Simulation of Gust Generator-Induced Wind Tunnel Flow Fields

Noah Aidan Conrad

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Aeronautics and Astronautics

University of Washington

2020

Reading Committee:

Eli Livne, Chair

Carl Knowlen

Program Authorized to Offer Degree:
Aeronautics and Astronautics

University of Washington

Abstract

Simulation of Gust Generator-Induced
Wind Tunnel Flow Fields

Noah Aidan Conrad

Chair of the Supervisory Committee:
Professor Eli Livne
Aeronautics and Astronautics

A mathematical/numerical investigation was carried out of the flow characteristics of wind tunnel test section flow influenced by a gust generation system. The University of Washington's 3'x3' Low-Speed wind tunnel was modeled using high-fidelity Computational Fluid Dynamics (CFD) simulations and using easy-to-program and inexpensive-to-run vortex lattice methods (VLM). Steady and unsteady CFD flow simulations were completed to study the nature of the flow field created by the gust generator in the wind tunnel. The gust generator induced gust angle-of-attack and the velocity of the flow field at various locations in the test section were determined and the aerodynamic effect on a test wing was also studied. In the time marching vortex lattice method developed, accounting for the effect of the walls of the wind tunnel on the forces felt by the test object was done via a system of mirror images. The VLM simulations were compared to classical Theodorsen solutions and to the CFD simulations to gauge the accuracy of the VLM simulations. In the presence of wind tunnel walls, the VLM simulations overestimated the forces felt by the test object when compared to the force predictions of the CFD simulations, but overall captured the time behavior well. Knowing the magnitude of difference between CFD and VLM results, VLM simulations can still be a useful tools, fast and inexpensive, for the design of wind tunnel gust generation systems and the positioning of wind tunnel models relative to them.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank Professor Livne for all his support and attention over the past two years. He served as an excellent role model for me in my studies and in my future career. His guidance challenged and engaged me, but also allowed me to enjoy my time at the University of Washington. For all that, I am very appreciative.

I would also like to thank Professor Knowlen for his help in writing and presenting my thesis.

Finally, I would like to thank my parents for their support throughout this entire process.

DEDICATION

To Mom, Dad, Liam and Kagan

Chapter 1

INTRODUCTION

1.1 Background and Motivation

Innovations in novel aerospace materials such as composites and wing design optimization lead to lighter airplanes. Lighter airplanes are desirable because they are cheaper and more fuel-efficient [27]. However, the design changes and the associated positive outcomes are also accompanied by increases in airframe flexibility. As [25] explains, more flexible wings are more sensitive to gust loading. This is problematic because elevated gust loading can damage the aircraft and limit the aircraft's effective lifespan as cited in [25], [30], and [31]. The need to meet dynamic gust load structural integrity requirements may lead to added structural weight. Gust excitation and the way aircraft respond to them also affect passenger and crew ride comfort [31]. The design for desirable gust response is, therefore, an important part of the multidisciplinary design of the modern airplane.

To eliminate the need for beefing up the structure to withstand gust loads in a passive way, Gust Load Alleviation (GLA) has been known and used for years to limit gust internal loading, especially for newer, lighter, and more flexible wings. Active GLA [25] has the potential to decrease the structural loading imposed on the aircraft and improve ride comfort. The result can be lighter and cheaper feasible designs [11]. Research activity in the area of control laws synthesis for gust alleviation has been an active field of research for years [25], [31], and [39]. But development of GLA control laws and systems requires an ability to work with mathematical models that are accurate and experimentally validated. New control laws can be studied theoretically or tested using numerical simulation methods, including high-fidelity modeling based on computational fluid dynamics (CFD). However, as [19] described, many flow fields, unsteady flow fields especially, are difficult to capture theoretically or by

using CFD. Therefore, some form of experimental validation is often still necessary.

1.2 Gust Generators

A gust generator in a wind tunnel is a device that sends gusts of predetermined shape and intensity into the wind tunnel's test section to excite aeroelastic wind tunnel models installed there. By creating a known and repeatable test section flow field (see [25], [11], and [19]), a gust generation system can be instrumental in developing and testing new control laws designed to alleviate gust loading. There exists a multitude of possible designs for gust generation systems in wind tunnels. Examples include a rig designed to oscillate an object of interest in a steady flow to simulate a stationary object in oscillating flow [5], a towing tank [5], a by-pass duct to control test section velocity [35], a harmonically-plunging flat plate [17], a series of flexible nozzles tilted up and down [26], a rotating slotted cylinder (RSC) located at the trailing edge of an airfoil [36], [41], a device to pump high-pressure air out the trailing edge of an airfoil [38], a modified circulation-controlled airfoil (CCA) [14], a moving bump which travels along the wall of a tunnel [12], a lattice of individual fans [42], or a group of oscillating airfoils at the inlet of the test section [11], [19], [25]. The University of Washington (UW) 3' x 3' Low-Speed wind tunnel (3x3 tunnel) is equipped with a gust generation system made of oscillating vanes at the entrance to its test section.

The UW's 3x3 low-speed tunnel is capable of wind speeds between 15 m/s and 60 m/s (34-135 mph). The test section is 3 ft by 3 ft by 8 ft. The gust generation system consists of two gust vanes placed vertically at the beginning of the test section, stretching from floor to ceiling. The gust vanes are slightly modified NACA0012 airfoils with a chord length of 10 in. Each vane is located 11 in from the closest wall and 14 in from the other vane, (+-7 in from centerline).

The system is capable of vane oscillation amplitudes of up to 13 degrees at low frequencies and up to 10 degrees at frequencies up to 50 Hz. In additional to sinusoidal motions the system can produce 1-cos gust at frequencies up to 50Hz.

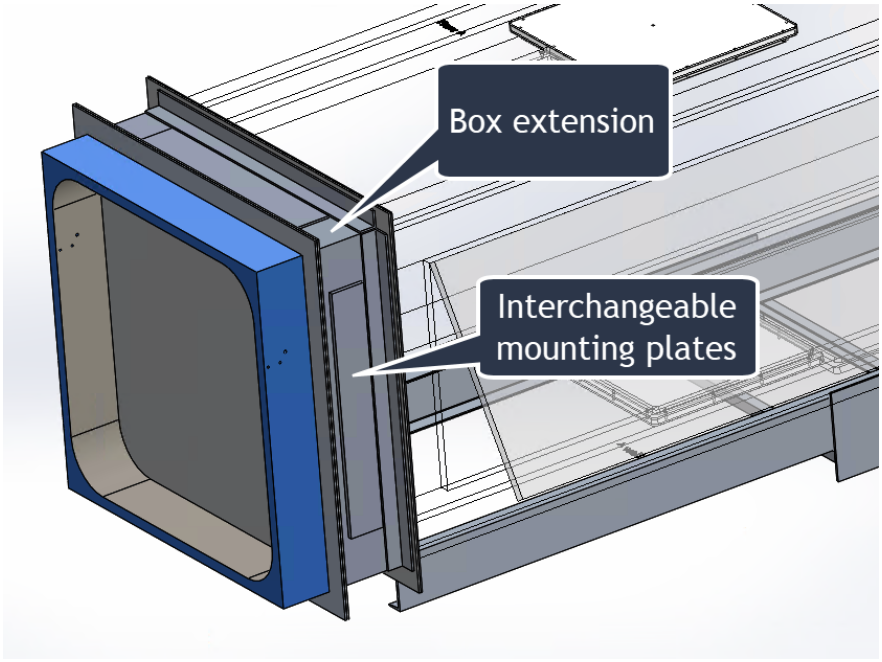

Figure 1.1: Mounting Plate to which Gust Vanes are Attached

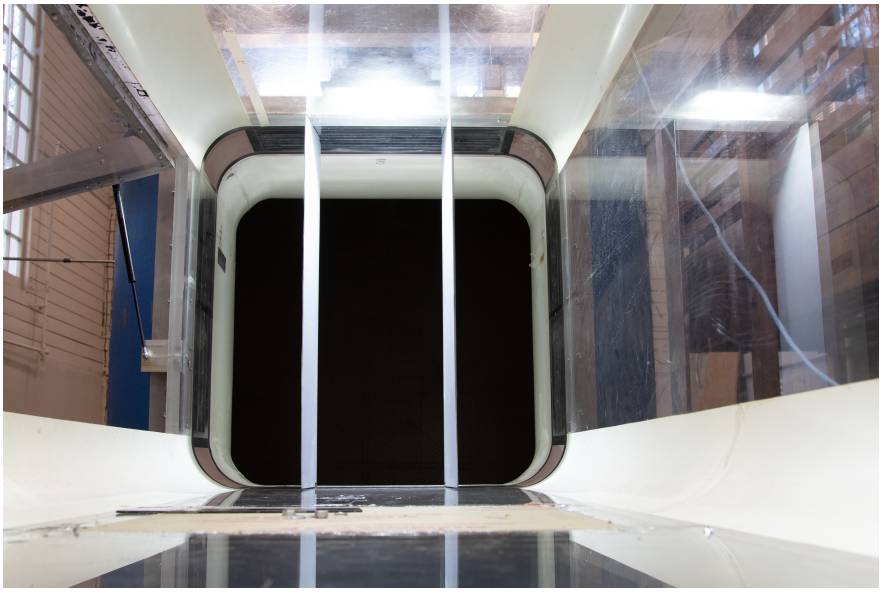

Figure 1.2: Gust Generator Configuration in Wind Tunnel

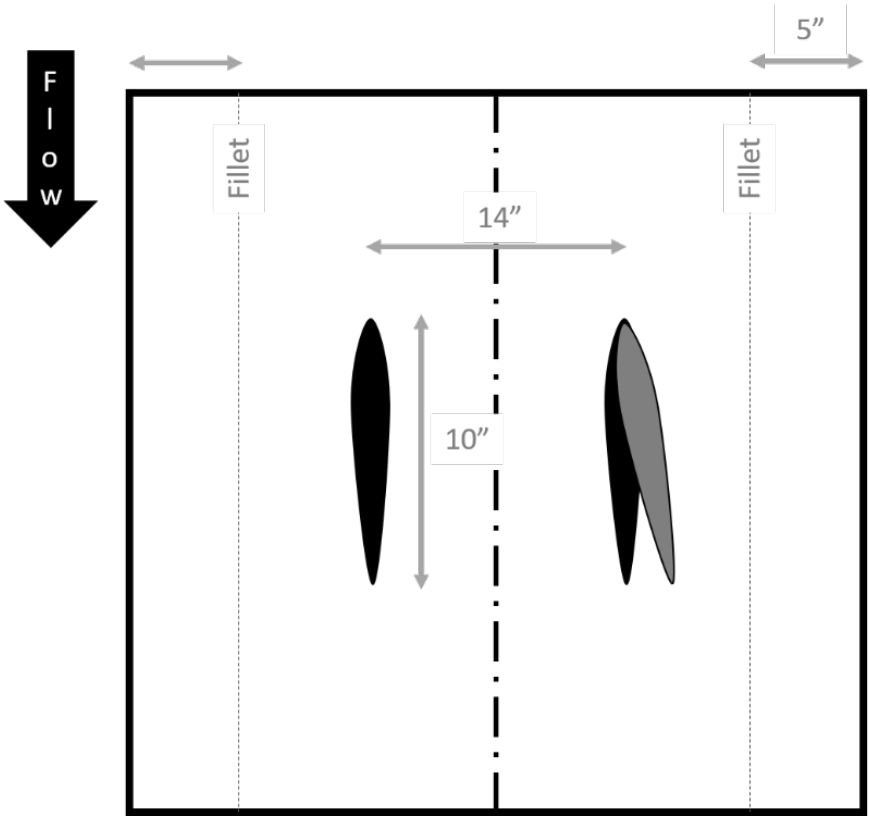

Figure 1.3: Gust Vanes Geometry: Top View

1.3 Flow Fields Created by Gust Generation Systems

In order to support gust load alleviation tests in the 3x3 tunnel, it was important to determine the characteristics of the flow field created by the gust system.

1.3.1 Experimental Analysis Methods

The flow fields of gust generation systems can be measured and mapped experimentally. Aeroelastic wind tunnel gust response test results can be compared to analytical predictions and flight test results, as shown in [29]. Hot-wire anemometer systems can also be used for flow field determination in the wind tunnel [5], [33], [35]. [3] and correlated with recorded forces on the test object using load cells. Other methods can be used too. [22] utilized

particle image velocimetry (PIV) to map the wind tunnel's flow field under the influence of the gust generator while [16] used an acoustic Doppler velocimeter to do the same.

1.3.2 Computational Analysis Methods

Computational methods can be used to analyze the flow field to complement experimental methods or when test results are not available. The nature of the flow fields created by gust generation systems in wind tunnels were previously determined in multiple ways by [25], [11], [19], [22], [24], [40], and [37] via the use of high-fidelity CFD. For example, [11] utilized an unsteady 2D CFD simulation using the OpenFoam code to determine the gust angle in the test chamber. A 2D simulation was used in their case because of geometry in the Polytechnic of Milan wind tunnel that did not change in the height direction. [19] employed ANSYS Fluent to run 2D time-transient simulations in order to study gust uniformity and flow interactions between gust vanes. Gust velocity angles in cross-sections of the tunnel for various distances behind the gust vanes were also considered. OpenFOAM with the PIMPLE solver was used by [25] to solve an unsteady, incompressible flow simulation. [25] completed simulations involving the gust vanes only and simulations involving both the gust vanes and an airfoil test airfoil in the test section. The gust vanes-only simulations were used to characterize the induced flow in the test section, with studies focusing on both the entire field and on centerline velocities more specifically. Additionally, gust angle amplitudes, mean gust angles, and transport delays were determined. Simulations involving both the gust vanes and an airfoil test object enabled determination of the loading induced on the airfoil and the flow induced by the gust system in the presence of an additional test object. That is, the simulations allowed studying the coupling between the gust vanes and the wing test object. Other solvers and other validation methods were employed by [22], [24], [40], and [37].

1.3.3 Other Analysis Methods

The downside of the use of CFD to determine the flow fields created by gust generation systems is the time and computational power needed to build computational meshes and

find solutions. Altough high-fidelity CFD models are capable of capturing nonlinear effects and flow field behavior in the presence of viscous effects (especially when no major separation is involved), it would be advantageous to have a faster method for estimation of a flow field, even if it may be slightly less accurate, as long as it would be quick to program and fast to run. Vortex models have already been used to efficiently determine flow field behavior under the influence of oscillating gust vanes in a fluid [15], especially in the 2D case. Therefore, for this work, vortex models were considered for potential suitability.

A vortex lattice method (VLM) was used to compute unsteady incompressible flow over an airfoil for the 2D case. [13] described dividing the airfoil of interest and the airfoil's wake into numerous elements. A discrete vortex was placed at the 1/4-chord of each element and a collocation point was placed at the 3/4-chord of each element. The circulation of each vortex was determined by requiring "the velocity induced by the discrete vortices to equal the downwash arising from the unsteady motion of the airfoil" [13]. The formulation developed by [13], originally for an oscillating airfoil in steady incoming flow, was applied by [7] to solve the pulsating-incoming-flow problem and by [18] to develop a new aeroservoelastic model with only real coefficients in the reduced-order model. [9] developed a simplified marching VLM for the 2D unsteady incompressible problem, motivated by his interest in flapping wings. The [9] method involved using a single element to describe the airfoil. A single vortex was modeled at the airfoil's 1/4-chord describing the airfoil's circulation. A single collocation point was modeled at the airfoil's 3/4-chord, used for describing the airfoil's motion. At each step, a new vortex was shed into the wake a prescribed distance behind the airfoil and all vortices in the wake moved a discrete distance in the wake as well. The strength of the shed vortex was determined by the "corresponding change in the bound [airfoil] vortex strength based on Helmholtz vorticity conservation law" [9]. The formulation of the marching VLM developed by [9], followed by [15] and [13], if sufficiently accurate, would be significantly faster and less computationally expensive than a comparative model using CFD.

1.4 Goals and Objectives

The present work builds upon [9]'s marching VLM method with the goal of quantifying the flow field created by the gust generation system in the UW 3x3 wind tunnel. The marching VLM, presented in [9], did not account for the effects of the walls of a wind tunnel or the effect of airfoils on each other in a multi-airfoil system - in our case: the individual gust vanes and the test wing. As is well known, [24], [1] and [6], walls can significantly impact the flow fields created by gust generators as well as flow fields around wind tunnel models. As [24] determined during gust generator design and validation, flow reflection off walls must be accounted for to achieve accurate results. Following a well established practice in low-speed aerodynamics to account for wall effects, a mirroring method is used in conjunction with the marching VLM here to account for the presence of walls. Such mirroring is described by [32]. In this work the marching VLM is compared to CFD simulations of the gust generation system in the wind tunnel to determine the accuracy of the marching VLM predictions and to determine the marching VLM's fitness for use in future wind tunnel GLA experiments.

1.5 Report Structure

Chapter 2 of the thesis describes the computational CFD approach used in this work. Chapter 3 presents the results of analysis validation. Validation of the computational CFD modeling adopted was used to build confidence in the CFD results in the case of no wind tunnel walls (for which other solutions are available) before proceeding to use the CFD models to simulate the flow fields in the wind tunnel. Chapter 4 presents the findings of the CFD computational studies regarding the flow field in the wind tunnel under a variety of conditions. The work now proceeds to present and discuss the marching vortex lattice method used. The theory underlying the vortex lattice methods is presented in chapter 5. VLM results are presented in chapter 6. Finally, the VLM results are compared to the CFD results in chapter 7. Chapter 8 concludes and summarizes the findings of the work.

Chapter 2

COMPUTATIONAL APPROACH

2.1 Governing Equations

In the CFD simulations the flow in the wind tunnel was modeled as viscous flow with a representative Reynolds number around $Re \approx$ 970000. The Reynolds Averaged Navier-Stokes (RANS) equations were solved. Velocity was averaged in the following way:

$$U_i = \overline{U}_i + u_i \tag{2.1}$$

where

$$\overline{U}_i = \frac{1}{\triangle t}\int_t^{t+\triangle t} U_i dt \tag{2.2}$$

The following continuity and momentum equations were used:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial t}(\ \rho U_j) \ = 0 \tag{2.3}$$

$$\frac{\partial \rho U_i}{\partial t} + \frac{\partial}{\partial x_j}(\ \rho U_i U_j) \ = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}(\ \tau_{ij} - \rho\overline{u}_i\overline{u}_j) \ + S_M \tag{2.4}$$

The turbulence model used in simulations was the k-ω based Shear Stress Transport (SST) model. The SST model is based on the Wilcox k-ω model, which "solves two transport equations, one for turbulent kinetic energy, k, and one for the turbulent frequency, ω" [2]. The Wilcox k-ω model is:

$$\frac{\partial(\ \rho k)}{\partial t} + \frac{\partial}{\partial x_j}(\ \rho U_j k) \ = \frac{\partial}{\partial x_j}[\ (\ \mu + \frac{\mu_t}{\sigma_\omega})\ \frac{\partial k}{\partial x_j}] \ + P_k - \beta'\rho k\omega + P_{kb} \tag{2.5}$$

$$\frac{\partial(\ \rho\omega)}{\partial t}+\frac{\partial}{\partial x_j}(\ \rho U_j\omega)\ =\frac{\partial}{\partial x_j}[\ (\ \mu+\frac{\mu_t}{\sigma_\omega})\ \frac{\partial\omega}{\partial x_j}]\ +\alpha\frac{\omega}{k}P_k-\beta\rho\omega^2+P_{\omega b} \tag{2.6}$$

The Wilcox model shows "strong sensitivities to freestream conditions" [2] so it was combined with the k-ϵ model, as demonstrated by [23], using blending functions. In the modified model, called the Baseline (BSL) k-ω model, the k-ω model is applied near the surface and the k-ϵ model is applied away from the surface.

$$\frac{\partial(\ \rho k)}{\partial t}+\frac{\partial}{\partial x_j}(\ \rho U_j k)\ =\frac{\partial}{\partial x_j}[\ (\ \mu+\frac{\mu_t}{\sigma_\omega})\ \frac{\partial k}{\partial x_j}]\ +P_k-\beta'\rho k\omega+P_{kb} \tag{2.7}$$

$$\frac{\partial(\ \rho\omega)}{\partial t}+\frac{\partial}{\partial x_j}(\ \rho U_j\omega)\ =\frac{\partial}{\partial x_j}[\ (\ \mu+\frac{\mu_t}{\sigma_{\omega 3}})\ \frac{\partial\omega}{\partial x_j}]+(\ 1-F_1)\ 2\rho\frac{1}{\sigma_{\omega 2}\omega}\frac{\partial k}{\partial x_j}\frac{\partial\omega}{\partial x_j}+\alpha_3\frac{\omega}{k}P_k-\beta_3\rho\omega^2+P_{\omega b} \tag{2.8}$$

The baseline k-ω model is further amended to the k-ω based SST model (the model used in the simulations) by accurately accounting for the transport of turbulent shear stress using "a limiter to the formulation of the eddy-viscosity" [2].

$$\nu_t=\frac{a_1 k}{max(\ a_1\omega, SF_2)} \tag{2.9}$$

where

$$\nu_t=\frac{\mu_t}{\rho} \tag{2.10}$$

The k-ω based SST model was chosen following some simulation shortcomings discovered while using the k-ϵ model and due to precedent set by previous research, specifically [25].

In addition, no heat transfer, combustion, thermal radiation, or buoyancy effects were considered.

2.2 Numerical Solver

The ANSYS CFX solver was used. Steady-state simulations utilized high-resolution turbulence numerics and a high-resolution advection scheme. Convergence criteria was based on

a root-mean-squared (RMS) convergence type with a residual target of 0.000001.

