UNIVERSITY OF CALIFORNIA,
IRVINE

Numerical and Analytical Investigation of the Unsteady Lift Characteristics of Plunging-Pitching-Flapping Wings

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Mechanical and Aerospace Engineering

by

Ryan Muoio

Thesis Committee:
Professor Haithem Taha, Chair
Professor Kenneth Mease
Professor Solmaz Kia

2016

DEDICATION

To:

The One who created me with: 1) enough determination to make it this far, and 2) the spark of lunacy to want to go even further in academia.

My mother, who supported me financially during my undergraduate career and who has always been there to offer me help when needed.

My father, who built me up with words of encouragement, always believing in me, even when I didn't believe in myself.

My friends, who always thought me to be smarter than I am; if it weren't for you, I wouldn't have pushed myself to try to attain the level of knowledge you thought I already had.

TABLE OF CONTENTS

Page

LIST OF FIGURES **v**

ACKNOWLEDGMENTS **vii**

CURRICULUM VITAE **viii**

ABSTRACT OF THE THESIS **ix**

1 Introduction **1**

2 Theodorsen **3**
2.1 Velocity Potentials 3
2.2 Induced Loads 6

3 Leishman **11**
3.1 State-Space Representation 11

4 Garrick **14**

5 Unsteady Vortex Lattice Method (UVLM) **17**
5.1 Biot-Savart: Function 18
5.2 Geometry 21
5.2.1 Time-Invariant Geometry: Lines 99-108 21
5.2.2 Time-Varying Geometry: Lines 134-148 23
5.3 Kinematics: Lines 114-142 24
5.4 Influence Coefficient Matrix (ICM): Lines 150-160 26
5.5 Right Hand Side: Lines 163-179 30
5.6 Solving the System: Lines 182-234 32
5.6.1 Pressure Coefficient C_p 33
5.6.2 Lift Coefficient C_L 37
5.7 Convection: Lines 237-270 38

6 Results **44**
6.1 Verification 44
6.2 Lift Shape Validation 46
6.2.1 Plunging 47

6.2.2 Pitching . . . 53
6.2.3 Flapping . . . 58
6.3 Root-Mean Square Error . . . 63
6.4 Phase Difference . . . 65
6.5 Lift Amplitude . . . 68
6.6 Mean Lift . . . 71

7 Conclusion and Future Work **76**
7.1 Conclusion . . . 76
7.2 Future Work . . . 77
7.2.1 Two-Dimensional . . . 77
7.2.2 Three-Dimensional . . . 78
.1 Appendix A . . . 80
.1.1 Geometric Constants . . . 80
.2 Appendix B . . . 80
.2.1 Unsteady Vortex Lattice Method Code . . . 80
.2.2 Theodorsen Code . . . 91
.2.3 Leishman Code . . . 93
.2.4 Phase Difference Code . . . 95

LIST OF FIGURES

Page

2.1 Conformal representation of the wing profile by a circle. 4
2.2 Parameters of the airfoil-flap combination. 5
2.3 Conformal representation with circulatory flow. 8

5.1 Coordinate system for UVLM approach. 18
5.2 Pictorial representation of the Biot-Savart law. 19
5.3 A look at the Rankine vortex model. 20
5.4 Discretization of the airfoil using collocation points. 22

6.1 Lift coefficient versus time. 46
6.2 A 2D visualization of an oscillating flap deflection. The black is the flat plate with a flap; the cyan is the wake convection. This image is from the UVLM mid-run. 47
6.3 Plunging for $k = 0.01$. 48
6.4 Plunging for $k = 0.2$. 49
6.5 Plunging for $k = 0.4$. 50
6.6 Plunging for $k = 0.6$. 51
6.7 Zoomed in case of plunging at $k = 0.01$ to show erroneous results of Leishman's formulation. 52
6.8 Pitching for $k = 0.01$. 54
6.9 Pitching for $k = 0.2$. 55
6.10 Pitching for $k = 0.4$. 56
6.11 Pitching for $k = 0.6$. 57
6.12 Flapping for $k = 0.01$. 59
6.13 Flapping for $k = 0.2$. 60
6.14 Flapping for $k = 0.4$. 61
6.15 Flapping for $k = 0.6$. 62
6.16 RMS_e trend for plunging, pitching, and flapping. 64
6.17 The phase difference ϕ for a plunging motion. 65
6.18 The phase difference ϕ for a pitching motion. 66
6.19 The phase difference ϕ for a flapping motion. 67
6.20 The lift amplitude for a plunging motion. 68
6.21 The lift amplitude for a pitching motion. 69
6.22 The lift amplitude for a flapping motion. 70
6.23 The mean lift for plunging at varying frequency. 72

6.24 The mean lift for pitching at varying frequency. 73
6.25 The mean lift for flapping at varying frequency. 74

7.1 Visualization of 3D UVLM code with a flap deflection. There are four chordwise panels and fourteen spanwise panels. The chord length is one meter and the span length is six meters. 79

ACKNOWLEDGMENTS

I would like to extend my thanks to the Department of Mechanical and Aerospace Engineering for granting me a Teaching Assitantship for final two quarters of my master's program. Without the financial support, I wouldn't have been able to complete this degree.

Professor Taha, without your patient guidance of my academic path, I doubt I would be where I am today. You accepted me as you student even though I had no experience in your area of expertise. You subsequently spent hours teaching me; and I spent hours not grasping the information as quickly as others may have, yet you continued to pour time into me. Your tutelage has inspired me to become great, both as a person and a teacher.

Friends, yes, I'm grateful to say I have many. To those in school with me, who brainstormed with me, showed me I was wrong, and taught me how to be right, you know who you are. But because I said I would, I'll name a few: Meng-Ting, Edgar, Tuan, James, Zeinab, Quentin, Vatche, and Joseph. You've helped me more than you know.

CURRICULUM VITAE

Ryan Muoio

EDUCATION

Master of Science in Mechanical and Aerospace **2016**
University of California, Irvine *Irvine, California*

Bachelor of Science in Physics **2012**
University of California, Riverside *Riverside, California*

RESEARCH EXPERIENCE

Graduate Researcher **2015–2016**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Teaching Assistant **2016**
UCI Classes: Mechanical Vibrations and Mechanical Lab *Irvine, California*

High School Tutor **2014**
Freelance for AP Physics students *Newport, California*

WORK EXPERIENCE

Component Engineer **2012–2014**
Extron Electronics *Anaheim, California*

Research Engineer **2016**
University of California, Irvine *Irvine, California*

ABSTRACT OF THESIS

Numerical and Analytical Investigation of the Unsteady Lift Characteristics of Plunging-Pitching-Flapping Wings

By

Ryan Muoio

Master of Science in Mechanical and Aerospace Engineering

University of California, Irvine, 2016

Professor Haithem Taha, Chair

Three methods for computing the unsteady lift response due to basic kinematic motion—plunging, pitching, and flapping—are compared. The first two methods are analytical models developed by Theodorsen and Leishman, while the third method is a numerical method known as the unsteady vortex lattice method. The MATLAB code for this numerical method is included. All calculations are evaluated for the potential flow case.

Chapter 1

Introduction

The lift response due to a oscillating flap deflection has been a topic of research since the 1930s, with it first being addressed for the harmonic case by Theodorsen in 1935 [1]. Restricting the theory to potential flow and the Kutta condition, Theodorsen resolved the solution of the problem into Bessel functions of the first and second kind and of zero and first order.

Two year later, in 1937, Garrick [2] extended Theodorsen's theory to include the horizontal force due to flap deflection. He used as a starting point the work of Wagner [3], then applied the compact methods outlined by von Kármán and Burgers [4] to treat the propulsion of an airfoil oscillating in three degrees of freedom. In doing so, Garrick developed a method to find the suction force, which can be used in both lift and drag calculations.

Skipping forward many decades, in 1994, Leishman [5] addressed the problem of unsteady lift produced by an oscillating flap, but he described the response using step concepts. Instead of taking into account solely harmonic motions like Theodorsen, Leishman considered the effect of non-uniform flow, much like that of von Kármán and Sears [6], R.T. Jones [7], and Wagner. Using the Duhamel superposition integral and an improved exponential approximation to Wagner's step lift function, Leishman obtained a controllable canonical form of the solution.

With analytical tools already developed, a numerical approach known as the unsteady vortex lattice method (UVLM) is used by the author to obtain results comparable to Theodorsen and Leishman for induced lift. The objective of this paper is to compare the UVLM to these two analytical lift-predicting methods.

The general motivation for this task is to better understand the difference between select analytical and numerical methods. It is desired to determine where and when this particular numerical approach can be used in place of the known analytical approaches, since Theodorsen's and Leishman's formulations are limited to rigid bodies, whereas the UVLM is not.

Chapter 2

Theodorsen

Theodore Theodorsen is considered a pioneer of classical unsteady aerodynamics, with many published works in the field. His paper including flap deflection [1] is referenced here.

Assumptions intrinsic to the classical model are that linearized partial differential equations and linear boundary conditions govern the problem, with loads calculated on a thin, rigid airfoil.

2.1 Velocity Potentials

Theodorsen begins his process by determining the velocity potentials created by the position and velocity of the individual parts in the whole of the airfoil system. The airfoil is first represented by a circle in the ξ-plane, such that $\xi = \chi + i\eta$. A source $2Q$ is placed at (χ_1, η_1) and a negative source $-2Q$ at $(\chi_1, -\eta_1)$, as shown in Figure 2.1.

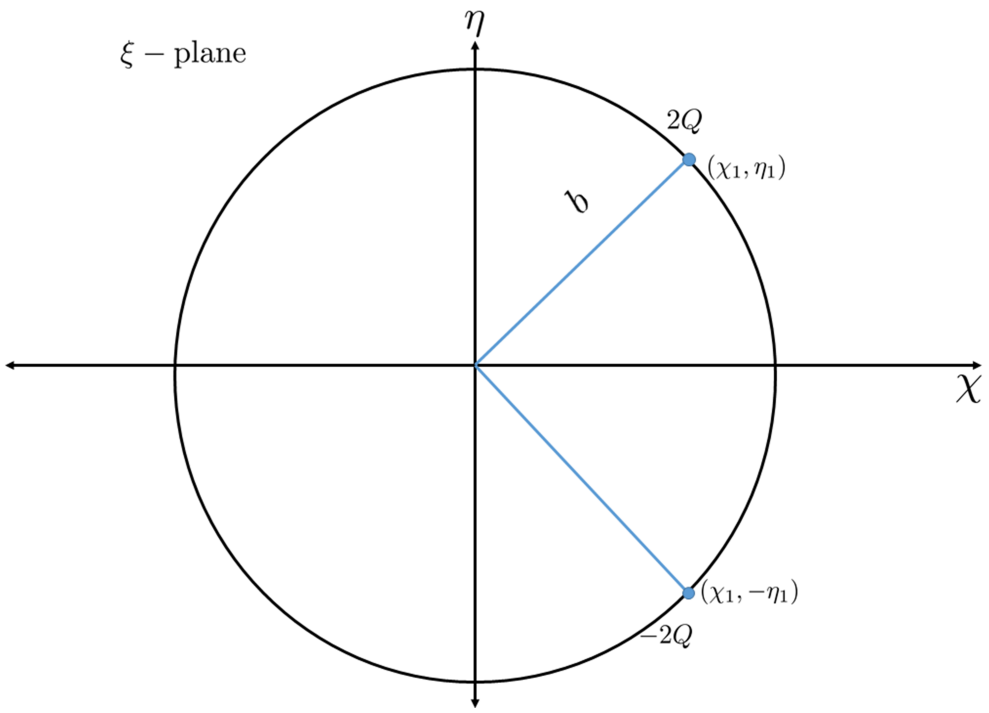

Figure 2.1: Conformal representation of the wing profile by a circle.

The flow around the circle is therefore

$$\varphi = \frac{Q}{2\pi} \ln \frac{(\chi - \chi_1)^2 + (\eta - \eta_1)^2}{(\chi - \chi_1)^2 + (\eta + \eta_1)^2}$$

Knowing that $|\xi| = \sqrt{\chi^2 + \eta^2} = b$, the above equation is reduced to a function of one variable.

$$\varphi = \frac{Q}{2\pi} \ln \frac{(\chi - \chi_1)^2 + (\sqrt{b^2 - \chi^2} - \sqrt{b^2 - \chi_1^2})^2}{(\chi - \chi_1)^2 + (\sqrt{b^2 - \chi^2} + \sqrt{b^2 - \chi_1^2})^2}$$

Using the Joukowski transformation

$$z = \frac{\xi}{2} + \frac{b^2}{2\xi} \tag{2.1}$$

the function φ on the circle gives directly the surface potential of the chord, the projection of the circle on the horizontal diameter in the physical plane. This transformation allows the velocity potentials for the multiple parts of the airfoil-flap combination to be treated as

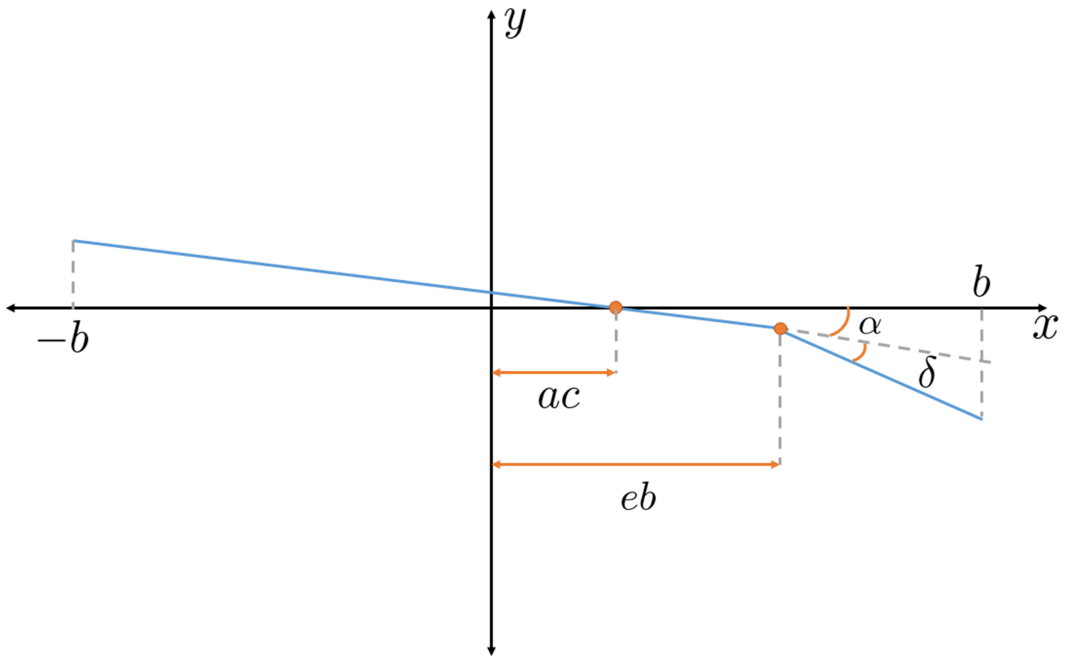

Figure 2.2: Parameters of the airfoil-flap combination.

follows:

1. The effect of the flap deflected at and angle δ gives rise to a function φ_δ by substituting $-U_\infty \delta b$ for Q.

$$\varphi_\delta = \frac{U_\infty \delta b}{\pi}\left[\sqrt{1-x^2}\cos^{-1} e - (x-e)\ln N\right]$$

   where e is the ratio of the flap to the chord length (meaning eb gives the position of the flap hinge point), δ is the flap deflection angle, U_∞ is the freestream velocity, and

$$N = \frac{1 - ex - \sqrt{1-x^2}\sqrt{1-e^2}}{x-e}$$

2. To account for the flap moving down at an angular velocity of $\dot{\delta}$, Q is set to be $Q = -(\chi_1 - e)\dot{\delta} b^2$, which gives

$$\varphi_{\dot{\delta}} = \frac{\dot{\delta} b^2}{2\pi}\left[\sqrt{1-e^2}\sqrt{1-x^2} + \cos^{-1} e(x-2e)\sqrt{1-x^2-(x-e)^2}\ln N\right]$$

3. To obtain the effect due to the pitching of the entire airfoil an angle α, $e = -1$ is placed in the expression for φ_δ, hence

$$\varphi_\alpha = U_\infty \alpha b \sqrt{1 - x^2}$$

4. To describe the rotation around point a at an angular velocity $\dot{\alpha}$, consider that this motion can be described by a rotation about the leading edge $e = -1$ at $\dot{\alpha}$, plus a vertical motion with velocity $-\dot{\alpha}(1 + a)b$. Therefore,

$$\varphi_{\dot{\alpha}} = \dot{\alpha} b^2 \left(\frac{1}{2} - a \right) \sqrt{1 - x^2}$$

5. To depict the airfoil in a downward heaving motion with velocity $\dot{h}$, the quantity $\dot{h}/U_\infty$ is used in place of the α in φ_α, giving

$$\varphi_{\dot{h}} = \dot{h} b \sqrt{1 - x^2}$$

2.2 Induced Loads

With the velocity potentials known, the local pressures can be integrated to obtain the forces on the airfoil-flap system.

The unsteady Bernoulli equation

$$p = -\rho \left(\frac{w^2}{2} + \frac{\partial \varphi}{\partial t} \right)$$

is employed, where w is local velocity. As per classical methodology, this equation is lin-

earized by setting $w = U_\infty + \partial\varphi/\partial x$ to give

$$p = -2\rho\left(U_\infty \frac{\partial\varphi}{\partial x} + \frac{\partial\varphi}{\partial t}\right)$$

The force on the entire airfoil is calculated by integrating the sum of partial derivatives of all the aforementioned velocity potentials, denoted as $\dot{\varphi}_{\text{total}}$, over the entire length of the chord.

$$L^{nc}(t) = -2\rho \int_{-b}^{b} \dot{\varphi}_{\text{total}} dx = -\rho b^2 [U_\infty \pi\dot{\alpha} + \pi\dot{h} - b\pi a\ddot{\alpha} - U_\infty F_4 \dot{\delta} - bF_1\ddot{\delta}] \tag{2.2}$$

This equation constitutes the total lift force produced by the non-circulatory flow. The coefficients F_4 and F_1 are geometric terms, which depend only on the size of the flap relative to the airfoil chord, and are expressed in Appendix A for a coordinate system centered at the midchord.

To take into account the circulatory flow, consideration is taken to determine the velocity potentials due to a surface of discontinuity of strength $\gamma(x)$ extending along the positive x-direction from the airfoil trailing edge to infinity. This is modeled by using the method of images, therefore placing a vortex of strength $-\Gamma_0$ at position χ_0 and a vortex of opposite strength Γ_0 at position b^2/χ_0 in the ξ-plane, as shown in Figure 2.3.

The resulting velocity potential is

$$\begin{aligned}
\varphi_\Gamma &= \frac{\Gamma_0}{2\pi}\left[\tan^{-1}\left(\frac{\eta}{\chi - \chi_0}\right) - \tan^{-1}\left(\frac{\eta}{\chi - \frac{b^2}{\chi_0}}\right)\right] \\
&= \frac{\Gamma_0}{2\pi}\tan^{-1}\frac{\left(\chi_0 - \frac{b^2}{\chi_0}\right)\eta}{\chi^2 - \left(\chi_0 + \frac{b^2}{\chi_0}\right)\chi + \eta^2 + b^2}
\end{aligned}$$

Bringing this equation into the physical plane requires equation 2.1, the Joukowski transfor-

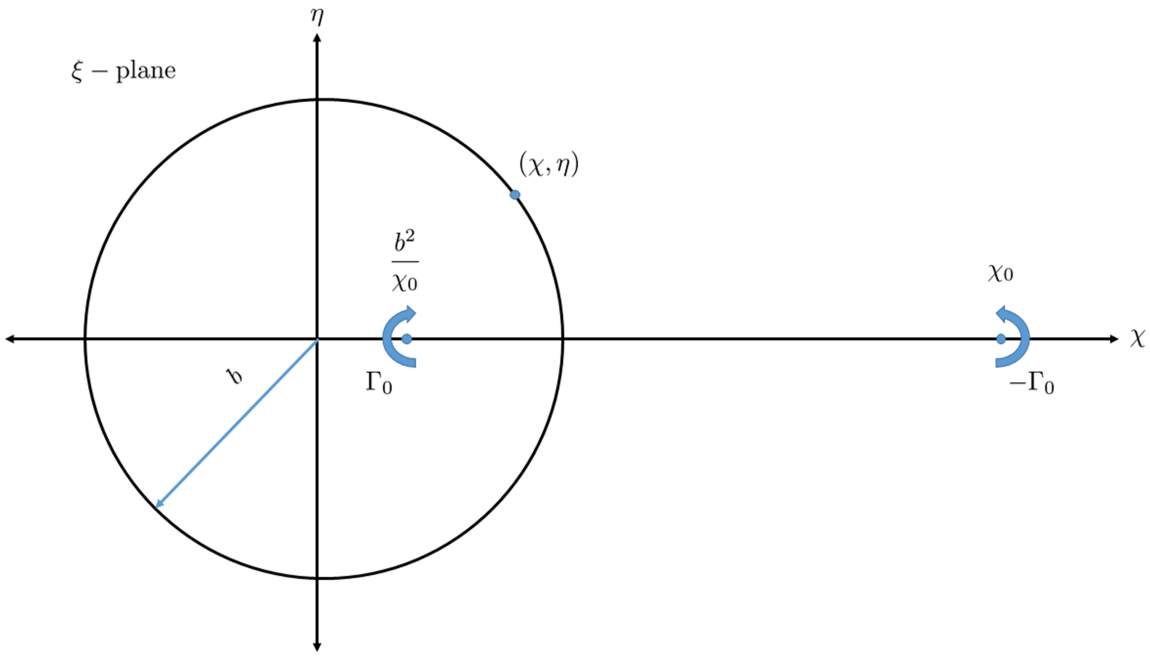

Figure 2.3: Conformal representation with circulatory flow.

mation, which gives the following relations:

$$\chi_0 = x_0 + \sqrt{x_0^2 - b^2}$$

$$\chi = x$$

$$\eta = \sqrt{b^2 - x^2}$$

The equation for the potential is

$$\varphi_\Gamma = -\frac{\Gamma_0}{2\pi} \tan^{-1} \frac{\sqrt{b^2 - x^2}\sqrt{x_0^2 - b^2}}{b^2 - x x_0} \tag{2.3}$$

which gives the clockwise circulation due to the element $-\Gamma_0$ at x_0. This is placed into the unsteady Bernoulli equation; but, to keep things linear, the element $-\Gamma_0$ is regarded as moving in the positive x-direction (with no y-component) at a speed of U_∞, such that

$$\frac{\partial \varphi_\Gamma}{\partial t} = \frac{\partial \varphi_\Gamma}{\partial x_0}$$

and therefore

$$p = -2\rho U_\infty \left(\frac{\partial \varphi_\Gamma}{\partial t} + \frac{\partial \varphi_\Gamma}{\partial x_0} \right)$$

Taking $\Gamma_0 = \gamma(x_0, t)dx_0$, the expression for the force on the whole airfoil is

$$L^c(t) = -\rho U_\infty \int_b^\infty \frac{x_0}{\sqrt{x_0^2 - b^2}} \gamma(x_0, t) dx_0$$

where γ is a function of the distance from the location of the first vortex element: $\gamma(x_0, t) = f(U_\infty t - x_0)$.

To calculate the magnitude of the circulation, the Kutta condition is imposed, which requires the velocity at the trailing edge ($x = b$) to be finite. Therefore,

$$\frac{\partial}{\partial x}(\varphi_\Gamma + \varphi_\alpha + \varphi_{\dot{h}} + \varphi_{\dot{\alpha}} + \varphi_\delta + \varphi_{\dot{\delta}}) = \text{finite}$$

Another way to view it is that the non-circulatory and circulatory velocities in the ξ-plane must add to zero. This is perhaps a more abstract way to view it. Regardless, the resulting equation is

$$\frac{1}{2\pi b} \int_b^\infty \gamma(x_0, t) \sqrt{\frac{x_0 + b}{x_0 - b}} dx_0 = U_\infty \alpha(t) + \dot{h}(t) + b\left(\frac{1}{2} - a\right)\dot{\alpha}(t) + \frac{F_{10}}{\pi} U_\infty \delta(t) + b\frac{F_{11}}{2\pi}\dot{\delta}(t)$$

which is the integral equation for circulation. The coefficients F_{10} and F_{11} are defined in Appendix A. With some manipulation, and with restricting the circulatory input to harmonic motion, the resulting circulatory lift with the imposed Kutta condition is

$$L^c(t) = -2\pi b\rho U_\infty \underbrace{\frac{\int_b^\infty \frac{\mu}{\sqrt{\mu^2-1}} e^{ik\mu} d\mu}{\int_b^\infty \sqrt{\frac{\mu+1}{\mu-1}} e^{ik\mu} d\mu}}_{C(k)} \left[U_\infty \alpha(t) + \dot{h}(t) + b\left(\frac{1}{2} - a\right)\dot{\alpha}(t) + \frac{F_{10}}{\pi} U_\infty \delta(t) + b\frac{F_{11}}{2\pi}\dot{\delta}(t) \right] \tag{2.4}$$

where $k = \omega b / U_\infty$ is the reduced frequency, $\mu = x_0/b$ is the normalized chord, and $C(k)$ is

known as the Theodorsen function, which can be written in terms of Hankel functions.

$$C(k) = \frac{H_1^{(2)}(k)}{H_1^{(2)}(k) + iH_0^{(2)}(k)}$$

The Theodorsen function is the transfer function of the aerodynamic system, and it describes the effect of shed wake vorticity. It should be noted that $C(k)$ is in the frequency domain, not the Laplace domain.

The total lift of the airfoil-flap system is computed by summing the non-circulatory and circulatory lifts.

$$\begin{aligned} L(t) = &-\rho b^2[U_\infty \pi\dot{\alpha} + \pi\dot{h} - b\pi a\ddot{\alpha} - U_\infty F_4\dot{\delta} - bF_1\ddot{\delta}] \\ &- 2\pi b\rho U_\infty C(k)\left[U_\infty\alpha(t) + \dot{h}(t) + b\left(\frac{1}{2} - a\right)\dot{\alpha}(t) + \frac{F_{10}}{\pi}U_\infty\delta(t) + b\frac{F_{11}}{2\pi}\dot{\delta}(t)\right] \end{aligned} \quad (2.5)$$

And so the coefficient of lift is:

$$\begin{aligned} C_L(t) = &-\frac{b}{U_\infty}[U_\infty \pi\dot{\alpha} + \pi\dot{h} - b\pi a\ddot{\alpha} - U_\infty F_4\dot{\delta} - bF_1\ddot{\delta}] \\ &- \frac{2\pi}{U_\infty}C(k)\left[U_\infty\alpha(t) + \dot{h}(t) + b\left(\frac{1}{2} - a\right)\dot{\alpha}(t) + \frac{F_{10}}{\pi}U_\infty\delta(t) + b\frac{F_{11}}{2\pi}\dot{\delta}(t)\right] \end{aligned} \quad (2.6)$$

Chapter 3

Leishman

3.1 State-Space Representation

Leishman begins his analysis with an equation exactly equivalent to equation 2.6, Theodorsen's lift equation for a flapped airfoil [5].

$$C_L(t) = \frac{\pi b}{U_\infty}\left[\ddot{h} + U_\infty\dot{\alpha} - ba\ddot{\alpha}\right] + \frac{b}{U_\infty^2}(-U_\infty F_4\dot{\delta} - bF_1\ddot{\delta}) + 2\pi C(k)\Bigg[\underbrace{\frac{\dot{h}}{U_\infty} + \alpha + b\left(\frac{1}{2} - a\right)\frac{\dot{\alpha}}{U_\infty}}_{\alpha_{qs}} + \underbrace{\frac{F_{10}\delta}{\pi} + \frac{bF_{11}\ddot{\delta}}{2\pi U_\infty}}_{\delta_{qs}}\Bigg] \quad (3.1)$$

As denoted, the quantities within the brackets of equation 3.1 are divided into two groupings, α_{qs} and δ_{qs}, for convenience and clarity of the following manipulations.

