. This value is found iteratively[41], although for small cases with few inputs, a good estimate can be made by comparing predictions with the original data. Since this is the case for this project and to quicken the kriging process, this process was used to obtain θ.
The Gaussian correlation function, $Z(x, y)$ can be found as the covariance matrix:

$$Cov(i,j) = exp\left(-\sum\left[\theta_x(x_i - x_j)^2 + \theta_y(y_i - y_j)^2\right]\right) \tag{3.48}$$

$$\text{or} \quad exp\left(-\sum\left[d^2\right]\right) \tag{3.49}$$

$$Cov(i,r) = exp\left(-\sum\left[\theta_x(x_i - x_r)^2 + \theta_y(y_i - y_r)^2\right]\right) \tag{3.50}$$

where i and j are the sample points and r is the required points, x and y are the design parameters and d is the distance between the points, which in this case is squared. Actually, the value to which it is raised can be between 0 and 2 and represents the differentiability of the response function with respect to the design parameters. Values close to 0 indicate that the function is not differentiable or smooth and closer to 2 is for differentiable functions[41].
The weight matrix, λ, that contains all the weights to obtain the required point from each point in the design space, can then be found as

$$\lambda = Cov(i,j)^{-1} Cov(i,r) \tag{3.51}$$

Then the prediction can be made as the summation of each appropriately weighted objective function value

$$\hat{P_r} = \sum_{i=0}^{n} \lambda_i \times P_i(x,y) \tag{3.52}$$

More details can be found in the references[41,134]. For the polynomial fit, up to order two polynomials are common. In some cases, constant values are sufficient, such as the mean value of all the data.

Figure 3.18 compares the data in Figure 3.17 as predicted by the polynomial, ANN and kriging metamodels. The kriging method tends to smooth the surface out more than the ANN, although this can be dealt with by tweaking the parameters. Nevertheless, the ANN and the kriging both seem to perform consistently well across all the cases with little change in parameters, slightly more so for the ANN than the kriging method. The advantage of the kriging is that training

time is not required. Both methods are capable of making reasonably accurate predictions. An example of the fortran kriging program is included in Appendix B.4.1.

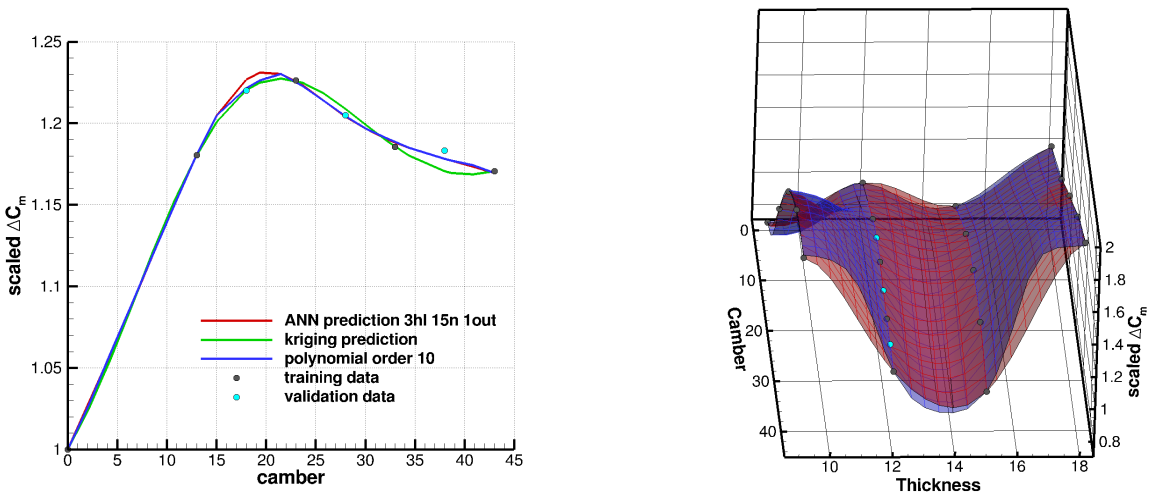


Figure 3.18: *Comparison of ANN prediction accuracy with polynomial of order 10 and kriging for ΔC_m. 3 hidden layers (hl) and 15 neurons (n) were used for the ANN. Number of inputs is kept constant at 2. The solver training and prediction comparison is included. ON the right, the blue surface is the ANN prediction and the red is the kriging prediction.*

3.5.4 Proper Orthogonal Decomposition (POD)

This is a mathematical technique that is used in many applications to compress data[135] . An analogy can be made with the Taylor series or the Fourier transform, where the part of the infinite equation that does not add significantly to the data is neglected. In the case of fluid dynamics, the flow is decomposed into modes. The modes that make major changes to the flow are retained and the rest are ignored.

The principle behind POD is that any function can be written as a linear combination of a finite set of functions, called basic functions. There are a number of different ways of performing this decomposition two of which are the Karhunen-Loève Decomposition (KLD) and Singular Value Decomposition (SVD) methods. The SVD method can be described as follows:
Let A be the data matrix, an $m \times n$ matrix where $m > n$. The SVD equation of A is:

$$A = USV^T \tag{3.53}$$

where U (whose columns are called left singular vectors - output basis vectors for A) is an $m \times mo$ matrix, S (whose diagonal elements are called singular values ordered in decreasing value order) is a $mo \times mo$ and V^T (whose rows are called the right singular vectors - input basis vectors for A) is $mo \times n$ matrix. More details can be found in Lawson[135] .

The Karhunen-Loève expansion is usually used to represent stochastic processes (i.e. a family of random variables with respect to time) as a combination of random deterministic time functions. The method of snapshots for the KLD method was introduced by Sirovich[136] . A snapshot is a set of data at certain spatial points at one particular time. So, the data matrix consists of two dimensions, the first being the spatial grid point data and the second being time. The snapshots are taken at regular intervals over the period of flow. In the KLD method, any snapshot can be expanded using eigen functions. Using velocity as an example,

$$u(x,t) = u_m(x) + \sum_{i=1}^{N_t} a_i(t)\Phi_i(x) \tag{3.54}$$

where $u_m(x)$ is the mean velocity, $\Phi_i(x)$ is the spatial KLD mode and $a_i(t)$ is the temporal KLD eigen function. To generate this decomposition, the first step is to calculate the covariance of the input data matrix,

$$B = (1/N)(A^T A) \tag{3.55}$$

The resulting matrix is symmetric and has dimensions of $N \times N$. The eigenvalues and eigen vectors of the covariance matrix are then computed. The matrix containing the eigen vectors are temporal KLD eigen functions $a_i(t)$. The original data matrix is then multiplied by the eigen vector to produce $\Phi_i(x)$. The eigenvalues represent the amount of energy stored in the mode i.e. its contribution to the overall flow.
Typically enough modes are stored so that 99% of the energy is captured:

$$\sum_{n=1}^{m} \lambda_n / \sum_{n=1}^{\infty} \lambda_n > 0.99 \tag{3.56}$$

Generally the KLD mode is faster and requires fewer modes to reconstruct the flow. Also, it is less sensitive to load imbalance and hence larger block sizes can be used. Even with increased number of snapshots i.e. larger matrices, the KLD execution time increases slower than the SVD method. The number of snapshots has a larger effect on execution time than the number of spatial point data. In addition, a method was found for compressing data as it is obtained for each time step from the solver, using the SVD method. No such method was found for the KLD[135] .

The POD described is not only used as a compression tool to reduce the size of data stored, but can also be used to predict missing data for required inputs, what has come to be commonly known as 'gappy' data[55] . From the theory shown, the model is reconstructed using the eigen functions that carry the highest energy. In this case, the aim is to be able to predict the loads of aerofoils that were not solved for using CFD. There are two methods of doing this: the POD/-Galerkin method and the POD/interpolation method. The POD/interpolation method is more suitable with data that does not have much correlation[137] .

The field data is reproduced as follows:

$$U = \sum_{i=1}^{p} \alpha_i \Phi^i \tag{3.57}$$

where U is the data that exists, p is the number of modes used for reconstruction, α_i is the i^{th} temporal POD coefficient and Φ^i is the i^{th} spatial POD basis vector.

The file containing the original data set (gappy file), U, and the required file are usually organised as the value at each spatial point down i.e. each row represents the value of U at a different point in space, and for each snapshot or time step across i.e. each column represents a time step so that you have a matrix of $n_{space} \times n_{snapshots}$. Using for example, the ΔC_l values for the aerofoils, the thickness values could replace the spatial data points and the camber values as the snapshots. So for example, the case described in the ANN section (Section 3.5.1) earlier is used where ΔC_l is the data matrix for A and it would look like this:

	0	**13**	**23**	**33**	**43**
9	0.977276	1.033416	3.000609	5.007123	7.083982
12	1.000000	0.642496	2.269264	3.946593	5.717450
15	1.137035	0.074969	1.296992	2.585237	4.025577
18	1.407559	0.693965	0.077982	0.971107	2.044992

First, a mask vector, n^k must be created which describes where the data is missing as follows:

$$n_i^k = 0 \text{ if } U_i^k \text{ is missing or incorrect} \tag{3.58}$$

$$n_i^k = 1 \text{ if } U_i^k \text{ is known and available} \tag{3.59}$$

where U_i^k is the i^{th} element of the vector U_k. Let g be the solution vector that has some elements missing. Then,

$$\tilde{g} = \sum_{i=1}^{p} b_i \Phi^i \tag{3.60}$$

where $\tilde{g}$ is the repaired vector. The error between the data that exists in both the solution with the missing data and the repaired vector is given by E,

$$E = g - \tilde{g} \tag{3.61}$$

Only the existing data are compared using the mask vector to mask out the new inputs. The coefficient b_i is varied so as to reduce the error. b can be found by differentiating Equation 3.61 with respect to b such that

$$Mb = f \tag{3.62}$$

where $M_{i,j} = (\Phi^i, \Phi^j)$ and $f = (g, \Phi^i)$. Solving this for b, g can be obtained and the missing data inserted into the original data matrix[54] .

Now, if a number of snapshots are missing i.e. entire capsules of data, then the first step is to fill in the missing data with random values or average values for the required points. Use this data matrix to obtain the POD basic vectors. As described above, use these matrices to obtain a corrected data matrix from Equation 3.60. The values from these intermediate repaired data are now used to reconstruct the missing data for the next iteration. This is continued until the algorithm converges or the maximum number of iterations is reached. For a more in-depth explanation of the KLD method, see the work done by Ly and Tran[138] .
This method however, is reliant on the availability of data in as many positions as possible and requires at least two elements of data in each snapshot. It also does not perform well with few data points. With smaller files, within each snapshot, if in addition to the required point, another point is missing, the accuracy of the prediction is highly compromised. For example, take the load, average C_m at a thickness of 12% and camber of 23 for the rotor optimisation case described in Section 3.5.1. Compare the data shown below. The difference is more than double the actual value. This large effect is because the initial matrix is small and so every point missing represents a large portion of the initial matrix.
For the matrix shown earlier predictions were made for the aerofoil of camber 33, thickness 12

Prediction method	Avg C_m
CFD	**1.296992**
1 point missing	1.28567
2 points missing	3.47769

and 18 using an ANN and the gappy POD method and compared. The comparison is shown 3.19 and as can be seen, the POD does not perform well with sparse data. Even with a greater number of points such as 20 variables for each design parameter, which is far above what is practically viable with high-fidelity CFD software, the ANN has superior performance as demonstrated here. The ANN predictions of 21 points of average C_m were used to show the POD's dependency on amount of data available to it, shown in Figure 3.20. With one point missing, the error is small, but as the number of missing points increases, the error grows. Also, the position of the missing data plays a role. If there are many points missing in one location as opposed to points spread over the full domain, the error can be larger as shown in Figure 3.20. This is one of the main advantages that the ANN has over the POD interpolation method in that even though ANNs require more time for training, the POD would require more points for higher accuracy which reduces its efficiency.
The gappy POD method was originally used to reconstruct images that were 'damaged' or unclear such as face recognition, satellite maps and CFD flow field data. In these cases, more data is

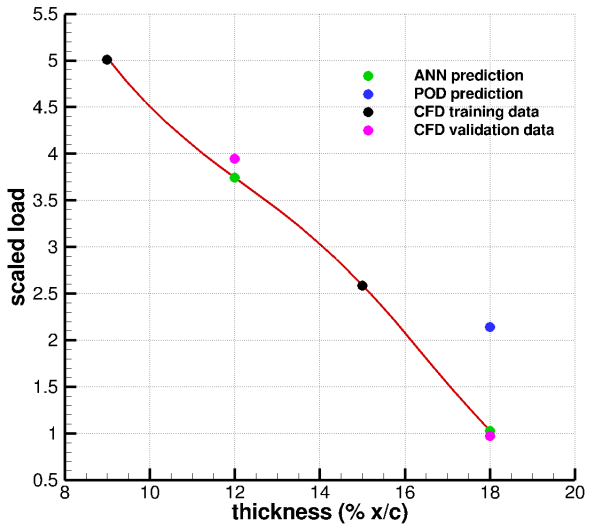


Figure 3.19: *Comparison of ANN prediction and POD prediction trained with the original database of 20 CFD points less the two that are predicted.*

available and at a comparatively smaller resolution. Therefore, the POD interpolation technique works well. However, in the case where very little data is available (which is expected when high fidelity CFD data is required) the trends are not predicted as accurately as other methods are able to predict.

The gappy POD method was coded in MATLAB and be found in Appendix B.2.1. It is self-contained within MATLAB and uses in-built functions of MATLAB. The POD method was also coded for other applications in HMB using fortran math packages. More information can be found in the POD Technical Note[139] .

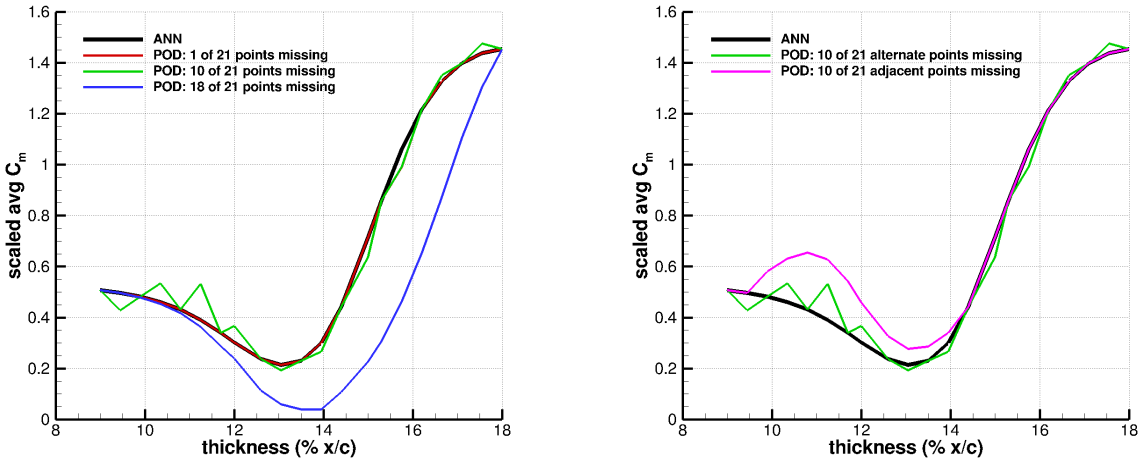


Figure 3.20: *Comparison of original data (ANN predictions of average C_m for varying NACA aerofoil thicknesses at R =50% - see Chapter5) with POD interpolation predictions comparing number and position of training data points.*

3.6 Genetic Algorithm Optimisation Method

For the optimisation, a non-gradient method in the form of a genetic algorithm (GA) was implemented and combined with the metamodels. Figure 3.21 shows the analogy and terminology used as applied to the optimisation of an aerofoil case: Selection, Crossover, Mutation, Competition, Survival. Two parents are first selected based on a roulette wheel technique. The roulette wheel

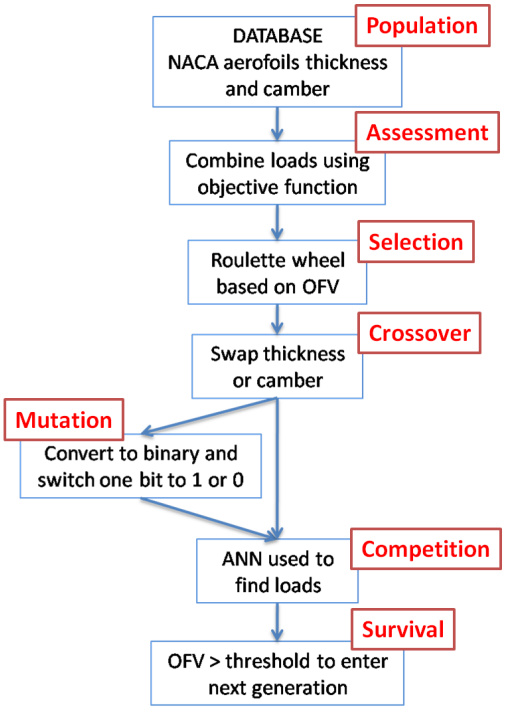


Figure 3.21: *Outline of the genetic algorithm employed for an aerofoil selection case and the analogy with genetics.*

is a file containing the full population of design points. However, each design point takes up as much space (i.e. it is repeated) in the file as is proportional to its fitness or performance by duplication i.e. the wheel is biased towards fitter individuals. A random selection is made from it, but since the wheel is biased, the evolution leads to better designs being created. The proportionality function for space on the roulette wheel is user-defined from linear to exponential and is necessary for convergence and stability. This is because if the number of individuals in the database is very high, the percentage of space taken up by fitter individuals on the roulette wheel reduces and the selections become less biased and more random. Therefore a better fitness assignment rule would be an exponential one rather than a linear one for example.

Once the parents are selected, their 'genes' are swapped or crossed over. A number of crossover methods have been developed. The most commonly used is random crossover and it is used in this project. Here, a gene or more are randomly selected and swapped producing two new offspring. In the illustration in Figure 3.21 either the thickness or camber is selected randomly and swapped between the two aerofoils.

For the mutation stage, the offspring parameters are converted to binaries of 10 bits analogous to genes. This representation is simple and effective. In the natural world, the genome is the most basic form of discretisation - similarly in the computing world, binary is the basic form of discretisation. A random bit is then chosen and changed to either 1 or 0. Mutation is necessary since it has been found that after a number of generations, some characteristics of the genes get 'lost'[64]. Mutation allows for these characteristics to be re-introduced into the gene pool and it also increases diversity which allows the global optimum to be found. Its probability is kept low by allowing only one point to be changed and there is a 50% chance that it will be changed to a

different value. So the probability of 1 bit changing is 1 in 20 for 10 bits. This prevents the change in the phenotype from happening too often which will cause the process to lose its evolutionary driving force. The probability of mutation can be increased by selecting more than 1 point for mutation or by switching the bit in the gene rather than assigning it a value it may already have or by swapping two bits in the code.

The resulting offspring is then assessed by employing the trained ANNs (along with its normalisation and denormalisation functions) and combining their output using the user-defined objective function. This objective function is then modified to stay within the user-defined constraints. There are two types of constraints exercised. One type limits the boundaries of the design space so that the predictions by the metamodel are accurate. The other includes physical constraints such as geometric e.g. thickness, minimum tip chord length etc. and aerodynamic constraints e.g. angle of attack, stall margin, drag-divergence Mach number. There are two ways to deal with these constraints - either as hard constraints or as soft constraints[50] . An option is set in the program that gives the user the ability to determine whether a constraint should be treated as a soft or hard constraint[140] . Soft constraints are beneficial initially as they increase the diversity of population preventing the GA from terminating pre-maturely before the global optimum is found.

After a number of such iterations, a pool of the offspring characteristics is created and a threshold value is set so that only the majority of the fitter individuals survive and pass on into the next generation pool. The fittest individuals are always carried through into the next generation. This is termed elitism. While the GA can converge without the help of elitism, the convergence takes longer and has a lower probability of being the global maximum since only mutation is capable of re-introducing new design characteristics or 'alleles' back into the pool and these may not be the best designs. Using elitism ensures that the best genes still exist in the gene pool (or 'live longer') and hence there is a higher probability of reaching the maximum value[64] . Cloning is avoided as it can change the selection process unfairly.

The genetic algorithm was coded in C and can be found in Appendix B.6.1. A script (Appendix B.6.2) is used to run it for the required number of generations. Details of how to operate the GA and its associated files can be found in the Optimisation Technical Note[140] , mentioned in the list of publications from this thesis.

3.7 Pareto Front Optimisation

The optimiser used here employs an objective function to determine the optimum and so the selection pressure is towards a small area in the design space. However, another way of finding the optimum is to find the designs that provide the best compromise between all the performance parameters. This is known as the Pareto front. The advantage of using a Pareto-front-type optimiser (PFO) is that it provides you with a range of design points that represent the best combination of performance measure parameters. In essence, the PFO method provides the user with the designs that give the best performance for each parameter and subsequently leaves the weighting of these performance parameters to them at the end of the optimisation process.
The advantage of using an objective-function-type optimiser (OFO) is that it allows designs that do not necessarily fall on the Pareto front to be included in the genetic algorithm's selection based on the end-point required. So, rather than spreading out the performance along a front, it clusters it at a specific objective.
For example in Figure 3.22, if C_M has less value in the objective of the design, point 2 is still a good design even though it may not fall on the Pareto front. However, having the Pareto front is a good indicator that there may be a slightly better design, possibly at point 4.

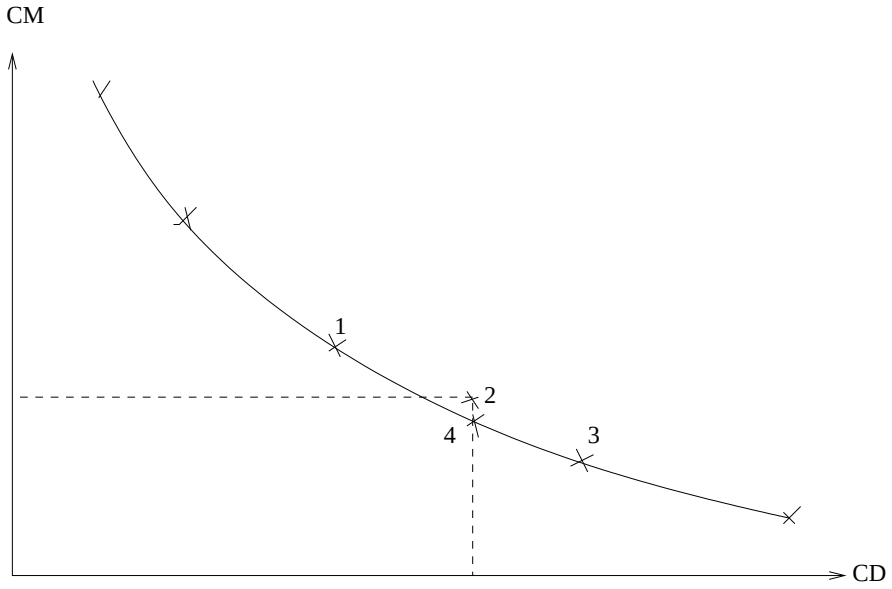


Figure 3.22: *Pareto front and objective function type optimisers.*

Chapter 4

Aerofoil Optimisation

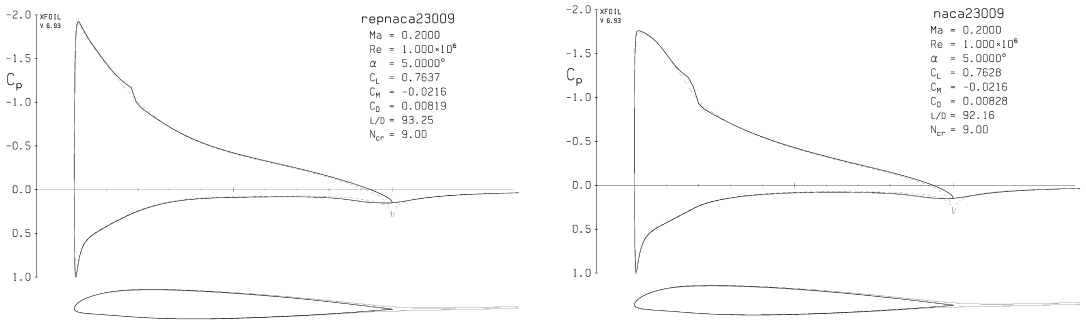
To demonstrate the optimisation framework on aerofoils, the RAE 2822 transonic aerofoil was used. The reason for using this aerofoil is that it is designed with a very specific objective in mind, which is to have good performance in terms of lift and drag at high Mach numbers, especially in the transonic to supersonic flow regime[141] . This makes it a good candidate for optimisation since its objectives are clear, and there is potential for further improvement within constraints such as lift-to-drag ratio and moments. The added advantage of using this aerofoil is that the experimental data on this aerofoil at these conditions are readily available. Also, it does not have a flexible parameterisation method and so this provides an opportunity to develop a generic method usable for shape optimisation. This parameterisation technique uses Chebyshev polynomials and is described in Section 3.2.1. Its application to the upper surface of the RAE 2822 aerofoil is also shown in Figure 3.2.

4.1 Parameterisation Technique

First the method was applied to a variety of aerofoils (NACA 0012, RAE 2822, NACA 23009 and ONERA OA213), and their C_p distribution compared in order to find what error convergence values were needed to obtain a shape curve that was accurate enough to produce the same results. For simple aerofoils like the NACA 0012, the parameterisation technique works very well with almost zero error in the shape using only a few coefficients. However for more complex aerofoils like the NACA 23009, the OA213 and the RAE 2822, more coefficients are required. Figure 4.1 is a C_p plot obtained from XFOIL for the NACA 23009 aerofoil and that of the reconstructed one. The error of the shape was 0.013728 with 3 coefficients and 0.005775 with 5. The moment values tend to be the most sensitive to the aerofoil shape. For both the actual and the reconstructed aerofoil, the values are within about 0.2%. The maximum error was for drag, approximately a 1% difference. For the ONERA OA 213 aerofoil which is used on wind turbines, the error was 0.068819 with 3 coefficients and 0.027594 with 5 coefficients. Figure 4.2 is the corresponding C_p plot. The error in moment coefficient is over 16% which is high suggesting that this error convergence value for the geometry is too large.

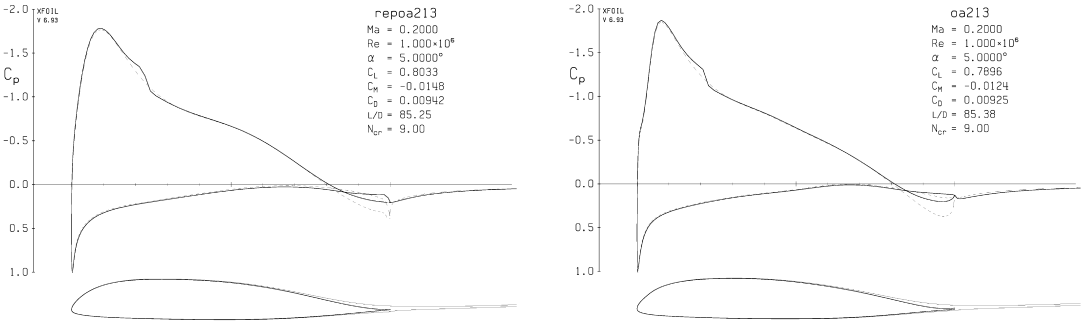
Therefore an error convergence less than approximately 0.01 is selected. However, it is also good practice to look at the recreated curve, even if the error convergence falls within this value. This is because the error is the sum of the absolute difference along the curve, therefore the same value can be obtained whether the error is a small but well distributed difference in the curves or a very accurate curve that has a large localised error, in which case the former would most likely produce better results.

For the RAE 2822 aerofoil, six coefficients were required to parameterise the upper surface with an error convergence of 0.00741. Figure 4.3 shows the C_p curve for the original RAE 2822 aerofoil and its parameterised version. The difference is minimal resulting in a difference in C_l, C_d and C_m of negligible values as shown in Table 4.1.



Parameterised NACA 23009 Original NACA 23009

Figure 4.1: *C_p plots predicted by* XFOIL *for the parameterised aerofoil NACA 23009 (a) and the original NACA 23009 aerofoil (b) at Mach 0.2 and Re = 1×10^6.*



Parameterised OA213 Original OA213

Figure 4.2: *C_p plots predicted by* XFOIL *for the parameterised ONERA aerofoil OA 213 (a) and the original ONERA aerofoil OA 213 (b) at Mach 0.2 and Re = 1×10^6. The error convergence was about 0.03 resulting in large differences in moments especially. This suggest a smaller error is required and hence a value of 0.01 was chosen.*

4.2 Objective Function

The objective was to improve the lift for a reduced amount of drag. Therefore, the lift-to-drag ratio (C_l/C_d) was the objective used. However, a constraint was placed on the drag coefficient (C_d) so that it never exceeded the original aerofoils drag value. This also ensured that the optimised aerofoil is one that will have the same or higher lifting capability than the original design. As this is a transonic aerofoil, an important constraint is to ensure that the drag divergence Mach number does not fall below the value for the original design. In addition, its moments must be maintained to within a small margin of the original value. The latter is quantified using the moment coefficient (C_m) about the quarter chord point of the aerofoil. The former is obtained analytically using Korn's method with a technology factor of 0.95[142] . The program for this can be found in Appendix B.12.

4.3 Optimisation

The variables to be optimised were the first three parameter coefficients of six used to define the upper surface of the aerofoil.

To create the initial population, steady calculations were run at the conditions described in Cook et al.[141] (Mach 0.725, Re 6.5 million, AoA = 2.92^o) using HMB to obtain a database of 27 points (full factorial combination of 3 values for each coefficient). An ANN was trained for each

RAE 2822 Aerofoil	C_l	C_d	C_m
original	0.7983	0.01888	-0.2912
parameterised	0.7981	0.01886	-0.2903

Table 4.1: *Table comparing aerodynamic coefficients of the original and parameterised RAE 2822 aerofoil.*

performance parameter using this data with the following ANN parameters: 3 inputs, 2 hidden layers, 15 neurons in each layer and 1 output. The variation in predictions of C_l/C_d with the three coefficients, α_1, α_2 and α_3, are shown in Figure 4.4. Increasing α_1 and α_2, the 1st and 2nd coefficients, decreases the lift-to-drag ratio and the trend is non-linear. However, increasing α_3 causes an increase of C_l/C_d and its effect is dependent on the values of α_1 and α_2, more evidently noticeable in Figure 4.4(b).
A GA was then used to select the best aerofoils. The objective function used here was simply the C_l/C_d. The constraints were a minimum value of C_l, a maximum value of C_m and a minimum value of M_{dd}. Two of the optimum aerofoils obtained along with the original are shown in Table 4.2. The optimum aerofoil was analysed using the HMB solver and these results are also shown in Table 4.2. Note the difference in the ANN and CFD predictions. The ANN predictions are reasonably accurate, but it worth mentioning that small errors in the ANN predictions of the objective function's components can add up to larger errors in the OFV. Hence, it is good practice to analyse the GA's selection using CFD to validate the optimum parameters. Also, it may be better to train an ANN with the OFV itself - but this reduces the flexibility of changing the objective function and constraint handling without re-training the ANN.

Figure 4.5 shows the Mach number contours around the three aerofoils in Table 4.2, and Figure 4.6 shows the original and the best optimised aerofoil. The optimised aerofoil has a maximum camber further aft than the original aerofoil so that a good pressure gradient is maintained for a larger part of the aerofoil, but to counter the change in moments that this causes, the front of the aerofoil, close to the leading edge is more rounded so that a lower pressure is maintained on the top surface at the front as can be seen from from the C_p plot in Figure 4.5.

Figure 4.7 shows the Pareto front output for the RAE 2822 transonic aerofoil optimisation. Here three performance parameters are used to create a 3D Pareto front using drag divergence Mach number, lift and drag coefficients. Again, it can be seen that the optimal selection using the objective function falls along the Pareto front. Using an objective function and constraints confines the optimal aerofoils to a small area on the front. In this case particularly, the Pareto front offers a huge variety of designs that can be used, which still makes it difficult to choose the optimum. The objective function helps to define how much compromise in one performance parameter can be allowed to improve another. The objective function was to optimise the lift-to-drag ratio while constraining M_{dd} and C_m.

RAE 2822 and Optimum								
Note	α_1	α_2	α_3	C_l	C_d	C_m	M_{dd}	C_l/C_d
original	0.029724	0.009149	0.001444	0.7981	0.01886	-0.2903	0.75194	42.317
CFD (b)	0.029724	0.009149	0.003444	0.8084	0.01767	-0.2875	0.75152	45.750
ANN (c)	0.028929	0.009283	0.004450	0.7823	0.01801	-0.2821	0.75490	43.437
CFD (c)	0.028929	0.009283	0.004450	0.8056	0.01762	-0.2868	0.74957	45.721

Table 4.2: *Data for the original RAE 2822 aerofoil and the modified parameterised upper surfaces of the aerofoils (1st 3 coefficients of 6). The most optimum aerofoil is (c) with a higher C_l/C_d, slightly lower drag divergence Mach number and slightly higher absolute C_m. Note the difference in the predictions of the ANN and CFD. Small errors in the components that make up the objective function, can add up to larger errors in the final value.*

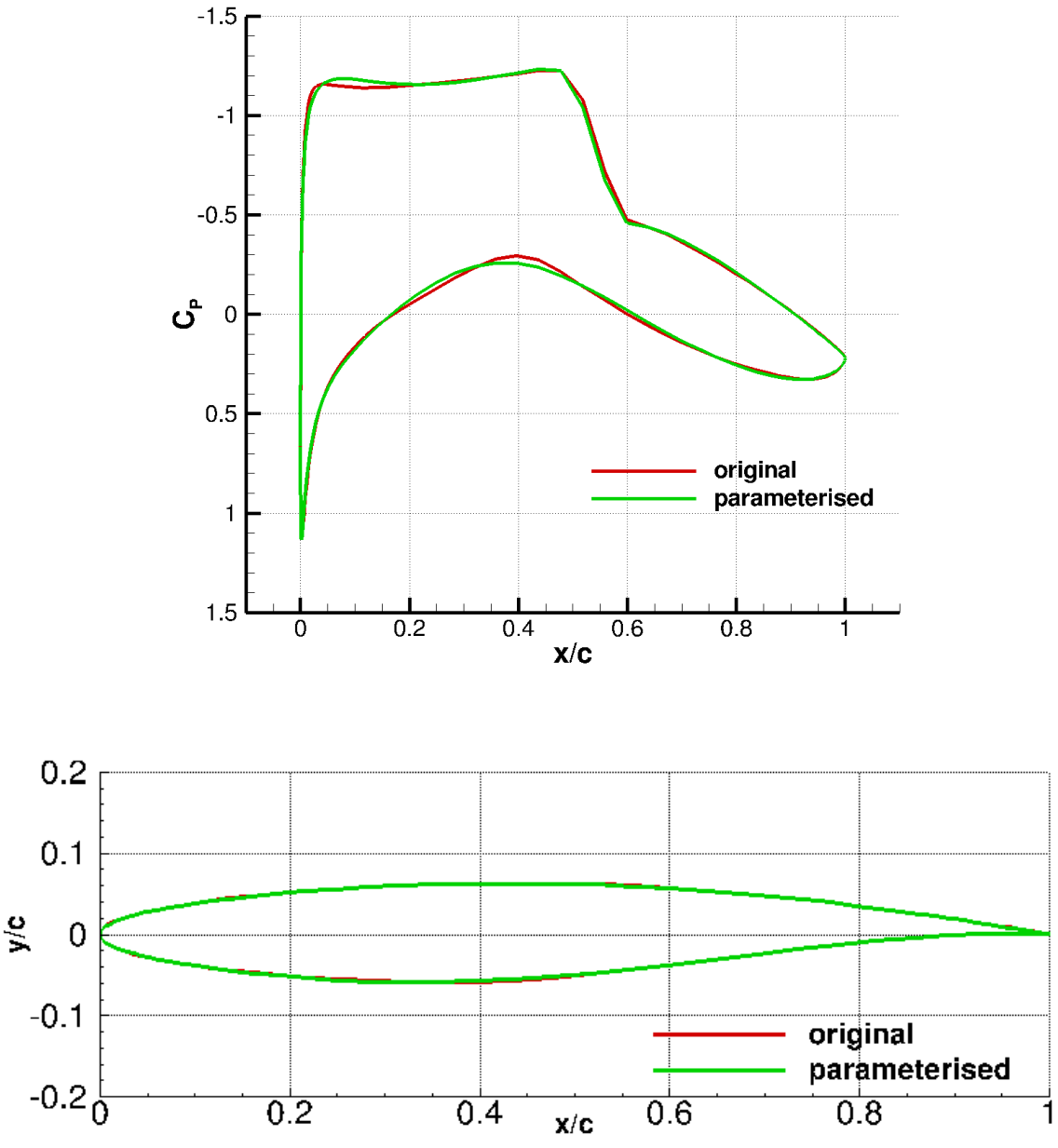
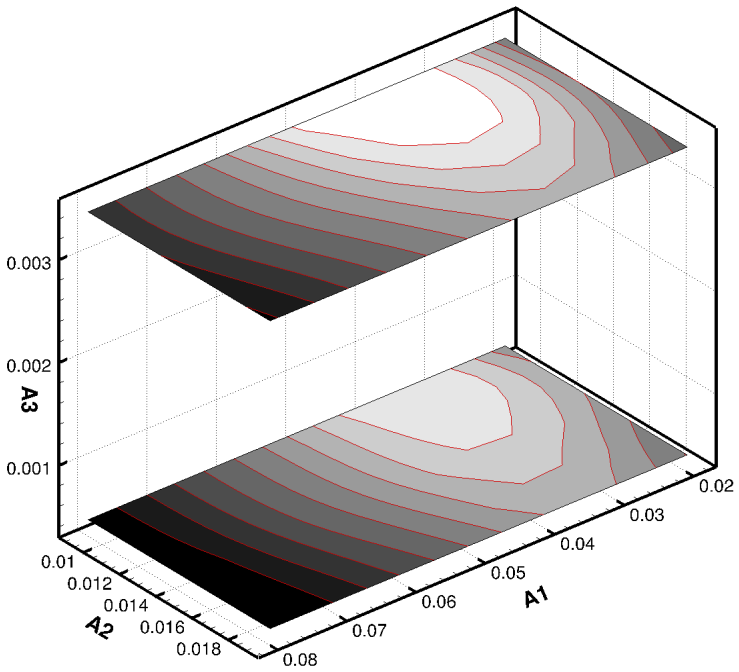
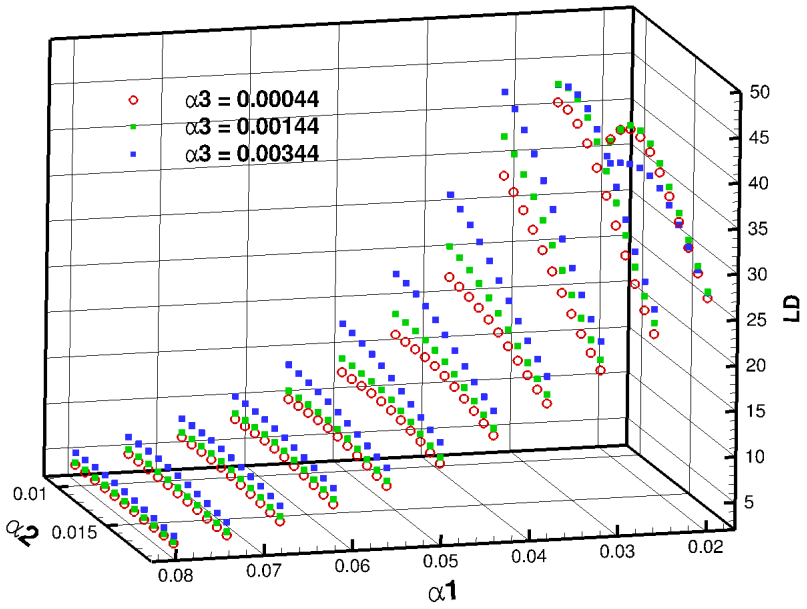


Figure 4.3: *C_p and aerofoil section comparison of the original and parameterised RAE 2822 at Mach = 0.725 and AoA = 2.92 degrees.*



(a)



(b)

Figure 4.4: *C_l/C_d ANN predictions of varying 1^{st} 3 coefficients of parameterised upper surface of RAE 2822. Optimum seems to aim for low α_1 and α_2 and high α_3. No constraints are implemented.*

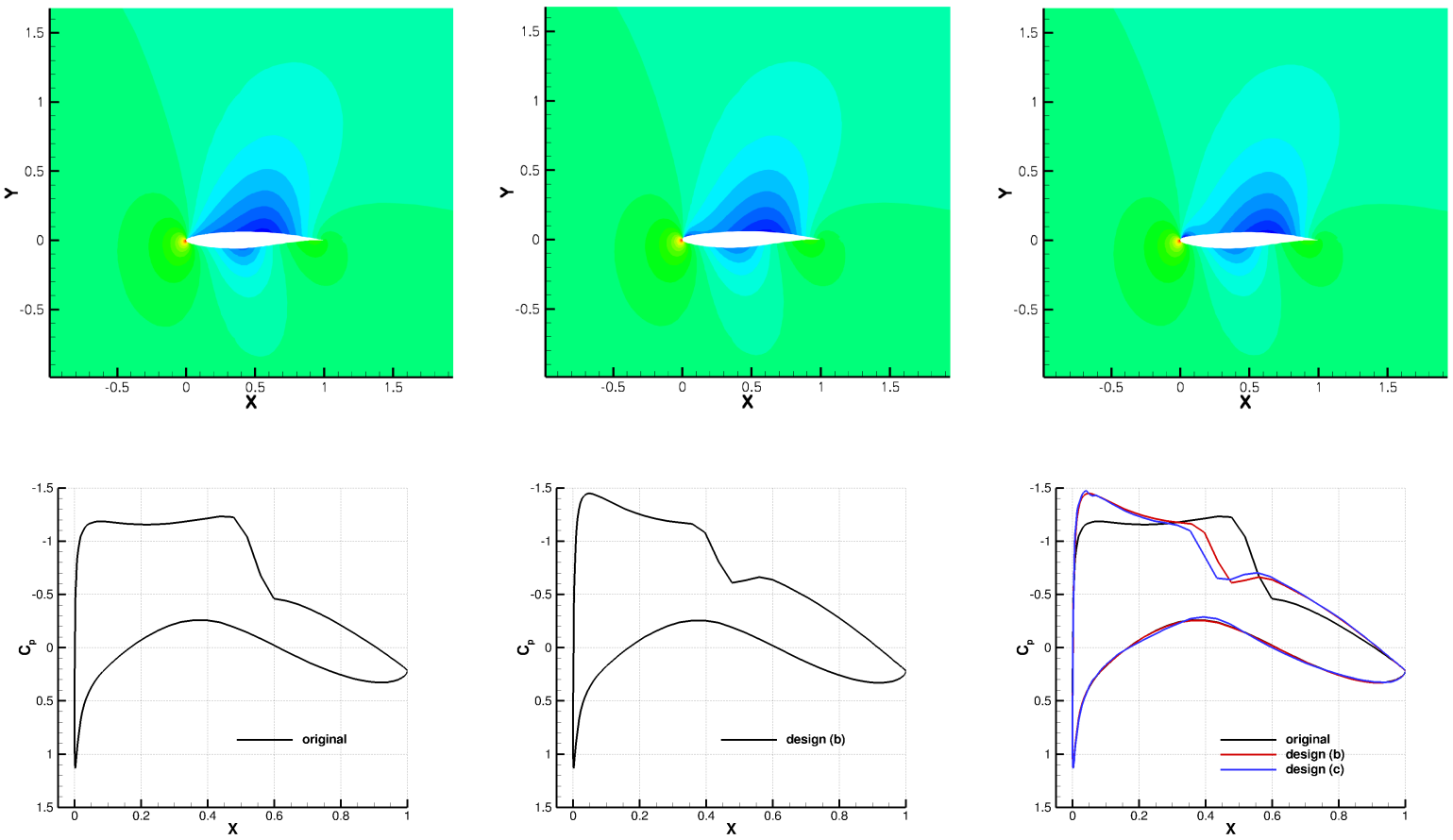


Figure 4.5: *Mach number contours and C_p for (a) the original RAE2822 aerofoil, (b) and (c) the optimised aerofoils in Table 4.2. Free stream Mach number = 0.725, AoA = 2.92^o, Re = 6.5×10^{-6}.*

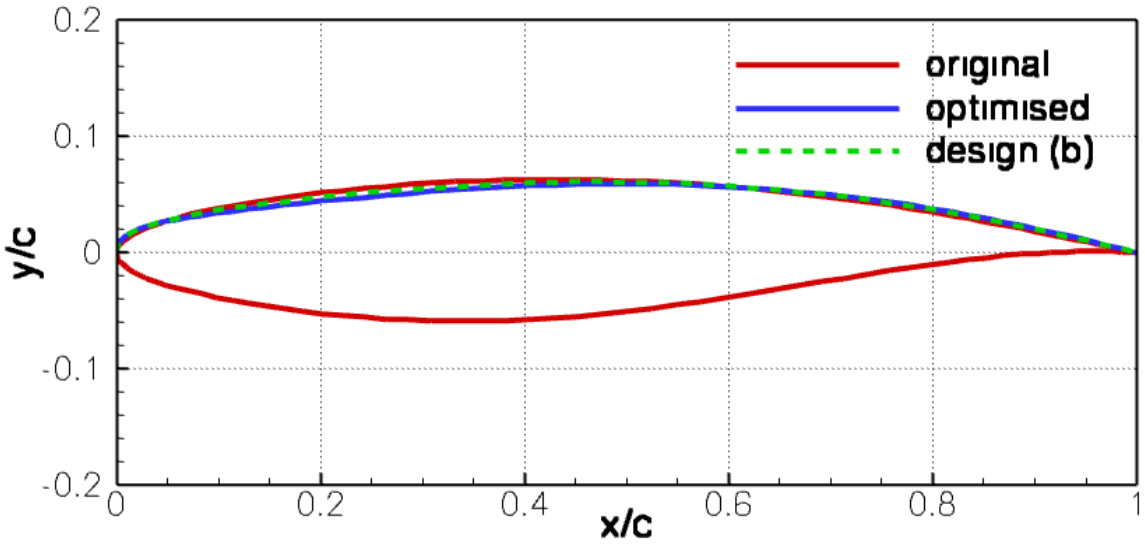


Figure 4.6: *The original RAE 2822 and the optimised aerofoil shape:* α_1 - 7.95×10^{-4}, α_2 + 1.34×10^{-4}, α_3 + 3.006×10^{-3}.

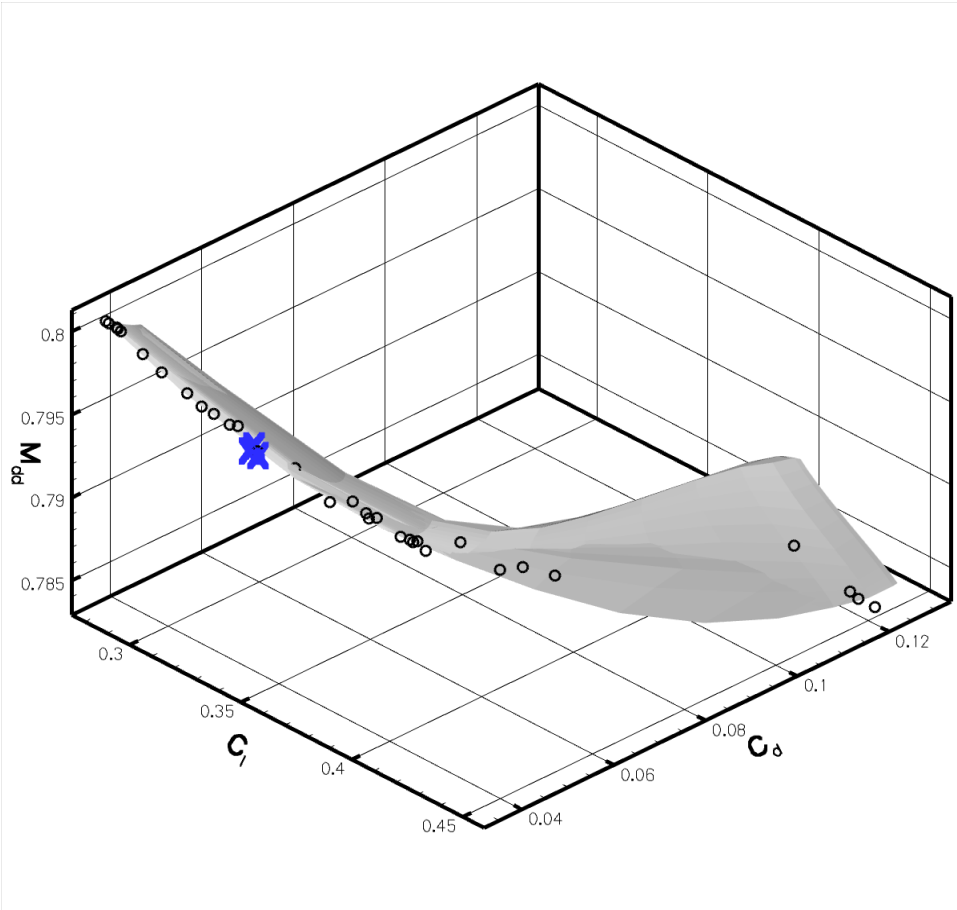
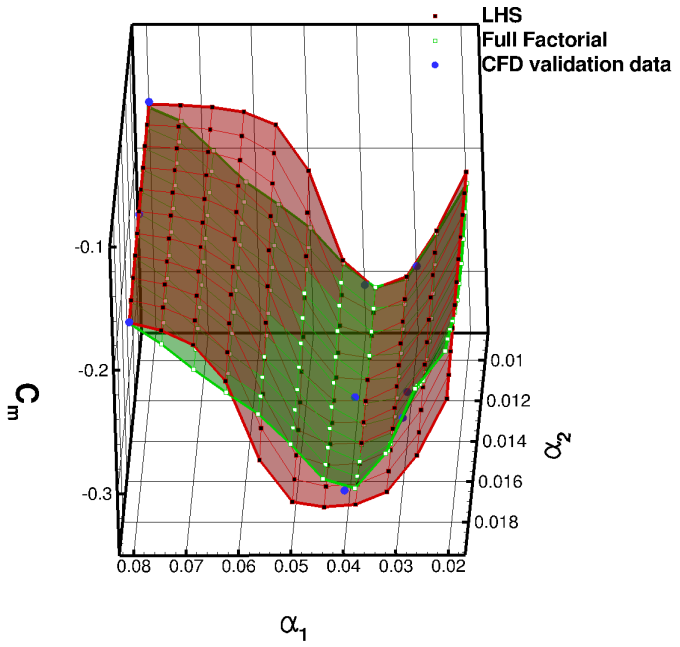


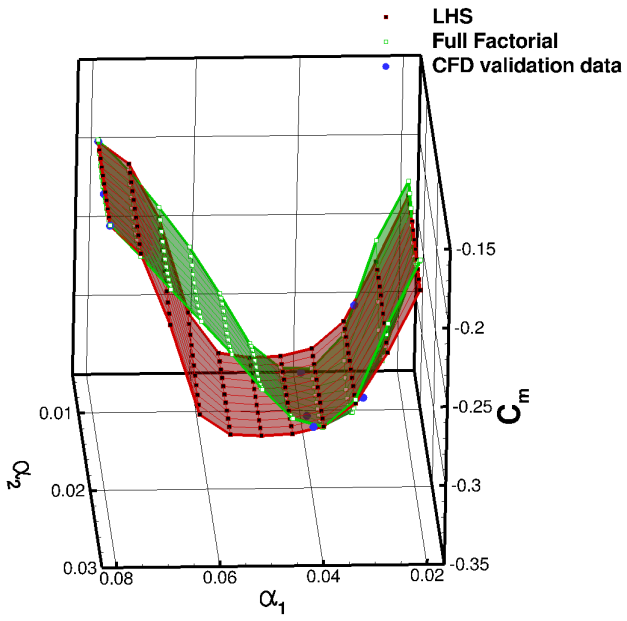
Figure 4.7: *Pareto front for the RAE 2822 aerofoil using drag divergence Mach number,* C_l *and* C_d*. The grey surface is a surface plot of these values. The black dots make up the Pareto front points and the blue spots are the GA's selection using the objective function.*

4.4 Sampling of the Design Space

This case was also used to test out the Latin Hypercube Sampling (LHS) method in comparison to the full factorial method. The validation data were additional points in the design space obtained using the HMB solver, but these results were not included in the training data for both the LHS and full factorial sampling techniques. Using the LHS method (with additional database boundary points) obtains the same trends but not as accurately as using the full factorial method as can be seen in Figure 4.8 for pitching moment, which had the worst prediction accuracy with the LHS. However, the trend was still captured and this was sufficient to obtain similar optimum design parameters.



$\alpha_3 = 0.00044$



$\alpha_3 = 0.00344$

Figure 4.8: *LHS comparison for the RAE 2822 moment coefficient.*

Chapter 5

Rotor Aerofoils Optimisation

In this case, the aerofoil sections for a rotor blade are optimised for forward flight. The simulated conditions are shown in Figure 5.1. Typically rotors tend to have up to three different sections

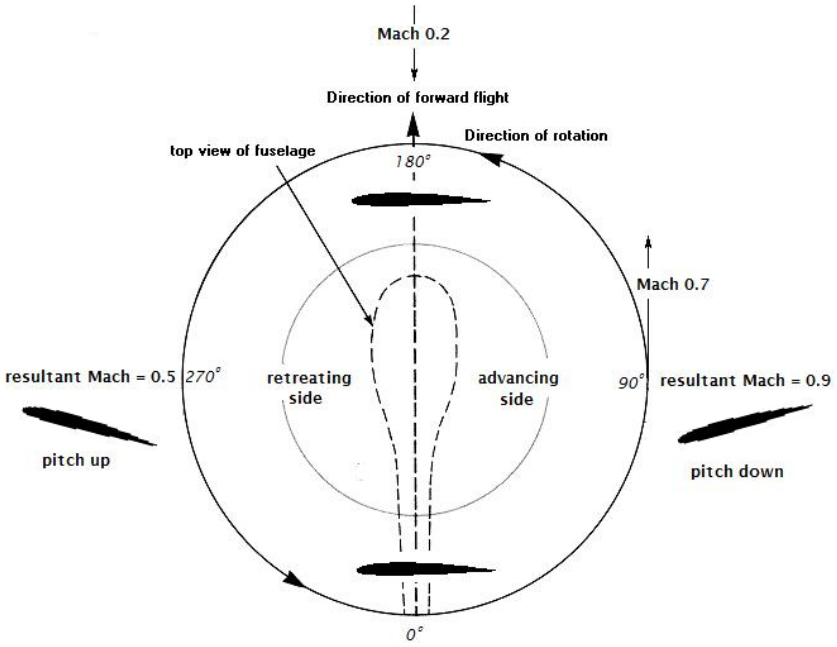


Figure 5.1: *Conditions of forward flight for optimisation.*

on a rotor blade. Usually these sections are selected to improve the efficiency of the rotors or to counter the disadvantages of varying other design aspects of the rotor such as twist, planform, etc. For this case, the assumption is that the rotor is a rigid blade with twist and a cyclic pitch amplitude of 8 degrees. Three stations along the blade are selected to be optimised. The design parameters to be optimised for were thickness and camber and to do this, the NACA 5-digit aerofoils were used as the parameterisation technique. Table 5 shows the parameters to be used in creating the initial population.
The CFD calculations were carried out by simulating the motion of the section as a dM/dt unsteady calculation, the details of which can be found in Section 2.4. Therefore for each section, the Mach number variation changes depending on the station. The details for each station are also included in Table 5. Each CFD calculation was carried out over three cycles and the results from the final revolution were used to obtain the performance of the section.

5.1 Performance Parameters

The objective here is to minimise the adverse effects of compressibility on the advancing side and high angles of attack at low Mach number on the retreating side of the rotor disk. Both

Radii (r/R)	0.5	0.75	0.9
Mach number range	0.15 - 0.55	0.325 - 0.725	0.43 - 0.83
AoA	10±8	7±8	5±8
NACA Thickness (t/c)	9, 12, 15 and 18		
NACA Camber (5-digit)	0, 130, 230, 330, 430		

Table 5.1: *Table showing test cases used for 2D optimisation.*

these aerodynamic phenomena not only cause increase in torque (and hence power requirements) and drag, but they also cause high pitching moments that put high demands on the pitch links. Therefore a good performance parameter to include in the objective function would be the pitching moment over a full cycle as well as the lift and drag coefficients. Figures 5.2 to Figures 5.7 show the variation in these three performance parameters for a full cycle (360^o azimuth) at r/R = 0.5 and 0.9.

Looking at the coefficient of drag first, it can be seen that stall is easily observable at the inboard station in Figure 5.2. For the NACA 0009, on the retreating side (close to $\psi = 270^o$, its average drag coefficient ($C_{d_{avg}}$) is much higher. Also the oscillations caused by the shedding of vortices are large and with lower frequency which is also an indication of stall. With increasing camber and thickness, the average drag drops and the oscillations reduce. On the advancing side, an increase in the drag can be observed with increasing camber due to compressibility effects. Also at high thickness values, there is a higher increase in drag with increased camber on the advancing side than with thinner aerofoils.
The effect of compressibility at high Mach number on the advancing side at the outboard station is also evident from the drag curves in Figure 5.3. The NACA 0009 does not have the steep drag rise and the NACA 0012 also behaves the same to some extent. All the other aerofoils have a steep drag rise on the advancing side. At higher thickness values, camber has little effect on the drag rise. Not much change in drag occurs on the retreating side with change in camber and thickness although the NACA0009 seems to stall. The blip in the curves, especially visible at thicknesses of 15% and 18% are due to the sudden form of a shock on the surface of the aerofoil as shown in Figure 5.8.
Overall, taking the average drag value will distinguish the designs that are vastly poorly performing sections from better ones. However, further performance parameters are necessary to fine-tune the optimisation for smaller changes in the thickness and camber.

Figures 5.4 and 5.5 show the pitching moment variation with azimuth at the inboard and outboard stations. For the moments, the aim is to try and keep it as close to zero as possible to reduce the load required on the pitch links. There are two parameters that can be used to do this viz. the average value, $C_{m_{avg}}$ and the peak-to-peak value, ΔC_m, over a full cycle. ΔC_m is a good indicator of the power required to maintain the blade's pitching moment, but it does not account for all the oscillations in the curve. Together with the average moment, it then gives a more complete picture. However, $C_{m_{avg}}$ also reflects the power required to maintain pitching moment. Therefore, using both parameters will increase its weighting in the objective function. Therefore, the decision was to use the $C_{m_{avg}}$ since it gives a more complete picture than ΔC_m. This is more visible in the 9% thickness plot at r/R = 0.5 in Figure 5.4, where the symmetrical aerofoil has one of the lowest ΔC_m but it has the most oscillations due to stall which is better reflected in its $C_{m_{avg}}$. At the outboard station, the $C_{m_{avg}}$ reflects the same objective as ΔC_m. This confirms the use of the $C_{m_{avg}}$ over ΔC_m.

The C_l curves in Figures 5.6 and 5.7 can also be used to show the effects of stall and compressibility, but this has been adequately dealt with using the drag and moment curves. However, the average C_l is used as a constraint.

It is accepted that a more fair comparison would be to compare the performance at the same lifting capability of the design, for example by considering the zero-lift angles as opposed to the geometric angle of attack. However, for an unsteady case such as this, it is difficult to match the lift load throughout the cycle or to even obtain the same average lift value, because of the many aerodynamic phenomena that occur based on shape. For the cases shown here, a maximum of 50% difference in average C_l was contained within the data for the very extreme cases where the un-cambered thin sections and the highly cambered thick sections were compared. This shows the complexity of the optimisation task though it could be addressed with further computations.

Once the performance parameters were established, the metamodel was employed. For the meta-model, ANNs were trained with the scaled data i.e. as a ratio to the reference section (NACA 0012) at the same conditions. The ANN parameters were 2 input, 2 hidden layers with 15 neurons each and 1 output for each performance parameter. The error convergence was set to 0.01 and convergence occurred at the end of the run. The trained ANN was then used to predict points every 10^{th} of the range. The results agreed well with the data obtained from the solver - most points overlapping exactly. The trends in all the performance parameters (i.e. the average and peak-to-peak values for all the coefficients as well as the gradient of the moment and drag curves) as predicted using ANNs can be seen in Figure 5.9. It can be seen that drag has more variation with thickness than with camber and moment seems to have similar sensitivity to both thickness and camber.

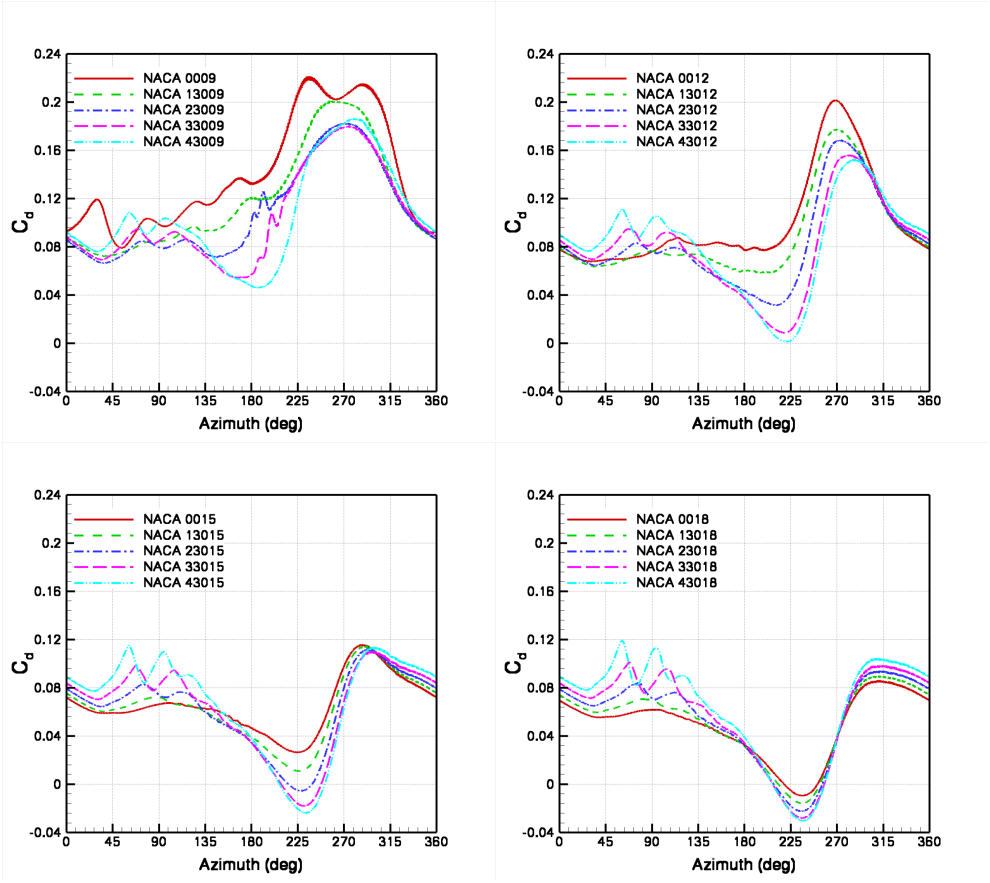


Figure 5.2: *C_d for varying NACA aerofoil camber - thickness 9, 12, 15 and 18% chord at the inboard station, $r/R = 0.5$. $M_{loc} = 0.35$, $M_\infty = 0.2$ in forward flight.*

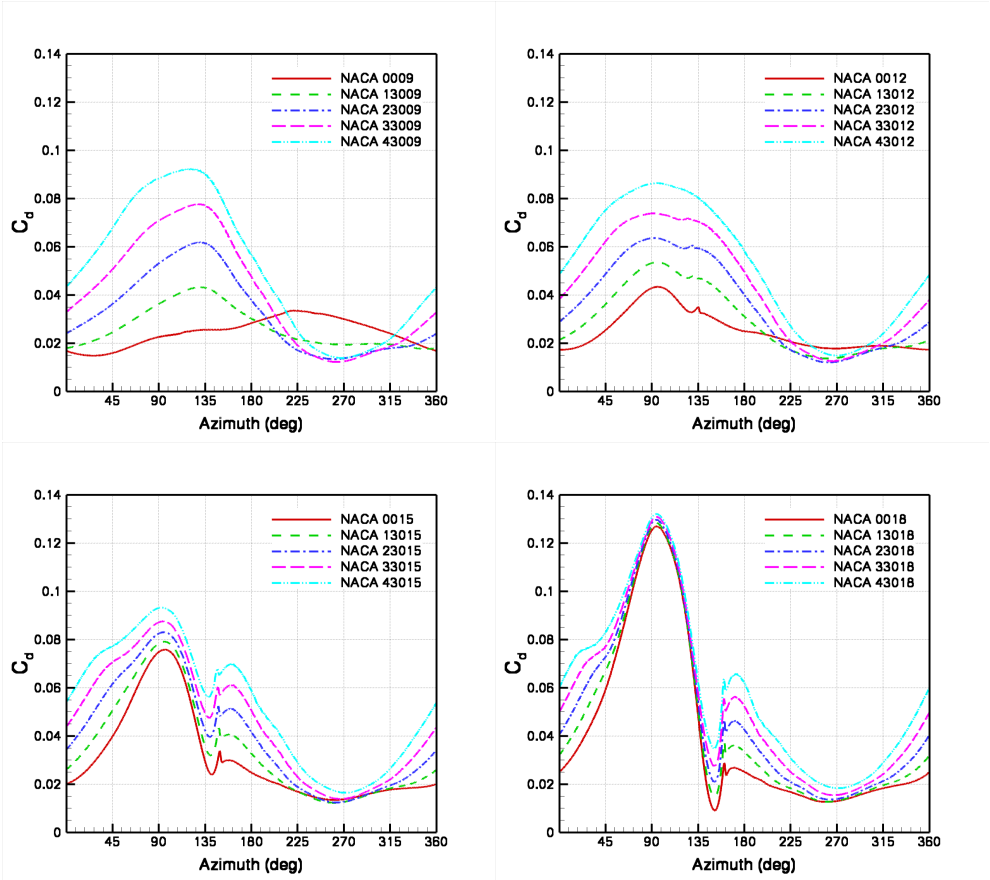


Figure 5.3: C_d *for varying NACA aerofoil camber - thickness 9, 12, 15 and 18% chord at the outboard station,* $R = 90\%$. $M_{loc} = 0.63$, $M_\infty = 0.2$ *in forward flight.*

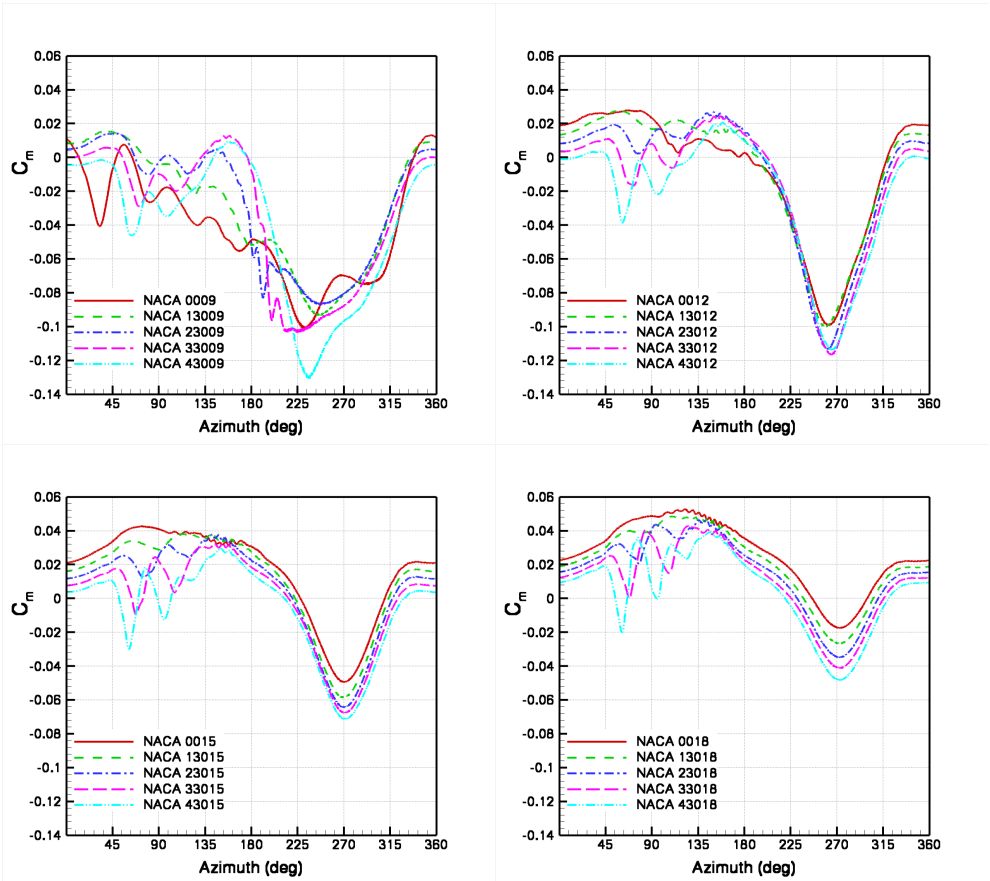


Figure 5.4: C_m *for varying NACA aerofoil camber - thickness 9, 12, 15 and 18% chord at the inboard station,* $R = 50\%$. $M_{loc} = 0.35$, $M_\infty = 0.2$ *in forward flight.*

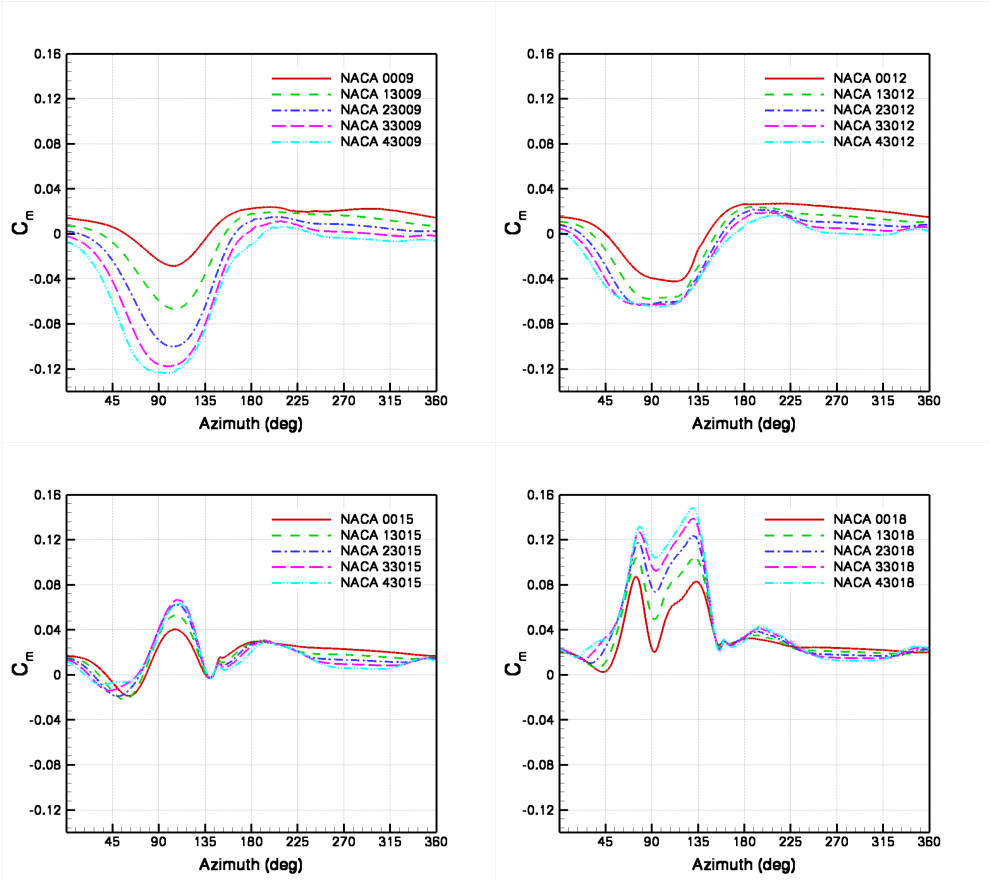


Figure 5.5: C_m *for varying NACA aerofoil camber - thickness 9, 12, 15 and 18% chord at the outboard station,* $R = 90\%$. $M_{loc} = 0.63$, $M_\infty = 0.2$ *in forward flight.*

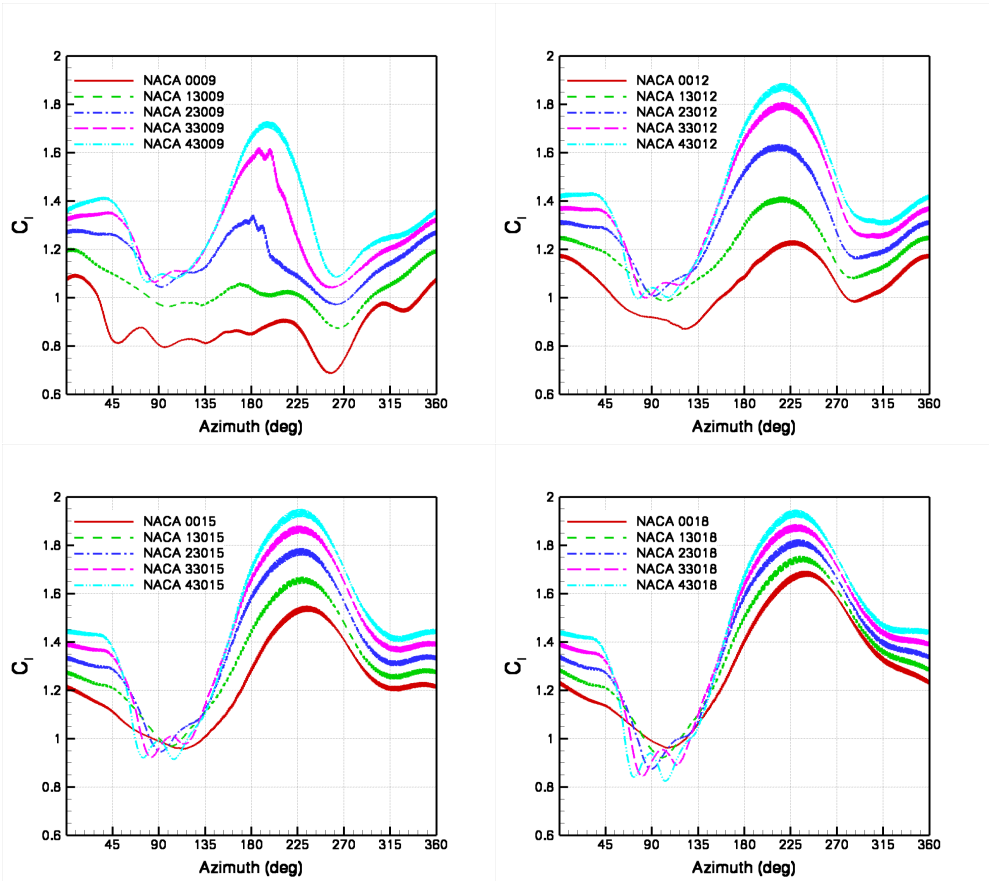


Figure 5.6: C_l *for varying NACA aerofoil camber - thickness 9, 12, 15 and 18% chord at the inboard station,* $R = 50\%$. $M_{loc} = 0.35$, $M_\infty = 0.2$ *in forward flight.*

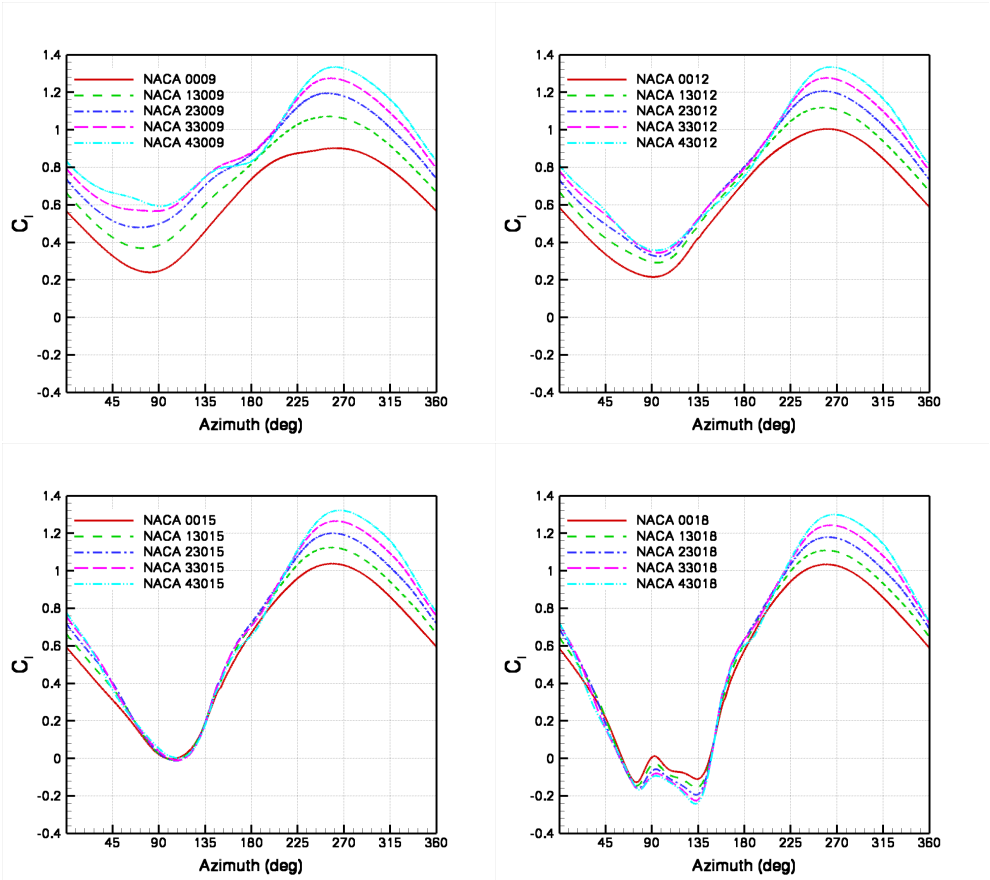


Figure 5.7: *C_l for varying NACA aerofoil camber - thickness 9, 12, 15 and 18% chord at the outboard station, $R = 90\%$. $M_{loc} = 0.63$, $M_\infty = 0.2$ in forward flight.*

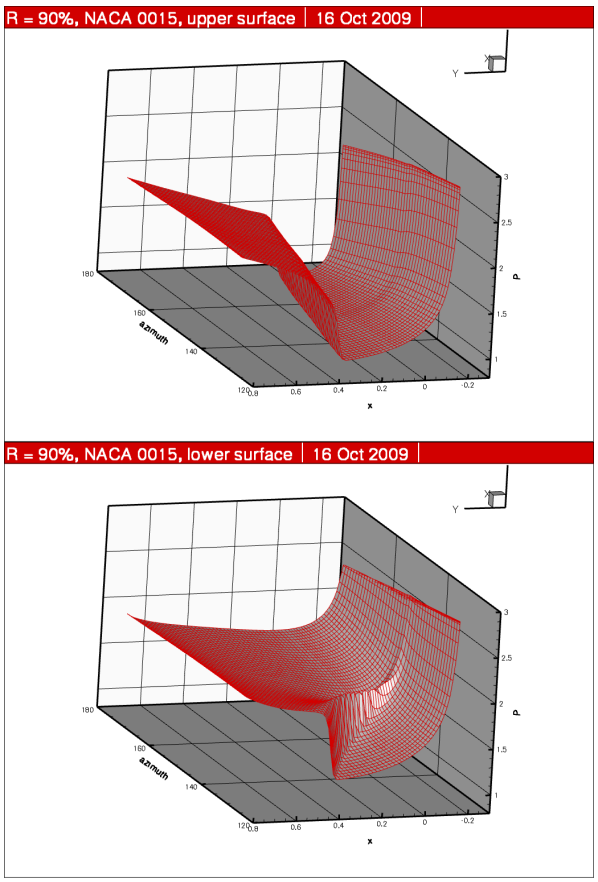


Figure 5.8: *Pressure contours on the lower and upper surface of NACA 0015 for increasing azimuth angle. $R = 90\%$, $M_{r/R}$ in hover would be 0.63 and $M_\infty = 0.2$ in forward flight.*

5.2 Objective Function

To create the objective function, the reference design to which all designs will be scaled is the NACA 0012 section so that the weightings are truly proportional to the requirements and are not affected by the actual values of the performance parameters. To capture the objectives, $\overline{C}_{d_{avg}}$ and $\overline{C}_{m_{avg}}$ over a complete revolution of the rotor was used to create the objective function while keeping $\overline{C}_{l_{avg}} \geq 1$. The drag component was the primary component in providing the selection pressure and the moment component, secondary. The guiding weights therefore, were determined by averaging the ratio of average $\overline{C}_{d_{avg}}$ to $\overline{C}_{m_{avg}}$:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{\overline{C}_{d_{avg}}}{\overline{C}_{m_{avg}}} = 1.2129 \tag{5.1}$$

where $n = 20$ is the number of sample points and the limiting weight for the loads were

$$w(\overline{C}_{d_{avg}}) = \frac{1}{1+1.2129} = 0.452 \tag{5.2}$$

$$w(\overline{C}_{m_{avg}}) = \frac{1.2129}{1+1.2129} = 0.548 \tag{5.3}$$

As there was large variance in this ratio and since $\overline{C}_{d_{avg}}$ is the primary component, the objective function used was

$$OFV = -0.5\overline{C}_{d_{avg}} - 0.3\overline{C}_{m_{avg}} + 0.8 \tag{5.4}$$

The ANN predictions of these two components at 50%R and 90%R and the effect camber and thickness have on them can be seen in Figure 5.9. Both drag and moment show that thick aerofoils are more suitable inboards and thinner ones, outboard, as expected. Moment also refines the optimisation of camber showing that more symmetrical aerofoils are suitable outboard and vice versa.

5.3 Optimisation

The GA employs these trained ANNs to obtain the optima. Figure 5.10 shows that the optima also fall on the Pareto front at all three stations. The objective function forces the global solution to a small region on the Pareto front. The combined Pareto and objective function method used the objective function to create the selection pressure for the parents for crossover, but does not determine the fitness. Table 5.2 shows the results of the GA for the inboard and outboard stations. Thinner more symmetrical sections are good for outboard stations and thicker, more cambered sections perform better more inboards, which is in agreement with most real-life helicopter rotors.

r/R = 0.5			**r/R = 0.9**		
Camber	Thickness	OFV	Camber	Thickness	OFV
33	15	0.44	9	9	0.30
43	16	0.42	7	11	0.14
32	14	0.36	5	10	0.12

Table 5.2: *Aerofoils selected by the GA for the inboard and outboard stations. The * values are scaled with the reference section, the NACA 0012.*

The kriging metamodel was also employed to compare its performance with the ANN predictions. A comparison between the optimum surface created by both is shown in Figure 5.11. It can be seen that kriging favours a smoother surface, although this did not significantly affect the outcome of the optimisation (a 1% difference in camber).