2.3 Computational Model

All CFD simulations were two-dimensional. As stated by [11], a 2D assumption was reasonable given the model was uniform in the y-direction (vertical) where, regarding figures in this work, the y-direction is in and out of the page. A 2D model is also much less computationally expensive than a 3D model of the system and the test section.

The computational CFD model was based on the geometry of the wind tunnel and the gust generator. In the 3 ft x 3 ft cross section of the tunnel, on a side view, the top wall was 18 in above centerline and the bottom wall was 18 in below centerline. The two gust vanes consisted of two NACA airfoils for which thickness could vary to study its effects. The vanes are shown in the side views located one on top of the other. The airfoils were identical, with a chord length of 10 in. The airfoils were placed symmetrically about the centerline of the tunnel. Each was positioned 7 in from the center, measured from the airfoil's centerline. The airfoils were 11 in from the nearest wall.

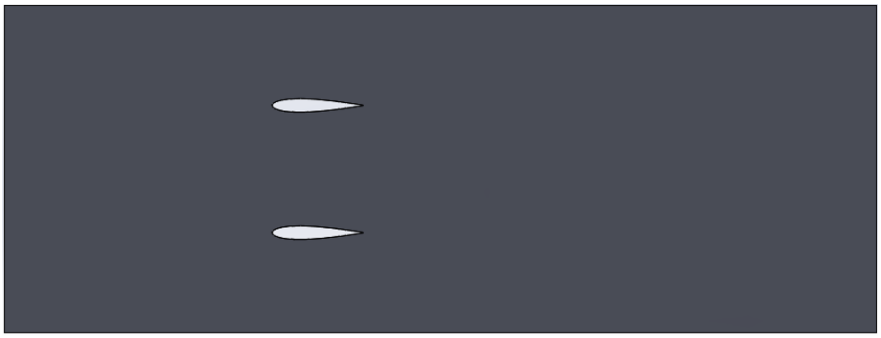

Figure 2.1: Basic Computational Model

Multiple variants of the basic computational model were considered. A single-airfoil configuration was considered with an airfoil located at the centerline of the tunnel. The single-airfoil configuration was modified for airfoils with different thicknesses, angles of attack

and chord lengths. The proximity of the walls to the airfoil was also changed in an effort to study the effect of this important parameter.

Figure 2.2: Single Airfoil Configuration Example

A three-airfoil configuration was considered with two symmetric NACA 00XX airfoils, arranged one on top of the other, as seen in the gust generation system, with one NACA 0012 airfoil, as the test airfoil, located on the tunnel centerline. The gust vanes and the test wing formed a coupled system. The distance of the test airfoil behind the gust vanes, as well as the test airfoil's angle of attack, were changed to evaluate their effects.

Figure 2.3: Three-Airfoil Configuration Example

A four-airfoil configuration was considered with two NACA 00XX airfoils (thickness varied in studies), arranged one on top of the other to create the gust generation system, with one NACA 0012 airfoil, as the test wing, located on the tunnel centerline, and with a test tail, another NACA 0012 airfoil, located 27 in behind the test wing to simulate wing a horizontal

tail in a flow field affected by the gust system. The gust vanes and the wing and tail structure formed a coupled system.

Figure 2.4: Four-Airfoil Configuration Example

2.4 Meshing Methodology

ICEM CFD was used to mesh all models. ICEM CFD was selected based on availability and on compatibility with the ANSYS CFX solver. A block structured grid was used to focus mesh fineness around the areas of interest. The mesh was concentrated around the airfoils and the airfoils' wakes. An H-H grid meshing technique was used. Clustering was applied along the edges of the airfoils to ensure finer mesh along the edges. Finer mesh along the edges enabled more accurate capture of boundary-layer behavior. H-grids were chosen for simplicity of construction [8]. A BiGeometric bunching law was employed. The mesh was constructed primarily of hexahedral and quadratic elements. The number of elements used was generally $\sim 10^5$ elements. However, the number of elements required was dictated by the complexity of the model's geometry.

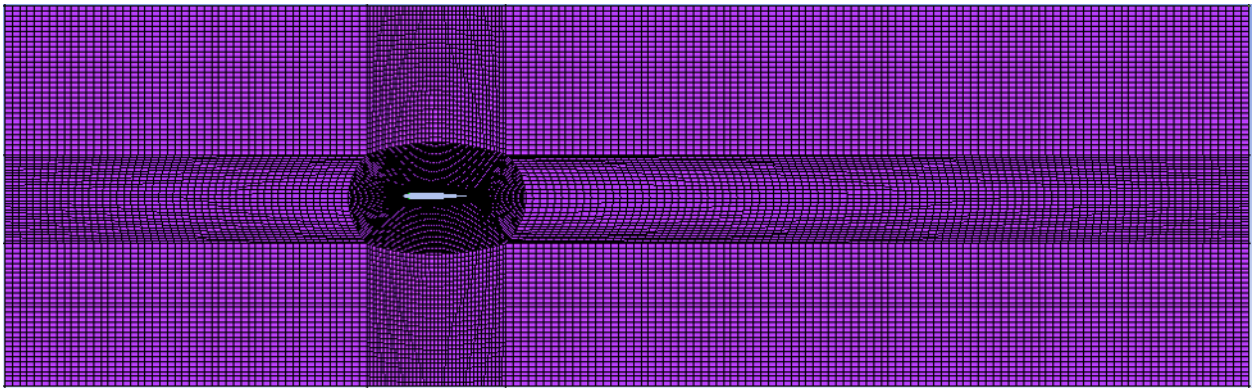

Figure 2.5: Mesh Example, Full Model

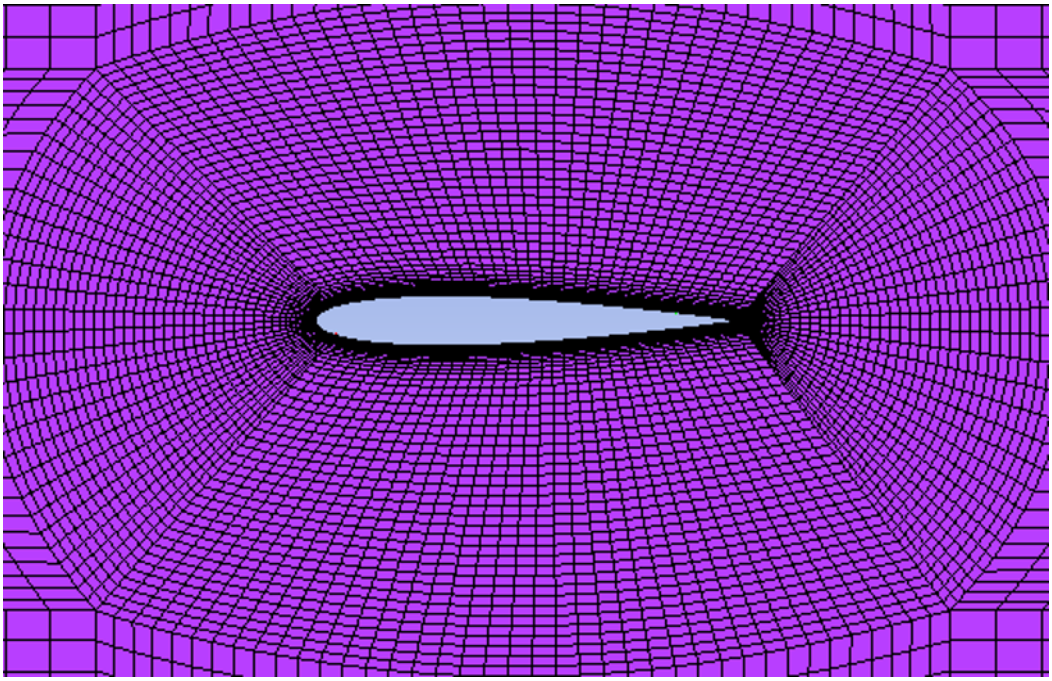

Figure 2.6: Mesh Example, Airfoil Close

2.5 Boundary Conditions

Test section inlet and outlet boundary conditions were specified in ANSYS CFX. Boundary conditions were the same for all CFD simulations. The inlet boundary was set for a speed

(normal to the face of the inlet) of 60 m/s. The outlet boundary was set for an average static pressure. The static pressure at the outlet was calculated based on the static pressure at the entrance to the wind tunnel and the flow velocity in the wind tunnel. The average static pressure at the outlet was 99.195 kPa. The top and bottom walls of the model and the airfoils were considered no-slip walls. The two faces of the model were related using a translational periodicity interface model to enforce the simulation's 2D nature.

2.6 Unsteady Simulation Methodology

The preceding description generally applied to both the steady-state and unsteady simulations. The steady-state and unsteady simulations shared the same model geometries, meshes, and boundary conditions. This is expected, as the steady-state simulations served as the initial conditions to the unsteady simulations. However, the unsteady simulations required slightly different governing equations and numerical solver setups. The unsteady simulations also required mesh motion, a feature not seen in the steady-state runs.

For unsteady flow, Unsteady Reynolds Averaged Navier-Stokes (URANS) equations were used, instead of RANS equations. URANS equations add an additional term in the RANS momentum equation to account for unsteady effects [34]. For unsteady simulations, a transient analysis solution technique was used. The transient scheme was a Second-Order Backward Euler scheme. The simulation used 32 timesteps to complete one period of oscillation.

Simulated airfoil oscillation was prescribed, as mentioned previously, using mesh deformation. The mesh motion model was a displacement-diffusion model with the displacement relative to the previous mesh [2]

$$\nabla \cdot (\Gamma_{disp} \nabla \delta) \ = 0 \tag{2.11}$$

The mesh stiffness was defined as 1/Volume Of Finite Volumes. The stiffness of each element varied inversely to the volume of the element. Airfoil mesh motion was defined using expressions relating the airfoil's desired position to a desired oscillation pattern for an airfoil with its origin at its 1/4-chord and centerline [20].

$$X_{airfoil}(\ t)\ = X_0 \cos\alpha_i - Y_0 \sin\alpha_i \tag{2.12}$$

$$Y_{airfoil}(\ t)\ = X_0 \sin\alpha_i + Y_0 \cos\alpha_i \tag{2.13}$$

where

$$\alpha_i = \theta_{max} \sin\frac{2\pi t}{T} \tag{2.14}$$

For airfoils not located with the airfoil's 1/4-chord at the origin, modifications to the expressions above were used to correctly define the desired airfoil motion. The airfoils in the unsteady simulation were considered no-slip walls as in the steady-steady simulations.

Chapter 3

COMPUTATIONAL VALIDATION OF CFD RESULTS

Computational validation, using known results, was required before the gust generation system was analyzed in order to ensure the CFD analysis produced reliable results. A mesh convergence study was completed to determine the mesh density needed for effective modeling. A single-airfoil case with no walls was considered to compare the lift-curve slope of a single airfoil measured by the CFD simulation to well-known values. The thin-airfoil unsteady Theodorsen solution, additionally, was compared to three unsteady simulations with different reduced frequencies to ensure proper modeling of unsteady phenomena. Finally, a positive-negative angle of attack (AOA) comparison was carried out to show that the CFD solver produced symmetric results. The validation tests built confidence in the subsequent CFD findings.

3.0.1 Mesh Convergence Study

A mesh convergence study was conducted to determine the required mesh fineness for reliable results. Models were blocked and meshed in the same manner for each repetition. However, the density of the mesh was controlled by scaling all meshing parameters evenly to modify the number of elements by approximately an order of magnitude each time.

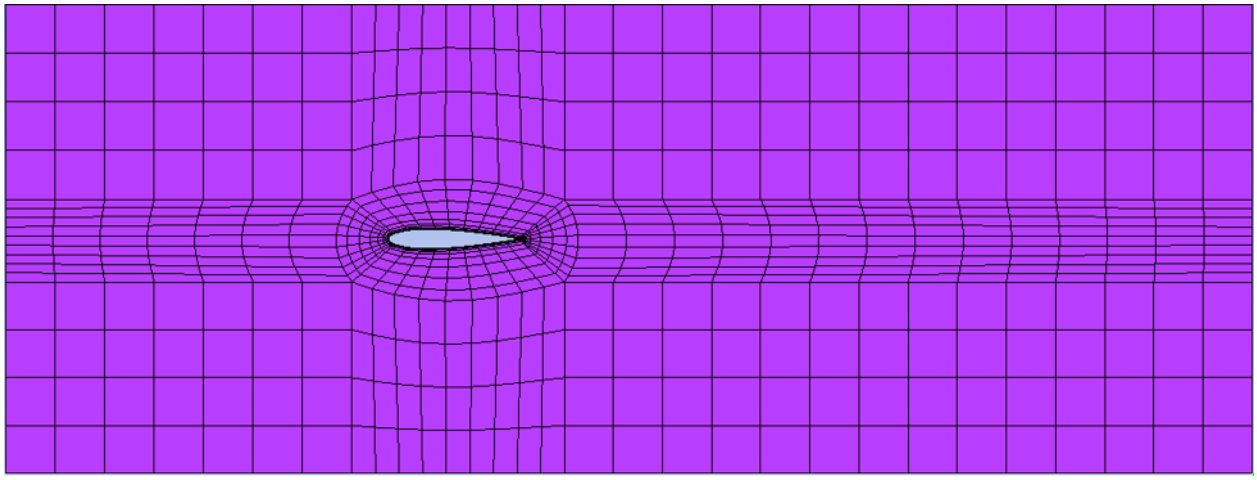

Figure 3.1: Mesh with 10^3 elements

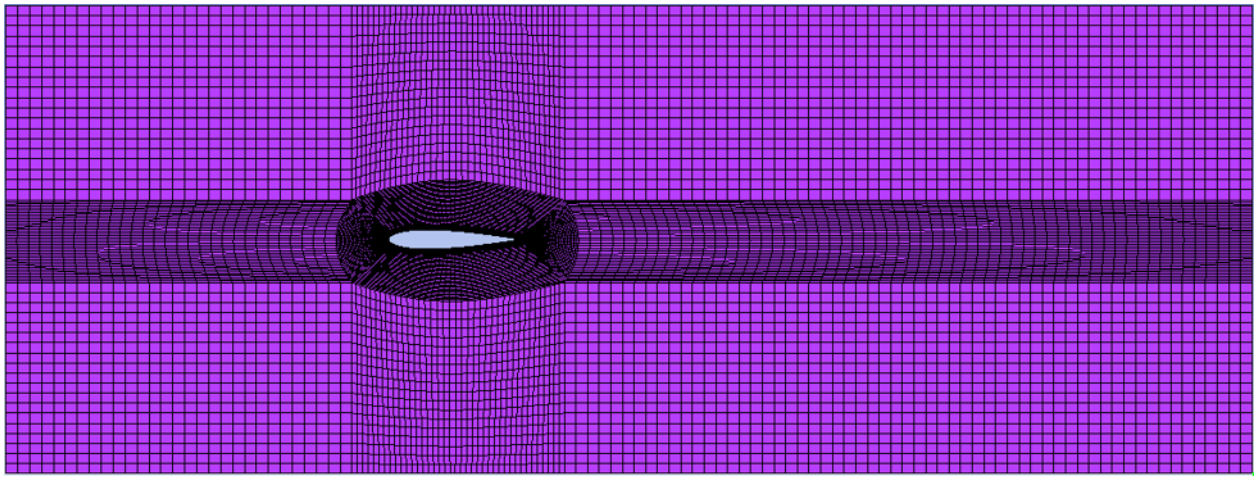

Figure 3.2: Mesh with 10^5 elements

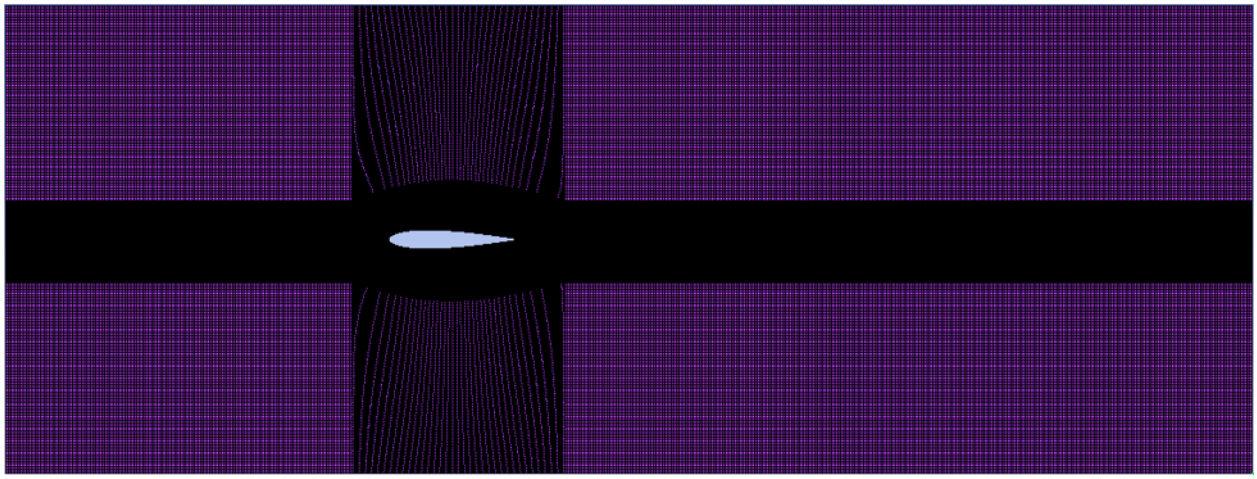

Figure 3.3: Mesh with 10^7 elements

Each mesh was used in an unsteady simulation. The airfoil was oscillated at 1 Hz with a maximum amplitude of oscillation of 10 degrees. Note that a 10" chord airfoil (chord=0.254 m) and a tunnel speed of 60 m/sec corresponds to a reduced frequency of 0.0133. The maximum normalized lift force (N/m) was calculated for each simulation. The lift forces from the various simulations were compared as functions of the number of elements. The results are plotted on a log-linear plot.

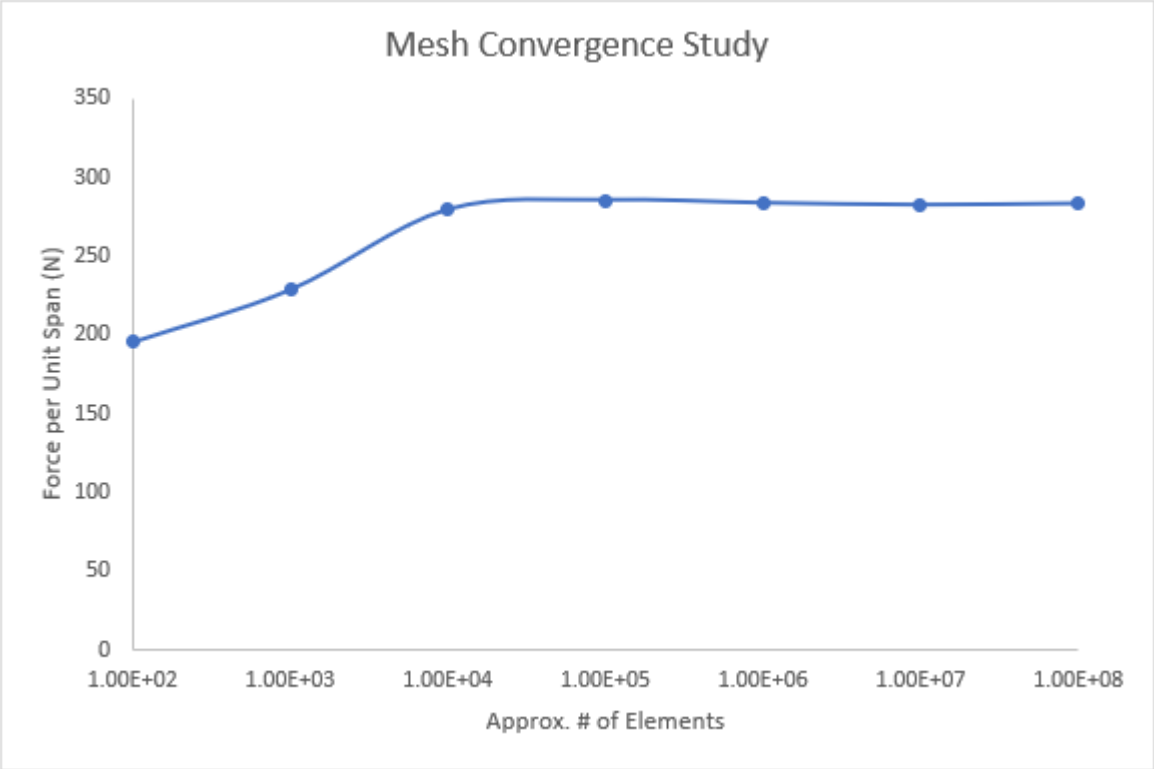

Figure 3.4: Mesh Convergence Study Results

The results of the mesh convergence study suggested, for the given model, that $\sim 10^4$ elements were sufficient for reliable calculations. A coarser mesh, it seemed, would drive the solver to underestimate the lift force acting on the airfoil and could not be trusted. Based on the above results, a minimum number of elements of $\sim 10^5$ elements was selected. A mesh of $\sim 10^5$ elements was selected because it was comfortably within the zone of

convergence demonstrated by the mesh convergence study and was computationally efficient. An unnecessarily fine mesh would slow computations significantly.

3.1 Single Airfoil Lift-Curve Slope Study

A single-airfoil case was considered in order to check if the CFD simulations could accurately reproduce the airfoil's theoretical lift-curve slope. The airfoil used for the study was a NACA 0015 airfoil with a chord length of 10 in. The airfoil was simulated at several angles of attack (AOAs) using steady-state simulations. The AOAs tested were 0 degrees, 3 degrees, 4.5 degrees, 6 degrees and 9 degrees. The dynamic pressure was calculated to be 2130 Pa.

$$q_D = \frac{1}{2}\rho U_\infty^2 \tag{3.1}$$

The normalized lifting force on the airfoil was calculated in each simulation. The normalized lift was divided by the dynamic pressure and the chord length to determine the lift coefficient.

$$C_L = \frac{L_{norm}}{q_D c} \tag{3.2}$$

The lift coefficient was plotted against alpha (AOA) to determine the lift-curve slope.

