Implementation of a trim routine in a rotor model for the numerical simulation of helicopter flow-fields

by

Vaneshen Naidoo

Thesis presented in partial fulfillment for the Degree of Master of Science

In the Department of Mechanical Engineering
University of Cape Town

November 5, 2006

Declaration

I hereby:

1. Grant the University of Cape Town free license to reproduce the aforementioned dissertation in whole or in part for the purpose of research;

2. Declare that:

   (a) The aforementioned dissertation is my own unaided work. Except for normal guidance of my supervisor, I have received no assistance apart from that stated below;

   (b) Except as stated below, neither the substance or any part of the thesis has been submitted in the past, or is being submitted for a degree in this, or any other University;

I have used the Vancouver referencing system for citation and referencing. Each significant contribution and quotation in this report from works by other people have been attributed and have been referenced and cited.

Signed by candidate

Signature:... Date:.....10/11/2006.....

Vaneshen Naidoo

Abstract

The aim of the current project is to develop, validate and implement a trim routine for a numerical rotor model, developed for the use in simulations of a helicopter exterior flow-field.

In this investigation a ROBIN fuselage geometry was utilised. Simulations of the fuselage without the rotor were carried out initially so that investigations into the computational grids and turbulence models could be done. The computational simulations were performed in the commercially available CFD solver, FLUENT®

Computational grids were created for the near wall modelling approach and wall function approach. Some of the more applicable turbulence models available in the solver were compared. For the wall function approach grids the $k - \epsilon$, and its variants, the RNG and realizable models were found to be suitable choices. For the near wall modelling approach grids used, the SST models performed the best.

The rotor model used during this investigation utilised a combination of blade element and actuator disk theory. Forces exerted by the rotor are calculated with the use of blade characteristics and flow properties. These forces were applied to the domain as momentum sources terms. The rotor model was incorporated with the CFD solver, through the use of a User Defined Function (UDF).

The method used to trim the rotor was the Newton-Raphson Iterative method. This trim routine was incorporated in the UDF used for the rotor model. Tests were conducted, on a 'rotor-alone' model, as well as the rotor and fuselage model. The trim routine was found to be rigorous and managed to trim the rotor in each of the tests conducted. Good agreement between experimental and numerical collective pitch angle and cyclic pitch coefficients were found. Also the effect of the fuselage on the trim conditions proved to be minimal.

Nomenclature

Symbols

A_1	Lateral cyclic pitch coefficient
A	Area (m^2)
B_1	Longitudinal cyclic pitch coefficient
C_d	Coefficient of drag
c	Chord (m)
C_l	Coefficient of lift
C_{M_x}	Coefficient of rolling moment
C_{M_y}	Coefficient of pitching moment
C_p	Coefficient of pressure
C_T	Coefficient of thrust
d	Diameter (m)
D	Drag (N)
L	Lift (N)
M_x	Coefficient of rolling moment $(N.m)$
M_y	Coefficient of pitching moment $(N.m)$
N	Number of blades
p	Pressure (Pa)

Dedications

This thesis is dedicated to my father who passed away suddenly on the 22^{nd} of April 2006. It is because of the love and support that both he and and my mother showed that I have managed to get my undergraduate degree, and complete this thesis. Despite all the hardships that he endured, he never gave up and because of his teachings nor will I. My only hope is that he is now resting in peace and watching us get along with a smile on his face. He will forever be missed.

Acknowledgments

Firstly I would like to thank God for giving me the strength, and granting me the serenity to achieve the things I can and the wisdom to do so.

I would like to also thank Dr. C.J. Meyer (Doc) for all his help, guidance, encouragement, and especially his tolerance of the situation and extra responsibilities that I faced during this project.

A special thanks has to go to Dwain Dunn, Darnell Engelbrecht, Jean-Paul Pelteret, Amy Barty, Kevin Appa, Andrew Mcbride, and Rene Heise who went beyond the call of duty to assist me.

Also thanks has to be given to the CFD postgraduate group, and undergraduate groups of 2005 and 2006, for their company during the long days and nights, as well as the cerecam group for the interesting discussions at lunch time.

Finally I would like to thank my parents, siblings, family and friends, especially my parents, who never stopped believing in me, for their encouragement and support through the years.

Q Torque (Nm)

r Blade element radial position, or radial unit vector (m)

Re Reynolds Number

R Rotor radius (m)

θ Blade pitch $(^\circ)$

T Thrust (N)

θ_0 Collective pitch angle $(^\circ)$

θ_1 blade twist $(^\circ)$

t Time (s)

$t_{R_{disk}}$ Rotor disk thickness (m)

v Velocity $(m.s^{-1})$

V Forward speed of the helicopter $(m.s^{-1})$

Greek Symbols

α Angle of attack $(^\circ)$

β Angle between the relative velocity vector and the rotor blade plane of rotation $(^\circ)$

ψ Azimuthal angle $(^\circ)$

δ_r Blade element radial thickness

μ Advance ratio

Ω Rotor rotation vector $(rad.s^{-1})$

ρ Density $(kg.m^{-3})$

Subscripts

d	Drag
i	Index
j	Index
k	Index
l	Lift
M_x	Rolling moment
M_y	Pitching moment
p	Pressure
r	Relative
T	Thrust
t	tangential

Acronyms

2D	Two dimensional
3D	Two dimensional
CFD	Computational Fluid Dynamics
RANS	Reynolds Averaged Navier-Stokes equations
RNG	Renormalization Group
RSM	Reynolds Stress Model
UDF	User Defined Function

Glossary

Actuator disk An infinitely thin disk, which occupies the same area as the rotor and generates a uniform induced velocity.

Advance ratio The velocity of the helicopter with respect to its rotor speed.

Azimuthal angle The horizontal angle between two lines or planes that intersect. The angle is usually take from a reference point.

Coning A phenomenon that occurs on a rotor, when the blades rise, during take off and flight, above the straight positions into a coned position.

Numerical diffusion A non real aspect which has the reputation of affecting the accuracy of solutions in a CFD model.

Parallelisation The process of making a serial UDF into one which can be used in a parallel environment.

Pitch angle Defines the angle of attack of the entire blade with respect to the disk plane.

Pitching Moment A moment that tends to pitch a flying body about its lateral axis.

Potential flow Flow in which the velocity of the flow is the gradient of a scalar function, known as the velocity potential.

Rolling Moment A moment that tends to rotate a flying body about its longitudinal axis.

Solidity ratio The ratio of rotor blade area to area swept by the rotor blades.

Swashplate A mechanism that turns non-rotating control movements into rotating control movements.

Tip effects A secondary, strong flow phenomenon that occurs around the tip of the blade.

Trim Routine A routine that is used to calculate the correct pitch angle and coefficients to achieve a thrust coefficient and eliminate moments around the hub.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background to investigation

The first semi-practical idea of a human-carrying helicopter was conceived by Leonardo da Vinci around 1490. However, it was not until the invention of the powered aircrafts in the 20th century that helicopters, also known as rotary-wing aircrafts, were developed. Throughout the years the basis of the helicopter configuration has not changed much, and most helicopters that exist today are based on the first single-rotor helicopter that was made by Igor Sikorsky in 1939 [15].

In comparison to conventional fixed-wing aircraft, helicopters found to be much more complex, more expensive to buy and operate, and are more limited in speed, range, and payload. The complexity is attributed to the unique aerodynamic features that a rotary wing exhibits, and to the many components that make up a helicopter. Although this is the case, they do have a great advantage in that they are highly manoeuvrable, since helicopters can hover in place, reverse, and above all take off and land vertically.

It is because of this advantage that much time and money has been put into research of rotary wing aircrafts. In previous years the investigations of the flow-fields that exist around helicopters were conducted through the use of wind tunnel experiments. However, this method has proven to be very expensive. Thus, over the last few decades various techniques have been suggested for the modelling of rotary-wing flow fields.

Early attempts to model helicopter rotor characteristics were based on one dimensional momentum theory developed for aeroplane propellers. The rotor is considered to be an infinitely thin disk, which occupies the same area as the rotor, generating a uniform induced velocity at the rotor disk. The disk is commonly known as an actuator disk. This method allowed basic simulation of rotor performance characteristics such as

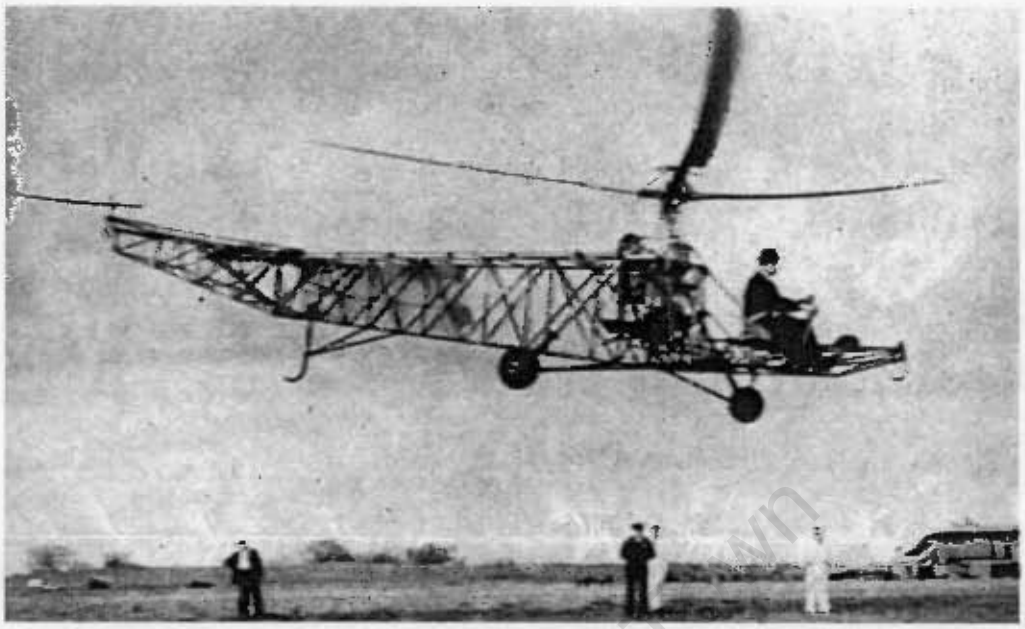

Figure 1.1: Igor Sikorsky at the controls of the VS-300 which is hovering during a public demonstration flight[15]

thrust and torque, and was later improved by accounting for rotation in the wake. It is deficient in two very important aspects. Firstly, momentum theory assumes uniform flow in the rotor wake, with no facility for defining azimuthal variations; and it makes no provision for information about the characteristics of the rotor blades. For these reasons, one dimensional momentum theories are not suitable for any but the most basic of simulations.

Wake methods such as the classic vortex theories, and variations thereof, were introduced in early attempts to model wakes induced by a rotary wing. This method is based on the assumption of potential flow, with the rotor blade being represented by a system of bound and trailing vortices, and the wake by a rigid non-contracting helical vortex sheet. The major shortfall of wake methods in general, is the assumption of potential flow. Despite these limitations, wake methods remain valuable tools for rotor modelling, largely to their ability to predict major flow characteristics in a cost efficient manner.

More recently, great interest has been raised in the investigation, and fluid flow modelling ability of the Computational Fluid Dynamics (CFD) packages for helicopter aerodynamics. CFD is a computer code that uses the numerical algorithms, as dictated by the Navier-Stokes equations, to analyse the fluid flow conditions during computer simulations. The CFD process is basically made up of three stages. These are the pre-

processor, solver and post processor stages, and the basis of this technique is the use of some form of discretisation scheme to permit numerical solution of certain governing equations over the domain of interest. It describes a relatively new discipline in fluid dynamics, made possible by the increasing power and availability of computing resources. However, at this time, the use of the full, unsteady Navier-Stokes equations for fluid flow around the individual rotor blades is beyond the capacity of computing resources available to most CFD users.

Various approaches have been demonstrated as means of modelling the effects of a rotary-wing, without necessarily modelling the individual blades themselves. While several approaches exist for representing time-averaged rotor effects, the fundamentals are commonly based on the calculation of aerodynamic loads directly from two dimensional (2D) blade-element theory. The most common methods used for accounting for the rotor's influence on the flow-stream, are the application of either internal boundary conditions, or momentum source terms, in the form of an actuator disk.

Several authors have used internal boundary conditions to represent a rotor in a time averaged manner, commonly in the form of a disk-like pressure boundary, coupled with an induced swirl velocity. This approach was used by Fetjek and Roberts [5] to simulate the wing/rotor interaction for a tilt rotor in hover, using the thin-layer Navier-Stokes equations to describe the flow-field. In the region of the rotor, the computational domain was excluded from the implicit solution, with flow properties being updated explicitly with values calculated by an independent rotor model. Chaffin and Berry [4] also used a similar technique to carry out the same task. They essentially modified the approach undertaken by Fetjek and Roberts [5] to represent the rotor with a disk boundary of zero thickness.

The use of momentum source terms to represent a rotor is, however, less complicated than the application of internal boundary conditions. This is because the rotor is represented simply by an actuator disk of finite thickness, whereby the rotor's influence is modelled in terms of the momentum it imparts to the fluid flowing though it.

Rajagopalan and Mathur [17] were early exponents of this theory in a rotary-wing application, the three dimensional (3D) analysis of a rotor in forward flight. The steady, incompressible, laminar Navier-Stokes equations were solved for the flow domain; the momentum source terms being explicitly added to the appropriate cells in the region of the actuator disk. Blade element theory was used to determine the aerodynamic loads with respect to radial and azimuth location, with blade pitch harmonics explicitly defined from experimental values, and a with a constant blade cone angle.

Momentum source terms have been used to represent the aerodynamic influence of fans and rotors in other applications. Recently, Meyer and Kröger [13] modelled the effect of an axial flow fan on the velocity field in the vicinity of the fan blades, with the use of momentum source terms. Hotchkiss [10] then created a numerical fan model, based on the methods used by Meyer and Kröger [13], and validated the accuracy of the model for both aligned and non-aligned inflow conditions. His results were compared to off-axis inflow fan conditions investigated, experimentally, by Stinnes and von Backstrom [19]. Agreement between experimental and numerical results was excellent.

Since the results were very favourable, Hotchkiss [10] modified the fan model to ensure that it could be used in a rotor application. One of the modifications made was the inclusion of a subroutine in the rotor model to define the cyclic pitch profile of the rotor in forward flight. It is necessary in helicopters to vary the blade pitch angle as a function of azimuth angle, to account for the free-stream velocity component in forward flight. This is to prevent pitching and rolling moments being created by unequal aerodynamic load characteristics along diametrically opposing blades.

Even though Hotchkiss [10] included this routine in the rotor model, accurate aerodynamic predictions in forward flight was not possible since optimisation of the pitch angle did not take place.

1.2 Objectives of this thesis

Since the investigation undertaken by Hotchkiss [10] did not include an optimisation of the pitch angle for the forward flight conditions, the objective for this project is to further develop the technique used by Hotchkiss [10], to simulate the flow-field surrounding the rotor in forward flight conditions, by optimising the pitch angle.

In accomplishing this task several other objectives have to be undertaken. These objectives are :

- To develop and validate a method that can be incorporated into the numerical rotor model and optimise the pitch angle correctly.
- Create the geometry for the helicopter fuselage and rotor, as well as the grid which can capture the flow conditions accurately.
- Investigate the various CFD solver variables which needs to be used.

1.3 Theoretical approach

It is strongly believed that CFD can be used to model and accurately predict the aerodynamics of the flow-field encountered by a rotary wing aircraft in various flight simulations. The present method used to optimise the pitch angle, is by incorporating a trim routine in the rotor model to specify the correct thrust values, as well as balancing both the pitching and rolling moments.

A modelling procedure similar to that of Meyer and Kröger [13] was used for the rotor model. It was however modified to incorporate the effects of azimuthal variations of blade geometry present in a helicopter rotor in forward flight. The rotor model presented is based on a representation of the rotor using momentum sources and is coupled with a commercially available CFD solver code, FLUENT®, through the use of user defined functions (UDF), written in the C programming language.

Steady, incompressible, viscous, Reynolds-averaged conservation equations are solved by FLUENT® for this investigation. Geometric details of the rotor need to be specified by the user. Momentum source terms are calculated by the rotor model, which receives flow data from the solver, calculates the aerodynamic loads from 2D blade element theory, and returns the source terms to the flow solver.

The trim routine was included in the UDF used for the rotor model. This was done so that the flow properties needed for calculations could be obtained from the solver, and so that the collective pitch angle and cyclic pitch coefficients could be corrected immediately. Correction of the collective pitch angle and cyclic pitch coefficients is the purpose of the trim routine, and was done to ensure that the correct aerodynamic loads would be predicted and assigned to the rotor in the flow solver.

1.4 Scope and Limitations

The main limitation in this project is computational power needed to generate the models, and reach final solutions. This is mainly because the rotor model code is not parallelised, which means that simulations, involving the rotor, could only be run on single processor, and a restriction on the number of cells used in the pre-processor phase was enforced. The documentation of previous and experimental investigations received was of poor quality and in black and white. Therefore, a comparison of the aerodynamic effects was unlikely.

Therefore, the simulations undertaken during this thesis will focus on optimisation of the CFD methods used to generate solutions for the helicopter fuselage, and imple-

mentation of the trim routine for the simulations concerning the rotor model.

1.5 Plan of Development

This report begins with a brief description of the literature reviewed. It then focuses on the geometry, boundary conditions, grid and CFD solver variables implemented for initial, helicopter fuselage, simulations. An evaluation of the turbulence models also takes place. A description of the rotor model used is then carried out, and this is followed by a description of the trim routine used, as well as the validation of the trim routine.

Chapter 2

Literature review

In this chapter, the equations that govern the fluid flow around a helicopter configuration are presented, and the methods used to model turbulence in CFD are discussed. A depiction of forward flight helicopter aerodynamics is also shown. Finally brief descriptions of the trim routine implementations used in previous investigations are revealed.

2.1 Governing Equations

The conservation equations that govern fluid flow around a helicopter configuration are given by the continuity and Navier-Stokes equations. These equations need to be solved so that the properties in the system of interest can be obtained.

2.1.1 Continuity Equation

The continuity equation which is also known as the conservation of mass it is given by:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0 \tag{2.1}$$

For an incompressible fluid, the continuity equation is given by

$$\frac{\partial}{\partial x_i}(u_i) = 0 \tag{2.2}$$

2.1.2 Navier-Stokes Equations

The Navier-Stokes equations for the conservation of momentum represents the momentum equations in each of the coordinate directions. The momentum equation is given

by

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j) = \rho b_i - \frac{\partial p}{\partial x_i} + \mu(\frac{\partial}{\partial x_i}\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}] - \frac{2}{3}\frac{\partial u_k}{\partial x_k}) + S_i \tag{2.3}$$

where S_i is a source term
b_i is the body force.

2.2 Turbulence Modelling

Flow past a helicopter configuration can be comprised of rapid velocity and pressure fluctuations known as turbulent flows. To be able to get an accurate representation of the turbulent flows mathematical models have been developed for CFD.

Turbulence models are mathematical models that are used to solve the unknown variables in the transformed Navier-Stokes equations that represent the velocity and pressure fluctuations. These models have been developed because the size and fluctuations of turbulence flows are unpredictable and therefore too computationally expensive to simulate directly in practical engineering calculations.

Two approaches can be used to transform the Navier-Stokes equations in such a way that that the turbulence fluctuations do not have directly simulated. These are the Reynolds-averaging and filtering (LES) approaches. Both of these approaches introduce additional terms into the Navier-Stokes equations which need to be solved by turbulence models. However, the Reynolds averaged approach is adopted by the more commonly used turbulence models since it is more applicable practical engineering applications, and is computationally cheaper than the lesser used LES approach [3].

2.2.1 Reynolds-Averaging

In Reynolds-averaging, the solution variables in the Navier-Stokes equations are decomposed into the mean and fluctuating component. Therefore the velocity, pressure and other scalar quantities are represented as follows:

$$\phi_i = \overline{\phi}_i + \phi_i' \tag{2.4}$$

where $\overline{\phi}_i$ represents the average(mean) component
ϕ_i' represents the fluctuation component.

The mean component is given by the following time average:

$$\overline{\phi}_i = \frac{1}{t}\int_{t_1}^{t_2} \phi_i dt \tag{2.5}$$

Substituting expressions of this form for the flow variables into the continuity and Navier-Stokes equations and taking a time averaged yields the following modified set of governing equations known as the Reynolds-averaged equations:

Reynolds averaged continuity equation

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0 \tag{2.6}$$

Reynolds Averaged Navier-Stokes Equation (RANS)

$$\frac{\partial}{\partial t}(\rho \overline{u}_i) + \frac{\partial}{\partial x_j}(\rho \overline{u}_i \overline{u}_j) - \rho b_i - \frac{\partial p}{\partial x_i} + \mu\left(\frac{\partial}{\partial x_j}\left[\frac{\partial \overline{u}_i}{\partial x_j}\right) + \frac{\partial \overline{u}_j}{\partial x_i}\right] - \frac{2}{3}\frac{\partial \overline{u}_k}{\partial x_k}\right) + S_i - \rho\overline{u_i' u_j'} \tag{2.7}$$

Additional terms, represented here by $\rho\overline{u_i' u_j'}$ which appear on the right-hand side of equation 2.7, now appear in the Navier-Stokes equation. These terms are known as the Reynolds- Stresses, and need to be modelled to achieve closure for the RANS equations. This is done through the use of various turbulence models [3].

2.2.2 Boussinesq Approach vs. Reynolds-Stress Transport Models

The Spalart-Allmaras, $k-\epsilon$ and $k-\omega$ turbulence models use the Boussinesq hypothesis to model the Reynolds0stresses. This method relates the Reynolds-Stresses to mean velocity gradients:

$$-\rho\overline{u_i' u_j'} = \mu_t\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}(\rho k + \mu_t \frac{\partial u_i}{\partial x_i})\delta_{ij} \tag{2.8}$$

The advantage of using this approach is the relatively low computational cost associated with the computation of the turbulent viscosity, μ_t. In the case of the Spalart-Allmaras model, only one additional transport equation, representing turbulent viscosity, is solved. For the $k-\epsilon$ and $k-\omega$ models, two additional transport equations are solved. These are namely for the turbulence kinetic energy, k and either the turbulence dissipation rate,ϵ or specific dissipation rate ω and μ_t are computed as a function of k and ϵ.

The alternative approach, embodied in the Reynolds-Stress Model, is to solve transport equations for each of the terms in the Reynolds- stress tensor. An additional scale-determining equation is also required. This means that five additional transport equations are required in 2D flows and seven additional equations must be solved in 3D.

Even though the Boussinesq hypothesis has the disadvantage of assuming μ_t is an isotropic scalar quantity, which is not strictly true, the models based on this approach perform very well. In most cases the additional computational expense of the Reynolds stresses model is not justified [3].

2.2.3 Wall bounded turbulent flows

Turbulent flows are significantly affected by the presence of walls, as the walls are the main source of mean vorticity and turbulence. Successful prediction of these wall bounded turbulent flows in the boundary layer, surrounding the helicopter, is thus a very important component of near wall modelling, because of the effect that it has on the results of numerical simulations. After all, it is in the near wall region that solution variables have large gradients, and the momentum and other scalar transports occur the most.

Previous experiments have shown that the near wall region is divided into three regions. These are the innermost layer, generally known as the viscous sublayer flow, which shows characteristics of laminar flow; the outer layer, known as the fully turbulent layer, where the flow is mostly turbulent; and the interim layer in which both types of flow effects are equally important.

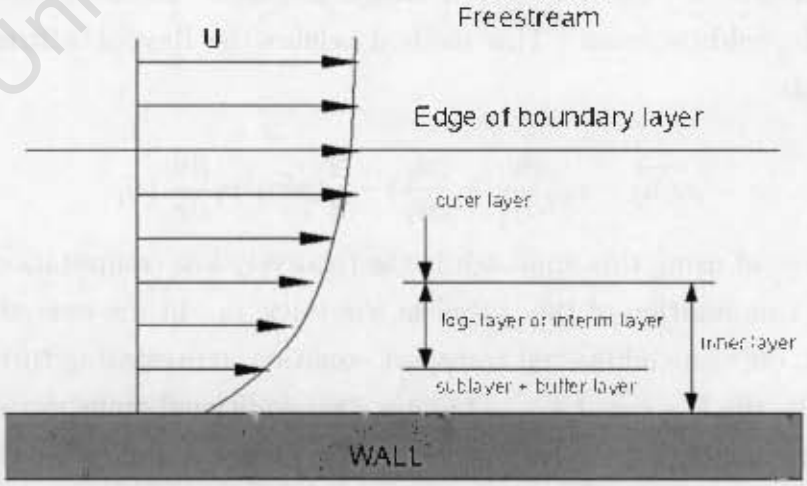

Figure 2.1: The three layers found in the near wall region [3]

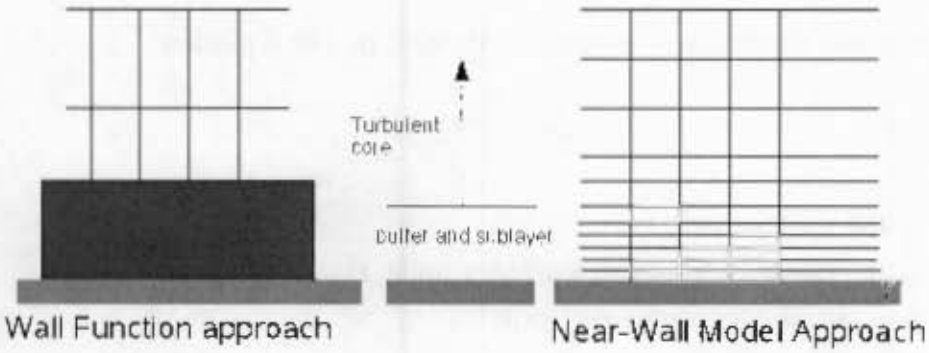

Figure 2.2: The two approaches used for near wall treatments [3]

Two approaches exist for the modelling of the near wall regions. In the first approach a semi-empirical formula called " a wall function" is used to bridge the gap between the wall and the fully turbulent region as the viscous layer is not resolved. The use of the wall functions removes the need to modify the turbulence models to account for the presence of the wall. In a second approach, which is known as the near wall modelling approach or enhanced wall treatment, the turbulence models are modified to account for the viscous layer by resolving the entire near wall regions above the wall with a mesh [3].

2.2.4 Grid considerations for turbulent flows

In the pre-processor phase the domain, over which the flow is being modelled, is created and boundary conditions are defined. The domain is also split up into many finite volumes, or cells, by a procedure known as meshing or grid creation. This is done so that the various unknown variables such as velocity, pressure and temperature can be calculated at the nodes situated inside the volumes [7].

In turbulent flow simulations strong interactions exist between the the mean flow and the turbulence. Therefore, the numerical results for turbulent flows tend to be more susceptible to grid dependency than those for laminar flows. Since most of the turbulence is generated in the boundary layer close to the helicopter, sufficiently fine meshes have to implemented in this region to resolve the rapid flow property changes that occur.

The near wall mesh can be checked by examining the wall y plus (y^+) values close to the helicopter fuselage.

Wall y^- is a non dimensional parameter defined by the equation :

$$y^+ = \frac{\rho u_t y_p}{\mu} \tag{2.9}$$

where ρ is the fluid density
y_p is the distance from the wall to a point P located in the first set of cells above the wall
u_t is the friction velocity
μ is the fluid viscosity at point P.

In the viscous sublayer of the near wall region the velocity varies linearly with the wall y^+ whilst, across the inner layer the log law is used in which the mean velocity is proportional to the logarithm of the wall y^-. These relationships provide the criteria for the selection of the near wall or wall function modelling.

If the wall y^+ value is in the region of 30-60, which is effectively a coarse mesh region in which the log law is valid, the wall function approach needs to be selected. Although the linear law is implemented when wall $y^+ < 11.225$, using a sufficiently fine mesh near the walls should be avoided, because the wall functions cease to be valid in the viscous layer. The upper bound depends on, amongst others, pressure gradients and Reynolds number. As the Reynolds number increases, the upper bounds tends to also increase. Wall y^+ values that are too large are not desirable, because the wake component becomes substantially large above the log-layer. Also a value close to the lower bound, of 30, is most desirable.

On the other hand the near wall model approach needs to be implemented when the mesh is very fine and the y^+ is equivalent or near to one. However, a higher wall y^+ of less than four is acceptable as long as it is well inside the viscous sublayer. At least 10 cells have to exist within the viscosity-affected near wall region (Reynolds number of less than 200) to be able to resolve the mean velocity and turbulent quantities in that region [3].

2.2.5 The Spalart-Allmaras model

The Spalart-Allmaras turbulence model was designed specifically for aerospace applications involving wall bounded flows, and has been shown to give good results for boundary layers subjected to adverse pressure gradients.

Originally this model was designed to be a low-Reynolds-number model, which required the viscous affected region of the boundary layer to properly resolved. However,

in the commercially available CFD package, FLUENT® the Spalart-Allmaras model has been modified to use wall functions when the mesh resolution is not sufficiently fine.

One equation models such as the Spalart-Allmaras model are said to be at a disadvantage in that they are unable to rapidly accommodate changes in length scale. Such might be necessary when the flow changes abruptly from a wall bounded to a free shear flow [3].

2.2.6 The $k-\epsilon$ model

The standard $k-\epsilon$ model has become the workhorse of practical engineering flow calculations since it was proposed by Launder and Spalding [11]. It has been shown to have reasonable accuracy as well as being robust for a wide range of applications. However, it performs poorly in swirling flows with streamline curvature, vortices, and rotation. Therefore, improvements have been made made to the model. Subsequently two k ϵ the variants, namely the RNG $k-\epsilon$ model and the realizable $k-\epsilon$ model have been created.

The RNG k ϵ was derived using the a rigorous statistical technique known as the Renormization group theory (RNG). This model is very similar to the standard model but includes refinements. These refinements include an additional term in its ϵ equation that improves the accuracy for rapidly strained flows. The effects of swirl on turbulence is accounted for which also enhances the accuracy for swirling flows, and an analytically derived differential formula that accounts for low Reynolds number effects is also provided.

On the other hand, the realizable model is a recent development and differs from the standard model by containing a new formulation for the turbulent viscosity and having a new transport equation for the dissipation rate, ϵ.

An advantage of the realizable model is that more accurately predicts the spreading rate of both planar and round jets. Thus it is likely to provide superior performance for flows involving rotation, boundary layers under strong adverse pressure gradients, separation, and recirculation. However, since this model is relatively new, it is still not clear in exactly which instances it out-performs both the standard and RNG models [3].

2.2.7 The $k-\omega$ model

The standard $k-\omega$ model in the CFD package FLUENT® is based on the Wilcox $k-\omega$ model [21]. This model incorporates modifications for low-Reynolds-number effects, compressibility, and shear flow spreading. Therefore, it predicts free shear shear flow spreading rates that are in close agreement with measurements for wakes, mixing layers, and plane, round, and radial jets. It is therefore very applicable to wall-bounded flows and free shear flows.

Another variation of the standard $k-\omega$ model is the shear stress transport (SST) model. The SST model was developed by Menter [12] to blend the robust and accurate formulation of the $k-\omega$ model in the near wall region with the free stream independence of the $k-\epsilon$ model in the far field.