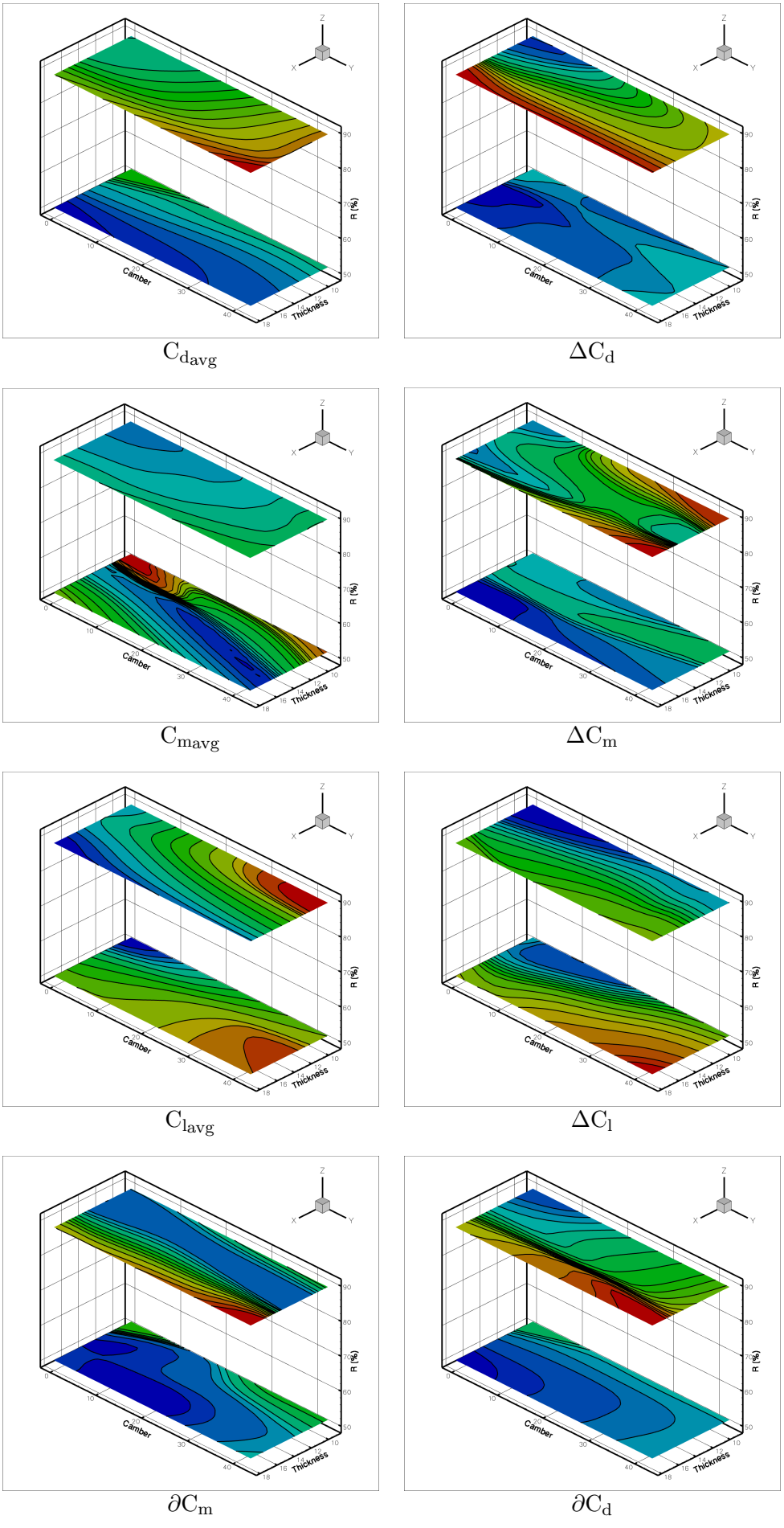
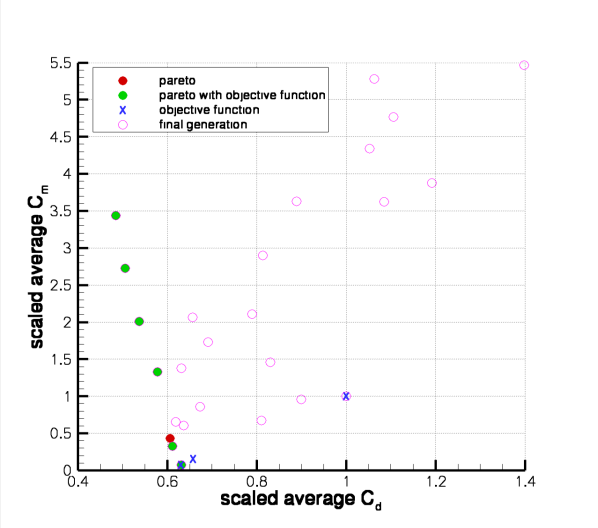
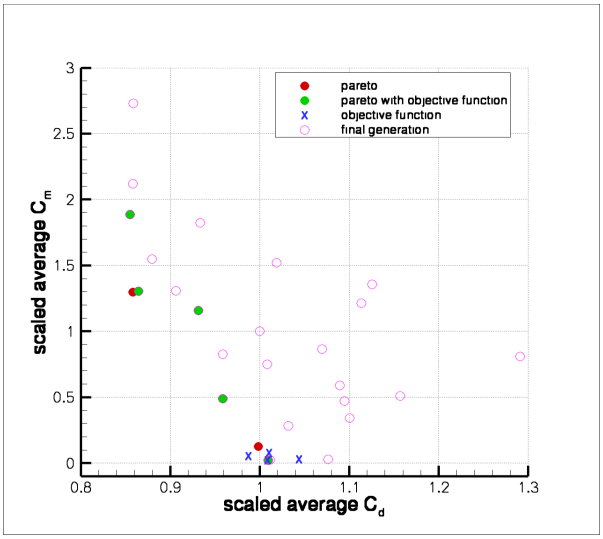


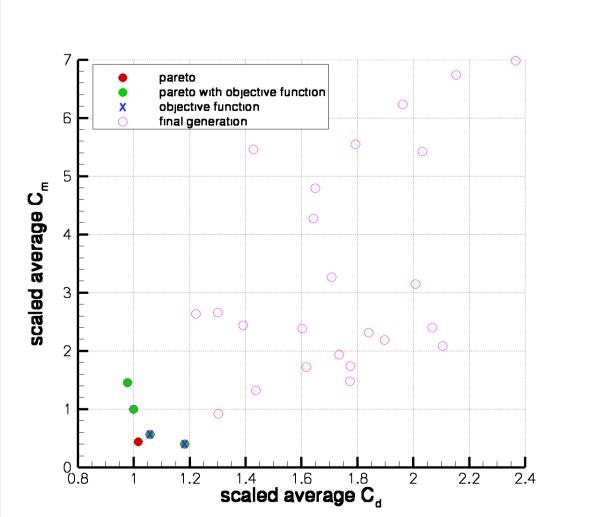
Figure 5.9: *Variation of loads for r/R = 50% and 90%; Red represents the upper extreme and blue, the lower extreme.*



Inboard station

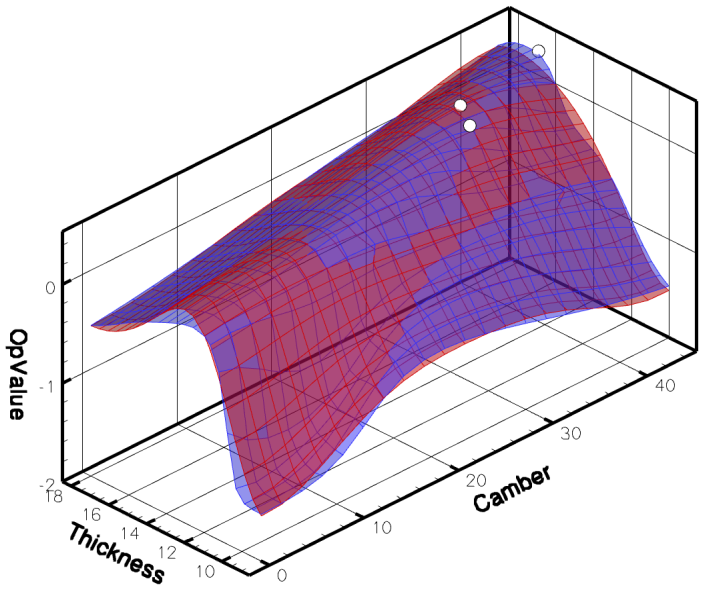


Midboard station

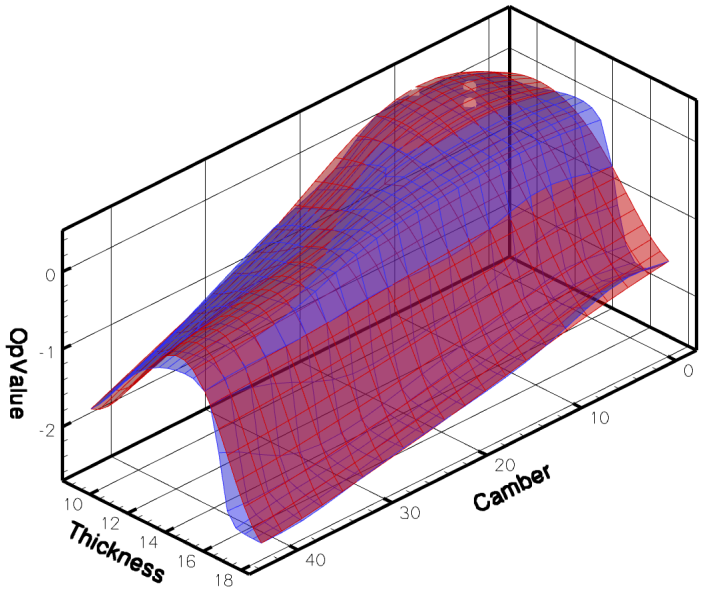


Outboard station

Figure 5.10: *Comparison for the dM/dt cases inboard, midboard and outboard stations of the use of pareto optimisation, objective functions and both i.e. objective function used for selection pressure on designs.*



Inboard station, 50% r/R



Outboard station, 90% r/R

Figure 5.11: *Comparison of the optimum surface created by the ANN (blue) and Kriging (red) metamodels. The GA selection is shown as white dots.*

Chapter 6

Wing Planform Optimisation

6.1 Design Space Generation

The optimisation of a wing planform was carried out to test the optimisation process with an increased number of parameters. The steady flow calculation was carried out at Mach 0.4 with a Reynolds number of approximately 2 million. The wing to be optimised is a multi-segment wing shown in Figure 6.1 that uses the NACA 23012 aerofoil throughout. It has three segments. The root is fixed and the tip chord is constrained in chord length but allowed to translate in the X-axis. The two mid-chords, M1 (closer to the root, z/b = 0.272) and M2 (closer to the tip, z/b = 0.927) and their corresponding positions (ΔM1 and ΔM2) as well as the tip chord position (see Figure 6.1), are the parameters to be optimised for optimal wing loading along the span, which is an elliptic distribution of loading. To create the required grids for this optimisation, an ICEM replay script was written to automate grid generation for the CFD simulation. The script can be found in Appendix B as Listing B.10.1. The objective was to have elliptic loading of the wing i.e. an elliptic lift distribution over the span of the wing without compromising the lift to drag ratio of the wing.

To modify a grid for the three segment planform, the four files that contain the sections at the root, M1, M2 and tip chord can be changed by directly modifying their length and position as required and the ICEM-hexa script can be *replayed* in ICEM to generate the geometry. A blocking file exists where each vertex has been previously constrained to a specifically named point. Therefore, once the new geometry is created, the blocking can be opened and using the '*snap to point*' button of ICEMCFD, the association is performed automatically and the grid can be set up.

In this way a number of grids were created and analysed using HMB. An example is shown in Figure 6.2. Sampling was done by taking a uniform distribution of points within the parameter boundaries (including the boundary values). As the optimisation proceeded, new points were added, especially wherever the GA suggested the optimum was, resulting in a higher resolution of the area where the optimum was expected to be in the design space. However, as this refining of the database occurred, new points were added elsewhere where the data seemed sparse so that the ANN was capable of making predictions of similar accuracy in other places of the design space. This adaptive sampling was carried out manually after the original design space was obtained, until the error between the target value and predicted value fell within a tolerance level.

There are optimisation algorithms that can do this sampling automatically, for example, as described in the paper by Haftka[143]. The Efficient Global Optimisation Algorithm (EGO) starts by creating a kriging metamodel and it keeps adding points until the prediction error falls inside a tolerance level. It builds the kriging model using all the data except a small subset of it. It then evaluates the error in prediction by calculating the rms error value between the original and predicted values for this small subset (similar to the work done for this case). It also estimates the probability of error along the predicted curve using the kriging variance as a measure of

uncertainty and then adds points where the uncertainty and error are high.

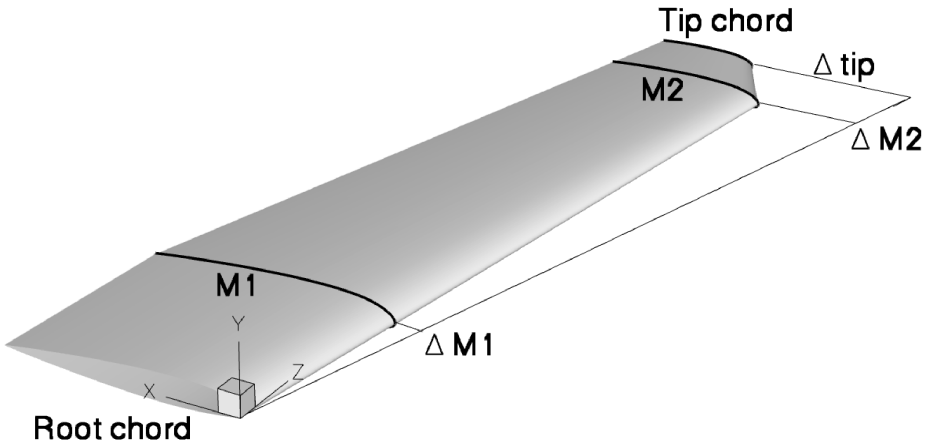


Figure 6.1: *Definition of the parameters for optimisation of the wing.*

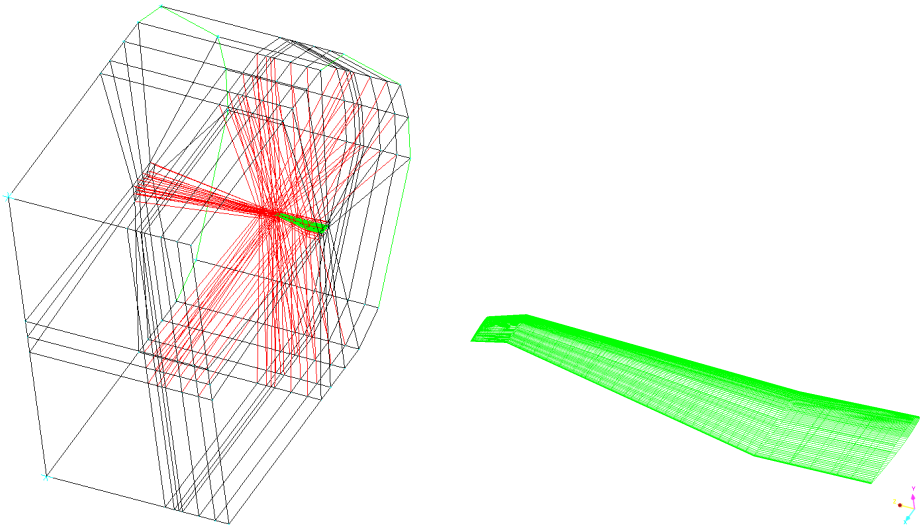


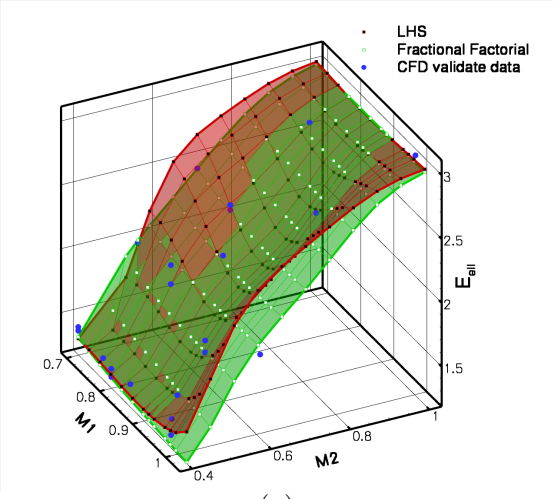
Figure 6.2: *Grid for the wing.*

6.2 Planform Optimisation

The optimisation was for elliptical loading which was measured as 'E_{ell}' by comparing the C_L distribution along the span to a parabolic fit of the curve and evaluating the difference between them (See Appendix B.11). Therefore the objective function was given as:

$$OFV = E_{ell} = \sum_{x=0}^{x=b/2} \sqrt{(C_l - C_{l_{ell}})^2} \tag{6.1}$$

where x is the spanwise location and b is the span of the wing. The greater the difference, the worse the design. In addition, C_L/C_D and C_L were constrained to be greater than the original wing's design values, which places a constraint on the drag as well. The C_L, and hence C_D, was constrained to be within a small margin over the original wing's values to allow the GA to explore the design space and increase the diversity of the population.



(a)

Figure 6.3: *LHS comparison for the near laminar flow wing elliptic loading.*

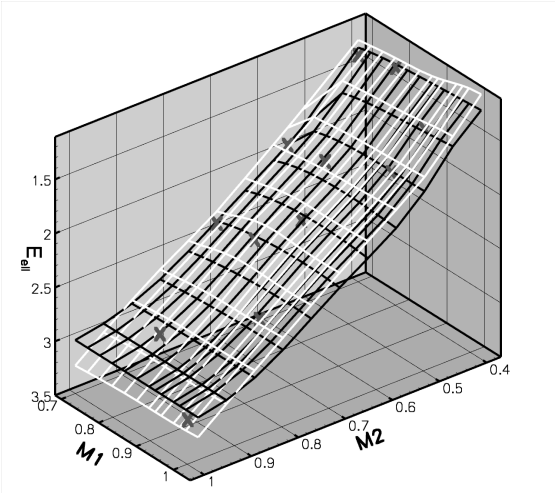
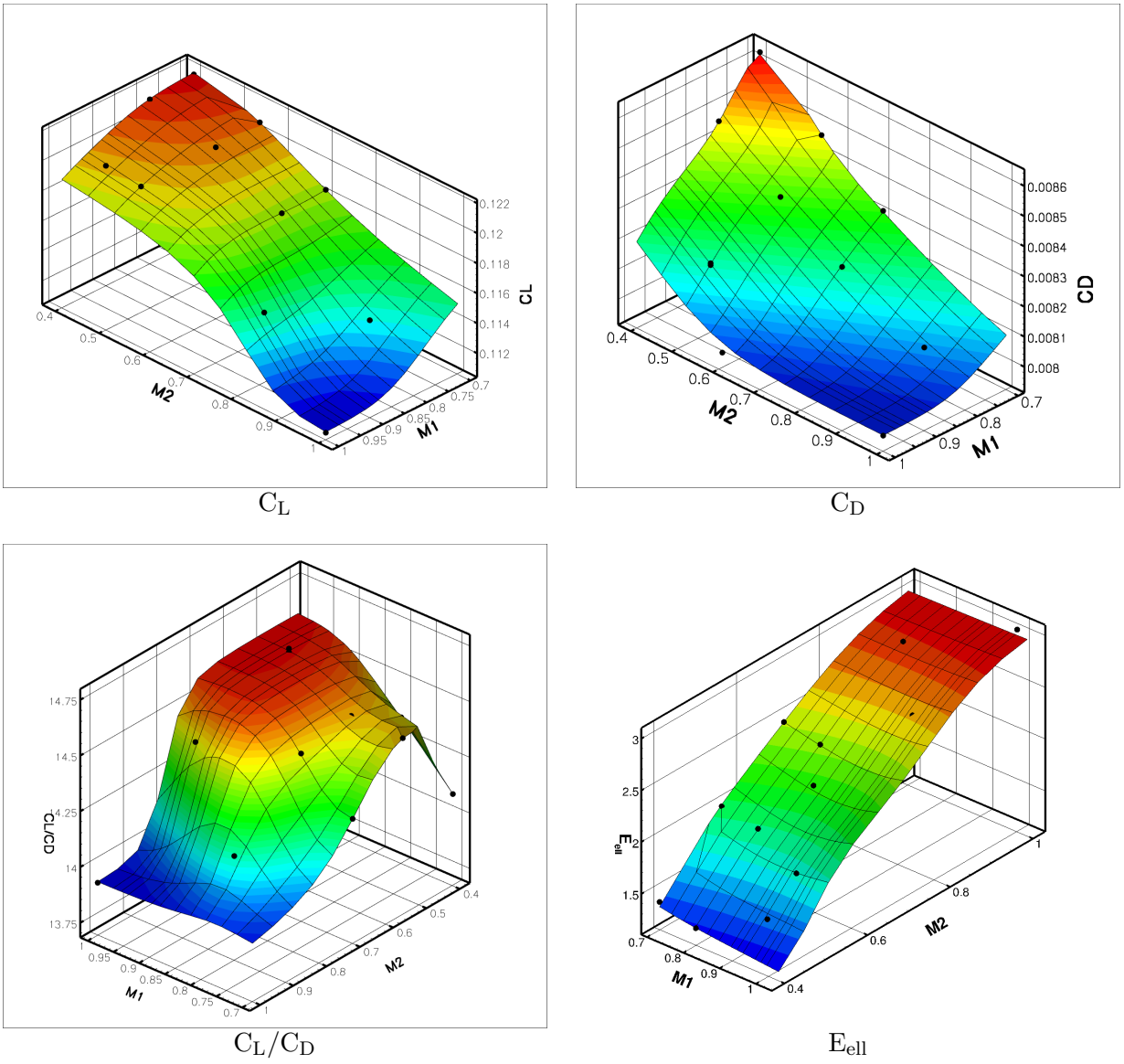


Figure 6.4: *ANN (black) and Kriging (white) comparison for the wing optimisation performance parameter, E_{ell}.*

Therefore four ANNs, for C_L, C_D, C_L/C_D and E_{ell}, were trained for each of the parameters normalised by the corresponding parameter of the original wing. As there are five variables and typically three values for each variable are used for training the ANNs, the number of design points required using a full-factorial method would be $3^5 = 243$. However, for this case, only 40 design points were used to build the database as it was sufficient for accurate predictions and this was validated with additional CFD points not included in the training data. An adaptive fractional factorial sampling method was used where the optimisation method itself played a part in the sampling to improve accuracy by populating more towards where the global optimum was likely to be found. A comparison with the LHS technique is shown in Figure 6.3. The LHS does not perform as well as there are fewer points used to train the ANN. Therefore, the fractional factorial method was used.

Kriging was also used to compare the two metamodels. The regression function was a one-degree polynomial and the correlation function was Gaussian. Figure 6.4 shows again that the ANN and kriging method both give similar predictions although the ANN was slightly more accurate. The difference in the optimum design by both metamodels was 2.4% in the OFV of the same design.

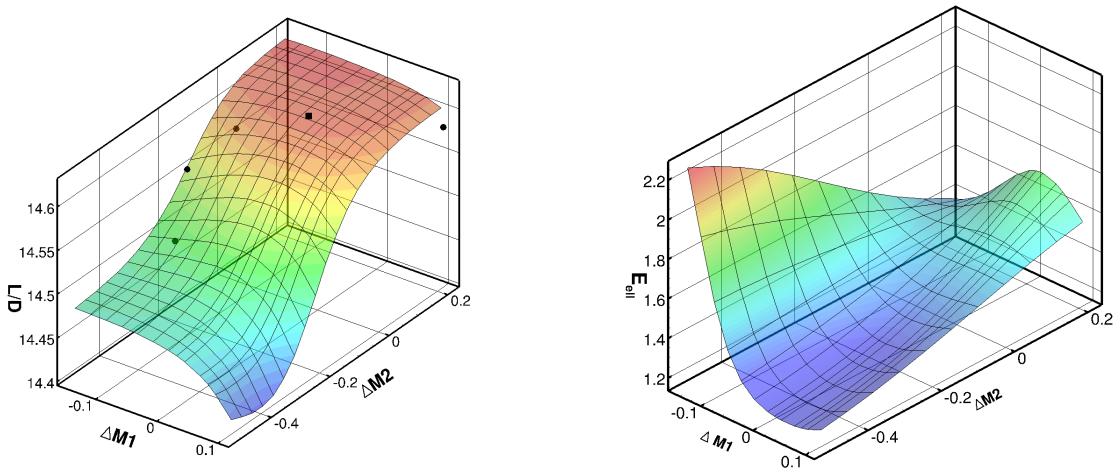
Figure 6.5 visualises the trends using the predictions of the ANN for M1 and M2 along with the training data for these parameters. It shows that a small inboard and outboard chord increases the lift but also the drag. However, when they are analysed together as C_L/C_D, a high chord for M1 and a low chord for M2 gives a good compromise which also favours better lift distribution. Figure 6.6 shows the variation of C_L/C_D and E_{ell} with the position of M1 and M2.



C_L C_D

C_L/C_D E_{ell}

Figure 6.5: *ANN prediction using CFD training data (black dots) for M1 and M2 values, shown at 0 $\Delta M1$ and $\Delta M2$.*



C_L/C_D E_{ell}

Figure 6.6: *ANN predictions of performance parameters as a function of the position of chord variables.*

The GA was run for 19 generations with 1000 iterations per generation. In this case, the fitness value was minimised as opposed to the other cases, where they are maximised. The results of the GA's optimisation are shown in Figure 6.8. The average fitness per generation in Figure 6.7 shows the convergence and stability of the GA for this case. This is defined as:

$$OFV^j_{avg} = \frac{1}{N}\sum_{i=0}^{i=N} OFV^j \tag{6.2}$$

where j is the generation number, i is the individual in a population of N individuals. Table 6.1 shows the CFD data and the ANN prediction for one of the optimum wings in comparison to the baseline wing and a rectangular wing.

The optimisation can be seen to select a more tapered wing with high values of M1 and low

M1	ΔM1	M2	ΔM2	ΔT	C_L	C_D	C_L/C_D	E_{ell}(%)	Note
1	-0.140	0.6086	-0.476	0	0.1162	0.00803	14.4585	37.64%	Original
1	0	1	0	0	0.1109	0.00797	13.9198	58.89%	rectangular
1.04178	-0.0134	0.4725	-0.0023	-0.096	0.1181	0.00808	14.6154	33.15%	ANN (a)
1.04178	-0.0134	0.4725	-0.0023	-0.096	0.1182	0.00809	14.6079	24.92%	CFD (a)

Table 6.1: *Table showing the wing designs analysed with CFD. E_{ell} is the percentage difference between a parabolic distribution and the C_L distribution. The GA's selection of optimum is also included (a). Design (a) is a new point created and predicted by the ANN. The CFD analysis of it is also shown for comparison.*

values of M2 and a moderately swept forward wing. The C_L/C_D of design (a) in Table 6.1 is higher than the original and its E_{ell} is lower. This wing has more taper and maintains some of the original wing's sweep as shown in Figure 6.9 along with C_L distributions. It seems to be that the taper accounts more for the elliptic loading and the position of the sections accounts more for the L/D (from the scales), although both are affected by all parameters.

Figure 6.10 compares the velocity magnitude of the optimised and original blade at a point behind the blade. It shows that for the optimised wing, the velocity is reduced behind the wing suggesting a lower vortex speed and hence better lift distribution. The improvement compared to the original wing as well as a rectangular wing is also visualised in Figure 6.11 using streamlines of the secondary flow behind the wing.

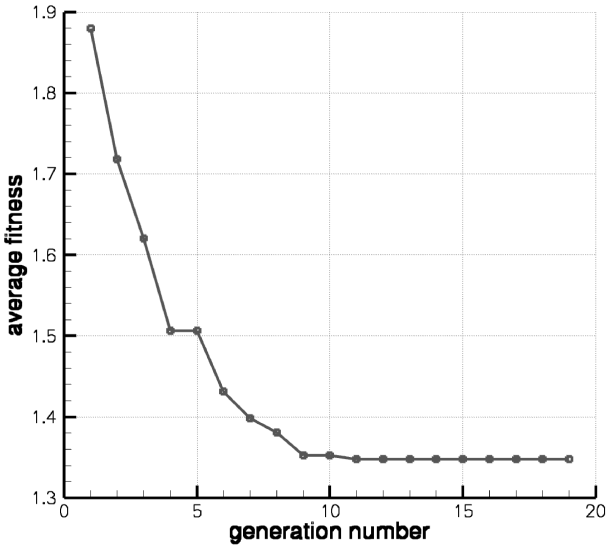
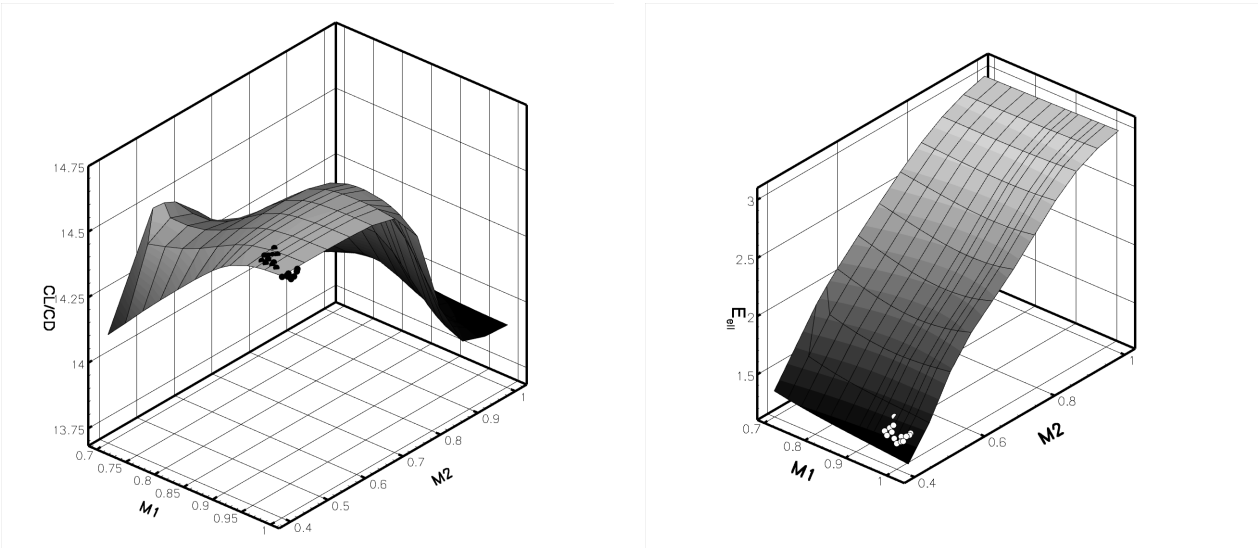


Figure 6.7: *The average fitness per generation. Note that, here the OFV is minimised.*

Figure 6.12 visualises the magnitude of velocity. Note the more spread out distribution



L/D E_{ell}

Figure 6.8: *GA selection of ANN prediction of L/D and elliptic loading.*

for the optimised wing. The vortex behind the rectangular wing is smaller. Since C_L is more or less the same and the angle of attack is the same, circulation must be the same, therefore a larger vortex means a slower speed vortex and vice versa and hence the optimised wing has a better lift distribution (take into account that the vortex is further back from the wing tip at the same slice position for the tapered wing). Also Figure 6.13 shows how far the vortex extends after the wing (taking into consideration the sweep for the GA optimum), indicating the strength of the vortex.

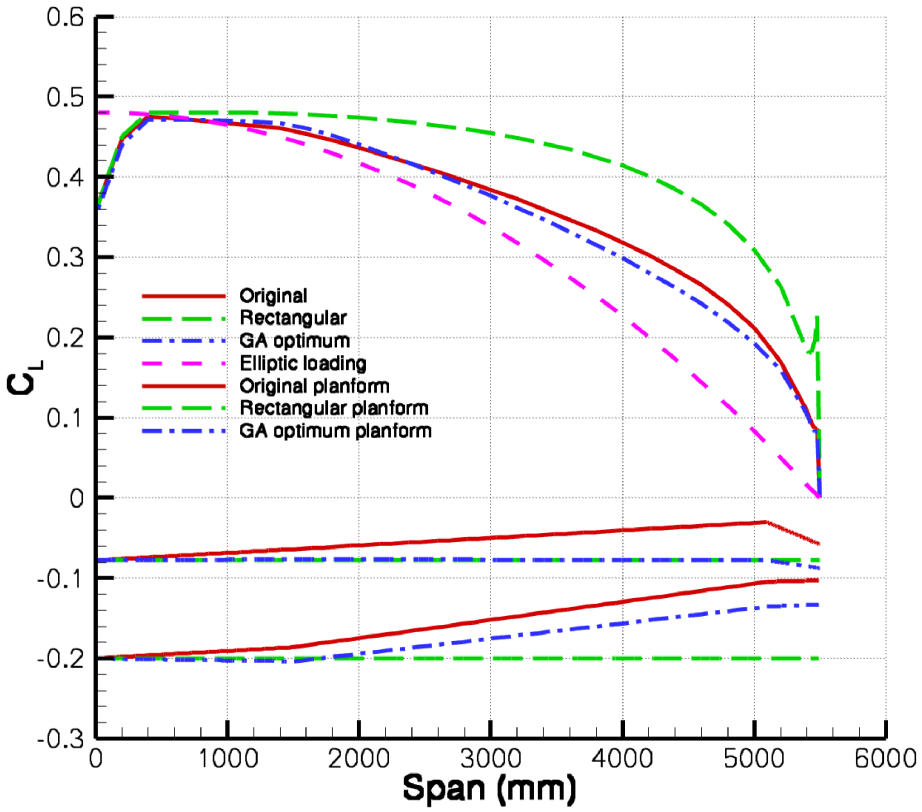


Figure 6.9: *Planform comparison of the designs in Table 6.1.*

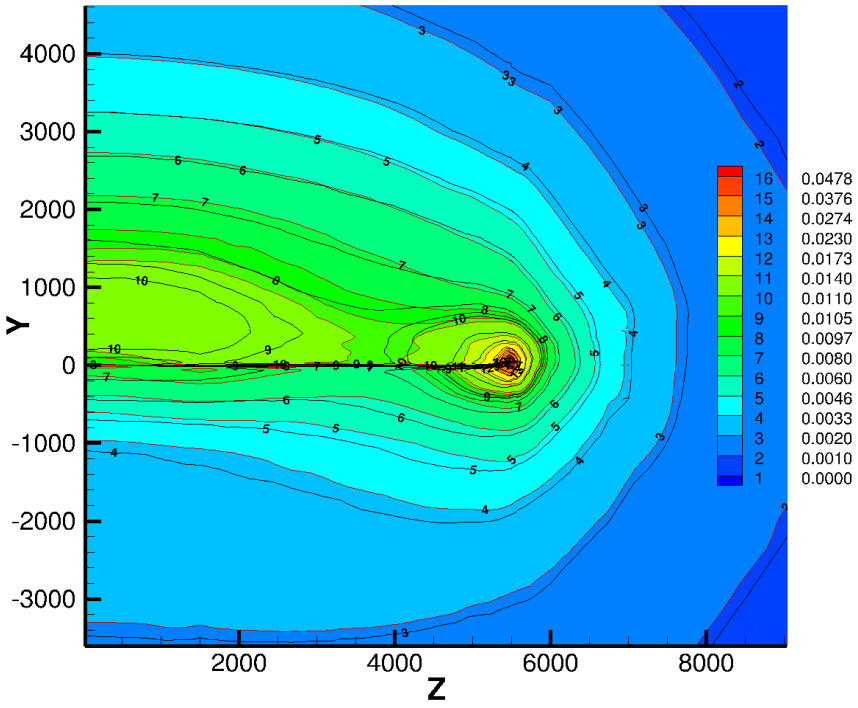
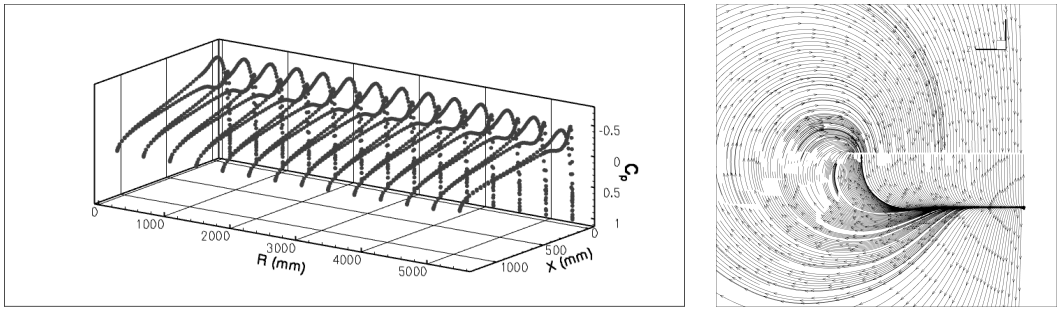
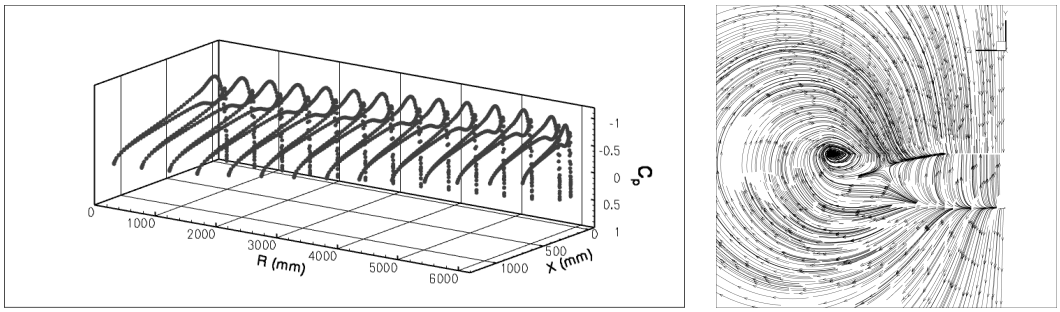


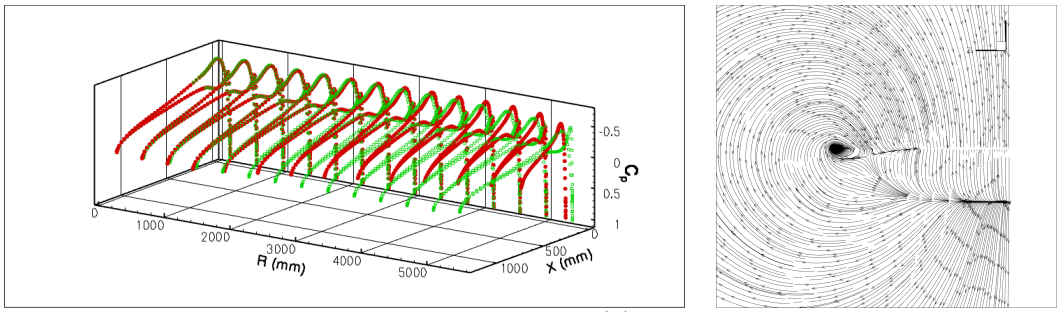
Figure 6.10: *Comparison of velocity magnitude contours for the optimised blade (red) and the original blade (black) at 2.3 chord lengths behind the trailing edge.*



Rectangular

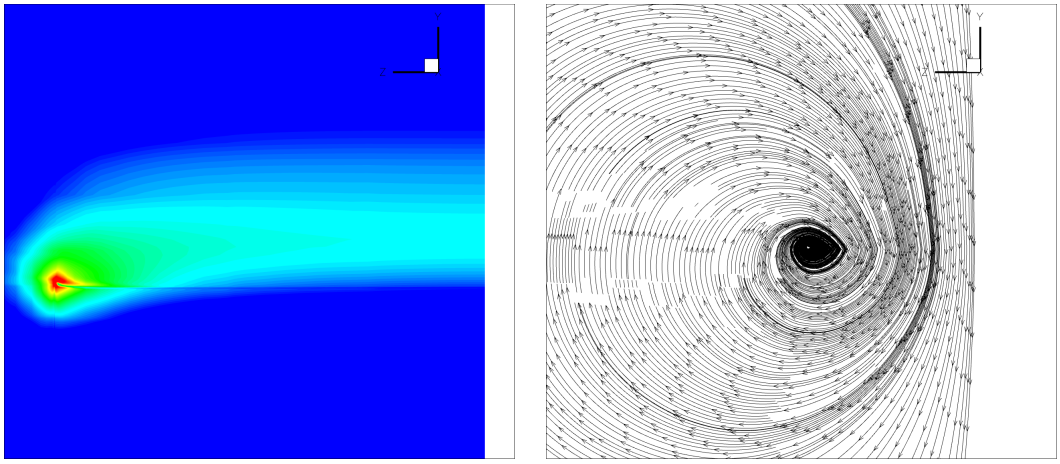


Original wing

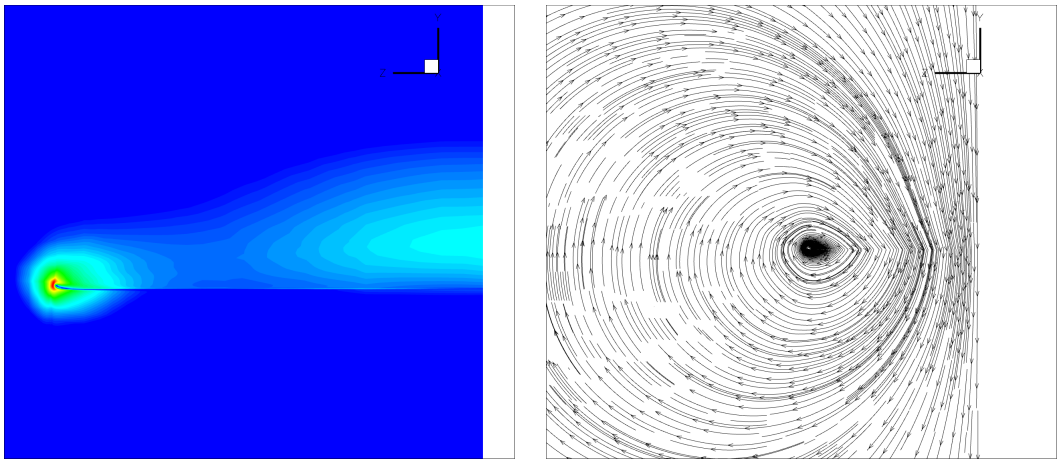


GA optimum new point - (a) in Table 6.1

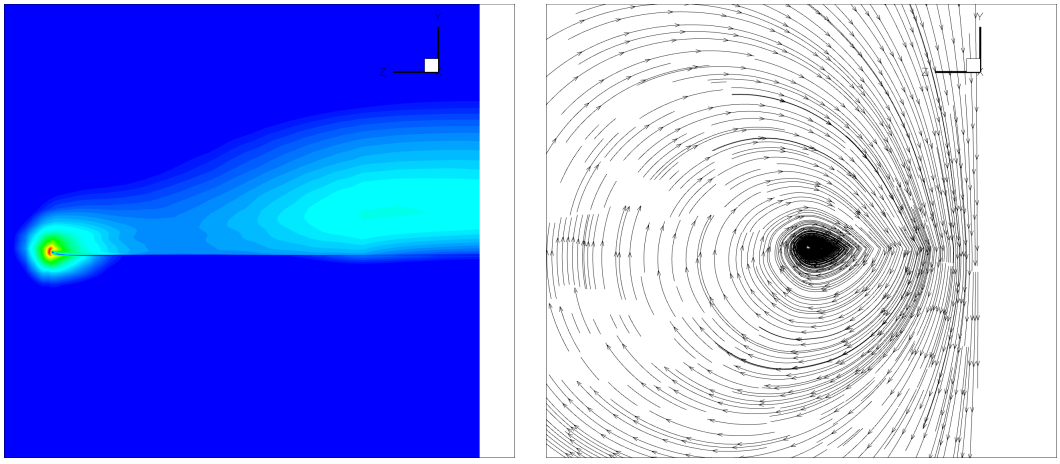
Figure 6.11: *C_p distribution of the designs in Table 6.1 and secondary flow visualisation a chord length behind the trailing edge. The green points are for the rectangular blade and are shown as a reference.*



Rectangular

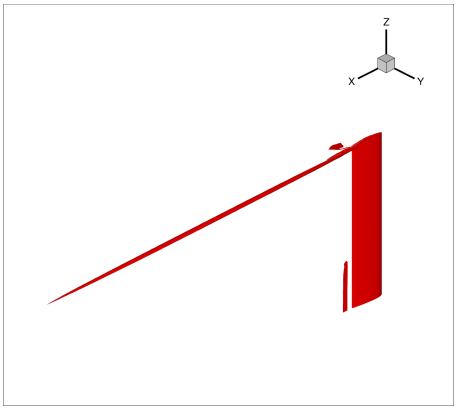


Original

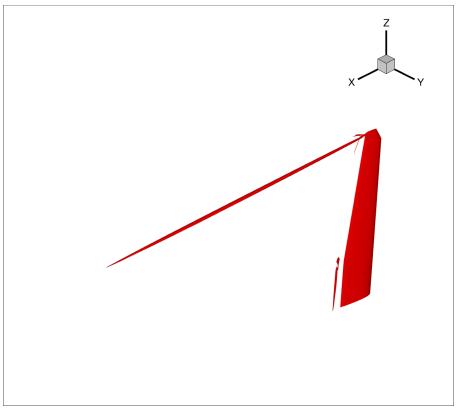


GA Optimum new point

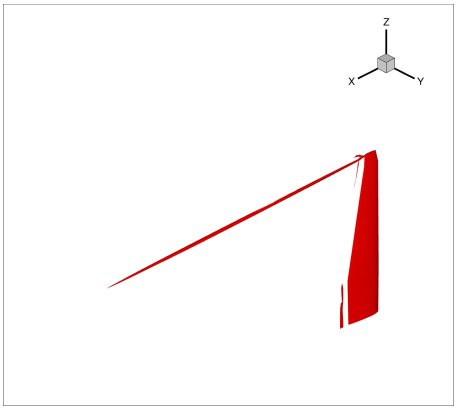
Figure 6.12: *C_L distribution visualisation using velocity magnitude (left), and streamline visualisation of secondary flow (right) at 2.5 chord lengths behind the trailing edge for the wings in Table 6.1.*



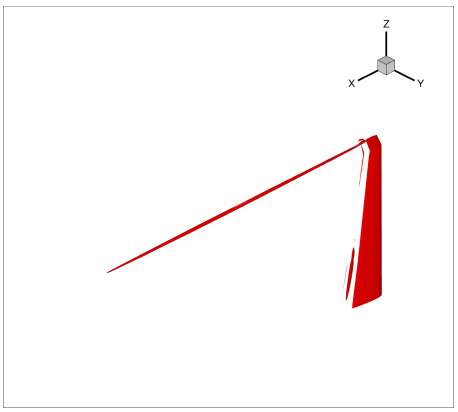
Rectangular



Original



GA Optimum new point



GA Optimum existing point

Figure 6.13: *Q criterion iso-surface* $= 1 \times 10^{-6}$ *showing extent of vortex tip.*

Chapter 7

Hovering Rotor Optimisation

A simple rotor with no coning or anhedral, aspect ratio of 16, linear twist, and a single section (NACA 0012) throughout the span (similar to the one used by Caradonna[12]) was used to analyse the change in performance that occurs due to the increase of twist at varying collective. The tip Mach number was 0.6 with a Reynold's number of about 1 million. The CFD solution was expected to be periodic, therefore the grid used was a quarter segment of the flow field with periodic boundary conditions as described in Section 2.3. The grid was a relatively coarse grid of approximately 2 million cells.

7.1 Performance Parameters

The performance parameters used here are the non-dimensional values of Figure of Merit (FM), thrust coefficient (C_T) and torque coefficient (C_Q). For hover, the Figure of Merit (FM) is a good measure of rotor efficiency and is defined as:

$$FM = \frac{\text{Induced Power}}{\text{Actual Power}} \tag{7.1}$$

Figure 7.1 shows the FM, C_T and C_Q for the hovering rotor. These plots were obtained by running a number of twisted blades at increasing collective angles (represented by the points on the figures). ANNs were trained to predict these performance parameters based on the collective and twist inputs to show the trends in the figures.

The general trend of the FM curve is that there is an increase in FM with increased collective until a certain point where the FM falls due to stall occuring resulting in a drop in the increase in thrust and an increase in the increase in torque which decreases the FM. The aim is to obtain a high FM and to keep it that way over a range of collective. A highly twisted blade is able to maintain a high FM over a bigger range of collective as shown in Figure 7.1 but at the cost of reducing that maximum value. Reducing the twist can obtain a higher FM but it is unlikely that the rotor will be operating at those conditions for a large enough proportion of the time to discard obtaining high FM for other conditions. Too low a twist results in stall outboards which also has the largest dynamic head and hence there is benefit in avoiding stall in this region.

This suggests that the gradient of the FM with respect to C_T would be a good component to capture the objective in addition to maximising the FM. Figure 7.2 shows the FM vs. C_T curve and the C_T vs. C_Q curve which is the hovering rotor equivalent of an aerofoil drag polar curve. With higher thrust, the torque increases greatly especially for less twisted blades. This may be because of the stronger vortices shed from the blade due to compressibility effects and relatively high angles of attack at the tip. With twist, the angle of attack at the tip is brought down and hence a lower torque is obtained. Figure 7.3 shows the pressure contours around the root and tip sections of the rotor with a twist of 0, 8 and 12° and a collective of 12°.

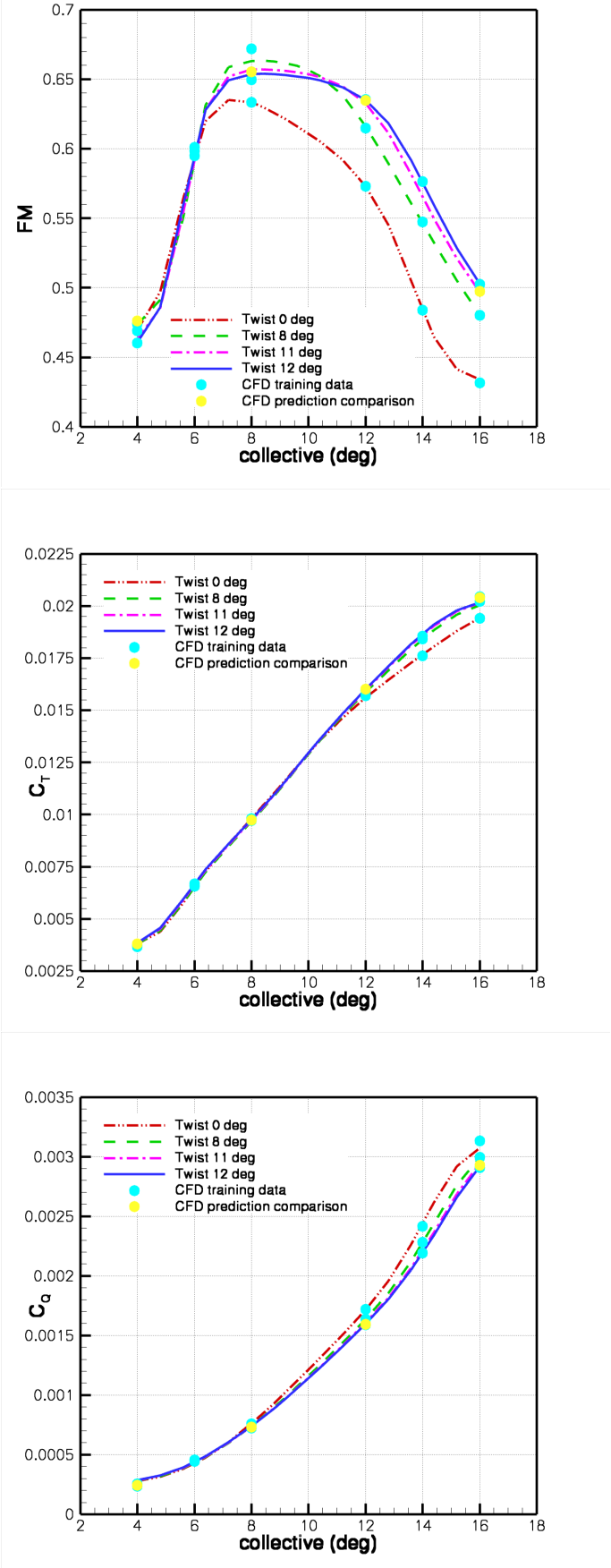
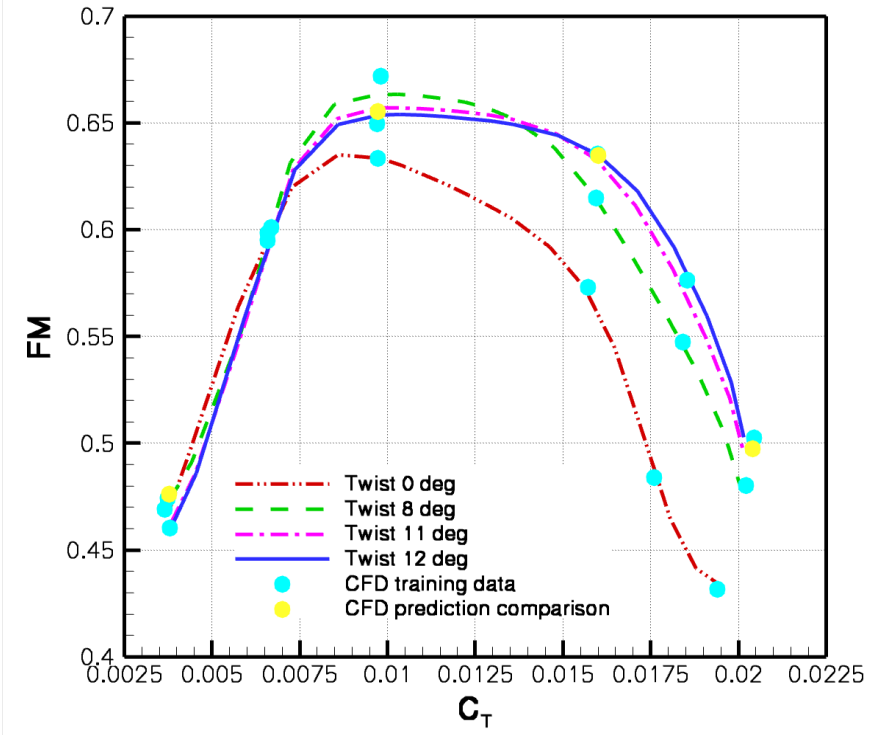
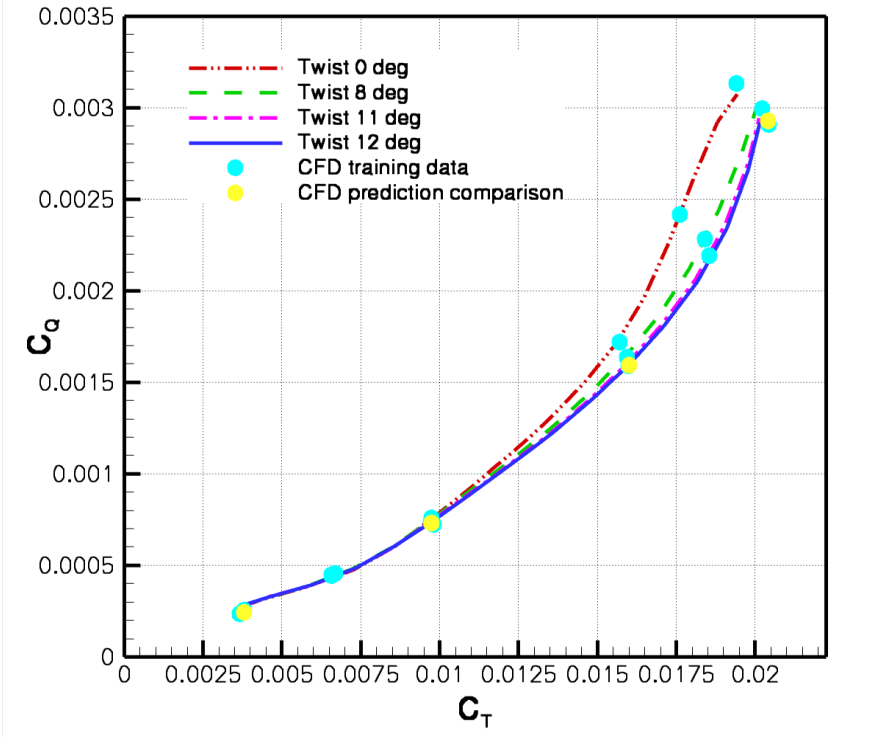


Figure 7.1: *Rotor in hover, Aspect Ratio = 16, Coning = 0, Anhedral = 0, Twist = 0, 8 and 12°: FM, C_T and C_Q vs collective angle. $M_{tip} = 0.6$.*



(a)



(b)

Figure 7.2: *ANN predictions of (a) FM vs C_T and (b) C_T vs. C_Q for 5 values of collective used to train the ANNs.*

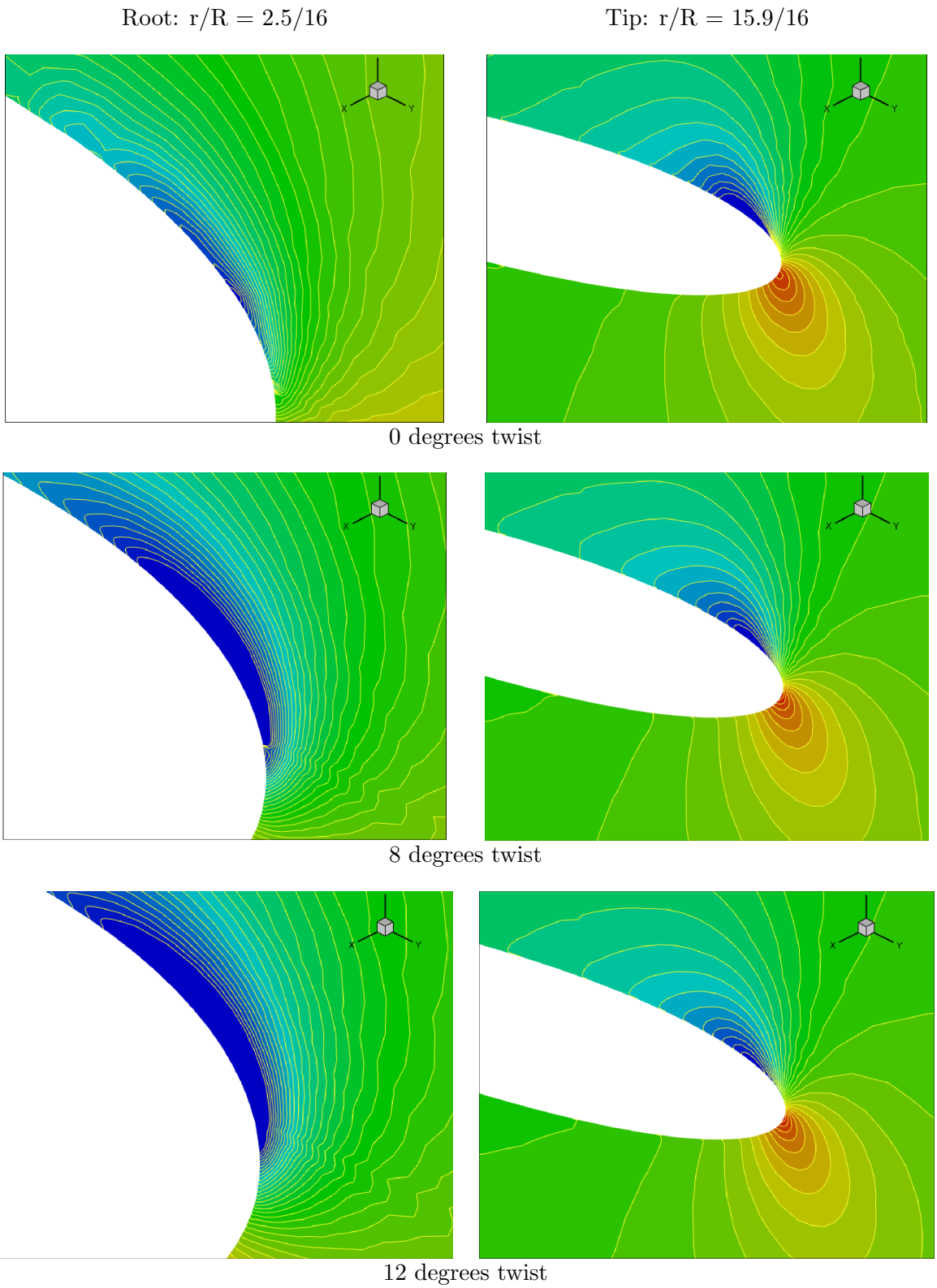


Figure 7.3: *Rotor in hover, Aspect Ratio = 16, Coning = 0, Anhedral = 0, Collective = 12^o and various twist distributions at the root and tip. $M_{tip} = 0.6$.*

7.2 Objective Function and Metamodel

The objective was to keep the FM constant at a high value for a wide range of collective angles. To capture this, two parameters were used viz. the maximum FM (FM_{max}) over a range of thrust values as well as the gradient of the FM with respect to C_T (∇FM). Eighteen sample points consisting of six collective angles at three linear twist angles (0, 8 and 12°) were obtained using CFD to create the initial database (these data points can be seen in Figure 7.1. ANNs were trained to be able to predict the parameters based on the collective and twist. This amount of data was required to obtain accurate predictions from the metamodels. Validation was carried out for an additional twist distribution of 11 degrees and the results agreed well as seen in Figures 7.1 and 7.2 even though the grids used were relatively coarse. Therefore, these predictions were used to obtain data regarding the two objective components, FM_{max} and ∇FM. Once the training for the components were complete, they were combined in an objective function that acted as the 'fitness' value for the optimiser. The objective function is given in Equation 7.2.

$$OFV = 0.35FM_{max} - 0.55\nabla FM + 0.2 \tag{7.2}$$

The weights in Equation 7.2 were such that the importance of having little change in FM must be greater than maximising FM for a single point. The average ratio of FM_{max} to ∇FM was 1:1.0396. Therefore, the limiting weights that would give ∇FM a greater than or equal weight to FM_{max} using the average ratio was:

$$w(FM_{max}) = \frac{1.0396}{1 + 1.0396} = 0.51 \tag{7.3}$$

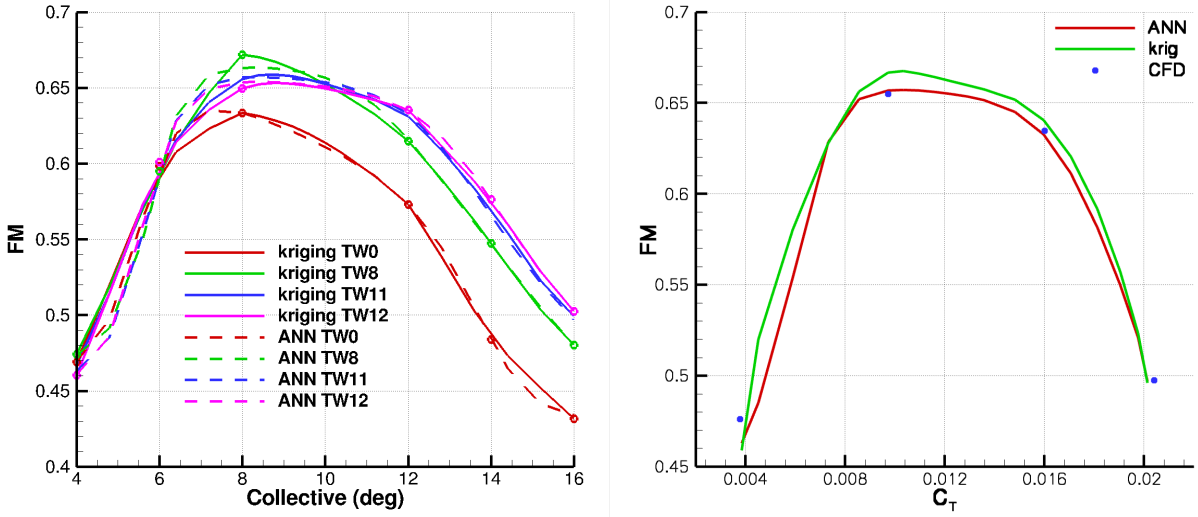
$$w(\nabla FM) = \frac{1}{1 + 1.0396} = 0.49 \tag{7.4}$$

However, a greater difference in the weights was selected to ensure that ∇FM always has a higher weighting than FM_{max}.

Kriging was also used to interpolate between the CFD data. Both, the ANN and kriging methods, predicted the data with approximately the same accuracy. Figure 7.4 shows the comparison. The case for 11 degrees twisted blade was not included in the training data. Therefore, it is used to show the comparison in prediction accuracy between the two methods relative to the CFD data for that twist distribution. The kriging tends to have a smoother prediction of the distribution for the known data. However, for the unknown data, the ANN predictions are more accurate and hence the ANN method was used for the optimisation. Figure 7.5 shows the predictions for these two objectives.

7.3 Blade Twist Optimisation

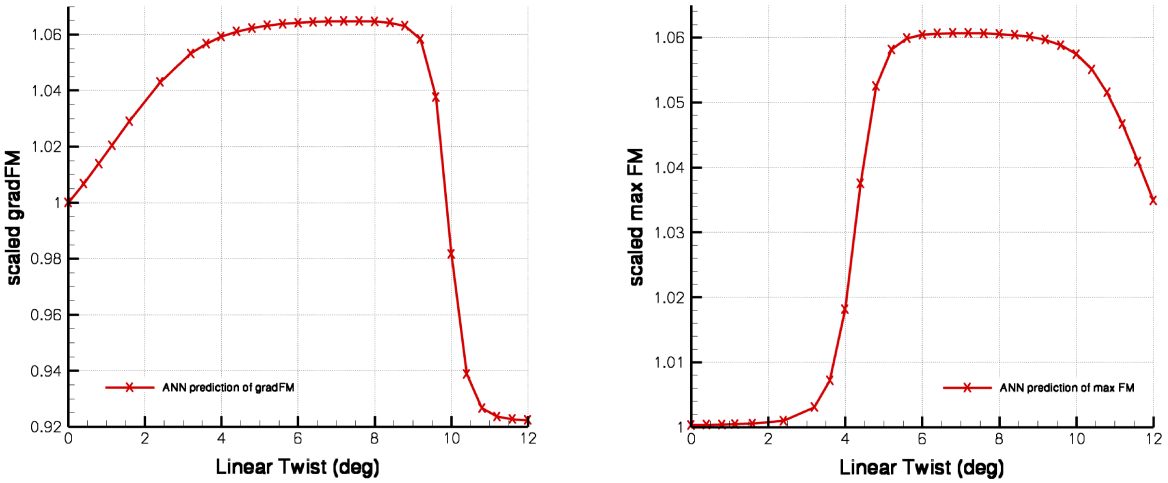
The GA was employed and run for five generations with 500 occurrences of crossovers in each generation, a 1 in 16 chance of mutation and a margin of fitness definition for the elite of 0.02. The time to complete this task was 21.2 seconds using a single Pentium processor, starting from a population of three. The optimum twist selected by the GA is shown in Figure 7.6. The optimum twist lies between 10 and 12°, with the maximum point being around 11°. Figure 7.7 shows the C_P distribution of the various twist distributions at a collective of 12 degrees. The stall can be seen clearly at the 0 degrees of twist blade whereas the twisted blades are free of stall and the pressure distribution is more uniform. This is also easily seen in the wake visualisation using the Q-criterion in Figure 7.8. The slight improvement gained by using 11° rather than 12° is difficult to find by comparison of the flow field but it is clearly visible in the FM vs. C_T curve in Figure 7.2. The performance of the 11° is only slightly degraded at higher collectives, i.e. it maximises the plateau by allowing only small drops in FM with collective, allowing higher achievable values of maximum FM compared to the 12° twist.



FM vs. Collective

FM vs. CT for 11^o

Figure 7.4: *ANN and kriging predictions and validation for FM.*



(a)

(b)

Figure 7.5: *ANN predictions of (a) FM_{max} vs. twist and (b) ∇FM vs. twist.*

7.3.1 Sampling Technique Test

This case was also used to demonstrate the effect of using the full factorial and Latin Hypercube (LHS) sampling techniques. Figure 7.9 shows the predictions by the ANN of FM against collective using three databases. When the LHS method using random selected points is used, the error in the predictions is large. However, it is known that the maximum FM is achieved at a higher collective with increasing twist. Therefore, selecting the point of 8 degrees collective at a twist of 8 degrees instead of at twist 0 degrees, made the predictions much more accurate. For cases where there are many maxima or minima and where such a prediction from experience cannot be made, the LHS is not as accurate. The LHS is usually useful where the amount of data points is large, unlike this case, and picking a set number of points that fit the criterion of the LHS (described in Section 3.3) does not result in big changes in the outcome.

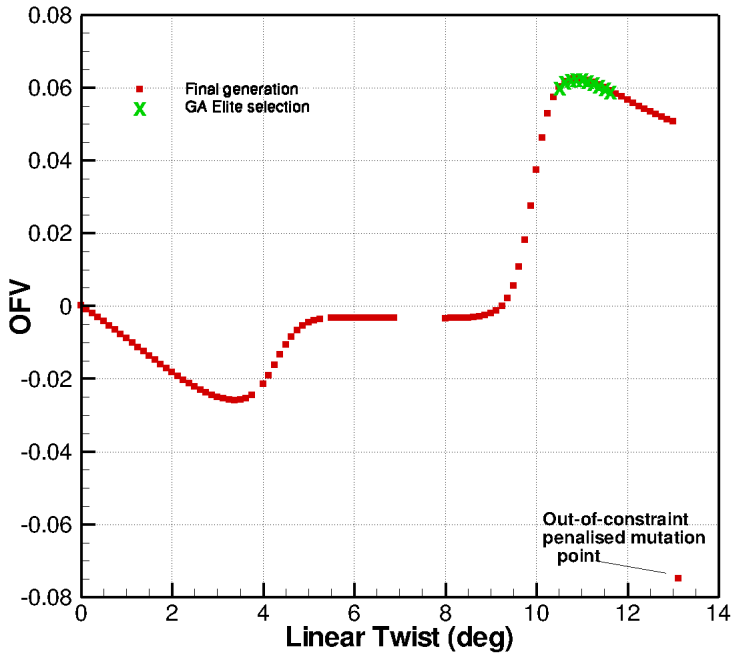
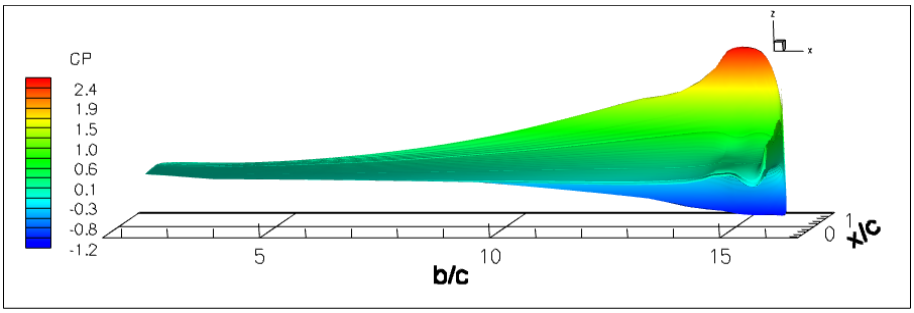
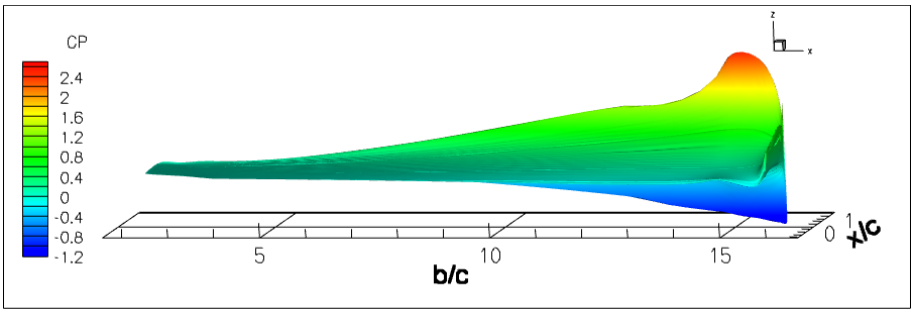


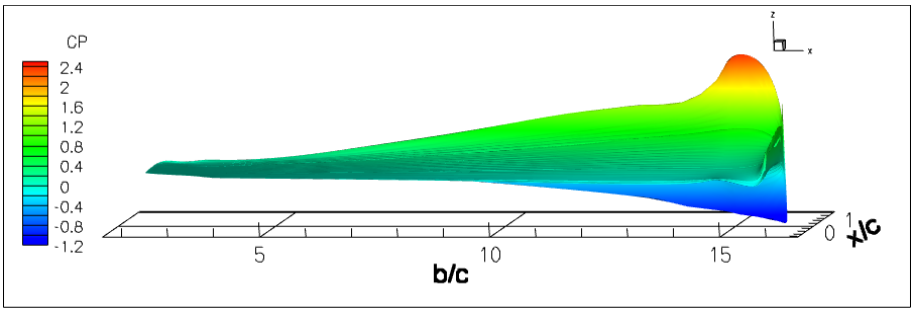
Figure 7.6: *GA selection of optimum twist based on the maximum FM and the change in FM with collective angle increase for 5 values of collective used to train the corresponding ANN.*



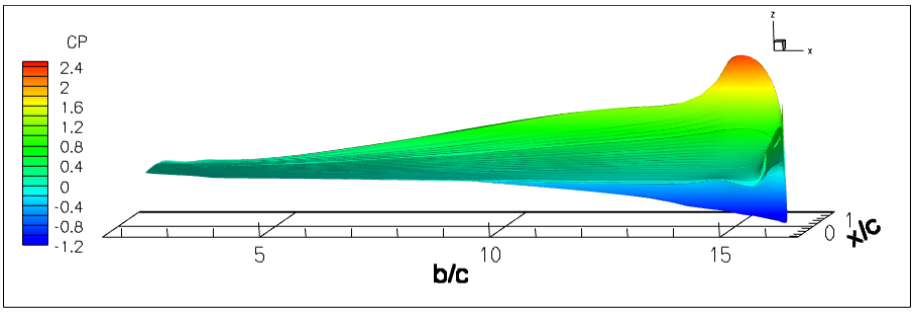
0 degrees twist



8 degrees twist



11 degrees twist



12 degrees twist

Figure 7.7: *Blade load (C_p) at 12 degrees of collective.*

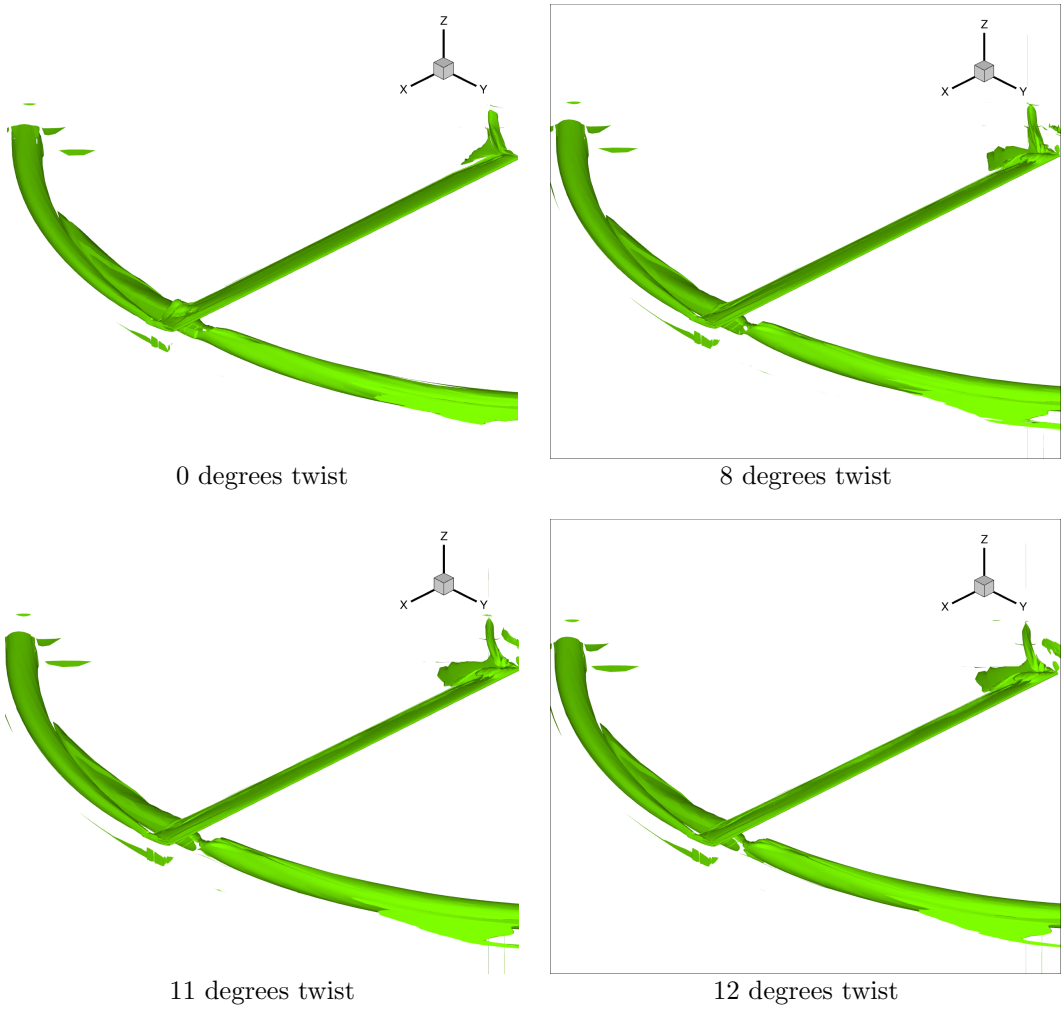


Figure 7.8: *Visualisation of tip vortices using Q criterion (value of 0.1) for a range of twists at $\theta = 12$ degrees.*

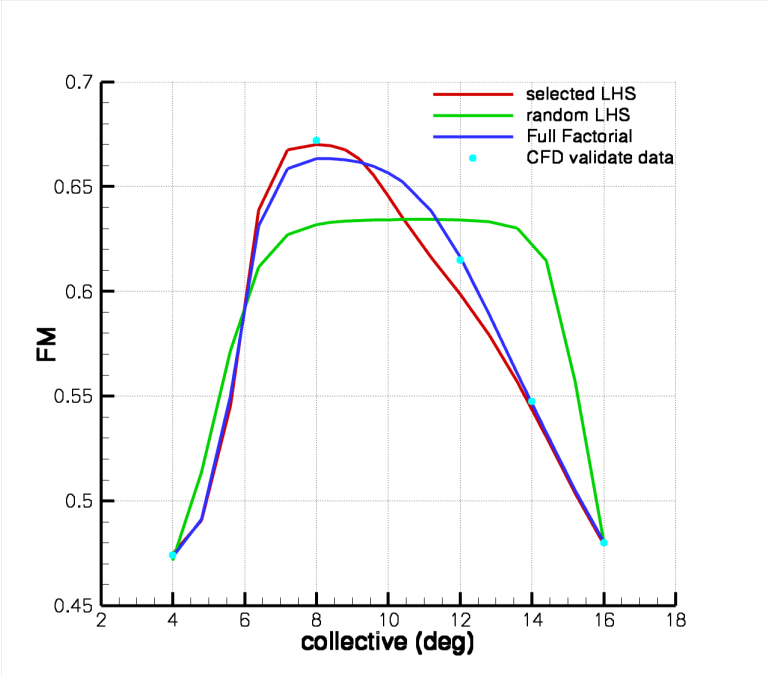


Figure 7.9: *LHS comparison for the FM of a rotor in hover.*

Chapter 8

Forward Flying Rotor Optimisation

For this case, the UH60-A rotor was used as a starting point because it was designed for high speed flight. It also has an extensive set of experimental data for forward flying rotors in the public domain. This rotor has an AR of 15.5. It has a tip sweep of 20° and there is also a reversal of twist near the tip. Figure 8.1 shows the twist and aerofoil distribution for the UH60-A blade. Also, blade torsional deformation was used with five harmonics to simulate the aeroelasticity of

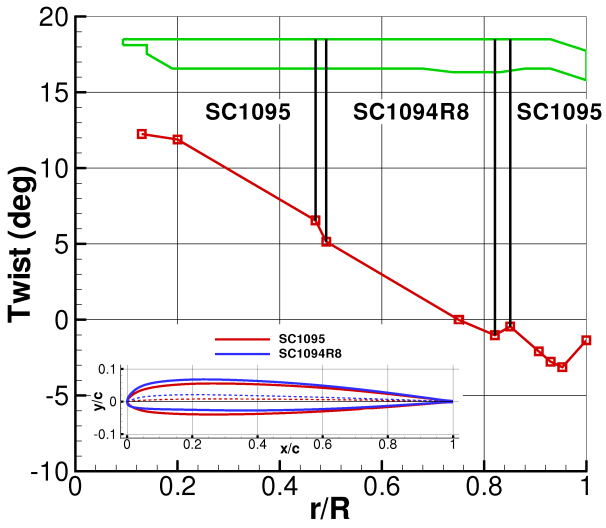


Figure 8.1: *UH60 rotor blade twist and aerofoil distribution.*

the blade as obtained from Datta et al.[144]. This, as well as the conditions of flight are given in Table 8.1. The trim conditions were: single flap harmonic with -0.7 for the cos term and -1 for the sin term, a single pitch harmonic of -2.39 for the cos term and 8.63 for the sin term and no lag harmonics.

8.1 Parameterisation and Grid Generation

The aim is to optimise the sweep and anhedral parameters of the planform for forward flight while constraining hover performance and then optimise the hover performance using twist. The sampling used was full factorial with four cases of anhedral and five of sweep as shown in Figure 8.2. The parameterisation method was simply the use of the angle in degrees of sweep and anhedral. The grid geometry was generated as follows. A is the sweep angle, and with reference to Figure 8.3, the following constants were defined:

- Area of tip = D + E + F = 1.1117
- span, q = r + p = 1.0737

Tip Mach number, M_{tip}	0.641
Advance ratio, μ	0.368
shaft angle	7.31°
Reynolds number, Re	0.5e-6
Freestream mach, M_∞	0.236256
Collective (built-in)	13.47°
Collective	11.6°
Cone angle	3.43°
AR	15.5
Nominal blade twist	-16°
Lock number, γ	8.0
Centre	0,0,0
Flap hinge	1,0,0
Lag hinge	1.1,0,0
Pitch centre	1.5,0,0
Twist harmonics:	
1st	cos: 1.4772260, sin: -1.108055
2nd	cos: -0.5285583, sin: 1.371093
3rd	cos: -0.1224511, sin: -0.098077
4th	cos: 0.5819061, sin: -0.244197
5th	cos: -0.2909757, sin: -0.386423

Table 8.1: *Table showing the conditions of flight for the UH60-A in forward flight.*

- pre-sweep chord (i.e. the chord before the swept part of the blade starts) = 1

$$x = \sqrt{2 - 2CosA} \tag{8.1}$$
$$B = 90 - A/2 \tag{8.2}$$
$$r = xCosA/2 \tag{8.3}$$
$$T1 = xSinA/2 \tag{8.4}$$
$$T2 = pTanA = (q - r)TanA \tag{8.5}$$
Also C1 is defined as: (8.6)
$$C1 = (C - 1)\frac{r}{q} + 1 \tag{8.7}$$

The areas D, E and F are given by:

$$D = \pi r^2 \times \frac{A}{2\pi} = \frac{A}{2} \tag{8.8}$$
$$E = \frac{1}{2}rC1 = \frac{r^2}{2q}(C - 1) + \frac{r}{2} \tag{8.9}$$
$$F = \frac{1}{2}(C1 + C)p = \frac{r^2}{2q}(1 - C) + \frac{q}{2}(C + 1) - r \tag{8.10}$$
$$Area = \frac{A}{2} - \frac{r}{2} + \frac{q}{2}(C + 1) = 1.1117 \tag{8.11}$$
Therefore C is defined as: (8.12)
$$C = \frac{2 \times Area - A + r - q}{q} \tag{8.13}$$

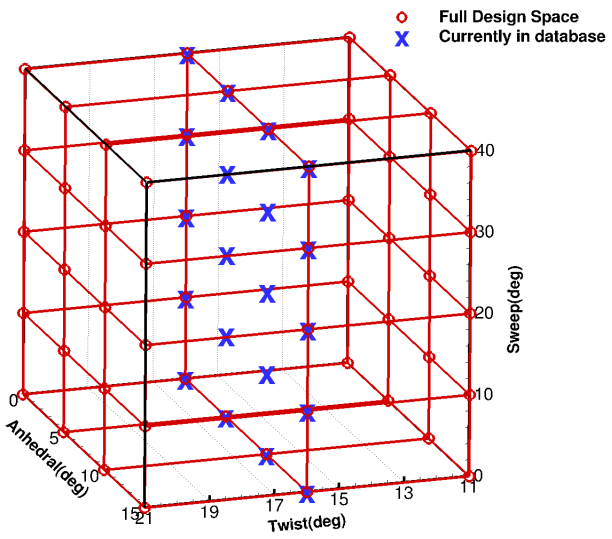


Figure 8.2: *UH60 optimisation for sweep, anhedral and twist CFD design points.*

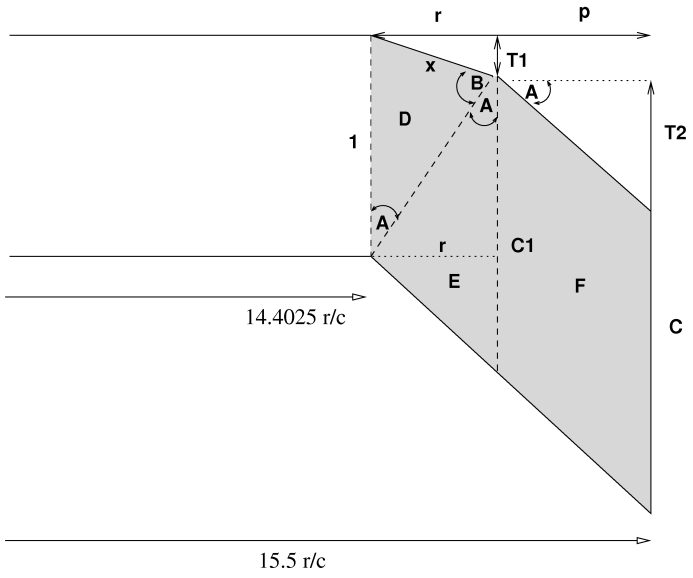


Figure 8.3: *UH60 tip geometry for sweep modification.*

At the tip there is also a reversal of twist and some inherent anhedral. Both of these are non-linear and can be approximated as a second-order polynomial. Since the value of r changes with the sweep angle, the twist and anhedral must be re-defined at the new point. These are defined as quadratics by solving for the three unknowns in each quadratic using three known points. Therefore, the twist as a function of span (for collective 13.47), X is given by:

$$\theta = 3.76696895X^2 - 110.837065731X + 823.338742751 \tag{8.14}$$

And the inherent anhedral as displacement of the quarter chord point from the Z-axis as a function of span, X is given by:

$$\Delta Z = -0.0196387563X^2 + 0.537139079948X - 3.67093898682 \tag{8.15}$$

Table 8.2 shows some examples of the geometry values for different sweep angles including the original of 20°. A small program to automate this is given in Appendix B.13. Also, the curvature

A (deg)	10	20	30	40
x	0.1743	0.3473	0.5176	0.6840
r	0.1736	0.3405	0.5000	0.6428
T1	0.01519	0.06031	0.13397	0.23396
T2	0.1587	0.2663	0.3312	0.3616
C	1.06991	1.06217	1.04875	1.01919
C1	1.01131	1.06417	1.02270	1.01149
θ (deg) at r	8.10695	8.04173	8.17481	8.45692
z at r	-0.01405	-0.02056	-0.02769	-0.03497

Table 8.2: *Geometry values for varying sweep. Areas are divided by c^2 and lengths by c.*

that forms the sweep at the leading edge is modified from an arc to a spline tangent to the straight part of the sweep for a smooth surface. This is done in ICEM by first forming the arc and then matching its point and tangent to the end curve of the sweep. The resulting planform curvature is shown in Figure 8.4.

