WING TRAILING VORTEX PATHS IN FORMATION FLIGHT

William P. Tipping-Woods
Supervisor: Prof. C. Redelinghuys

A dissertation submitted to the Department of Mechanical Engineering at the University of Cape Town, in partial fulfilment of the requirements for the degree of Master of Science in Mechanical Engineering.

September 2014

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.

2. I have used the $ISO/690 - Numerical Reference$ convention for citation and referencing. Each significant contribution to, and quotation in, this dissertation from the work(s) of other people has been attributed, and has been cited and referenced.

3. This dissertation is my own work.

4. I have not allowed, and will not allow anyone to copy my work with the intention of passing it off as their own work.

Signature: .....

Abstract

Formation flight has been shown to reduce the induced drag for a formation of aircraft. The mechanism by which this is achieved is caused by the wake velocity field of the aircraft. This field is dominated by wing-tip trailing vortices. The paths of these vortices become too complex for rigid wake models downstream of the second aircraft in the formation. To tackle this problem, a combined vortex lattice and vortex filament numerical model was developed. For each simulation the vortex lattice model determined the lift distribution which was applied to the vortex filament model. The vortex filament model used the Burnaham-Hallock vortex profile with a core size of 5% of the wing span to eliminate numerical instabilities. Individual components of the model were verified successfully against literature and the overall approach was validated against wind tunnel data. The wind tunnel data was extracted from apparatus designed and build as part of this study. The apparatus consisted of two NACA 0012 rectangular planform wings mounted in various formation positions and a tuft grid placed downstream of the wings to visualise the vortex paths. Test were performed with both wings at 8° angle of attack. Span-wise wing-tip overlap distances were set at 38%, 10%, 0% and -10% of the span, where 0% implies wing-tip alignment and a positive value indicates a wing-tip overlap. Vertical separations were set at -3%, 0% and 3% of the span for each span-wise wing-tip overlap condition apart from 38% which was only tested at 0 vertical separation. The formation outboard vortex paths were predicted well within the 3% span accuracy of the tuft grid. The predictions of the paths of the formation inboard vortices, however were less accurate. The errors were attributed to a combination of bias errors in the experimental apparatus as well as the pseudo-viscous effects of the Burnham-Hallock vortex profile.

Acknowledgements

Principally, I would like to thank my supervisor, Prof. Chris Redelinghuys, for his support and guidance throughout this project. I would also like to thank the other members of the Aeronautics Research Group at the University of Cape Town, Drewan Sanders and Jordan Adams. The time spend learning from all of their excellent knowledge of aeronautics was fascinating, humbling and endlessly enjoyable. The funding provided by the National Aerospace Center and the University of Cape Town allowed me to pursue my passion for aeronautics and is greatly appreciated.

Thanks must go to Glen Newins and his team, especially Dillon Jacobs and Hubert Tomlinson, for their professional advise and hard work with my experimental apparatus.

I would like to thank my parents, Cedric and Brigid, for their unwavering support and instilling the ambition that defines many of my decisions. Thank you to my sisters Dianne and Theresa for providing me with clarity and many forms of support. Finally, a thank you to Christine for her love, understanding and motivation.

Contents

List of Figures

List of Tables

Nomenclature

Symbols

$\bar{\eta}$	$\frac{\Delta y}{b}$	
$\bar{\mu}$	$\frac{r_c}{b}$	
$\bar{\sigma}$	Downwash Influence Factor	
$\bar{\tau}$	Moment Influence Factor	
$\bar{d}$	Representative Distance	m
$\bar{e}$	Oswald Efficiency Factor	
$\bar{y}$	Centroidal Distance	m
ϵ	Tunnel Blockage Correction Factor	
Γ_o	Maximum Circulation	$\mathrm{m^2 s^{-1}}$
Γ	Circulation Around the Body	$\mathrm{m^2 s^{-1}}$
μ	Dynamic Viscosity	$\mathrm{kgm^{-1} s^{-1}}$
ρ	Density	$\mathrm{kgm^{-3}}$
σ_e	Equivalent Stress for Distortion Energy Theory	Pa
$\sigma_{1,2,3}$	Principle Stresses	Pa
σ	Stress	Pa
τ^*	Tunnel Shape Factor	
τ	Shear Stress	Pa
$\vec{a}$	Direction Vector of Vortex Segment	
$\vec{b}$	Unit Vector of the Induced Velocity at Control Point	

$\vec{l}$	Vortex Filament Segment Length	m
$\vec{q}$	Induced Velocity at a Point	ms^{-1}
$\vec{r}$	Distance to the Point	m
$\vec{u}, \vec{v}, \vec{w}$	velocities in the $\vec{x}, \vec{y}, \vec{z}$ directions respectively	ms^{-1}
$\vec{x}, \vec{y}, \vec{z}$	Direction Axis	
A	Cross Sectional Area	m^2
AR	Aspect Ratio	
b	Wing Span	m
C	Dimensionless Coefficient	
c	Wing Chord	m
C^*	Cross Sectional Area of Wind Tunnel Test Section	m^2
C_{L_α}	Lift Slope Coefficient	
D	Drag	N
d	Diameter	m
F	Force	N
h	Height	m
I	Area Moment of Inertia	m^4
L	Lift	N
l	Length	m
L'	Lift Per Unit Span	Nm^{-1}
L^*	Rolling Moment	Nm
M	Moment	Nm
M^*	Pitching Moment	Nm
n	Number of Vortex Segments	
N^*	Yawing Moment	Nm
nf	Number of Filaments	
R	Reaction Force	N

r	Radius	m
r_c	Vortex Core Radius	
r_I	Distance Between the Body Axis and the Axis of Rotation	
Re	Reynold's Number	
S	Planform Area	m^2
U_F	Corrected Tunnel Speed	ms^{-1}
U_T	Un-Corrected Tunnel Speed	ms^{-1}
V	Velocity	ms^{-2}
w	Width	m
y	Wing Thickness	m

Subscripts

∞	Free Stream
$\vec{x}, \vec{y}, \vec{z}$	direction axes
brd	Braided Nylon Line Used in Tuft Grid
$circ$	Circle
cm	Center of Mass
cyl	cylinder
D	Drag
dr	Profile Drag
$gmts$	Grid Mounts
I	Area Moment of Inertia
i	indexing variable
j	Leading Aircraft
jk	Influence of Leading Aircraft on Trailing Aircraft
k	Trailing Aircraft
L	Lift

l	Rolling Moment
M	Moment
max	Maximum
min	Minimum
$rect$	Rectangle
res	Resultant
$sqrod$	Square Rod
tg	Tuft Grid
y	yield

Abbreviations

$RMSD$	Root Mean Square Difference
VFM	Vortex Filament Method
VLM	Vortex Lattice Method

Chapter 1

Introduction

1.1 Background

Investigations to date have confirmed that there is a significant aerodynamic advantage to be gained through the formation flight of two aircraft. Most notably, induced drag savings of 20% and fuel savings up to 18% were achieved using two F/A-18 aircraft flying in a close proximity formation [1,2]. Extended formation flight, defined by stream-wise separation distances of more than 10 spans between the aircraft, has also been investigate as a safer alternative to close proximity formation flight and proved advantageous in theoretical studies showing a $30-40\%$ induced drag reduction [3]. The practicality of long distance extended formation flight for current commercial airline schedules has been investigated and found to reduce overall operating costs by between 2% and 2.6% depending on the size of the cooperating fleet on established long haul flight schedules [4]. The necessary control capabilities have also been proven using the C-17 Globemaster with its current autopilot system achieving a $7-8\%$ fuel flow reduction [5].

Two main approaches have been used extensively in the modelling of formation flight. Rigid wake models, such as the rigid horseshoe vortex model and the vortex lattice method, and dynamic models such as the vortex filament method. The rigid wake models do not allow vortices to interact and are useful for two aircraft in close proximity. Once the wake behind the second aircraft in the formation is of interest, the flat wake methods become less reliable. The vortex filament method is less frequently used due to its relative complexity, but allows the trailing vortices to move about and interact with each other. A linearised vortex filament method has been successfully applied to multiple aircraft in formation by Ning [3]. This investigation highlighted the potential for the formation inboard vortices to move around and adversely affect aircraft further downstream in the formation.

1.2 Objectives and Scope of Investigation

The objective of this dissertation is to describe the paths of the mutually interacting wing trailing vortices in close proximity formation flight. In order to achieve this, a numerical model is developed in MatLab and an experimental apparatus is designed and built to validate the numerical approach. A combined vortex lattice method and vortex filament method is used to simulate two wings in various close proximity formations. The model is capable of simulating wings of different sizes under different different load distributions which can either be user defined or calculated by the vortex lattice method. The experimental analysis is carried out in the low speed closed loop wind tunnel at the University of Cape Town. The apparatus consists of two wings mounted in close proximity formation and a tuft grid mounted downstream of the trailing wing for flow visualisation purposes.

1.3 Plan of Development

The dissertation begins by describing the theory of aerodynamically advantageous formation flight, highlighting the wake structure as the critical factor. The various sections of the wake are then investigated alongside the benefits and restrictions of various modelling techniques. The chosen numerical method is described in detail in Chapter 3 with reference to the relevant mathematical equations and procedures presented in Appendix A. The wind tunnel apparatus design considerations are outlined in Chapter 4.3 with reference to the design calculations and wind tunnel blockage calculations in Appendix B as well as the design drawings in Appendix C. The basic experimental methods are also described in Chapter 4.3. The results and discussion in Chapter 5 begins by verifying the numerical model with results from literature for a single wing before validating the model against wind tunnel data for two wings. Chapter 5 also provides more detail of the experimental methods and data analysis techniques, and details the statistical approach to the two wing validation of the numerical model. Finally the conclusions are drawn and recommendations made in Chapter 6 and Chapter 7 respectively.

Chapter 2

Literature Review

2.1 The Theory of Formation Flight

The trailing aircraft achieves an aerodynamic advantage as it sees an increase in effective angle of attack due to its position in the up-wash generated by the lead aircraft [1,2,6]. This allows the trailing aircraft to trim to a smaller pitch angle compared to the lead aircraft, effectively rotating the lift and drag vectors as shown in figure 2.1. The lead aircraft creates this up-wash as a by-product of the pressure difference between the top and bottom surface of the wing as it generates lift. This pressure difference causes the air flowing over the top surface of the wing to move inboard of the wing tip and the air flowing under the bottom of the wing to move towards the tip of the wing, creating a vortex sheet when the flow meets at the trailing edge. The vortex sheet on each wing quickly rolls up into what is effectively a pair of counter rotating vortices trailing each wing slightly inboard of the wing tip. This produces downwash inboard of the vortex and an upwash outboard of the vortex [7]. The strength of this vertical velocity component of the flow determines the magnitude of the induced drag saving as well as the characteristic induced rolling moments experienced by the trailing aircraft. It is important to understand and predict the characteristics and position of these vortices in order to safely take advantage of the energy saving potential that they hold.

2.2 Vortex Wake Characteristics

The velocity field in the wake of an aircraft is dominated by the wing-tip trailing vortices. Donaldson [8], Spalart, [9], Rossow [10] and Gerz [11] give extensive reviews of the current literature describing the complex wake structure of conventional aircraft through experimental and theoretical studies. This vortex dominated wake can be analysed through three regions conveniently labelled by Takahashi et al. [12] as the roll-up region, the plateau

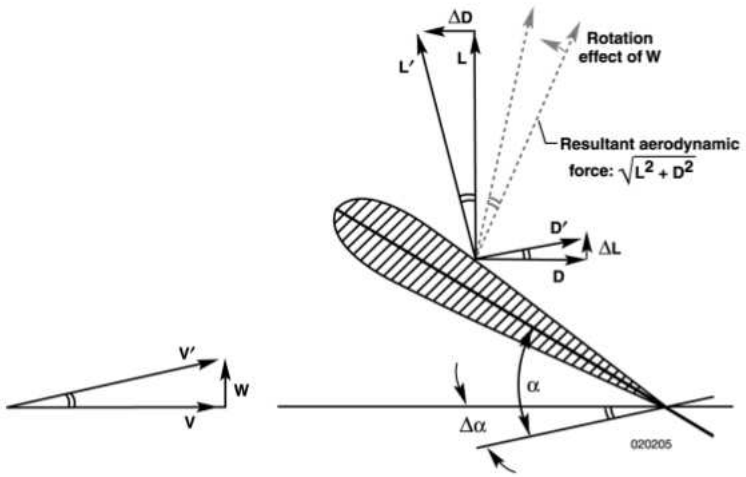


Figure 2.1: Rotation of lift and drag vectors due to upwash [1]

region and the decay region. The consensus is that, in a relatively unturbulent atmosphere, the trailing vortex sheet rolls up quickly, forming two counter rotating vortices of equal strength. The structure of these vortices is stable into the far field as the pair descends under self induction. Eventually large scale instabilities manifest and the vortices decay into a region of turbulence.

2.2.1 Roll Up

The roll up region is loosely considered complete when the wake has developed into a few well defined vortices [11]. Reports of the length of time taken for this to be complete range from almost instantaneous completion at the trailing edge [13,14] for a simple wing to about 5 to 9 spans downstream for a flapped wing [9,15]. The actual distance, as well as the structure and position of the rolled up vortices, depends highly on the wing span and loading [8,10]. Czech [15] shows that flap vortices (counter rotating compared to the wing-tip vortices) can be preserved up to 20 spans downstream, but the tail vortices (co-rotating with the wing-tip vortices in this case) have merged with the wing-tip vortices between 8 and 10 spans downstream. Although various investigations of specific configuration have been conducted, the quantitative rules of exactly how different parameters affect the roll up process are not well defined.

Prandtl applies a conservation of mechanical energy to model the roll up process by replacing the vortex sheet at the trailing edge of the wing with two equal strength vortices spaced according to the wing lift distribution.

This method assumes that the energy lost to induced drag must be equal to the energy shed into the wake and contained in the fluid motion in the Trefftz plane. Betz extended this understanding to provide a more rigorous model of the vortex structure linking it to wing drag as well as lift span loading [8]. This model has helped explain many observations. The more the lift distribution differs from uniform loading, the more the Betz model predicts the vorticity will be widely distributed. His method is only really applicable to two counter rotating vortices due to the assumptions that are made in the process, namely that the streamlines of each vortex do not influence each other. This results in concentric streamlines, which is only true due to the mutual induction and descent of the vortex pair [8]. This method provides a good description of the wake of a single aircraft after roll up is complete but does not tackle the questions of the process itself. A better understanding of the actual roll up process is important to ensure that the characteristics of vortex interaction can be understood and modelled to accurately simulate the wake structure for any combination of aircraft and control surface deflection. The fact that the roll-up process and most vortex interactions are considered to be inviscid due to the short time that they take [8, 9, 12, 16, 17] allows the roll up process to be modelled to good first order accuracy using numerical methods such as the vortex filament method.

2.2.2 Plateau

The plateau region extends for between 20 and 200 spans depending on the atmospheric conditions [16]. This region generally consists of a pair of counter rotating vortices which are stable and descend at a constant rate under mutual induction [10]. The rate of descent depends on the vortex strength and the separation distances [8]. For perspective it is interesting to note the report that for large aircraft, the vortices descend side by side at a rate of 120m to 150m per minute and level off 200m to 275m below the flight path [9, 18]. This levelling off could be due to the buoyancy effects, described by Scorer [19], of the warm exhaust gasses in the vortex cores. The stability of this plateau region is what gives promise to the extended formation flight investigations.

The Helmholtz vortex theorems state that, a vortex must extend to the boundary of a fluid or form a closed loop in the fluid, the strength of a vortex is constant along its length, fluid initially free of vorticity will remain free of vorticity [20]. These three laws form the foundation of the lift and wake theory which has been expanded from Pandtl's lifting line theorem including the horseshoe vortex methods, the vortex lattice methods and the vortex filament methods. They also lie at the heart of the idea that vortex wakes are stable into the far field and could hold energy saving potential.

2.2.3 Instabilities and Decay

There are multiple mechanisms described as contributing to the destruction of vortices [6, 8–10]. The long wavelength Crow instability [21] is the dominant phenomenon in the extended wake of a cruising aircraft in the absence of large scale turbulence [10]. These sinusoidal oscillations are in a plane angled to the horizontal so that the vortices grow closer together at the peak of the wave and further apart at the trough. The instabilities grow in amplitude until the vortices touch and create vortex rings far downstream in the wake. This instability was observed in practise and has been successfully described by vortex filament methods with only a small perturbation input. Vortex breakdown, sometimes incorrectly labelled as vortex bursting [9], is described by Donaldson [8] as the process of the axial velocity in the core of a vortex interacting suddenly and strongly with and tangential velocities around the vortex [16]. Both processes seem to result in a larger region of dispersed vorticity where angular momentum is conserved. The confusion between the two terms results from the fact that their outcome is similar and that bursting is not well understood [11]. Vortex bursting and breakdown can both occur in a single vortex, unlike the Crow instability which occurs only in a pair of vortices. The conservation of angular momentum through the bursting and breakdown processes imply that they do not decrease the potential aerodynamic advantage held in the wake [9, 10]. A viscous decay mechanism, vortex ageing [8], is sometimes over predicted in literature. General consensus is that it happens too slowly to be a dominant feature in the study of formation flight.

In wind tunnel investigations it is important to recognise the instabilities that are induced by the wind tunnel itself. This type of instability is generally referred to as vortex wandering or meandering [9,10,14]. It is of a much shorter wavelength than the Crow instability and begins in the near field, 1 chord length downstream [22], in wind tunnel investigations. It is present in a single vortex as well as a pair of vortices and is not well understood. The frequency of the instability wave has been reported with a difference of an order of magnitude between studies [12, 22] and the amplitude of the movement has been reported to be as large as a few centimetres. The frequency seems to be random and independent of Reynolds number. Although it is lower than the frequency of the turbulence that it is associated with, the wandering frequency is influenced heavily by turbulence levels in the wind tunnel. Small scale grid turbulence also induces a bend in the vortex path [23]. Wake vortex paths are also extremely sensitive to apparatus configuration, especially to the introduction of measurement probes [24].

2.2.4 Cross-Sectional Structure

Concentrated vorticity in inviscid, incompressible, two dimensional flow results in a velocity potential field throughout the fluid. This velocity field is characterised by an inverse relationship between induced velocity and distance from the origin of vorticity. It can be seen as that induced by a section through an infinite vortex filament of strength Γ, which is simply the circulation around any path enclosing the filament and is constant for a given vortex filament [7].

The Biot-Savart law, equation A.12, is a result of this potential theory which predicts the tangential velocity around a section of a vortex filament and can be integrated to give the induced velocity at a point due to an infinite vortex, a semi-infinite vortex or a finite straight vortex filament [7]. A derivation of this equation and its results can be found in reference [25]. This model results in an asymptotically infinite velocity as you approach the filament itself. This singularity cannot exist in a fluid and a vortex generated by an aircraft wing is a combination of irrotational flow around a viscous core of rotational flow [12]. The physical manifestation of this has been shown in the rolled up wake of an aircraft numerous times through experiment both in wind tunnels [26] [27] and in flight tests [28] [29] [30] and has been modelled theoretically through a number of different vortex profiles. The difference between the profiles does not have a significant effect on wake encounters modelled using the Vortex Lattice Method [18] if the aircraft are similar in size. However, the differences may influence the interaction of vortices in a dynamic wake modelling effort such the Vortex Filament Method.

Trailing vortex profile models range form the very simple Burnham-Hallock model through to more complex models such as the Lamb-Oseen model [11, 18, 31]. The choice of vortex model is heavily dependent on its intended application. The Lamb-Oseen model incorporates viscous effects of increasing core diameter with time and is necessary for some studies into the far field wake and for the understanding of vortex decay for optimisation of airport capacities [14, 18], but it has been shown to over estimate the viscous effects of vortex decay. A complex model like this would not be practical in modelling the wake roll-up process through numerical methods. Models such as the Burnham-Hallock, which are more easily integrated and simpler to utilise, have shown good correlation with experiments for the rolled up wake of cruising aircraft [32]. It is easier to justify substituting this type of model into a numerical scheme in preference to the Biot-Savart law because it does not contain a singularity and it closely matches the Biot-Savart velocity profile outside the core region. This is important because the singularity at the center of a Biot-Savart filament tends to be unwelcome in numerical

schemes, preventing convergence to a stable wake velocity field [33].

2.2.5 Core Size

The vortex core size is defined as the diameter between the two turning points of the velocity profile and is usually determined through experimental measurement. This measured core size for a specific experimental investigation is often applied to the simplified vortex profiles discussed in subsection 2.2.4. Numerous methods (the Betz Method [8] and Augmented Betz Method [3] being two of the most well known and accurate) have been employed to calculate the core size and structure in the wake of an aircraft, however the many stochastic variables make it difficult to find a generalised calculation, developed from first principles, that works for all cases [9]. Measurement techniques used to find vortex core sizes include Particle Image Velocimetry [14] and Hot Wire Anemometry [23]. As vortices are extremely sensitive to the presence of probes, PIV is considered more accurate as it is less intrusive. Even with PIV, the measured core size and position is sometimes overestimated due to vortex meandering. Conditional sampling and statistical methods help the investigation but do not eliminate the problem [24] [9]. Core sizes in the rolled up wake of an aircraft or experimental model are measured to be small, in the region of 0.7% to 6% of the wing span. There is some uncertainty surrounding the larger core sizes reported due to the measurement techniques [9]. The size of the core depends on the wing loading, and is reported stable up to 200 spans into a calm wake [34]. This stability measured so far downstream in the wake gives more promise to the investigation of energy saving in formation flight. Furthermore it gives weight to the idea of simplifying the vortex profile by neglecting decay in numerical models through the plateau region [12] [23] [14] [14] [9] [34]. In numerical studies, the core size of the vortex filaments is often determined by computational limits rather than experimental data [33]. The reason for this is that when the resolution of the model is increased and the vortex filaments are positioned close together, the large

2.3 Vortex Interactions

The experimental analysis of vortex interaction is challenging because it is a dynamic process and the cores are often not well defined [35]. Particle Image Velocimetry (PIV) seems to be the most promising approach [27]. The mutual interaction of two or more vortices can have three results depending on the initial conditions. They can merge, move apart or remain next to each other until long wavelength instabilities develop [36]. The spacing, circulation strengths, circulation directions and core size all define the outcome of the interaction [16] which can be broken down into two groups, counter-rotating interaction and co-rotating interaction.

2.3.1 Co-Rotating Vortex Pair

A co-rotating vortex pair of similar strength will orbit around each other and either merge if their cores overlap or move apart if they do not overlap [17] [37] [15]. If the pair starts out with a sufficiently large separation distance they will continue side by side until decay [8]. The merger happens quickly, it is complete within 1 orbital period, and is a three dimensional inviscid process in which the strength of the two vortices combine conserving circulation, energy and momentum [9]. The time to merger depends on the strengths of the vortices as well as the ratio of the vortex strengths. Stronger vortices and a bigger difference between the vortex strengths result in a faster merger process [17]. As the vortices rotate around each other, the weaker vortex is stripped into smaller filaments before being captured. The merger process has negligible affect on the size of the core and the resulting vortex stabilises immediately. This merger process is the basis of vortex roll-up.

2.3.2 Counter-Rotating Vortex Pair

The conservation principles of co-rotating vortex pairs holds true for counter rotating vortices where the circulations cancel each other out. The dynamics of a counter rotating pair are very different however. Depending on the circulation difference between the vortices, they are known to move in the cross-sectional plane by more than half a span during their interaction. The movement of the pair is in the direction from the initial positions of stronger vortex to the weaker vortex in an arc. The weaker vortex begins to develop short wave instabilities early on before it pinches off into rings of vorticity which are ejected into the wake. [16] The stronger vortex develops decay instabilities in the far wake. The structure of the final vortex is highly dependent on the relative strength of the vortices. The closer the strengths are of the vortices to one another, the more they will cancel each other out and result in a region of dissipated vorticity. This process takes much longer than the merger of two co-rotating vortices and results in a final vortex of a larger core diameter and weaker peak velocities. Investigation into the counter rotating pair created by the wing-tip vortices of a wing and tail plane configuration, with a -0.3 circulation ratio, showed that this merger is complete 20 spans downstream [35]. The counter-rotating vortex system is important for the study of formation flight because it will occur at the wing-tips inboard of the formation. The amount by which the circulation of the vortices cancels each other out is closely linked to the induced drag savings on the trailing aircraft [27]. The characteristics of the interaction and the paths of the vortices are extremely sensitive to their relative strengths and positions [36]. It is imperative to understand this interaction in order to predict the effects of increasing the size of the formation.

2.4 Wake Modelling in Formation Flight

The way in which one models formation flight depends on the individual aspects that are being investigated. A high fidelity simulation of a formation of aircraft which includes accurate aerodynamic models and a turbulence model coupled with flight mechanics and control would be extremely computationally expensive. Individual aspects are investigated in detail and then simplified as much as possible as the model grows and includes more aspects. For instance, once the roll-up process is understood under different conditions and trends found and modelled simply, the results can be used for flight models which would spend more computational time on the control aspects and less on the aerodynamics. An aim of this dissertation is to further the understanding of the paths of mutually interacting trailing vortices in order assist in the selection of a wake model suitable for formation flight modelling. Spalart, in his review of the knowledge of airplane trailing vortices, states that a model which predicts the paths of trailing vortices to 30% (of span, assumed by author) accuracy considering all the stochastic inputs, would be very impressive [9].

2.4.1 Flat Wake Models

Prandtl's lifting line theory [7, 25] has been applied to varying degrees of complexity with application to modelling aircraft and wake in formation flight. It consists of a series of horseshoe vortices placed along the quarter chord line of a wing in such a way as to model the lift distribution across the wing. The relationship between the lift and circulation across the wing is defined by the Kutta-Joukowski theorem and the trailing filaments of each horseshoe vortex make up the trailing vortex sheet. This sheet accounts for the loss in circulation as the lift per unit span reduces towards the wing-tips. Through this theory it is found that an elliptical lift distribution results in the lowest induced drag and is therefore the most efficient lift distribution. Taking these results and simplifying the method to a single horseshoe vortex per wing allows a much faster approximation of the aerodynamic coupling of formation flight. The span of this horseshoe vortex model is usually set to $\frac{\pi b}{4}$ for elliptically loaded wings [11, 18, 31, 38].

The vortex lattice method places a series of horseshoe vortices on the average surface of a wing and solves for the zero normal flow condition at control points between the filaments. The arrangement of the filaments and control points is generally distributed sinusoidally across the wing to give higher resolution towards the wing-tips where there is a higher circulation gradient. The arrangement can consist of horseshoe vortices placed along the quarter chord line, however this method neglects chord-wise lift distributions. To include chord-wise lift distributions, the discretisation of

the wing should take place with horseshoe vortices distributed in both the chord-wise and span-wise directions. In both cases the shed vorticity can either extend to infinity in the wake or form a closed loop by joining the trailing edge of the vortices with a starting vortex. Both methods therefore follow the Helmholtz vortex laws. The chord-wise distribution is less important to wake modelling than the span-wise distribution and is often neglected in wake models. The chord-wise discretisation is more important for detailed analysis of wing loading especially if the wing is geometrically complex. More detailed descriptions of the implementation and differences between the Horseshoe Vortex method and the Vortex Lattice method are presented in Chapter 3. Some applications and results from literature are presented below.

Blake shows that the horseshoe vortex model can grossly over predict the benefits of formation flight [38]. One application essentially combines the two wings into one long, low aspect ratio wing by placing the formation inboard vortices directly on one another. This results in a 100% induced drag reduction on the trailing wing which is unrealistic. More tactful application of the horseshoe vortex method can provide better results. Bizinos applies the horseshoe vortex model to simulate the effects of formation flight in the study of turbulence effects on passenger comfort. The lead aircraft is modelled by a single horseshoe vortex which is sufficient as the much smaller vortices of the horizontal stabiliser is captured by the large vortices of the main wing very quickly in the wake [15]. The trailing aircraft is modelled by two horseshoe vortices, one on the main wing and one on the horizontal stabiliser. The second horseshoe is necessary to capture all the aerodynamic effects of the turbulent simulation on the trailing aircraft. In this study the horseshoe vortex model was appropriate because the wake prediction was not the core focus, rather the accelerations experienced by a trailing aircraft in a wake affected by turbulence were.

The Vortex Lattice Method has been used extensively to predict the aerodynamic forces associated with formation flight [39]. Flight test data presented by Ray [1] as well as by Vachon [2] was compared to predicted induced drag saving from both the single horseshoe model as well as the vortex lattice model which had previously been presented by Blake [40]. This comparison showed that the flight test data matches both models well around the position of maximum aerodynamic advantage. In a more recent study, Blake [38] finds that although the general trend is accurate, the vortex lattice code used in the study over predicts the induced drag saving found through wind tunnel experiments by an average of 15%. The discrepancy is attributed to increased flow separation over the trailing wing caused by the 8 degree increase in effective angle of attack generated by the lead wing.

The fact that vortices in practice cannot sustain a pressure difference and begin to roll up, interact, move around and decay is likely to account for some of the discrepancy as well. Morgan [41] found a very similar percentage over prediction of the vortex lattice code compared to flight test data. Hemati et al. [42] apply the horseshoe vortex method to simulate the rolled up wake of the lead aircraft and a vortex lattice code to investigate the effects on the trailing aircraft. The simple lead aircraft model is appropriate and reduces the computational cost of the simulations. A recommendation from the study is that the relative motions of the vortex wakes should be studied further to obtain a clearer picture of formation flight. Overall the VLM seems to give good results in predicting the trends of the loads on the trailing aircraft, but a poor representation of the wake of the lead aircraft. The induced drag saving is consistently over predicted, especially with similar sized aircraft, and even when experimental results are used as an input for the induced upwash velocities experienced by the trailing wing [43].

Much of the research into the potential induced drag savings of formation flight acknowledge the effect of the relative positions of the aircraft as a defining factor of the magnitude of the aerodynamic advantage. It is more accurate to say that position of the the trailing aircraft relative to the vortex wake of the lead aircraft determines the induced drag savings. The predicted position of maximum aerodynamic advantage is therefore dependent on the wake model used for the lead aircraft. Inasawa shows experimentally that the most aerodynamically advantageous position for a trailing aircraft is close to the position of greatest interaction between the wingtip vortices inboard of the formation [27]. In this case the most advantageous position is a 5% wingtip overlap. The position of greatest interaction is 2.5% and gives a slightly lower advantage of 18% compared to the maximum advantage of 24%. It is important to note that these percentages are specific to the configuration of this experiment. At position of maximum advantage, the leading vortex impacts with the trailing wing and breaks into two vortices, one on the top and one on the bottom of the wing. The bottom vortex merges with the trailing wing vortex and decreases it's strength by up to 40% which is a major contributor to the induced drag saving of the trailing aircraft. This position would however be uncomfortable for the trailing aircraft passengers. In formation flight it is more likely that the aircraft would fly with a small span-wise separation between the wing-tips [44]. This position will still give good drag savings but will be easier to maintain and more practical in terms of passenger comfort [31].

The discussion so far has concentrated on the two aircraft formation. The concept of larger formations of aircraft has also been investigated and shown to have promise with some limitations. Due to dynamic throttling and

control deflection associated with station keeping, the percentage increase in fuel saving for the entire formation is reduced per aircraft added. As more aircraft enter the formation in the numerical vortex lattice model used by Blake [40], the upwash region is compounded and the lead aircraft has to increase its cruise speed and altitude in order to allow the trailing aircraft to not have to reduce throttle too much to maintain position. This will also reduce the fuel saving benefits of adding many aircraft to the formation. The flat wake model, as used by Blake, is even more problematic when applied to larger formations. The way in which the wake of multiple aircraft interact may have a big influence on the aircraft further back in the formation. As discussed in the vortex interaction section, a counter rotating pair with a small separation distance can move around significantly. This is just the case that will occur during formation flight with the vortices inboard of the interaction. This wake interaction has been identified as an important consideration in the analysis of the formation flight of more than two aircraft in particular [3, 42]. Ning et al. investigated this scenario using a linearised vortex filament method. It was found that the forward-most vortex pair in a multiple aircraft formation may roll over and adversely effect aircraft further back in the formation; thus suggesting that formations of more than 3 aircraft are not particularly advantageous [3].

2.4.2 Dynamic Wake Models

In many situations the trailing vortices of an aircraft can be modelled by assuming that the wake is rolled up into two distinct and stable vortices. With the introduction of extended formation flight of multiple aircraft, this model is no longer sufficient and vortex interaction must be considered. To fully understand the dynamics of the interaction of the wake a two aircraft, the vortices must be modelled by a dynamic wake simulation. Two approaches to this are the Eularian, CFD approach of Large Eddy Simulation (LES) and the Lagrangian approach of the vortex points and filaments. Both approaches have been used successfully to model different aspects of formation flight. CFD provides a comprehensive view of fluid flow but is extremely computationally expensive in the high gradient regions found in a vortex and may not always be practical for the simulation of multiple aircraft in formation flight. It is often used to check smaller domains where higher fidelity is needed or where compressibility and viscosity are important factors. For instance, Ning et al. investigated the concept of formation flight using a linearised vortex filament method to simulate the wake propagation for various formations of three aircraft [3] before moving to a CFD approach to investigate the compressibility effects for a smaller 2 ship formation. The linearised vortex filament method assumed a zero vertical and spanwise initial wingtip separation with the vortex profile calculated using an augmented Betz method for the roll-up process. Vortex decay was included with the

conservative Holzpfel model which is developed from the NavierStokes equations with the assumption of plane rotating flow. The compressibility study used a 2-dimensional Euler solver and focused on the vortex inboard of the formation of the lead aircraft and reduced the predicted drag reduction to 10% for the formation with a 5% wing-tip separation which is more in line with flight test results [44].

Vortex filament methods have been successfully applied to the wake study of aircraft to simulate the roll up and plateau regions [33] as well as the decay region for individual aircraft as well as multiple aircraft. The method can accurately model the long wavelength instabilities of the decay region given a small input perturbation [21, 36, 45, 46] as well as the short wavelength instabilities associated with counter-rotating vortex interaction. The VFM has it's origins in two dimensional time dependent numerical calculations using point vortices in the Trefftz plane. The Trefftz plane being a theoretical cross section of the aircraft wake, perpendicular to the free stream, located at an infinite position downstream of the wing. The theoretical plane allows certain assumptions to be made, depending on the application. In this case the assumption is that the bound vorticity on the wing and starting vortices does not effect the vortex wake. The point vortices of the two dimensional calculations are arranged to simulate the vortex sheet of the trailing edge which interact and roll up into two distinct vortices. These results were shown to be in good agreement with the Betz method [8]. Although vortices are three dimensional, two dimensional simulations are often sufficient to describe their behaviour [47]. The VFM extends this concept to three dimensions, although it neglects axial flow, by replacing the point vortices with straight vortex filaments connected with nodes which allow them to interact and move around to line up with the streamlines of the flow. In any numerical wake simulation, the vortex profile must be defined along with the distribution of the vortex filaments, both of which can be stumbling blocks. It is preferable to use the Biot-Savart law, with its roots in potential flow theory, as far as possible for VFM. This is not always possible though as the singularity at the filament tends to make the simulations unstable under certain conditions [8, 48].

Ehret links core size and filament length through a proportional relationship to define conditions by which the simulation will converge [33]. A danger if imposing a core size on the vortex filaments is that it introduces a pseudo-viscosity and linearises the roll-up procedure which we know happens quickly and is actually an inviscid process. Unfortunately numerical and computational limitations sometimes make this a necessity. Ehret's application of this method provides accurate results when compared with ground based wake measurements of a 747 [33, 49]. A second step to avoid

the numerical instabilities, which also increases the speed of the numerical calculation, is to allow the filaments to merge, under the conservation principles discussed earlier. The merger would be controlled to occur only once the filaments come within a certain radius of one another [8, 50]. This method reduces the number of filaments through the roll-up process and it is unclear as to the impact of the procedure. The placement of the filaments is of concern here as the increased resolution close to the wing-tips, which is generally preferred, would be destroyed by merging the filaments prematurely. If the vortex filament circulation strengths are defined as constant across the wing, the lift distribution is achieved by moving the filaments into appropriate positions to account for the change in circulation along the wingspan. This technique provides a more intuitive visual representation of the streamlines and flow field than defining the filament positions and adjusting the strengths to achieve a desired lift distribution. The merger process will not preserve this aspect of the flow visualisation as the strengths of the new filaments will change as their components merge. Merging will however result in a better visual representation of the final rolled up core position.

Chapter 3

Modelling Vortex Paths in Formation Flight

To investigate the interaction and the paths of the wing trailing vortices in formation flight, a computer model was developed in MatLab. The model was constrained to two rectangular planform wings in close proximity formation flight. The effects of winds, atmospheric turbulence, stratification and axial flow are omitted. The Vortex Filament Method (VFM) was chosen as the main tool to investigate the interaction of wingtip vortices. Ehret [33] used the method successfully to investigate the wake of a single aircraft and produced results that compared well to experimental data of a Boeing 747 [51]. The method allows a number of vortex filaments to interact on a single wing. The current investigation extends the method to two aircraft flying in formation with different lift distributions by combining it with a Vortex Lattice Method (VLM). The VLM is necessary because the VFM does not have the capability of solving for the circulation distribution due to input simulation conditions, rather it must have a known circulation distribution applied to it. Fig. 3.1 shows a block diagram of the various functions of the program which are discussed in detail in the rest of this section. Appendix E shows a thoroughly commented copy of the MatLab code which describes the intricacies of the methods applied to this investigation.