Rather than stopping here and using the frequency response method provided by Theodorsen, Leishman approaches this problem from the step response standpoint. Using Duhamel's

superposition intergral together with Wagner's step response, equation 3.1 can be written as

$$C_L(t) = \frac{\pi b}{U_\infty}\left[\ddot{h} + U_\infty\dot{\alpha} - ba\ddot{\alpha}\right] + \frac{b}{U_\infty^2}(-U_\infty F_4\dot{\delta} - bF_1\ddot{\delta}) + 2\pi\Bigg[\alpha_{qs}(0)\phi_W(s) + \int_s^0 \frac{d\alpha_{qs}}{d\sigma}(s-\sigma)d\sigma + \delta_{qs}(0)\phi_W(s) + \int_s^0 \frac{d\delta_{qs}}{d\sigma}(s-\sigma)d\sigma\Bigg] \quad (3.2)$$

where $s = U_\infty t/b$ is the time based on semichord lengths.

For evaluation of the Wagner function for practical uses, a second-order step response approximation is implemented, like of R.T. Jones [10] but with a more accurate coefficients.

$$\phi_W(s) = 1 - A_1 e^{-b_1 s} - A_2 e^{-b_2 s} \quad (3.3)$$

The main advantage of the exponential approximation of the Wagner function is that it has a simple Laplace transform, which allows the state-space equivalent of the Duhamel integral to be written in controllable canonical form.

Therefore, for the circulatory term describing airfoil motion, the state and output equations are

$$\begin{pmatrix}\dot{z}_1(t)\\ \dot{z}_2(t)\end{pmatrix} = \begin{bmatrix}0 & 1\\ -b_1b_2(U_\infty/b)^2 & -(b_1+b_2)(U_\infty/b)\end{bmatrix}\begin{pmatrix}z_1(t)\\ z_2(t)\end{pmatrix} + \begin{pmatrix}0\\ 1\end{pmatrix}\alpha_{qs}(t) \quad (3.4)$$

$$C^c_{L_\alpha}(t) = 2\pi\begin{bmatrix}(b_1b_2/2)(U_\infty/b)^2 & (A_1b_1 + A_2b_2)(U_\infty/b)\end{bmatrix}\begin{pmatrix}z_1(t)\\ z_2(t)\end{pmatrix} + \pi\alpha_{qs}(t) \quad (3.5)$$

For the flap motion,

$$\begin{pmatrix} \dot{z}_3(t) \\ \dot{z}_4(t) \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ -b_1 b_2 (U_\infty/b)^2 & -(b_1+b_2)(U_\infty/b) \end{bmatrix} \begin{pmatrix} z_3(t) \\ z_4(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \delta_{qs}(t) \tag{3.6}$$

$$C^c_{L_\delta}(t) = 2\pi \begin{bmatrix} (b_1 b_2/2)(U_\infty/c)^2 & (A_1 b_1 + A_2 b_2)(U_\infty/b) \end{bmatrix} \begin{pmatrix} z_3(t) \\ z_4(t) \end{pmatrix} + \pi \delta_{qs}(t) \tag{3.7}$$

The non-circulatory lift response

$$C^{nc}_L(t) = \frac{\pi b}{U_\infty} \left[\ddot{h} + U_\infty \dot{\alpha} - ba\ddot{\alpha} \right] + \frac{b}{U^2_\infty} (-U_\infty F_4 \dot{\delta} - bF_1 \ddot{\delta})$$

combined with equations 3.4 through 3.7 results in gives the total lift

$$C_L(t) = C^c_{L_\alpha}(t) + C^c_{L_\delta}(t) + C^{nc}_L(t) \tag{3.8}$$

Chapter 4

Garrick

Garrick [2] uses Theodorsen's analytical results and extends the derivation to include the forces in the horizontal direction in order to calculation the thrust (or propulsive force). In this paper, the drag forces are not the main focus, so the suction force is solely used to more accurately predict the lift.

The suction force is defined as

$$F_S = \pi \rho b S^2 \tag{4.1}$$

where S is obtained by the relation

$$S = \lim_{x \to -1} \gamma(x) \sqrt{\frac{x+1}{b}}$$

The value of S is finite since γ is infinite in the order of $1/\sqrt{x+1}$ at the leading edge $x = -1$. Due to this occurrence of infinity, it must be noted that ideal flow for an infinitely thin wing is impossible, which is the reason Garrick views this case as a limiting one: the airfoil is round and smooth at the leading edge but sharp at the trailing edge. Further explanation can be found in reference [4] and [8].

Deriving the equation for S, as shown by Garrick, yields

$$S = \frac{1}{2}\sqrt{2}\left[2C(k)Q - b\dot{\alpha} - \frac{2}{\pi}\sqrt{1-e^2}U_\infty\delta + \frac{F_4}{\pi}b\dot{\delta}\right]$$

where

$$Q = U_\infty\alpha + \dot{h} + b\left(\frac{1}{2} - a\right)\dot{\alpha} + \frac{1}{\pi}F_{10}U_\infty\delta + \frac{b}{2\pi}F_{11}\dot{\delta}$$

An alternate form of the S term is found by Ramesh [9]. After implementing ideas from an extended thin airfoil theory, the S term evaluates to

$$S = \lim_{\theta\to 0} U(t)A_0(t)\frac{1+\cos\theta}{\sin\theta}\sqrt{\frac{c}{2b}(1-\cos\theta)} \tag{4.2}$$

$$= \sqrt{2}U(t)A_0(t) \tag{4.3}$$

where $A_0(t)$ is the zeroth time-dependent Fourier coefficient defined as

$$A_0(t) = -\frac{1}{\pi}\int_0^\pi \frac{w(x,t)}{U(t)}d\theta \tag{4.4}$$

Here, $w(x,t)$ is the local downwash on the airfoil in the body z-direction, $U(t)$ is the freestream velocity, and θ is the variable resulting from Glauert's transformation:

$$x = \frac{c}{2}(1-\cos\theta)$$

Therefore, the suction force is

$$F_s = 2\pi\rho b U^2(t)A_0^2(t) \tag{4.5}$$

and the suction force coefficient is

$$C_{F_s} = 2\pi A_0^2 \tag{4.6}$$

Equation 4.6 is used to assist in more accurate lift predictions for the author's UVLM

calculations, as described in the following chapter.

Chapter 5

Unsteady Vortex Lattice Method (UVLM)

The description presented here of the UVLM is much more extensive and detailed than that of Theodorsen's, Leishman's, or Garrick's work, since the numerical method is the author's work. The underlying methodology is more clearly outlined, beginning with a statement of what is being addressed and followed with the fundamentals of the flow and specifics of the UVLM. In general, this section of the article is a walkthrough of the code programmed using the UVLM.

But before addressing anything further, the coordinate system is established to avoid confusion. The x-z coordinate system is defined with positive x being in the direction drawn from the leading edge to the trailing edge and the positive z being in the upward direction, as shown in figure 5.1.

With the axis set and understood, the rest of the description and analysis begins.

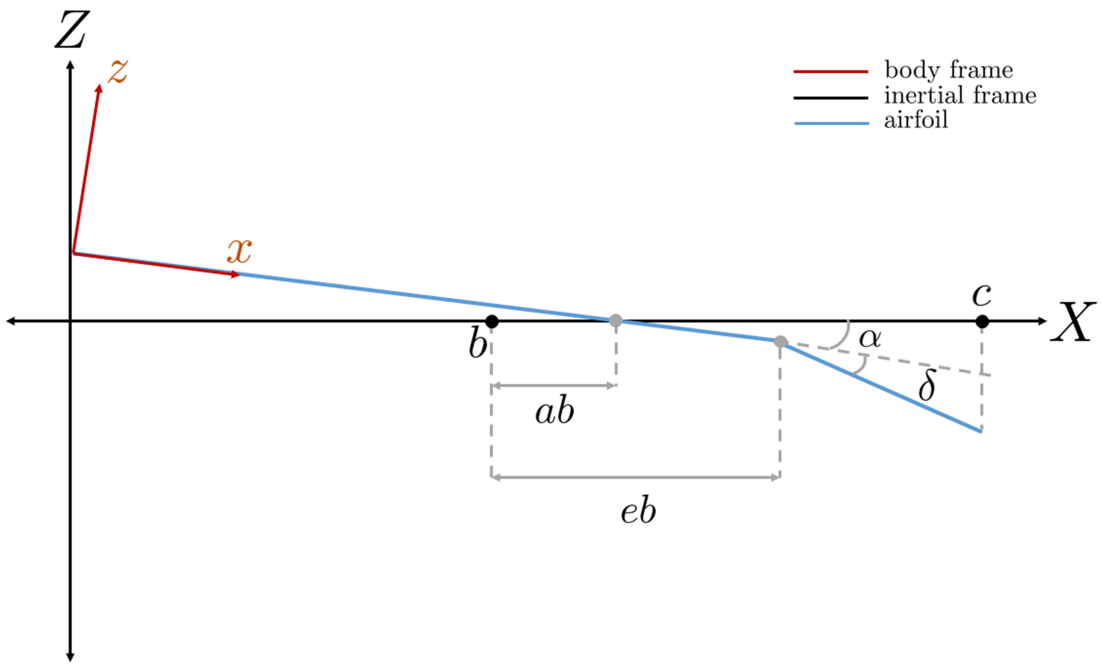

Figure 5.1: Coordinate system for UVLM approach.

5.1 Biot-Savart: Function

The Biot-Savart law is a fundamental relation in the theory of inviscid, incompressible flow, and it is a tool used in the ULVM. In general, the Biot-Savart law describes the flow field induced in the surrounding space of a curved vortex filament. If the circulation is taken about any path enclosing the filament, a constant value Γ is obtained; hence, the vortex filament strength is defined as Γ, where a clockwise flow is considered negative. According to Biot-Savart, the velocity **dq** induced at some point P by a directed segment **dl** is by definition

$$d\mathbf{q} = \frac{\Gamma}{4\pi} \frac{d\mathbf{l} \times \mathbf{r}}{|\mathbf{r}|^3}$$

where **r** is the radius vector from $d\mathbf{l}$ to the arbitrary point P, as seen figure 5.2.

For this two-dimensional flow case, only a *straight* vortex filament perpendicular to the airfoil is consider; straight, meaning the filament extends to $\pm\infty$ along the y-axis (the axis

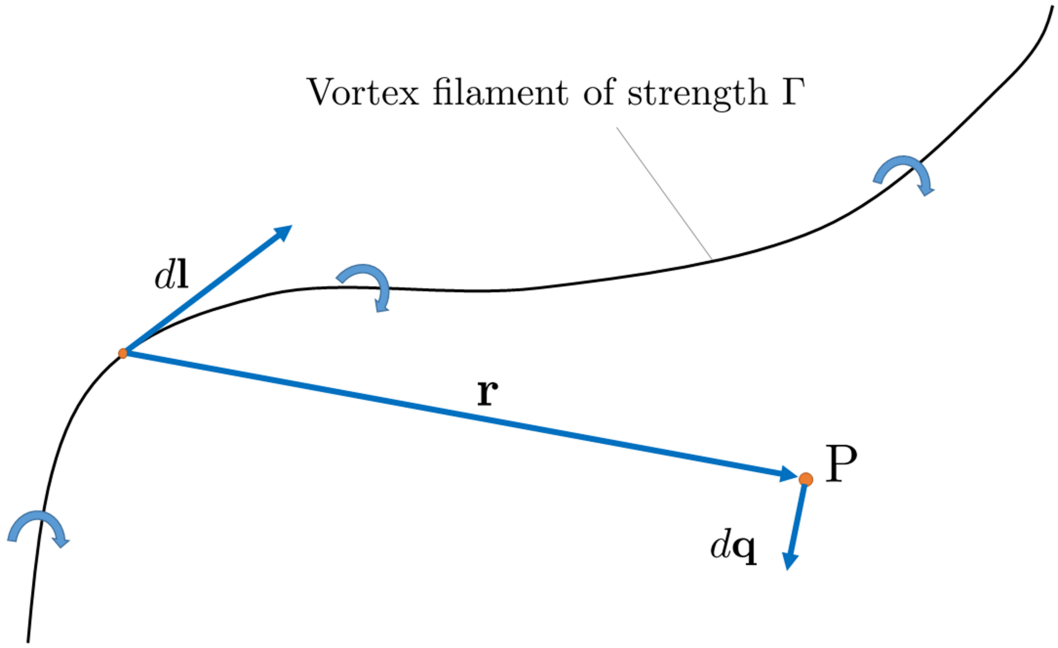

Figure 5.2: Pictorial representation of the Biot-Savart law.

not otherwise "present" in this analysis). Derivation yields

$$q = \frac{\Gamma}{2\pi r}$$

where q is the velocity induced at an arbitrary point P a distance r from the vortex filament of strength Γ. In the MATLAB code, the distance r is defined as r_{ij}, the distance between collocation point i and vortex element j. General formulae for x- and z-components of r_{ij} will be established in the following text.

Intuitively, it's obvious that as the discretization of the chord becomes finer (the number of panels becomes greater), the magnitude of r_{ij} will get very small, which will shoot the value of q to infinity. This result is not physical. To prevent this error, the Rankine vortex

characterization of inviscid flow is implemented.

$$q(r) = \begin{cases} \frac{r\Gamma}{2\pi R^2}, & \text{if } r \leq R \\ \frac{\Gamma}{2\pi r}, & \text{if } r > R \end{cases}$$

Inside the core radius, as shown in figure 5.3, viscosity is modeled to exist; however, beyond

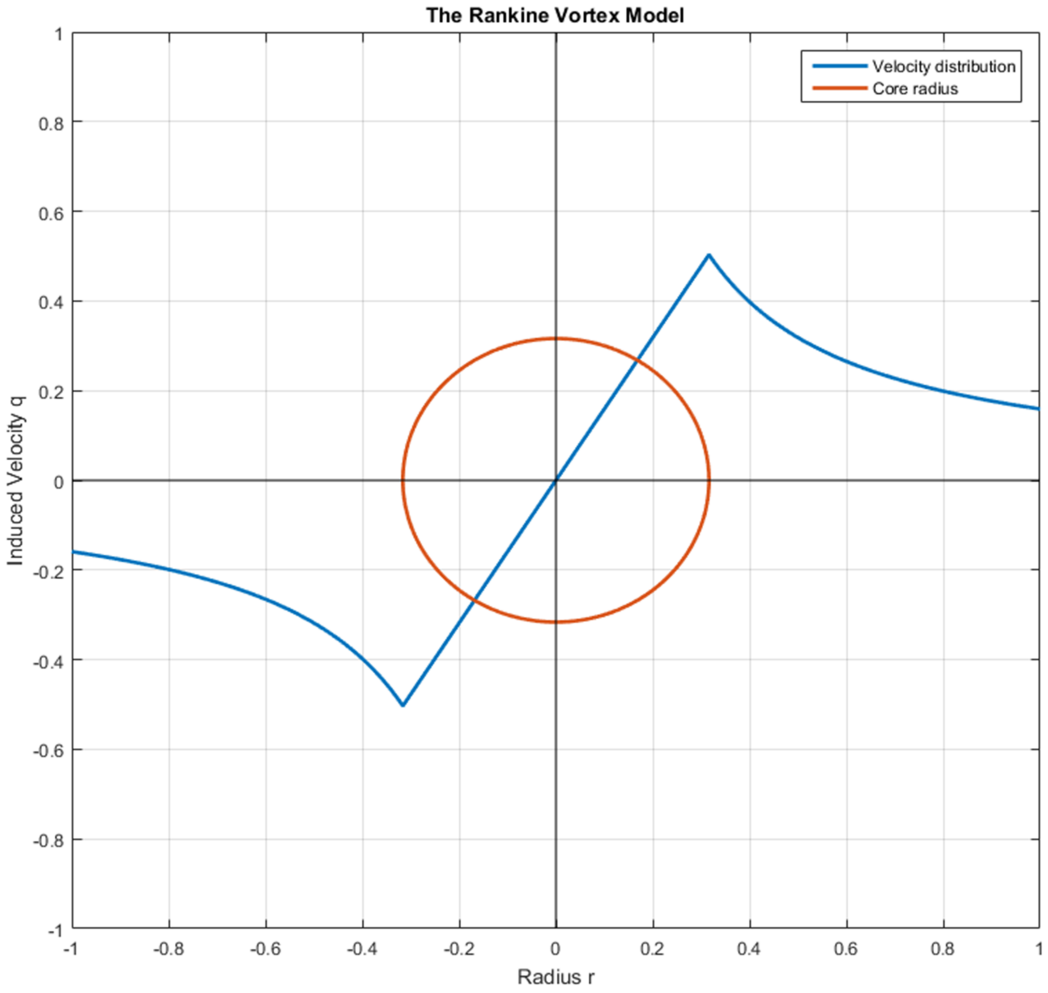

Figure 5.3: A look at the Rankine vortex model.

the core radius the flow is inviscid (i.e. potential flow exists). This combination of viscous and inviscid flow allows for a more correct modeling of potential flow when using the Biot-

Savart law to describe vortex-point interactions.

5.2 Geometry

5.2.1 Time-Invariant Geometry: Lines 99-108

With a necessary tool understood, the geometry of the airfoil is defined. The full chord length of the airfoil is denoted as c, and the flap portion of the chord is denoted as c_f. The ratio between c_f and c is

$$\lambda = \frac{c_f}{c}$$

Therefore, the flat airfoil portion is defined as having a length of $1 - \lambda$. Since the general idea behind the UVLM is the discretization of otherwise continuous quantities in order to numerically compute the output, the length of the airfoil (along with flap in the following section) must be discretized.

The discretization and elements imposed cannot be arbitrary, however. Because the airfoil of concern is thin, no sources are used, while the doublet distribution is approximated by n doublet elements, which is equivalent to n concentrated vortices at the panel edges. With prior knowledge of the geometry of the lumped vortex numerical method, it's clear that placing a vortex at the quarter chord (center of pressure) and a collocation point at the three-quarter chord of the panel automatically satisfies the Kutta condition ($\gamma(c, t) = 0$). Using this knowledge, the equivalent discretized model has the chord split into n panels, with each segment having a vortex element and collocation point at its relative quarter chord and three-quarter chord, respectively.

To find the number of panels found along the airfoil, the length of the airfoil is multiplied

with the total number of panels n along the entire chord.

$$K = (1 - \lambda)n$$

The value of K must be an integer value. There can be no fractions of panels since there can be no fractions of collocation points and vortex elements. (In this particular code, n must be a multiple of four to run.)

With the foundation of the method set, two geometries are now considered: time-invariant and time-varying. The location of the vortex elements $\mathbf{x}_{\text{bound}_j}$ and collocation points $\mathbf{x}_{\text{cp}_i}$ are needed. The time-invariant geometry has the normal vector $\mathbf{n}_i$ = constant with respect

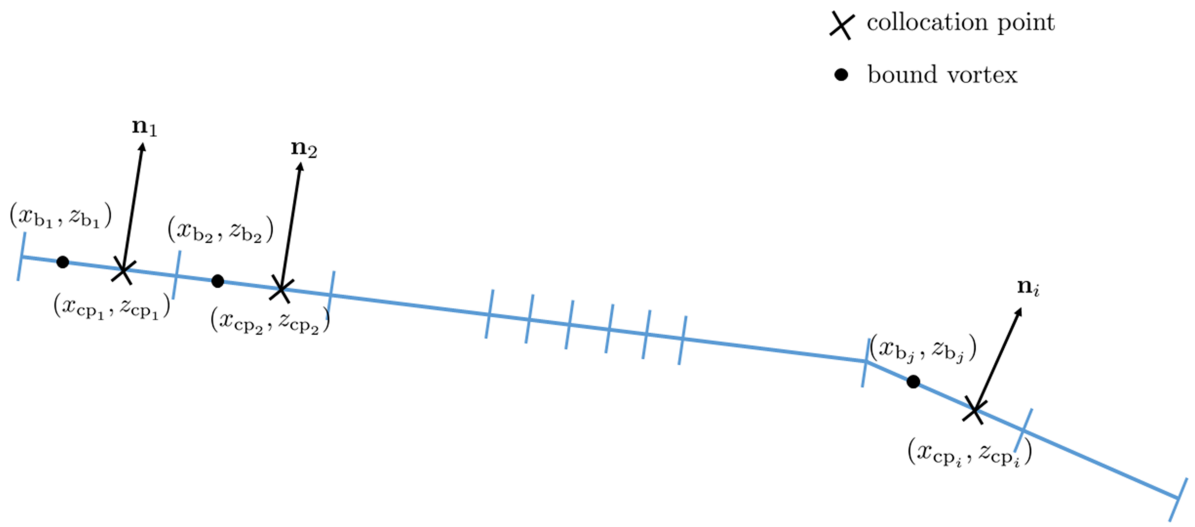

Figure 5.4: Discretization of the airfoil using collocation points.

to the body frame, with the $\mathbf{x}_{\text{bound}_j}$ and $\mathbf{x}_{\text{cp}_i}$ defined as

$$\mathbf{x}_{\text{cp}_i} = (x_{\text{cp}_i}, z_{\text{cp}_i}) = [(i - 1 + 3/4)c/n, 0]$$

$$\mathbf{x}_{\text{bound}_j} = (x_{\text{b}_j}, z_{\text{b}_j}) = [(j - 1 + 1/4)c/n, 0]$$

The z-value is zero here because there is no component of the plate in the z-direction of the body frame for panels $n = 1, 2, ..., K$.

The distance between $\mathbf{x}_{\text{cp}_i}$ and $\mathbf{x}_{\text{bound}_j}$ is

$$\mathbf{r}_{ij} = \mathbf{x}_{\text{cp}_i} - \mathbf{x}_{\text{bound}_j} = (x_{\text{cp}_i}, z_{\text{cp}_i}) - (x_{\text{b}_j}, z_{\text{b}_j})$$

With this, the 2D numerical Biot-Savart equation is constructed.

$$q_i = \sum_{j=1}^{K} \frac{\Gamma_j}{2\pi r_{ij}} \tag{5.1}$$

This is physically the magnitude of the contribution of *all* the vortex elements at the collocation point i. To find the contribution of *all* vortex elements on *all* the collocation points, i is extended to run through the loop iterations of $i = 1, 2, ..., K$.

5.2.2 Time-Varying Geometry: Lines 134-148

The angle δ varies with time, so the geometry formulation is generalized to describe this. The angle (with clockwise being negative) the flap creates with the airfoil is defined as

$$\delta(t) = A_\delta \cos \omega_\delta t$$

where Δ is the amplitude, ω_δ is the angular frequency of the flap motion, and t is the physical time. The varying geometry due to flapping is characterized by

$$\mathbf{n}_i = \begin{pmatrix} \sin\delta(t) \\ -\cos\delta(t) \end{pmatrix}$$

$$\mathbf{x}_{\mathrm{cp}_i} = \begin{pmatrix} c - c_f \\ 0 \end{pmatrix} + (i - K - 1 + 3/4)\frac{c_f}{n-K} \begin{pmatrix} \cos\delta(t) \\ -\sin\delta(t) \end{pmatrix}$$

$$\mathbf{x}_{\mathrm{bound}_j} = \begin{pmatrix} c - c_f \\ 0 \end{pmatrix} + (j - K - 1 + 1/4)\frac{c_f}{n-K} \begin{pmatrix} \cos\delta(t) \\ -\sin\delta(t) \end{pmatrix}$$

with $i = K+1, K+2, ..., n$. This time-varying geometry formulation along with its time-invariant analog gives the positions of all the collocation points and bound vortices, except the starting vortex, which is exactly at the trailing edge of the flap:

$$\mathbf{x}_{\mathrm{TE}} = \begin{pmatrix} c - c_f \\ 0 \end{pmatrix} + c_f \begin{pmatrix} \cos\delta(t) \\ -\sin\delta(t) \end{pmatrix}$$

5.3 Kinematics: Lines 114-142

With the geometry known, the motion of the system is now addressed.

The vertical translational motion, known as plunging or heaving, is described by a sinusoidal function

$$h(t) = H\sin(\omega_h t + \phi_h)$$

where H is the maximum heaving amplitude, ω_h is the heaving angular frequency, ϕ_h is the phase angle of the plunging motion, and t is the physical time; the former two are measured relative to pivot point ab (described later).

Given an airfoil traveling with a velocity $-U_\infty \hat{\mathbf{X}}$, as defined in the inertial and negative X-direction[1], and taking into account the effects of plunging, the resultant velocity $\mathbf{Q}_{\text{plunge}}$, as seen by the airfoil (body frame), is[2]

$$\mathbf{Q}_{\text{plunge}} = \begin{pmatrix} -U_\infty \cos\alpha - \dot{h}\sin\alpha \\ -U_\infty \sin\alpha + \dot{h}\cos\alpha \end{pmatrix}$$

where $\dot{h}$ is the time derivative of the vertical position due to plunging, and α is the angle of attack.

The coupled airfoil pitching is defined as a rotation about a point on the chord c at a distance ab from the midpoint of the chord, where a is the flip axis ratio; therefore, the instantaneous angle of attack (as seen in the airfoil frame) measured counterclockwise from the mean chord is

$$\alpha(t) = \alpha_m + \alpha_0 \sin(\omega_\alpha t + \phi_\alpha)$$

where α_m is the mean pitch angle, ϕ_α is the phase angle of the pitching motion, α_0 is the maximum pitching amplitude, and ω_α is the pitching angular frequency, the latter two of which are taken about the pivot point a.

Taking both the plunging and pitching effects into account, the resultant velocity $\mathbf{Q}_{\text{pp}}$ is defined as

$$\mathbf{Q}_{\text{pp}} = \begin{pmatrix} -U_\infty \cos\alpha - \dot{h}\sin\alpha \\ -U_\infty \sin\alpha + \dot{h}\cos\alpha - \dot{\alpha}\big(x - b(1+a)\big) \end{pmatrix} \tag{5.2}$$

where x is an arbitrary point along the body frame x-axis. In the code for the numerical method, this "arbitrary" point is a collocation point x_{cp} on the camberline.

[1]Notice that the inertial coordinate frame is denoted by capital letters, whereas as the body frame is denoted with lower-case letters.

[2]Note that all vectors in this paper are written in the conventional column vector format, whereas the vectors in the MATLAB code are typed in the row vector format. Be sure to remember this distinction when comparing the theory found in this paper to the code.

Note that when comparing this analytical result to the code, it is obvious that the pitching effect is not present initially; the code implements the pitching effect at a later section. This is due only to convenience, not necessity. It's easier to insert the pitching effect after the influence coefficient matrix a_{ij} has been created.

5.4 Influence Coefficient Matrix (ICM): Lines 150-160

To begin the derivation of the influence coefficient matrix[3] [11], the no-penetration boundary condition is imposed.

$$(\nabla\Phi_B + \nabla\Phi_W - \mathbf{V}_0 - \mathbf{v}_{\text{rel}} - \boldsymbol{\Omega} \times \mathbf{r}) \cdot \mathbf{n} = 0$$

Here, the perturbation potential Φ is divided into the airfoil potential Φ_B and the wake potential Φ_W. Defining the other terms: $\mathbf{V}_0$ is the velocity of the body (airfoil) system's origin relative to the inertial frame, $\mathbf{v}_{\text{rel}}$ is the velocity relative to the body frame, $\mathbf{r}$ is the position vector, $\boldsymbol{\Omega}$ is the rate of rotation of the body's frame of reference, and $\mathbf{n}$ is the vector normal to the airfoil's surface.