In addition to sweep, anhedral was optimised as well. There already exists a slight inherent non-linear anhedral for the tip of the UH-60 rotor. Initially, the anhedral to be optimised for was linear and added to the already existing inherent anhedral, i.e.

$$z_1 = rtan\phi \tag{8.16}$$

$$z_2 = (r+p)tan\phi \tag{8.17}$$

where z_1 is the additional drop at the mid-section and z_2 is the additional drop at the tip end. The mid-section between the start of the blade tip section and the tip was moved to the correct location for sweep and the anhedral. Then surfaces were created. Smoothing of the surface was applied between the mid-section and the initial point of the anhedral. Figure 8.5 shows this method of adding anhedral. The geometry produces a bulge at the point where the anhedral is implemented, which is unrealistic. Therefore, a non-linear smoothed arc transition of the anhedral was used instead. This produces a more realistic shape as shown in Figure 8.6.

Figure 8.4: *UH60 tip sweep geometry planform curvature. The black surface has a sweep of 10 degrees and the blue, a sweep of 40 degrees.*

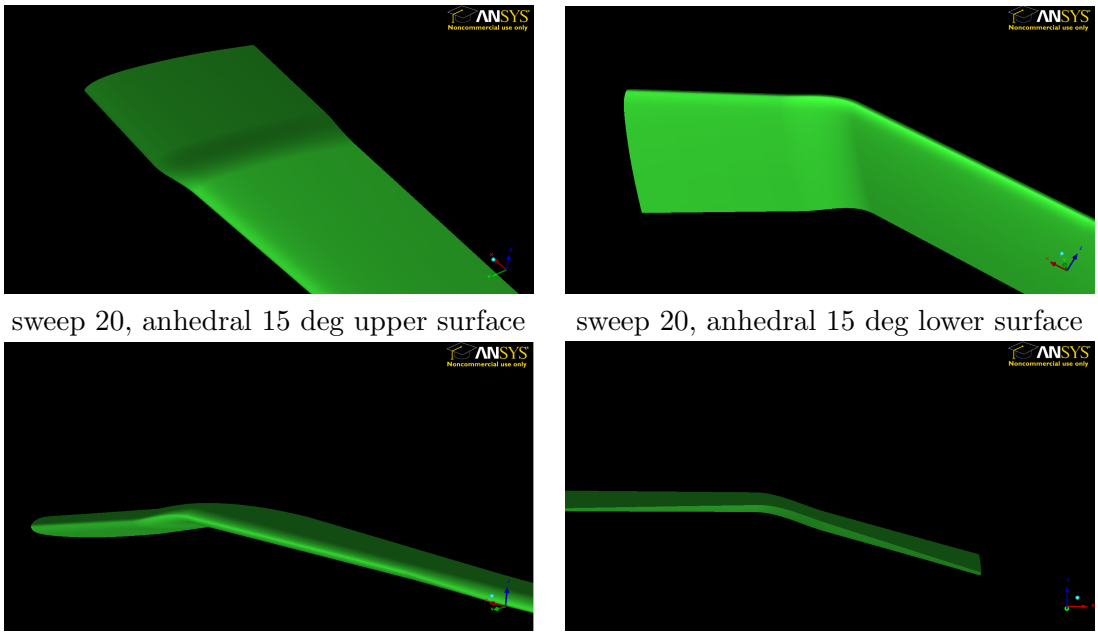


Figure 8.5: *Anhedral implementation using gradient matching between mid section and initiation section.*

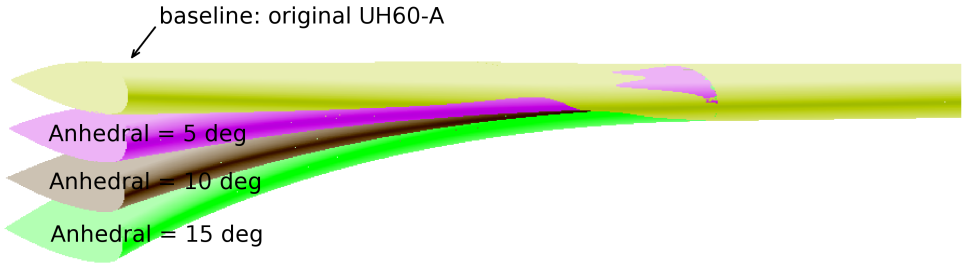


Figure 8.6: *Anhedral implementation for the UH60-A using a smoothed arc.*

8.2 Validation

Validation of the data was carried out for the original UH60-A rotor for two section along the span at r/R = 0.675 and 0.865. The experimental data was obtained from Coleman and Bousman[145]. Figure 8.7 shows the Mach squared C_n and C_m for these two sections. The time marching (TM) matches the experimental data quite accurately. The TM solution was obtained at every quarter degree of azimuth, although the full flow solution was obtained every 10 degrees. For the Harmonic Balance method (HB), four modes were used to construct the solution i.e. a solution exists for every 10 degrees of azimuth. Both methods were accurate in different regions. However, for the optimisation described here, the TM method was used for all the cases.

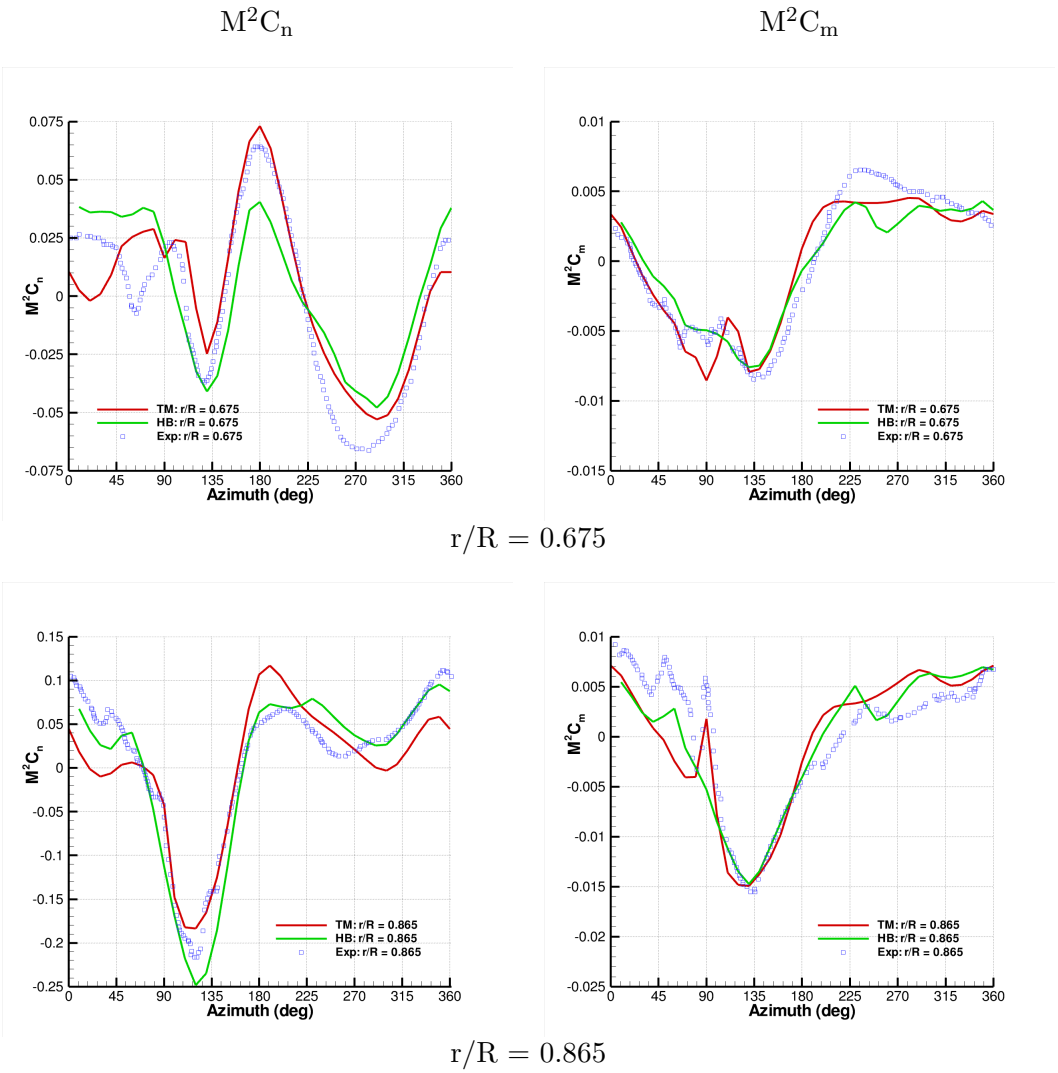
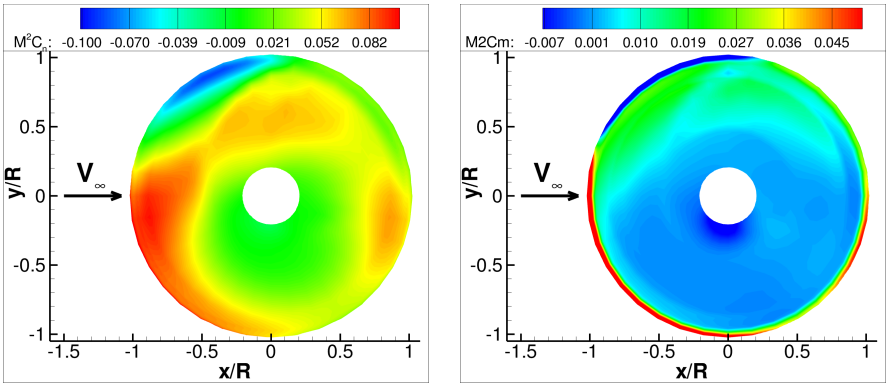


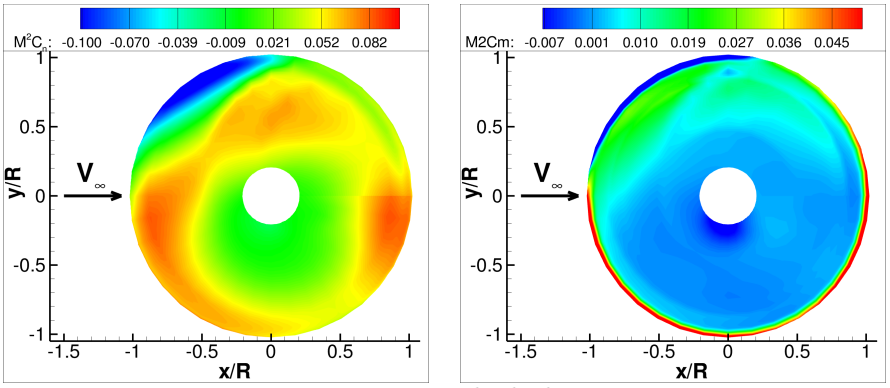
Figure 8.7: *Experimental data in comparison to CFD data by time marching (TM) and Harmonic Balance (HB).*

8.3 Results Analysis

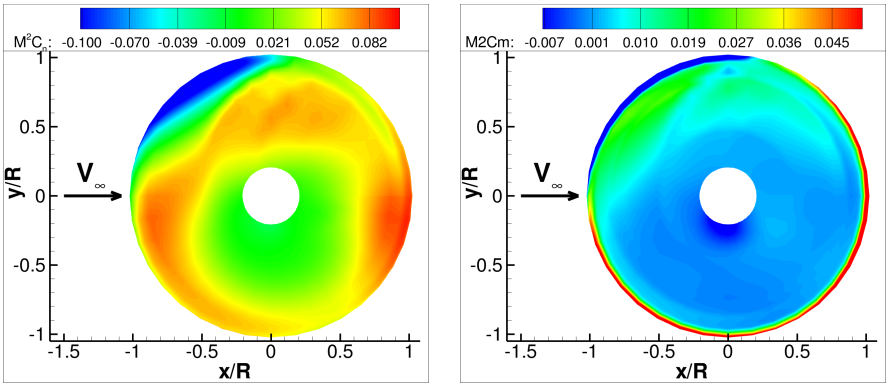
For the 20 cases used to build the initial design space (using a full factorial method),the lifting and pitching moment loads over the full disk were integrated. Some of these are shown in Figures 8.8 - 8.10. The effect of anhedral can be seen in the results for the M^2C_n and M^2C_m shown in Figures 8.8. Adding more anhedral loads the back of the disc more which distributes the disk loading more evenly. It tends to reduce the loading on the advancing side tip, but makes up for it on the retreating side. The moment is more evenly distributed as well.
Figure 8.9 compares the same loading for different sweeps with no anhedral. The loading is reduced overall, but more so at the back of the disk. The drop in the lift at the tip of the disk on the advancing side is reduced with more sweep. The moment distribution is also more even although the tip has a much higher pitching moment magnitude. This is expected with sweep, since the moment is taken about the pitch axis and since the distance of the swept part from the pitch axis is increased with more sweep, more pitching moment is observed. Figure 8.10 shows the same comparison but with an anhedral of 15 degrees. The top figure is the baseline design i.e. with 0 degrees anhedral and 20 degrees of sweep. Having more sweep reduces the loading at the back of the disk without adding load to the front of the disk. It also reduces the drop in lift on the advancing side tip. The anhedral and sweep both add more moment to the tip of the blade and hence the moment variation at the tip is highest for the case with most sweep and anhedral. However, this is made up with more favourable moments more inboards.



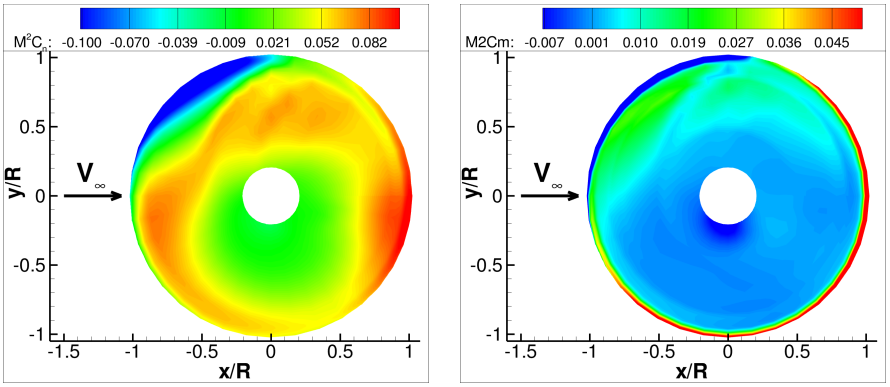
sweep 20, anhedral 0



sweep 20, anhedral 5

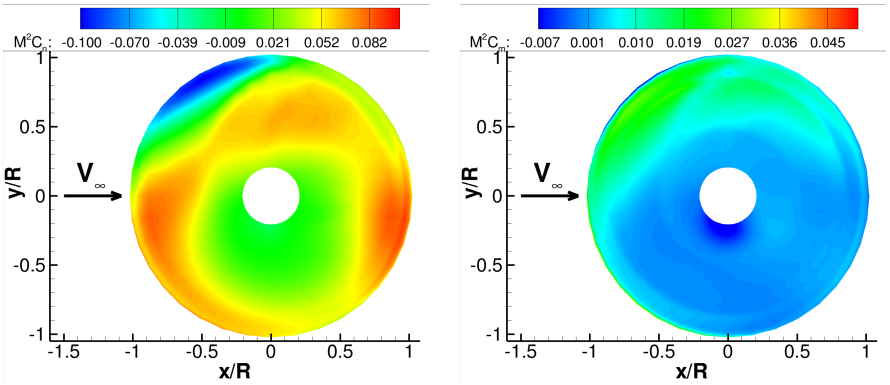


sweep 20, anhedral 10

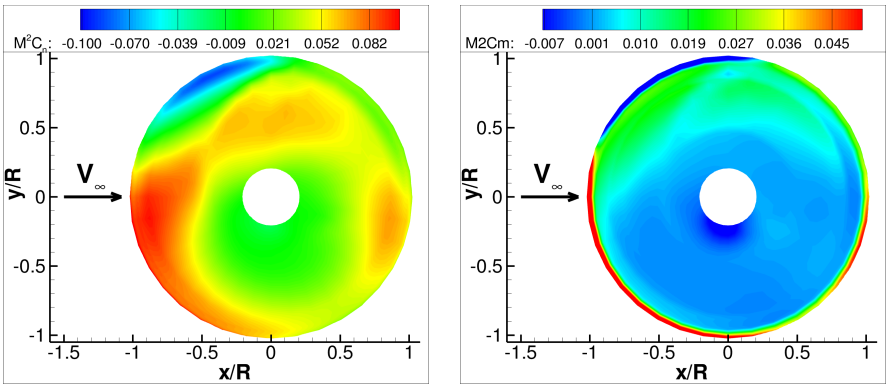


sweep 20, anhedral 15

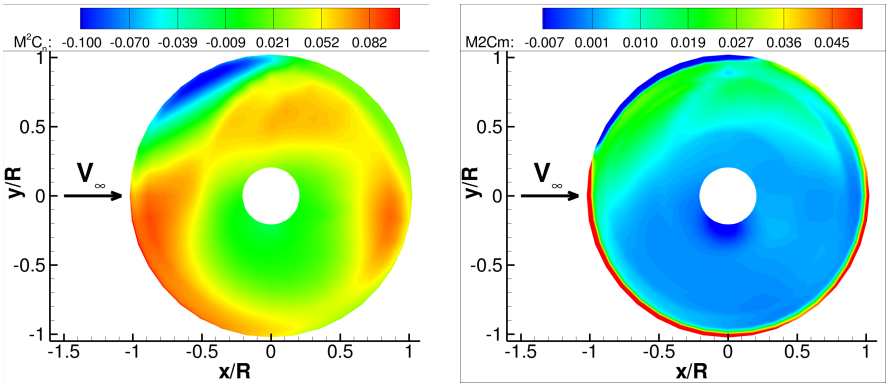
Figure 8.8: *M^2C_n(left) and M^2C_m(right) plots for the UH60 with 20 deg sweep and increasing anhedral.*



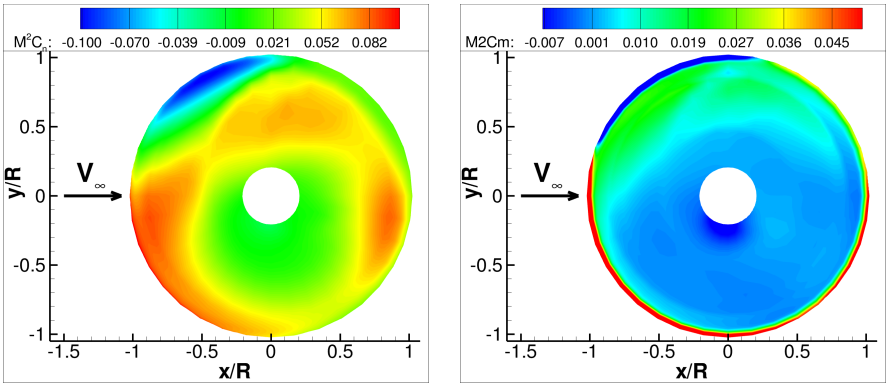
sweep 0, anhedral 0



sweep 20, anhedral 0

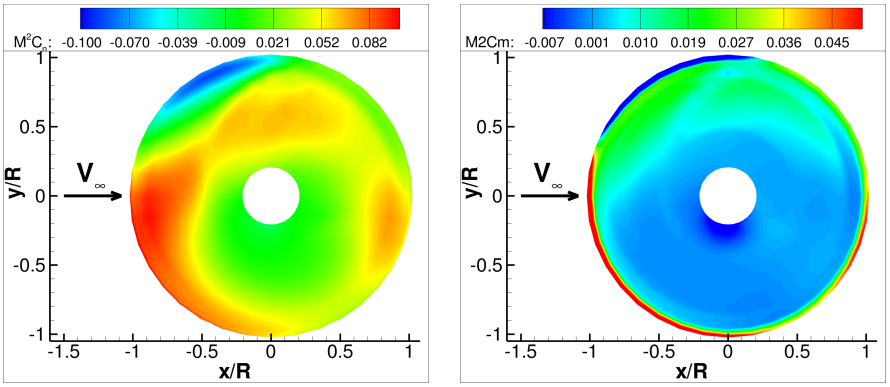


sweep 30, anhedral 0

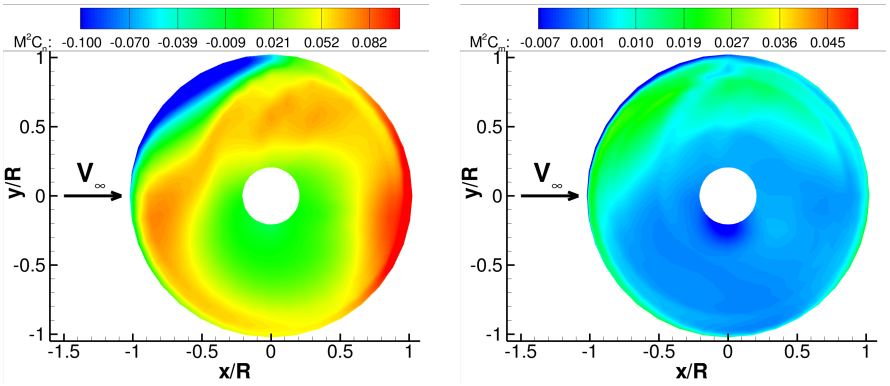


sweep 40, anhedral 0

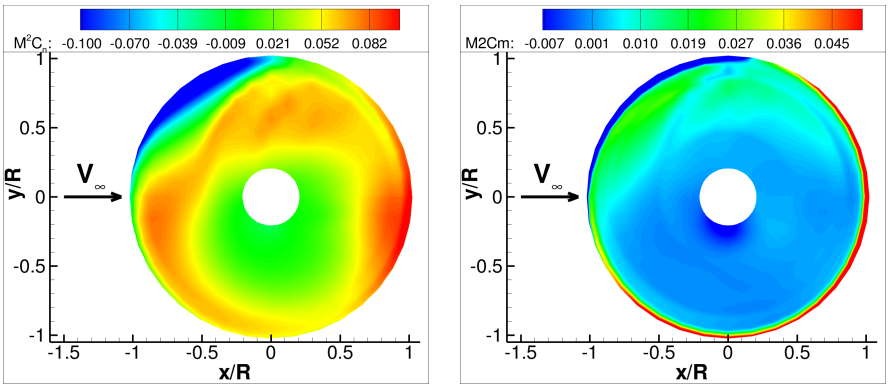
Figure 8.9: *M^2C_n(left) and M^2C_m(right) plots for the UH60 with different sweep and 0 deg anhedral.*



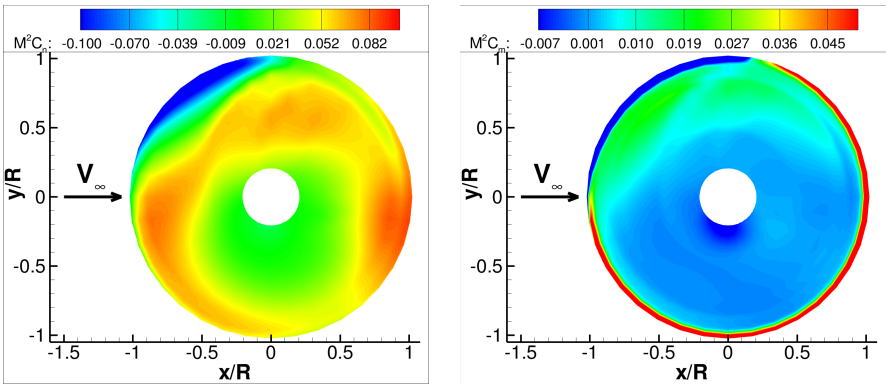
sweep 20, anhedral 0 - original



sweep 0, anhedral 15



sweep 20, anhedral 15



sweep 40, anhedral 15

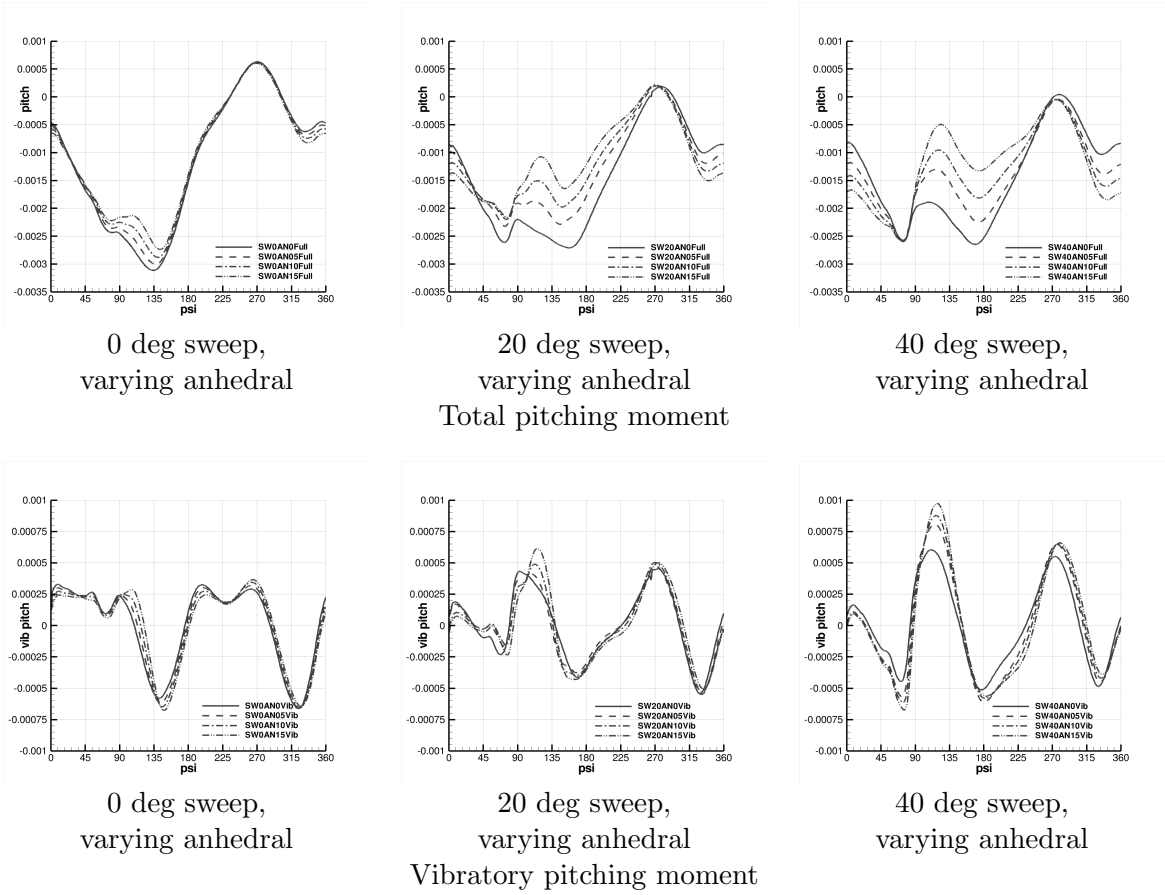
Figure 8.10: *M^2C_n(left) and M^2C_m(right) plots for the UH60 with different sweep and 15 deg anhedral as well as the original rotor.*

8.4 Performance Parameters

The objective was to optimise the UH60-A rotor anhedral and sweep to reduce the pitch loads, stall on the retreating side and shock effects on the advancing side whilst maintaining or improving the thrust, torque and vibratory loads. The stall and shock effects are observed as large changes in the pitching moment. Therefore the objective can be captured with a function that includes the average and peak-to-peak pitching moments. The other parameters can be incorporated as constraints; as a margin of change in thrust, torque and vibratory moments. The vibratory moment is calculated as the pitching moment less the mean and 1/rev oscillation.

The full and vibratory pitching moments for the blade from 0 to 360 degrees azimuth are shown in Figure 8.11 for varying sweep and anhedral. Adding more sweep decreases the peak-to-peak pitching moment. Adding anhedral does not have a large effect on the peak-to-peak moment but it changes the moment most around the front and advancing side of the disk. This change is larger with more sweep. The overall effect on the performance parameter is that it move the average pitching moment closer to zero pitching moment. Adding both, anhedral and sweep, increases the vibratory pitching moment and the effect of anhedral is more significant with more sweep.

Table 8.3 summarises the performance parameters to be used for the optimisation for a few



0 deg sweep, varying anhedral | 20 deg sweep, varying anhedral | 40 deg sweep, varying anhedral

Total pitching moment

0 deg sweep, varying anhedral | 20 deg sweep, varying anhedral | 40 deg sweep, varying anhedral

Vibratory pitching moment

Figure 8.11: *Total and vibratory pitching moments for a single blade over a revolution of the rotor in forward flight. The original rotor has 20 degrees sweep and 0 degrees anhedral.*

of the results to compare the effect of sweep and anhedral on these parameters. High anhedral tends to favour good average and peak-to-peak pitching moment, while more sweep favours good peak-to-peak but has worse average pitching moments. Both suffer higher amplitude vibratory pitching moment when increased.

Sweep (deg)	Anhedral (deg)	pitch$_{avg}$	Δ pitch
40	0	-0.001462	0.002690
30	0	-0.001468	0.002713
20	0	-0.001412	0.002794
20	5	-0.001296	0.002783
20	10	-0.001223	0.002743
20	15	-0.001150	0.002706
Vibratory			
40	0	0.000	0.001116
30	0	0.000	0.001032
20	0	0.000	0.001024
20	5	0.000	0.001012
20	10	0.000	0.001072
20	15	0.000	0.001153

Table 8.3: *Summary of performance of design points using moments of a single blade.*

8.5 Optimisation

The C_T for all the designs were within about 4% of each other and since further trimming was not used, the thrust was neglected as long as it fitted within 5% of the original rotor's performance. For C_Q, the coefficients were within about 5% of each other. The torque constraint was relaxed to allow for diversity, as long as the anhedral and sweep values were constrained within the boundaries of the database. It was treated as a soft constraint, i.e. the points that violated this constraint of greater than 5% increase in torque were included in the next generation but were penalised first.

To capture the objectives, the average pitching moment ($\overline{C_m^{pitch}}$) and the overall peak-to-peak pitching moment ($\Delta C_m{}^{\text{pitch}}$) would make up the components of the objective and the vibratory pitching moment peak-to-peak ($\Delta C_m{}^{\text{vib-pitch}}$) value would be added as a constraint to the torque coefficient.

The two components of the objective function were weighted equally. However, since on average, the ratio of $\Delta C_m{}^{\text{pitch}}$ to $\overline{C_m^{pitch}}$ is 1.164:1, the weights that would weight them equally were found to be:

$$\frac{1}{n}\sum_{i=0}^{n}\frac{\Delta C_m^{pitch}}{\overline{C_m^{pitch}}} = 1.164 \tag{8.18}$$

$$\Delta C_m^{pitch} : \overline{C_m^{pitch}} = 0.47 : 0.53 \tag{8.19}$$

So the overall objective function was:

$$OFV = -0.47\Delta C_m^{pitch} - 0.53\overline{C_m^{pitch}} + 1, \text{if} \Delta C_m^{vib-pitch} \leq 5\% \text{ and scaled } C_Q \leq 1.0$$

$$\text{otherwise, } OFV = -0.47\Delta C_m^{pitch} - 0.53\overline{C_m^{pitch}} + 1 - 0.5(\Delta C_m^{vib-pitch} - 1.05) - 0.5(C_Q - 1.02)$$

All the performance values of the design points were scaled with a reference rotor, which was the original rotor in this case. These scaled values were used to train the ANN. The ANN predictions are shown in Figure 8.12 for each of the components of the objective function. The ANNs accuracy was also estimated relative to the change in the performance of the design obtained using the CFD data. The maximum error in the objective function obtained was found to be 0.85%.

The GA was then used to find the optimum design using 500 iterations over five generations. $\Delta C_m{}^{\text{vib-pitch}}$ was constrained to be not more than 5% higher than that of the original blade and

C_Q not more than 2%. The result are also shown in Figure 8.13(a). Table 8.4 shows some of the designs used to train the ANNs, as well as an additional data point that had a sweep 17 degrees and an anhedral of 11 degrees used to validate the ANN in scaled values. This point performed better than the baseline case and was not too far from the optimal region. The average, peak-to-peak moments, and torque coefficient are reduced and there was a 5% increase in vibratory peak-to-peak moment. The points selected by the GA also lay on the Pareto front as shown in Figure 8.13(b). Again, it can be seen that the objective function method confines the optima to a region of the design space as opposed to a spread of the best compromise between the design points. The aerodynamic benefit obtained from the optimisation is due to the change in distribution of the loads. This is shown in Figure 8.14. Generally, the anhedral off-loads the tip of the rotor for most of the cycle. This allows the sweep to be reduced which reduces the average pitching moment.

Sweep(deg)	Anhedral(deg)	$\overline{C_m^{pitch}}$	$\Delta C_m{}^{\text{pitch}}$	$\Delta C_m{}^{\text{vib-pitch}}$	C_Q	OFV	Remark
20.0	0.00	1.0000	1.0000	1.0000	1.000	0.000	original
20.0	15.00	0.7485	0.8145	1.1245	0.906	0.183	best in initial population
17.1	11.00	0.7594	0.8239	1.0525	0.933	0.209	best new design by GA

Table 8.4: *Comparison of optimised and original UH60-A rotor blade in terms of pitching moment performance.*

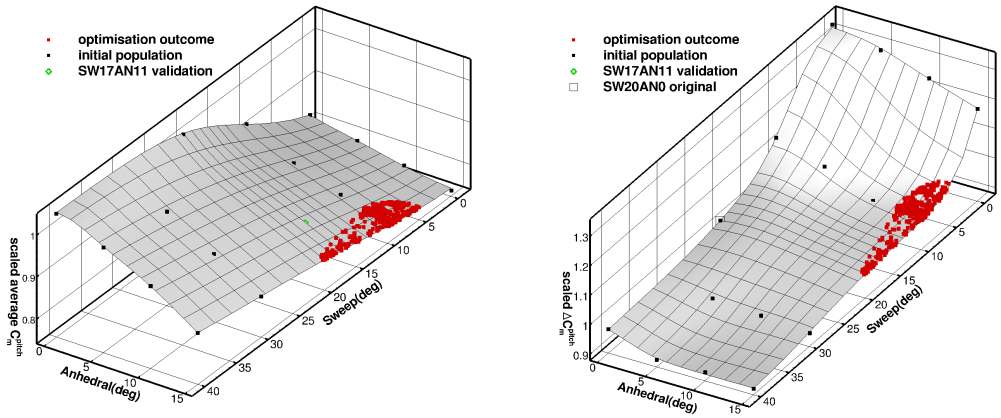
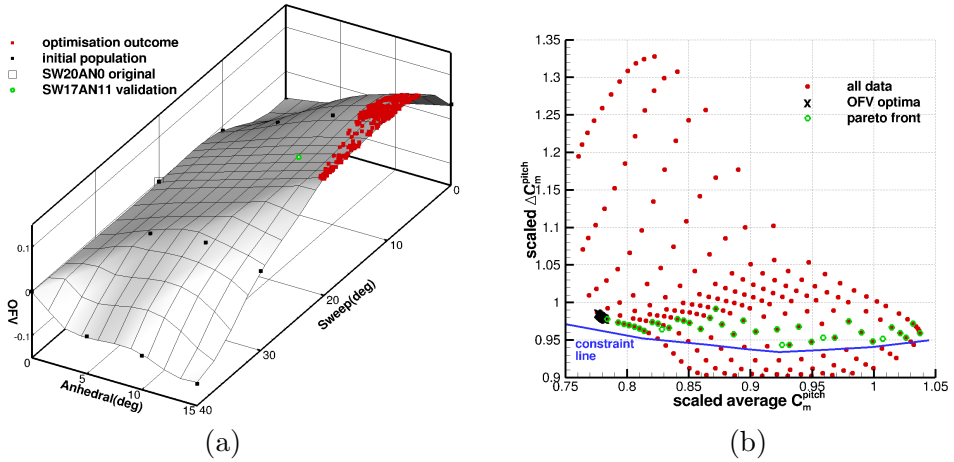
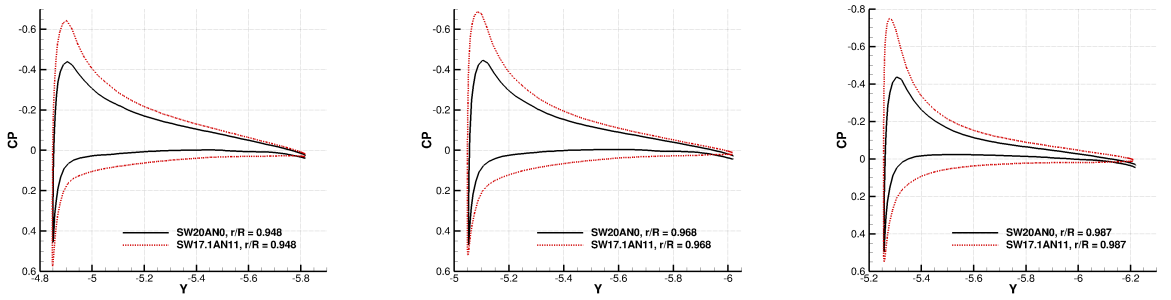


Figure 8.12: *ANN prediction for pitching moments of a blade with varying sweep and anhedral. Values were scaled with original blade of 0 deg anhedral, 20 deg sweep. Black dots on the plot are data used for training the ANN. GA selections are shown as red dots.*

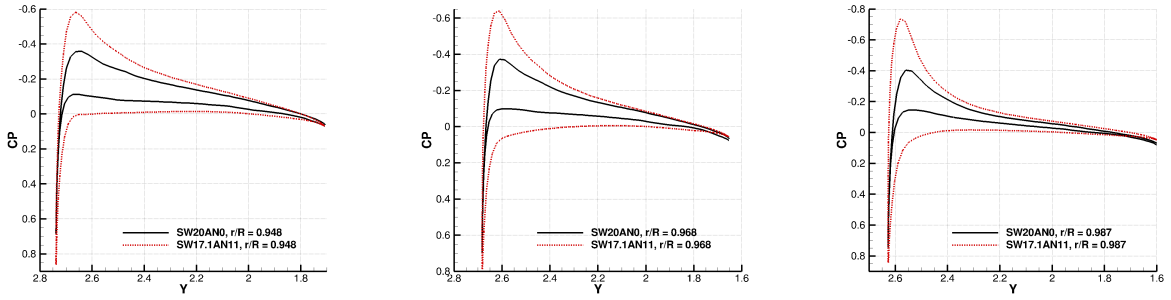


(a) (b)

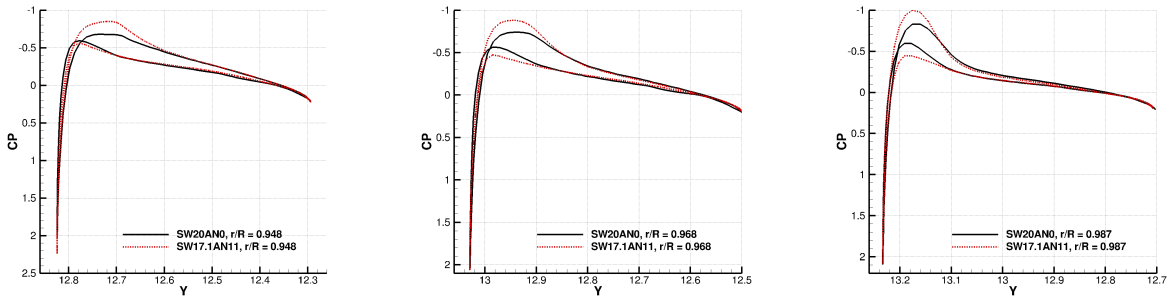
Figure 8.13: *(a) Genetic algorithm results, (b) Comparison between Pareto front optimisation and objective function selection.*



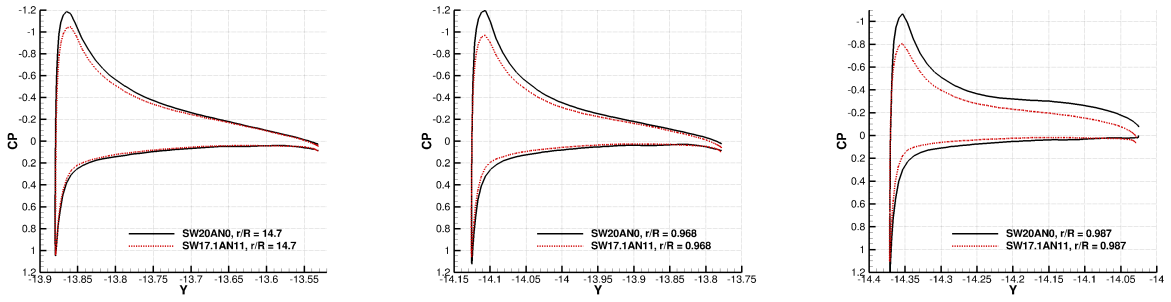
$\Psi = 70^o$



$\Psi = 100^o$



$\Psi = 150^o$



$\Psi = 340^o$

Figure 8.14: *C_p plots at different azimuth angles for the original (SW20AN0) and validation point (SW17.1AN11) UH60 rotors at the swept part of the blade, where $R = 15.5$.*

8.6 Hover Performance

To ensure that the optimised blade for forward flight does not compromise the performance in hover, the rotor with 17 degrees sweep and 11 degrees anhedral was analysed in hover. In addition, its twist was optimised for hover performance. The original blade had a twist of 16 degrees. In addition, a twist of 21 and 11 degrees was analysed using the HMB time marching method. As in the hover case in chapter 7, the optimisation was for maximising the FM over a range of thrust values. A database of 9 values were used i.e. three collective settings for each twist distribution. Table 8.5 shows the results.

It can be seen that the optimised blade performs better than the original blade even with the

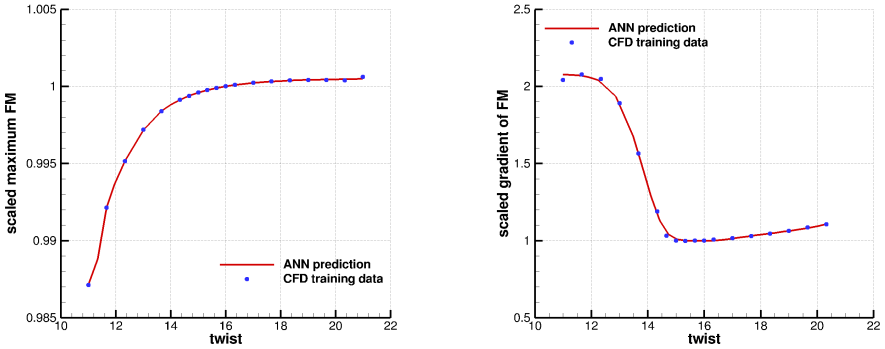
Sweep	Anhedral	Twist	C_T	C_Q	FM
17.1	11	11	2.20E-002	2.46E-003	0.646576
17.1	11	11	1.50E-002	1.11E-003	0.760250
17.1	11	11	9.78E-003	5.89E-004	0.707279
17.1	11	21	2.20E-002	2.05E-003	0.767487
17.1	11	21	1.49E-002	1.06E-003	0.785730
17.1	11	21	1.02E-002	6.20E-004	0.714826
17.1	11	16	2.20E-002	2.11E-003	0.744721
17.1	11	16	1.49E-002	1.07E-003	0.786982
17.1	11	16	9.94E-003	5.90E-004	0.721870
20	0	16	2.20E-002	2.25E-003	0.700278
20	0	16	1.50E-002	1.10E-003	0.767443
20	0	16	9.85E-003	5.96E-004	0.706070

Table 8.5: *Initial CFD database for Hover Optimisation.*

same twist distribution in terms of both FM and C_Q. An ANN was trained to predict the FM based on the thrust and these predicted values were used to create a database to train the ANN for the optimisation function parameter: FM_{max} and ∇FM. These predictions are shown in Figure 8.15. The ANN surface created for FM against collective and twist can be seen in Figure 8.16(a). The optimisation function was as follows:

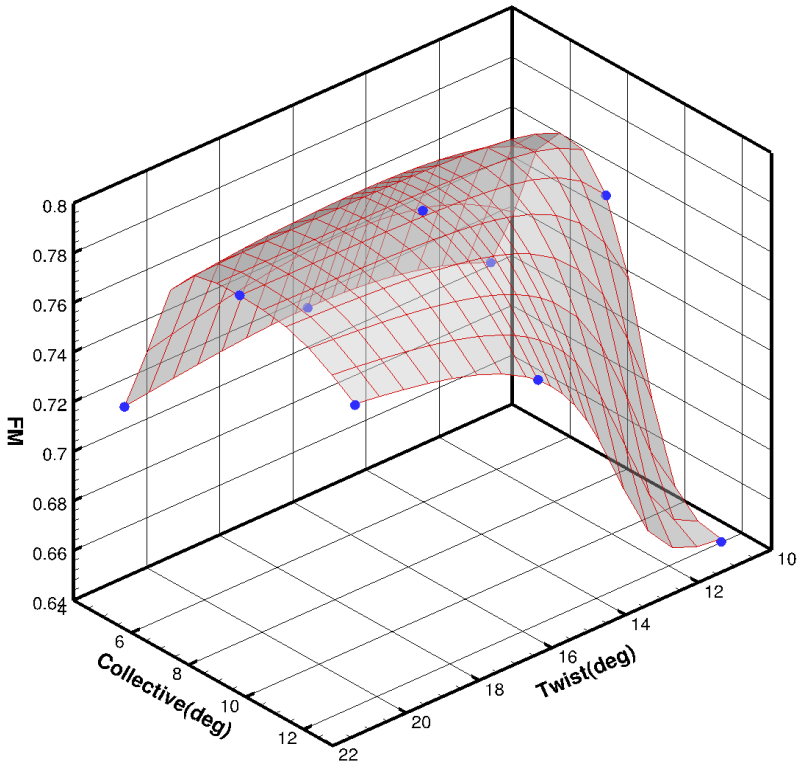
$$OFV = 0.48FM_{max} - 0.52\Delta FM - 0.04 \tag{8.20}$$

Once the GA was run, the optimum was found to be approximately 14.5 degrees of twist, a difference of -1.5 degrees from the original twist distribution as shown in Figure 8.16(b). As can be seen, the change in the OFV around that value of twist changes very little with twist. Therefore, it can be assumed that the hover performance is improved with the new planform design. This is expected, as it is well-accepted that in hover, less sweep and more anhedral benefits the rotor[13] .

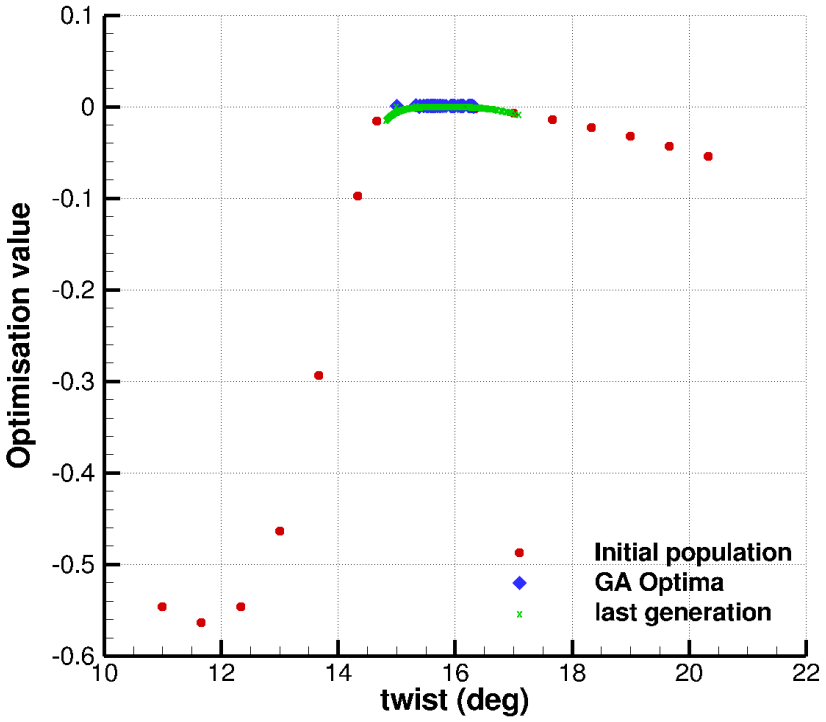


FM_{max} ∇FM

Figure 8.15: *ANN predictions of FM_{max} and ∇FM.*



(a)



(b)

Figure 8.16: *(a) ANN prediction of the FM, (b) Optimum twist distribution for hover of the blade optimised for forward flight.*

Chapter 9

Fuselage Parameterisation and Optimisation

Fuselage drag is a major contributor to the overall drag of a helicopter because of its bluff, not streamlined, shape and its additional components such as non-retractable landing gear, weapons, rear-facing surfaces etc. Also, helicopters tend to yaw and fly at some pitch which makes it difficult to obtain a streamlined fuselage at all conditions[146] .
At low speeds, the effect of the rotor wake on the fuselage also becomes significant[104] and further interactional effects contribute to the drag.

9.1 Grid Generation

For the grid generation, standard multi-block topologies (as described in Section 2.3) were generated using ICEM-Hexa and these were projected on the fuselage shapes. The system for HMB makes use of the ICEM-Hexa scripting language to generate geometries in an easy-to-use fashion. These are combined with pre-existing multi-block topologies to produce the meshes.
The outline of the mesh generation process is as follows:

- Step 1: Components of a generic fuselage are generated using ICEM replay files (extension *.rpl*) from parameterisation coefficients. Example of such a script can be found in Appendix B.8.1.
- Step 2: An ICEMCFD replay file loads the components in ICEM and produces points, curves and surfaces as shown in Figure 9.1.
- Step 3: A pre-defined topology is loaded in a far-field domain as shown in Figure 9.1.
- Step 4: The blocks are then re-associated with the geometry.
- Step 5: The mesh is exported to HMB format.

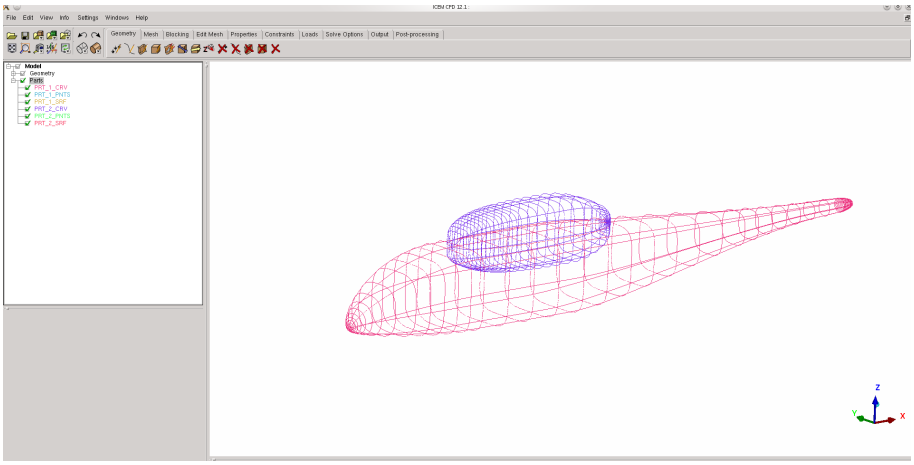
The replay files were generated using a fortran code[129] . This code (Appendix B.8.3 - B.8.2) reads in a set of parameterisation coefficients and creates a set of geometry points and an ICEMCFD replay file. The replay file can be run in ICEMCFD where it opens the point geometry created by the fortran code and creates the surfaces of the fuselage body. Depending on the geometry, this program has the ability to create matching patch surfaces or full lofted surfaces. Also, if point data is directly available, the program can read this data directly and create the replay files. It also has the ability to close the ends of the fuselage with a surface or at a point. Then another replay file builds the far-field geometry around the fuselage after which the blocking can be associated and the mesh generated.

This parameterisation method was applied to the ROBIN[104] and ROBIN-mod7[10] bodies. Grids were generated and some initial results were obtained which can be found in Appendix C. However,

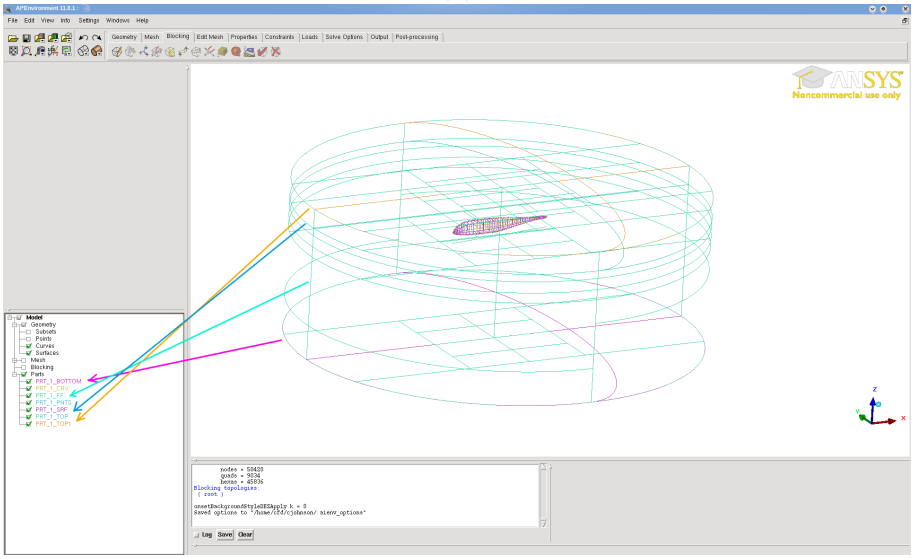
to demonstrate the optimisation process on a fuselage body, they were not selected in preference to the recent results from JAXA on their JMRTS fuselage[7] . This body was parameterised using the same technique for the ROBIN fuselage and the optimisation was performed using some of these parameters.

Number of blocks	160
Number of cells	3.3 million
Wall spacing	1×10^{-5}

Table 9.1: *Table summarising mesh properties for the fuselage.*



(a)



(b)

Figure 9.1: *(a) Step 2 and (b) Step 3 of the mesh generation process demonstrated with the ROBIN body.*

9.2 JAXA JMRTS Fuselage

The JMRTS (JAXA Multi-purpose Rotor Testing System) fuselage (Figure 9.2) developed by JAXA is a generic fuselage designed for investigating the aerodynamics of a rotor/fuselage in forward flight for a less aerodynamic fuselage than the ROBIN. Here, it is used as a demonstration of the capability of this optimisation procedure, specifically the parameterisation part, to improve the drag characteristics of a simplified fuselage. The conditions of flight are those used to obtain the experimental data i.e. Mach number of 0.175, Re = 1.1 million and an angle of attack of -2 degrees[7] .

The fuselage was parameterised using the method described in Section 3.2.3. For the JAXA

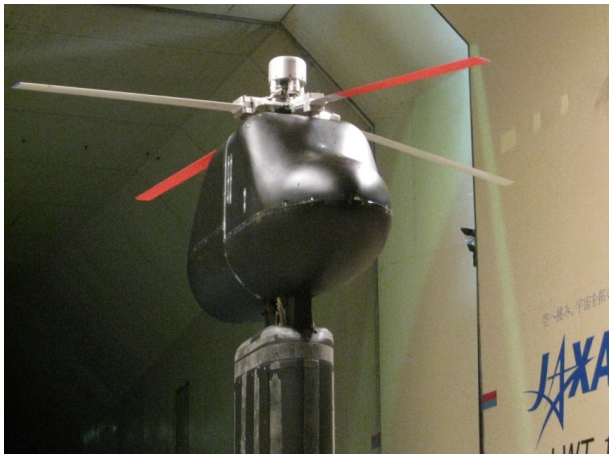


Figure 9.2: *JAXA JMRTS fuselage in the wind tunnel showing that the isolated fuselage data was obtained with the hub grips on*[7] .

JMRTS fuselage, 10 segments were used with 6 variables (8 coefficients each) in each segment, to represent the shape. For each segment, the additional parameter, N_{lw} was employed, since the curvature on the lower surface is different to that on the upper surface. The parameters are given in Table 9.2. The parameters modified to alter the shape are in italics. The value in bold is the control parameter as explained later.
The comparison between the recreated shape and the original shape can be seen in Figure 9.3 which also includes the experimental data on the top surface of the fuselage. The separation at the rear of the fuselage does not occur due to the lack of the hub on the fuselage, which was present in the experimental data. The pressure distribution at a number of stations along the longitudinal axis is also shown in Figure 9.4. An initial solution of the original geometry was obtained and the results showed that the front area of the fuselage (x = -0.68 to x = -0.4) produced approximately 61% of the total pressure drag, the doghouse area (x = -0.4 to x = -0.2) produced approximately 19% of the pressure drag and the back slant (x = 0.2 to 0.4), approximately 17%. This suggests that the front area of the fuselage could benefit from aerodynamic optimisation. The drag values were calculated by integrating the pressure and viscous forces over the surface of the body.

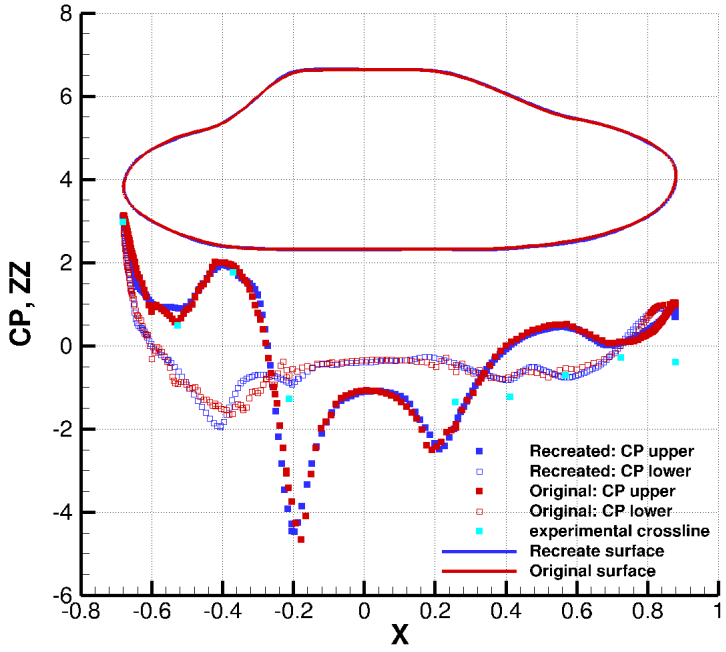


Figure 9.3: *JMRTS Fuselage C_p distribution along Y=0 (centerline). Freestream Mach number is 0.175, Reynold's number is 1.1 million and the angle of attack is -2 degrees. The mesh size is about 3 million cells.*

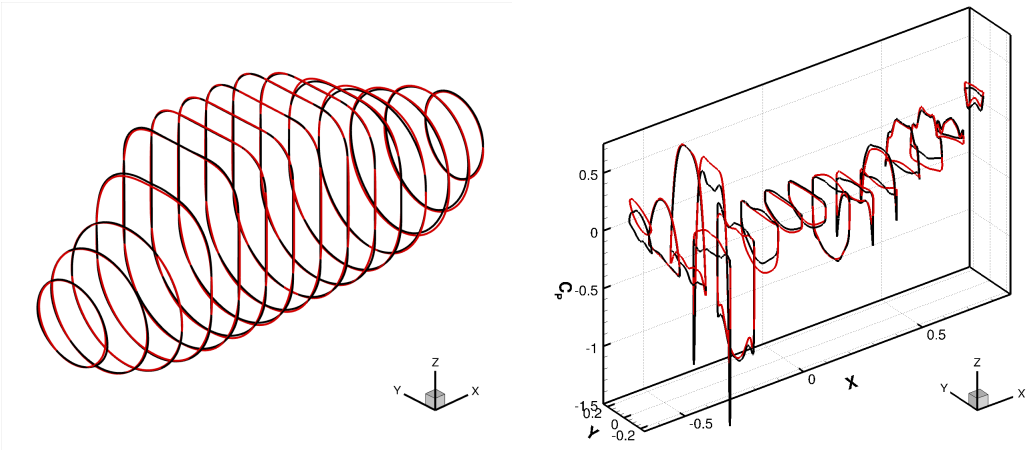


Figure 9.4: *Pressure distribution comparison between the original and the parameterised shape. Freestream Mach number is 0.175, Reynold's number is 1.1 million and the angle of attack is -2 degrees.*

9.3 Surface Parameterisation

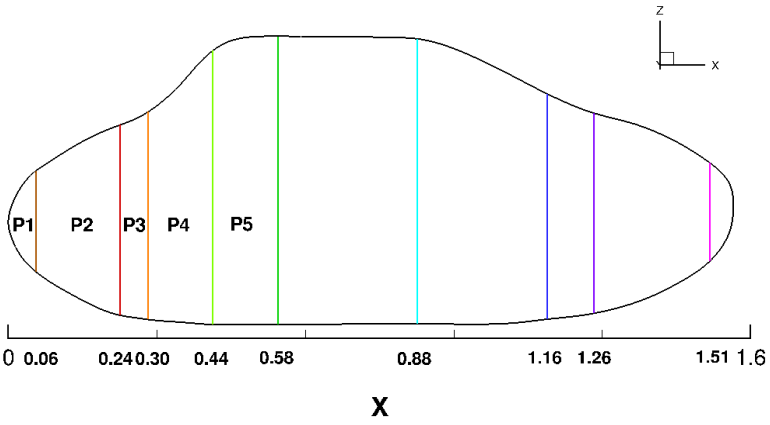
Parameter	C1	C2	C3	C4	C5	C6	C7	C8
Part 1: x = 0.015 to 0.06, Δx = 0.02								
H	0.300	-62.60	-0.150	0.700	4.000	*-0.210*	*1.500*	1.001
W	0.965	-0.950	-0.400	0.400	1.800	0.035	0.420	1.800
Y_o	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Z_o	0.050	-0.050	0.000	1.000	1.000	0.000	0.200	1.001
N_{up}	2.000	0.080	0.000	0.400	1.000	0.000	0.000	1.000
N_{lw}	2.000	-0.064	0.000	0.400	1.000	0.000	0.000	1.000
Part 2: x = 0.06 to 0.24, Δx = 0.05								
H	*0.460*	*-0.770*	*-0.500*	0.700	*2.500*	0.000	0.000	1.000
W	0.960	-1.000	-0.400	0.400	1.800	0.055	0.410	1.800
Y_o	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Z_o	*0.006*	*0.020*	-0.240	1.000	1.000	0.000	0.000	1.000
N_{up}	2.000	0.080	0.000	0.400	1.000	0.000	0.000	1.000
N_{lw}	2.100	-0.230	0.000	0.400	1.000	0.000	0.000	1.000
Part 3: x = 0.24 to 0.30, Δx = 0.05								
H	*0.345*	*2.000*	0.000	0.700	3.500	0.000	0.000	1.000
W	0.950	-1.000	-0.400	0.400	1.800	0.055	0.410	1.800
Y_o	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Z_o	*0.009*	*3.500*	-0.250	1.000	2.000	0.000	0.000	1.000
N_{up}	2.000	0.080	0.000	0.400	1.000	0.000	0.000	1.000
N_{lw}	2.000	-0.064	0.000	0.400	1.000	0.000	0.000	1.000
Part 4: x = 0.30 to 0.44, Δx = 0.05								
H	*0.400*	—***1.645***—	-0.220	0.920	1.500	0.000	0.000	1.000
W	0.446	-1.000	-0.460	0.500	4.000	0.000	0.000	1.000
Y_o	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000
Z_o	-0.005	0.850	-0.220	0.920	1.500	*0.005*	*0.855*	1.001
N_{up}	2.200	3.403	-0.280	0.200	2.000	0.000	0.000	1.000
N_{lw}	1.500	0.690	0.000	0.400	1.000	0.000	0.000	1.000
Part 5: x = 0.44 to 0.58, Δx = 0.06								
H	0.620	-3.950	-0.560	0.920	2.500	0.000	0.000	1.000
W	0.450	-1.000	-0.460	0.500	4.000	0.000	0.000	1.000
Y_o	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000
Z_o	0.099	-2.800	-0.560	1.000	2.500	0.000	0.000	1.000
N_{up}	3.866	0.800	-0.280	0.200	2.000	0.000	0.000	1.000
N_{lw}	1.550	0.650	0.000	0.400	1.000	0.000	0.000	1.000
Part 6: x = 0.58 to 0.88, Δx = 0.05								
H	0.620	-0.070	-0.580	1.000	2.000	0.000	0.000	0.000
W	0.448	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Y_o	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Z_o	0.099	-0.005	-0.580	1.000	2.000	0.000	0.000	0.000
N_{up}	5.600	0.000	0.000	1.000	0.000	0.000	0.000	0.000
N_{lw}	1.400	1.000	0.000	1.000	0.000	0.000	0.000	0.000
Part 7: x = 0.88 to 1.16, Δx = 0.05								
H	0.612	-1.000	-0.880	1.100	1.500	0.000	0.000	1.000
W	0.450	-0.525	-0.800	1.100	2.500	0.000	0.000	1.000
Y_o	0.000	-0.000	-0.000	0.000	0.000	0.000	0.000	0.000
Z_o	0.100	-0.400	-0.880	1.000	1.500	0.032	6.500	0.500
N_{up}	5.750	-10.00	-0.800	1.100	1.000	0.000	0.000	0.000
N_{lw}	2.680	-0.530	-0.600	0.500	1.000	0.000	0.000	0.000

Continued on Next Page...

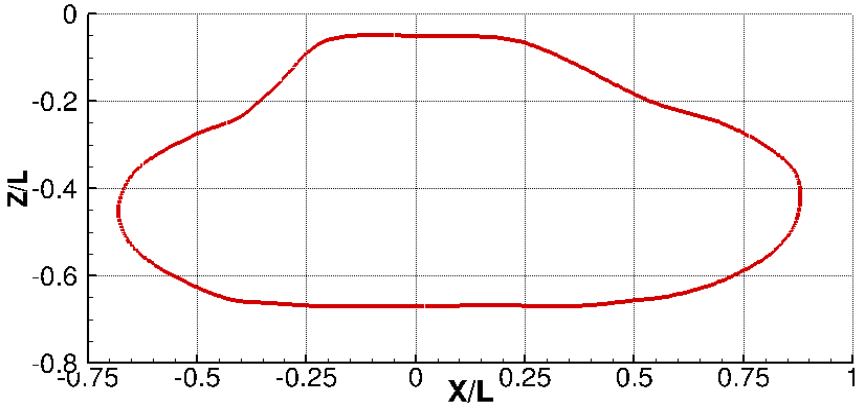
Table 9.2 – Continued

Parameter	C1	C2	C3	C4	C5	C6	C7	C8
Part 8: x = 1.16 to 1.26, Δx = 0.05								
H	1.000	-1.300	-0.900	1.100	1.000	0.312	0.360	0.500
W	0.450	-0.525	-0.800	1.100	2.500	0.000	0.000	1.000
Y_o	0.000	-0.000	-0.000	0.000	0.000	0.000	0.000	0.000
Z_o	1.000	0.290	-1.270	0.800	1.000	-0.094	0.120	0.330
N_{up}	3.750	-2.805	-0.665	1.100	1.000	0.000	0.000	0.000
N_{lw}	2.750	-0.580	-0.600	0.500	1.000	0.000	0.000	0.000
Part 9: x = 1.26 to 1.51, Δx = 0.05								
H	0.445	-1.400	-1.120	0.800	2.500	0.000	0.000	1.000
W	0.410	-3.400	-1.000	1.100	3.500	0.000	0.000	1.000
Y_o	0.000	-0.000	-0.000	0.000	0.000	0.000	0.000	0.000
Z_o	0.027	0.012	-1.280	1.000	1.000	0.000	0.000	0.000
N_{up}	2.600	-0.804	-0.800	1.100	1.000	0.000	0.000	0.000
N_{lw}	2.230	-0.550	-0.800	1.000	1.000	0.000	0.000	0.000
Part 10: x = 1.51 to 1.548, Δx = 0.02								
H	0.175	-2.400	-1.530	0.800	1.000	0.000	0.000	1.000
W	0.142	-2.500	-1.530	1.100	1.000	0.000	0.000	1.000
Y_o	0.000	-0.000	-0.000	0.000	0.000	0.000	0.000	0.000
Z_o	0.027	0.040	-1.280	1.000	1.000	0.000	0.000	0.000
N_{up}	4.900	-4.404	-0.800	1.100	1.000	0.000	0.000	0.000
N_{lw}	2.300	-0.600	-0.800	1.000	1.000	0.000	0.000	0.000

Table 9.2: *Parameters for the JMRTS Fuselage by JAXA.*



(a)



(b)

Figure 9.5: *(a) The parts along the x-axis that make up the parameterised JAXA fuselage, (b) The fuselage grid position showing x coordinate definitions (this is translated compared to Figure 9.5(a) so that the rotor centre is at the origin).*

Parts 1, 2, 3 and 4 define the front of the fuselage up to the point where the wind screen would end as shown in Figure 9.5. In the actual geometry, however, the fuselage is translated so that the rotor centre is at the origin as shown in Figure 9.5(b). The optimisation objective is to improve the drag characteristics of the fuselage by modifying the slope of this front area whilst maintaining the total volume of the fuselage.
To do this, it was decided to use the gradient of part 4 (P4) as the parameter to optimise and to modify all the other parameters related to the front part of the fuselage in association with its value, by matching the height (H) and the vertical position (Z_o) of each section as well as their gradients to P4, while constraining these values at P5 and P1. This was carried out as follows:
First, a new value for C2 of H, at part P4 ($P4.C2^*$) is defined. Then $P4.C1^*$ is found so that the height at x = 0.44 is matched to that of P5 using Equation 9.1 since this coefficient has direct control over the height.

$$P4.C1^* = H^{P5}_{x0.44} - P4.C2^* \left(\frac{0.44 + P4.C3}{P4.C4} \right)^{P4.C5} \tag{9.1}$$

Then using these two new coefficients, the height of P4 and its gradient, at x = 0.30, can be found.

$$H_{x0.3}^{P4^*} = P4.C1 + P4.C2\left(\frac{0.30 + P4.C3}{P4.C4}\right)^{P4.C5} \tag{9.2}$$

$$\nabla H_{x0.3}^{P4^*} = \frac{P4.C5P4.C2^*}{P4.C4}\left(\frac{0.30 + P4.C3}{P4.C4}\right)^{P4.C5-1} \tag{9.3}$$

C1 and C2 for P3 can then be found by matching the height and gradient of P3 to the new values for P4 at x = 0.30.

$$P3.C2^* = \frac{\nabla H_{x0.3}^{P4^*} P3.C4}{P3.C5\left(\frac{0.30+P2.C3}{P2.C4}\right)^{P2.C5-1}} \tag{9.4}$$

$$P3.C1^* = H_{x0.3}^{P4^*} - P3.C2^*\left(\frac{0.30 + P3.C3}{P3.C4}\right)^{P3.C5} \tag{9.5}$$

Similarly to Equations 9.3, the gradient and height can be found with the new coefficients at x = 0.24 for P3.
Now for P2, the gradient and height needs to also be matched to P1 at x = 0.06, where they meet. Therefore, to satisfy four constraints, four unknowns and equations are required. The four unknowns selected are C1, C2, C3 and C5 i.e. C4 is kept constant at a non-zero value. The equations to match are shown below:

$$\nabla H_{x0.24}^{P3^*} = \frac{P2.C5^*P2.C2^*}{P2.C4}\left(\frac{0.24 + P2.C3^*}{P2.C4}\right)^{P2.C5^*-1} \tag{9.6}$$

$$\nabla H_{x0.06}^{P1} = \frac{P2.C5^*P2.C2^*}{P2.C4}\left(\frac{0.06 + P2.C3^*}{P2.C4}\right)^{P2.C5^*-1} \tag{9.7}$$

$$H_{x0.24}^{P3^*} = P2.C1^* + P2.C2^*\left(\frac{0.24 + P2.C3^*}{P2.C4}\right)^{P2.C5^*} \tag{9.8}$$

$$H_{x0.06}^{P1} = P2.C1^* + P2.C2^*\left(\frac{0.06 + P2.C3^*}{P2.C4}\right)^{P2.C5^*} \tag{9.9}$$

For simplicity, this is done iteratively, by first selecting a value for $P2.C5^{H*}$. Then by dividing Eqn. 9.6 by Eqn. 9.7, $P2.C3^{H*}$ can be found as

$$P3.C3^* = \frac{0.06\left(\frac{\nabla H_{x0.24}^{P3^*}}{\nabla H_{x0.06}^{P1}}\right)^{1/(P3.C5^*-1)} - 0.24}{1 - \left(\frac{\nabla H_{x0.24}^{P3^*}}{\nabla H_{x0.06}^{P1}}\right)^{1/(P3.C5^*-1)}} \tag{9.10}$$

Then C2 and C1 can be found using Eqn. 9.6 and 9.8 respectively, as shown below:

$$P2.C2^* = -\left(\frac{\nabla H_{x0.24}^{P3^*} P2.C4}{P2.C5^{H*}}\right)\left(\frac{P2.C4^H}{0.24 + P2.C3^{H*}}\right)^{P2.C5^{H*}-1} \tag{9.11}$$

$$P2.C1^* = H_{x0.24}^{P3^*} - P2.C2^*\left(\frac{0.24 + P2.C3^*}{P2.C4}\right)^{P2.C5^*} \tag{9.12}$$

By subtracting Eqn. 9.9 from Eqn. 9.8, Eqn.9.13 is obtained.

$$P2.C2^*\left[\left(\frac{0.24 + P2.C3^*}{P2.C4}\right)^{P2.C5^*} - \left(\frac{0.06 + P2.C3^*}{P2.C4}\right)^{P2.C5^*}\right] = H_{x0.24}^{P3^*} - H_{x0.06}^{P1} \tag{9.13}$$

The LHS of Eqn.9.13 should equate to the RHS with the new values of C2, C1, C3 and C5 while C5 is changed iteratively till they converge.
The next part is to modify P1 so that it matches the gradient and the height at x = 0.06 and

matches the height at x = 0.015. Again, there are three unknowns required to satisfy three conditions. C3, C6 and C7 were used as the unknowns with the following equations:

$$H^{P2^*}_{x0.06} = P1.C7^* \left[P1.C1 + P1.C2 \left(\frac{0.06 + P1.C3^*}{P1.C4} \right)^{P1.C5} \right]^{1/P1.C8} + P1.C6^* \tag{9.14}$$

$$H^{P1}_{x0.015} = P1.C7^* \left[P1.C1 + P1.C2 \left(\frac{0.015 + P1.C3^*}{P1.C4} \right)^{P1.C5} \right]^{1/P1.C8} + P1.C6^* \tag{9.15}$$

$$\nabla H^{P1}_{x0.06} = \frac{P1.C7^* P1.C5 P1.C2}{P1.C8 P1.C4} \left[P1.C1 + P1.C2 \left(\frac{0.06 + P1.C3^*}{P1.C4} \right)^{P1.C5} \right]^{(1/P1.C8-1)} \tag{9.16}$$

$$\times \left(\frac{0.06 + P1.C3^*}{P1.C4} \right)^{P1.C5-1} \tag{9.17}$$

Subtracting Eqn.9.15 from Eqn.9.14,

$$H^{P2^*}_{x0.06} - H^{P1}_{x0.015} = P1.C7 \left[\left(P1.C1 + P1.C2 \left(\frac{0.06 + P1.C3^*}{P1.C4} \right)^{P} 1.C5 \right)^{1/P1.C8} - \left(P1.C1 + P1.C2 \left(\frac{0.015 + P1.C3^*}{P1.C4} \right)^{P} 1.C5 \right)^{1/P1.C8} \right]$$

$$P1.C6^* = H^{P2^*}_{x0.06} - P1.C7^* \left(P1.C1 + P1.C2 \left(\frac{0.06 + P1.C3^*}{P1.C4} \right)^{P1.C5} \right)^{1/P1.C8} \tag{9.18}$$

By starting with an initial guess for C3 and iteratively solving the above equations for C6 and C7, the final values for these three coefficients can be obtained.

Once the heights have been modified, the next step was to modify the Z_o coefficients so that the lower surface of the fuselage stays the same as the original. For P1 and P4, C6 and C7 are modified. For P2 and P3, C1 and C2 are modified. The new value of Z_o at a particular station, was found by finding half the difference between the new and the old height at that station i.e.

$$Z^* = Z + 0.5\,(H^* - H) \tag{9.19}$$

Then C1 and C2 were found as follows:

$$C2^* = \frac{Z^*_{x1} - Z^*_{xo}}{\left(\frac{0.24+C3}{C4}\right)^{C5} - \left(\frac{0.06+C3}{C4}\right)^{C5}} \tag{9.20}$$

$$C1^* = Z^*_{x1} - C2^* \left(\frac{0.24 + C3}{C4} \right)^{C5} \tag{9.21}$$

And C6 and C7 as follows:

$$C7^* = \frac{Z^*_{x1} - Z_{xo}}{\left(C1 + C2\left(\frac{0.06+C3}{C4}\right)^{C5}\right)^{1/C8} - \left(C1 + C2\left(\frac{0.015+C3}{C4}\right)^{C5}\right)^{1/C8}} \tag{9.22}$$

$$C6^* = Z^*_{x1} - C7^* \left(C1 + C2 \left(\frac{0.06 + C3}{C4} \right)^{C5} \right)^{1/C8} \tag{9.23}$$

Figure 9.6 shows the effect of changing $P4.C2^*$. The original is the red surface, while the white has a lower value and the cyan a higher value. Notice that at the front, the lower coefficient value takes up a greater volume (it is the outer most surface), whereas at the rear, the higher coefficient takes up a greater volume (the cyan has the outermost surface). This is so that the full volume of the body is kept constant and was carried out by modifying the width of the entire fuselage.

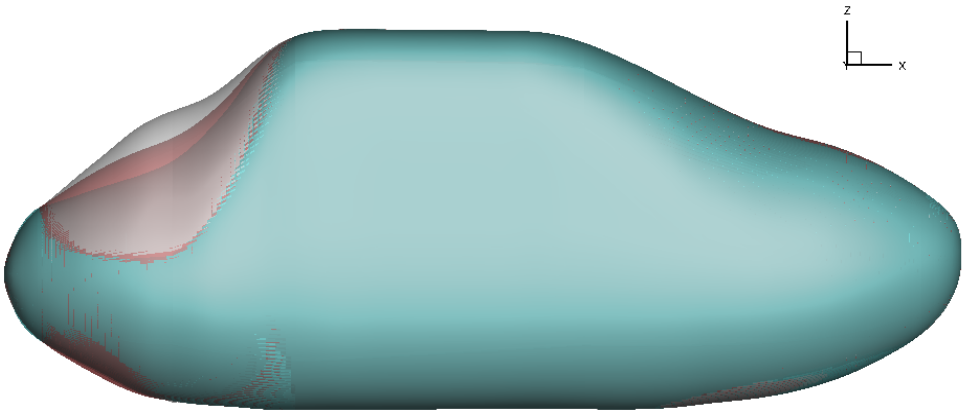


Figure 9.6: *JMRTS fuselage shape comparison between parameterised original, coefficient C2 = 1.645 (red), one with coefficient C2 = 1.24 (white) and one with coefficient C2 = 2.0 (cyan). The C_L for all of these bodies was approximately 0.00012 to 0.00013. The C_D varied from approximately 0.0251 to 0.0267 and the C_M (about the origin which is where the hub is centred) was negligible (of the order of 10^{-5}).*

9.4 JMRTS Optimisation Results

The parameterised body was compared to the original JMRTS fuselage shown previously in Figure 9.3. It shows the C_p distribution of the original and parameterised body along with the experimental data. Both sets of data fit quite close to each other. They also fit the experimental data quite accurately except for the rear part of the fuselage. This is because the isolated fuselage experiments were carried out with the hub grips still on and rotating, whereas the CFD solution was obtained without the hub grips and blade.
The overall drag of the original body was also very close to the parameterised shape as shown in Table 9.3. This suggests that the coefficient of Table 9.2 can accurately represent the JMRTS shape and that the parameterised body can be used in the optimisation process to represent the JAXA JMRTS fuselage.

In addition, to allow for the stagnation points near the front and rear of the fuselage to be somehow closer to reality since an isolated fuselage computation would have no influence from the rotor, an actuator disk (AD) was added. The AD simulates the effect of a rotor by creating a pressure difference on a single plane inside the flow[147] . ADs have been found to give reasonably correct predictions for the effects of rotor flows on fuselages[147] if modelled well. In this case, the employed AD was simply a uniform load distribution and produced a C_T value similar to settings of the JAXA experiments[7] . More details on the AD method can be found in the Technical Notes[148] . The disk was defined by its radius, centre and thickness as well as the C_T and μ of the rotor. The change in dimensionless pressure across the disk is then given by

$$\Delta P = \mu^2 C_T \tag{9.24}$$

The overall effect of the AD was to increase the magnitude of drag as shown in Table 9.3 but this resulted in a more realistic flow-field around the fuselage. Figures 9.7 compares the C_P distribution along the centre of the fuselage with and without the AD. The separation on the upper surface can also be seen in Figure 9.8. Figures 9.9 and 9.10 are qualitative visualisations of how the flow is deflected by the AD and the vortices formed.