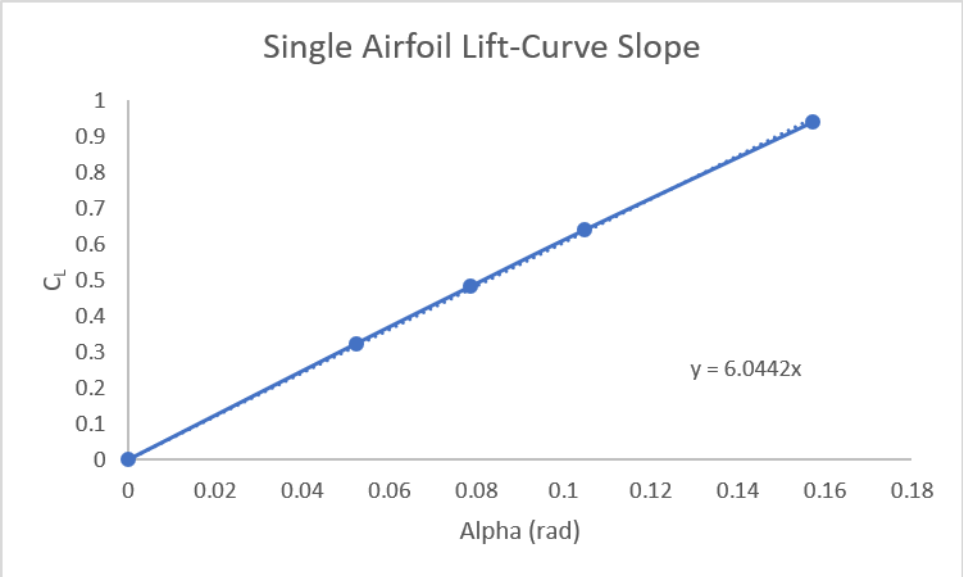

Figure 3.5: Single Airfoil Lift-Curve Slope from CFD

The lift-curve slope measured from the CFD simulations was $C_{L_\alpha} = 6.04/rad$. The canonical lift-curve slope value for a thin airfoil is 2π. The percent error between the measured value and the theoretical value was 3.80 percent. The airfoil simulated for the steady lift curve slope study was not a thin airfoil; it was a NACA 0015 airfoil. The C_L values obtained from the CFD simulations closely matched published results for NACA 0015 airfoils [10]. The lift-curve slope matching for a single airfoil was considered sufficiently accurate.

3.2 Theodorsen Solution Comparison

Unsteady oscillating airfoil CFD simulations were compared to the Theodorsen solution to validate the simulation CFD modeling. The Theodorsen solution is a closed-form solution to the problem of an oscillating thin airfoil in two-dimensional incompressible flow. It can be used to obtain lift and moment solutions analytically given sufficient initial conditions and geometric parameters, for a prescribed oscillation. The Theodorsen solution can consider both pitching and plunging motion. Here only pitching oscillations were considered to simulate the gust vanes in the UW's 3x3 tunnel.

Three CFD simulations were compared to the Theodorsen solution. The three simulations involved three different reduced frequencies. The reduced frequency was defined as:

$$k = \frac{\omega b}{U_\infty} \tag{3.3}$$

with b being the half-chord of the airfoil.

The reduced frequencies considered were $k = 0.013$, corresponding to a frequency of 1 Hz, $k = 0.16$, corresponding to a frequency of 12 Hz, and $k = 0.32$, corresponding to a frequency of 24 Hz. All simulations were run with a freestream velocity of 60 m/s and an airfoil chord length of 10 in. All simulations involved oscillations of 5 degree amplitudes.

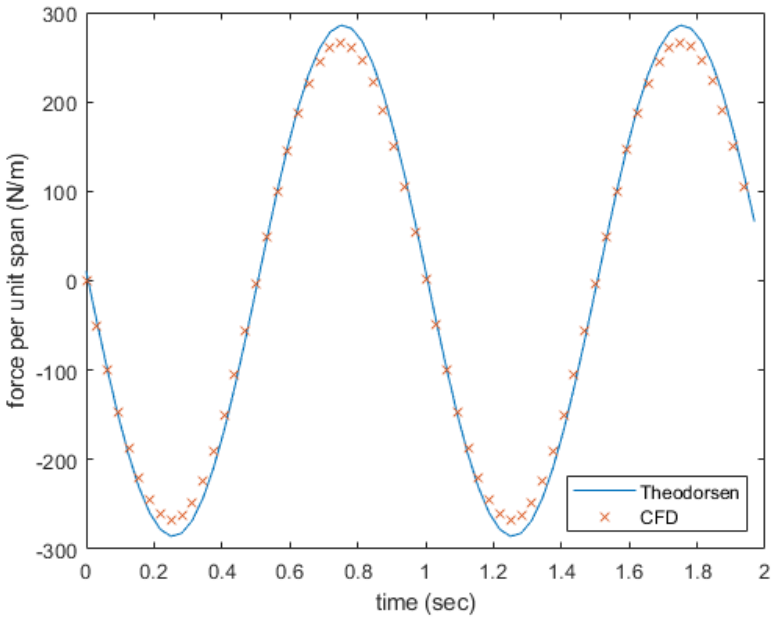

Figure 3.6: CFD / Theodorsen Matching - Low k

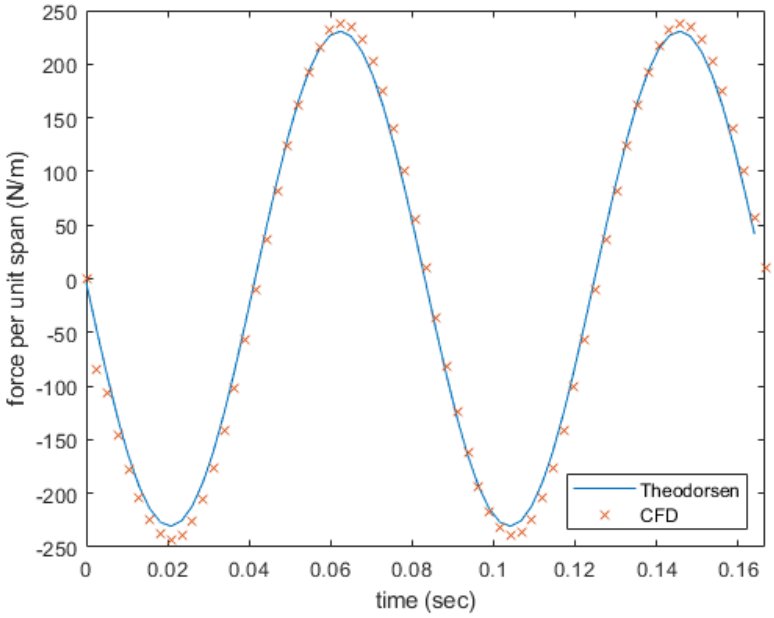

Figure 3.7: CFD / Theodorsen Matching - Medium k

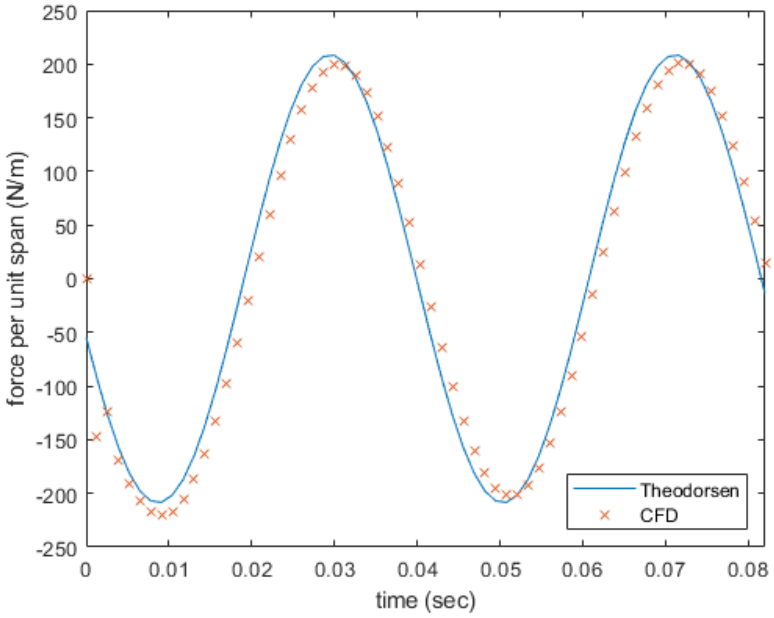

Figure 3.8: CFD / Theodorsen Matching - High k

The low reduced-frequency (1 Hz) CFD simulation effectively matched the phase of the Theodorsen solution. The maximum amplitude of the lift force determined by the CFD simulation was slightly less than that predicted by the Theodorsen solution, with a difference in the two calculations of about 8 percent at the peak. The medium reduced-frequency (12 Hz) CFD simulation demonstrated a minimal phase lag with respect to the Theodorsen solution. The maximum lift force demonstrated by the CFD simulation was also slightly higher than the Theodorsen prediction, with a difference in the two calculations of about 2 percent at the peak. Finally, the high reduced-frequency (24 Hz) CFD simulation also exhibited a phase lag, more pronounced than the phase lag exhibited by the 12 Hz simulation but still not substantial. The high reduced-frequency CFD simulation presented a lift force slightly less than that described by the Theodorsen solution, with the two calculations differing by about 5 percent at the peak.

Now, small differences between the CFD and Theodorsen solutions for the case of an oscillating airfoil in incompressible flow are expected. The Theodorsen solution assumes a thin airfoil (no thickness). The CFD simulations, instead, used a NACA 0015 airfoil, which featured a non-negligible thickness. Overall, based on comparisons to the Theodrsen solution, the CFD simulations were determined to be reliable.

3.3 Comparison of Positive-Negative AOA Results

Steady-state simulations of airfoils at positive and negative AOAs were compared to ensure the CFD simulations produced symmetric results. The basic computational model involving the two gust vanes with no test object was used. The airfoils were set at 3 degrees and -3 degrees, and 6 degrees and -6 degrees for comparisons.

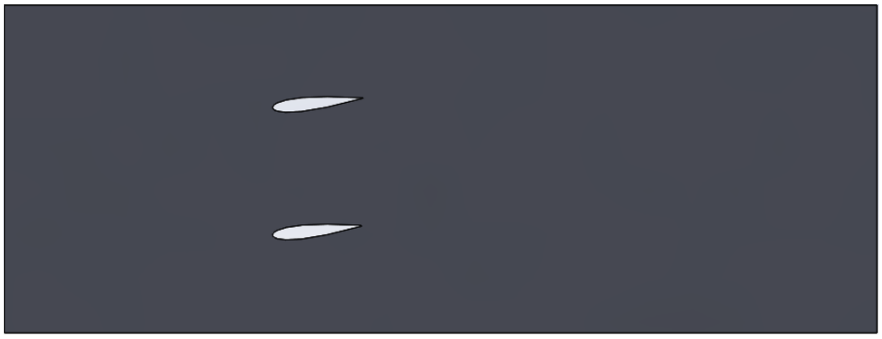

Figure 3.9: Basic Computational Model at -6 Degrees AOA Configuration

Figure 3.10: Positive and Negative AOA Comparison Results

AOA (deg)	Airfoil	Percent Error
3	A	0.62
6	A	0.21
3	B	0.31
6	B	0.70

Airfoil A was the upper airfoil and airfoil B was the lower airfoil. As shown in the above chart, the percent error in the lifting force determined by the positive and negative AOA CFD simulations was less than 1 percent for all airfoils at all angles of perturbation. It was determined the CFD simulations effectively produced symmetric results in symmetric cases.

Chapter 4

STUDIES OF GUST VANES AND WINGS IN THE WIND TUNNEL USING CFD-BASED SIMULATION

With the CFD capability described above, the flow field characteristics created by the gust generation system in the UW 3x3 tunnel were studied for a rich variety of cases. Some studies were motivated by the need to validate marching vortex lattice predictions (to be discussed in the following chapter). A wall-proximity effect study was carried out to determine the effect of the walls on the forces on the airfoils in the tunnel. An airfoil thickness-effect study was used to evaluate the effect of airfoil thickness on unsteady lift forces. Other studies were used to determine the behavior of the flow field in the wind tunnel given certain geometries and conditions. Steady-state and unsteady simulations of the gust generation system were completed in an attempt to map the flow field created by the gust generator. Also, a test wing-tail configuration was inserted into the tunnel and the gust generator's effects on the test object were analyzed for a variety of cases. All information gathered from computational CFD simulations was used to further illuminate the behavior of the flow field in the UW wind tunnel and to assist in evaluating the effectiveness of a VLM in predicting the flow field behavior.

4.1 Wall Effect Study

The impact of the proximity of the wind tunnel walls on the aerodynamic properties of an airfoil were investigated. A single airfoil along the centerline of the wind tunnel was considered. The distance between the walls and the airfoil was varied for various steady-state simulations. The airfoil was analyzed at two AOAs: 5 degrees and 10 degrees. The lifting force per unit depth (it is a 2D case) on the airfoil for each simulation was calculated.

The airfoil was a NACA 0015 airfoil. The chord length of the airfoil was 10 in. The distance between the walls and the airfoil was measured in units of chord length of the airfoil in an attempt to nondimensionalize the results.

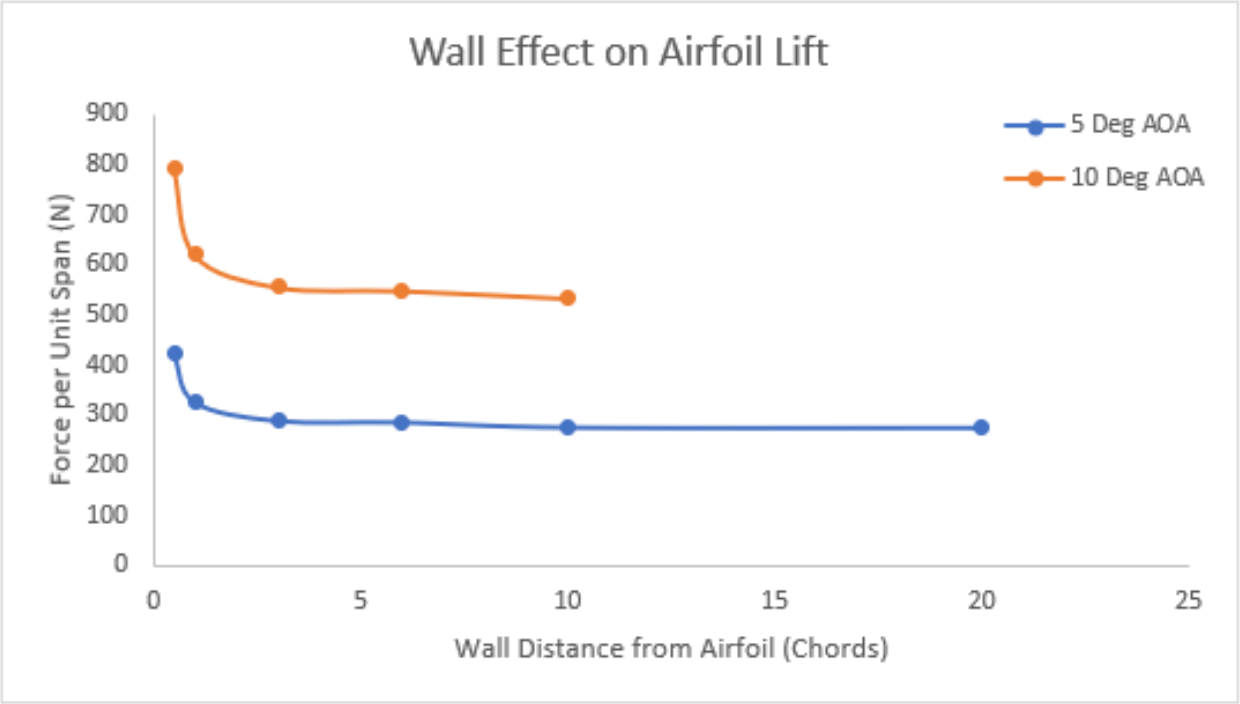

Figure 4.1: Wall Effect Study Results

The data shows that at close distances the walls can increase the lift on the airfoil substantially. For an airfoil placed within less than about five chord lengths of the wind tunnel walls, a distinct increase in lift should be expected. As the distance between the walls and the airfoil was increased past about five chord lengths, however, the effect of the walls became largely insignificant. The increased lift due to the walls was inversely related to the distance between the walls and the airfoil of interest. For the given gust generator, the gust vanes were located at about one chord length away from the wall. Although the gust generator configuration was slightly different than the configurations investigated in the given study (one airfoil vs. two airfoils), the results indicated the effects of the walls on

the lifting force experienced by the gust vanes cannot be ignored. The CFD (and the VLM simulations that would follow) must account for the effect of the walls on the gust generator and the flow field created by the gust generator.

4.2 Airfoil Thickness Study

The effect of the thickness of the airfoil on an unsteady simulation in a wind tunnel environment was investigated. As shown previously, walls in close proximity to an airfoil can cause an increase in the lift experienced by the airfoil. To investigate the effect of airfoil thickness, three airfoils were considered: NACA 0006, NACA 0009, and NACA 0012. The chord length of the airfoils was 10 in. The airfoils were placed along the centerline of the tunnel. That is, the distance from each wall (left and right) was 18" 0r 1.8 chords. The frequency of oscillation was 1 Hz (at a speed of 60 m/sec and a corresponding reduced frequency of 0.013). The amplitude of oscillation was 5 degrees.

Figure 4.2: Airfoil Thickness Study - Computational Setup for NACA 0009 Airfoil

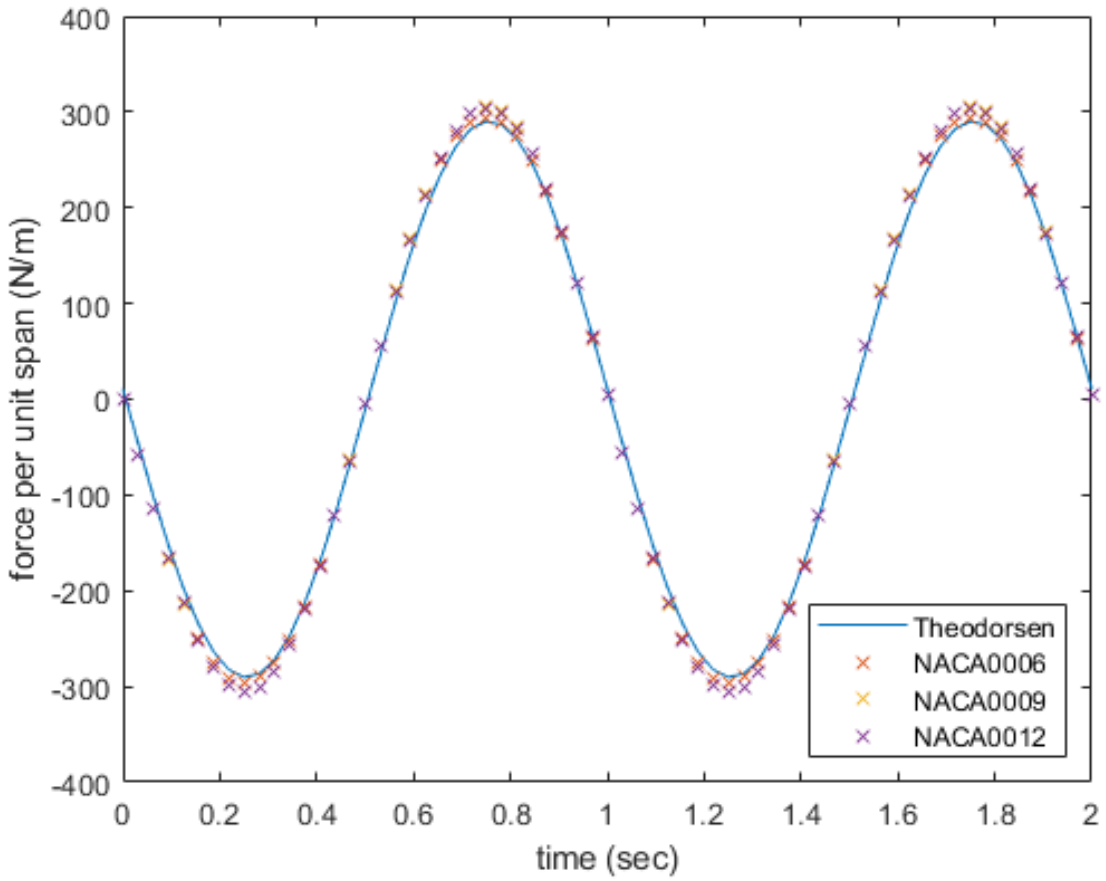

Figure 4.3: Airfoil Thickness Study - Theodorsen Comparison

Despite the fact that the Theodorsen solution for oscillating airfoils is for a no-walls case, computational CFD results here for the UW 3x3 tunnel case show quite close agreement with the Theodorsen solution for all airfoils. The CFD results match the phase of the Theodorsen solution and slightly over predict the amplitude. At first thought the simulations were expected to over predict the Theodorsen solution because of the wall effect. Additionally, the Theodorsen solution describes a thin airfoil while the airfoils considered in the CFD simulations were of non-negligible thickness. On further consideration it becomes clear that the thickness, as long as no separation is involved, does not affect lift forces of the airfoil in a major way because of its symmetric effect on the flow field. It does affect the flow field, of

course. The maximum lift force experienced by each airfoil increased as the thickness of the airfoil increased, but for the airfoils studied the effect is small.