The above equation is made into the left-hand side (LHS), which includes the potentials, and the right-hand side (RHS), which includes the kinematic velocity due to the motion of the airfoil and the velocity components induced by the wake.

$$\text{LHS} \equiv (\nabla\Phi_B + \nabla\Phi_W) \cdot \mathbf{n} = -(-\mathbf{V}_0 - \mathbf{v}_{\text{rel}} - \boldsymbol{\Omega} \times \mathbf{r}) \cdot \mathbf{n} \equiv \text{RHS} \tag{5.3}$$

The self-induced portion $(\nabla\Phi_B \cdot \mathbf{n})$ is established by first considering the influence due to a unit strength circulation Γ_j, which yields the influence coefficient a_{ij} in this numerical

[3]The following derivation is taken almost completely from Chapter 9 and 13 of *Low Speed Aerodynamics*, by Katz and Plotkin.

method.

$$a_{ij} = \nabla\Phi_{B_{ij}}|_{(\Gamma=1)} \cdot \mathbf{n}_i$$

This shows $\nabla\Phi_{B_{ij}} = \mathbf{q}_{ij} = (u, w)_{ij}$, which is the velocity component at collocation point i induced by the unit strength singularity element j.

The x- and z-components are determined by the Biot-Savart law as follows:

$$u_{ij} = \frac{\Gamma_j}{2\pi}\frac{z_{\mathrm{cp}_i} - z_{\mathrm{b}_j}}{r_{ij}^2} = \frac{\Gamma_j}{2\pi}\frac{z_{\mathrm{cp}_i} - z_{\mathrm{b}_j}}{(x_{\mathrm{cp}_i} - x_{\mathrm{b}_j})^2 + (z_{\mathrm{cp}_i} - z_{\mathrm{b}_j})^2}$$

$$w_{ij} = \frac{\Gamma_j}{2\pi}\frac{x_{\mathrm{cp}_i} - x_{\mathrm{b}_j}}{r_{ij}^2} = \frac{\Gamma_j}{2\pi}\frac{x_{\mathrm{cp}_i} - x_{\mathrm{b}_j}}{(x_{\mathrm{cp}_i} - x_{\mathrm{b}_j})^2 + (z_{\mathrm{cp}_i} - z_{\mathrm{b}_j})^2}$$

Notice that there is not a negative sign on the w_i value. That's because, as stated before, the positive z-direction points downwards.

As a reminder, since the vortex strengths Γ are unknown (and will be calculated for), $\Gamma = 1$ is taken to determine the induced velocity used in the evaluation of the influence coefficients.

$$u_{ij} = \frac{1}{2\pi}\frac{z_{\mathrm{cp}_i} - z_{\mathrm{b}_j}}{(x_{\mathrm{cp}_i} - x_{\mathrm{b}_j})^2 + (z_{\mathrm{cp}_i} - z_{\mathrm{b}_j})^2}$$

$$w_{ij} = \frac{1}{2\pi}\frac{x_{\mathrm{cp}_i} - x_{\mathrm{b}_j}}{(x_{\mathrm{cp}_i} - x_{\mathrm{b}_j})^2 + (z_{\mathrm{cp}_i} - z_{\mathrm{b}_j})^2}$$

Therefore, the induced velocity created by vortex j on collocation point i by a unit strength Γ is[4]

$$\mathbf{q}_{ij} = \frac{1}{2\pi[(x_{\mathrm{cp}_i} - x_{\mathrm{b}_j})^2 + (z_{\mathrm{cp}_i} - z_{\mathrm{b}_j})^2]}\begin{pmatrix} z_{\mathrm{cp}_i} - z_{\mathrm{b}_j} \\ x_{\mathrm{cp}_i} - x_{\mathrm{b}_j} \end{pmatrix} \tag{5.4}$$

Using this equation, the a_{ij}'s are found.

$$a_{ij} = \mathbf{q}_{ij} \cdot \mathbf{n}_i = \mathbf{q}_{ij}^T \mathbf{n}_i \tag{5.5}$$

[4]The following equation is the equation used in the Biot-Savart function of the MATLAB code.

Equation 5.5 is the general equation for finding the influence coefficients. Variations in geometry must be accounted for.

Next, the wake-induced portion of equation 5.3 is established by taking into account the influence due to a unit strength wake circulation Γ_{W_t}, yielding

$$a_{iw_t} = \nabla\Phi_{W_i} \cdot \mathbf{n}_i = \mathbf{q}_{iw_t}|_{(\Gamma=1)} \cdot \mathbf{n}_i$$

Considering both the self-induced and wake-induced portions results in the equation for the LHS.

$$\text{LHS} \equiv (\nabla\Phi_B + \nabla\Phi_W) \cdot \mathbf{n} = \sum_{j=1}^{n} a_{ij}\Gamma_j + a_{iW_t}\Gamma_{W_t} \tag{5.6}$$

with $i = 1, 2, 3..., n$, and Γ_{W_t} being the strength of the latest vortex. Note that this entire calculation is carried out at a single time t. (In the code, time steps are denoted by the letter k.) In matrix form, the LHS is written as

$$\text{LHS} \equiv \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & a_{1w} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} & a_{2w} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & a_{nW} \\ 1 & 1 & 1 & \dots & 1 & 1 \end{pmatrix} \begin{pmatrix} \Gamma_1 \\ \Gamma_2 \\ \vdots \\ \Gamma_n \\ \Gamma_{W_t} \end{pmatrix} \tag{5.7}$$

The LHS is complete, so the RHS is addressed next. The terms $(-\mathbf{V}_0 - \mathbf{v}_{\text{rel}} - \boldsymbol{\Omega} \times \mathbf{r})$ are replaced with with two groupings of terms: 1) an equivalent tangential and normal velocity $[U(t), W(t)]_i$, representing the kinematic motion of the airfoil; and 2) the velocity components induced by the wake vortices $(u_W, w_W)_i$, not including the velocity induced by the latest vortex.

To complete the RHS, the law of zero total circulation (Kelvin's Circulation Theorem) is

implemented. Referring back to the LHS equation, it's seen that the multiplication of the last row with the column of vortices represents one side of the equation defined as the Kelvin theorem. The mathematical representation of the Kelvin theorem is

$$\Gamma(t) + \Gamma_{W_t} = \Gamma(t - \Delta t) \tag{5.8}$$

where

$$\Gamma(t) = \sum_{j=1}^{n} \Gamma_j$$

resulting in

$$\sum_{j=1}^{n} \Gamma_j + \Gamma_{W_t} = \Gamma(t - \Delta t) \tag{5.9}$$

Comparing this result with 5.6, it's obvious the remaining term to be included on the RHS, in addition to kinematic motion and wake vortex terms, is $\Gamma(t-\Delta t)$, which is the circulation measured at the previous time step. Therefore, the full RHS is

$$\text{RHS} \equiv - \begin{pmatrix} [U(t) + u_W, W(t) + w_W]_1 \cdot \mathbf{n}_1 \\ [U(t) + u_W, W(t) + w_W]_2 \cdot \mathbf{n}_2 \\ \vdots \\ [U(t) + u_W, W(t) + w_W]_n \cdot \mathbf{n}_n \\ -\Gamma(t - \Delta t) \end{pmatrix} \tag{5.10}$$

Equating the 5.7 and 5.10 gives

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & a_{1W} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} & a_{2W} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & a_{nW} \\ 1 & 1 & 1 & \dots & 1 & 1 \end{pmatrix} \begin{pmatrix} \Gamma_1 \\ \Gamma_2 \\ \vdots \\ \Gamma_j \\ \Gamma_{W_t} \end{pmatrix} = - \begin{pmatrix} [U(t)+u_W, W(t)+w_W]_1 \cdot \mathbf{n}_1 \\ [U(t)+u_W, W(t)+w_W]_2 \cdot \mathbf{n}_2 \\ \vdots \\ [U(t)+u_W, W(t)+w_W]_n \cdot \mathbf{n}_n \\ -\Gamma(t-\Delta t) \end{pmatrix} \tag{5.11}$$

5.5 Right Hand Side: Lines 163-179

Though equation 5.11 is established, the entries of the RHS are yet unknown. To determine the values of the entries, the airfoil is viewed as a whole. First, the plunging motion of the airfoil is considered, taking the form

$$\mathbf{Q}_{\text{plunge}} = \begin{pmatrix} -U_\infty \cos\alpha - \dot{h}\sin\alpha \\ -U_\infty \sin\alpha + \dot{h}\cos\alpha \end{pmatrix} \tag{5.12}$$

Pitching is then added, yielding

$$\mathbf{Q}_{\text{pp}} = \begin{pmatrix} -U_\infty \cos\alpha - \dot{h}\sin\alpha \\ -U_\infty \sin\alpha + \dot{h}\cos\alpha - \dot{\alpha}\big(x - b(1+a)\big) \end{pmatrix}$$

where the *pp* subscript denotes pitch-plunge.

By way of reminder, the terms on the RHS are $(-\mathbf{V}_0 - \mathbf{v}_{\text{rel}} - \mathbf{\Omega} \times \mathbf{r})$. These are replaced with the equivalent $[U(t)+u_W, W(t)+w_W]$ terms, which include the kinematic velocities and the wake-induced velocities. The equation for $\mathbf{Q}_{\text{pp}}$ takes into account only the kinematic velocity; wake-induced velocities are added. These two portions of the RHS are denoted as

RHS_{kin} and RHS_W, respectively, where

$$\text{RHS} = \text{RHS}_{\text{kin}} + \text{RHS}_W$$

The RHS_W is evaluated first. Comparing the $U(t)$ and $W(t)$ terms to $\mathbf{Q}_{\text{pp}}$ gives

$$\begin{aligned} U(t) &= -U_\infty \cos\alpha - \dot{h}\sin\alpha \\ W(t) &= -U_\infty \sin\alpha + \dot{h}\cos\alpha - \dot{\alpha}\big(x - b(1+a)\big) \end{aligned}$$

The RHS is found by implementing the boundary condition requiring zero flow normal to the surface of the airfoil. The RHS_{kin}[5] is therefore

$$\begin{aligned} \text{RHS}_{\text{kin}_i} &= -\mathbf{Q}_{\text{pp}_i} \cdot \mathbf{n}_i = -\mathbf{Q}_{\text{pp}_i}^T \mathbf{n}_i \\ &= -\begin{pmatrix} -U_\infty \cos\alpha - \dot{h}\sin\alpha & -U_\infty \sin\alpha + \dot{h}\cos\alpha - \dot{\alpha}\big(x - b(1+a)\big) \end{pmatrix}_i \begin{pmatrix} 0 \\ 1 \end{pmatrix}_i \end{aligned}$$

The RHS_W is found next. The wake-induced effects are generated by each individual wake vortex shed off the trailing edge. At time step $k = 1$, no wake vortex has yet been created, which is the reason the variable `no_Wake` is initially set to zero in the input section of the code; but for time steps $k > 1$, wakes are present, with the quantity of wakes equal to $k - 1$.

To find RHS_W, the Biot-Savart law is applied to find the wake-induced velocity $\mathbf{q}_W$. Notice that a `for` loop with iteration number j is nested within the i loop. This j loop describes the effect each wake vortex has on each collocation point. The velocity induced by a vortex

[5] This is the first RHS term, which is denoted as `RHS(i,1)` in MATLAB.

of strength Γ_W is

$$\mathbf{q}_W = \frac{\Gamma_W}{2\pi} \frac{1}{[(x_{\mathrm{cp}_i} - x_{W_j})^2 + (z_{\mathrm{cp}_i} - z_{W_j})^2]} \begin{pmatrix} z_{\mathrm{cp}_i} - z_{W_j} \\ x_{\mathrm{cp}_i} - x_{W_j} \end{pmatrix} \tag{5.13}$$

where $j = 1, 2, ..., k$, with j denoting the wake vortex number in this instance.

RHS_{W_i} is found by using the familiar relationship

$$\begin{aligned} \mathrm{RHS}_{W_i} &= -\mathbf{q}_{W_i} \cdot \mathbf{n}_i = -\mathbf{q}_{W_i}^T \mathbf{n}_i \\ &= -\frac{\Gamma_W}{2\pi} \frac{1}{[(x_{\mathrm{cp}_i} - x_{W_j})^2 + (z_{\mathrm{cp}_i} - z_{W_j})^2]} \begin{pmatrix} z_{\mathrm{cp}_i} - z_{W_j} & x_{\mathrm{cp}_i} - x_{W_j} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix}_i \end{aligned}$$

The RHS is computed by summing the kinematic RHS with the wake-induced RHS:

$$\mathrm{RHS}_i = \mathrm{RHS}_{\mathrm{kin}_i} + \mathrm{RHS}_{W_i} \tag{5.14}$$

This is done on the final line of the nested j loop. The j loop is then exited, followed by the exiting of the i loop.

5.6 Solving the System: Lines 182-234

The main point of this section is to solve for the previously unknown Γ's and to find the C_L and C_T generated. That requires the pressure coefficient to first be found.

5.6.1 Pressure Coefficient C_p

The first line of code defines

$$\mathbf{\Gamma}_{\text{old}} = \mathbf{\Gamma}$$

where $\mathbf{\Gamma}$ is the vector of vortex strengths at time step k and $\mathbf{\Gamma}_{\text{old}}$ is the vector of vortex strengths from time step $k-1$.

The second line of code defines

$$\mathbf{\Gamma} = \mathbf{A}^{-1}\text{RHS} \tag{5.15}$$

where $\mathbf{A}$ is the influence coefficient matrix and RHS is the right-hand side solved for in previous sections. Perhaps an easier way to picture the above equation is in indicial form:

$$\Gamma_j = \sum_{j=1}^{n+1} a_{ij}^{-1}\text{RHS}_i \tag{5.16}$$

where a_{ij}^{-1} are the coefficients of the inverted matrix.

The important thing to notice is the order by which $\mathbf{\Gamma}_{\text{old}}$ and $\mathbf{\Gamma}$ are defined. First, $\mathbf{\Gamma}_{\text{old}}$ is defined; second, $\mathbf{\Gamma}$ is defined. The reason is due to the fact that $\mathbf{\Gamma}_{\text{old}}$ is desired to be based off the previous time step.

The $k-1$ vortex strength vector $\mathbf{\Gamma}_{\text{old}}$ can only be found if at least one full time step has already occurred, otherwise $\mathbf{\Gamma}$ cannot be calculated; $\mathbf{\Gamma}_{\text{old}}$ hinges on the fact that $\mathbf{\Gamma}$ has been calculated in the previous time step.

The next two lines of code set two values equal to zero.

$$\sum_{j=1}^{n+1} \Gamma_j = 0 \tag{5.17}$$

$$\sum_{j=1}^{n+1} \Gamma_{\text{old}_j} = 0 \tag{5.18}$$

These values are set to zero so that every time step sees them start at the same point of initialization in preparation for the `for` loop that follows.

The `for` loop initiates. This loop runs through the effects occurring on each panel, summing the effects from the current panel i with all previous iterations, ending only when the loops have run the length of the entire chord through the completion of nth loop. Equations (29) and (30) begin the loop, starting from no entries, with each subsequent loop adding the ith entry of the respective $\mathbf{\Gamma}$ and $\mathbf{\Gamma}_{\text{old}}$ vectors to the scalar sum.

A nested p loop is begun within the i loop. This p loop describes the velocity induced by the sum of the wake vortices generated at the passing of each time step. The Biot-Savart is used.

$$\mathbf{q}_{W_k} = \sum_{p=1}^{k-1} \mathbf{q}_{W_p} + \frac{\Gamma_{W_k}}{2\pi} \frac{1}{[(x_{\text{cp}_i} - x_{W_k})^2 + (z_{\text{cp}_i} - z_{W_k})^2]} \begin{pmatrix} z_{\text{cp}_i} - z_{W_k} \\ x_{\text{cp}_i} - x_{W_k} \end{pmatrix} \tag{5.19}$$

The first term describes the sum of the velocities induced from all previous time steps $p = 1, ..., k-1$, whereas the second term describes the velocity induced at the current time step k. (Note that the effects of this loop are noticed only after wakes have been generated, which occurs only after the first time step.)

Stepping out of the p loop, the next evaluation within the i loop begins. The kinematic velocity $\mathbf{Q}_{\text{pp}}$ is defined in the same way it was previously, but here it's done for a different purpose: here it's used to find $\mathbf{Q}_{\text{tot}}$, the total velocity as seen in the body frame.

$$\mathbf{Q}_{\text{tot}} = \mathbf{q}_{W_k} - \mathbf{Q}_{\text{pp}} \tag{5.20}$$

This $\mathbf{Q}_{\text{tot}}$ is needed in order to compute the resultant pressures and loads.

To find the pressure difference between upper and lower surfaces of the camberline near the

surface of the airfoil, the instantaneous Bernoulli equation is manipulated, resulting in[6]

$$\Delta p = p_l - p_u = \rho \left[\left(\frac{Q_\tau^2}{2} + \frac{\partial \Phi}{\partial t} \right)_u - \left(\frac{Q_\tau^2}{2} + \frac{\partial \Phi}{\partial t} \right)_l \right] \tag{5.21}$$

where Q_τ is the tangential velocity along the surface, $\partial\Phi/\partial t$ is the time derivative of the velocity potential, and the subscripts u and l denote the upper and lower surfaces, respectively. Before proceeding in finding the pressures, the Q_τ and $\partial\Phi/\partial t$ terms are defined.

Q_{τ_j} is defined as

$$Q_{\tau_j} = \mathbf{Q}_{\mathrm{tot}_j} \cdot \tau_j \pm \frac{\partial \Phi}{\partial \tau_j} \tag{5.22}$$

where $\partial\Phi/\partial\tau_j$ is the tangential derivative of the thin airfoil, the $\pm$ sign stands for the upper and lower surfaces, respectively, and the subscript j denotes the iteration number of the current panel, with $j = 1, ..., n$. The tangential derivative of equation 5.22 can be approximated as

$$\pm \frac{\partial \Phi}{\partial \tau_j} = \pm \frac{\gamma}{2} \approx \pm \frac{\Gamma_j}{2\Delta l_j}$$

where γ is the vortex strength per unit length and Δl_j is the jth panel length, which is c/n. Therefore, the tangential derivative is used in the form of

$$\pm \frac{\partial \Phi}{\partial \tau_j} \approx \pm \frac{\Gamma_j}{2(c/n)} \tag{5.23}$$

The tangential velocity at the jth panel is therefore

$$Q_{\tau_j} \approx \mathbf{Q}_{\mathrm{tot}_j} \cdot \tau_j \pm \frac{\Gamma_j}{2(c/n)} \tag{5.24}$$

With Q_{τ_j} defined[7], the next step is to define the $\partial\Phi/\partial t$ term of the instantaneous unsteady

[6]The following derivations, as noted for previous derivations, follow those outlined in Chapter 13 of *Low Speed Aerodynamics*, by Katz and Plotkin

[7]Notice that the code doesn't include the $\partial\Phi/\partial\tau$ term. For convenience, this term will be "ignored" here but added into the code's pressure term. No mathematical liberties are taken because of this; it is merely for coding convenience.

Bernoulli equation.

To find this velocity potential time derivative, a more fundamental definition is referenced first: the velocity potential.

$$\Phi^{\pm} = \pm \int \frac{\partial \Phi}{\partial \tau} d\tau$$

Applying the same approximation for the tangential derivative as used previously results in

$$\Phi^{\pm} = \pm \int \frac{\gamma}{2} d\tau = \pm \frac{1}{2} \int \frac{\Gamma}{d\tau} d\tau$$

Therefore, the velocity potential time derivative is

$$\pm \frac{\partial \Phi}{\partial t} = \pm \frac{\partial}{\partial t} \Big(\frac{1}{2} \int \frac{\Gamma}{d\tau} d\tau \Big)$$

which, for a numerical method, takes the form

$$\pm \frac{\partial \Phi}{\partial t} = \pm \frac{1}{2} \frac{\partial}{\partial t} \sum_{h=1}^{j} \Gamma_h \tag{5.25}$$

This equation defines the local potential as the vortices are summed from the leading edge to the jth vortex element along the camberline.

Substituting equations 5.22 and 5.24 into 5.21 gives

$$\Delta p_j = \rho \Bigg(\Bigg[\frac{1}{2} \Big([U(t) + u_W, W(t) + w_W]_j \cdot \tau_j + \frac{\Gamma_j}{2(c/n)} \Big)^2 + \frac{1}{2} \frac{\partial}{\partial t} \sum_{h=1}^{j} \Gamma_h \Bigg] - \Bigg[\frac{1}{2} \Big([U(t) + u_W, W(t) + w_W]_j \cdot \tau_j - \frac{\Gamma_j}{2(c/n)} \Big)^2 - \frac{1}{2} \frac{\partial}{\partial t} \sum_{h=1}^{j} \Gamma_h \Bigg] \Bigg)$$

which, when simplified, results in

$$\Delta p_j = \rho \Bigg([U(t) + u_W, W(t) + w_W]_j \cdot \tau_j \frac{\Gamma_j}{(c/n)} + \frac{\partial}{\partial t} \sum_{h=1}^{j} \Gamma_h \Bigg) \tag{5.26}$$

To determine the numerical form of the partial derivative, consider the definition of the standard derivative.

$$\frac{d\Gamma(t)}{dt} = \lim_{\Delta t \to 0} \frac{\Gamma(t) - \Gamma(t - \Delta t)}{\Delta t}$$

In a numerical method, the Δt doesn't tend to zero but is rather a discrete finite number, which will be kept in its denoted form of Δt; and, since Γ is a function of time, the partial derivative is the same as the standard derivative. The resultant numerical approximation is therefore

$$\frac{\partial}{\partial t} \sum_{h=1}^{j} \Gamma_h = \frac{1}{\Delta t} \sum_{h=1}^{j} \Big[\Gamma_{h,k} - \Gamma_{h,k-1} \Big] \tag{5.27}$$

where k denotes the current time step of evaluation. Plugging the above equation into the pressure equation yields the following pressure coefficient:

$$\Delta C_{p_j} = \frac{2}{U_\infty^2} \left([U(t) + u_W, W(t) + w_W]_j \cdot \tau_j \frac{\Gamma_j}{(c/n)} + \frac{1}{\Delta t} \sum_{h=1}^{j} \Big[\Gamma_{h,k} - \Gamma_{h,k-1} \Big] \right) \tag{5.28}$$

This is the equation used in the code to find the pressure difference between the upper and lower surfaces.

5.6.2 Lift Coefficient C_L

The lift is determined using the pressure equation, since lift is simply the force in the upward direction generated by the pressure difference. For added accuracy in the lift prediction, the suction force calculated by Garrick and reformulated by Ramesh is added onto the lift value obtained by the pressure difference using equation (4.6). Calculation is split into two portions due to there being two sections of the airfoil-flap system: 1) the airfoil; 2) the flap.

1. For the airfoil section ($j \leq K$), only the pitch angle must be taken into account in

order to rectify the force into the inertial frame.

$$C_L = \sum_{j=1}^{K} \Delta C_{p_j} \frac{c}{n} \cos\alpha_k + C_{F_s} \sin\alpha_k \tag{5.29}$$

2. The second section consists of the flap ($j > K$), which means both the pitch angle $\alpha(t)$ and the flap angle $\delta(t)$ are considered in order to accurately describe the lift.

$$C_L = \sum_{j=K}^{n} \Delta C_{p_j} \frac{c}{n} \cos(\alpha_k + \delta_k) + C_{F_s} \sin\alpha_k \tag{5.30}$$

These two portions of lift are handled by an `if` loop based on the iteration `j`.

5.7 Convection: Lines 237-270

The first `for` loop of this section generates the wake vector $\mathbf{\Gamma}_W$, with the number of entries equal to the number of wake vortices from all previous time steps $k-1$.

The number of wake vortices is equal to the number of time steps minus one.

$$n_W = k - 1 \tag{5.31}$$

The reason the number of vortices is not equal to the number of times steps is because unsteady flow doesn't elicit an instantaneous response. In other words, at $k = 1$, when the airfoil motion initiates, the starting vortex is not *instantly* developed in its entirety. Only after the first increment of time has passed is a starting vortex fully generated, and only then are its effects accounted for. The MATLAB code is written to take into account this lag.

The purpose of the loop is the generation of a vector of the form

$$\mathbf{\Gamma}_W = \begin{pmatrix} \Gamma_{W_k} \\ \Gamma_{W_{k-1}} \\ \vdots \\ \Gamma_{W_2} \\ \Gamma_{W_1} \end{pmatrix}$$

The code creates the vector by doing the following:

$$\mathbf{\Gamma}_W|_{k=1} = \begin{pmatrix} \Gamma_{W_1} \end{pmatrix} \longrightarrow \mathbf{\Gamma}_W|_{k=2} = \begin{pmatrix} \Gamma_{W_2} \\ \Gamma_{W_1} \end{pmatrix} \longrightarrow \ldots \longrightarrow \mathbf{\Gamma}_W = \begin{pmatrix} \Gamma_{W_k} \\ \Gamma_{W_{k-1}} \\ \vdots \\ \Gamma_{W_2} \\ \Gamma_{W_1} \end{pmatrix}$$

where each arrow signifies a successful run through an entire time loop. This process continues until the total number of time steps are completed.

The strength of the most recent wake vortex Γ_{W_k} is unknown and is evaluated by the Kelvin circulation theorem. The strength of the other wake vortices is known from previous time steps and their effect on the normal velocity will be transferred to the right-hand side, along with velocity components induced by the kinematic motion.

The lingering unknown is the most recent wake vortex.

Γ_{W_k} is solved for setting it equal to the $(n+1)$th entry of the column of Γ's, which is then evaluated, as are all the Γ's, by

$$\mathbf{\Gamma} = \mathbf{A}^{-1}\text{RHS}$$

as the code runs through the next time step.

For emphasis, it's best to state again that the most recent vortex cannot be known *at the current time step*. For example, Γ_{W_k} cannot be known at time step k, but it can be known at time step $k+1$. Another way to say it is: at time step k, only wake vortices up to $\Gamma_{W_{k-1}}$ can be known.

The next line of code: a summation of the entries of the wake vortex $\mathbf{\Gamma}_W$. Notice the placement of this line. It comes near the end of the time loop, at a point after influence coefficient matrix, after the right-hand side, and the lift have been calculated. It comes *after* the prominent calculations have been completed. Therefore, in essence, this sum collects all the wake vortices from the *previous* times steps.

For this concept to avoid confusion, actual placement of this line of code within the MATLAB file should be ignored. Even though its appearance and evaluation is in the current time step k, its purpose is not implemented until the following time step $k+1$; so, in effect, it is the sum of the wake circulation from previous time steps, which is the following:

$$\sum_{h=1}^{k-1} \Gamma_{W_h} = \Gamma(t - \Delta t)$$

When placed into what has already been written as the Kelvin theorem, the result is

$$\sum_{j=1}^{n} \Gamma_j + \Gamma_{W_k} = \sum_{h=1}^{k-1} \Gamma_{W_h} \tag{5.32}$$

with the only unknown being Γ_{W_k} (which, again, will be solved for at $k+1$).

The method for finding the strengths of the vortices is now defined, but that's useless until the positions of those vortices are known. Remember, the velocity induced on the airfoil by the vortices obeys the Biot-Savart law, a law that depends on the distance between a vortex and a point.

The next line of code (skipping over the already-defined n_W) gives an array filled with the

positions of the wakes vortices.

$$\mathbf{x}_W = \begin{pmatrix} x_{W_k} & x_{W_{k-1}} & \dots & x_{W_2} & x_{W_1} \\ z_{W_k} & z_{W_{k-1}} & \dots & z_{W_2} & z_{W_1} \end{pmatrix} \tag{5.33}$$

where $\mathbf{x}_{W_k} = \mathbf{x}_{\mathrm{b}_{n+1}}$. This line takes all the wakes vortices from the previous time steps and adds another entry corresponding to the wake vortex generated at the current time step. Note that although the strength of the most recent vortex is *not* known, the position *is* known; it always originates at the trailing edge of the airfoil.

