ANALYSIS OF UNSTEADY FLOWS WITH VORTEX BASED METHOD AND N-S SOLVER

BY PRUTHVI K JUJJAVARAPU

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Mechanical and Aerospace Engineering

Written under the direction of

DR. MITSUNORI DENDA

and approved by

_____

_____

_____

New Brunswick, New Jersey

January, 2016

ABSTRACT OF THE THESIS

ANALYSIS OF UNSTEADY FLOWS WITH VORTEX BASED METHOD AND N-S SOLVER

by Pruthvi K Jujjavarapu

Thesis Director: DR. MITSUNORI DENDA

Unsteady flows are very common in nature. Birds, insects, fish and other animals predominantly rely on unsteady phenomena for locomotion. In the past century the primary focus of aerodynamicists was on steady flows. Currently, there is a renewed interest in unsteady flows due to their applications in MAV's and energy extraction. The purpose of this thesis is to investigate unsteady flows with a traditional N-S solver and a new Vortex solver based on the Discreet Vortex Method(DVM). We found that the vortex method is very suitable to simulate flapping flight, especially for parametric studies before proceeding to a full fledged N-S solver. Our results show that the Vortex method is an order of magnitude faster than a N–S solver.

Acknowledgements

This thesis has been a very fruitful endeavour. Its the best learning experience I have had so far. I'm very thankful to my advisor Professor Mitsunori Denda for being a great mentor. He encouraged me to learn OpenFOAM. This was quite a challenge because I had to pick up a variety of skills like Linux, CFD etc. He was very supportive and gave me the freedom to explore new ideas. This thesis wouldn't have been possible without his guidance.

I would also like to thank the faculty who have taught me and made me a better engineer. A special thanks to my friend Ian for showing me the ropes to Linux and OpenFOAM. I thank my lab-mates and fellow researchers for engaging in those fantastic discussions and broadening my knowledge. I must say that the CFD online forums have been very helpful. This thesis wouldn't have been possible without the help of the various people I met there.

I thank Rutgers University for giving me the opportunity to pursue my masters and providing me with state-of-the-art facilities.

I am grateful towards my family and friends for supporting me in this endeavour.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Why unsteady flows?

The purpose of this thesis is to study unsteady flows using two different techniques and compare their performance. Before we delve into the details of unsteady flows it is important to understand why we are studying them. Unsteady flows occur everywhere and are usually undesirable. Structural engineers take extra measures to disrupt the vortex shedding behind tall chimneys which can otherwise destroy the structure. Flow separation is a highly undesirable phenomena for fixed wings and a lot of engineering effort is put into delaying stall. However, natures fliers have long figured a way to exploit the mechanisms of unsteady flight and fly efficiently. Until as recent as 1996, it was a mystery as to how insects can generate lift at very high angles of attack for which a fixed wing would stall. Therefore, it is a very wise idea to study insects for a better understanding of exploiting unsteady phenomena for good. In fact insects have provided a framework and acted as a strong catalyst for much of the research on unsteady flows. Owing to the above reasons its necessary to study insect flight and the framework on which much of the current literature is built upon.

At the end of this chapter there will be an overview of this thesis Section 1.6.

1.2 Brief introduction to insect flight

The reason behind the poor understanding of insect flight until recent times is due to the various challenges in observing and quantifying the kinematics of small, fast moving wings. For example the common fruit fly is 2-3 mm in length and has a wing beat frequency of 200 Hz [12]. The kinematics are vital because they are a pre-requisite for both

physical and numerical experiments. Recent improvements in high-speed videography have provided a wealth of kinematic data which made it possible to conduct realistic experiments. Directly measuring the forces on an insect wing is close to impossible. Flight forces have been measured on the body of tethered insects. This is not very effective because wing flap in tethering is not the same as free-flight and its not possible to separate the inertial forces from the aerodynamic forces.

1.3 Conventions and terminology

The terminology in flapping wing aerodynamics is derived from fixed wing aerodynamics.Wing span refers to the distance between the wing tips when they are stretched out, excluding the width of the thorax. Wing length refers to the base-to-tip length of each wing. Wing chord refers to any cut section of the wing generated by the intersection of a plane orthogonal to the wing planform. Aspect ratio is defined as the ratio between span to mean chord. Angle of attack (AOA) is the angle between the wing chord and the relative velocity vector of the undisturbed free stream flow. Effective AOA is defined as the angle between the wing chord and the relative velocity vector of the fluid in the immediate vicinity of the wing.

Flapping flight usually comprises of translational and rotational motions. The rotational motions come into play during stroke transition. Downstroke is defined as the dorsal-to-ventral motion and upstroke is the ventral-to-dorsal motion. When the wing undergoes a transition from downstroke-to-upstroke, the wing supinates rapidly, a rotation that causes the ventral surface of the wing to face the sky while the insect is flying horizontally. Pronation occurs at upstroke-to-downstroke transition and it causes the ventral surface of the wing to face the ground. By convention linear (or non-flapping) translation refers to airfoils translating linearly without any rotational motion. Flapping Translation is defined as an airfoil revolving about an axis (wing-abdomen joint in the case of an insect). Therefore by definition a 2-D wing is incapable of flapping translation. Infinite wings are abstractions of 2-D wings which do not cause any spanwise flow. Infinite wings are easier to analyze since they are computationally less demanding.

1.4 Prominent unsteady mechanisms

Of all the theories that have been put forward to explain the unusually high lift generated by insects only a few have been experimentally proven. Here we will briefly discuss about these mechanisms.

1.4.1 Delayed stall and leading edge vortex

Delayed stall refers to the fact that flow separation and subsequent stalling of the wing is delayed for a reasonable amount of time. This was first identified experimentally by Walker in 1931 [12]. Model aircraft wings experienced an increase in lift for a very short period of time during which the wing just exceeded angles of attack above steady-state-stall. This is explained by the formation of a leading edge vortex (LEV) which grows in size and remains attached to the wing. The LEV continues to grow in size until it can no longer remain attached and sheds into the wake. Immediately, vorticity is generated at the trailing edge and a trailing edge vortex (TEV) also sheds into the wake. This dynamic process is repeated continuously, generating the well known Karman vortex street. The formation of vorticity at the trailing edge disrupts the Kutta condition and impedes the ability of the wing to impart downward momentum to the wing, thereby reducing lift.

Insects however, do not suffer from vortex shedding due to a highly stable LEV. The exact structure of the LEV is dependent on the Reynolds Number. For high Reynolds numbers a steady span-wise flow is formed. This span-wise flow drains energy from the LEV and eventually joins with the tip vortex [8]. At low Reynold Numbers the LEV was found to be stable in the absence of a significant span-wise flow [5]. Irrespective of the reasons behind the stable LEV its effects are very prominent. It is the single most important reason for the high lift observed in insects.

1.4.2 Rotational forces or the Kramer effect

Rotational forces occur when an insect wing is pronating or supinating at stroke reversal. This effect is also called the Kramer effect after M.Kramer. Rotational forces

are observed as spikes in the lift and drag plots. These spikes are very significant and according to Dickinson et al. [7] they account for about 35% of the total lift produced throughout a stroke. This effect has been experimentally studied by them on a dynamically scaled model. They carefully measured the translation forces and subtracted it from the total force curve to confirm the origin of these peaks. It is believed that when a translating wing rotates, the Kutta condition is violated and the wing generates extra circulation to re-establish it. This extra circulation is believed to generate the extra force which can be either positive or negative depending on the timing of rotation. By studying the temporal effect of rotation on the aerodynamic forces they were able to confirm that rotational lift is a real phenomena.

1.4.3 Wing-Wake interaction or wake capture

Insect wings reciprocate and this raises the possibility of the wing intercepting its own wake. This is very obvious in the case of hovering when the insect is stationary and the wake is in the near vicinity of the insect. During stroke reversal both LEV and the TEV are shed into the wake. These shed vortices exert an influence on the nearby fluid which is proportional to the strength and relative position of the vortices. This phenomena aptly called wake capture, meaning the wing is trying to steal some of the energy in the wake. This was established by [7] using a dynamically scaled wing. This was confirmed by measuring forces on the wing after it was brought to rest. The forces lingered for several hundred milliseconds after the wing was stopped. This they reasoned is the result of the wake acting on the wing. The sign of these forces is once again highly dependent on the kinematics of the wing. It has also been proposed that insects utilize the effect of stroke timing on aerodynamic forces to change direction and execute complex aerial manoeuvres.

1.5 Contribution

The main contribution of this thesis is to perform a numerical evaluation of the vortex method and investigate how it compares to a N-S solver. To achieve this goal, numerous

simulations were conducted in OpenFOAM, an open source N-S solver to get familiar with its capabilities and limitations. A MATLAB implementation of the vortex method was used to simulate the same problem using the vortex method. A literature review was conducted to study the current state of computational methods for flapping flight. A paper by Liang et. al. [11] was chosen to be used as a reference to benchmark the simulations. A modified OpenFOAM solver was used to simulate the pitching and plunging motion of a flat plate. Parametric studies were conducted on the vortex method to study the influence of various parameters on the flow fields and the forces. An optimal set of parameters were discovered which consistently give solutions which are close to the N-S solver. Lastly, a very specific case Section 4.5.2 was chosen to prove that the vortex method is capable of predicting viscous forces. A framework was left behind for future researchers to use OpenFOAM for the purpose of pitching and plunging airfoils.

1.6 Overview of the thesis

Chapter 2 discusses about existing computational techniques and the need for simpler methods. It briefly talks about N-S solvers and Vortex based methods. The mathematical formulation of the Vortex method used in this thesis is discussed. Chapter 3 discusses about OpenFOAM, a N-S solver which has been used in this thesis. A very brief introduction is provided to the interested reader who may want to pursue OpenFOAM in the future. An OpenFOAM solver is benchmarked in this chapter. Chapter 4 compares the results of igVortex and OpenFOAM. The flow fields and aerodynamic forces of three cases are compared with each other. A comparison is made between the computational needs of igVortex and OpenFOAM. Chapter 5 concludes this thesis and discusses about future work.

Chapter 2

Existing computational techniques and the vortex method

CFD is a robust technique to study insect flight. It provides a wealth of information about flow structures and time varying forces which are not obtainable by experimental methods. In much of the current literature, CFD and dynamically scaled models went hand in hand to investigate and confirm the theories put forth for the underlying unsteady mechanisms. However, conventional Navier–Stokes (NS) Solvers are computationally expensive. Unsteady simulations require dynamic meshes which creates various challenges. The quality of the mesh deteriorates as the mesh deforms. This imposes a limit on the extent to which the mesh can deform. Various techniques like re-meshing and arbitrary mesh interface (AMI) exist to overcome this limitation but they further increase the computational expense. It is therefore necessary to have a computationally simple yet accurate solver to analyse unsteady flows.

2.1 Vortex based methods

Insects operate in a broad range of Reynolds numbers varying from 10 to 10^5 [12]. These Reynolds Numbers are high enough that they hint at the possibility of an inviscid approach. Even a Reynolds number of 10 means that the inertial forces are 10 times the viscous forces. Though viscous forces are negligible they strongly influence the aerodynamics by creating complex flow structures. Therefore an inviscid solver with the capability to produce elaborate flow structures is a possible solution.

Such solvers are called inviscid panel codes in unsteady literature. Various approaches have been made towards the implementation of vortex based methods [1,2,14]. All the implementations assume the fluid to be inviscid and the effects of viscosity embedded in the form of vortices and/or by the enforcement of the Kutta condition.

The common aspect of these methods is the use of Biot–Savart law to calculate the influence of the vortices at a certain location in the domain. These are mesh free methods unlike the Finite Volume Method (FVM) based solvers. This is very convenient because it requires less memory and the computation of the wing motion is simplified.

Though many attempts have been made, each has its own shortcomings. In this thesis we explore another version of the vortex method whose main distinction is that the Kutta condition is no longer enforced. Ansari et al. [1, 2] have attempted a vortex based method with the enforcement of the Kutta condition and have faced solver crashes during extreme wing motions. This was because of the creation of vortices with very high circulations during stroke reversals. The vortices shed after the creation of these high strength vortices were also of a large strength but opposite sign to enforce stagnation at the shedding region. This was also the case when the wing interacted with the wake. Considering that extreme wing motions and wing–wake interaction are the hall mark of insect flight this needs to be overcome for a successful solver.