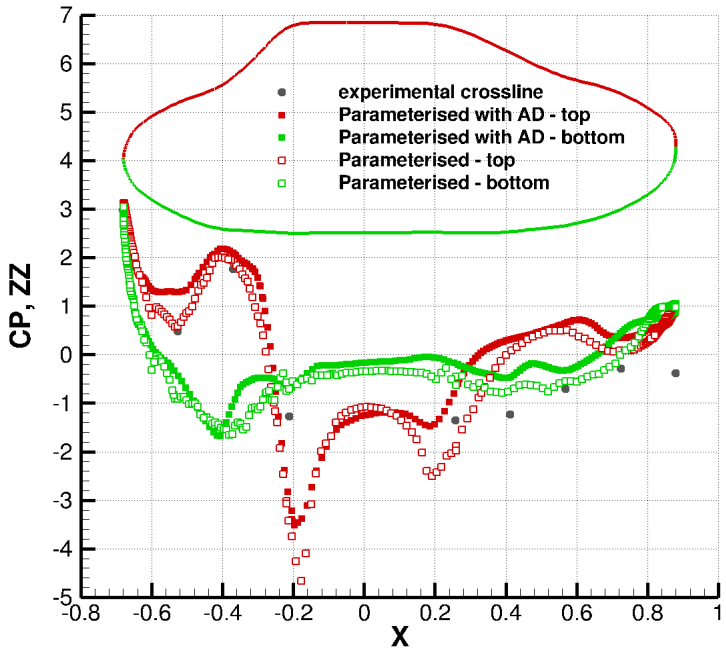
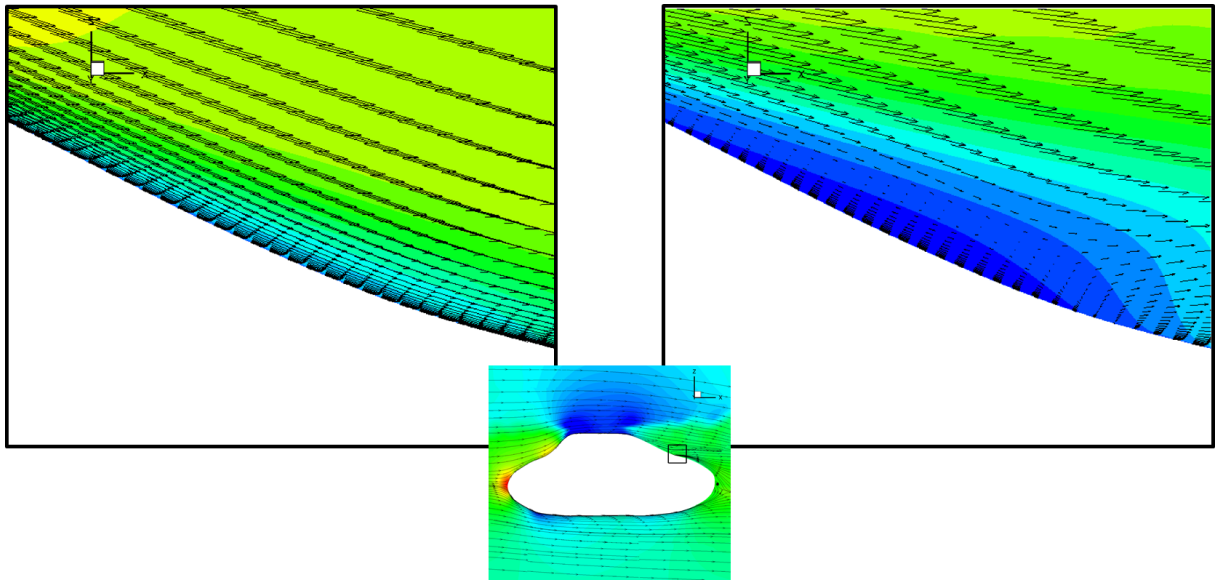
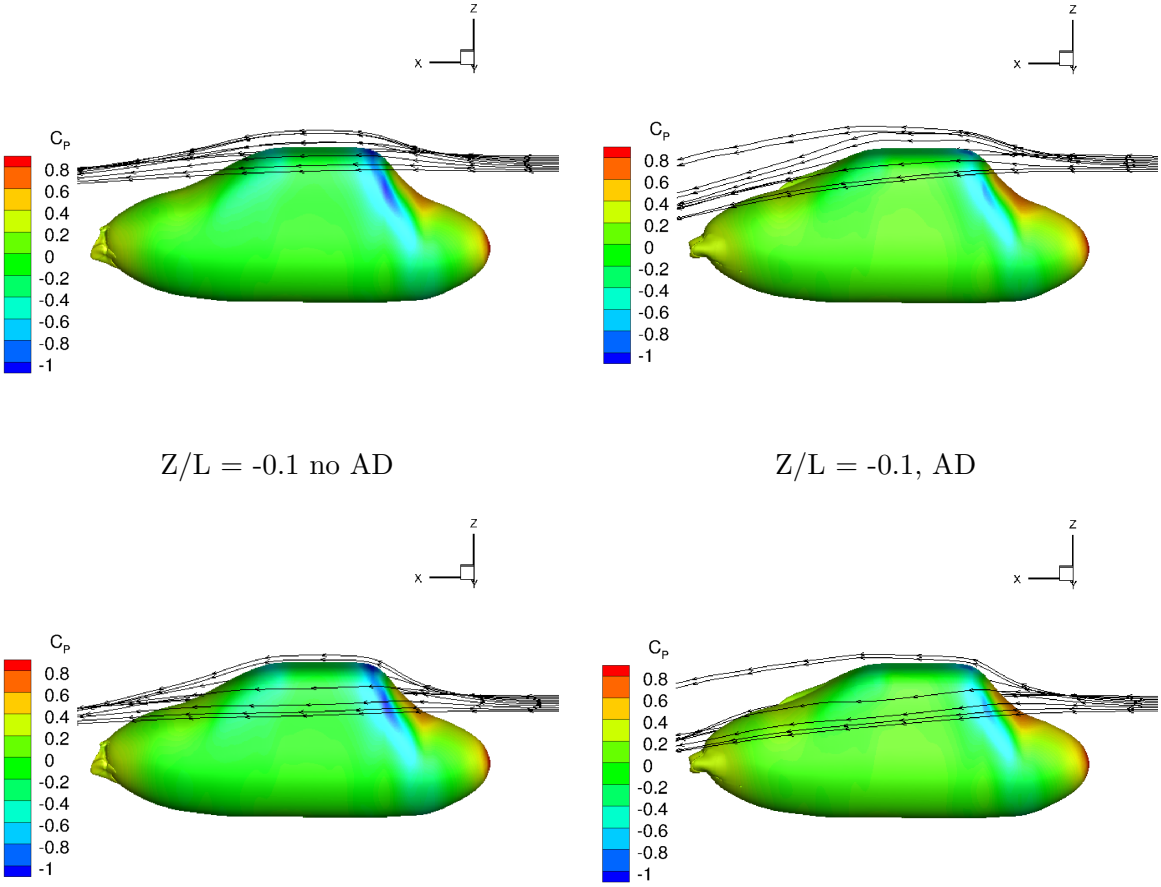


Figure 9.7: *Pressure distribution comparison of the parameterised fuselage with and without the actuator disk.* $C_T = 0.0047$, $\mu = 0.16$.



without AD with AD

Figure 9.8: *Flow visualisation of separation occuring at the rear of the JMRTS fuselage.* $C_T = 0.0047$, $\mu = 0.16$, $M_\infty = 0.175$.



Z/L = -0.1 no AD

Z/L = -0.1, AD

Z/L = -0.2 no AD

Z/L = -0.2, AD

Figure 9.9: *Flow visualisation through the actuator disk. Contour colours show C_P. The actuator disk centre is at $Z = 0$. $C_T = 0.0047$, $\mu = 0.16$*

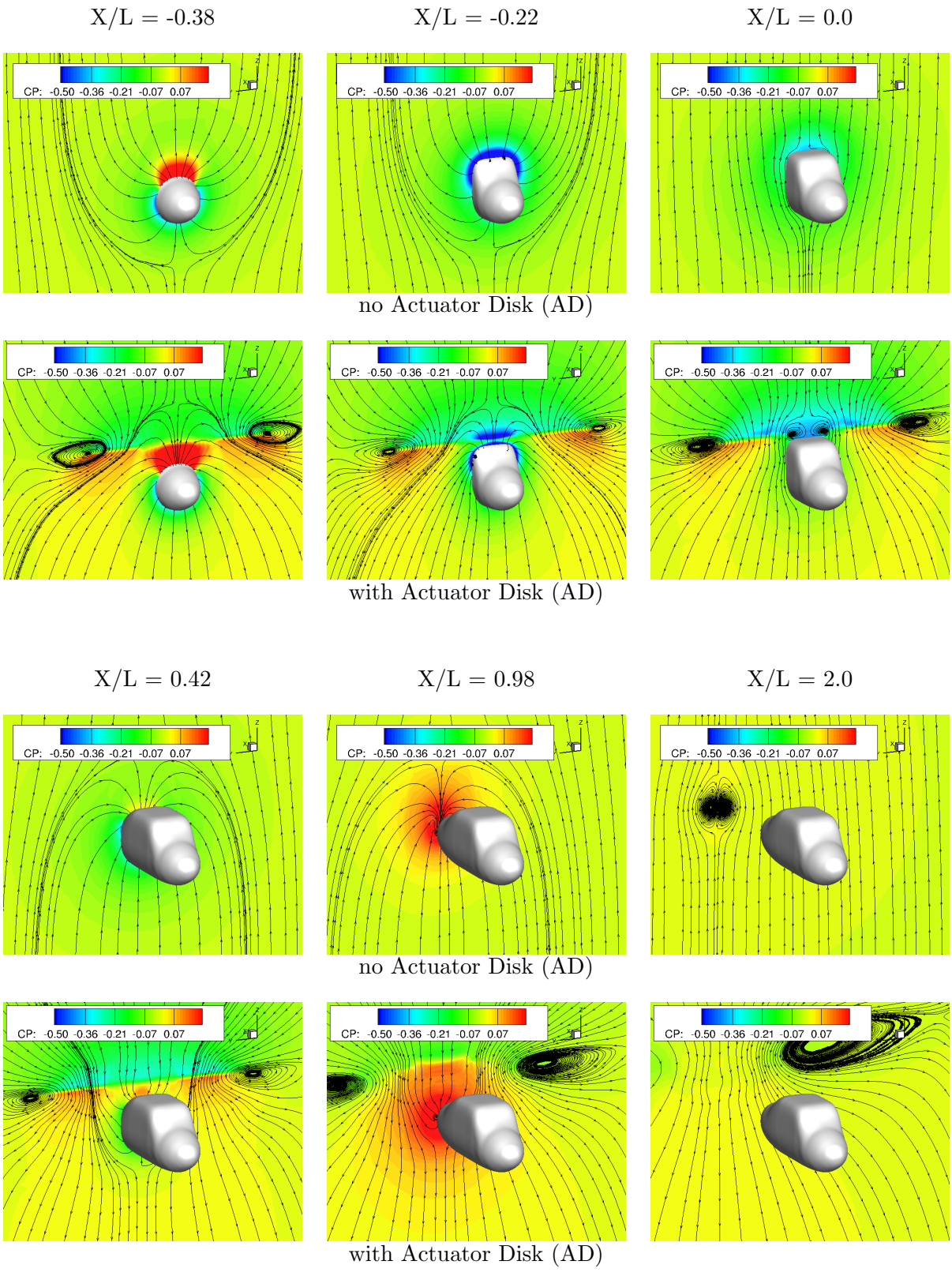


Figure 9.10: *Secondary flow visualisation at sections along the fuselage. Contour colours represent* C_P.

9.5 Drag Optimisation Results

9.5.1 Preliminary Drag Results

To obtain a general idea of which part of the fuselage to optimise, a modified fuselage was created that reduced the gradient of the windscreen area as shown in Figure 9.11. Figure 9.12 also shows the C_p distribution for some stations along the centreline for both fuselages. The pressure slices amplify the differences between the two shapes.

The total drag (pressure and friction) for the front part of the original, parameterised and modified shapes is shown in Table 9.3. As a result of the modification, a slight increase of the volume of the body was also obtained. This increase in volume of the body caused by the change in shape was approximately 0.8% and was combined with a 0.2% decrease in surface area. This shows that significant drag improvements can be made for an almost insignificant change in the total volume of the body.

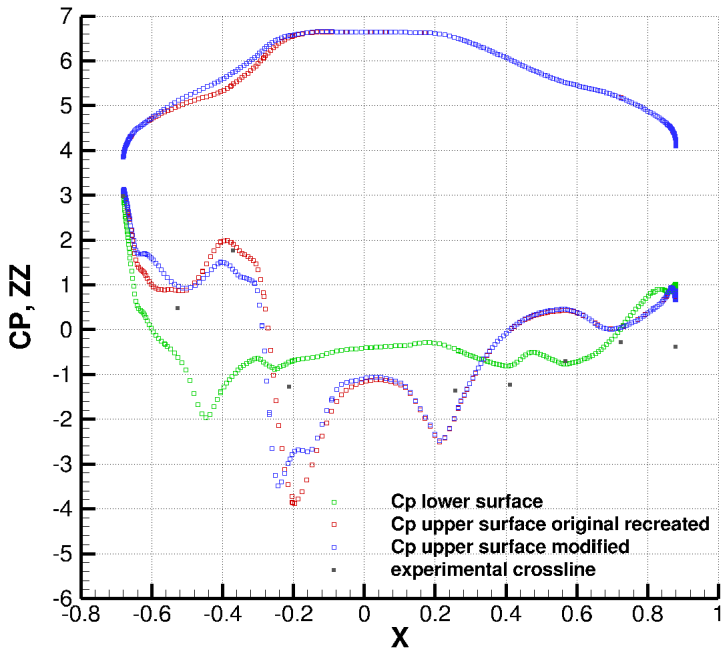
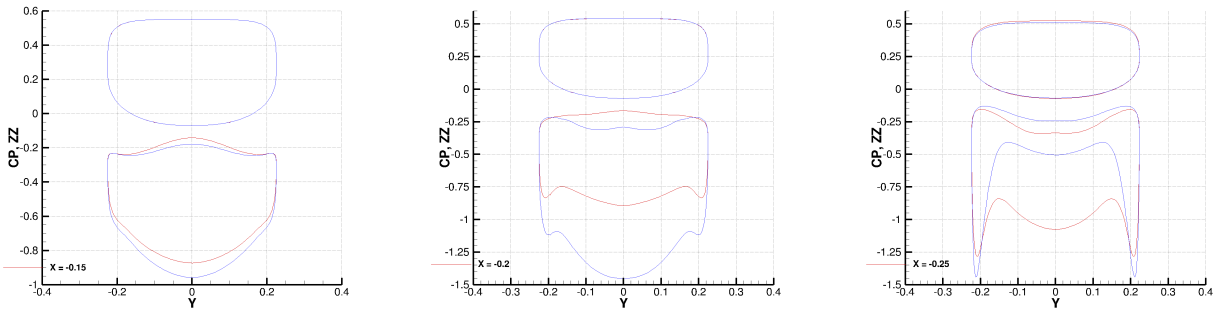


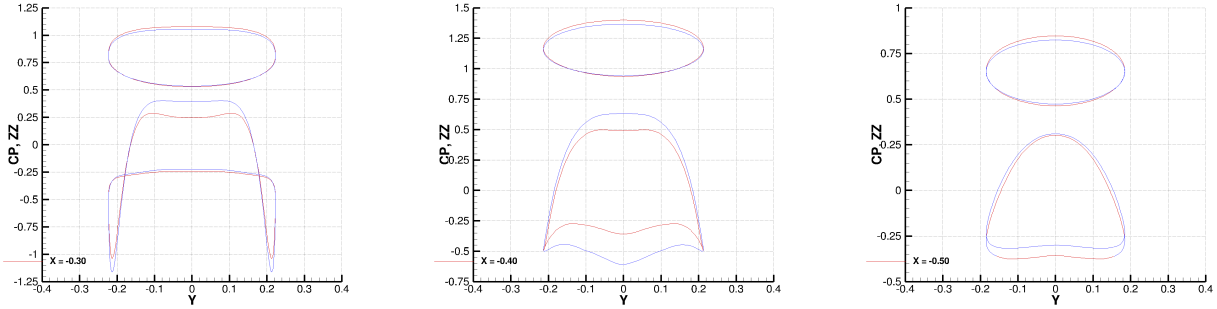
Figure 9.11: *C_p distribution along the centreline for the parameterised and the modified JAXA bodies.*

9.5.2 Optimisation of the Parameters

A number of variations of the JMRTS fuselage were made by modifying the $P4.C2^*$ coefficient of the parameterisation technique. Figure 9.13 shows some of these with their pressure contours. The stagnation point remained constant with a maximum displacement of approximately 4 mm between the two extreme designs. Each of them was computed with an acutator disk having a C_T of 0.0047 and $\mu = 0.16$. For each of these, the drag coefficient at the front of the fuselage i.e. up to and including the windscreen area, was calculated and the data are shown in Figure 9.14. Since there is only one parameter to optimise for, no Pareto front is obtained and the selection is simply the one with the lowest drag i.e. $P4.C2^* = 1.24$. The polar plot of this fuselage is compared to that of the original fuselage i.e. $P4.C2^* = 1.645$ in Figure 9.15. Against both angle of attack and lift coefficient, it can be seen that the optimised fuselage has a lower drag for the same lifting performance.



(X/L = -0.15) (X/L = -0.20) (X/L = -0.25)

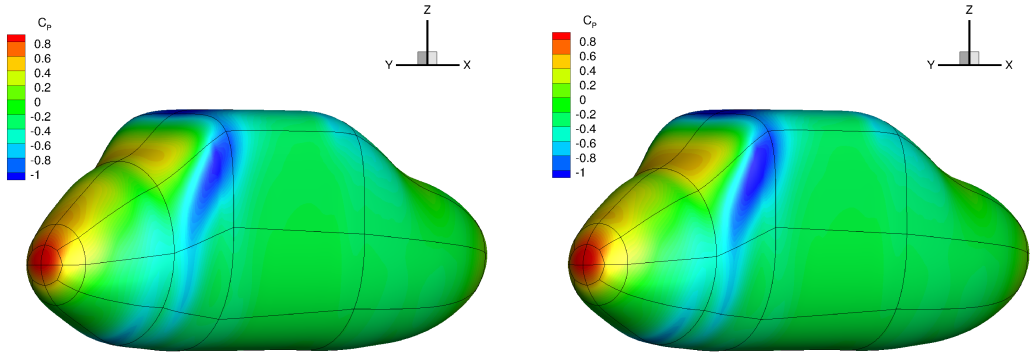


(X/L = -0.30) (X/L = -0.40) (X/L = -0.50)

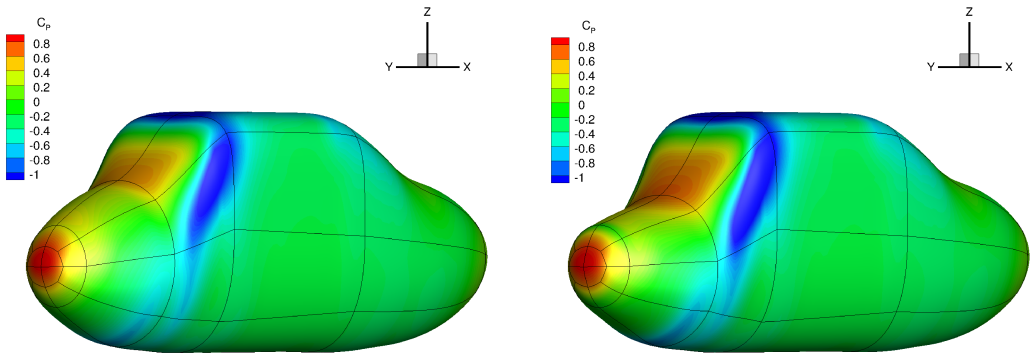
Figure 9.12: *C_p distribution differences along the fuselage of the original parameterised fuselage and the modified one (red is modified and blue is parameterised original).*

Description	C_D	C_L	C_{My}
Original full body (no AD)	0.02206	0.000474	-0.000201
Parameterised full body (no AD)	0.02378	-0.001412	0.000600
Original full body with AD	0.02710	-0.000247	0.000123
Parameterised full body with AD	0.02689	0.000105	0.000194
Original front fuselage	0.005816	$\simeq$0	N/A
Parameterised front fuselage	0.005732	$\simeq$0	N/A
Modified front fuselage	0.004060	$\simeq$0	N/A
Original front fuselage with AD	0.033378	$\simeq$0	N/A
Parameterised front fuselage with AD	0.033129	$\simeq$0	N/A
Modified front fuselage with AD	0.032805	$\simeq$0	N/A

Table 9.3: *Comparison of drag for the original, parameterised and modified JMRTS fuselages. The front is defined as the area containing the front cone and front slope i.e. up to x = -0.2 (or 0 to 0.44 in the parameter table, Table 9.2).*



$P4.C2^* = 1.24$ $P4.C2^* = 1.4$



$P4.C2^* = 1.645$ $P4.C2^* = 2.0$

Figure 9.13: *C_P variation for a selection of parameter values.*

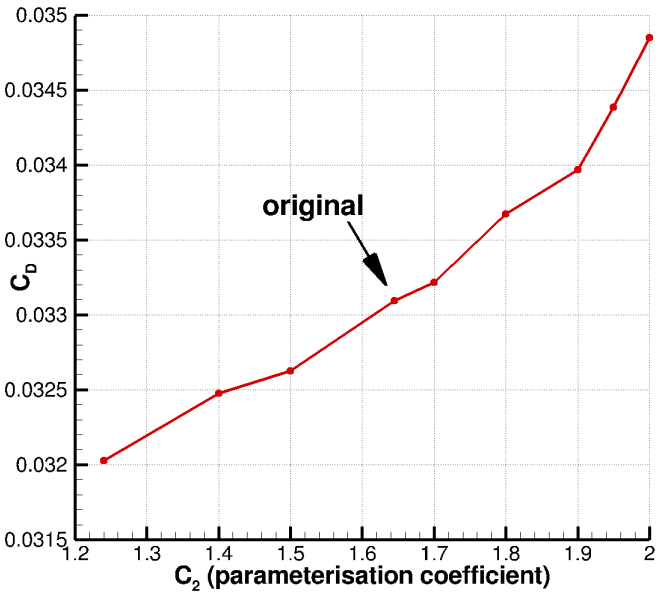


Figure 9.14: *Variation of drag coefficient for the front of the fuselage with change in parameterisation coefficient, C_2. The original body has a parameterisation coefficient of 1.645.*

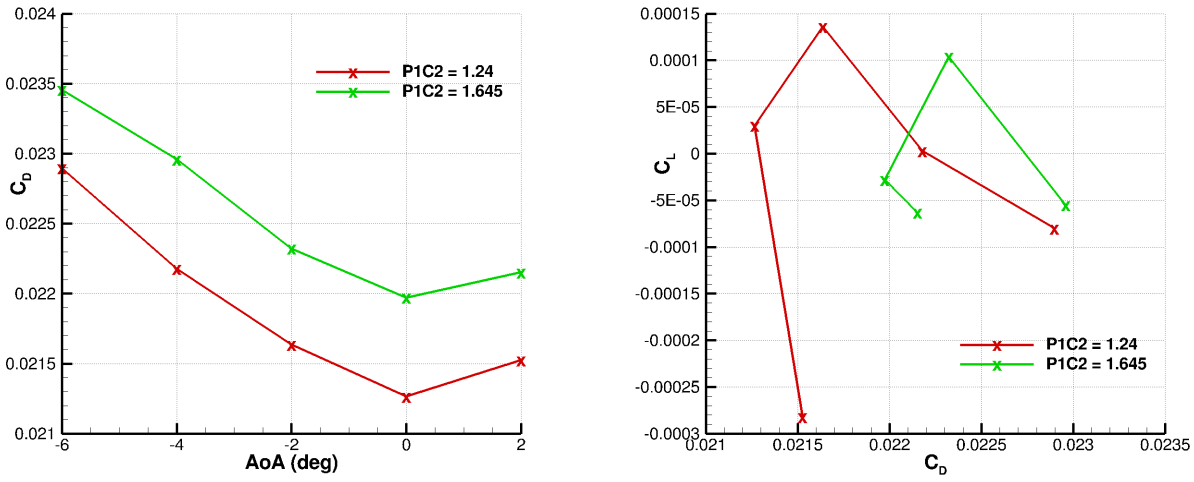


Figure 9.15: *Polar plot of the optimised and original showing Drag vs Angle of attack and Lift vs Drag coefficient. The green curve is for the original and the red for the optimised.*

Chapter 10

BERP-like Tip Optimisation

The BERP tip was designed for high speed forward flight without compromising hover[16] . The aim was to reduce transonic effects like drag rise and to increase rotor thrust by delaying stall using the planform and aerofoil selections[9] . The problems associated with this flight regime, are associated wth the fact that the effects of compressibility such as transonic flow and shockwaves become significant especially on the advancing blade. Typically thin aerofoils are used, but these tend to stall more easily at high angles of attack which occur on the retreating side.

The first step in the design of the BERP was the aerofoil selection. The aerofoils were selected such that thinner sections could be used to enable higher forward flight speeds. Camber was introduced to improve the stall capability of the blade on the retreating side (the aft-loaded RAE 9645)[9] . However, with camber comes increased downward pitching moments alleviated by using an aerofoil that counters the pitching moment inboards i.e. one with reflex (the RAE 9648). This resulting blade behaved well in terms of control requirements and twist loads[149] . The tip aerofoil used was the RAE 9634 which behaves well in transonic conditions.
The planform was then optimised to reduce high Mach number effects by first sweeping the tip of the blade back. This moves the aerodynamic centre of the swept part backwards causing control problems in the pitch axis. To counteract this, the swept part was translated forward which introduces a notch on the leading edge of the blade. The notch corners were smoothed to avoid flow separation. A "delta" tip was also incorporated so that a stable vortex formed at higher angles of attack on the retreating side to delay stall[9] .

Some of the beneficial characteristics of the blade are that the blade stall occurs first inboards of the notch and does not spread outwards. Favourable pressure gradients are observed outboard of the notch[9] . This is because at high angles of attack, such as at the retreating side, the vortex formed travels around the leading edge and the flow over the swept part remains attached and stable at large angles of attack as shown schematically in Figure 10.1. The BERP blade shows similar performance to a standard rotor blade at low speed flight, but superior performance in forward flight due to the absence of drag rise and flow separation on the retreating side due to the improved incidence capability[150] . The strong shock inboard of the notch does not progress outboards as it would normally do, therefore alleviating early shock induced separation even though the twist is high to maintain hover performance[150] . Anhedral in the tip was incorporated because it was found that the moment at the tip about the pitch axis was greatly increased due to a local lift about the tip edge[150] . In hover, the FM was improved due to the minimisation of blade area and overall, there were no penalties in hover performance. In the BERP IV version, the twist was optimised for hover to improve the hover performance without increasing the risk of high vibration[149] .
Further work by Srinivasan et al.,[151] investigated the influence of planform on airloads in hover by comparing a UH60-A, a rectangular UH60-A and a high-twist BERP blade. The BERP produced approximately the same FM with more thrust for the same collective and minimal shock-induced separation was observed in comparison to the UH60-A as well as a more tightly braided stable tip

vortex.

The BERP planform design is restricted, but based on the estimated information given in some of the literature,[152] a BERP-like tip was developed and used. Some additional details on the planform can also be found in Srinivasan et al.[151] such as that the notch starts about 80% of the span. The aim of this optimisation process is to quantify the improvements that a BERP-like tip can have on a typical high speed forward flying rotor. Such rotors tend to have swept tips and thin sections outboards on the rotor. The base rotor selected here is made of two sections, the HH-02 and the NACA 64A-006 at the tip (shown in Figure 10.2) and has a sweep of 20 degrees initiated at r/R = 0.92. It has an AR of approximately 13.7 and linear twist of -9 degrees. The optimisation was carried out primarily for forward flight, although hover conditions was analysed and constrained.

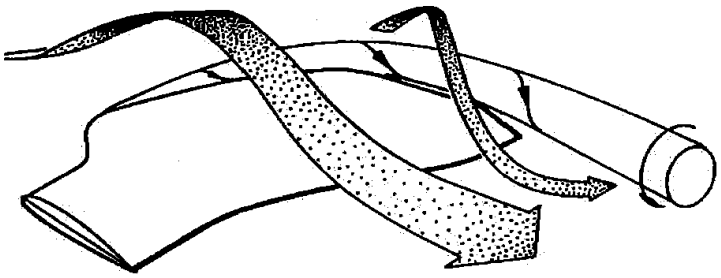
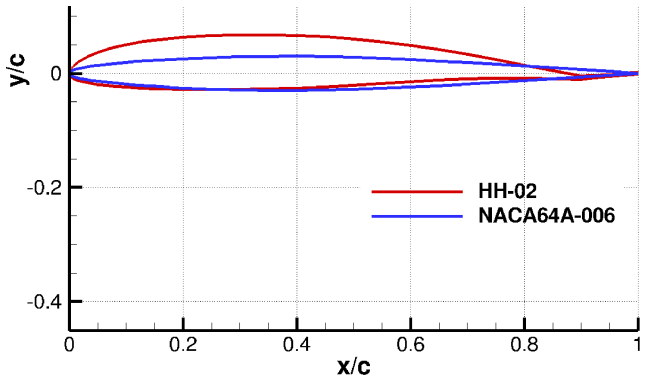
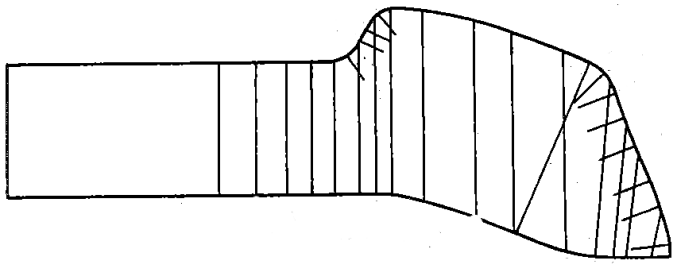


Figure 10.1: *Vortex along the leading edge for the BERP tip*[8] .



(a)

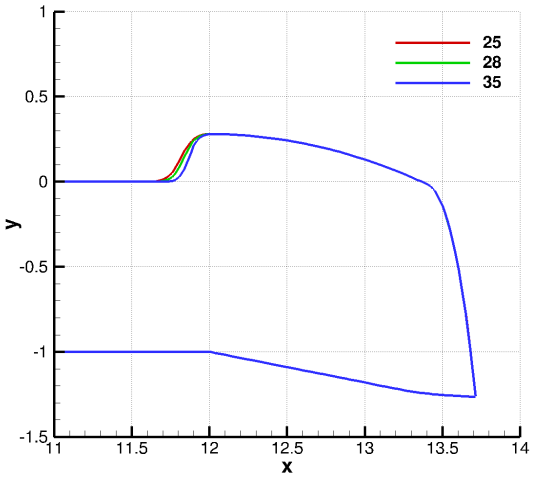


(b)

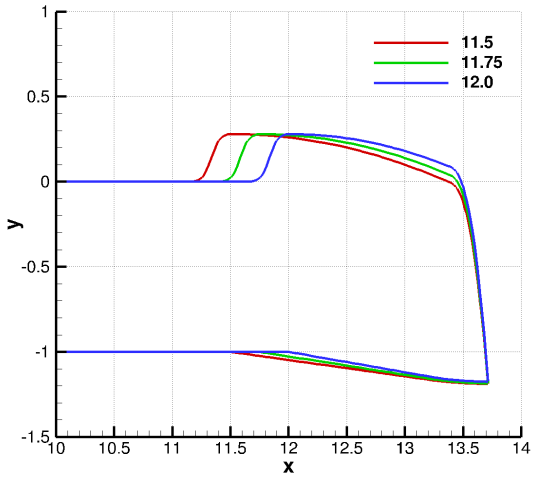
Figure 10.2: *(a) Cross section of the HH-02 and the NACA 64A-006 aerofoils used on this blade. The baseline twist is 9 degrees linear. (b) the original planform of the BERP rotor obtained from Brocklehurst*[9] .

10.1 Parameterisation Technique

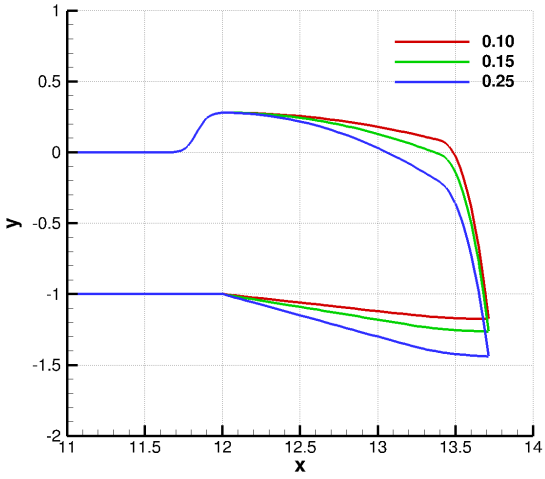
The BERP-like planform parameterisation technique is described in Section 3.2.2. Figure 10.3 shows some example of the three parameters to be varied viz. the notch gradient, the notch position and the sweep of the BERP-like tip.



Varying the gradient of the notch,
sweep = 0.15, notch position = 12



Varying the notch position,
sweep = 0.15, notch gradient = 35



Varying the tip sweep angle,
notch gradient = 25, notch position = 12

Figure 10.3: *Planform view of the parameter changes to the geometry.*

Having three values for each of the three parameters results in a design space containing 27 design points as shown in Figure 10.4. The reason why the design space is not a regular cube shape, is due to the values of sweep. This is because when the notch initiation point is varied, to get the tip point to be in the same place for the highest sweep value for example, the sweep gradient has to be changed accordingly, since it is defined by a parabola. This is shown in Figure 10.5 where the red curve represents a BERP with notch position at 12 and sweep parameter 0.25. To obtain the same sweep with a BERP tip with notch position at 11.5, a sweep parameter of 0.185 must be used instead of 0.25.

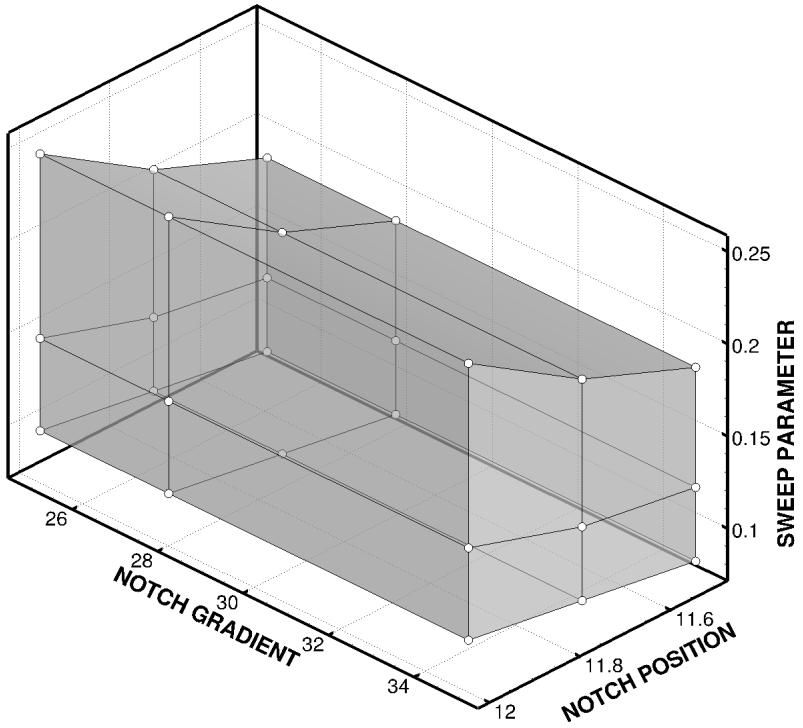


Figure 10.4: *Sample space for the planform of the BERP-like rotor.*

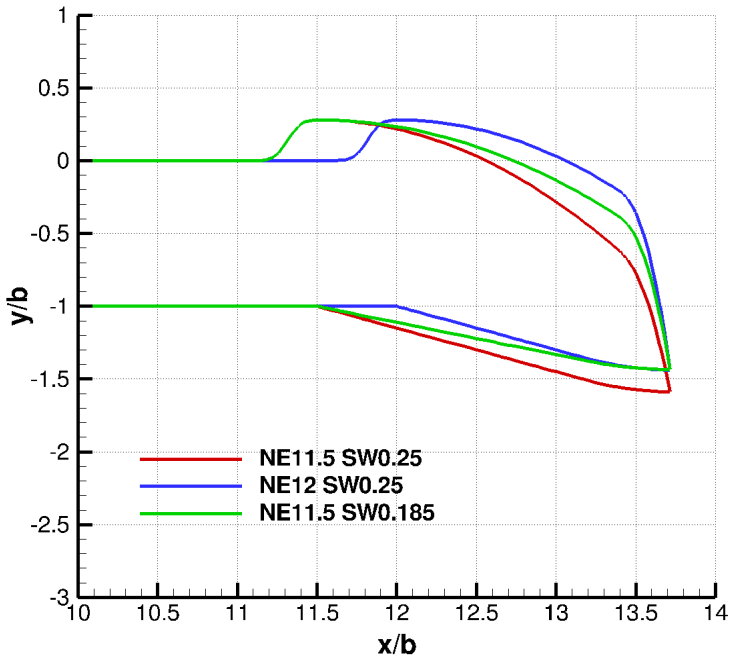
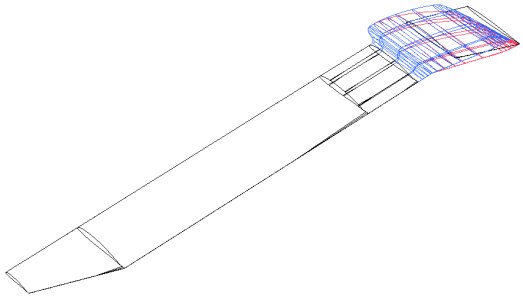
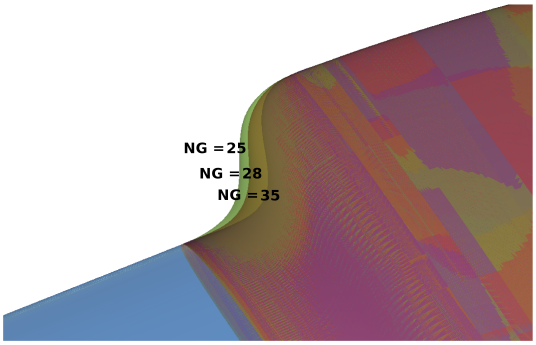


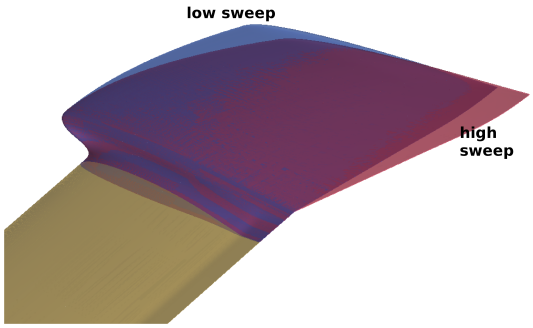
Figure 10.5: *The value of the sweep parameter must be changed to with the notch position for the same sweep.*



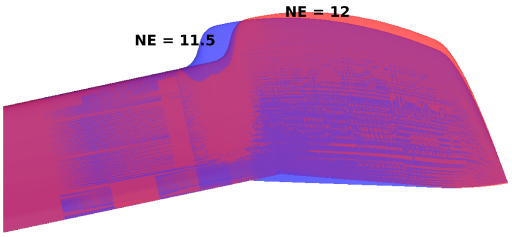
Base rotor with BERP modification



(b) Varying the gradient of the notch



(c) Varying the sweep



(d) Varying the initiation of the notch

Figure 10.6: *Visualisation of the three parameter changes to the blade tip geometry in ICEMCFD.*

10.2 Grid and Geometry Generation

Figure 10.6 shows how a swept rotor tip can be transformed into a BERP-like tip and the effect of varying the BERP parameters. The base rotor is made up of two sections - the HH-02 inboard (up to r/R = 0.92) and the NACA 64A-006 at the tip (r/R = 1). The aerofoil is linearly blended towards this latter section. This rotor has a rectangular tip swept back by 20 degrees, shown in Figure 10.7. When the BERP planform is applied to this tip, the non-linear variation of the chord means that the thickness changes non-linearly as well. To maintain the thickness so that it decreases linearly, from 9.6% (thickness of the HH-02) to 6% (thickness of the NACA 64A-006), sections are cut from the base rotor such that when they are scaled to the chord length required, they have the thickness value that satisfies the linear variation. The problem with this method is that the point of maximum thickness for each of these sections varies non-linearly and therefore the blade surface appears to be bumpy. To overcome this, the notch section is blended from the HH-02 to a NACA-64A section of the appropriate thickness and then the rest of the tip is built with NACA-64A sections of the required thickness so that at the tip, the thickness is 6%.
Another issue that arises is that the HH-02 aerofoil has a tab whilst the NACA 64A-006 does not. So a tab is introduced for the NACA64A and is kept constant till the tip, where the tip is rounded off. The tab is introduced by cutting the aerofoil curves at about 20% from the trailing edge and then rotating the latter part of the curves in the longitudinal axis of the blade so that it adds the required thickness for the tab. The curve is then blended by point, tangent and radius to the main part of the curve to remove any kinks.
Also, the twist is removed from where the BERP tip begins, to avoid having a dihedral trailing edge as shown in Figure 10.8. The trailing edge point is kept at a constant z-value and each section is twisted so that its chord line (not necessarily its quarter chord point) intersects (or its extension intersects) the reference pitch axis. This prevents dihedral from occurring.

A number of ICEM replay files are used to automate these steps, but some manual intervention is required in the creation of these grids. The scripts and their usage can be found in the HMB Technical Note, TN10-BERPGrid[153] .

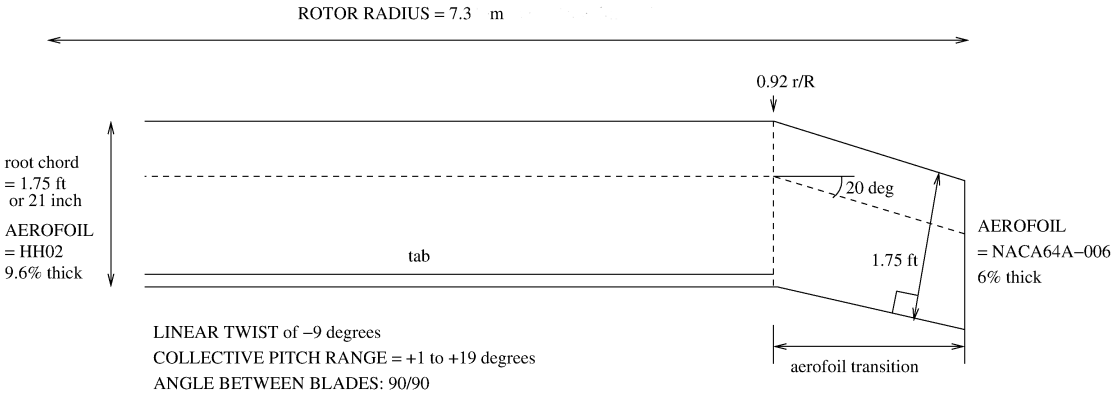


Figure 10.7: *Schematic of the original fast flying rotor blade used as an initial point for this optimisation exercise.*

10.2.1 Anhedral

The tip anhedral is implemented as follows. Let us assume that 10 degrees of anhedral is to be implemented starting from the station r/R = 0.946 to the tip. Then, for each station in between these two stations (r/R = 0.946 and r/R = 1.0), there will be a Δz distance by which that section should be translated downwards (Figure 10.9) which is found as:

$$\Delta z = \Delta x tan\theta \tag{10.1}$$

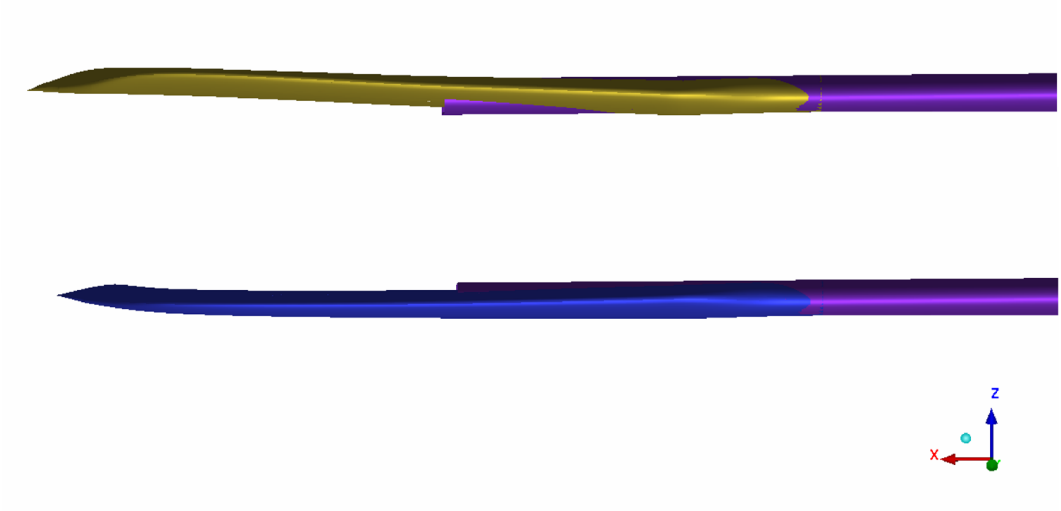


Figure 10.8: *Effect of twisting about the quarter chord point (top) or about the quarter chord line (bottom).*

Once each station has been translated, then the stations are joined by curves and surfaces are created.

10.3 Flight Conditions

The hover flight conditions for the blade were based on a rotor tip Mach number of 0.65 which was assumed from a tip speed of approximately 220m/s at ISA sea level conditions. The chord of the blade was 1.75 ft, therefore the Reynolds number was approximately 8 million. The weight of the aircraft was approximated by assuming a pressure altitude of 0 ft, a free-air temperature of 20° C, a wheel-height of 80 ft to ensure out-of-grounds operations and a torque factor of 1 with 100% rpm. The weight capability at these conditions is approximately 20,000 lb. This results in a C_T of 0.018 based on a rotor radius of 24 ft.

$$C_T = \frac{T}{1/2\rho A(\Omega R)^2} = \frac{9000 \times 9.81}{0.5 \times 1.225 \times \pi\,(7.3^2) \times 220^2} = 0.018 \tag{10.2}$$

For forward flight, the conditions are for high speed at reasonable thrust level. Therefore, the advance ratio, $\mu = 0.34$, $C_T = 0.0122$. This was obtained from typical maximum speed, minimum thrust for that speed based on empty weight + 20% or about 6.2 tonnes.

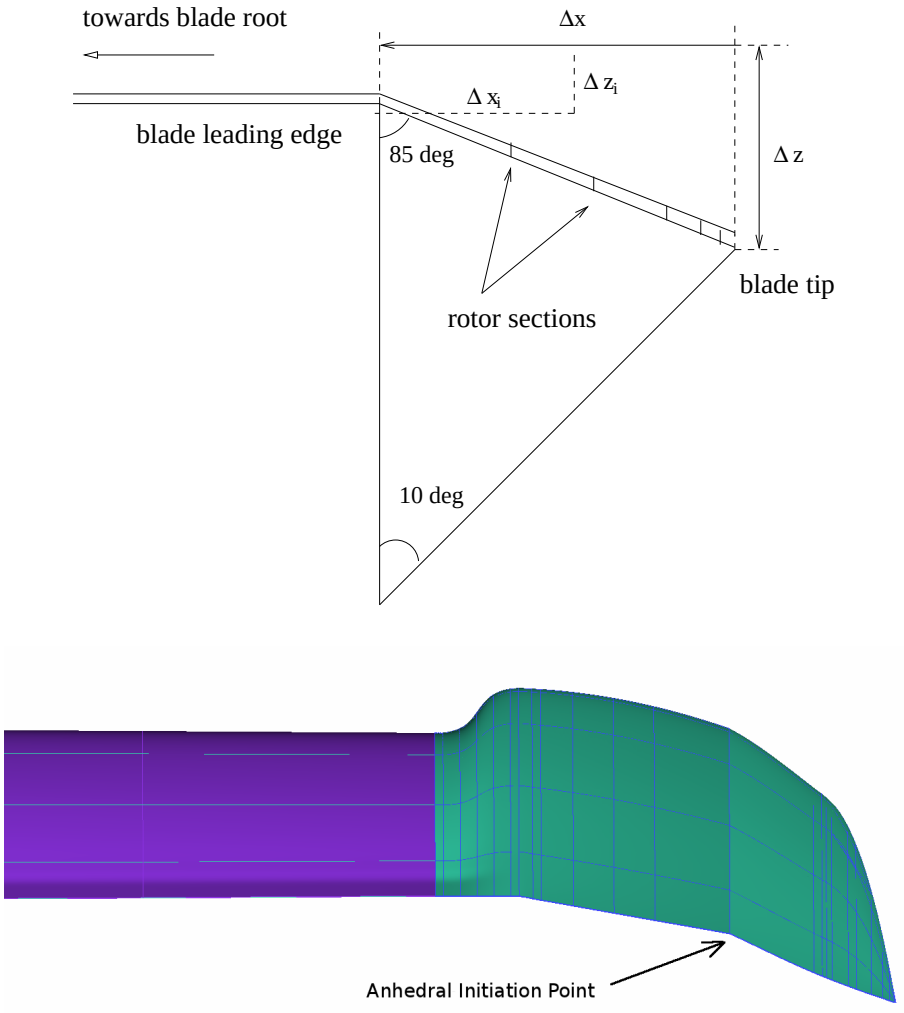


Figure 10.9: *Anhedral definitions for the BERP tip.*

10.4 Hover Results

The objective for this case, was to obtain a high FM over a wide thrust setting. Therefore the FM over a range of thrusts was obtained for the original rotor and a BERP variant. Figure 10.10 shows the comparison of these blades for increasing C_T/σ. The BERP rotor with the same twist and anhedral has a better FM at low thrust, but at higher thrust values, its performance drops to below that of the original rotor. With a higher twist, some of the performance of the original rotor is redeemed and with an anhedral of 20 degrees implemented, an improved rotor in hover is obtained.
The reason for this performance trend is that for the BERP rotor, the loading of the blade increases steeply where the BERP section starts as can be seen in Figure 10.11. Also the loading inboards is lower than the original rotor. With increasing twist, the inboard loading is increased which improves the FM. The anhedral reduces the outboard loading and thus the performance of the BERP blade matches the original rotor blade.

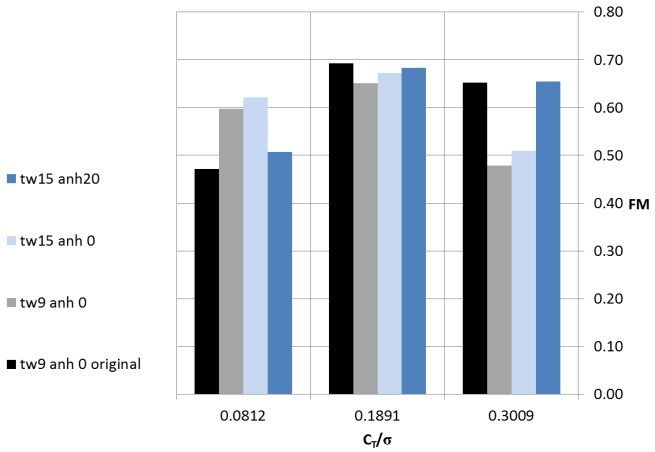


Figure 10.10: *Figure of Merit vs. thrust coefficient of the original blade and the BERP variants with varying twist and anhedral.*

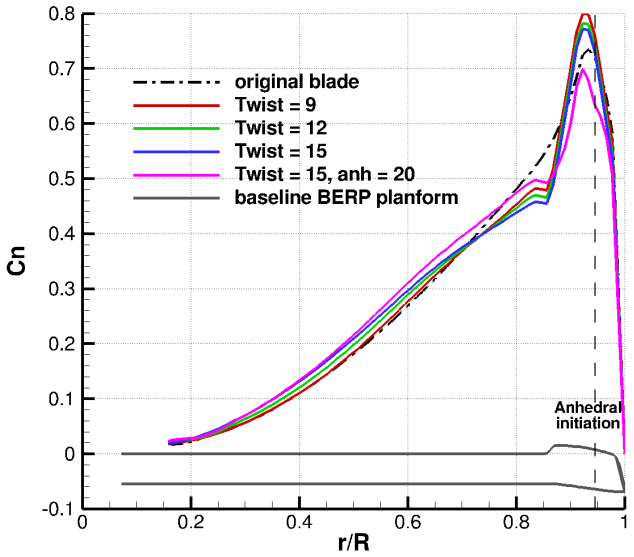
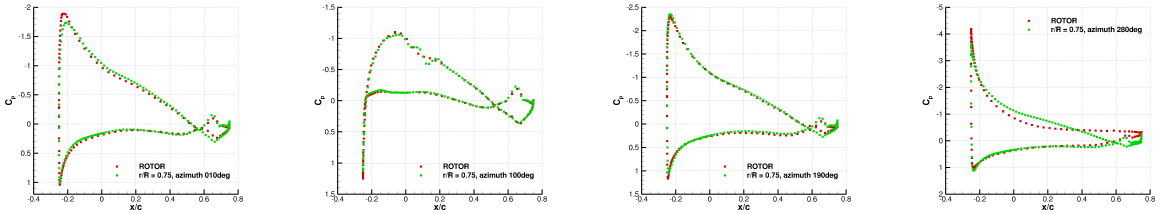


Figure 10.11: *Lift distribution along the span with varying twist at 13 degrees of collective.*

10.5 Forward Flight Results

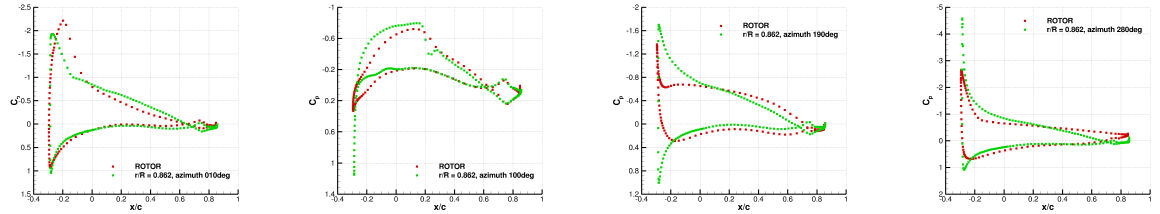
The original BERP rotor used aerofoils specifically designed for its overall performance[9] . However, the optimisation here only changes the planform while using generic aerofoil sections. To compare the effect of the aerofoil on the planform, steady flow 2D results were obtained at a number of sections and azimuths and compared to the flow of the section on the blade at the same conditions. This was done by obtaining the section at the rotor and running it as a steady 2D aerofoil to obtain the same lift coefficient at the same Mach and Reynolds conditions experienced on the rotor. In this way the downwash angle can be estimated as well even with some caution since 3D effects are not included.

Figure 10.12 shows the comparison of the aerofoil with the rotor section at r/R = 0.75 at 4 azimuth angles. It can be seen that the 3-dimensional effects on the rotor section play an important role in reducing the geometric angle via the downwash increment. The aerofoils also have large pitching moments especially more outboards on the retreating side as shown in Figure 10.13 and 10.14. This shows the importance of selecting good aerofoils for a BERP-like rotor and hence why much effort was put in to selecting the RAE aerofoils for the actual BERP rotor. A dM/dt optimisation similar to the one discussed in[154] prior to the planform optimisation would have likely produced a better rotor. However, regardless of the rotor sections, these aerofoils have been used in rotorcraft and this case only serves to highlight the planform optimisation. The high suction peaks of Figures 10.13 and 10.14 at 190 and 280 degrees of azimuth show the limitation of the employed aerofoils since they are pushed to high lift and high suction peaks.



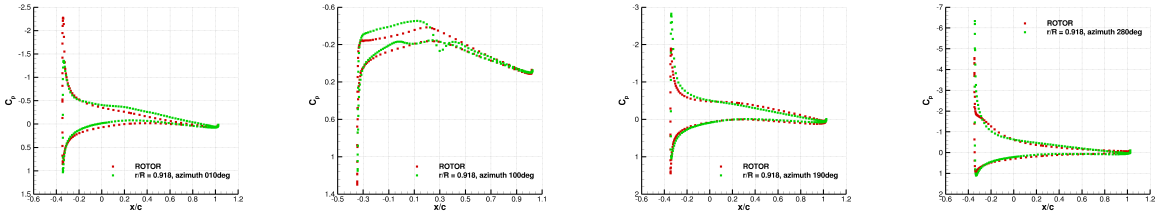
Azimuth = 10 deg | Azimuth = 100 deg | Azimuth = 190 deg | Azimuth = 280 deg

Figure 10.12: *Aerofoil comparison to section on BERP-like rotor at the same conditions before the notch, green is the 2D aerofoil and red is the rotor section, r/R = 0.75.*



Azimuth = 10 deg | Azimuth = 100 deg | Azimuth = 190 deg | Azimuth = 280 deg

Figure 10.13: *Aerofoil comparison to section on BERP-like rotor at the same conditions in the middle of the notch, green is the 2D aerofoil and red is the rotor section, r/R = 0.862.*



Azimuth = 10 deg | Azimuth = 100 deg | Azimuth = 190 deg | Azimuth = 280 deg

Figure 10.14: *Aerofoil comparison to section on BERP-like rotor at the same conditions after the notch, green is the 2D aerofoil and red is the rotor section, r/R = 0.918.*

r/R = 0.750			
Azimuth (deg)	rotor geometric angle (deg), A	2D incidence for same C_n (deg), B	"downwash" (deg), A - B
10	13.8571	5.40	8.46
100	15.6783	3.07	15.61
190	12.1430	7.00	5.14
280	17.0265	8.80	8.23
r/R = 0.862			
Azimuth (deg)	rotor (deg)	aerofoil (deg)	downwash (deg)
10	22.6805	4.80	17.88
100	14.6000	1.03	13.57
190	10.3370	4.30	6.04
280	15.8830	6.80	9.08
r/R = 0.918			
Azimuth (deg)	rotor (deg)	aerofoil (deg)	downwash (deg)
10	10.108	2.30	7.81
100	9.1738	0.55	8.62
190	8.3903	3.80	4.59
280	10.916	6.15	4.77

Table 10.1: *Table showing the effective downwash angle at 3 stations on the BERP-like blade; before the notch, in the middle of the notch and after the notch. Figures 10.12 to 10.14 correspond to this data. Rotor stands for the rotor section pitch angle (A), aerofoil for the equivalent C_n aerofoil angle (B) and "downwash" is the difference between A and B.*

10.5.1 Tip Sweep Effects

Table 10.2 shows the effect of sweep on the performance parameters of the BERP rotor. There is considerable loss in thrust with increased sweep. Therefore, for all cases shown, the rotor was trimmed to give the same thrust of approximately $C_T/\sigma = 0.09$. These results comparing the effect of sweep are shown in Figures 10.15 - 10.17 for the results with the highest notch gradient and the most inboard and outboard notch positions. With more sweep, it can be seen that the distribution of the lifting load is reduced at the back and outboards on the advancing side, and is increased at the front of the disk and more inboards on the advancing side. The load is also distributed more inboards in the spanwise direction at the notch for the blade with a more inboard notch.

The pitching moment is mostly negative on the advancing side and mostly positive on the retreating side. With increased sweep, the magnitude of the moment increases (Figure 10.18) since the moment is calculated about the pitch axis. Also as with M^2C_n, a more inboard notch spreads the moment peaks out.

The torque distribution shows a drop in C_Q where the notch of blade is. C_Q is highest at the back of the disk and lowest at the tip in the advancing side. These extremes increase in magnitude with more sweep. Overall, on the advancing side, the torque reduces with increased sweep and on the retreating side reaches maximum value. Figure 10.18 shows a 3D view of the load distributions comparing low and highly swept blade tips for the most inboard and outboard notch BERP-like tips. The torque distribution shows that the effect of sweep is increased when the notch is more outboard. The advantage of high sweep on the advancing side and low sweep on the retreating side is also shown. The moment distribution has similarities to the torque distribution and its magnitude is much higher at the front and back for the more swept tips. The lifting load distributions differ much less than the other performance parameters.

Figure 10.19 compares the blade loads at 4 azimuth positions and shows the effect of sweep at each location. As can be seen, high sweep offloads the tip at the back of the disk and increases

it at the front. On the advancing side, lift is maintained till just after the notch, where the highly swept blade loses lift quickly. The torque is low for higher sweep for most of the blade. At the tip, after the notch region, however, it increases rapidly. The pitching moment follows a similar pattern although the sweep does not have much of an effect inboards.

NE	NG	SWEEP	C_T/σ	C_Q	avg M^2C_M	ΔM^2C_M
After trimming						
11.75	28	0.09	0.0905	0.000192	-0.002527	0.010297
11.75	28	0.13	0.0900	0.000191	-0.001640	0.010290
11.75	28	0.21	0.0899	0.000186	-0.000197	0.011147

Table 10.2: *Sweep effects on performance comparison. NE is the notch position parameter and NG is the notch gradient parameter. avg M^2C_M is over one revolution and ΔM^2C_M is the peak-to-peak amplitude over one revolution.*

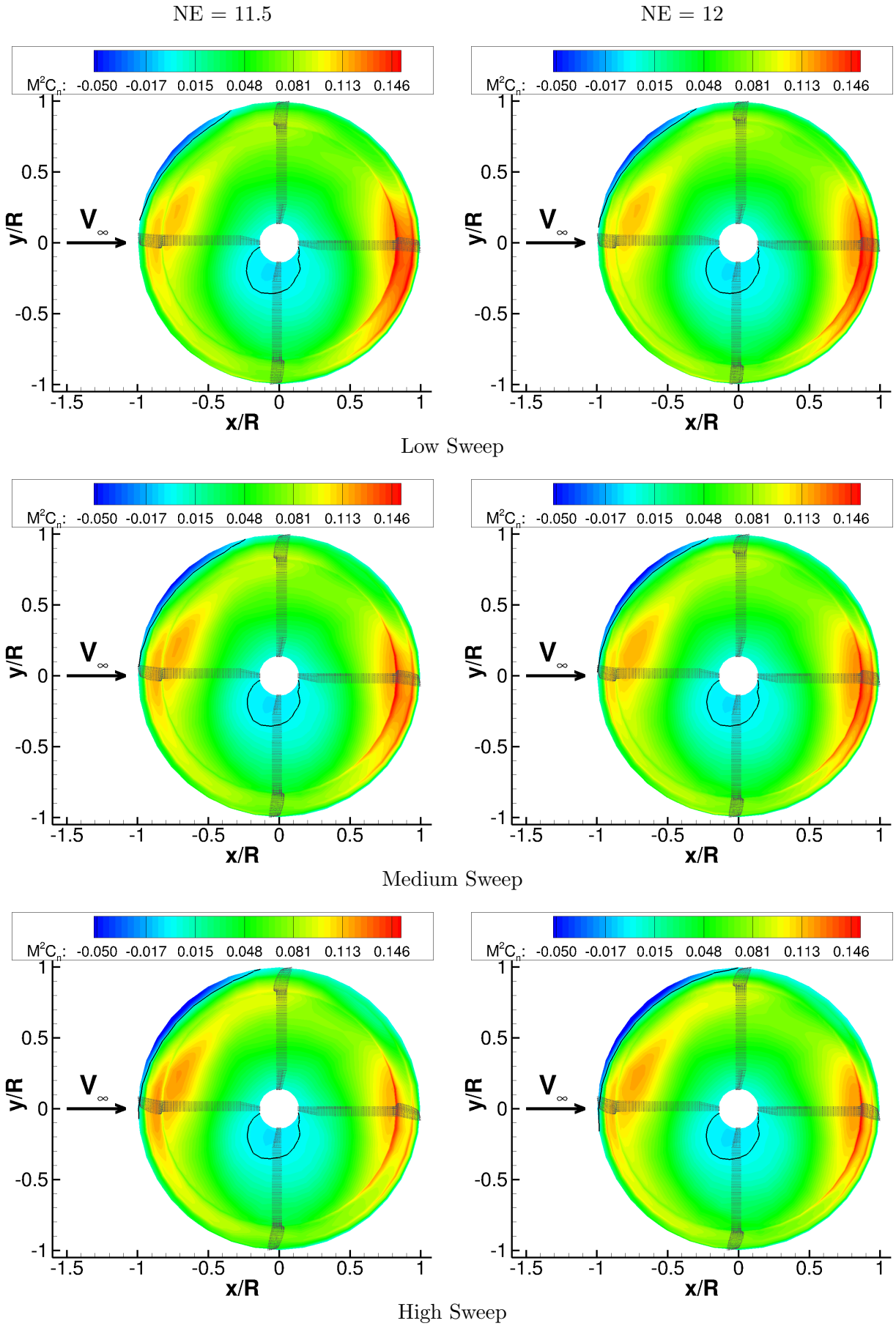


Figure 10.15: *M^2C_n for the BERP-like rotors with fixed parameters: NE = 11.5 and NE = 12, NG = 35 and variable sweep parameters. The black line indicates the $M^2C_n = 0$ line.*

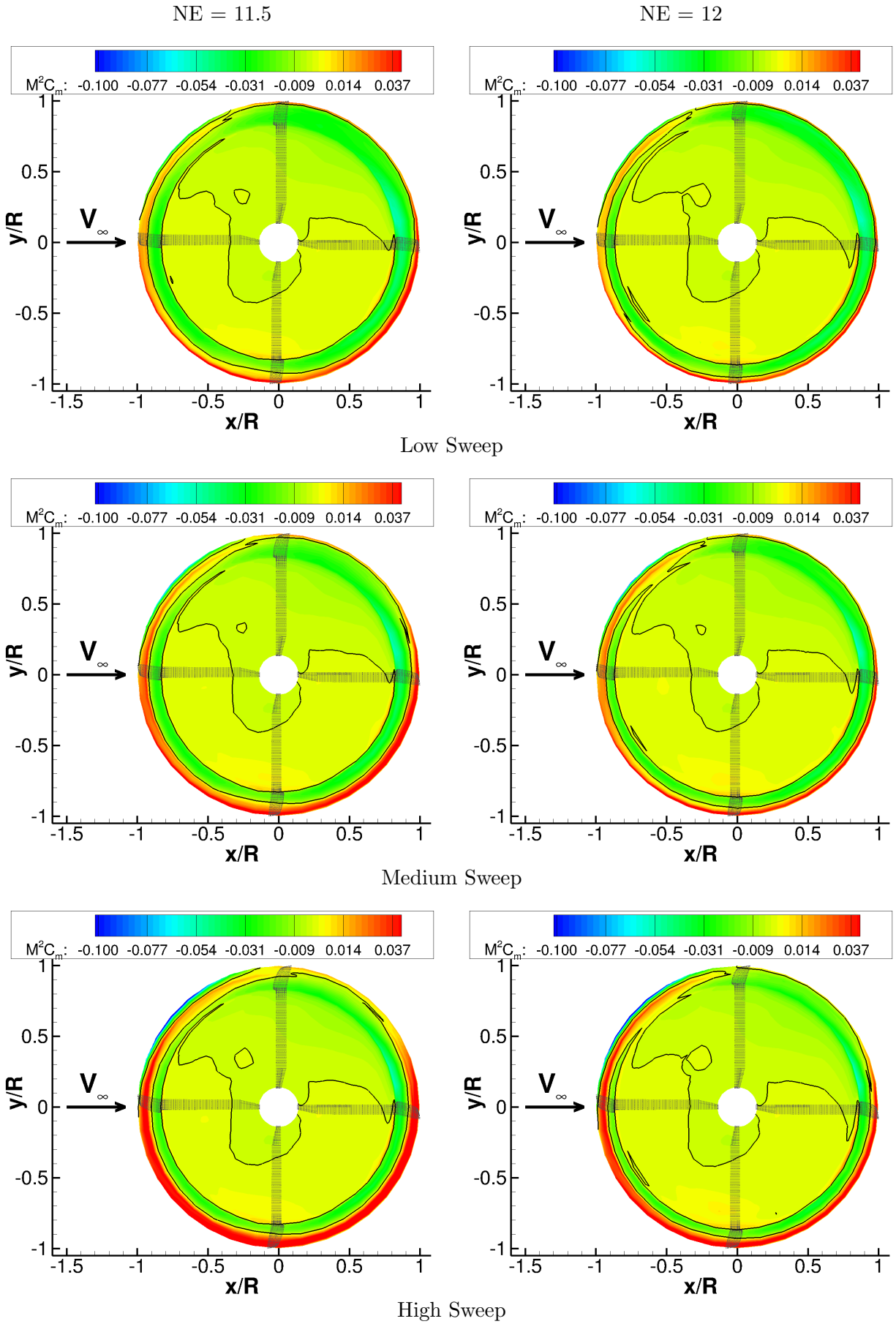


Figure 10.16: *M^2C_m for the BERP-like rotors with fixed parameters: NE = 11.5 and NE = 12, NG = 35 and variable sweep parameters. The black line indicates the $M^2C_m = 0$ line.*

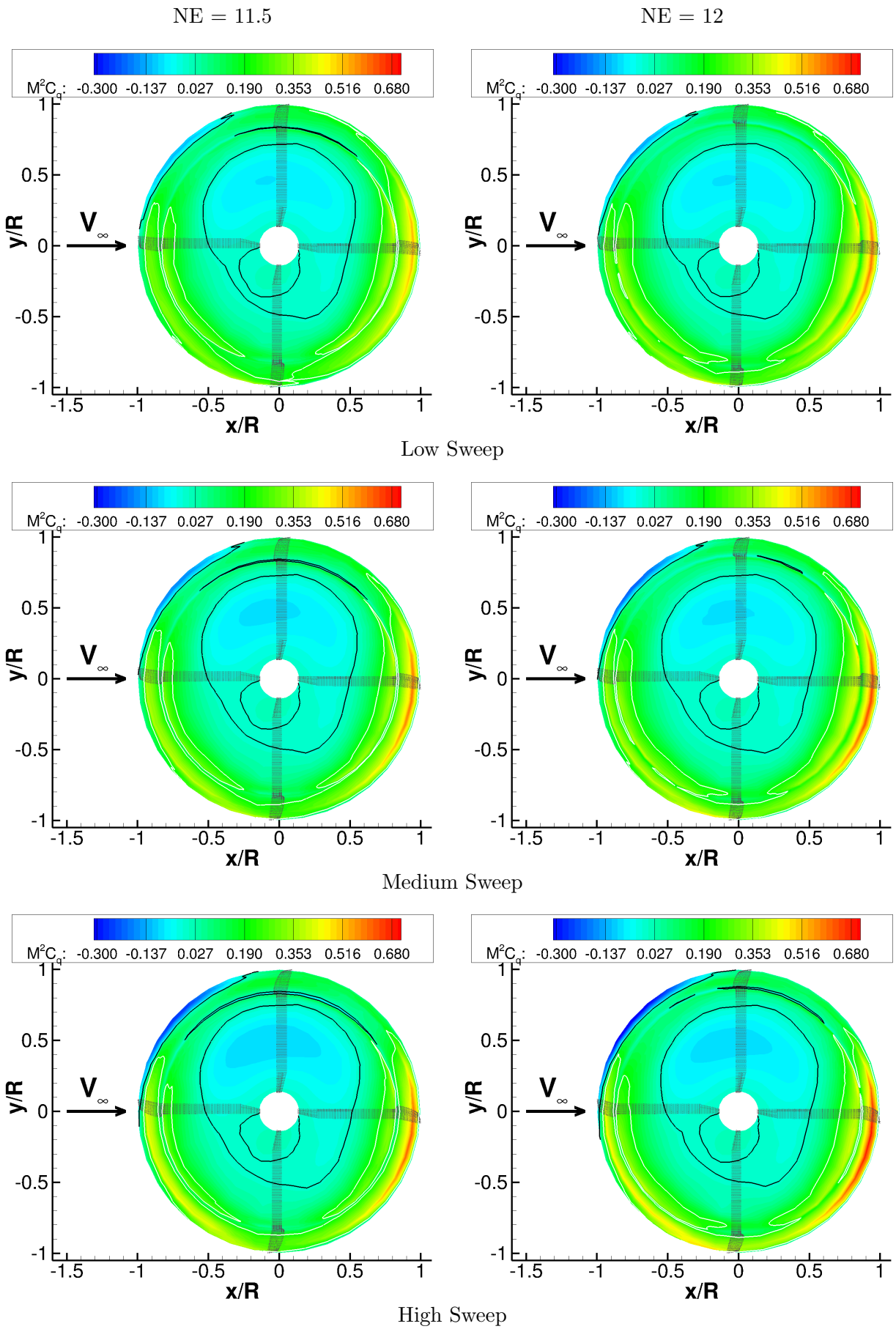
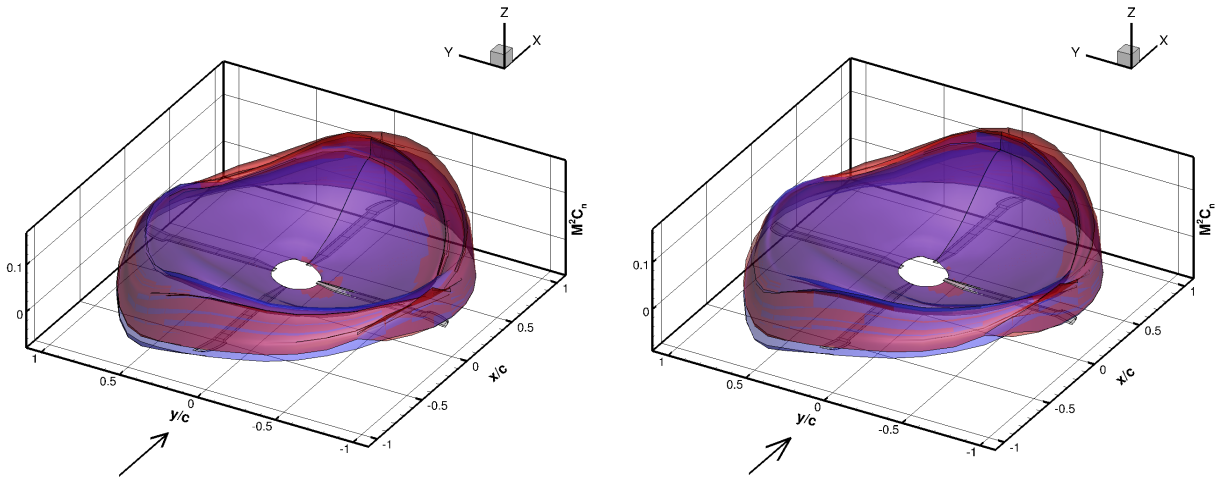
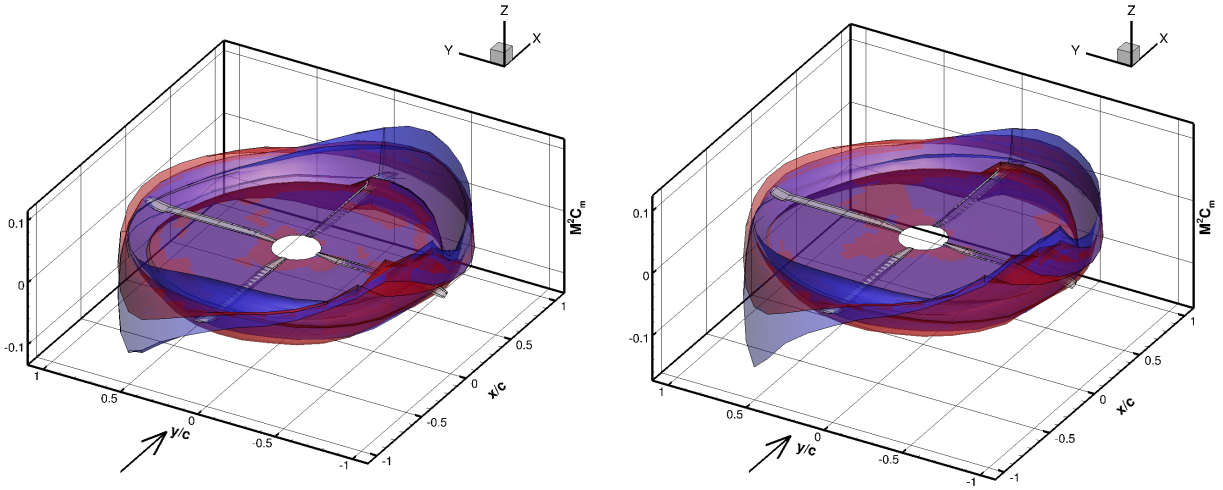


Figure 10.17: *M^2C_q for the BERP-like rotors with fixed parameters: NE = 11.5 and NE = 12, NG = 35 and variable sweep parameters. The black line indicates the $M^2C_q = 0$ line and the white line indicates the approximate middle value, $M^2C_q = 0.2$.*

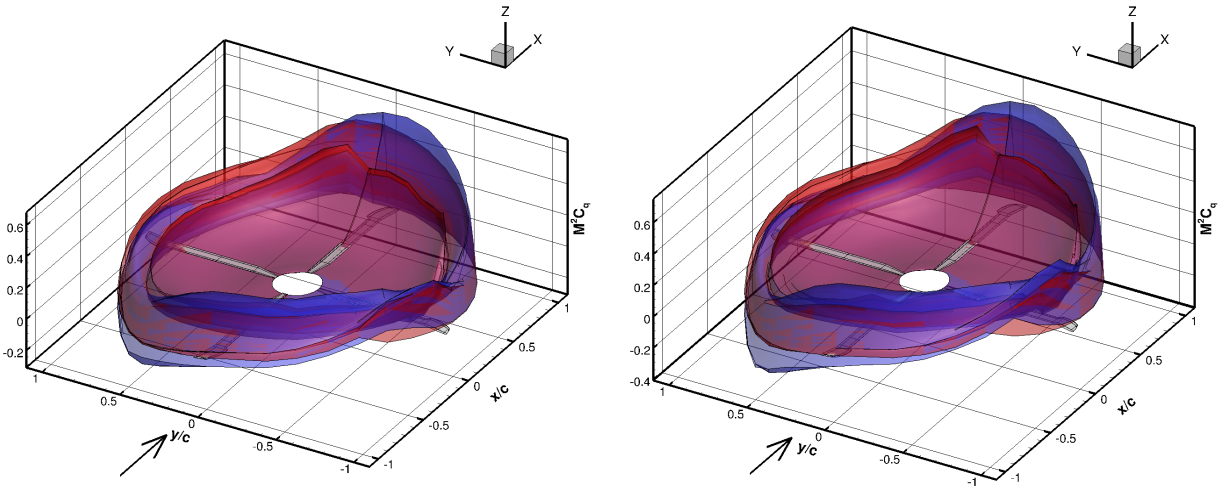
NE = 11.5 NE = 12



M^2C_n



M^2C_m



M^2C_q

Figure 10.18: *M^2C_n, M^2C_m and M^2C_q for the BERP-like rotors with fixed parameters: NG = 28 and variable sweep parameters for different notch positions. Red is lowest sweep, blue is highest sweep. The arrow shows the freestream direction.*

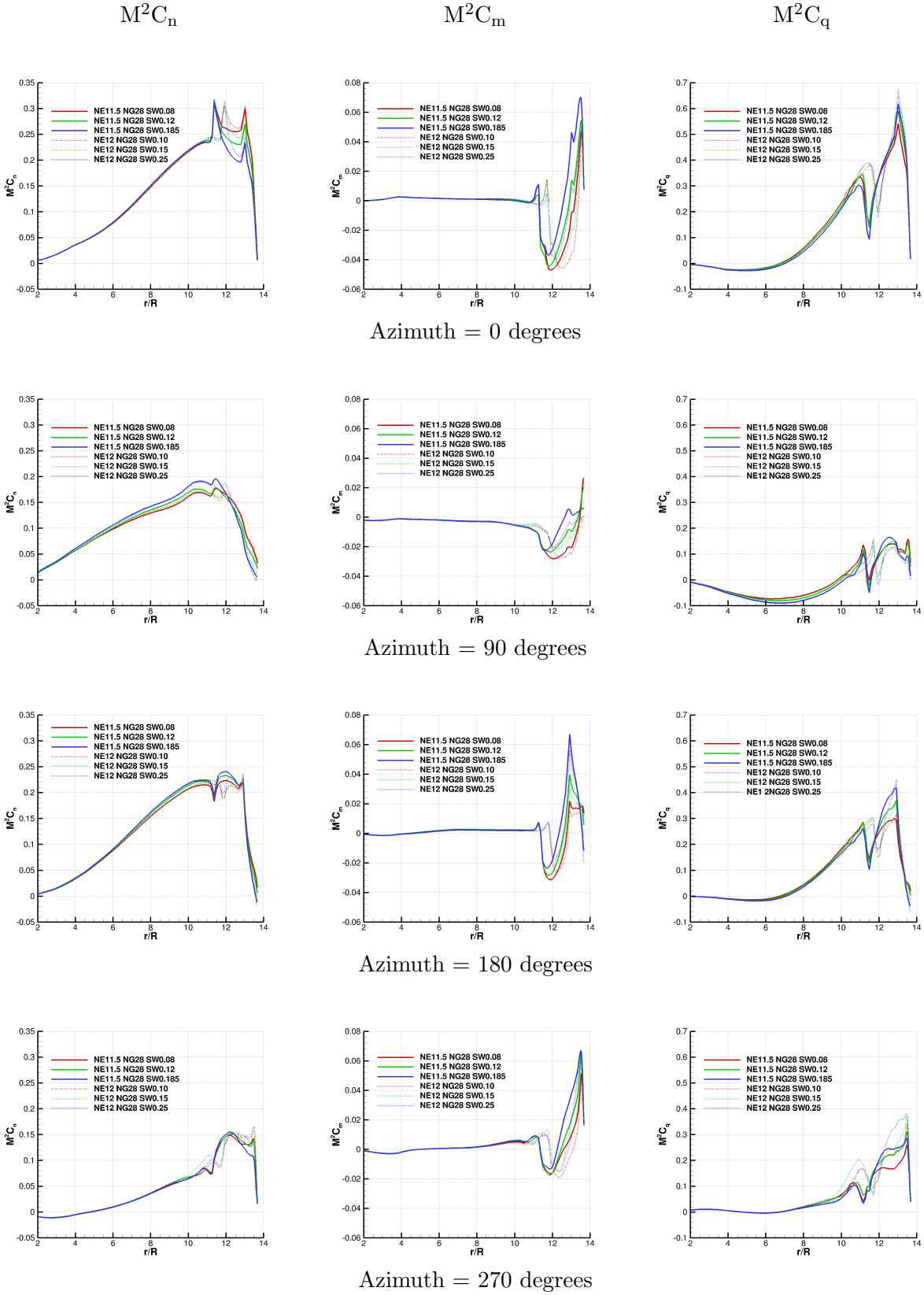


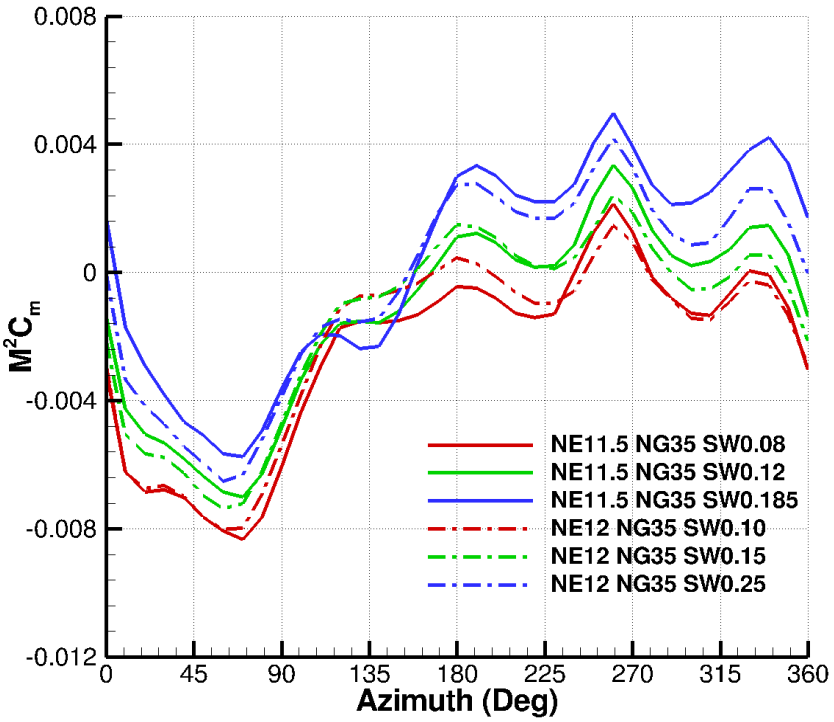
Figure 10.19: *Comparison at azimuth 0, 90, 180 and 270 degrees of the M^2C_n, M^2C_m and M^2C_q for BERP-like rotor with fixed parameters: NE = 11.5 and 12, NG = 28 and variable sweep parameters.*

10.5.2 Effect of Notch Offset

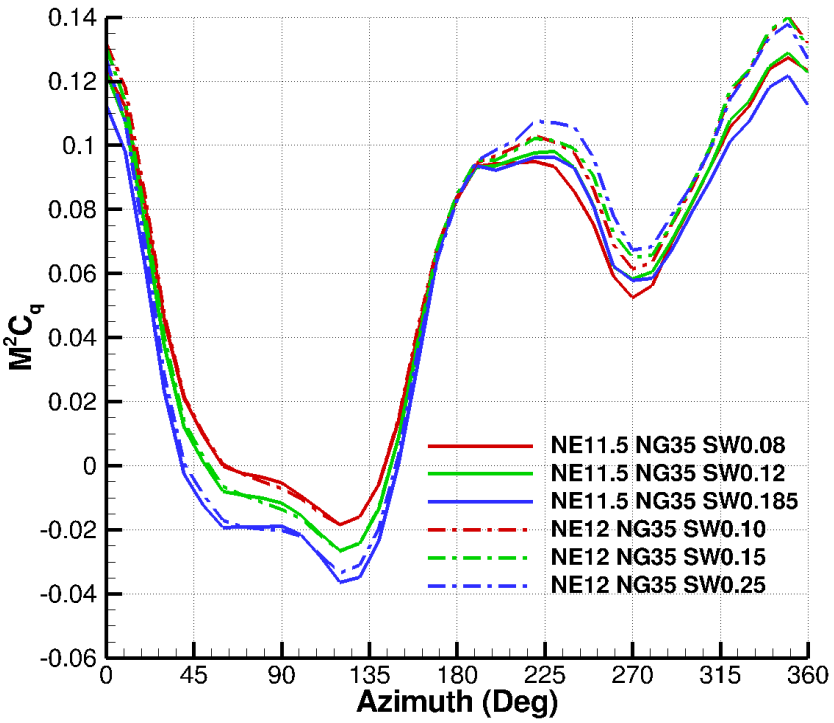
Figure 10.15 - 10.17 also compare the loads when the BERP area of the blade is increased i.e. when the notch is more inboards. Again, these are trimmed results although not much variation with notch position occurs in thrust with or without trimming as shown in Table 10.3. The effect of this parameter is to amplify the effect of the sweep parameter. For example, the redistribution of lift so that it is reduced at the back and increased at the front caused by sweep is larger in magnitude when the notch starts more inboard. The same can be seen for moment in Figure 10.16 where the region on the edge of the disk where moment is higher is thinner for the more outboard notch. For torque, the general trend is an increase with radial position. Where the notch occurs there is a drop in torque and then a continued increase followed by another drop where the anhedral occurs. With a more outboard notch, the torque continues to rise prior to reaching the notch for longer, therefore the latter part of the curve is higher. This can be seen in Figure 10.17 where the value of the reduced region at the notch is not as low when the notch is more outboard. Also, the torque further out from the notch is higher for the rotor with the more outboard notch. More inboards, on the advancing side, a decrease in torque is observed over a larger region and this brings the total value of the torque down as shown in Table 10.3. Figure 10.20 shows that the total torque on the blade is most affected by notch position on the retreating side.
The table 10.3 also shows that the peak-to-peak moment decreases (also seen in Figure 10.19) but the absolute average moment over a full revolution increases, the further outboard the notch is.