The SST $k-\omega$ model differs from the standard $k-\omega$ model in that it activates the standard $k-\omega$ model in the near wall region and the transformed $k-\epsilon$ model with the use of a blending function. It also found that the damped cross-diffusion derivative term in the ω is incorporated and the definition of the definition of the turbulent viscosity is modified to account for the transport of the turbulent shear stress. Finally the modelling constants are also different [3].

2.2.8 The Reynolds Stress Model(RSM)

The Reynold stress model is the most elaborate turbulence model that the CFD package, FLUENT®, provides. It closes the Reynolds-averaged Navier-Stokes equations by solving transport equations for the Reynolds stresses, together with an equation for the dissipation rate. This means that five additional equations are required for two dimensional (2D) flows and seven additional transport equations must be solved in three dimensional (3D) flows.

The exact form of the Reynolds stress transport equations may be derived by taking moments of the exact momentum equation. This a process wherein the exact momentum equations are multiplied by a fluctuating property, the product then being Reynolds averaged. Unfortunately, several of the terms in the exact equation are unknown and modelling assumptions are required to close the equations.

Since the RSM model accounts for the effects of streamline curvature, swirl, rotation, and rapid changes in strain rate in a more rigorous manner than the one-equation and two-equation models, it has greater potential to give accurate predictions for complex flows. However, the fidelity of the RSM predictions is still limited by the closure as-

sumptions employed to model various terms in the exact transports equations for the Reynolds stresses. The modelling of the pressure-strain and dissipation-rates is particularly challenging, and often considered to be responsible for compromising the accuracy of RSM predictions.

The RSM model,applied to various classes of flows, might not always yield results that are clearly superior to the simpler models. Therefore its results may not warrant the additional computational expense that its use requires. However, use of the RSM is a necessity when the flow features of interest are the result of anisotropy in the Reynold stresses [3].

2.3 Helicopter Aerodynamics

Most helicopter designs that exist today have the same basic layout. The layout that exists is commonly made up of a main rotor that provides lift and forward thrust, and a tail rotor that is used to counter main rotor torque, as well as for directional control. It should be noted that other concepts have been introduced to either replace or assist the tail rotor in cancelling the main rotor torque.

Helicopters are known for their ability to hover, and are also capable of both vertical and forward flight. The two basic methods for analysing the characteristics of a rotor, in hover, vertical flight and forward flight, are the momentum or energy methods and the blade element methods. The blade element method is necessary for accurate performance estimation and for establishing the limits of a rotor performance, but the momentum method provides a rapid means of obtaining a first estimate of the performance as well as valuable insight into the physics of the system [16].

The main area of concern in this report is the helicopter in forward flight. Therefore, some of the aerodynamic aspects of the helicopter in forward flight, that were analysed by the blade element method are introduced.

2.3.1 Tangential Velocity

In forward flight, the velocity acting on the blade element is a function of the both the radial station and the blade azimuthal position. The azimuthal angle, ψ, is defined as shown in Figure 2.3 with $\psi = 0$ over the tail. The velocity acting on the blade element is the vector sum of the velocity due to rotation, Ωr, and the forward speed of the helicopter, V. The velocity perpendicular to the leading edge or tangential to the chord

of the element is U_T, and is defined as :

$$U_T = \Omega r + V sin\psi \tag{2.10}$$

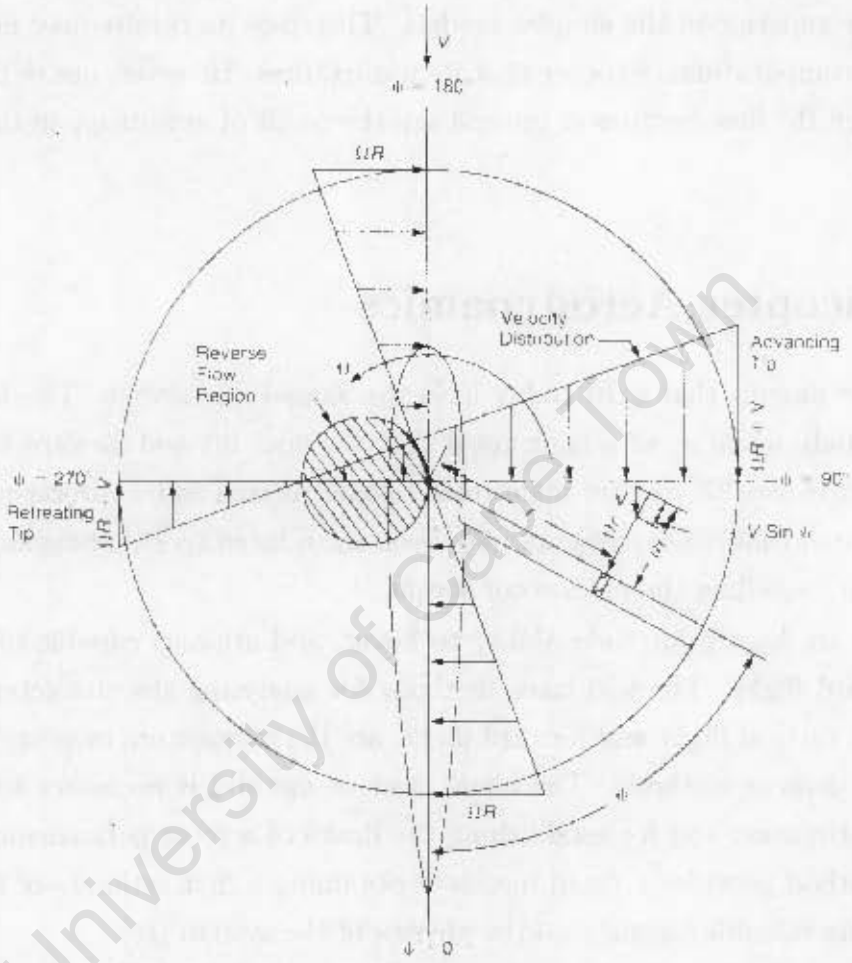

Figure 2.3: An overhead view of the tangential velocities of a helicopter body and rotor in forward flight [16]

Over the tail and nose, the blade element sees the same velocity as it would in hover, but on the advancing blade it sees a higher velocity, and on the retreating blade a lower velocity. It could even be found that there are elements where the velocity on the leading edge is actually negative. Thus the air strikes the trailing edge rather than the leading edge of the blade. The region in which the velocity perpendicular to the leading edge is actually negative is shown to be circular and the zone is called the reverse flow region.

If each blade had the same pitch setting, their angles of attack would be nearly the

same, but the difference in velocity would produce more lift on the advancing side than the retreating side. This would produce a rolling moment, which would be expected to roll the helicopter over [16].

2.3.2 Blade Flapping

Rotor blades on helicopters flap in response to centrifugal and aerodynamic forces they endure. This phenomenon is also dependent on the elastic and inertial characteristics of the blades. When the blades flap, the advancing blade, which initially had high lift, accelerates upward. As it accelerates upward, it also rotates toward the nose, where the local velocity is reduced to its mean value, so that no unbalanced lift exists and the blade stops accelerating. The retreating blade undergoes a similar experience except that it accelerates downwards as it rotates toward to a position over the tail. The angle of attack on the advancing blade is decreased while the angle of attack on the retreating blade is increased. This causes the rotor to reach flapping equilibrium when the local changes in the angle of attack are sufficient to compensate for the changes in dynamic pressure. Thus in this equilibrium condition, the rotor is not tilted sideways but is tilted fore and aft.

A small amount of lateral flapping will be generated due to another aerodynamic effect. As the rotor, produces lift, it is coned by the combination of lift and centrifugal forces. Coning of a rotor occurs when a blade, whether hinged or cantilevered from a hub, seeks an equilibrium coning angle that is a function of lift, centrifugal forces, and blade weight. The magnitude of coning is generally found by setting the moments at the hinge to zero.

In forward flight the lateral flapping occurs when the blade over the nose experiences air coming toward its lower surface, whilst the blade over the tail experiences air approaching it from the top. The result is that the angle of attack on the blade at $\psi = 0°$ is decreased whilst the angle of attack at $\psi = 180°$ is increased. The rotor compensates for this inequality 90° later by tilting up on the retreating side and down on the advancing side. This causes flapping velocities over the nose and over the tail that are exactly enough to compensate for the difference in angle of attack caused by coning.

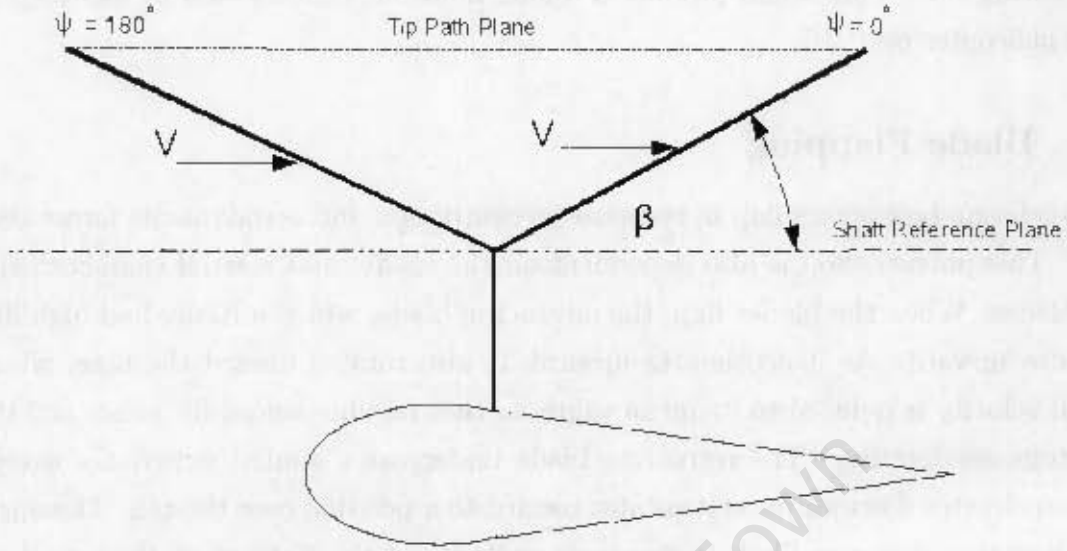

Figure 2.4: Side view of a helicopter and rotor which shows the velocity orientation causing lateral rotor tilt [16]

The flapping motion may be mathematically represented by a infinite Fourier series. However, only the first three terms need be considered since the second and higher harmonics represented by the remaining terms are relatively small and have very little effect on rotor thrust and torque. This Fourier equation is displayed below:

$$\beta = a_0 - a_{1_s} cos\psi - b_{1_s} sin\psi \tag{2.11}$$

where a_0 represents coning

a_{1_s} is the longitudinal flapping with respect to a plane perpendicular to the shaft defined as positive when the blade flaps down the tail and up at the nose

b_{1_s} is the lateral flapping defined as positive when the blade flaps down the advancing and up on the retreating side [16].

2.3.3 Cyclic Pitch

In the "early days" the pitch and roll control were generated by tilting the rotor and creating a thrust vector that had a moment arm with respect to the position of the center of gravity. However, as the size of the aircrafts became larger, the forces required to tilt

the rotor became so high that flight became difficult. At this point, a means of rotor control called cyclic pitch was developed. In this system, which is almost universally used at present, the pilot cyclically changes the pitch of the blades about the feathering bearings by a tilting mechanism known as a swashplate.

The cyclic pitch can be used for two purposes, namely to trim the tip path plane with respect to the mast and to produce control movements for maneuvering. In the first case, the pilot can mechanically change the angle of attack of the blades by the same amount as the flapping motion would have, thereby eliminating flapping. This can be used to diminish the entire effect of flapping or to leave just enough to balance pitching or rolling moments on the aircraft. In the second case, the pilot deliberately introduces an unbalanced lift distribution in order to make the rotor tilt for maneuvering. Whether being used for trim or for control, the cyclic pitch is equivalent to flapping in that the changes to rotor conditions due to one degree of cyclic pitch are the same as those due to one-degree change in flapping. Like the flapping, the blade pitch can be written in terms of a Fourier series:

$$\theta = \theta_0 - A_1 cos\psi - B_1 sin\psi + \frac{r}{R}\theta_1 \tag{2.12}$$

where θ_0 is the collective pitch that is required to produce enough rotor thrust to balance the weight and to compensate for the inflow

θ_1 is the blade twist

A_1 is the lateral cyclic pitch used to produce the rolling moment

B_1 is called the longitudinal cyclic pitch because it is used to produce pitching moments

r is the element radial position

R is the rotor tip radius.

It should be noted the use of cyclic pitch for trim makes it possible to eliminate the flapping hinges in the so-called rigid rotor designs. The rigid rotor designs have eliminated the flapping hinges which exist on the fully articulated rotor [16].

2.4 Trim Routine Implementations

2.4.1 Chaffin and Berrys Investigation of Helicopters Fuselage Aerodynamics

Chaffin and Berry [4] modified an incompressible Navier-Stokes code to model the effects of a helicopter in forward flight on the viscous aerodynamics of the fuselage. Interual boundary conditions were used to represent a rotor in time-averaged manner, in the form of an actuator disk with a disk boundary of zero thickness. Overset grids were used to allow more complex configurations to be modelled. Therefore, the rotor and fuselage geometry could be generated independently.

The effect of the rotor was imposed on the flow solution using boundary conditions on the rotor-surface, describing the difference in pressure and tangential velocities between the upper and lower regions. The thrust was modelled as a jump in pressure across the disk whilst swirl velocities were implemented by adding a jump in tangential velocity across the disks surface.

Unlike an actuator disk, the pressure jump was allowed to vary with radial and azimuthal locations on the disk and was computed from a fully coupled blade element. The compressibility effects were included implicitly when determining the forces generated on the blade.

This model included a trimming algorithm, matching the total rotor thrust to a prescribed value, and allowing the moments on the rotor to be balanced by adjusting the collective and cyclic pitch angles in an iterative manner during the solution process.

Calculations were made for an isolated rotor as well as for a rotor/fuselage geometry and compared with experimental inflow velocity data and experimental fuselage pressure data. The results were shown to be reasonably good. It was found that in the hub and nacelle region, the method had problems modelling the interaction of the separated region and the rotor wake. This was most likely because of the time averaged modelling used by the method.

2.4.2 Yang's hybrid method for rotors

Yang et al. [22] developed a hybrid Navier Stokes full potential solver for the efficient prediction of 3D unsteady viscous flow phenomena that occur over a helicopter rotors in forward flight.The method combined a Navier-Stokes analysis near the blade modelling the viscous flow and near wake, with a potential flow analysis in the far-field modelling

inviscid isentropic flow. A grid motion module was also developed to account for the blade motion, and elastic deformations. This hybrid analysis was validated through study of rotors in forward flight as the computed pressure coefficients were in good agreement with the experimental values.

The reason for the development of this model was because the authors felt the prediction capability and the solution efficiency of the current generation of CFD analysis was not practical for the helicopter industry. It was felt that the usefulness was limited because the rotor was not being trimmed, the tip vortex capturing suffers excessive numerical diffusion, and because the blade dynamics and aeroelasticity were not being adequately modelled.

Therefore, to ensure that accurate aerodynamic predictions could be possible, a Newton-Raphson Iterative method was used for the trimming procedure. This method was chosen since the relationship between the rotor and aerodynamic parameters; coefficient of thrust, coefficient of rolling moment, coefficient of pitching moment; and blade pitch angle is non linear.

Applying this procedure to the hybrid code was too expensive to be practical, thus it was implemented to form an out-most trimming loop in the full potential code. However, at times the rotor moments were approximately balanced in roll, but were out of balance in pitch. Thus, to eliminate the rolling and pitching moment, the lateral and longitudinal cyclic pitch angles were also trimmed manually.

2.4.3 FLUENTs' Virtual Blade Model(VBM)

The method that was used in the FLUENT® VBM model [18] to analyse the mutual aerodynamic interaction between multiple rotors and airframes, was created in the spirit of Zori et al. [23] and Yang et al. [22] and implemented into the general CFD package known as FLUENT®.

This technique modelled the rotors implicitly through source terms in the momentum equations. This allowed the effects of the blade to be accounted for without them being present. Unstructured grids were used in the rotor disks which allowed easy meshing of multiple rotor geometries in close proximity and convenient local mesh clustering. The non-linear, aerodynamic interaction between the rotor wakes with each other, and with structural components was solved by coupling the VBM with the governing flow field equations computed by the FLUENT® Navier-Stokes solvers.

It was also realised that accurate aerodynamic predictions are possible only if the rotors operate at desired thrust and zero roll and pitch moments about the hub. Thus

an automatic and robust trim routine was implemented to ensure that, at a particular flight speed, the model can calculate the correct collective pitch angle and cyclic pitch coefficients. Following Yang et al. [22] the iterative method used, was the Newton Raphson iterative method, and the purpose of the routine was to achieve the desired thrust coefficients and eliminate the roll and pitch moments about the hub. This model did not however calculate the flapping motion of the rotor but did include tip effects.

As a validation example, a well-studied single rotor airframe interaction case in the forward flight was examined and the pressure distributions on the airframe were found to compare well with experimental data. Results of two proof of concept examples were also carried out on an Apache-64 helicopter simulation considering both main and tail rotor, and the V-22 Osprey demonstrating the combination of the model with a dynamic mesh capability. The presented steady results for these examples were found to be credible.

Chapter 3

Model used in the investigation

Helicopters are made up of a number of components that are complex in shape. One of these components is the helicopter fuselage. Thus, before experimental flow conditions of a helicopter fuselage flow-field in a wind tunnel can be simulated, and resolved with the use of CFD, a fuselage model has to be created. This chapter describes the fuselage and external geometry created for this investigation. The method in which the geometry was created is also depicted.

3.1 ROBIN Geometry

The fuselage geometry created was that of a ROBIN (ROtor Body INteraction) configuration. The ROBIN configuration is representative of a generic helicopter. This configuration was chosen since comparative wind tunnel experimental data was found to be readily available as it was commonly used in several previous wind tunnel investigations.

As depicted in Figure 3.1 the ROBIN shape consists of an analytically defined body representing the fuselage, and an analytically defined pylon representing the fairing around the engines and transmission.

Coordinates of the ROBIN body are defined by super-ellipse equations [14]. For a given non-dimensional body longitudinal station (x/l), the non dimensional coordinates of the cross section (y/l and z/l) are obtained from the analytical functions of the model height (H), width (W), camber (Z_0), and elliptical power (N).

Each function has the same form, only the eight coefficients (C_1 to C_8) differ. The body is divided into four separate regions and the pylon is divided into two regions. Separate coefficients are used in the six regions for the four functions. The form of these

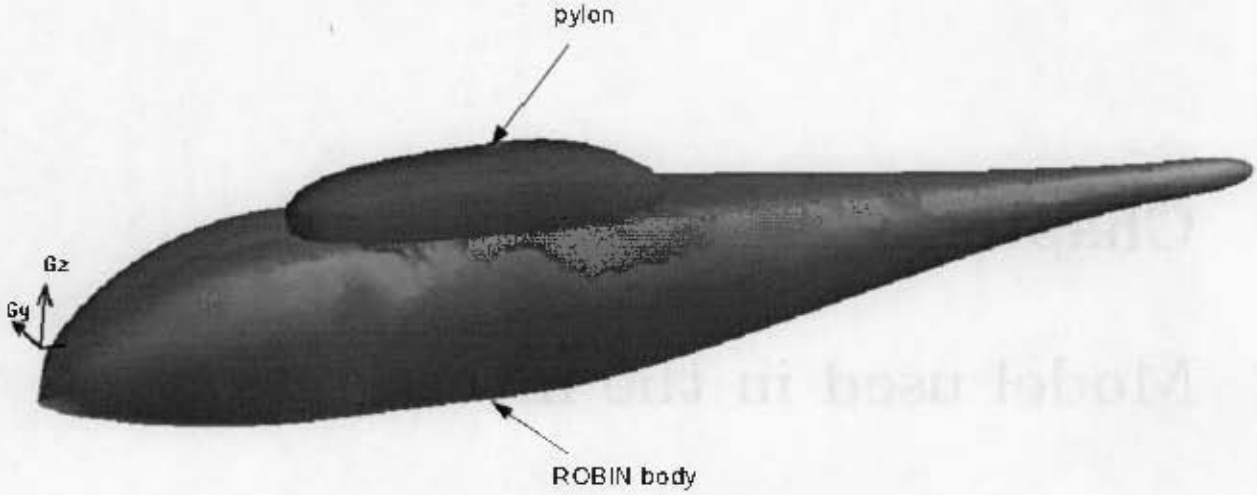

Figure 3.1: A depiction of the analytically defined ROBIN shape

functions are defined as follows [14]:

$$\begin{bmatrix} H(x/l) \\ W(x/l) \\ Z_0(x/l) \\ N(x/l) \end{bmatrix} = C_6 + C_7 \left(C_1 + C_2 \left(\frac{x/l + C_3}{C_4} \right)^{C_5} \right)^{C_8} \tag{3.1}$$

The coordinates of a given body station, x/l are defined using polar coordinates. The non dimensional radial coordinate for the cross section is defined as follows [14]:

$$r = \left(\frac{\left(\frac{H}{2}\frac{W}{2}\right)^N}{\left(\frac{H}{2}sin\varphi\right)^N + \left(\frac{W}{2}cos\varphi\right)^N} \right)^{1/N} \tag{3.2}$$

From the radial coordinate, the non dimensional coordinates on the cross section can be obtained form the following by varying φ from 0 to 2π

$$y/l = rsin\varphi \tag{3.3}$$

$$z/l = rcos\varphi \tag{3.4}$$

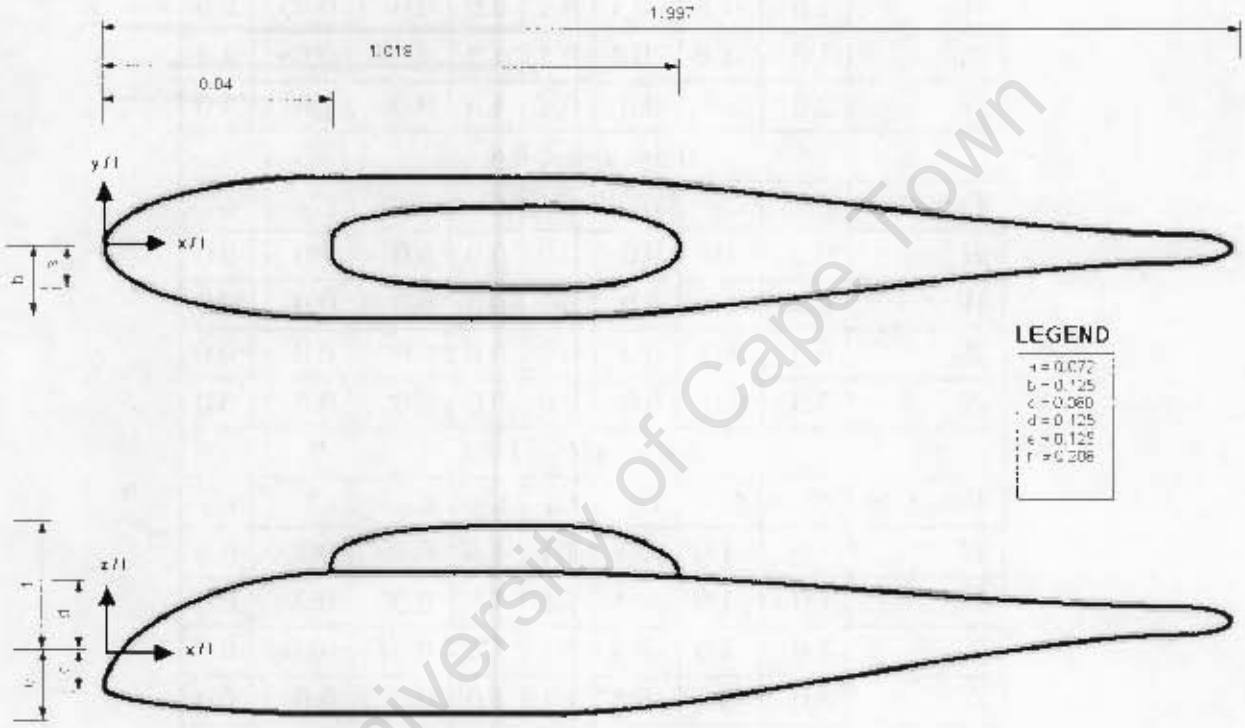

Figure 3.2: Sketch of the robin configuration indicating the non-dimensional coordinates

The values for the coefficients are listed in Tables 3.1 and 3.2:

$0.0 < x/l < 0.4$								
Function	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
H	1.0	-1.0	-0.4	0.4	1.8	0.0	0.25	1.8
W	1.0	-1.0	-0.4	0.4	2.0	0.0	0.25	2.0
Z_0	1.0	-1.0	-0.4	0.4	1.8	-0.08	0.08	1.8
N	2.0	3.0	0.0	0.4	1.0	0.0	1.0	1.0
$0.4 < x/l < 0.8$								
Function	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
H	0.25	0.0	0.0	0.0	0.0	0.0	0.0	0.0
W	0.25	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Z_0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$0.8 < x/l < 1.9$								
Function	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
H	1.0	-1.0	-0.8	1.1	1.5	0.05	0.2	0.6
W	1.0	-1.0	-0.8	1.1	1.5	0.05	0.2	0.6
Z_0	1.0	-1.0	-0.8	1.1	1.5	0.04	-0.04	0.6
N	5.0	-3.0	-0.8	1.1	1.0	0.0	0.0	0.0
$1.9 < x/l < 2.0$								
Function	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
H	1.0	-1.0	-1.9	0.1	2.0	0.0	0.05	2.0
W	1.0	-1.0	-1.9	0.1	2.0	0.0	0.05	2.0
Z_0	0.04	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 3.1: Coefficients to define body shape

$0.0 < x/l < 0.4$								
Function	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
H	1.0	1.0	-0.8	0.4	3.0	0.0	0.145	3.0
W	1.0	-1.0	-0.8	0.4	3.0	0.0	0.166	3.0
Z_0	0.125	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$0.4 < x/l < 0.8$								
Function	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
H	1.0	1.0	0.8	0.218	2.0	0.0	0.145	2.0
W	1.0	-1.0	-0.8	0.218	2.0	0.0	0.166	2.0
Z_0	1.0	-1.0	-0.8	1.1	1.5	0.065	0.06	0.6
N	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 3.2: Coefficients to define pylon shape

3.2 Creation of the model

The pre-processor that was used in this investigation was GAMBIT®. However, due the complexity of the ROBIN configuration and the lack of dimensions provided, coordinates had to be generated. These coordinates were created with the use of the mathematical software, MATLAB®. MATLAB® is a language for technical computing in which computation, visualization, and programming is carried out in a familiar mathematical notation.

A code was written in MATLAB®, to create the necessary coordinates needed to generate the ROBIN geometry, from the equations stated in the previous section (Refer to appendix A for the code and outputs). The coordinates were then exported to a text file (.IBL), so that it could be read in by the CAD package ProEngineer. ProEngineer is widely used CAD package that allows the user to generate complex geometries in the digital format required, and was used to create the ROBIN geometry because of its complexity.With the aid of the coordinates, a surface model of the ROBIN configuration was generated in ProEngineer, and exported to the pre-processor so that that it could be meshed. However, a problem was encountered since the transfer between the two CAD packages was not a smooth one. This was mainly due to the difference in tolerances. Ordinarily CAD systems use a loose (10^{-3}) tolerance, since it is usually good enough for their primary purpose and improves speed and memory requirements. Gambit® on the

other hand uses tolerance of 10^{-6}, since it needs precise accuracy for Boolean operations and splits. The difference can result in a gap between adjacent entities or between the boundary curve and surface data. This inadequacy was eventually overcome by scaling the model, and with the use of the healing option that is available in GAMBIT®.

3.3 ROBIN 'fuselage-only' simulations

In 1979, Freeman and Mineck [6] conducted extensive wind tunnel investigations, in NASA Langley's vertical or short takeoff and landing (V/STOL) closed return atmospheric tunnel, for the ROBIN configuration. This wind tunnel is also known as the 14 by 22-Foot Subsonic tunnel, and was designed for low speed testing of powered and high-lift configurations. The test section measures 6.63 meters wide and 4.42 meters high at the entrance and is 15.24 meters long.

In the investigation by Freeman and Mineck[6] time-averaged fuselage surface pressures of the ROBIN geometry with a 3.15-meter, four bladed articulated rotor was measured. This pressure data was gathered from pressure taps located along the fuselage of the ROBIN geometry at a Mach number of 0.062 and an effective Reynolds number of 4.46 x 10^6 with a three meter model. It should be noted that no pressure taps were located on the hub pylon cover.

Pressure data was also collected without a rotor system. Since experimental time-averaged pressure data was available for the ROBIN fuselage without a rotor, it was thought that initiating the computational investigation for a stand alone fuselage would be beneficial. The reason that this was thought to be beneficial was because it would be great preparation for simulations conducted with the rotor since certain aspects such as grid densities, turbulence models, boundary conditions could be compared.

For that reason, the complete model for the ROBIN 'fuselage-only' simulations was based on Freeman and Mineck [6] experimental investigation. A three meter fuselage was modelled at an angle of attack of zero degrees with zero yaw. The simulations were conducted at a Mach number of 0.062 and an effective Reynolds number of 4.46 x 10^6. Due to the symmetry, and time that could be saved computationally, only one half of the flow domain used by Freeman and Mineck [6] was simulated.

3.3.1 Boundary Conditions

Many boundary conditions are offered in the CFD pre-processor, GAMBIT®. In view of the fact that the experimental data from Freeman and Mineck [6] was to be used as a comparison for the body alone model the various boundary conditions, used experimentally, were assigned in the pre-processor. As seen by Figure 3.3 below the inlet boundary of the wind tunnel was specified to be a constant velocity inlet so that the experimental, air speed of 21.72 m/s could be enforced. The outlet boundary was specified as constant pressure boundary. Wall boundary conditions were applied to the ROBIN configuration to ensure that the external flow conditions just above could be captured. A symmetry boundary was assigned to the side domain boundary shown in Figure 3.3. Symmetry boundary conditions are used when the geometry of interest and the expected pattern of the flow have mirror symmetry. Free slip walls were also applied to the top, bottom and remaining side domain boundary.

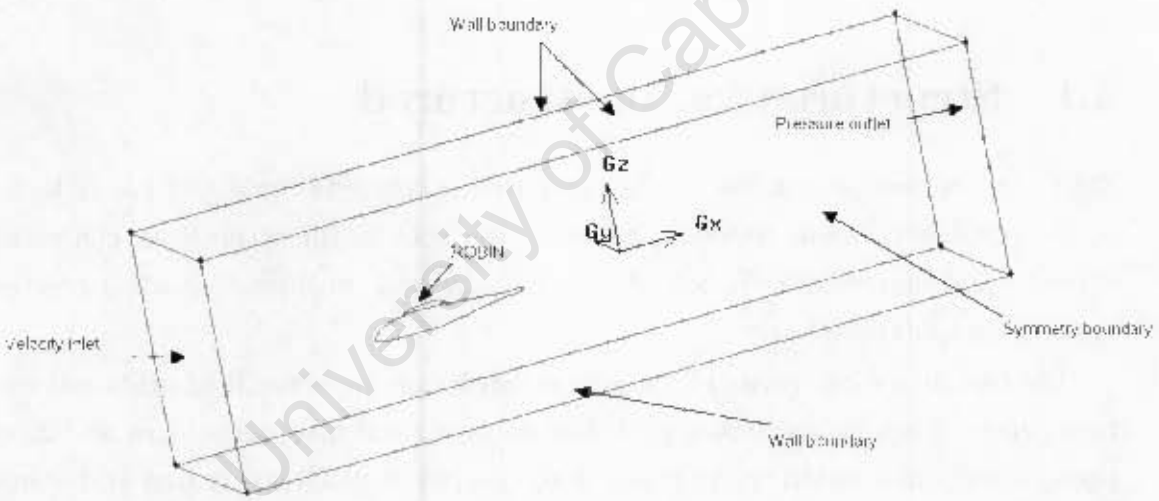

Figure 3.3: Geometry boundary conditions for the model that was used for the 'fuselage-only model' simulations

Chapter 4

Computational grid

The creation of the grid on a helicopter fuselage configuration is a fairly complex and lengthy procedure. This is because of flow variations which have to be taken into consideration, and the complexity of the model used. The structure of the grid used and explanations for the choices made, are therefore discussed in the following paragraphs.