2.2 Discreet vortex method

The vortex method used in this thesis is based on the Discreet Vortex Method (DVM) which was developed by Belotserkovsky [3, 4]. The wing is bound by discreet vortices which move along with the wing. At each time step the edge vortices are shed into the wake and each of the shed vortices influence each other and the wing. The forces on the wing can be calculated by integrating the unsteady Bernoulli equation or by using impulses and their derivatives introduced by Von Karman and Sears. The later approach is used here. It is computationally simpler at the expense of spatial resolution of forces on the wing. A MATLAB code was used to implement the vortex method used in this thesis. We have named the code as igVortex. The words igVortex and vortex method are used interchangeably in this thesis.

2.2.1 Assumptions

The vortex method in this thesis has a few simplifying assumptions.

1. The wing is thin and rigid
2. The fluid is assumed to be laminar, inviscid and incompressible.
3. The flow is irrotational everywhere except at the location of bound and wake vortices.
4. The shed vortices do not decay with time.
5. All vortices interact with each other at all times.
6. The smooth Biot–Savart law is used to avoid spurious velocities due to the singularity at the vortex core.
7. There is no spanwise flow i.e the flow is strictly two-dimensional.

2.2.2 Coordinate systems

Three coordinate systems were used to describe the local and global positions of the wings as shown in Figure 2.1 . The wing is represented by a solid line in the figure. Here we are looking at the sectional view of wing i.e the wing chord. The geometry of the wing is represented locally by the wing-fixed system $O-\xi\eta$ or globally by the space-fixed system $\tilde{O}-\tilde{\xi}\tilde{\eta}$. The wing is described by a set of (ξ,η) values, which remains constant. The global position is described by the corresponding set $(\tilde{\xi},\tilde{\eta})$, which varies with the motion of the wing. The wing undergoes two translational motions called lunge and heave and one rotational (pitch) motion. Lunge corresponds to the motion along the horizontal axis and heave corresponds to motion along the vertical axis in the space fixed system. The origin of the third coordinate system i.e the wing-translating system, $\hat{O}-\hat{\xi}\hat{\eta}$ is placed at the center of rotation of the wing. This origin, $\hat{O}$ can coincide with the origin of the wing fixed system, O or, in general, located at a distance a along the negative ξ axis, which defines the rotational offset. The axes of the wing-translating coordinate system, $\hat{\xi}$ and $\hat{\eta}$ are parallel to those of the wing-fixed system, $\tilde{\xi}$ and $\tilde{\eta}$, respectively. The wing-fixed system rotates and translates with the wing, the wing-translating system only translates while the space-fixed system is stationary.

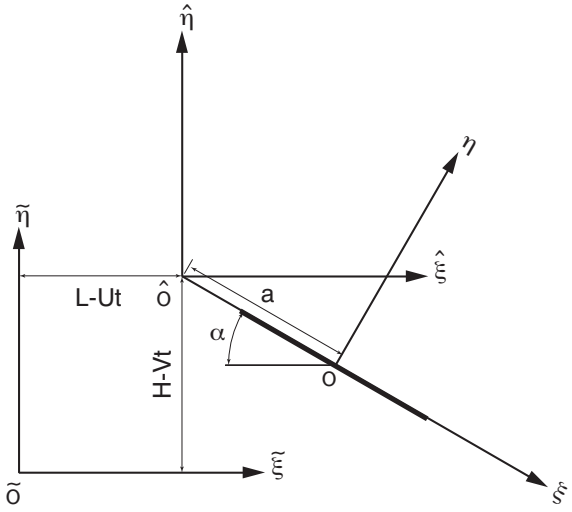


Figure 2.1: Space fixed ($\tilde{O} - \tilde{\xi}\tilde{\eta}$), wing-fixed ($O - \xi\eta$) and wing-translating ($\hat{O} - \hat{\xi}\hat{\eta}$) coordinate systems [6]

2.3 2-D Motion of flapping insect wing

2.3.1 Wing position

Insect wings have six degrees of freedom. Rotation about the three axes called the roll, pitch and yaw and three more translational motions. For insects with high aspect ratio wings, there is no significant spanwise flow. This justifies the two-dimensional approximations provided the right and left; wing pitch and roll are symmetric and the yaw is absent. In 2-D, the pitch is reproduced faithfully but the roll is recast as a translational motion which comprises the heave and lunge motions.

The rolling motion of a finite length insect wing in 3-D is described by the rotation around the body axis. In 2-D this motion is represented in the form of a translational motion of an infinitely long wing along the stroke plane. This is easy to visualize if one can imagine the wing rolling about the body axis which corresponds to the wing chord moving up and down when viewed from the side of the insect. Figure 1 in Appendix A shows the projection of the stroke plane with the two-dimensional plane as an straight inclined line(stroke line) with slope, β, measured positive counter-clockwise from the horizontal line. This angle is called the stroke angle. The translational motion is

decomposed into the horizontal (lunge), H, and vertical (heave), L, components. The wing rotates (pitching motion) about the wing span, denoted by α. The forward pitch, in the down-stroke direction is with $\alpha < 0$, is called pronation and the backward pitch, in the direction of the up-stroke with $\alpha > 0$, is called the supination.

We introduce a hypothetical 3D wing of finite span length l whose rolling is represented as a lunge and heave in the stroke plane. The rolling motion is defined by upper and lower stroke angles, ϕ_T and , ϕ_B, respectively undergone by this 3-D wing as shown in Figure 2.2. For simulations mimicking insect flight, these three values are taken from slow motion videos of flying insects. The time variation of the lunge and heave motions are expressed using sinusoidal functions.

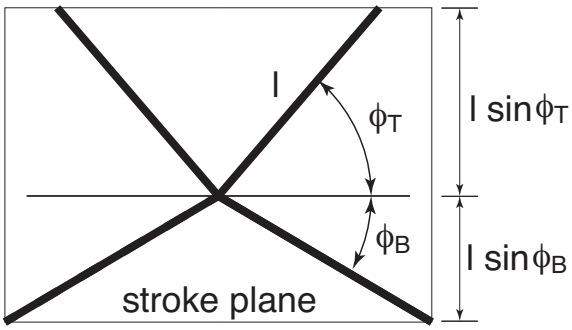


Figure 2.2: The stroke plane view of the right and left wings at extreme positions. Top (ϕ_T) and Bottom (ϕ_B) positions [6]

The pitch motion is assumed to take place at the extremes of each stroke as a sudden rotation. The pronation occurs at the top of the stroke by pitching the wing in the downward direction, while the supination is induced at the bottom of the stroke in the upward direction. However, pitch motions of some insects occur before reaching the top or bottom stroke point (advanced pitch) and others show the pitch after (delayed pitch). As instantaneous pitch is not physical and its a relatively smooth change. A smoothed step function is used to mimic the natural pitching motion of an insect.

A change of reference frame ensures that the the ambient air velocity, (U, V), is

incorporated by moving the wing itself in the direction opposite to the air velocity under zero ambient velocity. We assume the constant air velocity and the total translational motion of the wing is obtained by superposing the contributions from the flapping motion and the air velocity to give $(L - Ut, H - Vt)$, where t is the time.

All the wing position variables introduced are displayed in Figure 2.1. Here, $(L - Ut, H - Vt)$ gives the coordinates of the wing-translational system origin $\hat{O}$, α gives the slope of the ξ-axis of the wing-fixed system and a is the distance between the origins of the wing-fixed and wing-translating systems. The details of the wing motion are given in Appendix A.

2.3.2 Wing velocity

Figure 2.3 shows the translational velocity, $(\dot{L}-U, \dot{H}-V)$, of the wing-translating system origin and the superposed translational $(\dot{L} - U, \dot{H} - V)$ and rotational $(\dot{\alpha})$ velocities of the wing-fixed system origin. The center of rotation is located at the origin $\hat{O}$ of the wing-translational system. Given these wing velocity parameters, we can calculate the velocity of an arbitrary point $P = (\xi, \eta)$ on the wing, which, in the wing-translating system, is given by the coordinates $(\hat{\xi}, \hat{\eta})$. The linear velocity resulting from the rotation of point $\hat{P}$ around $\hat{O}$ is given by the cross product

$$\mathbf{r}_{\hat{O}\hat{P}} \times (-\dot{\alpha}\mathbf{k}), \tag{2.1}$$

where $\mathbf{k}$ is the out of plane unit vector.

In two-dimensions, it is convenient to use complex variables. Let $\hat{\zeta} = \hat{\xi} + i\hat{\eta}$ and $\zeta = \xi + i\eta$ to describe complex valued positions of the wing in the wing-translating and wing-fixed systems, respectively. They are related by

$$\hat{\zeta} = (a + \zeta)e^{-i\alpha}. \tag{2.2}$$

The complex valued velocity of 2.2 is given by

$$[V_{\hat{\xi}} + iV_{\hat{\eta}}]_{rot} = \dot{\alpha}\hat{\zeta}e^{-i\pi/2} = -i\dot{\alpha}\hat{\zeta}, \tag{2.3}$$

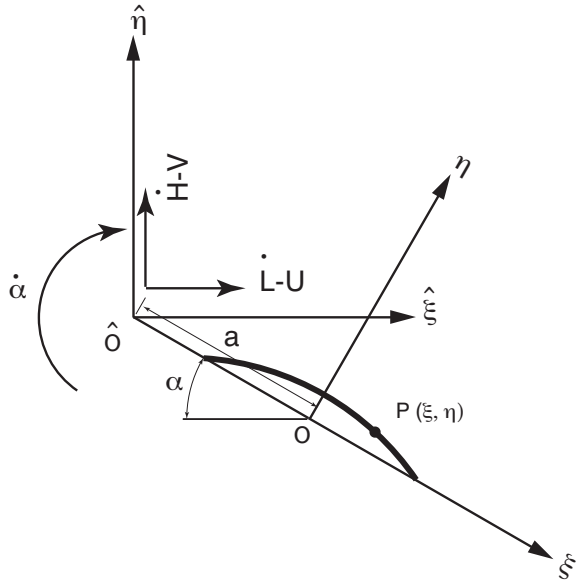


Figure 2.3: Translational and rotational velocities of the wing-translating and wing-fixed coordinate systems [6]

where $e^{-i\pi/2}$ rotates the complex position vector $\hat{\zeta}$ by 90 degrees in the clockwise direction. Combined with the complex valued velocity of the origin $\hat{O}$,

$$[V_{\hat{\xi}} + iV_{\hat{\eta}}]_{trans} = \dot{L} - U + i(\dot{H} - V), \tag{2.4}$$

we get the total velocity of the point $\hat{P}$ (or equivalently P) on the wing

$$V_{\tilde{\zeta}} = V_{\hat{\xi}} + iV_{\hat{\eta}} = (\dot{L} - U) + i(\dot{H} - V) - i\dot{\alpha}\hat{\zeta} = (\dot{L} - U) + i(\dot{H} - V) - i\dot{\alpha}(a + \zeta)e^{-i\alpha}. \tag{2.5}$$

Since the wing-translating and space-fixed coordinate axes are parallel, this gives the space-fixed expression of the wing velocity

$$V_{\tilde{\xi}} = V_{\hat{\xi}} = \dot{L} - U + \dot{\alpha}(\eta\cos(\alpha) - (a + \xi)\sin(\alpha)), \tag{2.6}$$

$$V_{\tilde{\eta}} = V_{\hat{\eta}} = \dot{H} - V + \dot{\alpha}((a + \eta)cos(\alpha) - \eta\sin(\alpha)). \tag{2.7}$$

in terms of the wing-fixed co-ordinates.

The camber of the wing is specified by the function $\eta = f(\xi)$ in the wing-fixed system. The complex unit vector of the wing is given by

$$n = n_{\tilde{\xi}} + i n_{\tilde{\eta}} = \frac{-df(\xi)/d\xi + i}{\sqrt{1 + (df(\xi)/d\xi)^2}} \tag{2.8}$$

in the wing-fixed system and by

$$\tilde{n} = n_{\tilde{\xi}} + i n_{\tilde{\eta}} = \hat{n} e^{-i\alpha} = \frac{-df(\xi)/d\xi + i}{\sqrt{1 + (df(\xi)/d\xi)^2}} e^{-i\alpha} \tag{2.9}$$

in the space-fixed system. The normal velocity of the wing is given, from 2.6, 2.7 and 2.9, by

$$V^{\tilde{n}} = \Re(\bar{V}_{\tilde{\zeta}} \tilde{n}) = V_{\tilde{\xi}} n_{\tilde{\xi}} + V_{\tilde{\eta}} n_{\tilde{\eta}} \tag{2.10}$$

where $\Re$ is the real part of a complex variable.