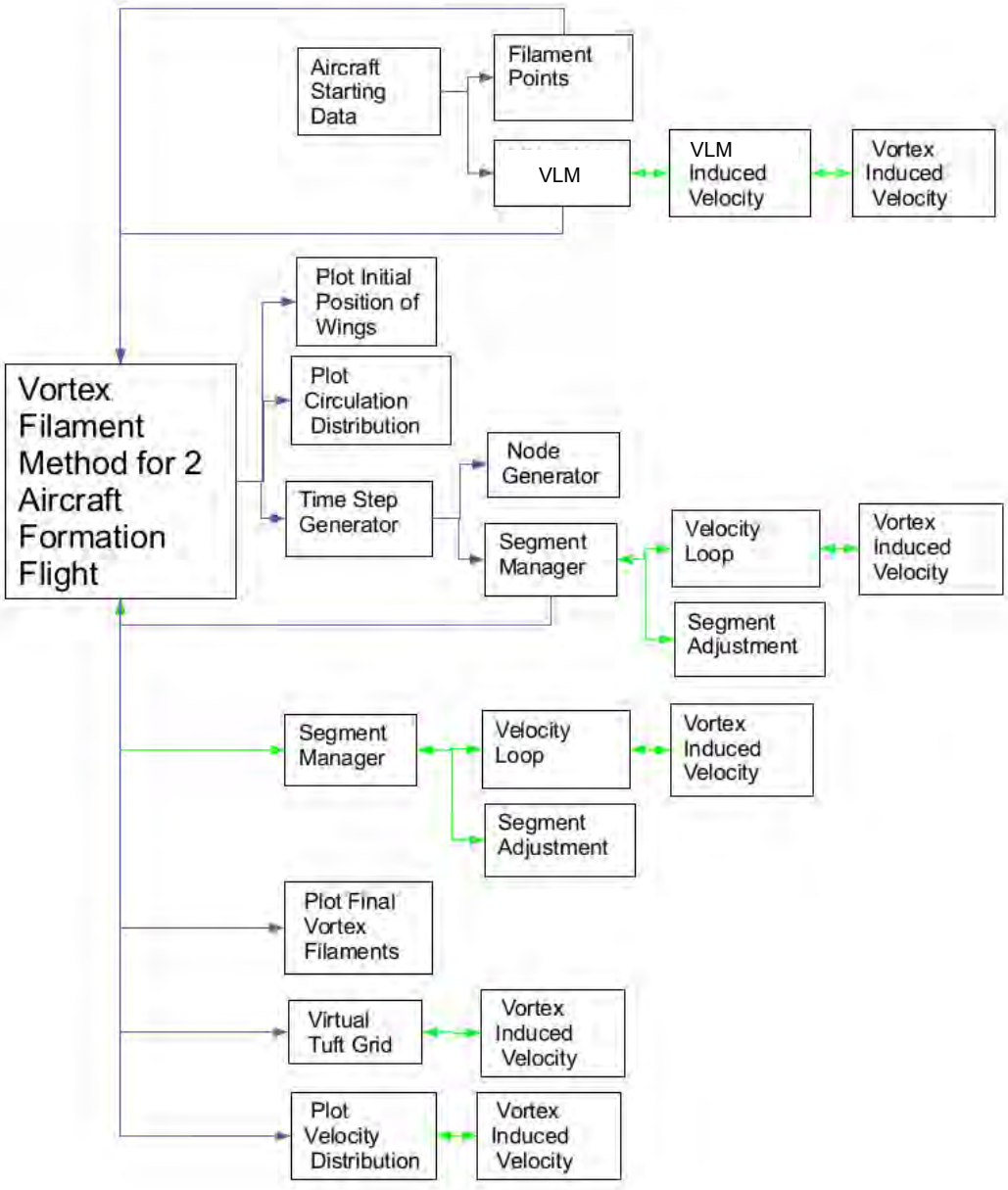


Figure 3.1: Block Diagram of MatLab Program

3.1 Vortex Filament Method for Two Aircraft in Formation Flight

The main script file calls the function files to perform the operations of the program and then presents the data in a graphical form making it easier to interpret, understand and compare to wind tunnel results. The program is capable of simulating various formations under different conditions and to specific levels of accuracy determined by the input parameters. Simulation Condition is one such parameter and is defined as either Wind Tunnel or Cruise. The Wind Tunnel option simulates two wings in a wind tunnel. The air speed and density, wing span and chord are 25 ms^{-1} and 1.2 kgm^{-3}, 0.3 m and 0.048 m respectively. The Cruise option sets the simulation up for two wings flying at 10 km in altitude with an air density of 0.35 kgm^{-3}. The wing span, chord and lift are defined by the parameters of a Boeing 747 under cruise conditions [52]. This simulation was mainly used in the development of the model for validation against results presented by Ehret [33] and Hallock [51], but is a necessity for the study to be relevant to the simulation of formation flight of passenger aircraft.

The lift distribution across wings in a wind tunnel differs from that of aircraft wings in trimmed flight. Distribution is another option in the program that can be set to Symmetric Lift or Non-Symmetric Lift. Symmetric Lift assumes that both wings have an elliptical and therefore symmetric lift distribution and is a closer approximation to two aircraft flying in formation flight than it is for a wind tunnel simulation. The reason for this is that the upwash region that the trailing aircraft flies through will induce a rolling moment on the trailing aircraft. The trailing aircraft will then trim to maintain position using various control surfaces and throttling techniques. Thin wing theory shows that an elliptical lift distribution will result in the lowest possible induced drag [53]. A low induced drag is a consideration for aircraft designers, therefore although the lift distribution will most likely not be perfectly elliptical and symmetric, it serves as a good approximation to assume that a commercial airliner in trimmed flight will have a lift distribution that is close to elliptical. This has also been showed by King and Gopalarathnam [54]. For this situation the lift generated by the trailing wing is defined to be the same as the lift generated by the lead wing for two identical aircraft which effectively simulates a trimmed formation of the two aircraft under cruise conditions.

In the wind tunnel however, the forces applied to the trailing wing to counteract the induced rolling moment due to the upwash created by the lead wing are provided by the support fixed to the wind tunnel and not

by control surfaces. This means that the side of the wing inboard of the formation will generate a greater lift force than the outboard side of the wing and the lift distribution will be far from elliptical. For this scenario, the Non-Symmetric option is selected. The lift of the wings in this case is not defined using the C_L but is rather calculated using the VLM described later on in this chapter. This vortex lattice approach assumes the two wings are at the same angle of attack and does not trim the wings. This more accurately describes the conditions of the wind tunnel set up. At the beginning of each simulation a number of other parameters must be defined:

- Positions of the wings relative to one another.
- Angle of attack of each wing.
- Number of filaments trailing each wing.
- Time period of the simulation.
- Time step.
- Vortex velocity profile (either Burnham-Hallock or Biot-Savart).
- Vortex core size (only defined and used for the Burnham-Hallock profile).

The length of wake to be generated and the length of each segment is controlled by the time period of the simulation and then the time step within the simulation.

3.2 Starting Data for Aircraft VLM

Once the conditions of the simulations are set, the Data for Simulation function takes the variables defined in the script file and applies the values to a structure which forms the complete set of data for the two wings for the entire simulation. Either the Filament Point function or the VLM function is called next to perform the calculations to discretise each wing into a series of horseshoe vortices placed on the quarter chord line of the wing and propagating into the wake where they are connected to starting vortices. The purpose of the starting vortices is to ensure the conservation of Helmholtz's second theorem [20] stating, in part, that a vortex cannot end in a fluid. The Filament Point function is used for the symmetrical lift distribution and the VLM function is used for the the eccentric lift distribution simulations.

The Filament Point function makes use of the elliptical distribution equation (A.3) to calculate the positions of vortex filaments along the quarter-chord line of each wing that will result in an elliptical and symmetrical lift distribution. First the maximum circulation is calculated using equation (3.1) which is the result of integrating the Kutta-Joukowski theorem for an elliptical lift distribution. The full integration is shown in Appendix A. Once the maximum circulation, Γ_o, is known it is divided by the number of filaments per wing to give the $\Delta\Gamma$, which is the change in circulation as each filament is added to the wing. As a new filament is added, the Γ in equation (A.3) increases by $\Delta\Gamma$ and the y-position of the corresponding filament is calculated.

$$\Gamma_o = \frac{4\Gamma}{\pi b} \tag{3.1}$$

3.3 Vortex Lattice Method

The VLM places a series of horseshoe vortices on the surface of a wing which is at an angle of attack to the free stream. The horseshoe vortices are distributed sinusoidally across the span of the wing and their relevant collocation points are placed inside each horseshoe vortex and on the $\frac{3}{4}$ chord line of the wing. The collocation points are also distributed using a sine distribution to find their spanwise locations. The sine distribution is used in order to give a better resolution in the region close to the wing-tips where the change in circulation per span is large. The strengths of the vortices are then solved to ensure zero flow through the surface of the wing at the collocation points. Once the strength of each vortex is known, the lift generated by each horseshoe vortex can be calculated using the Kutta-Joukowski formula and summed to give the total lift across the wing. The difference between the VLM method and the VFM is highlighted in Fig.s 3.2 and 3.3.

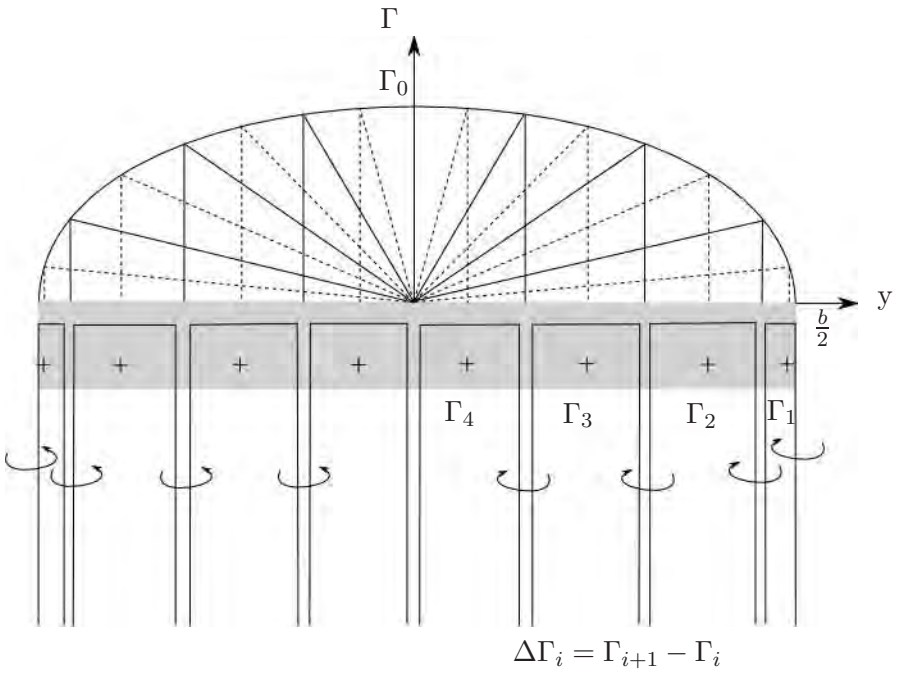


Figure 3.2: Discretisation for the Vortex Lattice Method

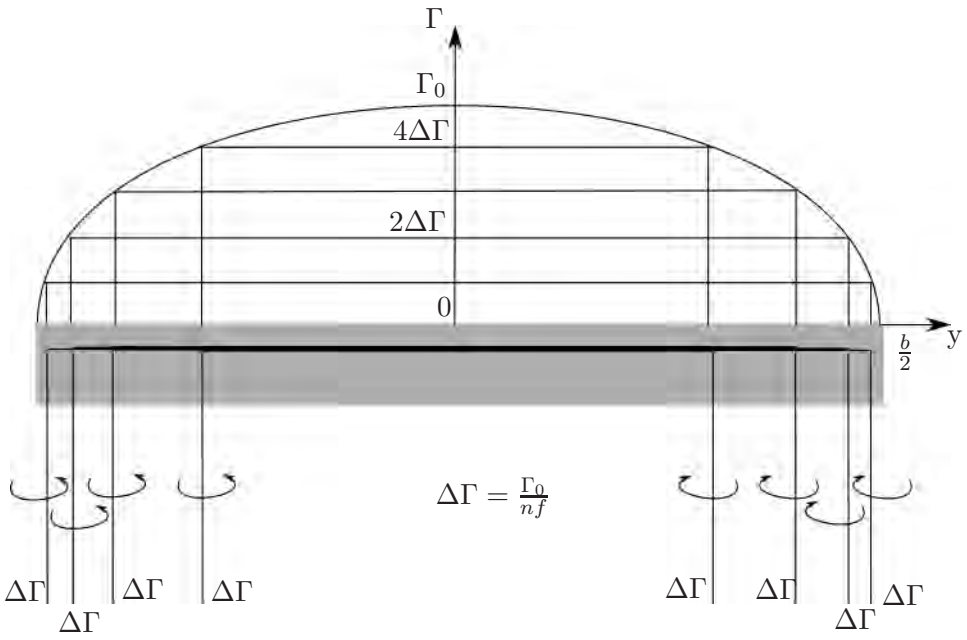


Figure 3.3: Discretisation for the Vortex Filament Method

3.4 Combined VLM and VFM

Both the VLM and the VFM use a numerical distribution of vortex filaments along the surface of a wing to account for the circulation that exists around a lift producing wing according to the Kutta-Joukowski theorem. The difference is that the VFM integrates vortex filaments of constant strength but variable position over the span of the wing while the VLM integrates a predefined length but variable strength series of vortex filaments over the span of the wing. In order to apply the VFM to the lift distribution solved for using the VLM, the maximum circulation had to be found. To do this, the total circulation calculated by the VLM was substituted into the elliptical circulation distribution function (A.3). Although the actual distribution can be far from elliptical, this operation is acceptable only to allow a value for $\Delta\Gamma$ to be defined that is in the correct range so as to allow for the required number of vortex filaments to be positioned on the wing. If $\Delta\Gamma$ is too small, the integration will result in too little circulation around the wing. If $\Delta\Gamma$ is too large, the simulation will result in an error as it tries to find a position for the requested number of filaments without exceeding the total circulation of the wing. This approximation is acceptable as the positioning of the vortex filaments is adjusted to ensure the correct distribution and total circulation is modelled as long as the value of $\Delta\Gamma$ is of the correct order of magnitude.

Once $\Delta\Gamma$ is defined, the horseshoe vortices of the VLM are integrated from the tip to the root of the wing. As the integration takes place, the value of the circulation is compared to $\Delta\Gamma$. When the integrated value is greater than or equal to the value of $\Delta\Gamma$ multiplied by the number of the filament in question, a linear interpolation across the spanwise length of the last VLM vortex filament integration is performed to give a better approximation of the position of the VFM vortex which will account for the same circulation distribution. The number of horseshoe vortices used in the VLM part of the program is 4 times as many as used in the VFM to ensure that none of the increases in circulation strength due to one of the VLM horseshoe vortices is larger than the increase due to a VFM vortex. This precaution helps the positioning of the vortex filaments in the VFM to be as accurate as possible by minimising the length over which the linear interpolation takes place. As in all simulations, there is a trade off between accuracy and computational speed. The VLM method is computationally cheap to run compared to the VFM, making it important to conserve as much accuracy by increasing the resolution of the VLM section of program. At this point, the positions of the wings as well as the circulation distributions are plotted to check for any

obvious errors in the set up of the simulation, before the computationally expensive VFM is started.

3.5 Filament Management

Filament Management makes use of various sub functions to generate and adjust the vortex filaments as shown in Fig. 3.4, finally producing a converged wake structure for the simulated conditions. The Time Step Generator function moves the wings forward through time and hence their virtual space by a distance defined by the time step length and the wing velocity. It calls the node generator function to shed the nodes, and starting filaments for the first iteration, from the trailing edge of the wing into its wake and define new nodes on the trailing edge in the updated position of the wing. It is appropriate to note that the strength of each vortex filament is constant in order to maintain Helmoltz's first vortex theorem which states that the strength of a vortex does not change along its length. Once this is completed in each time step, the segment manager is called to calculate the induced velocities due to every segment on all filaments of both wings at control points located at the midpoint of each segment of each filament on each wing. The process of calculating the induced velocities is discussed in detail in the vortex induced velocity section. After the induced velocities are calculated each segment is manoeuvred to line up with the direction vector of the induced velocity at its control point, effectively lining up the vortex core with the streamlines of the flow as per Helmholtz's second vortex theorem [20] which states, in part, that a vortex must move with the fluid. However once all the vortices have been adjusted, the velocity field induced by them changes and they are no longer lying along the streamlines of the flow which they are interacting with.

To deal with this discrepancy, the Segment Manager has two options associated with it. The first option is as described above when it is called as part of the time step generator. Once the wing has been propagated through all the required time steps and the wake has been defined, the segment manager is called upon again. In this capacity the function once more calculates the induced velocities in the same way as before, but before adjusting the segments, it compares the orientation of the vortex segments, $\vec{a_i}$, to the direction vectors of the induced velocities at their control points (mid points), $\vec{b_i}$. An error is defined using the root mean square difference (RMSD), shown adapted to this investigation in equation (3.2). If this error is larger than 10^{-5}, the segments are adjusted and the process is repeated until convergence is achieved.

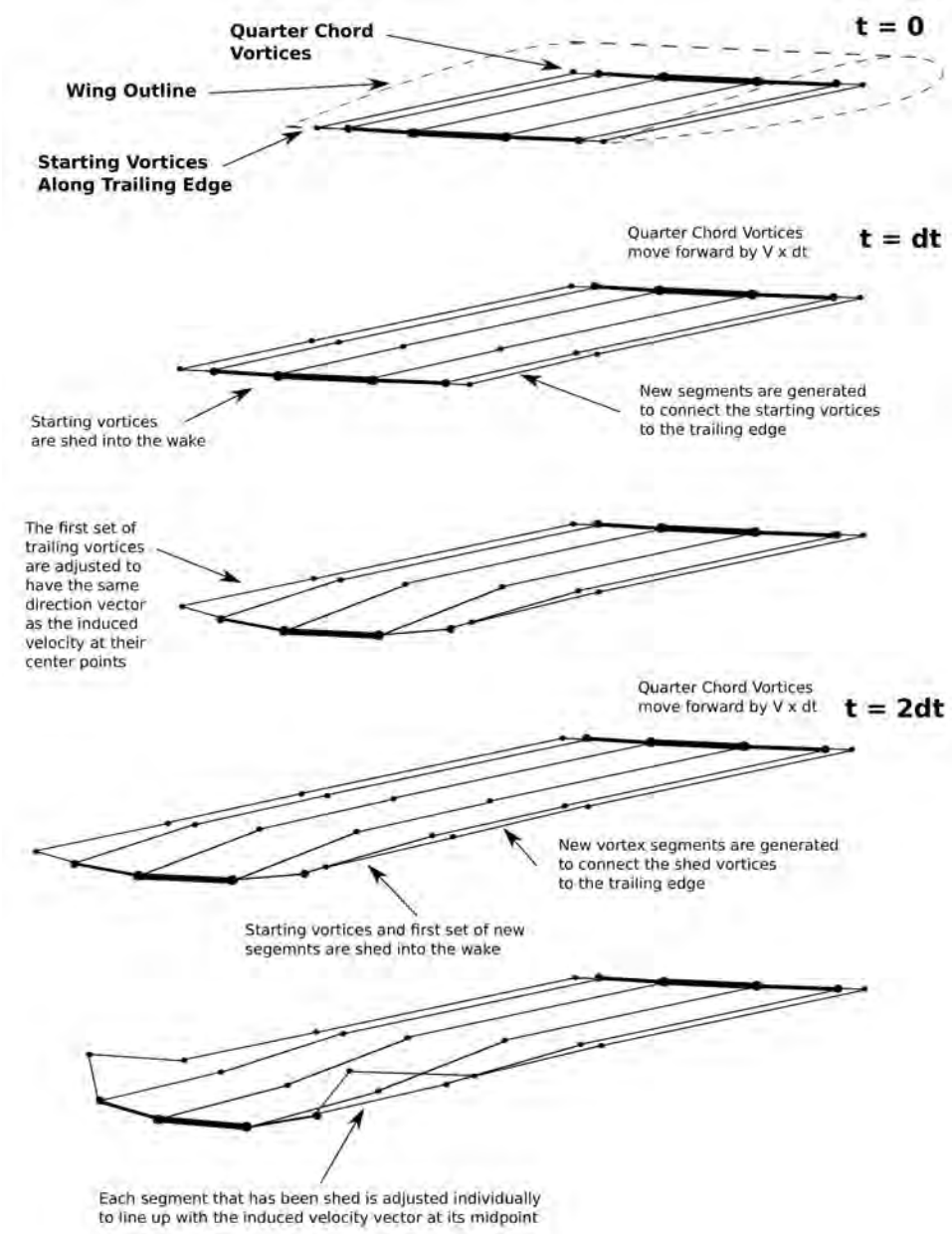


Figure 3.4: Generation and Adjustment of Vortex Filaments using the VFM

$$RMSD = \frac{\sum_{i=1}^{n} |\vec{a_i} \times \vec{b_i}|}{n} \tag{3.2}$$

3.6 Vortex Induced Velocity

The Vortex Induced Velocity function calculates the induced velocity at a point due to a finite vortex filament segment. The approach closely follows the procedure found in Katz and Plotkin [25] with one notable difference which is described in Appendix A.2 of this report. Whereas Katz used the vortex profile defined by the Biot-Savart law, the vortex induced velocity function can model the effects of the Burnham-Hallock vortex profile or the Biot-Savart vortex profile depending on the simulation conditions. Although the model is capable of simulating a Biot-Savart vortex profile, it was found through initial tests that the Biot-Savart model did not converge to a stable solution for the VFM while the Burnham-Hallock profile did. For this reason, it was decided to use the Burnham-Hallock model for the VFM and the Biot-Savart model for the VLM. The core size for the Burnham-Hallock model was set to 5% of the wing span to satisfy the convergence criteria put forward by Ehret while keeping the core as small as possible in order to limit the pseudo-viscous effects.

3.7 Virtual Tuft Grid

The virtual tuft grid was designed to give a picture of the velocity field downstream of the simulation which would be comparable to the experimental results. The vortex induced velocity function is used once again to calculate the induced velocity at a series of points due to every segment of each vortex filament in the simulation. In this case the series of points is defined to be at the nodes of a grid in the y-z plane downstream of the two virtual wings. The ratio of aperture of the grid to wing span is defined to be the same for the virtual tuft grid and the experimental tuft grid, $\frac{b}{30}$. The positions of the various vortex cores can be observed graphically or through a numerical output to make the data capture process easier. To pinpoint the various vortex cores, the program runs through the vertical component of the induced velocities at each node across a horizontal line of the tuft grid. The horizontal line is chosen to have a z value of zero. If the induced velocity at one node has the opposite sign to the induced velocity at the next node, the y ordinate of a vortex core must lie between these two nodes. The horizontal component of the induced velocity of the nodes in a vertical line are then compared. The y position is defined by the y ordinate of the first of the two nodes of each pair identified in the previous step. If the induced velocity changes sign between two nodes, the z ordinate of a vortex core must lie between these nodes. By linear interpolation, as shown only for the y ordinate in equation (3.3), the exact coordinates of each core can be found at each downstream grid location.

$$y = y_{i+1} - (y_{i+1} - y_i)\left(\frac{w_{tg_{i+1}}}{w_{tg_{i+1}}} - w_{tg_i}\right) \tag{3.3}$$

3.8 Wake Velocity Profile

The Wake Velocity Profile graphs only the vertical component of the velocity along a defined line across the wake. This capability is important to allow the model to be verified against experimental data in literature. As described in chapter 2, this is a common expression of the wake of an aircraft when addressing trailing wing-tip vortices, wake encounters and formation flight.

Chapter 4

Wind Tunnel Apparatus and Methods

In order to validate the numerical approach taken in this investigation, an experimental apparatus was designed. The main purpose of the experimental work was to find the positions of the wing-tip trailing vortices relative to the wings in the wind tunnel. The vortex positions need to be measured at different locations downstream of the trailing wing with the wings at different positions relative to one another. The length of wake to be investigated was limited to 3 spans downstream of the trailing aircraft. The close proximity and short length of wake to be investigated was limited by the capabilities of the wind tunnel equipment available at UCT.A rectangular planform was selected over swept and twisted wings for ease of manufacture and simplicity in writing the MatLab code to prove the concept that the vortex filament method can be extended to multiple aircraft for formation flight simulations. A symmetric profile was selected to allow for the results of positive and negative angles of attack to be compared as mirrored images. Any inconsistencies between these mirror images of the vortex positions could be attributed to inaccuracies in the positioning of the apparatus and accounted for in the relevant computer simulations. The NACA 0012 airfoil profile was used due to its prevalence in literature [10, 23, 51, 55] and because its characteristics compare well to thin wing theory at the Reynolds Numbers and angles of attack experienced in this study [56, 57].

4.0.1 Design Considerations

The wind tunnel apparatus was to consist of two NACA 0012 wings mounted in formation, trailed by a tuft grid and camera for flow visualisation, in the closed loop wind tunnel at UCT. The detailed design drawings can be found in Appendix C while the relevant aerodynamic and strength calculations are described in Appendix B alongside the wind tunnel correction calculations. The factors considered in the design and manufacture are:

Minimise Wind Tunnel Effects

The wall boundary constraints of a wind tunnel have various effects on the models and measurements. To minimise these effects, some basic geometrical constraints are generally adhered to. For instance, Pankhurst recommends that the total span of both wings must be less than 70% of the width of the wind tunnel test section [58] while Daugherty [59] recommends the span of the model to be less than 80% of the wind tunnel test section width. The supporting structure and adjustment mechanisms must also be taken into account and should be designed to influence the flow in the wind tunnel as little as practically possible. It is also important to be able to adjust the test variables from outside the tunnel to reduce the time needed for testing, by not having to make the tunnel safe for internal work between each test. The calculations and further explanations of the various effects are detailed in Appendix B.

Adjustable in x, y and z Axis

Formation flight benefits are very sensitive to the relative positioning of the aircraft trailing vortices. The spanwise positioning has the largest effect on the formation flight benefits, with the vertical positioning a close second. The streamwise positioning is mainly important due to the opportunity for the vortices to move around more with larger separation distances. In order to understand the interaction of these vortices it is necessary to study their behaviour in various positions.

Δx Δy Δz: The axis system is defined if Fig. 4.1. The stream-wise separation between the two wings was chosen to be constant at 1 span. This gives ample time for the wake of the lead wing to roll up but still gives the opportunity to measure the positions of the vortex cores at 3 positions in the limited length of the wind tunnel test section. The required change in z direction is $\pm 10\%$ of the span and the required change in y direction is $\pm 30\%$ of the span. Wind tunnel research conducted by Inasawa [27] inserted only half of the lead wing into the wind tunnel and showed the position of maximum vortex interaction at 2.5% overlap and the position of maximum aerodynamic advantage at 5% overlap with zero vertical offset. Flight test data presented by Ray et al. [1] shows the maximum aerodynamic advantage achieved at an overlap of around 15% with a vertical offset of about 10% of the span. Wind Tunnel test results in the selected region are therefore of great interest. Comparisons between wind tunnel results and simulated results will test the computer model under the most challenging parameters possible in steady cruise formation flight neglecting atmospheric turbulence.

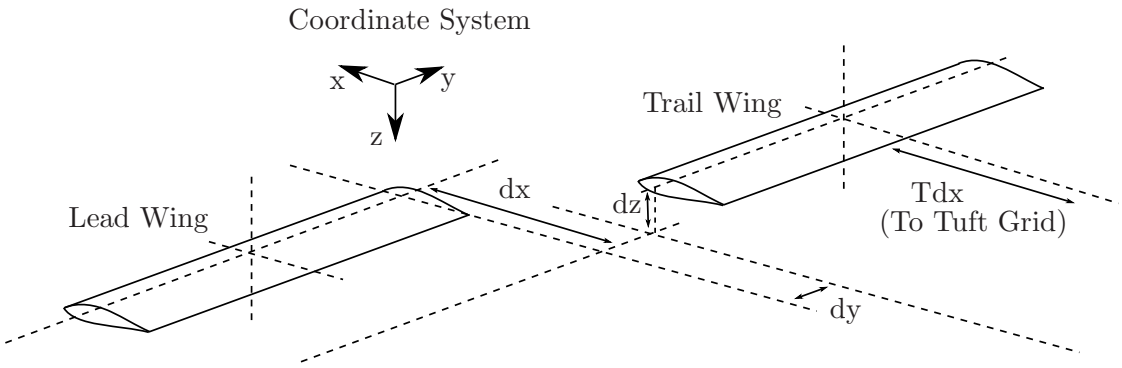


Figure 4.1: Coordinate System

The shape of the closed section of the wind tunnel was found to not be consistent along its length. This made it difficult to move the equipment to adjust one variable without affecting the other variables. This problem was tackled in different ways. New mounting brackets were installed on the top of the wind tunnel and a new Perspex top window was cut and installed on the brackets to allow the adjustments to take place as accurately as possible along the correct axis of the wind tunnel. The adjustment mechanisms and the slots in the Perspex window were designed to minimise the area of removed material in the test section, limiting distortions in the flow field. Post processing and statistical analysis of the data recorded in the wind tunnel was useful to compensate for this systematic error.

Adjustable Angle of Attack

The angle of attack of the wings determines the vortex strength at a constant wind speed. Formation flight effects induced forces on the trailing wing especially and it is important to be able to independently adjust the angles of each wing to take these effects into account in future tests. Although the test parameters are $\pm 8^{\circ}$, designing for the angle of attack of each wing to be adjustable between $\pm 12^{\circ}$ allows some room for calibration after installation, or for more versatility in future tests.

4.1 Flow Visualisation

The tuft grid has been used successfully to show the flow field in the wake of small aircraft [34] as well as in wind tunnel experiments [60]. It gives good qualitative results and can be extended to give quantitative results of the velocity distribution using a series of cameras and bilinear transformations to remove the errors of parallax and lens distortion [22]. The tuft grid is

inexpensive and simple to use although very time consuming and tedious to construct.

The major design considerations are to make sure that the flow in the wind tunnel is affected as little as possible by the introduction of the tuft grid and camera. This means making the frame thin and positioning it close to the wind tunnel walls while using thin and well tensioned chord to create the grid. The aperture of the grid and the ratio of aperture to tuft length were selected first through comparisons to the apparatus described by Bird [60] and then fine tuned through preliminary tests in the closed loop wind tunnel at UCT. Bird used 0.3 mm wire with 25.4 mm aperture and 76.2 mm tuft length in a 1.83 m by 1.83 m test section at 45 ms^{-1} with a wing root chord of 0.3 m and at various degrees of sweep translating into a span of between 140 mm and 914 mm.

With an expected vortex core radius of 3% [31, 40] an aperture of 20 mm was selected with a tuft length of 60 mm. This aperture did not give an accurate enough representation of the vortex core position and it was reduced to 10 mm in areas of the grid through which the vortices were expected to pass. The tuft length was shortened to 30 mm in accordance with the ratio used by Bird. Due to the weaker vortices resulting from the lower lift, the deflection of the short tufts was difficult to see using the camera available and the tuft length was extended back to 60 mm. The final parameters for the tuft grid are an aperture of 10 mm created with 0.3 mm diameter, 20 kg breaking strain braided nylon fishing line. The tuft length was settled at 60 mm.

The braided nylon line was modelled as a series of infinitely long, thin circular cylinders for the purposes of drag calculations. The effects of one line on the neighbouring lines was neglected due to the low ratio of the cross section of the line to the gap between each line [58]. The drag was then calculated for a single horizontal line multiplied by the number of horizontal lines and added to the drag of a single vertical line multiplied by the number of vertical lines. For the lines which are anchored through the slanted corners, the full length of the lines was used as an over estimation. The drag due to the tufts was considered small due to their very small cross sectional area and stream-wise orientation.

During the initial design of the tuft grid, a wooden frame was chosen due to its light weight, low cost and because it could easily be fashioned into a shape with low aerodynamic drag properties. Design calculations showed that a wooden frame would be strong enough but load tests on the wooden prototype showed that although the wood was strong enough, the pinned

and glued joints between the sections of the frame were not. The load tests were carried out by fixing the frame in a bench vice and loading masses onto the other end of the frame to simulate the bending moment that would be experienced in the wind tunnel. Aluminium was then chosen for its low weight, high strength and the fact that it could be welded together which would ensure strong joints. An additional benefit to the aluminium was that it did not flex excessively during the tensioning of the grid which made the process much easier.

An important design characteristic is the spacing of the holes that the grid is strung through. The holes are designed to be larger than the braid for ease of manufacture and assembly. This means that the braid will lie against one side of a hole in the top of the grid and the opposite side of the partner hole in the bottom of the grid. As can be seen in Fig. 4.2, the holes are therefore spaced in a pattern to ensure that the spacing of the braid is consistent and that each strand is parallel.

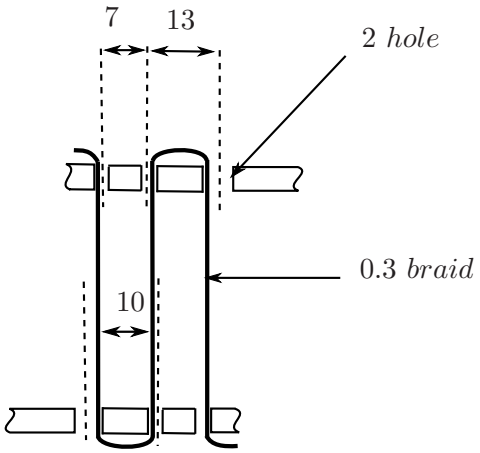


Figure 4.2: Tuft Grid Hole Spacing

4.2 Airfoils and Mountings

The design considerations for the wing can be broken up into two categories, function and manufacturing. In terms of functionality, the wing must be suspended from the quarter chord center point. The angle of attack must be adjusted about the quarter chord line. The mechanism must protrude into the wind tunnel as little as possible so as to not affect the flow in the wind tunnel excessively. The span of the wing was chosen to be 300 mm. The ratio of the span of the two wings to the wind tunnel cross section is therefore 69% which is within the recommendation by Pankhurst [58].

Figure 4.3: Wind Tunnel Apparatus Installed in Test Section

The surface roughness on the wing was kept rather course, although not specifically defined quantitatively, in order to decrease the propensity for flow separation to occur at higher angles of attack.

In terms of manufacture, the wings had to be large enough to allow for the chosen mechanism to be contained inside the wing while being small enough to not be greatly influenced by the walls of the wind tunnel. This resulted in a model with very small internal parts. The final design of the wing is shown in Fig. C.2 of Appendix C. Each wing was milled out of a single block of aluminium using a CNC milling machine. Tabs had to be left on the edges of the wing, as shown in Fig. C.3 of Appendix C so that the material could be clamped into position on the milling machine in such a way as to minimise the deflection of the material during the milling process. The tabs were cut and filed off by hand after the machining process was complete. A picture of the apparatus installed in the wind tunnel test section is shown in Fig. 4.3.

4.2.1 Adjustment Mechanisms

To maintain repeatability of the tests, a discrete adjustment mechanism was used for the angle of attack as well as the vertical position of the wings. The

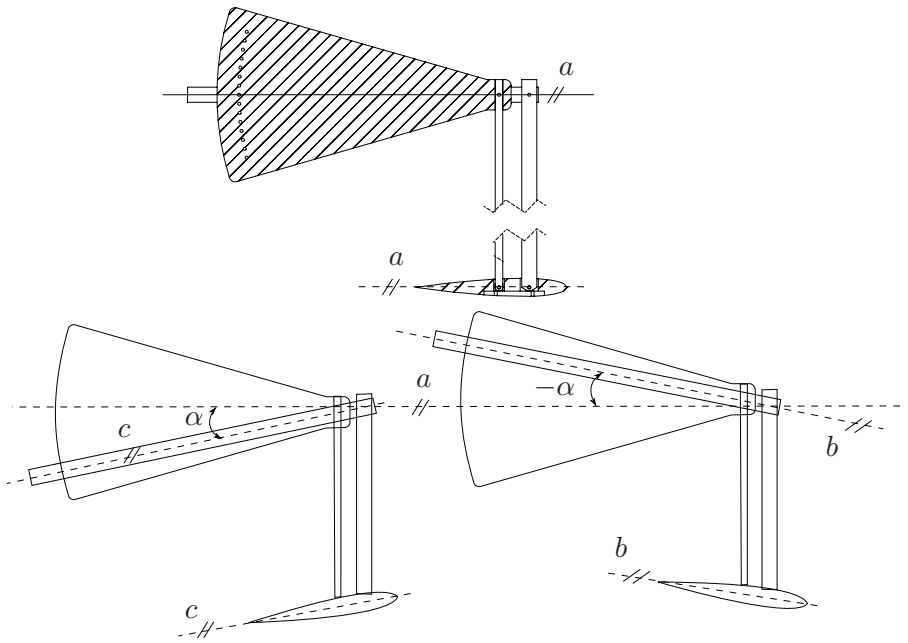


Figure 4.4: Angle of Attack Adjustment Mechanism

Angle of attack was designed to be adjusted in 2° intervals by the mechanism shown in Fig. 4.4. The relative y and z positions of the wings are adjustable in 10 mm increments and the x separation, in span length increments.

In order to allow for all of these variable to be manipulated while keeping the flow in the wind tunnel as consistent as possible, a new perpex top window was designed and installed on aluminium angle iron. The design allows for the lead wing to move in the y direction and the trailing wing and tuft grid to move in the x direction in separate channels. Once the wings were in position for a test, the holes were taped over as much as practical to maintain the flow consistency in the wind tunnel.

4.3 Methods

The simulations performed in MatLab produce results that are, by the nature of the analytical model, consistent given a set of input parameters and exactly mirrored between positive and negative angles of attack. This fact enables us to perform half the number if tests as were performed in the wind tunnel experiments.

Inasawa shows that the most aerodynamically advantageous position for a trailing aircraft is an overlap of 5% and largest interaction takes place at a 2.5% overlap [27]. The tests were therefore performed around this region

in order to test the computer model in the most extreme cases of vortex interaction. The positions tested were at y values corresponding to wingtip overlap in percentage of wing span of -10%, 0%, 10% and 38%. At each y position, apart from 38% overlap, the test was performed at z positions corresponding to wingtip separations of 10%, 0% and -10%. The tests were repeated for positive and negative angles of attack in each position to check for inconsistency and possible errors in the set up of the apparatus or the testing procedure. Assuming perfect accuracy of set up and measurement and consistency of conditions, the tests would be mirrored in the exact same way as we see in the MatLab simulation.

4.3.1 Apparatus Installation

When the wings were installed in the wind tunnel, tests were performed to ensure that they were mounted as accurately as possible relative to the wind tunnel walls. The vertical position of each wing-tip was measured to ensure that their roll angle was zero. The wings were lined up to ensure zero yaw angle visually through the top viewing panel of the wind tunnel. This was done by comparing the leading edge of the wing to a cross bar which was measured at each end to ensure that it was parallel to opening of the test section. The zero angle of attack was confirmed by running the wind tunnel with the tuft grid installed behind one wing at a time. The angle of attack of the wing was then adjusted until their was no sign of a trailing vortex.

4.3.2 Data Analysis

Video of the tuft grid in the wind tunnel was viewed in VLC Media player and still pictures captured using the *take snapshot* function. Three pictures were captured for each angle of attack under each relative positioning condition and at each downstream position of the tuft grid. The pictures were then opened in G3 Data Graph Analyser. The axis orientation was defined manually for each image as follows. The positions of the wing tips relative to the tuft grid were identified by looking at an image of the tuft grid positioned on the trailing edge of the wing at zero angle of attack. This process made the trailing wing the datum for the measurements with the wing-tip positions of the lead wing extrapolated from the experimental set-up. These known points remained constant on the tuft grid. They were re-identified in each image and the axis orientation and scale fixed to them. Once this position was found, the data points were selected by observation. The core was difficult to define exactly, as can be seen in Fig. 5.2, and the center of the distortion was selected as the vortex core position. The coordinates were then exported to a text file and imported into LibreOffice Calc. The data for each angle of attack was put into graphs with the numerical data in order to compare the vortex positions. The uncertainty of the selection

of the core position was shown through the use of error bars representing half a square of the tuft grid in each direction. The post processing of the data which was undertaken to investigate the sources of experimental errors is described through Chapter 5.

Chapter 5

Results and Discussion

Testing consisted of multiple stages each with a specific scope. Table 5.1 shows the simulation input variables for each stage of testing. For all of the wind tunnel tests and their relevant numerical simulations, the wings were held at 8° angle of attack. This angle was chosen in order to maximise the vortex strength while limiting the potential for flow separation on the trailing wing due to the induced angle of attack. Flow separation will begin after $C_{L_{max}}$ occurs at 10 ° [61]. Therefore, setting the airfoils at 8° will allow for the 2° induced angle of attack calculated in Appendix B.

Table 5.1: Input Conditions for Specific Simulations [62]

	Single Wing 747 Cruise	Single Wing Wind Tunnel	Two Wing Wind Tunnel
Air Density	0.35 kgm^{-3}	1.2 kgm^{-3}	1.2 kgm^{-3}
Core Diameter	5% b	5% b	5% b
Span (b)	60 m	0.3 m	0.3 m
Mean Aerodynamic Chord (c)	8 m	0.048 m	0.048 m
Velocity (u)	250 ms^{-1}	23 ms^{-1}	23 ms^{-1}
Maximum Circulation (Γ_{max})	700 m^2s^{-1}	N/A	N/A
Number of Fillaments	100	40	40
time step (Δt)	0.06 s	0.0032 s	0.0032 s
Simulation Length (t)	2.16 s	0.1216 s	0.1216 s
Aircraft Weight	396890 kg	N/A	N/A

5.1 Single Wing Validation

The first set of tests were run in order to verify and validate the VLM and VFM for a single wing against literature and wind tunnel experiments. The numerical model was run for a single wing wind tunnel simulation with input

variables set as described in table 5.1.