4.1 Structured vs. unstructured

Before any numerical solution can be computed, a computational grid has to be created on the physical domain. However, many factors such as the setup time, computational expense, and numerical diffusion have to be take into consideration when creating the optimal computational grid.

The two major categories of the grid construction are structured grids and unstructured grids. Each type of these grids has its own particular advantages and disadvantages. Ideally one would try to create a structured or block-structured grid, consisting of quadrilateral or hexahedral elements, since the numerical results are said to be significantly better. However, it has to be noted that this would only be true for simple flows.

The main reason for the better results is due to the fact that numerical diffusion is minimised when the flow is aligned with the mesh. The numerical diffusion phenomenon is a 'non real' aspect which has the reputation of affecting the accuracy of solutions in a CFD model. All practical numerical schemes for solving fluid flow contains a finite amount of numerical diffusion. This is because numerical diffusion arises from truncation errors that are a consequence of representing the fluid flow equations in discrete form [7]. The amount of numerical diffusion is inversely related to the resolution of the mesh.

Therefore, one way of dealing with numerical diffusion is to refine the mesh .

Often the use of structured grids is very time consuming or near impossible. This is often the case when the geometry is too complex. An unstructured grid that employs the triangular or tetrahedral cells can be used as an alternative. Also when the geometries are complex or the range of length scales of the flow is large, an unstructured mesh can be created with far fewer cells than the equivalent structured mesh. This is because a triangular or tetrahedral mesh allows clustering of cells in selected regions of the flow domain. Structured quadrilateral or hexahedral meshes will generally force cells to be placed in regions where they are not needed [1].

4.2 TGrid Unstructured grid generator

The unstructured volume grid created around the robin fuselage was generated using the additional pre-processor TGrid®.

TGrid® is a highly efficient, easy-to-use, unstructured grid generation program that can handle grids of virtually unlimited size and complexity, consisting of triangular, tetrahedral, hexahedral, prismatic, or pyramidal cells. TGrid is an intermediate component of the FLUENT® package, which also consists of the solver FLUENT® and the preprocessor GAMBIT®. It can generate volume meshes from the existing boundary meshes that are generally created in the initial pre-processor GAMBIT®.

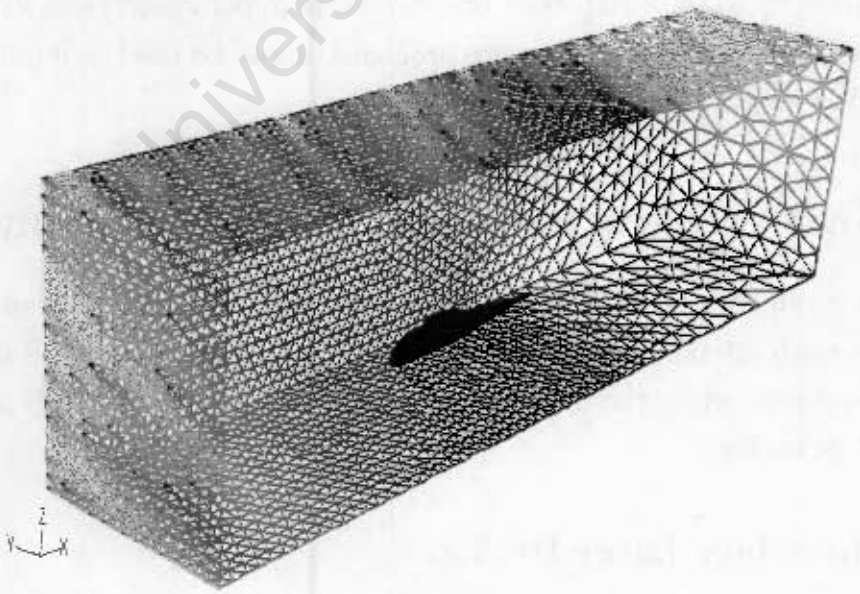

Figure 4.1: A depiction of the boundary mesh created in GAMBIT

Tgrid® was created to reduce the time and labour taken to create a mesh on complex geometries. The unstructured grid generation techniques of Tgrid®, couple basic geometric building blocks with extensive geometric data to highly automate the grid generation process. In addition, the generalised data structures employed in these schemes permit the addition and removal of cells to maximize accuracy and minimize memory and CPU requirements .

The general procedure to create a grid involves the reading of the boundary mesh into TGrid®. This boundary mesh is generally created with the pre-processor, GAMBIT®. Examination of the boundary mesh for topological problems such as free edges and duplicate nodes are then carried out. Once the boundary is topologically correct, a 3D surface mesh can be checked for poor face quality. Many quality-related problems can be solved easily with edge swapping, but more difficult problems may require direct manipulation of the faces and nodes

Generating the volume mesh is the next step. This can be automatically done or by proceeding through a series of steps. For hybrid grids, prisms or pyramids are firstly generated. This is followed by generating the triangular or tetrahedral volume cells. For grids containing only triangles or tetrahedral cells, either the automatic mesh generation procedure can be used, or each step can be performed manually. Any problems that exit on the volume mesh can then be checked. The presence of degenerate cells will prevent one from obtaining a solution, and very poor cells in critical areas will cause serious accuracy and convergence problems. If such cells cannot be improved either the boundary mesh needs to be improved or different mesh parameters will have to be used. At this point, if the mesh has no severe problems, it can be used as input the solver so that numerical solutions can be obtained [2].

4.3 Grid constructed around ROBIN geometry

The volume mesh constructed around the ROBIN geometry was made up hybrid mesh. The hybrid mesh contained an outer mesh region, in which the node distribution is controlled and created by the unstructured grid generator, and a mesh region close to the ROBIN geometry.

4.3.1 Boundary Layer Region

The near wall mesh was created in the boundary layer region around the ROBIN geometry. This was done so that the turbulent conditions that occur in the boundary layer

could be captured. Prismatic cells, with incrementally increasing spacing in the Z direction, were used in this region to ensure that mesh resolution could be easily controlled. Control of the mesh resolution was essential as the cells in this region needed to comply with the wall y^+ range of 30-60 for wall function approach or with a wall y^+ range of less than four for the near wall modelling approach.

The boundary layer mesh, which was created for the wall function approach, was 1.5 mm with the growth rate of 1.2. This was for a wall y plus of around 30. On the other hand, the boundary layer mesh, which was created for the near wall modelling approach, was 0.05mm with a growth rate of 1.035 for a wall y^+ approach of one, and 0.1mm with a growth rate of 1.05 for a wall y^+ approach of three.

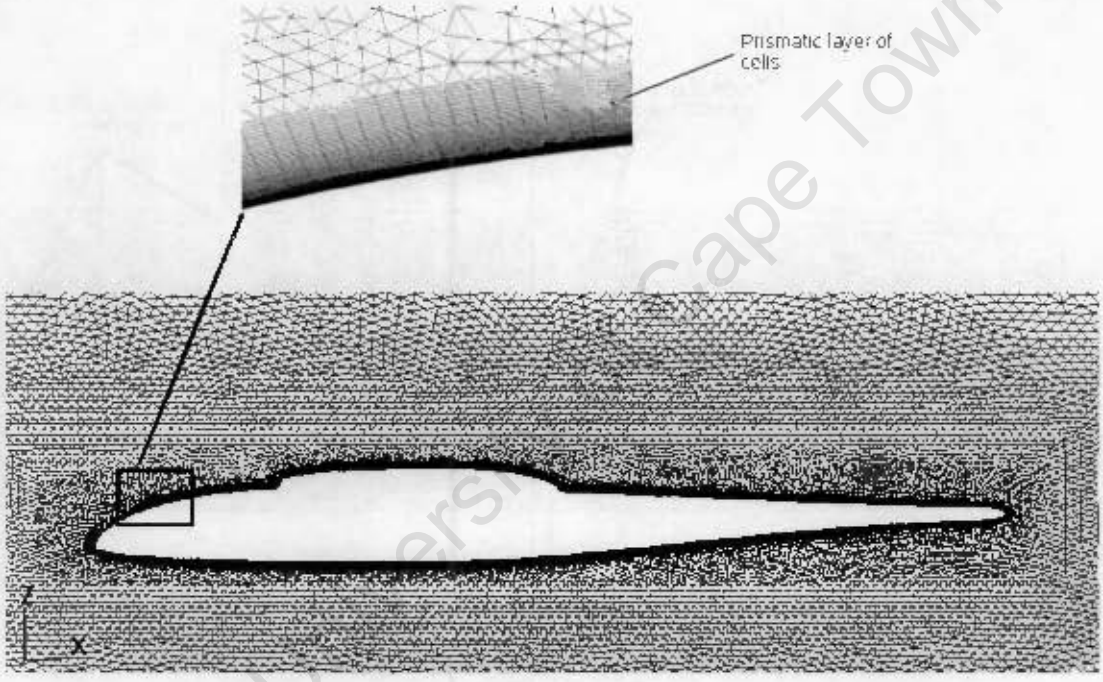

Figure 4.2: An indication of the prismatic boundary layer surrounding the helicopter fuselage for the near wall modelling approach

4.3.2 Outer Region

The outer mesh was created in the outskirts of the wind tunnel. Tetrahedral cells were used in this large region because fewer cells would be needed, which saves computational time and expense. The density of the mesh in the outer region was determined by the unstructured grid generator, and was directly proportional to the density of the prismatic

region surrounding boundary layer. Thus the outer region was made up of substantially more cells when a wall y^+ approach of one was attempted for the boundary layer mesh.

When a wall function approach was attempted, a wall y^+ value close to 30 was used. Alternatively grid independence was obtained for the wall y^+ values between one and three when the near wall modelling approach was used.

As seen in Figure 4.3, the unstructured mesh had a finer resolution near the fuselage so that any sharp gradients in flow properties close to the fuselage could be captured.

On average, between 2.4 and 3.0 million grid elements were used for the grids when near wall modeling approach (wall y^+ between 1-3) was implemented, and 1.8 million grid elements were used for the grids when the wall function approach (wall y^+ of 30) was implemented.

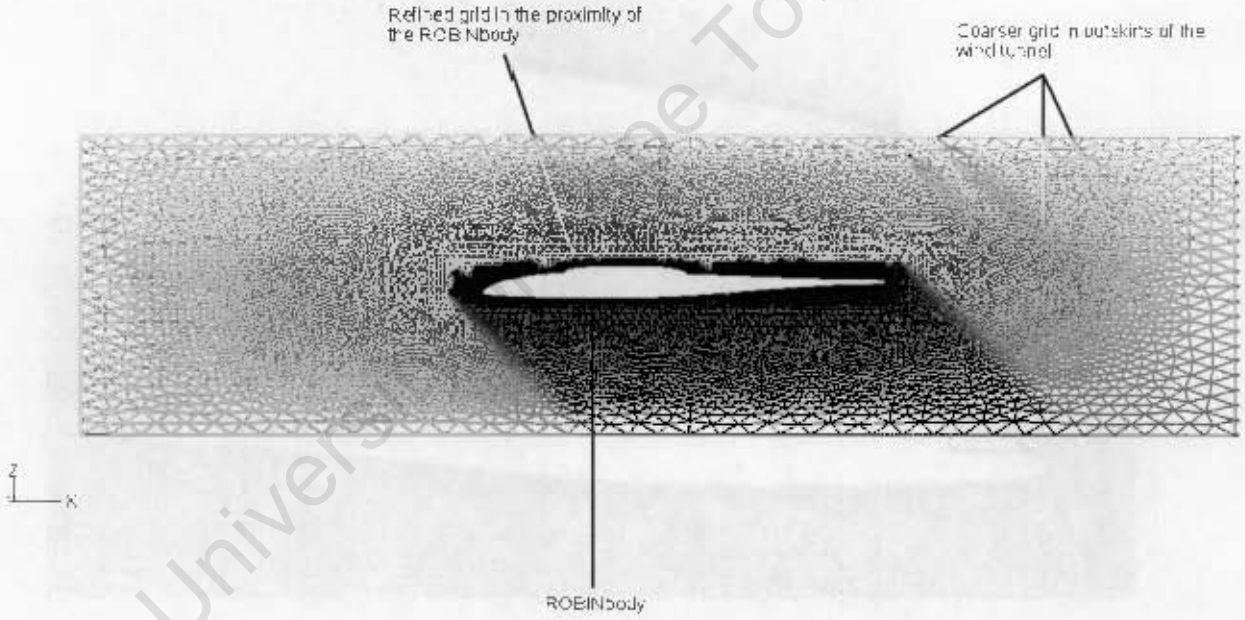

Figure 4.3: Detailed view of the computational grid around the helicopter fuselage for the near wall modelling approach

Chapter 5

Evaluation of turbulence models

A turbulent flow is expected around a helicopter configuration, during flight. This is because of the elevated speeds that helicopter flight occurs at, and also because of the complex nature of the flow-field found around a helicopter. Turbulence and vorticity will be found in the boundary layer surrounding the helicopter. The prediction of the turbulence in the boundary layer is especially important, as it will have an major effect on the accurate aerodynamic calculations. Turbulence models have been developed to predict turbulence for CFD. However, no single turbulence model is known to superior to the others for the above mentioned type of flow.

The aim of this section is thus to evaluate some of the more applicable turbulence models available in the solver, and to describe the numerical modeling strategy used in the solver to simulate the ROBIN fuselage.

5.1 CFD solver settings

The computational simulations, for the fuselage, were performed in the commercially available CFD code, FLUENT® on a dual processor, 64 bit machine. Using a dual processor saved computational time as the both processors could be used. The steady, incompressible, viscous, Reynolds-averaged conservation equations are solved in a Cartesian coordinate system.

A default segregated solver was used for all the simulations carried out. In this model, air was selected as the fluid type and the default settings of $1.225 kg/m^3$ and $1.7894 \times 10^{-5} kg/m-s$ were selected for the density and viscosity of the fluid, respectively.

5.1.1 Single Precision vs. Double Precision

The double precision solver was selected ahead of the single precision solver for the fuselage-only simulations, even though single precision solvers are more commonly used.

Double precision was chosen because it was computationally viable and it is said to be mainly used when there are very small variable differences, or where long, thin models, or any other model with highly differing length scales are used. This is because the single precision solver might not be able to accurately represent the values of the variables or the node coordinates.

The main difference between the two solvers is that the single precision uses six decimal places when doing calculations, whilst the double precision uses twelve decimal places [3].

5.1.2 Convergence criteria

In any simulation, the numerical accuracy of the simulation is checked by observing the residuals as it iterates. The residuals used to identify solutions in CFD are the difference in the amount of a variable entering and exiting a cell. In FLUENT®, the reported residual is actually the sum of the residuals of all the cells [3].

Solutions of the simulations are considered to be at an acceptable accuracy when the iterations of the solution have reached the convergence criteria specified.

The default, scaled settings of 10^{-4} for the residual convergence levels, which is the residual divided by the largest residual during the first five iterations, was used for all the variables.

5.2 Boundary Conditions

Many boundary conditions are offered in FLUENT®. Flow boundaries, are generally surfaces through which flow enters or exits the computational domain. The user is generally able to specify certain flow properties for flow crossing the boundary. However, this is dependent on the boundary condition chosen.

In the preprocessor, a constant velocity inlet was enforced at the inlet boundary. The outlet boundary was specified as constant pressure boundary. A symmetry boundary was assigned to the one of the side domain boundaries and no-slip walls were also applied to the top, bottom and remaining side domain boundaries. A range of other boundary conditions available in the solver were also tried for this current investigation.

In addition to the velocity inlet boundary, the pressure inlet boundary was tried at the inlet of the model. The pressure inlet, outlet and outflow boundaries were also tried at the outlet of the model. The pressure distribution along the helicopter fuselage was compared and no significant improvement in the accuracy was noticed when the alternative boundary conditions were used. Therefore, the initial boundary types specified in the pre-processor, which included the no slip wall boundary on the top, bottom and side walls was deemed acceptable. An explanation of each of these boundary types are, however, explained below.

5.2.1 Wall boundary

Wall boundary conditions are used to define bounded limits for the flow in the computational domain. In viscous flows, the no-slip boundary condition is applied at walls by default. However, varying degrees of slip and/or wall-velocities may be specified by the user [3].

5.2.2 Velocity inlet boundary

Velocity inlet boundary conditions are used to define the flow velocity, along with all relevant scalar properties of the flow at flow inlets. The total (or stagnation) properties of the flow are not fixed, so they will rise to whatever value is necessary to provide the prescribed velocity distribution. Generally, this condition is applied to the flow entering the computational domain. However, in special instances a velocity inlet may be used to define the flow velocity at flow exits [3].

5.2.3 Pressure inlet boundary

Pressure inlet boundary conditions are used to define the total fluid pressure at flow inlets, along with all other scalar properties of the flow. They are suitable for both incompressible and compressible flow calculations and the velocity direction may also be specified. Pressure inlet boundary conditions are primarily used when the inlet pressure is known but the flow rate and/or velocity is not known. In the case of flow, exiting through the boundary, the specified total pressure is used as the static pressure [3].

5.2.4 Pressure outlet boundary

Pressure outlet boundary conditions are used to define the static pressure at the outlet boundary. Generally, this condition is applied to flow exiting the computational domain, but for flows entering through the boundary, the specified static pressure is again used. The value of the specified static pressure is used only while the flow is subsonic. Should the flow become locally supersonic, the specified pressure will no longer be used and pressure will be extrapolated from the flow in the interior. All other flow quantities are extrapolated from the interior. If the flow reverses direction at the pressure outlet boundary, the user is allowed to define scalar quantities such as back flow conditions turbulence variables and velocity direction [3].

5.2.5 Outflow boundary

Outflow boundary conditions are used to model flow boundaries where details of the velocity and pressure of flow exiting the computational domain are not known prior to solution of the flow problem. They are appropriate where the exit flow is close to a fully developed condition, as the outflow boundary condition assumes a zero normal gradient for all flow variables except pressure. Importantly, any re-circulation across the boundary may lead to inaccurate results, since when flow enters the domain through an outflow boundary, scalar properties of the flow are not defined [3].

5.3 Discretisation schemes

The choice of a particular discretisation scheme is a compromise between numerical stability and accuracy. While first-order discretisation is numerically more stable than the second-order scheme, it will generally yield less accurate results, especially when the flow is not aligned with the grid. This is due to the occurrence of numerical diffusion.

In the current investigation a first-order upwind difference scheme was used ahead of the second-order upwind scheme for the momentum and pressure convection terms. The upwind scheme uses the variable values from the cell "upstream" from it. The use of second-order upwind scheme made no difference in predicting the pressure distribution along the helicopter fuselage. First order upwind schemes are also known to be quicker in reaching converged solutions than the second order upwind schemes. The QUICK scheme, a weighted average of second-order upwind difference and central difference schemes, was investigated for discretisation of the momentum terms, with no significant

improvement in accuracy.

Discretisation of the turbulence terms was accomplished using a first order upwind difference scheme. A more accurate solution of the turbulence terms was not beneficial as any turbulence model is at best an approximation of the effects of turbulence.

5.4 Turbulence Models

As part of the evaluation procedure of the turbulence models, consecutive tests were conducted on the grids created for the wall function approach (wall y^+ of 30) and near wall modelling approach (wall y^+ of 3). The turbulence models that were evaluated were the $k-\epsilon$, and its variants, the RNG and realizable models; $k-\omega$, and its variation, the SST model; as well as the Spalart-Allmaras, and RSM models.

The experimental data presented by Freeman and Mineck [6] was used as a comparison. The simulations conducted all took place at a Mach number of 0.062 and an effective Reynolds number of 4.46 x 10^6. Coefficients of pressure were compared along the four cross sections $x/l = 0.35, x/l = 1.17, x/l = 1.35, x/l = 1.54$. It is useful to note that the pressure coefficients used are defined in the following manner:

$$C_p = \frac{p - p_{\text{inf}}}{q_{\text{inf}}} \tag{5.1}$$

where p is the pressure at the point stipulated
p_{inf} is the reference pressure
q_{inf} is the reference dynamic pressure.

$x/l = 0.35$ is located before the pylon on the front section of the helicopter fuselage, whilst the other stations are located behind the pylon. The data on the cross sectional stations is compared and presented as function of the z/l non-dimensional coordinates. It is also important to note that model used experimentally contained a support strut at the bottom of the fuselage.

5.4.1 Wall function approach

Reasonable agreement with the experimental data was obtained at the first station $x/l = 0.35$ for all the turbulence models used. However, at the station $x/l = 1.17$, which occurs just behind the pylon the $k-\omega$ did not perform adequately. As shown in Figure 5.1 the $k-\omega$ over predicted the experimental data. Even though the turbulence models

performed adequately at this station, a huge discrepancy was found to exist between the experimental data and the numerical data at the side and bottom (negative z/l) of the fuselage. This is mainly because of the wake that is produced due to presence of the support strut at the bottom of the fuselage. As seen in Figure 5.2, at $z/l = 0.1$, all the models also over-predicted the wake that occurs behind the pylon.

The last two stations $x/l = 1.35, x/l = 1.54$ (Figure 5.3 and 5.4) also showed a separation point on the side of the fuselage, which can be seen by a sharp reversal of the pressure plot. None of the turbulence models captured the separation exactly. It was also found that the SST model hugely under-predicted the experimental coefficient of pressure distribution on the station, $x/l = 1.35$, whilst the $k - \omega$ over-predicted the coefficient of pressure distribution on the same station. On the last station ($x/l = 1.54$), the Spalart-Allmaras, $k - \omega$, and SST model hugely under predicted the experimental coefficient of pressure distribution.

Thus over the range of simulations, for the wall function approach grids, the $k - \epsilon$, and its variants, the RNG and realizable models were found to be suitable choices.

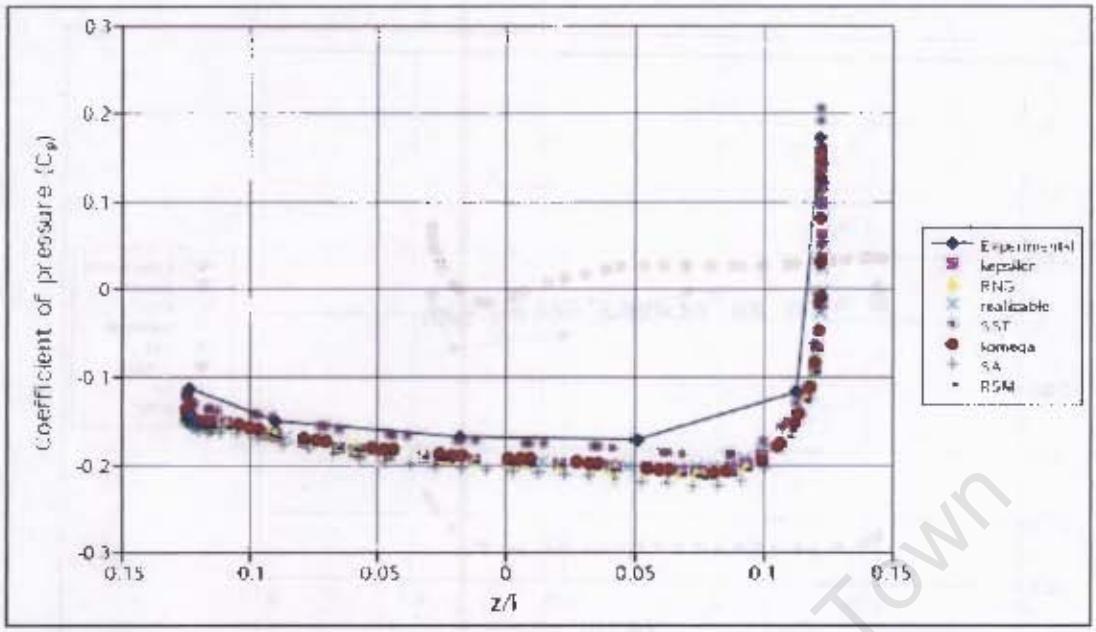

Figure 5.1: Coefficient of pressure distribution,ROBIN fuselage, at x/l 0.350 for the wall function approach grids

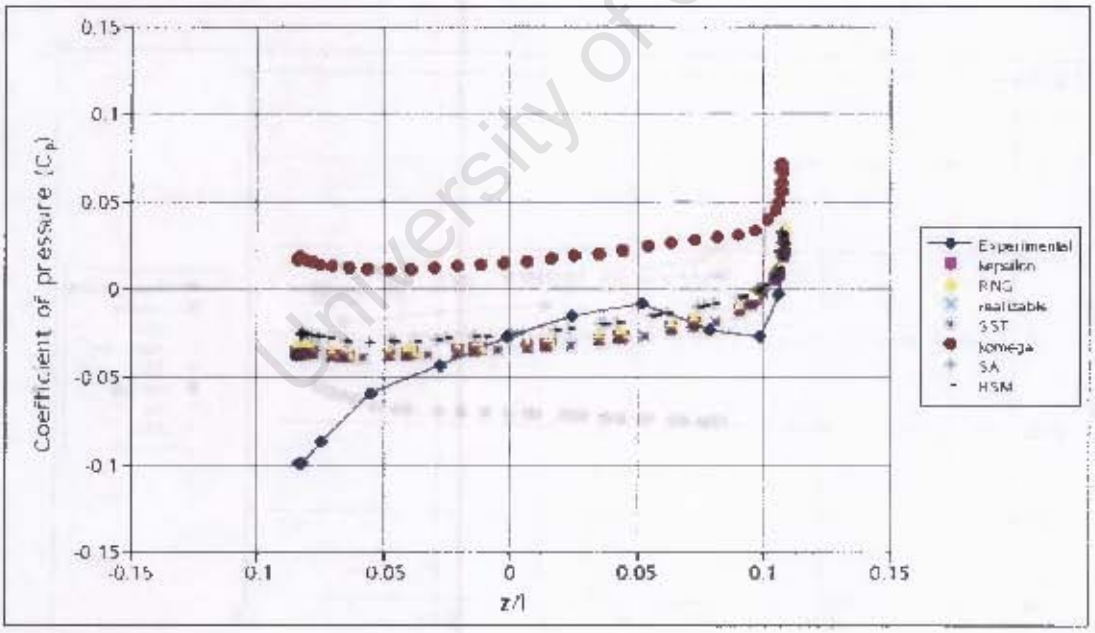

Figure 5.2: Coefficient of pressure distribution,ROBIN fuselage, at $x/l = 1.17$ for the wall function approach grids

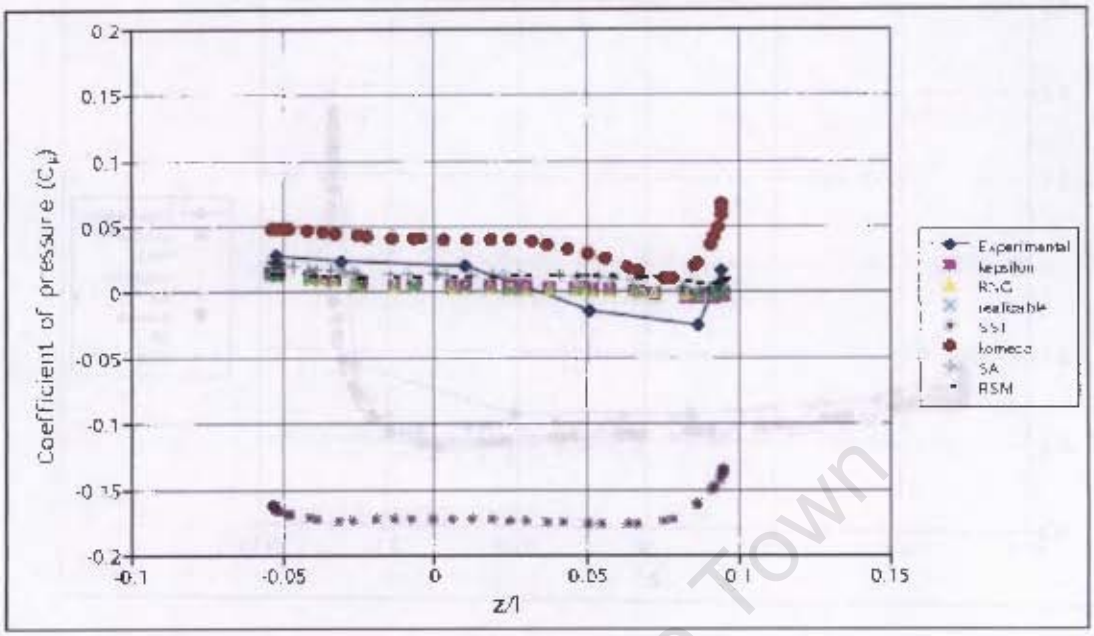

Figure 5.3: Coefficient of pressure distribution,ROBIN fuselage, at $x/l = 1.35$ for the wall function approach grids

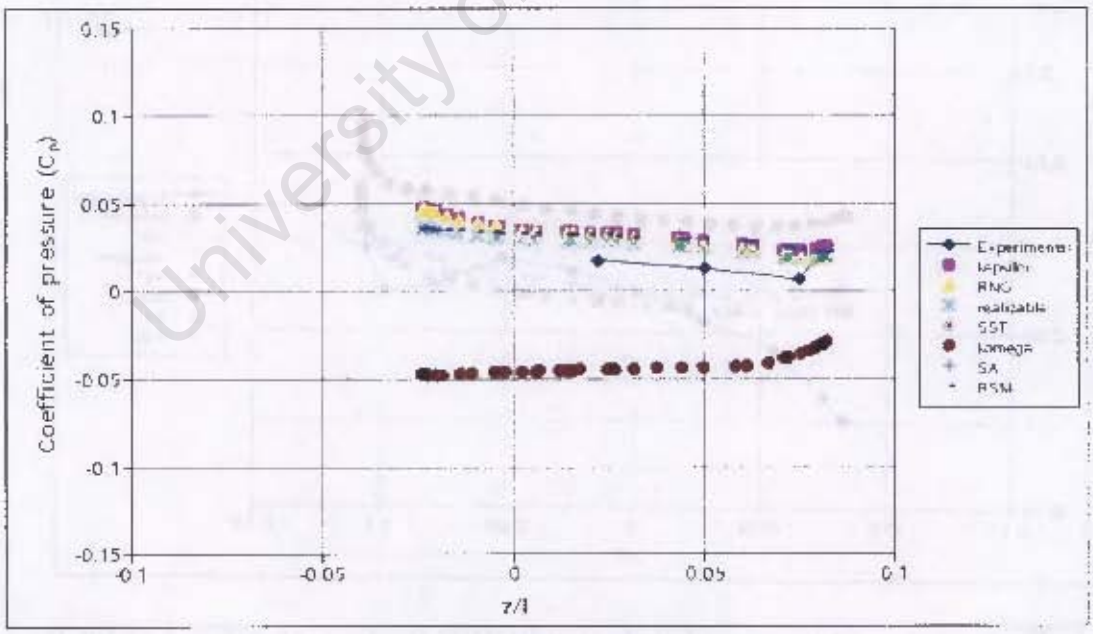

Figure 5.4: Coefficient of pressure distribution,ROBIN fuselage, at $x/l = 1.54$ for the wall function approach grids

5.4.2 Near wall modelling approach

As seen in Figure 5.5, at the first station $x/l = 0.35$, the Spalart-Allmaras and RSM slightly under-predicted the experimental coefficient of pressure distribution, whilst the other turbulence models had a good agreement with the data. On the second station, x/l 1.17, the SST, RSM and Spalart-Allmaras models predicted the coefficient of pressure distribution quite adequately. However, once again, a huge discrepancy was found to exist between the experimental data, and the numerical data at the side and bottom (negative z/l) of the fuselage. This is again attributed to the wake that is produced around the support strut. The SST model predicts the wake behind the pylon($z/l = 0.1$) closely but is unable to match the maximum coefficient of pressure measured at that station.