Next, another `for` loop is created, which accounts for two things: 1) the translational velocity of the wake vortices relative to the body frame; 2) the distance between the airfoil and the wake vortices.

1. The first line in this loop re-equates the overall airfoil velocity with the velocity of pure plunging motion, effectively removing the pitching term. The second line adds in a new pitching term

   $$-\dot{\alpha}\big(x_W - b(1+a)\big)$$

   which is with respect to the positions of the wake vortices. Together, these two lines give the velocity of the wake vortices relative to the body frame.

2. The final line of code in this loop defines the resultant positions of all the wake vortices relative to the body frame after the cycle of the current time step k. The equation is

   $$\mathbf{x}_{W_{\mathrm{new}}} = \mathbf{x}_W - \mathbf{Q}_{\mathrm{pp}}\Delta t \tag{5.34}$$

   where Δt is the change in time from the previous time step $k-1$ to the current step k.

There are two loops left in the code, both of which use the Biot-Savart law.

The first loop describes the contribution of the velocity induced by the bound vortices on the wake vortices. The equation takes the following form:

$$\mathbf{q}_{W_i,\mathrm{b}_j} = \frac{\Gamma_j}{2\pi} \frac{1}{[(x_{W_i} - x_{\mathrm{b}_j})^2 + (z_{W_i} - z_{\mathrm{b}_j})^2]} \begin{pmatrix} z_{W_i} - z_{\mathrm{b}_j} \\ x_{W_i} - x_{\mathrm{b}_j} \end{pmatrix} \tag{5.35}$$

where the subscript attached the to the $\mathbf{q}$ term denoting the bound-on-wake vortex effect. To get the new position of the wake vortices, the following is implemented:

$$\mathbf{x}_{W_{\mathrm{new}}} = \mathbf{x}_{W_{\mathrm{new}}} + \mathbf{q}_{W_i,\mathrm{b}_j} \Delta t$$

which yields a wake vortex position that takes into account the induced velocity effects of the bound vortices on the wake vortices.

The second loop describes the contribution of the velocity induced by the wake vortices *on* the wake vortices.

$$\mathbf{q}_{W_i,W_j} = \frac{\Gamma_j}{2\pi} \frac{1}{[(x_{W_i} - x_{W_j})^2 + (z_{W_i} - z_{W_j})^2]} \begin{pmatrix} z_{W_i} - z_{W_j} \\ x_{W_i} - x_{W_j} \end{pmatrix} \tag{5.36}$$

where the subscript attached to the $\mathbf{q}$ term denotes the wake-on-wake vortex effect. To get the new "final" position of the wake vortices within the current time loop, the same method is implemented.

$$\mathbf{x}_{W_{\mathrm{new}}} = \mathbf{x}_{W_{\mathrm{new}}} + \mathbf{q}_{W_i,W_j} \Delta t$$

With both bound-on-wake and wake-on-wake effects taken into account, the final wake vortex vector $\mathbf{x}_{W_{\mathrm{new}}}$ is known, which describes the position of every wake vortex relative to the body frame at the completion of the current time step.

One more thing remains. In order to properly take advantage of MATLAB's iterative abilities, the final wake vortex vector $\mathbf{x}_{W_{\mathrm{new}}}$ is redefined as the initial wake vortex vector $\mathbf{x}_W$

before sending it into the $(k+1)$th time step.

$$\mathbf{x}_{W_{\text{new}}} = \mathbf{x}_W$$

The code is now complete, and the theory behind it should be clear.

Chapter 6

Results

As a brief introduction to this chapter, let it be noted that the results obtained are for three strict cases: 1) plunging with amplitude $H = 0.02b$[1]; 2) pitching with amplitude $|\alpha| = 3^\circ$ and pitch axis location at the quarter chord ($a = -0.25$); and 3) flapping with amplitude $|\delta| = 5^\circ$ and with hinge axis at the three-quarter chord ($e = 0.5$).

6.1 Verification

In order to validate a numerical method, it is compared to analytical or experimental data. Since only analytical data is addressed here, the UVLM is contrained to be checked against mathematics.

However, before anything is validated, the analytical method is first verified to have been coded properly by the author. Theodorsen's paper does not contain a plot illustrating lift

[1]For the plunging case constrained by the parameter $H = 0.02b$, the effective angle of attack becomes greater than 5° at $k > 4.3$, which is the healthy angle limit of the methods based on thin airfoil theory. However, for the sake of equal comparison, the plunging case was tested all the way through $k = 10$, as were pitching and flapping.

versus, while Leishman's does. That said, how are both methods to be verified if only one is represented graphically?

It is good to remember that the difference between Theordorsen's and Leishman's formulations of lift is that the former is based on frequency response and the latter is based on indicial (step) response. Since Theordorsen's function and Wagner's function (used in Leishman's derivation) are related by means of a Fourier transform pair, the resulting lift from both methods will be almost exactly the same. Therefore, for the accuracy required in this paper, verification of Leishman's data is, by extension, verification of Theodorsen's.

The below figure shows the data digitized directly from Leishman's paper. On top of the digitized data are the plots generated in MATLAB. Since Leishman's method requires direct integration, the author makes use of MATLAB's built-in ordinary differential equation solver `ode45`, which implements the fourth/fifth-order Runge-Kutta method and therefore has variable step size. (Note that the state equations are integrated with respect to time.)

In figure 6.1[2], the dotted lines denote the digitized data and the solid lines denote the results outputted from the author's code. As can be seen, both align nearly on top of each other[3]. Though there are slight differences in phase, the discrepancies are due to the inaccuracies inherent to the digitization process when using the `GRABIT` downloadable MATLAB function. Such errors are well within the tolerable range; therefore, the code of the analytical formulation is verified to accurately represent Theodorsen's and Leishman's expressions.

[2]A nonlinear fit function `nlinfit` is used in MATLAB to produce the smooth plots seen here. Without the fit, the `ode45` function results in a jagged shape, likely due to its built-in time interval scheme.

[3]The code based on Theodorsen's formulation s used for $k = 0.01$. The reason is that Leishman's formulation does not give an accurate result. More on this is presented in the next section.

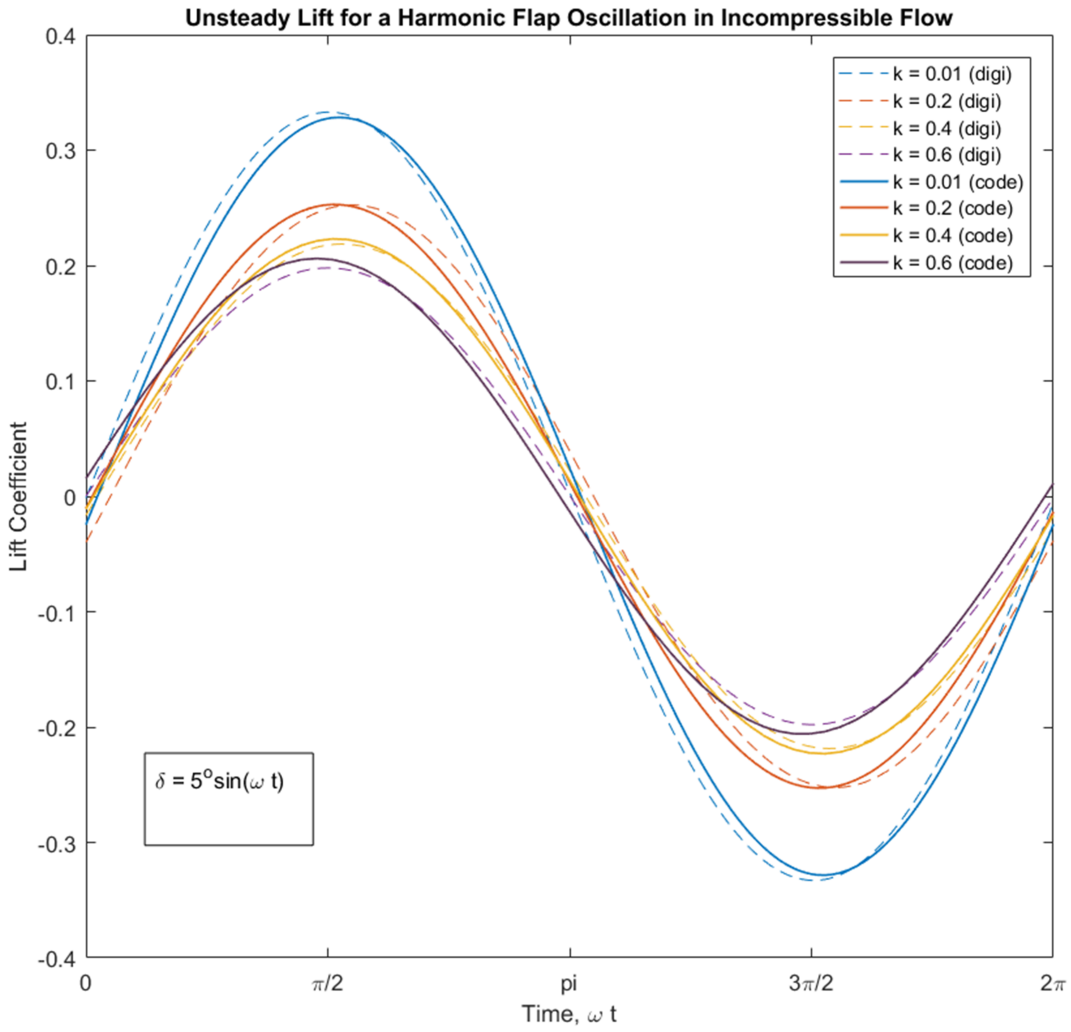

Figure 6.1: Lift coefficient versus time.

6.2 Lift Shape Validation

As a means of validating the UVLM, plots of lift versus time for all methods are shown in this section for three different motion and for four different reduced frequencies: 0.01,0.2,0.4,0.6. There are twelve plots in total.

Firstly, to help the reader better picture what is being discussed, figure **??** is presented.

With the image in mind, three subsections are created to better organize the presentation of the twelve plots .

6.2.1 Plunging

The plots are presented in order of increasing frequency, with the final plot in this subsection being the exception. At $k = 0.01$, the amplitude is too small to allow much analysis on the scale that 6.3 is held to. The scale is set for comparative purposes only.

Though the scale is useful for comparison of frequencies, it is not useful for comparison of methods at the lowest frequecy. Therefore, figure 6.7 is created, which is the same plot as figure 6.3 but at a different scale. The lift behavior at plunging frequency $k = 0.01$ can be discerned at this level.

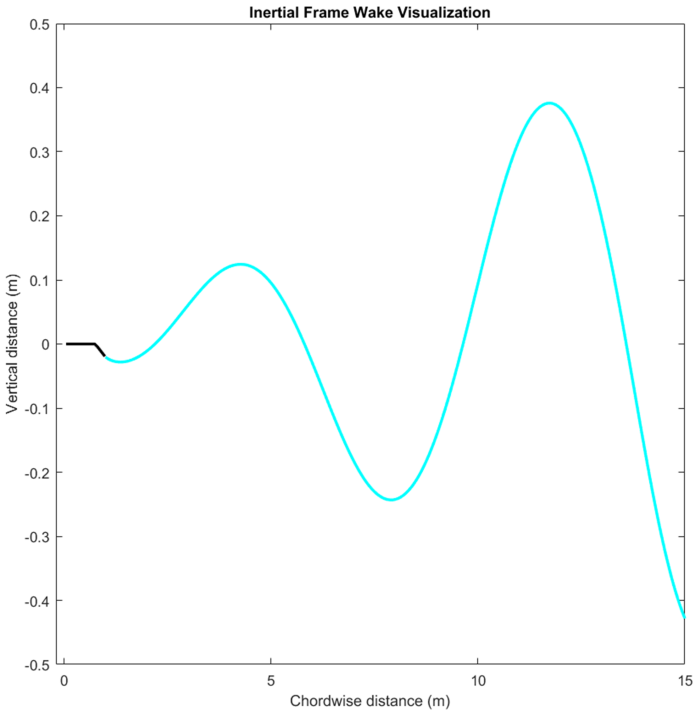

Figure 6.2: A 2D visualization of an oscillating flap deflection. The black is the flat plate with a flap; the cyan is the wake convection. This image is from the UVLM mid-run.

As is evident from figure 6.7, Leishman's formulation does not seem to accurately predict the lift at such low frequencies. This is an oddity. The reader is challenged to look into the author's code (found in the appendix) to find a mistake. Leishman's paper seems to show accurate lift predictions, though the code based on his paper does not—for extremely *low* frequencies, that is.

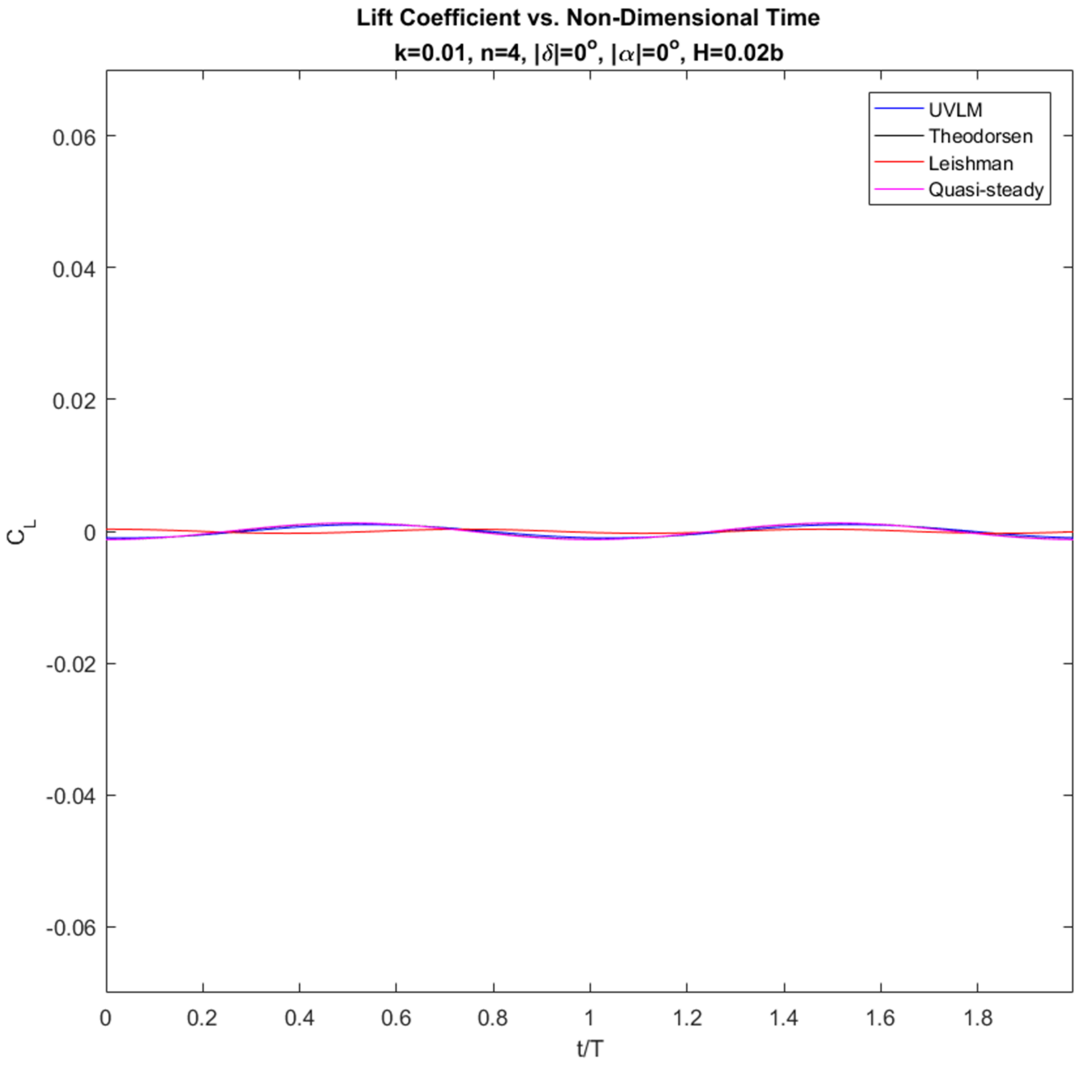

Figure 6.3: Plunging for $k = 0.01$.

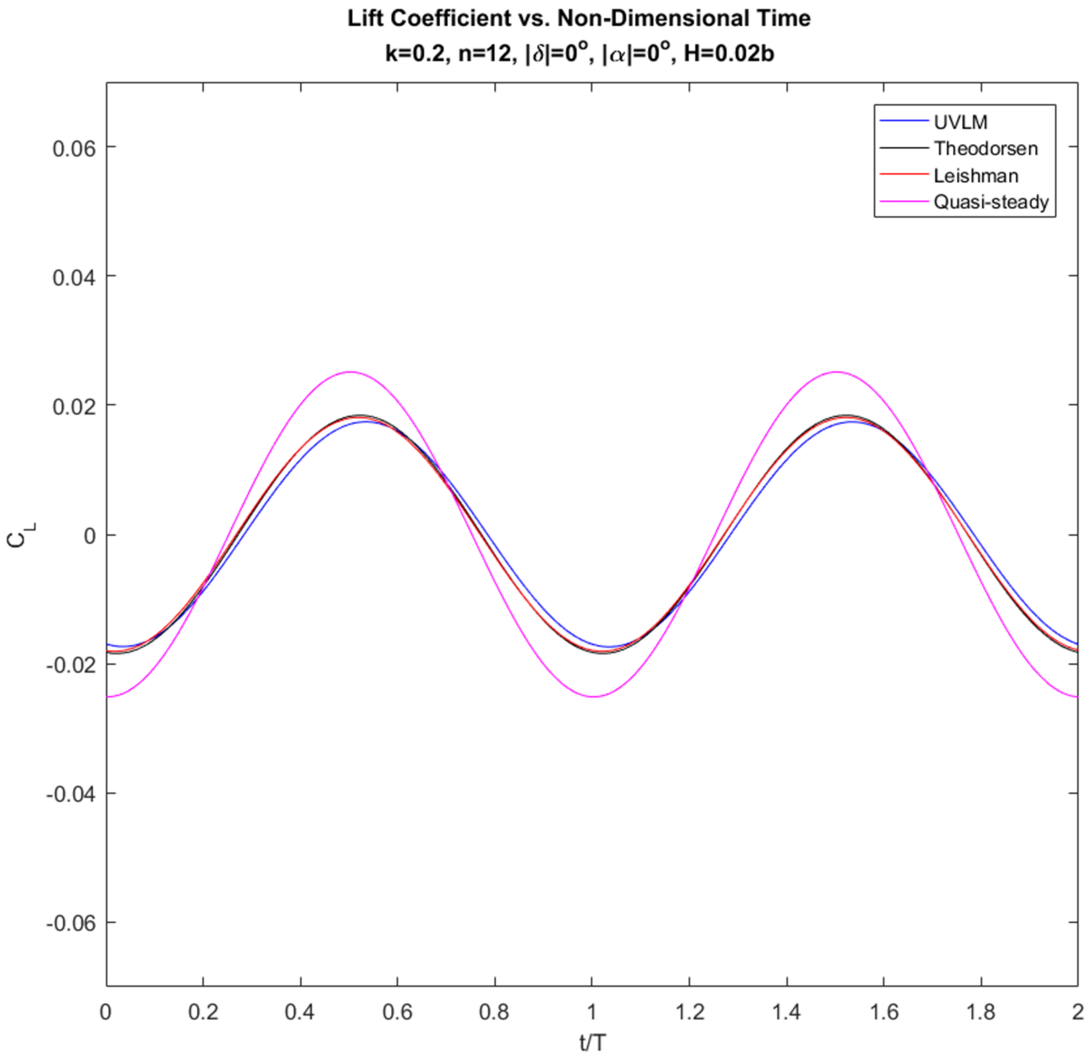

Figure 6.4: Plunging for $k = 0.2$.

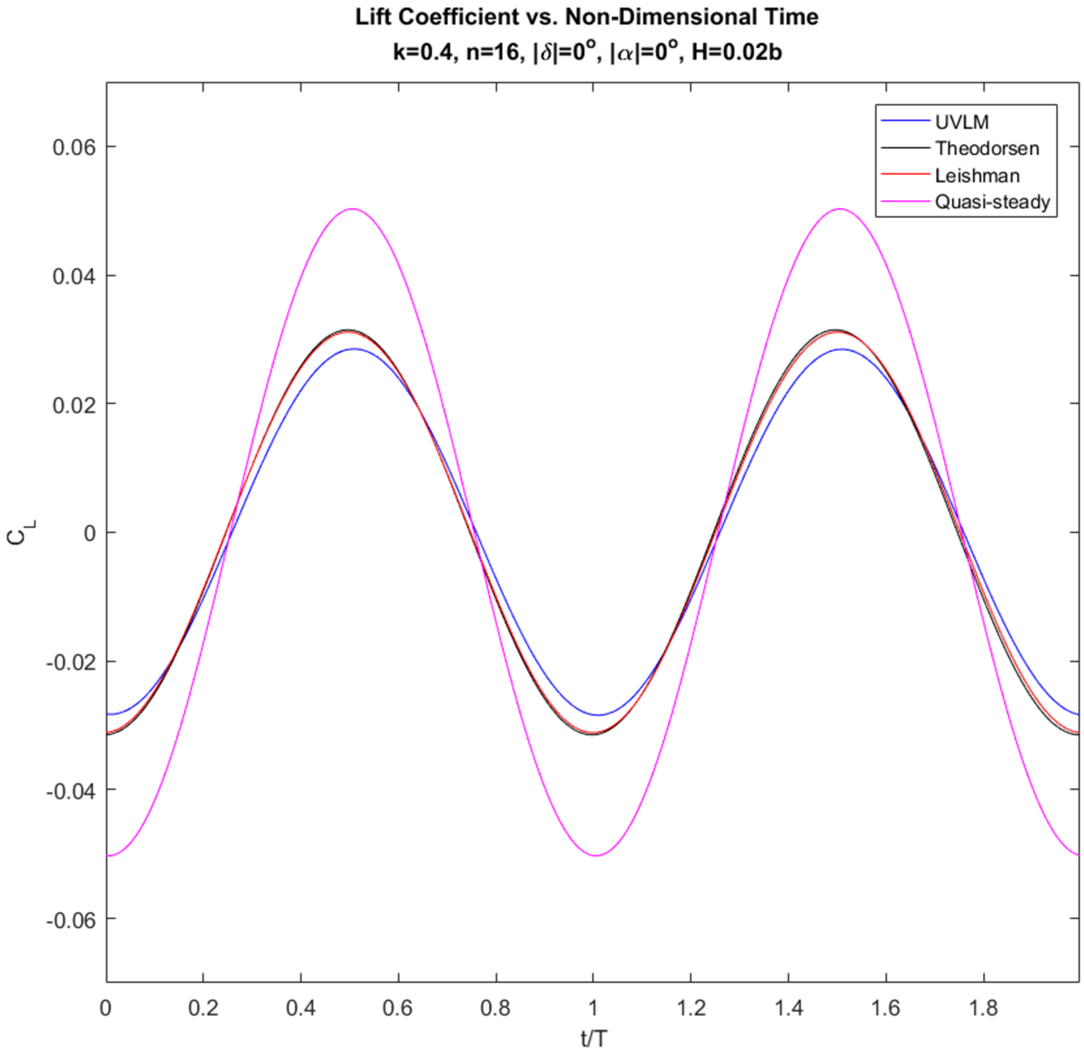

Figure 6.5: Plunging for $k = 0.4$.

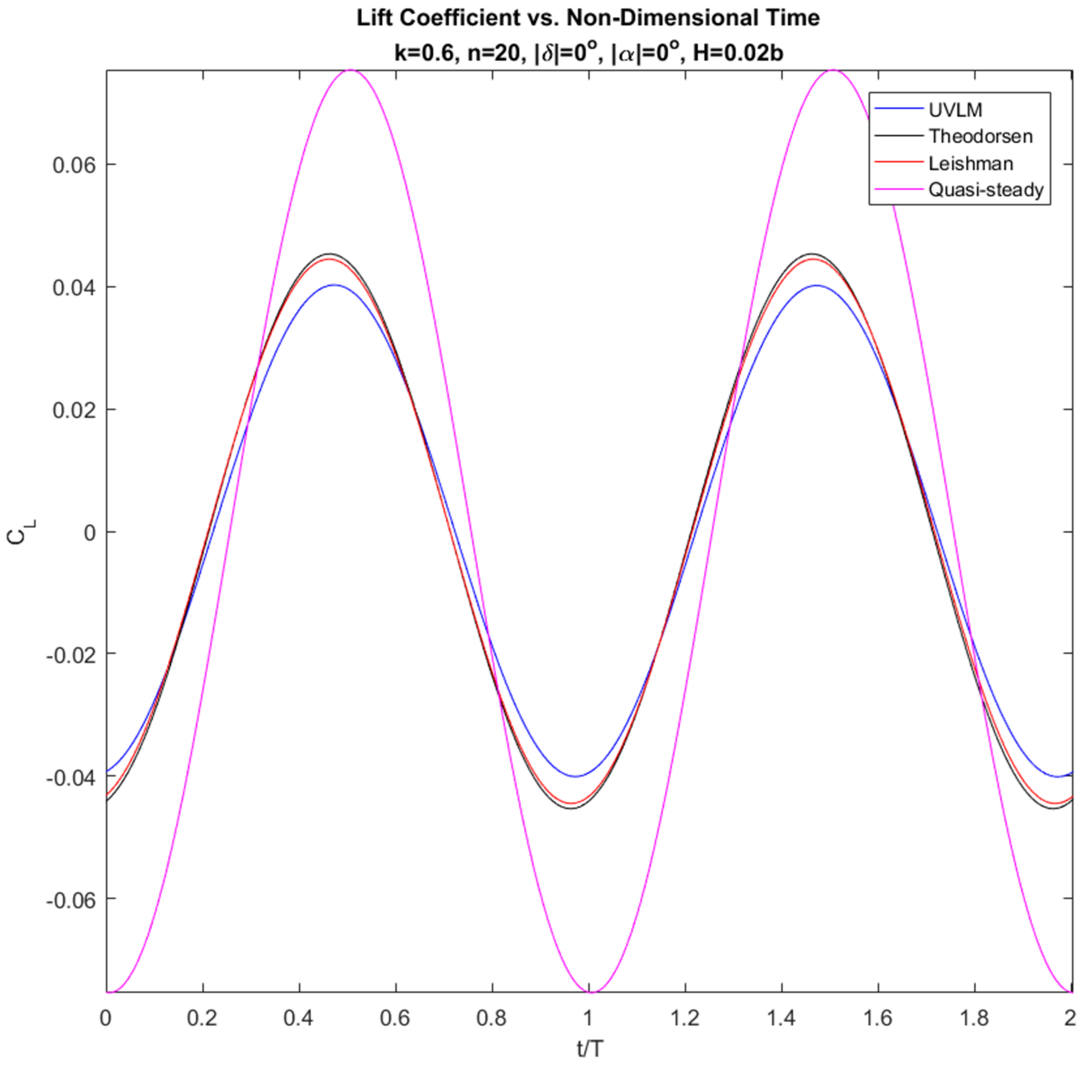

Figure 6.6: Plunging for $k = 0.6$.