NE	NG	SWEEP	C_Q	avg M^2C_M	ΔM^2C_M
After Trimming					
11.50	28	0.185	0.000186	0.000296	0.011556
11.75	28	0.21	0.000186	-0.000197	0.011147
12.00	28	0.25	0.000184	-0.000468	0.010621

Table 10.3: *Example of BERP spanwise notch position performance comparison. NE is the notch position parameter and NG is the notch gradient parameter. avg M^2C_M is over one revolution and ΔM^2C_M is the peak-to-peak amplitude over one revolution. The sweep values differ because the gradient of the parabola differs when the position of notch changes, but in actual fact, the sweep is the maximum sweep on all three rotors and it is the same amount of sweep.*



M^2C_m



M^2C_q

Figure 10.20: *Comparisons of the integrated loads, M^2C_m and M^2C_q for BERP-like rotor with fixed parameters: NE = 11.5 and 12, NG = 35 and variable sweep parameters.*

10.5.3 Effect of Notch Gradient

Figure 10.21 compares the performance of different notch gradients. The notch gradient has a smaller influence on the design than the other parameters tested. The difference in the lift and moment distribution do not change much. With a higher notch gradient, there is a slight increase in the pitching moment, evident from Table 10.4 where the average moment is slightly higher and the peak-to-peak value is lower, suggesting that a higher notch gradient provides better performance. The torque is not affected much at low sweep, but at higher sweep, notch gradient has slightly more influence on the torque as shown in Table 10.4 and Figure 10.21.

NE	NG	SWEEP	C_T/σ	C_Q	avg M^2C_M	ΔM^2C_M
12.00	25	0.10	0.0908812	0.000189	-0.002561	0.009611
12.00	28	0.10	0.0906629	0.000189	-0.002464	0.009566
12.00	35	0.10	0.0910765	0.000189	-0.002357	0.009489
11.75	25	0.21	0.0897566	0.000188	-0.000295	0.010960
11.75	28	0.21	0.0898069	0.000186	-0.000197	0.011147
11.75	35	0.21	0.0898057	0.000187	-0.000105	0.011019

Table 10.4: *Example of BERP spanwise notch gradient performance comparison. NE is the notch position parameter and NG is the notch gradient parameter. avg M^2C_M is over one revolution and ΔM^2C_M is the peak-to-peak amplitude over one revolution.*

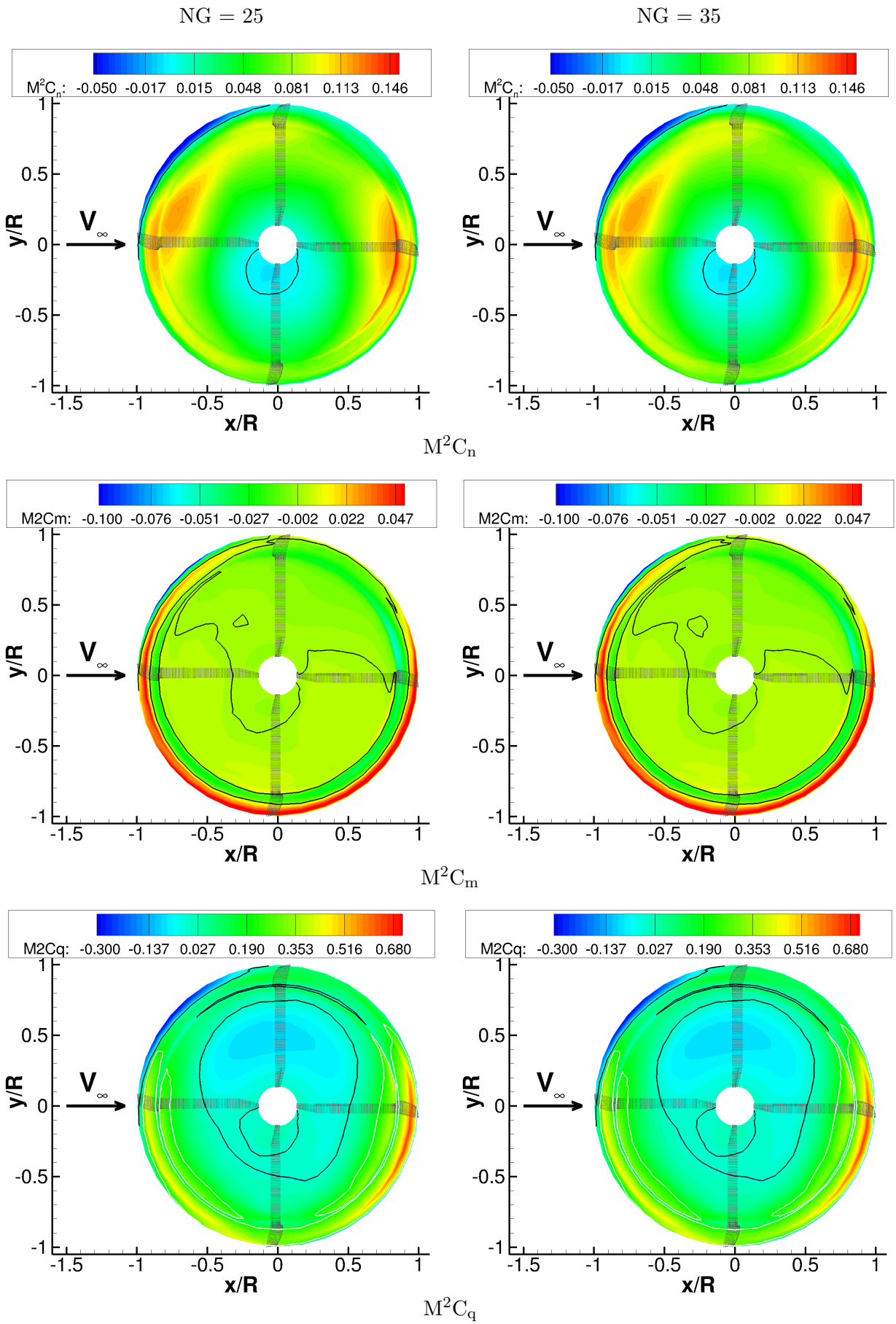


Figure 10.21: *M^2C_n, M^2C_m and M^2C_q for the BERP-like rotors with fixed parameters: NE = 11.75 and SW = 0.21 with varying NG. The black line indicates a contour level = 0 line and the white line indicates the approximate middle value for $M^2C_q = 0.2$.*

10.6 Integrated Loads

Figures 10.22 and 10.23 show the integrated loads of pitching moment and torque over the blade during one revolution for varying design parameters. The peak-to-peak moment value reduces with further outboard notch positions and the average moment tends to be more centred around zero when sweep is higher. The torque seems to be mostly affected by sweep and on the advancing and retreating side. The torque is reduced more on the advancing side than the increase on the retreating side. This is because it alleviates the compressibility effects. On the retreating side, the differences are more subtle. This data suggests that a highly swept blade would be optimal. Having a higher notch gradient would also improve the moments and having a notch more inboards would amplify the effects of the sweep. The quantities of the design parameters that make up the optimum design are obtained in the next section.
The vibratory pitching moment (defined as the pitching moment less the mean and 1/rev oscillation) in Figure 10.24 shows sensitivity to sweep mostly. However, its effect is captured in the peak-to-peak moment and hence it is not used in the objective evaluation.

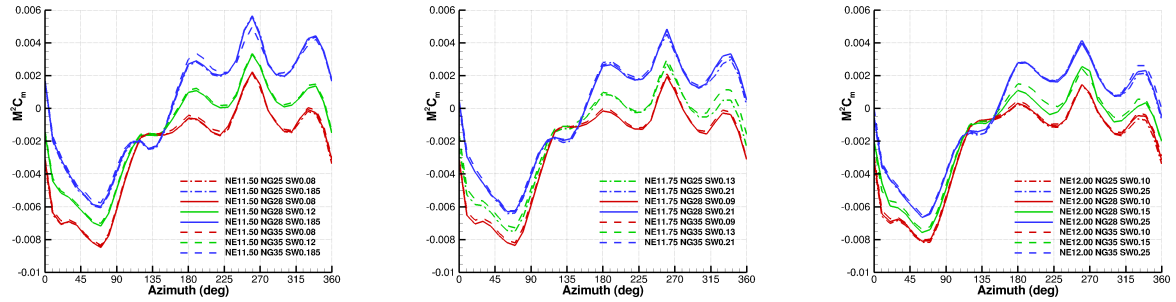


Figure 10.22: *M^2C_m integrated over the full blade at each azimuth for the BERP-like rotors.*

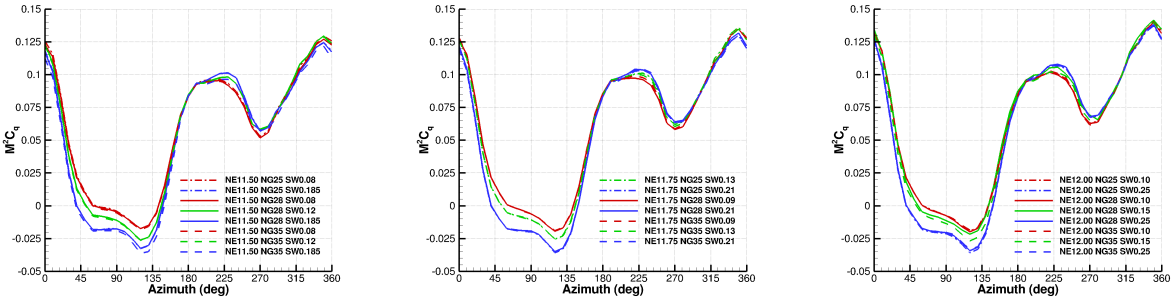


Figure 10.23: *M^2C_q integrated over the full blade at each azimuth for the BERP-like rotors.*

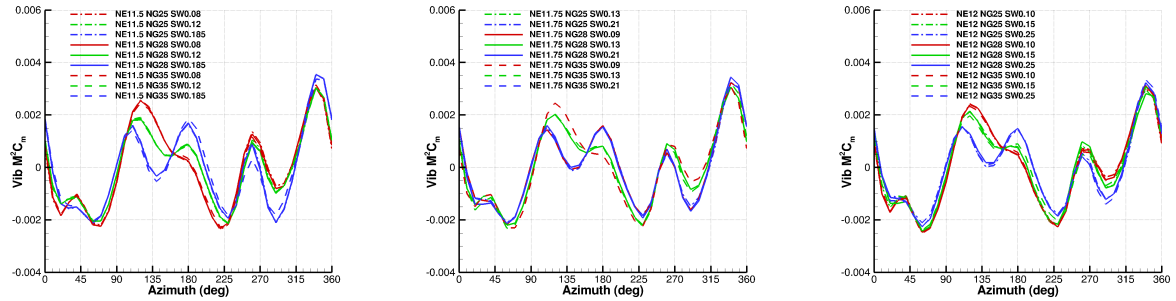


Figure 10.24: *Vibratory M^2C_m integrated over the full blade at each azimuth for the BERP-like rotors.*

10.7 Planform Optimisation

The original population obtained from the CFD results contained 27 points. The design parameters selected were the average pitching moment ($C_{m_{avg}}$), peak-to-peak pitching moment (ΔC_m) and the torque coefficient (C_q). The objective function weights were determined such that on average, each of these parameters had the same influence. This was determined using the data from the original population which was obtained using the high-fidelity CFD solver. First each design was scaled with the baseline design case. The baseline case chosen was the BERP most similar to the typical fast flying rotor as shown in Figure 10.25. The parameters for it are NE = 12, NG = 25, SW = 0.25. The average ratio of $C_{m_{avg}}$ to ΔC_m was found to be 2.7893:1 and the average ratio of ΔC_m to C_q was found to be 0.9548:1. Therefore the ratio of $C_{m_{avg}}$ to ΔC_m to C_q is obtained as: 2.6634 : 0.9548 : 1.0000. The weight for C_q was then calculated as:

$$w_{C_q} = \frac{2.6634}{2.6634 + 0.9548 + 1} = 0.5767 \tag{10.3}$$

Hence the weight of $C_{m_{avg}}$ and ΔC_m are given as:

$$w_{C_{m_{avg}}} = 0.5767/2.6634 = 0.2165 \tag{10.4}$$

$$w_{\Delta C_m} = 0.5767/0.9548 = 0.6040 \tag{10.5}$$

C_q was also used as a constraint. Since the rotors were trimmed to a $C_T/\sigma = 0.09$, C_T/σ did not need to be constrained.
From this data, it was determined that the most influential design parameter was the sweep, followed by the notch position and then the gradient. ANNs were trained for each of the performance parameters as shown in Figure 10.26. These parameters were used to find the optimum blade using a GA which was compare with the Pareto front shown in Figures 10.27 to 10.29. The ANNs accuracy was also estimated relative to the change in the performance of the design obtained using the CFD data. The maximum error in the objective function obtained was found to be 2.7%.
The comparison of the optimum with the original baseline blade is shown in Table 10.5 for the trimmed rotors. A much better avg M^2C_m was obtained and also a slightly better ΔM^2C_m. The performance of the resulting optimum relative to the baseline design is shown in Figures 10.30, 10.31 and 10.32. The black line indicates the contour line of the baseline design and the red is the optimum blade design. On the moment plot, it can be seen that the optimised blade has larger areas of lower moment especially on the retreating side but also on the outboard region of the advancing side. A similar trend can be seen for the torque plot. Figure 10.32 shows the difference in the objective function components between the optimum and the baseline design. For the regions of higher OFV, the areas enclosed by red are larger showing that the optimsed blade increases the area where performance is good and vice versa for areas of poorer performance.

The hover performance of the optimum blade was measured and compared to the baseline in Table 10.5. The results were obtained at a collective of 13 degrees. The C_T/σ is slightly less than the baseline design mostly due to the added solidity, but the FM obtained was higher. Figure 10.33 compares the C_P distribution for the BERP reference and optimised blade. It can be seen that the optimised one spreads the loading at the tip over more of the span. Therefore, overall, the optimised blade has better performance than the baseline blade especially in terms of moment where the average pitching moment was reduced to approximately a fifth of the baseline designs.

NE	NG	SWEEP	C_T/σ	C_Q	avg M^2C_M	ΔM^2C_m
11.75	35	0.21	0.0831761	0.000171	-0.000373	0.010782
After trim			0.0898057	0.000187	-0.000105	0.011019
Baseline			0.0904548	0.000186	-0.000517	0.010832

Hover Performance Comparison					
NE	NG	SWEEP	C_T/σ	FM	Collective (deg)
11.75	35	0.21	0.28957	0.6873	13
Baseline			0.30390	0.6543	13

Table 10.5: *BERP and baseline case (NE = 12, NG = 25, SW = 0.25) performance comparison related to Figure 10.30. NE is the notch position parameter and NG is the notch gradient parameter. avg M^2C_M is over one revolution and ΔM^2C_M is the peak-to-peak amplitude over one revolution*

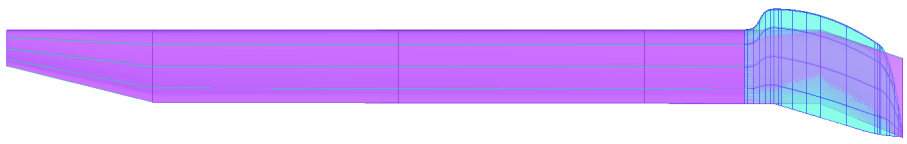


Figure 10.25: *The baseline BERP-like rotor in comparison to a swept tip design. The parameters for this rotor are NE = 12, NG = 25, SW = 0.25.*

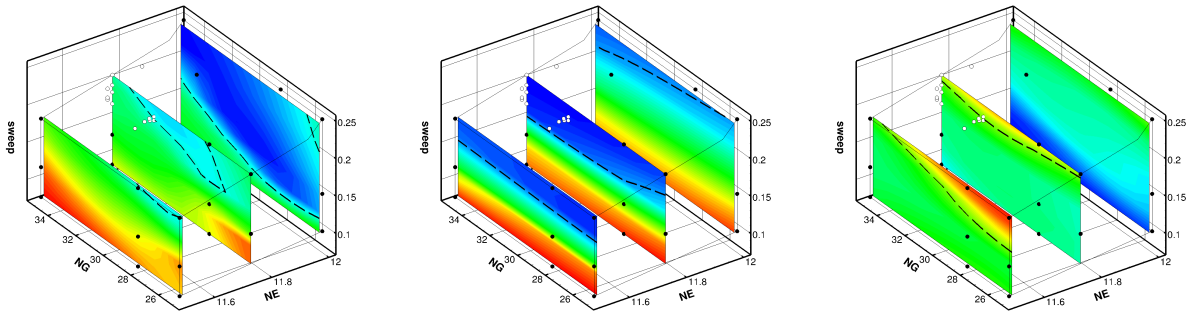


Figure 10.26: *ANN predictions with training data and GA selection shown for each of the performance parameters. The white dots are the GA optimal selection and the black dots are the CFD training data for the ANNs. The dashed line is where the contour level = 1 i.e. the value for the baseline design.*

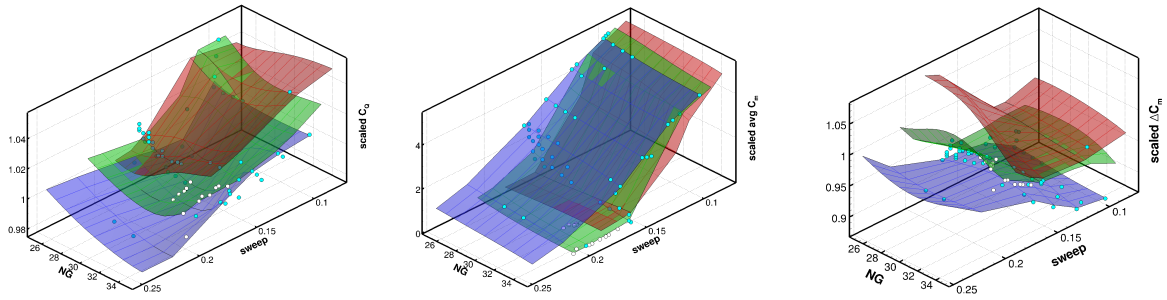


Figure 10.27: *Pareto front points compared with GA selection; red is NE = 11.5, green is NE = 11.75, blue is NE = 12. The white dots are the GA optimal selection and the cyan dots are the Pareto front solutions.*

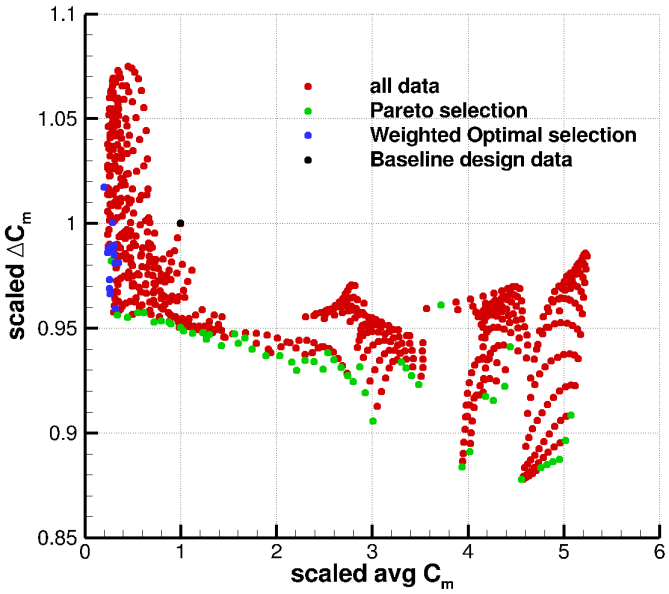
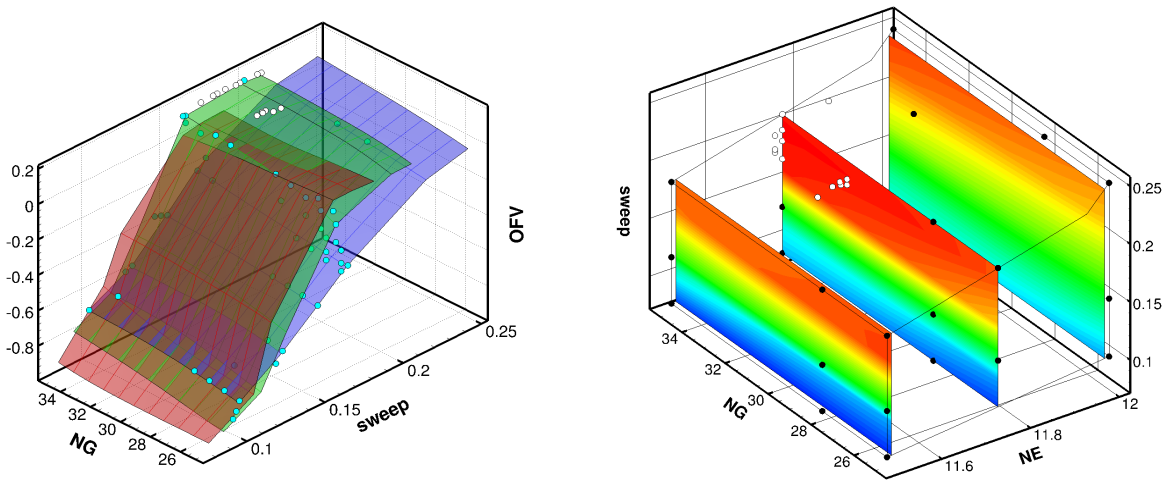


Figure 10.28: *Pareto front for the BERP-like design.*



(a) (b)

Figure 10.29: *(a) Pareto front points compared with GA selection; red is NE = 11.5, green is NE = 11.75, blue is NE = 12, (b) OFV contour colour map in the design space. The white dots are the GA optimal selection, the cyan dots are the Pareto selection and the black dots are the CFD training data for the ANNs.*

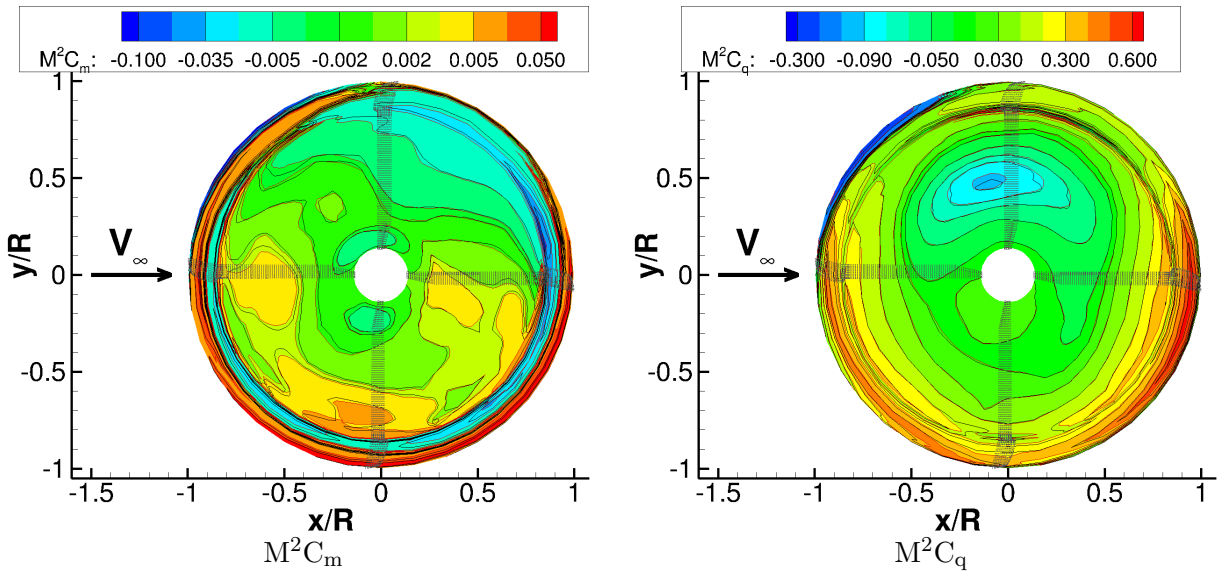


Figure 10.30: *Optimum (red contour lines) compared to reference (black contour lines) for M^2C_m and M^2C_q.*

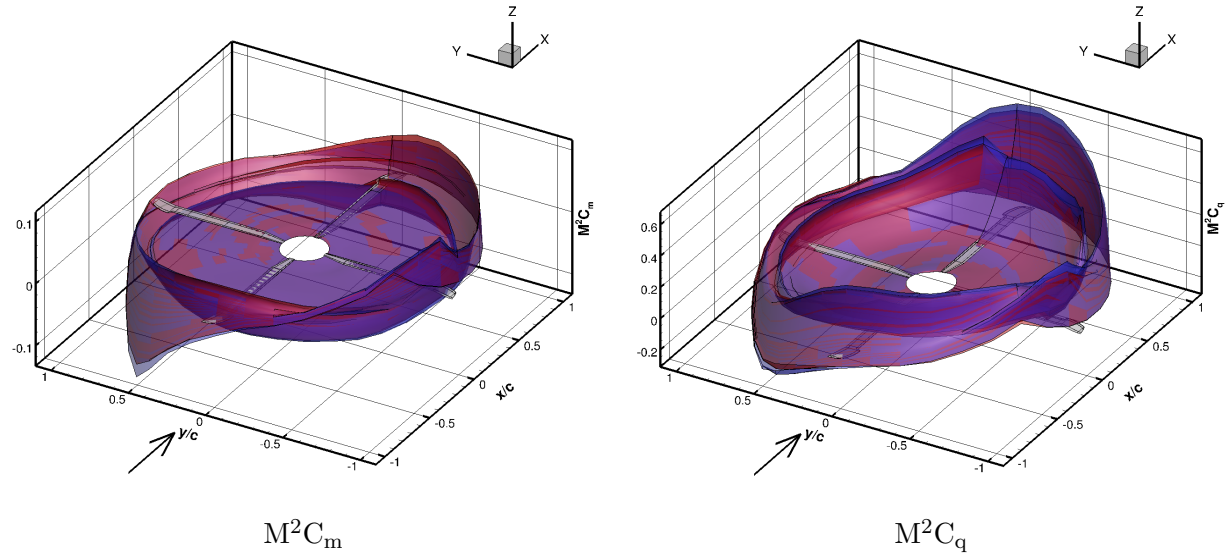


Figure 10.31: *3D view of load distribution for the optimum (red) and the original (blue) rotors.*

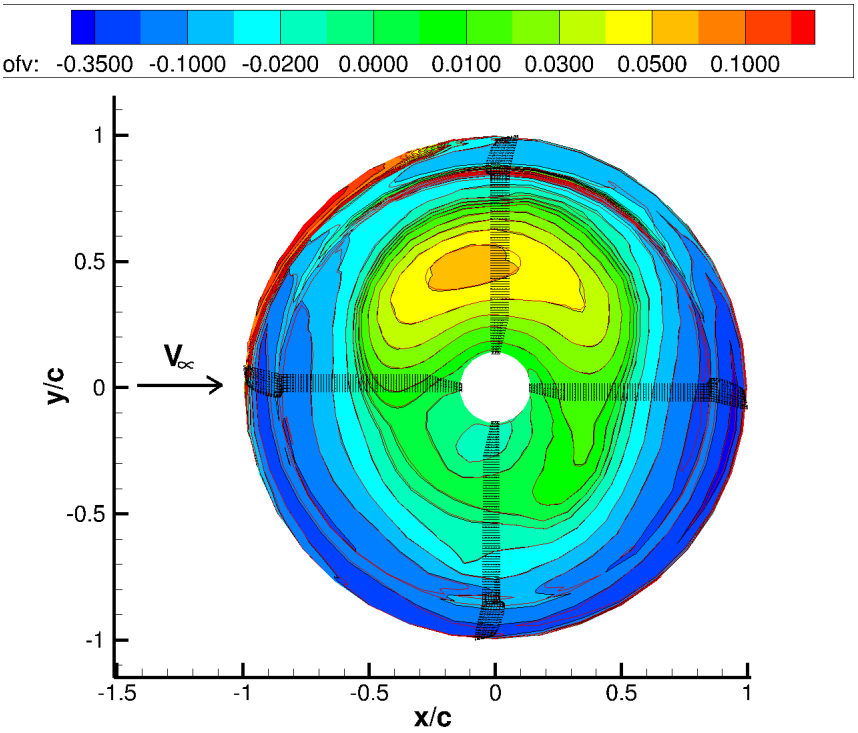
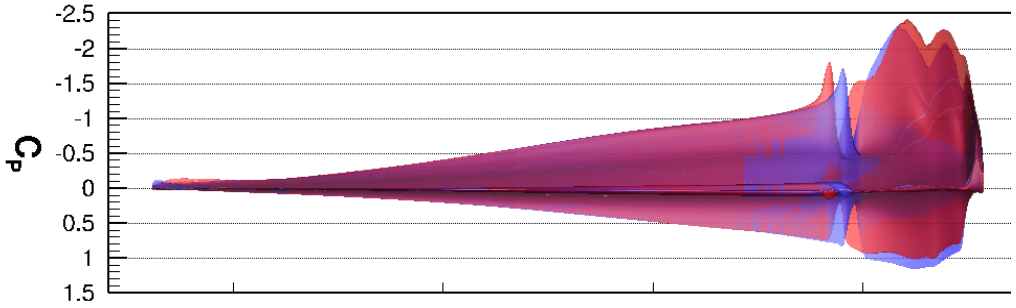


Figure 10.32: *OFV for the baseline and BERP-like rotor where the black contour lines represent the reference rotor with parameters: NE = 12, NG = 25, SW = 0.25, and the red lines represent the optimised rotor with parameters: NE = 11.75, NG = 35 and SW = 0.21 where OFV*$= -0.2 \times M^2 C_m - 0.6 \times M^2 C_q$.



C_P distribution comparison

Planform comparison

Figure 10.33: *C_p and planform distribution of the reference (blue) and optimised (red) BERP variant at high thrust (collective = 13 degrees) in hover.*

Chapter 11

Summary and Conclusions

11.1 Summary of the Optimisation Method

This thesis documents an optimisation method for helicopter rotor blades using Computational Fluid Dynamics (CFD). The objective was to "tweak" an existing good design to obtain even more performance out of it. The optimiser allows for high-fidelity CFD computations to be used to obtain accurate aerodynamic data for the objective, but at the same time, allows for this to be carried out efficiently by use of a lower order model, or metamodel. The information obtained from the metamodel is based on interpolation data from the high-fidelity model. This maintains accuracy and efficiency and makes the method usable for helicopter rotor aerodynamics. The main reason for this is that the optimisation method is decoupled from the high-fidelity CFD data. The two are linked through a database. The high-fidelity CFD data is used to populate the database to create a design space and the optimiser accesses this design space through a metamodel. In most cases, a parameterisation technique existed, but for some cases, specific parameterisation techniques had to be developed to create the design space.

The optimiser used was a Genetic Algorithm (GA). The reason for this is that the design space for rotor aerodynamics especially, is expected to be uneven and hence the likelihood of the optimiser getting trapped in local optima is high. Evolutionary methods avoid this problem and one of the most effective of these methods are GAs. GAs for all the cases were coupled with and relied on metamodels. A number of metamodels were tested, including polynomial and POD based methods, and the most promising ones were the Artificial Neural Networks (ANNs) and kriging methods.

Once all of this was put together, the method was demonstrated using a number of cases:

- transonic aerofoil test case in steady flow
- wing planform optimisation in steady flow
- optimal selection of rotor aerofoil sections in unsteady flow
- optimal selection of twist for hovering rotors in steady flow
- forward flying rotor optimisation in unsteady flow in conjunction with hover optimisation of the same
- optimisation of simplified fuselage bodies for drag reduction
- optimisation of a BERP-like rotor in forward flight constraining hover performance

For the transonic aerofoil case (Chapter 4), the RAE 2822 aerofoil was used and a parameterisation technique was created using the Chebyshev polynomial method (described in Section 3.2.1). Three parameters were optimised for high lift to drag ratio while constraining drag and

drag divergence Mach number. The results showed an 8% increase in L/D with negligible change in pitching moment and drag divergence Mach number. A version of the Latin Hypercube Sampling method (LHS) was also tested against the full factorial method.

For the wing planform optimisation in Chapter 6, the aim was to obtain elliptic loading by varying the chord and sweep while maintaining lift and drag. The wing was split into three sections and their position and width were defined as the five design parameters. Due to the relatively larger number of parameters, the sampling method used was a version of adaptive fractional factorial method, where points were added to the database where there was the most promise of finding the optimum. The results showed a 13% improvement in the elliptic loading and a small improvement in lift-to-drag while constraining maximum drag.

In Chapter 5 for the rotor section optimisation, dM/dt calculations were performed to simulate a rotor section in forward flight. NACA aerofoils were used and the design parameters were the thickness and camber as defined using the NACA numbering system. The objective was to reduce the effects of compressibility on the advancing side and of high angle of attack on the retreating side. This was captured using the moment curve over a full revolution of the rotor and the average drag. This case was also used to test the effect of the ANN parameters on its prediction accuracy and to compare the predictions of the kriging and ANN metamodels. The final design had resulted in over 50% reduction in the average pitching moment and at most, a 37% reduction in average drag both, inboard and outboard, over one revolution.

The optimisation method was then applied to full rotors in hover and in forward flight. Chapter 7 describes a simple rectangular rotor optimisation in hover for high Figure of Merit (FM) over a range of thrust coefficients. This was obtained by modifying the linear twist of the blade and measuring the highest FM and the gradient of the FM over a range of collective angles. The LHS method was tested here and its limitations for small sets of data was shown. The kriging method was also compared to the ANN for this case. The twist selected was slightly less than the maximum in the design space. The increase in the maximum FM from the original was 3%, but the maximum FM changed less with thrust than the baseline which is quantified by the gradient of the FM against the thrust. This was improved by 7.5%.

Chapter 8 describes the optimisation of the UH60-A rotor sweep and anhedral of the blade tip in both hover and forward flight. The parameterisation technique ensured that the area of the tip remained constant and that a smooth surface was maintained where the anhedral occurred. The objectives were to reduce stall effects on the retreating side and compressibility effects on the advancing side and the pitching moment was used to capture this via two parameters viz. the peak-to-peak and the average value over a full revolution. The vibratory pitching moment and the torque were constrained. The resulting blade had a lot more anhedral and slightly less sweep resulting in a 6.7% decrease in torque, a 17.6% decrease in the peak-to-peak pitching moment over a revolution and a 24% decrease in the average pitching moment in one revolution. The hover performance with the new planform was maintained.

The final rotor case was the BERP-like rotor tip (Chapter 10). Here a parameterisation technique was defined using a set of equations that captured three design parameters viz. the notch position, notch gradient and sweep of the tip. Due to the significant change in area, the rotor was trimmed to ensure the same thrust/solidity of the rotor. The objective was the same for the forward flight optimisation of the UH60-A rotor. The forward flight conditions were taken from a typical fast-flying, moderate lift rotor. The hover performance, in terms of FM, with a BERP modification was maintained by modifying the twist and anhedral. This rotor was then used to optimise the BERP-like rotor in forward flight and the final design was then analysed in hover to check for any performance compromise. The objectives were captured using the pitching moment curves as before, but torque was also included in the objective as well as a constraint. Here, the

Harmonic Balance method was used to obtain all the forward flight data for the initial design space as opposed to the time marching method used in all the other cases. This significantly improved the efficiency of the method. The final blade had a steep notch at about 86% of the span with high sweep resulting in an improvement in the average pitching moment over a revolution of 80%, although the peak-to-peak moment was increased by 1.7%. The torque did not change much and hover performance was maintained.

The method was also applied to fuselage optimisation in Chapter 9. As an example, the JM-RTS fuselage from JAXA was selected as a suitable test case. The parameterisation used was a modification of the super-elliptic equation technique used to define the ROBIN fuselage. The objective was to reduce the drag by changing the angle of the wind screen. This test case was mainly carried out for development of the parameterisation method. The generation of geometries is automated in ICEMCFD. Also a very simplified uniform actuator disk was included to simulate the downwash from the rotor on to the fuselage. A 3.3% reduction in drag was achieved for a shape that reduced the gradient of the windscreen area.

The conclusions drawn about the method and each of these test cases is explained further in the next section.

11.2 Conclusions

The CFD database for most of the cases were built using the HMB Time Marching method. However, the Harmonic Balance method was used for the BERP rotor case because of its much faster clock time in obtaining high-fidelity results. The HB method used was able to obtain a snapshot at every 10 degrees of azimuth. The only limitation was the memory required. With higher number of modes and higher grid resolution, the memory increases rapidly, hence, for the same number of snapshots as in the TM method, which requires a set number of modes and for a limited amount of memory, the grid resolution is sometimes reduced. Nevertheless, the results obtained were shown to be very similar to that obtained by time marching and any change in values was relative resulting in the same outcome.

The metamodel of choice was mostly the ANN and in some cases, the kriging method. These two methods were most popular because of their ability to model the design space accurately without overfitting or underfitting the data with changes in their parameters. Metamodels were checked aposteriori against HMB solutions. The optimiser developed was a GA. While a number of different evolutionary techniques could have been used, such as Simulated Annealing, Particle Swarm etc., from the literature, GAs are one of the most successful techniques in terms of implementation, ability to find the global optimum and relative efficiency. Within the objectives of this project, it performed well. Future work will include testing it against other non-gradient techniques.

The parameterisation techniques used were dependent on each case. The choice of parameters and their values depended on the experience of the user, the objectives required and the initial design.
For example, in the case of the transonic aerofoil optimisation, the RAE 2822 aerofoil was parameterised using Chebyshev polynomials. Six coefficients were required to define the shape. However, only the first three were optimised as they controlled the thickness, the camber and the position of maximum camber, which are known to significantly change the lift-to-drag ratio. The final design had more or less the same thickness and camber but with the camber pushed further backwards creating more favourable pressure gradients.

For the wing, the parameters were simply the chord length and position at two spanwise stations and the position of the tip. This effectively controlled the sweep and taper of the wing and

the objective was to obtain as close to elliptic loading as possible. The final design removed the forward sweep of the full wing, tapered the end a bit more and maintained the backward sweep at the tip. This reduced the loading more outboards which resulted in a lift distribution closer to an elliptic distribution due.

For the rotor section optimisation, a parameterisation technique already existed since NACA aerofoils were used. Again, the important characteristics chosen were the thickness and camber of the aerofoils. The optimum selection was for thick, cambered aerofoils, such as the NACA 33015, inboard of the rotor (r/R = 0.5) and thinner (9% thickness), more symmetrical (9% camber) aerofoils, outboards (r/R = 0.9). This is a well-known and tested theory since inboards the Mach number is slower therefore thicker aerofoils can be used. High camber is also advantageous because of the high angles of attack experienced inboard due to the twist of the blade. Outboards, the flow is more compressible and so the optimum tends to avoid shock effects. This case was a good case to use as a starting point to develop the optimisation method since the results expected were more or less known.

Similarly for the hovering rotor, high twist is expected to be the optimum for obtaining good Figure of Merit (FM). The compromise was between obtaining a high single point FM or a lower FM but for a bigger range of thrust. The optimum selected had 11 degrees of twist, which was less than the maximum twist tested of 12 degrees, showing that a wide design space was selected. The optimum also removed stall from the tip when compared to the untwisted blade.

For the UH60-A and the BERP-like rotors, twist as well as anhedral were optimised in hover. In both cases, however, forward flight was the main condition optimised for since this is the main condition that these rotors were designed for. Hover performance was then improved or constrained. In the UH60-A case, the parameters, sweep and anhedral were first optimised for forward flight, then the new design was tested in hover and found to maintain good performance. Nevertheless, twist was optimised for hover as well, and found to be approximately the same as the original rotors hover. Much more anhedral was added to the tip (11 degrees) which helped reduce the drop in moment on the advancing side which reduced the overall average moment. The sweep was slightly reduced (14.5%) since not as much of it was required as the anhedral off-loaded the tip of the rotor.

For the BERP-like rotor, a simulation of a typical fast flying, moderate thrust rectangular rotor with sweep was the baseline point. This planform was converted to a BERP-like tip using a parameterisation technique that used parabolic and sigmoidal curves. Using this technique, three important features of the BERP-like tip could be captured viz. the sweep, the notch gradient and the notch position. However, this baseline BERP-like tip had a lower performance in hover. Therefore, the twist and anhedral were first optimised in hover to obtain the same performance of the original rectangular swept rotor. More anhedral was added to offload the tip and a higher twist was added to regain the load more inboards. Once this blade was obtained, the planform was then optimised for forward flight. The parameter that had the biggest effect in load distribution was the sweep. The position of the notch had an effect on the peak-to-peak pitching moment and average moment. It also amplifies the effect of the BERP-like part of the blade since a more inboard notch means a larger BERP-like tip i.e. a lower rise in torque, a bigger rise in moment and so on. The notch gradient has a smaller effect on the performance, although a higher notch gradient reduces the torque slightly.
Overall, the optimum selection was a highly swept tip, with a high notch gradient and the initiation of the notch in between the two extremes. The reason for this was because its average pitching moment was very close to zero. Torque and peak-to-peak did not change much with notch position. Peak-to-peak moment was mainly affected by the sweep, where lower sweep was preferable. However, the effect was not large compared to the gain in average moment with higher sweep. A higher notch gradient also favoured a large decrease in the average moment. This rotor

was then tested in hover and maintained good hover performance over a range of thrust. In terms of the parameters used in Chapter 10, the optimum had a sweep of 0.21 (same as the baseline), a notch gradient of 35 (40% higher than the baseline) and a notch position of 11.75 (about 2% more inboards).

A further case that was run, was the JMRTS fuselage. The objective was to optimise the wind screen gradient for drag. Therefore, the parameters were linked to a single value that controlled this gradient. The automation of the grid generation part made it easy to modify this parameter and obtain results quickly. The final design was the one that had the least gradient, which was approximately 24% less than the baseline in terms of the parameterisation coefficient. This case was mainly performed for the parameterisation part for future work.

Overall, the method works well in fine-tuning parameters for improved aerodynamic performance efficiently. Some understanding of rotor design is still necessary and the aim was not to replace, but assist the designer in obtaining optimal designs. There is still much work that can be carried out for this endeavour as described below.

11.3 Future Work

The objective of this project was to create an efficient technique for the optimisation of aspects of rotor blades so that this can be of practical use. While, this objective has been successful, a further development would be to optimise the method itself so that a number of options are available to the user in terms of metamodels, optimisers and so on. Also, a more integrated approach would make it easier for the user. For example, a python script could be used to train the ANN and produce the results suggesting where further points may be required and once the error predictions fall within a given level, then the optimiser is run automatically and all the data presented to the user.

The adjoint method is a popular method for optimisation. With this method, the sensitivity of the optimisation is directly linked to the shape. It allows a more dynamic way of finding the optimum, although it allows perhaps too much flexibility for "tweaking" purposes since the sensitivity is linked to the shape via the grid points, rather than a selected set of parameters. The availability of this method, at least for small cases, along with the method described so far, would be advantageous to compare and develop.
Along these lines, a parameterisation method that automatically creates the CAD design would be required. In this project, the use of ICEMCFD replay files allowed some automation, especially for the wing case and the fuselage and part of the BERP-like rotor.

In terms of metamodels, further investigation into the POD technique would be useful. The POD method was found to be useful with good accuracy where large amounts of data are available. This may be the case depending on what the optimisation purpose is and the method used. For example, it may be useful to use it in conjunction with an adjoint method to predict flow. However, it would also be beneficial to explore a way of using this method accurately for small amounts of data. The advantage in the POD technique is that very little training time is required when compared to the ANN method and it is not sensitive to as many variables as in the case of ANNs and kriging. It would also be beneficial to be able to compare the metamodels prediction accuracy with change in the design points. This could be used to determine the resolution required in the initial database.

Further work could also be carried out for sampling methods. This area was not explored in much detail since the nature of this project only required small design spaces and hence, full factorial methods were used mostly, even though some work was done using fractional factorial and Latin Hypercube Sampling techniques. Adaptive techniques were used where points were added during the optimisation process itself to refine promising areas of the design space. This was done

manually. However, automation of this method would improve the efficiency of the overall code.

The objective function evaluation requires user experience, especially in determining what performance parameters capture the objectives. The weights also require some experience, although guided by the initial CFD data. However, the way the weights are determined could be further improved, by using the statistical data of the database, such as the the mean, standard deviation and variance of the objective components relative to each other. This allows the designer to equalise the weightings of the different objectives. However, what is also required, is a way of quantifying the value of a performance parameter for a specified design criterion. This will allow the designer to place the right weight for each component. A deeper analysis of the initial data, such as the coupling of performance parameters could lead to a better understanding of how much weight to place on each component.

The high-fidelity model used two methods viz. the Time Marching (TM) and Harmonic Balance (HB). The HB method was much more efficient in terms of wall clock time, however it is limited by its high memory usage. Further work on reducing the memory usage of this method would increase its accuracy as well as maintaining its efficiency. This would also allow further investigation in to more subtle changes in the BERP-like tip like the effect of the gradient of the notch and possibly other parameters not explored in this work.
In addition to the high-fidelity model, a more multi-disciplinary approach would be an important improvement. The inclusion of aeroelastic and aeroacoustic effects would allow the problems associated with these disciplines to be included in the optimisation process. These are significant issues for the rotorcraft industry. The current model will require changes for this to be included, but the overall concept can still be used to include these disciplines as long as good performance parameters that capture the objectives and constraints can be obtained.

A more advance model could also be used for the actuator disk in the fuselage optimisation. This would create more realistic results especially at faster flight conditions. Also, the inclusion of the hub in the geometry would produce results more similar to the experimental data.

For rotor section optimisation, a comparison at the same geometric angle of attack and was carried out. However a comparison at the same lifting capability would provide more insight into the advantage of one section over another.

For the wing planform optimisation case, another good optimisation problem would be to optimise the taper and twist of the planform for the same performance parameters.

In the future the aim is to also apply this method to investigate interactional aerodynamics such as rotor/fuselage, main rotor/tail rotor and other interactions. It is also hoped that with increasing computing power and efficient methods, the inclusion of other disciplines such as aeroelasticity would become feasible.

Bibliography

[1] S. A. Orr and R. P. Narducci. Framework for Multidisciplinary Analysis, Design, and Optimization With High-Fidelity Analysis Tools. NASA/CR-2009-215563, February 2009.

[2] J. L. Walsh. Performance Optimization of Helicopter Rotor Blades. *NASA Technical Memorandum*, 104054, 1991.

[3] R. M. Hicks, E. M. Murman, and G. N. Vanderplaats. An Assessment of Airfoil Design by Numerical Optimization. *NASA Technical Report*, TM X-3092, 1974.

[4] K. C. Tan, T. H. Lee, and E. F. Khor. Evolutionary Algorithms for Multi-Objective Optimization: Performance Assessments and Comparisons. *Journal of Artificial Intelligence*, 19:253 – 290, 2002.

[5] J. Hájek. Paremetrization of Airfoils and Its Application in Aerodynamic Applications. *Proceedings of Contributed Papers*, Part 1:233 – 240, 2007.

[6] A. Spentzos. *CFD Analysis of 3D Dynamic Stall*. PhD thesis, University of Glasgow, 2005.

[7] H. Sugawara, Y. Tanabe, and S. Saito. A Numerical Study of Rotor/Fuselage Interaction Based on the JMRTS Database. In *Heli Japan Conference*, volume T220-2, Saitama, Japan, November 2010.

[8] F. J. Perry. The Aerodynamics of the World Speed Record. In *43rd Annual Forum of the American Helicopter Society*, St;.Louis, USA, 1987.

[9] A. Brocklehurst and E. P. N. Duque. Experimental and Numerical Study of the British Experimental Rotor Programe Blade. In *AIAA 8th Applied Aerodynamics Conference*, volume AIAA-90-3008, Portland, Oregon, USA, August 1990.

[10] N. W. Schaeffler, B. G. Allan, C. Lienard, and A. Le Pape. Progress Towards Fuselage Drag Reduction via Active Flow Control: A Combined CFD and Experimental Effort. In *36th European Rotorcraft Forum*, volume 064, Paris, France, September 2010.

[11] D. C. Wilcox. Multiscale Model for Turbulent Flows. *AIAA Journal*, 26(11):1311 – 1320, November 1988.

[12] F. X. Caradonna. The Application of CFD to Rotary Wing Problems. *NASA Technical Memorandum*, March 1992.

[13] J. G. Leishman. *Principles of Helicopter Aerodynamics*. Cambridge Aerospace Series, 2ndedition edition, 2005.

[14] W. Johnson. NDARC NASA Design and Analysis of Rotorcraft: Theoretical Basis and Architecture. In *American Helicopter Society Aeromechanics Specialist Conference*, San Francisco, California, USA, January 2010.

[15] W. Johnson. NDARC - NASA Design and Analysis of Rotorcraft Validation and Demonstration. In *American Helicopter Society Aeromechanics Specialist Conference*, San Francisco, California, USA, January 2010.

[16] R. Harrison, S. Stacey, and B. Hansford. BERP IV The Design, Development and Testing of an Advanced Rotor Blade. In *American Helicopter Society 64th Annual Forum*, Montral, Canada, April/May 2008.

[17] R. Celi. Recent Applications of Design Optimization to Rotorcraft - A Survey. In *55th Annual Forum of the American Helicopter Society*, 1999.

[18] J. Pereira, D. I. Chopra, F. Bohorquez, M. Chehab, R. Couch, T. DuVall, J. Park, and B. Singh. *406-UM TerpRanger Light Helicopter Upgrade*. 2002.

[19] C. W. Acree, Jr. Integration of Rotor Aerodynamic Optimization with the Conceptual Design of a Large Civil Tiltrotor. In *American Helicopter Society Aeromechanics Specialist Conference*, San Francisco, California, USA, January 2010.

[20] P. B. Martin and J. G. Leishman. Trailing Vortex Measurements in the Wake of a Hovering Rotor Blade with Various Tip Shapes. In *58th Annual Forum of the AHS International*, June 2002.

[21] F. X. Caradonna and C. Tung. Experimental and Analytical Studies of a Model Helicopter Rotor in Hover. *NASA Technical Memorandum*, 81232, TR-81-A-23, September 1981.

[22] C. V. Cook. The Structure of Rotor Blade Tip Vortex. *AGARD CP-111*, September 1972.

[23] J. D. Anderson, Jr. *Computational Fluid Dynamics, The Basics with Applications*. McGraw-Hill International Editions, 1995.

[24] C. Keys, F. Tarzanin, and F. McHugh. Effect of twist on Helicopter Performance and Vibratory Loads. In *European Rotorcraft Forum*, 1987.

[25] N. Rajmohan, R. P. Marpu, L. N. Sankar, J. D. Baeder, and T. A. Egolf. Improved Prediction of Rotor Maneuvering Loads using a Hybrid Methodology. In *American Helicopter Society 67th Forum*, Virginia, USA.

[26] R. T. Biedron and E. M. Lee-Rausch. An Examination of Unsteady Airloads on a UH-60A Rotor: Computation versus Measurement. In *American Helicopter Society 68th Forum*, Fort Worth, Texas, May 2012.

[27] B. Glaz, P. P. Friedmann, and L. Liu. Surrogate Based Optimization of Helicopter Rotor Blades for Vibration Reduction in Forward Flight. *Structural Multidisciplinary Optimisation Journal*, 35:341 – 363, June 2007.

[28] B. Glaz, T. Goel, L. Liu, P. edmann, and R. Haftka. Multiple-Surrogate Apporach to Helicopter Rotor Blade Vibration Reduction. *AIAA Journal*, 47(1):271 – 282, January 2009.

[29] C. B. Allen, T. C. S. Rendall, and A. M. Morris. CFD-Based Twist Optimization of Hovering Rotors. *Journal of Aircraft*, 47(6):2075 – 2085, November 2010.

[30] K. Collins, J. Bain, N. Rajmohan, L. Sankar, T. A. Egolf, R. D. Janakiram, K. Brentner, and L. Lopes. Toward a High-Fidelity Helicopter Rotor Redesign Framework. In *AHS 64th Annual Forum*, Montréal, Canada, 2008.

[31] S. K. Rallabhandi and D. N. Mavris. Aircraft Geometry Design and Optimisation for Sonic Boom Reduction. *Journal of Aircraft*, 44(1):35 – 47, January - February 2007.

[32] N. J. Kolencherry and W. A. Crossley. Multi-Fidelity Optimization Strategies using Genetic Algorithms and Sequential Kriging Surrogates. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, volume AIAA 2012-0152, Nashville, Tennessee, USA, January 2012.

[33] H. M. Adelman and W. R. Mantay. Integrated Multidisciplinary Optimization of Rotorcraft: A Plan for Development. *NASA Technical Memorandum, AVSCOM Technical Memorandum 89-B-004*, 10-1617, May 1989.

[34] F. Bohorquez, D. Pines, and P. D. Samuel. Small Rotor Design Optimization Using Blade Element Momentum Theory and Hover Tests. *Journal of Aircraft*, 47(1):268 – 283, January/February 2010.

[35] P. B. A Le Pape. Numerical Optimization of Helicopter Rotor Aerodynamics Performance in Hover. In *Applied Aerodynamics Department*, ONERA, France, 2005.

[36] A. Le Pape. Numerical Aerodynamic Optimization of Helicopter Rotors: Multi-Objective Optimization in Hover and Forward Flight Conditions. In *31st European Rotorcraft Forum*, ONERA, France, 2005.

[37] M. Imiela. High-Fidelity Optimization Framework for Helicopter Rotors. In *35th European Rotorcraft Forum*, 2009.

[38] T. Mengistu and W. Ghaly. Aerodynamic Optimization of Turbomachinery Blades using Evolutionary Methods and ANN-based Surrogate Models. *Journal of Optimization Engineering*, 9:239 – 255, 2008.

[39] A. Samad and K.-Y. Kim. Shape Optimization of an Axial Compressor Blade by Multiobjective Genetic Algorithm. *Proc. IMechE Part A: Journal of Aerospace Engineering*, 222:599 – 611, 2008.

[40] J. Liu, Z. Han, and W. Song. Efficient Kriging-Based Aerodynamic Design of Transonic Airfoils: Some Key Issues. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, volume 2012-0967, Nashville, Tennessee, USA, January 2012.

[41] S. N. Lophaven, H. B. Nielson, and J. Søndergaard. DACE, A MATLAB Kriging Toolbox, Version 2.0. *Informatics and Mathematcial Modelling*, IMM-TR-2002012, August 2002.

[42] G. Bohling. Kriging. *Kansas Geological Survey*, October 2005.

[43] A. I. J. Forrester and A. J. Keane. Recent Advances in Surrogate-based Optimization. *Progress in Aerospace Sciences*, 45:50 – 79, 2009.

[44] B. Glaz, L. Liu, P. P. Friedmann, J. Bain, and L. N. Sankar. A Surrogate Based Approach to Reduced-Order Dynamic Stall Modelling. In *51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Orlando, Florida, USA, April 2010.

[45] B. Glaz, L. Liu, P. Friedmann, L. Bain, and L. Sankar. A Surrogate Based Approach to Reduced Order Dynamic Stall Modeling. In *6th AIAA Multidisciplinary Design Optimization Specialist Conference*, volume AIAA 2010-3042, Orlando, Florida, USA, April 2010.

[46] J. Peter and M. Marcelet. Comparison of Surrogate Models for Turbomachinery Design. *WSEAS Transactions on Fluid Mechanics*, 3(1), January 2008.

[47] M. Y. M. Ahmed and N. Qin. Metamodels for Aerothermodynamic Design Optimization of Hypersonic Spiked Blunt Bodies. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, volume AIAA 2010-1318, Orlando, Florida, USA, January 2010.

[48] T. C. S. Rendall and C. B. Allen. Evaluation of Radial Basis Functions for CFD Volume Data Interpolation. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, volume AIAA 2010-1414, Orlando, Florida, USA, January 2010.

[49] H. Sun, Y. Kim, S. Lee, and D. Lee. Technical Note: Aerodynamic Design of Helicopter Rotor Blade in Forward Flight Using Response Surface Methodology. *Journal of the American Helicopter Society*, 48(4), 2003.

[50] P. Hajela. Nongradient Methods in Multidisciplinary Design Optimization - Status and Potential. *Journal of Aircraft*, 36(1):255 – 265, January/February 1999.

[51] S. K. Nadarajah and C. Tatossian. Multi-Objective Aerodynamic Shape Optimization for Unsteady Viscous Flows. *Journal of Optimization Engineering*, 2008.

[52] C. Tatossian, S. K. Nadarajah, and P. Castonguay. Aerodynamic Shape Optimization of Hovering Rotor Blades using a Non-linear Frequency Domain Approach. In *46th AIAA Aerospace Sciences Meeting and Exhibit*, number AIAA 2008-322, Jan 2008.

[53] A. Oyama, T. Nonomura, and K. Fujii. Data Mining of Pareto-Optimial Transonic Airfoil Shapes Using Proper Orthogonal Decomposition. *Journal of Aircraft*, 47(5):1756 – 1762, September/October 2010.

[54] T. Bui-Thanh, M. Damodaran, and K. Willcox. Aerodynamic Data Reconstruction and Inverse Design Using Proper Orthogonal Decomposition. *AIAA Journal*, 42(8), August 2004.

[55] K. Willcox. Unsteady Flow Sensing and Estimation via the Gappy Proper Orthogonal Decomposition. January 2004.

[56] H. Gunes, S. Sirisup, and G. Karniadakis. Gappy Data: to krig or not to krig? *Journal of Computational Physics*, 212:358 – 382, 2006.

[57] Y. Tahara, D. Peri, E. F. Campana, and F. Stern. Computational Fluid Dynamics-Based Multiobjective Optimization of a Surface Combatant using a Global Optimization Method. *Journal of Marine Science Technology*, (13):95116, 2008.

[58] M. K. Karakasis and K. C. Giannakoglou. On the Use of Surrogate Evaluation Models in Multi-Objective Evolutionary Algorithms. In *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, July 2004.

[59] Y. Mack, T. Goel, W. Shyy, and R. Haftka. Surrogate Model-Based Optimization Framework: A Case Study in Aerospace Design. *Studies in Computational Intelligence Journal*, 51:323 – 342, 2007.

[60] D. Crowley, B. Robertson, R. Douglas, and D. N. Mavris. Aerodynamic Surrogate Modeling of Variable Geometry. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, volume AIAA 2012-0268, Nashville, Tennessee, USA, January 2012.

[61] T. J. Mackman, C. B. Allen, M. Ghoreyshi, and K. J. Badcock. Comparison of Adaptive Sampling Methods for Generation of Surrogate Aerodynamic Models. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, volume AIAA 2011-1171, Orlando Florida, Jan 2011.

[62] L. Cameron, J. M. Early, and R. McRoberts. Metamodel Assisted Multi-Objective Global Optimisation of Natural Laminar Flow Aerofoils. In *29th AIAA Applied Aerodynamics Conference*, Honolulu, Hawaii, June 2011.

[63] G. Petrone, C. de Nicola, D. Quagliarella, J. Witteveen, J. Axerio-Cilies, and G. Iaccarino. Wind Turbine Optimization Under Uncertainty with High Performance Computing. In *29th AIAA Applied Aerodynamics Conference*, volume AIAA 2011-3806, Honolulu, Hawaii, June 2011.

[64] R. Hirsh. *GADO: A Genetic Algorithm for Continuous Design Optimization.* PhD thesis, State University of New Jersey, 1998.

[65] G. N. Vanderplaats. CONMIN - A FORTRAN Program for Constrained Function Minimization, User's Manual. *NASA Technical Memorandum*, NASA TM-62,282, August 1973.

[66] H. Zhao, S. Wang, W. Han, and G. Feng. Aerodynamic design by Jointly Applying S2 Flow Surface Calculations and Modern Optimization Methods on Multistage Axial Turbines. *Frontiers of Energy and Power Engineering in China Journal*, 2(1):93 – 98, 2008.

[67] T. Watanabe, K. Matsushima, and K. Nakahashi. Aerodynamic Shape Optimization of a Near-Sonic Passenger Plane using Computational Fluid Dynamics. *Proc. IMechE Part G: Journal of Aerospace Engineering*, 222:1025 – 1035, 2008.

[68] L. Xiong, C. Yan, and Y. Zhiquan. Optimization Model for Rotor Blades of Horizontal Axis Wind Turbines. *Frontiers of Mechanical Engineering in China*, 2(4):483 – 488, 1997.

[69] K. Deb. Nonlinear Goal Programming Using Multi-Objective Genetic Algorithms. *Journal of the Operational Research Society*, 52:291 – 302, 2001.

[70] K. C. Hall and S. R. Hall. A Variational Method for Computing the Optimal Aerodynamic Performance of Conventional and Compound Helicopters. *Journal of the American Helicopter Society*, 55(4), 2010.

[71] J. Lee and P. Hajelat. Parallel Genetic Algorithm Implementation in Multidisciplinary Rotor Blade Design. *Journal of Aircraft*, 33(5):962 – 969, September/October 1996.

[72] P. Rauch, M. Gervais, P. Cranga, A. Baud, J.-F. Hirsch, A. Walter, and P. Beaumier. Blue Edge: The Design, Development and Testing of a New Blade Concept. In *American Helicopter Society 67th Forum*, Virginia, USA.

[73] R. Fletcher and C. Reeves. Function Minimization by Conjugate Gradients. *British Computer Journal*, 7(2), 1964.

[74] M. Schwabacher, T. Ellman, and H. Hirsh. Learning to set up Numerical Optimizations of Engineering Designs. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 12:173 – 192, 1998.

[75] C.-M. Chen and H.-M. Lee. An Efficient Gradient Forecasting Search Method Utilizing the Discrete Difference Equation Prediction Model. *Applied Intelligence Journal*, 16:43 – 58, 2002.

[76] B. Mohammadi and O. Pironneau. Shape Optimization in Fluid Mechanics. *Annual Review of Fluid Mechanics*, 36:255 – 279, 2004.

[77] S. G. Lehner, L. B. Lurati, G. C. Bower, E. J. Cramer, W. A. Crossley, F. Engelsen, I. M. Kroo, S. C. Smith, and K. E. Willcox. Advanced Multidisciplinary Optimization Techniques for Efficient Subsonic Aircraft Design. In *48^{th} AIAA Meeting Including the New Horizons Forum and Aerospace Exposition*, volume AIAA 2010-1321, Orlando, Florida, USA, January 2010.

[78] G. S. Dulikravich, T. J. Martin, B. H. Dennis, and N. F. Foster. Multidisciplinary Hybrid Constrained GA Optimization. In *EUROGEN99 - Evolutionary Algorithms in Engineering and Computer Science: Recent Advances and Industrial Applications*, May 1999.

[79] J. M. Badyrka, R. M. Jenkins, and R. J. Hartfield. Aerospace Design: A Comparitive Study of Optimizers. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, volume AIAA 2010-1311, Orlando, Florida, USA, January 2010.

[80] S. K. Mishra. Global Optimization By Particle Swarm Method: A Fortran Program. North-Eastern Hill University (NEHU), August 2006.

[81] M. B. Azaba and C. Ollivier-Goochb. High Order Aerodynamic Optimization Using New Hybrid Sequential Quadratic Programing-Particle Swarm Intelligence Technique. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, volume 2012-0730, Nashville, Tennessee, USA, January 2012.

[82] V. V. Vytla, P. G. Huang, and R. C. Penmetsa. Response Surface Based Aerodynamic Shape Optimization of High Speed Train Nose. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, volume AIAA 2010-1509, Orlando, Florida, USA, January 2010.

[83] C. Poloni, A. Giurgevich, L. Onesti, and V. Pediroda. Hybridization of a Multi-Objective Genetic Algorithm, a Neural Network and a Classical Optimizer for a Complex Design Problem in Fluid Dynamics. *Computer Methods in Applied in Applied Mechanics and Engineering*, 186:403 – 420, 2000.

[84] J. Régnier, B. Sareni, and X. Roboam. System Optimization by Multi-Objective Genetic Algorithms and Analysis of the Coupling between Variables, Constraints and Objectives. *International Journal for Computation and Mathematics in Electrical and Electronics Engineering*, 24(3):805 – 820, 2005.

[85] R. Carrese, H. Winarto, and X. Li. Integrating User-Preference Swarm Algorithm and Surrogate Modeling for Airfoil Design. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, volume AIAA-244265-371, Orlando Florida, January 2011.

[86] M. J. Daskilewicz and B. J. German. Observations on the Topology of Pareto Frontiers with Implications for Design Decision Making. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, volume AIAA 2012-0148, Nashville, Tennessee, USA, January 2012.

[87] J. Sobieszczanski-Sobieski and R. T. Haftka. Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments. *Structural Optimization*, 14:1–23, 1997.

[88] D. Whitley. A Genetic Algorithm Tutorial. Colarado State University.

[89] F. Gray. Pulse Code Communication. U.S. Patent 2632058, filed: Nov 1947, March 1953.

[90] A. Toffolo and E. Benini. Genetic Diversity as an Objective in Multi-Objective Evolutionary Algorithms. *Evolutionary Computation Journal*, 11(2):151 – 167, 2003.

[91] E. Benini and M. Cenzon. Calibration of a Meanline Centrifugal Pump Model Using Evolutionary Algorithms. *Proc. IMechE Part A: Journal of Power and Energy*, 223:835 – 847, April 2009.

[92] G. Janiga. A Few Illustrative Examples of CFD-based Optimization - Heat Exchanger, Laminar Burner and Turbulence Modeling. University of Magdeburg, Germany.

[93] L. González, E. Whitney, K. Srinivas, and J. Périaux. Optimum Multidisciplinary and Multi-Objective Wing Design in CFD using Evolutionary Techniques, 2006.

[94] A. A. Hassan and B. D. Charles. Airfoil Design for Helicopter Rotor Blades - A Three-Dimensional Approach. *Journal of Aircraft*, 34(2):197 – 205, April 1997.

[95] A. Jameson, L. Martinelli, and N. A. Pierce. Optimum Aerodynamic using the Navier-Stokes Equations. *Theoretical and Computational Fluid Dynamics*, 10:213 – 237, 1998.

[96] K. Mani, B. A. Lockwood, and D. J. Mavriplis. Adjoint-based Unsteady Airfoil Design Optimization with Application to Dynamic Stall. In *American Helicopter Society 68th Annual Forum*, Fort Worth, Texas, USA, May 2012.

[97] E. J. Nielsen, E. M. Lee-Rausch, and W. T. Jones. Adjoint-Based Design of Rotors Using the Navier-Stokes Equations in a Noninertial Reference Frame. In *American Helicopter Society 65th Forum*, Grapevine, Texas, USA, May 2009.

[98] X. Liu, Y. Chen, and Z. Ye. Optimization Model for Rotor Blades of Horizontal Axis Wind Turbines. *Frontiers of Mechanical Engineering in China Journal*, 2(4):483 – 488, 2007.

[99] S. G. Lekoudis and L. N. Sankar. A Method for Designing Three-Dimensional Configurations with Prescribed Skin Friction. *Journal of Aircraft*, 21(11), 1984.

[100] T. W. Drayna, G. V. Candler, and H. B. Johnson. A Nonlinear Sensitivity Solver for Aerodynamic Optimization and Rapid Design Space Exploration. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, volume AIAA 2011-26, Orlando Florida, January 2011.

[101] A. Garavello, R. Ponza, and E. Benini. Multi-Objective Aerodynamic Design of Tilt-Rotor Airframe Components by means of Genetic Algorithms and CFD. In *American Helicopter Society 68th Annual Forum*, Fort Worth, Texas, USA, May 2012.

[102] J. H. Sa, Y.-H. You, J.-S. Park, S. N. Jung, S. H. Park, and Y. H. Yu. Assessment of CFD/CSD Coupled Aeroelastic Analysis Solution for HARTII Rotor Incorporating Fuselage Effects. In *American Helicopter Society 67th Forum*, Virginia, USA, May 2011.

[103] C. A. Smith and M. D. Betzina. A Study of the Aerodynamic Interaction between a Main Rotor and a Fuselage. In *39th American Helicopter Society Forum*, St.Louis, Mo., USA, May 1983.

[104] J. D. Berry, V. Letnikov, I. Bavykina, and M. S. Chaffin. A Comparison of Interactional Aerodynamics Methods for a Helicopter in Low Speed Flight. *NASA Technical Memorandum*, NASA TM-1998-208420, AFDD TR-98-A-003, June 1998.

[105] J. D. Berry and S. L. Althoff. Computing Induced Velocity Pertubations Due to a Helicopter Fuselage in a Fresstream. *NASA Technical Memorandum 4113*, TR 89-B-001, 1989.

[106] R. E. Mineck and S. A. Gorton. Steady and Periodic Pressure Measurements on a Generic Helicopter Fuselage Model in the Presence of a Rotor. TM 2000-210286, June 2000.

[107] Y. Tanabe, S. Saito, N. Kobiki, K. Murota, H. Sugawara, K. Hayashi, and K. Hiraoka. An Experimental Study of Rotor/Fuselage Interation. In *35th European Rotorcraft Forum*, 2009.

[108] J. W. Lim, C. Tung, Y. H. Yu, C. L. Burley, T. Brooks, D. Boyd, B. van der Wall, O. Schneider, H. Richard, M. Raffel, P. Beaumier, J. Bailey, Y. Delrieux, K. Pengel, and E. Mercker. HART-II: Prediction of Blade-Vortex Interaction Loading.

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

[110] J. Hájek. Parameterization of Airfoils and its Application in Aerodynamic Optimization. Charles University, Prague, Czech Republic.

[111] P. Castonguay and S. K. Nadarajah. Effect of Shape Parameterization on Aerodynamic Shape Optimisation. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, USA, January 2007.

[112] W. S. Ghaly and T. T. Mengistu. Optimal Geometric Representation of Turbomachinery Cascades using NURBS. *Inverse Problems in Science and Engineering*, 11(5):359 – 373, October 2003.

[113] P. J. Schneider. NURB Curves: A guide for the Uninitiated. *Develop, the Apple Technical Journal*, (25), 2008.

[114] C.-Y. Joh. Development of a NURBS-based Wing Design Optimization System. *IEEE Explore*, 2008.

[115] B. M. Kulfan and J. E. Busoletti. Fundamental Parametric Geometry Shape Representations for Aircraft Component Shapes. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, volume AIAA 2006-6948, Portsmouth, Virginia, September 2006.

[116] D. P. Young. Technical Notes: Note on Parameterization of Airfoils. *AIAA Journal*, 49(1), January 2011.

[117] S. Osher and S. Chakravarthy. Upwind Schemes and Boundary Conditions with Applications to Euler Equations in General Geometries. *Journal of Computational Physics*, 50:447 – 481, January - February 1983.

[118] B. van Leer. Flux-vector splitting for the euler equations. In *Eighth International Conference on Numerical Methods in Fluid Dynamics*, volume 170 of Lecture Notes in Physics, pages 507 – 512, Berlin / Heidelberg, 1982.

[119] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, Cambridge, MA, 1994.

[120] F. R. Menter. Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications. *AIAA Journal*, 32(8):1598 – 1605, August 1994.

[121] M. Woodgate and G. N. Barakos. Implicit CFD Methods for Fast Analysis of Rotor Flows. In *36th European Rotorcraft Forum*, volume ERF Paper 14, Session 1 [CDROM], Paris, France, September 2009.

[122] M. A. Woodgate and K. J. Badcock. Implicit Harmonic Balance Solver for Transonic Flow with Forced Motions. *AIAA Journal*, 47(4):893 – 901, April 2009.

[123] J.-S. Jang, S. Choi, H.-I. Kwon, D.-K. Im, D.-J. Lee, and J.-H. Kwon. A Preliminary Study of Open Rotor Design Using a Harmonic Balance Method. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, volume AIAA 2012-1042, Nashville, Tenessee, USA, January 2012.

[124] ANSYS Inc., ICEM-CFD Hexa Mesh Generation Software. http://www.ansys.com/products/icemcfd.asp.

[125] R. Steijl, G. Barakos, and K. Badcock. A Framework for CFD Analysis of Helicopter Rotors in Hover and Forward Flight. *International Journal for Numerical Methods in Fluids*, 51:819 – 847, 2006.

[126] J. Jeong and F. Hussain. On the Identification of a Vortex. *Journal of Fluid Mechanics*, 285:69 – 94, 1995.

[127] G. Haller. An Objective Definition for a Vortex. *Journal of Fluid Mechanics*, 525:1 – 26, 2005.

[128] P. L. Chebyshev. Mémoir sur les nombres premiers. *Journal de Mathématiques Pures et Appliquées*, 17, 1852.

[129] C. S. Johnson. Parametric Mesh Generation for HMB. *HMB Technical Notes*, TN10-013-Fuselage, August 2010.

[130] T. W. Simpson, J. D. Peplinski, P. N. Koch, and J. K. Allen. Metamodels for Computer-based Engineering Design: Survey and recommendations. *Engineering with Computers Journal*, 17:129 – 150, 2001.

[131] A. Forrester, A. Sóbester, and A. Keane. *Engineering Design via Surrogate Modelling - a Practical Guide*. July 2008.

[132] S. Samarasinghe. *Neural Networks for Applied Sciences and Engineering - From Fundamentals to Complex Pattern Recognition*. Auerbach Publications, New York, 2006.

[133] J. H. Mathews and K. D. Fink. *Numerical Methods Using MATLAB*. Pearson Education, Inc., 2004.

[134] P. Mantovan and P. Secchi. *Complex Data Modeling and Computationally Intensive Statistical Methods*. Springer-Verlag, Italy, 2010.

[135] S. J. Lawson. Parallel Performance of Library Algorithms for Computational Engineering. University of Edinburgh and Liverpool, 2007.

[136] R. Everson and L. Sirovich. The Karhunen-Loève Procedure for Gappy Data, November 1994.

[137] M. Joyner. *An Application of a Reduced Order Computational Methodology for Eddy Current Based Non-destructive Evaluation Techniques*. PhD thesis, North Carolina State University, June 2001.

[138] H. V. Ly and H. T. Tran. Proper Orthogonal Decomposition for Flow Calculations and Optimal Control in a Horizontal CVD Reactor. *CRSC Tech. Rep. CRSC-TR98-13*, 1998.

[139] S. J. Lawson. Technical Note. *HMB Technical Notes*, TN05-014, June 2010.

[140] C. S. Johnson. Metamodel and Optimisation Procedure for HMB. *HMB Technical Notes*, TN10-022-Optimisation, November 2010.

[141] P. H. Cook, M. A. McDonald, and M. C. P. Firmin. Aerofoil RAE 2822 - Pressure Distributions, and Boundary Layer and Wake Measurements. *Experimental Data Base for Computer Program Assessment, AGARD Report AR 138*, 1979.

[142] W. H. Mason. *Configuration Aerodynamics Course*. Virginia Polytechnic Institute and State University, October 2006.

[143] F. A. C. Viana, R. T. Haftka, and L. T. Watson. Why not run the Efficient Global Optimization Algorithm with Multiple Surrogates? In *51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, volume AIAA 2010-3090, Orlando, Florida, USA, April 2010.

[144] A. Datta and I. Chopra. Validation of Structural and Aerodynamic Modeling Using UH-60A Flight Test Data. In *American Helicopter Society 59th Annual Forum*, Alexandria, VA, USA, May 2003.

[145] C. P. Coleman and W. G. Bousman. Aerodynamic Limitations of the UH-60A Rotor. *NASA Technical Memorandum*, 110396, August 1996.

[146] A. Filippone. Prediction of Aerodynamic Forces on a Helicopter Fuselage. *The Aeronautical Journal*, pages 175 – 184, March 2007.

[147] G. N. Barakos and R. Steijl. Computational Study of Helicopter Rotor-Fuselage Aerodynamic Interactions. *AIAA Journal*, 47(9), September 2009.

[148] C. S. Johnson. Actuator Disk Models in HMB. *HMB Technical Notes*, TN12-003, March 2012.

[149] K. Robinson and A. Brocklehurst. BERP IV - Aerodynamics, Performance and Flight Envelope. In *34th European Rotorcraft Forum*, Liverpool, UK, September 2008.

[150] F. J. Perry, P. G. Wilby, and A. F. Jones. The BERP Rotor - How does it work, and what has it beein doing lately. *Vertiflite*, 44(2):44–48, 1998.

[151] G. R. Srinivasan, V. Raghavan, E. P. N. Duque, and W. J. McCroskey. Flowfield Analysis of Modern Helicopter Rotors in Hover by Navier-Stokes Method. *Journal of the American Helicopter Society*, 38(3):3–13, July 1993.

[152] T. S. Beddoes. A 3-D Separation Model for Arbitrary Planforms. In *47th American Helicopter Society Forum*, pages 451–460, Phoenix, Arizona, USA, May 1991.

[153] C. S. Johnson. BERP tip Grid Generation Technique in ICEMCFD. *HMB Technical Notes*, TN10-BERPGrid, May 2011.

[154] C. Johnson and G. Barakos. A Framework for Optimising Aspects of Rotor Blades. *The Aeronautical Journal*, 115(1165):147 – 161, March 2011.

[155] Y. H. Yu, C. Tung, B. van der Wall, H.-J. Pausder, C. Burley, T. Brooks, P. Beaumier, Y. Delrieux, E. Mercker, and K. Pengel. The HART-II Test: Rotor Wakes and Aeroacoustics with Higher-Harmonic Pitch Control (HHC) Inputs - The Joint German/French/Dutch/US Project. In *American helicopter Society 58th Annual Forum*, Montreal, Canada, June 2002.