4.3 CFD Steady Flow Field Studies

Multiple steady-flow simulations were used to evaluate the characteristics of the flow in the tunnel. The gust generator vanes were perturbed to various degrees and the resulting flow behavior was examined. Specifically, the gust vanes were rotated 0 degrees, 3 degrees, 6 degrees, and 9 degrees. For each perturbation, five cross-sectional slices of the tunnel and three longitudinal slices of the tunnel were considered. Along each cross-sectional slice, the velocity profile in the wake and the gust AOA were considered. Gust AOA was calculated as:

$$gustAOA = \frac{v}{u} \tag{4.1}$$

wehere u and v are the longitudinal and lateral velocitiy components of the flow, respectively. Cross-sectional cuts were taken 0 chord lengths, 1 chord length, 2 chord lengths, 3 chord lengths, and 4 chord lengths behind the trailing edge of the airfoil. For each longitudinal slice, the velocity profile in the wake was considered. Longitudinal cuts were taken along the centerline of the tunnel and through the 1/4-chord of each airfoil, parallel to the centerline.

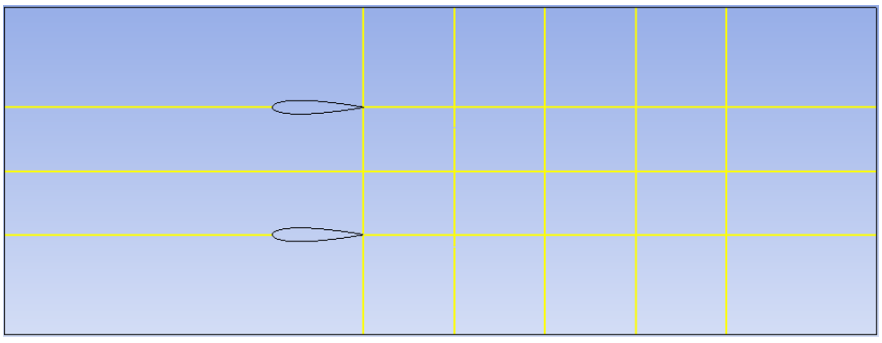

Figure 4.4: Gust Field Study Model Configuration with Cuts Illuminated

4.4 Gust AOA Distribution Plots

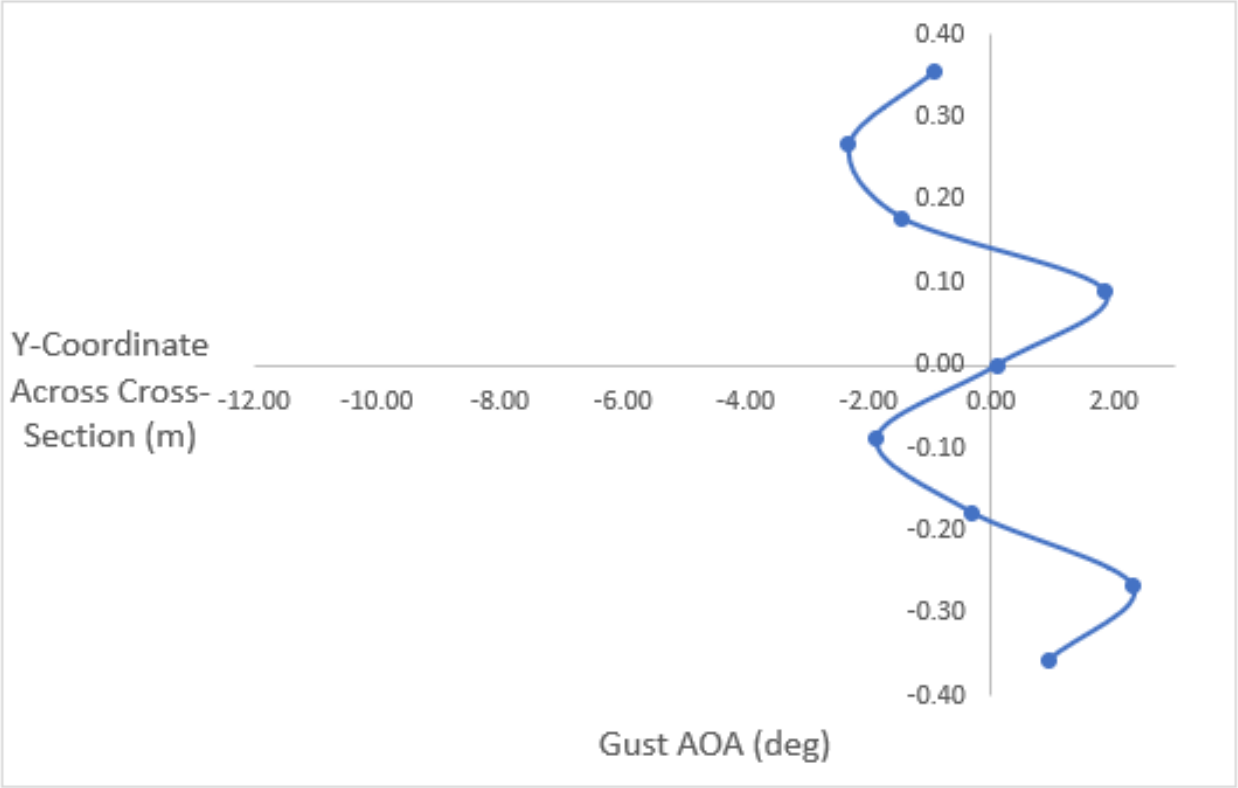

Figure 4.5: Gust AOA of Gust System at Zero Degrees AOA at Trailing Edge

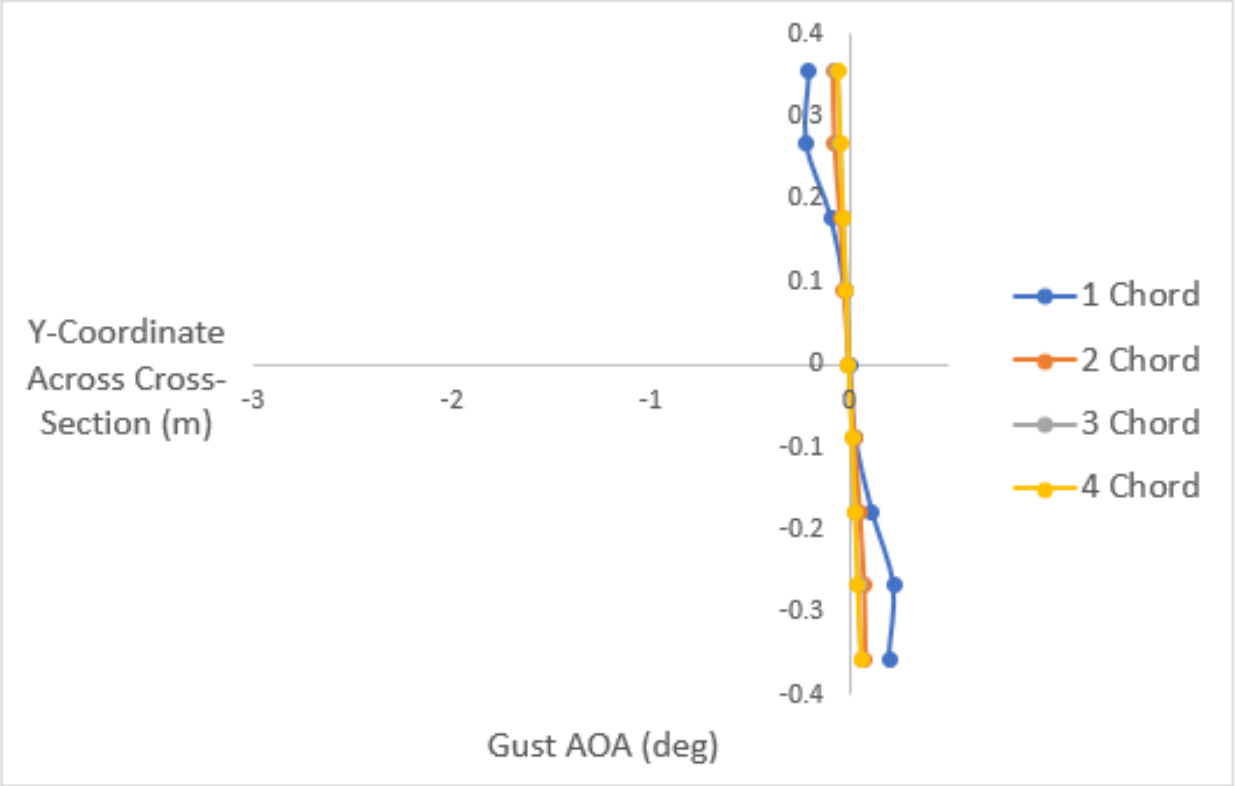

Figure 4.6: Gust AOA of Gust System at Zero Degrees AOA in Wake

For the zero degrees AOA case, at the trailing edge of the vanes, a significant non-zero gust AOA exists, especially in the near vicinity of the airfoils. Near the upper edge regions of both airfoils, a negative gust AOA exists while near the lower edges of both airfoils, a positive gust AOA exists. The pattern forms a periodic-looking distribution across the cross-section. Into the wake, the gust AOA for the zero AOA case dissipates quickly as the distance from the airfoil trailing edge into the wake is increased. At one chord length into the wake, a small gust AOA remains with negative gust AOA above the top airfoil and positive gust AOA below the bottom airfoil. A similar trend continues into the wake. However, at two chord lengths behind the trailing edge and continuing into the rest of the wake, there is little-to-no gust AOA.

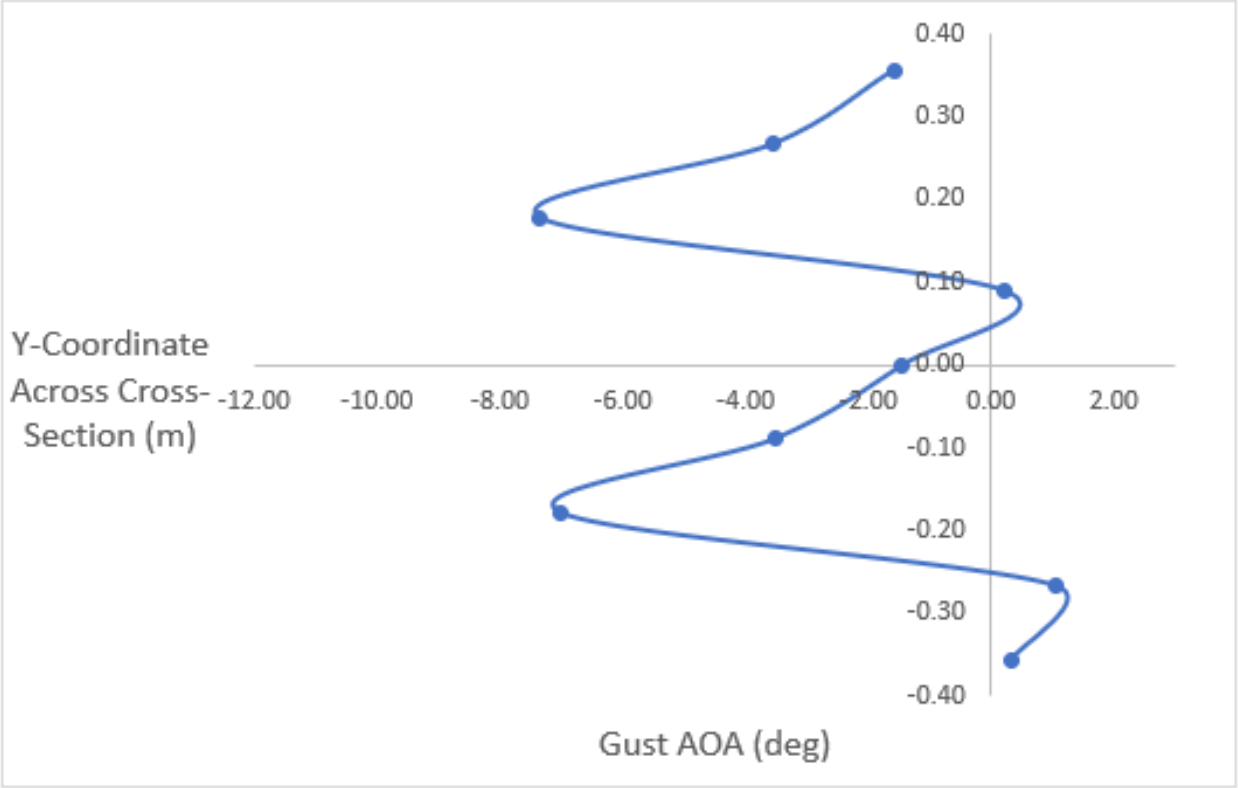

Figure 4.7: Gust AOA of Gust System at Three Degrees AOA at Trailing Edge

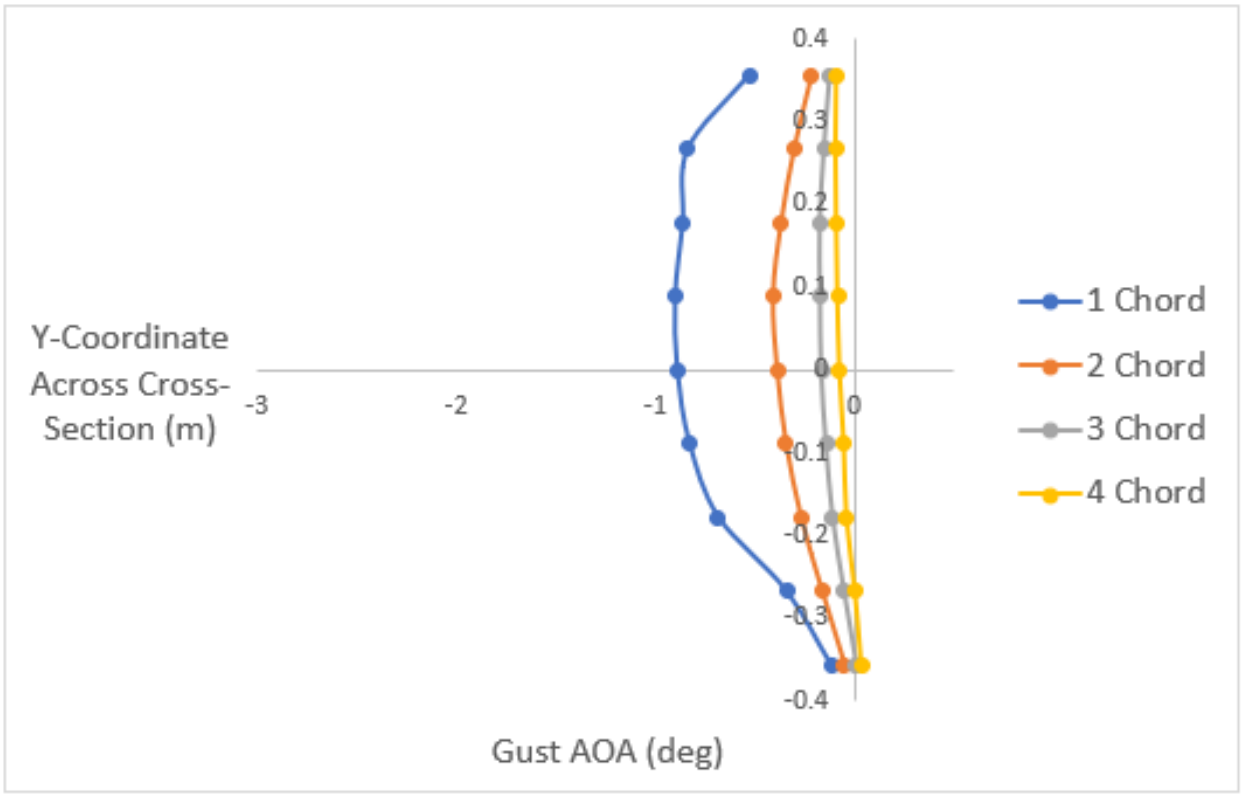

Figure 4.8: Gust AOA of Gust System at Three Degrees AOA in Wake

For the three degrees AOA case, the previous periodic-looking distribution from the zero AOA case seems to translate in the negative direction and transform slightly. The gust AOA at the trailing edges of the both vanes is distinctly negative and about three times greater than the largest magnitude gust AOA from the zero AOA case. The upper edge regions of the airfoils again demonstrate negative gust AOA sections while the lower edge regions of the airfoils again demonstrate positive gust AOA sections, similar to the zero AOA case. The wake behavior of the three degrees AOA case differs from that of the zero degrees AOA case. The cross-sectional cut located one chord length behind the trailing edge displays a distinct negative gust AOA across the entire cross-section with the most intense gust AOA in near the tunnel centerline. The gust AOA intensity decreases moving away from the centerline towards the walls. A similar trend is seen with the succeeding cuts at two, three and four

chord lengths behind the trailing edge. The magnitude of the gust AOA decreases, however, as the distance between the airfoil and the cross-sectional cut is increased.

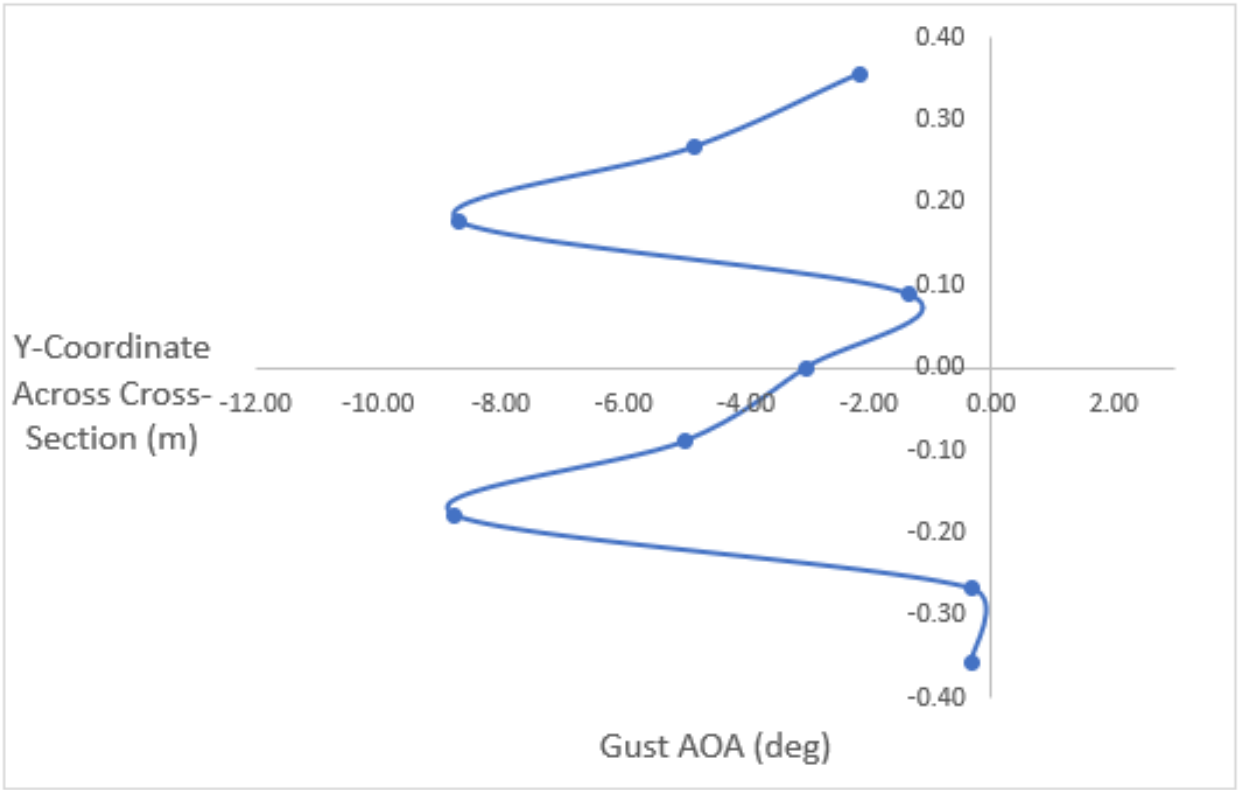

Figure 4.9: Gust AOA of Gust System at Six Degrees AOA at Trailing Edge

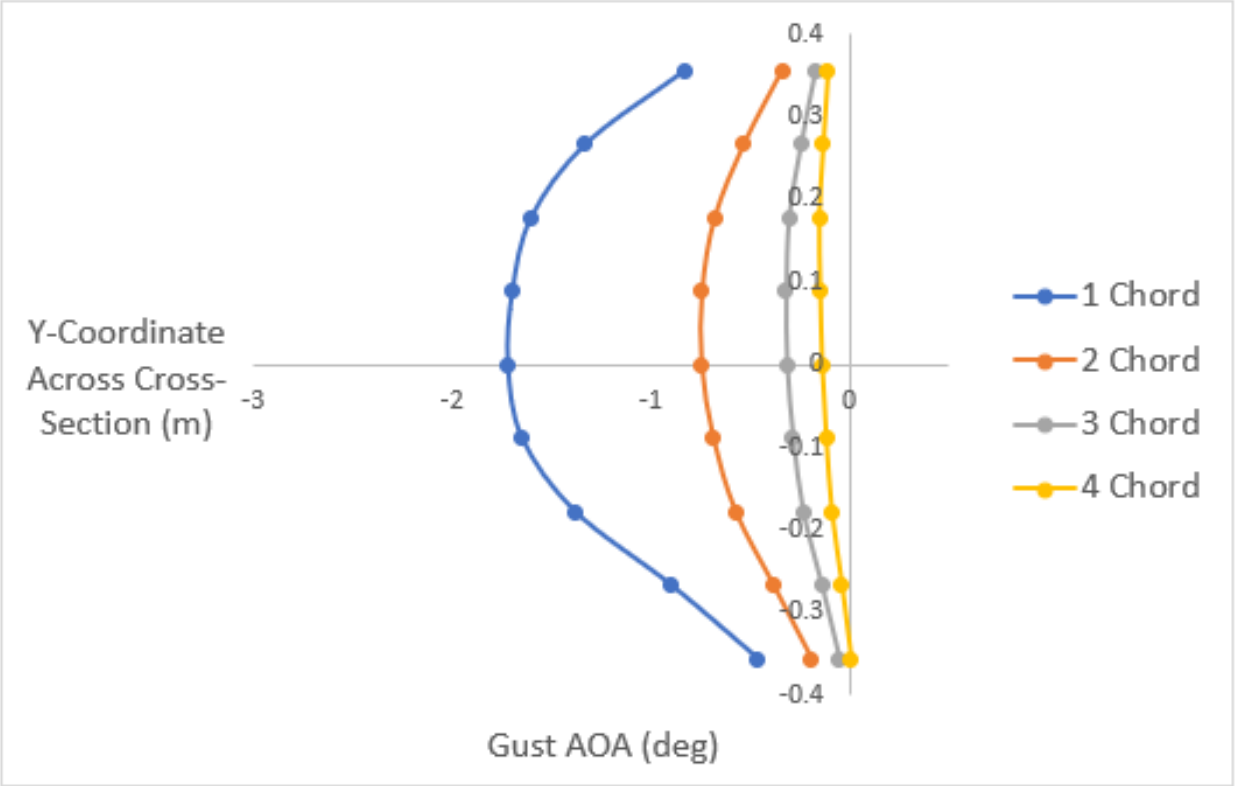

Figure 4.10: Gust AOA of Gust System at Six Degrees AOA in Wake

The distribution in the six degrees AOA case closely resembles that of the three AOA case with a slight translation to the left. At the airfoil's trailing edge, the gust AOA is entirely negative across the entire cross-sectional cut. The greatest magnitude gust AOA is found, as in the three AOA case, at the vanes' trailing edges. The gust AOA decreases towards the walls and toward centerline. The gust AOA in the wake follows the same pattern as the three degrees AOA case. The gust AOA in the wake is entirely negative. The more intense gust AOA is found at the center of the tunnel and closer to the vanes. As the distance from centerline or the distance into the wake is increased, the magnitude of the gust AOA decreases.