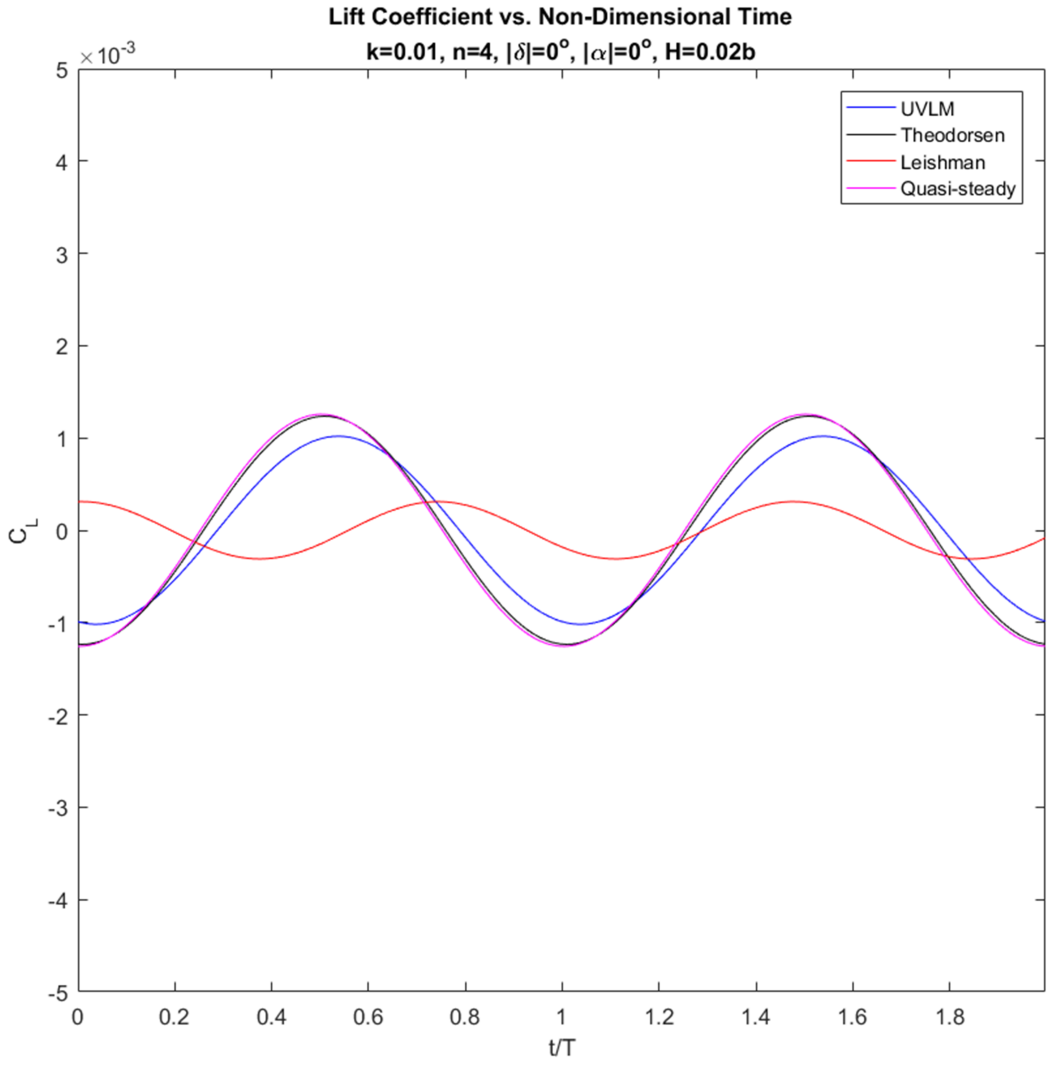

Figure 6.7: Zoomed in case of plunging at $k = 0.01$ to show erroneous results of Leishman's formulation.

6.2.2 Pitching

For pitching, there is an interesting relation that can be gleaned by viewing figures 6.8-6.11. Initially, there is an inversely proportional relation between lift and pitch frequency, but this exchange holds only until $k \approx 5$. At $k > 5$, a *proportional* relation emerges. This pattern is noticed and occurs at the transition somewhere between $k = 0.4$ and $k = 0.6$, as seen in figures 6.10 and 6.11, respectively. [4]

[4]More clear evidence of this relation is found by plotting the lift amplitude versus the frequency, as done in a following section.

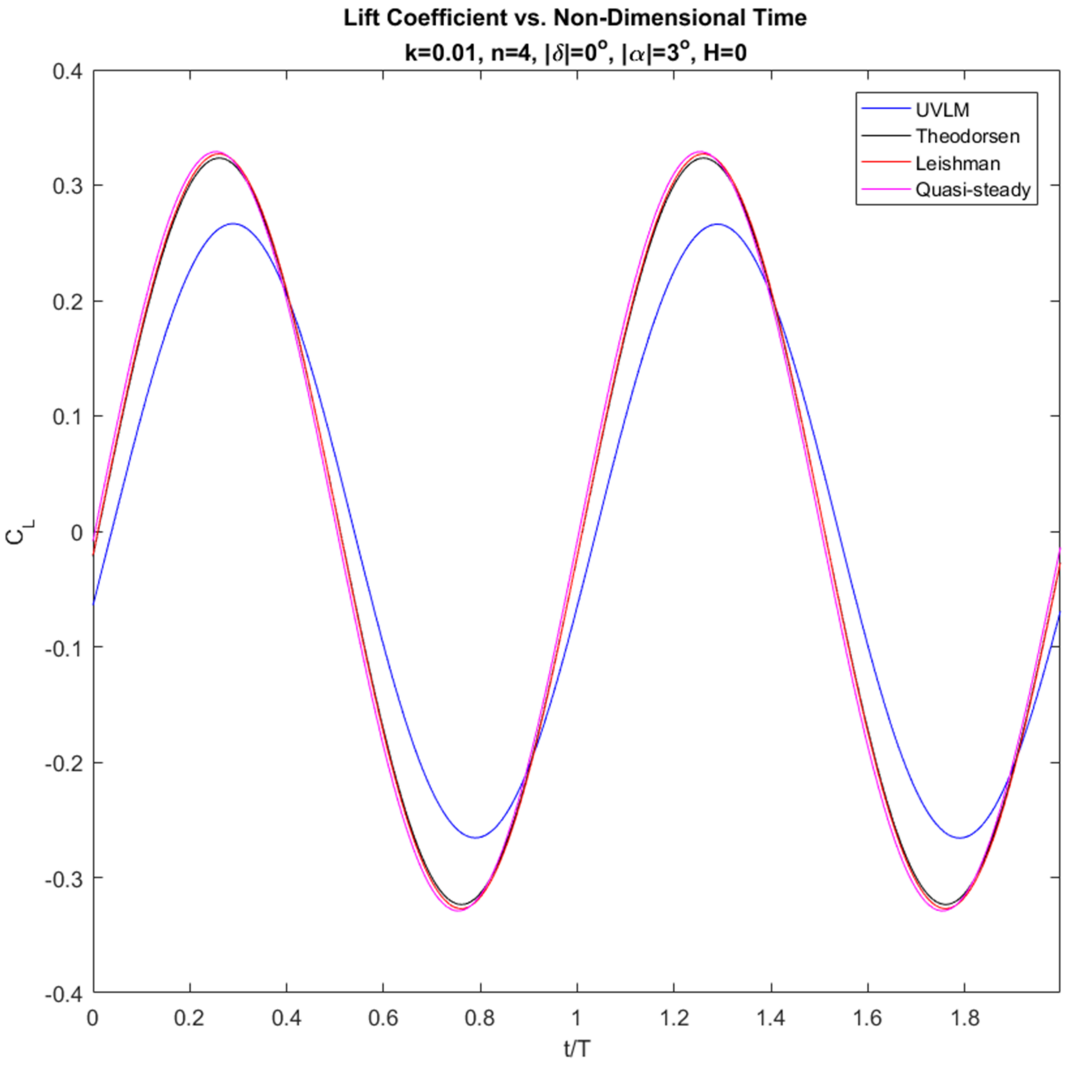

Figure 6.8: Pitching for $k = 0.01$.

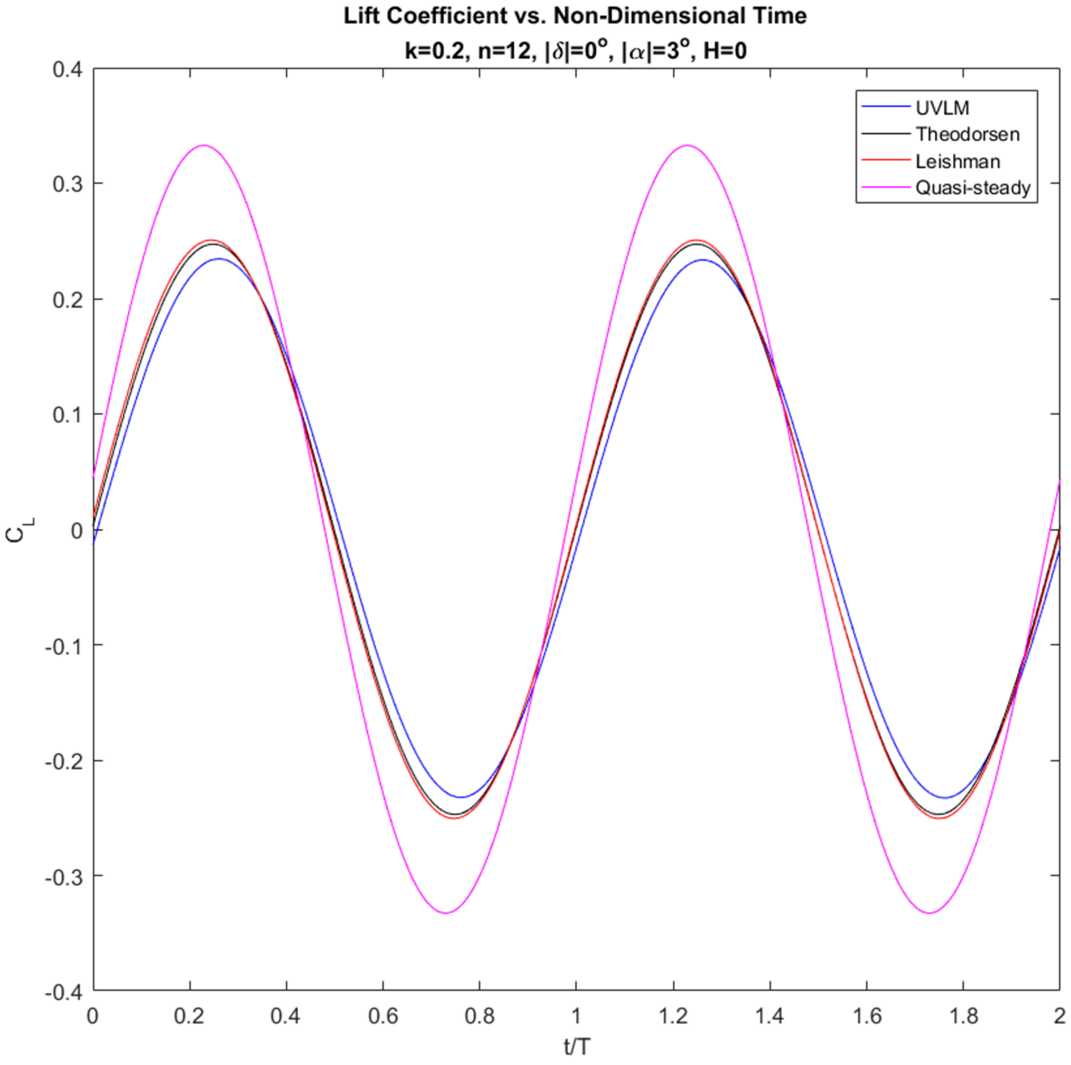

Figure 6.9: Pitching for $k = 0.2$.

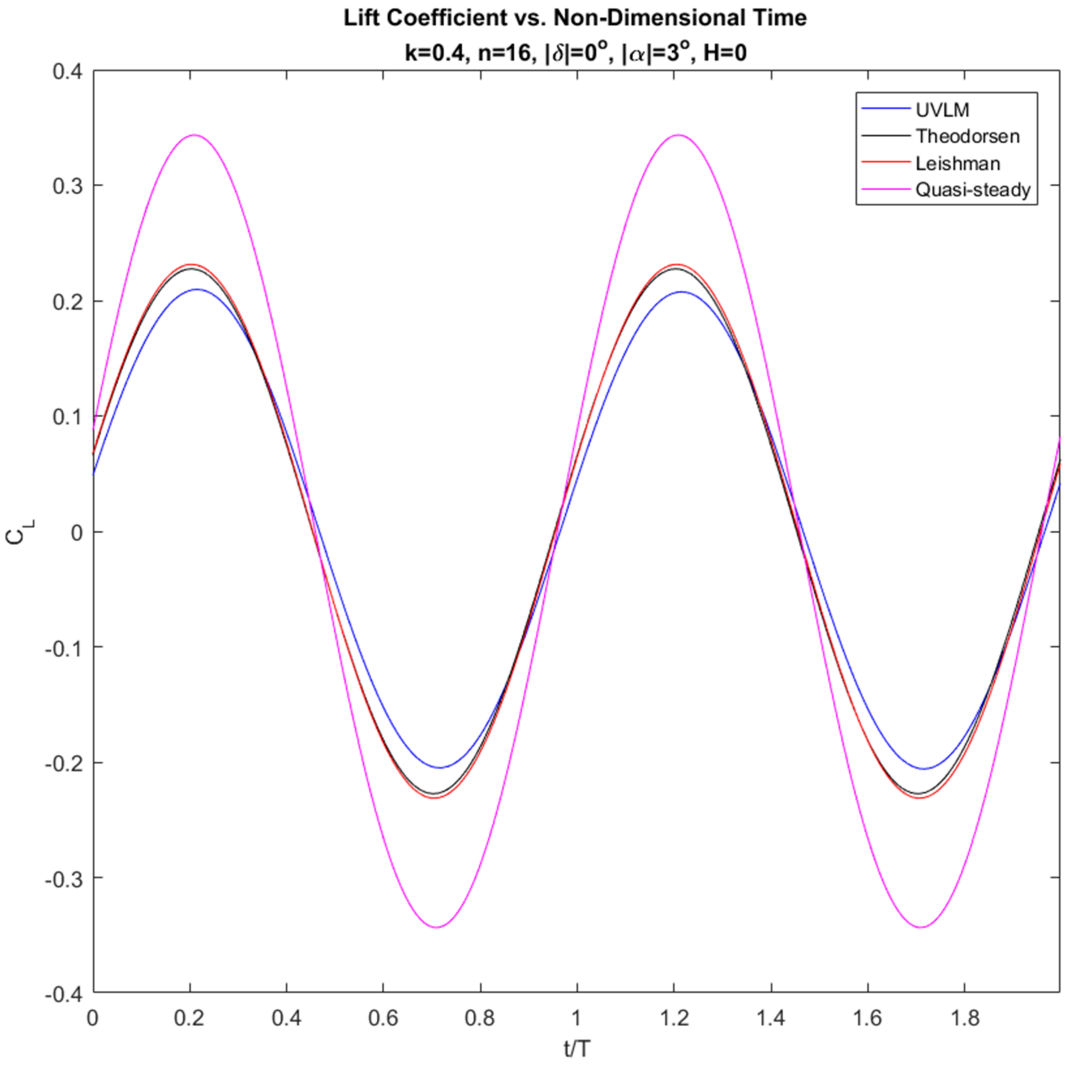

Figure 6.10: Pitching for $k = 0.4$.

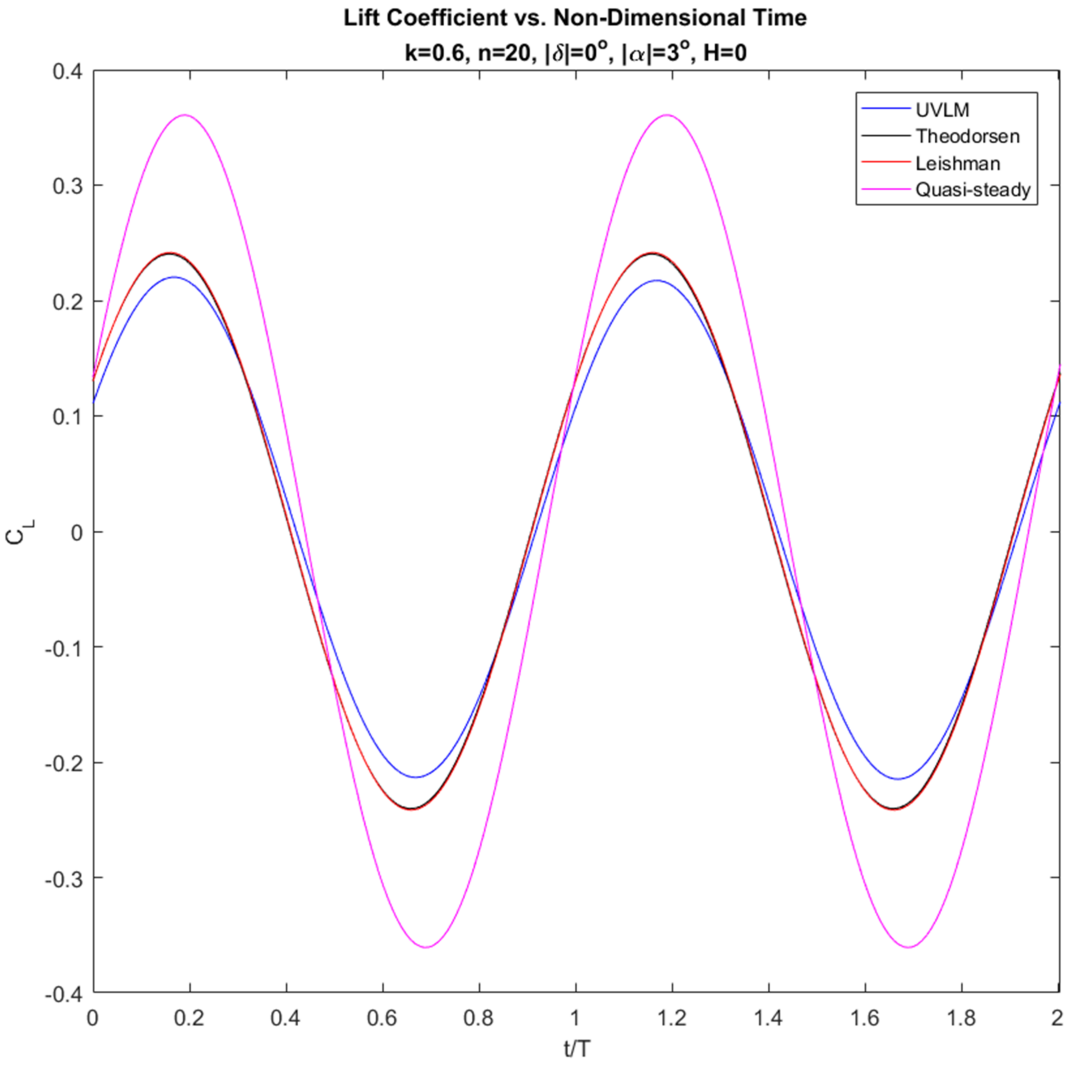

Figure 6.11: Pitching for $k = 0.6$.

6.2.3 Flapping

Flapping exhibits a similar pattern to that of pitching, in that lift initially decreases with increasing frequency until a certain transitional frequency is reached. At that value, which here is between $k = 1$ and $k = 2$ [5], the relation changes. Lift then increases proportionally with frequency.

It should also be noted that Leishman's plot for $k = 0.01$ is inaccurate, while for higher frequencies Leishman is accurate when compared to Theodorsen's. This is the same odd tendency noticed for the pitching case.

Again, the reader is invited to check the author's code if there exists any doubt about the results.

[5] This is more clearly seen in a following section regarding lift amplitude.

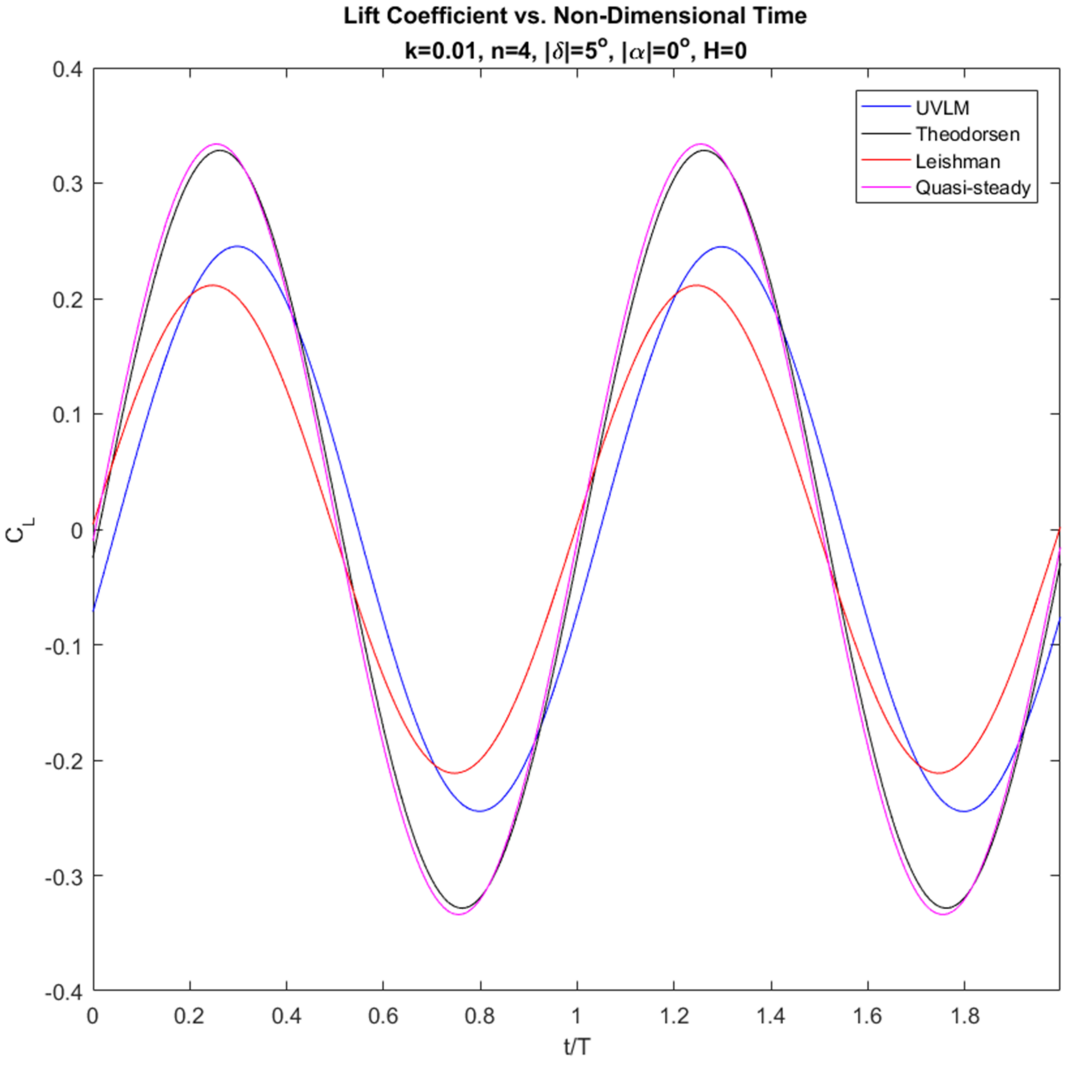

Figure 6.12: Flapping for $k = 0.01$.

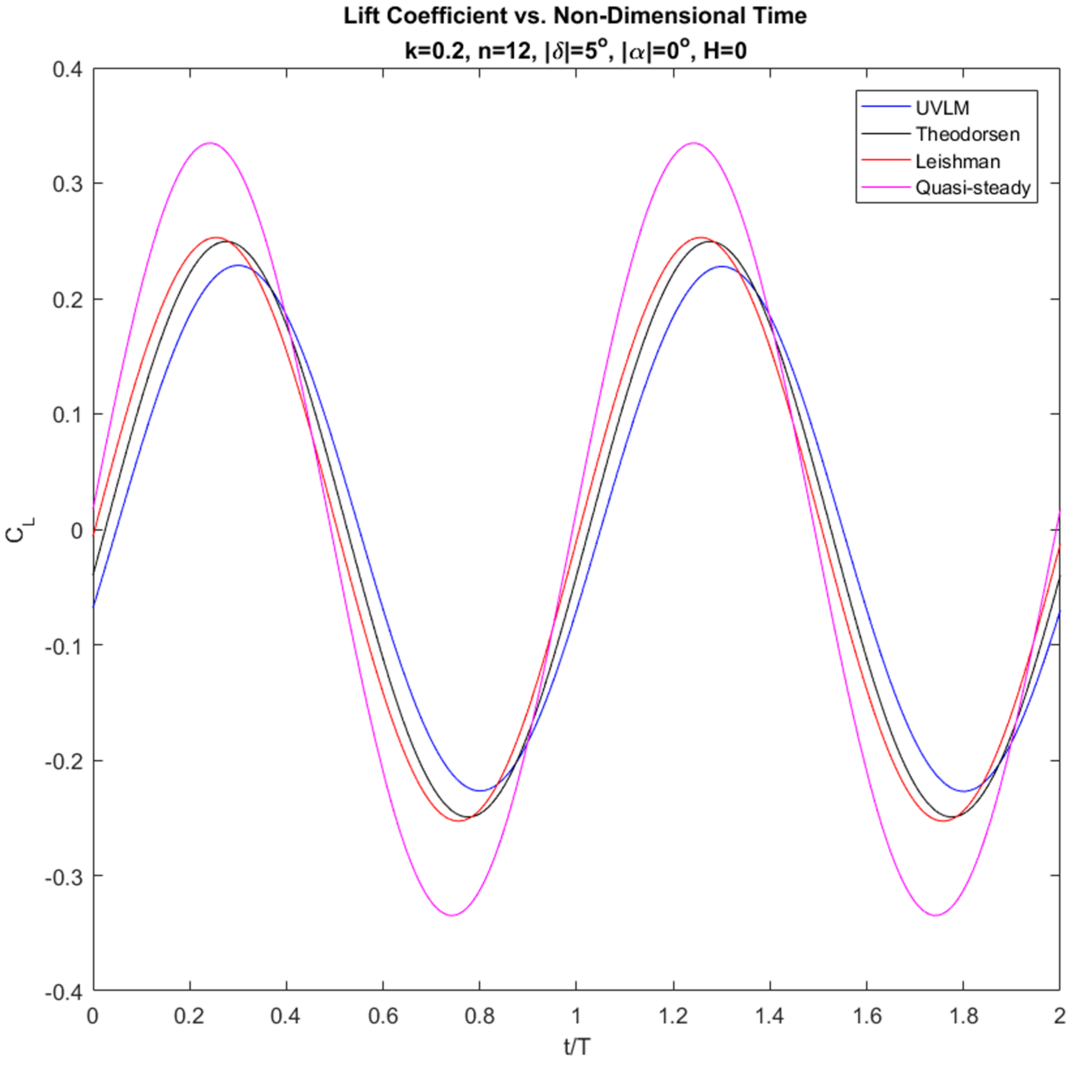

Figure 6.13: Flapping for $k = 0.2$.