On the station $x/l = 1.35$, the realizable, $k - \epsilon$ and $k - \omega$ models under-predicted the experimental data. The RNG and RSM models closely predicted the coefficient of pressure distribution at the separation point but deviated in other cross sectional regions. However, the SST model and the Spalart-Allmaras models had a reasonable agreement with the data. The SST and Spalart-Allmaras turbulence models did not, however, predict the separation point very well.

Only the SST model was able to have a reasonable agreement with the experimental data at the final station. The RSM model over predicted the coefficient of pressure distribution whilst the remaining, turbulence models under predicted the coefficient of pressure distribution. Therefore, for the range of turbulence models tested, for the near wall modelling approach grids used, the SST models performs the best over the range of compared experimental data.

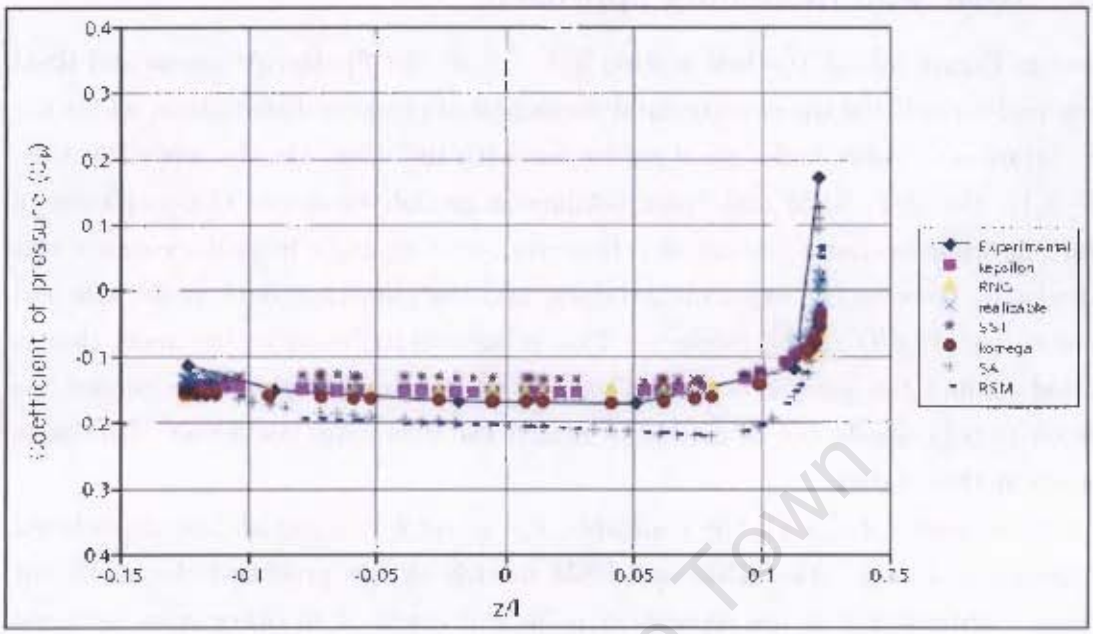

Figure 5.5: Coefficient of pressure distribution,ROBIN fuselage, at x/l 0.350 for the near wall modelling approach grids

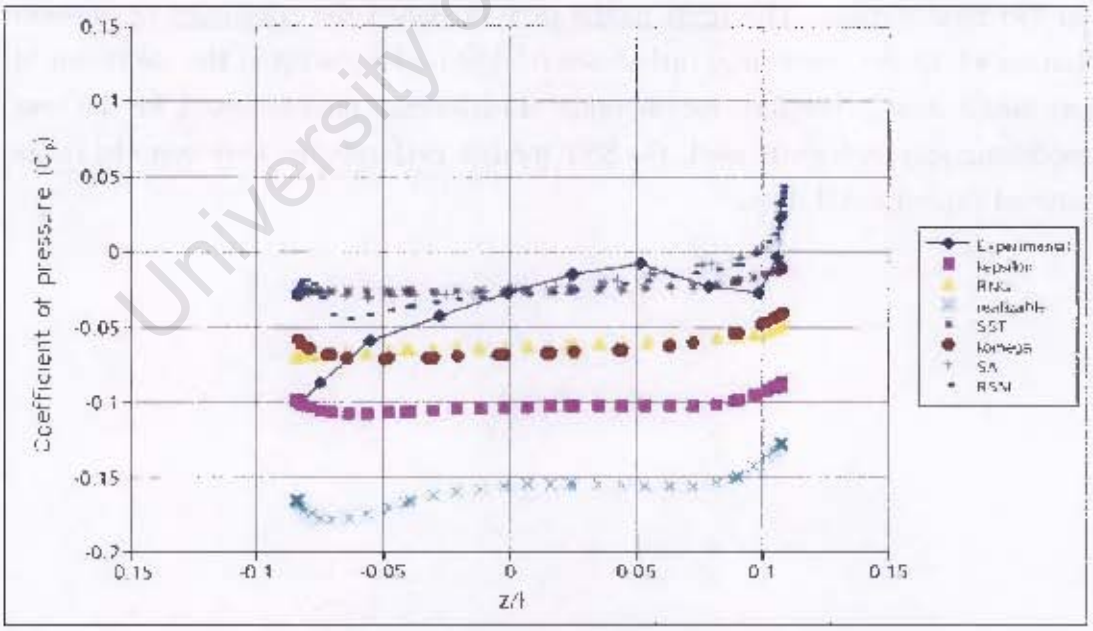

Figure 5.6: Coefficient of pressure distribution,ROBIN fuselage, at $x/l = 1.17$ for the near wall modelling approach grids

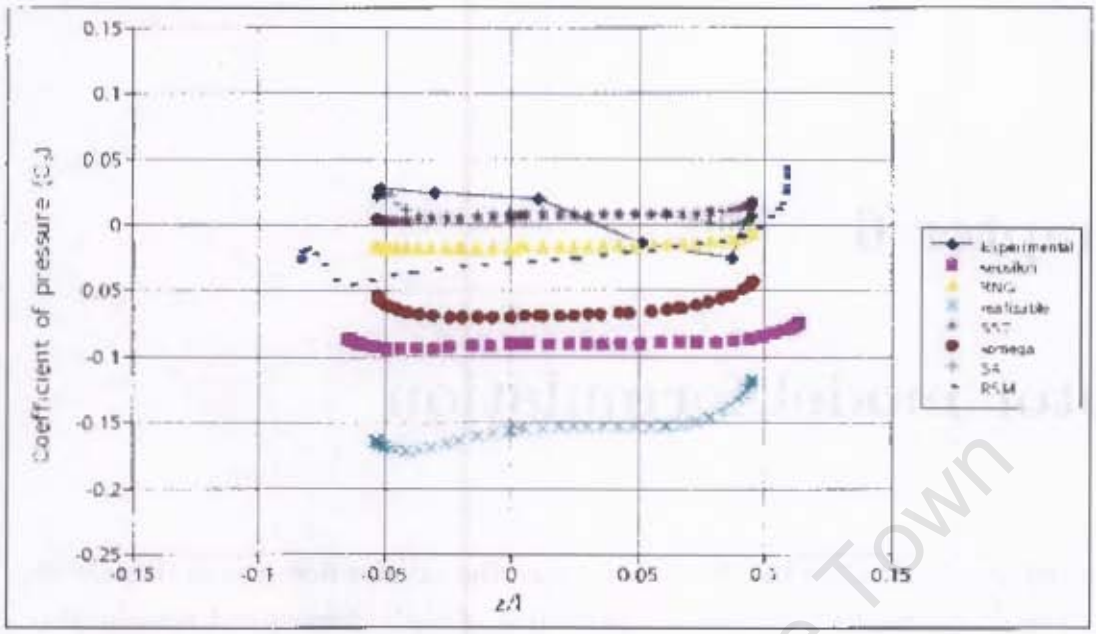

Figure 5.7: Coefficient of pressure distribution,ROBIN fuselage, at $x/l = 1.35$ for the near wall modelling approach grids

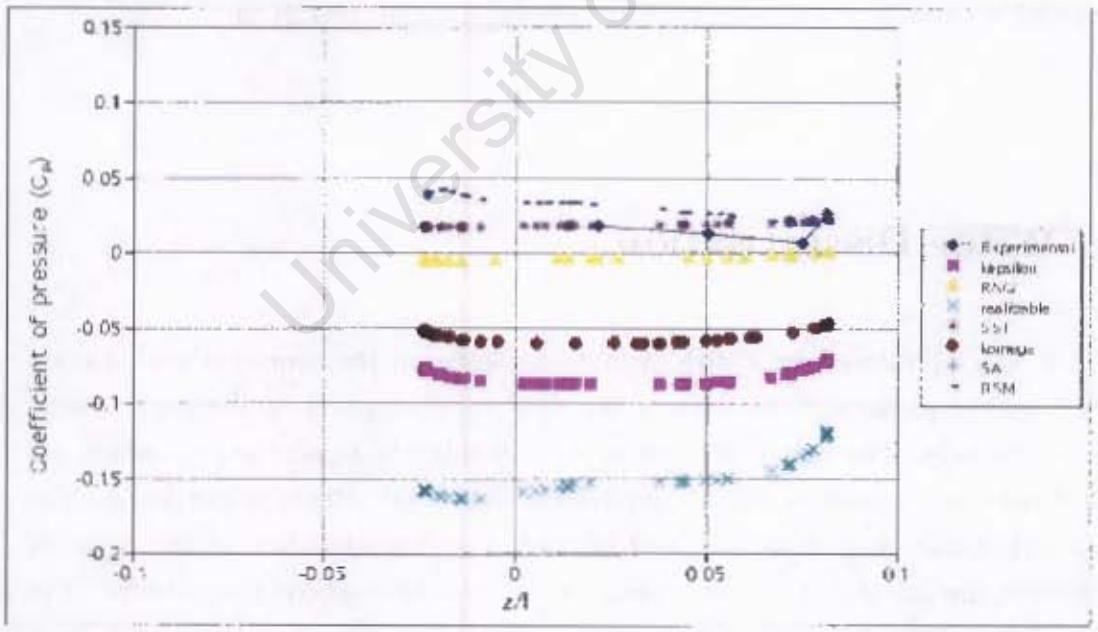

Figure 5.8: Coefficient of pressure distribution,ROBIN fuselage, at $x/l = 1.54$ for the near wall modelling approach grids

Chapter 6

Rotor model formulation

The model used to simulate the effects of rotor on the exterior flow-field in this investigation was a rotor model that used a combination of blade element and actuator disk theory. Forces exerted by the rotor are calculated with the use of blade characteristics and flow properties. These forces are applied to the domain as momentum sources terms. The rotor model was incorporated with the CFD solver, through the use of a User Defined Function (UDF). In this chapter, the mathematical formulation of the rotor model, written in the C programming language, is thus described.

6.1 Rotor Discretisation

The rotor was represented by a disk of finite thickness in the computational domain created, and the physical dimensions of the disk were described by the region swept by the rotor-blade. The disk is discretised by a number of regular annuli, which are further divided into elements. This is depicted in Figure 6.1. Blade properties, such as chord length, blade twist, thickness, and lift and drag characteristics, at the center of each element, are calculated and were assumed constant throughout that element. The actuator disk region, within the computational domain, is defined during construction of the computational grid, and the particular elements to which the momentum sources are to be applied are identified by a linking algorithm during the solution initialisation process.

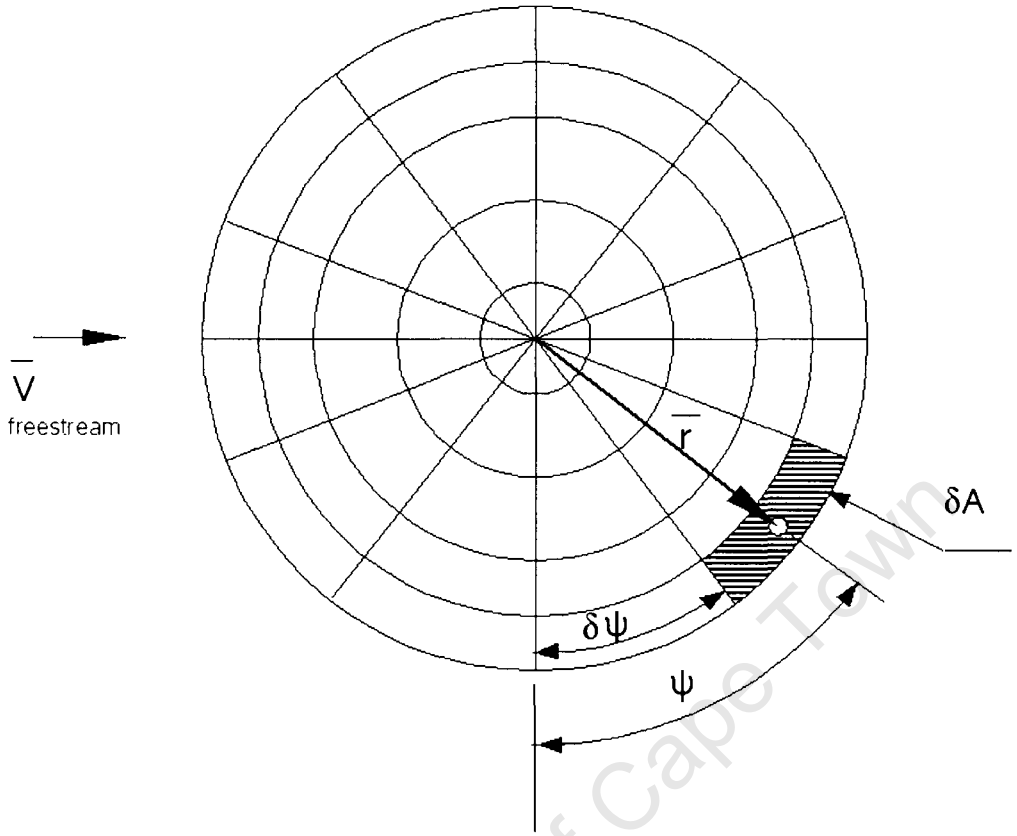

Figure 6.1: Rotor discretisation representation

6.2 Coordinate Systems

In the calculation of the momentum source terms four coordinate systems are used to describe the rotor and blade elements. The first cartesian coordinate system, $(\overline{X}, \overline{Y}, \overline{Z})$, is the global coordinate system for the computational domain. A second Cartesian system is defined, $(\overline{x}, \overline{y}, \overline{z})$, with z pointing along the axis of rotation, in the upstream direction, and x defined relative to the helicopter fuselage.

A rotor based cylindrical coordinate system,$(\overline{r}, \overline{\psi}, \overline{z})$, is also used, with ψ, the azimuth angle, defined relative to $\overline{x}$. A third Cartesian coordinate system is defined for each element Furthermore, when calculation of the blade element forces takes place, a third coordinate system is defined. This coordinate system was defined in same way as the second cartesian system. Transformation tensors were used to change the description of vectors from one coordinate system to another as convenient.

6.3 Calculation of the momentum sources

Values for the momentum sources terms are determined at the end of every iteration. Blade Element theory is used to determine the forces imparted on the fluid by the rotor blades, which may, with little manipulation, be directly substituted into the governing momentum equations as the source terms.

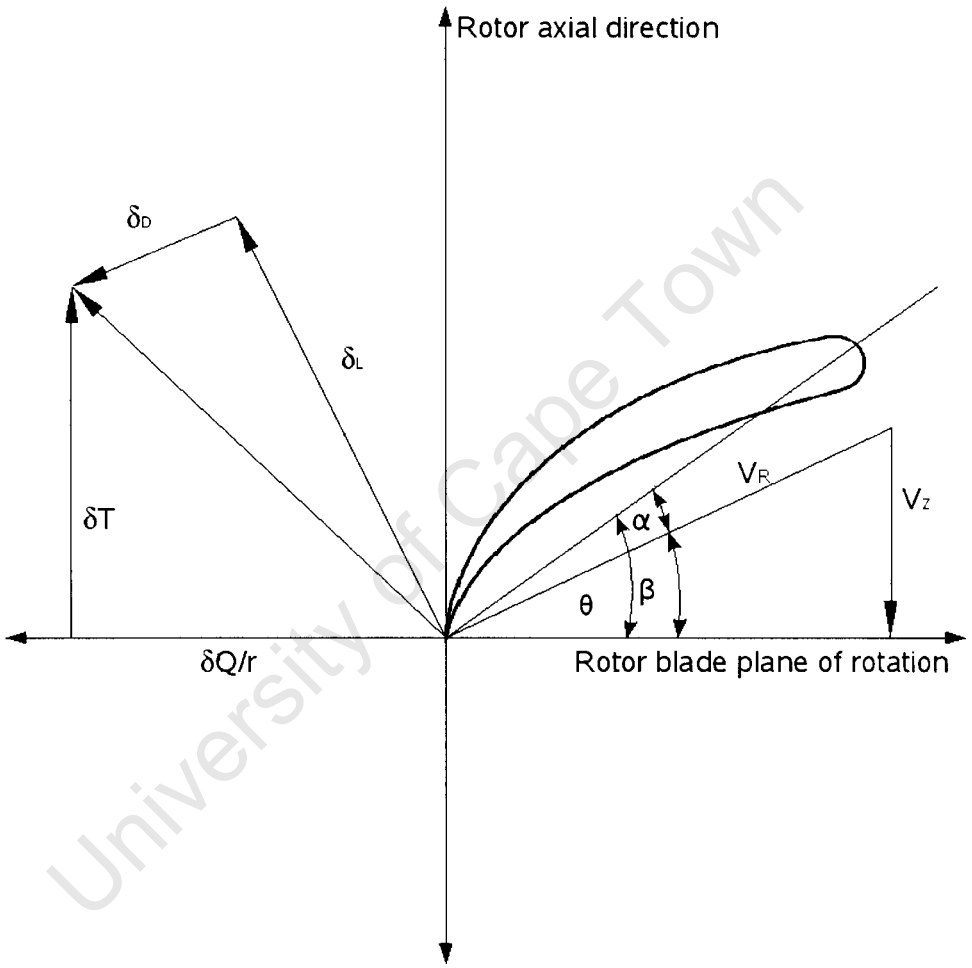

Figure 6.2: Blade element representation showing the relative velocity vector, and the resulting aerodynamic loads

According to Von Mises [20], the force exerted on the fluid stream at any location within the actuator disk is a function of the fluid velocity vector, $\overline{v_R}$, relative to the rotor-blade, as well as the lift and drag characteristics of the blade cross-sectional profile. Figure 6.2 depicts the relative velocity vector, as well as the resulting elemental lift, δL, and elemental drag force, δD, on a blade element, at a blade radius $\overline{r}$

The lift and drag forces are determined according to :

$$\delta L = \frac{1}{2}|v_R|^2 C_l.c.\delta r \tag{6.1}$$

$$\delta D = \frac{1}{2}|v_R|^2 C_d.c.\delta r \tag{6.2}$$

where C_l is the coefficients of lift
C_d is the coefficient of drag
c is the rotor blade chord length
$\overline{v_R}$ is the relative velocity vector
δr is the blade element thickness.

The solver provides the flow-field velocity, relative to the global coordinate system, at any prescribed location within the flow domain. Therefore, the velocity vectors at the actuator disk could be determined directly. However, the velocity field encountered by the two-dimensional blade elements of the rotor model differs considerably from the uniform velocity field for which the lift and drag coefficients are valid.

The most notable difference being the tangential velocity component of the velocity field on the two-dimensional blade element down stream side. To compensate for this discrepancy, the relative velocity vector, $\overline{v_R}$, at the two dimensional blade element is the average of the trailing edge vector and the free-stream velocity vector upstream of the two dimensional blade element. It should also be noted when the final value of $\overline{v_R}$ is calculated the rotation velocity of the rotor is compensated for.
From Figure 6.2 the blade element thrust, δT and torque, δQ are determined by

$$\delta T = \delta L \cdot cos\beta - \delta D \cdot sin\beta \tag{6.3}$$

$$\delta Q = (\delta L \cdot sin\beta + \delta D \cdot cos\beta) \cdot r \tag{6.4}$$

where β is the angle between the relative velocity vector, $\overline{v_R}$, and the rotor blade plane of rotation.

The blade forces were expressed as momentum sources/sinks in the governing equations, and the blade element thrust and torque need therefore to be expressed as a force per unit volume. It follows that

$$\frac{\delta T}{\delta V} = \frac{\sigma \cdot \delta T}{c \cdot \delta r \cdot t_{Rdisk}} \tag{6.5}$$

$$\frac{\delta Q}{\delta V} = \frac{\sigma \cdot \delta Q}{c \cdot \delta r \cdot t_{Rdisk}} \tag{6.6}$$

where δr is the element radial dimension
$t_{Rdisk}$is the rotor disk thickness
σ is defined as the blade solidity ratio:

$$\sigma = \frac{N \cdot c}{2\pi \cdot r} \tag{6.7}$$

Substitution of equations 7.1 to 6.6 into the equations 6.8 and 6.9 yields:

$$\frac{\delta T}{\delta V} = \frac{1}{2}\rho \overline{v_R}^2 \frac{\sigma}{t_{Fr}}(C_l \cdot cos\beta - C_d \cdot sin\beta) \tag{6.8}$$

$$\frac{\delta Q}{\delta V} = \frac{1}{2}\rho \overline{v_R}^2 \cdot r \cdot \frac{\sigma}{t_{Fr}}(C_l \cdot cos\beta + C_d \cdot sin\beta) \tag{6.9}$$

Since the relative velocity magnitude can be calculated and the other components are known, only the coefficients of lift and drag needs to be determined to resolve equations 6.8 and 6.9. The lift and drag characteristics of a profile section are a function of the Reynolds number and angle of attack,α, alone. This depends on the particular rotor being modelled and within a specified range of Mach numbers. The Reynolds number is calculated as:

$$Re = \frac{p \cdot |\overline{v_R}| \cdot c}{\mu} \tag{6.10}$$

where ρ is the density obtained directly from the solver
μ is the dynamic viscosity obtained directly from the solver
$|\overline{v_R}|$ is the relative velocity magnitude
c is the profile chord length.

The angle of attack is calculated according to:

$$\alpha = \theta - \beta \tag{6.11}$$

It is necessary in helicopters to vary the blade pitch angles as a function of azimuth angle, to account for the free-stream velocity component in forward flight. Controlling the geometric angle of attack, or the pitch angle, of the blades serves to keep the angle of attack constant for advancing and retreating blades. This is to prevent pitching moments

being created by unequal aerodynamic load characteristics along diametrically opposing blades. In the current simulations, the relationship between the pitch angle, θ, and the rotor azimuth angle, ψ, is described by the equation

$$\theta = \theta_0 - A_1 cos\psi - B_1 sin\psi + \frac{r}{R}\theta_1 \tag{6.12}$$

where θ_0 is the collective pitch angle
θ_1 is the blade twist
A_1 defines the lateral cyclic pitch angle
B_1 defines the longitudinal cyclic pitch angle
r is the element radial position
R is the rotor tip radius.

For α values falling outside the range of experimental data, lift and drag characteristics of for a flat plate were used. According to Hoerner [8] and Borst and Hoerner [9], the dimensionless lift and drag coeffiecients for a flat plate are given respectively as

$$C_l = C_{dmax} \cdot sin\alpha \cdot cos\alpha \tag{6.13}$$

and

$$C_d = C_{dmax} \cdot sin^2\alpha \tag{6.14}$$

A smooth transition between the airfoil and flat plate lift and drag characteristics was ensured with the introduction of a fourth order polynomial and trigonometric functions in the overlap regions. The resulting lift and drag curves are shown in Figure 6.3, as a function of angle of attack, α. Linear interpolation is used to determine C_l and C_d values at intermediate Reynolds numbers.

6.4 Calculation of moments

The rolling moment, values for the moment about the x-axis, and the pitching moment, moment about the y-axis, is function of the azimuthal angle and is determined at the end of very iteration. These moment values are of particular importance since the rotor needs to be balanced to operate correctly in the various flight conditions.

Once again the blade forces are expressed as momentum sources in the governing equations since the rolling moment and pitching moments were expressed as force per unit volume:

$$\frac{\delta M_x}{\delta V} = -\frac{1}{2}\rho\overline{v_R}^2 \cdot r \cdot \frac{\sigma}{t_{Fr}}(C_l \cdot cos\beta - C_d \cdot sin\beta) \cdot cos\psi \tag{6.15}$$

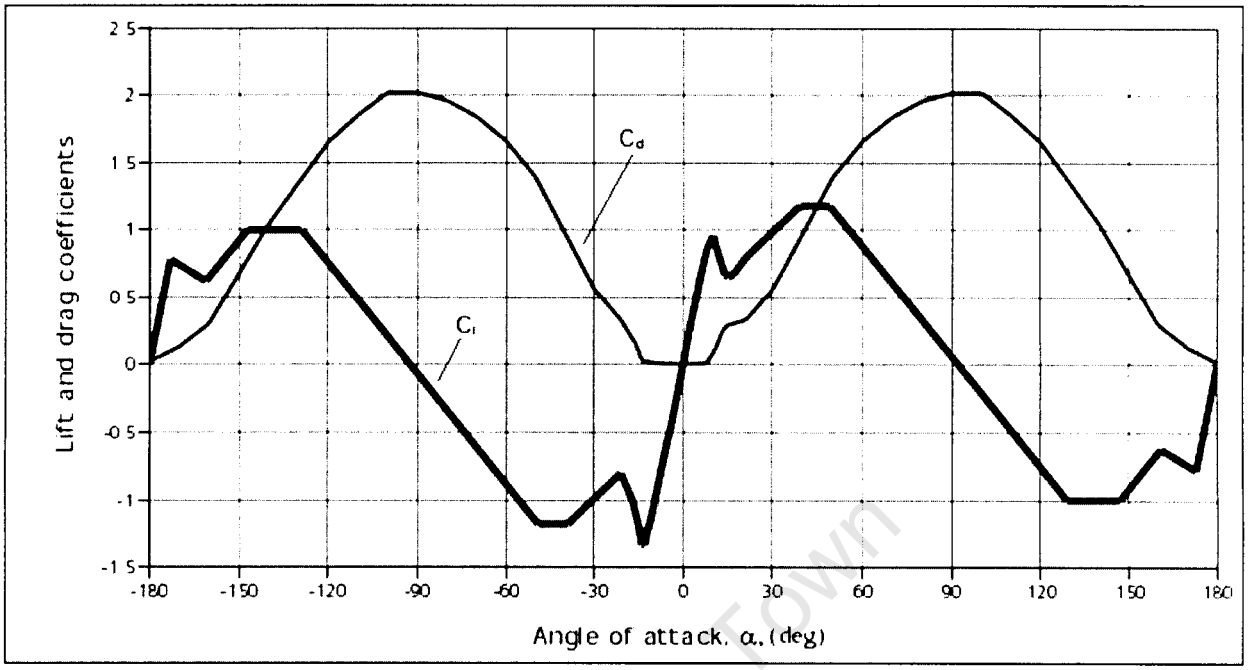

Figure 6.3: Lift and drag coefficients as a function of angle of attack, used for calculation of the aerodynamic forces at the blade elements defined in the rotor model[19]

$$\frac{\delta M_y}{\delta V} = \frac{1}{2}\rho\overline{v_R}^2 \cdot r \cdot \frac{\sigma}{t_{Fr}}(C_l \cdot cos\beta - C_d \cdot sin\beta) \cdot sin\psi \tag{6.16}$$

6.5 Blade Flapping

The blade flapping of the rotor is not calculated in the code. However, if the flapping motion is known the code can account for the first two harmonics of flapping by adding the flapping velocity to the velocity normal to the blade path. Further due to its relative unimportance in this investigation, the lead-lag motion is also ignored.

Chapter 7

Implementation and validation of the trim routine

This chapter presents the trim routine used to balance the rotor in forward flight conditions, as well as the tests conducted to validate this routine. The method used to trim the rotor was the Newton-Raphson Iterative method. Only a single processor was used in the verification of the trim routine, since this was one of the restrictions of the rotor model. Verification of the trim routine was initially carried out on a rotor model without the fuselage. This was mainly done because of the lesser computational time and expense required to test the trim routine. However, verification of the trim routine for the rotor and fuselage was performed for a few scenarios. Therefore, in this chapter, both of the cases in which the trim routine is validated are presented. In each of these cases validated the configuration of the model, the computational grid used, and results are depicted. Finally, the results are discussed.

7.1 Trim Routine

The trim routine that was created was implemented in the rotor model, and could be executed when the user desired. The trim routine implemented was able to calculate the correct collective pitch angle (θ_0) and cyclic pitch coefficients (A_1, B_1) in order to achieve the desired thrust coefficient, and eliminate moments around the hub for a particular flight speed. This relationship between the thrust and moment coefficients; and θ_0, A_1, B_1 are displayed in equations (7.1-7.3):

$$C_T = C_T(\theta_0, A_1, B_1) \tag{7.1}$$

$$C_{My} = C_{M_y}(\theta_0, A_1, B_1) \tag{7.2}$$

$$C_{Mx} = C_{M_x}(\theta_0, A_1, B_1) \tag{7.3}$$

where C_T represents the coefficients of thrust
C_{Mx} represents the rolling moment
C_{My} represents the pitching moment.

C_T, C_{Mx}, C_{My} are defined by the following equations:

$$C_T = \frac{T}{\rho\pi R^2(\Omega R)^2} \tag{7.4}$$

$$C_{M_y} = \frac{M_y}{\frac{1}{2}\rho\pi R^2(\Omega R)^2} \tag{7.5}$$

$$C_{M_x} = \frac{M_x}{\frac{1}{2}\rho\pi R^2(\Omega R)^2} \tag{7.6}$$

where ρ is the density of the fluid
R is the radius of the rotor
Ω is the rotational speed of the rotor.