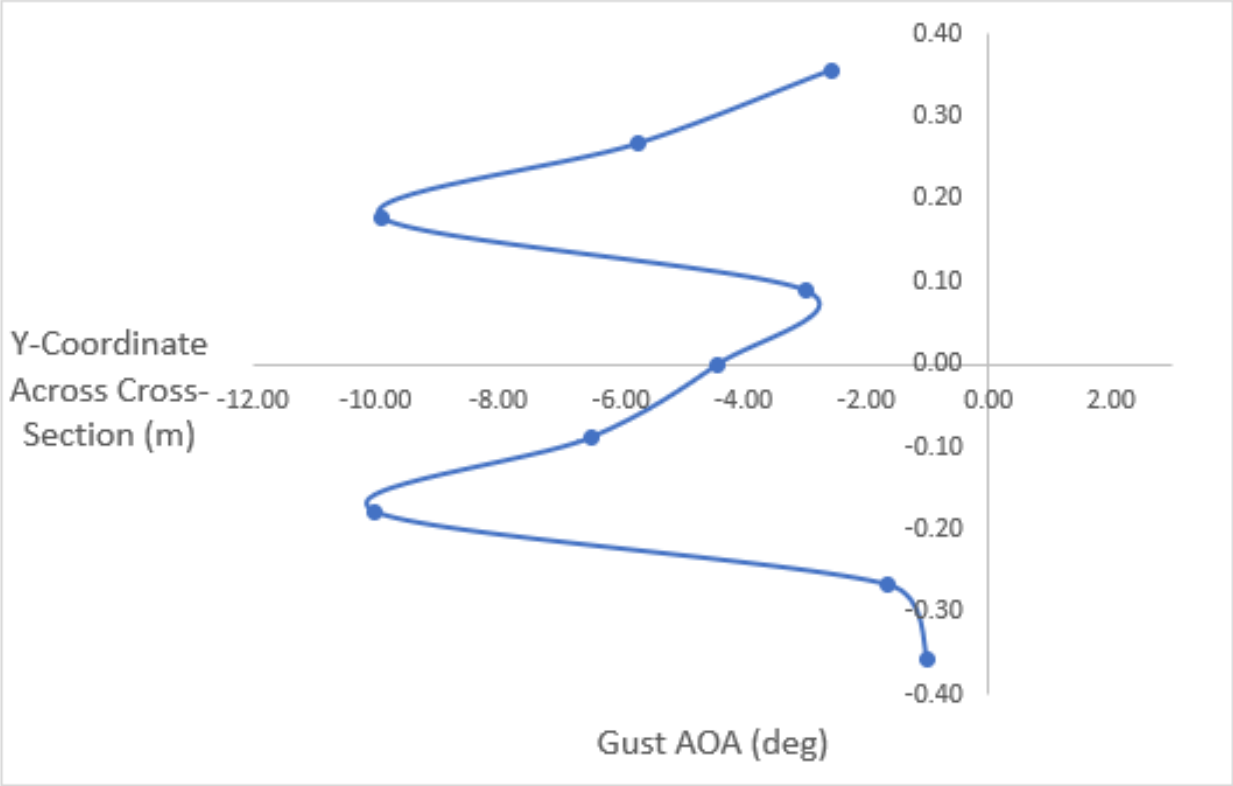

Figure 4.11: Gust AOA of Gust System at Nine Degrees AOA at Trailing Edge

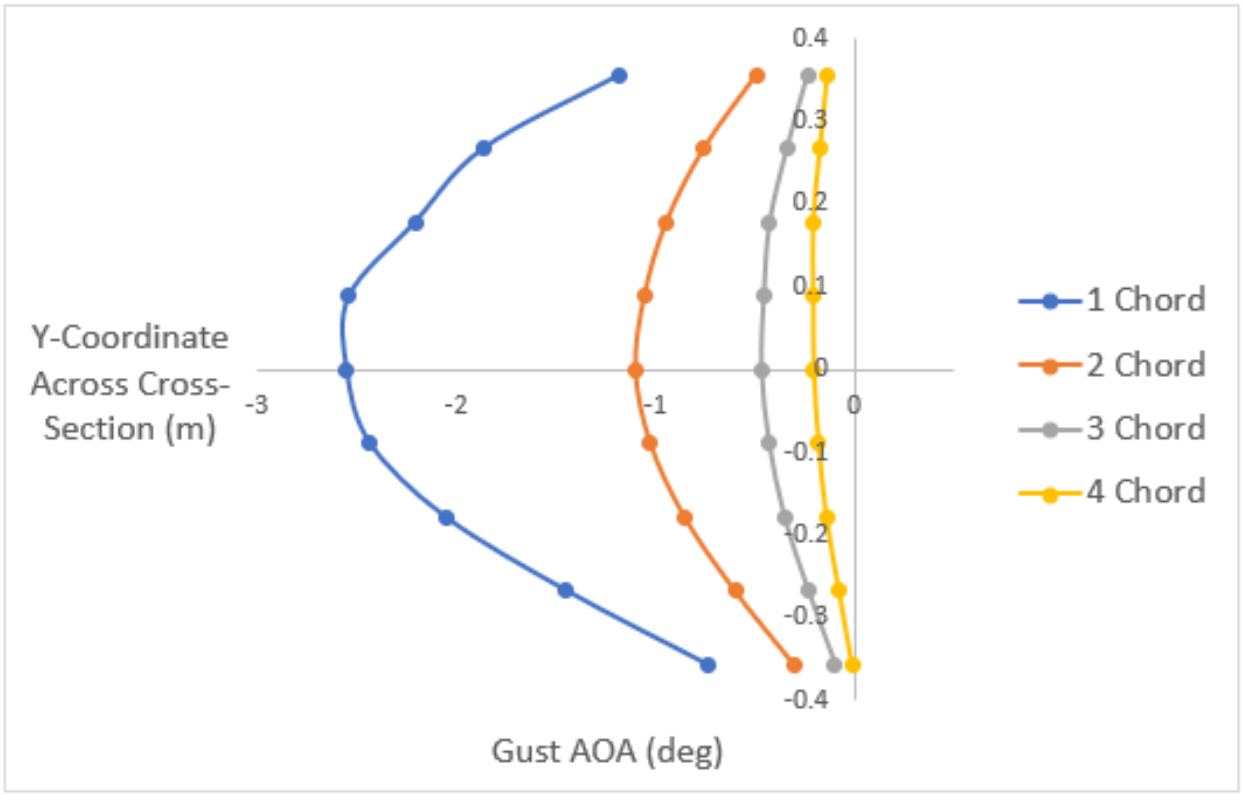

Figure 4.12: Gust AOA of Gust System at Nine Degrees AOA in Wake

The nine degrees AOA case follows the same trends as the six degrees AOA case. The magnitude of the maximum gust AOA at the trailing edge and in the wake is increased. However, again, the maximum magnitude gust AOA is found at the trailing edge of the airfoils. The magnitude of the gust AOA, which is negative at all points sampled, decreases towards the walls and into the wake.

The gust AOA plots indicate distinct gusting action near to the airfoils. The magnitude of the gust AOA increases at the trailing edges of the airfoils as airfoil AOA is increased. However, for the steady-state case, the gust AOA dissipates significantly into the wake. A higher vane AOA was successful in propagating gust AOA further into the wake than a lower vane AOA. However, after about two chord length behind the airfoils' trailing edges, the gust AOA becomes small.

4.5 Velocity Profiles at Cross-Sectional Cuts in Wake

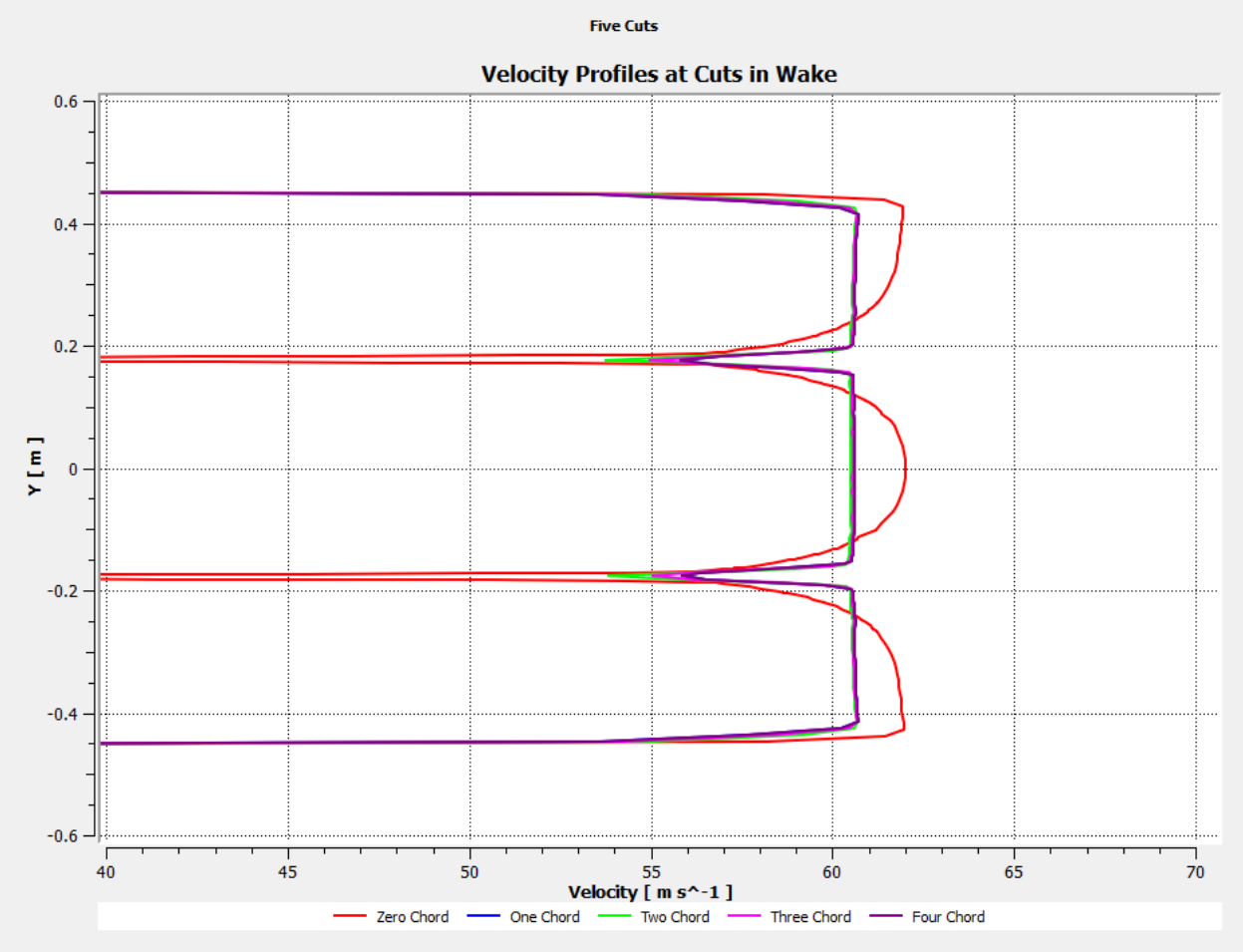

Figure 4.13: Velocity Profile at Cross-Sectional Cuts in Wake for Zero Degrees AOA

The velocity profile at cross-sectional cuts charts show the velocity at various cuts over the cross-section of the tunnel where Y represents the distance above or below the centerline of the tunnel in meters. The velocity profile at cross-sectional cuts in the wake for the zero degrees vanes AOA case shows the flow field is relatively unaffected by the gust vanes except for immediately behind the vanes. The horizontal velocity, u, was considered for all cases. At the trailing edge of the airfoils, the velocity is increased above and below the airfoils as the

flow accelerated around them. At the trailing edge exactly, the velocity is zero in accordance with the Kutta condition. However, the flow field effectively returns to the undisturbed state by one chord length behind the trialing edge of the airfoils. The only area in the flow field where the flow does not return to the previous flow condition is behind the airfoils in their direct wake. The figure shows that the velocity of the flow in the direct wake increases with each successive step away from the trailing edge of the airfoils in the direction of the flow.

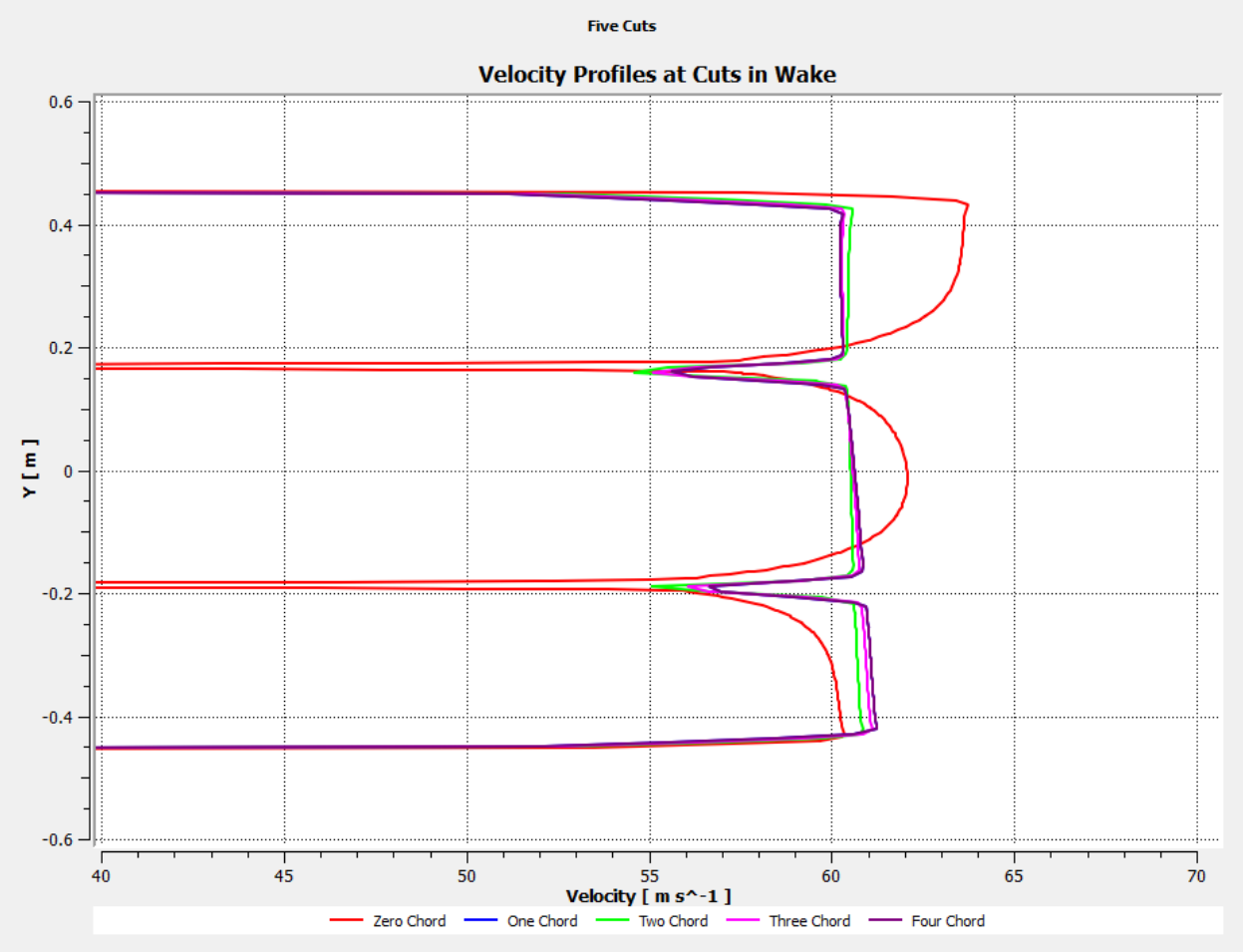

Figure 4.14: Velocity Profile at Cross-Sectional Cuts in Wake for Three Degrees AOA

The velocity profile at cross-sectional cuts in the wake for the three degrees AOA case shows a distinctly different flow field than the one produced by the zero degrees AOA con-

figuration. At the trailing edge cut, the flow velocity is highest above the top airfoil and lowest below the bottom airfoil. At progressive cuts into the wake, the velocity above the top airfoil decreases while the velocity below the bottom airfoil increases. Such a trend leads to a slightly asymmetric flow field at four chord lengths beyond the trailing edge where the bottom section of flow is faster than the middle and top sections of flow. Although slight, the effect of the gust vanes is perceptible into the wake, unlike in the zero degrees AOA case. Again, the velocity at the trailing edge of the airfoils is zero and slowly accelerates towards the freestream velocity further into the wake.

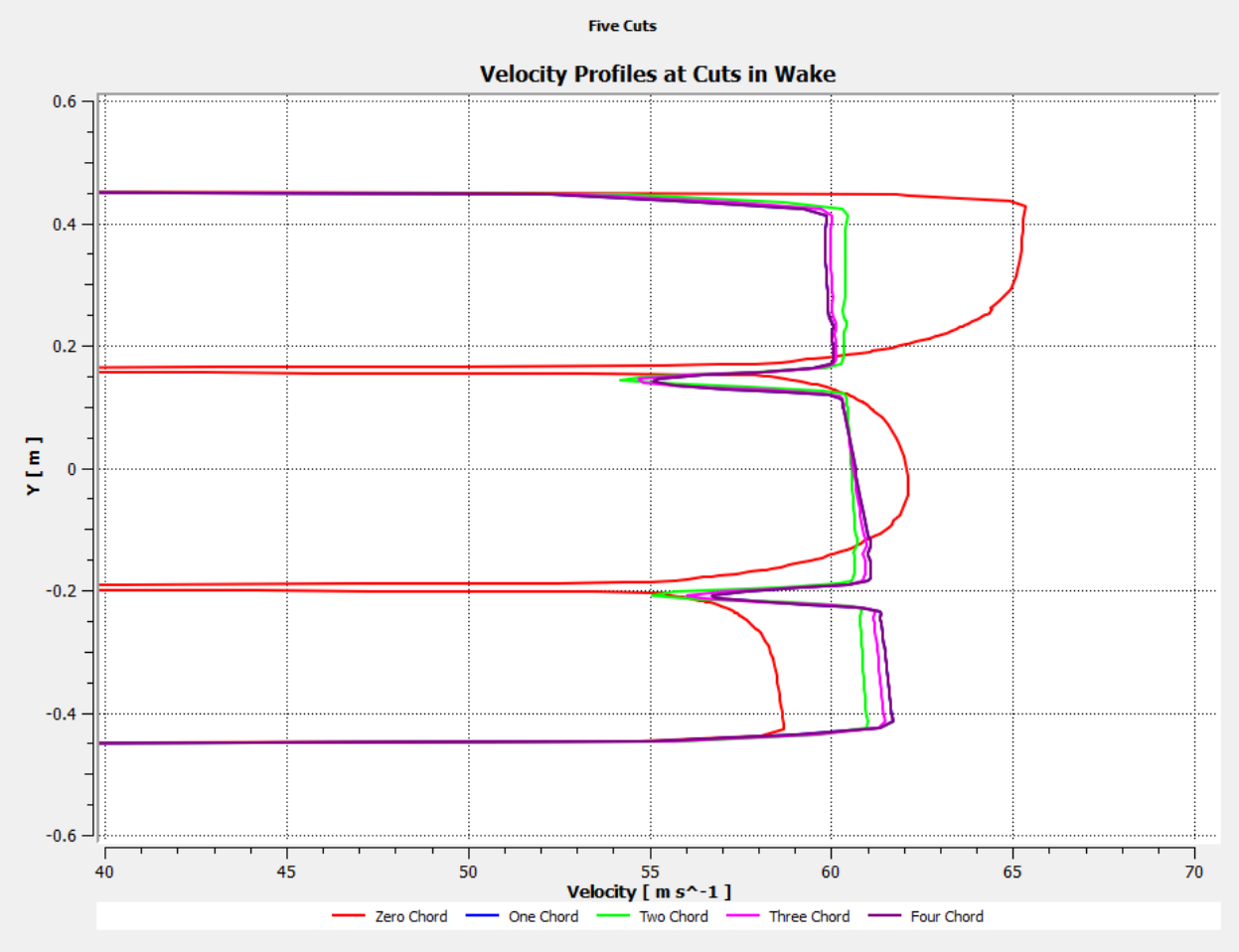

Figure 4.15: Velocity Profile at Cross-Sectional Cuts in Wake for Six Degrees AOA

The six degrees AOA case follows the same trends as the three AOA case with greater differences in magnitudes between velocities to the left and right of the airfoils. At the trailing edge, the highest velocity is along the top wall section and the lowest velocity is along the bottom wall section("top wall" and "bottom wall" are used here to denote the section between the top airfoil and the wall in the positive Y-direction and the section between the bottom airfoil and the wall in the negative Y-direction, respectively). The velocity along the top wall section decreases into the wake and the velocity along the bottom wall section increases into the wake. Such a pattern results again in a slightly asymmetric flow further into the wake of the same type described in the three degrees AOA case. The velocity trends exhibited by the wake directly behind the airfoils continue to develop. However, it appears the location of decreased velocity in the direct wake shifts slightly in the negative-y direction with as one ventures further into the wake.