2.4 Vortex equations

2.4.1 Single vortex

In the vortex method, the vortex is the fundamental unit. The complex potential function for a line vortex, with the circulation Γ and location $\tilde{\zeta}_0 = \tilde{\xi}_0 + i\tilde{\eta}_0$, is given by

$$\omega(\tilde{\zeta}) = -\frac{i\Gamma}{2\pi} \log(\tilde{\zeta} - \tilde{\zeta}_0), \tag{2.11}$$

where $\tilde{\zeta} = \tilde{\xi} + i\tilde{\eta}$ and $\tilde{\zeta}_0$ are complex position vectors in the space-fixed system. The circulation is positive counter-clockwise. The conjugate of the complex velocity induce by this vortex at $\tilde{\zeta}$ is given by

$$\overline{v(\tilde{\zeta})} = \frac{d\omega}{d\tilde{\zeta}} = -\frac{i\Gamma}{2\pi} \frac{1}{\tilde{\zeta} - \tilde{\zeta}_0} \tag{2.12}$$

where $v(\tilde{\zeta}) = v_{\tilde{\xi}} + i v_{\tilde{n}}$ and the overbar $\overline{()}$ indicates the complex conjugate.

When the distance between the observation point and the point vortex get closer, the velocity at the observation point increases. This trend becomes prevalent when the number of wake vortices increases. In order to avoid the excessive magnitude of velocity we have used a smooth Biot-Savart law discussed in the Appendix A.4.

2.4.2 Discretization of the wing

The wing is discretized by a given number of vortices m each with circulation Γ_j at locations $\tilde{\zeta_{0j}}(j = 1, 2, 3, \ldots m)$ and $m - 1$ collocation points $\tilde{\zeta}_i(i = 1, 2, \ldots, m - 1)$. Vortices are placed at the LE, TE and in-between. The spacing of the vortex points could be equidistant at the middle and gradually narrowed towards the edges. The collocation points are placed at the midpoints of the vortex points.

2.4.3 Influence coefficients

The complex conjugate velocity at the wing collocation point $\tilde{\zeta}_i$ at $\tilde{\zeta}_{0j}$ is given by

$$\bar{v}_{ij} = -\frac{i\Gamma_j}{2\pi}\frac{1}{\tilde{\zeta}_i - \tilde{\zeta}_{0j}} \tag{2.13}$$

The normal component of this velocity at the collocation point is given, from 2.10, by

$$v_{ij}^{\tilde{n}} = \Re(\bar{v}_{ij}\tilde{n}_i) = \frac{\Gamma_j}{2\pi}\Im(\frac{\tilde{n}_i}{\tilde{\zeta}_i - \tilde{\zeta}_{0j}}) \equiv V_{ij}^{\tilde{n}}\Gamma_j \tag{2.14}$$

where $\tilde{n_i}$ is given by Eq.2.9 at the collocation point, $\Im$ is the imaginary part of a complex variable, and

$$V_{ij}^{\tilde{n}} = \frac{1}{2\pi}\Im(\frac{\tilde{n}_i}{\tilde{\zeta}_i - \tilde{\zeta}_{0j}}), \tag{2.15}$$

is the influence coefficient.

2.5 System of equations for discrete vortices on the wing

2.5.1 Contribution from the bound vortices on the wing

Add contributions from the entire discreet bound vortices on the wing to get the normal velocity component at the collocation point $\tilde{\zeta}_i$,

$$v_i^{\tilde{n}} = \sum_{j=1}^{m} V_{ij}^{\tilde{n}}\Gamma_j \tag{2.16}$$

where $V_{ij}^{\tilde{n}}\Gamma_j$ is given by 2.15. The non-penetration condition requires that this normal velocity must be equal to the normal velocity of the wing, $V_i^{\tilde{n}}$, at each collocation point,

$$\sum_{j=1}^{m} V_{ij}^{\tilde{n}}\Gamma_j = V_i^{\tilde{n}}, \tag{2.17}$$

for collocation points $i = 1, 2, \ldots, m-1$. The conservation of vorticity provides the additional equation required to match the m unknowns, Γ_i, is given by the conservation of vortices,

$$\sum_{j=1}^{m} \Gamma_j = 0. \tag{2.18}$$

2.5.2 Contribution from the wake vortices

At each time step, a pair of vortices, from the LE ($j = 1$) and TE ($j = m$) edges, are shed from the wing such that during the p-th time step we have $2(p-1)$ wake vortices,

$$\Gamma_1^{[1]}, \Gamma_m^{[1]}, \Gamma_1^{[2]}, \Gamma_m^{[2]}, \ldots, \Gamma_1^{[p-1]}, \Gamma_m^{[p-1]},$$

located at,

$${}^{[p]}\tilde{\zeta}_1^{[1]}, {}^{[p]}\tilde{\zeta}_m^{[1]}, {}^{[p]}\tilde{\zeta}_1^{[2]}, {}^{[p]}\tilde{\zeta}_m^{[2]}, \ldots, {}^{[p]}\tilde{\zeta}_1^{[p-1]}, {}^{[p]}\tilde{\zeta}_m^{[p-1]},$$

where the pre-superscript, ${}^{[p]}()$ indicates the current step $[p]$ and the post-superscript $()^{[k]}$ where $((k = 1, 2, \ldots, p-1))$ indicates the originating time step. Notice that Γ's do not have the pre-superscript since their values remain constant once the vortices are shed into the flow.

The complex conjugate at the wing collocation point $\tilde{\zeta}_i$ due to the wake vortex pair $\Gamma_1^{[k]}$ and $\Gamma_m^{[k]}$ is given by

$${}^{[p]}\bar{v}_{i[k]}^{\tilde{n}} = -\frac{i}{2\pi}\left(\frac{\Gamma_1^{[k]}}{\tilde{\zeta}_i - {}^{[p]}\tilde{\zeta}_1^{[k]}} + \frac{\Gamma_m^{[k]}}{\tilde{\zeta}_i - {}^{[p]}\tilde{\zeta}_m^{[k]}}\right) \tag{2.19}$$

for $k = 1, 2, \ldots, p-1$.

The normal component of this velocity at the collocation point $\tilde{\zeta_i}$ is given, from Eq:2.10 given by

$$ {}^{[p]}\bar{v}^{\tilde{n}}_{i[k]} = \Re({}^{[p]}\bar{v}_{i[k]}\tilde{n_i}) = \frac{1}{2\pi}(\Gamma_1^{[k]}\Im(\frac{\tilde{n_i}}{\tilde{\zeta_i} - {}^{[p]}\tilde{\zeta}_1^{[k]}}) + \Gamma_m^{[k]}\Im(\frac{\tilde{n_i}}{\tilde{\zeta_i} - {}^{[p]}\tilde{\zeta}_m^{[k]}})) \equiv {}^{[p]}W^1_{i[k]}\Gamma_1^{[k]} + {}^{[p]}W^m_{i[k]}\Gamma_m^{[k]} \tag{2.20} $$

where

$$ {}^{[p]}W^1_{i[k]} = \frac{1}{2\pi}\Im(\frac{\tilde{n_i}}{\tilde{\zeta_i} - {}^{[p]}\tilde{\zeta}_1^{[k]}}), {}^{[p]}W^m_{i[k]} = \frac{1}{2\pi}\Im(\frac{\tilde{n_i}}{\tilde{\zeta_i} - {}^{[p]}\tilde{\zeta}_m^{[k]}}), \tag{2.21} $$

are the influence coefficients. The total normal component of this velocity is obtained by adding all contributions from $2(p-1)$ wake vortices to get

$$ {}^{[p]}\bar{v}^{\tilde{n}}_i = \sum_{k=1}^{p-1}({}^{[p]}W^1_{i[k]}\Gamma_1^{[k]} + {}^{[p]}W^m_{i[k]}\Gamma_m^{[k]}), \tag{2.22} $$

Eq.2.17 is now modified to give

$$ \sum_{j=1}^{m} V^{\tilde{n}}_{ij}\Gamma_j + {}^{[p]}v^{\tilde{n}}_i = V^{\tilde{n}}_i, \tag{2.23} $$

Notice that for the first step the wake is absent and ${}^{[1]}v^{\tilde{n}}_i = 0$.

2.6 Space and time resolution

The density of the bound vortices m on the wing determines the spacial resolution of the problem. Since the non-penetration condition is enforced only at the collocation points, a higher number of bound vortices can be used to get a better approximation of the non-penetration condition.

Two time increments are defined, one is based on the heave and the other on the pitch. The minimum of the two is chosen as the final value for every simulation. Smaller time increments will resolve finer features of the vortex movement. We choose the time increment to be related to the spacial resolution. The non-dimensional time-increment based on the heave is given by

$$\Delta t_d = \frac{c}{d}\frac{1}{m-1} \tag{2.24}$$

where c and d are the chord and stroke lengths, respectively. The physical interpretation of this relation is that for each time increment Δt_d, the wing travels in the vertical direction by an amount $d * \Delta t_d$ which is equal to one bound vortex gap, given by $c/(m-1)$.

In addition, it must be made sure that the pitching motion is also well resolved in the case when the pitching is much faster than the heave. The non-dimensional duration of pitch is given, from A.10

$$p = \frac{4}{\Delta t_p},$$

The time increment Δt has to be much smaller than Δt_p,

$$\Delta t_p = r_p \Delta t_p \tag{2.25}$$

to capture the pitching motion adequately. The value of r_p has been chosen to be 0.1. This means that there will be 10 snapshots of the pitching motion. This time increment is called as the pitch time increment. The smaller of 2.24 and 2.25 is chosen as the time increment for the whole simulation.

2.7 Time marching procedure

The flow is solved as an initial value problem with a time–marching algorithm. The basic steps will be described below.

Time t = 0

1. Discretize the wing. Specify the wing position and velocity.

2. The strength of each of the bound vortices is determined by the strict enforcement of the non–penetration condition at the surface of the wing, while also satisfying the Kelvins circulation theorem. This results in a set of $m + 1$ simultaneous algebraic equations.

- Non-penetration condition

$$\sum_{j=1}^{m} V_{ij}^{\tilde{n}} \Gamma_j = V_i^{\tilde{n}}, \tag{2.26}$$

where the L.H.S is the summation of the normal velocity induced at a collocation point i by the vortex j, $V_i^{\tilde{n}}$ is the normal velocity of the wing at collocation point i.

- Kelvins circulation theorem

$$\sum_{j=1}^{m} V_{ij}^{\tilde{n}} \Gamma_j = V_i^{\tilde{n}}, \tag{2.27}$$

3. The induced velocity of the bound and wake vortices, $v_1^{[1]}$ and $v_m^{[1]}$ at the LE, $\tilde{\zeta}_1^{[1]}$ and TE, $\tilde{\zeta}_m^{[1]}$ are calculated. Notice that there are no wake vortices yet so their contribution will be zero for this time–step.

4. Shed the bound vortices, $\Gamma_1^{[1]}$ and $\Gamma_m^{[1]}$ at the LE and TE as determined by the induced velocities using the forward Euler method, as follows

$$^{[2]}\tilde{\zeta}_1^{[1]} = \tilde{\zeta}_1^{[1]} + v_1^{[1]} \Delta t \qquad\qquad ^{[2]}\zeta_m^{[1]} = \tilde{\zeta}_m^{[1]} + v_m^{[1]} \Delta t$$

Time t = Δt.

1. Calculate the wing position and velocity.

2. The induced velocity at each collocation point, $\tilde{\zeta}_j^2$ is calculated by summing the contributions of the bound and wake vortices. Strengths of bound vortices $\Gamma_j^{[2]}$ are once again calculated by enforcing the non–penetration condition. Total circulation is conserved i.e the sum of the circulations of the bound and shed vortices.

- Non-penetration condition

$$\sum_{j=1}^{m} V_{ij}^{\tilde{n}} \Gamma_j^{[2]} +^{[1]} v_i^{\tilde{n}} = V_i^{\tilde{n}}, \tag{2.28}$$

- Kelvins circulation theorem

$$\sum_{j=1}^{m} \Gamma_j^{[2]} + \Gamma_1^{[1]} + \Gamma_m^{[1]} = 0. \tag{2.29}$$

3. The induced velocity of the bound and wake vortices, $v_1^{[2]}$ and $v_m^{[2]}$ at the LE, $\tilde{\zeta}_1^{[2]}$ and TE, $\tilde{\zeta}_m^{[2]}$ are calculated.