5.1.1 Lift Comparison

Lift coefficient results from the numerical model were compared with the expected lift coefficient worked out from classical aerodynamic theory and experimental results from literature. This was done using the single wing wind tunnel conditions and the Non-Symmetric lift distribution. The VLM predicts a C_L of 0.59 for a single thin wing at 8° angle of attack under small disturbance assumptions at a wind speed of $23ms^{-1}$. Increasing the resolution of the model did not have any significant effect on the lift coefficient. Thin wing theory predicts a $C_{l\alpha}$ of 2π for an infinitely long wing [7, 25, 53] which results in a C_l of 0.88 at an angle of attack of 8°. The value of $C_{L\alpha}$ for a wing is dependant on the actual profile shape as well as aspect ratio and planform shape.

Thickness Profile Effect Experimental data presented by Abbott [61] reduces the C_l to 0.80 for a NACA0012 wing section spanning a test section, effectively taking into account the thickness of the airfoil.

Aspect Ratio Effect Kats and Plotkin use a lifting line model for a finite wing to present the relationship between $C_{L\alpha}$ and aspect ratio shown in equation (5.1). This results in a $C_{L\alpha}$ of 4.7 for an aspect ratio of 6 with an elliptical lift distribution. The C_L on a single thin wing, under small disturbance assumptions inherent in the definition of the lift coefficient shown in equation B.11, is therefore calculated to be 0.66 for an elliptical lift distribution.

Planform Shape Effect The actual lift distribution on a rectangular planform wing is not elliptical. This can be accounted for using the Oswald efficiency factor calculated in appendix B to be, $e = 0.858$ [61] [7]. From Appendix B we can see that the viscous drag $C_{dr} = 0.009$ and the total drag is $C_D = 0.067$. Therefore the induced drag accounts for 87% of the total drag. As the Oswald efficiency factor affects the induced component only, increasing it by 16.5%. Since C_{Di} is proportional to C_L^2 as shown in equation (5.2) [63], it can be said that the Oswald efficiency factor decreases the lift coefficient by $1 - \sqrt{0.858}\ \% = 7.4\%$. This reduction brings the calculated C_L to 0.61, an acceptable 3% higher than the VLM used in this report.

$$C_{L_\alpha} = \frac{2\pi}{1 + \frac{2}{AR}} \tag{5.1}$$

$$C_{Di} = \frac{C_L^2}{e\pi AR} \tag{5.2}$$

Ehret presents the case of a single elliptically loaded wing modelled by the Vortex Filament method which was validated against flight test data for a Boeing 747 aircraft. In order to compare Ehret's model to the model described in this dissertation, the variable's had to be adjusted for the specific case that Ehret described. These parameters are shown in table 5.1 [52]. It was found that the VFM did not converge to the criteria required in this dissertation using the Biot-Savart vortex profile without a core. The tests were then run using the Burnham-Hallock profile with a core radius set to 5% of span as per the recommendations of Ehret who uses a low-order algebraic core within the Biot-Savart profile. The test in this case took 7 days to converge. A much faster test with a lower resolution taking only 24 hours to converge shows only a small difference in wake velocity profile. Table 5.2 compares these results to those of Ehret and to the 747 wake data [51] which Ehret used to validate the approach and shows that there is excellent agreement between the 747 data and the results of this investigation. The core sizes are reported in percentage span and are found in all cases through measuring the diameter between the turning points of the wake velocity profile plot through the centres of the vortex cores, 750 m downstream of the wing.

Table 5.2: Validation of Vortex Filament Method

	VFM	Ehret VFM	747 Wake
Peak Upwash	12 ms^{-1}	15 ms^{-1}	12 ms^{-1}
Vortex Core Size	14% b	±14% b	14% b

5.1.2 Experimental vs Numerical: Single Wing

The next step in validating the computer model is to compare the vortex paths of a single wing predicted by the VFM with those measured experimentally. As can be seen in Fig. 5.1 a region of vorticity is present along the trailing edge of the wing and the vorticity is stronger towards the tips of the wings as is expected. The video of this test, available upon request, shows the tufts in the section indicated buffeted by the vortex sheet in the early stages of roll up. Figure 5.2 shows that the vortex has formed into what can be approximated by a single vortex structure within 1 span length downstream of the wing (1Tdx). Figure 5.3 shows an image of the virtual tuft grid from which the vortex positions are extracted.

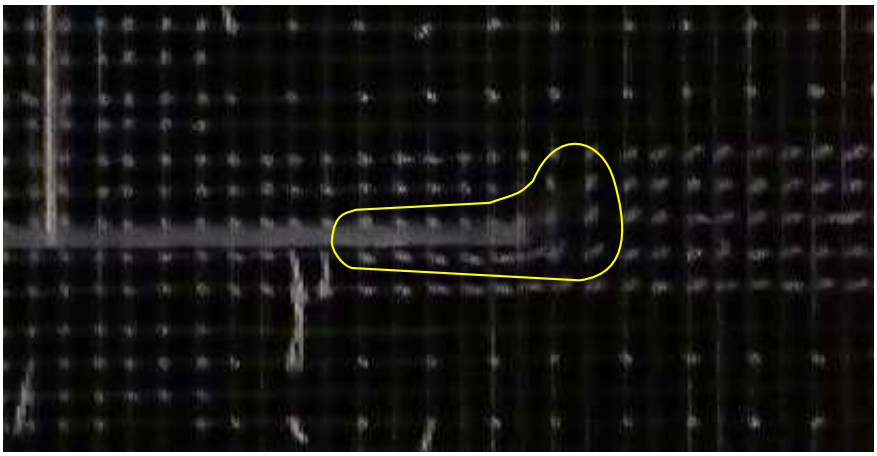


Figure 5.1: Tuft Grid Positioned Immediately Downstream of Trailing Edge of a Single Wing, Showing Vorticity Along the Trailing Edge of the Wing

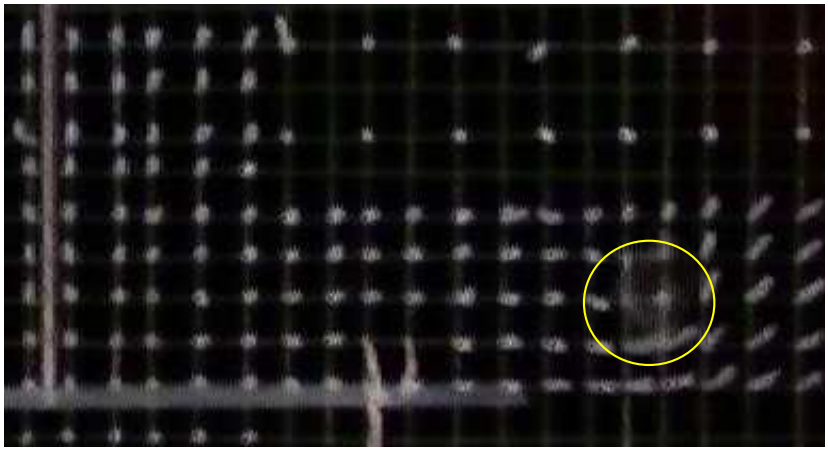


Figure 5.2: Tuft Grid Positioned 1 Span Downstream of a Single Wing, Showing Rolled Up Wingtip Vortex

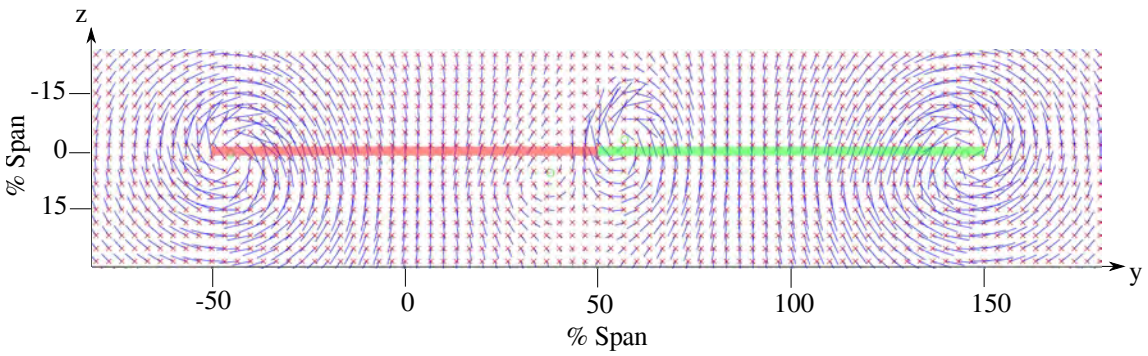


Figure 5.3: Virtual Tuft Grid showing simulated vortex positions 1 span downstream of trailing wing

Figures 5.4, 5.5 and 5.6 show the comparison between the VFM and corresponding wind tunnel results. The difference between the two sets of data is within the measurement accuracy of the tuft grid, however, an error analysis was performed. Some bias errors were found in the wind tunnel test section, which are discussed in detail in section 5.3, and in the positioning of the wing trailing edge in the MatLab simulation. The MatLab code did not model the gradient of the wing and hence placed the trailing edge at the same vertical position as the quarter chord line resulting in a consistent error of 1.7% of the span in the vertical position of the vortices. To compensate for the wind tunnel bias errors, the raw results were averaged and centred. The bias error from the numerical simulation is smaller than the positional accuracy of the wind tunnel apparatus and is accounted for in the 3% b error bars. Figures 5.7, 5.8 and 5.9 show the corrected results which indicate that the numerical vortices are generally slightly inboard of the experimental vortices. The error bars shown in the figures are a representation of 1 square of the tuft grid. It is interesting to note that as the resolution of the numerical model was increased, the simulated vortices rolled up closer to the wing-tip and hence closer to the experimentally measured vortex positions. Part of the reason for this could be attributed to the small losses in the numerical integration used in the VLM and VFM. The results show that the vortices move inwards and slightly downwards with time. At three spans downstream of the wing, the vortices have not yet settled into a consistent span-wise position and are still moving closer together. This makes it difficult to compare the measurements with literature which usually describes the rolled up separation distance and descent rate between the vortices. [9, 18, 33]

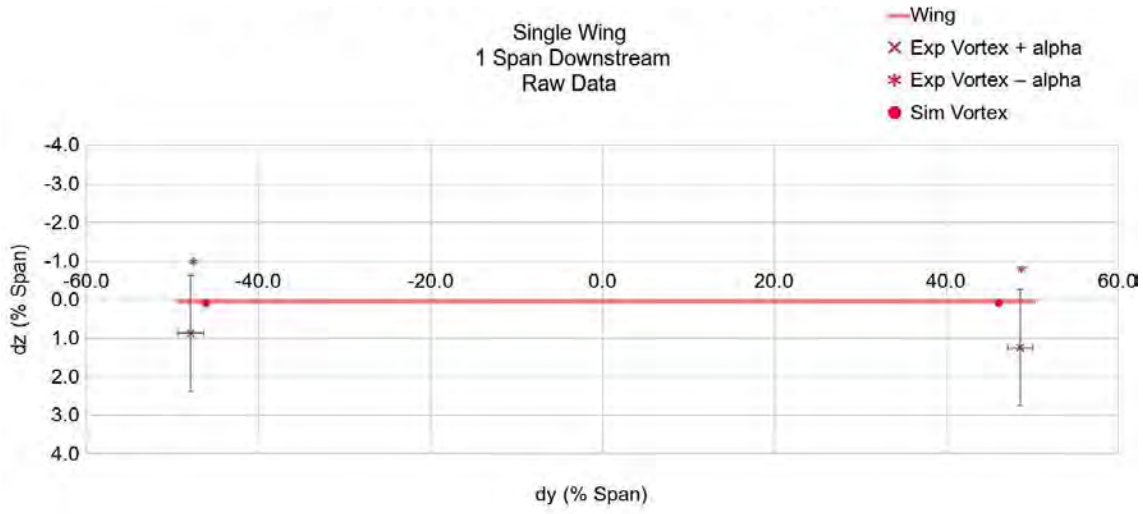


Figure 5.4: Simulated Vortex Position vs Experimental Vortex Position (Tuft Grid 1 Span Downstream of Trailing Wing, Raw Data)

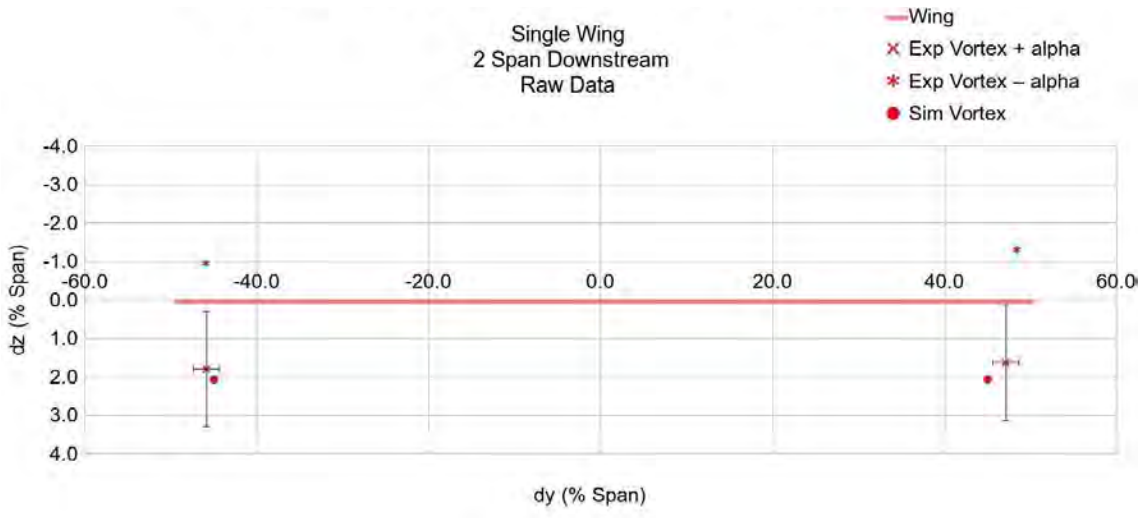


Figure 5.5: Simulated Vortex Position vs Experimental Vortex Position (Tuft Grid 2 Spans Downstream of Trailing Wing, Raw Data)

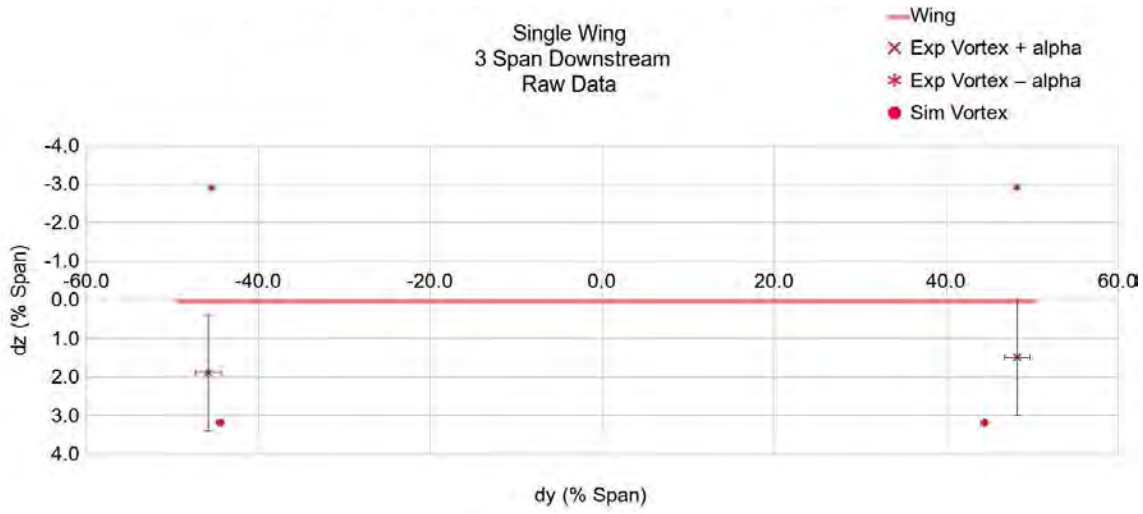


Figure 5.6: Simulated Vortex Position vs Experimental Vortex Position (Tuft Grid 3 Spans Downstream of Trailing Wing, Raw Data)

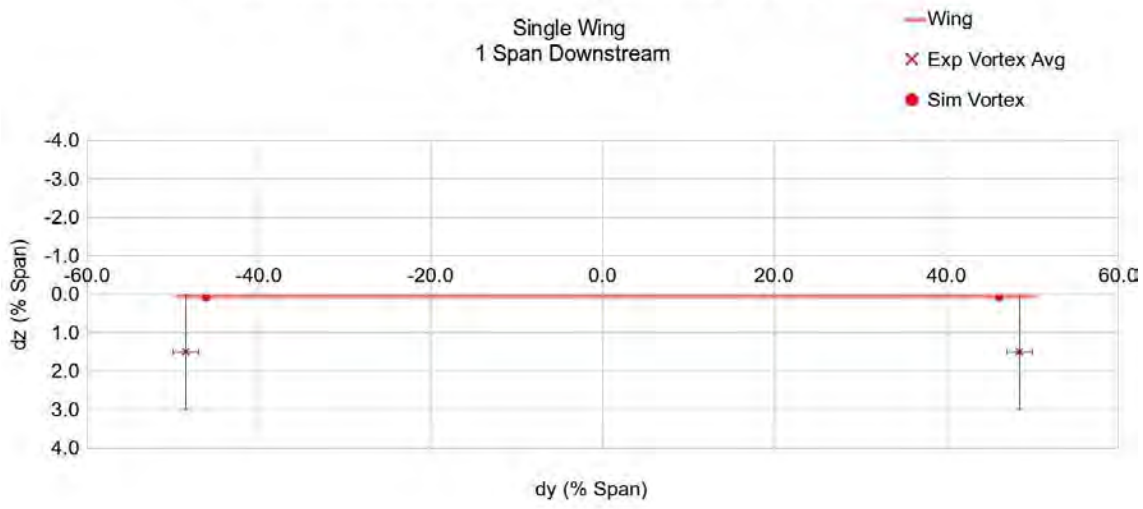


Figure 5.7: Simulated Vortex Position vs Experimental Vortex Position (Tuft Grid 1 Span Downstream of Trailing Wing)

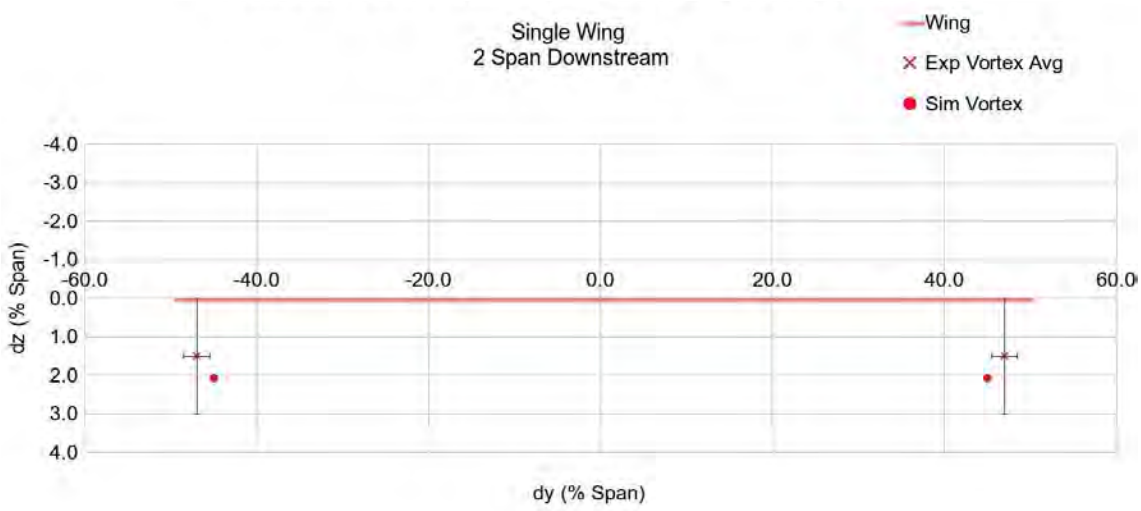


Figure 5.8: Simulated Vortex Position vs Experimental Vortex Position (Tuft Grid 2 Spans Downstream of Trailing Wing)

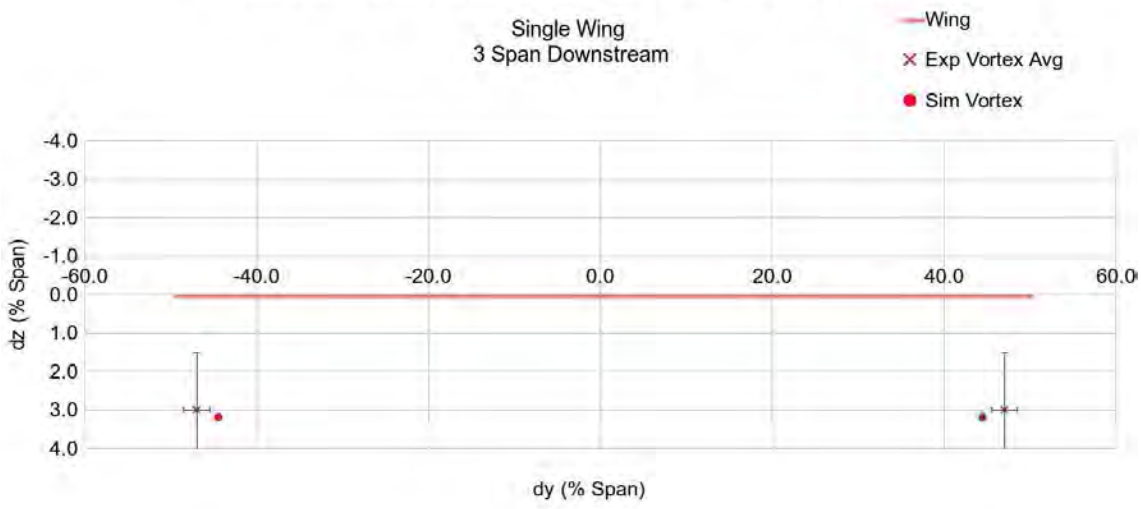


Figure 5.9: Simulated Vortex Position vs Experimental Vortex Position (Tuft Grid 3 Spans Downstream of Trailing Wing)

5.2 Two Wings in Formation

The model is extended to include a second wing to investigate the wing-tip vortex interaction and paths in formation flight. The parameters for the test are held the same as described in Table 5.1 for wind tunnel conditions and the coordinate system is defined in Fig. 4.1. The tests were run with a constant stream-wise separation of 1 b for span-wise wing-tip overlaps of, 38% b, 10% b, 0% b and -10% b. Vertical separations were set at -3.3% b, 0% b and 3% b (negative vertical separation meaning that the trailing wing is above the lead wing as per sign convention) for each span-wise position except 38% wing-tip overlap which was only conducted at 0 vertical separation. The tuft grid was placed at 3 different positions (1 chord length, 1 span length and 2 span lengths) downstream of the trailing wing for each formation configuration. It was found that with wing-tip overlap, increasing the resolution of the models resulted in increased predicted lift. The lift distribution found through the VLM for the wing-tip overlap condition showed variable results for the overlapping section of the trailing wing. The circulation over this section was often reversed by the interaction with the flat wake of the lead aircraft. Applying the VFM smoothed out theses inconsistencies in the lift distribution, but still resulted in a reduced increase in lift seen by trailing wing. This is understandable and well documented in literature as the trailing wing moves into the downwash region of the wake generated by the lead wing.

Increasing the resolution of the VLM did not have any effect on the lift predicted with zero overlap or a separation between the wingtips. With a 10% wing-tip overlap however, the increased resolution in the VFM narrowed the gap between the predicted lift of the two methods. As mentioned before, increasing the resolution of the VFM has dramatic effects on the run time. To keep the simulations to a manageable computational cost, the VFM was restricted to 40 filaments per wing.

Table 5.3 compares the predicted lift for both wings from different techniques. The VLM and VFM methods are those developed for this study which are described above. The third case uses, for the lead wing, the result produced by the small disturbance assumption for an elliptically loaded wing that is described in Section 5.1.1. For the trailing wing, the results presented by Bizinos [31] shown in equation B.16 and B.18 are used to calculate the induced lift coefficient on the trailing wing which takes into account the formation flight effects.

Table 5.3: Comparison between Lift Predicted by VFM Model, VLM Model and Thin Wing Theory + Bizinos Model with both wings at $\alpha = 8°$

		Lift Coefficient		
Condition	Method	Wing 1	Wing 2	% Increase
dy = 0; dz = 0	VLM	0.599	0.663	10.6%
	VFM	0.578	0.641	10.9%
	TWT + Bizinos	0.610	0.645	5.7 %
dy = 0; dz = 0.033	VLM	0.595	0.652	9.6%
	VFM	0.575	0.632	9.9%
dy = -0.1; dz = 0	VLM	0.595	0.573	-3.7%
	VFM	0.575	0.606	5.4%
	TWT + Bizinos	0.610	0.663	8.7%
dy = 0.1; dz = 0	VLM	0.595	0.630	5.9%
	VFM	0.575	0.610	6.1%
	TWT + Bizinos	0.610	0.636	4.3 %

Increasing the resolution of the VLM only for wingtip to wingtip had no effect on the VLM predicted Lift for either wing or on the VFM lift for either wing. The discrepancies highlighted in this table between the single horseshoe model presented by Bizinos and the multiple horseshoe model with and without the VFM roll up show the importance of the vortex spacing rather than the aircraft spacing. Depending on the vortex spacing of the wake model used, very different increases in lift can be found.

5.2.1 Vortex Paths

To validate the predicted paths of the wingtip trailing vortices in formation flight, the results of the measured vortex positions in the wind tunnel tests were compared to the predicted vortex positions using the VFM and VLM numerical model. The test conditions are shown in table 5.1. The data analysis took place in the same way as for the single wing comparisons without the averaging and adjustment of the vortex positions and included one extra step. The data for two wings was analysed using basic statistics. In this analysis, the data from all of the downstream locations of the tuft grid are displayed together for a specific trailing vortex. An error was defined between the experimental and numerical results separately in the Y and Z Cartesian axis. The spread sheet containing the complete data set and analysis can be found on the accompanying CD-rom. Figure 5.10 shows the error between the span-wise ordinates of the two data sets for all of the vortices. This Y error was defined to be positive if the vector from the numerical vortex position to the experimental vortex position is positive. The distribution of errors has a mean value of 2.3% b and a standard deviation

of 8.9% b. The mean error is less than the diameter of a vortex core which is a promising result. The large standard deviation is unfortunately less promising. The expected precision of the apparatus should give a standard deviation of 1.5% b or less. The mean value of the error shows that the numerical simulation tends to predict the vortex positions to be slightly to the port side of the experimental vortex positions in general. A shift like this is more likely due to an experimental bias error as inaccuracies in the model would be cancelled out by the symmetry. For instance if the model predicted the vortices to roll up closer together or further apart, the errors would cancel out between the port and starboard vortices in this specific analysis.

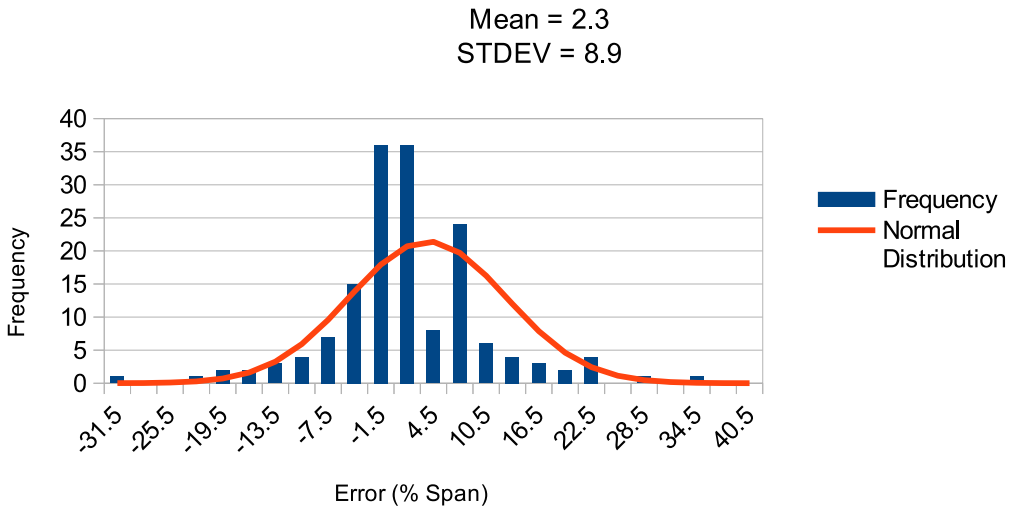


Figure 5.10: Frequency Distribution of the Error Between the Y Ordinates of the Numerically Predicted and Experimentally Measured Vortex Positions

Figure 5.11 shows the error between the Z ordinates of the two data sets. The error in this case was defined to be positive if the vector from the numerical vortex position to the experimental vortex position was positive for positive angles of attack and negative for negative angles of attack. This approach was chosen in order to be able to compare the data from the positive and negative angles of attack at the same time. This is possible because the results should be mirror images of each other in an ideal experimental set-up. Discrepancies between the results of positive and negative angles of attack provide insight into the sources of experimental errors. The error distribution for this case has a mean of -2.1% b and a standard deviation of 5.1% b. From this information it can be hypothesised that the numerical simulation predicts the vortex positions to descend faster than the experimental vortices. Descend must be defined here to mean a movement toward the ground when an aircraft is flying under conventional cruise conditions. While the mean error in both the y and z directions are not unacceptably large, the length of wake being investigate is relatively short and the large standard deviations are of some concern.

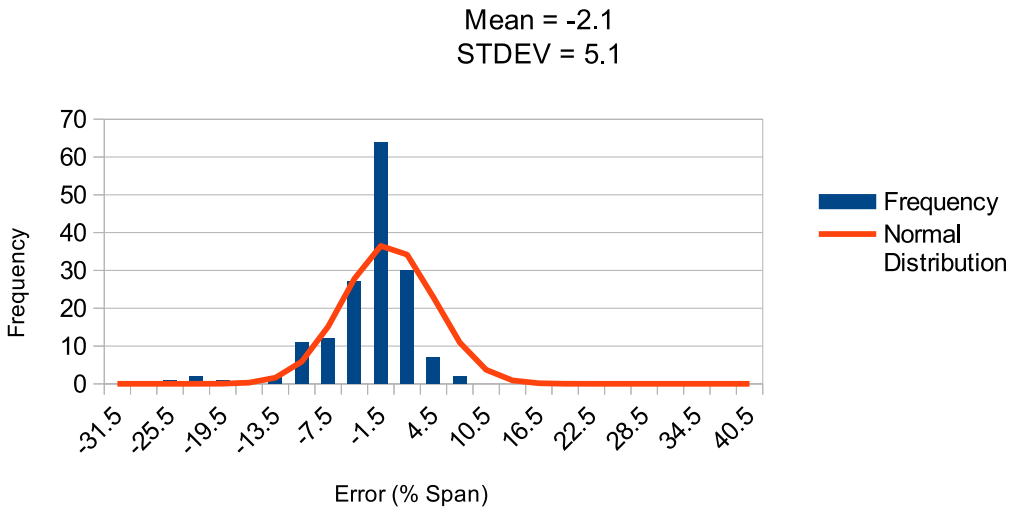


Figure 5.11: Frequency Distribution of the Error Between the Z Ordinates of the Numerically Predicted and Experimentally Measured Vortex Positions

Y Error Breakdown

To investigate when the simulated vortices roll up closer together or further apart than the experimental vortices, the results are analysed further. A closer look at the data sets shows that certain errors can be attributed to specific vortices to shed more light on validity of the numerical data. The mean error for the port vortex of the lead wing is 3.1% b as displayed in Fig. 5.12 and shows that the experimental vortex is found slightly inboard of the numerically predicted position. A standard deviation of 4.4% b for this vortex is slightly more encouraging. The mean error for the trailing wing starboard vortex is found to be 0.6% b with a standard deviation of 2.2% b as shown in Fig. 5.13. By the Y error definition, this mean error shows that the experimental vortex is very slightly outboard of the numerical vortex. If the bias error is taken as an experimental misalignment and the vortices inboard of the formation are excluded, these results can be interpreted to show that the numerical vortices roll up slightly inboard of the experimental vortices. This can be deduced because the numerical lead port vortex is more inboard of the corresponding experimental vortex than the trailing starboard numerical vortex is outboard of its experimental counterpart.

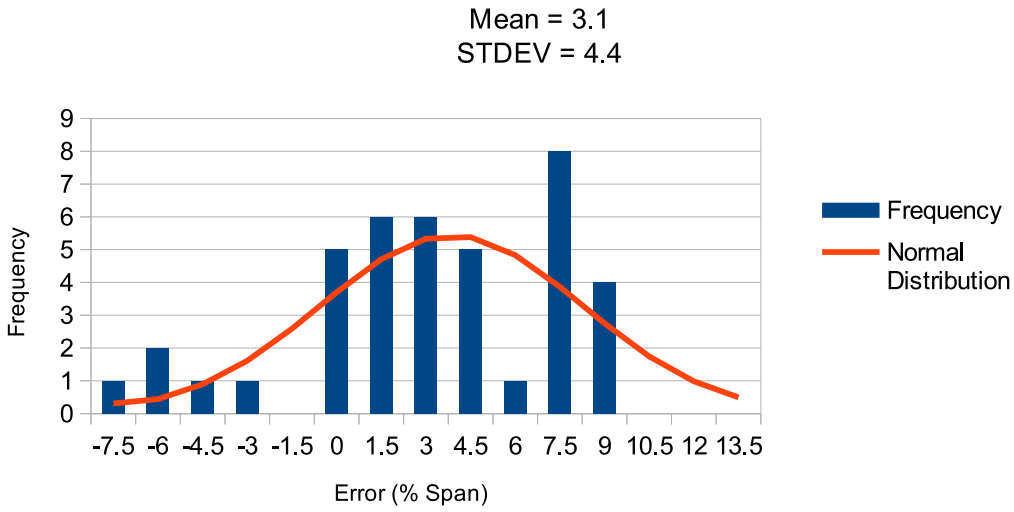


Figure 5.12: Frequency Distribution of the Error Between the Y Ordinates of the Numerically Predicted and Experimentally Measured Lead Wing Port Vortex Positions

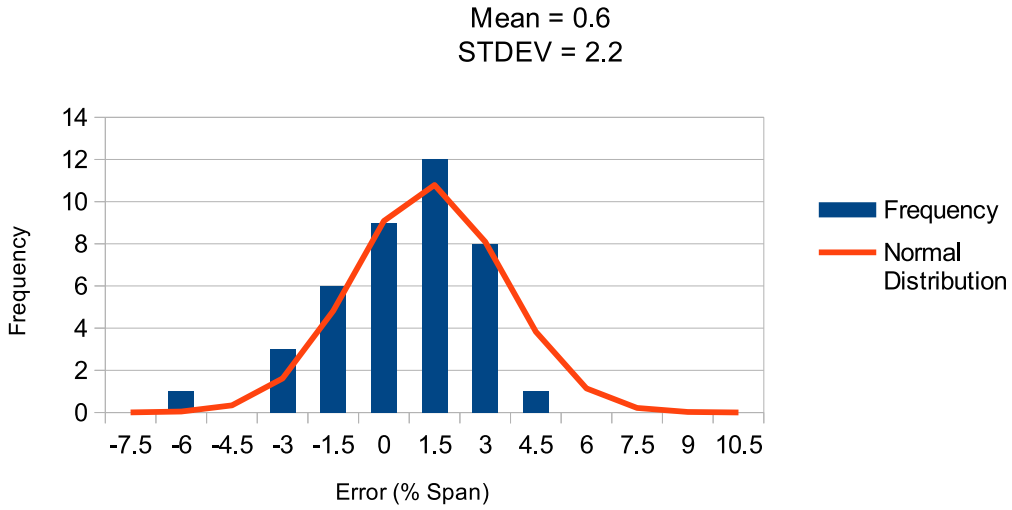


Figure 5.13: Frequency Distribution of the Error Between the Y Ordinates of the Numerically Predicted and Experimentally Measured Trail Wing Starboard Vortex Positions

The errors between the vortices associated with the wing-tips inboard of the formation are much larger and have a much greater standard deviation. Figures 5.14 and 5.15 indicate a mean error of 8.7% b for the lead wings starboard vortex and and -3.0% b for the trailing wings port vortex. The standard deviations are also much larger at 12.8% b and 7.7% b respectively. The general trend of the data still shows that the experimental vortices are offset slightly to the formation port side of numerical solution. This offset is much larger than the offset of the formation outboard vortices which means that they cannot both result from the same experimental bias. The difference between the errors shows that the numerical vortices are much further apart than the experimental vortices. The discussion in 5.2.2 sheds more light on possible sources of the large errors associated with the formation inboard vortices.