Appendix A

Fundamental Relationships for Rotorcraft Aeromechanics

A.1 Rotor

$$T = \dot{m}w \tag{A.1}$$

$$Tv_i = \frac{1}{2}\dot{m}w^2 = P \tag{A.2}$$

$$w = 2v_i \tag{A.3}$$

$$\frac{A_\infty}{A} = \frac{1}{2} \tag{A.4}$$

$$\tag{A.5}$$

Actually A is slightly bigger than that, because of viscous effects and a swirl component of velocity in the rotor wake induced by spinning blades($R_\infty = 0.78R$).

$$v_h = v_i = \sqrt{\left(\frac{T}{A}\right)\frac{1}{2\rho}} = \frac{P}{T} = \text{power loading} \tag{A.6}$$

$$P = Tv_h = \frac{T^{\frac{3}{2}}}{\sqrt{2\rho A}} = 2\rho A v_i^3 \tag{A.7}$$

pressure variations

$$\Delta p = p_2 - p_1 = \frac{T}{A} \tag{A.8}$$

$$\frac{T}{A} = p_2 - p_1 = \left(p_\infty + \frac{1}{2}\rho w^2 - \frac{1}{2}\rho v_i^2\right) - \left(p_\infty - \frac{1}{2}\rho v_i^2\right) = \frac{1}{2}\rho w^2 \tag{A.9}$$

If C_d is constant, for a rectangular blade, the profile power, ($\mathrm{P_o}$), is given by:

$$P_O = \frac{1}{8}\rho N_b \Omega^3 c C_d R^4 \tag{A.10}$$

A.1.1 Non-dimensionalisation

This is based on one dimensional flow so the value of inflow is assumed to be distributed uniformly over the disk. Since there are no viscous losses assumed, the power is the *ideal power* coefficient.

$$\lambda_h = \frac{v_i}{V_{tip}} = \sqrt{\frac{C_T}{2}} \tag{A.11}$$

$$C_P = \frac{P}{\frac{1}{2}\rho A V_{tip}^3} = \frac{C_T^{3/2}}{\sqrt{2}} \tag{A.12}$$

$$C_{P_O} = \frac{1}{8}\sigma C_d \text{ where } \sigma = \frac{N_b c}{\pi R} \tag{A.13}$$

This theory shows the correct trend, but underpredicts the actual power required. TO correct for this, a factor κ (the induced power correction factor) is included i.e. $C_P = \frac{\kappa C_T^{3/2}}{\sqrt{2}}$. It is found by experiment or more advanced blade element methods (usually about 1.15). It embodies non-uniform inflow, tip losses, wake swirl, finite no. of blades etc.
FM, Figure of merit is an additional way of determining the hover efficiency and is better in that it is non-dimensional unlike power loading.

$$FM = \frac{\text{Ideal power required to hover}}{\text{Actual power required to hover}} < 1 \tag{A.14}$$

Ideal power is the one calculated using the simple momentum theory above. i.e. no viscous effects (which add profile as well as some induced power). Actual is a measured value.
Note: FM is only comparative at the same disk loading because increasing the DL increases induced power relative to profile power.

$$FM = \frac{P_{ideal}}{\kappa P_{ideal} + P_{profile}} = \frac{\frac{C_T^{3/2}}{\sqrt{2}}}{\frac{\kappa C_T^{3/2}}{\sqrt{2}} + \frac{\sigma C_d}{8}} \tag{A.15}$$

Appendix B

Programs

B.1 MATLAB script to create POD modes and use them to interpolate

LISTING B.1: KLD_TEST_INTERPOLATEV.M

```
clear;
n=load('opfunctest_thick.txt');
[M,N] = size(n);

for i=1:N
    data(:,i) = n(:,i);
end

covar = (data*data')/M;                          % Calculate KLD of signals
[Vx,Sx] = eig(covar);
Vx = fliplr(Vx);
Ux = (data'*Vx);                                 % Ux is the spatial KLD modes, phi(x)
Vxt=Vx';                                         % matrix with eigen vectors contains the temporal
    eigenfunctions, a(t)
Sx = rot90(Sx);
Sx = fliplr(Sx);
for i=1:N
    svalues(i,1) = Sx(i,i);
end
eig_sum = sum(svalues);

modes=M;                                         % reconstruction
for i=1:modes
    Uxm(:,i) = Ux(:,i);
    Vxtm(i,:) = Vxt(i,:);
end
save U.txt Uxm -ASCII;
save V.txt Vxtm -ASCII;
rec = Uxm*Vxtm;
rec = rec';
save recreate.txt rec -ASCII;

%% Using averages
    for j=1:4
        for i=1:4
            Vxtm(i,M+j)=((1-0.5)*Vxtm(i,42+j)+(0.5)*Vxtm(i,43+j))/1;
        end
    end

inte=Uxm*Vxtm;
inte=inte';
save inte.txt inte -ASCII;

% %% Using polynomial interpolation
% Firstc=n(:,1);
% lastc=n(:,N);
% for i=1:M
%     FC(i,:)=[Firstc(i)^0 Firstc(i)^1 Firstc(i)^2 Firstc(i)^3];
% end
%
%     coeff=linsolve(FC,lastc);
%
% pred=3.5;
% predictthis=[pred^0 pred^1 pred^2 pred^3];
% for i=1:M
%     answer(i)=coeff(i,1)*predictthis(i);
% end
% answ=sum(answer);
```

B.2 MATLAB scripts for the Gappy POD method

B.2.1 Main program

LISTING B.2.1: GAP_POD.M

```
clear;
%thick
     =[9;9.45;9.9;10.35;10.8;11.25;11.7;12;12.6;13.05;13.5;13.95;14.4;15;15.3;15.75;16.2;16.65;17.09;17.55;18];

%A=load('R09T12_loads_incomplete.txt');
%N=load('R09T12_loads_predict.txt');
%A=load('../incomplete33.txt');
%N=load('../predict33.txt');
A=load('../incomplete.den.txt');
N=load('../predict.den.txt');
%A=load('matrix.txt');
%N=load('matrix_pred.txt');
%A=A';
%N=N';
cl=3; %required column
%masc=[1;1;0;0];
masc=[1;0;0;0;1;0;0;0;1;0;0;0;1;0;0;0;0;0;0;0;1]; % masc vector
%for iii=1:1
%    masc(iii)=1;
%end
%for iii=2:21
%    masc(iii)=0;
%end
%masc=masc';
[U,e,mean,nb]=POD(A,99);
for k=1:1
[UU,ee,MEAN,nbnb]=POD(N,99);% if you use mean, only 1 iteration is req
%masc=[1;1;1;1;1;1;1;1;0;1;1;1;1;1;1;1;0;0;0];
%G=GPOD(mean,U,nb,N,masc,cl);
G=GPOD_original(mean,U,nb,N,masc,cl);
N(:,cl)=G;
k;
end
% something is still not right - there must be a way of using more data in
% the A matrix without adding guessed values at nodes in the matrix where
% data is not available. This is when MEAN would be used instead of mean -
% but at this point, either can be used to obtain the same result, the only
% difference being the number of iterations required to obtain it.
```

B.2.2 POD function

LISTING B.2.2: POD.M

```
function [phi, lam, Xmean, nbasis]=POD(X,cenergy)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function calculates the POD basis vectors
%
% Function inputs:
% X       :  (n x nsnap) snapshot matrix
%            n = number of states in full space
%            nsnap = number of snapshots
% cenergy :  how much energy of the ensemble you want
%            to capture, i.e. cenergy = 99.9 (percent)
%
% Function outputs:
% phi     :   (n x nsnap) matrix containing POD basis vectors
% lam     :   (nsnap x 1) vector containing POD eigenvalues
% Xmean   :   (n x 1) the mean of the ensemble X
% nbasis  :   number of POD basis vectors you should use
%             to capture cenergy (percent) of the ensemble
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% calculate the mean of the snapshots
nsnap = size(X,2);
Xmean = sum(X,2)/nsnap;
% Obtain new snapshot ensemble with zero mean
for i=1:nsnap
    X1(:,i) = X(:,i)-Xmean;
end
%   METHOD OF SNAPSHOTS
% calculate the empirical correlation matrix C
C = X1'*X1/nsnap;
% Calculate the POD basis
[evectorC,evalueC] = eig(C);
phi = X1 * evectorC;
% Normalize the POD basis
for i=1:nsnap
    phi(:,i) = phi(:,i)/norm(phi(:,i),2);
end
% return the POD eigenvalues in a vector
lam = diag(evalueC);
% Rearrange POD eigenvalues, vectors in descending order.
% Note that the correlation matrix C is symmetric, so SVD and EIG
% will give the same evectorC and evalueC but they are already in
% descending order and hence we don't need to rearrange evectorC and evalueC
% if SVD is used
lam = rot90(lam,2);
phi = fliplr(phi);

%%%       Find the number of POD basis vectors capturing cenergy (percent) of energy   %%%%

% total energy
tenergy = sum(lam);
energy = 0.;
nbasis = 0;
i = 1;
while (((energy/tenergy)*100) < cenergy)
    energy = energy + lam(i);
    i = i+1;
```

```
end
nbasis = i;
for i=1:nbasis
    PHI(:,i)=phi(:,i);
end
phi=PHI;
% plot eigenvalues corresponding to nbasis vectors
%plot(lam(1:nbasis)/tenergy/100,'*')
%xlabel('Number of POD eigenvalues')
%ylabel('Energy captured')
%grid on
```

B.2.3 Gappy POD function

LISTING B.2.3: GPOD.M

```
function [g_repaired]=GPOD(Xmean,phi,nbasis,g,mask,col)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function use the Gappy POD method to repair the gappy
% data g
%
% Function inputs:
% Xmean      :  (n x 1) ensemble mean, obtained from POD.m function
% phi        :  (n x nsnap) POD basis, obtained from POD.m function
%               n = number of states in full space
%               nsnap = number of snapshots
% nbasis     : number of POD basis vectors we use (usually captures 99.9% of energy),
%              obtained from POD.m function
% g          : (n x 1) gappy vector g needs to be repaired
% mask       : (n x 1) mask vector corresponding to the gappy vector g
%
% Function outputs:
% g_repaired:  (n x 1) the result after reparing gappy vector g
%
% All the variable names are very close to what we presented in the references
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m=size(g,2);
for k=1:m
% The size of the problem
n=size(g,1);
% subtracted by the ensemble mean
g(:,col)=g(:,k);%-Xmean;
% gappy vector with mean subtracted
g(:,k)=mask.*g(:,k);

% GAPPY POD METHOD
% Find all the coefficients M_ij = (phi_i,phi_j) of the system Mb = f
% using the gappy inner product
for i = 1:nbasis
    for j = 1:nbasis
        M(i,j) = (mask.*phi(:,i))'*phi(:,j);
    end
    % Find the right hand side f of the system Mb = f
    % Note that the gappy inner product is absorbed in g
    % since g = mask.*g
    gt=g';
    f(i,k) = gt(k,:)*phi(:,i);
end
end
M;
det(M);
f;
% solve for the POD coefficients, note that the size of the system
% is very small = number of POD basis vectors used
b = M\f ;
for k=1:m
% Finding the intermediate repaired vector g_tidle via POD expansion
g_tidle = phi*b;
% The gappy vector with mean subtracted is now repaired. Note that
% in the presence of noise, we should not perform this repair but
% use g_tidle instead since it is better.
%for j=1:n
%    if mask(j,1)==0
        g(j,k)=g_tidle(j,1);
%    end
%end
% The final repaired vector
g_repaired=g(:,k);%+Xmean;
end
% Here the whole matrix is used as opposed to GPOD_original.m where only
% the row where a prediction is required is used.
```

B.3 Artificial Neural Network code

B.3.1 Readme file

Normalise the data first by running normatest.f
2 files are produced. the second file to which you write to with a user-defined filename will be produced with the normal- ised data if you set segmentation line to 0 and the output on screen shows the min and max of all the inputs and outputs. In addition a file called norma.dat is created containing the number of columns, the max and min of the original file and of the normalised file. This file is used for denormalising.

****
This program nntest.f, performs the training of a neural network
ilay is the maximum number of layers that can be implemented with this program
imax1, *imax2* limits the size of the matrices
inp is a matrix that contains the patterns and its size is limited by *imax1* and *imax2*
Therefore *imax1* must be > the number of patterns
And *imax2* must be > the number of neurons i.e. *numhidden*
numinput is the number of inputs
numoutput is the number of outputs
numpatterns is the number of patterns
nml is the number of layers not including input and output
numhidden is the number of hidden neurons in each layer
weightIH is a matrix containing the weights of the input to hidden layer neurons (*imax2* x *imax2*)
weightHH is a matrix containing the weights of the hidden to the next hidden layer (*imax2* x *imax2* for each layer)
weightHO is a matrix containing the weights of the hidden to the output layer (*imax2* x *imax2*)
deltaweight for each of these is the change in the corresponding weight matrix due to a change in the output which occurs in the back propagation
An input file is required that contains data such as those below:
- patfile is the pattern file i.e. the training data
the number of columns in the patfile = no. of inputs + no. of outputs
This file should be a normalized version of the original file (normalized using program norma.f)
- istart determines if the training should be started afresh (hot start*see note at end) or continue from a previous run (cold start)
- epoch is the number of times the patterns are run through the network
- alpha is the learning momentum
- eta is the learning rate
where the new weight at time n is given by eta x previous output + alpha x weight at time n-1
- ifresh is the refresh rate. It is the number of epochs after which a file is written
- cerror is the error it is trying to converge to
- wfile is a file to write in the final values of the weights etc.
perceptron is the hidden layers. The number of weights in the perceptron is the number of connections between hidden layers not including the weights from input to hidden and hidden to output.
There is a loop containing all the patterns nested in a loop for all the epochs
To initialise, a matrix WEIGHTS.MTX is created that contains random values for the weights
The first line contains the epoch number and the *error0*
The second line contains the number of inputs, layers, hidden neurons and outputs
The lines after that contain the values for the weights in the matrices *weightIH,HH,HO*
These weights are then read from these matrices and the number of weights is calculated
The patfile is then read into a matrix input for the inputs and target for the outputs and the number of patterns are counted
The matrix 'list' contains an integer for each pattern (a count).
This 'list' matrix is mixed up randomly for each epoch and hence the patterns are introduced randomly.
This prevents the ANN from memorising, increases the learning rate and improves its capabilities of predicting unknown points.
The values are then propagated forward from IH to HH, HH to nHH and HH to HO where matrices sum hold the final value as it moves through the ANN.
Each loop uses the sigmoid transfer function to get across the neuron to its output (which is the input of the synaptic gap)
In the last loop from the HH to the HO, the error is calculated and the pattern with the worst error is denoted i1.

This is printed on screen as the ANN is trained.

Then backpropagation is performed and the weights re-adjusted as the loops are carried out from output to input to obtain the change in weight. The weights themselves are then adjusted using previous weights.
For each epoch, the outputs on screen are the epoch number, *error1, errormax* and *i1.*

The final values of weights, epochs etc. are put in a file with the name as the last row of the input file.

error 0 is the very first error i.e. of epoch 1. error is calculated for each pattern and *error1* is calculated as *error/error0.*
This *error1* is what must converge to *cerror. errormax* is the maximum error at a particular pattern called *i1.*

****

To predict the output, run the predict file, ensuring the number of inputs, outputs and layers are correct.
Enter the file and read the output in the file predict.pat.

****

Denormalise the predicted file by running the program denormastest.f
edit if necessary, to get appropriate tecplot file

IMPORTANT: If you require extrapolated data, make sure that when the data is normalized, the extrapolated points are included.
In the case of training, include them in normlalisation and then delete them from the normalized file before beginning training.
Then re-introduce them when predicting and then denormalise.

Note: For hot starting, change epoch final number to required value in the testfile or wfile.
Also, the more number of layers and neurons the more discontinuities in the drop in error during training.
The more number outputs, the more uneven the predictions are.

B.3.2 Normalising the inputs

This is required so that the sensitivity of the sigmoid activation function to the input does not diminish with large input values. See Section 3.5.1 of the thesis.
LISTING B.3.2: NORMATEST.F

```
      program cg2tec2
      character*30 filein, fileout1,fileout2
      real*8 value(200000,20),mx(20),mn(20)
      real*8 maxi(20),mini(20),lamda,beta
      integer itime, ntime
      integer answer
69    format(20(1X,f15.10))
70    format(20(1X,E16.8))

      print*,'do you want to use previously created norm.dat file?'
      read*,answer
      print*,'enter file to read data from'
      read*,filein
      if (answer.eq.0) then
            print*,'enter number of columns'
            read*,icols
      endif

c     print*,'enter max and min for each column'
      do i=1,icols
c           read*,mini(i),maxi(i)
            mini(i)=0.0000000001
            maxi(i)=0.9999999999
      enddo
      print*,'enter files to write data to'
      read*,fileout1,fileout2
c     read*,fileout1

      print*,'enter line number for segmentation of files'
      read*,nl
c     nl=100000000

      open(1,file=filein)
      do i=1,999999
            read(1,*,END=666)(value(i,j),j=1,icols)
```

```
        enddo
666     continue
        ilines=i-1
        close(1)
        print*,'INPUT FILE READ'

        do j=1,icols
                mn(j)= 99999.0
                mx(j)=-99999.0
        enddo

        do i=1,ilines
                do j=1,icols
                        mn(j)=dmin1(mn(j),value(i,j))
                        mx(j)=dmax1(mx(j),value(i,j))
                enddo
        enddo

        do i=1,icols
                print*,mn(i),mx(i)
        enddo

        do i=1,ilines
                do j=1,icols
                        rdif=mx(j)-mn(j)
                        if (rdif.eq.0.0) rdif=0.0000001
                        lamda=(maxi(j)-mini(j))/rdif
                        beta=mini(j)-lamda*mn(j)
                        value(i,j)=value(i,j)*lamda+beta
c                       value(i,j)=(value(i,j)-mn(j))/(mx(j)-mn(j))
                enddo
        enddo

        open(1,file='norma.dat')
        write(1,*)icols
        do i=1,icols
                write(1,*)mini(i),maxi(i)
        enddo
        do i=1,icols
                write(1,*)mn(i),mx(i)
        enddo
        close(1)

        open(1,file=fileout1)
        open(2,file=fileout2)
        do i=1,ilines
                if (i.le.nl) write(1,70)(value(i,j),j=1,icols)
                if (i.gt.nl) write(2,70)(value(i,j),j=1,icols)
        enddo
        close(1)
        close(2)
        print*,'OUTPUT FILEs WRITTEN'
        end
```

B.3.3 Training the ANN

The training is used to obtain the weights for the neurons to give accurate predictions of unknown outputs. See Section 3.5.1 of the thesis.

LISTING B.3.3: NNTEST_ADAPTIVE_ETA.F

```
        program neural_net
        parameter(imax1=650,imax2=25,ilay=25)
        integer epoch,istart
        integer i,j,k,p,np,list(imax1)
        integer numpattern,numinput,numhidden,numoutput,nml
        real*8 input(0:imax1,0:imax2),target(0:imax1,0:imax2)
        real*8 output(0:imax1,0:imax2)
        real*8 hidden(ilay,imax1,imax2)
        real*8 sum(0:imax1,0:imax2),summ(0:imax1)
        real*8 weightIH(0:imax2,0:imax2)
        real*8 weightHH(ilay,0:imax2,0:imax2)
        real*8 weightHO(0:imax2,0:imax2)
        real*8 deltaO(0:imax1),deltaH(ilay,0:imax1)
        real*8 deltaweightIH(0:imax1,0:imax2)
        real*8 deltaweightHO(0:imax1,0:imax2)
        real*8 deltaweightHH(ilay,0:imax1,0:imax2),inp(imax1,imax2)
        real*8 error,eta,alpha,smallwt,error1,errormax,r
        real*8 errorprev,delerror
        character*50 patfile
        character*40 wfile

11        format(60(f12.6,2X))
12        format(A50)
13        format(A1)

        smallwt=0.5
        iepoch=1
        error0=0.0

c       patfile is the pattern file i.e. the training data
        open(11,file='input')
        read(11,*)patfile
        read(11,*)istart
        read(11,*)numinput
        read(11,*)nml
        read(11,*)numhidden
        read(11,*)numoutput
        read(11,*)ifresh
```

```
      read(11,*)cerror
      read(11,*)eta
      read(11,*)alpha
      read(11,*)wfile
      close(11)

900   format (D8.3','D8.3)
950   format (D8.3','D8.3','D8.3)
      open(12,file='weights.mtx')
      write(12,*) '1    0.1'
      write(12,*)numinput,nml,numhidden,numoutput

c     intitialise input_to_hidden weights
      do j=1,numhidden
            do i=1,numinput
                  deltaweightIH(i,j)=0.0
                  weightIH(i,j)=2.0*(rand(0))*smallwt
                  write(12,900) weightIH
            enddo
      enddo
c      intitialise hidden_to_output weights
      do k=1,numoutput
            do j=1,numhidden
                  deltaweightHO(j,k)=0.0
                  weightHO(j,k)=2.0*(rand(0))*smallwt
                  write(12,900) weightHO
            enddo
      enddo
c      initialise hidden1 to hidden2 weights
      do il=1,nml
      do i=1,numhidden
            do j=1,numhidden
                  deltaweightHH(il,j,i)=0.0
                  weightHH(il,j,i)=2.0*(rand(0))*smallwt
                  write(12,950) weightHH
            enddo
      enddo
      enddo
      close(12)

      if (istart.eq.1) then
            write(*,*) '# Hot starting...'
            open(12,file=wfile)
            read(12,*)iepoch,error0
            read(12,*)numinput,nml,numhidden,numoutput
            do i=0,numinput
                  do j=0,numhidden
                        read(12,*)weightIH(i,j)
                  enddo
            enddo
            do i=0,numhidden
                  do k=0,numoutput
                        read(12,*)weightHO(i,k)
                  enddo
            enddo
            do il=1,nml
            do i=0,numhidden
                  do j=0,numhidden
                        read(12,*)weightHH(il,i,j)
                  enddo
            enddo
            enddo
            write(*,*) '# weight functions read'
            close(12)
      else
            write(*,*) '# Cold starting'
      endif

c      calculate number of weight functions
      norm1=(numinput*numhidden)+(nml-1)*(numhidden**2)+
     &      numhidden*numoutput
      write(*,*) '# ',norm1,' total weight components'
      write(*,*) '# ',(nml-1)*(numhidden*numhidden),
     &      ' weight components in perceptron'
c
      open(15,file=patfile)
      i=1
10    list(i)=i
         read(15,*,END=66)(inp(i,j), j=1,numinput+numoutput)
         do j=1,numinput
            input(i,j)=inp(i,j)
         enddo
         do j=1,numoutput
            target(i,j)=inp(i,j+numinput)
         enddo
      i=i+1
      goto 10
66    continue
      numpattern=i-1
      write(*,*) '# ',numpattern,' patterns'
      close(15)

      do epoch=iepoch,90000000
            do i=1,numpattern-1
                  number=int(rand(0)*float(2))
                  if (number.eq.1) then
                        iii=list(i)
                        list(i)=list(i+1)
                        list(i+1)=iii
                  endif
            enddo
            error=0.0
            errormax=-10.0e+6
```

```
                do np=1,numpattern
                        p=list(np)

c       propagate forward from input to hidden1
                        do j=1,numhidden
                                sum(p,j)=weightIH(0,j)
                                do i=1,numinput
                                    sum(p,j)=sum(p,j)+
     &                                      input(p,i)*weightIH(i,j)
                                enddo
                                hidden(1,p,j)=1.0/(1.0+exp(-sum(p,j)))
                        enddo

c       propagate forward from hidden-1 to hidden-n
                        do il=1,nml-1
                        do j=1,numhidden
                              sum(p,j)=weightHH(il,0,j)
                              do i=1,numhidden
                                 sum(p,j)=sum(p,j)+
     &                            hidden(il,p,i)*weightHH(il,i,j)
                              enddo
                              hidden(il+1,p,j)=1.0/(1.0+exp(-sum(p,j)))
                        enddo
                        enddo

c       propagate forward from hidden-n to output, calculate error & remedy per output node
                        do k=1,numoutput
                                sum(p,k)=weightHO(0,k)
                                do j=1,numhidden
                                    sum(p,k)=sum(p,k)+
     &                                hidden(nml,p,j)*weightHO(j,k)
                                enddo
                                output(p,k)=1.0/(1.0+exp(-sum(p,k)))
                                if (epoch.eq.1) error0=error
                                    r=target(p,k)-output(p,k)
                                    r=abs(r)
c                                   r=0.5*r*r
                                    error=error+r
                                    errormax=dmax1(errormax,r)
                                if (errormax.eq.r) il=p
                                deltaO(k)=(target(p,k)-output(p,k))*
     &                            (output(p,k)*(1.0-output(p,k)))
                        enddo
                        if (epoch.eq.1) error0=error
                        error1=error/error0


c       backpropagate from output to hidden-n, use output remedy to calculate remedy per hidden-n node
                        do j=1,numhidden
                                summ(j)=0.0
                                do k=1,numoutput
                                    summ(j)=summ(j)+
     &                                weightHO(j,k)*deltaO(k)
                                enddo
                                deltaH(nml,j)=summ(j)*hidden(nml,p,j)*
     &                            (1.0-hidden(nml,p,j))
                        enddo

c       backpropagate from hidden-n to hidden1, use hidden-n remedy to calculate remedy per hidden-n-1
     node
                        do il=nml-1,1,-1
                        do j=1,numhidden
                                summ(j)=0.0
                                do jk=1,numhidden
                                    summ(j)=summ(j)+
     &                                weightHH(il,j,jk)*deltaH(il+1,jk)
                                enddo
                                deltaH(il,j)=summ(j)*
     &                            hidden(il,p,j)*(1.0-hidden(il,p,j))
                        enddo
                        enddo

c       backpropagate from hidden1 to input & use hidden1 remedy to re-calculate i-h1 weights
                        do j=1,numhidden
                           deltaweightIH(0,j)=eta*deltaH(1,j)+
     &                                        alpha*deltaweightIH(0,j)
                           weightIH(0,j)=weightIH(0,j)+
     &                                   deltaweightIH(0,j)
                           do i=1,numinput
                              deltaweightIH(i,j)=
     &                           eta*input(p,i)*deltaH(1,j)+alpha*
     &                           deltaweightIH(i,j)
                              weightIH(i,j)=weightIH(i,j)+
     &                                      deltaweightIH(i,j)
                           enddo
                        enddo

c       propagate from hidden1 to hidden_n & use hidden2 remedy to recalculate h_n-h_n+1 weights
                        do il=1,nml-1
                        do j=1,numhidden
                                deltaweightHH(il,0,j)=eta*deltaH(il+1,j)
     &                           +alpha*deltaweightHH(il,0,j)
                                weightHH(il,0,j)=weightHH(il,0,j)+
     &                           deltaweightHH(il,0,j)
                                do i=1,numhidden
                                    deltaweightHH(il,i,j)=
     &                               eta*hidden(il,p,i)*deltaH(il+1,j)
     &                              +alpha*deltaweightHH(il,i,j)
                                    weightHH(il,i,j)=weightHH(il,i,j)+
     &                               deltaweightHH(il,i,j)
                                enddo
                        enddo
                        enddo
```

```
c         propagate from hidden3 to output & use output remedy to re-calculate h3-o weights
                        do k=1,numoutput
                                deltaweightHO(0,k)=
     &                          eta*deltaO(k)+alpha*deltaweightHO(0,k)
                                weightHO(0,k)=weightHO(0,k)+
     &                          deltaweightHO(0,k)
                                do j=1,numhidden
                                deltaweightHO(j,k)=eta*hidden(nml,p,j)*
     &                          deltaO(k)+alpha*deltaweightHO(j,k)
                                weightHO(j,k)=weightHO(j,k)+
     &                                       deltaweightHO(j,k)
                                enddo
                        enddo
                enddo
                if ((epoch/ifresh)*ifresh.eq.epoch) then
                        call weights(epoch,error0,numinput,nml,
     &          numhidden,numoutput,weightIH,weightHH,weightHO,wfile)
                      delerror=1
                      if (epoch.eq.1*ifresh) errorprev=error1
                      if (epoch.gt.1*ifresh) then
                          delerror=sqrt((error1*error1)/
     &                    (errorprev*errorprev))
                          if (delerror.lt.0.50.and.eta*1.05.
     &                    lt.1.and.error1.lt.errorprev) eta=eta*1.05
                          if (delerror.gt.1.05.and.eta*0.85.gt.0.01
     &                    .and.error1.gt.errorprev*1.05) eta=eta*0.85
                      endif
                      write(*,*) epoch,error1,errormax,i1,
     &                errorprev,delerror,eta
                      errorprev=error1
                endif
                if (error1.lt.cerror) goto 1000
        enddo
1000    continue
        call weights(epoch,error0,numinput,nml,numhidden,numoutput,
     &              weightIH,weightHH,weightHO,wfile)

        write(*,*) epoch,error1,errormax
        write(*,*) 'The End'
        stop
        end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine weights(epoch,error0,ninput,nl,nhidden,noutput,
     &            wweightIH,wweightHH,wweightHO,wwfile)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        parameter (imax3=25,ilay=25)
        integer epoch
        real*8 wweightHH(ilay,0:imax3,0:imax3)
        real*8 wweightHO(0:imax3,0:imax3),wweightIH(0:imax3,0:imax3)
        character*40 wwfile
11      format(6(f12.6,2X))

c       weights
        open(111,file=wwfile,status='UNKNOWN')
        write(111,*)epoch,error0
        write(111,*)ninput,nl,nhidden,noutput
        do i=0,ninput
                do j=0,nhidden
                        write(111,*)wweightIH(i,j)
                enddo
        enddo
        do i=0,nhidden
                do k=0,noutput
                        write(111,*)wweightHO(i,k)
                enddo
        enddo
        do il=1,nl
        do i=0,nhidden
                do j=0,nhidden
                        write(111,*)wweightHH(il,i,j)
                enddo
        enddo
        enddo
        close(111)
        return
        end
```

B.3.4 Predicting the outputs

The is used to predict the output for a set of input parameters. See Section 3.5.1 of the thesis.

LISTING B.3.4: PREDICTTEST.F

```
        program predict6
        parameter (imax3=99,ilay=35)
        real sumout(0:imax3),sumH(ilay,0:imax3)
        real inp(0:imax3),outp(0:imax3)
        real weightHH(ilay,0:imax3,0:imax3)
        real weightHO(0:imax3,0:imax3)
        real weightIH(0:imax3,0:imax3)
        character*32 filename1
        character*50 dummy
        character*1  reply
60      format(99(1X,F12.8))
666     format(A50)

        open(1,file='testfile')
        read(1,*)iepoch,error
```

```
      print*,'epoch is',iepoch
      read(1,*)numinput,nml,numhidden,numoutput
      do i=0,numinput
            do j=0,numhidden
                  read(1,*)weightIH(i,j)
            enddo
      enddo
      do i=0,numhidden
            do k=0,numoutput
                  read(1,*)weightHO(i,k)
            enddo
      enddo
      do il=1,nml
      do i=0,numhidden
            do j=0,numhidden
                  read(1,*)weightHH(il,i,j)
            enddo
      enddo
      enddo
      close(1)

      print*,'enter filename with input conditions
     &                  (output written to predict.pat)'
      read*,filename1


      open(1,file=filename1)
      open(2,file='predict.pat')
      do iii=0,9999999
            read(1,*,END=69)(inp(j),j=1,numinput)
c     calculate total signal to j-th neuron on hidden1 layer
            do j=1,numhidden
                  sumH(1,j)=weightIH(0,j)
                  do i=1,numinput
                        sumH(1,j)=sumH(1,j)+inp(i)*weightIH(i,j)
                  enddo
                  sumH(1,j)=1.0/(1.0+exp(-sumH(1,j)))
            enddo
c     calculate total signal to j-th neuron on hidden_n layer
            do il=1,nml-1
            do j1=1,numhidden
                  sumH(il+1,j1)=weightHH(il,0,j1)
                  do j2=1,numhidden
                        sumH(il+1,j1)=sumH(il+1,j1)+sumH(il,j2)*weightHH(il,j2,j1)
                  enddo
                  sumH(il+1,j1)=1.0/(1.0+exp(-sumH(il+1,j1)))
            enddo
            enddo
c     calculate total signal to k-th neuron on output layer
            do k=1,numoutput
                  sumout(k)=weightHO(0,k)
                  do j=1,numhidden
                        sumout(k)=sumout(k)+sumH(nml,j)*weightHO(j,k)
                  enddo
                  sumout(k)=1.0/(1.0+exp(-sumout(k)))
            enddo
            write(2,60)(inp(j),j=1,numinput),(sumout(k),k=1,numoutput)
      enddo
69    continue
      close(2)
      close(1)
      end
```

B.3.5 Denormalising the outputs

The denormalisation undoes the previous normalisation step. it denormalises the output after predictions are made. See Section 3.5.1 of the thesis.

LISTING B.3.5: DENORMATEST.F

```
      program cg2tec2
      character*30 filein, fileout
      character*70 dummy(3)
      integer reply
      real*8 value(999999,99),mx(99),mn(99)
      real*8 maxi(99),mini(99),lamda,beta
      integer itime, ntime
70    format(99(1X,E16.8))
77    format(A70)
222   format(D8.3' 'D8.3' 'D8.3)

      print*,'enter file to read data from'
      read*,filein
      print*,'is it a tecplot file? (1/0)'
      read*,reply
      print*,'enter number of lines per block'
      read*,nblock


      open(1,file='norma.dat')
      read(1,*)icols
      do i=1,icols
            read(1,*)mini(i),maxi(i)
      enddo
      do i=1,icols
            read(1,*)mn(i),mx(i)
      enddo
      close(1)
```

```
      i=1

      open(15,file=filein)
      do iii=1,10000000
            if (reply.eq.1)  then
                  read(15,77,END=66)dummy
                  do ib=1,nblock
                        read(15,*)(value(i,j),j=1,icols)
                        i=i+1
                  enddo
            else
                  read(15,*,END=66)(value(i,j),j=1,icols)
                  i=i+1
            endif
      enddo
66    continue
      ilines=i-1
      close(15)

      print*,'INPUT FILE READ'

      do i=1,ilines
            do j=1,icols
                  lamda=(maxi(j)-mini(j))/(mx(j)-mn(j))
                  beta=mini(j)-lamda*mn(j)
                  value(i,j)=(value(i,j)-beta)/lamda
            enddo
      enddo

      print*,'Do you want to make the output file a tecplot file?
     &  (1/0)'
      read*,reply

      open(11,file='denormed.dat')
      i=1
      do iii=1,909090909
            if (reply.eq.1)  then
                  write(11,*)'TITLE="Rotor Prediction"'
                  write(11,*)'VARIABLES="R" "Camber" "Thickness" "OpValue"'
                  write(11,*)'ZONE I=7, J=10, K=10, DATAPACKING=POINT'
                  do ib=1,nblock
                        write(11,70)(value(i,j),j=1,icols)
                        i=i+1
                  enddo
            else
                  write(11,*)(value(i,j),j=1,icols)
                  i=i+1
            endif
            if (i.eq.ilines+1) goto 777
      enddo
777   continue
      close(11)
      print*,'OUTPUT FILE WRITTEN'
      end
```

B.4 Kriging Metamodel

B.4.1 Main Fortran Program

```
      PROGRAM KRIG
c     ------------------------------------------------------------------
c     Kriging for 2 parameters
c     theta values are optimised such that psi is reduced. One theta is
c     kept constant (tht) and the other altered.
c     The values are normalised to between 0 and 1 first (suffix n).
c     Then the mean and standard deviation found and used to
c     normalise them (suffix z).
c     Prediction occurs for a set file of data.
c     Change in delta is the addition of a small value.
c     N is the total number of training data points.
c     ------------------------------------------------------------------

      INTEGER N
      PARAMETER(N=20)
      REAL A1(N),A2(N),ofv(N),A1z(N),A2z(N)
      REAL ofvz(N)
      REAL A1n(N),A2n(N),ofvn(N)
      REAL dist(N,N),distr(N),covk(N,N),CCC(N),lambda(N),F(N)
      REAL bb(N),FH(N),YH(N),CH(N,N)
      REAL meanA1,meanA2,meanV,sigma
      REAL pi,sA1,sA2,sV,prV,rA1,rA2
      REAL thA1,thA2,deter
      REAL det(2),work(N)
      REAL psii(N)
      REAL maxA1,minA1,maxA2,minA2,maxV,minV
      INTEGER aa(N), H, job

      pi = 3.141592654
      OPEN (10, FILE='training_data_CMdelta.txt', STATUS='OLD')

      DO J=1,N
         READ(10,*) A1(J),A2(J),ofv(J)
      ENDDO

c     *--- FIND MAX AND MIN OF ALL COLUMNS ----------------------------*
      DO J=1,N
         minA1=dmin1(minA1,A1(J))
         maxA1=dmax1(maxA1,A1(J))
         minA2=dmin1(minA2,A2(J))
         maxA2=dmax1(maxA2,A2(J))
         minV=dmin1(minV,ofv(J))
         maxV=dmax1(maxV,ofv(J))
      ENDDO

c     *--- NORMALISE DATA IN ALL COLUMNS ------------------------------*
      DO J=1,N
        A1n(J)=(A1(J)-minA1)/(maxA1-minA1)
        A2n(J)=(A2(J)-minA2)/(maxA2-minA2)
        ofvn(J)=(ofv(J)-minV)/(maxV-minV)
      ENDDO

c     *--- CALCULATE MEAN AND SD AND NORMALISE ------------------------*
      meanA1=SUM(A1n)/N
      meanA2=SUM(A2n)/N
      meanV=SUM(ofvn)/N
      sA1=sqrt(SUM((A1n-meanA1)**2)/(REAL(N-1)))
      sA2=sqrt(SUM((A2n-meanA2)**2)/(REAL(N-1)))
      sV=sqrt(SUM((ofvn-meanV)**2)/(REAL(N-1)))

      A1z=(A1n-meanA1)/sA1
      A2z=(A2n-meanA2)/sA2
      ofvz=(ofvn-meanV)/sV

c     *--- REQUIRED PARAMETER VALUES ----------------------------------*
c     File to read inputs for prediction from is predict_this.txt
c     File that predictions are going to be written to is
c     predicted_CMdeltapoly0.pat
      OPEN(12,file='predicted_CMdeltapoly0.pat')
      OPEN (11, FILE='predict_this.txt', STATUS='OLD')
      DO K=1,209
      print *,K
      READ(11,*) reqA1,reqA2
      rA1=reqA1
      rA2=reqA2
      reqA1=(((reqA1-minA1)/(maxA1-minA1))-meanA1)/sA1
      reqA2=(((reqA2-minA2)/(maxA2-minA2))-meanA2)/sA2

c     *--- INITIALISE THETA AND ALL VALUES ----------------------------*
      thA1=1.3167
      thA2=0.1
      delta=0.001
      psi=0
      beta=0
      beta2=0
      sigma=0
c     *--- DISTANCE BETWEEN ALL POINTS --------------------------------*
      DO I=1,N
      DO J=1,N
      dist(I,J)=(thA1*(A1z(J)-A1z(I))**2+thA2*(A2z(J)-A2z(I))**2)**(0.5)
c       print *,dist(I,J)
      ENDDO
      ENDDO

c     *--- DISTANCE BETWEEN REQUIRED AND ALL --------------------------*
      DO J=1,N
      distr(J)=(thA1*(A1z(J)-reqA1)**2+thA2*(A2z(J)-reqA2)**2)**(0.5)
```

```
      ENDDO

c     *--- COVARIANCE BETWEEN ALL POINTS - Gaussian ---------------------*
      DO I=1,N
c     initialise some matrices to be used later
       psii(I)=0
       lambda(I)=0
       F(I)=1
       FH(I)=0
       YH(I)=0
      DO J=1,N
       covk(I,J)=EXP(-(dist(I,J)**2))
      ENDDO
      ENDDO

c     *--- COVARIANCE BETWEEN REQUIRED AND ALL - Gaussian ---------------*
      DO I=1,N
       CCC(I)=EXP(-distr(I)**2)
c      print *,CCC(I)
      ENDDO

c     *--- CHOLESKY DECOMPOSITION AND INVERTED --------------------------*
      CH=covk
      CALL schdc(CH,1,N,work,0,0,info)
      CALL sgeco(CH,N,N,aa,H,bb)
      CALL sgedi(CH,N,N,aa,det,work,01)
      DO I=1,N
      DO J=1,N
      FH(I)=FH(I)+CH(I,J)*F(J)
      YH(I)=YH(I)+CH(I,J)*ofvz(J)
      ENDDO
      beta=beta+FH(I)*FH(I)
      beta2=beta2+FH(I)*YH(I)
      ENDDO
      beta=beta2/beta
c      print *,'beta',beta

c     *--- CALCULATE INVERSE OF COVARIANCE ------------------------------*
      CALL sgeco(covk,N,N,aa,H,bb)
      CALL sgedi(covk,N,N,aa,det,work,11)
      deter=det(1)*10**det(2)
      DO I=1,N
      DO J=1,N
       psii(I)=psii(I)+(ofvz(J)-F(J)*beta)*covk(I,J)
      ENDDO
       sigma=sigma+(psii(I)*(ofvz(I)-F(I)*beta)/N)
      ENDDO
c     get psi
      psi=deter**(1/N) * sigma
c      psi=-0.5*(N*LOG(sigma)+LOG(deter))
c      print *,'deter is',deter
      print *,'psi is',psi,'prepsi is',prepsi
c      print *,'sigma is',sigma

       print *,'thA1',thA1,'thA2',thA2,'thA3',thA3

c      *--- CALCULATE LAMBDA ---------------------------------------------*
       DO I=1,N
       DO J=1,N
         lambda(I)=lambda(I)+CCC(J)*covk(J,I)
       ENDDO
       ENDDO

c     *--- PREDICT VALUE ------------------------------------------------*
      prV=0
      DO I=1,N
       prV=prV+lambda(I)*ofvz(I)
      ENDDO

c     *--- DENORMALISE --------------------------------------------------*
      prV=prV * sV + meanV
      prV=prV * (maxV-minV) + minV
      WRITE(12,*)rA1,rA2,prV
      ENDDO

c     *--- WRITE PREDICTION PARAMETERS TO A FILE ------------------------*
      OPEN(111,file='predictparameterCMdelta.txt',status='UNKNOWN')
      WRITE(111,*) minA1,maxA1,minA2,maxA2,minV,maxV,meanA1
     & ,meanA2,meanV,sA1,sA2,sV,thA1,thA2
      DO I=1,N
      DO J=1,N
      WRITE (111,*) covk(I,J)
      ENDDO
      ENDDO
      CLOSE(111)
      CLOSE(12)
      END
```

B.4.2 Fortran Subroutines and Run Script

The fortran subroutines to be used along with the main program are:

- sgedi.f
- sgeco.f
- sgefa.f
- sscal.f

- saxpy.f
- isamax.f
- sswap.f
- sasum.f
- sdot.f
- schdc.f

They are available through the *LINPACK* package from www.netlib.org.
The way to run the the kriging method is using the following script:

```
gfortran -c -g sgedi.f
gfortran -c -g sgeco.f
gfortran -c -g sgefa.f
gfortran -c -g sscal.f
gfortran -c -g saxpy.f
gfortran -c -g isamax.f
gfortran -c -g sswap.f
gfortran -c -g sasum.f
gfortran -c -g sdot.f
gfortran -c -g schdc.f
gfortran -c -g OFV_krig_zbeta_alltheta_testpoly0CMavg.f
gfortran -g -o krig OFV_krig_zbeta_alltheta_testpoly0CMavg.o sgedi.o sgefa.o sscal.o saxpy.o sswap.o
    isamax.o sasum.o sdot.o sgeco.o schdc.o
./krig
rm *.o
```

B.5 Polynomial interpolation

The polynomial interpolation technique creates a polynomial fit of a specified degree through a set of data. See Section 3.5.2 of the thesis.

LISTING B.5: POLYNOMIAL10_INTERPOLATION.M

```
clear;
    p=10; %number of thickness values
    q=10; %number of camber values
    matrixA=load('R09_loads_dCm_training_poly10.txt');
    for i=1:p%10
        thick(i)=matrixA(q*i,2);%(10*i,2);
        for j=1:q%10
            OP_per_thick(j,i)=matrixA((i-1)*q+j,3);%((i-1)*10+j,3);
            OP_per_camber=OP_per_thick';
        end
    end
    camber=matrixA(1:q,1);%(1:10,1);
    for i=1:q%10
        C(i,:)=[camber(i)^0 camber(i)^1 camber(i)^2 camber(i)^3 camber(i)^4 camber(i)^5 camber(i)^6
            camber(i)^7 camber(i)^8 camber(i)^9];
    end
    for i=1:p
        T(i,:)=[thick(i)^0 thick(i)^1 thick(i)^2 thick(i)^3 thick(i)^4 thick(i)^5 thick(i)^6 thick(i)^7
            thick(i)^8 thick(i)^9];
    end
    for i=1:p%10
        prec(:,i)=linsolve(C,OP_per_thick(:,i));
    end
    for i=1:q
        pret(:,i)=linsolve(T,OP_per_camber(:,i));
    end
    %prediction
    aerothick=12;
    aerocamber=[4.30000005784115734E-006;
   2.1499999959150000;
   4.2999999961299995;
   6.4500004263449995;
   8.5999999965599994;
   10.749999996774999;
   12.900000426989999;
   15.049999567204999;
   18.000014997500003;
   19.349999567634999;
   21.499999997850001;
   23.650000428064999;
   25.800000858279997;
   28.000008138499997;
   30.099999568710000;
   32.249999998924999;
   34.400000429140000;
   38.000002569499998;
   38.699999139570004;
   40.849999569784998;
   43.000000000000000];
for ip=1:21
    predict_thiscamber=[aerocamber(ip)^0 aerocamber(ip)^1 aerocamber(ip)^2 aerocamber(ip)^3 aerocamber(ip
        )^4 aerocamber(ip)^5 aerocamber(ip)^6 aerocamber(ip)^7 aerocamber(ip)^8 aerocamber(ip)^9];
    predict_thisthick=[aerothick^0 aerothick^1 aerothick^2 aerothick^3 aerothick^4 aerothick^5 aerothick
        ^6 aerothick^7 aerothick^8 aerothick^9];

    k=0;
    for i=1:q
        diff=camber(i)-aerocamber(ip);
        if diff >= 0
            k=k+1;
            diffcamb(k)=diff;
        end
    end
    closestcamberabove=min(diffcamb)+aerocamber(ip);
    kk=0;
    for i=1:p
        diff=thick(i)-aerothick;
        if diff >= 0
            kk=kk+1;
            diffthick(kk)=diff;
        end
    end
    closestthickabove=min(diffthick)+aerothick;

    highc=p-kk+1; %this is the one you change to the closest higher value to aerothick
    if highc==1
        lowc=highc;
    else
        lowc=highc-1;
    end
    hight=q-k+1; %this is the one you change to the closest higher value to aerocamber
    if hight==1
        lowt=hight;
    else
        lowt=hight-1;
    end
    c1value=predict_thiscamber*prec(:,lowc);
    c2value=predict_thiscamber*prec(:,highc);
    t1value=predict_thisthick*pret(:,lowt);
    t2value=predict_thisthick*pret(:,hight);
    cvalue=c2value-(thick(highc)-aerothick)/(thick(highc)-thick(lowc))*(c2value-c1value);
    tvalue=t2value-(camber(hight)-aerocamber(ip))/(camber(hight)-camber(lowt))*(t2value-t1value);
    value(ip,1)=aerocamber(ip);
    value(ip,2)=aerothick;
    value(ip,3)=(cvalue+tvalue)/2
```

```
end
```

B.6 Genetic Algorithm code

B.6.1 Genetic Algorithm in C

This program and its requirements are described in detail in the Technical Note.[140] To run the program, use the run script in Section B.6.2 where the number of generations can also be set. It requires a number of files that describe the inputs, constraints, objective function weights, the settings, the initial population and so on. The process is described in Section 3.6 of the thesis.

LISTING B.6.1: GA_TRIAL.C

```
/* GENETIC ALGORITHM
 *
 * Use the run script to run this program.
 *
 */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>

/*--- FUNCTION DEFINITIONS ---*/
float binaryfunc(float);                /* function for converting decimal to binary */
float bintodec(void);                   /* function for converting binary to decimal */

struct {                                /* structure for each chromosome */
        float in[12];                   /* inputs characteristic for an individual */
        float out[12];                  /* and output for that individual */
        float ofv;                      /* OFV or fitness value */
        int index;                      /* identity number */
        int norm;                       /* normalised fitness integer - for roulette */
        } grid[1000];

main(int argc, char *argv[])
{
int no_of_in, no_of_out, no_of_con;     /* number of inputs, outputs and constraints */
int no_of_hcon, no_of_scon;             /* number of hard and soft constraints */
FILE *norm;
printf("GA version 25\n\n");
/*--- FIND REQUIRED DATA ---*/
norm=fopen("inputparams.in","r");
fscanf(norm,"no. of inputs: %d\n",&no_of_in);
fscanf(norm,"no. of outputs: %d\n",&no_of_out);
fscanf(norm,"no. of hard const: %d\n",&no_of_hcon);
fscanf(norm,"no. of soft const: %d\n",&no_of_scon);
fclose(norm);
float VAR[no_of_in+no_of_out+1],value;  /* reads the loads from initial pool i.e. solver data */
float ngrid;                            /* reads the input values on the roulette wheel */
float emargin,threshold;                /* reads settings for elitsm margin and survival threshold
    fitness */
float ve, minve, maxve, max;            /* used to find max and minimum OF values for normalizing and
    elitism */
float maxi[no_of_in], maxi2[no_of_in];  /* selects two input values to cross over for each input variable
    */
float selectvar[no_of_in];              /* is the selected variable after mutation */
float elit[no_of_in];                   /* used to select elite variable */
float JJ, apre[no_of_in];               /* fortran functions predicted inputs */
float aout[no_of_out];                  /* fortran functions predicted outputs */
float a5a[no_of_in],b5b[no_of_out],c5c; /* used to compare inputs so individuals are not repeated in new
    gen file */
float con,cccc;                         /* constant value to add to obj func and final fitness value */
float weight[no_of_out],cns[no_of_con]; /* weights for objective function and constraints */
float mx[no_of_in],mn[no_of_in];        /* maximum and mimimum of inputs to normalise with */
int a, b, i, ii, j, jj, k, kk, countline, gen;  /* counters */
int counter;                            /* counters */
int normalizeto, gen_max;               /* normalize fitness values from 0 to ... and max
    iterations */
int rounum, bk, rouindex, rouli;        /* roulettewheel file info */
int select_upper, bin_upper;            /* upper limit of the random number generator */
int sel1pt, sel2pt, crpt, mupt, mut;    /* random selection points in roulette wheel and cross,
    mutation point */
int selgene[no_of_in];                  /* gene selection to crossover */
int con_num[no_of_con];                 /* constraint parameter number (starts from 0 - i.e. 1st
    output is 0) */
int con_rel[no_of_con];                 /* constraint relation number (greater than = 1, less
    than = 0)*/
int con_sh[no_of_con];                  /* soft or hard constraint, soft is 0, hard is 1 */
float con_w[no_of_con];                 /* soft constraint weight */
int bin[no_of_in][2][10];               /* for conversion back to decimal from binary */

FILE *selected, *input, *output, *denorm, *penalty, *roulette, *roulselect, *binread, *binwrite;
FILE *pool, *poolread1, *poolread, *poolelit, *onlyelit, *ofvweights, *constraint, *setting;
char filename[128],strch[50];

/*--- READ SETTINGS, CONSTRAINTS AND WEIGHTS FILES ---*/

setting = fopen("set","r");
fscanf(setting,"%f\n%d\n%d\n%f\n%f\n",&emargin,&normalizeto,&gen_max,&threshold)==4;
fclose(setting);
no_of_con=no_of_hcon+no_of_scon;
constraint = fopen("constraints.dat","r");
for(i=0;i<no_of_con;i++)
{fscanf(constraint,"%d %f %d %d %f\n",&con_num[i], &cns[i], &con_rel[i], &con_sh[i], &con_w[i])==5;}
fclose(constraint);
ofvweights = fopen("ofvweights.txt","r");
jj=0;
while((j = getc(ofvweights)) != EOF)
        {
```

```
                ungetc(j,ofvweights);
                fscanf(ofvweights,"%f\n",&weight[jj])==1;
                jj=jj+1;
        }
fclose(ofvweights);
con=0;
for(i=0;i<no_of_out;i++)
{con=con+weight[i];}
con=-1*con;                             // value to neutralise Objective function so reference value has
     a value of 0.
printf("con:%f\n",con);

/*--- READ INITIAL POPULATION FILE ---*/
for(i=0;i<no_of_in;i++)                 // initialise max and min values
{mx[i]=-1000; mn[i]=1000;}
jj=0;
strcpy(filename,argv[1]);
input = fopen(filename,"r");
while((j = getc(input)) != EOF )
        {
                ungetc(j,input);
                for(i=0;i<(no_of_in+no_of_out);i++)
                {
                        fscanf(input,"%f ",&VAR[i])==1;
                }
                fscanf(input,"%f ",&value)==1;
                grid[jj].ofv=value;
                for(i=0;i<no_of_in;i++)
                {
                        grid[jj].in[i]=VAR[i];
                        if(grid[jj].in[i]>=mx[i]){mx[i]=grid[jj].in[i];}
                        if(grid[jj].in[i]<=mn[i]){mn[i]=grid[jj].in[i];}
                }
                for(i=0;i<no_of_out;i++)
                {
                        grid[jj].out[i]=VAR[i+no_of_in];
                }
                grid[jj].index=jj;
                jj=jj+1;
        }
fclose(input);
printf("%d\n ", jj);

/*----- SELECTION - ROULETTE BASED -----*/
/*--- First normalize the fitness values ---*/
minve=10000;                            // initialize minimum fitness
maxve=-10000;                           // initialize maximum fitness
for(i=0;i<jj;i++)
        {
                ve=grid[i].ofv;
                if (ve<minve)
                {minve=ve;}
                if (ve>maxve)
                        {maxve=ve;
                                for(j=0;j<no_of_in;j++)
                                {elit[j]=grid[i].in[j];}
                        }
        }
printf("maxve = %f\n",maxve);
/*--- ELITISM - best aerofoils > certain fitness in the initial pool put into next generation i.e. pool.
     dat ---*/
strcpy(filename,argv[2]);
poolelit = fopen(filename,"w");
onlyelit = fopen("best.txt","w");
for(i=0;i<jj;i++)
        {
                if( (grid[i].ofv >= (maxve - emargin)) )
                {
                a=0;
                for(k=0;k<no_of_con;k++)         // account for hard constraints here
                {
                        if((con_rel[k]==1) &&(con_sh[k]==1) )
                        {if(grid[i].out[con_num[k]]>=cns[k]){a=a+1;}}
                        else if((con_rel[k]==0) &&(con_sh[k]==1) )
                        {if(grid[i].out[con_num[k]]<=cns[k]){a=a+1;}}
                }
                if(a==no_of_hcon)
                        {for(j=0;j<no_of_in;j++)
                        {fprintf(poolelit,"%f\t",grid[i].in[j]);
                        fprintf(onlyelit,"%f\t",grid[i].in[j]);}
                        for(j=0;j<no_of_out;j++)
                        {fprintf(poolelit,"%f ",grid[i].out[j]);
                        fprintf(onlyelit,"%f ",grid[i].out[j]);}
                        fprintf(poolelit,"%f\n",grid[i].ofv);
                        fprintf(onlyelit,"%f\n",grid[i].ofv);
                        }
                }
        }
fclose(poolelit);
fclose(onlyelit);
/*------------------------------ end of elitism ----------------------------------------*/

for(i=0; i<jj; i++)      // normalise to get number of repetitions in roulettewheel
        {
                ngrid=(grid[i].ofv - minve) / (maxve - minve) * normalizeto ;
                grid[i].norm = (int)ngrid;
        }
/*--- Put it on a roulette wheel which is the file roulettewheel.txt ---*/
countline=0;
roulette=fopen("roulettewheel.txt","w");
for(i=0;i<jj;i++)
        {
                rouindex = grid[i].index;
```

```
                rounum = grid[i].norm;
                for(k=0;k<rounum;k++)
                {
                        fprintf(roulette,"%d  \n",rouindex);
                        countline=countline+1;
                }
        }
fclose(roulette);
int roul[countline];            /* array going to contain roulette wheel */
roulselect=fopen("roulettewheel.txt","r");
for(a=0;a<countline;a++)
        {
                fscanf(roulselect,"%d\n",&rouli)==1;
                roul[a]=rouli;
        }
fclose(roulselect);

srand(time(NULL));

/*---------- REPEAT FOR GEN NUMBER OF RUNS TO POPULATE NEXT GEN ----------*/
for(gen=0;gen<gen_max;gen++)
{
/*--- SELECT 2 PARENTS TO CROSSOVER ---*/
select_upper=countline;
sel1pt=rand()%select_upper;
sel2pt=rand()%select_upper;
for(j=0;j<no_of_in;j++)
        {
        maxi[j]=grid[roul[sel1pt]].in[j];
        maxi2[j]=grid[roul[sel2pt]].in[j];
        }
printf("SELECTED: ");
for(j=0;j<no_of_in;j++)
{printf("var%d parent1 = %f, var%d parent2 = %f\t",j,maxi[j],j,maxi2[j]);}
printf("\n");

/*--- CROSSOVER ---*/
for(j=0;j<no_of_in;j++)
        {
        selgene[j]=rand()%no_of_in;
        if(selgene[j]==1)
                {
                max=maxi2[j];
                maxi2[j]=maxi[j];
                maxi[j]=max;
                }
        }

printf("CROSSED: ");
for(j=0;j<no_of_in;j++)
{printf("var%d child1 = %f, var%d child2 = %f\t",j,maxi[j],j,maxi2[j]);}
printf("\n");
/*--- normalise first between 0 and 30 so it fits in limited binary number---*/
for(j=0;j<no_of_in;j++)
{maxi[j]=(maxi[j]-mn[j])/(mx[j]-mn[j]) * 30;
maxi2[j]=(maxi2[j]-mn[j])/(mx[j]-mn[j]) * 30;}

/*--- CONVERSION TO BINARY CODING I.E. PHENOTYPE TO GENOTYPE ---*/
for(j=0;j<no_of_in;j++)
{bk=binaryfunc(maxi[j]);
binread=fopen("thebinary","r");
fscanf(binread,"%d %d %d %d %d %d %d %d %d %d",&bin[j][0][0],&bin[j][0][1],&bin[j][0][2],&bin[j][0][3],&
     bin[j][0][4],&bin[j][0][5], &bin[j][0][6],&bin[j][0][7],&bin[j][0][8],&bin[j][0][9])==10;
fclose(binread);
bk=binaryfunc(maxi2[j]);
binread=fopen("thebinary","r");
fscanf(binread,"%d %d %d %d %d %d %d %d %d %d",&bin[j][1][0],&bin[j][1][1],&bin[j][1][2],&bin[j][1][3],&
     bin[j][1][4],&bin[j][1][5], &bin[j][1][6], &bin[j][1][7], &bin[j][1][8],&bin[j][1][9])==10;
fclose(binread);
}

bin_upper=10;           /* upper bound for mutation point is max binary digits */
//printf("Offspring in binary(gene) format:\n");
for(k=0;k<no_of_in;k++)
{
//      printf(" variable %d: \n",k);
//      for(j=0;j<2;j++)
//      {
//      printf(" %d: ",j+1);
//              for(i=0;i<bin_upper;i++)
//              {
//              printf("%d",bin[k][j][i]);
//              }
//      printf("\n");
//      }
}
/*--- MUTATION ---*/
for(j=0;j<no_of_in;j++)
        {
        mupt=rand()%bin_upper;          /* random number generator < upper bound */
        mut=rand()%2;                   /* mutation bit i.e. replace with this bit (1 or 0) */
        bin[j][0][mupt]=bin[j][0][mut];
        bin[j][1][mupt]=bin[j][1][mut];
        }
//printf("mutated:\n");
//for(k=0;k<no_of_in;k++)
//{
//      printf("variable: %d\n",k);
//      for(j=0;j<2;j++)
//      {
//      printf(" %d: ",j+1);
//              for(i=0;i<bin_upper;i++)
//              {
```

```
//                printf("%d",bin[k][j][i]);
//                }
//        printf("\n");
//        }
//}
/*--- CONVERSION BACK TO PHENOTYPE I.E. DECIMAL ---*/
for(k=0;k<no_of_in;k++)
{
        binwrite=fopen("thebinary2","w");
        for(j=0;j<2;j++)
        {
                for(i=0;i<bin_upper;i++)
                {
                        fprintf(binwrite, "%d ",bin[k][j][i]);
                }
                fprintf(binwrite, "\n");
        }
        fclose(binwrite);
        selectvar[k]=bintodec();
}

/*--- denormalise ---*/
for(k=0;k<no_of_in;k++)
{selectvar[k]=selectvar[k] / 30 * (mx[k]-mn[k]) + mn[k];}

selected=fopen("neww.txt","w");
for(k=0;k<no_of_in;k++)
{fprintf(selected,"%f\t",selectvar[k]);
printf("final offspring: %f\t",selectvar[k]);}
fprintf(selected,"\n");
fclose(selected);

/*--- FINDING OPTIMIZATION FUNCTION VALUE USING TRAINED ANN ---*/

/*--- normalizing ---*/
normatest_(&no_of_in);
norm=fopen("fileout.txt","r");

/*--- predicting and denormalizing outputs ---*/
for(k=0;k<no_of_out;k++)
{
        kk=k+1;
        predicttest_(&kk);
        denormatest_(&kk);
        penalty=fopen("denormed.dat","r");
        for(j=0;j<no_of_in;j++)
        {fscanf(penalty,"%f ", &apre[j]);}
        fscanf(penalty,"%f\n",&aout[k]);
        fclose(penalty);
}

/*---------- PENALTY FOR VALUES OUTSIDE CONSTRAINTS ----------*/
cccc=con;
for(k=0;k<no_of_out;k++)
{
        cccc=cccc + weight[k] * aout[k];
}
for(k=0;k<no_of_in;k++)
        {if (apre[k] < mn[k]) {cccc=cccc - (10 * (mn[k] - apre[k]));}
        if (apre[k] > mx[k]) {cccc=cccc - (10 * (apre[k] - mx[k]));}}
for(k=0;k<no_of_con;k++)          // soft constraints
        {       if((con_rel[k]==1) && (con_sh[k]==0) )
                {if (aout[con_num[k]] < cns[k]) {cccc=cccc - (con_w[k] * (cns[k] - aout[con_num[k]]));}}
                if((con_rel[k]==0) && (con_sh[k]==0) )
                {if (aout[con_num[k]] > cns[k]) {cccc=cccc - (con_w[k] * (- cns[k] + aout[con_num[k]]))
                     ;}}
        }
for(k=0;k<no_of_out;k++)
        {printf("%f ",aout[k]);}
printf("%f\n",cccc);

/*--- ADD TO POOL WITHOUT REPETITION AND WITH PENALTY FACTOR ADDED ---*/
j=0;
ii=1;
counter=0;
strcpy(filename,argv[2]);
poolread = fopen(filename,"r+");
while((j < ii))
{
        i=0;
        ii=0;
        poolread1 = fopen(filename,"r");
        while((i = getc(poolread1)) != EOF)
                {ungetc(i,poolread1);
                for(i=0;i<no_of_in;i++)
                        {fscanf(poolread1,"%f ",&a5a[i])==1;}
                for(i=0;i<no_of_out;i++)
                        {fscanf(poolread1,"%f ",&b5b[i])==1;}
                fscanf(poolread1,"%f\n",&c5c)==1;
                ii=ii+1;}
        fclose(poolread1);
        for(i=0;i<no_of_in;i++)
                {fscanf(poolread,"%f ",&a5a[i])==1;}
        for(i=0;i<no_of_out;i++)
                        {fscanf(poolread,"%f ",&b5b[i])==1;}
        fscanf(poolread,"%f\n",&c5c)==1;
        i=0;
        k=0;
        while(i<no_of_in)
        {
                if (sqrt((a5a[i]-apre[i])*(a5a[i]-apre[i]))<=0.00001) {k=k+1;}
                i=i+1;
                if((i==no_of_in) && (k!=no_of_in)) {counter=counter+1;}
```

```
        }
        if (k==no_of_in) {break;}
        if(( counter == ii) && (cccc>threshold) )        // if whole file checked and above threshold...
        {
                a=0;kk=0;
                while(kk<no_of_con)                      // ... and hard constraints are met...
                {       if ((con_rel[kk]==1) && (con_sh[kk]==1))
                        {if(aout[con_num[kk]]>=cns[kk]) {a=a+1;}}
                        else if ((con_rel[kk]==0) && (con_sh[kk]==1))
                        {if(aout[con_num[kk]]<=cns[kk]) {a=a+1;}}
                        kk=kk+1;
                }
                if(a==no_of_hcon) // (hard constraints met)... add to next generation.
                {
                        for(i=0;i<no_of_in;i++)
                        {fprintf(poolread,"%f\t",apre[i]);
                        printf("%f\t",apre[i]);}
                        for(i=0;i<no_of_out;i++)
                        {fprintf(poolread,"%f ",aout[i]);
                        printf("%f ",aout[i]);}
                        fprintf(poolread,"%f\n",cccc);
                        printf("%f\n",cccc);
                }
        }
        j=j+1;
}
fclose(poolread);
}
return 0;
}

/*--- FUNCTION FOR THE CONVERSION TO BINARY - 6-digit before decimal point, 3-digit after decimal point
    ---*/
float binaryfunc(float p)
{
int div, rem, i, n, bin[7], bindec[3], bk, count;
float fr, r, frac, number;
FILE *binf;
i=0;
                fr = (int)p;
                div = p;
                while (div > 0)
                        {
                                fr = fr /2;
                                div = fr;
                                r = ( fr - div ) *2;
                                rem = r;
                                bin[6-i] =rem;
                                i=i+1;
                                fr = div;
                        }
                if (i<7)
                        {
                                n = 7-i;
                                while(n > 0)
                                {
                                        bin[n -1]=0;
                                        n = n -1;
                                }
                        }
bk=bin[4];

/* fraction part */
frac = p - (int)p;
number = frac;
count = 0;
while(number>0)
{
        number = number * 2;
        bindec[count] = (int)number;
        number = number - (int)number;
        count = count + 1;
        if (count==3) break;
}
if (count<3)
{
        n=count;
        while (n<3)
        {
                bindec[n]=0;
                n = n + 1;
        }
}

binf=fopen("thebinary","w");
fprintf(binf,"%d %d %d %d %d %d %d %d %d %d \n",bin[0],bin[1],bin[2],bin[3],bin[4],bin[5],bin[6],bindec
    [0],bindec[1],bindec[2]);
fclose(binf);
return bk;
}

/*--- FUNCTION FOR THE CONVERSION OF BINARY TO DECIMAL ---*/

float bintodec(void)
{
        FILE *binread, *output;
        float sum1, sum2;
        int i, val1, val2, bin[2][10];
        binread=fopen("thebinary2","r");
        for(i=0;i<2;i++)
        {
        fscanf(binread,"%d %d %d %d %d %d %d %d %d %d\n", &bin[i][0],&bin[i][1],&bin[i][2],&bin[i][3],&
            bin[i][4],&bin[i][5],&bin[i][6],&bin[i][7],&bin[i][8],&bin[i][9]);
```

```
        }
        fclose(binread);
        sum1=0;
        sum2=0;
        i=0;
        for (i=0;i<7;i++)
                {
                        val1 = bin[0][6-i];
                        sum1 = sum1 + val1 * pow(2,i);
                        val2 = bin[1][6-i];
                        sum2 = sum2 + val2 * pow(2,i);
                }
        i=0;
        for(i=0;i<3;i++)
                {
                        val1 = bin[0][9-i];
                        sum1 = sum1 + val1 * pow(0.5,i+1);
                        val2 = bin[1][9-i];
                        sum2 = sum2 + val2 * pow(0.5,i+1);
                }
        output = fopen("new.txt","w");
        fprintf(output,"%f %f \n",sum2, sum1);
        fclose(output);
        return sum1;
}
```

B.6.2 Running the GA code in B.6.1

LISTING B.6.2: RUN

```
gfortran -c normatest.f
gfortran -c predicttest.f
gfortran -c denormatest.f
gfortran -c ga.c
gfortran -o ga ga.o denormatest.o predicttest.o normatest.o
./ga training_data.txt.ofv gen_01.dat
#./ga gen_03.dat gen_04.dat

a=2
while [ "$a" -le "3" ]
do
        echo "$a"
        let "b=a-1"
        echo "$b"
        ./ga gen_0"$b".dat gen_0"$a".dat
        let "a+=1"
done

rm thebinary thebinary2 roulettewheel.txt fileout.txt new.txt neww.txt predict.pat denormed.dat
rm *.o

exit 0
```

B.6.3 Running the Pareto GA code in B.6.1

This program obtains the Pareto Front using a similar technique to the GA.

LISTING B.6.3: GA_TRIALV24PARETO_2INP.C

```
/*This is the main file. It finds 2 parents to crossover, and crosses over random genes in
integer form.
The program finds the pareto optimal. However it uses the sum of objectives to cause a selection
 pressure.
The trained ANN is used to predict the required outputs and the resulting weights of the neurons
are stored in the file 'testfile' along with the number of inputs, layers, neurons and outputs.
First, the offspring aerofoils (file is new.txt) are normalised (file is fileout), then its
optimization value predicted (file is predict.pat) and then denormalized (file is denormed.dat).
Penalty is included if the values fall outside the constraints.
The roulette method is used for selection i.e. normalization of the fitness values is done
for the entire file before selection occurs. The roulette itself is a file containing the
points with higher fitness repeated as many times as is proportional to its fitness, so that
it has a higher probability of being picked.
A pool of the offspring characteristics is created.
Elitism can be used here.
The conversion to binary is a function that can be called and also aerofoils out of
constraints are included in the pool but with the penalty value, since this is assumed to
make the GA converge faster.
The GA is repeated with the new pool for a certain number of iterations.
Mutation has been implemented.
The input file is generic so data can be read from the pool.
Fractional values for the characteristics have been introduced and the option of excluding
values that violate constraints.
In this version, structures have been used instead of matrices to save memory space.
Here, the genes are swapped as opposed to swapping bits of the genes which is what prevented
the GA from converging without elitism.
The loads are predicted from more than 1 ANN. Each ANN is assigned to predict a small
number of similar loads e.g. Avg Cm and Delta Cm.
User-defined constraints such as minimum lift, etc have been implemented.
*/

/* Note that the OFV is not read but calculated when the initial pool is read - so if you want
   it to include values outside the constraints with the penalty added, it wont add the penalty.
   For this to happen, it must be read from the file.*/
/************************************ BUGS ****************************************/
/*
 *
```

```
 */
/*****************************************************************************************/

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>

/*--- FUNCTION DEFINITIONS ---*/
float binaryfunc(float);                /* function for converting decimal to binary */
float bintodec(void);                   /* function for converting binary to decimal */

struct {                                /* structure for each chromosome */
        float A1;                       /* gene for 1st coefficient */
        float A2;                       /* gene for 2nd coefficient */
        float A3;                       /* gene for 3rd coefficient */
        float CQ;                       /* CD */
        float CM;                       /* CM */
        float MDD;                      /* MDD */
        float ofv;                      /* OFV or fitness value CL/CD */
        int index;                      /* identity number */
        int norm;                       /* normalised fitness integer - for roulette */
        } grid[1000];

main(int argc, char *argv[])
{
float a1,a2,a3,ct,cq,cm,mdd,value;      /* reads the twist, collective and OF from initial pool i.e.
     solver data */
float ngrid;                            /* reads the twist and collective values on the roulette wheel */
float emargin,threshold;                /* reads settings for elitsm margin and constraint inclusion
     threshold */
float ve, minve, maxve, max;            /* used to find max and minimum OF values for normalizing and
     elitism */
float maxA1, max2A1, maxA2, max2A2, maxA3, max2A3; /* selects two thickness values and camber values to
     cross over */
float selectA1, selectA2, selectA3;     /* is the individual bit of the binary used for conversion back
     to decimal */
float elit_A1, elit_A2, elit_A3;        /* used to select elite thickness and elite camber values */
float JJ, aa, bb, cc;                   /* used in the fortran functions to read from the files and elite
      R */
float aaaa, bbbb, cccc, dddd;           /* used in the fortran functions to read from the files */
float aaa,bbb,ddd,eee,fff;              /* used in the fortran functions to read from the files */
float a5a,b5b,d5d,e5e,g5g,f5f,c5c,h5h;  /* used in the fortran functions to read from the files */
float cta,cqa,cma,mdda;                 /* used in the fortran functions to read from the files */
float conCM,marginCM,minLift,minMDD;    /* user defined constraints */
float minofv,maxofv,gCm,gMD,gCQ;
int counter;                            /* grid counters */
int normalizeto, gen_max;               /* normalize fitness values from 0 to ... and max iterations */
int rounum, bk, rouindex, rouli;        /* roulettewheel file info */
int select_upper, bin_upper, mmg_upper; /* upper limit of the random number generator */
int nin,a,b,i,ii,j,jj,k,kk,countline,gen;       /* counters */
int sel1pt, sel2pt, crpt, mupt, mut;            /* random selection points in roulette wheel and cross,
     mutation point */
int selgene, sel1gene, sel2gene, sel3gene;      /* gene selection to crossover */

int binA1[2][8],binA2[2][8],binA3[2][8],holder[1][8]; /* for conversion back to decimal from binary */

FILE *selected, *input, *output, *norm, *denorm, *penalty, *roulette, *roulselect, *binread, *binwrite, *
     setting;
FILE *pool, *poolread1, *poolread, *poolelit, *onlyelit,*temp;
temp = fopen("norma.dat","r");
fscanf(temp,"%d\n",&nin);
float mxA[nin],mnA[nin];                        /* for normalising for mutation binary */
char filename[128];

printf("GA version 24\n\n");
setting = fopen("set","r");
fscanf(setting,"%f\n%d\n%d\n%f\n",&emargin,&normalizeto,&gen_max,&threshold)==4;
printf("%f %d %d %f\n",emargin,normalizeto,gen_max,threshold);
fclose(setting);

for(j=0;j<nin;j++)
{
        fscanf(temp,"%f %f\n",&mnA[j],&mxA[j]);
}
for(j=0;j<nin;j++)
{
        fscanf(temp,"%f %f\n",&mnA[j],&mxA[j]);
}
fclose(temp);
/*--- USER-DEFINED CONSTRAINTS ---*/

/* open a file to read from*/
maxofv=-10000;
minofv=10000;
jj=0;
strcpy(filename,argv[1]);
input = fopen(filename,"r");
while((j = getc(input)) != EOF )
        {
                ungetc(j,input);
                fscanf(input,"%f %f %f %f %f %f %f\n",&a1,&a2,&a3,&cq,&cm,&mdd,&value)==7;
                grid[jj].A1=a1;
                grid[jj].A2=a2;
                grid[jj].A3=a3;
                grid[jj].CQ=cq;
                grid[jj].CM=cm;
                grid[jj].MDD=mdd;
                grid[jj].ofv=value;
                if (grid[jj].ofv>maxofv) {maxofv = grid[jj].ofv;}
                if (grid[jj].ofv<minofv) {minofv = grid[jj].ofv;}
                grid[jj].index=jj;
```

```
                jj=jj+1;
        }
//printf("%d\n ", jj);
fclose(input);
printf("A1: max: %f, min: %f\n ", mxA[0],mnA[0]);
printf("A2: max: %f, min: %f\n ", mxA[1],mnA[1]);
printf("A3: max: %f, min: %f\n ", mxA[2],mnA[2]);

/*----- SELECTION - ROULETTE BASED -----*/
strcpy(filename,argv[2]);
poolelit = fopen(filename,"w");
onlyelit = fopen("best.txt","w");
for(i=0;i<jj;i++)
        {
                gCm=grid[i].CM; gMD=grid[i].MDD; gCQ=grid[i].CQ;
                for (j=0;j<jj;j++)
                {
                        //if ((grid[j].CM <= gCm) && (grid[j].MDD <= gMD))
                        if ((sqrt((grid[j].CM - gCm) * (grid[j].CM - gCm)) <= 0.05) && (grid[j].MDD <=
                             gMD) && (grid[j].CQ <= gCQ))
                                {a=j; gCm=grid[j].CM; gMD = grid[j].MDD; gCQ = grid[j].CQ;}
                }
                printf("a is %d\n",a);
        fprintf(poolelit,"%f\t%f\t%f\t%f %f %f %f\n",grid[i].A1,grid[i].A2,grid[i].A3,grid[i].CQ,grid[i].
             CM,grid[i].MDD,grid[i].ofv);
        fprintf(onlyelit,"%f\t%f\t%f\t%f %f %f %f\n",grid[a].A1,grid[a].A2,grid[a].A3,grid[a].CQ,grid[a].
             CM,grid[a].MDD,grid[a].ofv);
        }
fclose(poolelit);
fclose(onlyelit);
/*------------------------------ end of elitism ----------------------------------------*/

for(i=0; i<jj; i++)
        {
                ngrid=(grid[i].ofv - minofv) / (maxofv - minofv) * normalizeto ;
                grid[i].norm = (int)ngrid;
//              printf("normgrid %d\n",grid[i].norm);
        }
/*--- Put it on a roulette wheel which is the file roulettewheel.txt ---*/
countline=0;
roulette=fopen("roulettewheel.txt","w");
for(i=0;i<jj;i++)
        {
                rouindex = grid[i].index;
                rounum = grid[i].norm;
                for(k=0;k<rounum;k++)
                {
                        fprintf(roulette,"%d  \n",rouindex);
                        countline=countline+1;
                }
        }
fclose(roulette);
//printf("countline %d \n", countline);
int roul[countline];            /* array going to contain roulette wheel */
roulselect=fopen("roulettewheel.txt","r");
for(a=0;a<countline;a++)
        {
                fscanf(roulselect,"%d\n",&rouli)==1;
                roul[a]=rouli;
        }
fclose(roulselect);

srand(time(NULL));

/*----- Repeat for gen number of runs to populate pool.dat -----*/
for(gen=0;gen<gen_max;gen++)
{
/*--- Select 2 parents to crossover ---*/
select_upper=countline;
sel1pt=rand()%select_upper;
sel2pt=rand()%select_upper;
sel1gene=rand()%mmg_upper;
sel2gene=rand()%mmg_upper;
sel3gene=rand()%mmg_upper;
//printf("selpt %d %d\n",sel1pt,sel2pt);
//printf("roul %d %d\n",roul[sel1pt],roul[sel2pt]);
maxA1=grid[roul[sel1pt]].A1;
max2A1=grid[roul[sel2pt]].A1;
maxA2=grid[roul[sel1pt]].A2;
max2A2=grid[roul[sel2pt]].A2;
maxA3=grid[roul[sel1pt]].A3;
max2A3=grid[roul[sel2pt]].A3;
//printf("PARENTS: maxA1=%f max2A1=%f maxA2=%f, max2A2=%f, maxA3=%f, max2A3=%f\n",maxA1,max2A1,maxA2,
     max2A2,maxA3,max2A3);

/*--- CROSSOVER ---*/
mmg_upper=2;
if (sel1gene==0)
{max=max2A1;
max2A1=maxA1;
maxA1=max;}
if (sel2gene==0)
{max=max2A2;
max2A2=maxA2;
maxA2=max;}
if (sel3gene==0)
{max=max2A3;
max2A3=maxA3;
maxA3=max;}

//printf("CROSSED: maxA1=%f max2A1=%f maxA2=%f, max2A2=%f, maxA3=%f, max2A3=%f\n",maxA1,max2A1,maxA2,
     max2A2,maxA3,max2A3);
```

```
/*--- normalise first between 1 and 10 so it fits in limited binary number---*/
maxA1=(maxA1-mnA[0])/(mxA[0]-mnA[0]) * 30;
maxA2=(maxA2-mnA[1])/(mxA[1]-mnA[1]) * 30;
maxA3=(maxA3-mnA[2])/(mxA[2]-mnA[2]) * 30;

/*--- CONVERSION TO BINARY CODING I.E. PHENOTYPE TO GENOTYPE ---*/

/*convert to binary*/
bk=binaryfunc(maxA1);
binread=fopen("thebinary","r");
fscanf(binread,"%d %d %d %d %d %d %d %d",&binA1[0][0],&binA1[0][1],&binA1[0][2],&binA1[0][3],&binA1
    [0][4],&binA1[0]
[5], &binA1[0][6],&binA1[0][7])==8;
fclose(binread);
bk=binaryfunc(max2A1);
binread=fopen("thebinary","r");
fscanf(binread,"%d %d %d %d %d %d %d %d",&binA1[1][0],&binA1[1][1],&binA1[1][2],&binA1[1][3],&binA1
    [1][4],&binA1[1]
[5], &binA1[1][6], &binA1[1][7])==8;
fclose(binread);

bk=binaryfunc(maxA2);
binread=fopen("thebinary","r");
fscanf(binread,"%d %d %d %d %d %d %d %d",&binA2[0][0],&binA2[0][1],&binA2[0][2],&binA2[0][3],&binA2
    [0][4],&binA2[0]
[5], &binA2[0][6],&binA2[0][7])==8;
fclose(binread);
bk=binaryfunc(max2A2);
binread=fopen("thebinary","r");
fscanf(binread,"%d %d %d %d %d %d %d %d",&binA2[1][0],&binA2[1][1],&binA2[1][2],&binA2[1][3],&binA2
    [1][4],&binA2[1]
[5], &binA2[1][6], &binA2[1][7])==8;
fclose(binread);

bk=binaryfunc(maxA3);
binread=fopen("thebinary","r");
fscanf(binread,"%d %d %d %d %d %d %d %d",&binA3[0][0],&binA3[0][1],&binA3[0][2],&binA3[0][3],&binA3
    [0][4],&binA3[0]
[5], &binA3[0][6],&binA3[0][7])==8;
fclose(binread);
bk=binaryfunc(max2A3);
binread=fopen("thebinary","r");
fscanf(binread,"%d %d %d %d %d %d %d %d",&binA3[1][0],&binA3[1][1],&binA3[1][2],&binA3[1][3],&binA3
    [1][4],&binA3[1]
[5], &binA3[1][6], &binA3[1][7])==8;
fclose(binread);

bin_upper=8;              /* upper bound for mutation point is max binary digits */
for(j=0;j<2;j++)
{
        printf("A1 %d: ",j+1);
        for(i=0;i<bin_upper;i++)
        {
        printf("%d",binA1[j][i]);
        }
        printf("\n");
}
for(j=0;j<2;j++)
{
        printf("A2 %d: ",j+1);
        for(i=0;i<bin_upper;i++)
        {
        printf("%d",binA2[j][i]);
        }
        printf("\n");
}
for(j=0;j<2;j++)
{
        printf("A3 %d: ",j+1);
        for(i=0;i<bin_upper;i++)
        {
        printf("%d",binA3[j][i]);
        }
        printf("\n");
}

/*--- MUTATION ---*/
mupt=rand()%bin_upper;            /* random number generator < upper bound */
mut=rand()%mmg_upper;             /* mutation bit i.e. replace with this bit (1 or 0) */
binA1[0][mupt]=binA1[0][mut];
binA1[1][mupt]=binA1[1][mut];
mupt=rand()%bin_upper;            /* random number generator < upper bound */
mut=rand()%mmg_upper;             /* mutation bit i.e. replace with this bit (1 or 0) */
binA2[0][mupt]=binA2[0][mut];
binA2[1][mupt]=binA2[1][mut];
mupt=rand()%bin_upper;            /* random number generator < upper bound */
mut=rand()%mmg_upper;             /* mutation bit i.e. replace with this bit (1 or 0) */
binA3[0][mupt]=binA3[0][mut];
binA3[1][mupt]=binA3[1][mut];

//printf("mutated:\n");
//for(j=0;j<2;j++)
//{
//        printf("twist %d: ",j+1);
//        for(i=0;i<bin_upper;i++)
//        {
//        printf("%d",bintwist[j][i]);
//        }
//        printf("\n");
//}
//for(j=0;j<2;j++)
//{
//        printf("coll %d: ",j+1);
```

```
//        for(i=0;i<bin_upper;i++)
//        {
//        printf("%d",bincoll[j][i]);
//        }
//        printf("\n");
//}
/*--- CONVERSION BACK TO PHENOTYPE I.E. DECIMAL ---*/
binwrite=fopen("thebinary2","w");
for(j=0;j<2;j++)
{
        for(i=0;i<bin_upper;i++)
        {
                fprintf(binwrite, "%d ",binA1[j][i]);
        }
        fprintf(binwrite, "\n");
}
fclose(binwrite);
selectA1=bintodec();
binwrite=fopen("thebinary2","w");
for(j=0;j<2;j++)
{
        for(i=0;i<bin_upper;i++)
        {
                fprintf(binwrite, "%d ",binA2[j][i]);
        }
        fprintf(binwrite, "\n");
}
fclose(binwrite);
selectA2=bintodec();
binwrite=fopen("thebinary2","w");
for(j=0;j<2;j++)
{
        for(i=0;i<bin_upper;i++)
        {
                fprintf(binwrite, "%d ",binA3[j][i]);
        }
        fprintf(binwrite, "\n");
}
fclose(binwrite);
selectA3=bintodec();

/*--- denormalise ---*/
selectA1=selectA1 / 30 * (mxA[0]-mnA[0]) + mnA[0];
selectA2=selectA2 / 30 * (mxA[1]-mnA[1]) + mnA[1];
selectA3=selectA3 / 30 * (mxA[2]-mnA[2]) + mnA[2];

selected=fopen("neww.txt","w");
fprintf(selected,"%f %f %f \n",selectA1,selectA2,selectA3);
fclose(selected);
//printf("selected:\t%f\t%f\t%f\n",selectA1,selectA2,selectA3);

/*--- FINDING OPTIMIZATION FUNCTION VALUE USING TRAINED ANN ---*/

/*--- normalizing ---*/
normatest_(&selectA1,&selectA2,&selectA3);
norm=fopen("fileout.txt","r");
fscanf(norm,"%f %f %f\n",&aa,&bb,&cc);
fclose(norm);
printf("normalised:\taa = %f, bb = %f, cc = %f\n", aa,bb,cc);

/*--- predicting value CQ ---*/
predicttestcq_(&aa,&bb,&cc);
denorm=fopen("predict.pat","r");
fscanf(denorm,"%f %f %f %f \n",&aaa,&ddd,&eee,&fff);
fclose(denorm);
//printf("predicted:\taaa = %f, ddd= %f, eee= %f, fff=%f\n", aaa,ddd,eee,fff);
denormaltestcq_(&aaa,&ddd,&eee,&fff);
penalty=fopen("denormed.dat","r");
fscanf(penalty,"%f %f %f %f\n", &aaaa,&bbbb,&dddd,&cqa);
fclose(penalty);
printf("denormalised CQ:\taaaa = %f, bbbb = %f, dddd = %f, cd = %f\n", aaaa,bbbb,dddd,cqa);

/*--- predicting value CM ---*/
predicttestcm_(&aa,&bb,&cc);
denorm=fopen("predict.pat","r");
fscanf(denorm,"%f %f %f %f \n",&aaa,&ddd,&eee,&fff);
fclose(denorm);
//printf("predicted:\taaa = %f, ddd= %f, eee= %f, fff=%f\n", aaa,ddd,eee,fff);
denormaltestcm_(&aaa,&ddd,&eee,&fff);
penalty=fopen("denormed.dat","r");
fscanf(penalty,"%f %f %f %f\n", &aaaa,&bbbb,&dddd,&cma);
fclose(penalty);
printf("denormalised CM:\taaaa = %f, bbbb = %f, dddd = %f, cm = %f\n", aaaa,bbbb,dddd,cma);

/*--- predicting value MDD ---*/
predicttestmdd_(&aa,&bb,&cc);
denorm=fopen("predict.pat","r");
fscanf(denorm,"%f %f %f %f \n",&aaa,&ddd,&eee,&fff);
fclose(denorm);
//printf("predicted:\taaa = %f, ddd= %f, eee= %f, fff=%f\n", aaa,ddd,eee,fff);
denormaltestmdd_(&aaa,&ddd,&eee,&fff);
penalty=fopen("denormed.dat","r");
fscanf(penalty,"%f %f %f %f\n", &aaaa,&bbbb,&dddd,&mdda);
fclose(penalty);
printf("denormalised MDD:\taaaa = %f, bbbb = %f, dddd = %f, mdd = %f\n", aaaa,bbbb,dddd,mdda);

/*---------- PENALTY FOR VALUES OUTSIDE CONSTRAINTS ----------*/
cccc= 0;
printf("%f %f %f, cccc %f\n",aaaa,bbbb,dddd,cccc);

/*------ ADD TO POOL without repetition and with penalty factor added ------*/
/*------ another version of ADD TO POOL without repetition and with penalty factor added ------*/
j=0;
```

```
ii=1;
counter=0;
strcpy(filename,argv[2]);
poolread = fopen(filename,"r+");
while((j < ii)) /*= getc(input)) != EOF )*/
{
        i=0;
        ii=0;
        poolread1 = fopen(filename,"r");
        while((i = getc(poolread1)) != EOF)
                {ungetc(i,poolread1);
                fscanf(poolread1,"%f %f %f %f %f %f %f\n",&a5a,&b5b,&d5d,&f5f,&g5g,&c5c,&e5e)==7;
                ii=ii+1;}
        fclose(poolread1);
//      printf("ii = %d \n",ii);
        fscanf(poolread,"%f %f %f %f %f %f %f\n",&a5a,&d5d,&e5e,&f5f,&g5g,&c5c,&e5e)==7;
        if (sqrt((a5a-aaaa)*(a5a-aaaa)<0.00001) && sqrt((b5b-bbbb)*(b5b-bbbb)<0.00001) && sqrt((d5d-dddd)
            *(d5d-dddd)<0.00001)) break;
        if ((aaaa < mnA[0]) || (aaaa > mxA[0]) || (bbbb > mxA[1]) || (bbbb<mnA[1]) || (dddd > mxA[2]) ||
            (dddd < mnA[2])) break;
//        else if ((a5a!=aaaa) && (b5b!=bbbb) )
        else if ((sqrt((a5a-aaaa)*(a5a-aaaa))>0.00001) && (sqrt((b5b-bbbb)*(b5b-bbbb))>0.00001) && (sqrt
            ((d5d-dddd)*(d5d-dddd))>0.00001))
        {
                counter=counter+1;
                if(( counter == ii))    /* can include or exclude minimum fitness values */
                {fprintf(poolread,"%f\t%f\t%f\t%f %f %f %f\n",aaaa,bbbb,dddd,cqa,cma,mdda,cccc);}
        }
        j=j+1;
}
fclose(poolread);
}
return 0;
}

/*--- FUNCTION FOR THE CONVERSION TO BINARY - 6-digit before decimal point, 3-digit after decimal point
    ---*/
float binaryfunc(float p)
{
int div, rem, i, n, bin[5], bindec[3], bk, count;
float fr, r, frac, number;
FILE *binf;
i=0;
                fr = (int)p;
                div = p;
                while (div > 0)
                        {
                                fr = fr /2;
                                div = fr;
                                r = ( fr - div ) *2;
                                rem = r;
                                bin[4-i] =rem;
                                i=i+1;
                                fr = div;
                        }
                if (i<5)
                        {
                                n = 5-i;
                                while(n > 0)
                                {
                                        bin[n -1]=0;
                                        n = n -1;
                                }
                        }
bk=bin[4];

/* fraction part */
frac = p - (int)p;
number = frac;
count = 0;
while(number>0)
{
        number = number * 2;
        bindec[count] = (int)number;
        number = number - (int)number;
        count = count + 1;
        if (count==3) break;
}
if (count<3)
{
        n=count;
        while (n<3)
        {
                bindec[n]=0;
                n = n + 1;
        }
}

binf=fopen("thebinary","w");
fprintf(binf,"%d %d %d %d %d %d %d %d \n",bin[0],bin[1],bin[2],bin[3],bin[4],bindec[0],bindec[1],bindec
    [2]);
fclose(binf);
return bk;
}

/*--- FUNCTION FOR THE CONVERSION OF BINARY TO DECIMAL ---*/

float bintodec(void)
{
        FILE *binread, *output;
        float sum1, sum2;
        int i, val1, val2, bin[2][8];
        binread=fopen("thebinary2","r");
```

```
        for(i=0;i<2;i++)
        {
        fscanf(binread,"%d %d %d %d %d %d %d %d\n", &bin[i][0],&bin[i][1],&bin[i][2],&bin[i][3],&bin[i
             ][4],&bin[i][5],&bin[i][6],&bin[i][7]);
        }
        fclose(binread);
        sum1=0;
        sum2=0;
        i=0;
        for (i=0;i<5;i++)
                {
                        val1 = bin[0][4-i];
                        sum1 = sum1 + val1 * pow(2,i);
                        val2 = bin[1][4-i];
                        sum2 = sum2 + val2 * pow(2,i);
                }
        i=0;
        for(i=0;i<3;i++)
                {
                        val1 = bin[0][7-i];
                        sum1 = sum1 + val1 * pow(0.5,i+1);
                        val2 = bin[1][7-i];
                        sum2 = sum2 + val2 * pow(0.5,i+1);
                }
        output = fopen("new.txt","w");
        fprintf(output,"%f %f \n",sum2, sum1);
        fclose(output);
        return sum1;
}
```