4. Shed the bound vortices, $\Gamma_1^{[2]}$ and $\Gamma_m^{[m]}$ as determined by the induced velocities using the forward Euler method, as follows

$$^{[3]}\tilde{\zeta}_1^{[2]} = \tilde{\zeta}_1^{[2]} + v_1^{[2]}\Delta t \qquad ^{[3]}\tilde{\zeta}_m^{[2]} = \tilde{\zeta}_m^{[2]} + v_m^{[2]}\Delta t$$

5. Convect the wake vortices, $\Gamma_1^{[1]}$ and $\Gamma_m^{[1]}$ from the previous time-step by calculating the induced velocities, $^{[2]}v_1^{[1]}$ and $^{[2]}v_m^{[1]}$ from every other vortex and using the forward Euler method as follows.

$$^{[3]}\tilde{\zeta}_1^{[1]} = \tilde{\zeta}_1^{[1]} +^{[2]} v_1^{[1]}\Delta t \qquad ^{[3]}\tilde{\zeta}_m^{[1]} = \tilde{\zeta}_m^{[1]} +^{[2]} v_m^{[1]}\Delta t$$

Repeat the process until the end of the time-marching procedure.

2.8 Forces and Moments

2.8.1 Method of Impulses

From Kelvins circulation theorem it is known that the magnitude of vorticity remains zero if the problem started with zero vorticity. In two dimensions each vortex is a line vortex with an infinite length in the out-of-plane direction. Since vortex tubes can only end on a surface we need to amend this vortex with another vortex having the same magnitude of circulation but in the opposite sense. These two vortices can be joined at infinity to form a closed vortex loop in the extended three-dimensional space.

During the time-stepping procedure there will me numerous wake vortices and it may not be possible to find out the exact partner of a given vortex that has the same magnitude of circulation as the original with the opposite sign. However, as long as the sum of the circulation of all vortices is zero, it is always possible to break down the given distribution into an alternative, but equivalent, distribution of vortices consisting of pairs of vortices each with the same magnitude but opposite signs.

A pair of such line vortices are considered with circulation Γ and $-\Gamma$ (located at

$\tilde{\zeta}_1$ and $\tilde{\zeta}_2$, respectively). The convention is chosen such that the sign of the counter-clockwise circulation is positive. The goal is to calculate the impulses induced by this pair onto the air in terms of the velocity potential function (Lamb, 1932). If a single line vortex is considered the problem becomes unbounded, since the air extends to infinity. A pair of vortices as described above will eliminate this problem. In the derivation we have used the complex variable potential 2.11 function for the line vortex. The complex variable linear impact vector for the pair is given by

$$\tilde{\mathcal{I}} = -i\rho\Gamma(\tilde{\zeta}_1 - \tilde{\zeta}_2), \tag{2.30}$$

where ρ is the density of air. The real value of the angular impulse is given by

$$\tilde{\mathcal{I}}_A = -\frac{1}{2}\rho\Gamma(|\tilde{\zeta}_1|^2 - |\tilde{\zeta}_2|^2), \tag{2.31}$$

In the implementation, the impulses are assigned for the individual vortex with the linear and angular impulses $-i\rho\Gamma\tilde{\zeta}$ and $-i\rho\Gamma|\tilde{\zeta}|^2$, where Γ is a signed circulation with positive counter-clockwise. The time derivatives of the linear and angular impulses will provide the force and moment exerted by the vortex onto the air. It is important to recognize that the force and moment acting on the wing are obtained by reversing the signs of these obtained for the air mass.

2.8.2 Wing-Translating system for force calculation

In implementation, it is convenient to calculate the angular momentum about the origin of the wing-translating system which is where the wing is attached to its body rather than about the space-fixed system. The challenge is that its origin is not stationary and the formula for the angular momentum obtained by replacing the complex coordinate $\tilde{\zeta}$ with $\hat{\zeta}$ does not represent the real angular impulse. We introduce another space-fixed system that has the same origin as the wing-translating system and calculate the angular impulse in this coordinate system. This system needs to be updated continuously to reflect the fact that the new space-fixed system is not stationary and moves at each time step. However, at each time step, the new space-fixed system is the legitimate system

for the calculation of the impulses.

The transformation between the space-fixed system $\tilde{O}-\tilde{\xi}\tilde{\eta}$ and the wing-translating system $\hat{O}-\hat{\xi}\hat{\eta}$ is given by

$$\tilde{\zeta} = r + \hat{\xi}, \tag{2.32}$$

where $r = L - U_\infty t + i(H - V_\infty t)$. Substitute this relation into Eq. 2.30 and 2.31 and simplify to get

$$\tilde{\mathcal{I}} = \hat{\mathcal{I}}, \qquad \tilde{\mathcal{I}}_A = \hat{\mathcal{I}}_A + \Im(\bar{r}\hat{\mathcal{I}}), \tag{2.33}$$

where

$$\hat{\mathcal{I}} = -i\rho\Gamma\hat{\zeta}, \qquad \hat{\mathcal{I}}_A = -\frac{1}{2}\rho\Gamma|\hat{\zeta}|^2. \tag{2.34}$$

Now take the time derivative in Eq.2.33 to get the force and moment,

$$\tilde{\mathcal{F}} = \dot{\tilde{\mathcal{I}}} = \dot{\hat{\mathcal{I}}}, \qquad \tilde{\mathcal{M}} = \dot{\tilde{\mathcal{I}}}_A = \dot{\hat{\mathcal{I}}}_A + \Im(\dot{\bar{r}}\hat{\mathcal{I}}) + \Im(\bar{r}\dot{\hat{\mathcal{I}}}) \tag{2.35}$$

Now we switch from the original space-fixed system to the new space-fixed system that is placed on top of the wing-translating system, for which $r = 0$, giving the updated relations,

$$\tilde{\mathcal{F}} = \dot{\tilde{\mathcal{I}}} = \dot{\hat{\mathcal{I}}}, \qquad \tilde{\mathcal{M}} = \dot{\tilde{\mathcal{I}}}_A = \dot{\hat{\mathcal{I}}}_A + \Im(\bar{r}\dot{\hat{\mathcal{I}}}) \tag{2.36}$$

where $\dot{\bar{r}} = \dot{L} - i\dot{H}$ is the conjugate complex velocity of the co-ordinate origin $\hat{O}$ of the wing-translating system due to the genuine wing motion; note that the ambient fluid velocity, which contributes as the relative velocity of the co-ordinate origin, should not be included here. The force and moment acting along the wing is obtained by reversing the signs. The force and moment in 2-D are calculated per unit depth in the out-of-plane direction.

2.8.3 Flow visualization

An advantage of vortex based methods is that the flow field is automatically generated. The position and velocity of the wake vortices is a true representation of the flow field. The wake vortices behave like smoke in a wind tunnel. From a computational perspective the flow field is only calculated at the locations of interest i.e the vicinity of the wing and the wake. This is one of the primary reasons why vortex methods are faster than a Navier–Stokes based solver.

Chapter 3

OpenFOAM a Navier–Stokes based solver

The inviscid panel code, igVortex, discussed in this thesis has been validated using OpenFOAM [13]. This chapter will briefly discuss about OpenFOAM and how it was used to simulate flapping problems.

3.1 Brief talk about OpenFOAM

OpenFOAM (Open Field Operation and Manipulation) is an open source CFD toolbox written in C++. It is capable of simulating a variety of problems like aerodynamics, solid mechanics, heat transfer, electromagnetics, financial problems etc. OpenFOAM has a very steep learning curve due to its lack of an integrated GUI and poor documentation. OpenFOAM is typically run from the Linux terminal i.e command line interface. This puts additional stress on a beginner because of the variety of the skills to be learnt. To become very proficient in OpenFOAM requires the mastery of fluid dynamics, scientific computing, software development and the Linux shell.

OpenFOAM can be modified relatively easily to create customized solvers and add additional capabilities. The code is open for viewing and editing unlike closed source commercial packages. It is free to use on any number of machines. This reduces licensing costs which become prohibitive in the case of commercial codes.

3.1.1 Case structure

A typical case file in OpenFOAM contains three dictionaries named "**0**", "**system**" and "**constant**". The "**0**" folder contains all the data regarding the initial condition. The "**constant**" folder contains the mesh, fluid properties and other parameters. The "**system**" folder contains all the data regarding the solver like numerical schemes, time

step calculation, residual data etc. All the data is stored in text files which can be edited in any text editor. Once a simulation is run, the data is stored in folders named after the time at which the data was saved.

Lets take a look at a **Temperature** boundary condition file called **T** in the "**0**" folder.

```
1  dimensions      [0 0 0 1 0 0 0];
2  internalField   uniform 0;
3  boundaryField
4  {
5      inlet
6      {
7          type            fixedValue;
8          value           uniform 1;
9      }
10     outlet
11     {
12         type            zeroGradient;
13     }
14     upperWall
15     {
16         type            zeroGradient;
17     }
18     lowerWall
19     {
20         type            zeroGradient;
21     }
22     frontAndBack
23     {
24         type            empty;
25     }
26 }
```

The dimensions entry is used to define the dimensions of the scalar field. OpenFOAM follows the following format:

Entry No.	Property	SI unit
1	Mass	kilogram (kg)
2	Length	metre (m)
3	Time	second (s)
4	Temperature	Kelvin (K)
5	Quantity	mole(mol)
6	Current	ampere(A)
7	Luminous intensity	candela(cd)

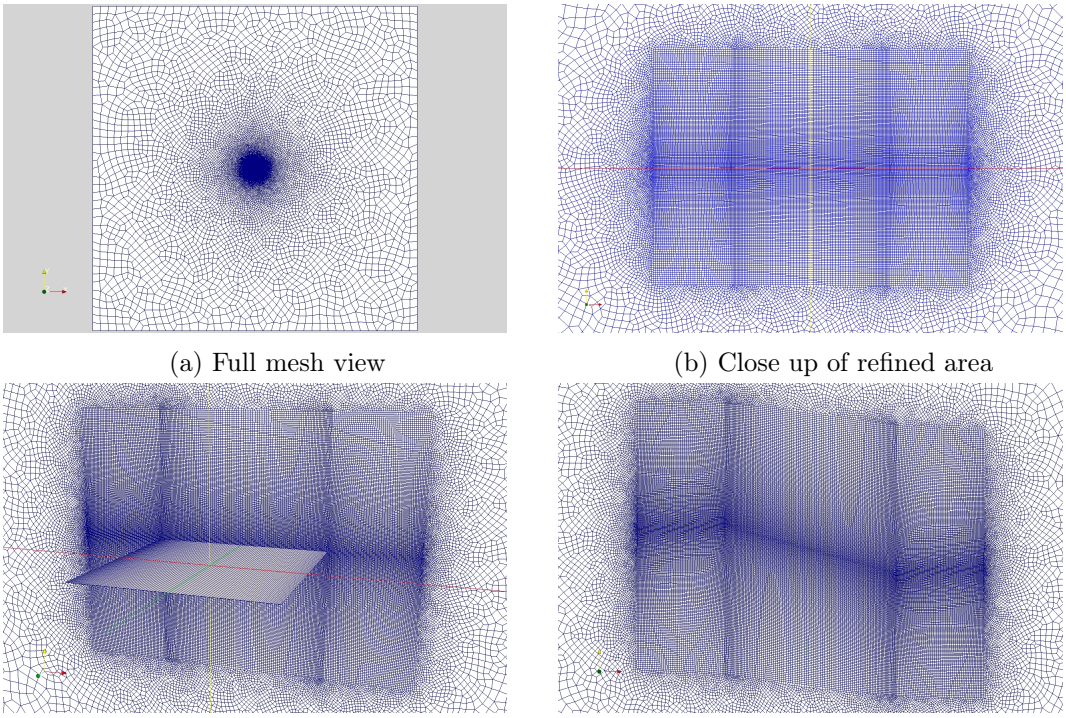
The entry internalField is used to define the initial condition of the domain. The keyword uniform signifies that the temperature is uniform throughout the domain. The entry boundary field takes in the boundary conditions at the patches defined during meshing. fixedValue is a Dirichlet boundary condition. zeroGradient is a Neumann boundary condition which sets the gradient to the normal to zero. Empty boundary condition is used to tell the software that the problem is in 2-D. A complete discussion of the boundary conditions in OpenFOAM is beyond the scope of this thesis.