Following Yang et al. [22] a Newton-Raphson iterative method was employed to trim the rotor. Given an initial guess for θ_0, A_1, B_1; and the desired thrust and moment coefficients ($C_{T^{desired}}; C_{M_x{}^{desired}}; C_{M_y{}^{desired}}$), equation 7.7 was used by the the trim routine to determine the change in $\Delta\theta_0, \Delta A_1, \Delta B_1$:

$$\begin{bmatrix} C_{T^0} \\ C_{M_x{}^0} \\ C_{M_y{}^0} \end{bmatrix} + \begin{bmatrix} \frac{\partial C_T}{\partial \theta_0} & \frac{\partial C_T}{\partial A_1} & \frac{\partial C_T}{\partial B_1} \\ \frac{\partial C_{M_x}}{\partial \theta_0} & \frac{\partial C_{M_x}}{\partial A_1} & \frac{\partial C_{M_x}}{\partial B_1} \\ \frac{\partial C_{M_x}}{\partial \theta_0} & \frac{\partial C_{M_y}}{\partial A_1} & \frac{\partial C_{M_y}}{\partial B_1} \end{bmatrix} \begin{bmatrix} \Delta\theta_0 \\ \Delta A_1 \\ \Delta B_1 \end{bmatrix} = \begin{bmatrix} C_{T^{desired}} \\ C_{M_x{}^{desired}} \\ C_{M_y{}^{desired}} \end{bmatrix} \tag{7.7}$$

A step by step explanation of the method used, in the trim routine, to determine the solution and various aspects of equation 7.7 are described below :

1. With the initial guess for θ_0, A_1, B_1 set the flow solver was run till convergence of the flow-field, to get the initial rotor performance (C_{T^0}, $C_{M_x{}^0}$, $C_{M_y{}^0}$).

2. Another θ_0 value was then guessed and the simulation was restarted. A_1 and B_1 remained unchanged.

3. Once convergence was reached, the new rotor performance was obtained from the rotor model code and the first column of the Jacobian matrix, $\begin{bmatrix} \frac{\partial C_T}{\partial \theta_0} & \frac{\partial C_T}{\partial A_1} & \frac{\partial C_T}{\partial B_1} \\ \frac{\partial C_{M_x}}{\partial \theta_0} & \frac{\partial C_{M_x}}{\partial A_1} & \frac{\partial C_{M_x}}{\partial B_1} \\ \frac{\partial C_{M_x}}{\partial \theta_0} & \frac{\partial C_{M_y}}{\partial A_1} & \frac{\partial C_{M_y}}{\partial B_1} \end{bmatrix}$ was calculated. The components of the Jacobian matrix were calculated using a linear approximation method. This calculation was done relative to the initial rotor performance characteristics and the initial guesses for θ_0, A_1, B_1.

4. θ_0 was changed back to the initial value and a new A_1 was guessed. B_1 remained unchanged.

5. When convergence was reached once again, the rotor performance was obtained from the rotor model code and the second column of the Jacobian matrix was calculated.

6. Another B_1 value was then guessed and A_1 was changed back to the initial value. θ_0 remained unchanged.

7. Once convergence was reached, the new rotor performance was obtained from the rotor model code and the third column of the Jacobian matrix was calculated.

8. With the use of equation 7.7 the changes to θ_0, A_1, B_1 was calculated.

9. A simulation was conducted with the new θ_0, A_1, B_1 values. Once convergence was reached the rotor performance was obtained. If this rotor performance was equivalent to the desired values then the new θ_0, A_1, B_1 were correct.

10. If the rotor performance was not equivalent to the desired values then the process was restarted from step one with the current rotor performance values being seen as the initial rotor performance values.

The relationship between the rotor aerodynamic parameters C_T, C_{M_x}, C_{M_y} and θ_0, A_1, B_1 were found to be non-linear. However, the diagonal components of the Jacobian matrix $(\frac{\partial C_T}{\partial \theta_0}, \frac{\partial C_{M_x}}{\partial A_1}, \frac{\partial C_{M_y}}{\partial B_1})$, were found to be at least ten times greater than the non-diagonal components of the Jacobian matrix. This meant that even though the relationship between the aerodynamic parameters and θ_0, A_1, B_1 was non-linear, it could be possible to assume a linear relationship. Thus, because of this finding it would be practical to implement linear techniques to solve for θ_0, A_1, B_1. In future investigations methods such as the secant method, could be used instead of the Newton-Raphson method in order to find the optimum θ_0, A_1, B_1 needed to trim the rotor.

7.2 Validation of the trim routine

Validation of the trim routine was a fairly long process. Because of the computational restrictions of the rotor model code, only a single processor could be used at any one time. Therefore, to save computational time, verification of the trim routine was carried out on a model with a rotor, but no fuselage (rotor-alone model). The effects of the fuselage on the results were not known but it was felt that this was sufficient for initial testing of the routine. However, before the trim routine verification could begin, a simple test of the rotor in a state of hover was simulated. This was carried out to ensure that the rotor would be balanced, as it should be, in a hover condition. A few simulations of the rotor and fuselage were carried out to check the results and effects that the fuselage would have on θ_0, A_1, B_1. Therefore, in the validation of the trim routine the testing of the rotor model with the rotor-alone model was carried out. This was followed by the testing of the trim routine in forward flight conditions, for both the rotor-alone as well as the rotor and fuselage simulations.

7.2.1 Isolated Rotor

Rotor Model Configuration

In order to use the rotor model successfully, critical information about the rotor system was required. This included the rotational speed, blade profiles, dimensions, and number of blades. These values were used as inputs in the rotor model code, as well as defining the construction of the actuator disk within the computational domain. The root cutout used was 24 % of the radius, and the rotor rotated in an anticlockwise direction. In the numerical rotor model geometry created, the effects of the rotor hub were neglected.

Rotation speed (rpm)	Number of blades	Blade profile	Blade chord (m)	Rotor diameter (m)	Blade angle of twist (deg)
2000	4	NACA 0012	0.066	1.72	-8

Table 7.1: Main rotor characteristics used for the rotor model configuration

The characteristics used in this investigation for the main rotor are given in Table 7.1, and were taken from the experimental investigation carried out by Mineck et al. [14]. Mineck et al. [14] conducted a wind tunnel test of a generic helicopter fuselage model with an independently mounted rotor to obtain steady and periodic pressure data on the helicopter body. The model was tested at four advance ratios and three thrust coefficients. It was also useful to note that Mineck et al. [14] used a 2-meter rotor test system with a four bladed articulated rotor with no significant pitch-flap coupling. In the investigation it was found that the rotor wake induced changes in the steady pressure coefficients at the two lowest advance ratios as the wake flowed around the body.

Mineck et al. [14] also found that the unsteady pressure coefficients were marked by four peaks associated with the passage of the four rotor blades. Blade passage effects were largest on the nose and tail boom of the model. In addition the magnitude of the pulse increased with the rotor thrust coefficient.

Computational grid for the rotor-alone simulations

The computational grids for the rotor-alone simulations were created in the pre-processor, GAMBIT[20]. Apart from the structured actuator disk region, the computational domain was meshed using an unstructured tetrahedral grid. This allowed relatively simple manipulation of the grid density around the actuator disk region. The dimensions of the computational domain was representative of the 14 by 22 foot subsonic tunnel. The test section measures 6.63 meters wide and 4.42 meters feet high at the entrance and is 15.24 meters long. About 1.3 million grid elements were used to mesh the entire domain. As seen in Figure 7.1, mesh refinement occurs in locality of the actuator disk region, with the grid becoming increasingly coarse toward the domain boundaries.

As required by the rotor model code, the actuator disk region was meshed using a regular structured grid. This is shown in Figure 7.2. 50 elements were used in the radial direction and 100 equally spaced elements were used around the circumference, to make up an actuator disk containing 5000 cells. A axial thickness of 15 mm was used for the actuator, upstream and downstream disks, with an axial spacing of 30mm between the

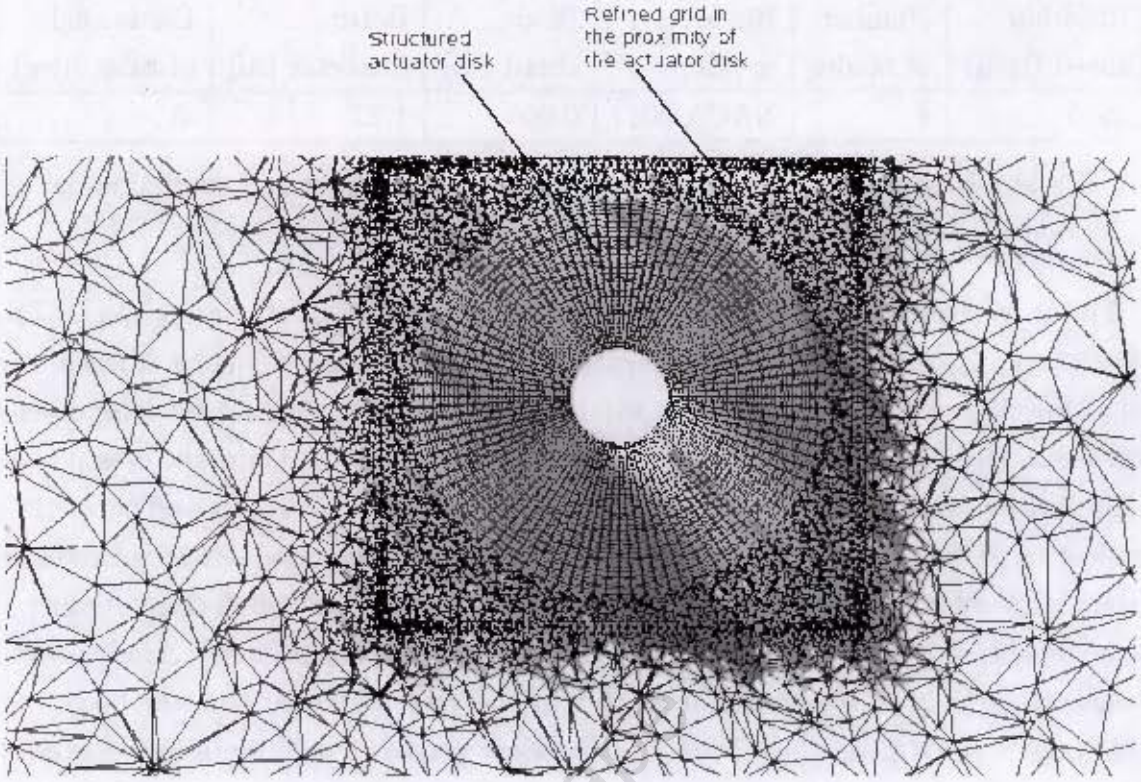

Figure 7.1: Plan view of the computational grid in the region of the actuator disk

disks.

Dimensions of the actuator disk were based on those used by Mineck et al. [14] in their investigation. In Figure 7.3, a section through the actuator disk region shows the different zones used in the rotor model.

Coning was not included in the creation of the computational grid on the rotor disk. This was mainly because values of coning were not available. These values are generally determined during experimental investigations.

Initial testing of the Rotor Model

Initial testing of the rotor model without the trim routine was carried out to ensure that the numerical rotor behaved in the correct manner. This initial testing was carried out on the rotor-alone model. The rotor was modelled in a surrounding region of atmosphere in a state of hover. The atmospheric outer region was created by assigning each of the boundary faces as pressure inlet boundaries.

A value of 5 degrees was assigned to the collective pitch angle, θ_0, whilst zero degrees were assigned to the cyclic pitch coefficients, A_1 and B_1. No angle of twist was assigned.

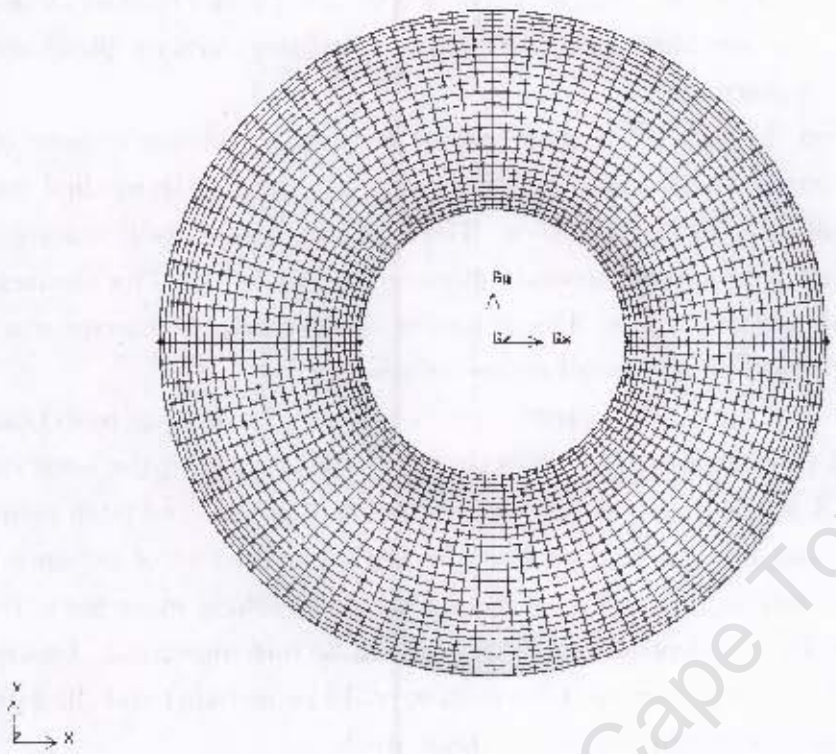

Figure 7.2: An actuator disk computational grid comprising of 5000 cells

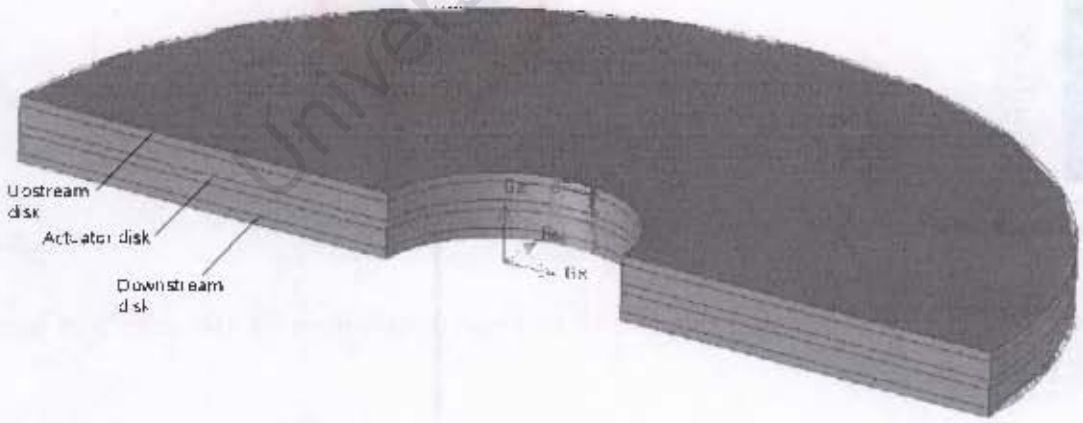

Figure 7.3: A section view through the actuator disk region, showing the disk-zones used

The above-mentioned values were assigned to check if the rotor would have zero moments about the hub and a non zero thrust value. This is the conditions that the physical rotor would have in a hover state.

In the CFD solver, FLUENT®, a single precision solver was chosen because of the computational limitations. The default, scaled settings of 10^{-4} for the residual convergence levels, was used for all the variables. The standard $k-\epsilon$ model was used for turbulence modelling. A first-order upwind difference scheme was used for the momentum and pressure convection terms. Discretisation of the turbulence terms was also accomplished using a first order upwind difference scheme.

The rotor behaved in the desired manner since a non zero thrust value was obtained, and the rolling and pitching moment values that were obtained from the rotor model code were within 2 % of being zero. A further verification of the roll and pitch moments being close to zero can also be seen by the symmetrical coefficient of pressure (C_p) distribution on the rotor in Figure 7.4. If the rolling and pitching moments were not close to zero, the coefficient of pressure (C_p) plot would be unsymmetrical. This result proved to be very favourable since the trim routine could be included and the forward flight simulations could be conducted to test the routine.

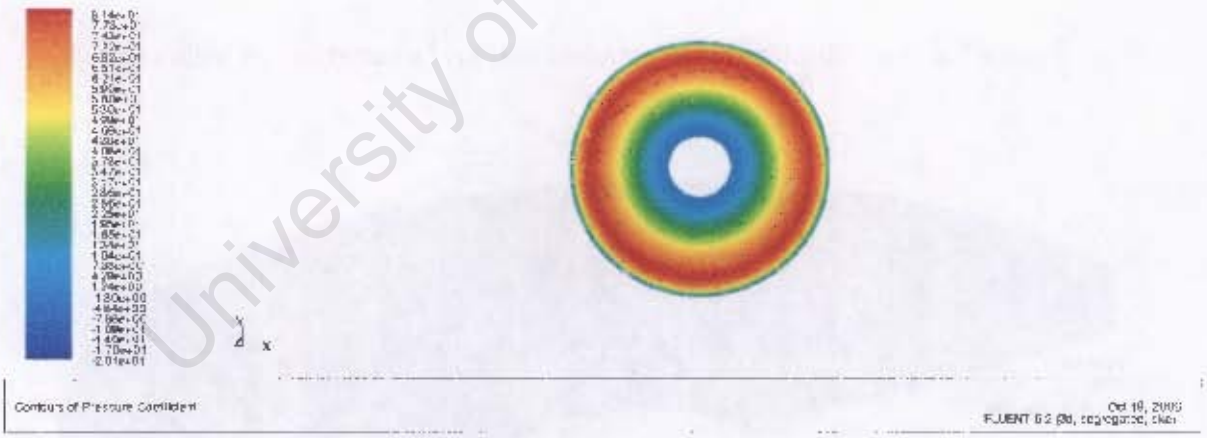

Figure 7.4: The symmetrical coefficient of pressure distribution for the rotor in a hovering condition

Trim routine verification for the rotor-alone model

To verify that the trim routine calculated needed to trim the rotor, several tests were conducted on the rotor-alone model for the forward flight condition. Forward flight is an extremely complex aerodynamic problem to model. This is because of the main rotor being subjected to proportionately large off-axis flow components when a helicopter is in forward flight. In addition, further complexities arise because of the reversed flow that may be experienced over a section of the retreating blades, particularly at high advance ratios.

A complete model was used for these simulations and the boundary conditions applied non slip walls on the top, bottom, and side domain boundaries with a constant velocity-inlet boundary at the front to enforce the forward flight velocity. The exit of the flow domain was set as a constant pressure-inlet boundary condition.

The simulations that took place used the same solver settings that were used for the initial testing and were carried out at the advance ratios (μ) and thrust coefficients C_T shown in Table 7.2

An advance ratio (μ) is a commonly used value used to describe the velocity of a helicopter with respect to its rotor speed. As seen in 7.8, μ is defined as the ratio of free-stream velocity (v_{inf}) and rotor blade tip speed (v_t).

$$\mu = \frac{v_{inf}}{v_t} \tag{7.8}$$

The conditions for the simulations were based on the experimental conditions used by Mineck et al. [14]. The values obtained by Mineck et al. [14] during their investigation could be used as a comparison. Initial guesses for θ_0, A_1, B_1 were obtained analytically with the methods used by Prouty [16]. Generally when the analytical values were utilised the correct trimming coefficients were reached within three simulations.

The θ_0, A_1, B_1 values used by Mineck et al. [14] to obtain trim conditions experimentally, as well as those obtained numerically in this investigation are displayed in the Tables 7.2 and 7.3 below.

μ	C_T	θ_0	A_1	B_1
0.051	0.00636	11.90	-1.30	1.30
0.051	0.00804	13.60	-1.30	1.40
0.151	0.00643	10.30	-2.70	2.40
0.151	0.00802	12.00	-2.90	2.50
0.232	0.00643	10.40	-0.40	3.80
0.232	0.00803	11.90	-1.30	4.0

Table 7.2: Minock et al. [14] experimental results for the numerical conditions used in the rotor-alone simulations

μ	C_T	θ_0	A_1	B_1
0.051	0.00636	12.10	-2.33	1.08
0.051	0.00804	13.92	-2.43	1.41
0.151	0.00643	9.82	-1.61	1.97
0.151	0.00802	11.28	-1.96	2.52
0.232	0.00643	9.57	-0.99	2.79
0.232	0.00803	11.07	-1.42	3.74

Table 7.3: Numerical results for the conditions used in the rotor-alone simulations

7.2.2 Rotor and Fuselage

In the simulations conducted for the rotor and fuselage configuration the same computational domain, boundary conditions, solver settings and rotor characteristics as those used for the rotor-alone, forward flight simulations were utilised. The center of the rotor, with respect to the fuselage, was located at $x/l = 0.696, y/l = 0.051, z/l = 0.322$. These coordinates duplicated those used by Mineck et al. [14] in their investigation.

Computational grid of the rotor and fuselage simulations

In the region surrounding the helicopter fuselage and actuator disk, sharp gradients in flow properties were expected due to the rotor down wash and flow interactions around the body-structure. A size function was therefore used to refine the unstructured mesh size in this critical region, and to control the rate at which the grid would become progressively courser. The grid became courser as it moved, outwards, toward the domain boundaries. The maximum cell size was thus at the domain boundaries. The grid used is demonstrated in Figure 7.5, showing the ROBIN and structured actuator disk region embedded in an unstructured mesh.

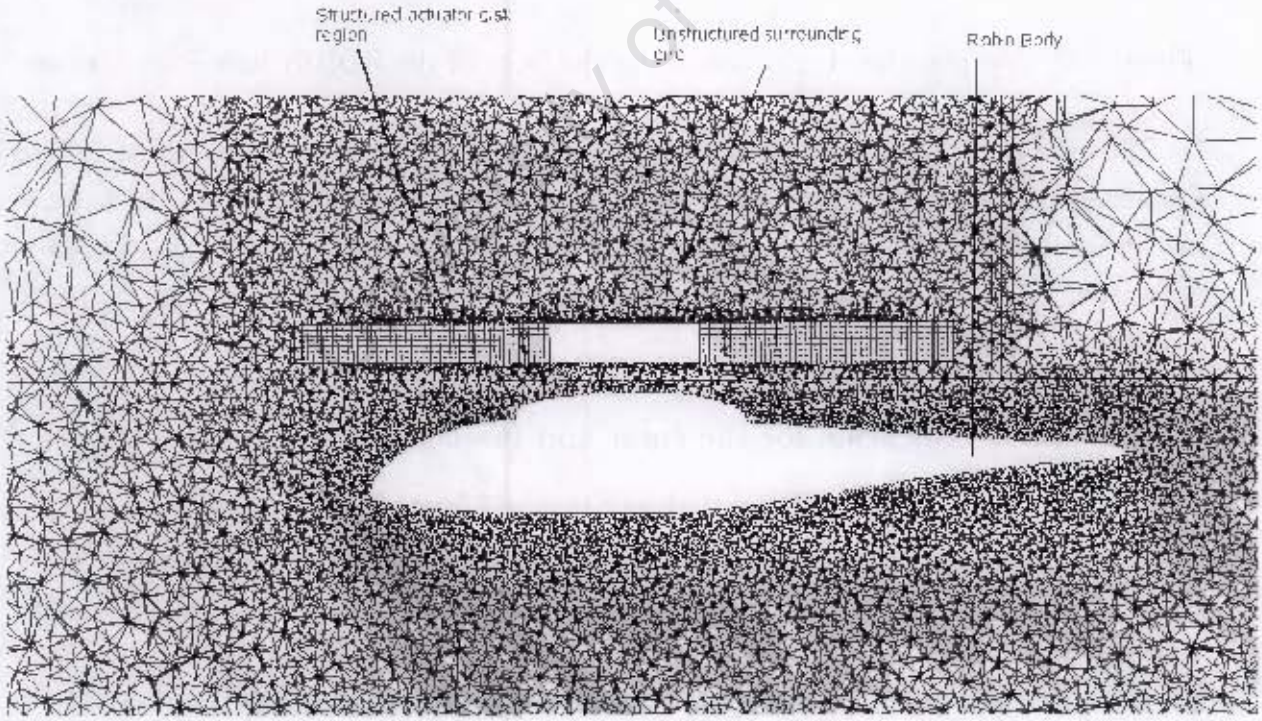

Figure 7.5: Detailed view of the computational grid around the helicopter fuselage, showing the structured actuator disk region embedded within an unstructured grid

An initial grid size of 150 mm was specified for meshing the helicopter fuselage (see Figure 7.6), using a growth rate of 1.05 and a maximum elements size of 600 mm. The growth rate defines the size ratio of the current row of elements to the previous row's elements. A total number of about 2 million cells was used to mesh the entire domain.

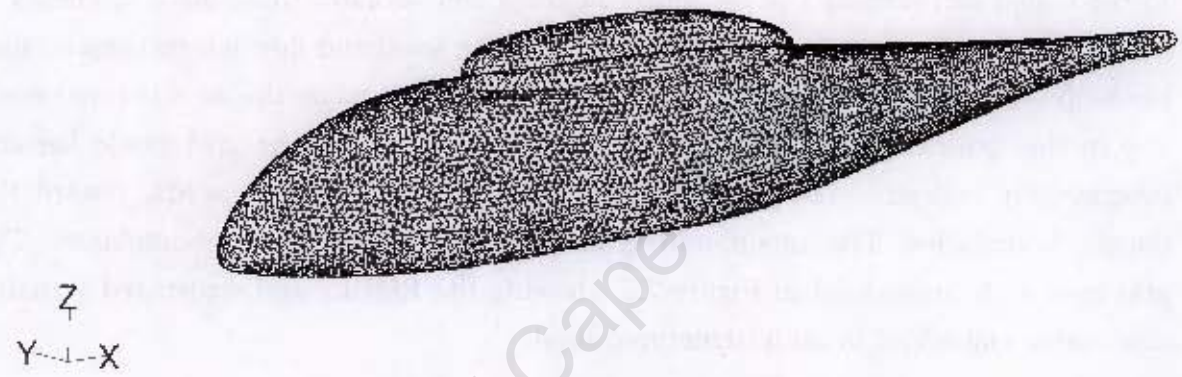

Figure 7.6: Computational grid describing the faces of the ROBIN helicopter fuselage

The use of size functions allowed greater control over the computational grid. This permitted the total number of grid elements to be reduced significantly, without necessarily sacrificing computational accuracy and stability, since the grid could be refined in critical regions, while retaining a course structure in the majority of the computational domain.

Trim routine verification for the rotor and fuselage model

Due to the computational expense that was required for the rotor and fuselage simulations only three verification simulations were conducted. The simulations at the thrust coefficients and advance ratios in Table 7.4 were conducted. The θ_0, A_1, B_1 values that were used to trim the rotor for the rotor and fuselage model, and rotor-alone simulations are also displayed Table 7.4, along with the experimental values used by Mineck et al [14]. About 4 or 5 iterations of the trim routine were needed to balance the rotor for the rotor and fuselage simulations. Generally the rolling moment and thrust coefficient were balanced within three simulations, but were out of balance in pitch.

μ	C_T	Experimental	Isolated Rotor	Rotor and Fusealage
0.051	0.00636	θ_0 = 11.90	θ_0 = 12.10	θ_0 = 12.10
		A_1 = -1.30	A_1 = -2.33	A_1 = -2.33
		B_1 = 1.30	B_1 = 1.08	B_1 = 1.08
0.151	0.00643	θ_0 = 10.30	θ_0 = 9.82	θ_0 = 9.84
		A_1 = -2.70	A_1 = -1.61	A_1 = -1.86
		B_1 = 2.40	B_1 = 1.97	B_1 = 1.99
0.231	0.00657	θ_0 = 10.40	θ_0 = 9.57	θ_0 = 9.54
		A_1 = -0.40	A_1 = -0.99	A_1 = -1.45
		B_1 = 3.80	B_1 = 2.79	B_1 = 2.75

Table 7.4: A comparison of the experimental and numerical results for both the validation cases at the test conditions of the rotor and fuselage simulations

7.2.3 Results and Discussion

Even though the experimental values were available for the θ_0, A_1, B_1, the visual aerodynamic effects were unclear in the documentation acquired. This was because the documentation received was in black and white, and the effects could only be seen on colour. Therefore only the numerical θ_0, A_1, B_1 values could be used for evaluation and discussion.

For the rotor-alone simulations, rolling and pitch moments errors were found to be below 1.0%, when compared to the desired values, and the thrust error was found to be below 2.0% for the conditions specified. A reasonably good agreement was found to exist between the experimental and numerical results for the rotor-alone simulations.

On the other hand, for the rotor and fuselage simulations the rolling and pitch moments errors were found to be below 2.4% and the thrust error was found to be below 4.1%. As seen in Table 7.4, the effect of the fuselage on θ_0, A_1, B_1 needed to trim the rotor, is not large. The difference between θ_0, A_1, B_1 of the rotor-alone and, rotor and fuselage simulations were found to be greater at higher advance ratios.

The difference in the experimental and numerical values can be attributed to the lack of coning. It must also be remembered that in a physical helicopter configuration, the hub in the center of the rotor rotates. Rotation of the hub was not accommodated for in the rotor model, which means that the effects that it has on the flow-field surrounding is not accounted for. Theoretically the rotor wake should also influence the trimming of the rotor, and thus on the difference in results.

Chapter 8

Summary and conclusions

This report has presented the development, validation and implementation of a trim routine for a numerical rotor model, developed for use in simulations of a helicopter exterior flow-field.

In Chapter 1, the reason, approach and limitations for this investigation were presented. A historical discussion of the methods used to model rotary-wing flow fields, leading up to the current momentum source technique used to model the effects of rotor, was depicted. The need for the implementation of the trim routine was also revealed.

A literature review was presented in Chapter 2. The equations that govern the fluid flow around a helicopter configuration were presented, along with the methods used to model turbulence in CFD. A depiction of forward flight helicopter aerodynamics was also shown. Finally brief descriptions of the trim routine implementations used in previous investigations were also revealed.

In Chapter 3, the fuselage and external geometry created for this investigation were described. The method in which the geometry was created was also described. Chapter 4 presented the computational grids created around the helicopter fuselage. The grid used, choices made, and methods used to create the grids were also described.

Evaluation of the turbulence models and the numerical modeling strategy for the helicopter fuselage simulations were described in Chapter 5. First order discretisation schemes were found to be acceptable discretisation choices for the pressure, convection and turbulence terms. Turbulence models were compared for grids that utilised both wall function approach, and near wall modelling. For the wall function approach grids, the $k-\epsilon$, and its variants, the RNG and realizable models were found to be suitable choices. For the range of turbulence models tested, for the near wall modelling approach grids used, the SST models performed the best.

Chapter 6 presented a mathematical description of the methodology used for the rotor model, which was implemented through the use of user-defined functions, written in the C programming language.

In Chapter 7, the trim routine used to balance the rotor in forward flight conditions was described and tested. The method used to balance the rotor was the Newton-Raphson Iterative method. Tests were conducted, initially on a rotor-alone model because of the computational restrictions that were enforced by the rotor model. When compared to the desired values, the collective pitch angle and cyclic pitch coefficients for these tests compared favourably. Good agreements between the numerical and experimental values were also found. When the trim routine was tested on the rotor and fuselage model, the effect of the fuselage on the collective pitch angle and cyclic pitch coefficients proved to minimal.

Although the restrictions prevented a realistic aerodynamic comparison of the numerical and experimental data, it can be concluded that the trim routine implemented was able to trim the rotor successfully. This finding was based on outputs of the rotor model code. However, greater verification of the helicopter aerodynamics needs to be conducted.