B.7 Chebyshev Parameterisation for Aerfoils

This program obtains a set of coefficients based on the Chebyshev equations from a set of data points that can recreate the data points. The coefficients can be modified to create variants of the aerofoil. The theory of this method is described further in Section 3.2.1 of the thesis.

```
/*
 * CHEBYSHEV PARAMETERISATION
 * alpha is the matrix containing the coefficients to reproduce the shape.
 * input file is coordinates file.
 *
 */

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>

main(int argc, char *argv[])
{
FILE *input1, *output;
char filename[128];

int i, It, k, ktot, n_of_x,j,n,c;
n=57;
c=7;
float x,y,kk,E;
float XPTS[n], YPTS[n], D[n][c], gamma[n], T[n][c+2], alpha[c];
float gE1[n], gradE[c], hessE[c], gE[n][c], hE[n][c], F[n];

/*--- OPEN AND READ INPUT FILE ---*/
strcpy(filename,argv[1]);
input1 = fopen(filename,"r");                    /* File containing co-ordinates of geometry to be
    parameterised */
i=0;
while((j = getc(input1)) != EOF )
        {
        ungetc(j,input1);
        fscanf(input1,"%f %f \n", &x,&y)==2;
        XPTS[i]=x;
        YPTS[i]=y;
        i=i+1;
        }
fclose(input1);
n_of_x=i;                                        /*--- Number of x points ---*/
/*--- INITIALISE COEFFICIENTS ---*/
ktot=c+2;
for(k=0;k<(ktot-2);k++)
        {
        alpha[k]=(float)k+1;
        }

/*--- NEW REPRESENTATION OF CURVE ---*/
for(k=0;k<ktot;k++)
        {
        kk=k;
        for(i=0;i<n_of_x;i++)
                {
                gamma[i]=acos(2 * sqrt(XPTS[i])-1);
                T[i][k]=cos(kk * gamma[i]);
                }
        }
for(k=0;k<(ktot-2);k++)
        {
        for(i=0;i<n_of_x;i++)
                {
                D[i][k]=T[i][k]-T[i][k+2];
                }
        }

strcpy(filename,argv[2]);                        /*--- Output file to write to ---*/
output = fopen(filename,"w+");

/*--- ITERATIVELY CALCULATE GRADIENT AND HESSIAN ---*/
for(It=0;It<200;It++)
        {
        /*--- Initialise gradient and hessian matrices to 0 ---*/
        for(i=0;i<n_of_x;i++)
                {
                for(k=0;k<(ktot-2);k++)
                        {
                        gE[i][k] = 0;
                        hE[i][k] = 0;
                        gE1[i] = 0;
                        gradE[k] = 0;
                        hessE[k] = 0;
                        F[i] = 0;
                        E = 0;
                        }
                }
        /*--- Fill in the matrices ---*/
        for(i=0;i<n_of_x;i++)
                {
                for(k=0;k<(ktot-2);k++)
                        {
                        gE1[i] = gE1[i] + alpha[k] * D[i][k];
                        }
                }
```

```
        for(i=0;i<n_of_x;i++)
                {
                for(k=0;k<(ktot-2);k++)
                        {
                        gE[i][k] = (gE1[i] - YPTS[i]) * D[i][k];
                        hE[i][k] = D[i][k] * D[i][k];
                        }
                }
        for(k=0;k<(ktot-2);k++)
                {
                for(i=0;i<n_of_x;i++)
                        {
                        gradE[k] = gradE[k] + gE[i][k];
                        hessE[k] = hessE[k] + hE[i][k];
                        }
                }

        /*--- FIND NEW ALPHA ---*/
        for(k=0;k<(ktot-2);k++)
                {
                alpha[k] = alpha[k] - gradE[k] / hessE[k];
                }

        /*--- REPRODUCE NEW CURVE AND FIND ERROR ---*/
// If you want to force the program to use a set of coefficients, edit them here and uncomment
//alpha[0]=0.043557;
//alpha[1]=0.002801;
//alpha[2]=-0.006735;
//alpha[3]=-0.002220;
//alpha[4]=-0.002753;
//alpha[5]=-0.001562;
//alpha[6]=-0.000050;

        for(i=0;i<n_of_x;i++)
                {
                for(k=0;k<(ktot-2);k++)
                        {
                        F[i] = F[i] + alpha[k] * D[i][k];
                        }
                E = E + sqrt((F[i] - YPTS[i]) * (F[i] - YPTS[i]));
                }
        }
        printf("error: %f \n",E);
for(i=0;i<n_of_x;i++)
        {
        fprintf(output,"%f %f \n",XPTS[i],F[i]);
        }
fclose(output);
for(k=0;k<(ktot-2);k++)
        {
        printf("%f\n",alpha[k]);          /* Write new coefficients */
        }
return 0;
}
```

B.7.1 Chebyshev Parameterisation for Wings

The method in B.7 can be applied to wings using the program below.

```
/* PARAMETERISATION PROGRAM
 * This program takes 3 arguments:
 *      1. file containing wing leading edge existing coordinates
 *      2. file containing wing trailing edge existing coordinates
 *      3. file to write wing coordinates to
 * The number of sections used to define the wing can be specified.
 * The coefficients are stored in files called alpha...txt. These
 * can be modified manually to change the wing dimensions.
 * The original wing coefficients are obtained by running the
 * program parameterwing.c for each specific curve.
 * Note that the aerofoil coefficients are stored in the files:
 *      - alphasu.txt for the upper surface
 *      - alphasl.txt for the lower surface
 * These are also obtained from parameterwing.c which automatically
 * stores these coefficients in alphas.txt.
 * The number of points on the wing edge and on the aerofoil must be
 * specified.
 */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>

main(int argc, char *argv[])
{
FILE *input1, *output, *coefficients;
char filename[128];

int N,i,It,k,ktot,n_of_x,n_of_y,j,n,c,na,ca,sec,s,div;
n=101;                  /* No. of points on wing planform */
c=4;                    /* No. of coefficients for planform */
na=57;                  /* No. of points on aerofoil surface */
ca=7;                   /* No. of coefficients for aerofoil */
sec=15;                 /* Number of sections along span */
float x,y,a,kk,E,minxle,maxxle,minyle,maxyle;
float minxte,maxxte,minyte,maxyte;
float XPTSLE[n], YPTSLE[n], DLE[n][c], gammaLE[n], TLE[n][c+1], alphaLE[c];
float XPTSTE[n], YPTSTE[n], DTE[n][c], gammaTE[n], TTE[n][c+1], alphaTE[c];
```

```
float YPTSa[na], YPTSf[na][sec], ZPTS[na][sec], Da[na][ca], gammaa[na], Ta[na][ca+1], alphaa[ca];
float YPTSl[na], ZPTSl[na][sec], Dl[na][ca], gammal[na], Tl[na][ca+1], alphal[ca];
float XSECLE[sec],XSECTE[sec],FSECLE[sec],FSECTE[sec];
float FLE[n], FTE[n], Fa[na],Fl[na];

div=n/sec;
minxle=minxte=1000;
maxxle=maxxte=-1000;
minyle=minyte=1000;
maxyle=maxyte=-1000;

/*--- OPEN AND READ INPUT FILE FOR LE ---*/
strcpy(filename,argv[1]);        // original wing LE file
input1 = fopen(filename,"r");
i=0;
s=0;
while((j = getc(input1)) != EOF )
        {
        ungetc(j,input1);
        fscanf(input1,"%f %f \n",&x,&y)==2;
        XPTSLE[i]=x;
        if(x<minxle) {minxle=x;}
        if(y<minyle) {minyle=y;}
        if(x>maxxle) {maxxle=x;}
        if(y>maxyle) {maxyle=y;}
        i=i+1;
        }
fclose(input1);
/*--- OPEN AND READ INPUT FILE FOR TE ---*/
strcpy(filename,argv[2]);        // original wing TE file
input1 = fopen(filename,"r");
i=0;
s=0;
while((j = getc(input1)) != EOF )
        {
        ungetc(j,input1);
        fscanf(input1,"%f %f \n",&x,&y)==2;
        XPTSTE[i]=x;
        if(x<minxte) {minxte=x;}
        if(y<minyte) {minyte=y;}
        if(x>maxxte) {maxxte=x;}
        if(y>maxyte) {maxyte=y;}
        i=i+1;
        }
fclose(input1);
printf("%d %f %f %f %f\n",i,minxle,maxxle,minyle,maxyle);
n_of_x=i;
ktot=c+1;
/*--- NORMALISE ---*/
for(j=0;j<n_of_x;j++)
        {
        XPTSLE[j]=(XPTSLE[j]-minxle)/(maxxle-minxle);
        XPTSTE[j]=(XPTSTE[j]-minxte)/(maxxte-minxte);
        }

coefficients = fopen("alphasLE.txt","r");
for(j=0;j<c;j++)
        {
        fscanf(coefficients,"%f \n",&a)==1;
        alphaLE[j]=a;
        //printf("%f \n",a);
        }
fclose(coefficients);
coefficients = fopen("alphasTE.txt","r");
for(j=0;j<c;j++)
        {
        fscanf(coefficients,"%f \n",&a)==1;
        alphaTE[j]=a;
        //printf("%f \n",a);
        }
fclose(coefficients);

/*--- NEW REPRESENTATION OF CURVE ---*/
for(k=0;k<ktot;k++)
        {
        kk=k;
        for(i=0;i<n_of_x;i++)
                {
                gammaLE[i]=acos(2 * sqrt(XPTSLE[i])-1);
                gammaTE[i]=acos(2 * sqrt(XPTSTE[i])-1);
                TLE[i][k]=cos(kk * gammaLE[i]);
                TTE[i][k]=cos(kk * gammaTE[i]);
                }
        }
for(k=0;k<(ktot-1);k++)
        {
        for(i=0;i<n_of_x;i++)
                {
                DLE[i][k]=TLE[i][k]-TLE[i][k+1];
                DTE[i][k]=TTE[i][k]-TTE[i][k+1];
                }
        }

/*--- INITIALISE NEW CURVE ---*/
for(i=0;i<n_of_x;i++)
        {
        FTE[i]=0; FLE[i] = 0;
        }
/*--- REPRODUCE NEW CURVE ---*/
for(i=0;i<n_of_x;i++)
        {
        for(k=0;k<(ktot-1);k++)
                {
```

```
                FLE[i] = FLE[i] + alphaLE[k] * DLE[i][k];
                FTE[i] = FTE[i] + alphaTE[k] * DTE[i][k];
                }
        }
/*--- DENORMALISE ---*/
for(i=0;i<n_of_x;i++)
        {
        FLE[i] = FLE[i] * (maxyle - minyle) + minyle;
        FTE[i] = FTE[i] * (maxyte - minyte) + minyte;
        XPTSLE[i] = XPTSLE[i] * (maxxle - minxle) + minxle;
        XPTSTE[i] = XPTSTE[i] * (maxxte - minxte) + minxte;
        if(((float)i/div-i/div)==0)
                {XSECLE[s]=XPTSLE[i];
                 FSECLE[s]=FLE[i];
                 FSECTE[s]=FTE[i];
                printf("%d,%f,%f,%f\n",s,XSECLE[s],FSECLE[s],FSECTE[s]);
                 s=s+1;}
        }

n_of_y=na;
ktot=ca+1;
/*--------------- AEROFOIL CURVE ----------------*/
coefficients = fopen("alphasu.txt","r"); /* Read coefficients upper */
for(j=0;j<ca;j++)
        {
        fscanf(coefficients,"%f \n",&a)==1;
        alphaa[j]=a;
        //printf("%f \n",a);
        }
fclose(coefficients);
coefficients = fopen("alphasl.txt","r"); /* Read coefficients lower */
for(j=0;j<ca;j++)
        {
        fscanf(coefficients,"%f \n",&a)==1;
        alphal[j]=a;
        //printf("%f \n",a);
        }
fclose(coefficients);
/*--- Coordinates for aerofoil evenly spaced so that order can be reversed on wing ---*/
for(i=0;i<n_of_y;i++)
        {
        YPTSa[i] = (float)i/(n_of_y-1);
//      printf("%f\n",YPTSa[i]);
        }
/*--- NEW REPRESENTATION OF CURVE ---*/
for(k=0;k<ktot;k++)
        {
        kk=k;
        for(i=0;i<n_of_y;i++)
                {
                gammaa[i]=acos(2 * sqrt(YPTSa[i])-1);
                Ta[i][k]=cos(kk * gammaa[i]);
                }
        }
for(k=0;k<(ktot-1);k++)
        {
        for(i=0;i<n_of_y;i++)
                {
                Da[i][k]=Ta[i][k]-Ta[i][k+1];
                }
        }
/*--- INITIALISE NEW CURVE ---*/
for(i=0;i<n_of_y;i++)
        {
        Fa[i] = 0; Fl[i] = 0;
        }
/*--- REPRODUCE NEW CURVE ---*/
for(i=0;i<n_of_y;i++)
        {
        for(k=0;k<(ktot-1);k++)
                {
                Fa[i] = Fa[i] + alphaa[k] * Da[i][k];
                Fl[i] = Fl[i] + alphal[k] * Da[i][k];
                }
        }
/*--- DENORMALISE ---*/
strcpy(filename,argv[3]);        /*--- Output file to write to ---*/
output = fopen(filename,"w+");
for(j=0;j<sec;j++)
        {
        for(i=0;i<n_of_y;i++)
                {
                ZPTS[i][j]= Fa[i] * (FSECLE[j]-FSECTE[j]);
                ZPTSl[i][j]= Fl[i] * (FSECLE[j]-FSECTE[j]);
                N=n_of_y-1-i;
                YPTSf[i][j] = YPTSa[N] * (FSECLE[j]-FSECTE[j]) + FSECTE[j];
                fprintf(output,"%f %f %f %f\n",XSECLE[j],YPTSf[i][j],ZPTS[i][j],ZPTSl[i][j]);
                }
        }
fclose(output);

return 0;
}
```

B.8 Fuselage Parameterisation Technique

B.8.1 Program to Create New Fuselage from Parameters

This program creates a set of data points and an ICEMCFD replay file to create a fuselage geometry. A number of options are available and many parts can be combined. It automates the grid generation of a fuselage. Section 3.2.3 describes the technique in more detail. An example input file is shown in Appendix B.8.3.

```
      PROGRAM ROBIN
c
c PROGRAM to find the coordinates of a fuselage using the robin body typre parameterisation.
c From NASA paper Appendix in JAXA folder.
c
c This program allows you to rotate parts, for example the tail plane etc.
c Option 0 does a lofting surface with ful curves, Option 1 makes segmented curves.
c The number of segments for a segmented surface can be set along with this option.
c Ensure that the number of points is a multiple of the number of segments.
c The ends of the body can be closed at a point or at a surface.
c*******************************************************************************************
c
      INTEGER MAXSTATIONS,MAXPART,MAXSEG,MAXPOINT,PTS
      PARAMETER(MAXSTATIONS=100,MAXPART=10,MAXSEG=12,MAXPOINT=72)
      INTEGER I,J,seg,num(MAXPART),K,KK,fn(MAXSEG,MAXPART),fnv
      INTEGER nseg,M
      REAL delta(MAXSEG,MAXPART)
      COMPLEX fh,r,fhw,fw
      COMPLEX hh(MAXPOINT),ww(MAXPOINT),zo(MAXPOINT),nn(MAXPOINT)
      COMPLEX yo(MAXPOINT)
      REAL x,l,y,zz,phi,pi,II,JJ
      REAL H(8,MAXSEG,MAXPART),W(8,MAXSEG,MAXPART)
      REAL Z(8,MAXSEG,MAXPART),N(8,MAXSEG,MAXPART)
      REAL YF(8,MAXSEG,MAXPART)
      REAL xu(MAXSEG,MAXPART),xl(MAXSEG,MAXPART),segm(MAXSEG,MAXPART)
      REAL dx,xstart(MAXPART),xend(MAXPART)
      REAL xx(MAXSTATIONS),yy(MAXSTATIONS,MAXPOINT)
      REAL zzz(MAXSTATIONS,MAXPOINT)
      CHARACTER*100 filenam,filename(MAXPART),fil(MAXPART),sta_c,pnt_c
      CHARACTER(len=100)::sta_x,sta_y,sta_z,srf_c,pnt_cn,sta_cn,rot_ax
      CHARACTER(len=100)::rx_c,ry_c,rz_c,rotang_c
      CHARACTER(len=100)::x_st,y_st,z_st,x_en,y_en,z_en
      CHARACTER(len=100)::sta_c1,sta_c5,sta_c3,sta_cf1
      CHARACTER(len=2)::number,axis
      INTEGER IPART,nstations(MAXSTATIONS),NPARTS,option(MAXPART),III
      INTEGER pcounter,scounter,ccounter,stcounter,fcounter,maxfcounter
      INTEGER minfcounter,counter4,mdpoint1,mdpoint2,nicem
      INTEGER rotoption(MAXPART),pntoption(MAXPART)
29    FORMAT(F4.8)
c
      pi=3.141592654
c
c     --- DEFINE COEFFICIENTS FOR HEIGHT, WIDTH, CAMBER, ELLIPTIC POWER ----------------
      print*,'Enter number of existing ICEM part geometries:'
      read(5,*)nicem
      WRITE( number, '(i2)' ) nicem+1
      number = adjustl(number)
      OPEN (51,file='parametric_fuselage'//trim(number)//'.dat',STATUS='
     &OLD')
      IPART=1
100   read(51,*,END=1000)filenam,option(IPART),nseg       ! read the body and the options for surfacing in
     ICEM
      read(51,*)num(IPART),xstart(IPART),xend(IPART)      ! read the number of segments and the start and
          end x co-ordinates of the part
      READ(51,*)rotoption(IPART),rx_c,ry_c,rz_c,axis,rotang_c
      READ(51,*)pntoption(IPART),x_st,y_st,z_st,x_en,y_en,z_en
      DO J=1,num(IPART)
          READ(51,*) segm(J,IPART),xl(J,IPART),xu(J,IPART),fn(J,IPART),
     &          delta(J,IPART)                                  ! ... and parameter
          READ(51,*) (H(I,J,IPART),I=1,8)                       ! H coefficients
          READ(51,*) (W(I,J,IPART),I=1,8)
          READ(51,*) (YF(I,J,IPART),I=1,8)
          READ(51,*) (Z(I,J,IPART),I=1,8)
          READ(51,*) (N(I,J,IPART),I=1,8)
      ENDDO
      IPART=IPART+1
      GOTO 100
1000  CLOSE (51)
      NPARTS=IPART-1
      print*, 'NPARTS',NPARTS
c
c     --- DEFINE OUTPUT FILE AND X STATIONS ---------------------------------------------
      pcounter=0
      ccounter=0
      scounter=0
      stcounter=0
      fcounter=0


      DO IPART=1,NPARTS                    ! ------------------------------LOOP OVER PARTS
      WRITE( number, '(i2)' ) nicem+IPART  ! convert part number to character for filename
      number = adjustl(number)             ! adjust number so no gaps in filename
      filename(IPART) = 'parametric_fuselage_out'//trim(number)//'.dat' !icemscript part filename
      OPEN(61,FILE=filename(IPART),ACCESS='APPEND',STATUS='NEW')

      x=xstart(IPART) ! first station
c
      I=1
      DO WHILE(x.lt.xend(IPART))           ! ------------------------------LOOP OVER STATIONS
         DO J=1,num(IPART)
```

```
            if((x.le.xu(J,IPART)).AND.(x.ge.xl(J,IPART))) then
              seg=segm(J,IPART)
              dx=delta(J,IPART)
              fnv=fn(J,IPART)
            endif
          ENDDO
          xx(I)=x

c       --- FIND COEFFICIENTS FOR DEFINING r ------------------------------------------
c       H ------------------
          if (H(4,seg,IPART).eq.0) then
             hh(I)=H(1,seg,IPART)
          else
             fh=H(1,seg,IPART)+H(2,seg,IPART)*(abs((x
     &          +H(3,seg,IPART))/H(4,seg,IPART)))**H(5,seg,IPART)
             hh(I)=fh
             if (H(8,seg,IPART).eq.0 .or. H(8,seg,IPART).eq.1) goto 1111
             hh(I)=H(7,seg,IPART)*(abs(fh))**(1.0/H(8,seg,IPART))
     &            +H(6,seg,IPART)
          endif
1111      continue
c       W ------------------
          if (W(4,seg,IPART).eq.0) then
             ww(I)=W(1,seg,IPART)
          else
             fh=W(1,seg,IPART)+W(2,seg,IPART)*(abs((x
     &          +W(3,seg,IPART))/W(4,seg,IPART)))**W(5,seg,IPART)
             ww(I)=fh
             if (W(8,seg,IPART).eq.0 .or. W(8,seg,IPART).eq.1) goto 2222
             ww(I)=W(7,seg,IPART)*(abs(fh))**(1.0/W(8,seg,IPART))+
     &             W(6,seg,IPART)
          endif
2222      continue
c       X ------------------
          if (YF(4,seg,IPART).eq.0) then
             yo(I)=YF(1,seg,IPART)
          else
             fh=YF(1,seg,IPART)+YF(2,seg,IPART)*(abs((x
     &          +YF(3,seg,IPART))/YF(4,seg,IPART)))**YF(5,seg,IPART)
             yo(I)=fh
             if (YF(8,seg,IPART).eq.0 .or. YF(8,seg,IPART).eq.1)goto 5555
             yo(I)=YF(7,seg,IPART)*(abs(fh))**(1.0/YF(8,seg,IPART))+
     &             YF(6,seg,IPART)
          endif
5555      continue
c       Z ------------------
          if (Z(4,seg,IPART).eq.0) then
             zo(I)=Z(1,seg,IPART)
          else
             fh=Z(1,seg,IPART)+Z(2,seg,IPART)*(abs((x
     &          +Z(3,seg,IPART))/Z(4,seg,IPART)))**Z(5,seg,IPART)
             zo(I)=fh
             if (Z(8,seg,IPART).eq.0 .or. Z(8,seg,IPART).eq.1) goto 3333
             zo(I)=Z(7,seg,IPART)*(abs(fh))**(1.0/Z(8,seg,IPART))+
     &             Z(6,seg,IPART)
          endif
3333      continue
c       N ------------------
          if (N(4,seg,IPART).eq.0) then
             nn(I)=N(1,seg,IPART)
          else
             fh=N(1,seg,IPART)+N(2,seg,IPART)*(abs((x
     &          +N(3,seg,IPART))/N(4,seg,IPART)))**N(5,seg,IPART)
             nn(I)=fh
             if (N(8,seg,IPART).eq.0 .or. N(8,seg,IPART).eq.1) goto 4444
             nn(I)=N(7,seg,IPART)*(abs(fh))**(1.0/N(8,seg,IPART))+
     &             N(6,seg,IPART)
          endif
4444      continue
c
          K=1
          KK=1
c         FIND r, phi, y, z ---------------------------------------------------------------------
          DO J=1,MAXPOINT ! ------------------------------------------LOOP OVER AZIMUTH
             JJ=J
             phi=(JJ-1)/MAXPOINT * pi * 2
               fhw = (hh(I)*ww(I)/4)**nn(I)
               fh = abs((hh(I)/2 * sin(phi)))**nn(I)
               fw = abs((ww(I)/2 * cos(phi)))**nn(I)
               r=(fhw/(fh + fw))**(1 ./ nn(I))                    ! radial coordinates
c
c       --- CONVERT RADIAL TO CARTESIAN COORDINATES ------------------------------------------
               er=0.00001
               y=(r)*sin(phi)+(yo(I))
               zz=(r)*cos(phi)+(zo(I))
c       --- ordering points for icem file ---
               yy(I,J)=y
               zzz(I,J)=zz
          ENDDO    !J LOOPEND
          print *,'nn', nn(I),'hh',hh(I),'x',x,'zz',zo(I),'r', r
c         --- set distribution function ---
          IF(fnv.eq.0) x=x+dx
          IF(fnv.eq.1) x=x*dx
          IF(fnv.eq.2) x=x+x*dx
          IF(fnv.eq.3) x=x+x*dx*(1+dx)
          IF(fnv.eq.4) x=x+x*dx*(1-dx)
          IF(fnv.eq.5) x=x+dx*(1.018-x)
          I=I+1
        ENDDO       !I LOOPEND
        nstations(IPART)=I-1
        print *,'nstations',nstations(IPART)
c       --- WRITE FILE IN ORDER ------------------------------------------------------------
        DO I=1,nstations(IPART)
```

```
          DO J=1,MAXPOINT
              WRITE(61,*),xx(I),yy(I,J),zzz(I,J)
          ENDDO
        ENDDO
        CLOSE(61)
c
c       --- CREATE ICEM REPLAY FILE ------------------------------------------------------------
        fil(IPART) = 'icemscript'//trim(number)//'.rpl'          !icemscript part filename
        OPEN(71,FILE=fil(IPART),STATUS='unknown')
        write(71,*)'ic_set_meshing_params global 0'
        write(71,*)'ic_undo_group_begin'
        write(71,*)'ic_set_meshing_params global 0 gttol 0.00001 gtrel 1'
        write(71,*)'ic_regenerate_tris'
        write(71,*)'ic_undo_group_end'
        write(71,*)'ic_undo_group_begin'
        write(71,*)'ic_geo_cre_geom_input /home/cfd/cjohnson/Fuselage/ROBI
       &N/parametric_fuselage_out'//trim(number)//'.dat 0.00001 input PRT_
       &'//trim(number)//'_PNTS pnt CRVS {} SURFS {}'
        write(71,*)'ic_boco_solver'
        write(71,*)'ic_boco_clear_icons'
        write(71,*)'ic_csystem_display all 0'
        write(71,*)'ic_csystem_set_current global'
        write(71,*)'ic_boco_nastran_csystem reset'
        write(71,*)'ic_geo_new_family GEOM'//trim(number)//''
        write(71,*)'ic_boco_set_part_color GEOM'//trim(number)//''
        IF(pntoption(IPART).eq.1) then
         write(71,*)'ic_point {} PRT_'//trim(number)//'_PNTS pnt.end.0 '//
       &trim(x_st)//','//trim(y_st)//','//trim(z_st)//''
         write(71,*)'ic_point {} PRT_'//trim(number)//'_PNTS pnt.end.1 '//
       &trim(x_en)//','//trim(y_en)//','//trim(z_en)//''
        ENDIF
c        --- rotating option for tail plane or cylindrical hub for example ---
        IF(rotoption(IPART).eq.1) then
c        --- first find axis of rotation ---
           IF (axis.eq.'x') rot_ax = '1 0 0'
           IF (axis.eq.'y') rot_ax = '0 1 0'
           IF (axis.eq.'z') rot_ax = '0 0 1'
           write(71,*)'ic_save_model_for_undo'
           write(71,'(A)',advance='no')'ic_move_geometry point names {'
           DO I=1,nstations(IPART)*MAXPOINT
              WRITE(pnt_c,'(i5)') I-1
              pnt_c = adjustl(pnt_c)
              write(71,'(A)',advance='no')'pnt'//trim(pnt_c)//' '
           ENDDO
           write(71,*)'} rotate '//trim(rotang_c)//' rotate_axis {'//trim(
       &rot_ax)//'} cent {'//trim(rx_c)//' '//trim(ry_c)//' '//trim(rz_c)//
       &'}'
           write(71,*)'ic_geo_reset_data_structures '
           write(71,*)'ic_geo_configure_one_attribute surface shade solid
       &'
        ENDIF
c       --- OPTION 1 TO CREATE LOFT SURFACE ----------------------------------
        IF(option(IPART).eq.0) then
         DO I=1,nstations(IPART)
          WRITE( sta_c,'(i5)') I+ccounter    ! convert station number to character for curvename
          sta_c = adjustl(sta_c)                    ! adjust number so no gaps in filename
          write(71,'(A)',advance='no') 'ic_curve point PRT_'//trim(number)
       &//'_CRV crv.'//trim(sta_c)//' {'
          DO J=1,MAXPOINT
           WRITE(pnt_c,'(i5)') (J-1)+(MAXPOINT*(I-1))+pcounter!convert point number to character for
                pointname
           pnt_c = adjustl(pnt_c)                   !adjust number so no gaps in filename
           write(71,"(A)",advance='no') 'pnt'//trim(pnt_c)//' '
          ENDDO
           if(I.eq.1) mdpoint1=(J-1)+(MAXPOINT*(I-1))+pcounter-MAXPOINT/2! finding midpoint to split curve
                to close end surface
           if(I.eq.nstations(IPART)) mdpoint2=(J-1)+(MAXPOINT*(I-1))+
       &                pcounter-MAXPOINT/2! finding midpoint to split curve to close end surface
           WRITE(pnt_c,'(i5)') (1-1)+(MAXPOINT*(I-1))+pcounter!convert point number to character for
                pointname
           pnt_c = adjustl(pnt_c)                   !adjust number so no gaps in filename
          write(71,*)'pnt'//trim(pnt_c)//'}'
         ENDDO
         WRITE(srf_c,'(i5)') IPART+scounter !convert point number to character for pointname
          srf_c = adjustl(srf_c)                  !adjust number so no gaps in filename
          write(71,'(A)',advance='no')'ic_geo_cre_srf_loft_crvs PRT_'//trim
       &(number)//'_SRF srf.'//trim(srf_c)//' 0.00001 {'
         DO I=1,nstations(IPART)
           WRITE( sta_c,'(i5)') I+stcounter          ! convert station number to character for curvename
           sta_c = adjustl(sta_c)                    ! adjust number so no gaps in filename
           write(71,'(A)',advance='no')'crv.'//trim(sta_c)//' '
         ENDDO
          write(71,*)'} 4 0 1'
          write(71,*)'ic_set_dormant_pickable point 0 {}'
          write(71,*)'ic_set_dormant_pickable curve 0 {}'
c       --- close ends of part - first split, then surface ---
         WRITE(sta_c,'(i5)') ccounter+1
          sta_c=adjustl(sta_c)
         WRITE(pnt_c,'(i5)') mdpoint1
          pnt_c=adjustl(pnt_c)
          write(71,*) 'ic_curve split GEOM crv.'//trim(sta_c)//'a {crv.'//t
       &rim(sta_c)//' pnt'//trim(pnt_c)//'}'
          IF(pntoption(IPART).eq.1) then
           write(71,*)'ic_curve point PRT_'//trim(number)//'_CRV crv.str.0
       &{pnt.end.0 pnt'//trim(pnt_c)//'}'
           write(71,*)'ic_curve point PRT_'//trim(number)//'_CRV crv.str.1
       &{pnt.end.0 pnt0}'
           write(71,*) 'ic_surface 2-4crvs PRT_'//trim(number)//'_SRF srfen
       &d.'//trim(number)//'.a {0.00001 {crv.str.0 crv.'//trim(sta_c)//' c
       &rv.str.1}}'
           write(71,*) 'ic_surface 2-4crvs PRT_'//trim(number)//'_SRF srfen
       &d.'//trim(number)//'.a {0.00001 {crv.str.0 crv.'//trim(sta_c)//'a
       &crv.str.1}}'
```

```
      ELSE
       write(71,*) 'ic_surface 2-4crvs PRT_'//trim(number)//'_SRF srfen
     &d.'//trim(number)//'.a {0.00001 {crv.'//trim(sta_c)//' crv.'//trim
     &(sta_c)//'a}}'
       write(71,*) 'ic_set_dormant_pickable point 0 {}'
       write(71,*) 'ic_set_dormant_pickable curve 0 {}'
      ENDIF

c     --- OPTION 2 TO CREATE nseg SURFACES ----------------------------
      ELSEIF(option(IPART).eq.1) then
       IF(pntoption(IPART).eq.1) then
        write(71,*)'ic_curve point PRT_'//trim(number)//'_CRV crv.str.0
     &{pnt.end.0 pnt0}'
        DO M=1,(nseg-1)
         WRITE(pnt_c,'(i5)') MAXPOINT/nseg*M
         pnt_c=adjustl(pnt_c)                                    ! single point joined to first
              station at 4 points
         WRITE(sta_c,'(i5)') M
         sta_c=adjustl(sta_c)
         write(71,*)'ic_curve point PRT_'//trim(number)//'_CRV crv.str.'
     &//trim(sta_c)//' {pnt.end.0 pnt'//trim(pnt_c)//'}'
        ENDDO
        WRITE(pnt_c,'(i5)') MAXPOINT*(nstations(IPART)-1)
        pnt_c=adjustl(pnt_c)                                     ! single point joined to last
             station at 4 points
        write(71,*)'ic_curve point PRT_'//trim(number)//'_CRV crv.end.0
     &{pnt.end.1 pnt'//trim(pnt_c)//'}'
        DO M=1,(nseg-1)
         WRITE(sta_c,'(i5)') M
         sta_c=adjustl(sta_c)
         WRITE(pnt_c,'(i5)') MAXPOINT*nstations(IPART)-MAXPOINT/nseg*M
         pnt_c=adjustl(pnt_c)
         write(71,*)'ic_curve point PRT_'//trim(number)//'_CRV crv.end.'
     &//trim(sta_c)//' {pnt.end.1 pnt'//trim(pnt_c)//'}'
        ENDDO
       ENDIF
        DO I=1,(nstations(IPART)*nseg)                           ! nseg segments, therefore nseg
             times the curves
         WRITE( sta_c,'(i5)') I+ccounter                         ! so now ccounter counts each
              curve bit
         sta_c = adjustl(sta_c)                                  ! adjust number so no gaps in
              filename
         write(71,'(A)',advance='no') 'ic_curve point PRT_'//trim(number
     &)//'_CRV crv.'//trim(sta_c)//' {'
         DO J=1,(MAXPOINT/nseg)                                  ! count ony upto 1/nseg segment
              points
          WRITE(pnt_c,'(i5)') (J-1)+((MAXPOINT/nseg)*(I-1))+pcounter   ! pcounter takes into account part
               , I takes into account station
          pnt_c = adjustl(pnt_c)                                 ! adjust number so no gaps in
               filename
          write(71,"(A)",advance='no') 'pnt'//trim(pnt_c)//' '
         ENDDO      ! **** END J LOOP
         IF(((J-1)+((MAXPOINT/nseg)*(I-1))+pcounter).NE.
     &       (pcounter+(I+(nseg-1))/nseg*MAXPOINT)) then         ! if its not the end of the curve,
          join curve to next curve-bit's 1st point
          WRITE(pnt_c,'(i5)') (J-1)+((MAXPOINT/nseg)*(I-1))+pcounter
          pnt_c = adjustl(pnt_c)                                 ! adjust number so no gaps in
               filename
         ELSE                                                    ! if it is end of curve, join it
              to start of 1st curve-bits 1st point
           WRITE(pnt_c,'(i5)') (1-1)+((MAXPOINT/nseg)*(I-nseg))+pcounter
           pnt_c = adjustl(pnt_c)                                ! adjust number so no gaps in
                filename
         ENDIF
         write(71,*) 'pnt'//trim(pnt_c)//'}'
        ENDDO       ! *** END I LOOP
        J=0
        minfcounter=fcounter
        DO WHILE(J.lt.(MAXPOINT * (nstations(IPART)-1)))
          WRITE(pnt_c,'(i5)') J+pcounter  ! first curve point
          pnt_c=adjustl(pnt_c)
          WRITE(pnt_cn,'(i5)') J+pcounter+MAXPOINT ! second curve point
          pnt_cn=adjustl(pnt_cn)
          WRITE(sta_c,'(i5)') fcounter
          sta_c = adjustl(sta_c)
          write(71,*)'ic_curve point PRT_'//trim(number)//'_CRV crvf.'/
     &/trim(sta_c)//' {pnt'//trim(pnt_c)//' pnt'//trim(pnt_cn)//'}'     ! write longitundinal fuselage
         curves
         J=J+(MAXPOINT/nseg)
         fcounter=fcounter+1
        ENDDO       ! *** END WHILE LOOP
       maxfcounter=fcounter-1
       DO I=minfcounter,(maxfcounter-nseg)
        WRITE(sta_c,'(i5)') I
        sta_c = adjustl(sta_c)                                    ! adjust number so no gaps in
             filename
        WRITE(sta_cn,'(i5)') I+nseg
        sta_cn = adjustl(sta_cn)                                  ! adjust number so no gaps in
             filename
        write(71,*) 'ic_geo_mod_crv_match_crv crvf.'//trim(sta_c)//' cr
     &vf.'//trim(sta_cn)//' 2 1 {2 1 1 0 0}'
        if(I.lt.(minfcounter+nseg)) then
         write(71,*) 'ic_delete_geometry curve names crvf.'//trim(sta_c
     &)//' 0'
         write(71,*) 'ic_geo_set_name curve crvf.'//trim(sta_c)//'.1 cr
     &vf.'//trim(sta_c)//''
        endif
        write(71,*) 'ic_delete_geometry curve names crvf.'//trim(sta_cn
     &)//' 0'
        write(71,*) 'ic_geo_set_name curve crvf.'//trim(sta_cn)//'.1 cr
     &vf.'//trim(sta_cn)//''
       ENDDO
      IF(pntoption(IPART).eq.1) then
```

```
      DO i=0,(nseg-1)                                                  ! match curves at start and end to
           the main body
       WRITE(sta_c,'(i5)') I
       sta_c = adjustl(sta_c)
       write(71,*)'ic_geo_mod_crv_match_crv crv.str.'//trim(sta_c)//'
     &crvf.'//trim(sta_c)//' 2 1 {2 1 1 1 0}'
       write(71,*)'ic_delete_geometry curve names crv.str.'//trim(sta_
     &c)//' 0'
       write(71,*)'ic_geo_set_name curve crv.str.'//trim(sta_c)//'.1 c
     &rv.str.'//trim(sta_c)//''                                        ! end of matching for front
       WRITE(sta_c,'(i5)') (nseg-1)-I
       sta_c = adjustl(sta_c)
       WRITE(sta_cn,'(i5)') maxfcounter-(nseg-1)+I
       sta_cn = adjustl(sta_cn)
       write(71,*)'ic_geo_mod_crv_match_crv crv.end.'//trim(sta_c)//'
     &crvf.'//trim(sta_cn)//' 2 2 {2 1 1 1 0}'
       write(71,*)'ic_delete_geometry curve names crv.end.'//trim(sta_
     &c)//' 0'
       write(71,*)'ic_geo_set_name curve crv.end.'//trim(sta_c)//'.1 c
     &rv.end.'//trim(sta_c)//''
      ENDDO
      DO M=0,(nseg/2-1)
       WRITE(sta_c,'(i5)') M
       sta_c = adjustl(sta_c)
       WRITE(sta_cn,'(i5)') nseg/2+M
       sta_cn = adjustl(sta_cn)
       write(71,*)'ic_geo_mod_crv_match_crv crv.str.'//trim(sta_c)//'
     &crv.str.'//trim(sta_cn)//' 1 1 {2 1 1 0 0}'
       write(71,*)'ic_geo_mod_crv_match_crv crv.end.'//trim(sta_c)//'
     &crv.end.'//trim(sta_cn)//' 1 1 {2 1 1 0 0}'
      ENDDO
     ENDIF
      counter4=0
      DO I=minfcounter,(maxfcounter)
       counter4=counter4+1                                             ! to check if its the 4th curve
           so it loops back to first curve
       WRITE(sta_c,'(i5)') I
       sta_c = adjustl(sta_c)                                          ! adjust number so no gaps in
           filename
       WRITE(sta_c5,'(i5)') ccounter+I-minfcounter+(nseg+1)
       sta_c5 = adjustl(sta_c5)                                        ! adjust number so no gaps in
           filename
       WRITE(sta_c3,'(i5)') I-(nseg-1)
       sta_c3 = adjustl(sta_c3)                                        ! adjust number so no gaps in
           filename
       WRITE(sta_cf1,'(i5)') I+1
       sta_cf1 = adjustl(sta_cf1)                                      ! adjust number so no gaps in
           filename
       WRITE(sta_c1,'(i5)') ccounter+I-minfcounter+1
       sta_c1 = adjustl(sta_c1)                                        ! adjust number so no gaps in
           filename
       if(counter4.ne.nseg) then
        write(71,*) 'ic_surface 2-4crvs PRT_'//trim(number)//'_SRF srf
     &c.'//trim(sta_c)//' {0.00001 {crvf.'//trim(sta_c)//' crv.'//trim(s
     &ta_c5)//' crvf.'//trim(sta_cf1)//' crv.'//trim(sta_c1)//'}}'
       else
        write(71,*) 'ic_surface 2-4crvs PRT_'//trim(number)//'_SRF srf
     &c.'//trim(sta_c)//' {0.00001 {crvf.'//trim(sta_c)//' crv.'//trim(s
     &ta_c5)//' crvf.'//trim(sta_c3)//' crv.'//trim(sta_c1)//'}}'
        counter4=0
       endif
       write(71,*) 'ic_set_dormant_pickable point 0 {}'
       write(71,*) 'ic_set_dormant_pickable curve 0 {}'
      ENDDO
c     --- make closing surface at start of part ---
      IF(pntoption(IPART).eq.1) then
       DO I=0,(nseg-1)
        WRITE(sta_c,'(i5)') I
        sta_c = adjustl(sta_c)
        IF (I.eq.(nseg-1)) then
         WRITE(sta_cn,'(i5)') I-(nseg-1)
         WRITE(sta_c1,'(i5)') I+1
        ELSE
         WRITE(sta_cn,'(i5)') I+1
         WRITE(sta_c1,'(i5)') I+1
        ENDIF
        sta_cn = adjustl(sta_cn)
        sta_c1 = adjustl(sta_c1)
        write(71,*)'ic_surface 2-4crvs PRT_'//trim(number)//'_SRF srf.s
     &tr.'//trim(sta_c)//' {0.00001 {crv.str.'//trim(sta_c)//' crv.'//tr
     &im(sta_c1)//' crv.str.'//trim(sta_cn)//'}}'
       ENDDO
      ELSE
       write(71,'(A)',advance='no')'ic_geo_cre_crv_concat PRT_'//trim(n
     &umber)//'_CRV crv.c.str1 0.000001 {'
       DO I=1,nseg/2
        WRITE(sta_c,'(i5)') I
        sta_c = adjustl(sta_c)
        write(71,'(A)',advance='no')'crv.'//trim(sta_c)//' '
       ENDDO
       write(71,*) '}'
       write(71,'(A)',advance='no')'ic_geo_cre_crv_concat PRT_'//trim(n
     &umber)//'_CRV crv.c.str0 0.000001 {'
       DO I=nseg/2+1,nseg
        WRITE(sta_c,'(i5)') I
        sta_c = adjustl(sta_c)
        write(71,'(A)',advance='no')'crv.'//trim(sta_c)//' '
       ENDDO
       write(71,*) '}'
       write(71,'(A)',advance='no') 'ic_surface 2-4crvs PRT_'//trim(num
     &ber)//'_SRF srfend.a.'//trim(number)//' {0.00001 {crv.c.str1 crv.c
     &.str0}}'
       write(71,*) ''
```

```
          write(71,*) 'ic_set_dormant_pickable point 0 {}'
          write(71,*) 'ic_set_dormant_pickable curve 0 {}'
         ENDIF
        ENDIF
c       --- add the right count value for part depending on segmented or full curves
        IF(option(IPART).eq.0) then
         pcounter=pcounter+MAXPOINT*nstations(IPART)
         ccounter=ccounter+nstations(IPART)
         scounter=scounter+0
         stcounter=stcounter+nstations(IPART)
c       --- close ends of part - first split, then surface ---
         WRITE(sta_c,'(i5)') ccounter
         sta_c=adjustl(sta_c)
         WRITE(pnt_c,'(i5)') mdpoint2
         pnt_c=adjustl(pnt_c)
         write(71,*) 'ic_curve split GEOM crv.'//trim(sta_c)//'a {crv.'//t
     &rim(sta_c)//' pnt'//trim(pnt_c)//'}'
         IF(pntoption(IPART).eq.1) then
           WRITE(pnt_c,'(i5)') pcounter-MAXPOINT
           pnt_c = adjustl(pnt_c)
           write(71,*)'ic_curve point PRT_'//trim(number)//'_CRV crv.end.1
     & {pnt.end.1 pnt'//trim(pnt_c)//'}'
           WRITE(pnt_c,'(i5)') mdpoint2
           pnt_c = adjustl(pnt_c)
           write(71,*)'ic_curve point PRT_'//trim(number)//'_CRV crv.end.0
     & {pnt.end.1 pnt'//trim(pnt_c)//'}'
           WRITE(sta_c1,'(i5)') ccounter
           sta_c1 = adjustl(sta_c1)
           write(71,*)'ic_surface 2-4crvs PRT_'//trim(number)//'_SRF srf.e
     &nd.'//trim(sta_c)//' {0.00001 {crv.end.1 crv.'//trim(sta_c1)//' cr
     &v.end.0}}'
           write(71,*)'ic_surface 2-4crvs PRT_'//trim(number)//'_SRF srf.e
     &nd.'//trim(sta_c)//' {0.00001 {crv.end.1 crv.'//trim(sta_c1)//'a c
     &rv.end.0}}'
         ELSE
           write(71,*) 'ic_surface 2-4crvs PRT_'//trim(number)//'_SRF srfen
     &d.'//trim(number)//'.a {0.00001 {crv.'//trim(sta_c)//' crv.'//trim
     &(sta_c)//'a}}'
           write(71,*) 'ic_set_dormant_pickable point 0 {}'
           write(71,*) 'ic_set_dormant_pickable curve 0 {}'
           write(71,*)'ic_undo_group_end'
         ENDIF
        ELSE          ! if the option is segmented curves, then nseg times the number of stations per part.
         pcounter=pcounter+MAXPOINT*nstations(IPART)
         ccounter=ccounter+nseg*nstations(IPART)
         stcounter=stcounter+nseg*nstations(IPART)
c
c       --- make closing surface at end of part ---
         IF(pntoption(IPART).eq.1) then
         DO I=0,(nseg-1)
           WRITE(sta_c,'(i5)') I
           sta_c = adjustl(sta_c)
           IF (I.eq.(nseg-1)) then
            WRITE(sta_cn,'(i5)') I-(nseg-1)
            WRITE(sta_c1,'(i5)') nstations(IPART)*nseg
           ELSE
            WRITE(sta_cn,'(i5)') I+1
            WRITE(sta_c1,'(i5)') nstations(IPART)*nseg-I-1
           ENDIF
           sta_cn = adjustl(sta_cn)
           sta_c1 = adjustl(sta_c1)
           write(71,*)'ic_surface 2-4crvs PRT_'//trim(number)//'_SRF srf.e
     &nd.'//trim(sta_c)//' {0.00001 {crv.end.'//trim(sta_c)//' crv.'//tr
     &im(sta_c1)//' crv.end.'//trim(sta_cn)//'}}'
           write(71,*) 'ic_set_dormant_pickable point 0 {}'
           write(71,*) 'ic_set_dormant_pickable curve 0 {}'
         ENDDO
         ELSE
           write(71,'(A)',advance='no')'ic_geo_cre_crv_concat PRT_'//trim(n
     &umber)//'_CRV crv.c.end1 0.000001 {'
          DO I=0,nseg/2-1
           WRITE(sta_c,'(i5)') ccounter-I
           sta_c = adjustl(sta_c)
           write(71,'(A)',advance='no')'crv.'//trim(sta_c)//' '
          ENDDO
           write(71,*) '}'
           write(71,'(A)',advance='no')'ic_geo_cre_crv_concat PRT_'//trim(n
     &umber)//'_CRV crv.c.end0 0.000001 {'
          DO I=1,nseg/2
           WRITE(sta_c,'(i5)') ccounter-nseg+I
           sta_c = adjustl(sta_c)
           write(71,'(A)',advance='no')'crv.'//trim(sta_c)//' '
          ENDDO
           write(71,*) '}'
           write(71,'(A)',advance='no') 'ic_surface 2-4crvs PRT_'//trim(num
     &ber)//'_SRF srfend.b.'//trim(number)//' {0.00001 {crv.c.end1 crv.c
     &.end0}}'
           write(71,*) ''
           write(71,*) 'ic_set_dormant_pickable point 0 {}'
           write(71,*) 'ic_set_dormant_pickable curve 0 {}'
         ENDIF
        ENDIF
         write(71,*)'ic_save_tetin /home/cfd/cjohnson/Fuselage/ROBIN/robin_
     &scriptpart'//trim(number)//'.tin 0 0 {} {} 0'
         CLOSE(71)
         print *,'p',pcounter,'c',ccounter,'s',scounter,'f',fcounter,'stati
     &on',stcounter
        ENDDO          ! IPARTS LOOP
c
c
        END
```

B.8.2 Program to Create New Fuselage from Data Points

This program is the same as the one in B.8.1 except that it reads data points directly and creates the replay file. An example input file is given in Appendix B.8.4.

```
      PROGRAM ROBIN
c
c PROGRAM to find the coordinates of a fuselage using the robin body typre parameterisation.
c From NASA paper Appendix in JAXA folder.
c
c Option 0 does a lofting surface with ful curves, Option 1 makes segmented curves.
c rotoption allows you to rotate the body.
c*************************************************************************************
c
      INTEGER MAXSTATIONS,MAXPART,MAXSEG,MAXPOINT,PTS
      PARAMETER(MAXSTATIONS=50,MAXPART=10,MAXSEG=4,MAXPOINT=52)
      INTEGER I,J,seg,num(MAXPART),K,KK,fn(MAXSEG,MAXPART),fnv
      REAL delta(MAXSEG,MAXPART)
      COMPLEX fh,r,fhw,fw
      COMPLEX hh(MAXPOINT),ww(MAXPOINT),zo(MAXPOINT),nn(MAXPOINT)
      COMPLEX yo(MAXPOINT)
      REAL x,l,y,zz,phi,pi,II,JJ
      REAL H(8,MAXSEG,MAXPART),W(8,MAXSEG,MAXPART)
      REAL Z(8,MAXSEG,MAXPART),N(8,MAXSEG,MAXPART)
      REAL YF(8,MAXSEG,MAXPART)
      REAL xu(MAXSEG,MAXPART),xl(MAXSEG,MAXPART),segm(MAXSEG,MAXPART)
      REAL dx,xstart(MAXPART),xend(MAXPART)
      REAL xx(MAXSTATIONS,MAXPOINT),yy(MAXSTATIONS,MAXPOINT)
      REAL zzz(MAXSTATIONS,MAXPOINT)
      CHARACTER*100 filenam,filename(MAXPART),fil(MAXPART),sta_c,pnt_c
      CHARACTER(len=100)::sta_x,sta_y,sta_z,srf_c,pnt_cn,sta_cn,rot_ax
      CHARACTER(len=100)::sta_c1,sta_c5,sta_c3,sta_cf1,rx_c,ry_c,rz_c
      CHARACTER(len=100)::rotang_c
      CHARACTER(len=2)::number, axis
      INTEGER IPART,nstations(MAXSTATIONS),NPARTS,option(MAXPART),III
      INTEGER pcounter,scounter,ccounter,stcounter,fcounter,maxfcounter
      INTEGER minfcounter,counter4,mdpoint1,mdpoint2,nicem,rotoption
29    FORMAT(E3.2)
c
      pi=3.141592654
c
c     --- DEFINE COEFFICIENTS FOR HEIGHT, WIDTH, CAMBER, ELLIPTIC POWER ----------
      print*,'Enter number of existing ICEM part geometries:'
      read(5,*)nicem
      WRITE( number, '(i2)' ) nicem+1
      number = adjustl(number)
      OPEN (51,file='parametric_fuselage_readpoints'//trim(number)//'.da
     &t',STATUS='OLD')
      IPART=1
      read(51,*)num(IPART),xstart(IPART),xend(IPART)                    ! read the number of segments and
           the start and end x co-ordinates of the part
      DO J=1,num(IPART)
           READ(51,*) segm(J,IPART),xl(J,IPART),xu(J,IPART),nstations(IPA
     &RT),rotoption
           READ(51,*) rx_c,ry_c,rz_c
           READ(51,*) axis, rotang_c
      ENDDO
      IPART=IPART+1
1000  CLOSE (51)
      NPARTS=IPART-1
      print*,'cent: ',rx_c,ry_c,rz_c,axis
c
c     --- DEFINE OUTPUT FILE AND X STATIONS ---------------------------------------
      pcounter=0
      ccounter=0
      scounter=0
      stcounter=0
      fcounter=0
      option(IPART)=1

      DO IPART=1,NPARTS                    ! ---------------------------------LOOP OVER PARTS
      WRITE( number, '(i2)' ) nicem+IPART  ! convert part number to character for filename
      number = adjustl(number)             ! adjust number so no gaps in filename
      print *,'nstations',nstations(IPART)
c
c     --- CREATE ICEM REPLAY FILE -------------------------------------------------
      fil(IPART) = 'icemscript'//trim(number)//'.rpl'         !icemscript part filename
      OPEN(71,FILE=fil(IPART),STATUS='unknown')
      write(71,*)'ic_set_meshing_params global 0'
      write(71,*)'ic_undo_group_begin'
      write(71,*)'ic_set_meshing_params global 0 gttol 0.00001 gtrel 1'
      write(71,*)'ic_regenerate_tris'
      write(71,*)'ic_undo_group_end'
      write(71,*)'ic_undo_group_begin'
      write(71,*)'ic_geo_cre_geom_input /home/cfd/cjohnson/Fuselage/ROBI
     &N/parametric_fuselage_out_pnts'//trim(number)//'.dat 0.00001 input
     & PRT_'//trim(number)//'_PNTS pnt CRVS {} SURFS {}'
      write(71,*)'ic_boco_solver'
      write(71,*)'ic_boco_clear_icons'
      write(71,*)'ic_csystem_display all 0'
      write(71,*)'ic_csystem_set_current global'
      write(71,*)'ic_boco_nastran_csystem reset'
      write(71,*)'ic_geo_new_family GEOM'//trim(number)//''
      write(71,*)'ic_boco_set_part_color GEOM'//trim(number)//''
c     --- OPTION 2 TO CREATE 4 SURFACES -------------------------------
        DO I=1,(nstations(IPART)*4)                                     ! 4 segments, therefore 4 times
             the curves
         WRITE( sta_c,'(i5)') I+ccounter                                ! so now ccounter counts each
              curve bit
         sta_c = adjustl(sta_c)                                         ! adjust number so no gaps in
              filename
         write(71,'(A)',advance='no') 'ic_curve point PRT_'//trim(number
```

```
     &)//'_CRV crv.'//trim(sta_c)//' {'
          DO J=1,(MAXPOINT/4)                                    ! count ony upto 1/4 segment
               points
           WRITE(pnt_c,'(i5)') (J-1)+((MAXPOINT/4)*(I-1))+pcounter      ! pcounter takes into account part
               , I takes into account station
            pnt_c = adjustl(pnt_c)                               ! adjust number so no gaps in
                 filename
           write(71,"(A)",advance='no') 'pnt'//trim(pnt_c)//' '
          ENDDO      ! **** END J LOOP
          IF(((J-1)+((MAXPOINT/4)*(I-1))+pcounter).NE.
     &      (pcounter+(I+3)/4*MAXPOINT)) then                    ! if its not the end of the curve,
           join curve to next curve-bit's 1st point
           WRITE(pnt_c,'(i5)') (J-1)+((MAXPOINT/4)*(I-1))+pcounter
           pnt_c = adjustl(pnt_c)                                ! adjust number so no gaps in
                filename
          ELSE                                                   ! if it is end of curve, join it
               to start of 1st curve-bits 1st point
            WRITE(pnt_c,'(i5)') (1-1)+((MAXPOINT/4)*(I-4))+pcounter
            pnt_c = adjustl(pnt_c)                               ! adjust number so no gaps in
                 filename
          ENDIF
          write(71,*) 'pnt'//trim(pnt_c)//'}'
         ENDDO       ! *** END I LOOP
          J=0
          minfcounter=fcounter
          DO WHILE(J.lt.(MAXPOINT * (nstations(IPART)-1)))
            WRITE(pnt_c,'(i5)') J+pcounter  ! first curve point
            pnt_c=adjustl(pnt_c)
            WRITE(pnt_cn,'(i5)') J+pcounter+MAXPOINT ! second curve point
            pnt_cn=adjustl(pnt_cn)
            WRITE(sta_c,'(i5)') fcounter
            sta_c = adjustl(sta_c)
            write(71,*)'ic_curve point PRT_'//trim(number)//'_CRV crvf.'/
     &/trim(sta_c)//' {pnt'//trim(pnt_c)//' pnt'//trim(pnt_cn)//'}'     ! write longitundinal fuselage
           curves
           J=J+(MAXPOINT/4)
           fcounter=fcounter+1
          ENDDO       ! *** END WHILE LOOP
         maxfcounter=fcounter-1
         DO I=minfcounter,(maxfcounter-4)
          WRITE(sta_c,'(i5)') I
          sta_c = adjustl(sta_c)                                 ! adjust number so no gaps in
               filename
          WRITE(sta_cn,'(i5)') I+4
          sta_cn = adjustl(sta_cn)                               ! adjust number so no gaps in
               filename
          write(71,*) 'ic_geo_mod_crv_match_crv crvf.'//trim(sta_c)//' cr
     &vf.'//trim(sta_cn)//' 2 1 {2 1 1 0 0}'
          if(I.lt.(minfcounter+4)) then
           write(71,*) 'ic_delete_geometry curve names crvf.'//trim(sta_c
     &)//' 0'
           write(71,*) 'ic_geo_set_name curve crvf.'//trim(sta_c)//'.1 cr
     &vf.'//trim(sta_c)//''
          endif
          write(71,*) 'ic_delete_geometry curve names crvf.'//trim(sta_cn
     &)//' 0'
          write(71,*) 'ic_geo_set_name curve crvf.'//trim(sta_cn)//'.1 cr
     &vf.'//trim(sta_cn)//''
         ENDDO
         counter4=0
         DO I=minfcounter,(maxfcounter)
          counter4=counter4+1                                    ! to check if its the 4th curve
               so it loops back to first curve
          WRITE(sta_c,'(i5)') I
          sta_c = adjustl(sta_c)                                 ! adjust number so no gaps in
               filename
          WRITE(sta_c5,'(i5)') ccounter+I-minfcounter+5
          sta_c5 = adjustl(sta_c5)                               ! adjust number so no gaps in
               filename
          WRITE(sta_c3,'(i5)') I-3
          sta_c3 = adjustl(sta_c3)                               ! adjust number so no gaps in
               filename
          WRITE(sta_cf1,'(i5)') I+1
          sta_cf1 = adjustl(sta_cf1)                             ! adjust number so no gaps in
               filename
          WRITE(sta_c1,'(i5)') ccounter+I-minfcounter+1
          sta_c1 = adjustl(sta_c1)                               ! adjust number so no gaps in
               filename
          if(counter4.ne.4) then
           write(71,*) 'ic_surface 2-4crvs PRT_'//trim(number)//'_SRF srf
     &c.'//trim(sta_c)//' {0.0001 {crvf.'//trim(sta_c)//' crv.'//trim(s
     &ta_c5)//' crvf.'//trim(sta_cf1)//' crv.'//trim(sta_c1)//'}}'
          else
           write(71,*) 'ic_surface 2-4crvs PRT_'//trim(number)//'_SRF srf
     &c.'//trim(sta_c)//' {0.0001 {crvf.'//trim(sta_c)//' crv.'//trim(s
     &ta_c5)//' crvf.'//trim(sta_c3)//' crv.'//trim(sta_c1)//'}}'
           counter4=0
          endif
          write(71,*) 'ic_set_dormant_pickable point 0 {}'
          write(71,*) 'ic_set_dormant_pickable curve 0 {}'
         ENDDO
c        --- make closing surface at start of part ---
         write(71,'(A)',advance='no') 'ic_surface 2-4crvs PRT_'//trim(numb
     &er)//'_SRF srfend.a.'//trim(number)//' {0.0001 {'
         DO I=1,4
          WRITE(sta_c,'(i5)') ccounter+I
          sta_c = adjustl(sta_c)
          write(71,'(A)',advance='no')'crv.'//trim(sta_c)//' '
         ENDDO
         write(71,*) '}}'
         write(71,*) 'ic_set_dormant_pickable point 0 {}'
         write(71,*) 'ic_set_dormant_pickable curve 0 {}'
c       --- add the right count value for part depending on segmented or full curves
```

```
        pcounter=pcounter+MAXPOINT*nstations(IPART)
        ccounter=ccounter+4*nstations(IPART)
        stcounter=stcounter+4*nstations(IPART)
c
c       --- make closing surface at end of part ---
        write(71,'(A)',advance='no') 'ic_surface 2-4crvs PRT_'//trim(numb
     &er)//'_SRF srfend.b.'//trim(number)//' {0.0001 {'
        DO I=0,3
         WRITE(sta_c,'(i5)') ccounter-I
         sta_c = adjustl(sta_c)
         write(71,'(A)',advance='no')'crv.'//trim(sta_c)//' '
        ENDDO
        write(71,*) '}}'
        write(71,*) 'ic_set_dormant_pickable point 0 {}'
        write(71,*) 'ic_set_dormant_pickable curve 0 {}'
c       --- rotating option for tail plane for example ---
        IF(rotoption.eq.1) then
c       --- first find axis of rotation ---
           write(71,*)'ic_save_model_for_undo'
           write(71,'(A)',advance='no')'ic_move_geometry surface names {sr
     &fend.a.'//trim(number)//' srfend.b.'//trim(number)//' '
           DO I=1,fcounter
             WRITE(sta_c,'(i5)') I-1
             sta_c = adjustl(sta_c)
             write(71,'(A)',advance='no')'srfc.'//trim(sta_c)//' '
           ENDDO
           IF (axis.eq.'x') rot_ax = '1 0 0'
           IF (axis.eq.'y') rot_ax = '0 1 0'
           IF (axis.eq.'z') rot_ax = '0 0 1'
           write(71,*)'} rotate '//trim(rotang_c)//' rotate_axis {'//trim(
     &rot_ax)//'} cent {'//trim(rx_c)//' '//trim(ry_c)//' '//trim(rz_c)//
     &/'}'
           write(71,'(A)',advance='no')'ic_move_geometry curve names {'
           DO I=1,ccounter
              WRITE(sta_c,'(i5)') I
              sta_c = adjustl(sta_c)
              write(71,'(A)',advance='no')'crv.'//trim(sta_c)//' '
           ENDDO
           DO I=1,fcounter
             WRITE(sta_c,'(i5)') I-1
             sta_c = adjustl(sta_c)
             write(71,'(A)',advance='no')'crvf.'//trim(sta_c)//' '
           ENDDO
           write(71,*)'} rotate '//trim(rotang_c)//' rotate_axis {'//trim(
     &rot_ax)//'} cent {'//trim(rx_c)//' '//trim(ry_c)//' '//trim(rz_c)//
     &/'}'
           write(71,'(A)',advance='no')'ic_move_geometry point names {'
           DO I=1,pcounter
              WRITE(pnt_c,'(i5)') I-1
              pnt_c = adjustl(pnt_c)
              write(71,'(A)',advance='no')'pnt'//trim(pnt_c)//' '
           ENDDO
           write(71,*)'} rotate '//trim(rotang_c)//' rotate_axis {'//trim(
     &rot_ax)//'} cent {'//trim(rx_c)//' '//trim(ry_c)//' '//trim(rz_c)//
     &/'}'
           write(71,*)'ic_geo_reset_data_structures '
           write(71,*)'ic_geo_configure_one_attribute surface shade solid
     &'
        ENDIF
        write(71,*)'ic_save_tetin /home/cfd/cjohnson/Fuselage/ROBIN/robin
     &_scriptpart'//trim(number)//'.tin 0 0 {} {} 0'
        CLOSE(71)
       print *,'p',pcounter,'c',ccounter,'f',fcounter,'station',stcounter
       ENDDO          ! IPARTS LOOP
c
       CLOSE(61)
c
       END
```

B.8.3 Parametric Fuselage data files for parametric_fuselage.f

B.8.4 Parametric Fuselage data files for parametric_fuselage_separateparts.f

parametric_fuselage1.dat:

```
MainBody 1
4 0.001 1.99
1 0.001 0.4 1 1.6
 1.000 -1.000 -0.400  0.400  1.800  0.000  0.250  1.800
 1.000 -1.000 -0.400  0.400  2.000  0.000  0.250  2.000
 0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 1.000 -1.000 -0.400  0.400  1.800 -0.080  0.080  1.800
 2.000  3.000  0.000  0.400  1.000  0.000  1.000  1.000
2 0.4 0.8 0 0.1
 0.250  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 0.250  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
```

Mainbody	0						
4	0.001	1.99					
1	0.001	0.4	1	1.6			
1.000	-1.000	-0.400	0.400	1.800	0.000	0.250	1.800
1.000	-1.000	-0.400	0.400	2.000	0.000	0.250	2.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1.000	-1.000	-0.400	0.400	1.800	-0.080	0.080	1.800
2.000	3.000	0.000	0.400	1.000	0.000	1.000	1.000
2	0.4	0.8	0	0.1			
0.250	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.250	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.8	1.9	0	0.1			
1.000	-1.000	-0.800	1.100	1.500	0.050	0.200	0.600
1.000	-1.000	-0.800	1.100	1.500	0.050	0.200	0.600
0.000	-0.000	-0.000	0.000	0.000	0.000	-0.000	0.000
1.000	-1.000	-0.800	1.100	1.500	0.040	-0.040	0.600
5.000	-3.000	-0.800	1.100	1.000	0.000	0.000	0.000
4	1.9	2.0	4	0.002			
1.000	-1.000	-1.900	0.100	2.000	0.000	0.050	2.000
1.000	-1.000	-1.900	0.100	2.000	0.000	0.050	2.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.040	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Pylon	1						
2	0.40001	1.01799					
1	0.4	0.8	3	0.035			
1.000	-1.000	-0.800	0.400	3.000	0.000	0.200	3.000
1.000	-1.000	-0.800	0.400	3.000	0.000	0.172	3.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.125	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.8	1.018	0	0.05			
1.000	-1.000	-0.800	0.218	2.000	0.000	0.200	2.000
1.000	-1.000	-0.800	0.218	2.000	0.000	0.172	2.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1.000	-1.000	-0.800	1.100	1.500	0.065	0.060	0.600
5.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table B.1: *Example input file with 2 parts : fuselage (containing 4 segments) and pylon (containing 2 segments).*

```
 0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 5.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
3 0.8 1.9 0 0.1
 1.000 -1.000 -0.800  1.100  1.500  0.050  0.200  0.600
 1.000 -1.000 -0.800  1.100  1.500  0.050  0.200  0.600
 0.000 -0.000 -0.000  0.000  0.000  0.000 -0.000  0.000
 1.000 -1.000 -0.800  1.100  1.500  0.040 -0.040  0.600
 5.000 -3.000 -0.800  1.100  1.000  0.000  0.000  0.000
4 1.9 2.0 4 0.002
 1.000 -1.000 -1.900  0.100  2.000  0.000  0.050  2.000
 1.000 -1.000 -1.900  0.100  2.000  0.000  0.050  2.000
 0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 0.040  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 2.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
```

parametric_fuselage2.dat

```
Pylon 1
2 0.40001 1.01799
1 0.4 0.8 3 0.035
 1.000 -1.000 -0.800 0.400  3.000  0.000  0.200  3.000
 1.000 -1.000 -0.800 0.400  3.000  0.000  0.172  3.000
 0.000  0.000  0.000 0.000  0.000  0.000  0.000  0.000
 0.125  0.000  0.000 0.000  0.000  0.000  0.000  0.000
 5.000  0.000  0.000 0.000  0.000  0.000  0.000  0.000
2 0.8 1.018 0 0.05
 1.000 -1.000 -0.800 0.218  2.000  0.000  0.200  2.000
 1.000 -1.000 -0.800 0.218  2.000  0.000  0.172  2.000
 0.000  0.000  0.000 0.000  0.000  0.000  0.000  0.000
 1.000 -1.000 -0.800 1.100  1.500  0.065  0.060  0.600
 5.000  0.000  0.000 0.000  0.000  0.000  0.000  0.000
```

parametric_fuselage3.dat

```
Sponson1 0
2 0.50001 1.0001
1 0.5 0.85 0 0.02
 1.000 -1.000 -0.800 0.400  2.000  0.000  0.050  0.500
 1.000 -1.000 -0.800 0.400  2.000  0.000  0.050  0.500
 0.125  0.000  0.000 0.000  0.000  0.000  0.000  0.000
 -.050  1.000  0.000 0.000  0.000  0.000  0.000  0.000
 5.000  0.000  0.000 0.000  0.000  0.000  0.000  0.000
2 0.85 1.1 0 0.02
 1.000 -1.000 -0.800 0.218  2.000  0.000  0.050  1.100
 1.000 -1.000 -0.800 0.218  2.000  0.000  0.050  1.100
 0.125  0.000  0.000 0.000  0.000  0.000  0.000  0.000
 -.050  1.000  0.000 0.000  0.000  0.000  0.000  0.000
 5.000  0.000  0.000 0.000  0.000  0.000  0.000  0.000
Sponson2 0
2 0.50001 1.0001
1 0.5 0.85 0 0.02
 1.000 -1.000 -0.800 0.400  2.000  0.000  0.050  0.500
 1.000 -1.000 -0.800 0.400  2.000  0.000  0.050  0.500
 -.125  0.000  0.000 0.000  0.000  0.000  0.000  0.000
 -.050  1.000  0.000 0.000  0.000  0.000  0.000  0.000
```

```
 5.000  0.000  0.000 0.000  0.000  0.000  0.000  0.000
2 0.85 1.1 0 0.02
 1.000 -1.000 -0.800 0.218  2.000  0.000  0.050  1.100
 1.000 -1.000 -0.800 0.218  2.000  0.000  0.050  1.100
 -.125  0.000  0.000 0.000  0.000  0.000  0.000  0.000
 -.050  1.000  0.000 0.000  0.000  0.000  0.000  0.000
 5.000  0.000  0.000 0.000  0.000  0.000  0.000  0.000
```

B.9 BERP Tip Parameterisation

For the BERP like parameterisation, the BERP-like tip was defined by leading edge and trailing edge planform curves and scale factors for the aerofoils to maintian linearly decreasing thickness. Program B.9.1 creates the trailing edge planform, program B.9.2 creates the leading edge planform and program B.9.3 combines these two curves and also creates a set of scale factors for the chords of the aerofoils and their required thickness to create the BERP-like tip. The programs must be run in that order. More details on how this is done and steps on how to generate the grid can be found in the Technical Note.[153] Section 3.2.2 of the thesis describes the theory.

B.9.1 Program to Create Planform Co-ordinates for Trailing Edge

```
/*
 * ------------------------------ BERP-LIKE TIP PARAMETERISATION ------------------------------
 * This program parameterises a BERP-like rotor tip T.E. using the sweep gradient, sw, as the
 * gradient of a linear curve.
 * Some parameters are automatically obtained, such as the the zero gradient points of the
 * sweep and the delta polynomial.
 * A number of files are required:
 * 1. The parameter.txt - contains the 3 parameters.
 * 2. The defining_xpts.txt - defines the x coordinates where the parameters start and stop.
 * 3. The stations.txt - defines the x coordinates - (will be useful when making ICEM grid).
 * 4. An output file to write the data to - this is an argument to the run command.
 * This program must be run before the leading edge define.c one as it creates the value ytip
 * which is used in the other program.
 * And this ytip value must be deleted from the defining_xpts.txt file each time the program
 * is run as it appends it to the file.
 * ------------------------------------------------------------------------------------------
 */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>

main(int argc, char *argv[])
{
FILE *input, *define, *param, *output;
char filename[128];

int i,j,jj;
float x,y,xns,xne,xse,xde,dy,xne_orig,xse_orig,yadd,ytetip;
float lastpoint,firstpoint,miny,maxy;
float g,sw,del,zg_pt,dy1,Area,AR,swte;
float XX[100],YY[100];

/*--- OPEN AND READ INPUT FILE ---*/
param = fopen("parameters.txt","r");
fscanf(param,"g: %f\n",&g);                 // gradient of the notch sigmoid
fscanf(param,"sw: %f\n",&sw);               // gradient of the linear sweep equation - same as for L.E.
fscanf(param,"del: %f\n",&del);             // not required - only used in L.E.
fscanf(param,"swte: %f\n",&swte);           // factor to multiply the gradient by to change the gradient
    compared to L.E. = movement of tip point.
fclose(param);

define = fopen("defining_xpts.txt","r");
fscanf(define,"notch start: %f\n",&xns);
fscanf(define,"notch end: %f\n",&xne);
fscanf(define,"delta y: %f\n",&dy);
fscanf(define,"sweep end: %f\n",&xse);
fscanf(define,"delta end: %f\n",&xde);
//fscanf(define,"y T.E. tip: %f\n",&ytetip);
xse = xse -0.15;
xse_orig = xns;                             // for the original tip, (this case assumed to be when g = 28), x
    coord at the start of the notch.
xne_orig = xne-0.1771;                      // for the original tip, (this case assumed to be when g = 28), x
    coord at the end of the notch.
fclose(define);

jj=0;
strcpy(filename,argv[1]);
output = fopen(filename,"w");               // filename to write coordinate data to.
input = fopen("stations.txt","r");
while((j = getc(input)) != EOF )
        {
        ungetc(j,input);
        fscanf(input,"%f\n",&x)==1;
        if(x<=xne)
                {
                        y=0;                                        // for points before the notch,
                            y = leading edge coordinate.
                        fprintf(output,"%f %f\n",x,y-1);
                }
        /*--- SWEEP SECTION ---*/
        if((x>xne) && (x<=xse))
                {
                        if(x<xne_orig) {y = dy;}                    // if its behind the maximum of
                            the parabola, keep it constant to notch height.
                        else
                        {y = -swte*sw*(x-xne_orig);}                // maxima at 0, therefore add
                            notch height to match preceeding curve.
                        if(xne>xne_orig)                            // if the notch end is after the
                              zero gradient x coord ...
```

```
                        {yadd = -swte*sw*(xne-xne_orig);                        // an additional delta y
                             must be added to keep the notch height the same.
                        y = -swte*sw*(x-xne_orig)-yadd;}
                        fprintf(output,"%f %f\n",x,y-1);
                        lastpoint = -swte*sw*(xse-xne);          // this point is needed to match the
                             curve point to the delta curve's first point.
                }
        /*--- END SECTION ---*/
        if((x>xse) && (x<=xde))
                {
                        zg_pt = xde-xse;                                        // zero gradient point is at the
                             tip.
                        del = swte*sw/(3.5*(xde-xse+zg_pt));
                        firstpoint = del*pow(((xde-xse)+zg_pt),3.5);
                        dy1 = lastpoint-firstpoint;                             // the diff in y to join the
                             delta part to the swept part
                        y = del*pow(((xde-x)+zg_pt),3.5)+dy1;
                        fprintf(output,"%f %f\n",x,y-1);
                }
                jj=jj+1;
        }
ytetip=y;
printf("ytetip: %f\n",ytetip);
fclose(input);
fclose(output);

define = fopen("defining_xpts.txt","a");
fprintf(define,"y T.E. tip: %f\n",ytetip-1);
fclose(define);


/*--- CALCULATE AREA ---*/
Area = 0;
i = 0;
strcpy(filename,argv[1]);
input = fopen(filename,"r");
while((j = getc(input)) != EOF )
        {
        ungetc(j,input);
        fscanf(input,"%f %f \n",&XX[i],&YY[i])==2;
        YY[i] = YY[i] - ytetip;
        if(i>=1)
                {Area = Area + sqrt(((XX[i]-XX[i-1])*(YY[i]+YY[i-1])/2) * ((XX[i]-XX[i-1])*(YY[i]+YY[i
                     -1])/2));}
        i=i+1;
        }
fclose(input);
printf("%d Area: %f \n",i,Area);
printf("%d AR: %f \n",i,AR);

return 0;
}
```

B.9.2 Program to Create Planform Co-ordinates for Leading Edge

```
/*
 * ---------------------------------- BERP-LIKE TIP PARAMETERISATION ----------------------------------
 * This program parameterises a BERP-like rotor tip L.E. using three parameters:
 * 1. Notch gradient, g - the gradient of a sigmoid curve
 * 2. Sweep gradient, sw - the gradient of a parabolic curve
 * 3. Delta gradient, del - the gradient of a polynomial of order 2.5.
 * To make these independent of each other, some other parameters are automatically obtained,
 * such as the the zero gradient points of the sweep parabola and the delta polynomial.
 * The notch height can be specified, but is kept constant regardless of the changes in the
 * three parameters.
 * A number of files are required:
 * 1. The parameter.txt - contains the 3 parameters.
 * 2. The defining_xpts.txt - defines the x coordinates where the parameters start and stop.
 * 3. The stations.txt - defines the x coordinates - (will be useful when making ICEM grid).
 * 4. An output file to write the data to - this is an argument to the run command.
 * ------------------------------------------------------------------------------------------------
 */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>

main(int argc, char *argv[])
{
FILE *input, *define, *param, *output;
char filename[128];

int i,j,jj;
float x,y,xns,xne,xse,xde,dy,dx,xne_orig,xse_orig,yadd,xtetip,ytetip,area_orig;
float lastpoint,firstpoint,miny,maxy;
float g,sw,del,zg_pt,dy1,Area,AR,AR_orig;
float XX[100],YY[100];

/*--- OPEN AND READ INPUT FILE ---*/
param = fopen("parameters.txt","r");
fscanf(param,"g: %f\n",&g);                 // gradient of the notch sigmoid
fscanf(param,"sw: %f\n",&sw);               // gradient of the sweep parabola
fscanf(param,"del: %f\n",&del);             // gradient of the delta sweep definition curve - only used as an
     initial guess - not a parameter
fclose(param);

define = fopen("defining_xpts.txt","r");
fscanf(define,"notch start: %f\n",&xns);
```

```
fscanf(define,"notch end: %f\n",&xne);
fscanf(define,"delta y: %f\n",&dy);
fscanf(define,"sweep end: %f\n",&xse);
fscanf(define,"delta end: %f\n",&xde);
fscanf(define,"y T.E. tip: %f\n",&ytetip);

dx=2*4.584/g;                           // obtained to start the sweep from when the gradient is 0.001 or
    something.
xns = xne-dx;                           // so xne is when the gradient of the sigmoid curve becomes
    0.001.
xse_orig = 11.1172;                     // for the original tip, (this case assumed to be when g = 40), x
    coord at the start of the notch.
xne_orig = 11.500;                      // for the original tip, (this case assumed to be when g = 40), x
    coord at the end of the notch.
area_orig = 16.238148;                  // original Area - calculated
AR_orig = 13.7143;                      // original Aspect Ratio - from paper
printf("dx: %f, xns %f\n",dx,xns);
fclose(define);

jj=0;
strcpy(filename,argv[1]);
output = fopen(filename,"w");           // filename to write coordinate data to.
input = fopen("stations.txt","r");
while((j = getc(input)) != EOF )
        {
        ungetc(j,input);
        fscanf(input,"%f\n",&x)==1;
        if(x<xns)
                {
                        y=0;                                            // for points before the notch,
                            y = leading edge coordinate.
                        fprintf(output,"%f %f\n",x,y);
                }
        /*--- NOTCH SECTION ---*/
        if((x>=xns) && (x<=xne))
                {
                        miny = 1/(1+exp(-g*(xns-xns-dx/2)));
                        maxy = 1/(1+exp(-g*(xne-xns-dx/2)));
                        y = 1/(1+exp(-g*(x-xns-dx/2)));                 // normalised sigmoid curve
                        y = (y-miny)/(maxy-miny)*dy;                    // denormalised to notch height,
                            dy
                        fprintf(output,"%f %f\n",x,y);
                }
        /*--- SWEEP SECTION ---*/
        if((x>xne) && (x<=xse))
                {
                        if(x<xne_orig) {y = dy;}                        // if its behind the maximum of
                            the parabola, keep it constant to notch height.
                        else
                        {y = -sw*(x-xne_orig)*(x-xne_orig)+dy;}         // maxima at 0, therefore add
                            notch height to match preceeding curve.
                        if(xne>xne_orig)                                // if the notch end is after the
                             zero gradient x coord ...
                        {yadd = -sw*(xne-xne_orig)*(xne-xne_orig);      // an additional delta y must be
                             added to keep the notch height the same.
                        y = -sw*(x-xne_orig)*(x-xne_orig)+dy-yadd;}
                        fprintf(output,"%f %f\n",x,y);
                        lastpoint = -sw*(xse-xne_orig)*(xse-xne_orig)+dy;// this point is needed to match
                             the curve point to the delta curve's first point.
                }
        /*--- DELTA SECTION ---*/
        if((x>xse) && (x<=xde))
                {
                        zg_pt = pow(((2*sw*(xse-xne_orig))/(2.5*del)),0.667); // find zero gradient point
                             of pow 2.5 curve that matches gradient of sweep parabola.
                        firstpoint = -del*pow(((xse-xse)+zg_pt),2.5);
                        dy1 = lastpoint-firstpoint;                     // the diff in y to join
                            the delta part to the swept part
                        del = -(ytetip-dy1)/pow(((xde-xse)+zg_pt),2.5);
                        zg_pt = pow(((2*sw*(xse-xne_orig))/(2.5*del)),0.667); // find zero gradient point
                             of pow 2.5 curve that matches gradient of sweep parabola.
                        firstpoint = -del*pow(((xse-xse)+zg_pt),2.5);
                        dy1 = lastpoint-firstpoint;                     // the diff in y to join
                            the delta part to the swept part
                        y = -del*pow(((x-xse)+zg_pt),2.5)+dy1;
                        if(xne>xne_orig){y = y - yadd;}
                        if(y >= ytetip)
                        {fprintf(output,"%f %f\n",x,y);}
                        xtetip = pow(((dy1 - ytetip)/del),0.4)-zg_pt+xse;
                }
                jj=jj+1;
        }
        fprintf(output,"%f %f\n",xtetip,ytetip);
fclose(input);
fclose(output);

/*--- CALCULATE AREA ---*/
Area = 0;
i = 0;
strcpy(filename,argv[1]);
input = fopen(filename,"r");
while((j = getc(input)) != EOF )
        {
        ungetc(j,input);
        fscanf(input,"%f %f \n",&XX[i],&YY[i])==2;
        YY[i] = YY[i] - ytetip;
        if(i>=1)
                {Area = Area + sqrt(((XX[i]-XX[i-1])*(YY[i]+YY[i-1])/2) * ((XX[i]-XX[i-1])*(YY[i]+YY[i
                    -1])/2));}
        i=i+1;
        }
fclose(input);
AR = AR_orig - (Area - area_orig);
```

```
printf("%d orig_area %f Area: %f \n",i,area_orig,Area);
printf("%d orig AR %f AR: %f \n",i,AR_orig,AR);

return 0;
}
```

B.9.3 Program to Obtain Scaling Factor for Chord of Sections at the BERP Tip

```
/*
 * ---------------------------- BERP-LIKE TIP PARAMETERISATION SCALING ----------------------------
 * This program uses the output from the planform programs, 'define.c' and 'define_lower.c' to
 * determine the chord and the proportional thickness of the sections so that the thickness is
 * linearly blended towards the tip.
 * 2 inputs and 1 output are required.
 * input 1 = the leading edge coordinate data e.g. data.txt
 * input 2 = the trailing edge coordinate data e.g. data_lower.txt
 * output = file to write the chord and thickness data e.g. data_thick.txt
 * It also prints out a file called 'aerofoil_thickness.dat' which section along the APACHE
 * blade has the right thickness so that when it is scaled up in chord for the BERP tip, it
 * still allows a linearly blending thickness towards the tip.
 * ------------------------------------------------------------------------------------------------
 */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>

main(int argc, char *argv[])
{
FILE *input1, *input2, *output;
char filename[128];

int i,j,jj,n;
float x,y,xse_orig,AR_orig,thk_orig,thk_tip,x_orig,reqx,req_th;
n = 500;
float X[n],YLE[n],YTE[n],thick[n],actual_thk[n],xx[n],thk[n];
double chord[n];

/*--- CONSTANTS ---*/
xse_orig = 11.3672;                     // BERP tip begins
thk_orig = 9.6;                         // HH02 aerofoil thickness (inboard)
thk_tip = 6;                            // NACA 64A-006 aerofoil thickness (tip)
AR_orig = 13.7143;                      // original Aspect Ratio - and Radius of rotor

/*--- FIND POSITION TO CUT TO GET REQUIRED THICKNESS SECTION ---*/
output = fopen("aerofoil_thickness.dat","w");
for(i=0;i<n;i++)
{
        x_orig=12.6172;                 // original APACHE's sweep start location (r/R).
        xx[i] = (AR_orig-x_orig) / n * i;
        thk[i] = (thk_tip-thk_orig)/(AR_orig-x_orig)*xx[i] + thk_orig;
        xx[i]=xx[i]+x_orig;
        fprintf(output,"%f\t%f\n",xx[i],thk[i]);
}
fclose(output);

/*--- OPEN AND READ INPUT FILE ---*/
jj=0;
strcpy(filename,argv[1]);
input1 = fopen(filename,"r");           // filename with leading edge planform coordinates.
while((j = getc(input1)) != EOF )
        {
        ungetc(j,input1);
        fscanf(input1,"%f %f\n",&X[jj],&YLE[jj])==2;
        jj=jj+1;
        }
fclose(input1);
jj=0;
strcpy(filename,argv[2]);
input2 = fopen(filename,"r");           // filename with trailing edge planform coordinates.
while((j = getc(input2)) != EOF )
        {
        ungetc(j,input2);
        fscanf(input2,"%f %f\n",&X[jj],&YTE[jj])==2;
        jj=jj+1;
        }
fclose(input2);

/*--- CALCULATE CHORD LENGTH ---*/
strcpy(filename,argv[3]);
output = fopen(filename,"w");
fprintf(output,"X\t\tYLE\t\tYTE\t\tchord\t\tthick\t\taero_t\t\tt*c\t\tx\t\tthk\n");
for(i=0;i<jj;i++)
        {
        chord[i]=YLE[i]-YTE[i];
        if(X[i]>=xse_orig)
        {thick[i]=(thk_tip-thk_orig)/(AR_orig-xse_orig)*(X[i]-xse_orig) + thk_orig;}    // original rotor
             thickness variation.
        else {thick[i]=thk_orig;}
        actual_thk[i]=thick[i]/chord[i];        // chord variation means we actually need this thickness
             value for the aerofoil.
        j=0;
        while (j<n)
        {
                if(sqrt((actual_thk[i]-thk[j])*(actual_thk[i]-thk[j]))>0.005)
                {j=j+1;}
```

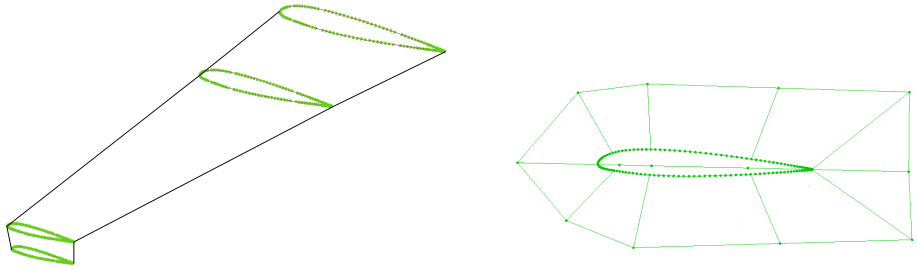
```
                else {reqx = xx[j]; req_th = thk[j]; break;}
            }
            fprintf(output,"%f\t%f\t%f\t%lf\t%f\t%f\t%f\t\%f\t%f\n",X[i],YLE[i],YTE[i],chord[i
                ]*0.9999126196031187,thick[i],actual_thk[i],actual_thk[i]*chord[i],reqx,req_th);
            }
fclose(output);


return 0;
}
```

B.10 ICEMCFD Grid Script for Three Segment Wing

The script below automatically creates a grid geometry for a wing of three segments at fixed locations along the span, i.e. 4 aerofoils can be specified at 4 locations - the root, midchord 1 (M1), midchord 2 (M2) and the tip, in that order. All the remaining points, curves and surfaces are related to these points, in terms of ratios. The only points that are specified independently are the farfield points and symmetry surface points.
First all the points on the aerofoils and the aerofoil curves are imported. Note that the curves are imported because the first line in the aerofoil point file contains the number of points making half the curve and the number of curves to make plus one. So for example if the aerofoil is made up of 256 points and you would like to generate a single curve joining all the points, the first line would read:

128 2

The ICEM environment is as in Figure B.1(a). Once the points and curves are available, the points



(a) (b)

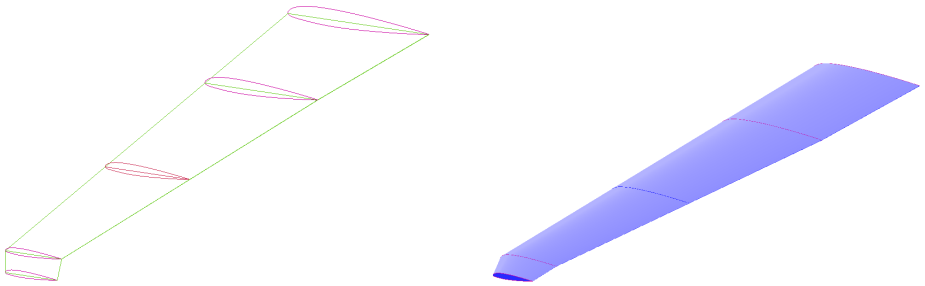
Figure B.1: *(a)Imported points of aerofoils. (b) Points around the aerofoil to which vertices will be assosciated to.*

to associate the blocking vertices around the wing are specified in the format shown in Figure B.1(b). These are given as ratios of the distance between a point on the chord line and a point on the aerofoil curve. This is done for each section at which vertices are going to be associated. Where there are no curves imported and where vertices are required, a curve is created as a parameteric cut in the surface and points are generated along this curve. Then the curves and surfaces around the wing are generated (Figure B.2(a)). An important point to remember when creating ICEM scripts is to match the curves at a point (also make sure tolerances are 10^{-6} or lower at all stages). Also the matching should be done to a single curve or have a hierarchy so that the curves do not simply shift from one 'side' to the other leaving gaps in the surface.
The full flow field is shown in Figure B.3(a). The other important points are the ones around the tip of the blade. There is a slight expansion in size of the blocking at the tip towards the far field. To capture this expansion, scaling of the assosciation points about the quarter chord point was carried out. This can lead to slight morphing of the general aerofoil shape as shown in Figure B.3(b). This can be manually tweaked to get a good geometry, or perhaps it may be better to scale this from the far field side.