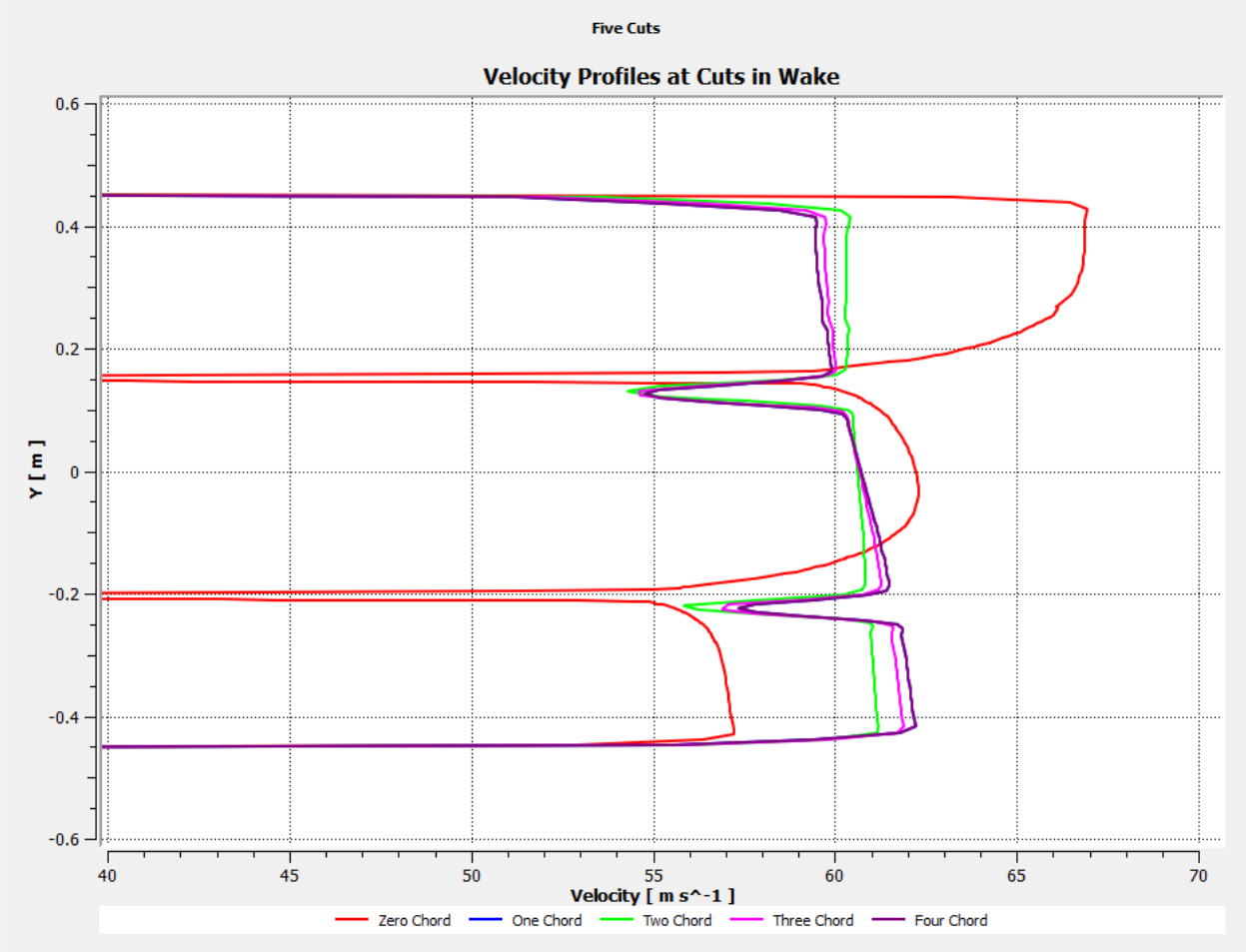

Figure 4.16: Velocity Profile at Cross-Sectional Cuts in Wake for Nine Degrees AOA

The nine degrees AOA case continues the trends previously shown by the six degrees AOA and three degrees AOA cases. All previous qualitative descriptions for the six degrees AOA case apply to the nine degrees AOA case. The nine degrees AOA case differs from the six degrees AOA case in that the nine degrees AOA case demonstrates greater differences in magnitude for the previously considered behavior.

4.6 Velocity in Wake Along Longitudinal Cuts of Wind Tunnel (in the incoming-flow direction)

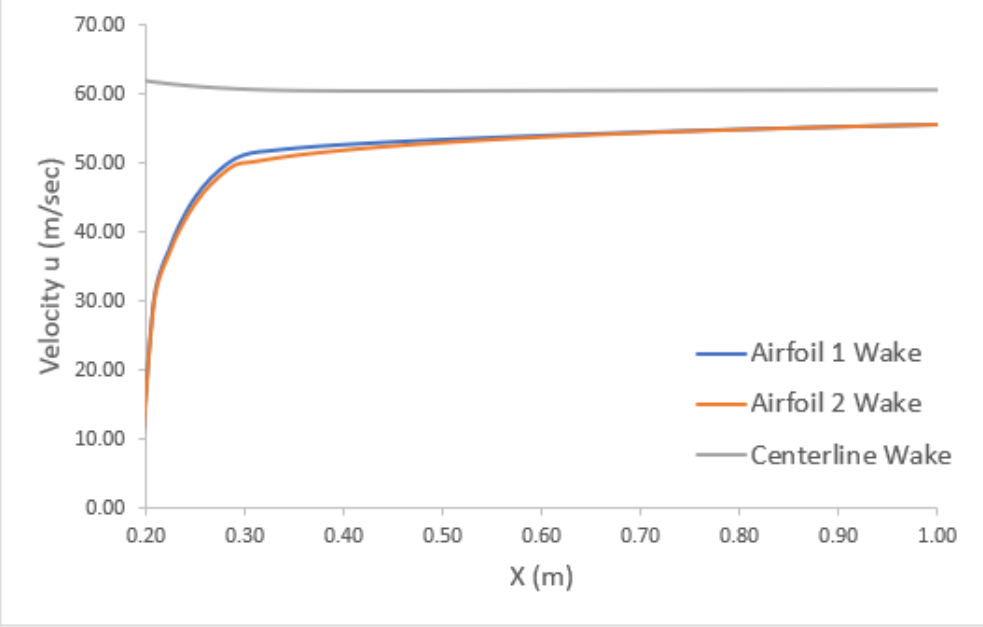

Figure 4.17: Steady-State Longitudinal Velocity Profiles: Zero Degrees AOA Case

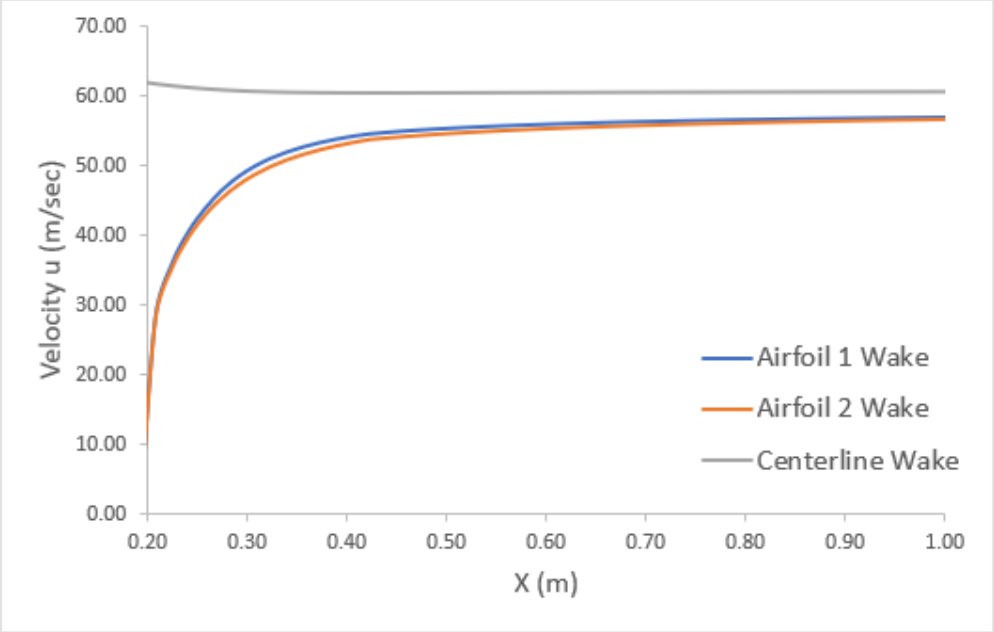

Figure 4.18: Steady-State Longitudinal Velocity Profiles: Three Degrees AOA Case

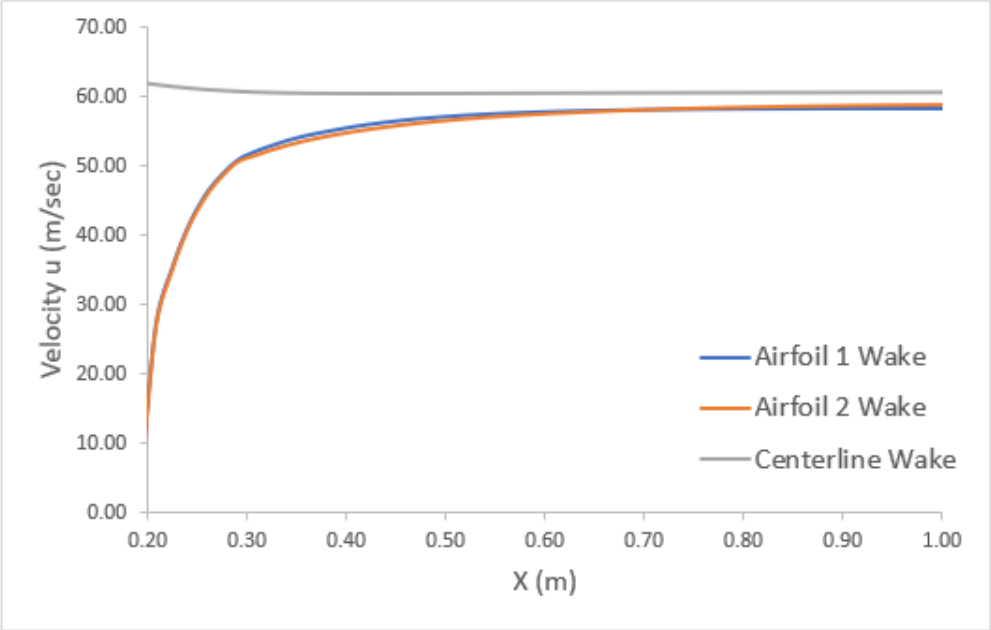

Figure 4.19: Steady-State Longitudinal Velocity Profiles: Six Degrees AOA Case

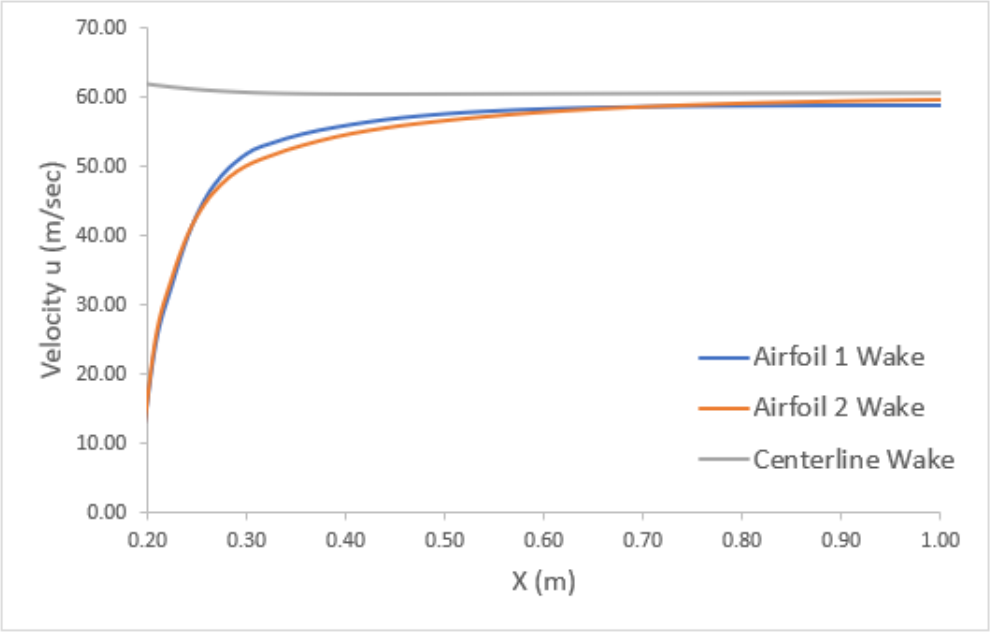

Figure 4.20: Steady-State Longitudinal Velocity Profiles: Nine Degrees AOA Case

The plots of the horizontal velocity, u, in the wake along longitudinal cuts, one along the centerline of the tunnel and two behind the trailing edges of the airfoils, look nearly identical for each of the four cases. Each case shows the enforcement of the Kutta Condition at the airfoil's trailing edge. Of note, the four plots also illustrate the changing magnitudes of the velocities in different areas of the flow field. Near the gust vanes' trailing edge, the upper airfoil's wake has a greater velocity than the lower airfoil's wake (since in the tunnel the vanes are mounted vertically, there is a left and a right vane relative to the centerline in the direction of the flow). As a greater distance into the wake is considered, the velocity of the lower airfoil's wake increases at a greater rate than the velocity of the upper airfoil's wake. Such a result is consistent with the wake velocity trends discussed previously when considering the velocity profiles at various cross-sectional cuts in the wake.

The gust angle distributions along the longitudinal cuts for each gust vane perturbation were also plotted.

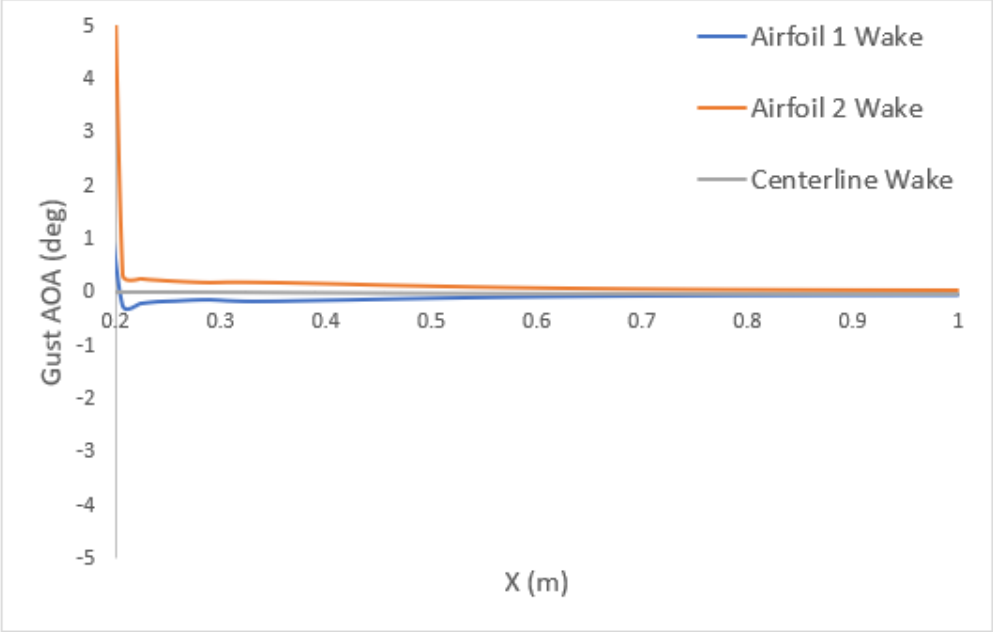

Figure 4.21: Steady-State Longitudinal Gust Angle Distribution: Zero Degrees AOA Case

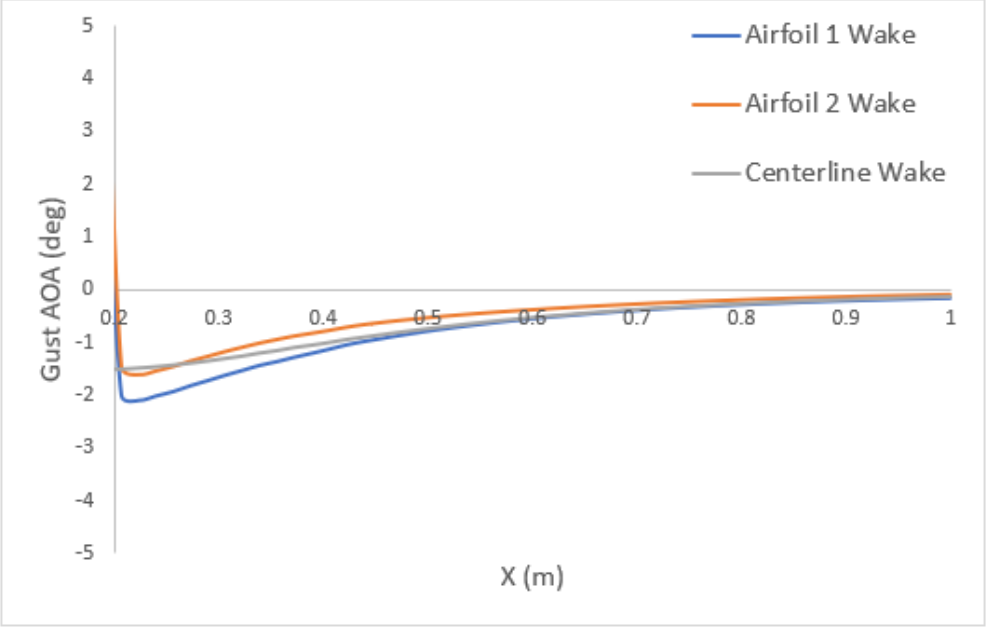

Figure 4.22: Steady-State Longitudinal Gust Angle Distribution: Three Degrees AOA Case

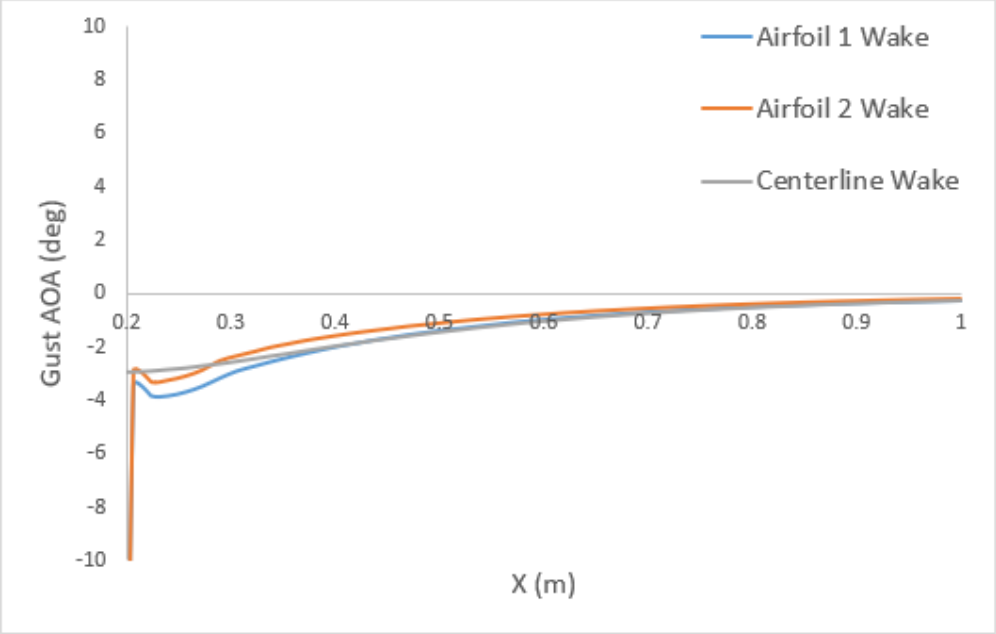

Figure 4.23: Steady-State Longitudinal Gust Angle Distribution: Six Degrees AOA Case

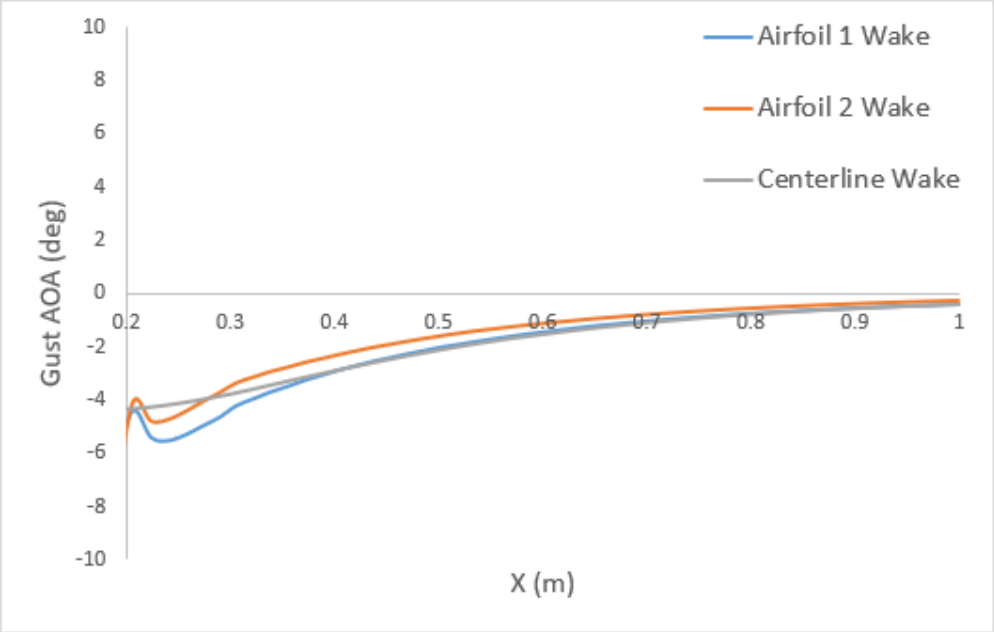

Figure 4.24: Steady-State Longitudinal Gust Angle Distribution: Nine Degrees AOA Case

The gust angle distribution plots for each case show the distinct non-negligible gust AOA in the vicinity of the oscillating vanes. As one progresses further into the wake, the gust AOA dissipates in an exponential manner. Greater gust vane perturbation leads to more significant gust AOA near the vanes and a more gradual dissipation of gust AOA in the wake like in the previous investigations, .