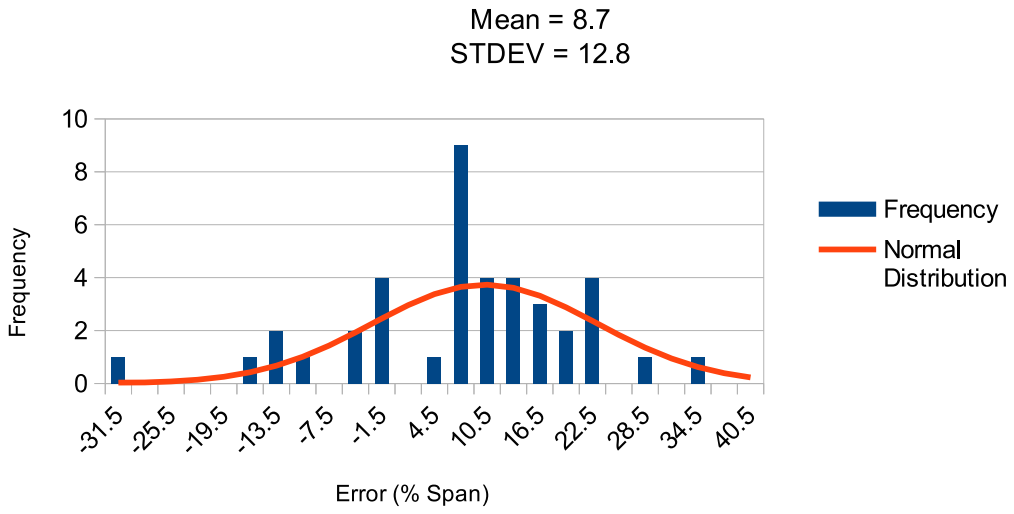


Figure 5.14: Frequency Distribution of the Error Between the Y Ordinates of the Numerically Predicted and Experimentally Measured Lead Wing Starboard Vortex Positions

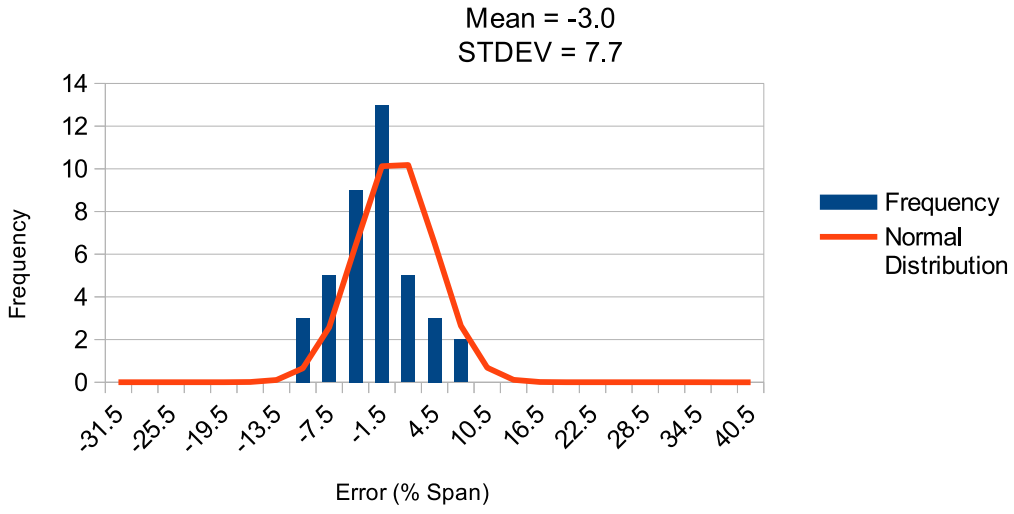


Figure 5.15: Frequency Distribution of the Error Between the Y Ordinates of the Numerically Predicted and Experimentally Measured Trail Wing Port Vortex Positions

Z Error Breakdown

The Z component of the errors between the numerical and experimental vortex positions is very small with an acceptable standard deviation for the vortices outboard of the formation as shown in Fig. 5.16 and 5.17. Including the inboard vortices again shows larger errors with greater standard deviations as shown in Fig. 5.18 and 5.19. Overall, the formation outboard vortices are well predicted. The numerical formation inboard vortices seem to interact with a larger separation distance and do not move around as much as their experimental counterparts. Although graphs of all of the results are too numerous to include, Appendix D presents the comparison between the numerical and experimental vortex positions for the various zero span-wise wing-tip overlap experiments. These results show graphically that the experimental vortices move around more and interact much more closely than the numerical vortices in the formation inboard region.

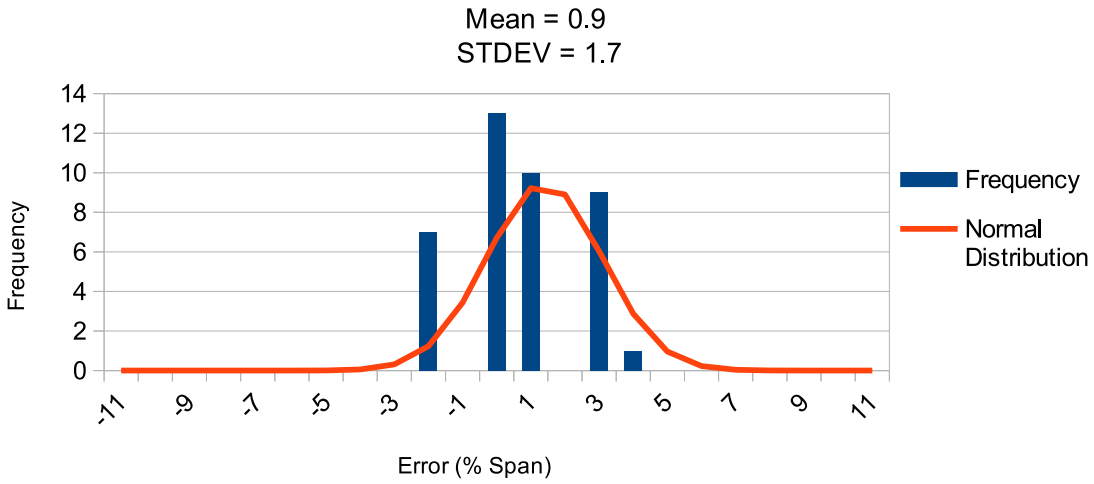


Figure 5.16: Frequency Distribution of the Error Between the Z Ordinates of the Numerically Predicted and Experimentally Measured Outboard Vortex Positions at Positive Angle of Attack

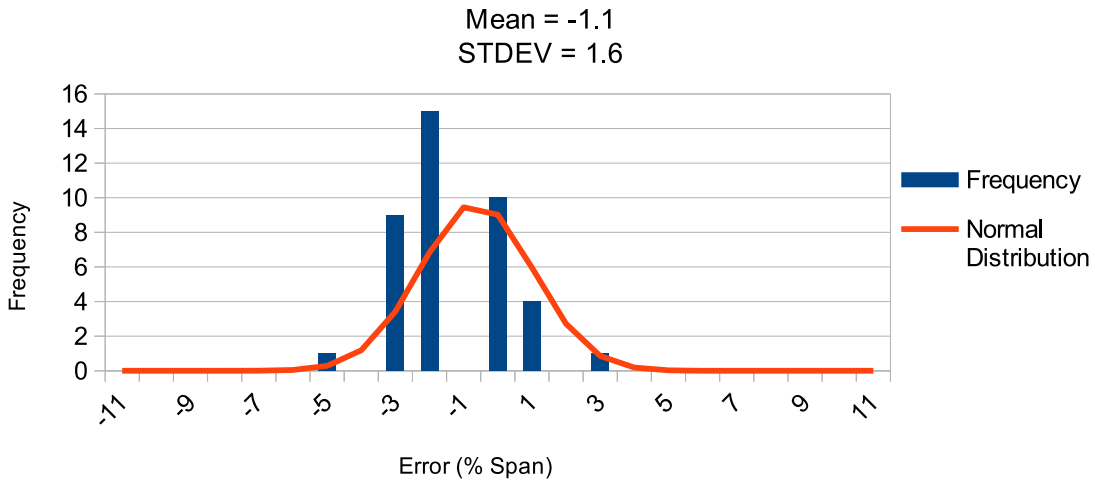


Figure 5.17: Frequency Distribution of the Error Between the Z Ordinates of the Numerically Predicted and Experimentally Measured Outboard Vortex Positions at Negative Angle of Attack

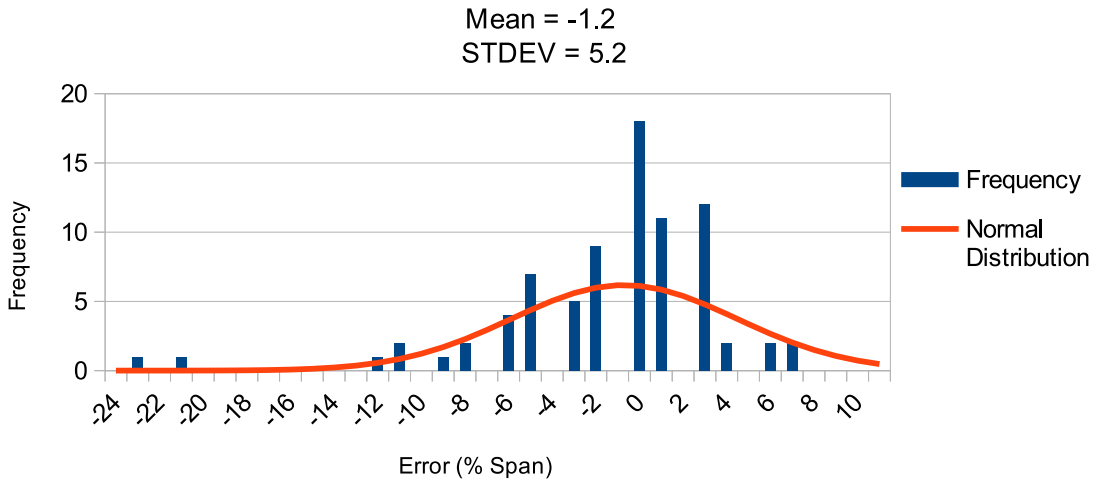


Figure 5.18: Frequency Distribution of the Error Between the Z Ordinates of the Numerically Predicted and Experimentally Measured Vortex Positions at Positive Angle of Attack

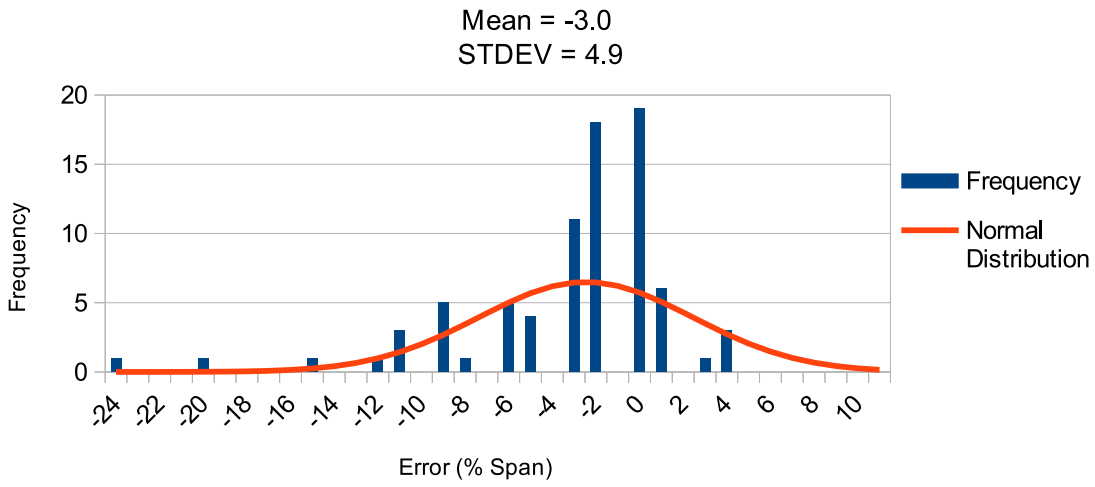


Figure 5.19: Frequency Distribution of the Error Between the Z Ordinates of the Numerically Predicted and Experimentally Measured Vortex Positions at Negative Angle of Attack

5.2.2 Vortex Core Size and Merging

The core size predicted by the numerical model starts off rather large at about 15% b 1 span downstream of the trailing wing. At this point the vortex filaments have not had time to completely roll around one another. The core size reduces dramatically to about 5% b at 3 spans downstream of the trailing wing once the filaments have had a chance to orbit around each other completely. this is contrary to experimental results which show a fully rolled up vortex core less than 1 b downstream of the wing. The complete cancellation of the formation inboard vortices was not achieved in the numerical simulation although it was found experimentally at 0 vertical separation and 0 span-wise wing-tip overlap, 2 spans downstream of the trailing wing. This cancellation can be seen in Fig. D.3 by the absence of the experimental data points.

At a 10% wing-tip overlap, there was significant change in the wake structure, shown in Fig. 5.20, as the formation inboard vortices interacted as shown in Fig. 5.21. The result is a more dispersed region of turbulence than defined vortex cores which relates well to results of vortex interaction studies in literature. The numerical model does not have a mechanism by which the vortex filaments can merge as they come into the range of each others pseudo-viscous cores. Efforts to reduced the core size of each filament in the numerical simulation were not successful and resulted in the numerical instability described by Ehret [33]. Reducing the core size or implementing

the Biot-Savart vortex profile will result in larger induced velocities close to the filaments. This larger induced velocity may speed up the numerical roll-up prediction as well as allow vortices which are in close proximity to one another to interact and move around faster and closer together. Both of these results would show better correlation to the wind tunnel experiments conducted in this study. Allowing a merger of the filaments that are very close together, in a similar fashion to Leonards approach [50], may remove the numerical instability that has so far been tackled by introducing a filament core. Further investigation is needed in order to determine if there is merit in these ideas.

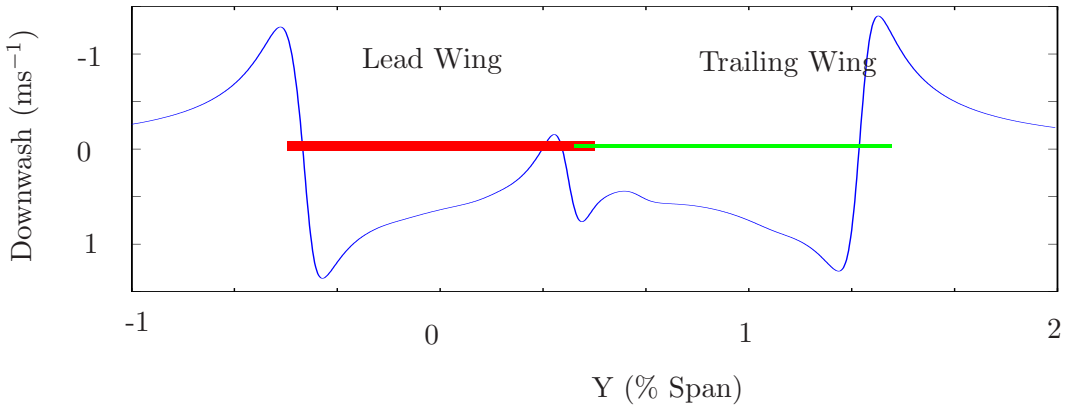


Figure 5.20: Wake Velocity Profile for 10% Span-Wise Overlap, 0 Vertical Separation, 2 Spans Downstream

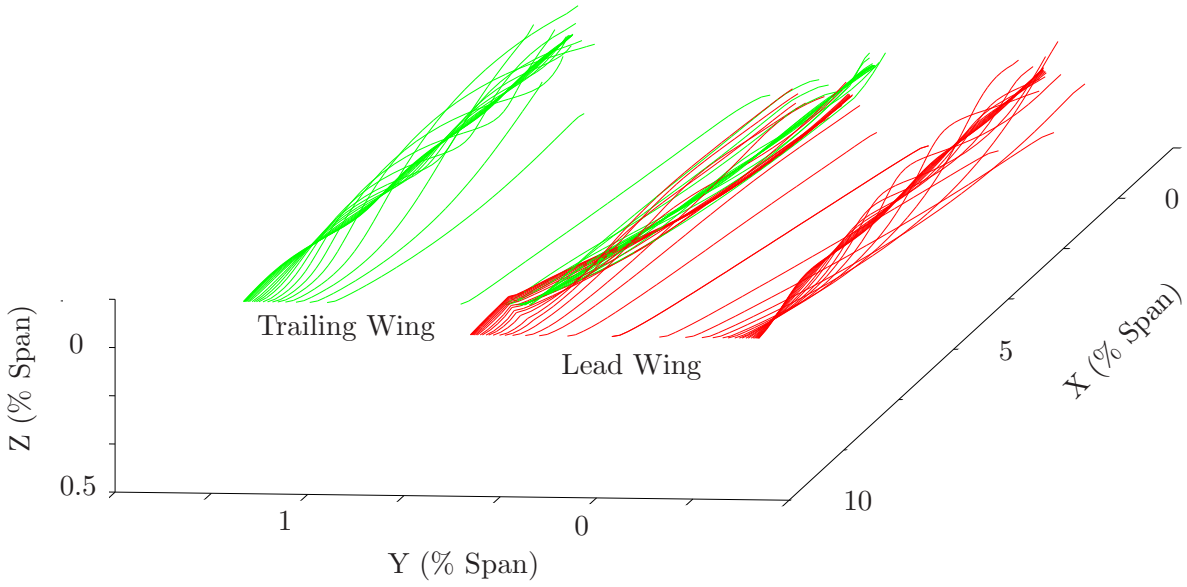


Figure 5.21: Filaments of Numerical Model Lined Up with Stream Lines for 10% Span-Wise Overlap, 0 Vertical Separation

5.3 Experimental Uncertainty

The wind tunnel results introduce uncertainty due to the following:

Vortex Wandering Wind tunnel research has found that the precise core size and position is difficult to measure due to vortex wandering: A phenomenon whereby the vortex oscillates diagonally about a mean position with an amplitude of about that of the predicted core diameter and a large wavelength. Due to the large wavelength, this phenomenon will not have a large effect on the predicted vortex positions [22]. The aperture of the tuft grid is 3% of the span of the wing. As the measurements are recoded by observation of video footage and still pictures, the position of the core cannot be reliably recorded with a resolution better than 1.5% of the span of the wing. The introduction of the camera only 1 span downstream of the tuft grid also adds to the uncertainty of the measurements of the vortex positions.

Wind Tunnel Precision For a single wing, theory predicts that the two trailing vortices should roll up at the same vertical position and their spanwise position should be mirrored about the center. This was not the case in

the experimental results and the discrepancy was up to 2% b of the span for both the vertical and span-wise positions. Figure 5.6 shows the wind tunnel results for one condition. Upon investigating the cause of the error between the numerical and experimental data, a bias error in the apparatus was discovered. As the tuft grid is moved backwards through the wind tunnel, it rotates slightly due to a change in shape of the test section. The amount by which it changes was measured and is shown in Fig. 5.22 and 5.23. Bias errors such as this and other apparatus misalignments can be compensated for to some extent. Averaging the recorded positions of the vortices between the positive and negative angles of attack can theoretically suppress the error. It is important to note that when this post processing takes place for wings with vertical separations, care must be taken to ensure that the correct results are averaged. For example, if the angle of attack of both wings is changed from negative to positive, and the trailing wing is slightly above the lead wing for the negative angle of attack, it must be moved to be slightly below the lead wing for positive angle of attack. This will ensure theoretical symmetry in the results. Figure 5.9 shows the processed data which gives a much better visual representation of the accuracy of the numerical model.

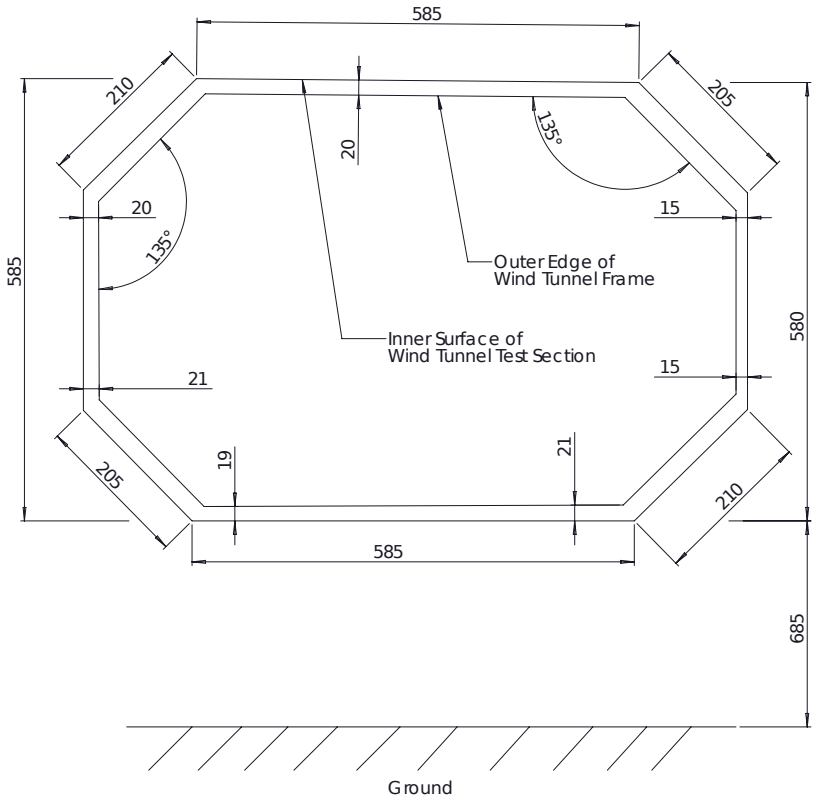


Figure 5.22: Sketch showing the position of the tuft grid relative to the wind tunnel walls at Tdx = 0

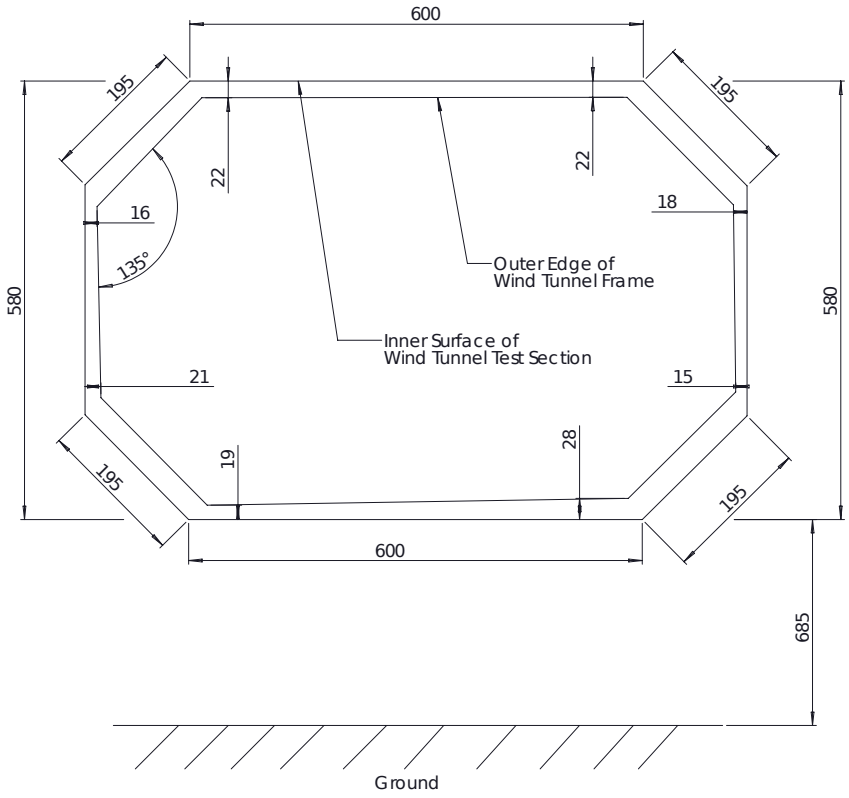


Figure 5.23: Sketch showing the position of the tuft grid relative to the wind tunnel walls at Tdx = 3

Numerical Uncertainty The positions of the vortices predicted by the numerical model move closer towards the wing tip as the number of filaments modelled for each wing increases. A simulation of 40 filaments per wing which creates 40 horseshoe vortices, a good correlation was found between the simulated and experimental results for one wing. The simulation takes around 24 hours to run to achieve this accuracy and resolution. This computational expense is approaching what could be expected for some simple CFD models. Literature on formation flight always describes the formation positions in terms of aircraft separation which is logical, but the benefits of formation flight are sensitive to the relative position of the trailing aircraft to the vortex generated by the lead aircraft. The rolled up position of the vortex wake of the lead aircraft is dependent on the strength of the vortices, which is proportional to lift and dependent on aspect ratio. The flat wake models tend to predict the trailing vortices of a single wing to be closer together than what happens in practise.

Chapter 6

Conclusions

The paths of mutually interacting wing-tip vortices in formation flight were modelled using a combined vortex lattice and vortex filament numerical model. The results were compared to literature and to flow visualisation wind tunnel tests conducted in the low speed closed loop wind tunnel at UCT. The combined numerical model was necessary in order to account for the lift distribution on the trailing wing in the wind tunnel experiments. The flow visualisation was performed by a tuft grid mounted downstream of the wings in the wind tunnel. The wings were set at a constant stream-wise separation of 1 span, while the vertical and horizontal separations were adjusted in discrete iterations around the area or greatest vortex interaction. The formation outboard vortices do not move around much from their initial y-z coordinates. The formation inboard vortices, however, interact strongly with each other and are displaced by up to 10% of the span of the wing in the y-z plane. It is important to note that the paths of the vortices are stable and repeatable. The complete cancellation of the formation inboard vortices was found experimentally to occur at 0 vertical separation and 0 span-wise wing-tip overlap, 2 spans downstream of the trailing wing.

The numerical simulation showed excellent agreement with the single wing model, described by Ehret, on which it was based. The lift calculations for the numerical methods could only be compared to theoretical values as the wind tunnel apparatus did not include lift and drag instrumentation. There were some discrepancies in the values which highlighted the dangers associated with the flat wake models in wing-tip overlap conditions. The VLM gave unrealistic, jagged lift distributions with wing-tip overlap. Increasing the VLM resolution helped the predicted lift distribution to an extent and the application of the VFM smoothed the lift distribution into a more expected profile. Increasing the resolution of the VLM did not have any effect on the lift predicted with zero overlap or a separation between the wingtips. With a 10% wing-tip overlap however, the increased resolution in the VFM

narrowed the gap between the predicted lift of the two methods. Increasing the resolution of the VFM has dramatic effects on the run time. To keep the simulations to a manageable computational cost, the VFM was restricted to 40 filaments per wing for the majority of the numerical simulations. Increasing the resolution of the VLM only, for the positions of zero overlap or any separation between the wing-tips, had no effect on the lift predicted for either wing by the VLM or VFM. Overall, the combined model performed well through these tests although more verification should take place with wind tunnel lift tests.

The numerical vortices were predicted to descend slightly faster than, and roll-up slightly inboard of, the experimental vortices in this near field study. These two observations are strongly related as the closer together a pair of vortices are, the faster they will descend. As the resolution of the model was increased, the vortex positions were predicted to be closer to the experimentally observed vortex positions, however, computational cost and time constraints limited the resolution of the numerical simulations. An error analysis was conducted on the raw data to determine the precision and accuracy of the results. The mean error in the y and z directions between the positions of the numerical vortices and the experimental vortices was 2.3% and -2.1% of the wing span, respectively. These values are only slightly larger than the expected diameter of the vortex core and are acceptable. The standard deviation of the error was, however, of some concern. Upon further analysis, it was found that the formation inboard vortices contributed the most to the mean error and standard deviation of the results. The y errors for the starboard vortex of the lead wing, for instance, have a mean error of 8.7% of the span of the wing and a standard deviation of 12.8% of the wing span. The mean error and standard deviation for the port vortex of the lead wing are dramatically less at 3.1% and 4.4% of the span of the wing respectively.

The error analysis also found that the numerical vortices are predicted to be slightly to the port side of the experimental vortices. Part of this error was attributed a bias error found in the tuft grid mounting. The large error and standard deviation associated with the formation inboard vortices is hypothesised to be due to the pseudo-viscous effects introduced by the core of the Burnham-Hallock vortex profile. The core was introduced to eliminate numerical instabilities in the model by lowering the induced velocity close to the filaments. This slowed down the interaction of the filaments as can be seen in the change in rolled-up core size from 1 span downstream to 3 spans downstream of a single wing. At 1 span the core size is found to be 15% of the wing span and at 3 spans downstream, it is a more appropriate 5% of the wing span. Overall, the formation outboard vortices

are well predicted, however, the numerical formation inboard vortices seem to interact with a larger separation distance and do not move around as much as their experimental counterparts.

Chapter 7

Recommendations

In order to validate the numerical model completely, a more advanced wind tunnel should be used. Ideally a longer test section of up to 20 spans with non-invasive measuring techniques such as Particle Image Velocimetry should be utilised. Incorporating lift and drag measurements into the wind tunnel apparatus would be advantageous to fine tune the numerical model.

Reducing the core size or implementing the Biot-Savart vortex profile for the VFM may be useful by increasing the interaction intensity of filaments in close proximity. A model such as this could better represent the speed of inviscid roll-up and vortex interaction. Allowing a merger of the filaments that are very close together, in a similar fashion to Leonards approach [50], may remove the numerical instability that has so far been tackled by introducing a filament core. Merging filaments that are close together, only after the intricacies of the roll up process have been captured, could maintain the fidelity of the model while greatly reducing computational expense. For instance, once the filaments have rolled up completely, the velocity profile that is produced could be applied to a single filament to represent each rolled-up core.

Use of a more efficient programming language and parallel processing will also increase the speed of the simulation dramatically. The method by which the vortices are adjusted could possibly be improved by drawing on the logic of numerical methods such as the bisection method or Newtons method to search for the settled position of the filaments more quickly. One approach could be to predict and over estimate the movement of the filaments, before adjusting them back towards their settled position. By applying these methods and using a more intricate vortex lattice method, it is the opinion of the author that a fast, high fidelity aircraft wake model can be developed for the study of formation flight.

References

[1] Ronald J. Ray, Brent R. Cobleigh, M Jake Vachon, and Clinton St. John. Flight Test Techniques Used to Evaluate Performance Benefits During Formation Flight. In *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, number August, Monterey, 2002. NASA; 1998, American Institute of Aeronautics and Astronautics.

[2] M Jake Vachon, Ronald J Ray, Kevin R Walsh, and Kimberly Ennix. F / A-18 Performance Benefits Measured During the Autonomous Formation Flight Project. In *AIAA Atmospheric Flight Mechanics Conference*, number August, Monterey, 2002. American Institute of Aeronautics and Astronautics.

[3] S Andrew Ning, Tristan C Flanzer, and Ilan M Kroo. Aerodynamic Performance of Extended Formation Flight. *Journal of Aircraft*, 48(3):855–865, 2011.

[4] Jia Xu, S. Andrew Ning, Geoffrey Bower, and Ilan Kroo. Aircraft Route Optimization for Formation Flight. *Journal of Aircraft*, 51(2):490–501, March 2014.

[5] Joe Pahle, Dave Berger, Mike Venti, Chris Duggan, Jim Faber, and Kyle Cardinal. An Initial Flight Investigation of Formation Flight for Drag Reduction on the C-17 Aircrat. In *AIAA Atmospheric Flight Mechanics Conference*, Minneapolis, 2012. American Institute of Aeronautics and Astronautics.

[6] Ilan Kroo. Drag Due to Lift: Concepts for Prediction and Reduction. *Annual Review of Fluid Mechanics*, 33:587–617, 2001.

[7] J D Anderson. *Fundamentals of Aerodynamics*, volume 1984 of *McGraw-Hill series in aeronautical and aerospace engineering*. McGraw-Hill Education, New, fifth edition, 2011.

[8] Coleman duP. Donaldson and Alan J. Bilanin. Vortex Wakes of Conventional Aircraft. Technical report, Advisory Group for Aerospace Research and Development, Princeton, May 1975.

[9] Philippe R. Spalart. Airplane Trailing Vortices. *Annual Review of Fluid Mechanics*, 30:107–138, January 1998.

[10] Vernon J. Rossow. Lift-Generated Vortex Wakes of Subsonic Transport Aircraft. *Progress in Aerospace Sciences*, 35(6):507–660, August 1999.

[11] Thomas Gerz, Frank Holzapfel, and Denis Darracq. Commercial aircraft wake vortices. *Progress in Aerospace Sciences*, 38:181–208, 2002.

[12] R K Takahashi, K W Mcalister, and Technical Memorandum. Preliminary study of a wing-tip vortex using laser velocimetry. *Nasa Technical Memorandum*, 88343(January), 1987.

[13] A Shekarriz, T C Fu, J Katz, and T T Huang. Near-field behavior of a tip vortex. *AIAA journal*, 31(1):112–118, 1993.

[14] Omer A. Alsayed, Waqar Asrar, and Ashraf a. Omar. Evaluation of Aircraft Wing-Tip Vortex Using PIV. In M. A. Wahid, J. M. Sheriff, N. A. C. Sidik, and S. Samion, editors, *The 10th Asian International Conference on Fluid Machinery*, pages 382–390, Kuala Lumpur, 2010. American Institude of Physics.

[15] Michael Czech, Gregory Miller, Jeffrey Crouch, and Michail Strelets. Predicting the near-field evolution of airplane trailing vortices. *Comptes Rendus Physique*, 6:451–466, May 2005.

[16] J. M. Ortega, R. L. Bristol, and Ö. Savas. Experimental Study of the Instability of Unequal-Strength Counter-Rotating Vortex Pairs. *Journal of Fluid Mechanics*, 474(January 2003):35–84, January 2003.

[17] A. L. Chen, J. D. Jacob, and Ö. Sava. Dynamics of corotating vortex pairs in the wakes of flapped airfoils. *Journal of Fluid Mechanics*, 382(March 1999):155–193, March 1999.

[18] Liu Chao. *Wake Vortex Encounter Analysis with Different Wake Vortex Models Using Vortex-Lattice Method*. PhD thesis, Delft University of Technology, 2007.

[19] R. S. Scorer. Natural aerodynamics. 1958.

[20] P. G. Saffman. *Vortex Dynamics*. Cambridge University Press, 1995.

[21] S. C. Crow. Stability Theory for a Pair of Trailing Vortices. *AIAA Journal*, 8(12):2172 – 2179, 1970.

[22] C. del Pino, J. M. Lopez-Alonso, L. Parras, and R. Fernandez-Feria. Dynamics of the Wing-Tip Vortex in the Near Field of a NACA 0012 Aerofoil. *The Aeronautical Journal*, 115(1164), 2011.

[23] M. L. Beninati and J. S. Marshall. An experimental study of the effect of free-stream turbulence on a trailing vortex. *Experiments in Fluids*, 38:244–257, 2005.

[24] William J. Devenport, Michael C. Rife, Stergios I. Liapis, Gordon J. Follin, J S Zsoldos, and C M Vogel. The structure and development of a wing-tip vortex. *Journal of Fluid Mechanics*, 312(1):67, April 1996.

[25] Joseph Katz and Allen Plotkin. *Low-Speed Aerodynamics*, volume 13. Cambridge University Press, New York, second edition, 2001.

[26] J U Klar, C Breitsamter, S Hickel, and N Adams. Integrated Experimental-Numerical Analysis of High-Agility Aircraft Wake Vortex Evolution. *Journal of aircraft*, 48(6):2050–2058, 2011.

[27] Ayumu Inasawa, Fumihide Mori, and Masahito Asai. Detailed Observations of Interactions of Wingtip Vortices in Close-Formation Flight. *Journal of Aircraft*, 49(1):206–213, 2012.

[28] Ravindra V Jategaonkar, Dietrich Fischenberg, and Wolfgang Gruenhagen. Aerodynamic Modeling and System Identification from Flight Data-Recent Applications at DLR. *Journal of Aircraft*, 41(4):681–691, July 2004.

[29] David A Hinton and Chris R Tatnall. A Candidate Wake Vortex Strength Definition for Application to the NASA Aircraft Vortex Spacing System (AVOSS). Technical Report September, NASA, Hampton, 1997.

[30] David C. Burnham and James N. Hallock. Measurements of Wake Vortices Interacting with the Ground. *Journal of Aircraft*, 42(5):1179–1187, September 2005.

[31] Nicholas Bizinos and Christiaan Redelinghuys. Tentative Study of Passenger Comfort During Formation Flight Within Atmospheric Turbulence. *Journal of Aircraft*, 50(3):886–900, May 2013.

[32] Dietrich Fischenberg. A Method to Validate Wake Vortex Encounter Models From Flight Test Data. In *27th International Congress of the Aeronautical Sciences*, Nice, 2010. International Congress of the Aeronautical Sciences.

[33] T. Ehret and H. Oertel Jr. Calculation of Wake Vortex Structures in the Near-Field Wake Behind Cruising Aircraft. *Atmospheric Environment*, 32(18):3089–3095, 1998.

[34] B W McCormick, H E Sherrier, and J L Tangler. Structure of trailing vortices(Aircraft trailing vortex structure studies involving flight and

model testing predict vortex geometry and velocity field downstream). *Journal of Aircraft*, 5:260–267, 1968.

[35] Alexander Allen and Christian Breitsamter. Experimental Investigation of Counter-Rotating Four Vortex Aircraft Wake. *Aerospace Science and Technology*, 13(2-3):114–129, March 2009.

[36] J. D. Crouch. Instability and transient growth for two trailing-vortex pairs. *Journal of Fluid Mechanics*, 350(November 1997):311–330, November 1997.

[37] Man Jia, Yuan Gao, Fei Huang, Sen-yue Lou, Ji-lin Sun, and Xiao-yan Tang. Vortices and vortex sources of multiple vortex interaction systems. *Nonlinear Analysis: Real World Applications*, 13(5):2079–2095, October 2012.

[38] W B Blake and D R Gingras. Comparison of predicted and measured formation flight interference effects. Technical report, DTIC Document, 2001.

[39] A W Bloy, M JoumaA, K A Lea, M G West, and M Jouma'A. Lateral aerodynamic interference between tanker and receiver in air-to-air refueling. *Journal of Aircraft*, 30(5):705–710, 1993.

[40] William Blake and Dieter Multhopp. Design, Performance and Modeling Considerations for Close Formation Flight. In *23rd Atmospheric Flight Mechanics Conference*, number AIAA-98-4343, pages 476–486, Boston, 1998. American Institute of Aeronautics and Astronautics.

[41] Michael T Morgan. *A Study in Drag Reduction of Close Formation Flight*. PhD thesis, United States Airforce Institute of Technology, 2005.

[42] Maziar S. Hemati, Jeff D. Eldredge, and Jason L. Speyer. Wake Sensing for Aircraft Formation Flight. *Journal of Guidance, Control, and Dynamics*, 37(2):513–524, March 2014.

[43] Vernon J. Rossow. Validation of Vortex-Lattice Method for Loads on Wings in Lift-Generated Wakes. *Journal of Aircraft*, 32(6):1254–1262, 1995.

[44] S Andrew Ning and Ilan Kroo. Compressibility Effects of Extended Formation Flight. In *29th AIAA Applied Aerodynamics Conference*, pages 1–24, 2011.

[45] David Fabre, Laurent Jacquin, and Antoine Loof. Optimal perturbations in a four-vortex aircraft wake in counter-rotating configuration. *Journal of Fluid Mechanics*, 451(January 2002):319 – 328, January 2002.

[46] D. Margerit, P. Brancher, and A. Giovannini. Implementation and Validation of a Slender Vortex Filament Code: Its Application to the Study of a Four-Vortex Wake Model. *International Journal for Numerical Methods in Fluids*, 44(August 2003):175–196, 2004.

[47] C. Eugene Wayne. Vortices and Motion. *Notices of the AMS*, 58(1), 2011.

[48] S. A. Dovgii and A. V. Shekhovtsov. An Improved Vortex Lattice Method for Nonstationary Problems. *Journal of Mathematical Sciences*, 104(6):1615–1627, 2001.

[49] D. C. Burnham, J. N. Hallock, I. H. Tombach, M. R. Brashears, and M. R. Barber. Ground-Based Measurements of the Wake Vortex Characteristics of a B-747 Aircraft in Various Configurations. Technical report, Transport Systems Center, Cambridge, December 1978.

[50] A Leonard. Computing three-dimensional incompressible flows with vortex elements. *Annual Review of Fluid Mechanics*, 17(1):523–559, 1985.

[51] J. N. Hallock, D. C. Burnham, I. H. Tombach, M. R. Brashears, A. D. Zalay, and M. R. Barber. Ground-based measurements of the wake vortex characteristics of a B747 aircraft in various configurations. January 1978.

[52] Technical Characteristics – Boeing 747-400.

[53] Holt Ashley. *Engineering Analysis of Flight Vehicles*. Addison-Wesley Publishing Company, 2012.

[54] Rachel M. King and Ashok Gopalarathnam. No Title. *AIAA Journal*, 42(5):1188, 2005.

[55] Ömer Sava. Experimental Investigations on Wake Vortices and their Alleviation. *Comptes Rendus Physique*, 6:415–429, May 2005.

[56] Robert T. Jones. Correction of the Lifting-Line Theory for the Effect of the Chord. Technical report, NACA, Washington, 1941.

[57] H Julian Allen. General Theory of Airfoil Sections Having Arbitrary Shape or Pressure Distribution. Technical report, NACA, Moffett Field, 1945.

[58] R. C. Pankhurst and D. W. Holder. *Wind Tunnel Technique*. Sir Isaac Pitman & Sons, Ltd, London, 1952.