3.1.2 Flapping Aerodynamics in OpenFOAM

OpenFOAM has a solver called ”pimpleDyMFoam" which can solve transient problems with a dynamic mesh. Pimple stands for pisco + simple solver. DyM stands for the dynamic mesh solving capabilities. An attempt was made to use an Arbitrary Meshing Interface in the wing fixed co-ordinate system with little success. Therefore we had to use a vertex based mesh motion solver. OpenFOAM doesn't support both pitching and plunging motion out of the box so two mesh motion solvers, namely, oscillatingDisplacement and angularOscillatingDisplacement, had to be combined to achieve this. The former is for harmonic translating motion and the latter is for harmonic rotating motion. This was first implemented by Hassan Kassem [10] and he shared it with me when we were discussing about mesh motion capabilities of OpenFOAM. The code will not be included here for brevity and the reader is encouraged to visit the Github repository included in the references.

3.1.3 Mesh generation

OpenFOAM's native meshers blockMesh and snappyHexMesh were used initially and later on Gmsh [9] was adopted because of its simplicity and the capability to generate hybrid meshes. High quality hybrid meshes were generated very quickly with gmsh while snappyHexMesh use to take a lot of time to generate a similar quality mesh. However, gmsh is not capable of meshing very complex geometries and snappyHexMesh is the only way. SnappyHexMesh requires a hexahedral background mesh which must be larger than



(a) Full mesh view

(b) Close up of refined area

(c) Flat-plate extruded

(d) Mesh Deformation

Figure 3.1: Infinitely thin flat-plate meshed using gmsh and the createBaffle utility

the geometry that is being meshed. It also needs an STL file of the geometry. Very complex geometries like propellers have been meshed using snappyHexMesh. Gmsh is a scripting language which is very useful for meshing simple geometries. It can generate a mesh with either triangles or quadrilaterals.

Since the MATLAB implementation of the vortex method, igVortex, can only simulate infinitely thin flat plates, the same was used for comparison with OpenFOAM. OpenFOAM provides a utility called createBaffles which splits an internal face into two

different faces. This is very convenient to simulate infinitely thin flat plates. The mesh in the figure 3.1 was created using gmsh and visualized using paraview. Unlike a typical mesh there is no cavity in the domain. The wing is just a collection of internal faces that have been split into two faces and defined as walls. Due to mesh quality issues, the extent of pitching is restrained to 20 degrees. Remeshing is not supported by OpenFOAM and a good amount of work needs to be put in to automate the process. The author did not have the time to implement this.

3.1.4 Post-processing

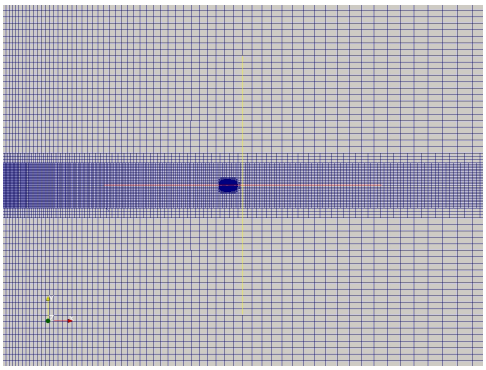
This section will be concluded with one last note about post-processing in OpenFOAM. Post-processing is done by Paraview, also an opensource visualization tool. It is a very advanced tool with the capability to handle large data sets and various other features. Third party post-processing tools can also be used like Ensight, FieldView and Fluent. OpenFOAM is a very good open-source alternative to commercial packages. The interested reader is encouraged to pursue it. Bugs are not un-common in OpenFOAM and the maintenance team does a very good job taking care of them. Most bugs are fixed within a day or two of being reported. This I can vouch from personal experience.

3.2 Solver validation

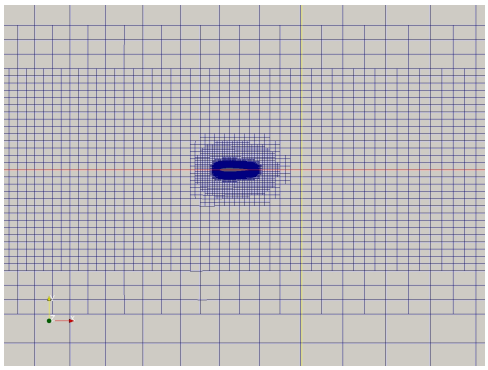
We validated OpenFOAM's solver, pimpleDyMFoam against the results obtained by Liang et al. They have simulated a simple plunging motion using higher-order spectral difference methods and have obtained very accurate results. OpenFOAM is able to capture the flow accurately and the force coefficients 3.2 to a satisfactory level. Mesh independence studies were performed with a mesh of double the elements and there was no significant improvement in the results.

This is a case of pure plunge with $\omega = 1.15$ and plunge amplitude, $h = 0.08c$. The geometry is a NACA0012 with a sharp trailing edge. There was an under prediction of 13% of the force coefficients by OpenFOAM. The discrepancy in results is due to the differences in discretization and mesh quality. However the time-varying trend was

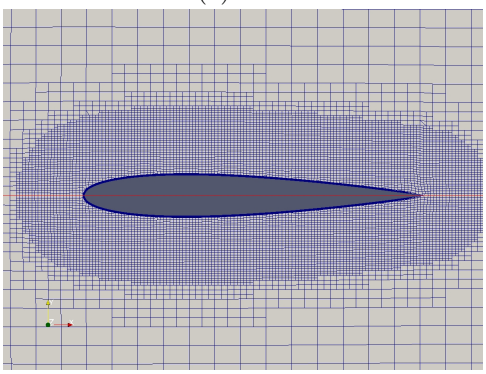
captured very satisfactorily. The mesh 3.2 was generated using OpenFOAM's native meshing tools blockMesh and snappyHexmesh.



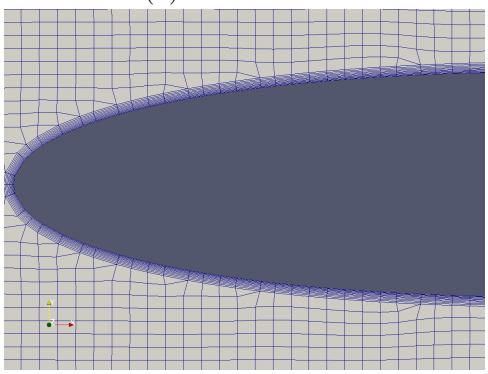
(a) Full mesh view



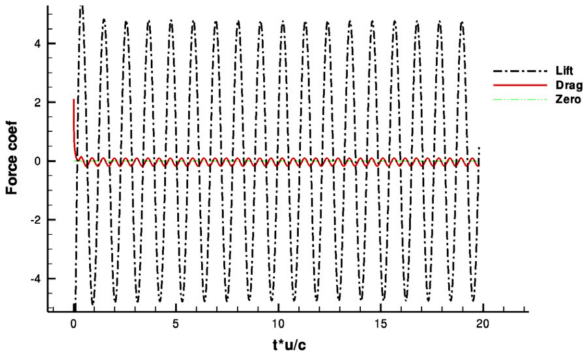
(b) Initial Refinement levels



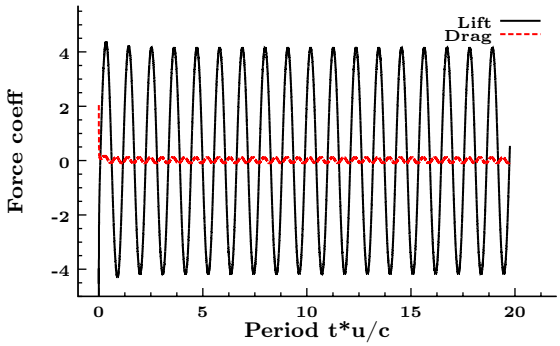
(c) Final refinement levels



(d) Close up of boundary layer



(e) Liang et al. [11]



(f) OpenFOAM

Figure 3.2: Mesh and force coefficients comparison

Chapter 4

Numerical Performance

4.1 Effect of spatial and temporal resolution

The spatial resolution is determined by the number of bound vortices, m. Three or more bound vortices, one at the LE, one at the TE and the others in between are used to discretize the wing. The time increment is determined by computing the minimum of two values. The first value is determined in terms of m and the stroke line length d. This time increment depends on the lunge motion and is denoted by Δt_d. The second value, Δt_p is determined by the pitch speed, p. The smaller of the two is used for the actual time increment. The time increment is non-dimensionalized by half a flapping period. The reason for choosing two candidates is to make sure that the flow is well resolved. For flapping with very high pitch speeds the second candidate is smaller. For flapping motions in which the plunge dominates, the first candidate is smaller.

Figure 4.1 shows the variation of lift force for a single period for various values of m in a combined flapping (heave and lunge) and rotation (pitching). The input parameters correspond to the flight of a crane fly, obtained from studying a slow motion video. The values are listed in a Table 4.1 with the stroke plane angle, $\beta = 10 degrees$. The non-dimesnsionalized time increment determined by Eq.2.24, using the parameters from Table 4.1, is $\Delta t_d = 0.267/(m-1)$. The time increment determined by the pitch using Eq. 2.25 with $p = 5$ and $r_p = 0.1$ gives $\Delta t_p = 0.08$. Figure 4.1 (a) shows the lift force for $m = 3, 5, 10, 15, 20$. Figure 4.1(b) shows the lift force for $m = 20, 25, 30, 35$. For all cases except $m = 3$, the time increment is determined by Δt_d (determined by lunge). This is because the speed of pitch is slow. We notice that the solution does not change much for values of m greater than 20 indicating solver stability.

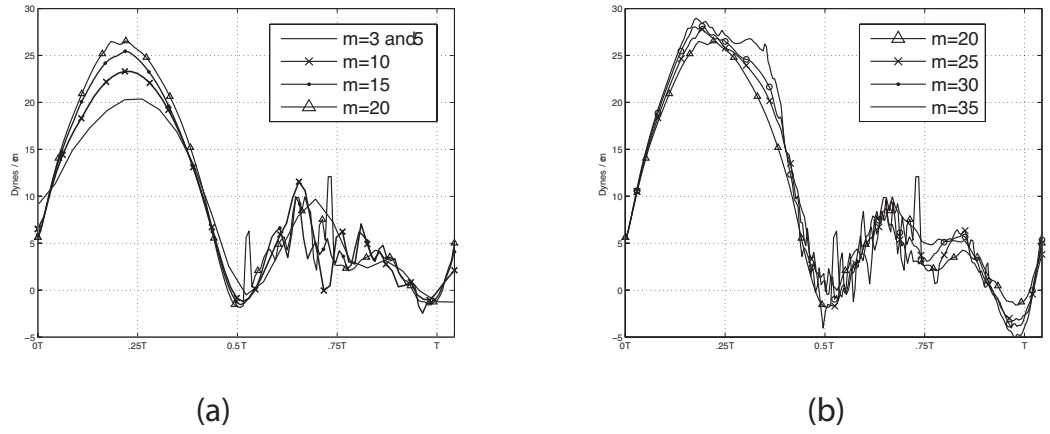


Figure 4.1: Lift force variation for varying m. (a) $m = 3, 5, 10, 15, 20$, (b) $m = 20, 25, 30, 35$. [6]

Translational velocity	$U = 100cm/s$, $V = 0cm/s$
Span and chord lengths	$l = 1cm$, $c = 0.5cm$
Top and bottom stroke angles	$\phi_T = 30$ degrees, $\phi_B = -90$ degrees
Stroke plane angle	0 degrees $\leq \beta <$ 360 degrees
Flapping frequency	$f = 50Hz$
Pitching speed, amplitude, offset	$p = 5$, 45 degrees, 0
Pitch axis offset	$a = 0cm$

Table 4.1: Flight parameters for cranefly.