Chapter 9

Recommendations

The following recommendations are made for future investigations:

9.1 Parallelise the rotor model

One of the shortcomings of the rotor model was that it could only be used on a single processor. This presents a very large computational restriction. It is therefore, recommended that the rotor model be altered, so that it can be run on more than one processor, and so that it can accommodate more than one rotor per simulation.

9.2 Acquire greater computational resources

To, fully, accurately predict the aerodynamics of the flow-field surrounding the helicopter fuselage, greater computational resources will be needed to generate grids with more elements in regions of interest, and to computationally save time in solving the flow.

9.3 Conduct an experimental investigation

To completely validate the accuracy of CFD in predicting the flow variations and aerodynamic forces experimental work needs to be carried out. Creation of a physical model and wind tunnel testing is thus advised.

9.4 Include coning and hub effects

When the actuator disk grid is created it is recommended that the value of coning be taken into consideration. However, these coning values will have to be determined or obtained from experimental investigations. A spinning hub should also be included on the geometry so that its effects on the flow-field can be accounted for. It would be possible to alter the rotor model and geometry so that it can account for the rotation and aerodynamic loading due to the hub.

9.5 Investigate the use of linear techniques for the trim routine

Since a possible linear relationship was found to exist between the pitch angles and the coefficients of thrust, and moments it is recommended that a linear technique be investigated as a possible alternative for rotor trim, as one of these methods may be found to be more suitable.

References

[1] GAMBIT 2.2. *Users Guide.* Fluent Inc, Jan 2005.

[2] TGrid 3.6. *Users Guide.* Fluent Inc, Jan 2005.

[3] FLUENT 6.2. *Users Guide.* Fluent Inc, Jan 2005.

[4] M.S. Chaffin and J.D. Berry. Navier-stokes simulation of a rotor using a distributed pressure disk method. *Proceedings of the American Helicopter Society, 51st Annual Forum, Fort Worth, TX*, 1995.

[5] I. Fetjek and L. Roberts. Navier-stokes computation of wing/rotor interaction for a tilt rotor in hover. *AIAAJ.*, 30(11):2595–2603, 1992.

[6] C.E. Freeman and R.E. Mineck. Fuselage surface pressure measurements of a helicopter wind tunnel model with a 3.15-meter diameter single rotor. *NASA TM-80051*, 1979.

[7] Versteeg H.K. and Malalasekera. *An introduction to Computational Fluid Dynamics.* Prentice Hall, Harlow, 1995.

[8] S.F. Hoerner. Fluid-dynamic drag. *Published by the author*, 1965.

[9] S.F. Hoerner and Borst H.V. Fluid-dynamic lift. *Published by Mrs. L.A. Hoerner*, 1975.

[10] P.J. Hotchkiss. Development of a rotor model for the numerical simulation of helicopter exterior flow-fields. Master's thesis, University of Cape Town, 2004.

[11] B.E. Launder and Spalding D.B. The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 3:269–289, 1974.

[12] F.R. Menter. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA journal*, 32(8):1598–1605, 1994.

[13] C.J. Meyer and D.G. Kröger. Numerical simulation of the flow field in the vicinity of an axial flow fan. *International Journal for Numerical Methods in Fluids*, 36:947–969, 2001.

[14] R.E. Mineck and Gorton S.A. Steady and periodic pressure measurements on a generic helicopter fuselage model in the presence of a rotor. *NASA/TM-2000-210286*, 2000.

[15] U.S. Centennial of Flight Commission. `http://www.centennialofflight.gov/essay/Rotary/Sikorsky_VS300/HE8.htm`. Last accessed on: 16 October 2006.

[16] R.W. Prouty. *Helicopter Performance, Stability, and Control.* Krieger Publishing Company, Malabar, Florida, 1995.

[17] R.G. Rajagopalan and S.J. Mathur. Three dimensional analysis of a rotor in forward flight. *J.Amer.Helicopter Soc.*, 1993.

[18] M.R. Ruith. Unstructured, multiplex rotor source model with thrust and moment trimming-fluent's vbm model. *AIAA-2005-5217; AIAA: Reston, VA*, 2005.

[19] W.H. Stinnes and T.W. von Backström. Effect of cross-flow on the performance of air-cooled heat exchanger fan. *Applied Thermal Engineering*, 22, 2002.

[20] R. Von Mises. Theory of flight. *McGraw-Hill Book Company Inc, London*, 1945.

[21] D.C. Wilcox. Turbulence modeling for cfd. *DCW Industries, Inc., La Canada, California*, 1998.

[22] Z. Yang, L.N. Sankar, M. Smith, and O. Bauchau. Recent improvements to a hybrid method for rotors in forward flight. *AIAA-2000-260; AIAA: Reston, VA*, 2000.

[23] L.A.J. Zori and R.G. Rajagopalan. Navier-stokes calculation of rotor airframe in forward forward flight. *J.Amer.Helicopter Soc.*, 40, 1995.

Appendix A

The coordinates of the ROBIN configuration used during this investigation is defined by super-ellipse equations. It consists of an analytically defined body representing the fuselage, and an analytically defined pylon representing the fairings around the engines and transmission. The exact form of these equations is shown in Chapter 3.

This chapter, therefore, deals with the code that was written in Matlab to create the coordinates of the ROBIN configuration. Flowcharts describing the logic of the routines are attached, along with the outputs of the code. A copy of the code used is also indicated in this appendix.

A.1 Software Description

The software is comprised of a main program (main.m), several function files (pylonone.m, pylontwo.m, robinone.m, robintwo.m, robinthree.m, robinfour.m) to generate the coordinates of the robin configuration, and another function file (calcval.m) which generates the values for the the analytical functions of the model height (H), width (W), camber (Z_0), and elliptical power (N). Only the main file needs to be run to generate the outputs, since the function files merely completes intermediate tasks needed to generate the coordinates.

A.2 Flowcharts

A generic flowchart is indicated for the function files needed to generate the coordinates of the robin configuration. This is because the same method is applied in each of these function files. Flowcharts for main.m and calcval.m are also shown

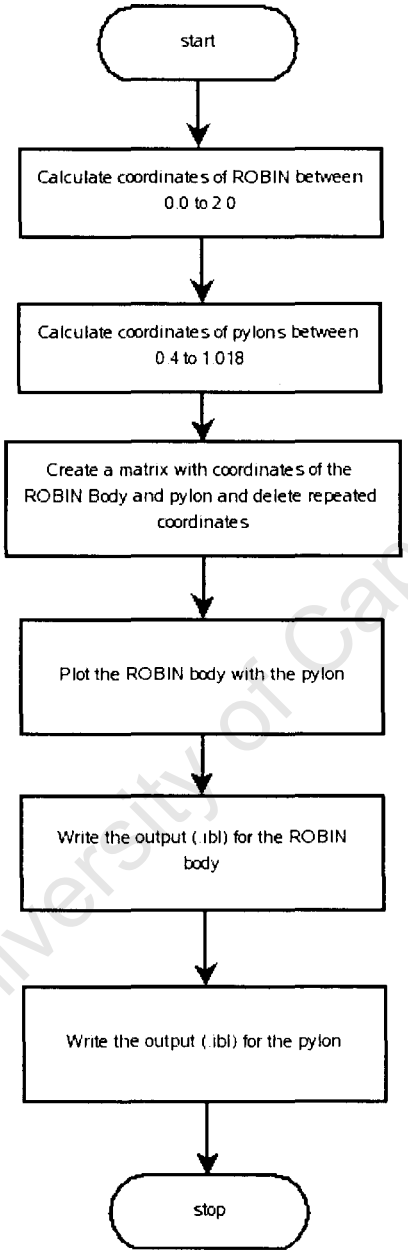

Figure A.1: Flowchart depicting the structure of main.m

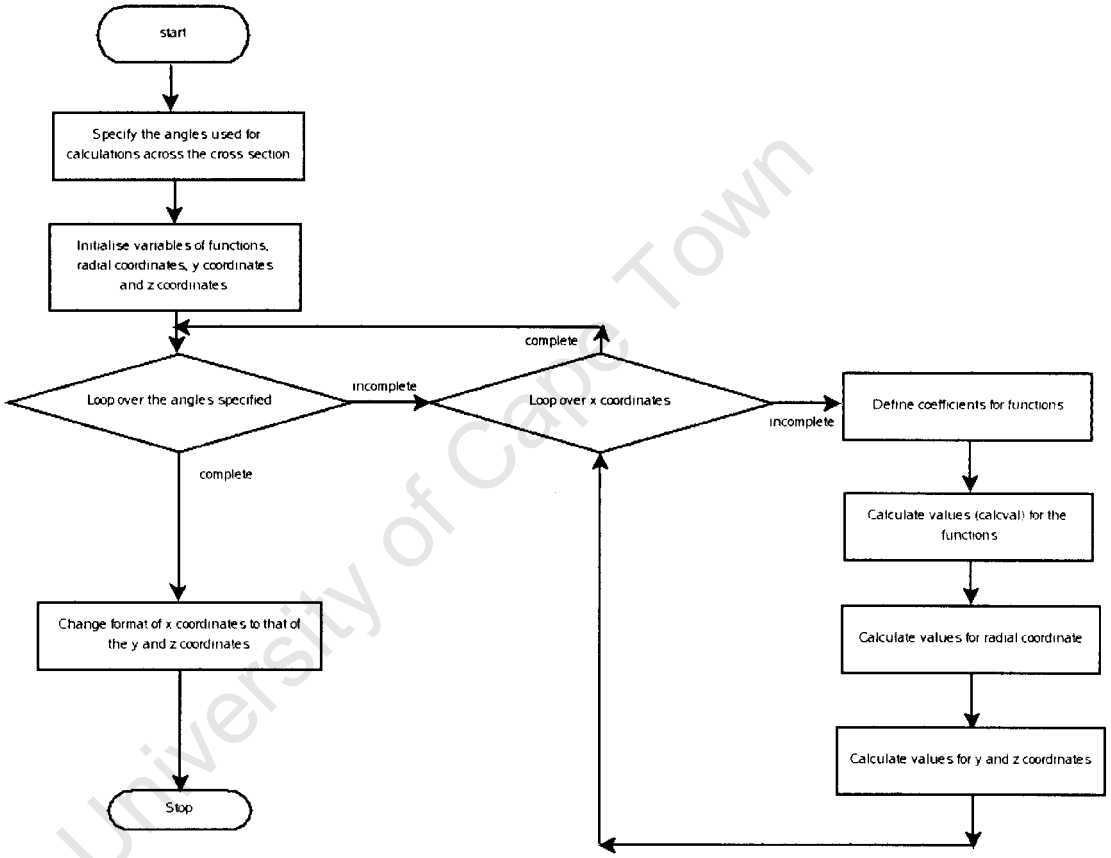

Figure A.2: Flowchart depicting the structure of function files needed to generate the coordinates

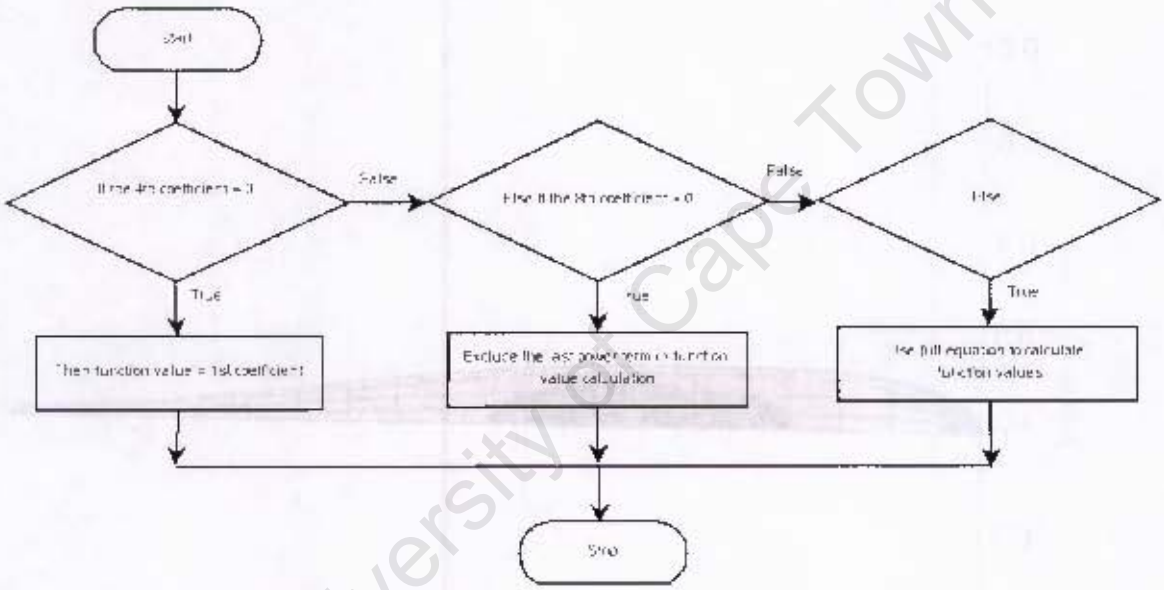

Figure A.3: Flowchart depicting the structure of function file calcval.m

A.3 Outputs of the code

A figure depicting the shape of the ROBIN configuration is an output of the code. The code also outputs the cross sectional coordinates of the ROBIN body and pylon in a data (.IBL) file. This was carried out so that the data files could be the imported by the CAD package ProEngineer. Having the coordinates being imported allowed simpler geometry creation in ProEngineer. Thus, in this section a figure depicting the ROBIN helicopter fuselage is shown. The cross sectional coordinates for the ROBIN body and pylon in the (.IBL) format is also depicted.

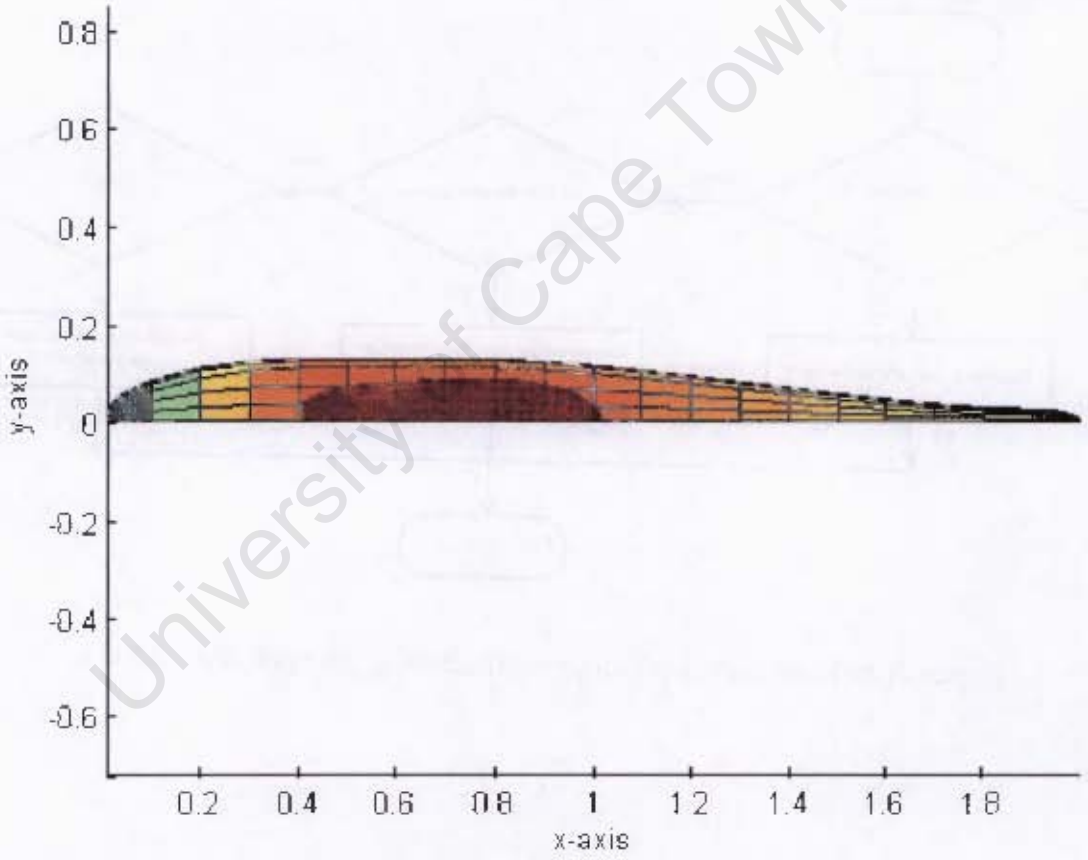

Figure A.4: The robin configuration as it is outputs from the Matlab code written

A.3.1 Pylon coordinates in the .IBL format

```
Open Index Arclength
 Begin section !   1
 Begin curve !    1
    0.410000      0.000000     0.155320
    0.410000      0.008123     0.155315
    0.410000      0.017393     0.155126
    0.410000      0.027926     0.152926
    0.410000      0.033896     0.144570
    0.410000      0.034692     0.134296
    0.410000      0.034711     0.125000
    0.410000      0.034692     0.115704
    0.410000      0.033896     0.105430
    0.410000      0.027926     0.097074
    0.410000      0.017393     0.094874
    0.410000      0.008123     0.094685
    0.410000      0.000000     0.094680
 Begin section !   2
 Begin curve !    2
    0.420000      0.000000     0.162879
    0.420000      0.010148     0.162874
    0.420000      0.021730     0.162637
    0.420000      0.034889     0.159889
    0.420000      0.042347     0.149449
    0.420000      0.043342     0.136613
    0.420000      0.043365     0.125000
    0.420000      0.043342     0.113387
    0.420000      0.042347     0.100551
    0.420000      0.034889     0.090111
    0.420000      0.021730     0.087363
    0.420000      0.010148     0.087126
    0.420000      0.000000     0.087121
 Begin section !   3
 Begin curve !    3
```

```
   0.430000      0.000000      0.167994
   0.430000      0.011519      0.167988
   0.430000      0.024664      0.167719
   0.430000      0.039600      0.164600
   0.430000      0.048065      0.152750
   0.430000      0.049194      0.138181
   0.430000      0.049220      0.125000
   0.430000      0.049194      0.111819
   0.430000      0.048065      0.097250
   0.430000      0.039600      0.085400
   0.430000      0.024664      0.082281
   0.430000      0.011519      0.082012
   0.430000      0.000000      0.082006
Begin section !   4
Begin curve !    4
   0.440000      0.000000      0.171917
   0.440000      0.012570      0.171910
   0.440000      0.026914      0.171616
   0.440000      0.043213      0.168213
   0.440000      0.052450      0.155282
   0.440000      0.053682      0.139384
   0.440000      0.053712      0.125000
   0.440000      0.053682      0.110616
   0.440000      0.052450      0.094718
   0.440000      0.043213      0.081787
   0.440000      0.026914      0.078384
   0.440000      0.012570      0.078090
   0.440000      0.000000      0.078083
Begin section !   5
Begin curve !    5
   0.450000      0.000000      0.175105
   0.450000      0.013424      0.175097
   0.450000      0.028743      0.174784
   0.450000      0.046150      0.171150
   0.450000      0.056014      0.157340
```

```
   0.450000      0.057330      0.140362
   0.450000      0.057361      0.125000
   0.450000      0.057330      0.109638
   0.450000      0.056014      0.092660
   0.450000      0.046150      0.078850
   0.450000      0.028743      0.075216
   0.450000      0.013424      0.074903
   0.450000      0.000000      0.074895
Begin section !   6
Begin curve !    6
   0.460000      0.000000      0.177782
   0.460000      0.014141      0.177775
   0.460000      0.030279      0.177444
   0.460000      0.048616      0.173616
   0.460000      0.059008      0.159068
   0.460000      0.060394      0.141182
   0.460000      0.060426      0.125000
   0.460000      0.060394      0.108818
   0.460000      0.059008      0.090932
   0.460000      0.048616      0.076384
   0.460000      0.030279      0.072556
   0.460000      0.014141      0.072225
   0.460000      0.000000      0.072218
Begin section !   7
Begin curve !    7
   0.470000      0.000000      0.180079
   0.470000      0.014756      0.180072
   0.470000      0.031597      0.179727
   0.470000      0.050732      0.175732
   0.470000      0.061576      0.160551
   0.470000      0.063022      0.141887
   0.470000      0.063056      0.125000
   0.470000      0.063022      0.108113
   0.470000      0.061576      0.089449
   0.470000      0.050732      0.074268
```

```
   0.470000      0.031597      0.070273
   0.470000      0.014756      0.069928
   0.470000      0.000000      0.069921
Begin section !   8
Begin curve !    8
   0.480000      0.000000      0.182079
   0.480000      0.015292      0.182071
   0.480000      0.032744      0.181714
   0.480000      0.052574      0.177574
   0.480000      0.063811      0.161842
   0.480000      0.065310      0.142500
   0.480000      0.065346      0.125000
   0.480000      0.065310      0.107500
   0.480000      0.063811      0.088158
   0.480000      0.052574      0.072426
   0.480000      0.032744      0.068286
   0.480000      0.015292      0.067929
   0.480000      0.000000      0.067921
Begin section !   9
Begin curve !    9
   0.490000      0.000000      0.183838
   0.490000      0.015763      0.183830
   0.490000      0.033753      0.183462
   0.490000      0.054194      0.179194
   0.490000      0.065778      0.162977
   0.490000      0.067323      0.143039
   0.490000      0.067359      0.125000
   0.490000      0.067323      0.106961
   0.490000      0.065778      0.087023
   0.490000      0.054194      0.070806
   0.490000      0.033753      0.066538
   0.490000      0.015763      0.066170
   0.490000      0.000000      0.066162
Begin section !   10
Begin curve !    10
```

```
   0.500000      0.000000      0.185397
   0.500000      0.016181      0.185388
   0.500000      0.034647      0.185010
   0.500000      0.055629      0.180629
   0.500000      0.067520      0.163983
   0.500000      0.069106      0.143517
   0.500000      0.069144      0.125000
   0.500000      0.069106      0.106483
   0.500000      0.067520      0.086017
   0.500000      0.055629      0.069371
   0.500000      0.034647      0.064990
   0.500000      0.016181      0.064612
   0.500000      0.000000      0.064603
Begin section !   11
Begin curve !     11
   0.600000      0.000000      0.194344
   0.600000      0.018578      0.194334
   0.600000      0.039779      0.193900
   0.600000      0.063870      0.188870
   0.600000      0.077523      0.169758
   0.600000      0.079344      0.146260
   0.600000      0.079387      0.125000
   0.600000      0.079344      0.103740
   0.600000      0.077523      0.080242
   0.600000      0.063870      0.061130
   0.600000      0.039779      0.056100
   0.600000      0.018578      0.055666
   0.600000      0.000000      0.055656
Begin section !   12
Begin curve !     12
   0.700000      0.000000      0.197120
   0.700000      0.019322      0.197110
   0.700000      0.041372      0.196659
   0.700000      0.066428      0.191428
   0.700000      0.080627      0.171550
```

```
   0.700000      0.082521      0.147111
   0.700000      0.082565      0.125000
   0.700000      0.082521      0.102889
   0.700000      0.080627      0.078450
   0.700000      0.066428      0.058572
   0.700000      0.041372      0.053341
   0.700000      0.019322      0.052890
   0.700000      0.000000      0.052880
Begin section !   13
Begin curve !    13
   0.800000      0.000000      0.197500
   0.800000      0.019424      0.197490
   0.800000      0.041590      0.197036
   0.800000      0.066777      0.191777
   0.800000      0.081051      0.171795
   0.800000      0.082955      0.147228
   0.800000      0.083000      0.125000
   0.800000      0.082955      0.102772
   0.800000      0.081051      0.078205
   0.800000      0.066777      0.058223
   0.800000      0.041590      0.052964
   0.800000      0.019424      0.052510
   0.800000      0.000000      0.052500
Begin section !   14
Begin curve !    14
   0.825000      0.000000      0.196679
   0.825000      0.019295      0.196669
   0.825000      0.041316      0.196219
   0.825000      0.066337      0.190994
   0.825000      0.080516      0.171144
   0.825000      0.082408      0.146739
   0.825000      0.082452      0.124658
   0.825000      0.082408      0.102577
   0.825000      0.080516      0.078172
   0.825000      0.066337      0.058321
```

```
   0.825000      0.041316      0.053097
   0.825000      0.019295      0.052646
   0.825000      0.000000      0.052636
Begin section !   15
Begin curve !     15
   0.850000      0.000000      0.194601
   0.850000      0.018906      0.194591
   0.850000      0.040481      0.194150
   0.850000      0.064997      0.189031
   0.850000      0.078890      0.169581
   0.850000      0.080744      0.145669
   0.850000      0.080787      0.124034
   0.850000      0.080744      0.102399
   0.850000      0.078890      0.078487
   0.850000      0.064997      0.059037
   0.850000      0.040481      0.053918
   0.850000      0.018906      0.053477
   0.850000      0.000000      0.053467
Begin section !   16
Begin curve !     16
   0.875000      0.000000      0.191305
   0.875000      0.018238      0.191295
   0.875000      0.039051      0.190869
   0.875000      0.062701      0.185931
   0.875000      0.076103      0.167169
   0.875000      0.077891      0.144101
   0.875000      0.077933      0.123230
   0.875000      0.077891      0.102359
   0.875000      0.076103      0.079292
   0.875000      0.062701      0.060529
   0.875000      0.039051      0.055592
   0.875000      0.018238      0.055165
   0.875000      0.000000      0.055156
Begin section !   17
Begin curve !     17
```

```
   0.900000      0.000000      0.186706
   0.900000      0.017259      0.186697
   0.900000      0.036956      0.186294
   0.900000      0.059337      0.181621
   0.900000      0.072021      0.163865
   0.900000      0.073712      0.142035
   0.900000      0.073752      0.122284
   0.900000      0.073712      0.102533
   0.900000      0.072021      0.080703
   0.900000      0.059337      0.062947
   0.900000      0.036956      0.058274
   0.900000      0.017259      0.057871
   0.900000      0.000000      0.057862
Begin section !   18
Begin curve !    18
   0.925000      0.000000      0.180616
   0.925000      0.015913      0.180608
   0.925000      0.034074      0.180236
   0.925000      0.054709      0.175928
   0.925000      0.066403      0.159556
   0.925000      0.067963      0.139429
   0.925000      0.068000      0.121218
   0.925000      0.067963      0.103008
   0.925000      0.066403      0.082880
   0.925000      0.054709      0.066509
   0.925000      0.034074      0.062201
   0.925000      0.015913      0.061829
   0.925000      0.000000      0.061821
Begin section !   19
Begin curve !    19
   0.950000      0.000000      0.172658
   0.950000      0.014095      0.172651
   0.950000      0.030179      0.172322
   0.950000      0.048456      0.168506
   0.950000      0.058814      0.154006
```

```
   0.950000      0.060196      0.136179
   0.950000      0.060228      0.120049
   0.950000      0.060196      0.103920
   0.950000      0.058814      0.086093
   0.950000      0.048456      0.071593
   0.950000      0.030179      0.067777
   0.950000      0.014095      0.067448
   0.950000      0.000000      0.067440
Begin section !   20
Begin curve !    20
   0.975000      0.000000      0.162022
   0.975000      0.011583      0.162016
   0.975000      0.024801      0.161746
   0.975000      0.039820      0.158610
   0.975000      0.048332      0.146694
   0.975000      0.049467      0.132044
   0.975000      0.049494      0.118790
   0.975000      0.049467      0.105535
   0.975000      0.048332      0.090885
   0.975000      0.039820      0.078970
   0.975000      0.024801      0.075834
   0.975000      0.011583      0.075563
   0.975000      0.000000      0.075557
Begin section !   21
Begin curve !    21
   1.000000      0.000000      0.146297
   1.000000      0.007729      0.146293
   1.000000      0.016548      0.146112
   1.000000      0.026570      0.144020
   1.000000      0.032250      0.136069
   1.000000      0.033007      0.126294
   1.000000      0.033025      0.117449
   1.000000      0.033007      0.108605
   1.000000      0.032250      0.098830
   1.000000      0.026570      0.090879
```

```
   1.000000      0.016548      0.088787
   1.000000      0.007729      0.088606
   1.000000      0.000000      0.088602
Begin section !   22
Begin curve !    22
   1.001800      0.000000      0.144776
   1.001800      0.007348      0.144772
   1.001800      0.015733      0.144600
   1.001800      0.025261      0.142611
   1.001800      0.030661      0.135052
   1.001800      0.031381      0.125758
   1.001800      0.031398      0.117350
   1.001800      0.031381      0.108942
   1.001800      0.030661      0.099648
   1.001800      0.025261      0.092089
   1.001800      0.015733      0.090100
   1.001800      0.007348      0.089928
   1.001800      0.000000      0.089924
Begin section !   23
Begin curve !    23
   1.003600      0.000000      0.143163
   1.003600      0.006942      0.143159
   1.003600      0.014865      0.142997
   1.003600      0.023867      0.141118
   1.003600      0.028969      0.133976
   1.003600      0.029650      0.125195
   1.003600      0.029666      0.117250
   1.003600      0.029650      0.109306
   1.003600      0.028969      0.100525
   1.003600      0.023867      0.093383
   1.003600      0.014865      0.091503
   1.003600      0.006942      0.091341
   1.003600      0.000000      0.091338
Begin section !   24
Begin curve !    24
```

```
   1.005400      0.000000      0.141441
   1.005400      0.006508      0.141438
   1.005400      0.013935      0.141286
   1.005400      0.022373      0.139524
   1.005400      0.027156      0.132829
   1.005400      0.027794      0.124597
   1.005400      0.027809      0.117150
   1.005400      0.027794      0.109703
   1.005400      0.027156      0.101472
   1.005400      0.022373      0.094777
   1.005400      0.013935      0.093015
   1.005400      0.006508      0.092863
   1.005400      0.000000      0.092859
Begin section !   25
Begin curve !    25
   1.007200      0.000000      0.139586
   1.007200      0.006038      0.139583
   1.007200      0.012928      0.139442
   1.007200      0.020758      0.137807
   1.007200      0.025195      0.131596
   1.007200      0.025787      0.123959
   1.007200      0.025801      0.117050
   1.007200      0.025787      0.110140
   1.007200      0.025195      0.102503
   1.007200      0.020758      0.096292
   1.007200      0.012928      0.094657
   1.007200      0.006038      0.094516
   1.007200      0.000000      0.094513
Begin section !   26
Begin curve !    26
   1.009000      0.000000      0.137565
   1.009000      0.005523      0.137563
   1.009000      0.011827      0.137434
   1.009000      0.018989      0.135938
   1.009000      0.023048      0.130256
```

```
   1.009000      0.023590      0.123270
   1.009000      0.023602      0.116949
   1.009000      0.023590      0.110628
   1.009000      0.023048      0.103642
   1.009000      0.018989      0.097960
   1.009000      0.011827      0.096464
   1.009000      0.005523      0.096335
   1.009000      0.000000      0.096332
Begin section !   27
Begin curve !    27
   1.010800      0.000000      0.135327
   1.010800      0.004951      0.135324
   1.010800      0.010600      0.135208
   1.010800      0.017020      0.133868
   1.010800      0.020658      0.128775
   1.010800      0.021144      0.122513
   1.010800      0.021155      0.116848
   1.010800      0.021144      0.111182
   1.010800      0.020658      0.104921
   1.010800      0.017020      0.099827
   1.010800      0.010600      0.098487
   1.010800      0.004951      0.098371
   1.010800      0.000000      0.098369
Begin section !   28
Begin curve !    28
   1.012600      0.000000      0.132783
   1.012600      0.004296      0.132781
   1.012600      0.009200      0.132680
   1.012600      0.014771      0.131517
   1.012600      0.017928      0.127097
   1.012600      0.018349      0.121663
   1.012600      0.018359      0.116746
   1.012600      0.018349      0.111829
   1.012600      0.017928      0.106395
   1.012600      0.014771      0.101975
```

```
   1.012600      0.009200      0.100812
   1.012600      0.004296      0.100712
   1.012600      0.000000      0.100709
Begin section !   29
Begin curve !    29
   1.014400      0.000000      0.129766
   1.014400      0.003515      0.129764
   1.014400      0.007527      0.129682
   1.014400      0.012086      0.128730
   1.014400      0.014669      0.125113
   1.014400      0.015013      0.120667
   1.014400      0.015022      0.116644
   1.014400      0.015013      0.112621
   1.014400      0.014669      0.108175
   1.014400      0.012086      0.104559
   1.014400      0.007527      0.103607
   1.014400      0.003515      0.103525
   1.014400      0.000000      0.103523
Begin section !   30
Begin curve !    30
   1.016200      0.000000      0.125839
   1.016200      0.002491      0.125838
   1.016200      0.005334      0.125780
   1.016200      0.008564      0.125106
   1.016200      0.010394      0.122543
   1.016200      0.010638      0.119393
   1.016200      0.010644      0.116542
   1.016200      0.010638      0.113692
   1.016200      0.010394      0.110541
   1.016200      0.008564      0.107979
   1.016200      0.005334      0.107304
   1.016200      0.002491      0.107246
   1.016200      0.000000      0.107245
```