[59] J .C Daugherty. Unitary Plan Wind Tunnel Blockage Recommendations. Technical report, NASA Ames, Moffett Field, 1984.

[60] John D. Bird. Tuft-Grid Surveys at Low Speeds for Delta Wings. Technical Report February 1969, NASA Langley Research Center, Hampton, 1969.

[61] Ira H. Abbott and Albert E. Von Doenhoff. *Theory of Wing Sections Including a Summary of Airfoil Data*. General Publishing Company, Toronto, 2 edition, 1959.

[62] Todd Jones, John W Rustenburg, Donald A Skinn, Daniel O Tipps, and Thomas DeFiore. Statistical Data for the Boeing-747-400 Aircraft in Commercial Operations. Technical Report January, U.S. Department of Transportation, Dayton, 2005.

[63] G. R. Spedding and J. McArthur. Span Efficiencies of Wings at Low Reynolds Numbers. *Journal of Aircraft*, 47(1):120–128, January 2010.

[64] Robert C. Juvinall and Kurt M. Marshek. *Fundamentals of Machine Component Design*. Wiley & Sons, Hoboken, fourth edition, 2006.

[65] John A. Roberson and Clayton T. Crowe. *Engineering Fluid Mechanics*. Houghton Mifflin Company, Boston, 3rd edition, 1985.

Appendix A

Aerodynamic Derivations

A.1 Kutta-Joukowski integration

The Kutta-Joukowski theorem, equation A.1, relates the circulation around a circular cylinder to the lift it generates due to the free stream velocity in the fluid. Assuming incompressible flow, the theorem can be extended to a 2 dimensional body of arbitrary cross section such as an airfoil [7].

$$L' = \rho_\infty V_\infty \Gamma \tag{A.1}$$

By integrating equation A.1 along the span of the airfoil we can calculate the total lift generated as shown in equation A.2. The $_i nfty$ subscripts have been dropped for convenience.

$$L = \rho V \int_{-\frac{b}{2}}^{\frac{b}{2}} \Gamma(y)\, dy \tag{A.2}$$

To find the lift generated by a finite length airfoil, we must know the circulation distribution along the span and substitute it into the integral. For the purposes of this investigation we use an elliptical circulation distribution described by equation (A.3).

$$\Gamma(y) = \Gamma_o \sqrt{1 - \frac{y^2}{(\frac{b}{2})^2}} \tag{A.3}$$

Sub (A.3) into (A.2):

$$L = \rho V \int_{-\frac{b}{2}}^{\frac{b}{2}} \Gamma_o \sqrt{1 - \frac{y^2}{(\frac{b}{2})^2}}\, dy \tag{A.4}$$

Because the distribution is symmetrical about the midspan, $y = 0$:

$$L = 2\rho V \times \int_{0}^{\frac{b}{2}} \Gamma_o \sqrt{1 - \frac{y^2}{(\frac{b}{2})^2}}\, dy \tag{A.5}$$

let:

$$y = \frac{b}{2}\sin\theta \tag{A.6}$$

&

$$dy = \frac{b}{2}\cos\theta \; d\theta \tag{A.7}$$

$\therefore$ when $y = \frac{b}{2}$, $\theta = \frac{\pi}{2}$
& when $y = 0$, $\theta = 0$

Sub (A.6) & (A.7) into (A.5):

$$L = 2\rho V\Gamma_o \times \int_0^{\frac{\pi}{2}} \sqrt{1 - \frac{(\frac{b}{2})^2 \sin^2\theta}{(\frac{b}{2})^2}} \times \frac{b}{2}\cos\theta \; d\theta \tag{A.8}$$

but:
$1 - \sin^2\theta = \cos^2\theta$
& $\cos^2\theta = \frac{1+\cos 2\theta}{2}$

$$\therefore L = \rho V b\Gamma_o \int_0^{\frac{\pi}{2}} \frac{1+\cos 2\theta}{2} \; d\theta \tag{A.9}$$

integrating gives:

$$L = \rho V \frac{b\Gamma_o}{2}\left[\theta + \frac{\sin 2\theta}{2}\right]_0^{\frac{\pi}{2}} \tag{A.10}$$

Hence the total lift generated by a finite airfoil with an elliptical lift distribution is:

$$L = \rho V \frac{\pi b\Gamma_o}{4} \tag{A.11}$$

A.2 Vortex Induced Velocity Derivation

The derivation described below exactly follows the process shown in Katz [25] with one addition. While Kats shows the induced velocity at a point due to a vortex filament calculated using the Biot-Savart law, equation (A.12), this derivation includes a viscous core as modelled by the Burnham-Hallock vortex profile (A.21).

$$\vec{dq} = \frac{\Gamma}{4\pi}\frac{\vec{dl} \times \vec{r}}{|\vec{r}|^3} \tag{A.12}$$

The definition of a cross product, equation (A.13), allows us to write equation (A.12) in the scalar form shown in equation (A.14), integrated along the length of the segment of vortex filament as defined in figure A.1.

$$\vec{dl} \times \vec{r} = |dl||r| \sin\beta \ \vec{n} \tag{A.13}$$

where $\vec{n}$ is the unit vector of the induced velocity, with it's direction defined by the right hand rule as shown in figure A.1.

$$q = \frac{\Gamma}{4\pi} \int_{\beta_1}^{\beta_2} \frac{\sin\beta}{r^2} dl \tag{A.14}$$

from figure A.1, it can be seem that:

$$r = \frac{d}{\sin\beta} \tag{A.15}$$

$$l = \frac{-d}{\tan\beta} \tag{A.16}$$

$\therefore$

$$dl = \frac{d}{\sin^2\beta} d\beta \tag{A.17}$$

sub (A.15) & (A.17) into (A.14)

$$q = \frac{\Gamma}{4\pi d} \int_{\beta_1}^{\beta_2} \sin\beta \ d\beta \tag{A.18}$$

$$q = \frac{\Gamma}{4\pi d} (\cos\beta_1 - \cos\beta_2) \tag{A.19}$$

To find the induced velocity due to an infinite straight vortex filament, $\beta_1 = 0$ and $\beta_2 = \pi$. The result is equation (A.20).

$$q = \frac{\Gamma}{2\pi d} \tag{A.20}$$

Equation (A.21) is for the same case but assumes a viscous core as modelled by the Burnham-Hallock profile.

$$q = \frac{\Gamma}{2\pi} \left(\frac{d}{d^2 + r_c^2} \right) \tag{A.21}$$

Equation (A.19) can be adjusted to include the Vortex core as shown in (A.22)

$$q = \frac{\Gamma}{4\pi} \left(\frac{d}{d^2 + r_c^2} \right) (\cos\beta_1 - \cos\beta_2) \tag{A.22}$$

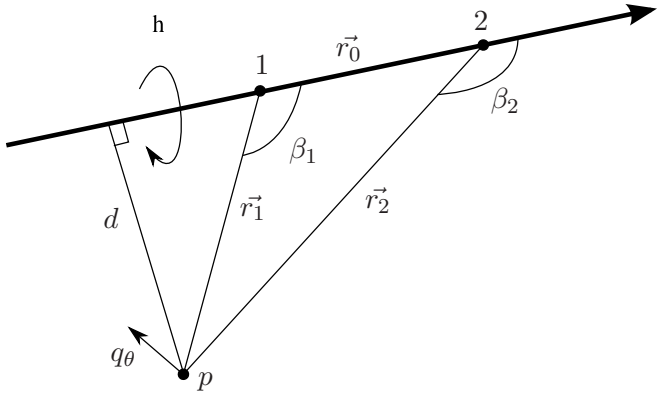


Figure A.1: Induced Velocity at a Point Due to a Vortex Filament

A.2.1 Numerical Computation

Katz explains the method described below to rewrite equation (A.19) in a vector form as defined by the vector notations in figure A.1. The variables of equation (A.1) can be written as:

$$\vec{r_o} = \vec{r_1} - \vec{r_2} \tag{A.23}$$

$$d = \frac{|\vec{r_1} \times \vec{r_2}|}{|\vec{r_o}|} \tag{A.24}$$

$$\cos\beta_1 = \frac{\vec{r_o} \cdot \vec{r_1}}{|\vec{r_o}||\vec{r_1}|} \tag{A.25}$$

$$\cos\beta_2 = \frac{\vec{r_o} \cdot \vec{r_2}}{|\vec{r_o}||\vec{r_2}|} \tag{A.26}$$

with the direction of the induced velocity defined by the unit vector:

$$n = \frac{\vec{r_1} \times \vec{r_2}}{|\vec{r_1} \times \vec{r_2}|} \tag{A.27}$$

Substituting (A.23) through (A.26) into (A.19) & (A.22) and multiplying by the unit vector gives (A.28) & (A.29) respectively.

$$\vec{q} = \frac{\Gamma}{4\pi} \frac{|\vec{r_o}|}{|\vec{r_1} \times \vec{r_2}|} \left(\frac{\vec{r_o} \cdot \vec{r_1}}{|\vec{r_o}||\vec{r_1}|} - \frac{\vec{r_o}\vec{r_2}}{|\vec{r_o}||\vec{r_2}|} \right) \frac{\vec{r_1} \times \vec{r_2}}{|\vec{r_1} \times \vec{r_2}|} \tag{A.28}$$

$$\vec{q} = \frac{\Gamma}{4\pi} \left(\frac{\frac{|\vec{r_1} \times \vec{r_2}|}{|\vec{r_o}|}}{\left(\frac{|\vec{r_1} \times \vec{r_2}|}{|\vec{r_o}|} \right)^2 + E^2} \right) \left(\frac{\vec{r_o} \cdot \vec{r_1}}{|\vec{r_o}||\vec{r_1}|} - \frac{\vec{r_o}\vec{r_2}}{|\vec{r_o}||\vec{r_2}|} \right) \frac{\vec{r_1} \times \vec{r_2}}{|\vec{r_1} \times \vec{r_2}|} \tag{A.29}$$

Appendix B

Design Calculations

B.1 General Strength Equations

The general strength equations can all be found in reference [64] and are listed below for convenience.

Area Moments of Inertia

Moment of Inertia of a Rectangle:

$$I_{rect} = \frac{wh^3}{12} \tag{B.1}$$

Moment of Inertia of a Circle:

$$I_{circ} = \frac{\pi d^4}{64} \tag{B.2}$$

Parallel Axis Theorem:

$$I = I_{cm} + Al_I^2 \tag{B.3}$$

Stress Equations

Direct Stress:

$$\sigma = \frac{F}{A} \tag{B.4}$$

Bending Stress:

$$\sigma = \frac{M\bar{y}}{I} \tag{B.5}$$

Principal Stresses:

$$\sigma_{1,2} = \frac{\sigma_{\vec{x}} + \sigma_{\vec{y}}}{2} \pm \sqrt{\tau_{xy}^2 + \left(\frac{\sigma_{\vec{x}} - \sigma_{\vec{y}}}{2}\right)^2} \tag{B.6}$$

Equivalent Stress for Distortion Energy Theory:

$$\sigma_e = \frac{\sqrt{2}}{2}\left[(\sigma_1 - \sigma_2)^2 + ...\,(\sigma_3 - \sigma_1)^2 + ...\,(\sigma_3 - \sigma_2)^2\right]^{\frac{1}{2}} \tag{B.7}$$

Maximum Shear Stress Theory:

$$\tau_{max} = \sqrt{\tau_{xy}^2 + \left(\frac{\sigma_{\vec{x}} - \sigma_{\vec{y}}}{2}\right)^2} \tag{B.8}$$

B.2 Aerodynamic Equations

General Aerodynamic Equations

The following equations are basic aerodynamic equations which can be found in references [7, 25] and are listed here for convenience.

Reynolds Number:

$$Re = \frac{V_\infty \bar{d} \rho}{\mu} \tag{B.9}$$

Aspect Ratio:

$$AR = \frac{b}{c} \tag{B.10}$$

Lift Coefficient Definition:

$$F_{\vec{z}} = -\frac{C_L \rho V_\infty^2 S}{2} \tag{B.11}$$

Drag Coefficient Definition:

$$F_{\vec{x}} = \frac{C_D \rho V_\infty^2 S}{2} \tag{B.12}$$

The total drag of a wing is made up of profile drag and induced drag. The former is obtained from airfoil section data. The second term in equation (B.13) represents the induced drag. The Oswald Efficiency Factor, $\bar{e}$, is a correction to account for the fact that the wing is finite and therefore not as efficient as would be predicted by two dimensional thin airfoil theory [63].

Total Drag Coefficient:

$$C_D = C_{dr} + \frac{C_L^2}{\pi \bar{e} AR} \tag{B.13}$$

Oswald Efficiency Factor:

$$\bar{e} = 1.78(1 - 0.045\ AR^{0.68}) - 0.64 \tag{B.14}$$

Rolling Moment Coefficient

$$L^* = \frac{\delta C_l \rho V_\infty^2 S b}{2} \tag{B.15}$$

Equations for Formation Flight Effect

The equations in this section are adapted from [31] to account for the induced effects due to the formation of two airfoils. Bizinos used an approximate method of simulating the two aircraft formation. The trailing aircraft was modelled by one horseshoe vortex for the wing and one for the horizontal stabilizer. The horizontal stabiliser was not simulated for the lead aircraft as it was assumed that the much smaller vortices generated by it would be captured by the main trailing vortices in the near field wake. The Burnham-Hallock core profile with a core radius of 3% of the span was used to calculate the induced effects on the trailing aircraft. The effects of the quarter chord bound vortex were shown to be negligible for the specific study at longitudinal separations of more than 1 span. The components of the induced coefficients which are attributed to the horizontal and vertical stabilisers, including induced side force coefficient and induced pitching and yawing moment coefficients, were neglected from the equations presented below as the models used in this investigation consist of only a rectangular planform wing.

Downwash Influence Factor:

$$\bar{\sigma}_{jk} = ln \left| \frac{\left(\left(\bar{\eta} - \frac{\pi}{4} \right)^2 + \bar{\mu}^2 \right) \left(\left(\bar{\eta} + \frac{\pi}{4} \right)^2 + \bar{\mu}^2 \right)}{(\bar{\eta}^2 + \bar{\mu}^2)} \right| \tag{B.16}$$

Moment Influence Factor:

$$\begin{aligned} \bar{\tau}_{jk} = -2\bar{\mu} \left(\tan^{-1} \left(\frac{\bar{\eta} - \frac{\pi}{4}}{\bar{\mu}} \right) + \tan^{-1} \left(\frac{\bar{\eta} + \frac{\pi}{4}}{\bar{\mu}} \right) - 2 \tan^{-1} \left(\frac{\bar{\eta}}{\bar{\mu}} \right) \right) ... \\ -\bar{\eta}\ ln \left| \frac{\left(\left(\bar{\eta} - \frac{\pi}{4} \right)^2 + \bar{\mu}^2 \right) \left(\left(\bar{\eta} + \frac{\pi}{4} \right)^2 + \bar{\mu}^2 \right)}{(\bar{\eta}^2 + \bar{\mu}^2)} \right| ... \\ -\frac{\pi}{8} ln \left| \frac{\left(\bar{\eta} + \frac{\pi}{4} \right)^2 + \bar{\mu}^2}{\left(\bar{\eta} - \frac{\pi}{4} \right)^2 + \bar{\mu}^2} \right| \end{aligned} \tag{B.17}$$

Induced Lift Coefficient:

$$\Delta C_{L_k} = -\frac{C_{L_j}}{\pi AR} \bar{\sigma}_{jk} \tag{B.18}$$

Induced Drag Coefficient:

$$\Delta C_{Di_k} = \frac{2 C_{L_j} C_{L_k}}{\pi^3 AR} \bar{\sigma}_{jk} \tag{B.19}$$

Induced Rolling Moment Coefficient:

$$\Delta C_{l_k} = \frac{C_{L_j}}{\pi AR} \bar{\tau}_{jk} \tag{B.20}$$

B.3 Tuft Grid Calculations

B.3.1 Drag Calculations

The drag coefficient for the braided nylon used in the tuft grid is determined using the Reynolds Number and the graph found in reference [65] which was generated from experimental results. Table B.1 shows the values of other relevant variables at the appropriate conditions.

Table B.1: Design Calculation Variables [58, 65]

Symbol	Value
μ	3.77×10^{-7} kgm^{-1}s^{-1}
ρ	1.20 kgm^{-3}
V_∞	25 ms^{-1}
d_{brd}	0.0003 m
d_{gmts}	0.005 m
l_{brd}	66.29 m
$C_{D_{cyl}}$	0.6
$C_{D_{sqrods}}$	2
d_{sqrods}	0.013 m
l_{sqrods}	2.54 m
r_I	0.0075 m

$Re_{brd} = 23873$
$A_{brd} = d_{brd} \times l_{brd}$
$\quad = 0.019887\ m^2$
$A_{sqrods} = d_{sqrods} \times l_{sqrods}$
$\quad = 0.03302\ m^2$

$\therefore$ from (B.12)
$F_{x_{brd}} = 4.47\ N$
$F_{x_{sqrods}} = 49.53\ N$

$F_{x_{tg}} = F_{x_{brd}} + F_{x_{sqrods}}$
$F_{x_{tg}} = 54.00\ N$

Since the tuft grid is symmetrical about the horizontal axis as well as the vertical axis, it is acceptable to say that the tuft drag force acts at the center of the tuft grid. The maximum bending stress can then be calculated using equation (B.5):

$M = F_{D_{tg}} \times l$

where $l = 0.313$ and is the length of the moment arm.
$M = 16.9\ Nm$
$I_{cm} = 3.068 \times 10^{-11}$
$I = 3.068 \times 10^{-11} + 2Ar_I^2$

The second term of the parallel axis theorem is multiplied by two because there are two shafts supporting the tuft grid as shown in figure B.1

$A = \pi \frac{d_{gmts}^2}{4}$
$\quad = 1.9 \times 10^{-5}$
$I = 3.068 \times 10^{-11} + 2 \times 10^{-9}\ m^2$
$\quad = 2.003 \times 10^{-9}$
$y = 0.01\ m$
$\sigma_{\vec{x}} = 8.44 \times 10^7\ MPa$

$\therefore$ from (B.7):
$\sigma_e = 8.44 \times 10^7\ MPa$
This is the same result as would have been obtained by the simple Maximum Normal Stress Theory because the problem involves only simple bending. The yield strength for different carbon steels is in the magnitude of 10^8 which is much larger than the calculated σ_e which indicates that the design will not fail.

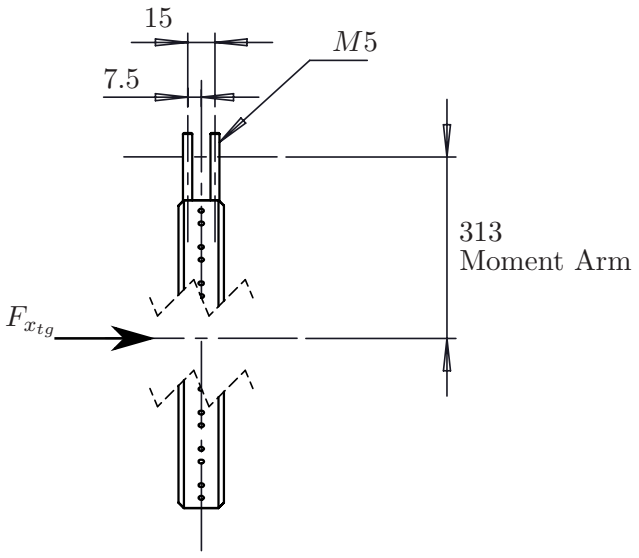


Figure B.1: Tuft Grid Mountings

B.4 Airfoil Calculations

Table B.2 and table B.3 show the relevant data for a NACA 0012 airfoil profile. The Reynolds number for each wings is calculated to be 3.74×10^6 at the test section wind speed of 25 ms^{-1} through equation (B.9) with the wing chord as the characteristic length.

Table B.2: NACA 0012 Profile Coordinates [61]

x (% c)	y (% c)	x (% c)	y (% c)
0	0	30	6.002
1.25	1.894	40	5.803
2.5	2.615	50	5.294
5.0	3.555	60	4.563
7.5	4.200	70	3.664
10	4.638	80	2.623
15	5.345	90	1.448
20	5.737	95	0.807
25	5.941	100	0.126

Table B.3: NACA 0012 Airfoil Data [61]

Symbol	Value
α	8°
b	0.3 m
c	0.047 m
AR	6.38
y	6 mm
m	0.16 kg
C_L	0.8
C_{dr}	0.009
C_M	0

The critical parts of the wing and assembly are the shaft that holds it in place as well as the pin which attaches the shaft to the wing as labelled A and B respectively in figure B.2.

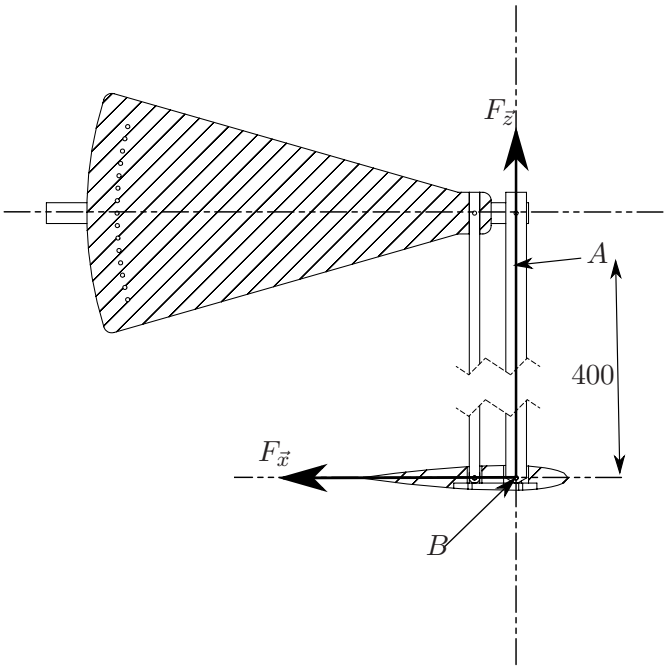


Figure B.2: Lift and Drag Diagram

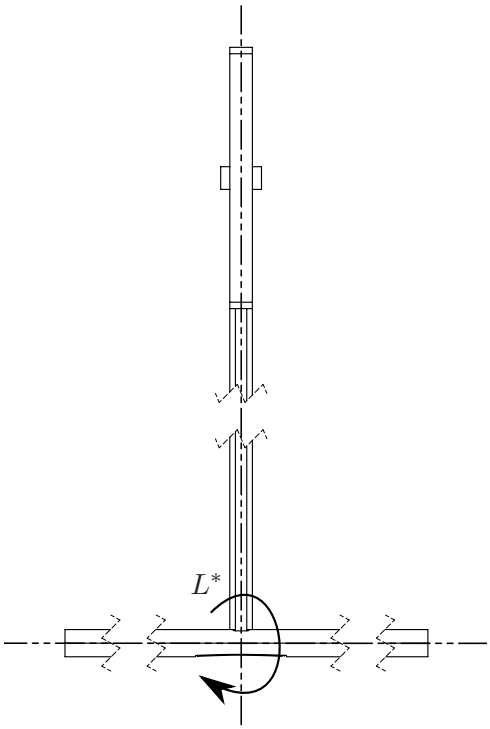


Figure B.3: Rolling Moment Diagram

The pins in the wing assemble are fashioned out of drill bits which demonstrate brittle fracture properties. The Maximum-Normal-Stress Theory is used in this case [64].
The formation flight effects increase the lift coefficient of the trailing wing, reduce the drag coefficient of the trailing wing provided there is less than 50 % wingtip overlap and induce a rolling moment on the trailing wing. For the strength calculations, a worst case scenario is assumed with maximum induced lift with no induced drag reduction on the trailing wing.

Lift Calculations

From equation (B.16) with $\bar{\mu} = \frac{r_c}{b} = 0.03$ and $\bar{\eta} = 0.76$ corresponding to the position of maximum predicted advantage:

$\bar{\sigma}_{jk} = -5.0544$

$\therefore$ Equation (B.18) gives:
$\Delta C_{L_k} = 0.2$
which is added to the lift coefficient for an isolated wing giving:
$C_{L_k} = 1$

Substituting this value into equation (B.11) results in:
$F_{\vec{z}} = -5.3$ N

Rolling Moment Calculations

From equation (B.17):
$\bar{\tau}_{jk} = 1.0903$

$\therefore$ Equation (B.20) gives:

$\Delta C_{l_k} = 0.043$

Substituting this into equation (B.20) gives:

$\Delta L^* = 0.0700$ Nm

Drag Calculations

Using the result of equation (B.16) in equation (B.19) gives:
$\Delta C_{D_k} = -0.0408$ N

The drag calculated without the formation flight effects into account is obviously larger and will be used for the worst case scenario strength calculations. The Oswald efficiency factor, $\bar{e}$, is given by equation (B.14):

$\bar{e} = 0.858$

Substituting this into equation (B.13) gives:
$C_D = 0.067$
which in turn is substituted into equation (B.12) resulting in:
$F_{x_{wing}} = 0.35$ N

Strength of Wing Support Strut

Due to lift: The mass of the wing must be included. It is added to the lift force to account for negative angle of attack resulting in a lift force in the same direction as the the weight of the wing. Combining the weight of the wing and the lift generated by it and substituting into equation (B.4):
$\sigma_{\vec{z}} = 0.350$ MPa

Due to drag and rolling moment: The drag force creates a bending moment which is combined with the rolling moment by Pythagoras Theorem:

$$M = \sqrt{\left(0.4F_{x_{wing}}\right)^2 + \Delta L^{*2}}$$

from (B.2) $I = 3.068 \times 10^{-11}$
Substituting these values into equation (B.5) gives: $\sigma_b = 30.720$ MPa

Adding σ_b and $\sigma_{\vec{z}}$ because they are in the same direction results in:
$\sigma_{res} = 31.07$ MPa
This is much smaller than the yield strength of the material and the part will not fail.

Strength of Wing Mounting Pin

The dominant stress in the pin joint is assumed to be shear stress. The rolling moment adds complexity to the analysis by increasing the shear stress on one side and decreasing the stress on the other side. Figure B.4 shows the forces in the vertical plane on a cross sectioned sketch of the relevant part of the wing mounting to help the conceptual analysis. The forces and the rolling moment are resolved in figure B.5 with the calculations presented below using the sign conventions as sketched. Again the weight of the wing must be included.
Vertical Plane:

$$R_{A_{\vec{z}}} = \frac{F_{\vec{z}}}{2} + \frac{\Delta L^*}{0.016}$$
$$= 7.83 \text{ N}$$
$$R_{B_{\vec{z}}} = \frac{F_{\vec{z}}}{2} - \frac{\Delta L^*}{0.016}$$
$$= 0.93 \text{ N}$$

Horizontal Plane:

$$R_{A_{\vec{x}}} = R_{B_{\vec{x}}} = \frac{F_{\vec{x}}}{2}$$
$$= 0.18 \text{ N}$$

Resultant:
$R_{A_{res}}$ is clearly the largest shear force in the z direction and is combined with the horizontal shear stress using Pythagoras Theorem:

$$R_{A_{res}} = \sqrt{R_{A_{\vec{z}}}^2 + R_{A_{\vec{x}}}^2}$$
$$= 7.83 \text{ N}$$

Substituting this into equation (B.4) gives:
$\tau = 9.97$ MPa

This is much smaller than the yield strength of the material and the part will not fail.

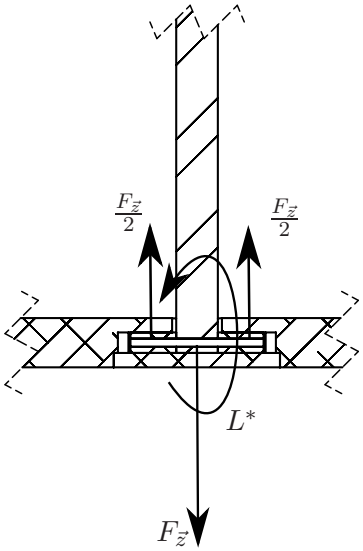


Figure B.4: Wing Mount Pin Lift Force and Rolling Moment Diagram

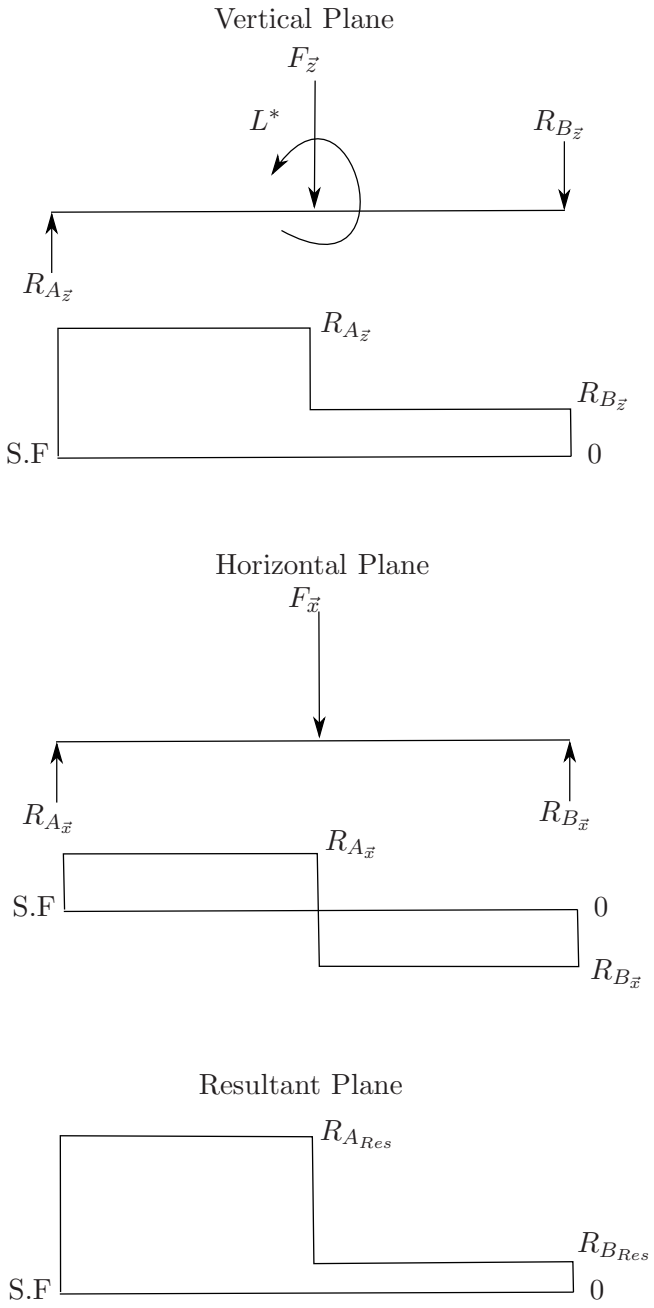


Figure B.5: Shear Force Analysis of Wing Mounting Pin

B.4.1 Blockage Calculations

While conducting wind tunnel experiments at low speeds, it is important to ensure that the Reynolds Number is the same as for the models which the

wind tunnel results will be compared to. In the case of this investigation, simulating a wind tunnel in the numerical model, these numbers can be set to be the same. The flow pattern in a wind tunnel tends to differ from that of a free air model. These differences arise from the limited cross sectional area of the wind tunnel as well as the various support structures used to hold the wind tunnel models in place.

The first interference effect to be discussed is the boundary constraint. There cannot be any flow through the walls of the wind tunnel. This zero normal flow condition affects the flow around the wind tunnel models. The flow above the aerofoil is compressed and results in an increase in axial velocity above the wing, a decrease in axial velocity below the wing and therefore an increase in circulation around the wing. Since lift is proportional to circulation, this results in an increase in lift on the wing. Pankhurst [58] has broken this general problem down into categories which have compounding effects.

Solid Blockage

Solid Blockage effects result in an increase in axial velocity past the wing due to the narrowing of the passage that the air flows through because of the cross section on the model taking part of the cross section of the wind tunnel. To account for this while still maintaining a constant mass flow rate, the velocity of the air must increase.

$$U_F = U_T(1 + \epsilon) \tag{B.21}$$

For a finite wing Pankhurst recommends equation (B.22) and $\tau^* = 0.75$ is recommended for an octagonal test section [58].

$$\epsilon = \left(\frac{\pi}{4}\right)^{\frac{1}{2}} \tau^* \left(\frac{A}{C^*}\right)^{\frac{3}{2}} \tag{B.22}$$

$\therefore \epsilon = 3.96 \times 10^{-4}$
$U_F = 1.0004\ U_T$
Therefore the increase in wind velocity due to the solid blockage effects is negligible.

Wake Blockage

The Wake Blockage effect is due to the displacement of the wake of the model evident in a reduction in pressure directly downstream of the model. It results in the decrease in the axial velocity in the wake of the model and a corresponding increase in the velocity of the free stream. The effects are considered negligible for wings of finite aspect ratio in a closed test section.

Lift Effect

The Lift Effect is due to the trailing and bound vortices of an aerofoil. It is similar to ground effect but on all four walls of the wind tunnel. This effect is described as the most relevant correction to account for in low speed tests. It results in an increase in effective angle of attack and also influences the streamlines in the wake of the model.

$$\Delta\alpha = \delta\frac{S}{C}C_L \tag{B.23}$$

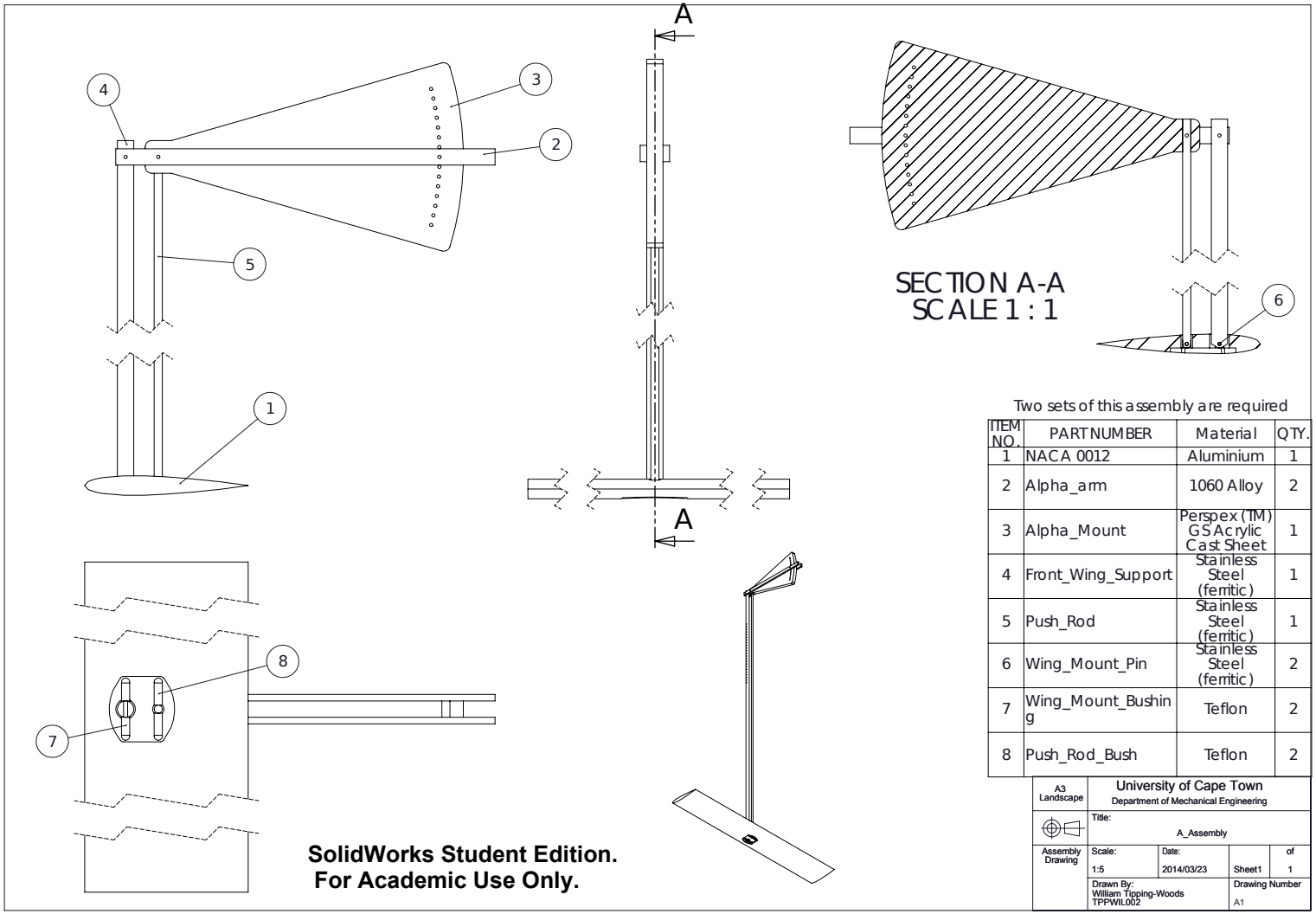
δ is a factor which takes into account the shape of the tunnel, relative size of the wing to the test section, boundary effects and the lift distribution on the wing. Pankhurst recommends $\delta = 0.133$ for the relevant experimental setup. Each wing will therefore see an increase in effective angle of attack, $\Delta\alpha = 0.2\,\text{deg}$. This value is also considered negligible.

Static Pressure is another effect described by Pankhurst. It deals with drag caused by horizontal buoyancy and flow acceleration. Only the lift induced component of drag is of major consideration in this experimental procedure and therefore this effect is not taken into account. The final wind tunnel effect described by Pankhurst is the Wall Boundary Layer effect. It is mainly relevant to an aerofoil spanning a wind tunnel and explains the difference between the wind tunnel results and the expected 2-D theory. It is therefore not relevant to this study.

These effects can all be modelled in a numerical simulation by placing images if the test wings twice the distance away from the test wing that then test wing is away from the wall. This creates the zero flow condition and the wind tunnel effects due to it. Due to computational cost, it is better to apply a correction factor to the input conditions rather than model the walls of the wind tunnel using images of vortices.