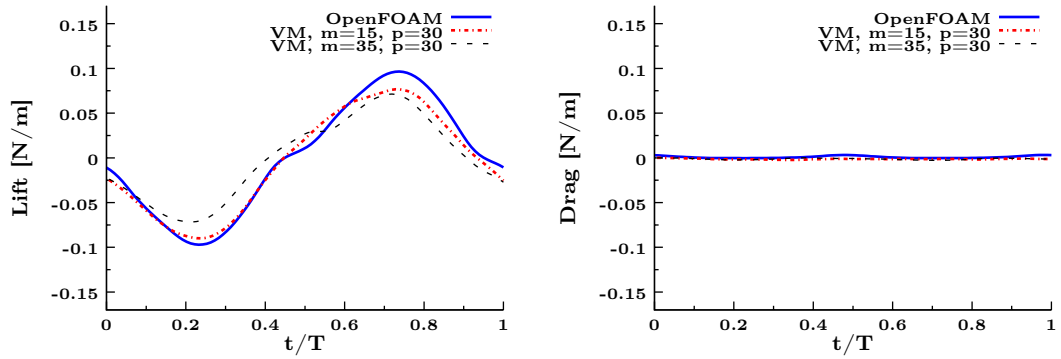
4.2 Comparison of results obtained by vortex method and OpenFOAM

In this section we investigate whether the solution of the vortex method converges to the solution of the N-S solver. Both the flow fields and the aerodynamic forces were compared between OpenFOAM and the vortex method. Two cases were compared, one of pure plunging and the other a combination of pitch and plunge.

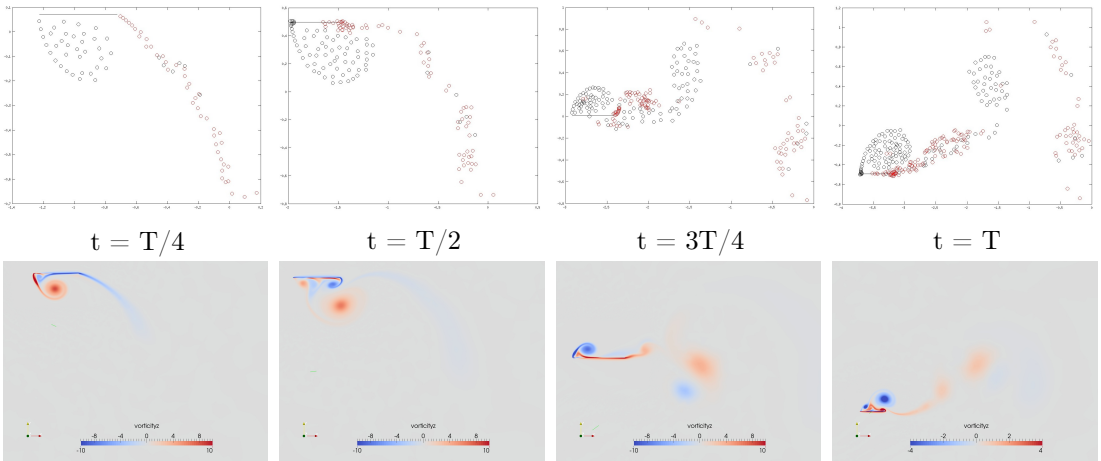
4.3 Pure plunging

The flow parameters are given in Table 4.2. The plunging motion is sinusoidal. There is a reasonable agreement between the flow field generated by vortex method and that of OpenFOAM. The vortex remains attached for a longer period in the case of vortex method while it is shed and smaller vortices are formed in OpenFOAM.

Figure 4.2 shows the comparison of drag and lift forces for $m = 15, 35$ and OpenFOAM. The time increment based on the heave is given by $\Delta t_d = 0.5/(m-1)$. The values of m used here give the results that are closest to the OpenFOAM solution among other values tried. This will be discussed later on.



(a)

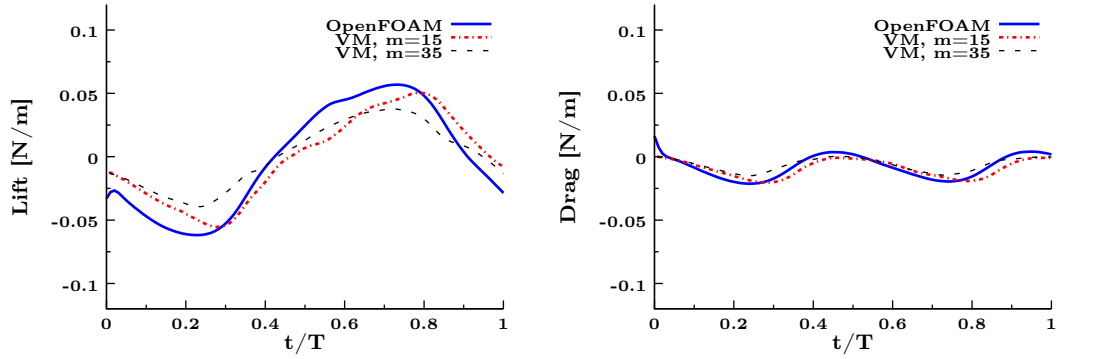


(b)

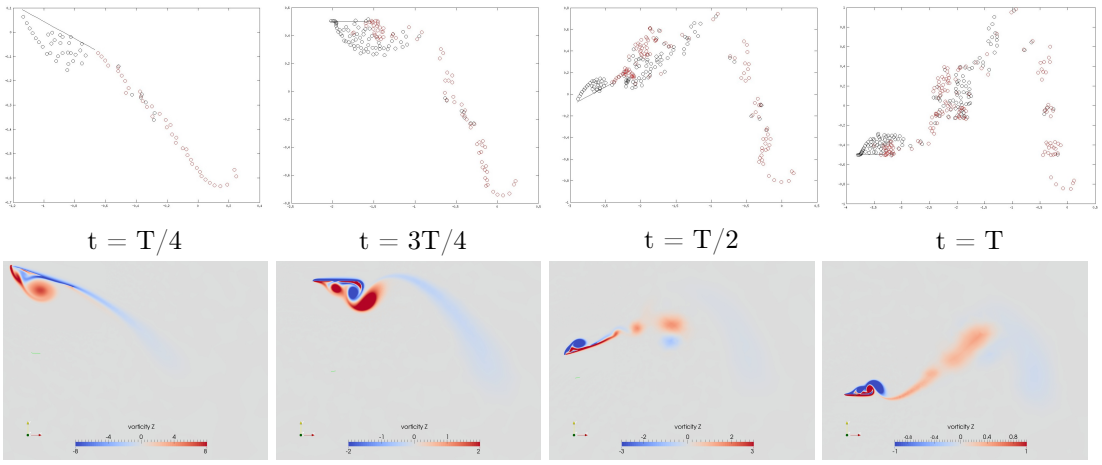
Figure 4.2: (a) Comparison of lift and drag forces for pure plunge obtained by the vortex method and OpenFOAM, (b) Corresponding vortex fields by the vortex method (top) and OpenFOAM (bottom) [6]

4.4 Pitching and plunging airfoil

The ambient flow velocity is 0.2 m/s with a wing flapping frequency of 0.028 Hz. The chord length and stroke amplitude are 1 meter each. The chord based Reynolds number is 1850. The pitch motion is out of phase with the plunge. Pitch amplitude is 19 degrees which was the limit due to mesh considerations. First we look at the flow field 4.3 which is quite accurately captured by the vortex method. The figures on the left are from the vortex method and those on the right are from OpenFOAM. The equations of motion are as follows $h = h_0 \sin \omega t$ $\theta = \theta_0 \cos \omega t$.



(a)



(b)

Figure 4.3: (a) Comparison of lift and drag forces for combined heave and pitch obtained by the vortex method and OpenFOAM, (b) Corresponding vortex fields by the vortex method (top) and OpenFOAM (bottom) [6]

The parameters used in igVortex, the MATLAB implementation are given in Table 4.3. Here we noticed that the time step based on the heave was not able to resolve the

forces well enough. By manually choosing a time increment based on the pitch with a value of $p = 30$ and $r_p = 0.1$, $\Delta t_p = 1/75 \approx 0.0133$ we found that there was the best agreement. This tendency was found to be consistent with various other cases with a pitch amplitude varying from 10 to 20 degrees and even the Karman vortex street 4.5.2 discussed ahead. Once again the forces are best captured by an m value between 15 and 35.

4.4.1 Tables with experimental values

Translational velocity	$U = 20cm/s$, $V = 0cm/s$
Span and chord lengths	$l = 2000cm$, $c = 100cm$
Top and bottom stroke angles	$\phi_T = 5.739$ degrees, $\phi_B = -5.739$ degrees
Stroke plane angle	90 degrees
Flapping frequency	$f = 0.028Hz$
Pitching speed, amplitude, offset	$p = 30$, 0 degrees, 0
Pitch axis offset	$a = 0cm$

Table 4.2: Parameters for pure heave. See Figure 4.2.

Translational velocity	$U = 20cm/s$, $V = 0cm/s$
Span and chord lengths	$l = 2000cm$, $c = 100cm$
Top and bottom stroke angles	$\phi_T = 5.739$ degrees, $\phi_B = -5.739$ degrees
Stroke plane angle	90 degrees
Flapping frequency	$f = 0.028Hz$
Pitching speed, amplitude, offset	$p = 30$, 19.27 degrees, 0

Table 4.3: Parameters for combined heave and pitch. See Figure 4.3.

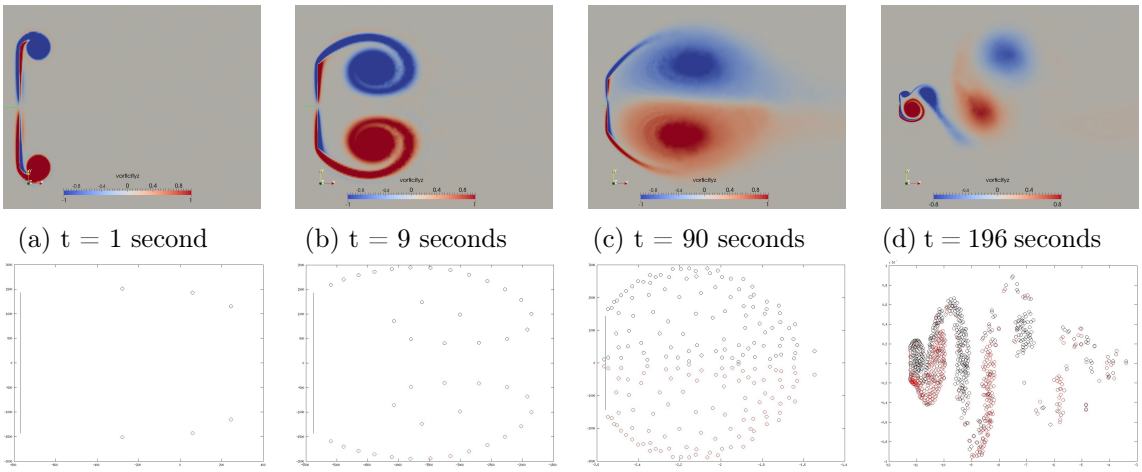
4.5 Is the vortex method capable of predicting viscous forces?

4.5.1 Role of viscosity

The version of the vortex method implemented here does not attempt to replicate viscosity in the traditional sense i.e a shearing force, boundary layer and the no-slip condition. Instead, the discreet bound and wake vortices replicate the flow structures due to the rotational nature of the flow induced by the point vortices on the surrounding fluid. This is consistent with the flow regime. For example, for flows with a Reynolds number greater than 10, the inertial force is 10 times larger than the viscous force. In such a situation, one must question the role of viscosity and the reason for unsteady phenomena like the vortex street and their influence on the aerodynamic forces. The only reason for the drag force to be significant enough is not due to the direct effects of viscosity but instead, because of the flow structures and the subsequent generation of inertial forces.

At the beginning of this thesis a few unsteady phenomena Section 1.4 pertinent to insect flight were discussed. Each of the phenomena are an indirect cause of viscosity. For example, the generation of the leading edge vortex which is regarded as the primary reason for the high lift generation in insects is a creation of viscosity. However, the increase in force is not due to viscosity itself but rather by the prevention of flow separation which is an indirect phenomena. The word indirect is used to refer to any effect other than the traditional boundary layer and the no-slip condition.

In fact, discretizing the wing with bound vortices actually corresponds to the boundary layer concept in a certain sense. Since the induced velocity gradually increases from zero at the center to a certain fixed value at the vortex core, it is quite similar to the boundary layer approximation and the no slip condition. To confirm that the vortex method is capable of replicating the effects of viscosity, the flow Section 4.5.2 past a cylinder was simulated. We found that the vortex method is able to replicate the flow structures created by viscosity and also generate similar aerodynamic forces as was verified with OpenFOAM. This is discussed further in the next section.