A.3.2 ROBIN coordinates in the .IBL format

```
Open Index Arclength
 Begin section !   1
 Begin curve !    1
    0.010000      0.000000    -0.043599
    0.010000      0.005834    -0.044023
    0.010000      0.011733    -0.045473
    0.010000      0.017550    -0.048245
    0.010000      0.022777    -0.052645
    0.010000      0.026487    -0.058698
    0.010000      0.027776    -0.065795
    0.010000      0.026487    -0.072892
    0.010000      0.022777    -0.078945
    0.010000      0.017550    -0.083345
    0.010000      0.011733    -0.086116
    0.010000      0.005834    -0.087566
    0.010000      0.000000    -0.087990
 Begin section !   2
 Begin curve !    2
    0.020000      0.000000    -0.026803
    0.020000      0.008536    -0.027384
    0.020000      0.017164    -0.029512
    0.020000      0.025541    -0.033699
    0.020000      0.032763    -0.040324
    0.020000      0.037534    -0.049183
    0.020000      0.039031    -0.059240
    0.020000      0.037534    -0.069297
    0.020000      0.032763    -0.078156
    0.020000      0.025541    -0.084781
    0.020000      0.017164    -0.088969
    0.020000      0.008536    -0.091097
    0.020000      0.000000    -0.091677
 Begin section !   3
 Begin curve !    3
```

```
   0.030000      0.000000    -0.013745
   0.030000      0.010649    -0.014403
   0.030000      0.021445    -0.017000
   0.030000      0.031847    -0.022297
   0.030000      0.040542    -0.030738
   0.030000      0.045948    -0.041833
   0.030000      0.047496    -0.054144
   0.030000      0.045948    -0.066456
   0.030000      0.040542    -0.077551
   0.030000      0.031847    -0.085992
   0.030000      0.021445    -0.091289
   0.030000      0.010649    -0.093885
   0.030000      0.000000    -0.094544
Begin section !   4
Begin curve !    4
   0.040000      0.000000    -0.002715
   0.040000      0.012442    -0.003407
   0.040000      0.025112    -0.006345
   0.040000      0.037262    -0.012578
   0.040000      0.047148    -0.022619
   0.040000      0.052960    -0.035649
   0.040000      0.054486    -0.049840
   0.040000      0.052960    -0.064030
   0.040000      0.047148    -0.077061
   0.040000      0.037262    -0.087102
   0.040000      0.025112    -0.093335
   0.040000      0.012442    -0.096273
   0.040000      0.000000    -0.096965
Begin section !   5
Begin curve !    5
   0.050000      0.000000     0.006970
   0.050000      0.014022     0.006271
   0.050000      0.028372     0.003081
   0.050000      0.042093    -0.003968
   0.050000      0.052983    -0.015471
```

```
   0.050000      0.059050    -0.030238
   0.050000      0.060515    -0.046060
   0.050000      0.059050    -0.061883
   0.050000      0.052983    -0.076650
   0.050000      0.042093    -0.088153
   0.050000      0.028372    -0.095202
   0.050000      0.014022    -0.098392
   0.050000      0.000000    -0.099091
Begin section !   6
Begin curve !    6
   0.060000      0.000000     0.015665
   0.060000      0.015446     0.014977
   0.060000      0.031333     0.011603
   0.060000      0.046495     0.003828
   0.060000      0.058251    -0.009036
   0.060000      0.064464    -0.025394
   0.060000      0.065848    -0.042667
   0.060000      0.064464    -0.059940
   0.060000      0.058251    -0.076298
   0.060000      0.046495    -0.089163
   0.060000      0.031333    -0.096937
   0.060000      0.015446    -0.100312
   0.060000      0.000000    -0.101000
Begin section !   7
Begin curve !    7
   0.070000      0.000000     0.023587
   0.070000      0.016746     0.022921
   0.070000      0.034058     0.019415
   0.070000      0.050563     0.010987
   0.070000      0.063071    -0.003162
   0.070000      0.069350    -0.020994
   0.070000      0.070642    -0.039576
   0.070000      0.069350    -0.058158
   0.070000      0.063071    -0.075990
   0.070000      0.050563    -0.090139
```

```
   0.070000      0.034058    -0.098566
   0.070000      0.016746    -0.102073
   0.070000      0.000000    -0.102739
Begin section !   8
Begin curve !     8
   0.080000      0.000000     0.030876
   0.080000      0.017945     0.030240
   0.080000      0.036589     0.026644
   0.080000      0.054354     0.017623
   0.080000      0.067523     0.002253
   0.080000      0.073804    -0.016955
   0.080000      0.075000    -0.036731
   0.080000      0.073804    -0.056507
   0.080000      0.067523    -0.075715
   0.080000      0.054354    -0.091085
   0.080000      0.036589    -0.100106
   0.080000      0.017945    -0.103703
   0.080000      0.000000    -0.104339
Begin section !   9
Begin curve !     9
   0.090000      0.000000     0.037634
   0.090000      0.019058     0.037032
   0.090000      0.038956     0.033379
   0.090000      0.057911     0.023817
   0.090000      0.071660     0.007279
   0.090000      0.077896    -0.013222
   0.090000      0.078995    -0.034094
   0.090000      0.077896    -0.054966
   0.090000      0.071660    -0.075467
   0.090000      0.057911    -0.092005
   0.090000      0.038956    -0.101567
   0.090000      0.019058    -0.105220
   0.090000      0.000000    -0.105822
Begin section !   10
Begin curve !     10
```

```
   0.100000      0.000000      0.043933
   0.100000      0.020097      0.043368
   0.100000      0.041178      0.039686
   0.100000      0.061264      0.029628
   0.100000      0.075523      0.011967
   0.100000      0.081674     -0.009751
   0.100000      0.082680     -0.031636
   0.100000      0.081674     -0.053520
   0.100000      0.075523     -0.075239
   0.100000      0.061264     -0.092899
   0.100000      0.041178     -0.102958
   0.100000      0.020097     -0.106639
   0.100000      0.000000     -0.107205
Begin section !   11
Begin curve !     11
   0.200000      0.000000      0.089855
   0.200000      0.027684      0.089604
   0.200000      0.057814      0.086421
   0.200000      0.086788      0.073072
   0.200000      0.103486      0.046033
   0.200000      0.107896      0.015196
   0.200000      0.108253     -0.013715
   0.200000      0.107896     -0.042626
   0.200000      0.103486     -0.073463
   0.200000      0.086788     -0.100503
   0.200000      0.057814     -0.113851
   0.200000      0.027684     -0.117034
   0.200000      0.000000     -0.117285
Begin section !   12
Begin curve !     12
   0.300000      0.000000      0.115428
   0.300000      0.031904      0.115331
   0.300000      0.067409      0.113020
   0.300000      0.102008      0.098273
   0.300000      0.118259      0.064542
```

```
   0.300000      0.120918     0.028665
   0.300000      0.121031    -0.003735
   0.300000      0.120918    -0.036135
   0.300000      0.118259    -0.072012
   0.300000      0.102008    -0.105743
   0.300000      0.067409    -0.120490
   0.300000      0.031904    -0.122802
   0.300000      0.000000    -0.122899
Begin section !   13
Begin curve !    13
   0.400000      0.000000     0.125000
   0.400000      0.033484     0.124965
   0.400000      0.071277     0.123455
   0.400000      0.108819     0.108819
   0.400000      0.123455     0.071277
   0.400000      0.124965     0.033484
   0.400000      0.125000     0.000000
   0.400000      0.124965    -0.033484
   0.400000      0.123455    -0.071277
   0.400000      0.108819    -0.108819
   0.400000      0.071277    -0.123455
   0.400000      0.033484    -0.124965
   0.400000      0.000000    -0.125000
Begin section !   14
Begin curve !    14
   0.500000      0.000000     0.125000
   0.500000      0.033484     0.124965
   0.500000      0.071277     0.123455
   0.500000      0.108819     0.108819
   0.500000      0.123455     0.071277
   0.500000      0.124965     0.033484
   0.500000      0.125000     0.000000
   0.500000      0.124965    -0.033484
   0.500000      0.123455    -0.071277
   0.500000      0.108819    -0.108819
```

```
   0.500000      0.071277    -0.123455
   0.500000      0.033484    -0.124965
   0.500000      0.000000    -0.125000
Begin section !   15
Begin curve !    15
   0.600000      0.000000     0.125000
   0.600000      0.033484     0.124965
   0.600000      0.071277     0.123455
   0.600000      0.108819     0.108819
   0.600000      0.123455     0.071277
   0.600000      0.124965     0.033484
   0.600000      0.125000     0.000000
   0.600000      0.124965    -0.033484
   0.600000      0.123455    -0.071277
   0.600000      0.108819    -0.108819
   0.600000      0.071277    -0.123455
   0.600000      0.033484    -0.124965
   0.600000      0.000000    -0.125000
Begin section !   16
Begin curve !    16
   0.700000      0.000000     0.125000
   0.700000      0.033484     0.124965
   0.700000      0.071277     0.123455
   0.700000      0.108819     0.108819
   0.700000      0.123455     0.071277
   0.700000      0.124965     0.033484
   0.700000      0.125000     0.000000
   0.700000      0.124965    -0.033484
   0.700000      0.123455    -0.071277
   0.700000      0.108819    -0.108819
   0.700000      0.071277    -0.123455
   0.700000      0.033484    -0.124965
   0.700000      0.000000    -0.125000
Begin section !   17
Begin curve !    17
```

```
   0.800000      0.000000      0.125000
   0.800000      0.033484      0.124965
   0.800000      0.071277      0.123455
   0.800000      0.108819      0.108819
   0.800000      0.123455      0.071277
   0.800000      0.124965      0.033484
   0.800000      0.125000      0.000000
   0.800000      0.124965     -0.033484
   0.800000      0.123455     -0.071277
   0.800000      0.108819     -0.108819
   0.800000      0.071277     -0.123455
   0.800000      0.033484     -0.124965
   0.800000      0.000000     -0.125000
Begin section !   18
Begin curve !    18
   0.900000      0.000000      0.122284
   0.900000      0.032267      0.122234
   0.900000      0.068506      0.120466
   0.900000      0.104043      0.105853
   0.900000      0.118656      0.070317
   0.900000      0.120423      0.034078
   0.900000      0.120474      0.001811
   0.900000      0.120423     -0.030457
   0.900000      0.118656     -0.066695
   0.900000      0.104043     -0.102232
   0.900000      0.068506     -0.116845
   0.900000      0.032267     -0.118613
   0.900000      0.000000     -0.118663
Begin section !   19
Begin curve !    19
   1.000000      0.000000      0.117449
   1.000000      0.030103      0.117378
   1.000000      0.063705      0.115374
   1.000000      0.096216      0.101250
   1.000000      0.110340      0.068739
```

```
   1.000000      0.112344     0.035136
   1.000000      0.112416     0.005034
   1.000000      0.112344    -0.025069
   1.000000      0.110340    -0.058671
   1.000000      0.096216    -0.091183
   1.000000      0.063705    -0.105306
   1.000000      0.030103    -0.107310
   1.000000      0.000000    -0.107382
Begin section !   20
Begin curve !    20
   1.100000      0.000000     0.111445
   1.100000      0.027414     0.111346
   1.100000      0.057786     0.109125
   1.100000      0.086766     0.095803
   1.100000      0.100088     0.066823
   1.100000      0.102309     0.036450
   1.100000      0.102408     0.009037
   1.100000      0.102309    -0.018377
   1.100000      0.100088    -0.048749
   1.100000      0.086766    -0.077729
   1.100000      0.057786    -0.091051
   1.100000      0.027414    -0.093272
   1.100000      0.000000    -0.093371
Begin section !   21
Begin curve !    21
   1.200000      0.000000     0.104717
   1.200000      0.024399     0.104582
   1.200000      0.051184     0.102176
   1.200000      0.076377     0.089899
   1.200000      0.088654     0.064706
   1.200000      0.091060     0.037921
   1.200000      0.091195     0.013522
   1.200000      0.091060    -0.010877
   1.200000      0.088654    -0.037662
   1.200000      0.076377    -0.062855
```

```
   1.200000      0.051184    -0.075132
   1.200000      0.024399    -0.077538
   1.200000      0.000000    -0.077673
Begin section !   22
Begin curve !     22
   1.300000      0.000000     0.097604
   1.300000      0.021211     0.097424
   1.300000      0.044232     0.094876
   1.300000      0.065571     0.083835
   1.300000      0.076613     0.062496
   1.300000      0.079160     0.039475
   1.300000      0.079341     0.018264
   1.300000      0.079160    -0.002947
   1.300000      0.076613    -0.025969
   1.300000      0.065571    -0.047307
   1.300000      0.044232    -0.058349
   1.300000      0.021211    -0.060896
   1.300000      0.000000    -0.061077
Begin section !   23
Begin curve !     23
   1.400000      0.000000     0.090408
   1.400000      0.017982     0.090171
   1.400000      0.037227     0.087540
   1.400000      0.054804     0.077866
   1.400000      0.064479     0.060288
   1.400000      0.067109     0.041043
   1.400000      0.067346     0.023062
   1.400000      0.067109     0.005080
   1.400000      0.064479    -0.014165
   1.400000      0.054804    -0.031743
   1.400000      0.037227    -0.041417
   1.400000      0.017982    -0.044048
   1.400000      0.000000    -0.044284
Begin section !   24
Begin curve !     24
```

```
   1.500000      0.000000      0.083420
   1.500000      0.014843      0.083116
   1.500000      0.030456      0.080471
   1.500000      0.044510      0.072230
   1.500000      0.052751      0.058176
   1.500000      0.055396      0.042563
   1.500000      0.055700      0.027720
   1.500000      0.055396      0.012877
   1.500000      0.052751     -0.002736
   1.500000      0.044510     -0.016790
   1.500000      0.030456     -0.025031
   1.500000      0.014843     -0.027675
   1.500000      0.000000     -0.027980
Begin section !   25
Begin curve !    25
   1.600000      0.000000      0.076950
   1.600000      0.011933      0.076567
   1.600000      0.024218      0.073980
   1.600000      0.035123      0.067156
   1.600000      0.041946      0.056251
   1.600000      0.044534      0.043966
   1.600000      0.044917      0.032033
   1.600000      0.044534      0.020101
   1.600000      0.041946      0.007816
   1.600000      0.035123     -0.003090
   1.600000      0.024218     -0.009913
   1.600000      0.011933     -0.012500
   1.600000      0.000000     -0.012884
Begin section !   26
Begin curve !    26
   1.700000      0.000000      0.071352
   1.700000      0.009407      0.070874
   1.700000      0.018839      0.068396
   1.700000      0.027103      0.062869
   1.700000      0.032630      0.054604
```

```
   1.700000      0.035109      0.045173
   1.700000      0.035586      0.035765
   1.700000      0.035109      0.026358
   1.700000      0.032630      0.016926
   1.700000      0.027103      0.008662
   1.700000      0.018839      0.003135
   1.700000      0.009407      0.000657
   1.700000      0.000000      0.000179
Begin section !   27
Begin curve !    27
   1.800000      0.000000      0.067085
   1.800000      0.007467      0.066479
   1.800000      0.014713      0.064093
   1.800000      0.020990      0.059600
   1.800000      0.025483      0.053323
   1.800000      0.027868      0.046077
   1.800000      0.028475      0.038610
   1.800000      0.027868      0.031143
   1.800000      0.025483      0.023897
   1.800000      0.020990      0.017620
   1.800000      0.014713      0.013127
   1.800000      0.007467      0.010742
   1.800000      0.000000      0.010135
Begin section !   28
Begin curve !    28
   1.900000      0.000000      0.065000
   1.900000      0.006470      0.064148
   1.900000      0.012500      0.061651
   1.900000      0.017678      0.057678
   1.900000      0.021651      0.052500
   1.900000      0.024148      0.046470
   1.900000      0.025000      0.040000
   1.900000      0.024148      0.033530
   1.900000      0.021651      0.027500
   1.900000      0.017678      0.022322
```

```
   1.900000      0.012500      0.018349
   1.900000      0.006470      0.015852
   1.900000      0.000000      0.015000
Begin section !   29
Begin curve !    29
   1.910000      0.000000      0.064875
   1.910000      0.006438      0.064027
   1.910000      0.012437      0.061542
   1.910000      0.017589      0.057589
   1.910000      0.021542      0.052437
   1.910000      0.024027      0.046438
   1.910000      0.024875      0.040000
   1.910000      0.024027      0.033562
   1.910000      0.021542      0.027563
   1.910000      0.017589      0.022411
   1.910000      0.012437      0.018458
   1.910000      0.006438      0.015973
   1.910000      0.000000      0.015125
Begin section !   30
Begin curve !    30
   1.920000      0.000000      0.064495
   1.920000      0.006340      0.063660
   1.920000      0.012247      0.061213
   1.920000      0.017321      0.057321
   1.920000      0.021213      0.052247
   1.920000      0.023660      0.046340
   1.920000      0.024495      0.040000
   1.920000      0.023660      0.033660
   1.920000      0.021213      0.027753
   1.920000      0.017321      0.022679
   1.920000      0.012247      0.018787
   1.920000      0.006340      0.016340
   1.920000      0.000000      0.015505
Begin section !   31
Begin curve !    31
```

```
   1.930000      0.000000      0.063848
   1.930000      0.006172      0.063036
   1.930000      0.011924      0.060653
   1.930000      0.016863      0.056863
   1.930000      0.020653      0.051924
   1.930000      0.023036      0.046172
   1.930000      0.023848      0.040000
   1.930000      0.023036      0.033828
   1.930000      0.020653      0.028076
   1.930000      0.016863      0.023137
   1.930000      0.011924      0.019347
   1.930000      0.006172      0.016964
   1.930000      0.000000      0.016152
Begin section !   32
Begin curve !     32
   1.940000      0.000000      0.062913
   1.940000      0.005930      0.062132
   1.940000      0.011456      0.059843
   1.940000      0.016202      0.056202
   1.940000      0.019843      0.051456
   1.940000      0.022132      0.045930
   1.940000      0.022913      0.040000
   1.940000      0.022132      0.034070
   1.940000      0.019843      0.028544
   1.940000      0.016202      0.023798
   1.940000      0.011456      0.020157
   1.940000      0.005930      0.017868
   1.940000      0.000000      0.017087
Begin section !   33
Begin curve !     33
   1.950000      0.000000      0.061651
   1.950000      0.005604      0.060913
   1.950000      0.010825      0.058750
   1.950000      0.015309      0.055309
   1.950000      0.018750      0.050825
```

```
   1.950000      0.020913      0.045604
   1.950000      0.021651      0.040000
   1.950000      0.020913      0.034396
   1.950000      0.018750      0.029175
   1.950000      0.015309      0.024691
   1.950000      0.010825      0.021250
   1.950000      0.005604      0.019087
   1.950000      0.000000      0.018349
Begin section !   34
Begin curve !    34
   1.960000      0.000000      0.060000
   1.960000      0.005176      0.059319
   1.960000      0.010000      0.057321
   1.960000      0.014142      0.054142
   1.960000      0.017321      0.050000
   1.960000      0.019319      0.045176
   1.960000      0.020000      0.040000
   1.960000      0.019319      0.034824
   1.960000      0.017321      0.030000
   1.960000      0.014142      0.025858
   1.960000      0.010000      0.022679
   1.960000      0.005176      0.020681
   1.960000      0.000000      0.020000
Begin section !   35
Begin curve !    35
   1.970000      0.000000      0.057854
   1.970000      0.004621      0.057245
   1.970000      0.008927      0.055462
   1.970000      0.012624      0.052624
   1.970000      0.015462      0.048927
   1.970000      0.017245      0.044621
   1.970000      0.017854      0.040000
   1.970000      0.017245      0.035379
   1.970000      0.015462      0.031073
   1.970000      0.012624      0.027376
```

```
   1.970000      0.008927      0.024538
   1.970000      0.004621      0.022755
   1.970000      0.000000      0.022146
Begin section !   36
Begin curve !     36
   1.980000      0.000000      0.055000
   1.980000      0.003882      0.054489
   1.980000      0.007500      0.052990
   1.980000      0.010607      0.050607
   1.980000      0.012990      0.047500
   1.980000      0.014489      0.043882
   1.980000      0.015000      0.040000
   1.980000      0.014489      0.036118
   1.980000      0.012990      0.032500
   1.980000      0.010607      0.029393
   1.980000      0.007500      0.027010
   1.980000      0.003882      0.025511
   1.980000      0.000000      0.025000
Begin section !   37
Begin curve !     37
   1.990000      0.000000      0.050897
   1.990000      0.002820      0.050526
   1.990000      0.005449      0.049437
   1.990000      0.007706      0.047706
   1.990000      0.009437      0.045449
   1.990000      0.010526      0.042820
   1.990000      0.010897      0.040000
   1.990000      0.010526      0.037180
   1.990000      0.009437      0.034551
   1.990000      0.007706      0.032294
   1.990000      0.005449      0.030563
   1.990000      0.002820      0.029474
   1.990000      0.000000      0.029103
Begin section !   38
Begin curve !     38
```

```
   1.991000      0.000000      0.050365
   1.991000      0.002683      0.050012
   1.991000      0.005183      0.048977
   1.991000      0.007329      0.047329
   1.991000      0.008977      0.045183
   1.991000      0.010012      0.042683
   1.991000      0.010365      0.040000
   1.991000      0.010012      0.037317
   1.991000      0.008977      0.034817
   1.991000      0.007329      0.032671
   1.991000      0.005183      0.031023
   1.991000      0.002683      0.029988
   1.991000      0.000000      0.029635
Begin section !   39
Begin curve !    39
   1.992000      0.000000      0.049798
   1.992000      0.002536      0.049464
   1.992000      0.004899      0.048485
   1.992000      0.006928      0.046928
   1.992000      0.008485      0.044899
   1.992000      0.009464      0.042536
   1.992000      0.009798      0.040000
   1.992000      0.009464      0.037464
   1.992000      0.008485      0.035101
   1.992000      0.006928      0.033072
   1.992000      0.004899      0.031515
   1.992000      0.002536      0.030536
   1.992000      0.000000      0.030202
Begin section !   40
Begin curve !    40
   1.993000      0.000000      0.049189
   1.993000      0.002378      0.048876
   1.993000      0.004594      0.047958
   1.993000      0.006498      0.046498
   1.993000      0.007958      0.044594
```

```
   1.993000      0.008876      0.042378
   1.993000      0.009189      0.040000
   1.993000      0.008876      0.037622
   1.993000      0.007958      0.035406
   1.993000      0.006498      0.033502
   1.993000      0.004594      0.032042
   1.993000      0.002378      0.031124
   1.993000      0.000000      0.030811
Begin section !   41
Begin curve !    41
   1.994000      0.000000      0.048529
   1.994000      0.002208      0.048239
   1.994000      0.004265      0.047387
   1.994000      0.006031      0.046031
   1.994000      0.007387      0.044265
   1.994000      0.008239      0.042208
   1.994000      0.008529      0.040000
   1.994000      0.008239      0.037792
   1.994000      0.007387      0.035735
   1.994000      0.006031      0.033969
   1.994000      0.004265      0.032613
   1.994000      0.002208      0.031761
   1.994000      0.000000      0.031471
Begin section !   42
Begin curve !    42
   1.995000      0.000000      0.047806
   1.995000      0.002020      0.047540
   1.995000      0.003903      0.046760
   1.995000      0.005520      0.045520
   1.995000      0.006760      0.043903
   1.995000      0.007540      0.042020
   1.995000      0.007806      0.040000
   1.995000      0.007540      0.037980
   1.995000      0.006760      0.036097
   1.995000      0.005520      0.034480
```

```
   1.995000      0.003903      0.033240
   1.995000      0.002020      0.032460
   1.995000      0.000000      0.032194
Begin section !   43
Begin curve !    43
   1.996000      0.000000      0.047000
   1.996000      0.001812      0.046761
   1.996000      0.003500      0.046062
   1.996000      0.004950      0.044950
   1.996000      0.006062      0.043500
   1.996000      0.006761      0.041812
   1.996000      0.007000      0.040000
   1.996000      0.006761      0.038188
   1.996000      0.006062      0.036500
   1.996000      0.004950      0.035050
   1.996000      0.003500      0.033938
   1.996000      0.001812      0.033239
   1.996000      0.000000      0.033000
Begin section !   44
Begin curve !    44
   1.997000      0.000000      0.046078
   1.997000      0.001573      0.045871
   1.997000      0.003039      0.045263
   1.997000      0.004298      0.044298
   1.997000      0.005263      0.043039
   1.997000      0.005871      0.041573
   1.997000      0.006078      0.040000
   1.997000      0.005871      0.038427
   1.997000      0.005263      0.036961
   1.997000      0.004298      0.035702
   1.997000      0.003039      0.034737
   1.997000      0.001573      0.034129
   1.997000      0.000000      0.033922
Begin section !   45
Begin curve !    45
```

```
   1.998000      0.000000      0.044975
   1.998000      0.001288      0.044805
   1.998000      0.002487      0.044308
   1.998000      0.003518      0.043518
   1.998000      0.004308      0.042487
   1.998000      0.004805      0.041288
   1.998000      0.004975      0.040000
   1.998000      0.004805      0.038712
   1.998000      0.004308      0.037513
   1.998000      0.003518      0.036482
   1.998000      0.002487      0.035692
   1.998000      0.001288      0.035195
   1.998000      0.000000      0.035025
Begin section !   46
Begin curve !    46
   1.999000      0.000000      0.043527
   1.999000      0.000913      0.043407
   1.999000      0.001763      0.043054
   1.999000      0.002494      0.042494
   1.999000      0.003054      0.041763
   1.999000      0.003407      0.040913
   1.999000      0.003527      0.040000
   1.999000      0.003407      0.039087
   1.999000      0.003054      0.038237
   1.999000      0.002494      0.037506
   1.999000      0.001763      0.036946
   1.999000      0.000913      0.036593
   1.999000      0.000000      0.036473
```