Appendix C

Design Drawings



ITEM NO.	PART NUMBER	Material	QTY.
1	NACA 0012	Aluminium	1
2	Alpha_arm	1060 Alloy	2
3	Alpha_Mount	Perspex (TM) GS Acrylic Cast Sheet	1
4	Front_Wing_Support	Stainless Steel (ferritic)	1
5	Push_Rod	Stainless Steel (ferritic)	1
6	Wing_Mount_Pin	Stainless Steel (ferritic)	2
7	Wing_Mount_Bushing	Teflon	2
8	Push_Rod_Bush	Teflon	2

Figure C.1: Wing Assembly

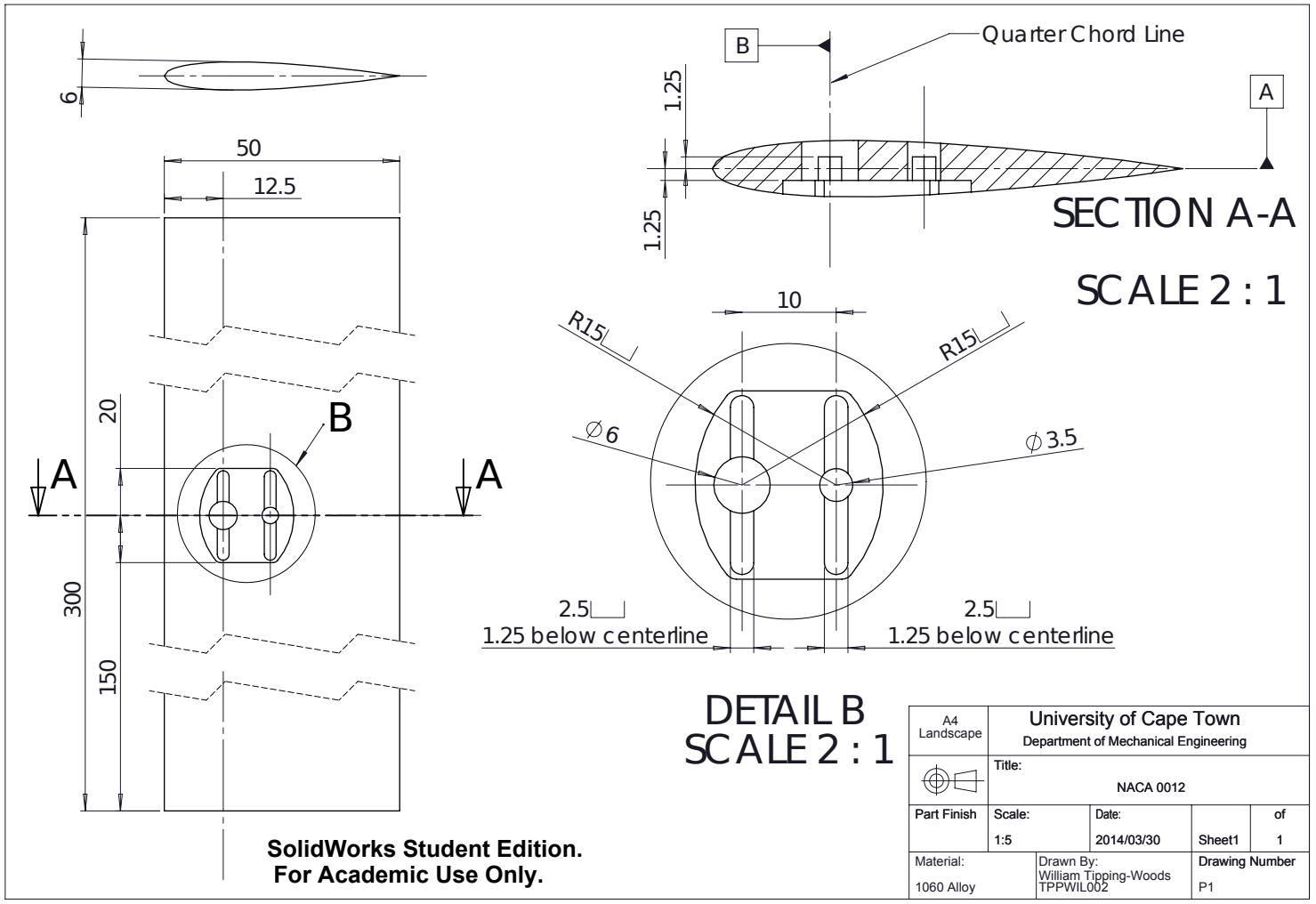


Figure C.2: NACA0012 Design Drawing

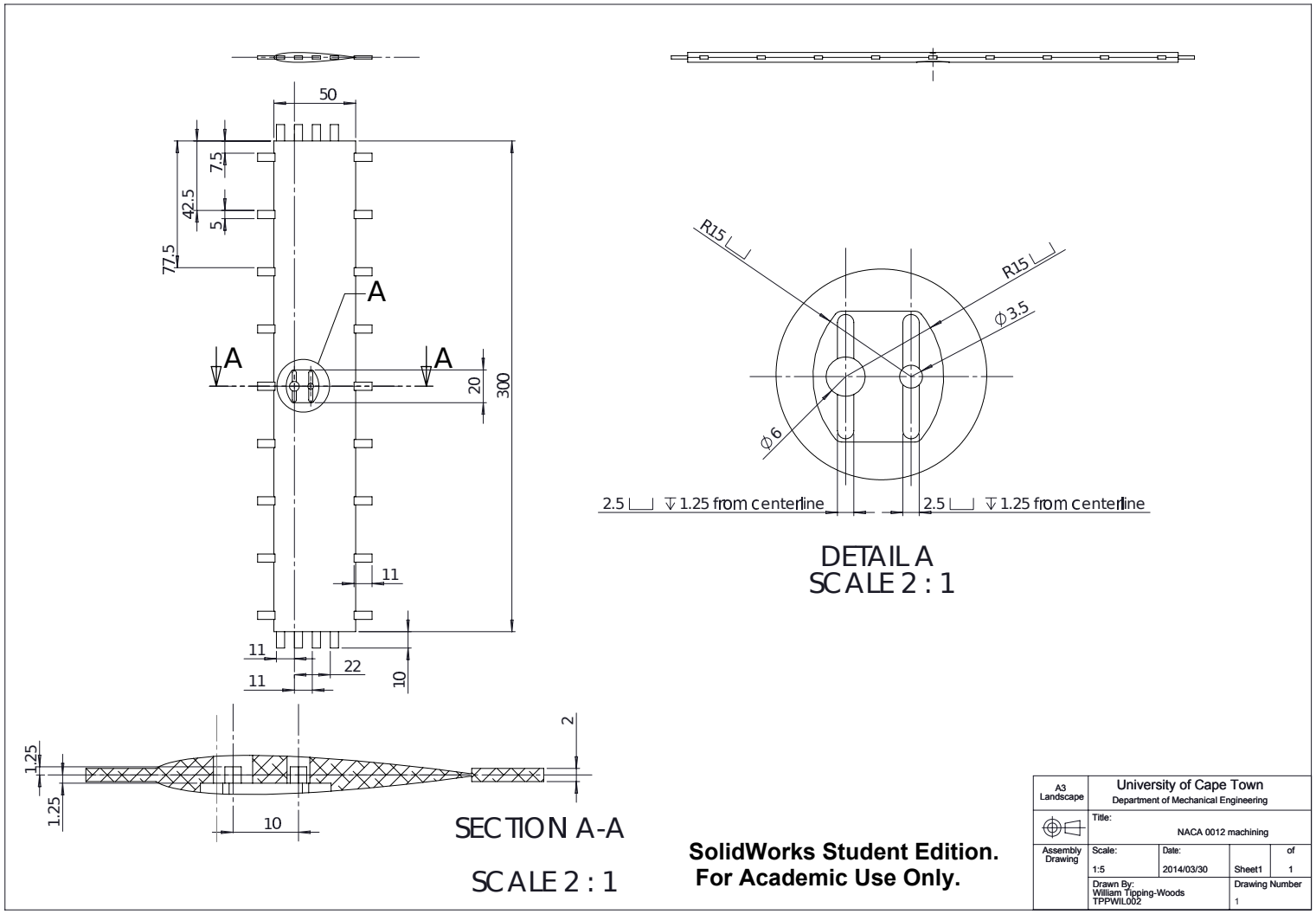
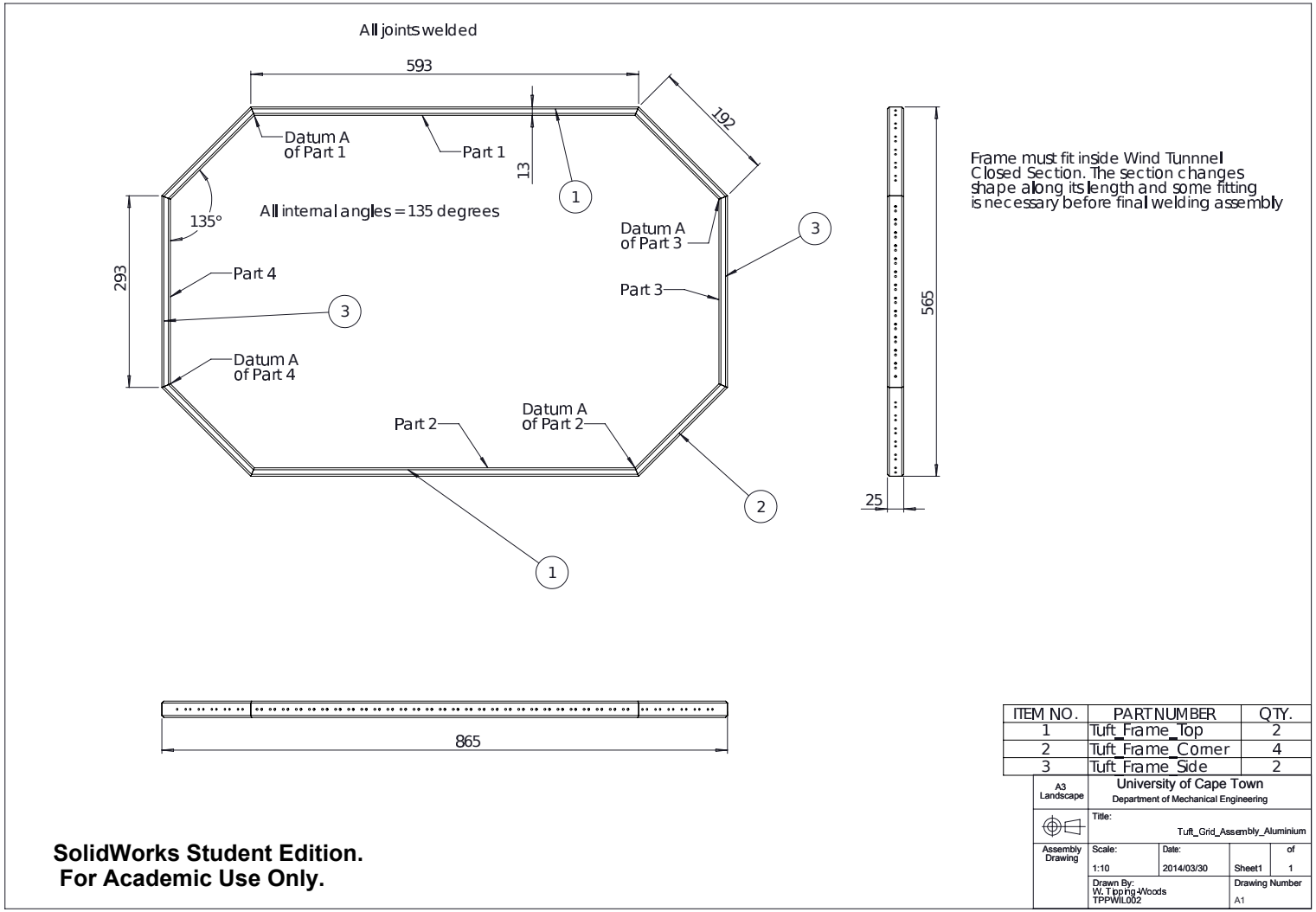


Figure C.3: NACA0012 Machining Design Drawing



ITEM NO.	PART NUMBER	QTY.
1	Tuft_Frame_Top	2
2	Tuft_Frame_Corner	4
3	Tuft_Frame_Side	2

A3 Landscape	University of Cape Town Department of Mechanical Engineering			
	Title: Tuft_Grid_Assembly_Aluminium			
Assembly Drawing	Scale: 1:10	Date: 2014/03/30	Sheet1	of 1
	Drawn By: W. Tipping-Woods TPPWIL002		Drawing Number A1	

Figure C.4: Tuft Grid

Appendix D

Two Wing Graphs

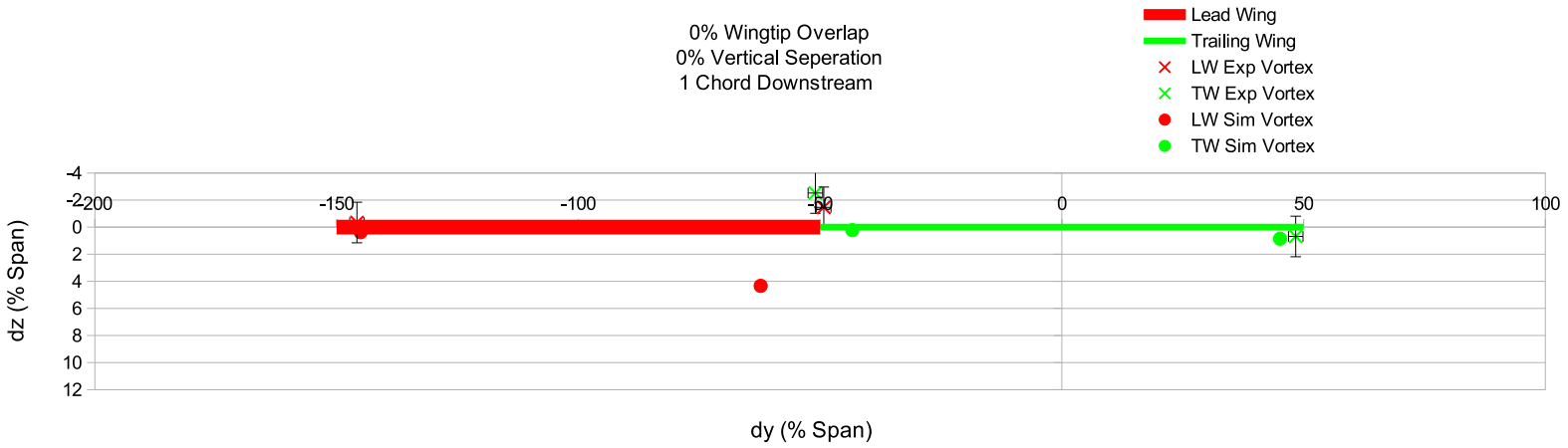


Figure D.1: Simulated Vortex Position vs Experimental Vortex Position 1 Chord Downstream of Two Wings

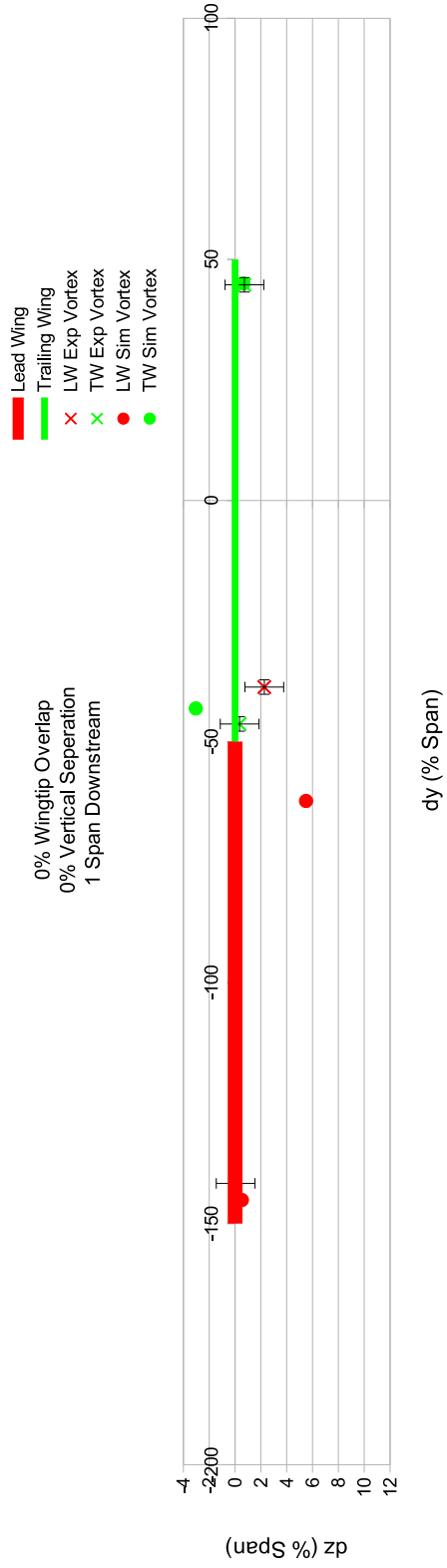


Figure D.2: Simulated Vortex Position vs Experimental Vortex Position 1 Span Downstream of Two Wings

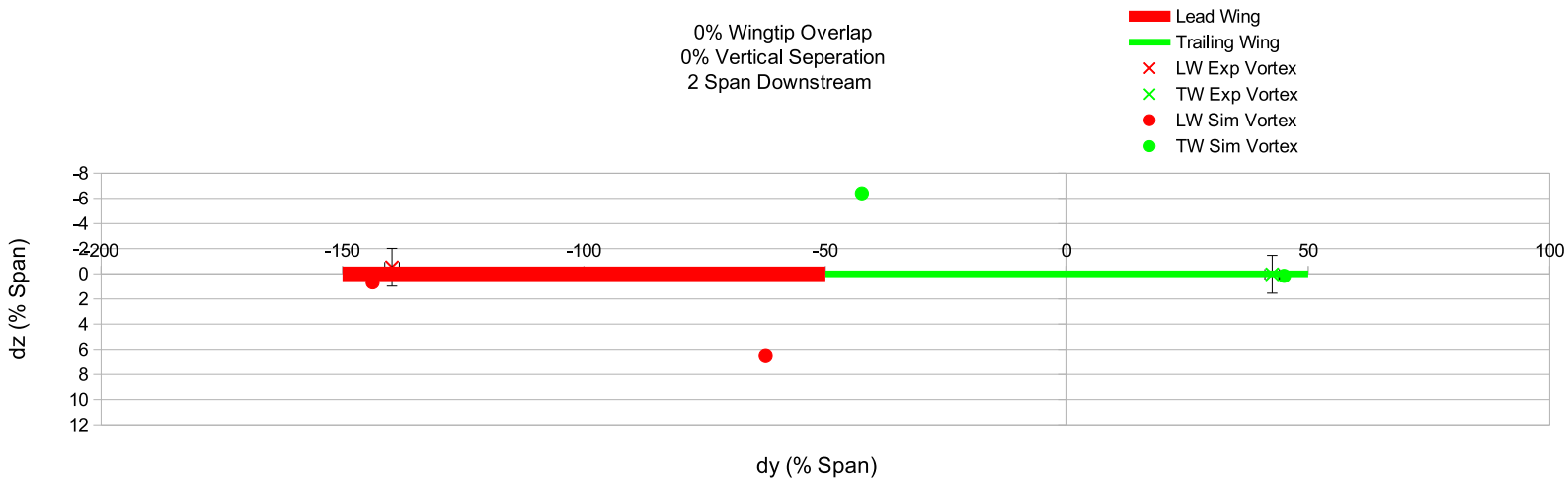


Figure D.3: Simulated Vortex Position vs Experimental Vortex Position 2 Spans Downstream of Two Wings

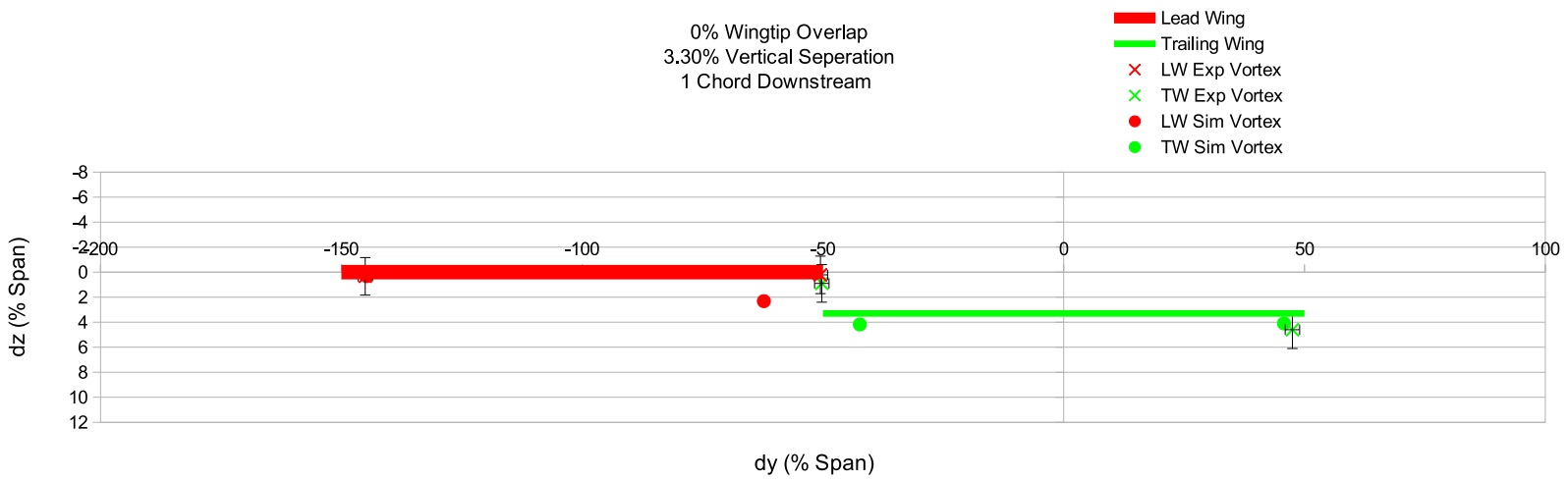


Figure D.4: Simulated Vortex Position vs Experimental Vortex Position 1 Chord Downstream of Two Wings

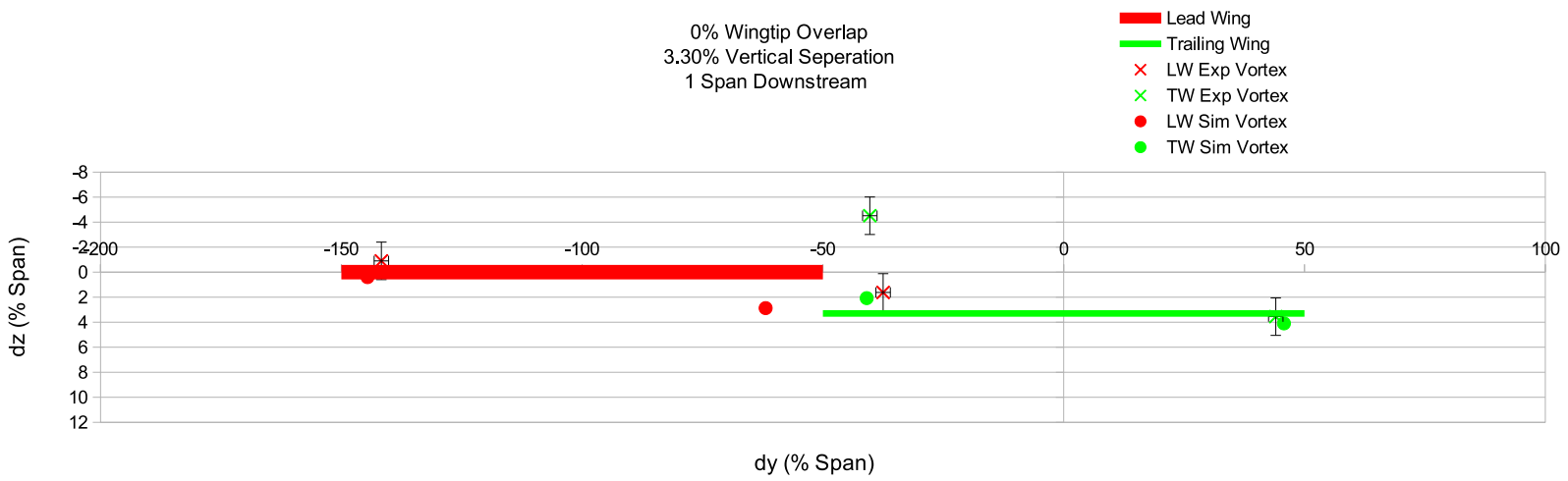


Figure D.5: Simulated Vortex Position vs Experimental Vortex Position 1 Span Downstream of Two Wings

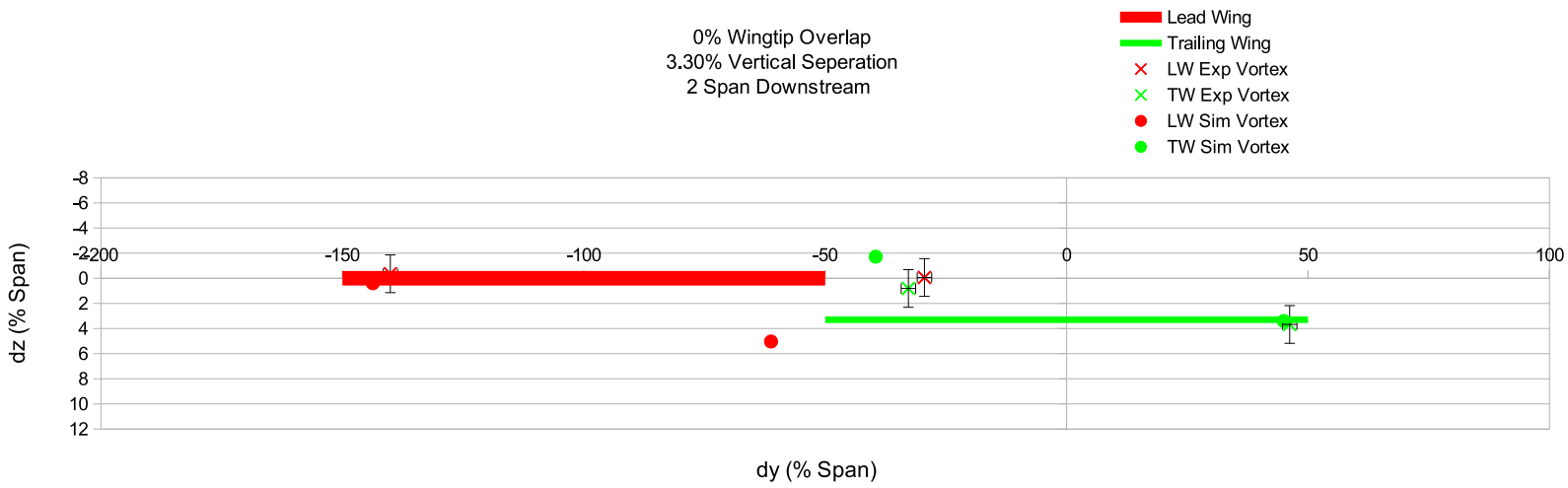


Figure D.6: Simulated Vortex Position vs Experimental Vortex Position 2 Spans Downstream of Two Wings

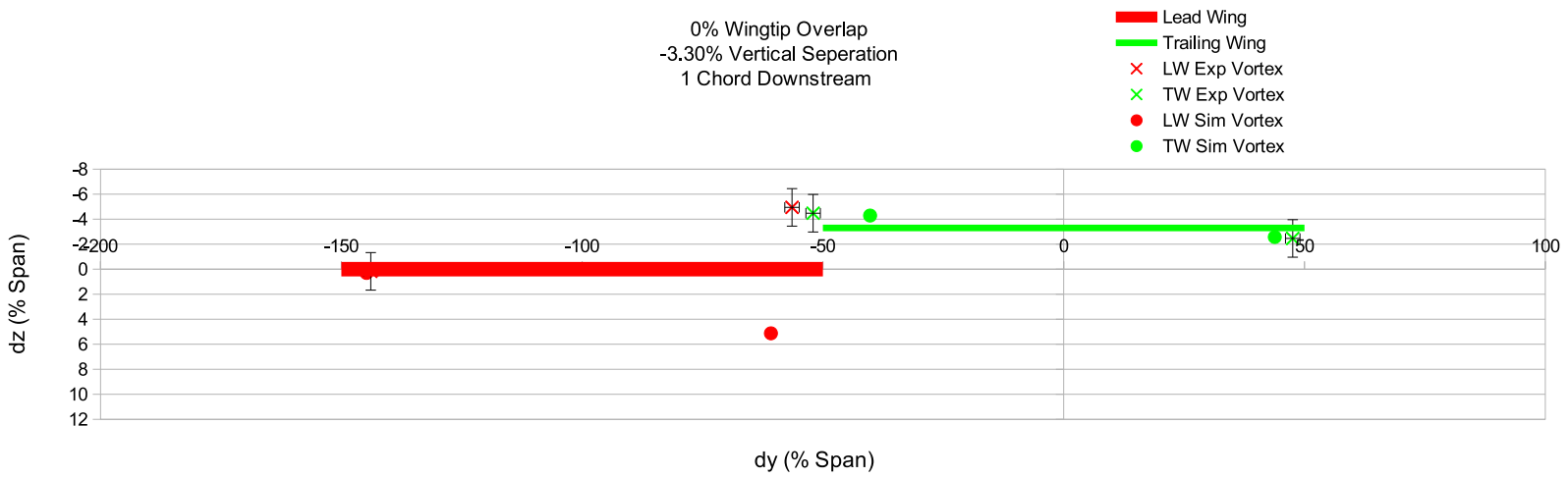


Figure D.7: Simulated Vortex Position vs Experimental Vortex Position 1 Chord Downstream of Two Wings

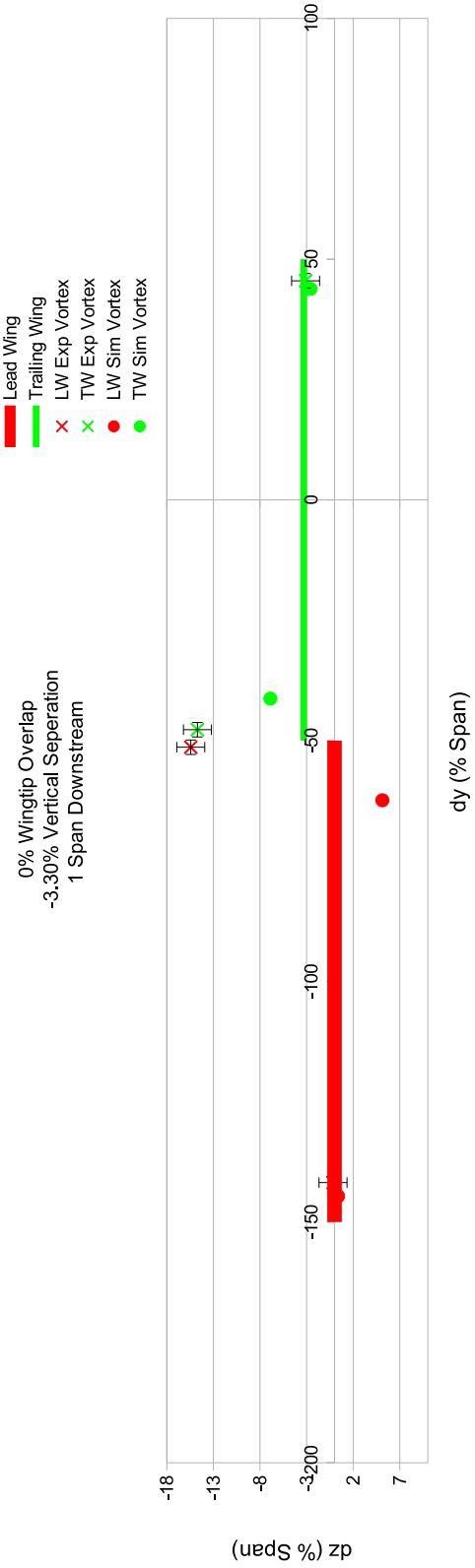


Figure D.8: Simulated Vortex Position vs Experimental Vortex Position 1 Span Downstream of Two Wings

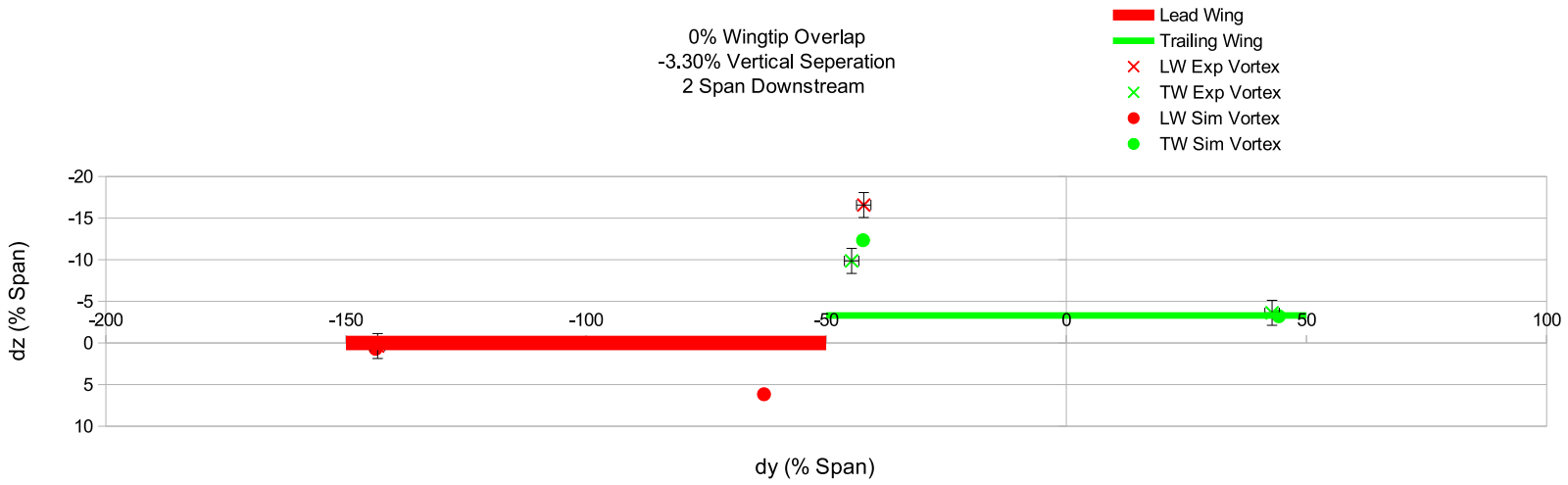


Figure D.9: Simulated Vortex Position vs Experimental Vortex Position 2 Spans Downstream of Two Wings