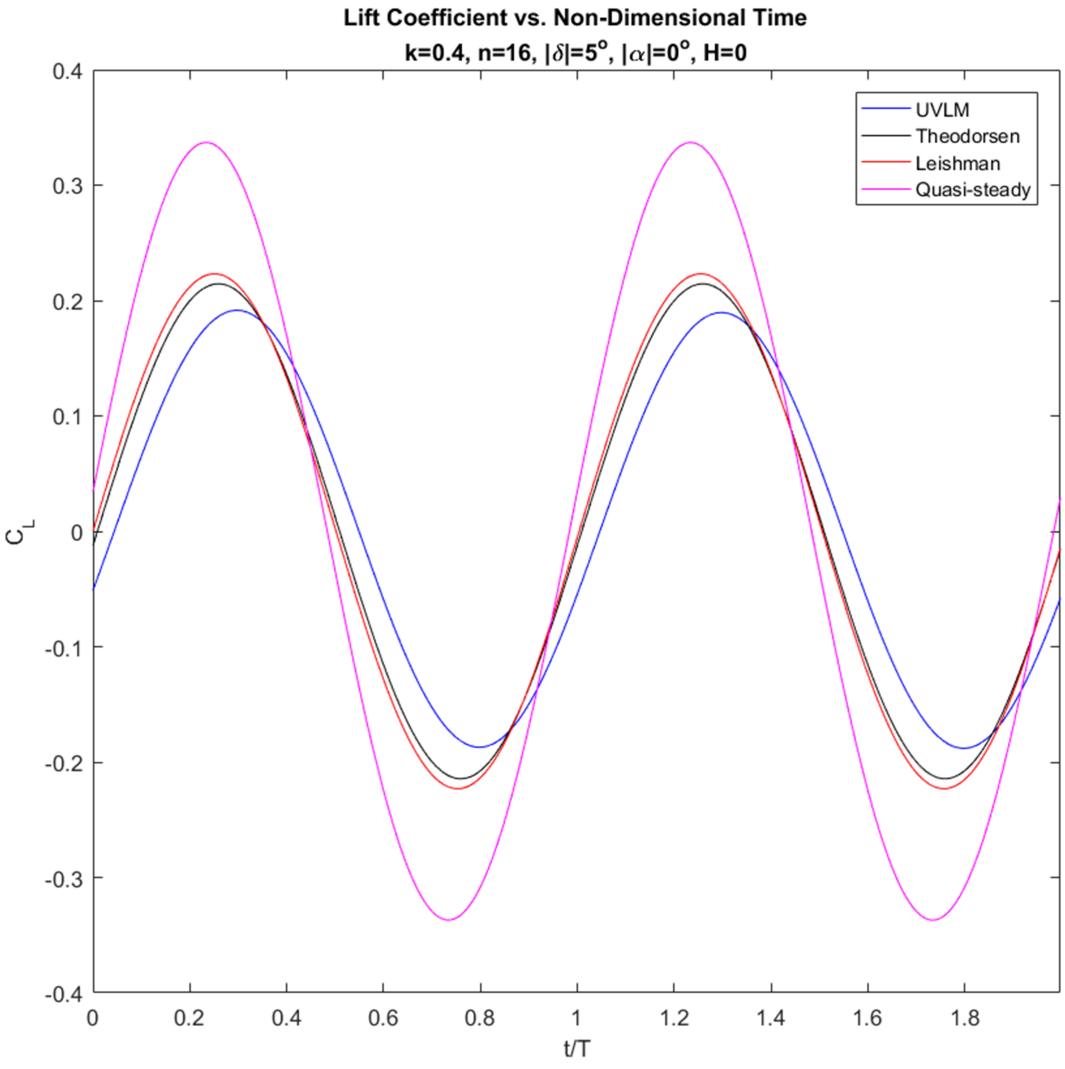

Figure 6.14: Flapping for $k = 0.4$.

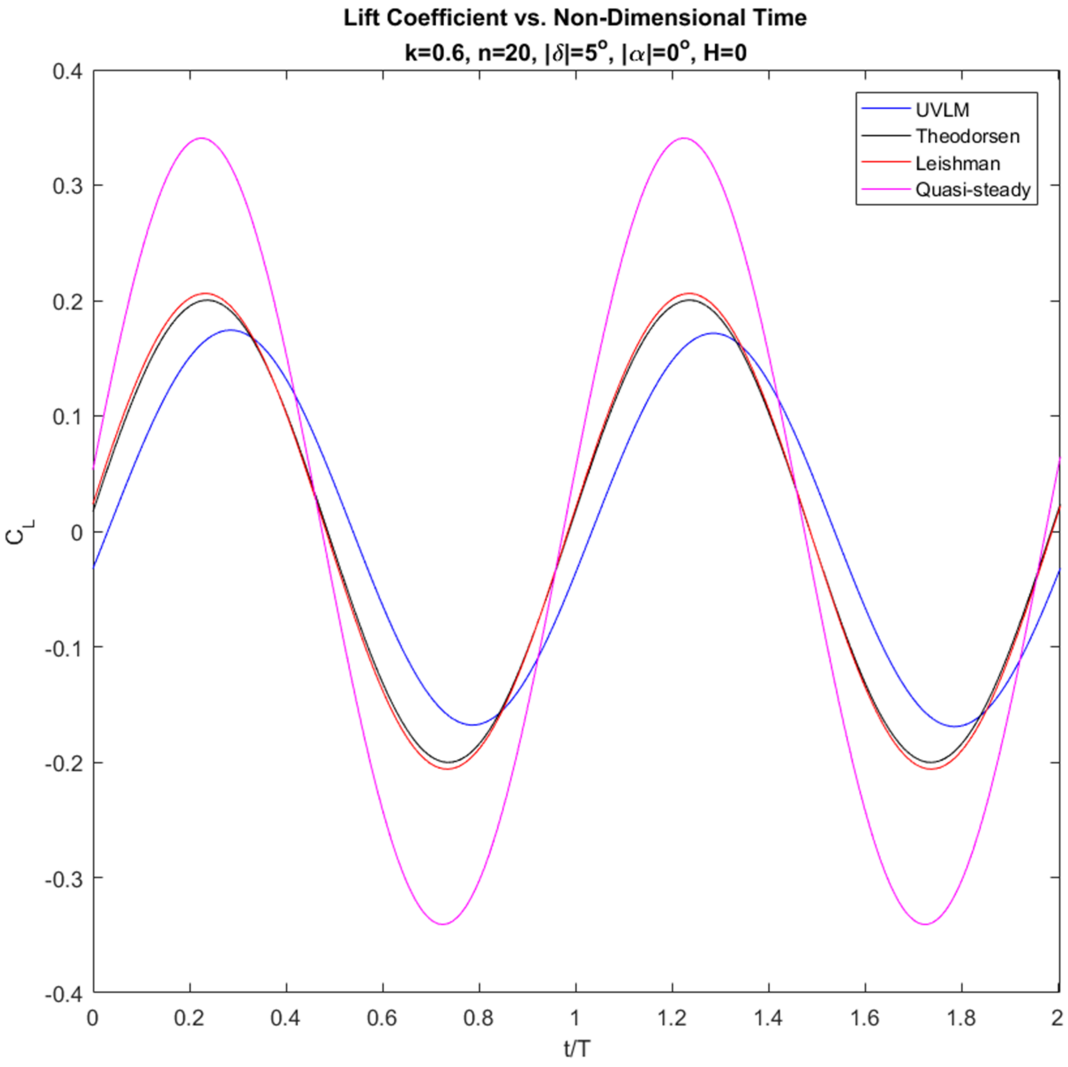

Figure 6.15: Flapping for $k = 0.6$.

Lift Shape Validation The plotting of lift for varying motions and frequencies demonstrates a few interesting patterns, but the overall point of this section is to validate the accuracy of the UVLM in predicting the output shape of the lift curve. As is seen by the plots, the UVLM holds to the proper sinusoidal output when given an sinusoidal input, just as the analytical methods. The UVLM, like Leishman, can also handle non-harmonic inputs (unlike Theodorsen's formulation), but a demonstration of that is not found in this paper.

6.3 Root-Mean Square Error

In order to establish the frequency range for which the select numerical method is accurate, it is useful to investigate the root-mean square error (RMS_e) of the lift coefficient as it varies with the reduced frequency. The error is defined as

$$\mathrm{RMS}_e = 100\% \times \sqrt{\frac{\sum_{i=1}^{T}(C_{L_{\mathrm{UVLM}_i}} - C_{L_{\mathrm{Theo}_i}})}{T}} \tag{6.1}$$

where the T here denotes the final time step, and the argument in the square root is the difference between the UVLM and Theodorsen formulations of lift.

Applying equation (6.1) to the plunging, pitching, and flapping results in the plot below.

As described by Robert J. S. Simpson[12], it is noted that, given a discretization, the UVLM shows increasing discrepancies with analytical theory. However, using such knowledge, the plot for figure 6.16 was generated using an increasingly dense set of collocation points in proportion to an increase in reduced frequency. A quantitative relation between discretization and frequency was not used, but a qualitative approach was.

Even so, it is clear by that the UVLM diverges from the analytical values in proportion to the frequency. The motion of flapping generates a much higher degree of error than the other

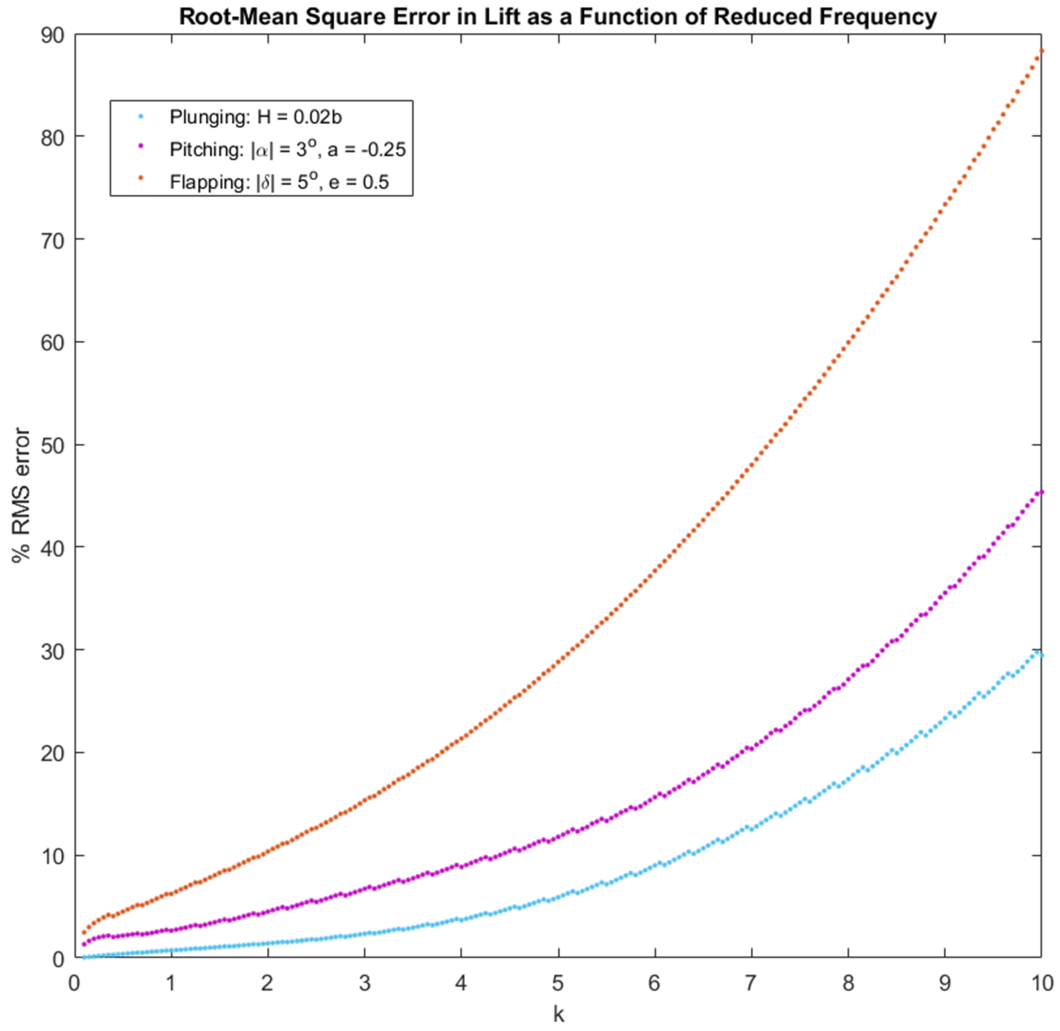

Figure 6.16: RMS_e trend for plunging, pitching, and flapping.

two motions. The reason behind such a difference is not clearly understood, but it likely has something to do with the boundary conditions imposed on the geometry.

In conclusion, the UVLM is best suited for lower frequencies, unless the error is somehow accounted for and corrected.

6.4 Phase Difference

It is useful to investigate the effects frequency and kinematic motion have on the phase ϕ of the unsteady lift with respect to the quasi-steady lift. There is a certain lag of the circulatory lift that is intrinsic to the fluid, but geometric motion and frequency seem to play a role in the degree of the that lag, as is shown in the below figures.

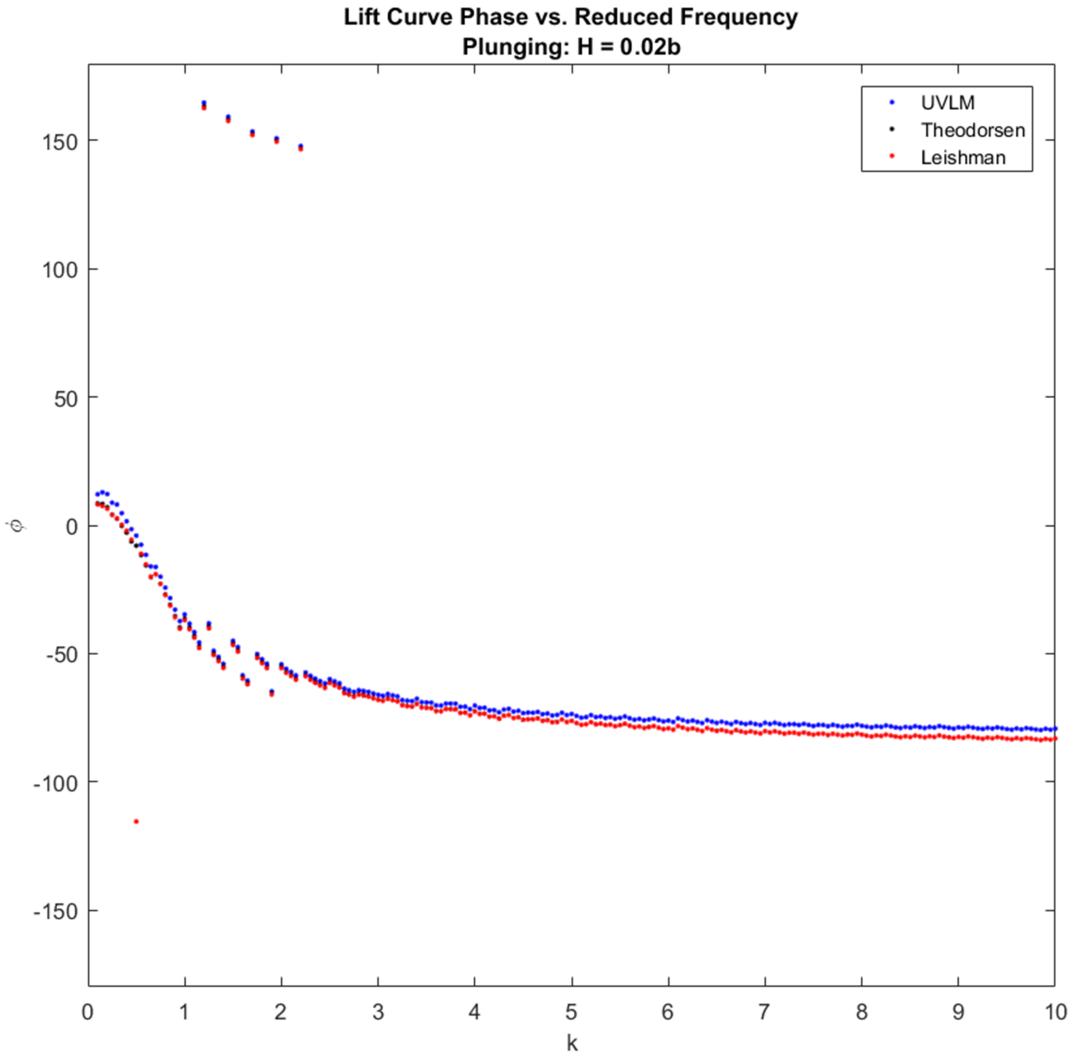

Figure 6.17: The phase difference ϕ for a plunging motion.

The general pattern noticed from figures 6.17-6.19 is that the phase difference ϕ is inversely proportional to the frequency. In essence, the unsteady lift shifts from lagging behind the

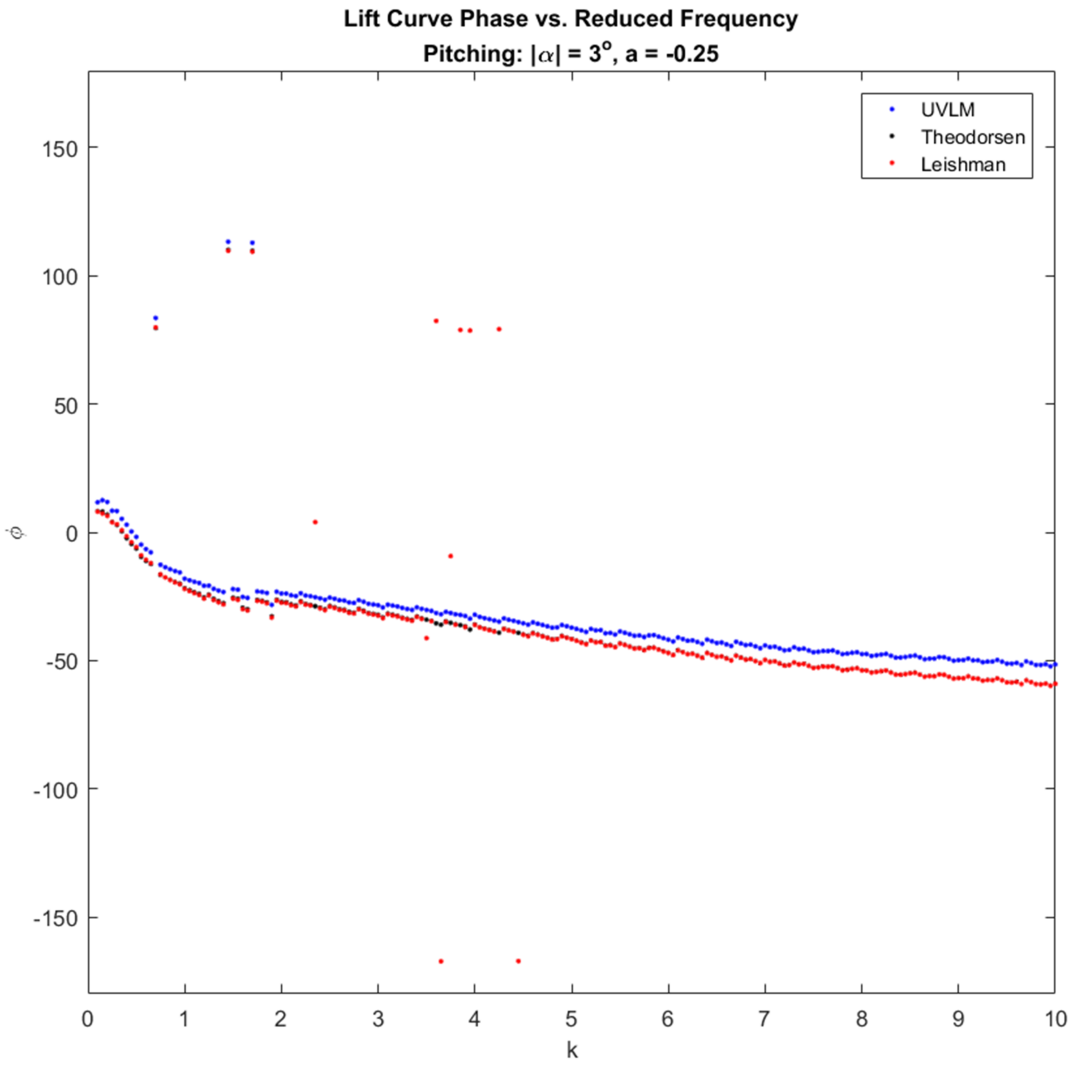

Figure 6.18: The phase difference ϕ for a pitching motion.

quasi-steady lift at low frequency ($k < 0.3$) to leading ahead of the quasi-steady lift at high frequency ($k > 0.3$).

The plunging motion has the steepest slope, starting at a positive ϕ then dropping toward $\phi = -60°$ as k grows. Phase of the pitching motion shows a similar trend of starting positive, then decreasing. The same can be said of the flap deflection oscillation; yet, unlike the former two, the phase yielded by the UVLM for the flapping motion diverges quite a large degree from the analytical methods.

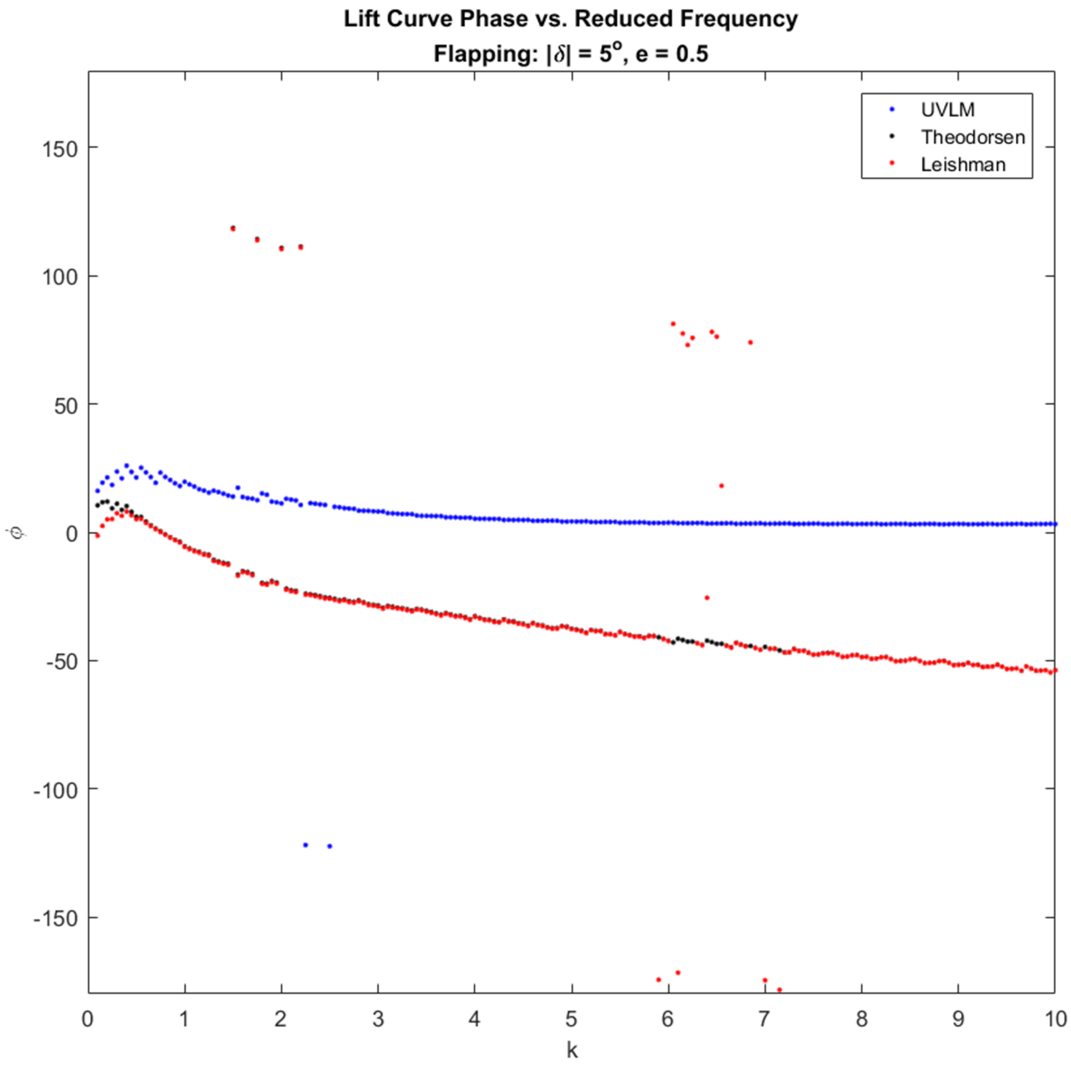

Figure 6.19: The phase difference ϕ for a flapping motion.

Phase Validation The UVLM demonstrates accurate phase prediction for plunging and pitching motions when compared to Theodorsen and Leishman. Contrarily, the flapping phase prediction of the UVLM does not match well with analytical results.

6.5 Lift Amplitude

As in the previous sections of this chapter, this section aims to find a pattern in the results obtained via the three methods applied for lift prediction. Here, the focus is on the variation of lift amplitude with frequency.

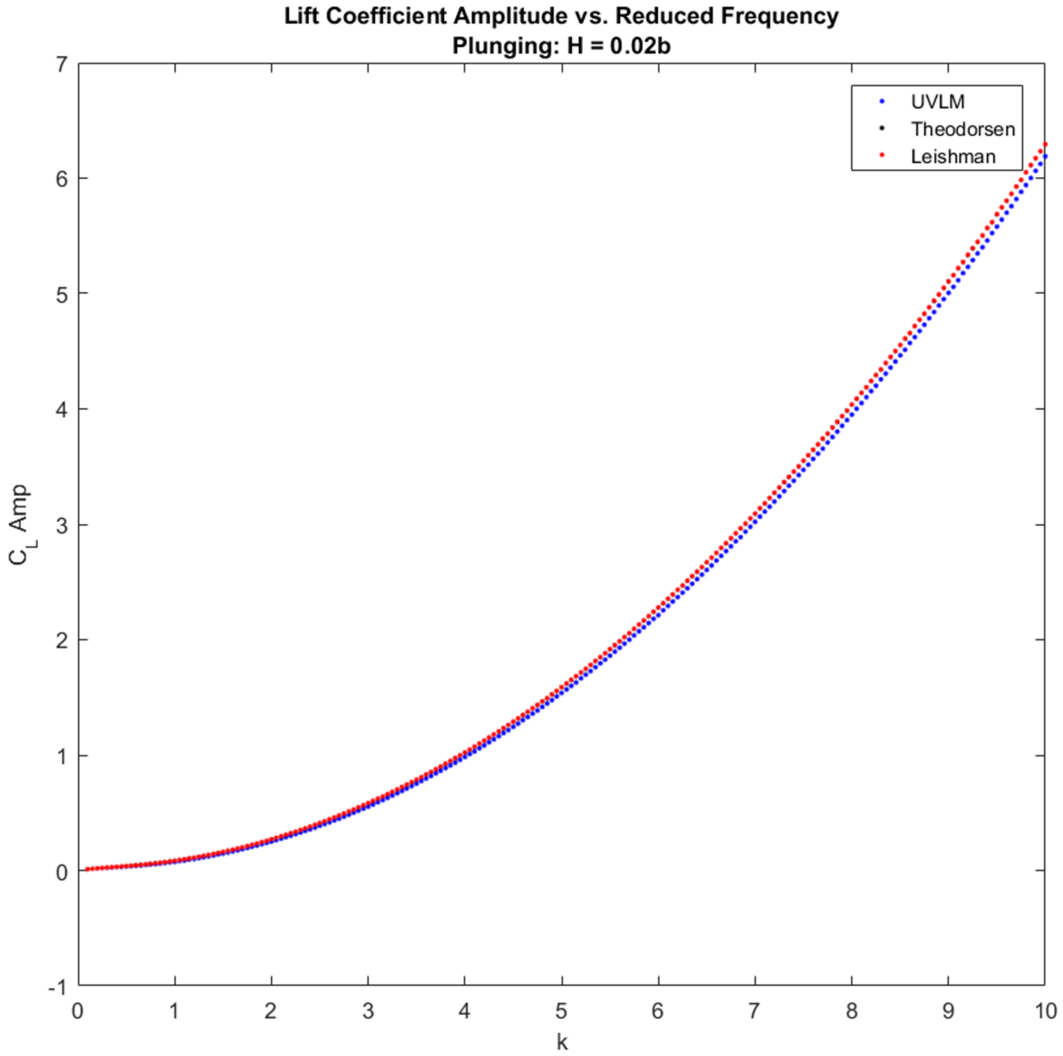

Figure 6.20: The lift amplitude for a plunging motion.