(a) t = 1 second (b) t = 9 seconds (c) t = 90 seconds (d) t = 196 seconds

Figure 4.4: Growth of the Karman vortex street behind a flat-plate in crossflow. Notice the formation of flow structures in (b) and the growing wake in (c).

4.5.2 The Karman vortex street

The Karman vortex street is the definitive test to confirm that the vortex method is able to capture the effects of viscosity. The flow is from left to right with a velocity of 0.2 m/s. The chord based Reynold number is 1850. For this case the time increment was manually entered in igVortex because there are no pitching or plunging motions. A time increment corresponding to a value of $p = 30$ is found to consistently give results closest to that of OpenFOAM. The reason for choosing this case is that the flow is well known and the structure formation is not interrupted by advection. From the Fig.4.4 it is evident that the vortex method captures the vortex formation very accurately. The shedding frequency is also similar as can be confirmed from the force plot. It must be noted that the shedding frequency is dependent on various numerical phenomena and we are not interested in obtaining an accurate estimate of the frequency. For forced flows like that of a flapping insect, the flow is quickly resolved and the flow structures are very dependent on the wing motion.

To get a feel of how the forces are being generated in igVortex, the MATLAB implementation, it helps to study the flow field generated in the vortex method and the corresponding force plot. The force plot was almost constant until the wake was symmetric. The moment the wake started to shed the force curve spiked and showed a sinusoidal nature. The force itself is not viscous in the traditional sense but rather

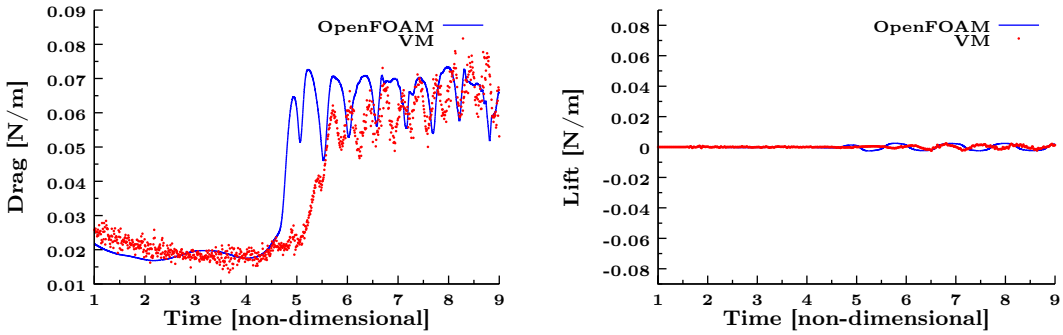


Figure 4.5: Lift and Drag for Karman vortex street

a consequence of wake asymmetry. The wake asymmetry is caused by the irrotational bound vortices. The curling up of the shed vortices into a single larger vortex occurs only due to the irrotational nature of the bound/wake vortices and their spatial distribution. A purely inviscid formulation is incapable of generating such a flow.

The lift force generated in Fig.4.5 can be easily understood from a momentum and control volume approach. The asymmetric wake creates an impulse at an angle to the wing. In order to enforce the no-penetration condition, the bound vortices must generate a flow of similar magnitude and direction at the surface of the wing. If we construct a control volume around the wing it can be easily seen that there is a force in the vertically upward direction.

From the arguments set forth above and the experimental evidence it can be concluded that the effects of viscosity have indeed been replicated. It can be said that the vortex method implemented here is in-between the inviscid approach employed in the potential flow theory and the viscous approach used in N-S solvers.

4.6 Optimum value of m

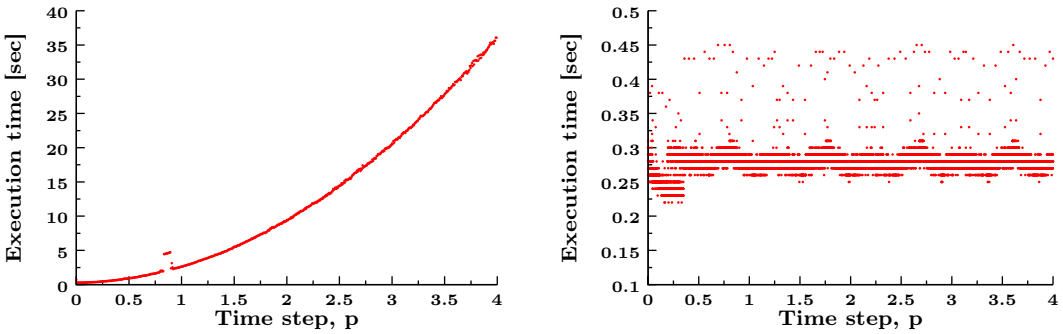
A parametric study was conducted for the above simulations by using different values of m ranging from $m = 3$ to $m = 50$. We found that the solution was rapidly converging towards the N-S solution for increasing values of m and stabilized at around $m = 25 : 35$. However, it began to deviate from the OpenFOAM solution after $m = 35$. The conclusion is that the vortex method does not converge towards the N-S solution as we increase the value of m. For large values of m, the vortex core becomes smaller and this

increases the effect of the singularity when vortices get close to the wing or even each other. Numerical difficulties arose in the form of sharp spikes in the force plots caused by the singularity.

For small values of m the solution is qualitatively inferior but still captures the trends correctly with a very small computational effort. This indicates that the vortex method is very suitable for a parametric study of various flight scenarios at low values of m before moving to higher values. Higher values of m reduce the time step and consecutively shed more vortices into the wake, increasing computation time. Therefore it is important to know the limitations of this method and use it in the range of m for which the solver gives reasonable results.

4.7 Computational performance of the vortex method

We have noticed a significant improvement in computation time with igVortex compared to OpenFOAM. One important thing to be kept in mind when interpreting these results is that igVortex is written in MATLAB and OpenFOAM is based in C++. This means that a significant amount of performance improvement is achievable by just implementing igVortex in C++. The computation time of igVortex increases parabolically Fig.4.6 with each time-step unlike OpenFOAM in which the only variation is due to the adaptive time-step. However the forces converge very quickly because of the wake being convected away and there is no need to compute for more than one oscillation.



(a) igVortex (b) OpenFOAM

Figure 4.6: Execution time per time-step.

In the table above we compare the computation time for each case. The simulations

Case	**OpenFOAM**	**Vortex Method**	**Speed up**
Pitch and plunge	57.8 minutes	2.7 minutes	21.4
Pure Plunge	55 minutes	2.8 minutes	19.6

Table 4.4: Performance

in OpenFOAM were conducted in parallel on 6 cores. We experimentally calculated that the simulations take 1.8 times longer to compute on a single core. This value changes slightly for different simulations but it doesn't change significantly for a given mesh and flow parameters. The values below are the computation time per oscillation of the flat-plate. There was a speed up of ≈ 20 with remarkable agreement between the forces and wake.

The vortex method is very suitable for cases when the flow field is quickly established. The reason for its speed is that there are very few computations required for each time step. However this keeps increasing as the wake vortices are generated. In the case of a N-S solver the computations required to proceed ahead for one time step are many. For a flow problem like that of the Karman vortex street 4.5.2, it is better to use a N-S solver because the flow takes significant time to develop.

Chapter 5

Conclusions and future work

In this thesis we have investigated the performance of two different techniques to solve the problem of flapping flight. We have seen the mathematical formulation of VM in which the wing is represented by a number of bound vortices. Two vortices, one from the LE and the other from the TE are shed at every time step. Convection of the wake vortices is achieved by a forward Euler method. A vortex core model was implemented to avoid artificially high velocities when the distance between the vortex and the observation point becomes small. The distinguishing feature of the vortex method in this thesis is that the Kutta condition is not enforced at all. This has greatly improved the stability of the solution compared to another version of the solver where the Kutta condition was enforced.

In order to simulate the same problem using a Navier–Solver, OpenFOAM was chosen because of it being open-source and customizable. Since OpenFOAM does not support two degrees of freedom mesh motion, we combined two dynamic mesh motion solvers to achieve the effect. A lot of effort was put into generating high quality meshes with OpenFOAM's native tools and third party tools like gmsh. The results obtained from OpenFOAM were validated Fig.3.2 against those of Liang et al. [11]. There was reasonable agreement between the two. Despite the differences in magnitude there was very good temporal agreement and the force peaks were captured very accurately. In addition, a crucial bug was identified and reported to the maintenance team at OpenCFD.

In order to compare the performance we chose three cases; pitch and plunge Fig.4.3, pure plunge Fig.4.2, vortex street Fig.4.4 and found reasonable agreement in the forces with both solvers. The most remarkable feature is the similarity between the wake

generated by both the solvers. Parametric studies were conducted with the vortex method to find a range of valued for which the solver is accurate. It has been found that the solver gives the best results for $m = 20 : 35$. This limit must be kept in mind when using the vortex method. It is recommended to be used as a preliminary check to gain a quick insight into the flapping pattern before moving onto a N-S solver to get the finer details of the flow.

Viscous effects in the traditional sense no longer apply for the vortex method employed here. This method can be thought of as a hybrid of the potential flow theory and the viscous N-S approach. It has been experimentally 4.5.2 proved that the effects of viscosity have been accurately captured in the flow structures and not as a shearing force. A completely inviscid formulation is incapable of such a flow structure.

A remarkable speed up, Table.4.4 of ≈ 20 was observed. Due to the fact that the vortex method converges quickly it is not necessary to compute for more that one oscillation.

5.1 Future work

There is huge scope for future work with the vortex method. It has to be extended to work for any geometry and include wing deformation effects. From a computational perspective there are the possibilities of converting it to a compiled language and subsequent parallelization. From the Fig.4.6 it seems that it is necessary to discard the effects of vortices far from the wing. This ”cutoff" will limit the computation time per time step which has a parabolic growth with the current formulation. This has been implemented by Ansari [2] by using a Lamb-Oseen vortex which is a vortex that decays with time. They also implemented an amalgamation technique in which vortices which are closer than a certain threshold are merged together to form a single stronger vortex. The other way to implement this is to either neglect vortices which are far enough from the wing or limit the extent to which vortices can interact with each other.

A more rigorous validation needs to be done under extreme wing motions. The smooth Biot–Savart law was able to reduce the spikes in the force plots but not eliminate

them. The size of the vortex core needs to be varied and tested to see if it can reduce the spikes without compromising the solution.

References

[1] SA Ansari, R Zbikowski, and K Knowles. Non-linear unsteady aerodynamic model for insect-like flapping wings in the hover. part 1: methodology and analysis. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 220(2):61–83, 2006.

[2] SA Ansari, R Zbikowski, and K Knowles. Non-linear unsteady aerodynamic model for insect-like flapping wings in the hover. part 2: implementation and validation. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 220(3):169–186, 2006.

[3] S.M. Belotserkovsky, V.N. Kotovskii, M.I. Nisht, and R.M. Fedorov. *Two-dimensional separated flows*. CRC Press, 1992.

[4] S.M. Belotserkovsky and I.K. Lifanov. *Method of discrete vortices*. CRC Press, 1992.

[5] James M Birch and Michael H Dickinson. Spanwise flow and the attachment of the leading-edge vortex on insect wings. *Nature*, 412(6848):729–733, 2001.

[6] M. Denda, Pruthvi K. Jujjavarapu, and Brandon C. Jones. A vortex approach for unsteady insect flight analysis in 2-d. *European Journal of Computational Mechanics*, 2015. review.

[7] Michael H Dickinson, Fritz-Olaf Lehmann, and Sanjay P Sane. Wing rotation and the aerodynamic basis of insect flight. *Science*, 284(5422):1954–1960, 1999.

[8] Charles P Ellington, Coen Van Den Berg, Alexander P Willmott, and Adrian LR Thomas. Leading-edge vortices in insect flight. 1996.

[9] Christophe Geuzaine and Jean-Francois Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[10] Hassan Kassem. twodoscillatingdisplacement. `https://github.com/HIKassem/twoDOscillatingDisplacement`, 2015.

[11] Chunlei Liang, Kui Ou, Sachin Premasuthan, Antony Jameson, and ZJ Wang. High-order accurate simulations of unsteady flow past plunging and pitching airfoils. *Computers & Fluids*, 40(1):236–248, 2011.