Appendix E

Matlab Code

E.1 Vortex Filament Method for 2 Aircraft Formation Flight

```
clear all
clc

%% Define simulation %%

sim = 'wind tunnel';                    % defines the simulation as either wind tunnel or cruise
distro = 'nommetric lift';              % symmetric lift distribution or non-symmetric (nommetric) lift distribution
dt = 0.0032;                            % dt = time step // seconds (s)
tf = 0.064;                             % tf = total time // seconds (s)
nf = 6;                                 % number of filaments per wing, nf = 4 produces two chord wise vortices on each wing resulting in two trailing vortex filaments on each semi-wing, therefore 4 trailing vortices on each wing.
ns = tf/dt;                             % number of segments per filament
lead_alpha = 8;                         % angle of attack of the lead wing // degrees
delta_alpha = 0;                        % difference between the angle of attack of the lead wing and the angle of attack of the trailing wing (+ive means trailing wing is at a greater angle of attack
```

```
    than the lead wing)
14
15 % x = +ive in the direction from the tail to the nose
    of the aircraft
16 % y = +ive from left (port) wingtip to right (
    starboard) wingtip
17 % z = +ive towards the ground
18 delta = 0;                                % Spanwise, Y
    separation, 0 = wingtip to wingtip, -ive means
    wingtips overlap // spanlengths
19 delta_x = 1;                              % Streamwise,
    X separation, 0 = wingtip to wingtip, = +ive means
    trailing wing is behind lead wing // spanlengths
20 delta_z = 0;                              % Vertical, Z
    separation, 0 = wingtip to wingtip +ive means
    trailing wing is below lead wing //
21
22
23 filename = ['nf = ' num2str(nf) ' ns = ' num2str(ns) '
    lead a = ' num2str(lead_alpha) ' delta a = '
    num2str(delta_alpha) ...
24     ' dx = ' num2str(delta_x) ' dy = ' num2str(delta)
        ' dz = ' num2str(delta_z) num2str(distro)
       num2str(sim)...
25     ' data.mat']; % creates filename to save data
       under
26
27 % Define which graphs to plot for the simulation, ye =
    plot, no = do not
28 % plot
29 plot_initial_positions = 'no';          % plot a graph
    of the quarter chord nodes of the filaments
30 plot_circulation_distribution = 'no';   % plot the
    circulation distribution for each wing
31 plot_error = 'ye';                      % plot a graph
    of the error between filament and streamlines (
    induced velocities)
32 plot_final = 'ye';                      % plot final
    vortex filaments
33 plot_tuft_grid = 'ye';                  % plot the
    virtual tuft grid
34 plot_wake_distribution = 'ye';          % plot
    vertical velocity distribution at a defined place
    in the wake
```

```
35 plot_induced_alpha = 'ye';                % plot the
       induced angle of attack accross the trailing wing
       span
36
37 t = 0;                                    % t = time (
       start time set to 0) // seconds (s)
38 nt = tf/dt;                               % nt = number
       of time steps
39 nt = round(nt);                           % ensure that
       the number of time steps is a whole number
40
41
42 if sim == 'wind tunnel'
43     % this is for the wind tunnel simulation
44     planexb = 1;                          % set the
           sampling distance for plots downstream of the
           trailing wing
45     aperture = 0.01;                      % set the
           aperture of the tuft grid
46     planez = 0;                           % set the
           vertical position of the vertical velocity
           sampling in the wake
47
48 else
49     % this is for the cruise simulation
50     planexb = 1;                          % set the
           sampling distance for plots downstream of the
           trailing wing
51     aperture = 2;                         % set the
           aperture of the tuft grid
52     planez = 0;                           % set the
           vertical position of the vertical velocity
           sampling in the wake
53
54 end
55
56 %% import starting data for both aircraft %
57 % the sub funcitons also define the positions of the
       vortices across the
58 % wing. The VLM is used for the wind tunnel case in
       order to find the lift
59 % distribution and position the VFM vortices to
       simulate the correct lift
60 % distribution and wake structure
```

```
61
62 [a] = starting_data_for_aircraft_LLM (nt, sim, distro,
       nf, delta, lead_alpha, delta_alpha, delta_x,
     delta_z);
63
64
65 %% plot the position of the initial nodes %%
66
67 % plot to show where the starting positions of the
     wings are as a check to
68 % make sure the code is working so far.
69 if plot_initial_positions ~= 'no';
70
71     for i = 1:((a(1).nf)/2)
72         ploti1(i,:) = a(1).s(1).f(i).n(1,:);
73         ploti2(i,:) = a(1).s(1).f(i).n(nt+1,:);
74         ploti3(i,:) = a(1).s(2).f(i).n(1,:);
75         ploti4(i,:) = a(1).s(2).f(i).n(nt+1,:);
76         ploti5(i,:) = a(2).s(1).f(i).n(1,:);
77         ploti6(i,:) = a(2).s(1).f(i).n(nt+1,:);
78         ploti7(i,:) = a(2).s(2).f(i).n(1,:);
79         ploti8(i,:) = a(2).s(2).f(i).n(nt+1,:);
80
81
82     end
83     figure;
84
85     plot3(ploti1(:,1), ploti1(:,2), ploti1(:,3), ':Xg'
         ); hold on;
86     plot3(ploti2(:,1), ploti2(:,2), ploti2(:,3), ':Xg'
         ); hold on;
87     plot3(ploti3(:,1), ploti3(:,2), ploti3(:,3), ':Xg'
         ); hold on;
88     plot3(ploti4(:,1), ploti4(:,2), ploti4(:,3), ':Xg'
         ); hold on;
89     plot3(ploti5(:,1), ploti5(:,2), ploti5(:,3), ':Xr'
         ); hold on;
90     plot3(ploti6(:,1), ploti6(:,2), ploti6(:,3), ':Xr'
         ); hold on;
91     plot3(ploti7(:,1), ploti7(:,2), ploti7(:,3), ':Xr'
         ); hold on;
92     plot3(ploti8(:,1), ploti8(:,2), ploti8(:,3), ':Xr'
         ); hold on;
93     set(gca,'zdir','reverse')
```

```
94      set(gca,'ydir','reverse')
95  else
96  end
97
98  %% plot the circulation distribution for each wing %%
99  if plot_circulation_distribution ~= 'no';
100
101     x = 2;
102     gamma2(x-1) = a(2).gamma_delta;
103     gamma1(x-1) = a(1).gamma_delta;
104
105     for i = 1:((a(2).nf-1)/2)
106
107         gamma2(x) = gamma2(x-1) + a(2).gamma_delta;
108         gamma1(x) = gamma1(x-1) + a(1).gamma_delta;
109         x = x + 1;
110
111     end
112
113     for i = 1:((a(2).nf)/2)
114
115         plotfil1(i) = a(1).s(1).f(a(2).nf/2 +1-i).n(nt
                +1,2);
116         plotfil1(i+ nf/2) = a(1).s(2).f(a(2).nf/2 +1-i
                ).n(nt+1,2);
117         plotfil2(i) = a(2).s(1).f(a(2).nf/2 +1-i).n(nt
                +1,2);
118         plotfil2(i+ nf/2) = a(2).s(2).f(a(2).nf/2 +1-i
                ).n(nt+1,2);
119     end
120
121     figure2 = figure;
122     plot(plotfil1(:), gamma1(:), 'g'); hold on;
123     plot(plotfil2(:), gamma1(:), 'g'); hold on;
124
125 else
126 end
127
128
129 %% Start time step %%
130 % begins moving the wings forward in time, shedding
      the trailing vortex
131 % segments, one per time step. At each time step the
      positions of the
```

```
132 % vortex segments are adjusted once in order to line
    up with the
133 % streamlines
134 tic
135
136 [a, i, nt] = time_step_generator_v3_5 (a, t, dt, nt);
137
138 toc
139
140 %% this keeps the solver going until convergance is
    found %%
141 tic
142
143 % [a, nt] = time_step_solver_v3_5 (a, nt, i, run for
    convergence, run for redistribution);
144 % Segment Manager
145 [a, nt] = time_step_solver_v3_5 (a, nt, i, 1);
146 % save the output data
147 save (['/home/will/Dropbox/Will Shared with ARG/Tippiz
    MK 3.4.2 eccentric lift distribution/Fillament
    Method/Data Output/', filename])
148
149 toc
150 %% plot final positions of vortices
151 if plot_final ~= 'no';
152
153     figure4 = figure;
154     for infed_wing = 1:2
155         for infed_side = 1:2
156             for infed_fil = 1:(a(infed_wing).nf/2)
157                 if (infed_wing == 1)
158
159                     plot3( a(infed_wing).s(infed_side)
                        .f(infed_fil).n(:,1), a(
                        infed_wing).s(infed_side).f(
                        infed_fil).n(:,2), a(infed_wing
                        ).s(infed_side).f(infed_fil).n
                        (:,3), 'r-' ); hold on;
160
161                 else
162
163                     plot3( a(infed_wing).s(infed_side)
                        .f(infed_fil).n(:,1), a(
                        infed_wing).s(infed_side).f(
```

```
                        infed_fil).n(:,2), a(infed_wing
                        ).s(infed_side).f(infed_fil).n
                        (:,3), 'g-' ); hold on;
164
165                 end
166
167             end
168         end
169     end
170     % directions of the graph are adjusted to line up
            with the defined
171     % coordinate system
172     set(gca,'zdir','reverse')
173     set(gca,'ydir','reverse')
174     daspect([1,1,1])
175
176 else
177 end
178
179
180
181 %% Builds the virtual tuft grid
182 tic
183 planex = a(2).s(1).f(1).n(nt) - a(2).c - a(2).b * (
        planexb);                   % sets the positon on the
        tusft grid plane from the trailing edge according
        to the number of spans, planexb, defined at the
        start of the simulation
184 [vtg_qi, n, m] = virtual_tuft_grid (a, i, nt, aperture
        , planex);                  % calculates the induced
        velociteis at the nodes of the tuft grid and
        creates the tufts as unit vectors assigned to each
        node
185 % Plots the tuft grid
186 if plot_tuft_grid ~= 'no';
187
188     figure5 = figure;
189
190     for vtg_i = 1:n-1
191         for vtg_ii = 1:m-1
192
193             x(1) = vtg_qi(vtg_i, vtg_ii).point(1,1);
194             x(2) = vtg_qi(vtg_i, vtg_ii).point(1,1) +
                    vtg_qi(vtg_i, vtg_ii).unit(1,1);
```

```
195             y(1) = vtg_qi(vtg_i, vtg_ii).point(1,2);
196             y(2) = vtg_qi(vtg_i, vtg_ii).point(1,2) +
                    vtg_qi(vtg_i, vtg_ii).unit(1,2);
197             z(1) = vtg_qi(vtg_i, vtg_ii).point(1,3);
198             z(2) = vtg_qi(vtg_i, vtg_ii).point(1,3) +
                    vtg_qi(vtg_i, vtg_ii).unit(1,3);
199
200             plot3(x(1), y(1), z(1), ':rx'); hold on;
201             plot3(x,y,z); hold on;
202
203         end
204     end
205 %%  Find Core Location %%
206
207     % Finds the Vortex Core Positions
208
209     % Y positions of the vortex cores
210     corz = 1; %initialises variable
211
212     for vtg_i = 1:n-1
213
214         % The z ordinate of the initial search is set
                at a thrid, (m/3), of
215         % the hight of the domain. This was chosen as
                it is the height of
216         % the lead wing in the domain and puts the
                search in the ball park
217         % of where the vortices are expected. In
                complex cases, the
218         % visualisation of the tuft grid will
                highlight any anomalies.
219
220         % multiplies the z component of the induced
                velocity at two
221         % neighboring nodes.
222         bisec = vtg_qi(vtg_i, m/3).unit(1,3) .* vtg_qi
                (vtg_i+1, (m/3)).unit(1,3);
223
224         % if the result is negative, it means that
                they have opposite signs
225         % and that there is a point between them where
                 the vertical
226         % velocity is 0, indicating the presence of a
                vortex core
```

```
227
228         if bisec <= 0
229             cory = corz;
230             % performs a linear interpolation to find
                    a better
231             % representation of the y ordinate of the
                    vortex core
232             int_len_v_y = norm(vtg_qi(vtg_i+1, (m/3)).
                    unit(1,3) - vtg_qi(vtg_i, (m/3)).unit
                    (1,3));
233             int_frac_y = norm(vtg_qi(vtg_i+1, (m/3)).
                    unit(1,3) / int_len_v_y);
234             int_len_p_y = norm(vtg_qi(vtg_i+1, (m/3)).
                    point(1,2) - vtg_qi(vtg_i, (m/3)).point
                    (1,2));
235
236             corz = cory;
237
238             % Z position of the vortex cores
239             % this for loop runs down the z axis in
                    the same way as the
240             % parent for loop ran across the y axis.
241             for vtg_ii = 1:m-1
242
243                 bisec2 = vtg_qi(vtg_i, vtg_ii).unit
                        (1,2) .* vtg_qi(vtg_i, vtg_ii+1).
                        unit(1,2);
244
245                 if bisec2 <= 0
246                     % performs a linear interpolation
                            to find a better
247                     % representation of the z ordinate
                            of the vortex core
248                     int_len_v_z = norm(vtg_qi(vtg_i,
                            vtg_ii+1).unit(1,2) - vtg_qi(
                            vtg_i, vtg_ii).unit(1,2));
249                     int_frac_z = norm(vtg_qi(vtg_i,
                            vtg_ii+1).unit(1,2) /
                            int_len_v_z);
250                     int_len_p_z = norm(vtg_qi(vtg_i,
                            vtg_ii+1).point(1,3) - vtg_qi(
                            vtg_i, vtg_ii).point(1,3));
251
252                     % builds a matrix of the core
```

```
                        positions to prepare for
253                     % a plot
254                     core(corz,1) = vtg_qi(vtg_i+1,(m
                            /3)).point(1,2) - (int_len_p_y*
                            int_frac_y);
255                     core(corz,2) = vtg_qi(vtg_i,
                            vtg_ii+1).point(1,3) - (
                            int_len_p_z*int_frac_z);
256                     planex_for_plot(corz,1) = planex;
257
258                     corz = corz + 1; % iterates the
                            core number to identify the
                            next core through the for loop
259
260                 else
261                 end
262             end
263
264
265
266         else
267         end
268     end
269
270     % plots the core positions on the tuft grid
271     plot3(planex_for_plot(:,1), core(:,1), core(:,2),
            ':go'); hold on;
272
273     % changes direction of axis to match chosen
            convention
274     set(gca,'zdir','reverse')
275     set(gca,'ydir','reverse')
276     daspect([1,1,1])
277     % changes the view of the tift grid to be from
            behind to match the wind
278     % tunnel data
279     az = 270;
280     el = 0;
281     view(az,el);
282     hold off;
283
284 else
285 end
286
```

```
287 toc
288 %% plot velocity distribution in the wake
289
290 aperture = (aperture/10);   % creates the sampling
       apperture for the wake distribution plot
291 if plot_wake_distribution ~= 'no';
292     [plotwake] = wake_vel_profile (a, i, nt, aperture,
          planex, planez);
293     figure6 = figure;
294     plot(plotwake(:,1), plotwake(:,2))
295     set(gca,'ydir','reverse')
296 else
297 end
298
299 %% Fillament method Lift
300 % sums the lift generated by the HS vortices of the
       VFM in order to be able
301 % to check for losses when changing form the VLM to
       the VFM
302 for i = 1:nf/2
303     fillift1(i) = a(1).rho * a(1).v(1,1) * a(1).
          gamma_delta * norm( a(1).s(1).f(i).n(nt+1,2) -
          a(1).s(2).f(i).n(nt+1,2));
304     fillift2(i) = a(2).rho * a(2).v(1,1) * a(2).
          gamma_delta * norm( a(2).s(1).f(i).n(nt+1,2) -
          a(2).s(2).f(i).n(nt+1,2));
305 end
306
307 filliftT1 = sum(fillift1);
308 filliftT2 = sum(fillift2);
```

E.2 Aircraft Starting Data

```
1 function [a] = starting_data_for_aircraft_LLM (nt, sim
      , distro, nf, delta,lead_alpha, delta_alpha,
      delta_x, delta_z)
2
3 % Define Axis System again for convenience
4     % +z is down
5     % +y is down right hand wing, starboard
6     % +x is flight direction
7
8 %% Aircraft_Start_Conditions %%
9
```

```
10      if strcmp(sim, 'wind tunnel')
11          % for wind tunnel siumulation
12          % a(1) creates the access structure for the
                lead aircraft, allowing
13          % properties to be defined specifically for
                that wing
14          a(1).rho = 1.2;
15          a(1).v = [23,0,0];

               % a1.v = aircraft 1. velocity // meters per
                second (ms^-1)
16          a(1).b = 0.300;                 % Scalar
                                                % a1.b =
               aircraft 1. span // meters (m)
17          a(1).c = 0.048;                 % Scalar
                                                % a1.c =
               chrod length of wing on aircraft 1 //
               meters (m)
18          a(1).dc = a(1).c * (3/4);       % Scalar
                                                % a1.dc =
               distance from quarter chord point to
               trailing edge on aircraft 1 // meters (m)
19          a(1).wcp = [0,0,0];

               % a1.wcp = aircraft 1. wing center point
               and quarter chord point // meters (m)
20          a(1).L = -5;                    % Negative due to
               axis system                % a1.L = aircraft 1.
                lift // Newtons (N)
21          a(1).nf = nf;                   % Scalar
                                                % a1.nf =
               number of filaments accross the whole wing
               1 //
22          a(1).E = 0.015;                 % Scalar
                                                % Vortex
               core size
23
24      else
25          % for crusie simulation of two 747 aircraft
26          a(1).rho = 0.35;

               % rho = density // kg/m (0.35 at 11.3 km
               altitude)
27          a(1).v = [250,0,0];
```

```
            % a1.v = aircraft 1. velocity // meters per
              second (ms^-1)
28          a(1).b = 60;                % Scalar
                                        % a1.b =
            aircraft 1. span // meters (m)
29          a(1).c = 8;                 % Scalar
                                        % a1.c =
            chrod length of wing on aircraft 1 //
            meters (m)
30          a(1).dc = a(1).c * (3/4);   % Scalar
                                        % a1.dc =
             distance from quarter chord point to
            trailing edge on aircraft 1 // meters (m)
31          a(1).wcp = [0,0,0];

            % a1.wcp = aircraft 1. wing center point
            and quarter chord point // meters (m)
32          a(1).L = -3968900; % Negative due to axis
            system +Z is down           % a1.L =
            aircraft 1. lift // Newtons (N)
33          a(1).nf = nf;
34          a(1).E = 3;
35
36      end
37
38      % Creates the structure for the second aircraft
           and gives the
39      % assigns properties to it.
40      a(2).rho = a(1).rho;
41      a(2).v = a(1).v;
42      a(2).b = a(1).b;
43      a(2).c = a(1).c;
44      a(2).dc = a(1).c * (3/4);
45      a(2).L =  a(1).L;
46      a(2).nf = a(1).nf;
47      % creates the separtion varibles from the script
           file inputs
48      dx = -a(1).b * delta_x;
49      dz = a(1).b * delta_z;
50      dy = (a(1).b + a(2).b) / 2;
51      delta_ = a(1).b * delta;
52      dtot = [dx,dy,dz] + [0,delta_,0];
53      a(2).wcp = a(1).wcp + dtot;
```

```
54      a(2).E = a(1).E;
55
56    %% Aircraft_filament_coordinates %%
57      if strcmp(distro, 'symmetric lift')
58          % for a symmetric lift distribution, the
              elliptical circulation
59          % distribution function is used to position
              the vortes filaments
60          % across the wing
61
62          for wing = 1:2
63              % finds the maximum circulation from the
                  lift
64              a(wing).gamma_o = -(4 * a(wing).L) / (a(
                  wing).rho * norm(a(wing).v) * pi() * a(
                  wing).b);   % a1.gamma_0 = maximum
                  circulation on wing 1 // (m^2/s)
65              % finds the total circulation from the
                  result of the intgration of the
66              % elliptical circulation distribution
                  function
67              a(wing).gamma_tot = -a(wing).L/ (a(wing).
                  rho * norm(a(wing).v));
                                        % a1.gamma_tot
                  = total circulation of wing 1 // (m^2/s
                  )
68              % defines the circulation of each of the
                  trailing vortices to
69              % be equal
70              a(wing).gamma_delta = (a(wing).gamma_o /
                  ((a(wing).nf/2) + 1));          % a1.
                  gamma_delta = circulation strenght for
                  each fillament on wing 1 // (m^2/s)
71
72
73
74              % calculating the spanwise distribution of
                   filaments with a constant
75              % gamma. The points are plotted along the
                  quarter chord point of the
76              % wing
77              for side = 1:2
78                  % starboard = right half of wing = s
                      (1)
```

```
79                  gamma = 0; % initial condition
80                  for i = 1:(a(wing).nf/2)
81
82                      % adds a new piece of shed
                            circualation to find a new y
                            working
83                      % towards the center of the wing
84                      gamma = gamma + a(wing).
                            gamma_delta;
85                      if side == 1
86
87                          % a(1).s(1).f(i).n(nt+1,:) is
                                the quarter chord
                                coordinate
88                          a(wing).s(side).f(i).n(nt+1,:)
                                = a(wing).wcp +...
89                              [0,(fillament_points_v3_5
                                    (gamma, a(wing).gamma_o
                                    , a(wing).b, 0))];
90                      else
91                          a(wing).s(side).f(i).n(nt+1,:)
                                = a(wing).wcp -...
92                              [0,(fillament_points_v3_5
                                    (gamma, a(wing).gamma_o
                                    , a(wing).b, 0))];
93
94                      end
95
96                      % creates points for initial
                            position of starting vortex
                            along the
97                      % trailing edge of the wing
98                      a(wing).s(side).f(i).n(1,:) = a(
                            wing).s(side).f(i).n(nt+1,:) -
                            [a(wing).dc,0,0];
99
100
101                 end
102             end
103         end
104
105
106     else
107
```

```
108         % to take into account a non-symetric lift
                distribution accross the
109         % wings caused by the influence of the wings
                on each other, a
110         % Vortex Lattice method is used
111
112         [a] = LLM(a, nt, lead_alpha, delta_alpha);
113
114     end
115
116 end
```

E.3 Filament Points

```
1  %% fillament position function %%%
2
3  % usses the ellyptical lift distribution equation,
       with y as the subject of
4  % the formula to find the starting position of each
       filament
5
6
7  function [y] = fillament_points_v3_5 (gamma, gamma_o,
       b)
8
9      y = (b/2) * (1 - (gamma/(gamma_o))^2) ^ (1/2);
10
11 end
```

E.4 Vortex Lattice Method

```
1  %% Vortex Lattice Model
2  % this model is used to solve for the lift
       distribution on the wings
3  % calls VLM_induced_velocity_loop
4
5  function [a] = VLM(a, nt, lead_alpha, delta_alpha)
6  tic
7
8  rho = a(1).rho;                          % the density
       is defined from the structure of aircraft 1
9  alpha = lead_alpha*pi()/180;             % the angle of
       attack is converted to radians
```

```
10 delta_alpha = delta_alpha*pi()/180;      % the cange in
       angle of attack is converted to radians
11 ndis = a(1).nf*22;                       % the number
      of horseshoe vortices placed along the quarter
      chord line is made larger than the number fo
      filaments use din the VFM
12 theta = pi()/2 + (pi()/ndis);            % the start of
       the sinusoidal discretisation of the wing is
      defined in radians
13
14 %% Creates the end points of the quarter chord segment
       of the chordwise distribution of horsshoe vortice
      %%
15 for i = 1:ndis+1
16
17     theta = theta - (pi()/ndis);
18
19     a(1).VLM_points(i,:) = a(1).wcp(1,:) + [0, (a(1).b
           /2)*sin(theta), 0];
20     a(2).VLM_points(i,:) = a(2).wcp(1,:) + [0, (a(2).b
           /2)*sin(theta), 0];
21
22 end
23
24 theta = pi()/2 + (pi()/(ndis*2));   % re-initialises
      the theta start point
25 %% Creates the collocation points for each horseshoe
      vortex along the half chord line of the wing %%
26 for i = 1:ndis
27
28     theta = theta - (pi()/(ndis*2))*2;
29
30     a(1).VLM_collocation(i,:) = a(1).wcp + [-a(1).c/2,
           (a(1).b/2)*sin(theta), 0];
31     a(2).VLM_collocation(i,:) = a(2).wcp + [-a(2).c/2,
           (a(2).b/2)*sin(theta), 0];
32
33 end
34
35 figure;
36
37 % plots the points of the chordwise vortex segments
      and the collocation
38 % points
```

```
39 plot3( a(1).VLM_points(:,1), a(1).VLM_points(:,2), a
   (1).VLM_points(:,3), ':Xr'); hold on;
40 plot3( a(2).VLM_points(:,1), a(2).VLM_points(:,2), a
   (2).VLM_points(:,3), ':Xb'); hold on;
41 plot3( a(1).VLM_collocation(:,1), a(1).VLM_collocation
   (:,2), a(1).VLM_collocation(:,3), ':Or'); hold on;
42 plot3( a(2).VLM_collocation(:,1), a(2).VLM_collocation
   (:,2), a(2).VLM_collocation(:,3), ':Ob'); hold off;
43 set(gca,'zdir','reverse')
44 set(gca,'ydir','reverse')
45
46
47 %% calculate the induced velocity at each collocation
   point
48 % initialise the induced collocation velocity matrix
49 for colvi = 1:ndis
50     a(1).VLM_collocation_velocity(colvi).HS = zeros(
       ndis*2,3);
51     a(2).VLM_collocation_velocity(colvi).HS = zeros(
       ndis*2,3);
52 end
53
54 %% Matrix A = downwash velocities
55 % initialise the downwash velocity matrix
56 A = zeros(ndis*2, ndis*2);
57
58 %% Matrix X = Gammas
59 % initialise the circulation matrix
60 X = ones(ndis*2, 1);
61
62 %% Matrix B = Normal component of free stream
63 % initialise the normal velocity matrix
64 B = ones(ndis*2,1);
65
66 % Velocity is negative because it is for the free
   stream, not the aircraft
67 % speed. To take into account the lift generated by
   the angle of attack,
68 % the freestream is titled relative to the axis system
    of the wing rather
69 % than the wing being tilted relative to the axis
   system of the free
70 % stream. This method simplies and speeds up the
   calculation process.
```

```
71
72 for Bi = 1:ndis
73     % to take into account the effects of each wing on
           each other, the two
74     % wings are combined into one matrix
75     B(Bi,1) = - sin(alpha)*(a(1).v(1,1));
76     B(Bi+ndis,1) = - sin(alpha + delta_alpha)*(a(1).v
           (1,1));
77 end
78
79 %% Inducec velocities of wing 1 on wing 1 (1,1)
80
81 %% (1,1)
82 winginfed = 1; % wing 1 is being influenced
83 winginfing = 1; % wing 2 is inducing the velocities
84
85 [a] = VLM_induced_velocity_loop (a, X, winginfed,
       winginfing, ndis, a(1).E);
86
87 % Build A Matrix = Induced velocities due to the
       horseshoe vortices from
88 % the output of the VLM_induced_velocity_loop function
89 for C = 1:ndis
90     for H = 1:ndis
91         CH = a(1).VLM_collocation_velocity(C).HS(H,:);
92         A(C,H) = CH(1,3);
93     end
94 end
95
96 %% Induced velocities of wing 2 on wing 1 (1,2)
97
98 %% (1,2)
99 winginfed = 1;  % wing 1 is being influenced
100 winginfing = 2; % wing 2 is inducing the velocities
101
102 [a] = VLM_induced_velocity_loop (a, X, winginfed,
       winginfing, ndis, a(1).E);
103
104 % Continue to build A Matrix = Induced velocities
105
106 for m = 1:ndis
107     for n = 1:ndis
108         mn = a(1).VLM_collocation_velocity(m).HS(n +
               ndis,:);
```

```
109             A(m,n+ndis) = mn(1,3);
110         end
111     end
112
113     %% Induced velocities of wing 1 on wing 2 (2,1)
114
115     %% (2,1)
116     winginfed = 2;
117     winginfing = 1;
118
119     [a] = VLM_induced_velocity_loop (a, X, winginfed,
            winginfing, ndis, a(1).E);
120
121     % Continue to build A Matrix = Induced velocities
122
123     for m = 1:ndis
124         for n = 1:ndis
125             mn = a(2).VLM_collocation_velocity(m).HS(n,:);
126             A(m+ndis,n) = mn(1,3);
127         end
128     end
129
130     %% Induced velocities of wing 2 on wing 2 (2,2)
131
132     %% (2,2)
133     winginfed = 2;
134     winginfing = 2;
135
136     [a] = VLM_induced_velocity_loop (a, X, winginfed,
            winginfing, ndis, a(1).E);
137
138
139     % Continue to build A Matrix = Induced velocities
140
141     for m = 1:ndis
142         for n = 1:ndis
143             mn = a(2).VLM_collocation_velocity(m).HS(n +
                    ndis,:);
144             A(m+ndis,n+ndis) = mn(1,3);
145         end
146     end
147
148     %% Solve for matrix for X %%
149     % solves for the circulation strengths that satisfy
```

```
    the condition of zero normal flow
% through the collocation points on both wings

% this -B takes it to the other side of the equation
B = -(B);
X = A\B;

toc

%% find the lift due to each HSV
% uses the Kutta-Joukowski theorem to find the lift
    due to each horseshoe
% vortex

for wing = 1:2

    for coli = 1:ndis
        if wing == 1
            colip = coli;
        else
            colip = ndis + coli;
        end
        a(wing).VLM_collocation_lift(coli) = rho *
            norm(a(wing).v) * X(colip) * norm(a(wing).
            VLM_points(coli + 1,:) - a(wing).VLM_points
            (coli,:));
    end
end
% sums up the lift produced by each wing
a(1).Lift = -sum(a(1).VLM_collocation_lift);
a(2).Lift = -sum(a(2).VLM_collocation_lift);

%% plot lift distribution
%
figure;
plot(a(1).VLM_collocation(:,2), a(1).
    VLM_collocation_lift, 'r'); hold on;
plot(a(2).VLM_collocation(:,2), a(2).
    VLM_collocation_lift, 'b'); hold off;


%% position of VFM QC nodes


```

```
187 % Assuming elliptical distribution just in order to
    chose
188 % a gamma_delta that is almost appropriate
189
190 % for loop works through each wing individually
191 for wing = 1:2
192
193     % finds the maximum circulation from the lift
194     a(wing).gamma_o = -(4 * a(wing).Lift) / (a(wing).
        rho * norm(a(wing).v) * pi() * a(wing).b);   %
        a1.gamma_0 = maximum circulation on wing 1 // (
        m^2/s)
195     % defines the constant strength of the vortex
        filaments
196     a(wing).gamma_delta = ( a(wing).gamma_o / ((a(wing
        ).nf/2) + 1));
197
198     % works through each side of the wing
199     for side = 1:2
200
201         % Depending on which wing is
202         if side == 1 && wing == 1
203             intergration = 0;
204             i = 0;
205
206             % works through each filament
207             for fil = 1:a(wing).nf/2
208
209                 % checks the circulation of the
                    horseshoe vortices from the
210                 % tip of the wing towards the center
                    until the strenght on
211                 % the iteration is greater than or
                    equal to the required
212                 % strength of the filament in question
                    . This is an
213                 % important part if the simulation. In
                     the VLM, the
214                 % circulation strength of a descrete
                    horseshoe vortex
215                 % is variable and accounts for the
                    lift over that section
216                 % of wing entirely. In the VFM, the
                    horseshoe vortices of
```

```
217                 % equal strenghts are layered ontop of
                        each other in
218                 % specific positions in order to
                        account for the
219                 % distribution of vorticity along the
                        span of the wing. In
220                 % the VLM the shed vorticity is the
                        difference between two
221                 % neighboring horsesoe vortices,
                        whereas in the VFM, the
222                 % shed vorticity is the streamwise
                        portion of the horseshoe
223                 % vortex itself.
224
225                 % Keeps running through the HSV's of
                        the VLM until the
226                 % strength is greater than or equal to
                        the VFM vortes
227                 % strength multiplied by the filament
                        in question.
228                 while norm(intergration) < norm(a(wing
                        ).gamma_delta*fil)
229                     i = i + 1;
230                     intergration = X(i);
231                     if norm(X(i)) >= norm(X(i+1))
232                         break
233                     end
234                 end
235                 % performs a linear interpolation
                        between the points of the
236                 % VLM quarter chord HSV segment so
                        that the VFM filament
237                 % can be placed so as to account for
                        the same shed
238                 % vorticity distribution as the VLM
239                 interpolation = (X(i) - a(wing).
                        gamma_delta*fil) .* (a(wing).
                        VLM_points(i,:) - a(wing).
                        VLM_points(i-1,:));
240                 % defines the nodes of the VFM along
                        the quarter cord line
241                 a(wing).s(side).f(fil).n(nt+1,:) = a(
                        wing).VLM_points(i,:) +
                        interpolation;
```

```
242                 % defines the nodes of the VFM along
                        the trailing edge of
243                 % the wing. These nodes create the
                        startign vortices and
244                 % get shed into the wake of the wing.
245                 a(wing).s(side).f(fil).n(1,:) = a(wing
                        ).s(side).f(fil).n(nt+1,:) - [a(
                        wing).dc,0,0];
246             end
247
248             % the elseif functions allow for the
                    positions of the VFM
249             % vortices to be found from tip to root on
                    each side of each
250             % wing
251         elseif side == 2 && wing == 1
252
253             intergration = 0;
254             i = ndis + 1;
255             for fil = 1:a(wing).nf/2
256
257                 while norm(intergration) < norm(a(wing
                        ).gamma_delta*fil)
258                     i = i - 1;
259                     intergration = X(i);
260                     if norm(X(i)) >= norm(X(i-1))
261                         break
262                     end
263                 end
264                 interpolation = (X(i) - a(wing).
                        gamma_delta*fil) .* (a(wing).
                        VLM_points(i,:) - a(wing).
                        VLM_points(i+1,:));
265
266                 a(wing).s(side).f(fil).n(nt+1,:) = a(
                        wing).VLM_points(i,:) -
                        interpolation;
267                 a(wing).s(side).f(fil).n(1,:) = a(wing
                        ).s(side).f(fil).n(nt+1,:) - [a(
                        wing).dc,0,0];
268
269             end
270         elseif side == 1 && wing == 2
271
```

```
272             intergration = 0;
273             i = ndis;
274             ii = 0;
275             for fil = 1:a(wing).nf/2
276 
277                 while norm(intergration) < norm(a(wing
                        ).gamma_delta*fil)
278                     i = i + 1;
279                     ii = ii + 1;
280                     intergration = X(i);
281                     if norm(X(i)) >= norm(X(i+1))
282                         break
283                     end
284                 end
285                 interpolation = (X(i) - a(wing).
                        gamma_delta*fil) .* (a(wing).
                        VLM_points(ii,:) - a(wing).
                        VLM_points(ii-1,:));
286 
287                 a(wing).s(side).f(fil).n(nt+1,:) = a(
                        wing).VLM_points(ii,:) +
                        interpolation;
288                 a(wing).s(side).f(fil).n(1,:) = a(wing
                        ).s(side).f(fil).n(nt+1,:) - [a(
                        wing).dc,0,0];
289 
290             end
291         elseif side == 2 && wing == 2
292 
293             intergration = 0;
294             i = 2*ndis + 1;
295             ii = ndis + 1;
296             for fil = 1:a(wing).nf/2
297 
298                 while norm(intergration) < norm(a(wing
                        ).gamma_delta*fil)
299                     i = i - 1;
300                     ii = ii - 1;
301                     intergration = X(i);
302                     if norm(X(i)) >= norm(X(i-1))
303                         break
304                     end
305                 end
306                 interpolation = (X(i) - a(wing).
```

```
                    gamma_delta*fil) .* (a(wing).
                    VLM_points(ii,:) - a(wing).
                    VLM_points(ii+1,:));
307
308                 a(wing).s(side).f(fil).n(nt+1,:) = a(
                    wing).VLM_points(ii,:) -
                    interpolation;
309                 a(wing).s(side).f(fil).n(1,:) = a(wing
                    ).s(side).f(fil).n(nt+1,:) - [a(
                    wing).dc,0,0];
310
311             end
312         end
313     end
314 end
```

E.5 Vortex Lattice Induced Velocity

```
1  % Uses the vortex_induced_velocity function to
       calculate the induced
2  % velocities at each of the colocation points.
3  % the "side" input is defined to mean Biot-Savart (3)
       or Burnham-Hallock
4  % vortex profile.
5
6  function [a] = VLM_induced_velocity_loop (a, X,
       winginfed, winginfing, ndis, E)
7          % For the VLM, the Biot-Savart profile is used
8          side = 3;
9
10 for coli = 1:ndis                    % influenced point
11     for HSii = 1:ndis                % influencing HS
12
13         % the if functions allow the induced
               velocities at each collocation
14         % point to be found due to all HS vortices of
               both wings
15         if winginfed == 1 && winginfing == 1
16             circulation = X(HSii);
17             HSiip = HSii;
18         else
19         end
20
21         if winginfed == 1 && winginfing == 2
```

```
22              circulation = X(ndis +HSii);
23              HSiip = ndis + HSii;
24          else
25          end
26
27          if winginfed == 2 && winginfing == 1
28              circulation = X(HSii);
29              HSiip = HSii;
30          else
31          end
32
33          if winginfed == 2 && winginfing == 2
34              circulation = X(ndis +HSii);
35              HSiip = ndis + HSii;
36          else
37          end
38
39          % first induced velocity is found due to QC
                segments
40          % therefore infinity is defiend as 0 for a
                finite vortex filament
41          infinity = 0;
42
43          influenced_node_1 = a(winginfed).
                VLM_collocation(coli,:);
44          influenced_node_2 = a(winginfed).
                VLM_collocation(coli,:);
45          influencing_node_1 = a(winginfing).VLM_points(
                HSii,:);
46          influencing_node_2 = a(winginfing).VLM_points(
                HSii+1,:);
47          % circulation is negative because I'm working
                from right to left on
48          % the wing
49          a(winginfed).VLM_collocation_velocity(coli).HS
                (HSiip,:) =...
50              a(winginfed).VLM_collocation_velocity(coli
                    ).HS(HSiip,:) + ...
51              vortex_induced_velocity (influenced_node_1
                    , influenced_node_2, influencing_node_1
                    , influencing_node_2, (-circulation), E
                    , infinity, side);
52      end
53  end
```

```
54
55 for coli = 1:ndis                          % influenced point
56     for HSii = 1:ndis                      % influencing HS
57
58         if winginfed == 1 && winginfing == 1
59             circulation = X(HSii);
60             HSiip = HSii;
61         else
62         end
63
64         if winginfed == 1 && winginfing == 2
65             circulation = X(ndis +HSii);
66             HSiip = ndis + HSii;
67         else
68         end
69
70         if winginfed == 2 && winginfing == 1
71             circulation = X(HSii);
72             HSiip = HSii;
73         else
74         end
75
76         if winginfed == 2 && winginfing == 2
77             circulation = X(ndis +HSii);
78             HSiip = ndis + HSii;
79         else
80         end
81
82         % due to right hand trailing vortices,
             infinite set to 1 for semi
83         % infinte vortex
84         infinity = 1;
85
86         influenced_node_1 = a(winginfed).
             VLM_collocation(coli,:);
87         influenced_node_2 = a(winginfed).
             VLM_collocation(coli,:);
88         influencing_node_1 = a(winginfing).VLM_points(
             HSii,:);
89         influencing_node_2 = a(winginfing).VLM_points(
             HSii,:) - [a(1).b*20, 0, 0];
90
91         a(winginfed).VLM_collocation_velocity(coli).HS
             (HSiip,:) =...
```

```
92              a(winginfed).VLM_collocation_velocity(coli
                    ).HS(HSiip,:) +...
93               vortex_induced_velocity (
                    influenced_node_1, influenced_node_2,
                    influencing_node_1, influencing_node_2
                    , (circulation), E, infinity, side);
94
95          % due to left hand trailing vortices
96          % circulation is negative because of the
                direction of the vortex,
97          % opposite between sides of the wing
98          infinity = 1;
99
100         influenced_node_1 = a(winginfed).
                VLM_collocation(coli,:);
101         influenced_node_2 = a(winginfed).
                VLM_collocation(coli,:);
102         influencing_node_1 = a(winginfing).VLM_points(
                HSii+1,:);
103         influencing_node_2 = a(winginfing).VLM_points(
                HSii+1,:) - [a(1).b*20, 0, 0];
104
105         a(winginfed).VLM_collocation_velocity(coli).HS
                (HSiip,:) =...
106             a(winginfed).VLM_collocation_velocity(coli)
                    .HS(HSiip,:) +...
107              vortex_induced_velocity (influenced_node_1
                    , influenced_node_2, influencing_node_1
                    , influencing_node_2, (-circulation), E
                    , infinity, side);
108
109     end
110 end
```

E.6 Vortex Inducted Velocity

```
1 % uses the Biot-Savart model or the Burnham-Hallock
      model to solve for the
2 % induced velocity at a point due to a vortex segment
3 % imports:
4
5 %%% within the code:
6     % r = matrix of vectors between the control point
          and the nodes of the
```

```
7           % vortex segment
8           % Cross = cross product of r vectors in
                question
9           % Modsq = the modulous of the cross product
                squared
10          % r_ = length of the
11          % circulation = vortex circulation strenght
12          % r_o_r = r_o dot product with
13          % r_o = vector of the vortex segment
14          % i = index unit used within each solver
                iteration
15          % E = core radius
16  % outputs:
17          % q = the induced velocity at point pc due to
                line segment between node f and node f+1
18
19      % infinity is for a semi infinite vortex, 1 = semi
            infinite
20
21  function [qi] = vortex_induced_velocity (
        influenced_node_1, influenced_node_2,
        influencing_node_1, influencing_node_2, circulation
        , E, infinity, side)
22          % sets the influenced point to be the mid
                point of the filament
23          influenced_point = ((influenced_node_1) + (
                influenced_node_2)) / 2;
24
25          r1 = influencing_node_1 - influenced_point;
                                    % vector from
                control point on vortex a to beginning of
                vortex i
26          r2 = influencing_node_2 - influenced_point;
                                    % vector from
                control point on vortex a to end of vortex
                i
27
28                  Cross = cross_product(r1, r2);

                        % cross product of ri and ri+1
29                  Mod = norm(Cross);

                        % Modulous of Cross Product of ri
                        and ri+1
```

```
30
31                  r_1 = norm(r1);

                       % lenght of ri
32                  r_2 = norm(r2);

                       % length of ri+1
33
34          r_o = r1 - r2;

               % vector of the vortex segment i
35          r_o_ = norm(r_o);

               % Length of vortex segment i
36
37          r_o_r1 = dot_product(r_o,r1);
                                                    %
               vortex segment r_oi doted with ri
38                r_o_r2 = dot_product(r_o,r2);

                       % vortex segment r_oi doted with ri
                       +1
39
40          h_ = Mod/r_o_;

               % height or perpendicular distance from
               control point to vector
41          if side == 3
42              hE = 1/h_;

                   % Biot-Savart Profile
43          else
44              hE = h_/((h_^2)+(E^2));

                   % height divided by height squared +
                   core diameter squared(used for Burnham
                   Hallock Model)
45          end
46          ang1 = (r_o_r1./(r_o_ * r_1));
47          ang2 = (r_o_r2./(r_o_ * r_2));
48
49          % here the angle goes to pi() to extend the
               vortex to infinity.
50          % Instead of changing the geomertry, i have
```

```
            chosen to just reverse
51          % the circulatino on the other side of the
            horseshoe voertex which
52          % gives the same result but is more in line
            with how the fillamnet
53          % method works
54
55          if (infinity == 1)
56              ang2 = -1;
57          else
58
59          end
60
61
62          Ang = ang1 - ang2;
63          Dir = (Cross)./Mod;

               % Gives the direction of the vortex
64
65          % protects the function from returning NaN
            when the vortices are in line
66          if (Mod == (0))
67              qi = [0,0,0];
68          else
69
70              qi = (circulation / (4*pi)) .* hE .* Ang
                .* Dir;                 % Velocity
                Equation
71
72          end
73
74  end
```

E.7 Time Step Generator

```
1   function [a, i, nt] = time_step_generator_v3_5 (a, t,
        dt, nt)
2   % this is the magnitude of the segments in meters
3   a(1).dn = norm(a(1).v) * dt;
4   a(2).dn = norm(a(2).v) * dt;
5
6   for i = 1:nt
7       % displays the time step to keep track of the
            progress of the
```

```
8       % simulation
9       t = t + dt
10
11      %% Generate the new nodes and propegates the
            quarter chord nodes
12
13      [a] = node_generator_v3_5 (a,i,nt);
14
15
16      %% Calculate the influence on each point due to
            each segment %%
17
18      [a, nt] = time_step_solver_v3_5 (a, nt, i, 0);
19
20
21  end
```

E.8 Node Generator

```
1   function [a] = node_generator_v3_5 (a,i,nt)
2
3       % each ii generates the new points behind the
            trailing edge for each
4       % filament
5       for ii = 1:2
6           % ii = wing number
7           for iii = 1:a(1).nf/2
8               % iii = filament number
9               for iiii = 1:2
10                  % iiii = side number
11                  a(ii).s(iiii).f(iii).n(i+1,:) = a(ii).
                        s(iiii).f(iii).n(i,:) + [a(ii).dn
                        ,0,0];
12                  a(ii).s(iiii).f(iii).n(nt+1,:) = a(ii)
                        .s(iiii).f(iii).n(nt+1,:) + [a(ii).
                        dn,0,0];
13              end
14          end
15      end
16
17  end
```