B.10.1 Example of ICEMCFD script for wing

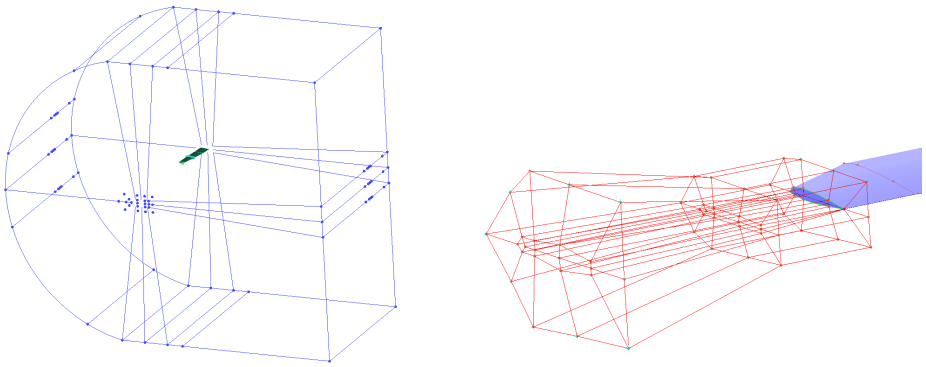
These replay files only create the geometry of the wing. Once they are run, a blocking file is required to associate the geometry with and then the mesh is generated.

```
ic_set_meshing_params global 0
ic_undo_group_begin
ic_set_meshing_params global 0 gttol 0.000001 gtrel 1    ### this sets the toerance value
ic_regenerate_tris
```



(a) (b)

Figure B.2: *(a) Curves around the wing. (b) Surface around the wing.*



(a) (b)

Figure B.3: *(a)Full flow field. (b) Blocking and points around the wing tip.*

```
ic_geo_cre_geom_input /home/cfd/cjohnson/NLF2/NACA23012aerofoil/naca23012_1222c.txt 0.000001 input PNTS
    pnt CRVS crv SURFS {}  ### this reads the input files
ic_boco_solver
ic_boco_clear_icons
ic_csystem_display all 0
ic_csystem_set_current global
ic_boco_nastran_csystem reset
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.00 {pnt0 pnt127}     ### create a curve between pt 0 and pt 127
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.00 {crv.00 0.5}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.01 {pnt64 pnt.00}
ic_geo_delete_family GEOM
ic_geo_cre_geom_input /home/cfd/cjohnson/NLF2/NACA23012aerofoil/N23012_M1p94_M2p52_aboutLE/
    naca23012_1222c_midchord1_p94_BACK0.txt 0.000001 input PNTS pnt CRVS crv SURFS {}
ic_boco_solver
ic_boco_clear_icons
ic_csystem_display all 0
ic_csystem_set_current global
ic_boco_nastran_csystem reset
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.02 {pnt255 pnt128}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.01 {crv.02 0.5}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.03 {pnt192 pnt.01}
ic_geo_delete_family GEOM
ic_geo_cre_geom_input /home/cfd/cjohnson/NLF2/NACA23012aerofoil/N23012_M1p94_M2p52_aboutLE/
    naca23012_1222c_midchord2_p52_BACK0.txt 0.000001 input PNTS pnt CRVS crv SURFS {}
ic_boco_solver
ic_boco_clear_icons
ic_csystem_display all 0
ic_csystem_set_current global
ic_boco_nastran_csystem reset
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.04 {pnt256 pnt383}
ic_geo_delete_family GEOM
```

```
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.02 {crv.04 0.5}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.05 {pnt320 pnt.02}
ic_geo_delete_family GEOM
ic_geo_cre_geom_input /home/cfd/cjohnson/NLF2/NACA23012aerofoil/naca23012_1222c_tipchord_p4511m_BACKp25c.
    txt 0.000001 input PNTS pnt CRVS crv SURFS {}
ic_boco_solver
ic_boco_clear_icons
ic_csystem_display all 0
ic_csystem_set_current global
ic_boco_nastran_csystem reset
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.06 {pnt511 pnt384}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.03 {crv.06 0.5}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.07 {pnt448 pnt.03}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.08 {pnt384 pnt256}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.09 {pnt.02 pnt.03}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.10 {pnt511 pnt383}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.11 {pnt320 pnt448}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.12 {pnt320 pnt192}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.13 {pnt192 pnt64}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.14 {pnt256 pnt128}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.15 {pnt.02 pnt.01}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.16 {pnt383 pnt255}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.17 {pnt128 pnt0}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.18 {pnt.01 pnt.00}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_curve point PNTS crv.19 {pnt255 pnt127}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.04 {crv.01 0.25}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.04a {crv.01 0.1}     ### create a point on curve crv.01 at 0.1 from its
    starting point
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.05 {crv.03 0.25}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.05a {crv.03 0.1}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.06 {crv.05 0.25}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.06a {crv.05 0.1}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.07 {crv.07 0.25}
```

```
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.97 {crv.07 0.15}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.98 {crv.07 0.1}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.08 {crv.07 0.7}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.09 {crv.05 0.7}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.10 {crv.03 0.7}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point crv_par PNTS pnt.11 {crv.01 0.7}
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.23a pnt.98+(pnt459-pnt.98)*10
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.15a pnt.98+(pnt437-pnt.98)*6     ### create a point between point 98 and 437 at a
    ratio of 0.6 from pnt 98
ic_geo_delete_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.12 pnt.04+(pnt43-pnt.04)*5
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.12 pnt.04+(pnt43-pnt.04)*5
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.16 pnt.11+(pnt23-pnt.11)*9
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.13a pnt.05+(pnt171-pnt.05)*5
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.14 pnt.06+(pnt299-pnt.06)*5
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.15 pnt.07+(pnt427-pnt.07)*5
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.18 pnt.09+(pnt279-pnt.09)*9
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.19 pnt.08+(pnt407-pnt.08)*9
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.28 pnt.04+(pnt64-pnt.04)*2.5
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.29 pnt.05+(pnt192-pnt.05)*2.5
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.30 pnt.06+(pnt320-pnt.06)*2.5
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.31 pnt.07+(pnt448-pnt.07)*2.5
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.32 pnt.11+(pnt.00-pnt.11)*2.5
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.33 pnt.10+(pnt.01-pnt.10)*2.5
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.34 pnt.09+(pnt.02-pnt.09)*2.5
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.35 pnt.08+(pnt.03-pnt.08)*2.5
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.36 pnt.12+(pnt.16-pnt.12)*2
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
```

```
ic_point {} PNTS pnt.17a pnt.10+(pnt151-pnt.10)*9
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.37 pnt.13a+(pnt.17a-pnt.13a)*2
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.38 pnt.14+(pnt.18-pnt.14)*2
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.39 pnt.15+(pnt.19-pnt.15)*2
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.20 pnt.04+(pnt85-pnt.04)*9
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.20a pnt.04a+(pnt75-pnt.04a)*10
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.12a pnt.04a+(pnt53-pnt.04a)*6
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.21a pnt.05a+(pnt203-pnt.05a)*10
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.13aa pnt.05a+(pnt181-pnt.05a)*6
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.22a pnt.06a+(pnt331-pnt.06a)*10
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.14a pnt.06a+(pnt309-pnt.06a)*6
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.24 pnt.11+(pnt105-pnt.11)*12
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.42 pnt.20+(pnt.24-pnt.20)*1.9
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.21 pnt.05+(pnt213-pnt.05)*9
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.25 pnt.10+(pnt233-pnt.10)*12
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.43 pnt.21+(pnt.25-pnt.21)*1.9
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.22 pnt.06+(pnt341-pnt.06)*9
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.26 pnt.09+(pnt361-pnt.09)*12
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.44 pnt.22+(pnt.26-pnt.22)*1.9
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.23 pnt.07+(pnt469-pnt.07)*9
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.27 pnt.08+(pnt489-pnt.08)*12
ic_geo_delete_family GEOM
ic_geo_new_family GEOM
ic_boco_set_part_color GEOM
ic_point {} PNTS pnt.45 pnt.23+(pnt.27-pnt.23)*1.9
ic_geo_delete_family GEOM
ic_point {} GEOM pnt.47 -17000,0,0            ### create a point at xyz coordinates
ic_point {} GEOM pnt.50 25000,0,0
ic_point {} GEOM pnt.51 25000,18000,0
ic_point {} GEOM pnt.52 25000,-18000,0
ic_point {} GEOM pnt.58 -2500,-18000,0
ic_point {} GEOM pnt.59 -2500,18000,0
ic_curve point GEOM crv.20 {pnt.50 pnt.51}
ic_curve point GEOM crv.21 {pnt.50 pnt.52}
ic_point {} GEOM pnt.60 25000,2000,0
ic_point {} GEOM pnt.61 25000,-2000,0
ic_curve point GEOM crv.22 {pnt.50 pnt.32}
ic_curve point GEOM crv.23 {pnt.60 pnt.36}
ic_curve point GEOM crv.24 {pnt.61 pnt.42}
ic_curve point PNTS crv.25 {pnt.42 pnt.32}
ic_curve point PNTS crv.26 {pnt.32 pnt.36}
ic_point {} GEOM pnt.64 3500,18000,0
ic_point {} GEOM pnt.64a 5500,18000,0
```

```
ic_point {} GEOM pnt.65 3500,-18000,0
ic_point {} GEOM pnt.65a 5500,-18000,0
ic_point {} GEOM pnt.66 pnt.59+(pnt.64-pnt.59)*0.5
ic_point {} GEOM pnt.67 pnt.58+(pnt.65-pnt.58)*0.5
ic_curve point GEOM crv.34 {pnt.64 pnt.51}
ic_curve point GEOM crv.35 {pnt.65 pnt.52}
ic_curve point GEOM crv.36 {pnt.64 pnt.36}
ic_curve point GEOM crv.37 {pnt.64 pnt.66}
ic_curve point GEOM crv.38 {pnt.66 pnt.16}
ic_curve point GEOM crv.39 {pnt.66 pnt.59}
ic_curve point GEOM crv.40 {pnt.59 pnt.12}
ic_curve point GEOM crv.41 {pnt.65 pnt.42}
ic_curve point GEOM crv.42 {pnt.65 pnt.67}
ic_curve point GEOM crv.43 {pnt.67 pnt.24}
ic_curve point GEOM crv.44 {pnt.67 pnt.58}
ic_curve point GEOM crv.45 {pnt.58 pnt.20}
ic_curve point PNTS crv.46 {pnt.36 pnt.16}
ic_curve point PNTS crv.47 {pnt.16 pnt.12}
ic_curve point PNTS crv.48 {pnt.42 pnt.24}
ic_curve point PNTS crv.49 {pnt.24 pnt.20}
ic_curve point PNTS crv.50 {pnt.32 pnt.00}
ic_curve point GEOM crv.51 {pnt.47 pnt.28}
ic_curve point PNTS crv.52 {pnt.28 pnt64}
ic_curve arc GEOM crv.53 {pnt.59 pnt.47 pnt.58}
ic_geo_cre_crv_concat GEOM crv.54 0.0000001 {crv.35 crv.42 crv.44}   join curves together
ic_delete_geometry curve names {crv.35 crv.42 crv.44}
ic_geo_cre_crv_concat GEOM crv.55 0.0000001 {crv.34 crv.37 crv.39}
ic_delete_geometry curve names {crv.34 crv.37 crv.39}
ic_geo_cre_crv_concat GEOM crv.56 0.0000001 {crv.20 crv.21}
ic_delete_geometry curve names {crv.20 crv.21}
ic_surface 2-4crvs GEOM srf.00 {0.0001 {crv.54 crv.56 crv.55 crv.53}}  ### create new surface
ic_set_dormant_pickable point 0 {}
ic_set_dormant_pickable curve 0 {}
ic_surface 2-4crvs PNTS srf.01 {0.0001 {crv.17 crv0 crv.19 crv1}}
ic_set_dormant_pickable point 0 {}
ic_set_dormant_pickable curve 0 {}
ic_geo_mod_crv_match_crv crv.17 crv.00 2 1 {0 1 1 1 0}
ic_delete_geometry curve names crv.17 0
ic_geo_set_name curve crv.17.1 crv.17
ic_geo_mod_crv_match_crv crv.17 crv0 2 1 {0 1 1 1 0}
ic_geo_mod_crv_match_crv crv.19 crv0 2 2 {0 1 1 1 0}
ic_delete_geometry curve names crv.19 0
ic_geo_set_name curve crv.19.1 crv.19
ic_geo_mod_crv_match_crv crv.00 crv.19 2 2 {0 1 1 1 0}
ic_delete_geometry curve names crv.00 0
ic_geo_set_name curve crv.00.1 crv.00
ic_geo_mod_crv_match_crv crv.00 crv.17 1 2 {0 1 1 1 0}
ic_geo_mod_crv_match_crv crv.16 crv1 2 2 {0 1 1 1 0}
ic_delete_geometry curve names crv.16 0
ic_geo_set_name curve crv.16.1 crv.16
ic_geo_mod_crv_match_crv crv.14 crv1 2 1 {0 1 1 1 0}
ic_delete_geometry curve names crv.14 0
ic_geo_set_name curve crv.14.1 crv.14
ic_geo_mod_crv_match_crv crv.02 crv.14 2 2 {0 1 1 1 0}
ic_delete_geometry curve names crv.02 0
ic_geo_set_name curve crv.02.1 crv.02
ic_geo_mod_crv_match_crv crv.02 crv.16 1 2 {0 1 1 1 0}
ic_geo_mod_crv_match_crv crv.08 crv2 2 1 {0 1 1 1 0}
ic_delete_geometry curve names crv.08 0
ic_geo_set_name curve crv.08.1 crv.08
ic_geo_mod_crv_match_crv crv.10 crv2 2 2 {0 1 1 1 0}
ic_delete_geometry curve names crv.10 0
ic_geo_set_name curve crv.10.1 crv.10
ic_geo_mod_crv_match_crv crv.04 crv.10 2 2 {0 1 1 1 0}
ic_delete_geometry curve names crv.04 0
ic_geo_set_name curve crv.04.1 crv.04
ic_geo_mod_crv_match_crv crv.04 crv.08 1 2 {0 1 1 1 0}
ic_geo_mod_crv_match_crv crv.08 crv3 1 1 {0 1 1 1 0}
ic_geo_mod_crv_match_crv crv.10 crv3 1 2 {0 1 1 1 0}
ic_geo_mod_crv_match_crv crv.06 crv.10 1 1 {0 1 1 1 0}
ic_delete_geometry curve names crv.06 0
ic_geo_set_name curve crv.06.1 crv.06
ic_geo_mod_crv_match_crv crv.06 crv.08 2 1 {0 1 1 1 0}
ic_surface 2-4crvs PNTS srf.01 {0.0001 {crv.08 crv2 crv.10 crv3}}
ic_set_dormant_pickable point 0 {}
ic_set_dormant_pickable curve 0 {}
ic_surface 2-4crvs PNTS srf.02 {0.0001 {crv.08 crv.04 crv.10 crv.06}}
ic_set_dormant_pickable point 0 {}
ic_set_dormant_pickable curve 0 {}
ic_geo_mod_crv_match_crv crv.17 crv1 1 1 {0 1 1 1 0}
ic_geo_mod_crv_match_crv crv.19 crv1 1 2 {0 1 1 1 0}
ic_geo_mod_crv_match_crv crv.17 crv0 2 1 {0 1 1 1 0}
ic_geo_mod_crv_match_crv crv.19 crv0 2 2 {0 1 1 1 0}
ic_surface 2-4crvs PNTS srf.03 {0.0001 {crv.17 crv0 crv.19 crv1}}
ic_set_dormant_pickable point 0 {}
ic_set_dormant_pickable curve 0 {}
ic_surface 2-4crvs PNTS srf.04 {0.0001 {crv.02 crv.17 crv.00 crv.19}}
ic_set_dormant_pickable point 0 {}
ic_set_dormant_pickable curve 0 {}
ic_geo_mod_crv_match_crv crv.14 crv1 2 1 {0 1 1 1 0}
ic_geo_mod_crv_match_crv crv.16 crv1 2 2 {0 1 1 1 0}
ic_geo_mod_crv_match_crv crv.14 crv2 1 1 {0 1 1 1 0}
ic_geo_mod_crv_match_crv crv.16 crv2 1 2 {0 1 1 1 0}
ic_geo_mod_crv_match_crv crv.04 crv.14 1 1 {0 1 1 1 0}
ic_delete_geometry curve names crv.04 0
ic_geo_set_name curve crv.04.1 crv.04
ic_geo_mod_crv_match_crv crv.04 crv.16 2 1 {0 1 1 1 0}
ic_surface 2-4crvs PNTS srf.05 {0.0001 {crv.14 crv1 crv.16 crv2}}
ic_set_dormant_pickable point 0 {}
ic_set_dormant_pickable curve 0 {}
ic_surface 2-4crvs PNTS srf.06 {0.0001 {crv.04 crv.14 crv.02 crv.16}}
ic_set_dormant_pickable point 0 {}
ic_set_dormant_pickable curve 0 {}
```

```
ic_geo_set_part surface {srf.02 srf.01 srf.06 srf.05 srf.04 srf.03} WING 0
ic_geo_duplicate_set_fam_and_data surface srf.00 srf.00.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.54 crv.54.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.41 crv.41.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.45 crv.45.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.43 crv.43.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.53 crv.53.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.51 crv.51.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.40 crv.40.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.55 crv.55.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.38 crv.38.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.36 crv.36.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.56 crv.56.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.23 crv.23.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.22 crv.22.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.24 crv.24.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.48 crv.48.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.50 crv.50.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.25 crv.25.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.26 crv.26.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.46 crv.46.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.47 crv.47.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.49 crv.49.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.52 crv.52.0 {} _0
ic_geo_duplicate_set_fam_and_data curve crv.01 crv.01.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.65 pnt.65.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.65a pnt.65a.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.67 pnt.67.0 {} _0
ic_geo_duplicate_set_fam_and_data point GEOM.67 GEOM.67.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.47 pnt.47.0 {} _0
ic_geo_duplicate_set_fam_and_data point GEOM.68 GEOM.68.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.66 pnt.66.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.59 pnt.59.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.64 pnt.64.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.64a pnt.64a.0 {} _0
ic_geo_duplicate_set_fam_and_data point GEOM.81 GEOM.81.0 {} _0
ic_geo_duplicate_set_fam_and_data point GEOM.69 GEOM.69.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.51 pnt.51.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.61 pnt.61.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.50 pnt.50.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.60 pnt.60.0 {} _0
ic_geo_duplicate_set_fam_and_data point GEOM.82 GEOM.82.0 {} _0
ic_geo_duplicate_set_fam_and_data point GEOM.70 GEOM.70.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.52 pnt.52.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.58 pnt.58.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.32 pnt.32.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.36 pnt.36.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.16 pnt.16.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.12 pnt.12.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.42 pnt.42.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.24 pnt.24.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.20 pnt.20.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.28 pnt.28.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.11 pnt.11.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.04 pnt.04.0 {} _0
ic_save_model_for_undo
ic_move_geometry surface names srf.00.0 translate {0 0 20000}
ic_move_geometry curve names {crv.54.0 crv.41.0 crv.45.0 crv.43.0 crv.53.0 crv.51.0 crv.40.0 crv.55.0 crv
    .38.0 crv.36.0 crv.56.0 crv.23.0 crv.22.0 crv.24.0 crv.48.0 crv.50.0 crv.25.0 crv.26.0 crv.46.0 crv
    .47.0 crv.49.0 crv.52.0 crv.01.0} translate {0 0 20000}
ic_move_geometry point names {pnt.65.0 pnt.65a.0 pnt.67.0 GEOM.67.0 pnt.47.0 GEOM.68.0 pnt.66.0 pnt.59.0
    pnt.64.0 pnt.64a.0 GEOM.81.0 GEOM.69.0 pnt.51.0 pnt.61.0 pnt.50.0 pnt.60.0 GEOM.82.0 GEOM.70.0 pnt
    .52.0 pnt.58.0 pnt.32.0 pnt.36.0 pnt.16.0 pnt.12.0 pnt.42.0 pnt.24.0 pnt.20.0 pnt.28.0 pnt.11.0 pnt
    .04.0} translate {0 0 20000}
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_curve point GEOM crv.57 {GEOM.82 pnt.52.0}
ic_curve point GEOM crv.58 {pnt.61.0 pnt.61}
ic_curve point GEOM crv.59 {pnt.50 pnt.50.0}
ic_curve point GEOM crv.60 {pnt.60.0 pnt.60}
ic_curve point GEOM crv.61 {GEOM.81 pnt.51.0}
ic_curve point GEOM crv.62 {pnt.64.0 pnt.64}
ic_curve point GEOM crv.62a {pnt.64a.0 pnt.64a}
ic_curve point GEOM crv.63 {pnt.66 pnt.66.0}
ic_curve point GEOM crv.64 {pnt.59.0 GEOM.68}
ic_curve point GEOM crv.65 {pnt.47 pnt.47.0}
ic_curve point GEOM crv.66 {pnt.65.0 pnt.65}
ic_curve point GEOM crv.66a {pnt.65a.0 pnt.65a}
ic_curve point GEOM crv.67 {pnt.67 pnt.67.0}
ic_curve point GEOM crv.68 {GEOM.67.0 GEOM.67}
ic_surface 2-4crvs GEOM srf.07 {0.0001 {crv.56.0 crv.61 crv.56 crv.57}}
ic_set_dormant_pickable point 0 {}
ic_set_dormant_pickable curve 0 {}
ic_surface 2-4crvs GEOM srf.08 {0.0001 {crv.54 crv.57 crv.54.0 crv.68}}
ic_set_dormant_pickable point 0 {}
ic_set_dormant_pickable curve 0 {}
ic_surface 2-4crvs GEOM srf.09 {0.0001 {crv.55.0 crv.61 crv.55 crv.64}}
ic_set_dormant_pickable point 0 {}
ic_set_dormant_pickable curve 0 {}
ic_surface 2-4crvs GEOM srf.10 {0.0001 {crv.64 crv.53 crv.68 crv.53.0}}
ic_set_dormant_pickable point 0 {}
ic_set_dormant_pickable curve 0 {}
ic_geo_set_part surface srf.00 SYMM 0
ic_geo_set_part surface {srf.09 srf.07 srf.08 srf.10} FF 0
ic_geo_set_part surface srf.00.0 FF 0
ic_delete_geometry point names {pnt.16.0 pnt.12.0 pnt.36.0 pnt.32.0 pnt.42.0 pnt.24.0 pnt.20.0 pnt.11.0
    pnt.04.0 pnt.28.0} 0 1
ic_set_dormant_pickable point 0 {}
ic_delete_geometry curve names {crv.49.0 crv.48.0 crv.25.0 crv.26.0 crv.46.0 crv.47.0 crv.01.0 crv.50.0
    crv.52.0} 0 1
ic_set_dormant_pickable curve 0 {}
ic_point {} GEOM pnt.69 2000,0,20000
ic_point {} GEOM pnt.69a 2000,400,20000
```

```
ic_point {} GEOM pnt.69aa 2000,-400,20000
ic_point {} GEOM pnt.70 2500,0,20000
ic_point {} GEOM pnt.71 2500,1000,20000
ic_point {} GEOM pnt.72 2500,-1000,20000
ic_point {} GEOM pnt.73 1500,-1000,20000
ic_point {} GEOM pnt.74 1500,1000,20000
ic_point {} GEOM pnt.75 1500,0,20000
ic_point {} GEOM pnt.75a 1500,400,20000
ic_point {} GEOM pnt.75aa 1500,-400,20000
ic_point {} GEOM pnt.76 500,0,20000
ic_point {} GEOM pnt.76a 500,400,20000
ic_point {} GEOM pnt.76aa 500,-400,20000
ic_point {} GEOM pnt.77 500,1000,20000
ic_point {} GEOM pnt.78 500,-1000,20000
ic_point {} GEOM pnt.79 -1000,0,20000
ic_point {} GEOM pnt.80 -300,0,20000
ic_point {} GEOM pnt.80a -600,400,20000
ic_point {} GEOM pnt.80aa -600,-400,20000
ic_point {} GEOM pnt.80a1 -2000,0,20000
ic_point {} GEOM pnt.80a2 -1200,1000,20000
ic_point {} GEOM pnt.80a3 -1200,-1000,20000
ic_point crv_par GEOM pnt.83 {crv.53 0.6}
ic_point crv_par GEOM pnt.83a {crv.53 0.9}
ic_point crv_par GEOM pnt.84 {crv.53 0.4}
ic_point crv_par GEOM pnt.84a {crv.53 0.1}
ic_point crv_par GEOM pnt.85 {crv.53.0 0.4}
ic_point crv_par GEOM pnt.85a {crv.53.0 0.1}
ic_point crv_par GEOM pnt.86 {crv.53.0 0.6}
ic_point crv_par GEOM pnt.86a {crv.53.0 0.9}
ic_point projcurv GEOM pnt.87 {pnt.28 crv.65}
ic_point projcurv GEOM pnt.88 {pnt.29 crv.65}
ic_point projcurv GEOM pnt.89 {pnt.30 crv.65}
ic_point projcurv GEOM pnt.90 {pnt.31 crv.65}
ic_curve point GEOM crv.69 {pnt.86 pnt.83}
ic_curve point GEOM crv.69a {pnt.86a pnt.83a}
ic_curve point GEOM crv.70 {pnt.85 pnt.84}
ic_curve point GEOM crv.70a {pnt.85a pnt.84a}
ic_point projcurv GEOM pnt.91 {pnt.90 crv.70}
ic_point projcurv GEOM pnt.92 {pnt.89 crv.70}
ic_point projcurv GEOM pnt.93 {pnt.88 crv.70}
ic_point projcurv GEOM pnt.94 {pnt.90 crv.69}
ic_point projcurv GEOM pnt.95 {pnt.89 crv.69}
ic_point projcurv GEOM pnt.96 {pnt.88 crv.69}
ic_geo_duplicate_set_fam_and_data point pnt.31 pnt.31.0 {} _0   ### make a copy of a point - used to
    translate a copied point
ic_geo_duplicate_set_fam_and_data point pnt.15 pnt.15.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.19 pnt.19.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.39 pnt.39.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.35 pnt.35.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.45 pnt.45.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.27 pnt.27.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.23 pnt.23.0 {} _0
ic_save_model_for_undo
ic_move_geometry point names {pnt.31.0 pnt.15.0 pnt.19.0 pnt.39.0 pnt.35.0 pnt.45.0 pnt.27.0 pnt.23.0}
    translate {0 0 500}  ### translate copied point
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_geo_duplicate_set_fam_and_data point pnt.98 pnt.101 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.07 pnt.102 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.08 pnt.105 {} _0
ic_geo_duplicate_set_fam_and_data point pnt407 pnt407.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt489 pnt489.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt427 pnt427.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt469 pnt469.0 {} _0
ic_move_geometry point names {pnt.101 pnt.102 pnt.105 pnt407.0 pnt489.0 pnt427.0 pnt469.0} translate {0 0
    500}
ic_point {} GEOM pnt.100 pnt.102+(pnt.101-pnt.102)*1.8
ic_point {} GEOM pnt.107 pnt.101+(pnt.105-pnt.101)*1.7
ic_point {} GEOM pnt427a pnt.102+(pnt427.0-pnt.102)*0.8
ic_geo_duplicate_set_fam_and_data point pnt.107 pnt.107up {} _0
ic_geo_duplicate_set_fam_and_data point pnt.107 pnt.107dw {} _0
ic_move_geometry point names pnt.107up translate {0 20 0}
ic_move_geometry point names pnt.107dw translate {0 -20 0}
ic_geo_duplicate_set_fam_and_data point pnt.15a pnt.15a.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.23a pnt.23a.0 {} _0
ic_move_geometry point names {pnt.15a.0 pnt.23a.0} translate {0 0 500}
ic_geo_reset_data_structures
ic_point {} PNTS pnt.118 pnt.105+(pnt489.0-pnt.105)*1.5
ic_point {} PNTS pnt.119 pnt.105+(pnt407.0-pnt.105)*1.5
ic_point {} GEOM pnt437a pnt.119+(pnt427a-pnt.119)*1.4
ic_point {} GEOM pnt459a pnt.118+(pnt469.0-pnt.118)*1.4
ic_point {} PNTS pnt.120 pnt.45.0+(pnt.39.0-pnt.45.0)*0.45
ic_delete_geometry point names {pnt407.0 pnt489.0} 0 1
ic_set_dormant_pickable point 0 {}
ic_geo_set_part point {pnt.04a pnt.07 pnt.12a pnt.16 pnt.20a pnt23 pnt43 pnt53 pnt64 pnt75 pnt85 pnt.98
    pnt.100 pnt.101 pnt.102 pnt.105} IMP_PTS 0
ic_geo_set_part point {pnt105 pnt151 pnt171 pnt181 pnt203 pnt213 pnt233 pnt279 pnt299 pnt309 pnt320
    pnt331 pnt341} IMP_PTS 0
ic_geo_set_part point {pnt.105.0 pnt.107up pnt.107dw pnt.107 pnt.118 pnt.119 pnt192 pnt361 pnt407 pnt489
    pnt427 pnt427a pnt437 pnt437a pnt448 pnt459 pnt459a pnt469 pnt469.0} IMP_PTS 0
ic_geo_set_part point {pnt.15 pnt.15a pnt.19 pnt.23 pnt.23a pnt.27 pnt.31 pnt.35 pnt.39 pnt.45 } IMP_PTS
    0
ic_geo_set_part point {pnt.15.0 pnt.15a.0 pnt.19.0 pnt.23.0 pnt.23a.0 pnt.27.0 pnt.31.0 pnt.35.0 pnt.39.0
    pnt.45.0 } IMP_PTS 0
ic_delete_geometry point names pnt.120 0 1
ic_set_dormant_pickable point 0 {}
ic_geo_duplicate_set_fam_and_data point pnt.15a.0 pnt.15a.0.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.23a.0 pnt.23a.0.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt459a pnt459a.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt437a pnt437a.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt469.0 pnt469.0.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt427a pnt427a.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.119 pnt.119.0 {} _0
```

```
ic_geo_duplicate_set_fam_and_data point pnt.118 pnt.118.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.107 pnt.107.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.107dw pnt.107dw.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.107up pnt.107up.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.105 pnt.105.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.102 pnt.102.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.101 pnt.101.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.100 pnt.100.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.27.0 pnt.27.0.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.45.0 pnt.45.0.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.35.0 pnt.35.0.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.39.0 pnt.39.0.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.19.0 pnt.19.0.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.15.0 pnt.15.0.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.31.0 pnt.31.0.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.23.0 pnt.23.0.0 {} _0
ic_save_model_for_undo
ic_move_geometry point names {pnt.15a.0.0 pnt.23a.0.0 pnt459a.0 pnt437a.0 pnt469.0.0 pnt427a.0 pnt.119.0
     pnt.118.0 pnt.107.0 pnt.107dw.0 pnt.107up.0 pnt.105.0 pnt.102.0 pnt.101.0 pnt.100.0 pnt.27.0.0 pnt
     .45.0.0 pnt.35.0.0 pnt.39.0.0 pnt.19.0.0 pnt.15.0.0 pnt.31.0.0 pnt.23.0.0} translate {0 0 1000}
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_move_geometry point names {pnt.15a.0.0 pnt.23a.0.0 pnt459a.0 pnt437a.0 pnt469.0.0 pnt427a.0 pnt.107.0
     pnt.107dw.0 pnt.107up.0 pnt.101.0 pnt.100.0 pnt.27.0.0 pnt.45.0.0 pnt.35.0.0 pnt.39.0.0 pnt.19.0.0
     pnt.15.0.0 pnt.31.0.0 pnt.23.0.0} scale {1.2 1.6 1} cent {pnt.102.0}
ic_move_geometry point names {pnt.119.0 pnt.118.0 } scale {1.05 0.50 0} cent {pnt.105.0}
ic_save_model_for_undo
ic_move_geometry point names {pnt.15.0.0 pnt.19.0.0 pnt.39.0.0} translate {0 100 0}
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_save_model_for_undo
ic_move_geometry point names {pnt.45.0.0 pnt.27.0.0 pnt.23.0.0} translate {0 -100 0}
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_save_model_for_undo
ic_move_geometry point names pnt.31.0.0 translate {70 0 0}
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_save_model_for_undo
ic_move_geometry point names pnt.31.0.0 translate {-70 0 0}
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_save_model_for_undo
ic_move_geometry point names pnt.31.0.0 translate {-70 0 0}
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_save_model_for_undo
ic_move_geometry point names {pnt.39.0.0 pnt.35.0.0 pnt.45.0.0} translate {100 0 0}
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_save_model_for_undo
ic_move_geometry point names {pnt.119.0} translate {0 50 0}
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_save_model_for_undo
ic_move_geometry point names {pnt.118.0} translate {0 -50 0}
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_geo_duplicate_set_fam_and_data point pnt.91 pnt.91.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.90 pnt.90.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.94 pnt.94.0 {} _0
ic_save_model_for_undo
ic_move_geometry point names {pnt.91.0 pnt.90.0 pnt.94.0} translate {0 0 500}
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_geo_duplicate_set_fam_and_data point pnt.91 pnt.91.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.90 pnt.90.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.94 pnt.94.0 {} _0
ic_save_model_for_undo
ic_move_geometry point names {pnt.91.1 pnt.90.1 pnt.94.1} translate {0 0 1000}
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_geo_duplicate_set_fam_and_data point pnt.91 pnt.91.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.90 pnt.90.0 {} _0
ic_geo_duplicate_set_fam_and_data point pnt.94 pnt.94.0 {} _0
ic_save_model_for_undo
ic_move_geometry point names {pnt.91.2 pnt.90.2 pnt.94.2} translate {0 0 1500}
ic_geo_reset_data_structures
ic_geo_configure_one_attribute surface shade wire
ic_delete_geometry point names {pnt.91.1 pnt.90.1 pnt.94.1} 0 1
ic_set_dormant_pickable point 0 {}
ic_point projcurv GEOM pnt.126 {pnt.32 crv.59}    ### project a point to a curve
ic_point projcurv GEOM pnt.127 {pnt.33 crv.59}
ic_point projcurv GEOM pnt.128 {pnt.34 crv.59}
ic_point projcurv GEOM pnt.129 {pnt.35 crv.59}
ic_point projcurv GEOM pnt.130 {pnt.35.0 crv.59}
ic_point projcurv GEOM pnt.131 {pnt.35.0.0 crv.59}
ic_point projcurv GEOM pnt.132 {pnt.131 crv.60}
ic_point projcurv GEOM pnt.133 {pnt.130 crv.60}
ic_point projcurv GEOM pnt.134 {pnt.129 crv.60}
ic_point projcurv GEOM pnt.135 {pnt.128 crv.60}
ic_point projcurv GEOM pnt.136 {pnt.127 crv.60}
ic_point projcurv GEOM pnt.137 {pnt.131 crv.58}
ic_point projcurv GEOM pnt.138 {pnt.130 crv.58}
ic_point projcurv GEOM pnt.139 {pnt.129 crv.58}
ic_point projcurv GEOM pnt.140 {pnt.128 crv.58}
ic_point projcurv GEOM pnt.141 {pnt.127 crv.58}
ic_surface 2-4crvs CRVS srf.11 {0.0001 {crv3 crv.06}}
ic_set_dormant_pickable point 0 {}
ic_set_dormant_pickable curve 0 {}
ic_geo_set_part surface srf.11 WING 0
ic_geo_cre_crv_iso_crv WING crv.71 srf.05 0.5 1 0 0
ic_geo_configure_one_attribute surface shade wire
```

```
ic_point crv_par WING pnt.1142 {crv.71 0.0196078431373}   ### create a point at a ratio of a curve.
ic_point crv_par WING pnt.1143 {crv.71 0.0392156862746}
ic_point crv_par WING pnt.1144 {crv.71 0.0588235294119}
ic_point crv_par WING pnt.1145 {crv.71 0.0784313725492}
ic_point crv_par WING pnt.1146 {crv.71 0.0980392156865}
ic_point crv_par WING pnt.1147 {crv.71 0.117647058824}
ic_point crv_par WING pnt.1148 {crv.71 0.137254901961}
ic_point crv_par WING pnt.1149 {crv.71 0.156862745098}
ic_point crv_par WING pnt.1150 {crv.71 0.176470588236}
ic_point crv_par WING pnt.1151 {crv.71 0.196078431373}
ic_point crv_par WING pnt.1152 {crv.71 0.21568627451}
ic_point crv_par WING pnt.1153 {crv.71 0.235294117648}
ic_point crv_par WING pnt.1154 {crv.71 0.254901960785}
ic_point crv_par WING pnt.1155 {crv.71 0.274509803922}
ic_point crv_par WING pnt.1156 {crv.71 0.294117647059}
ic_point crv_par WING pnt.1157 {crv.71 0.313725490197}
ic_point crv_par WING pnt.1158 {crv.71 0.333333333334}
ic_point crv_par WING pnt.1159 {crv.71 0.352941176471}
ic_point crv_par IMP_PTS pnt.1160 {crv.71 0.372549019609}
ic_point crv_par WING pnt.1161 {crv.71 0.392156862746}
ic_point crv_par WING pnt.1162 {crv.71 0.411764705883}
ic_point crv_par WING pnt.1163 {crv.71 0.431372549021}
ic_point crv_par WING pnt.1164 {crv.71 0.450980392158}
ic_point crv_par WING pnt.1165 {crv.71 0.470588235295}
ic_point crv_par WING pnt.1166 {crv.71 0.490196078433}
ic_point crv_par WING pnt.1167 {crv.71 0.50980392157}
ic_point crv_par WING pnt.1168 {crv.71 0.529411764707}
ic_point crv_par WING pnt.1169 {crv.71 0.549019607844}
ic_point crv_par WING pnt.1170 {crv.71 0.568627450982}
ic_point crv_par WING pnt.1171 {crv.71 0.588235294119}
ic_point crv_par WING pnt.1172 {crv.71 0.607843137256}
ic_point crv_par IMP_PTS pnt.1173 {crv.71 0.627450980394}
ic_point crv_par WING pnt.1174 {crv.71 0.647058823531}
ic_point crv_par WING pnt.1175 {crv.71 0.666666666668}
ic_point crv_par WING pnt.1176 {crv.71 0.686274509806}
ic_point crv_par WING pnt.1177 {crv.71 0.705882352943}
ic_point crv_par WING pnt.1178 {crv.71 0.72549019608}
ic_point crv_par WING pnt.1179 {crv.71 0.745098039217}
ic_point crv_par WING pnt.1180 {crv.71 0.764705882355}
ic_point crv_par WING pnt.1181 {crv.71 0.784313725492}
ic_point crv_par WING pnt.1182 {crv.71 0.803921568629}
ic_point crv_par WING pnt.1183 {crv.71 0.823529411767}
ic_point crv_par WING pnt.1184 {crv.71 0.843137254904}
ic_point crv_par WING pnt.1185 {crv.71 0.862745098041}
ic_point crv_par WING pnt.1186 {crv.71 0.882352941179}
ic_point crv_par WING pnt.1187 {crv.71 0.901960784316}
ic_point crv_par WING pnt.1188 {crv.71 0.921568627453}
ic_point crv_par WING pnt.1189 {crv.71 0.94117647059}
ic_point crv_par WING pnt.1190 {crv.71 0.960784313728}
ic_point crv_par WING pnt.1191 {crv.71 0.980392156865}
ic_point intersect IMP_PTS pnt.17aa {crv.71 crv.12} tol 0.0000001
ic_point crv_par WING pnt.1192 {crv.71 0}
ic_point crv_par WING pnt.1193 {crv.71 1}
ic_curve point WING crv.72 {pnt.1193 pnt.1192}
ic_point crv_par WING pnt.1194 {crv.72 0.5}
ic_curve point WING crv.20aa {pnt.17aa pnt.1194}
ic_point crv_par WING pnt.17q {crv.20aa 0.25}
ic_point crv_par WING pnt.17q3 {crv.20aa 0.7}
ic_point crv_par WING pnt.17p1 {crv.20aa 0.1}
ic_point {} IMP_PTS pnt.sup1 pnt.17q+(pnt.1160-pnt.17q)*5
ic_point {} IMP_PTS pnt.slw1 pnt.17q+(pnt.1173-pnt.17q)*9
ic_point crv_par IMP_PTS pnt.su2 {crv.71 0.14}
ic_point crv_par IMP_PTS pnt.sl2 {crv.71 0.86}
ic_point {} IMP_PTS pnt.slw2 pnt.17q3+(pnt.sl2-pnt.17q3)*12
ic_point {} IMP_PTS pnt.sup2 pnt.17q3+(pnt.su2-pnt.17q3)*9
ic_point {} IMP_PTS pnt.stup pnt.sup1+(pnt.sup2-pnt.sup1)*1.9
ic_point {} IMP_PTS pnt.stlw pnt.slw1+(pnt.slw2-pnt.slw1)*1.8
ic_point crv_par IMP_PTS pnt.su1 {crv.71 0.46}
ic_point {} IMP_PTS pnt.slup pnt.17p1+(pnt.su1-pnt.17p1)*6
ic_point crv_par IMP_PTS pnt.sl1 {crv.71 0.545}
ic_point {} IMP_PTS pnt.sllw pnt.17p1+(pnt.sl1-pnt.17p1)*10
ic_point {} IMP_PTS pnt.sle pnt.17q+(pnt.17aa-pnt.17q)*2.5
ic_point {} IMP_PTS pnt.ste pnt.stup+(pnt.stlw-pnt.stup)*0.5
ic_point projcurv IMP_PTS pnt.17ff {pnt.sle crv.65}
ic_point projcurv IMP_PTS pnt.17ffu {pnt.17ff crv.70}
ic_point projcurv IMP_PTS pnt.17ffl {pnt.17ff crv.69}
ic_point projcurv IMP_PTS pnt.17ffb {pnt.ste crv.59}
ic_point projcurv IMP_PTS pnt.17ffbu {pnt.17ffb crv.60}
ic_point projcurv IMP_PTS pnt.17ffbl {pnt.17ffb crv.58}
ic_point projcurv IMP_PTS pnt.b1.1 {pnt.05 crv.68}
ic_point projcurv IMP_PTS pnt.b1.2 {pnt.17q crv.68}
ic_point projcurv IMP_PTS pnt.b1.3 {pnt.06 crv.68}
ic_point projcurv IMP_PTS pnt.b1.4 {pnt.07 crv.68}
ic_point projcurv IMP_PTS pnt.b1.5 {pnt.102 crv.68}
ic_point projcurv IMP_PTS pnt.b1.6 {pnt.102.0 crv.68}
ic_point projcurv IMP_PTS pnt.b2.1 {pnt.05 crv.67}
ic_point projcurv IMP_PTS pnt.b2.2 {pnt.17q crv.67}
ic_point projcurv IMP_PTS pnt.b2.3 {pnt.06 crv.67}
ic_point projcurv IMP_PTS pnt.b2.4 {pnt.07 crv.67}
ic_point projcurv IMP_PTS pnt.b2.5 {pnt.102 crv.67}
ic_point projcurv IMP_PTS pnt.b2.6 {pnt.102.0 crv.67}
ic_point projcurv IMP_PTS pnt.b3.1 {pnt.05 crv.66}
ic_point projcurv IMP_PTS pnt.b3.2 {pnt.17q crv.66}
ic_point projcurv IMP_PTS pnt.b3.3 {pnt.06 crv.66}
ic_point projcurv IMP_PTS pnt.b3.4 {pnt.07 crv.66}
ic_point projcurv IMP_PTS pnt.b3.5 {pnt.102 crv.66}
ic_point projcurv IMP_PTS pnt.b3.6 {pnt.102.0 crv.66}
ic_point projcurv IMP_PTS pnt.b4.1 {pnt.b3.1 crv.57}
ic_point projcurv IMP_PTS pnt.b4.2 {pnt.b3.2 crv.57}
ic_point projcurv IMP_PTS pnt.b4.3 {pnt.b3.3 crv.57}
ic_point projcurv IMP_PTS pnt.b4.4 {pnt.b3.4 crv.57}
ic_point projcurv IMP_PTS pnt.b4.5 {pnt.b3.5 crv.57}
ic_point projcurv IMP_PTS pnt.b4.6 {pnt.b3.6 crv.57}
```

```
ic_point projcurv IMP_PTS pnt.b5.1 {pnt.b3.1 crv.66a}
ic_point projcurv IMP_PTS pnt.b5.2 {pnt.b3.2 crv.66a}
ic_point projcurv IMP_PTS pnt.b5.3 {pnt.b3.3 crv.66a}
ic_point projcurv IMP_PTS pnt.b5.4 {pnt.b3.4 crv.66a}
ic_point projcurv IMP_PTS pnt.b5.5 {pnt.b3.5 crv.66a}
ic_point projcurv IMP_PTS pnt.b5.6 {pnt.b3.6 crv.66a}
ic_point projcurv IMP_PTS pnt.t1.1 {pnt.05 crv.64}
ic_point projcurv IMP_PTS pnt.t1.2 {pnt.17q crv.64}
ic_point projcurv IMP_PTS pnt.t1.3 {pnt.06 crv.64}
ic_point projcurv IMP_PTS pnt.t1.4 {pnt.07 crv.64}
ic_point projcurv IMP_PTS pnt.t1.5 {pnt.102 crv.64}
ic_point projcurv IMP_PTS pnt.t1.6 {pnt.102.0 crv.64}
ic_point projcurv IMP_PTS pnt.t2.1 {pnt.05 crv.63}
ic_point projcurv IMP_PTS pnt.t2.2 {pnt.17q crv.63}
ic_point projcurv IMP_PTS pnt.t2.3 {pnt.06 crv.63}
ic_point projcurv IMP_PTS pnt.t2.4 {pnt.07 crv.63}
ic_point projcurv IMP_PTS pnt.t2.5 {pnt.102 crv.63}
ic_point projcurv IMP_PTS pnt.t2.6 {pnt.102.0 crv.63}
ic_point projcurv IMP_PTS pnt.t3.1 {pnt.05 crv.62}
ic_point projcurv IMP_PTS pnt.t3.2 {pnt.17q crv.62}
ic_point projcurv IMP_PTS pnt.t3.3 {pnt.06 crv.62}
ic_point projcurv IMP_PTS pnt.t3.4 {pnt.07 crv.62}
ic_point projcurv IMP_PTS pnt.t3.5 {pnt.102 crv.62}
ic_point projcurv IMP_PTS pnt.t3.6 {pnt.102.0 crv.62}
ic_point projcurv IMP_PTS pnt.t4.1 {pnt.t3.1 crv.61}
ic_point projcurv IMP_PTS pnt.t4.2 {pnt.t3.2 crv.61}
ic_point projcurv IMP_PTS pnt.t4.3 {pnt.t3.3 crv.61}
ic_point projcurv IMP_PTS pnt.t4.4 {pnt.t3.4 crv.61}
ic_point projcurv IMP_PTS pnt.t4.5 {pnt.t3.5 crv.61}
ic_point projcurv IMP_PTS pnt.t4.6 {pnt.t3.6 crv.61}
ic_point projcurv IMP_PTS pnt.t5.1 {pnt.t3.1 crv.62a}
ic_point projcurv IMP_PTS pnt.t5.2 {pnt.t3.2 crv.62a}
ic_point projcurv IMP_PTS pnt.t5.3 {pnt.t3.3 crv.62a}
ic_point projcurv IMP_PTS pnt.t5.4 {pnt.t3.4 crv.62a}
ic_point projcurv IMP_PTS pnt.t5.5 {pnt.t3.5 crv.62a}
ic_point projcurv IMP_PTS pnt.t5.6 {pnt.t3.6 crv.62a}
ic_point projcurv IMP_PTS pnt.ct.1 {pnt.90.2 crv.70a}
ic_point projcurv IMP_PTS pnt.ct.2 {pnt.88 crv.70a}
ic_point projcurv IMP_PTS pnt.ct.3 {pnt.17ff crv.70a}
ic_point projcurv IMP_PTS pnt.ct.4 {pnt.89 crv.70a}
ic_point projcurv IMP_PTS pnt.ct.5 {pnt.90 crv.70a}
ic_point projcurv IMP_PTS pnt.ct.6 {pnt.90.0 crv.70a}
ic_point projcurv IMP_PTS pnt.cb.1 {pnt.90.2 crv.69a}
ic_point projcurv IMP_PTS pnt.cb.2 {pnt.88 crv.69a}
ic_point projcurv IMP_PTS pnt.cb.3 {pnt.17ff crv.69a}
ic_point projcurv IMP_PTS pnt.cb.4 {pnt.89 crv.69a}
ic_point projcurv IMP_PTS pnt.cb.5 {pnt.90 crv.69a}
ic_point projcurv IMP_PTS pnt.cb.6 {pnt.90.0 crv.69a}
ic_save_tetin /home/cfd/cjohnson/NLF2/NACA23012aerofoil/N23012_M1p94_M2p52_aboutLE/script_NACA23012.tin 0
     0 {} {} 0   ### save geometry
```

B.11 Elliptic and Load Distribution Calculation for a Wing

```
/* PROGRAM TO FIT AND COMPARE A PARABOLIC DISTRIBUTION OF CL OVER A WING */

/* This program reads a CL distribution file, normalises it and fits a parabola through it and then
 * denormalises it.
 * total load or CT.
 * 2 arguments are needed: 1) The CL file to read the span and CL values from them.
 *                         2) The file to write a parabola to.
 * ------------------------------------------------------------------------------------------------ */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>

struct {                                    /* structure */
        float CL;                           /* CL value */
        float B;                            /* span value */
        } curve[1000];

main(int argc, char *argv[])
{

float Error;                                /* Difference between parabola and CL curve */
int i,j, jj;                                /* counters */
float CL, b, cl;                            /* Load variables */
float minCL,maxCL, minspan, maxspan;        /* Normalising limits */
float x[1000],y[1000],E[1000];              /* Parabola points matrices */
char filename[128];
FILE *input, *out;

/*--- ALL CL READ AND FIT PARABOLA ---*/
strcpy(filename,argv[1]);
input = fopen(filename,"r");
jj=0;
maxCL = -1000; minCL = 1000;
maxspan = -1000; minspan = 1000;
while((j = getc(input)) != EOF )
        {
                ungetc(j,input);
                fscanf(input,"%f %f \n",&b,&cl)==2;
                curve[jj].CL=cl;
                curve[jj].B=b;
                if (curve[jj].CL >= maxCL) {maxCL = curve[jj].CL;}
                if (curve[jj].CL <= minCL) {minCL = curve[jj].CL;}
                if (curve[jj].B <= minspan) {minspan = curve[jj].B;}
```

```
                if (curve[jj].B >= maxspan) {maxspan = curve[jj].B;}
                jj=jj+1;
        }
fclose(input);
/*--- NORMALISE ---*/
for (i=0;i<jj;i++)
{
        x[i] = (curve[i].B - minspan) / (maxspan - minspan);
        curve[i].CL = (curve[i].CL - minCL) / (maxCL - minCL);
}

/*--- PARABOLA ---*/
strcpy(filename,argv[2]);
out=fopen(filename,"w");
for (i=0;i<jj;i++)
{
        y[i]=1-x[i]*x[i];
        fprintf(out,"%f %f\n",x[i] * (maxspan - minspan) + minspan,y[i] * (maxCL - minCL) + minCL);
}
Error = 0;
for (i=1;i<jj;i++)
{
        E[i]=(y[i] - curve[i].CL) * (y[i] - curve[i].CL) / (curve[i].CL * curve[i].CL);
        Error= Error+E[i];
}
printf("Error = %f\n\n",Error);
fclose(out);
return 0;
}
```

B.12 Korn's Method to Find Drag Divergence Mach Number

```
/*
 * KORN'S METHOD TO FIND DRAG DIVERGENCE MACH NUMBER.
 * The Technology Factor used here is 0.95.
 *
 */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>

main(int argc, char *argv[])
{
FILE *input,*output;

char filename[128];

float A1,A2,A3,CL,x,t,MDD;
int jj,j;

/* open files */
jj=0;
strcpy(filename,argv[2]);
output = fopen(filename,"w");
strcpy(filename,argv[1]);
input = fopen(filename,"r");
fscanf(input,"A1\t\tA2\t\tA3\t\tCL\t\tmaxt xcoord\tt/c\t\tMDD\n");
while((j = getc(input)) != EOF )
        {
                ungetc(j,input);
                fscanf(input,"%f %f %f %f %f %f\n",&A1,&A2,&A3,&CL,&x,&t)==6;
                MDD = 0.95 - CL/10 - t;
                fprintf(output,"%f %f %f %f\n",A1,A2,A3,MDD);
                jj=jj+1;
        }

fclose(input);
fclose(output);
return 0;
}
```

B.13 UH60-A Parameter Modification Program

```
/*
 * PROGRAM TO FIND THE COORDINATES TO MODIFY THE UH60 TIP WITH SWEEP.
 * See Figure for specific defintions.
 */

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>

main(int argc, char *argv[])
{
float pi,Area,span,tipR,sweep,x,r,T1,T2,C,C1,theta,z,z1,z2,anh;
pi=3.141592654;

/*--- DEFINE CONSTANTS ---*/
Area = 1.111672518;
span = 1.0737;
tipR = 14.4025;

printf("Enter sweep and anhedral angle in deg:\n");
scanf("%f %f",&sweep,&anh);
sweep = sweep /180 * pi;
anh = anh /180 * pi;

/*--- CALCULATE ---*/
x = sqrt(2 - 2 * cos (sweep));
r = x * cos (sweep/2);
T1 = x * sin (sweep/2);
T2 = (span -r) * tan (sweep);
C = (2*Area - sweep + r - span) / span;
C1 = ((C-1) * r / span) + 1;
theta = 3.76696895 * pow((tipR + r),2) - 110.837065731 * (tipR + r) + 823.338742751;    // tip twist
z = - 0.0196387563 * pow((tipR + r),2) + 0.537139079948 * (tipR + r) - 3.67093898682;   // inherent
    anhedral
z1 = r * tan (anh);
z2 = span * tan (anh);

printf("sweep:\t%f\n",sweep * 180/pi);
printf("x:\t\t%f\n",x);
printf("r (pos):\t\t%f\n",r);
printf("T1(inner):\t\t%f\n",T1);
printf("T2 (tip):\t\t%f\n",T2);
printf("C (tip):\t\t%f\n",C);
printf("C1 (inner):\t\t%f\n",C1);
printf("theta at r:\t\t%f\n",theta);
printf("z at r:\t\t%f\n",z);
printf("z1 at r:\t\t%f\n",z1);
printf("z2 at tip:\t\t%f\n",z2);

return 0;
}
```

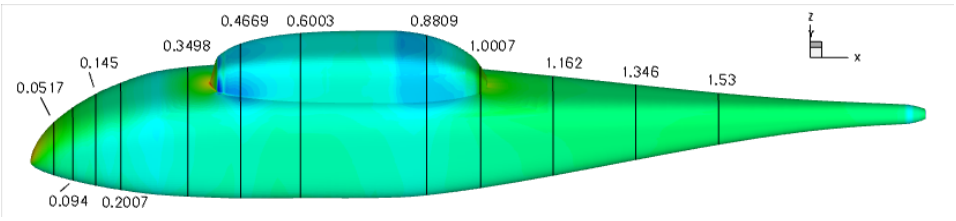
Appendix C

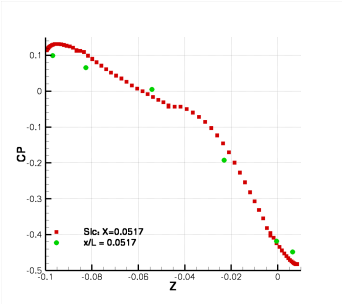
Other Related Data

C.1 ROBIN Fuselage

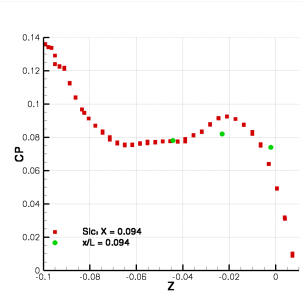
The ROBIN (ROtor Body INteraction) body was created to investigate the rotor wake-fuselage interaction[104]. It is a simplified geometry representing a helicopter fuselage albeit, more streamlined than is typically found. The shape of the fuselage is determined mathematically by a set of parameters that make up super-ellipsoid equations[105]. These equations create a curve at a cross-section along the length of the fuselage. By varying these parameters along the length of the fuselage, the shape can be modified. Different parts can be made and positioned together, such as the doghouse and the main fuselage of the ROBIN body. This is described in more detail in Section 3.2.3.

The grid size was approximately 3 million cells and needed 144 blocks. The conditions for CFD were a Mach number of 0.062 and a Reynold's number of approximately 3 million. Figure C.1 shows the C_P comparison with experimental data for the isolated fuselage obtained from Berry and Althoff[105]. The results are well matched until the doghouse, after which deterioration in the comparison with the experiments begins to occur especially after the dog house region. This may be due to the fact that the hub was not modelled for the CFD simulation. This same conclusion was drawn in the report by Berry and Althoff[105].

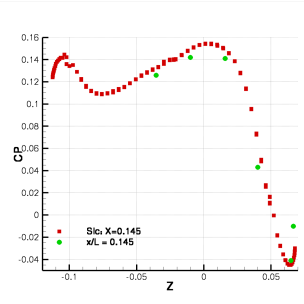




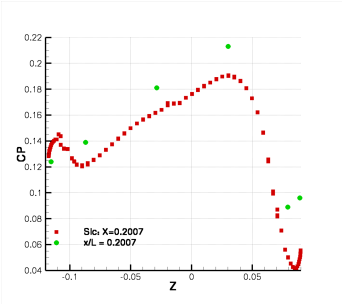
x/L = 0.0517



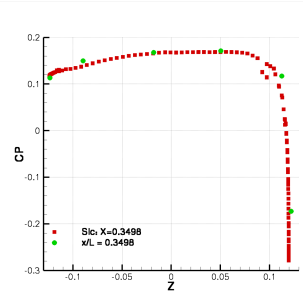
x/L = 0.094



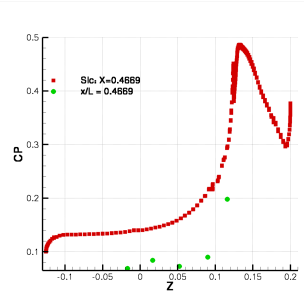
x/L = 0.145



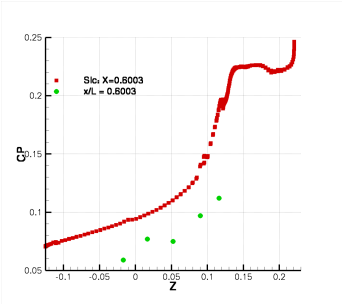
x/L = 0.2007



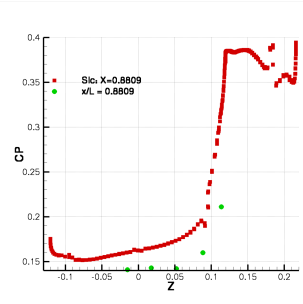
x/L = 0.3498



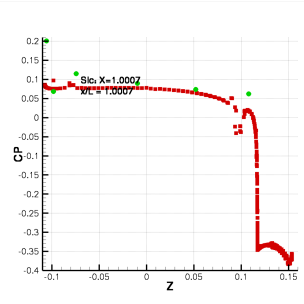
x/L = 0.4669



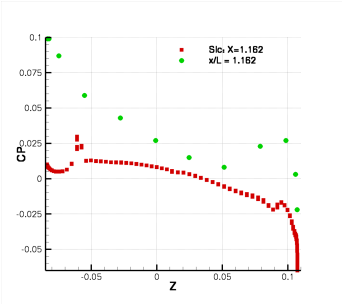
x/L = 0.6003



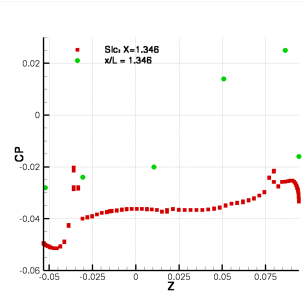
x/L = 0.8809



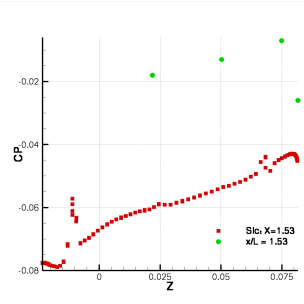
x/L = 1.0007



x/L = 1.162



x/L = 1.346



x/L = 1.53

Figure C.1: *ROBIN fuselage compare C_p with experimental data. Mach number = 0.062, Re = 3×10^{-6}.*

C.2 ROBIN-mod7 Fuselage

The ROBIN-mod7 is a modification of the original ROBIN (ROtor Body INteraction) fuselage. It was part of a joint effort by the United States and France to analyse and study fuselage drag reduction[10] since a significant amount of the power used to overcome drag is used to overcome fuselage drag. Experimental work was carried out in a wind tunnel and RANS based calculations were made using a number of different turbulence models on the isolated fuselage. Active flow control was employed to reduce the drag at the rear ramp of the fuselage where separation occurs. The experiment was conducted at a Mach number of 0.1 and a fuselage-length based Reynold's number of 1.6 million. Computations were carried out using OVERFLOW developed by NASA and elsA developed by ONERA. It was found that the inclusion of the sting and windtunnel in the CFD model, made little difference to the separation. The fuselage was analysed at two angles of attack, 0 and 5 degrees. Mach 0.1 was selected here because C_D was constant above Mach 0.09. Figure C.2 shows the HMB solution along with the experimental data for this fuselage at 0 degrees angle of attack and free stream Mach number of 0.1. The calculation was run with a CFL number of one (although this could be increased comfortably) and the $\kappa - \omega$ turbulence model. The grid size was approximately 3 million cells. The total drag coefficient obtained was approximately 0.012 based on the front view cross-sectional area of the fuselage with a negligible lift coefficient. Figure C.3 shows how the rear ramp gradient can be changed by modifying the parameters, showing a possible problem suitable for optimisation. The pressure contours and streamlines are also shown depicting separation at the rear of the fuselage.

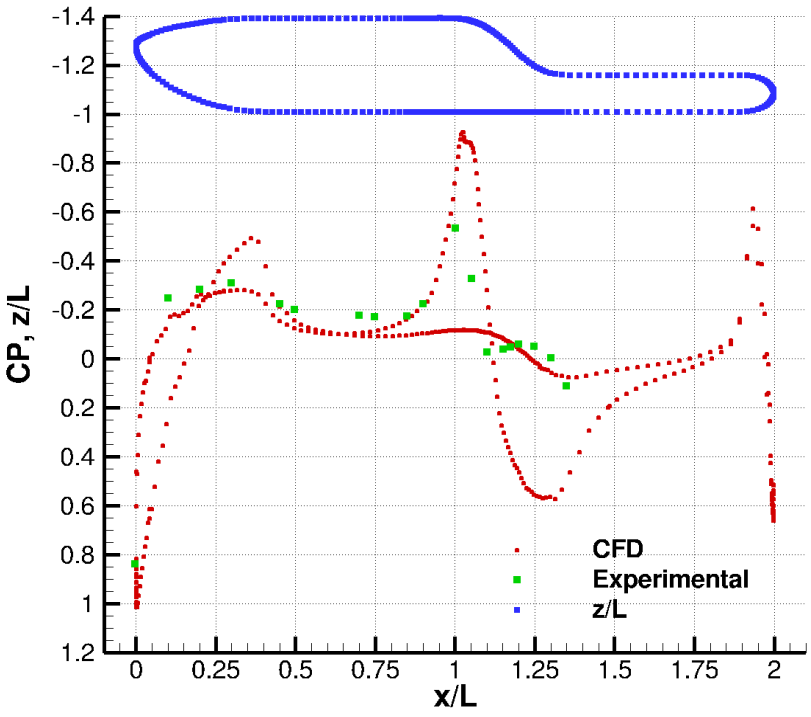
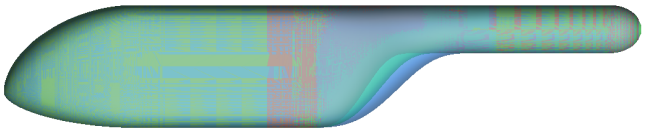
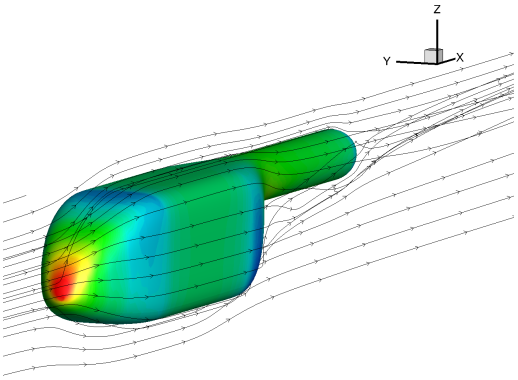


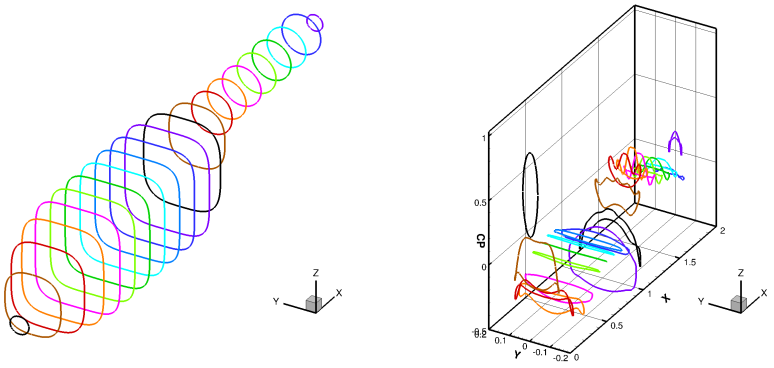
Figure C.2: *ROBIN-mod7 fuselage C_p comparison from Schaeffler et al. at 0 degrees angle of attack and Mach number 0.1*[10]



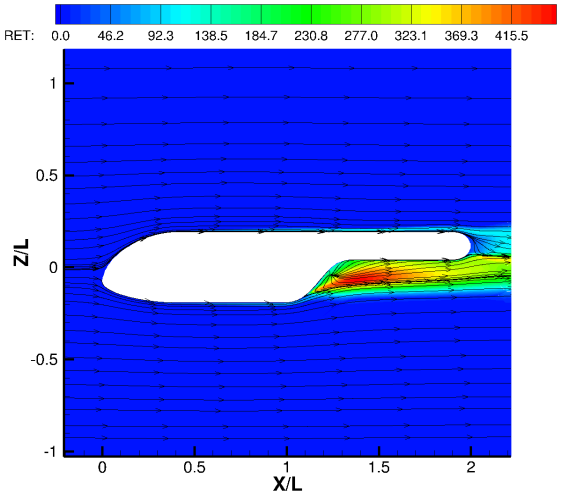
(a)



(b)



(c)



(d)

Figure C.3: *ROBIN-mod7 fuselage (a) with varying rear-ramp gradient, (b) with surface pressure contours, (c) with C_p distributions along the fuselage main axis (d) with Reynolds turbulence and streamlines showing separated region.*

C.3 HARTII

The aim of the HART-II (Higher harmonic control Aeroacoustics Rotor Test) rotor test was to investigate rotor wake measurements using PIV techniques and obtaining data about airloads, acoustics and blade deflection[108] . The rotor test stand, called ROTEST, was used to support the rotor (shown in Figure C.4). The test facility, consists of the drive system, the rotor balance, the control system, and the measuring system. It is designed for model rotors and contains all the test systems while causing the least interference acoustically and aerodynamically as possible. $(www.dlr.de/ft/en/desktopdefault.aspx/tabid - 1387/1915_read - 3372//r/)$

The test was carried out at a free stream Mach number of 0.2 and Reynold's number of 2 million with an angle of attack of -6 degrees[155] . The grid size was approximately 2 million cells. The C_L obtained was 0.002838 and the C_D was 0.163, 79% of which was pressure drag.

Figure C.4: *HARTII test showing ROTEST system.*

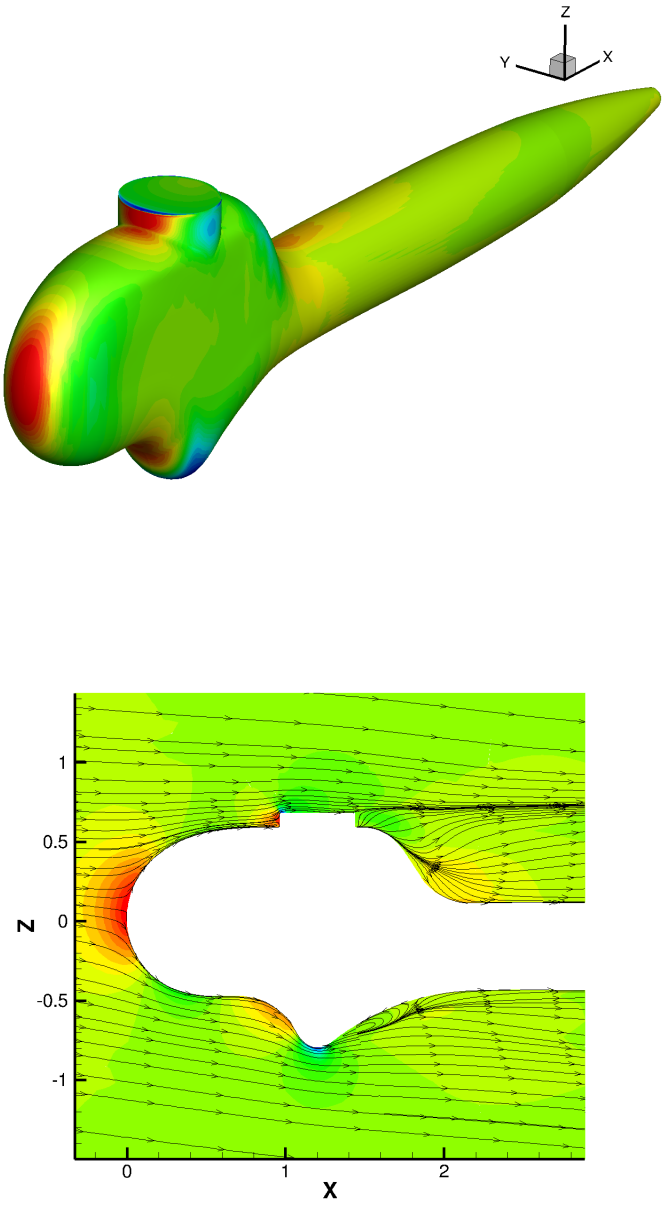


Figure C.5: *HARTII fuselage at angle of attack -6 degrees, Mach 0.1 and Reynold's number 1 million. The contours are pressure contours.*

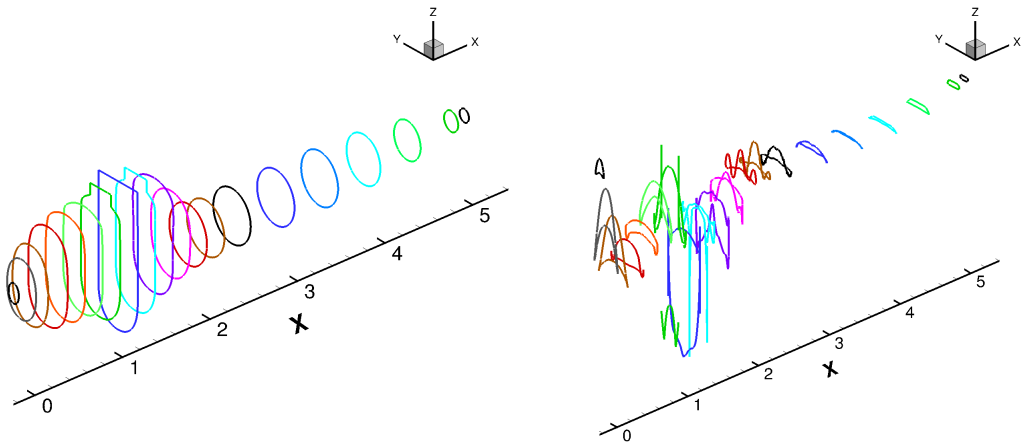
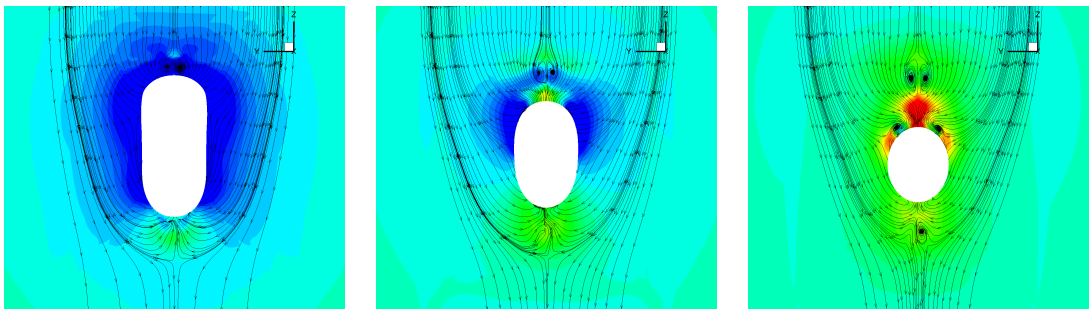


Figure C.6: *HARTII fuselage at angle of attack -6 degrees, Mach 0.1 and Reynold's number 1 million. Slices showing the pressure curves along the longitudinal axis.*



x = 1.6 x = 1.8 x = 2.0

Figure C.7: *HARTII fuselage at angle of attack -6 degrees, Mach 0.1 and Reynold's number 1 million. Pressure contours and streamlines showing sdeparated regions.*

Appendix D

CD Contents

PhD thesis

Publications:
Journal Papers:
Optimisation of Aspects of Rotor Blades in Forward Flight, Int. J. Engineering Systems Modelling and Simulation, 2012.
A Framework for Optimising Aspects of Rotor Blades, The Aeronautical Journal, 2011.
A Framework for the Optimisation of a BERP-like Blade, Journal of Aircraft (review).
Papers in Conference Proceedings:
Optimisation of Aspects of Helicopter Rotor Blades and Fuselage, 37th ERF, 2011.
Optimising Aspects of Rotor Blades in Forward Flight, 49th AIAA, 2011.
Rotor Tip Optimisation in Forward Flight, 46th Symposium of AARB (ERF), 2011.
Development of a Framework for Optimising Aspects of Rotor Blades, 66th AHS, 2010.

ICEMCFD Projects
BERP-like tip: Forward flight, hover grids and ICEM replay files.
UH60-A: Forward flight, hover grids for optimum UH60-A blade.
Rectangular Blade: Hover grid for the NACA 0012 blade.

Fuselage:
ROBIN: replay script, geometry and blocking files with additional parts.
ROBIN-mod7: replay script, geometry and blocking for 3 versions of ROBIN-mod7.
JAXA JMRTS: replay script, geometry and blocking, program files for modification.

Aerofoils:
NACA example: NACA 0009.
RAE 2822.

Wing Planform: geometry, blocking, grid and replay file.

Poster : Optimisation of Rotor Blades using Computational Fluid Dynamics, C.S. Johnson and G.N. Barakos.
Internal Reports
TN10-013: A Parametric Mesh Generation for HMB, containing an example.
TN12-003: Actuator Disk Models in HMB.
TN10-022: A Metamodel and Optimisation Procedure for HMB, containing an ANN, Kriging and Genetic Algorithm Package.
TN10-BERP: BERP tip Grid Generation Technique in ICEMCFD.
TN10-UH60: Parameterisation Technique for the UH60-A blade.