The largest C_L amplitude is produced by the plunging motion, which is no surprise; pitching follows in second; and flapping follows as a distant third. The accuracy of the UVLM relative to Theodorsen and Leishman follows in the same order of motion type: plunging, pitching,

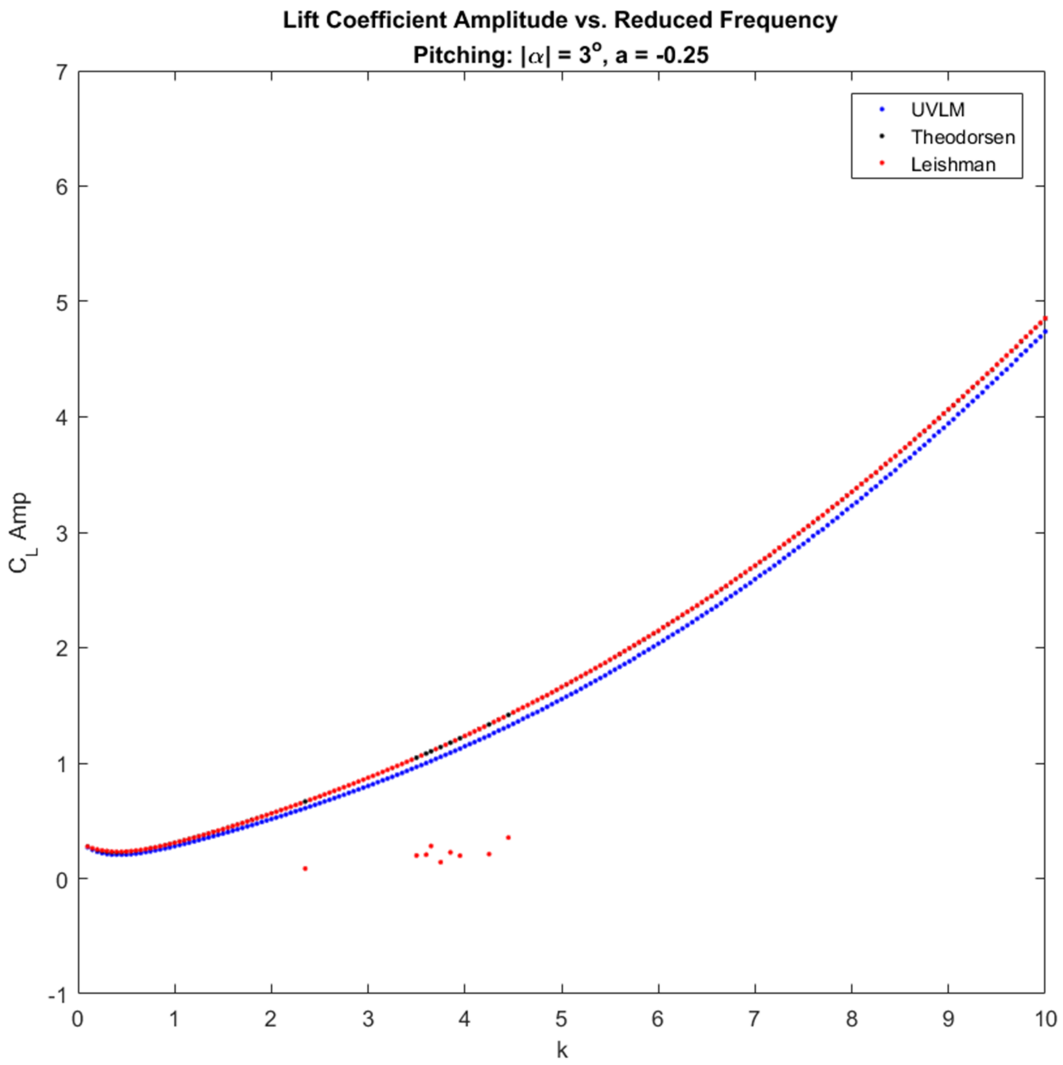

Figure 6.21: The lift amplitude for a pitching motion.

then flapping.

Figure 6.20 shows the analytical and numerical methods on top of each other, with a slow divergence as the frequency increases. Figure 6.21 show a similar trend, though its divergence is accelerated, though not to a great degree; and the rate of divergence seems to level off at around $k = 5$. Figure 6.22 shows a large divergence of values after $k > 1$.

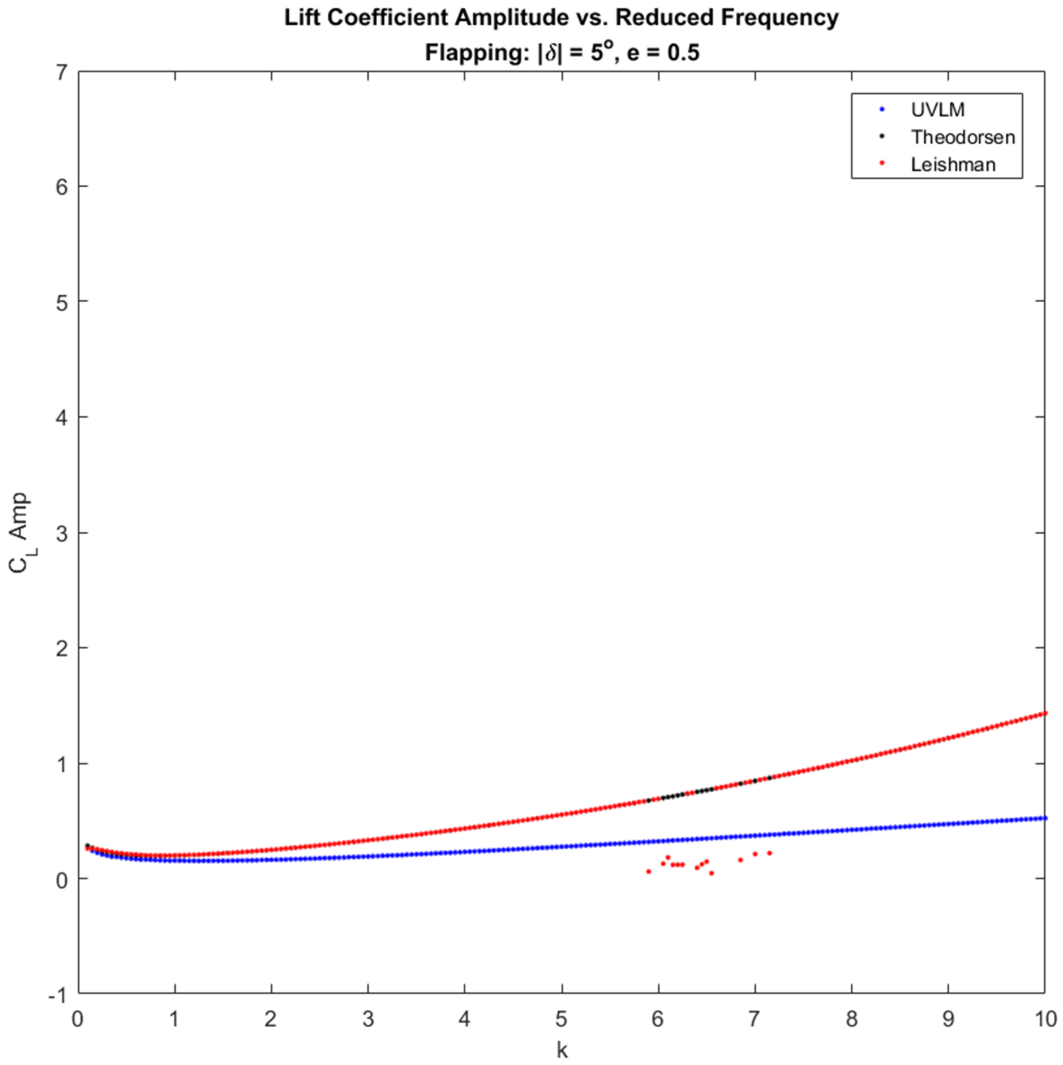

Figure 6.22: The lift amplitude for a flapping motion.

Amplitude Validation Based on the aforementioned plots, the UVLM in accurate in predicting the lift amplitude for plunging and pitching motions. For flapping motions, the UVLM amplitude diverges quickly from analytical results.

6.6 Mean Lift

The amplitude of the lift does not tell the full story; it says nothing in regard to whether or not a net non-zero lift force is produced. But determining the mean lift does just that.

For the UVLM, the mean lift of the UVLM is calculated using the following equation.

$$\overline{C}_L = \frac{1}{T} \sum_{k=1}^{T} C_{L_k} \tag{6.2}$$

The `mean` function in MATLAB is used to calculate the mean of the analytical methods, but for the numerical method it does not yield results as accurate as can be. Therefore, equation 6.2 is used explicitly for UVLM calculations, which in turn produces smooth results comparable to using `mean` for Theodorsen and Leishman.

Regarding the plots, a fifth order polynomial fit was to make the trend more easily recognizable. The exact points on the

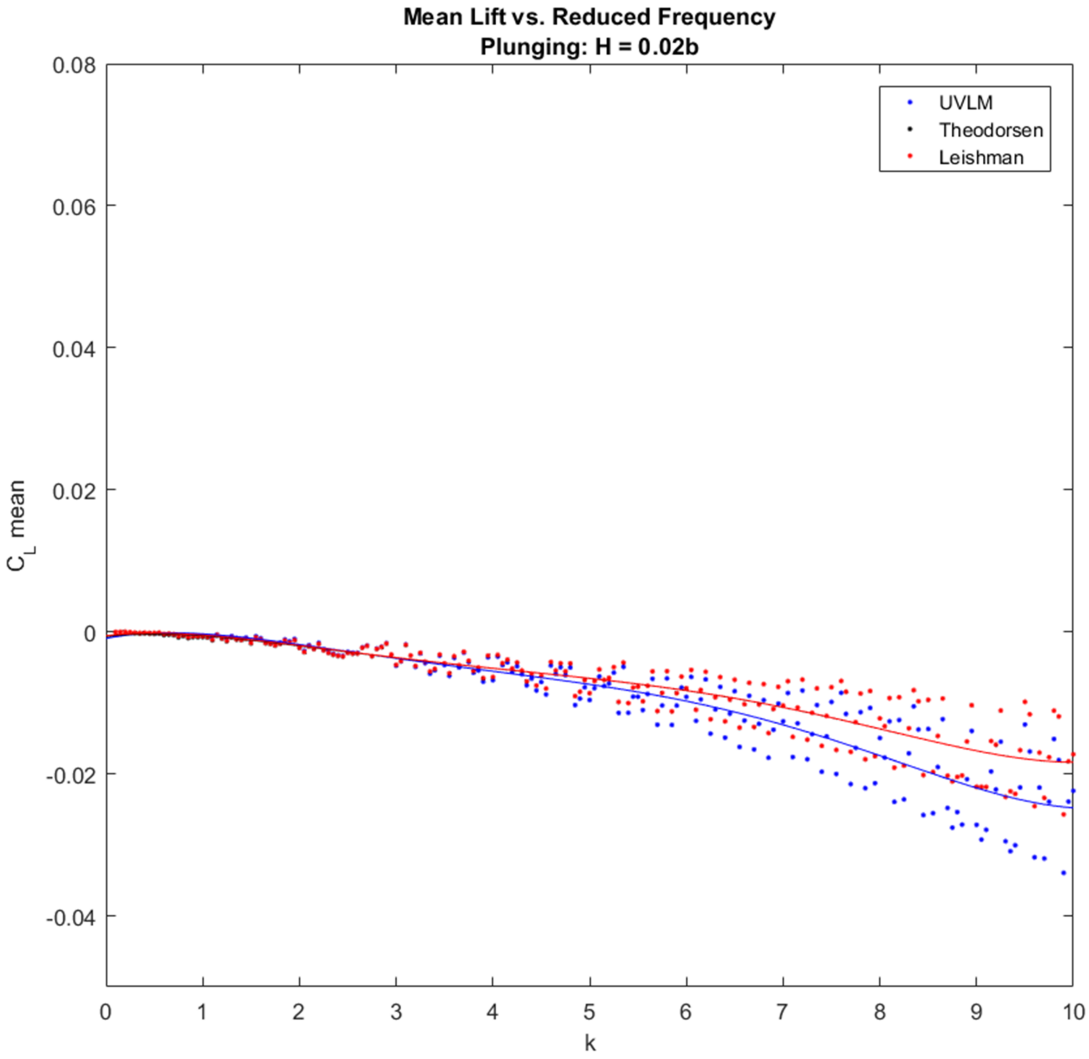

Figure 6.23: The mean lift for plunging at varying frequency.

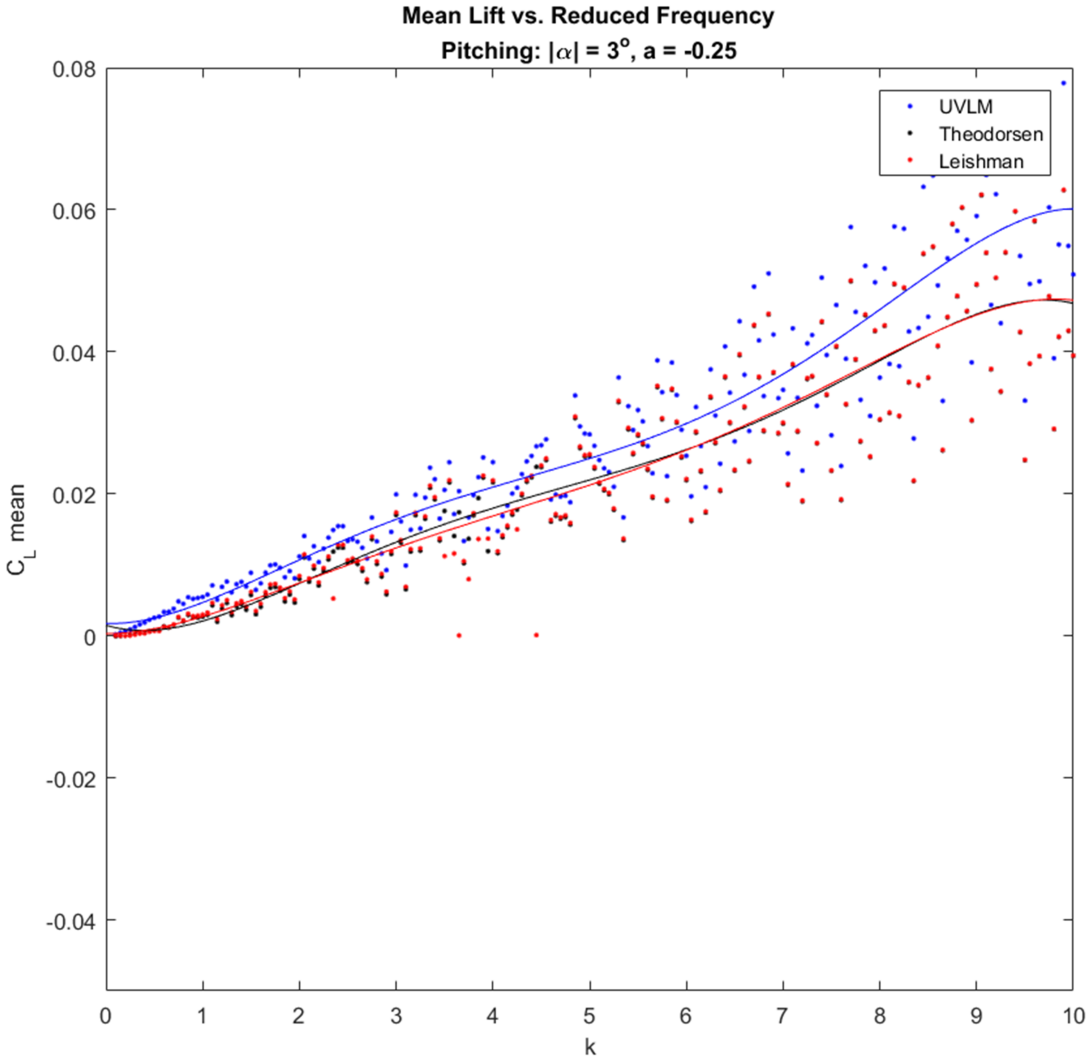

Figure 6.24: The mean lift for pitching at varying frequency.

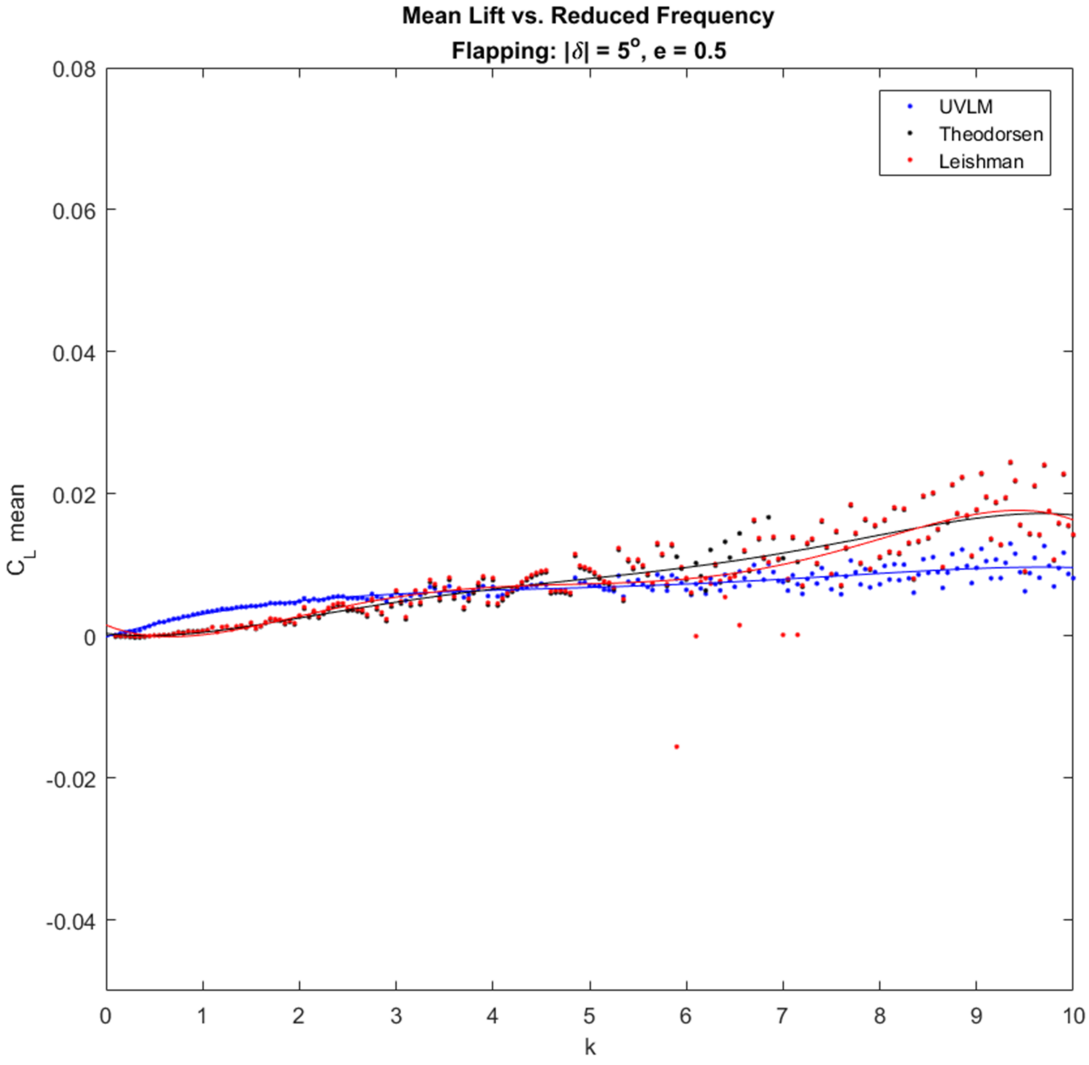

Figure 6.25: The mean lift for flapping at varying frequency.

Mean Lift Validation The plots for the mean lift are quite scattered. The $\overline{C}_L$ values vary in a seemingly random way as the frequency changes. It can be noted, however, that by viewing the polynomial fit line the UVLM maintains a similar trend to that of the analytical methods. Divergence occurs at high frequencies, which is the normal behavior of the select numerical method.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This article addresses the general effects of time-dependent motions on the unsteady lift response in potential flow for unsteady vortex lattice method (UVLM). Many things are explored, and a summary of the results is given in the following paragraphs.

The UVLM outputs a signal representative of the input signal. For example, as seen in the Lift Shape section, the output is sinusoidal for a given sinusoidal input. Unlike Theodorsen's formulation, the UVLM, like Leishman's description, can handle inputs other than those harmonic in nature. These, however, are not explored in this paper.

Regarding RMS error, it is shown that the error is proportional to frequency. The higher the frequency, the greater degree to which the UVLM deviates from analytical models. It is concluded, therefore, that the select numerical method be used for frequencies of $k \leq 1$.

The phase difference ϕ of the UVLM lift against the quasi-steady lift is well in line with the analytical methods for plunging and pitching motions of the parameters explored ($H = 0.02b$

and $|\alpha| = 3^o$ at $a = -0.25$, respectively), but such accuracy is not true for the prescribed flapping motion ($|\delta| = 5^o$). Flap deflection measured with the UVLM starts at a different value than Theodorsen and Leishman, even at low k, and it diverges even further as k increases. It is suggested that such patterns be considered if the UVLM is to be used to phase-sensitive studies.

As for lift, the amplitude of the UVLM measures closely with the two analytical methods for plunging and pitching. For flapping, the UVLMs amplitude measures closely up to $k \approx 1$, but it diverges for $k > 1$. This error in flapping is expected after examining the RMS, as noted previously.

When examining the mean lift of the motions, it is seen that plunging $\overline{C}_L$ decreases with increasing frequency, pitching $\overline{C}_L$ increase with increasing frequency, and flapping $\overline{C}_L$ shares the same trend as pitching, though the slope is much smaller. The most effective lift-producing motion of those prescribed is pitching.

In conclusion, the degree to which the UVLM matches with analytical results depends on motion type and frequency. Regarding frequency, lower frequency tends to lead to the least amount of error; and, as for motion, plunging has a smaller error margin than either pitching or flapping.

7.2 Future Work

7.2.1 Two-Dimensional

The present paper has tended to the analytical and numerical approach of lift response induced by a two-dimensional airfoil with an oscillating flap in unsteady flow. One area of research has been left out: experimental. Further investigation of the lift response to a sinu-

soidal input must be confronted through experimentation. The author hopes to collaborate with a colleague to tackle this particular endeavor.

The purpose of combining analytical, numerical, and experimental results is to eventually develop a reduced-order model of aerodynamic phenomena. From this model, the author plans to exploit the tools of geometric control in order to predict unintuitive effects of various combinations of pitching, plunging, and flapping airfoils, with the main goal of using unconventional means to increase lift or decrease drag for fighter aircraft maneuvers.

7.2.2 Three-Dimensional

Beyond the two-dimensional numerical approach, there is the more accurate three-dimensional approach. Though adding an extra dimension increases the computational effort, the results are more practical. The UVLM used in this paper can be extended to a third dimension, though not without any lack of complexity. The author has already made attempts to write the code and currently has preliminary—albeit inaccurate—results.

The erroneous results will not be put into this paper, but a figure of the visualization of a single 3D UVLM run is below.

It is the author's hope to perfect the 3D UVLM code, then exaggerate the aspect ratio in order to validate it against a 2D analytical method. Of course, a comparison with experimental data is also desired, but that will not be the first mode of validation.

Following aerodynamic validation, the 3D UVLM will ideally be combined with a structural model to mimick phenomena other than those exhibited by rigid bodies.

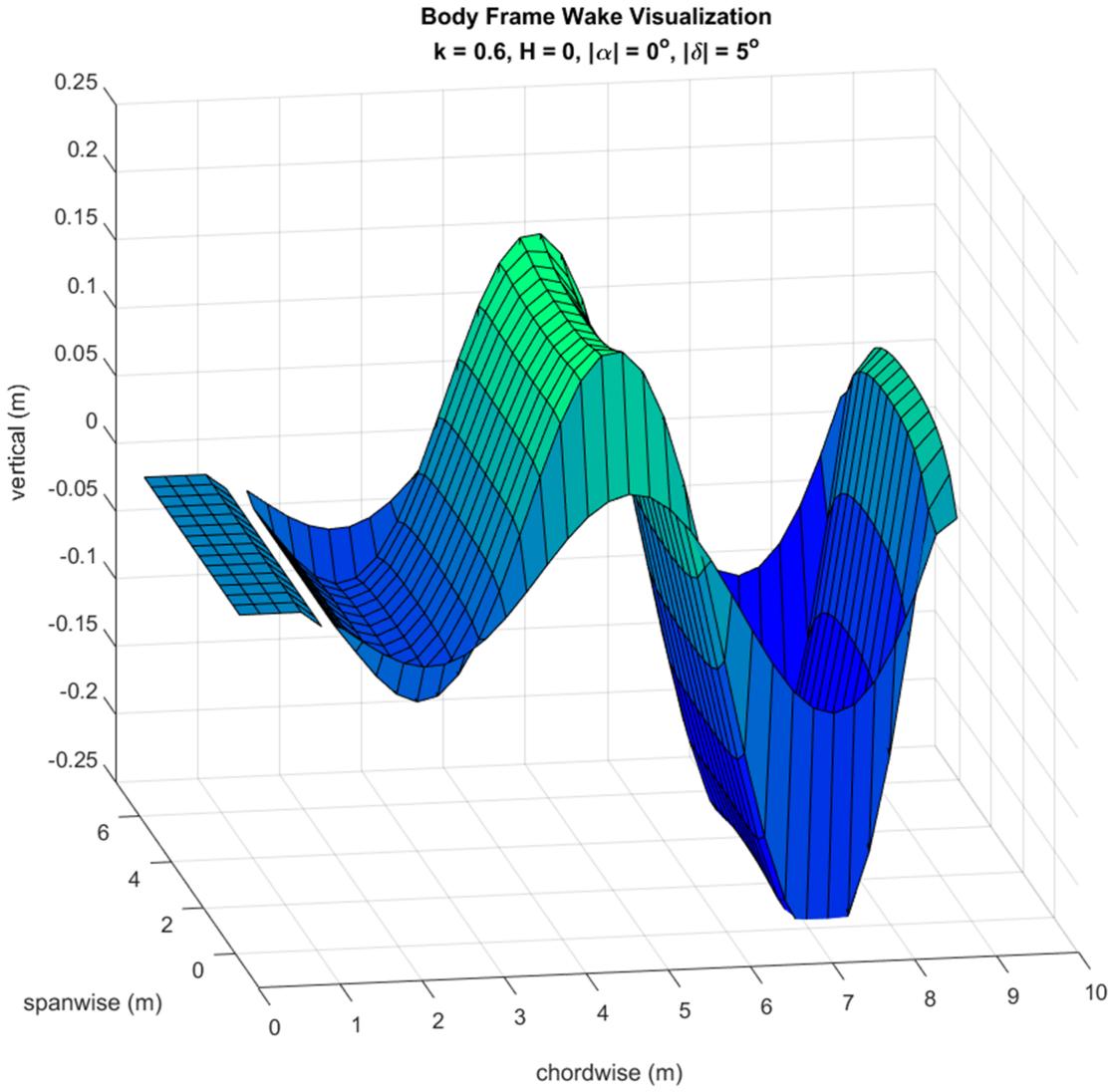

Figure 7.1: Visualization of 3D UVLM code with a flap deflection. There are four chordwise panels and fourteen spanwise panels. The chord length is one meter and the span length is six meters.