[12] Sanjay P Sane. The aerodynamics of insect flight. *The journal of experimental Biology*, 206(23):4191–4208, 2003.

[13] Henry G Weller, G Tabor, Hrvoje Jasak, and C Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in physics*, 12(6):620–631, 1998.

[14] Rafal Zbikowski. On aerodynamic modelling of an insect-like flapping wing in hover for micro air vehicles. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 360(1791):273–290, 2002.

Appendices

A Vortex method

A.1 Lunge and heave

The stroke length, d can be calculated using the wing length, $0.5l$ and the top and bottom stroke angles ϕ_T and ϕ_B.

$$d = d_T + d_B \tag{A.1}$$

where $d_T = 0.5l\sin(\phi_T)$ and $d_B = 0.5l\sin(-\phi_B)$, the minus sign is due to the chosen convention. The wing-translating system is located on the stroke line and moves along it. In the global co-ordinates its position is given by $(L - U_\infty, H - V_\infty)$, in terms of the heave, H and lunge, L. The lunge and heave are described by the sinusoidal function

$$L = \frac{1}{2}(d\cos\frac{2\pi(t+\tau)}{T} + e)\cos\beta, \tag{A.2}$$

$$H = \frac{1}{2}(d\cos\frac{2\pi(t+\tau)}{T} + e)\sin\beta, \tag{A.3}$$

where $e = d_T - d_B$ is the stroke length difference parameter between the top and bottom strokes, τ and T are the phase shift and the period of motion and β is the stroke plane angle. When $\tau = 0$, the motion starts from the top, while for $\tau = T/2$, the motion starts from bottom. By varying the phase shift from 0 to T, we can produce sinusoidal motions with various starting positions. The rates of lunge and heave are given by the time derivatives

$$\dot{L} = \frac{\pi d}{T}\cos\frac{2\pi(t+\tau)}{T}\cos\beta, \tag{A.4}$$

$$\dot{H} = \frac{\pi d}{T}\cos\frac{2\pi(t+\tau)}{T}\sin\beta, \tag{A.5}$$

A.2 Pitch(Rotation)

Wing rotation γ occurs about the origin $\hat{O}$ of the wing-translating system, as shown in Figure 1. Clockwise rotations are designated as positive. At 0 pitch, the wing chord line remains perpendicular to the stroke plane. With a non-zero pitch γ, the angle of attack of the chord line is given by $\alpha = \pi/2 - (\beta - \gamma)$. In an ideal situation the pitching motion occurs instantaneously at the ends of the up stroke and down stroke. The sudden pitching motion is described by a step function as shown in Figure 2, which correspond to the lunge and heave motion with $\tau = 0$. If the perfect pitch occurs exactly at the top and bottom then its called symmetric pitch. The timing of the pitch can be either before the top or bottom (advanced pitch) or after them (delayed pitch). These timings are specified by a timing offset parameter μ.

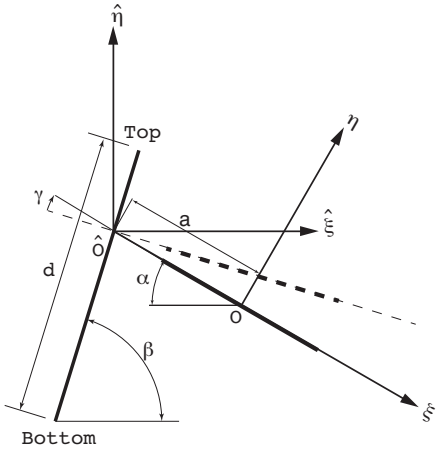


Figure 1: Stroke line of length d and angle β. Pitch angle γ and attack angle α [6]

It's physically impossible for an insect to achieve the perfect pitch. Instead, the pitch motion occurs over a finite period of time. This looks like a smoothed pitch motion which is described as follows

$$f_{ti} = \frac{2}{1 + e^{-2p(t-t_i)}}, \tag{A.6}$$

which describes the step function that jumps from 0 to 1 at $t = t_i$ when $p \to \infty$. Each

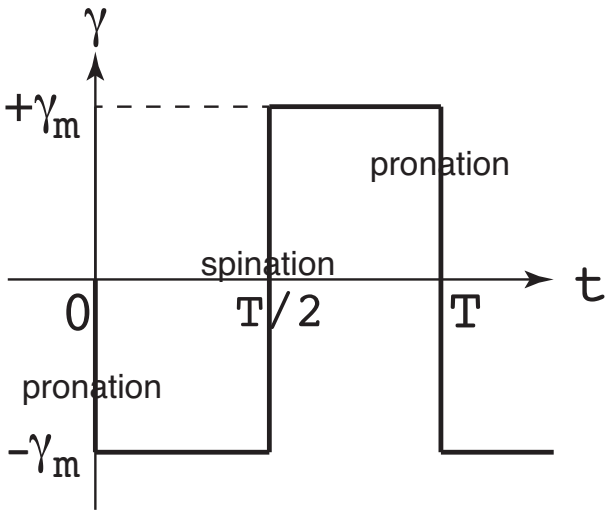


Figure 2: Ideal pitch motion [6]

stroke reversal is obtained by using the amplitudes $+2\gamma_m$ $-2\gamma_m$, respectively. The entire series of smooth pitching in one period is given by the superposed smooth step functions given by

$$\gamma = \gamma_m(1 - f_0 + f_{T/2} - f_T). \tag{A.7}$$

Figure 3 shows variations of smooth pitching with symmetric, advanced and delayed pitching.

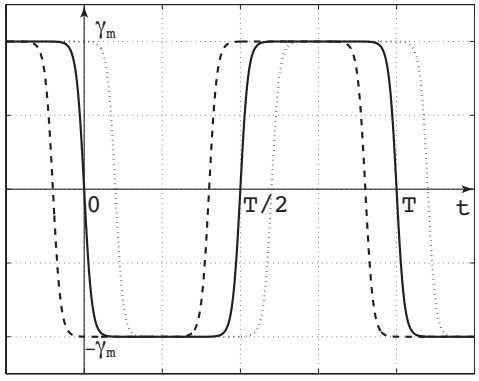


Figure 3: Smoothed pitch rotation with symmetric (solid), advanced (dashed) and delayed (dotted) pitch. [6]

The pitch rate is given by the time derivative

$$\dot{f}_{ti} = \frac{4pe^{-2p(t-t_i)}}{1+e^{-2p(t-t_i)^2}}, \tag{A.8}$$

in

$$\dot{\gamma} = \gamma_m(-f_0 + \dot{f}_{T/2} - \dot{f}_T). \tag{A.9}$$

If the timing of the pitch is offset, then replace t_i above by $ti + \mu$, where $\mu < 0$ for advanced and $\mu > 0$ for delayed pitch.

A.3 Physical meaning of pitch parameter, p

Let Δt_p be the time the approximate step function f_{ti} takes to complete the smooth increase of pitch by the amount $2\gamma_m$. The velocity variation has a maximum at $t = t_i$ as shown in Figure 4 and has a maximum value of $\gamma_m p$ as obtained from Eq.A.8. The average velocity over the pitching period can be calculated by inscribing a triangle with base length Δt_p and the height $\gamma_m p$. This average velocity, $1/2\gamma_m p$ acts over a period of Δt_p to produce the total jump $2\gamma_m$. This gives rise to the following relation

$$p = \frac{4}{\Delta t_p} \tag{A.10}$$

Since the slowest pitching motion corresponds to a pitching motion which occurs for half a translational period Δt_p, the minimum value of the pitch parameter is $p_m in = 8/T$. This is the slowest possible pitch motion that can be achieved and it is very similar to a sinusoidal motion. The estimation here is an approximation since the inscribed triangle cannot fully replicate the bell shaped variation. Therefore, the minimum value of p needs to be set at slightly higher than $8/T$.

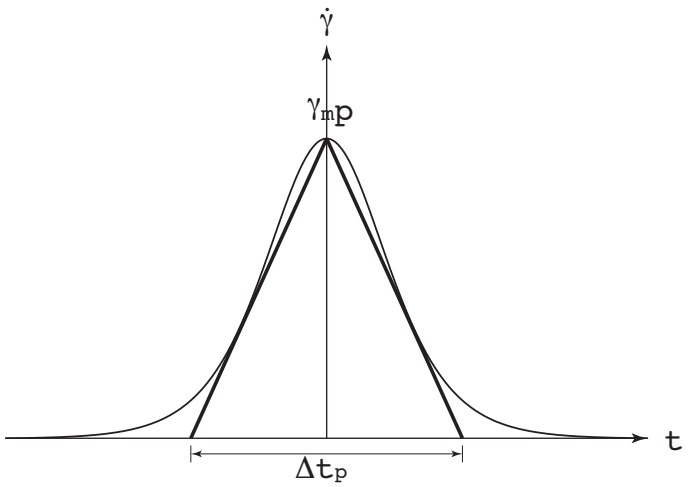


Figure 4: Pitch velocity variation, [6]

A.4 Smooth Biot-Savart law

The Biot-Savart law states that for a vortex of strength Γ at a given location ζ_0, the conjugate of the complex velocity at a location ζ is given by the following.

$$\overline{v}(\tilde{\zeta}) = -\frac{i\Gamma}{2\pi}\frac{1}{\tilde{\zeta} - \tilde{\zeta_0}} \tag{A.11}$$

The induced velocity blows up at locations very close to the vortex core, $|\tilde{\zeta} - \tilde{\zeta_0}| \approx 0$. This causes un-physical velocities when vortices come close to each other. To avoid such problems a vortex core model is used, also called the smooth Biot-Savart kernel in literature. A vortex core of radius r_c is chosen such that the induced velocity linearly increases from zero at the origin, to the value at the boundary of the core, $v(\zeta_{r_c})$.

$$v(\zeta) = -\frac{i\Gamma}{2\pi}\frac{1}{\zeta - \zeta_0}(\frac{r}{r_c})^2, \qquad r = |\zeta - \zeta_0| \leq r_c. \tag{A.12}$$

The radius of the vortex core r_c is given by half the distance between two bound vortices

$$r_c = 0.5\frac{c}{m-1}, \tag{A.13}$$

where c is the chord length of the wing and m is the number of bound vortices.

B OpenFOAM

B.1 How to read the source code?

It is very possible that the boundary condition you are looking for is not documented in neither the OpenFOAM user guide or the programmers guide but is available in OpenFOAM. Therefore its important to know how to dig into the code and learn about OpenFOAM's capabilities on your own. An example will be discussed below to give future readers an idea of how to search for documentation in OpenFOAM.

Go to the Github page of the OpenFOAM version you are using, 2.4.x in my case. Navigate to OpenFOAM-2.4.x/src/finiteVolume/fields/fvPatchFields/derived/. Here you will find a long list of folders. Each folder contains a .C file and a .H file. As an example lets consider the oscillatingFixedValue folder. In the folder there are four files. Go to the oscillatingFixedValueFvPatchField.H file. There is a section called Description which begins from line 30. Here are the contents of the Description section.

B.2 Description of oscillatingFixedValueFvPatchField.H

This boundary condition provides an oscillating condition in terms of amplitude and frequency.

$$x_p = (1 + asin(2\pi ft))x_{ref} + x_o where$$

Variable	x_p	x_{ref}	x_o	a	f	t
Description	patch values	patch reference values	patch offset values	amplitude	frequency [1/s]	time [s]

Patch usage:

Property	Description	Required	Default value
refValue	reference value	yes	
offset	offset value	no	0.0
amplitude	oscillation amplitude	yes	
frequency	oscillation frequency	yes	

Example of the boundary condition specification

```
myPatch
{
    type            oscillatingFixedValue;
    refValue        uniform 5.0;
    offset          0.0;
    amplitude       constant 0.5;
    frequency       constant 10;
}
```

Note

The amplitude and frequency entries are DataEntry types, able to describe time varying functions. The example above gives the usage for supplying constant values.

Now it should be pretty clear what the boundary condition is for and how it has to be implemented. This is a sinusoidal boundary condition. Notice the note which informs that the amplitude and frequency are of DataEntry type. To find out how to implement a time-varying function repeat the above process by going through the code in the DataEntry folder. A quick way to locate a folder in OpenFOAM is to use the following command in the Linux Terminal.

```
1  $ locate DataEntry | grep src
```

The above command will output all the files and folders with the name ”DataEntry" in the ”src" folder. Comb through the folders and read the descriptions given. You will find out how to implement a time-varying value.