4.7 Unsteady Gust Field Study

The unsteady gust field studies investigated the flow field behavior in the wind tunnel under the influence of oscillatory motion by the gust generator vanes. The gust generator vanes were oscillated at three amplitudes: 3 degrees, 6 degrees and 9 degress. The gust generator was also oscillated at two frequencies: 1 Hz (corresponding to a reduced frequency of $k = 0.013$) and 24 Hz (corresponding to a reduced frequency or $k = 0.32$). The flow field, for a given amplitude and frequency of oscillation, was analyzed at four timesteps corresponding to 0.25T, 0.5T, 0.75T and T where T is the length of the period.

4.7.1 Low Reduced-Frequency (1 Hz, 60 m/sec) Simulations

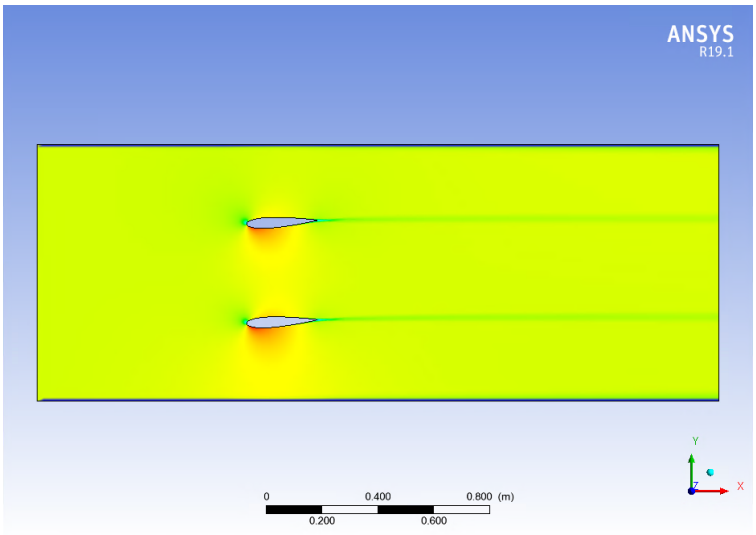

Figure 4.25: Unsteady Simulation: Three-Degree Oscillation, Low k at $\frac{T}{4}$ (Snapshot)

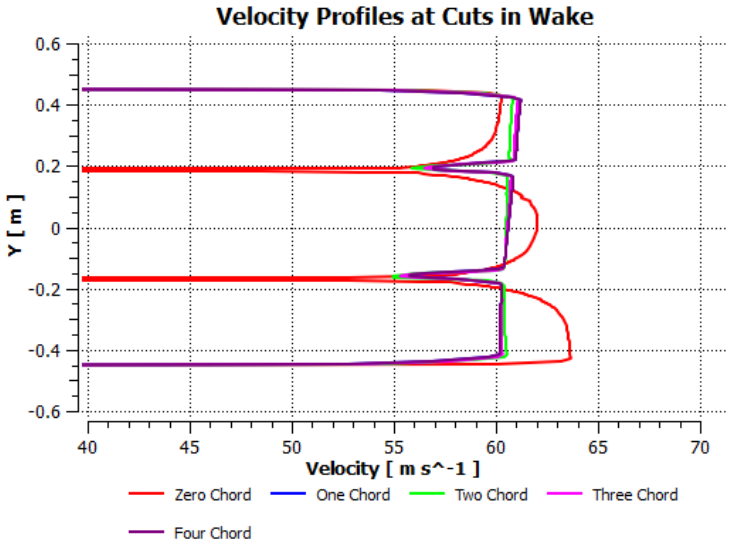

Figure 4.26: Unsteady Simulation: Three-Degree Oscillation, Low k at $\frac{T}{4}$ (Chart)

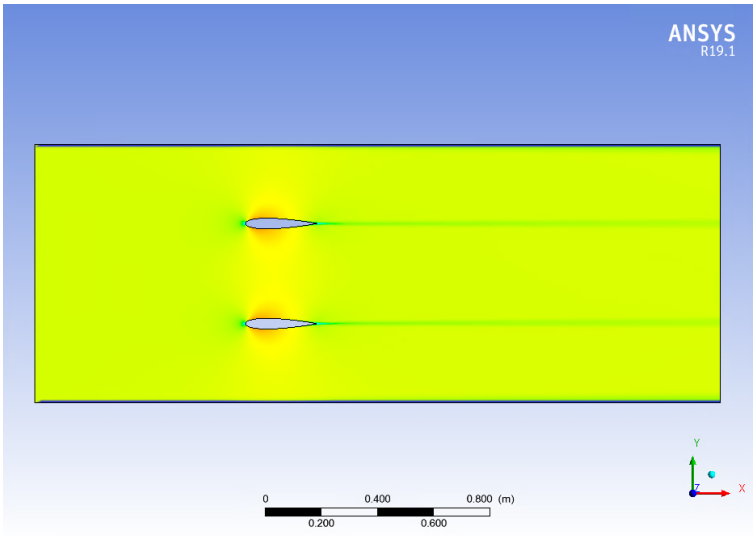

Figure 4.27: Unsteady Simulation: Three-Degree Oscillation, Low k at $\frac{T}{2}$ (Snapshot)

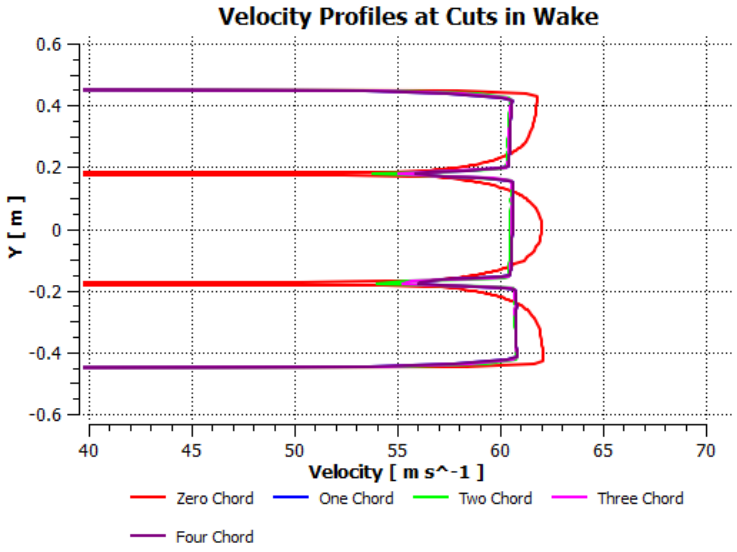

Figure 4.28: Unsteady Simulation: Three-Degree Oscillation, Low k at $\frac{T}{2}$ (Chart)

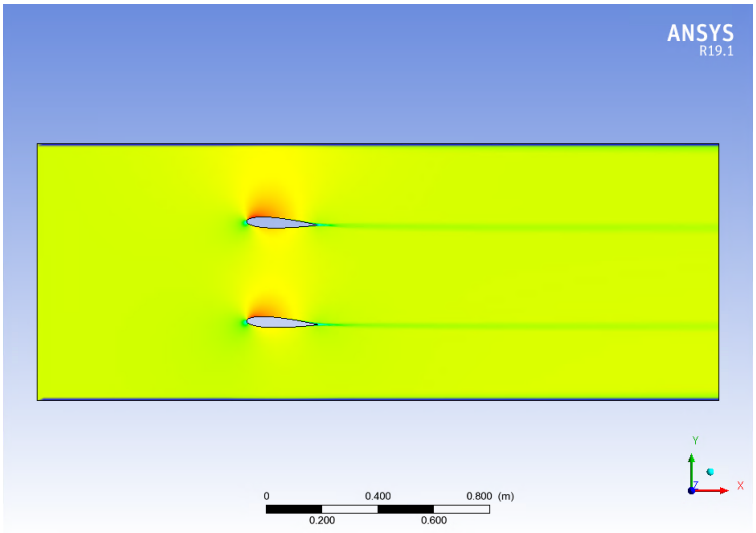

Figure 4.29: Unsteady Simulation: Three-Degree Oscillation, Low k at $\frac{3T}{4}$ (Snapshot)

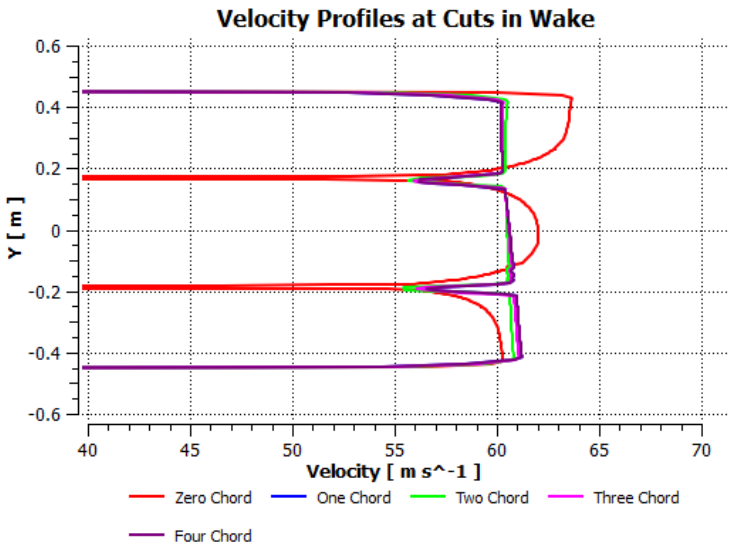

Figure 4.30: Unsteady Simulation: Three-Degree Oscillation, Low k at $\frac{3T}{4}$ (Chart)

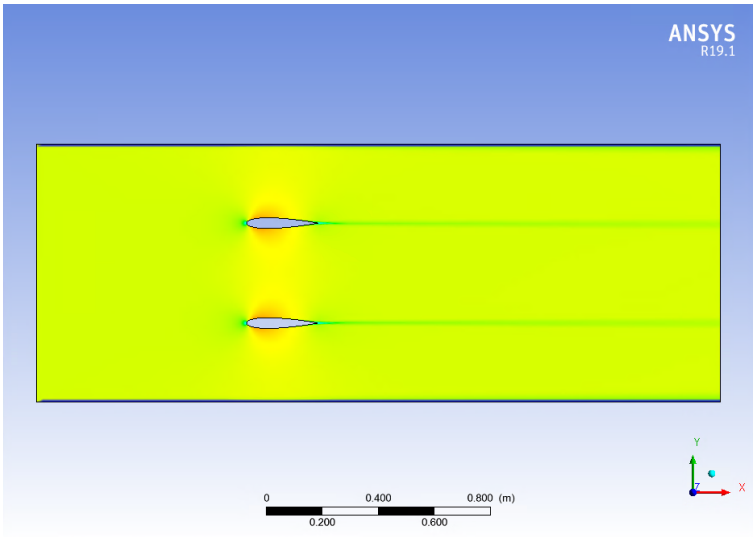

Figure 4.31: Unsteady Simulation: Three-Degree Oscillation, Low k at T (Snapshot)

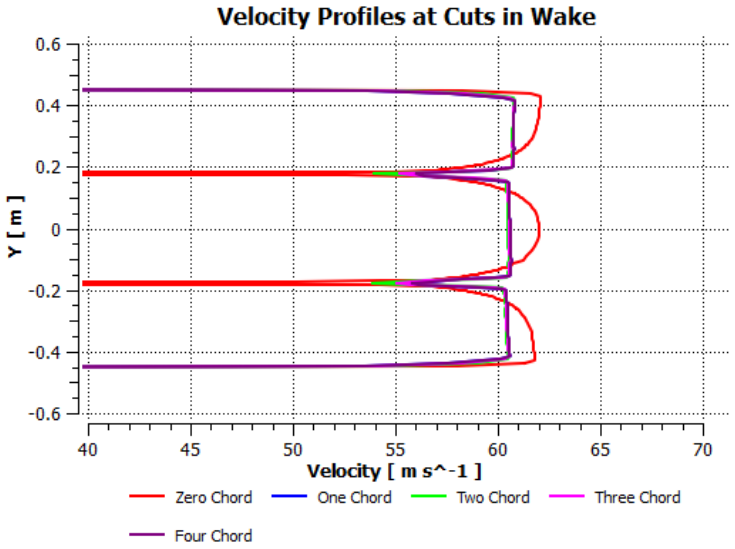

Figure 4.32: Unsteady Simulation: Three-Degree Oscillation, Low k at T (Chart)

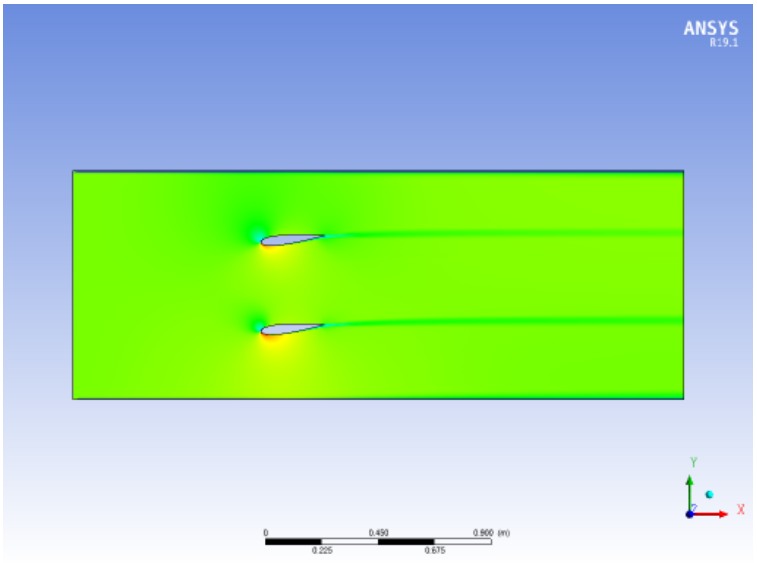

Figure 4.33: Unsteady Simulation: Six-Degree Oscillation, Low k at $\frac{T}{4}$ (Snapshot)

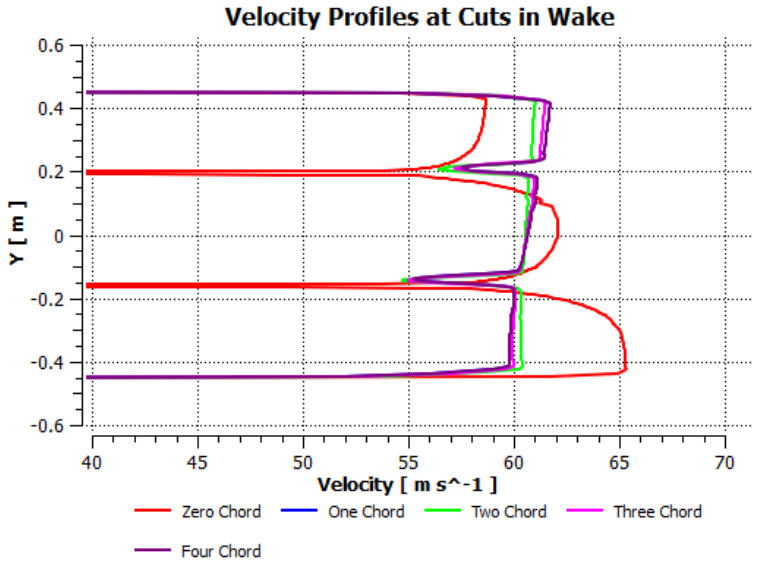

Figure 4.34: Unsteady Simulation: Six-Degree Oscillation, Low k at $\frac{T}{4}$ (Chart)

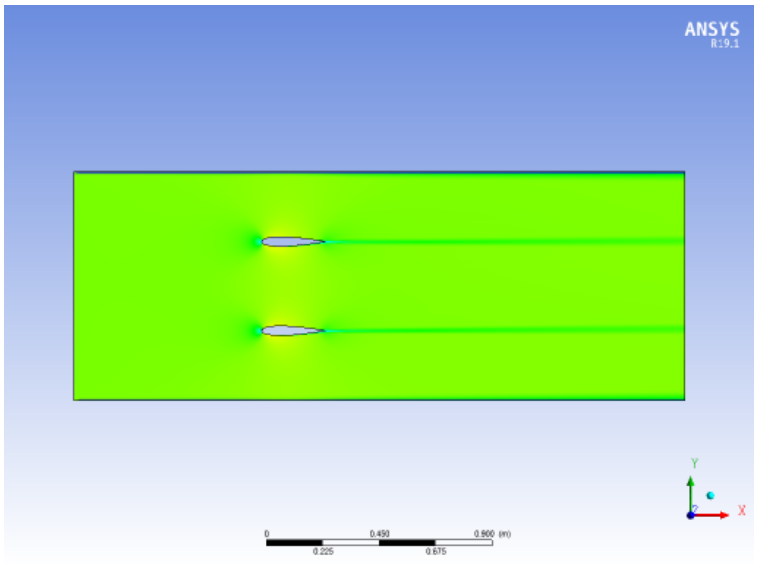

Figure 4.35: Unsteady Simulation: Six-Degree Oscillation, Low k at $\frac{T}{2}$ (Snapshot)

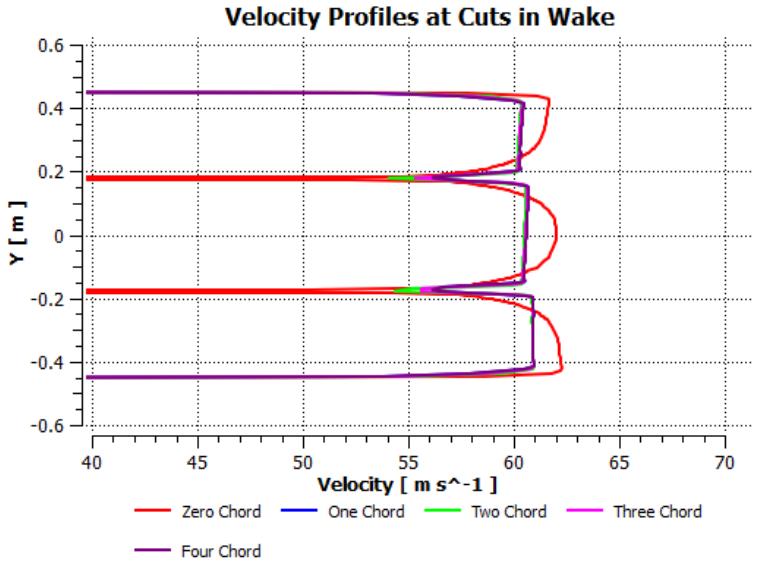

Figure 4.36: Unsteady Simulation: Six-Degree Oscillation, Low k at $\frac{T}{2}$ (Chart)