.1 Appendix A

.1.1 Geometric Constants

$$F_1 = -\frac{1}{3}\sqrt{1-e^2}(2+e^2) + e\cos^{-1} e$$

$$F_4 = -\cos^{-1} e + e\sqrt{1-e^2}$$

$$F_{10} = \sqrt{1-e^2} + \cos^{-1} e$$

$$F_{11} = \cos^{-1} e(1-2e) + \sqrt{1-e^2}(2-e)$$

$$F_{20} = F_{10} - 2\sqrt{1-e^2}$$

.2 Appendix B

.2.1 Unsteady Vortex Lattice Method Code

```
1  %----This routine finds the lift on a rigid airfoil using the the Unsteady
2  %    Vortex Lattice Method. Refer to "Low Speed Aerodynamics," chapter 13,
3  %    for details beyond the author's thesis.
4
5  %    Author: Ryan Muoio
6  %    Date: 9/6/2016
7
8  clear all;
9  % close all;
10 clc;
11 tic
12
```

```
13 %% Sign conventions
14
15 % UVLM:          +ve right, +ve up, +ve clockwise
16 % Theodorsen:    +ve right, +ve down, +ve clockwise
17 % Leishman:      +ve right, +ve up, +ve clockwise
18
19
20 %% Input
21
22 n = 24; %use multiples of 4 for all other comparisons
23 b = 0.5;
24 c = 2*b;
25
26 %----Leishman/Theodorsen variables
27 M = 0.5;
28 v_sound = 340.3; %speed of sound at sea level in m/s
29 Uinf = M*v_sound;
30 mu = 1.983*10^-5;
31 rho = 1.225; %for air, units kg/m^3
32 Re = rho*Uinf*c/mu;
33 kk = 1;
34 f = Uinf*kk/(2*pi*b);
35 H = 0.02*b;
36 Str = kk*H/(pi*c);
37
38 %----Plunging
39 % H = Str*pi*c/kk;
40 % H = 0.1*b;
41 % H = Str*Re*mu/(2*f*rho*c);
42 phi_h = 0;
43
44 %----Pitching
45 Alpha = 0*pi/180;
```

```
46 alpha_m = 0*pi/180;
47 a = -0.25;
48 phi_alpha = pi/2;
49
50 %----Flapping
51 e = 0.5; %hinge axis ratio as measured from midchord toward TE
52 cflap = b*(1-e);
53 lambda = cflap/c;
54 K = (1-lambda)*n;
55 delta0 = 0*pi/180;
56 Delta = 0*pi/180;
57 phi_delta = pi/2;
58
59 no_cycles = 5;
60 no_cyc_cut = 2;
61
62
63 %% Initialization
64
65 panel = c/n;
66
67 cut_off = panel*10^(-6); %core radius value found by Taha in indep. study
68
69 Period = 1/f;
70 w = 2*pi*f;
71
72 nt = 1;
73 dt = panel/(nt*Uinf);
74 t = 0:dt:no_cycles*Period;
75 % t = 0:dt*10:no_cycles*Period;
76 lt = length(t);
77
78 %----Initial wake setup
```

```
79 G = zeros(n+1,1);
80 G_Wake = 0; %sum of wake vortices
81 no_Wake = 0;
82 xz_Wake = []; %position of wake vortices
83 G_tot_old = 0;
84
85
86 %% Pre-Allocation
87
88 %----UVLM
89 xz_bound=zeros(n+1,2); xz_cp=zeros(n,2); Nvec=zeros(n,2); Tvec=zeros(n,2);
90 hdot=zeros(lt,1); hdot2=zeros(lt,1); alpha=zeros(lt,1);
91 alphadot=zeros(lt,1); alphadot2=zeros(lt,1); delta=zeros(lt,1);
92 deltadot=zeros(lt,1); deltadot2=zeros(lt,1);
93 RHS=zeros(n+1,1); A=zeros(n+1,n+1); p_diff=zeros(1,n);
94 Cp_diff=zeros(1,n); Cl=zeros(1,lt); D=zeros(1,lt); P=zeros(1,lt);
95 CP_UVLM=zeros(1,lt); Thrust_UVLM=zeros(1,lt); CT_UVLM=zeros(1,lt);
96 eta_UVLM=zeros(1,lt); CD_UVLM=zeros(1,lt); Cl_avg=zeros(1,lt);
97
98
99 %% Time-Invariant Geometry
100
101 for i = 1:K
102     xz_bound(i,1) = (i-1+0.25)*c/n;
103     xz_bound(i,2) = 0;
104     xz_cp(i,1) = (i-1+0.75)*c/n;
105     xz_cp(i,2) = 0;
106     Nvec(i,[1 2]) = [0 1];
107     Tvec(i,[1 2]) = [1 0];
108 end
109
110
111 %% Time Loop
```

```
112
113 for k = 1:lt
114 %% Kinematics
115
116     h(k) = H*sin(w*t(k));
117     hdot(k) = w*H*cos(w*t(k));
118     hdot2(k) = -w^2*H*sin(w*t(k));
119     alpha(k) = alpha_m+Alpha*sin(w*t(k));
120     alphadot(k) = w*Alpha*cos(w*t(k));
121     alphadot2(k) = -w^2*Alpha*sin(w*t(k));
122     delta(k) = delta0+Delta*sin(w*t(k)+phi_delta);
123     deltadot(k) = w*Delta*cos(w*t(k)+phi_delta);
124     deltadot2(k) = -w^2*Delta*sin(w*t(k)+phi_delta);
125     alphaeff(k) = alpha(k)+atan(hdot(k)/Uinf);
126     if alphaeff(k)*180/pi > 5
127         error('You are a moron. This code is not made for such high ...
              effective AoAs.');
128     end
129
130     %----Freestream moving to the right
131     Q_plunge(k,[1 2]) = [-Uinf*cos(alpha(k))-hdot(k)*sin(alpha(k)),...
132                 -Uinf*sin(alpha(k))+hdot(k)*cos(alpha(k))];
133
134
135 %% Time-Variant Geometry
136
137     for i = K+1:n
138         xz_bound(i,1) = (c-cflap)+(i-K-1+0.25)*cflap/(n-K)*cos(delta(k));
139         xz_bound(i,2) = -(i-K-1+0.25)*cflap/(n-K)*sin(delta(k));
140         xz_cp(i,1) = (c-cflap)+(i-K-1+0.75)*cflap/(n-K)*cos(delta(k));
141         xz_cp(i,2) = -(i-K-1+0.75)*cflap/(n-K)*sin(delta(k));
142         Nvec(i,[1 2]) = [sin(delta(k)) cos(delta(k))];
143         Tvec(i,[1 2]) = [cos(delta(k)) -sin(delta(k))];
```

```
144     end
145
146     xz_bound(n+1,1) = (c-cflap)+cflap*cos(delta(k));
147     xz_bound(n+1,2) = -cflap*sin(delta(k));
148
149
150 %% Influence Coefficient Matrix
151
152     for i = 1:n
153         for j = 1:n
154             q = Biot_Savart(xz_cp(i,:),xz_bound(j,:),cut_off);
155             A(i,j) = sum(q.*Nvec(i,:));
156         end
157         q = Biot_Savart(xz_cp(i,:),xz_bound(n+1,:),cut_off);
158         A(i,n+1) = sum(q.*Nvec(i,:));
159     end
160     A(n+1,:) = ones(1,n+1); %Conservation of circulation
161
162
163 %% Right-Hand Side
164
165     for i = 1:n
166         %----Airfoil motion
167         Q_pp = Q_plunge(k,:);
168         Q_pp(2) = Q_pp(2)-alphadot(k)*(xz_cp(i,1)-b*(1+a));
169         RHS(i,1) = sum(Q_pp.*Nvec(i,:)); %RHS kinematic motion
170
171         %----Wake effect
172         for j = 1:no_Wake
173             q = Biot_Savart(xz_cp(i,:),xz_Wake(j,:),cut_off)*G_W(j);
174             RHS(i) = RHS(i)-sum(q.*Nvec(i,:));
175         end
176     end
```

```
177
178      %----Conversation of circulation
179      RHS(n+1) = -G_Wake;
180
181
182  %% Solving the System: Vortex Strengths, Pressures, and Loads
183
184      G_old = G; %doesn't contribute anything on first time step
185      G = A\RHS; %solves for vortex strengths
186
187      G_sum = 0; G_old_sum = 0; Cll = 0; Drag = 0; Power = 0; Lift = 0;
188
189      for i = 1:n
190          G_sum = G_sum+G(i); %current iteration's vortices
191          G_old_sum = G_old_sum+G_old(i); %previous iteration's vortices
192
193          %----Velocity induced by the wake vortices
194          q_Wake = zeros(1,2);
195          for p = 1:no_Wake
196              q_Wake = q_Wake+Biot_Savart(xz_cp(i,:),xz_Wake(p,:),...
197                  cut_off)*G_W(p);
198          end
199          w_wdown(i) = q_Wake(2); %wake-induced downwash
200
201          %----Velocity induced by bound vortices
202          for j = n
203              q = q+Biot_Savart(xz_cp(i,:),xz_bound(j,:),cut_off);
204          end
205          w_bdown(i) = q(2);
206
207          Q_pp = Q_plunge(k,:);
208          Q_pp(2) = Q_pp(2)-alphadot(k)*(xz_cp(i,1)-b*(1+a));
209          Q = q_Wake-Q_pp; %induced velocity relative to the airfoil
```

```
210          Qn = sum(Q.*Nvec(i,:)); %normal velocity
211          Qt = sum(Q.*Tvec(i,:)); %tangential velocity
212
213          %----Pressure difference
214          p_diff(i) = rho*(Qt*G(i)/panel+(G_sum-G_old_sum)/dt);
215          Cp_diff(i) = p_diff(i)/(0.5*rho*Uinf^2);
216
217
218           %% Katz and Plotkin: Method 1b (geometric angles)
219           alpha_i = atan(Qn/Qt); %local angle of attack
220           if i <= K
221              Cll = Cll+Cp_diff(i)*panel*cos(alpha(k));
222           else
223              Cll = Cll+Cp_diff(i)*panel*cos(alpha(k)+delta(k));
224           end
225      end
226
227      %----Suction force: see Garrick's paper and Ramesh's paper
228      integrand_0(i) = w_bdown(i)*(1-(1-2*xz_cp(i,1)/c)^2)^(-1/2);
229      A0 = -2/(c*Uinf*pi)*trapz(xz_cp(:,1),integrand_0); %aerodynamic coeff.
230      S = sqrt(2)*Uinf*A0;
231      Fs = pi*b*rho*S^2;
232
233      %----Lift at each time step
234      Cl(k) = Cll/c+Fs/(rho*Uinf^2*b)*sin(alpha(k));
235
236
237  %% Wake Roll-Up (Convection)
238
239      for ii = no_Wake:-1:1 %creates a loop that increases vortex quantity
240                            %at each time step
241          G_W(ii+1) = G_W(ii);
242      end
```

```
243
244      G_W(1) = G(n+1); %sets the first wake vortex as the starting vortex
245      G_Wake = G_Wake+G_W(1); %sums the wake starting vortices from all steps
246      no_Wake = no_Wake+1;
247      xz_Wake = [xz_bound(n+1,:); xz_Wake];
248
249      for p = 1:no_Wake
250
251          %----Airfoil motion
252          Q_pp = Q_plunge(k,:);
253          Q_pp(2) = Q_pp(2)-alphadot(k)*(xz_Wake(p,1)-b*(1+a));
254          xz_Wake_new(p,:) = xz_Wake(p,:)-Q_pp*dt;
255
256          %----Effect of bound vortices' velocity on wake vortices
257          for j = 1:n
258              q = Biot_Savart(xz_Wake(p,:),xz_bound(j,:),cut_off)*G(j);
259              xz_Wake_new(p,:) = xz_Wake_new(p,:)+q*dt;
260          end
261
262          %----Effect of wake vortices' velocity on the wake vortices
263          for j = 1:no_Wake
264              q = Biot_Savart(xz_Wake(p,:),xz_Wake(j,:),cut_off)*G_W(j);
265              xz_Wake_new(p,:) = xz_Wake_new(p,:)+q*dt;
266          end
267
268      end
269
270      xz_Wake = xz_Wake_new;
271
272  end
273
274  toc
275  rdf = num2str(kk); rdf = strcat('k=',rdf);
```

```
276 chord = num2str(c); chord = strcat('c=',chord);
277 panels = num2str(n); panels = strcat('n=',panels);
278 Reynolds = num2str(round(Re,-6)); Reynolds = strcat('Re=',Reynolds);
279 Strouhal = num2str(Str); Strouhal = strcat('St=',Strouhal);
280 FlapDef = num2str(Delta*180/pi); FlapDef = ...
       strcat('|\delta|=',FlapDef,'^o');
281 Pitch = num2str(Alpha*180/pi); Pitch = strcat('|\alpha|=',Pitch,'^o');
282 Plunge = num2str(H); Plunge = strcat('H=',Plunge);
283
284 if no_cyc_cut ~= 0
285     Plotcut = round(no_cyc_cut/no_cycles*lt); %removes transient cycles
286     Cl_UVLM = Cl(Plotcut:lt);
287     Cl_avg_UVLM = Cl_avg(Plotcut:lt);
288     CP_UVLM = CP_UVLM(Plotcut:lt);
289     eta_UVLM = eta_UVLM(Plotcut:lt);
290     t_UVLM = t(Plotcut:lt);
291     delta_UVLM = delta(Plotcut:lt);
292     Thrust_UVLM = Thrust_UVLM(Plotcut:lt);
293     CT_UVLM = CT_UVLM(Plotcut:lt);
294     CD_UVLM = CD_UVLM(Plotcut:lt);
295     alphaeff = alphaeff(Plotcut:lt);
296 end
297
298 %----Period-Averaged Propuslive Efficiency
299 eta_avg = dt/((no_cycles-no_cyc_cut)*Period)*sum(eta_UVLM);
300 AvgEta = num2str(eta_avg); AvgEta = strcat('\eta_a=',AvgEta);
301
302 %----Period-Averaged Thrust Coefficient
303 CT_Mean = dt/((no_cycles-no_cyc_cut)*Period)*sum(CT_UVLM);
304
305
306 %% Comparison with Theodorsen (Lift)
307 % Theodorsen:   +ve right, +ve down, +ve clockwise
```

```
308
309 [Cl_Theo,Cl_Q_Theo,T4,T10,C,QC,alpha,alphadot,alphadot2,Uinf,...
310     delta,deltadot,deltadot2,hdot2,L] = Theodorsen(b,rho,w,Uinf,H,Alpha,...
311                                         alpha_m,a,e,delta0,Delta,kk,t,lt);
312
313 Cl_Theo = Cl_Theo(Plotcut:lt);
314 Cl_Q_Theo = Cl_Q_Theo(Plotcut:lt);
315 delta_Theo = delta_UVLM;
316 t_Theo = t(Plotcut:lt);
317
318
319 %% Comparison with Leishman (Lift)
320 % Leishman:      +ve right, +ve up, +ve clockwise
321
322 [Cl_Leish,t,delta] = Leishman(t,e,H,Alpha,alpha_m,delta0,Delta,a,b,c,...
323                           Uinf,w);
324 delta_Leish = delta;
325
326 %----Nonlinear fit to give a smoother graph
327 my_func = @(beta,t) beta(1)*sin(beta(2)*t+beta(3));
328 amp = (max(Cl_Leish)-min(Cl_Leish))/2;
329 beta0 = [amp; w; 0];
330 beta = nlinfit(t,Cl_Leish',my_func,beta0);
331 t_Leish = t_UVLM;
332 Cl_Leish = my_func(beta,t_Leish);
333
334
335 %% Root-Mean Square Error
336
337 for i = 1:length(t_UVLM)
338     argument(i) = (Cl_UVLM(i)-Cl_Theo(i))^2;
339 end
340 RMS = 100*sqrt(sum(argument)/length(t_UVLM));
```

```
341
342
343 %% Amplitude & Phase Shift
344
345 AmpL_UVLM = (max(Cl_UVLM)-min(Cl_UVLM))/2;
346 AmpL_Theo = (max(Cl_Theo)-min(Cl_Theo))/2;
347 AmpL_Leish = (max(Cl_Leish)-min(Cl_Leish))/2;
348
349 %----The phase differences are taken relative to the quasi-steady lift of
350      %Theodorsen.
351 PhaseD_UVLM = Phase_diff(Cl_Q_Theo,Cl_UVLM);
352 PhaseD_Theodorsen = Phase_diff(Cl_Q_Theo,Cl_Theo);
353 PhaseD_Leishman = Phase_diff(Cl_Q_Theo,Cl_Leish);
```

.2.2 Theodorsen Code

```
1  function [Cl_Theo,Cl_Q_Theo,T4,T10,C,QC,alpha,alphadot,alphadot2,Uinf,...
2      delta,deltadot,deltadot2,hdot2,L] = Theodorsen(b,rho,w,Uinf,H,Alpha,...
3                                              alpha_m,a,e,delta0,Delta,kk,t,lt)
4
5  % H = -H; %changing sign convention to match Theodorsen's
6
7  %----Pre-Allocation
8  hdot=zeros(lt,1); hdot2=zeros(lt,1); alpha=zeros(lt,1);
9  alphadot=zeros(lt,1); alphadot2=zeros(lt,1); delta=zeros(lt,1);
10 deltadot=zeros(lt,1); deltadot2=zeros(lt,1);
11 hdotC=zeros(1,lt); alphaC=zeros(1,lt); alphadotC=zeros(1,lt);
12 deltaC=zeros(1,lt); deltadotC=zeros(1,lt);
13
14 %----Theodorsen Function
```

```
15 C = besselh(1,2,kk)/(besselh(1,2,kk)+1i*besselh(0,2,kk));
16 phi = angle(C);
17
18 %----Constants
19 T1 = -1/3*sqrt(1-e^2)*(2+e^2)+e*acos(e);
20 T4 = -acos(e)+e*sqrt(1-e^2);
21 T10 = sqrt(1-e^2)+acos(e);
22 T11 = acos(e)*(1-2*e)+sqrt(1-e^2)*(2-e);
23
24 if Alpha ~= 0
25    Alpha = -Alpha;
26    alpha_m = -alpha_m;
27 end
28 if Delta ~= 0
29     Delta = -Delta;
30     delta0 = -delta0;
31 end
32
33 for k = 1:lt
34
35     hdot(k) = w*H*cos(w*t(k));
36     hdot2(k) = -w^2*H*sin(w*t(k));
37     alpha(k) = alpha_m+Alpha*sin(w*t(k));
38     alphadot(k) = w*Alpha*cos(w*t(k));
39     alphadot2(k) = -w^2*Alpha*sin(w*t(k));
40     delta(k) = delta0+Delta*sin(w*t(k));
41     deltadot(k) = w*Delta*cos(w*t(k));
42     deltadot2(k) = -w^2*Delta*sin(w*t(k));
43
44     %----Angles including the phase angle of the Theodorsen function
45     hdotC(k) = w*H*cos(w*t(k)+phi);
46     alphaC(k) = alpha_m+Alpha*sin(w*t(k)+phi);
47     alphadotC(k) = w*Alpha*cos(w*t(k)+phi);
```

```
48      deltaC(k) = delta0+Delta*sin(w*t(k)+phi);
49      deltadotC(k) = w*Delta*cos(w*t(k)+phi);
50
51      %----Quasi-steady lift of Theordorsen
52      Q(k) = ...
            -2*pi*rho*Uinf*b*(Uinf*alpha(k)+hdot(k)+b*(1/2-a)*alphadot(k)+...
53           T10/pi*Uinf*delta(k)+b/(2*pi)*T11*deltadot(k));
54      Cl_Q_Theo(k) = Q(k)/(rho*Uinf^2*b);
55
56      %----Quasi-steady with C(k) effect
57      QC(k) = Uinf*alphaC(k)+hdotC(k)+b*(1/2-a)*alphadotC(k)+...
58           T10/pi*Uinf*deltaC(k)+b/(2*pi)*T11*deltadotC(k);
59
60      %----Total lift on the airfoil-aileron system
61      L(k) = ...
            -rho*b^2*(Uinf*pi*alphadot(k)+pi*hdot2(k)-pi*b*a*alphadot2(k)-Uinf*T4*...
62           deltadot(k)-T1*b*deltadot2(k))-2*pi*rho*Uinf*b*abs(C)*(QC(k));
63      Cl_Theo(k) = L(k)/(rho*Uinf^2*b);
64
65  end
```

.2.3 Leishman Code

```
1   function [Cl_Leish,t,delta] = Leishman(t,e,H,Alpha,alpha_m,delta0,Delta,...
2                                       a,b,c,Uinf,w)
3
4   H = -H; %changing sign back to match Leishman's convention
5
6   %----Results from second-order step response approximation to Wagner's.
7       %A nonlinear least-squares fit using a constrained optimization
```

```
8       %algorithm was used by Leishman.
9   A1 = 0.2048;
10  A2 = 0.2952;
11  b1 = 0.0557;
12  b2 = 0.333;
13
14  %----Geometric Coefficients
15  F1 = e*acos(e)-1/3*(2+e^2)*sqrt(1-e^2);   %F1=T1
16  F4 = e*sqrt(1-e^2)-acos(e);               %F4=T4
17  F10 = sqrt(1-e^2)+acos(e);                %F10=T10
18  F11 = (1-2*e)*acos(e)+(2-e)*sqrt(1-e^2);  %F11=T11
19
20  %----Integration of controllable canonical form
21  t0 = 0;
22  tf = t(end);
23  tspan = [t0 tf];
24  z0 = [0 0 0 0];
25  ode_func = @(t,z) ...
        Leishman_nested(t,z,w,H,Alpha,alpha_m,delta0,Delta,a,b,...
26      Uinf,b1,b2,F10,F11);
27  options = odeset('RelTol',1e-9);
28  [t,z] = ode45(ode_func,tspan,z0,options);
29
30  %----Displacements/Angles and derivatives
31  hdot = w*H*cos(w*t);
32  hdot2 = -w^2*H*sin(w*t);
33  alpha = alpha_m+Alpha*sin(w*t);
34  alphadot = w*Alpha*cos(w*t);
35  alphadot2 = -Alpha*w^2*sin(w*t);
36  delta = delta0+Delta*sin(w*t);
37  deltadot = w*Delta*cos(w*t);
38  deltadot2 = -w^2*Delta*sin(w*t);
39
```

```
%----Lift Coefficient
for i = 1:size(t)

    %----Quasi-steady angles
    alpha_qs(i) = hdot(i)/Uinf+alpha(i)+b*(1/2-a)*alphadot(i)/Uinf;
    delta_qs(i) = F10*delta(i)/pi+b*F11*deltadot(i)/(2*pi*Uinf);

    %----Airfoil motion (alpha)
    Cl_a(i) = 2*pi*[(b1*b2/2)*(Uinf/b)^2,(A1*b1+A2*b2)*(Uinf/b)]*...
        [z(i,1); z(i,2)]+pi*alpha_qs(i);

    %----Flapping motion (delta)
    Cl_d(i) = 2*pi*[(b1*b2/2)*(Uinf/c)^2,(A1*b1+A2*b2)*(Uinf/b)]*...
        [z(i,3); z(i,4)]+pi*delta_qs(i);

    %---Non-circulatory
    Cl_nc(i) = ...
       pi*b/Uinf*(hdot2(i)/Uinf+alphadot(i)-(b*a/Uinf)*alphadot2(i))+...
        b/Uinf^2*(-Uinf*F4*deltadot(i)-b*F1*deltadot2(i));

end
Cl_Leish = Cl_a+Cl_d+Cl_nc;
```

.2.4 Phase Difference Code

```
% Credit for the outline of this code goes to Shashank G. Sawant
% PhaseD: output phase difference (in radians)
% Cl: first sinusoidal signal
% Cl2 (or Cl3): second sinusoidal signal
% Note: Cl and Cl2 (or Cl3) should have the same frequency
```

```
6  function PhaseD = Phase_diff(Cl,Cl3)
7  L = length(Cl);
8  if(L ~= length(Cl3))
9      error('You are definitely dumb. The length of the two sinusoidal ...
           input vectors is not same!');
10 end
11
12 %----The following block calculates the FFT
13 NFFT = 2^nextpow2(L);
14 CL = fft(Cl,NFFT)/L; %Fourier Transform of signal Cl
15
16 %----The following block calculates the phase of the most significant
17     % frequency component
18 [value,index] = max(2*abs(CL(1:NFFT/2+1)));
19 Phase_Cl = angle(CL(index));
20
21 %----The following block calculates the FFT
22 NFFT = 2^nextpow2(L);
23 CL3 = fft(Cl3,NFFT)/L; %Fourier Transform of signal Cl3
24
25 %----The following block calculates the phase of the most significant
26     % frequency component
27 [value,index] = max(2*abs(CL3(1:NFFT/2+1)));
28 Phase_Cl3 = angle(CL3(index));
29
30 %----The following is the phase difference between the two signals
31 PhaseD = Phase_Cl-Phase_Cl3;
32
33 %----The code below limits the output to PhaseD={-pi,pi}radians
34  if PhaseD > pi
35        PhaseD = PhaseD-2*pi;
36  elseif PhaseD < -pi
37        PhaseD = PhaseD+2*pi;
```

```
38   else
39         PhaseD = PhaseD;
40   end
41
42  return
43  end
```

References

[1] Theodore Theodorsen. General Theory of Aerodynamic Instability and the Mechanism of Flutter. Technical Report NACA-TR-496, National Advisory Committee for Aeronautics. Langley Aeronautical Lab., Langley Field, VA, United States, January 1935.

[2] I.E. Garrick. Propulsion of a Flapping and Oscillating Airfoil. Technical Report NACA-TR-567, National Advisory Committee for Aeronautics. Langley Aeronautical Lab., Langley Field, VA, United States, January 1937.

[3] Wagner, Herbert: ´Uber die Entstehung des dynamischen Auftriebes von Tragfl´ugeln. Z. f.a. M. M., Band 5, Heft 1, Feb. 1925,S. 17-35.

[4] von Kármán, Th., and Burgers, J. M.: General Aerodynamic Theory—Perfect Fluids. Aerodynamic Theory, W. F. Durand, ed., vol II, Julius Springer (Berlin), 1935.

[5] J. Gordon Leishman. Unsteady lift of a Flapped airfoil by Indicial Concepts. *Journal of Aircraft*, 31(2):288-297, 1994.

[6] von Kármán, Th., and Sears, W.R., "Airfoil Theory for Non-Uniform Motion," *Journal of the Aeronautical Sciences*, Vol 5, No. 10, 1938, pp. 379-390.

[7] Jones, R.T, "Calculation of the Motion of an Airplane Under the Influence of Irregular Disturbances," *Journal of the Aeronautical Sciences*, Vol 3, No. 12, 1936, pp. 419-425.

[8] Pistolesi, Enrico: Aerodinamica. Unione Tipografico—Editrice Torinese, 1932.

[9] Ramesh, Kiran, "An Unsteady Airfoil Theory Applied to Pitching Motions Validated Against Experiment and Computation," *Theoretical Computational Fluid Dynamics*, **27**, 843-864 (2013).

[10] Jones, R. T.,"Operational Treatment of the Non-Uniform Lift Theory in Airplane Dynamics." 1938. Technical Note 667, NASA.

[11] Katz, J., and Plotkin, A., *Low-Speed Aerodynamics*, Cambridge Univ. Press, Cambridge, England, U.K., 2001, pp. 369-447.

[12] Simpson, R. J. S., and Palacios, R, "Induced-Drag Calculations in the Unsteady Vortex Lattice Method," *AIAA Journal*, Vol 51, No. 7, 2013, pp. 1775-1779.