A.4 Actual code

A hard copy of the software is presented in this section:

```
%FILE: main.m{E}
%{L}robinone, robintwo, robinthree, robinfour, pylonone, pylontwo
%Prepared by: Vaneshen Naidoo
%Purpose: To create the non dimensional coordinates of the
%ROBIN helicopter configuration and write it out to an .IBL file
%Overall method:
%1. Calculates the coordinates of the ROBIN 0.0 < x/l < 0.1
%2. Calculates the coordinates of the ROBIN 0.1 < x/l < 0.4
%3. Calculates the coordinates of the ROBIN 0.4 < x/l < 0.8
%4. Calculates the coordinates of the ROBIN 0.8 < x/l < 1.9
%5. Calculates the coordinates of the ROBIN  1.9 < x/l < 1.99
%6. Calculates the coordinates of the ROBIN 1.99 < x/l < 2.0
%7. Calculates the coordinates of the pylon 0.4 < x/l < 0.5
%8. Calculate the coordinates of the pylon 0.4 < x/l < 0.8
%9. Calculates the coordinates of the pylon 0.8 < x/l < 1.0
%10.Calculates the coordinates of the pylon 1.0 < x/l < 1.018
%11.Creates a matrix with the co-ordinates of the ROBIN and deletes the any
%repeated coordinates
%12.Creates a matrix with the co-ordinates of the pylon and deletes the any
%repeated coordinates
%13. Plots the ROBIN with the pylon
%14. Writes the output (.ibl) file for the ROBIN body
%14. Writes the output (.ibl) file for the pylon

clc
clear all
%------------------------------------------------------------

%Calculates the coordinates of the ROBIN  0.0 < x/l < 0.1

x_beg8 = 0.0; %add sumtin here
```

```
x_inc8 = 0.01;
x_end8 = 0.1;
interval8 = ((x_end8 - x_beg8) / x_inc8 ) + 1;
x_plot8 = zeros(interval8, 13);
y_plot8 = zeros(interval8, 13);
z_plot8 = zeros(interval8, 13);
[x_plot8,y_plot8,z_plot8] = robinone(x_beg8, x_inc8, x_end8, interval8);
%-------------------------------------------------------------

%Calculates the coordinates of the ROBIN 0.1 < x/l < 0.4

x_beg = 0.1;
x_inc = 0.1;
x_end = 0.4;
interval = ((x_end - x_beg) / x_inc ) + 1;
x_plot = zeros(interval, 13);
y_plot = zeros(interval, 13);
z_plot = zeros(interval, 13);
[x_plot,y_plot,z_plot] = robinone(x_beg, x_inc, x_end,interval);

%-------------------------------------------------------

%Calculates the coordinates of the ROBIN 0.4 < x/l < 0.8

x_beg2 = 0.4;
x_inc2 = 0.1;
x_end2 = 0.8;
interval_2 = ((x_end2 - x_beg2) / x_inc2 ) + 1;
x_plot2 = zeros(interval_2, 13);
y_plot2 = zeros(interval_2, 13);
z_plot2 = zeros(interval_2, 13);
[x_plot2,y_plot2,z_plot2] = robintwo(x_beg2, x_inc2, x_end2,interval_2);

%-------------------------------------------------------
%Calculates the coordinates of the ROBIN  0.8 < x/l < 1.9
```

```
x_beg3 = 0.8;
x_inc3 = 0.1;
x_end3 = 1.9;
interval_3 = 12;% ((x_end3 - x_beg3) / x_inc3 ) + 1;
x_plot3 = zeros(interval_3, 13);
y_plot3 = zeros(interval_3, 13);
z_plot3 = zeros(interval_3, 13);
[x_plot3,y_plot3,z_plot3] = robinthree(x_beg3, x_inc3, x_end3,interval_3);

%-------------------------------------------------------
%Calculates the coordinates of the ROBIN  1.9 < x/l < 1.99

x_beg4 = 1.9;
x_inc4 = 0.01;
x_end4 = 1.99;
interval_4 = ((x_end4 - x_beg4) / x_inc4 ) + 1;
x_plot4 = zeros(interval_4, 13);
y_plot4 = zeros(interval_4, 13);
z_plot4 = zeros(interval_4, 13);
[x_plot4,y_plot4,z_plot4] = robinfour(x_beg4, x_inc4, x_end4,interval_4);

%-------------------------------------------------------
%Calculates the coordinates of the ROBIN 1.99 < x/l < 2.0

x_beg7 = 1.99;
x_inc7 = 0.001;
x_end7 = 2.0;
interval_7 = ((x_end7 - x_beg7) / x_inc7 ) + 1;
x_plot7 = zeros(interval_7, 13);
y_plot7 = zeros(interval_7, 13);
z_plot7 = zeros(interval_7, 13);
[x_plot7,y_plot7,z_plot7] = robinfour(x_beg7, x_inc7, x_end7,interval_7);

%-------------------------------------------------------
```

```
%Calculates the coordinates of the pylon 0.4 < x/l < 0.5

x_beg9 = 0.4;
x_inc9 = 0.01;
x_end9 = 0.5;
interval9 = 11 ; %((x_end9 - x_beg9) / x_inc9 ) + 1;
x_plot9 = zeros(interval9, 13);
y_plot9 = zeros(interval9, 13);
z_plot9 = zeros(interval9, 13);
[x_plot9,y_plot9,z_plot9] = pylonone(x_beg9, x_inc9, x_end9,interval9);

%-------------------------------------------------------

%Calculates the coordinates of the pylon 0.5 < x/l < 0.8


x_beg5 = 0.5;
x_inc5 = 0.1;
x_end5 = 0.8;
interval5 = ((x_end5 - x_beg5) / x_inc5 ) + 1;
x_plot5 = zeros(interval5, 13);
y_plot5 = zeros(interval5, 13);
z_plot5 = zeros(interval5, 13);
[x_plot5,y_plot5,z_plot5] = pylonone(x_beg5, x_inc5, x_end5,interval5);

%-------------------------------------------------------

%Calculates the coordinates of the pylon 0.8 < x/l < 1.0

x_beg6 = 0.8;
x_inc6 = 0.025;
x_end6 = 1.00;
interval6 = 9; %((x_end6 - x_beg6) / x_inc6 ) + 1; % x_end / x_beg
x_plot6 = zeros(interval6, 13);
y_plot6 = zeros(interval6, 13);
```

```
z_plot6 = zeros(interval6, 13);
[x_plot6,y_plot6,z_plot6] = pylontwo(x_beg6, x_inc6, x_end6,interval6);

 %-------------------------------------------------------
 %Calculates the coordinates of the pylon 1.0 < x/l < 1.018

x_beg10 = 1.00;
x_inc10 = 0.0018;
x_end10 = 1.018;
interval10 = 11; %((x_end6 - x_beg6) / x_inc6 ) + 1; % x_end / x_beg
x_plot10 = zeros(interval10, 13);
y_plot10 = zeros(interval10, 13);
z_plot10 = zeros(interval10, 13);
[x_plot10,y_plot10,z_plot10] = pylontwo(x_beg10, x_inc10, x_end10,interval10);

 %-------------------------------------------------------

% creates a matrix with the co-ordinates of the robin
x_finalplotter = [x_plot8;x_plot;x_plot2;x_plot3;x_plot4;x_plot7];

% deletes repeated co ordinates

x_finalplotter(12,:) = [ ];
x_finalplotter(15,:) = [ ];
x_finalplotter(19,:) = [ ];
x_finalplotter(30,:) = [ ];
x_finalplotter(39,:) = [ ];
x_finalplotter(1,:) = [ ];
x_finalplotter(47,:) = [ ];
%-------------------------------------------------------
y_finalplotter = [y_plot8;y_plot;y_plot2;y_plot3;y_plot4;y_plot7];
y_finalplotter(12,:) = [ ];
y_finalplotter(15,:) = [ ];
y_finalplotter(19,:) = [ ];
y_finalplotter(30,:) = [ ];
```

```
y_finalplotter(39,:) = [ ];
y_finalplotter(1,:) = [ ];
y_finalplotter(47,:) = [ ];

%-------------------------------------------------------
z_finalplotter = [z_plot8;z_plot;z_plot2;z_plot3;z_plot4;z_plot7];
z_finalplotter(12,:) = [ ];
z_finalplotter(15,:) = [ ];
z_finalplotter(19,:) = [ ];
z_finalplotter(30,:) = [ ];
z_finalplotter(39,:) = [ ];
z_finalplotter(1,:) = [ ];
z_finalplotter(47,:) = [ ];

%-------------------------------------------------------

% creates a matrix with the co-ordinates of the pylon
x_pylon = [x_plot9;x_plot5;x_plot6;x_plot10];

% deletes repeated co ordinates
x_pylon(12,:) = [ ];
x_pylon(15,:) = [ ];
x_pylon(23,:) = [ ];
x_pylon(1,:) = [ ];
x_pylon(31,:) = [ ];
%-------------------------------------------------------
y_pylon = [y_plot9;y_plot5;y_plot6;y_plot10];
y_pylon(12,:) = [ ];
y_pylon(15,:) = [ ];
y_pylon(23,:) = [ ];
y_pylon(1,:) = [ ];
y_pylon(31,:) = [ ];
%-------------------------------------------------------
z_pylon = [z_plot9;z_plot5;z_plot6;z_plot10];
z_pylon(12,:) = [ ];
```

```
z_pylon(15,:) = [ ];
z_pylon(23,:) = [ ];
z_pylon(1,:) = [ ];
z_pylon(31,:) = [ ];
%-------------------------------------------------------

%plots the robin with the pylon
hold
surf(x_finalplotter,y_finalplotter,z_finalplotter);
surf(x_pylon,y_pylon,z_pylon);
xlabel('x-axis'), ylabel('y-axis'), zlabel('z-axis');
axis equal;
hold off
%-------------------------------------------------------
% Writes the output (.ibl) file for the ROBIN
count01 = 1;
count03 = 1;
fid = fopen('the_robin.ibl','w');
fprintf(fid, 'Open Index Arclength');
 for count01 = 1:46,
     fprintf(fid, '\n Begin section !   %d',count01);
     fprintf(fid, '\n Begin curve !    %d', count01);
     for count03 = 1:13,
         fprintf(fid, '\n%12.6f  %12.6f %12.6f', x_finalplotter(count01,count03)
         y_finalplotter(count01,count03), z_finalplotter(count01,count03));
     end
 end
fclose(fid);
%-------------------------------------------------------
% Writes the output (.ibl) file for the pylon
pyl01 = 1;
pyl03 = 1;
fid = fopen('the_pylon.ibl','w');
fprintf(fid, 'Open Index Arclength');
 for pyl01 = 1:30,
```

```
    fprintf(fid, '\n Begin section !   %d',pyl01);
    fprintf(fid, '\n Begin curve !    %d', pyl01);
    for pyl03 = 1:13,
        fprintf(fid, '\n%12.6f  %12.6f %12.6f', x_pylon(pyl01,pyl03),
        y_pylon(pyl01,pyl03),z_pylon(pyl01,pyl03));
    end
 end
fclose(fid);
%-------------------------------------------------------
```

```
%Name of Function: pylonone.m
%Prepared by: Vaneshen Naidoo
%Purpose: To generate the pylon shape between 0.4 < x/l < 0.8
%Parameters: x_beg, x_inc, x_end, interval
%Return Value: x_plot,y,z(output non dimensional coordinates)
%Calls To: calcval.m

%METHOD:

%2. Specifies the angles for the calculations of the coordinates on the
%cross section.
%2. Intialises variables for the functions H,W,Zo,N(vectors);
%radial coordinate(R) as well as Upper and Lower parts of the equation which
%calculates the radial coordinate(matrices) ; y(y/l) coordinates(vector);
%z(z/l)coordinates(vector)
%3. Loops over the angles specified
%   3.1. Loops over the range of the x/l values specified
%   3.2. Specifies the coefficients for H,W,Zo,N
%   3.3. Calculates H,W,Zo,N
%   3.4. Calculates R(non dimensional radial coordinate)
%   3.5. Calculates y/l and z/l
%4. Changes the format of x/l to that of y/l and z/l

function [x_plot,y,z] = pylonone(x_beg,x_inc,x_end,interval)
    % Intialise variables
    A=0;
    value = 0.0;
    theta_beg = 0;
    theta_end = pi;
    theta_inc = pi/12;
    H = zeros(interval,1);
    W = zeros(interval,1);
    Zo = zeros(interval,1);
    N = zeros(interval,1);
```

```
R = zeros(interval,13);
Upper = zeros(interval,13);
Lower = zeros(interval,13);
y = zeros(interval,13);
z = zeros(interval,13);

xx = x_beg : x_inc : x_end; % assigns a vector for x coordinates
x_plot = repmat(xx, [13 1]); %swaps the colums and rows of the x coordinates
% looping over the angles specified
for theta = theta_beg : theta_inc : theta_end
    for x = x_beg : x_inc : x_end % looping over x for 0.0 to 0.4
        A= A+1 ;
        % defining coeff for H
        H_c = [1.0; -1.0; -0.8; 0.4; 3.0; 0.0; 0.145; 3.0];
        % defining coeff for W
        W_c = [1.0; -1.0; -0.8; 0.4; 3.0; 0.0; 0.166; 3.0];
        % defining coeff for Zo
        Z_c = [0.125; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0];
        % defining coeff for N
        N_c = [5.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0];
        [value] = calcval(H_c,x);
        H(A) = [value];  % calculates H
        [value] = calcval(W_c,x);
        W(A) = [value];  % calculates W
        [value] = calcval(Z_c,x);
        Zo(A) = [value];  % calculates Zo
        [value] = calcval(N_c,x);
        N(A) = [value];  % calculates N

        % calculates the value for R(radial coordinate)
        Upper(A) =((H(A)/2)*(W(A)/2))^(N(A));
        Lower(A) = 1 / (((abs((H(A)/2)*sin(theta)))^(N(A))) +
        ((abs((W(A)/2)*cos(theta)))^(N(A))));
        if Lower(A) > 10e20
            Lower(A) = 0 ;
```

```
            end
            R(A) = (Upper (A) * Lower(A)) .^ (1/(N(A)))  ;

            y(A) = R(A)*sin(theta);  % calculates y/l
            z(A) = R(A)*cos(theta) + Zo(A); % calculates z/l
        end
    end
% arranges x/l coordinates in the same format as the y/l and z/l
% coordinates
x_plot = x_plot';
```

```
%Name of Function: pylontwo.m
%Prepared by: Vaneshen Naidoo
%Purpose: To generate the pylon shape between 0.8 < x/l < 1.018
%Parameters: x_beg, x_inc, x_end, interval
%Return Value: x_plot,y,z(output non dimensional coordinates)
%Calls To: calcval.m

%METHOD:

%1. Specifies the angles for the calculations of the coordinates on the
%cross section.
%2. Intialises variables for the functions H,W,Zo,N(vectors);
%radial coordinate(R) as well as Upper and Lower parts of the equation which
%calculates the radial coordinate(matrices) ; y(y/l) coordinates(vector);
%z(z/l)coordinates(vector)
%3. Loops over the angles specified
%   3.1. Loops over the range of the x/l values specified
%   3.2. Specifies the coefficients for H,W,Zo,N
%   3.3. Calculates H,W,Zo,N
%   3.4. Calculates R(non dimensional radial coordinate)
%   3.5 Calculates y/l and z/l
%4. Changes the format of x/l to that of y/l and z/l

function [x_plot,y,z] = pylontwo(x_beg,x_inc,x_end,interval)

    A=0;
    value = 0.0;
    theta_beg = 0;
    theta_end = pi;
    theta_inc = pi/12;
    H = zeros(interval,1);
    W = zeros(interval,1);
    Zo = zeros(interval,1);
    N = zeros(interval,1);
```

```
R = zeros(interval,13);
Upper = zeros(interval,13);
Lower = zeros(interval,13);
y = zeros(interval,13);
z = zeros(interval,13);

xx = x_beg : x_inc : x_end;
x_plot = repmat(xx, [13 1]);
for theta = theta_beg : theta_inc : theta_end
    for x = x_beg : x_inc : x_end % looping over x for 0.8 to 1.018
        A= A+1 ;
        % defining coeff for H
        H_c = [1.0; -1.0; -0.8; 0.218; 2.0; 0.0; 0.145; 2.0];
        % defining coeff for W
        W_c = [1.0; -1.0; -0.8; 0.218; 2.0; 0.0; 0.166; 2.0];
        % defining coeff for Zo
        Z_c = [1.0; -1.0; -0.8; 1.1; 1.5; 0.065; 0.06; 0.6];
        % defining coeff for N
        N_c = [5.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0];
        [value] = calcval(H_c,x);
        H(A) = [value]; % calculates H
        [value] = calcval(W_c,x);
        W(A) = [value]; % calculates W
        [value] = calcval(Z_c,x);
        Zo(A) = [value]; % calculates Zo
        [value] = calcval(N_c,x);
        N(A) = [value]; % calculates N

        % calculates the value for R(radial coordinate)
        Upper(A) =((H(A)/2)*(W(A)/2))^(N(A));
        Lower(A) = 1 / (((abs((H(A)/2)*sin(theta)))^(N(A)))
        + ((abs((W(A)/2)*cos(theta)))^(N(A))));
        if Lower(A) > 10e20
            Lower(A) = 0 ;
        end
```

```
        R(A) = (Upper (A) * Lower(A)) .^ (1/(N(A)));

        y(A) = R(A)*sin(theta); % calculates y/l
        z(A) = R(A)*cos(theta) + Zo(A);  % calculates z/l
    end
end
 % arranges x/l coordinates in the same format as the y/l and z/l
% coordinates
x_plot = x_plot';
```

```
%Name of Function: robinone.m
%Prepared by: Vaneshen Naidoo
%Purpose: To generate the ROBIN body shape between 0.0 < x/l < 0.4
%Parameters: x_beg, x_inc, x_end, interval
%Return Value: x_plot,y,z(output non dimensional coordinates)
%Calls To: calcval.m

%METHOD:

%1. Specifies the angles for the calculations of the coordinates on the
%cross section.
%2. Intialises variables for the functions H,W,Zo,N(vectors);
%radial coordinate(R) as well as Upper and Lower parts of the equation which
%calculatesthe radial coordinate(matrices) ; y(y/l) coordinates(vector);
%z(z/l)coordinates(vector)
%3. Loops over the angles specified
%   3.1. Loops over the range of the x/l values specified
%   3.2. Specifies the coefficients for H,W,Zo,N
%   3.3. Calculates H,W,Zo,N
%   3.4. Calculates R(non dimensional radial coordinate)
%   3.5. Calculates y/l and z/l
%4. Changes the format of x/l to that of y/l and z/l

function [x_plot,y,z] = robinone(x_beg, x_inc, x_end, interval)

    % Intialise variables
    A=0;
    value = 0.0;
    theta_beg = 0;
    theta_end = pi;
    theta_inc = pi/12;
    H = zeros(interval,1);
    W = zeros(interval,1);
    Zo = zeros(interval,1);
```

```
N = zeros(interval,1);
R = zeros(interval,13);
Upper = zeros(interval,13);
Lower = zeros(interval,13);
y = zeros(interval,13);
z = zeros(interval,13);

xx = x_beg : x_inc : x_end; % assigns a vector for x coordinates
x_plot = repmat(xx, [13 1]); %swaps the colums and rows of the x coordinates
% looping over the angles specifiedSepc
for theta = theta_beg : theta_inc : theta_end
    for x = x_beg : x_inc : x_end  % looping over x/l for 0.0 to 0.4
        A= A+1 ;
        % defining coeff for H
        H_c = [1.0; -1.0; -0.4 ; 0.4 ; 1.8; 0.0 ; 0.25; 1.8];
        % defining coeff for W
        W_c = [1.0; -1.0; -0.4 ; 0.4 ; 2.0;  0.0; 0.25;  2.0];
        % defining coeff for Zo
        Z_c = [1.0; -1.0; -0.4; 0.4; 1.8; -0.08; 0.08; 1.8];
        % defining coeff for N
        N_c = [2.0; 3.0; 0.0; 0.4; 1.0; 0.0; 1.0; 1.0];
        [value] = calcval(H_c,x);
        H(A) = [value]; % calculates H
        [value] = calcval(W_c,x);
        W(A) = [value];  % calculates W
        [value] = calcval(Z_c,x);
        Zo(A) = [value]; % calculates Zo
        [value] = calcval(N_c,x);
        N(A) = [value];  % calculates N

        % calculates the value for R(radial coordinate)
        Upper(A) =((H(A)/2)*(W(A)/2))^(N(A));
        Lower(A) = 1 / (((abs((H(A)/2)*sin(theta)))^(N(A))) +
        ((abs((W(A)/2)*cos(theta)
        % If the lower part of the equation is too great let it = 0
```

```
            if Lower(A) > 10e20
                Lower(A) = 0 ;
            end
        R(A) = (Upper (A) * Lower(A)) .^ (1/(N(A)));

        y(A) = R(A)*sin(theta);% calculates y/l
        z(A) = R(A)*cos(theta) + Zo(A); %calculates z/l
    end
end
% arranges x/l coordinates in the same format as the y/l and z/l
% coordinates
x_plot = x_plot';
```

```
%Name of Function: robintwo.m
%Prepared by: Vaneshen Naidoo
%Purpose: To generate the ROBIN body shape between 0.4 < x/l < 0.8
%Parameters: x_beg, x_inc, x_end, interval
%Return Value: x_plot,y,z(output non dimensional coordinates)
%Calls To: calcval.m

%METHOD:

%1. Specifies the angles for the calculations of the coordinates on the
%cross section.
%2. Intialises variables for the functions H,W,Zo,N(vectors);
%radial coordinate(R) as well as Upper and Lower parts of the equation which
%calculates the radial coordinate(matrices) ; y(y/l) coordinates(vectors);
%z(z/l)coordinates(vectors)
%3. Loops over the angles specified
%   3.1. Loops over the range of the x/l values specified
%   3.2. Specifies the coefficients for H,W,Zo,N
%   3.3. Calculates H,W,Zo,N
%   3.4. Calculates R(non dimensional radial coordinate)
%   3.5. Calculates y/l and z/l
%4. Changes the format of x/l to that of y/l and z/l

function [x_plot,y,z] = robintwo(x_beg,x_inc,x_end,interval)

    % Intialise variables
    A=0;
    value = 0.0;
    theta_beg = 0;
    theta_end = pi;
    theta_inc = pi/12;
    H = zeros(interval,1);
    W = zeros(interval,1);
    Zo = zeros(interval,1);
```

```
N = zeros(interval,1);
R = zeros(interval,13);
Upper = zeros(interval,13);
Lower = zeros(interval,13);
y = zeros(interval,13);
z = zeros(interval,13);

xx = x_beg : x_inc : x_end; % assigns a vector for x coordinates
x_plot = repmat(xx, [13 1]);%swaps the colums and rows of the x coordinates
% looping over the angles specified
for theta = theta_beg : theta_inc : theta_end
    for x = x_beg : x_inc : x_end % looping over x/l for 0.4 to 0.8
        A= A+1 ;
        % defining coeff for H
        H_c = [0.25; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0];
        % defining coeff for W
        W_c = [0.25; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0];
        % defining coeff for Zo
        Z_c = [0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0];
        % defining coeff for N
        N_c = [5.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0];
        [value] = calcval(H_c,x);
        H(A) = [value];  % calculates H
        [value] = calcval(W_c,x);
        W(A) = [value];  % calculates W
        [value] = calcval(Z_c,x);
        Zo(A) = [value]; % calculates Zo
        [value] = calcval(N_c,x);
        N(A) = [value]; % calculates N

        % calculates the value for R(radial coordinate)
        Upper(A) =((H(A)/2)*(W(A)/2))^(N(A));
        Lower(A) = 1 / (((abs((H(A)/2)*sin(theta)))^(N(A))) +
        ((abs((W(A)/2)*cos(theta)))^(N(A))));
        % If the lower part of the equation is too great let it = 0
```

```
        if Lower(A) > 10e20
         Lower(A) = 0 ;
        end
        R(A) = (Upper (A) * Lower(A)) .^ (1/(N(A)));

        y(A) = R(A)*sin(theta); % calculates y/l
        z(A) = R(A)*cos(theta) + Zo(A); % calculates z/l
    end
end
% arranges x/l coordinates in the same format as the y/l and z/l
% coordinates
x_plot = x_plot';
```

```
%Name of Function: robinthree.m
%Prepared by: Vaneshen Naidoo
%Purpose: To generate the ROBIN body shape between 0.8 < x/l < 1.9
%Parameters: x_beg, x_inc, x_end, interval
%Return Value: x_plot,y,z(output non dimensional coordinates)
%Calls To: calcval.m

%METHOD:

%1. Specifies the angles for the calculations of the coordinates on the
%cross section.
%2. Intialises variables for the functions H,W,Zo,N(vectors);
%radial coordinate(R) as well as Upper and Lower parts of the equation which
%calculates the radial coordinate(matrices) ; y(y/l) coordinates(vector);
%z(z/l)coordinates(vector)
%3. Loops over the angles specified
%   3.1. Loops over the range of the x/l values specified
%   3.2. Specifies the coefficients for H,W,Zo,N
%   3.3. Calculates H,W,Zo,N
%   3.4. Calculates R(non dimensional radial coordinate)
%   3.5. Calculates y/l and z/l
%4. Changes the format of x/l to that of y/l and z/l

function [x_plot,y,z] = robinthree(x_beg,x_inc,x_end,interval)

    % Intialise variables
    A=0;
    value = 0.0;
    theta_beg = 0;
    theta_end = pi;
    theta_inc = pi/12;
    H = zeros(interval,1);
    W = zeros(interval,1);
    Zo = zeros(interval,1);
```

```
N = zeros(interval,1);
R = zeros(interval,13);
Upper = zeros(interval,13);
Lower = zeros(interval,13);
y = zeros(interval,13);
z = zeros(interval,13);

xx = x_beg : x_inc : x_end; % assigns a vector for x coordinates
x_plot = repmat(xx, [13 1]); %swaps the colums and rows of the x coordinates
% looping over the angles specified
for theta = theta_beg : theta_inc : theta_end
    for x = x_beg : x_inc : x_end % looping over x for 0.8 to 1.9
        A= A+1 ;
        % defining coeff for H
        H_c = [1.0; -1.0; -0.8; 1.1; 1.5; 0.05; 0.2; 0.6];
        % defining coeff for W
        W_c = [1.0; -1.0; -0.8; 1.1; 1.5; 0.05; 0.2; 0.6];
        % defining coeff for Zo
        Z_c = [1.0; -1.0; -0.8; 1.1; 1.5; 0.04; -0.04; 0.6];
        % defining coeff for N
        N_c = [5.0; -3.0; -0.8; 1.1; 1.0; 0.0; 0.0; 0.0];
        [value] = calcval(H_c,x);
        H(A) = [value]; % calculates H
        [value] = calcval(W_c,x);
        W(A) = [value]; % calculates W
        [value] = calcval(Z_c,x);
        Zo(A) = [value]; % calculates Zo
        [value] = calcval(N_c,x);
        N(A) = [value]; % calculates N
        % calculates the value for R(radial coordinate)
        Upper(A) =((H(A)/2)*(W(A)/2))^(N(A));
        Lower(A) = 1 / (((abs((H(A)/2)*sin(theta)))^(N(A))) +
        ((abs((W(A)/2)*cos(theta)))^(N(A))));
         % If the lower part of the equation is too great let it = 0
        if Lower(A) > 10e20
```

```
            Lower(A) = 0 ;
        end
        R(A) = (Upper (A) * Lower(A)) .^ (1/(N(A)));

        y(A) = R(A)*sin(theta);% calculates y/l
        z(A) = R(A)*cos(theta) + Zo(A); % calculates z/l
    end
end
% arranges x/l coordinates in the same format as the y/l and z/l
% coordinates
x_plot = x_plot';
```

```
%Name of Function: robinfour.m
%Prepared by: Vaneshen Naidoo
%Purpose: To generate the ROBIN body shape between 1.9 < x/l < 2.0
%Parameters: x_beg, x_inc, x_end, interval
%Return Value: x_plot,y,z(output non dimensional coordinates)
%Calls To: calcval.m

%METHOD:

%1. Specifies the angles for the calculations of the coordinates on the
%cross section.
%2. Intialises variables for the functions H,W,Zo,N(vectors);
%radial coordinate(R) as well as Upper and Lower parts of the equation which
%calculates the radial coordinate(matrices) ; y(y/l) coordinates(vector);
%z(z/l)coordinates(vector)
%3. Loops over the angles specified
%   3.1. Loops over the range of the x/l values specified
%   3.2. Specifies the coefficients for H,W,Zo,N
%   3.3. Calculates H,W,Zo,N
%   3.4. Calculates R(non dimensional radial coordinate)
%   3.5. Calculates y/l and z/l
%4. Changes the format of x/l to that of y/l and z/l

function [x_plot,y,z] = robinfour(x_beg,x_inc,x_end,interval)

    A=0;
    value = 0.0;
    theta_beg = 0;
    theta_end = pi;
    theta_inc = pi/12;
    H = zeros(interval,1);
    W = zeros(interval,1);
    Zo = zeros(interval,1);
    N = zeros(interval,1);
```

```
R = zeros(interval,13);
Upper = zeros(interval,13);
Lower = zeros(interval,13);
y = zeros(interval,13);
z = zeros(interval,13);

xx = x_beg : x_inc : x_end;  % assigns a vector for x coordinates
x_plot = repmat(xx, [13 1]);
%swaps the colums and rows of the x coordinates
for theta = theta_beg : theta_inc : theta_end
    for x = x_beg : x_inc : x_end % looping over x for 1.9 to 2.0
        A= A+1 ;
        % defining coeff for H
        H_c = [1.0; -1.0; -1.9; 0.1; 2.0; 0.0; 0.05; 2.0];
        % defining coeff for W
        W_c = [1.0; -1.0; -1.9; 0.1; 2.0; 0.0; 0.05; 2.0];
        % defining coeff for Zo
        Z_c = [0.04; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0];
        % defining coeff for N
        N_c = [2.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0];
        [value] = calcval(H_c,x);
        H(A) = [value]; % calculates H
        [value] = calcval(W_c,x);
        W(A) = [value]; % calculates W
        [value] = calcval(Z_c,x);
        Zo(A) = [value]; % calculates Zo
        [value] = calcval(N_c,x);
        N(A) = [value]; % calculates N

        Upper(A) =((H(A)/2)*(W(A)/2))^(N(A));
        Lower(A) = 1 / (((abs((H(A)/2)*sin(theta)))^(N(A))) +
        ((abs((W(A)/2)*cos(theta)))^(N(A))));
        if Lower(A) > 10e20
            Lower(A) = 0 ;
        end
```

```
        R(A) = (Upper (A) * Lower(A)) .^ (1/(N(A)));

        y(A) = R(A)*sin(theta);
        z(A) = R(A)*cos(theta) + Zo(A);
    end
end
% arranges x/l coordinates in the same format as the y/l and z/l
% coordinates
x_plot = x_plot';
```

```
%Name of Function: robinthree.m
%Prepared by: Vaneshen Naidoo
%Purpose: To generate the ROBIN body shape between 0.8 < x/l < 1.9
%Parameters: x_beg, x_inc, x_end, interval
%Return Value: x_plot,y,z(output non dimensional coordinates)
%Calls To: calcval.m

%METHOD:

%1. Specifies the angles for the calculations of the coordinates on the
%cross section.
%2. Intialises variables for the functions H,W,Zo,N(vectors);
%radial coordinate(R) as well as Upper and Lower parts of the equation which
%calculates the radial coordinate(matrices) ; y(y/l) coordinates(vector);
%z(z/l)coordinates(vector)
%3. Loops over the angles specified
%   3.1. Loops over the range of the x/l values specified
%   3.2. Specifies the coefficients for H,W,Zo,N
%   3.3. Calculates H,W,Zo,N
%   3.4. Calculates R(non dimensional radial coordinate)
%   3.5. Calculates y/l and z/l
%4. Changes the format of x/l to that of y/l and z/l

function [x_plot,y,z] = robinthree(x_beg,x_inc,x_end,interval)

    % Intialise variables
    A=0;
    value = 0.0;
    theta_beg = 0;
    theta_end = pi;
    theta_inc = pi/12;
    H = zeros(interval,1);
    W = zeros(interval,1);
    Zo = zeros(interval,1);
```

```
N = zeros(interval,1);
R = zeros(interval,13);
Upper = zeros(interval,13);
Lower = zeros(interval,13);
y = zeros(interval,13);
z = zeros(interval,13);

xx = x_beg : x_inc : x_end; % assigns a vector for x coordinates
x_plot = repmat(xx, [13 1]); %swaps the colums and rows of the x coordinates
% looping over the angles specified
for theta = theta_beg : theta_inc : theta_end
    for x = x_beg : x_inc : x_end % looping over x for 0.8 to 1.9
        A= A+1 ;
        % defining coeff for H
        H_c = [1.0; -1.0; -0.8; 1.1; 1.5; 0.05; 0.2; 0.6];
        % defining coeff for W
        W_c = [1.0; -1.0; -0.8; 1.1; 1.5; 0.05; 0.2; 0.6];
        % defining coeff for Zo
        Z_c = [1.0; -1.0; -0.8; 1.1; 1.5; 0.04; -0.04; 0.6];
        % defining coeff for N
        N_c = [5.0; -3.0; -0.8; 1.1; 1.0; 0.0; 0.0; 0.0];
        [value] = calcval(H_c,x);
        H(A) = [value]; % calculates H
        [value] = calcval(W_c,x);
        W(A) = [value]; % calculates W
        [value] = calcval(Z_c,x);
        Zo(A) = [value]; % calculates Zo
        [value] = calcval(N_c,x);
        N(A) = [value]; % calculates N
        % calculates the value for R(radial coordinate)
        Upper(A) =((H(A)/2)*(W(A)/2))^(N(A));
        Lower(A) = 1 / (((abs((H(A)/2)*sin(theta)))^(N(A))) +
        ((abs((W(A)/2)*cos(theta)))^(N(A))));
         % If the lower part of the equation is too great let it = 0
        if Lower(A) > 10e20
```

```
            Lower(A) = 0 ;
        end
        R(A) = (Upper (A) * Lower(A)) .^ (1/(N(A)));

        y(A) = R(A)*sin(theta);% calculates y/l
        z(A) = R(A)*cos(theta) + Zo(A); % calculates z/l
    end
end
% arranges x/l coordinates in the same format as the y/l and z/l
% coordinates
x_plot = x_plot';
```

```
%Name of Function: calcval.m
%Prepared by: Vaneshen Naidoo
%Purpose: To calculate the values of H,W,Zo,N
%Parameters: co,xinc
%Return Value: value(H,W,Zo,N)

%METHOD:
%1. Inputs the coefficients and the x/l value
%2. If C4 = 0 the output is = C1
%3. Otherwise if C8 = 0 then exclude the last term in the equation when
%calculating the output.
%4. Otherwise just use the equation provided to calculate the outputx

function [value] = calcval(co,xinc)

    if co(4) == 0
        value = co(1);
    elseif co(8) == 0
        value = co(1) + co(2)*((abs(((xinc + co(3))/co(4))))^(co(5)));
    else
        value =  co(6) + co(7)*((co(1) + co(2)*((abs(((xinc +
        co(3))/co(4))))^(co(5))))^(1/co(8)));
    end
```