E.9 Segment Manager

```
% Segment Manager

function [a, nt] = time_step_solver_v3_5 (a, nt, i, xt
    )
    % the xt variable tells the function that the
       nodes are still being
    % shed and that it does not need to check for
       convergance yet.
    % xt = 1 means that the node generation has
       stopped and that the
    % function must continue running untill
       convergance is found
    if xt == 0;
        % Calculates the induced velocities at each
           collocation point due
        % to all the segments on both wings
        [a, nt] = Velocity_Loop_v3_5 (a, nt, i);

       %% Adjust each point according to the influence

        [a] = adjust_segments_v3_5 (a,i);

    else

        %% Check for convergence %%
        % once the vortex filaments have all been
           generated, the adjustment
        % continues to take place until the segments
           line up closely enough
        % with the streamlines of the flow

            figure;
            % initialises the variables
            xx = 1;
            errorr2 = zeros(2,1);
            errorr3 = zeros(2,1);
            errorr4(1) = 0.0001;    % forces the
               simulation into while loop for the
               first iteration
            errorr4(2) = 0;

            while (errorr4(xx) > 1e-5) % condition for
                convergence to be declared

```

```
34                  errorr4(1) = 0;
35                  xx = xx+1;
36                  % Calculates the induced velocities at
                        each collocation point due
37                  % to all the segments on both wings
38                  [a, nt] = Velocity_Loop_v3_5 (a, nt, i
                        );
39
40                  % convergence check compares the
                        direction vectors of all
41                  % of the segments with the direction
                        vectors of the induced
42                  % velocities at their collocation
                        points.
43
44                  for infed_wing = 1:2
45                      for infed_side = 1:2
46                          for infed_fil = 1:(a(
                                infed_wing).nf/2)
47                              for infed_node = 1:(i)
48                                  % it only checks up to
                                        the point of the
49                                  % trailing edge of the
                                        wing
50                                  % finds the segment
                                        vector
51                                  vectorcheck(infed_node
                                        ,:) =...
52                                      a(infed_wing).s(
                                            infed_side).f(
                                            infed_fil).n(
                                            infed_node,:) -
                                             a(infed_wing).
                                            s(infed_side).f
                                            (infed_fil).n(
                                            infed_node+1,:)
                                            ;
53                                  % finds the induced
                                        velocity vector
54                                  indvelcheck(infed_node
                                        ,:) =...
55                                      a(infed_wing).s(
                                            infed_side).f(
                                            infed_fil).
```

```
                                        indvel(
                                        infed_node,:);
56                                  % finds the cross
                                        product of the two
                                        in
57                                  % order to find out
                                        how large the
58                                  % difference between
                                        the two are
59                                  errorr(infed_node,:)
                                        =...
60                                      norm(cross(
                                            vectorcheck,
                                            indvelcheck));
61                              end
62                              % sum node errors
63                              errorr1(infed_fil) = sum(
                                    errorr);
64                          end
65                          % sum filament errors
66                          errorr2(infed_side) = sum(
                                errorr1);
67                      end
68                      % sum side errors
69                      errorr3(infed_wing) = sum(errorr2)
                            ;
70                  end
71                  % sum wing errors and find the mean
                        error and display in
72                  % order to keep track of the
                        convergance.
73                  errorr4(xx) = sum(errorr3) / (2*a(1).
                        nf*nt)
74                  % plot the progress of the convergence
75                  plot(errorr4); hold on; drawnow;
76
77                  if xx == 2
78                      errorr4(xx) = errorr4(xx) + 1
79                  else
80                  end
81
82                  % adjust the segments to line up with
                        the newly calculated
83                  % induced velocity. That way the
```

```
                    results are one more
84                % iteration accurate than the
                    convergence criteria
85                [a] = adjust_segments_v3_5 (a,i);
86
87            end
88        end
89    end
```

E.10 Velocity Loop

```
1
2   % The Velocity Loop runs through the filaments and
        calculates the induced
3   % velocities at each collocation point due to every
        vortex segment on both
4   % wings.
5
6   function [a, nt] = Velocity_Loop_v3_5 (a, nt, i)
7   % The first set of for loops runs from end of wake up
        to the trailing edge.
8
9   % Define influenced wing
10  for infed_wing = 1:2
11      % Define influenced side
12      for infed_side = 1:2
13          % Define influenced filament, starting from
                wingtip to root
14          for infed_fil = 1:(a(infed_wing).nf/2)
15              % initialise the induced velocity matrix
16              a(infed_wing).s(infed_side).f(infed_fil).
                    indvel = zeros(nt,3);
17              % Define influenced collocation point
18              for infed_node = 1:(i)
19                  % Define influencing wing
20                  for infing_wing = 1:2
21                      % Define influencing side
22                      for infing_side = 1:2
23                          % Define influencing filament
24                          for infing_fil = 1:(a(
                                infed_wing).nf/2)
25                              % Define influencing
                                    segment
26                              for infing_node = 1:(i)
```

```
27                                  % induced velocities due to the
28                                  % initially streamwise filaments
29
30                                  influenced_node_1 = a(infed_wing).s(infed_side).f(infed_fil).n(infed_node,:);
31                                  influenced_node_2 = a(infed_wing).s(infed_side).f(infed_fil).n((infed_node + 1),:);
32                                  influencing_node_1 = a(infing_wing).s(infing_side).f(infing_fil).n(infing_node,:);
33                                  influencing_node_2 = a(infing_wing).s(infing_side).f(infing_fil).n((infing_node + 1),:);
34                                  circulation = a(infing_wing).gamma_delta;
35                                  E = a(infing_wing).E;
36
37                                  % because the solver works from the
38                                  % back to the front of the wake, the if
39                                  % loops keeps the circulation opposite
40                                  % between the sides using the right hand
41                                  % rule for circulation strenght.
42
```

```
43                                      if (infing_side == 1)
44                                          circulation =
                                                circulation
                                                *(-1);
45
46                                      else
47
48                                      end
49
50                                      a(infed_wing).s(
                                            infed_side).f(
                                            infed_fil).indvel(
                                            infed_node,:) = ...
51                                      a(infed_wing).s(
                                            infed_side).f(
                                            infed_fil).indvel(
                                            infed_node,:) +...
52                                      vortex_induced_velocity
                                            (influenced_node_1,
                                             influenced_node_2,
                                             influencing_node_1
                                            ,
                                            influencing_node_2,
                                             circulation, E, 0,
                                             0);
53                                  end
54                              end
55                          end
56                      end
57                  end
58              end
59          end
60  end
61
62
63  % the if function is created in order to allow to
        simulation to deal with
64  % either the case of a single pair of trailing
        vortices from each wing or
65  % the case of multiple trailing vortices of each wing
66  if a(1).nf ~= 2
67      % induced velocity due to the quarter chord
            vortices
68      for infed_wing = 1:2
```

```
69          for infed_side = 1:2
70              for infed_fil = 1:(a(infed_wing).nf/2)
71                  for infed_node = 1:(i)
72                      for infing_wing = 1:2
73                          for infing_side = 1:2
74                              for infing_fil = 1:(a(
                                    infing_wing).nf/2-1)
75                                  % infing_node is nt+1
                                        for quarter chord
76                                  % vortex
77                                  influenced_node_1 = a(
                                        infed_wing).s(
                                        infed_side).f(
                                        infed_fil).n(
                                        infed_node+1,:);
78                                  influenced_node_2 = a(
                                        infed_wing).s(
                                        infed_side).f(
                                        infed_fil).n((
                                        infed_node),:);
79                                  influencing_node_1 = a
                                        (infing_wing).s(
                                        infing_side).f(
                                        infing_fil).n(nt
                                        +1,:);
80                                  influencing_node_2 = a
                                        (infing_wing).s(
                                        infing_side).f(
                                        infing_fil+1).n((nt
                                        +1),:);
81                                  circulation = a(
                                        infing_wing).
                                        gamma_delta;
82                                  E = a(infing_wing).E;
83                                      % the circulation
                                            is multiplied
                                            by the
84                                      % filament number
                                            because the
                                            segment
85                                      % circulation
                                            between the two
                                             points
86                                      % consists of
```

```
                                    multiple
                                    vortices
87                                % superimposed on
                                    one another
                                    because
88
89                                if (infing_side ==
                                     1)
90                                    circulation =
                                        circulation
                                        *(-1)*
                                        infing_fil;
91                                else
92                                    circulation =
                                        circulation
                                        *infing_fil
                                        ;
93                                end
94
95                              a(infed_wing).s(
                                    infed_side).f(
                                    infed_fil).indvel(
                                    infed_node,:) = ...
96                              a(infed_wing).s(
                                    infed_side).f(
                                    infed_fil).indvel(
                                    infed_node,:) +...
97                              vortex_induced_velocity
                                    (influenced_node_1,
                                     influenced_node_2,
                                     influencing_node_1
                                    ,
                                    influencing_node_2,
                                     circulation, E, 0,
                                     0);
98                          end
99                      end
100
101                     % for each wing, this section
                            of code takes into
102                     % account the vortex segment
                            across the center of
103                     % the wing from the inner most
                            filment of on side
```

```
104                         % to the inner most filament
                                of the other side.
105                         influenced_node_1 = a(
                                infed_wing).s(infed_side).f
                                (infed_fil).n(infed_node
                                +1,:);
106                         influenced_node_2 = a(
                                infed_wing).s(infed_side).f
                                (infed_fil).n((infed_node)
                                ,:);
107                         influencing_node_1 = a(
                                infing_wing).s(1).f(
                                infing_fil+1).n(nt+1,:);
108                         influencing_node_2 = a(
                                infing_wing).s(2).f(
                                infing_fil+1).n((nt+1),:);
109                         circulation = a(infing_wing).
                                gamma_delta;
110                         E = a(infing_wing).E;
111
112                         circulation = circulation
                                *(-1) * (infing_fil + 1);
113
114                         a(infed_wing).s(infed_side).f(
                                infed_fil).indvel(
                                infed_node,:) = ...
115                         a(infed_wing).s(infed_side).f(
                                infed_fil).indvel(
                                infed_node,:) +...
116                         vortex_induced_velocity(
                                influenced_node_1,
                                influenced_node_2,
                                influencing_node_1,
                                influencing_node_2,
                                circulation, E, 0, 0);
117
118                     end
119                 end
120             end
121         end
122     end
123     % induced velocity due to fixed streamwise
            vortices on wing surface
124     for infed_wing = 1:2
```

```
125         for infed_side = 1:2
126             for infed_fil = 1:(a(infed_wing).nf/2)
127                 for infed_node = 1:(i)
128                     for infing_wing = 1:2
129                         for infing_side = 1:2
130                             for infing_fil = 1:(a(
                                    infing_wing).nf/2)
131
132                                 influenced_node_1 = a(
                                        infed_wing).s(
                                        infed_side).f(
                                        infed_fil).n(
                                        infed_node,:);
133                                 influenced_node_2 = a(
                                        infed_wing).s(
                                        infed_side).f(
                                        infed_fil).n(
                                        infed_node+1,:);
134                                 influencing_node_1 = a
                                        (infing_wing).s(
                                        infing_side).f(
                                        infing_fil).n(i
                                        +1,:);
135                                 influencing_node_2 = a
                                        (infing_wing).s(
                                        infing_side).f(
                                        infing_fil).n(nt
                                        +1,:);
136                                 circulation = a(
                                        infing_wing).
                                        gamma_delta;
137                                 E = a(infing_wing).E;
138
139                                     if (infing_side ==
                                            1)
140                                         circulation =
                                                circulation
                                                *(-1);
141                                     else
142                                     end
143
144                                 a(infed_wing).s(
                                        infed_side).f(
                                        infed_fil).indvel(
```

```
                                        infed_node,:) = ...
145                                   a(infed_wing).s(
                                        infed_side).f(
                                        infed_fil).indvel(
                                        infed_node,:) +...
146                                   vortex_induced_velocity
                                        (influenced_node_1,
                                         influenced_node_2,
                                         influencing_node_1
                                        ,
                                        influencing_node_2,
                                         circulation, E, 0,
                                         0);
147                                end
148                            end
149                        end
150                    end
151                end
152            end
153        end
154
155        % induced velocity due to the starting
156        % vortices and the free stream
157        for infed_wing = 1:2
158            for infed_side = 1:2
159                for infed_fil = 1:(a(infed_wing).nf/2)
160                    for infed_node = 1:(i)
161                        for infing_wing = 1:2
162                            for infing_side = 1:2
163                                for infing_fil = 1:(a(
                                       infing_wing).nf/2-1)
164
165                                    % infing_node is 1
                                         for starting
                                        vortex
166                                    influenced_node_1
                                        = a(infed_wing)
                                        .s(infed_side).
                                        f(infed_fil).n(
                                        infed_node,:);
167                                    influenced_node_2
                                        = a(infed_wing)
                                        .s(infed_side).
                                        f(infed_fil).n
```

```
((infed_node +
    1),:);
168 influencing_node_1
    = a(
    infing_wing).s(
    infing_side).f(
    infing_fil).n
    (1,:);
169 influencing_node_2
    = a(
    infing_wing).s(
    infing_side).f(
    infing_fil+1).n
    ((1),:);
170 circulation = a(
    infing_wing).
    gamma_delta;
171 E = a(infing_wing)
    .E;
172
173 if (infing_side ==
     2)
174     circulation =
        circulation
        *(-1)*
        infing_fil;
175 else
176     circulation =
        circulation
        *infing_fil
        ;
177 end
178
179 a(infed_wing).s(
    infed_side).f(
    infed_fil).
    indvel(
    infed_node,:) =
     ...
180 a(infed_wing).s(
    infed_side).f(
    infed_fil).
    indvel(
    infed_node,:)
```

```
                                    +...
181                                vortex_induced_velocity
                                    (
                                    influenced_node_1
                                    ,
                                    influenced_node_2
                                    ,
                                    influencing_node_1
                                    ,
                                    influencing_node_2
                                    , circulation,
                                    E, 0, 0);
182
183                         end
184                 end
185
186                 influenced_node_1 = a(
                        infed_wing).s(infed_side).f
                        (infed_fil).n(infed_node,:)
                        ;
187                 influenced_node_2 = a(
                        infed_wing).s(infed_side).f
                        (infed_fil).n((infed_node +
                         1),:);
188                 influencing_node_1 = a(
                        infing_wing).s(1).f(
                        infing_fil+1).n(1,:);
189                 influencing_node_2 = a(
                        infing_wing).s(2).f(
                        infing_fil+1).n((1),:);
190                 circulation = a(infing_wing).
                        gamma_delta;
191                 E = a(infing_wing).E;
192
193                 circulation = circulation*(
                        infing_fil + 1);
194
195                 a(infed_wing).s(infed_side).f(
                        infed_fil).indvel(
                        infed_node,:) = ...
196                 a(infed_wing).s(infed_side).f(
                        infed_fil).indvel(
                        infed_node,:) +...
197                 vortex_induced_velocity(
```

```
                                influenced_node_1,
                                influenced_node_2,
                                influencing_node_1,
                                influencing_node_2,
                                circulation, E, 0, 0);
198
199
200                         end
201                         a(infed_wing).s(infed_side).f(
                                infed_fil).indvel(infed_node,:)
                                 = ...
202                         a(infed_wing).s(infed_side).f(
                                infed_fil).indvel(infed_node,:)
                                 -...
203                         a(infed_wing).v;
204                     end
205                 end
206             end
207         end
208
209
210 % This section of code takes into account the case
      where each wing consists
211 % of a single HSV with one trailing vortex placed on
      the side fo each wing.
212 else
213
214     for infed_wing = 1:2
215         for infed_side = 1:2
216             for infed_fil = 1:(a(infed_wing).nf/2)
217                 for infed_node = 1:(i)
218                     for infing_wing = 1:2
219                         for infing_side = 1:2
220                             for infing_fil = 1:(a(
                                    infing_wing).nf/2)
221                                 % This code is not
                                        needed for a
                                    single
222                                 % HSV on each wing
223 %                               % infing_node is nt
      +1 for quarter chord
224 %                               % vortex
225 %                               influenced_node_1 =
      a(infed_wing).s(infed_side).f(infed_fil).n(
```

```
    infed_node+1,:);
226 %                                   influenced_node_2 =
    a(infed_wing).s(infed_side).f(infed_fil).n((
    infed_node),:);
227 %                                   influencing_node_1 =
     a(infing_wing).s(infing_side).f(infing_fil).n(nt
    +1,:);
228 %                                   influencing_node_2 =
     a(infing_wing).s(infing_side).f(infing_fil+1).n((
    nt+1),:);
229 %                                   circulation = a(
    infing_wing).gamma_delta;
230 %                                  E = a(infing_wing).E
    ;
231 %
232 %                                       if (infing_side
    == 1)
233 %                                           circulation
    = circulation*(-1)*infing_fil;
234 %                                       else
235 %                                           circulation
    = circulation*infing_fil;
236 %                                       end
237 %
238 %                                   a(infed_wing).s(
    infed_side).f(infed_fil).indvel(infed_node,:) = ...
239 %                                   a(infed_wing).s(
    infed_side).f(infed_fil).indvel(infed_node,:) +...
240 %
    vortex_induced_velocity(influenced_node_1,
    influenced_node_2, influencing_node_1,
    influencing_node_2, circulation, E, 0, 0);
241                                 end
242                             end
243
244                             influenced_node_1 = a(
                                    infed_wing).s(infed_side).f
                                    (infed_fil).n(infed_node
                                    +1,:);
245                             influenced_node_2 = a(
                                    infed_wing).s(infed_side).f
                                    (infed_fil).n((infed_node)
                                    ,:);
246                             influencing_node_1 = a(
```

```
                                infing_wing).s(1).f(
                                infing_fil).n(nt+1,:);
247                             influencing_node_2 = a(
                                infing_wing).s(2).f(
                                infing_fil).n((nt+1),:);
248                             circulation = a(infing_wing).
                                gamma_delta;
249                             E = a(infing_wing).E;
250
251                             circulation = circulation
                                *(-1) * infing_fil;
252
253                             a(infed_wing).s(infed_side).f(
                                infed_fil).indvel(
                                infed_node,:) = ...
254                             a(infed_wing).s(infed_side).f(
                                infed_fil).indvel(
                                infed_node,:) +...
255                             vortex_induced_velocity(
                                influenced_node_1,
                                influenced_node_2,
                                influencing_node_1,
                                influencing_node_2,
                                circulation, E, 0, 0);
256
257                         end
258                     end
259                 end
260             end
261         end
262
263
264         % induced velocity due to fixed streamwise
              vortices on wing surface
265         for infed_wing = 1:2
266             for infed_side = 1:2
267                 for infed_fil = 1:(a(infed_wing).nf/2)
268                     for infed_node = 1:(i)
269                         for infing_wing = 1:2
270                             for infing_side = 1:2
271                                 for infing_fil = 1:(a(
                                    infing_wing).nf/2)
272                                     % infing_node is nt+1
                                        for quarter chord
```

```
273                 % vortex
274                 influenced_node_1 = a(
                        infed_wing).s(
                        infed_side).f(
                        infed_fil).n(
                        infed_node,:);
275                 influenced_node_2 = a(
                        infed_wing).s(
                        infed_side).f(
                        infed_fil).n(
                        infed_node+1,:);
276                 influencing_node_1 = a
                        (infing_wing).s(
                        infing_side).f(
                        infing_fil).n(i
                        +1,:);
277                 influencing_node_2 = a
                        (infing_wing).s(
                        infing_side).f(
                        infing_fil).n(nt
                        +1,:);
278                 circulation = a(
                        infing_wing).
                        gamma_delta;
279                 E = a(infing_wing).E;
280
281                     if (infing_side ==
                            1)
282                         circulation =
                            circulation
                            *(-1);
283                     else
284                     end
285
286                 a(infed_wing).s(
                        infed_side).f(
                        infed_fil).indvel(
                        infed_node,:) = ...
287                 a(infed_wing).s(
                        infed_side).f(
                        infed_fil).indvel(
                        infed_node,:) +...
288                 vortex_induced_velocity
                        (influenced_node_1,
```

```
                                                influenced_node_2,
                                                influencing_node_1
                                               ,
                                               influencing_node_2,
                                                circulation, E, 0,
                                                0);
289                                         end
290                                     end
291                                 end
292                             end
293                         end
294                     end
295                 end
296
297                 % induced velocity due to the starting
298                 % vortices and the free stream
299                 for infed_wing = 1:2
300                     for infed_side = 1:2
301                         for infed_fil = 1:(a(infed_wing).nf/2)
302                             for infed_node = 1:(i)
303                                 for infing_wing = 1:2
304                                     for infing_side = 1:2
305                                         for infing_fil = 1:(a(
                                                infing_wing).nf/2)
306                                                 % This code is
                                                     not needed
                                                     for a
307                                                 % signle HSV
                                                     on each
                                                     wing
308 %                                               % infing_node is
                                                 1 for starting vortex
309 %
                                                influenced_node_1 = a(infed_wing).s(infed_side).f(
                                                infed_fil).n(infed_node,:);
310 %
                                                influenced_node_2 = a(infed_wing).s(infed_side).f(
                                                infed_fil).n((infed_node + 1),:);
311 %
                                                influencing_node_1 = a(infing_wing).s(infing_side).
                                                f(infing_fil).n(1,:);
312 %
                                                influencing_node_2 = a(infing_wing).s(infing_side).
                                                f(infing_fil+1).n((1),:);
```

```
313 %                                       circulation = a(
    infing_wing).gamma_delta;
314 %                                      E = a(
    infing_wing).E;
315 %
316 %                                       if (infing_side
    == 2)
317 %                                           circulation
    = circulation*(-1)*infing_fil;
318 %                                       else
319 %                                           circulation
    = circulation*infing_fil;
320 %                                       end
321 %
322 %                                       a(infed_wing).s(
    infed_side).f(infed_fil).indvel(infed_node,:) = ...
323 %                                       a(infed_wing).s(
    infed_side).f(infed_fil).indvel(infed_node,:) +...
324 %
    vortex_induced_velocity(influenced_node_1,
    influenced_node_2, influencing_node_1,
    influencing_node_2, circulation, E, 0, 0);
325
326                             end
327                         end
328
329                         influenced_node_1 = a(
                                infed_wing).s(infed_side).f
                                (infed_fil).n(infed_node,:)
                                ;
330                         influenced_node_2 = a(
                                infed_wing).s(infed_side).f
                                (infed_fil).n((infed_node +
                                 1),:);
331                         influencing_node_1 = a(
                                infing_wing).s(1).f(
                                infing_fil).n(1,:);
332                         influencing_node_2 = a(
                                infing_wing).s(2).f(
                                infing_fil).n((1),:);
333                         circulation = a(infing_wing).
                                gamma_delta;
334                         E = a(infing_wing).E;
335
```

```
336                             circulation = circulation*
                                    infing_fil;
337
338                             a(infed_wing).s(infed_side).f(
                                    infed_fil).indvel(
                                    infed_node,:) = ...
339                             a(infed_wing).s(infed_side).f(
                                    infed_fil).indvel(
                                    infed_node,:) +...
340                             vortex_induced_velocity(
                                    influenced_node_1,
                                    influenced_node_2,
                                    influencing_node_1,
                                    influencing_node_2,
                                    circulation, E, 0, 0);
341
342
343                         end
344                         a(infed_wing).s(infed_side).f(
                                infed_fil).indvel(infed_node,:)
                                 = ...
345                         a(infed_wing).s(infed_side).f(
                                infed_fil).indvel(infed_node,:)
                                 -...
346                         a(infed_wing).v;
347                     end
348                 end
349             end
350         end
351 end
```

E.11 Segment Adjustment

```
1  function [a] = adjust_segments_v3_5 (a, i)
2      % Saves a copy of the data so far, incase the
           simulation is stopped
3      % prematurely.
4           save (['/home/will/Dropbox/Will Shared with
               ARG/Tippiz MK 3.4.2 eccentric lift
               distribution/Fillament Method/Data Output/'
               , 'winn'])
5      % adjusts the segments to have the same direction
           vector as the induced
6      % velocity that was calculated at their
```

```
        collocation points in the
7     % time_step_solver function
8     for infed_wing = 1:2
9         for infed_side = 1:2
10            for infed_fil = 1:(a(infed_wing).nf/2)
11                for infed_nodei = 1:(i)
12
13                    % reverse the itterations to work
                          from the anchor point
14                    % = nt at the trailing edge of the
                           wing
15                    infed_node = i+1-infed_nodei;
16                    % find unit vector of induced
                          velocity at each point
17                    a(infed_wing).s(infed_side).f(
                          infed_fil).indvel(infed_node,:)
                           =...
18                    (a(infed_wing).s(infed_side).f(
                          infed_fil).indvel(infed_node,:)
                          /...
19                    norm(a(infed_wing).s(infed_side).f
                          (infed_fil).indvel(infed_node
                          ,:)))*...
20                    norm(a(infed_wing).dn);
21
22                    a(infed_wing).s(infed_side).f(
                          infed_fil).n(infed_node,:) =...
23                    a(infed_wing).s(infed_side).f(
                          infed_fil).n(infed_node+1,:)
                          +...
24                    a(infed_wing).s(infed_side).f(
                          infed_fil).indvel(infed_node,:)
                          ;
25
26                    % indvel is multiplied by the
                          magnitude of the length
27                    % of the segment
28
29                end
30            end
31        end
32    end
33 end
```

E.12 Virtual Tuft Grid

```
function [vtg_qi, n, m] = virtual_tuft_grid (a, i, nt,
    aperture, planex)

% define the dimentions of the grid
n = (a(1).b*3/(aperture));      % set to 3 spans
    accross
m = (a(1).b/(2*aperture));      % set to half a span
    deep
for vtg_i = 1:n
    for vtg_ii = 1:m

        % defines the y and z ordinates for the point
            in question
        vtg_y = -a(1).b + aperture*(vtg_i - 1);
        vtg_z = -(a(1).b/4) + aperture*(vtg_ii - 1);

        vtg_qi(vtg_i, vtg_ii).point = [planex, vtg_y,
            vtg_z];
        % builds and defines the initial velocity at
            the point in question
        vtg_qi(vtg_i, vtg_ii).velocity = [0, 0, 0];

        influenced_node_1 = vtg_qi(vtg_i, vtg_ii).
            point;
        influenced_node_2 = vtg_qi(vtg_i, vtg_ii).
            point;

        %% Calculates the induced velocities %%

        % The first set of for loops runs from end of
            wake up to the trailing edge.
        % The tuft grid has not been expanded to take
            into account the case
        % of a single horseshoe vortex on each wing.
            Alterations in line
        % with the Velocity_Loop_v3_5 function will
            give it the capability
        % if desired
        % Calculates the induced velocities
        % Define influencing wing
        for infing_wing = 1:2
```

```
30              % Define influencing side
31              for infing_side = 1:2
32                  % Define influencing filament
33                  for infing_fil = 1:(a(infing_wing).nf
                        /2)
34                      % Define influencing segment
35                      for infing_node = 1:(i)
36                          % induced velocities due to
                                the
37                          % initially streamwise
                                filaments
38
39                          influencing_node_1 = a(
                                infing_wing).s(infing_side)
                                .f(infing_fil).n(
                                infing_node,:);
40                          influencing_node_2 = a(
                                infing_wing).s(infing_side)
                                .f(infing_fil).n((
                                infing_node + 1),:);
41                          circulation = a(infing_wing).
                                gamma_delta;
42                          E = a(infing_wing).E;
43
44                          % because the solver works
                                from the
45                          % back to the front of the
                                wake, the if
46                          % loops keeps the circulation
                                opposite
47                          % between the sides using the
                                right hand
48                          % rule for circulation
                                strenght.
49
50                           if (infing_side == 1)
51                               circulation =
                                     circulation*(-1);
52
53                           else
54
55                           end
56
57                          vtg_qi(vtg_i,vtg_ii).velocity
```

```
                                = vtg_qi(vtg_i,vtg_ii).
                                velocity +
                                vortex_induced_velocity(
                                influenced_node_1,
                                influenced_node_2,
                                influencing_node_1,
                                influencing_node_2,
                                circulation, E, 0, 0);
58
59                          end
60                      end
61                  end
62              end
63
64              % induced velocity due to the quarter chord
                    vortices
65              for infing_wing = 1:2
66                  for infing_side = 1:2
67                      for infing_fil = 1:(a(infing_wing).nf
                            /2-1)
68                          % infing_node is nt+1 for quarter
                                chord
69                          % vortex
70                          influencing_node_1 = a(infing_wing
                                ).s(infing_side).f(infing_fil).
                                n(nt+1,:);
71                          influencing_node_2 = a(infing_wing
                                ).s(infing_side).f(infing_fil
                                +1).n((nt+1),:);
72                          circulation = a(infing_wing).
                                gamma_delta;
73                          E = a(infing_wing).E;
74
75                          % the circulation is multiplied by
                                 the
76                          % filament number because the
                                segment
77                          % circulation between the two
                                points
78                          % consists of multiple vortices
79                          % superimposed on one another
                                because
80
81
```

```
82                          if (infing_side == 1)
83                              circulation = circulation
                                    *(-1)*infing_fil;
84                          else
85                              circulation = circulation*
                                    infing_fil;
86                          end
87
88                      vtg_qi(vtg_i,vtg_ii).velocity =
                            vtg_qi(vtg_i,vtg_ii).velocity +
                             vortex_induced_velocity(
                            influenced_node_1,
                            influenced_node_2,
                            influencing_node_1,
                            influencing_node_2, circulation
                            , E, 0, 0);
89
90                  end
91              end
92              % for each wing, this section of code
                    takes into
93              % account the vortex segment across the
                    center of
94              % the wing from the inner most filment of
                    on side
95              % to the inner most filament of the other
                    side.
96
97              influencing_node_1 = a(infing_wing).s(1).f
                    (infing_fil+1).n(nt+1,:);
98              influencing_node_2 = a(infing_wing).s(2).f
                    (infing_fil+1).n((nt+1),:);
99              circulation = a(infing_wing).gamma_delta;
100             E = a(infing_wing).E;
101
102             circulation = circulation *(-1) * (
                    infing_fil + 1);
103             vtg_qi(vtg_i,vtg_ii).velocity = vtg_qi(
                    vtg_i,vtg_ii).velocity +
                    vortex_induced_velocity(
                    influenced_node_1, influenced_node_2,
                    influencing_node_1, influencing_node_2,
                     circulation, E, 0, 0);
104
```

```
105         end
106
107         % induced velocity due to fixed streamwise
                vortices on wing surface
108
109         for infing_wing = 1:2
110             for infing_side = 1:2
111                 for infing_fil = 1:(a(infing_wing).nf
                        /2-1)
112
113                     influencing_node_1 = a(infing_wing
                            ).s(infing_side).f(infing_fil).
                            n(i+1,:);
114                     influencing_node_2 = a(infing_wing
                            ).s(infing_side).f(infing_fil).
                            n(nt+1,:);
115                     circulation = a(infing_wing).
                            gamma_delta;
116                     E = a(infing_wing).E;
117
118                         if (infing_side == 1)
119                             circulation = circulation
                                    *(-1);
120                         else
121                         end
122
123                     vtg_qi(vtg_i,vtg_ii).velocity =
                            vtg_qi(vtg_i,vtg_ii).velocity +
                             vortex_induced_velocity(
                            influenced_node_1,
                            influenced_node_2,
                            influencing_node_1,
                            influencing_node_2, circulation
                            , E, 0, 0);
124
125                 end
126             end
127         end
128
129
130         % induced velocity due to the starting
131         % vortices and the free stream
132
133         for infing_wing = 1:2
```

```
134             for infing_side = 1:2
135                 for infing_fil = 1:(a(infing_wing).nf
                        /2-1)
136
137                     % infing_node is 1 for
                            starting vortex
138                     influencing_node_1 = a(
                            infing_wing).s(infing_side)
                            .f(infing_fil).n(1,:);
139                     influencing_node_2 = a(
                            infing_wing).s(infing_side)
                            .f(infing_fil+1).n((1),:);
140                     circulation = a(infing_wing).
                            gamma_delta;
141                     E = a(infing_wing).E;
142
143                         if (infing_side == 2)
144                             circulation =
                                    circulation*(-1);
145                         else
146                         end
147
148                     vtg_qi(vtg_i,vtg_ii).velocity
                            = vtg_qi(vtg_i,vtg_ii).
                            velocity +
                            vortex_induced_velocity(
                            influenced_node_1,
                            influenced_node_2,
                            influencing_node_1,
                            influencing_node_2,
                            circulation, E, 0, 0);
149
150                 end
151             end
152         end
153
154         vtg_qi(vtg_i,vtg_ii).velocity = vtg_qi(vtg_i,
                vtg_ii).velocity - a(1).v;
155
156
157         % creates the unit vector
158         vtg_qi(vtg_i, vtg_ii).unit = ((vtg_qi(vtg_i,
                vtg_ii).velocity ./ norm(vtg_qi(vtg_i,
                vtg_ii).velocity))) * 0.5;
```

```
159
160     end
161
162 end
```

E.13 Wake Velocity Profile

```
1  function [plotwake] = wake_vel_profile (a, i, nt,
       aperture, planex, planez)
2  % The waek velocity plot has not been expanded to take
       into account the case
3  % of a single horseshoe vortex on each wing.
       Alterations in line
4  % with the Velocity_Loop_v3_5 function will give it
       the capability
5  % if desired
6
7  start = -a(1).b;    % sets the start point for the
       sampling
8  for pvi=1:(a(1).b*3/aperture) % sets the domain size
       for sampling to 3 spans
9
10     wplot = start + aperture*(pvi-1); % changes the
           node on each for loop iteration
11     plotwake(pvi,:) = [wplot,0];    % initialises the
           matrix to save the velocity to
12
13     influenced_node_1 = [planex, wplot, planez];    %
           defines the influenced node in terms that the
           induced velocity function can interpret
14     influenced_node_2 = [planex, wplot, planez];    %
           defines the influenced node in terms that the
           induced velocity function can interpret
15
16     % The first set of for loops runs from end of wake
           up to the trailing edge.
17     for infing_wing = 1:2
18         for infing_side = 1:2
19             for infing_fil = 1:(a(infing_wing).nf/2)
20
21                 for infing_node = 1:(i)
22                     % induced velocities due to the
23                     % initially streamwise filaments
24
```

```
25                      influencing_node_1 = a(infing_wing
                            ).s(infing_side).f(infing_fil).
                            n(infing_node,:);
26                      influencing_node_2 = a(infing_wing
                            ).s(infing_side).f(infing_fil).
                            n((infing_node + 1),:);
27                      circulation = a(infing_wing).
                            gamma_delta;
28                      E = a(infing_wing).E;
29
30                      % because the solver works from
                            the
31                      % back to the front of the wake,
                            the if
32                      % loops keeps the circulation
                            opposite
33                      % between the sides using the
                            right hand
34                      % rule for circulation strenght.
35
36                      if (infing_side == 1)
37                          circulation = circulation*(-1)
                                ;
38
39                      else
40
41                      end
42
43                      v = vortex_induced_velocity(
                            influenced_node_1,
                            influenced_node_2,
                            influencing_node_1,
                            influencing_node_2, circulation
                            , E, 0, 0);
44                      plotwake(pvi,2) = plotwake(pvi,2)
                            + v(1,3);
45
46                  end
47              end
48          end
49      end
50
51          % induced velocity due to the quarter
52          % chord vortices
```

```
53
54      for infing_wing = 1:2
55          for infing_side = 1:2
56              for infing_fil = 1:(a(infing_wing).nf/2-1)
57                  % infing_node is nt+1 for quarter
                        chord
58                  % vortex
59                  influencing_node_1 = a(infing_wing).s(
                        infing_side).f(infing_fil).n(nt
                        +1,:);
60                  influencing_node_2 = a(infing_wing).s(
                        infing_side).f(infing_fil+1).n((nt
                        +1),:);
61                  circulation = a(infing_wing).
                        gamma_delta;
62                  E = a(infing_wing).E;
63
64                      if (infing_side == 1)
65                          circulation = circulation*(-1)
                                *infing_fil;
66                      else
67                          circulation = circulation*
                                infing_fil;
68                      end
69                  v = vortex_induced_velocity(
                        influenced_node_1,
                        influenced_node_2,
                        influencing_node_1,
                        influencing_node_2, circulation, E,
                         0, 0);
70                  plotwake(pvi,2) = plotwake(pvi,2) + v
                        (1,3);
71
72              end
73          end
74          % for each wing, this section of code takes
                into
75          % account the vortex segment across the center
                of
76          % the wing from the inner most filment of on
                side
77          % to the inner most filament of the other side
                .
78
```

```
79          influencing_node_1 = a(infing_wing).s(1).f(
                infing_fil+1).n(nt+1,:);
80          influencing_node_2 = a(infing_wing).s(2).f(
                infing_fil+1).n((nt+1),:);
81          circulation = a(infing_wing).gamma_delta;
82          E = a(infing_wing).E;
83
84          circulation = circulation *(-1) * (infing_fil
                + 1);
85          v = vortex_induced_velocity(influenced_node_1,
                 influenced_node_2, influencing_node_1,
                influencing_node_2, circulation, E, 0, 0);
86          plotwake(pvi,2) = plotwake(pvi,2) + v(1,3);
87
88      end
89
90          % induced velocity due to fixed streamwise
                vortices on wing surface
91
92      for infing_wing = 1:2
93          for infing_side = 1:2
94              for infing_fil = 1:(a(infing_wing).nf/2-1)
95                  % infing_node is nt+1 for quarter
                        chord
96                  % vortex
97                  influencing_node_1 = a(infing_wing).s(
                        infing_side).f(infing_fil).n(i+1,:)
                        ;
98                  influencing_node_2 = a(infing_wing).s(
                        infing_side).f(infing_fil).n(nt
                        +1,:);
99                  circulation = a(infing_wing).
                        gamma_delta;
100                 E = a(infing_wing).E;
101
102                     if (infing_side == 1)
103                         circulation = circulation*(-1)
                                ;
104                     else
105
106                     end
107
108                 v = vortex_induced_velocity(
                        influenced_node_1,
```

```
                    influenced_node_2,
                    influencing_node_1,
                    influencing_node_2, circulation, E,
                     0, 0);
109                 plotwake(pvi,2) = plotwake(pvi,2) + v
                    (1,3);
110
111             end
112         end
113     end
114
115
116         % induced velocity due to the starting
117         % vortices and the free stream
118
119     for infing_wing = 1:2
120         for infing_side = 1:2
121             for infing_fil = 1:(a(infing_wing).nf/2-1)
122
123                 % infing_node is 1 for starting
                    vortex
124                 influencing_node_1 = a(infing_wing
                    ).s(infing_side).f(infing_fil).
                    n(1,:);
125                 influencing_node_2 = a(infing_wing
                    ).s(infing_side).f(infing_fil
                    +1).n((1),:);
126                 circulation = a(infing_wing).
                    gamma_delta;
127                 E = a(infing_wing).E;
128
129                 if (infing_side == 2)
130                     circulation = circulation*(-1)
                        ;
131                 else
132
133                 end
134
135                 v = vortex_induced_velocity(
                    influenced_node_1,
                    influenced_node_2,
                    influencing_node_1,
                    influencing_node_2, circulation
                    , E, 0, 0);
```

```
136                         plotwake(pvi,2) = plotwake(pvi,2)
                                + v(1,3);
137
138                 end
139             end
140         end
141
142     end
143
144 end
```