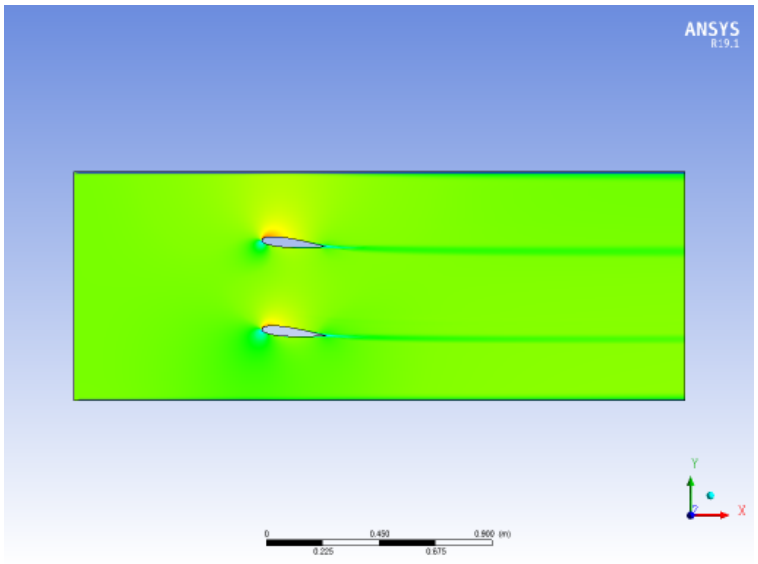

Figure 4.37: Unsteady Simulation: Six-Degree Oscillation, Low k at $\frac{3T}{4}$ (Snapshot)

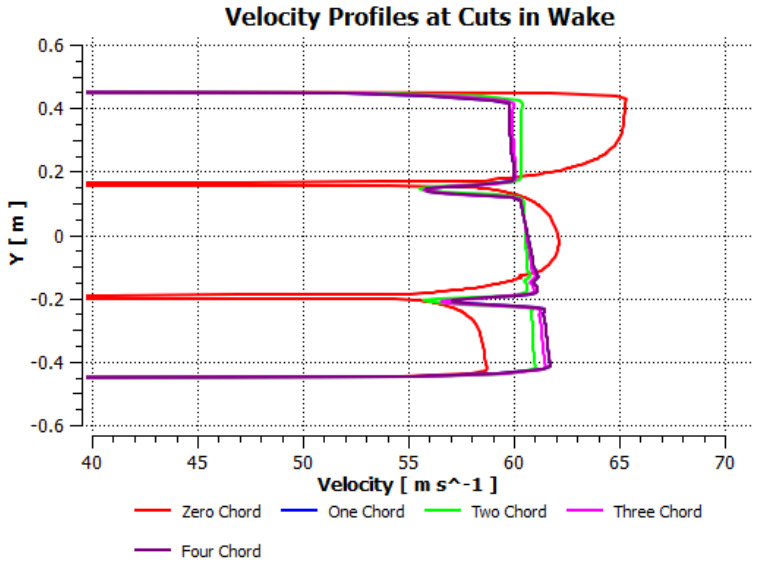

Figure 4.38: Unsteady Simulation: Six-Degree Oscillation, Low k at $\frac{3T}{4}$ (Chart)

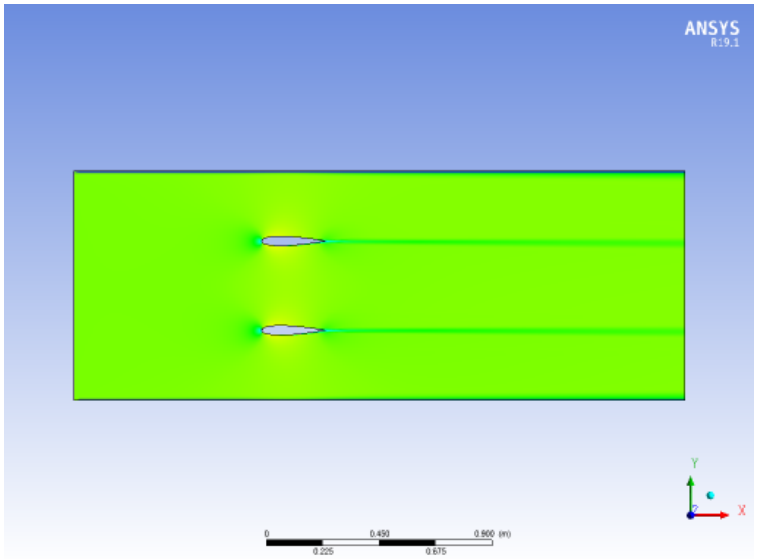

Figure 4.39: Unsteady Simulation: Six-Degree Oscillation, Low k at T (Snapshot)

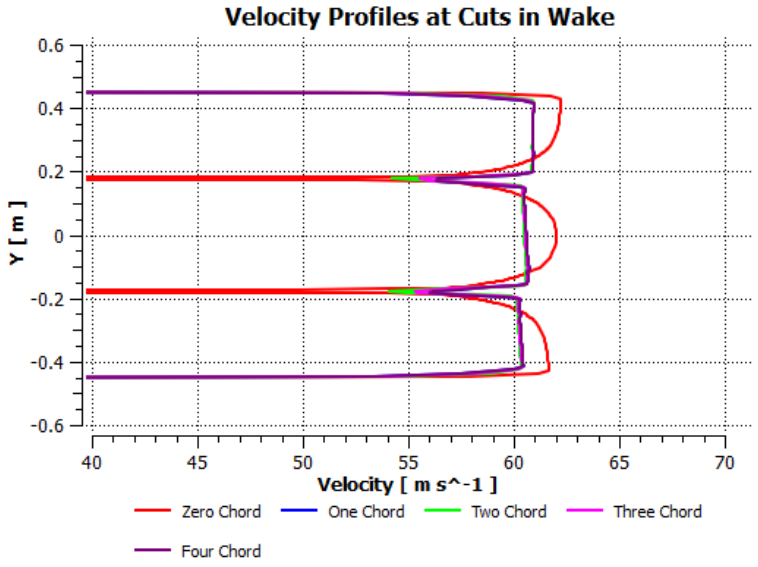

Figure 4.40: Unsteady Simulation: Six-Degree Oscillation, Low k at T (Chart)

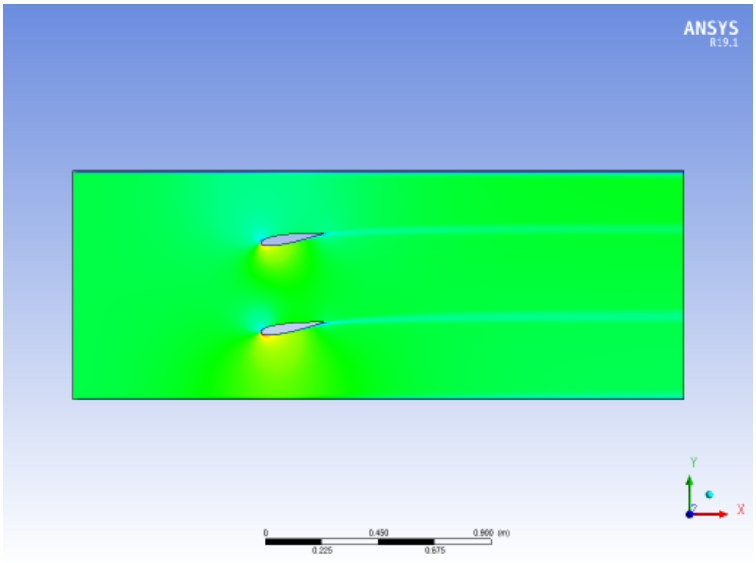

Figure 4.41: Unsteady Simulation: Nine-Degree Oscillation, Low k at $\frac{T}{4}$ (Snapshot)

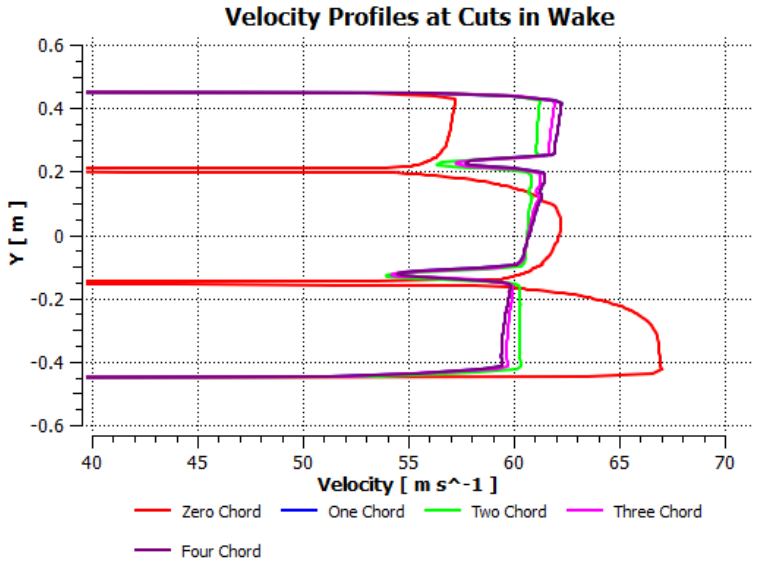

Figure 4.42: Unsteady Simulation: Nine-Degree Oscillation, Low k at $\frac{T}{4}$ (Chart)

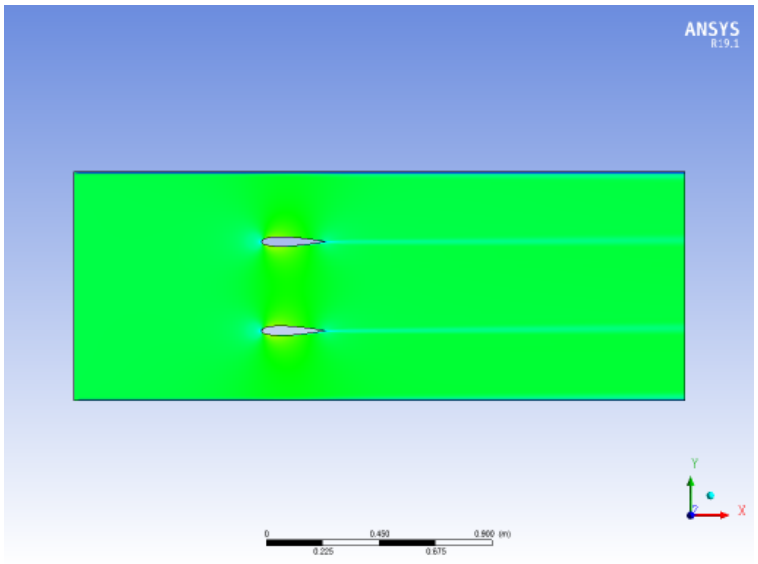

Figure 4.43: Unsteady Simulation: Nine-Degree Oscillation, Low k at $\frac{T}{2}$ (Snapshot)

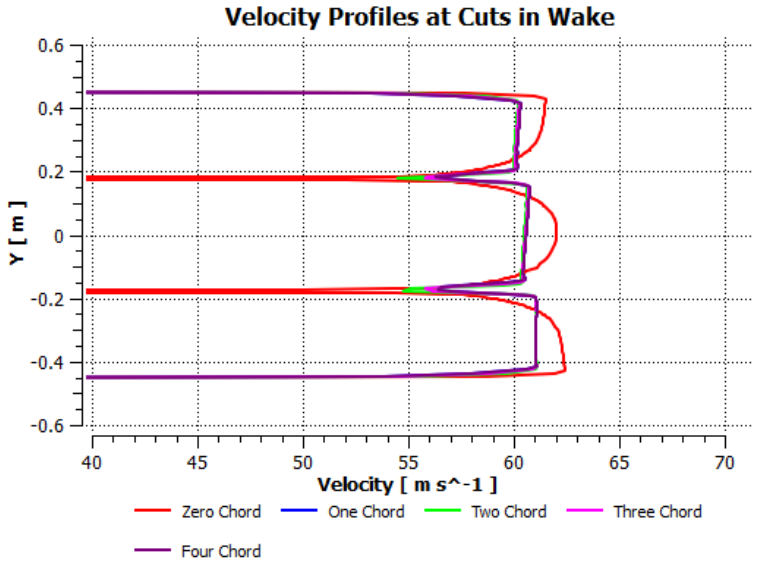

Figure 4.44: Unsteady Simulation: Nine-Degree Oscillation, Low k at $\frac{T}{2}$ (Chart)

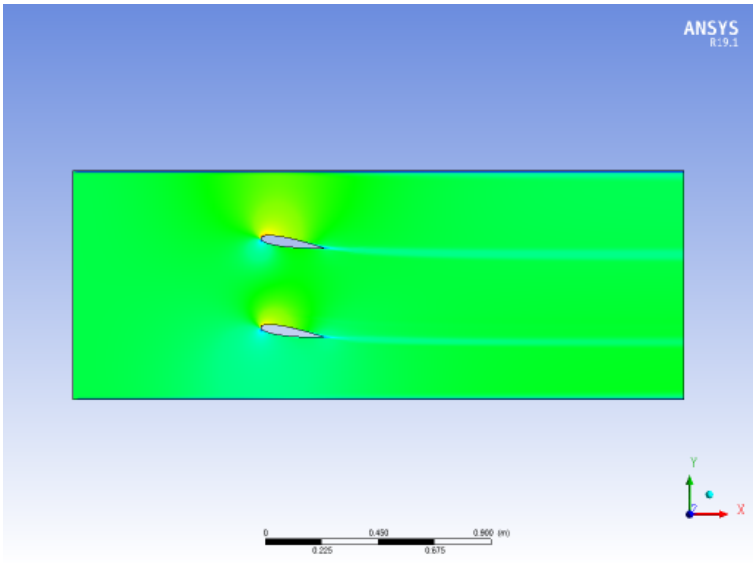

Figure 4.45: Unsteady Simulation: Nine-Degree Oscillation, Low k at $\frac{3T}{4}$ (Snapshot)

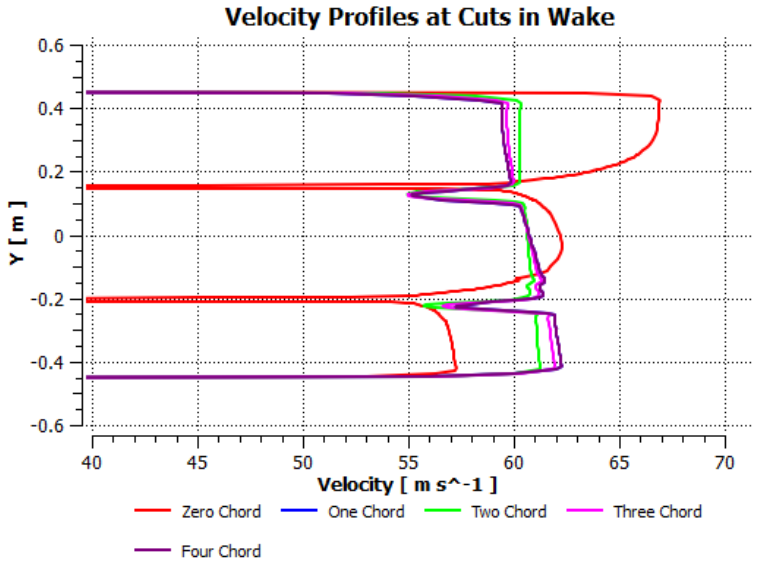

Figure 4.46: Unsteady Simulation: Nine-Degree Oscillation, Low k at $\frac{3T}{4}$ (Chart)

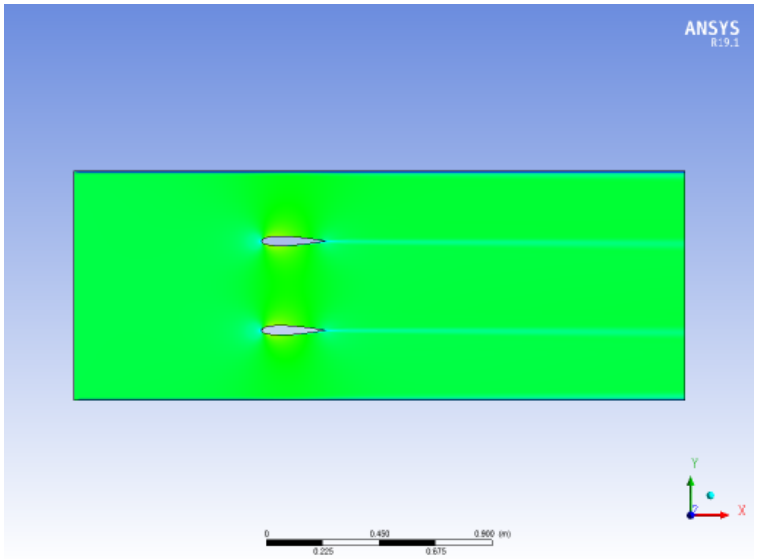

Figure 4.47: Unsteady Simulation: Nine-Degree Oscillation, Low k at T (Snapshot)

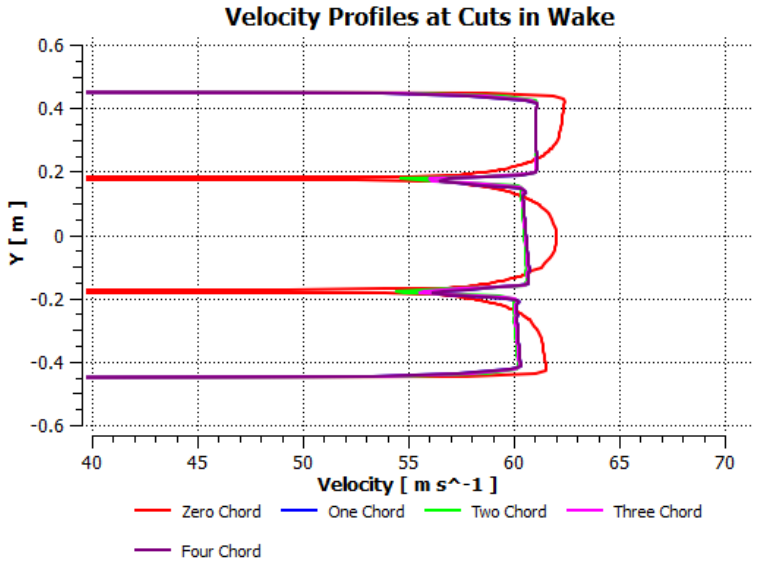

Figure 4.48: Unsteady Simulation: Nine-Degree Oscillation, Low k at T (Chart)

The plots of the velocity profiles at cross-sectional cuts in the wake show that the low reduced-frequency simulations match the steady-state simulation results closely. Looking at the $0.25T$ and $0.75T$ timesteps, where the model is at the maximum amplitude of oscillation, the maximum velocities measured in the unsteady simulation mirror the maximum velocities found in the steady-state cases for 3 degree, 6 degree and 9 degree perturbations. Considering the $0.5T$ and T timesteps, where the model is at zero AOA, a slight asymmetry can be seen in the maximum velocity between the bottom (right) airfoil and the bottom (right) wall compared to the maximum velocity between the top (left) airfoil and the top (left) wall. However, even for the nine-degree oscillation case, the difference is less than 1 m/s. The above results indicate, for such a low reduced-frequency, the unsteady effects can be effectively ignored. The flow changes at a slow enough rate that it can be considered quasi-steady and analyzed with reasonable confidence using steady-state methods.

4.7.2 High Reduced-Frequency (24 Hz) Simulations

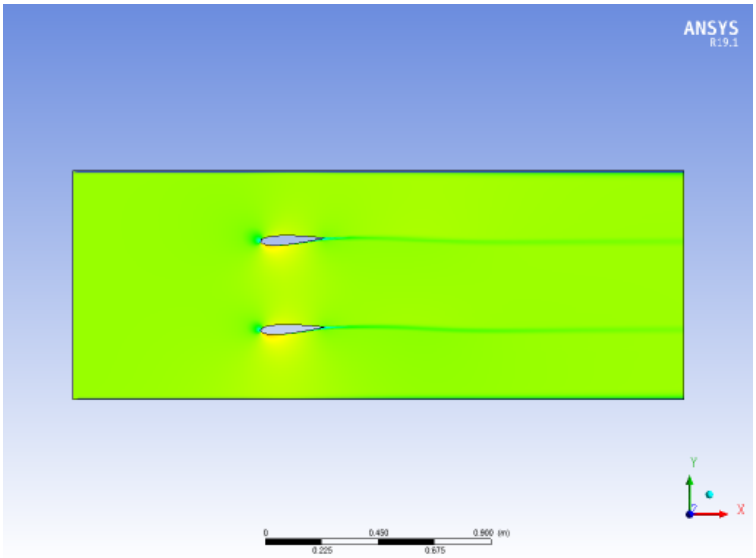

Figure 4.49: Unsteady Simulation: Three-Degree Oscillation, High k at $\frac{T}{4}$ (Snapshot)

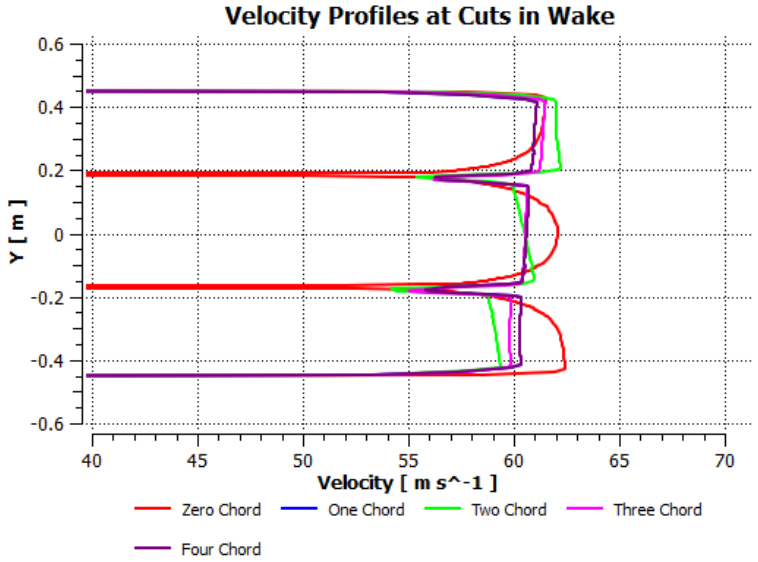

Figure 4.50: Unsteady Simulation: Three-Degree Oscillation, High k at $\frac{T}{4}$ (Chart)

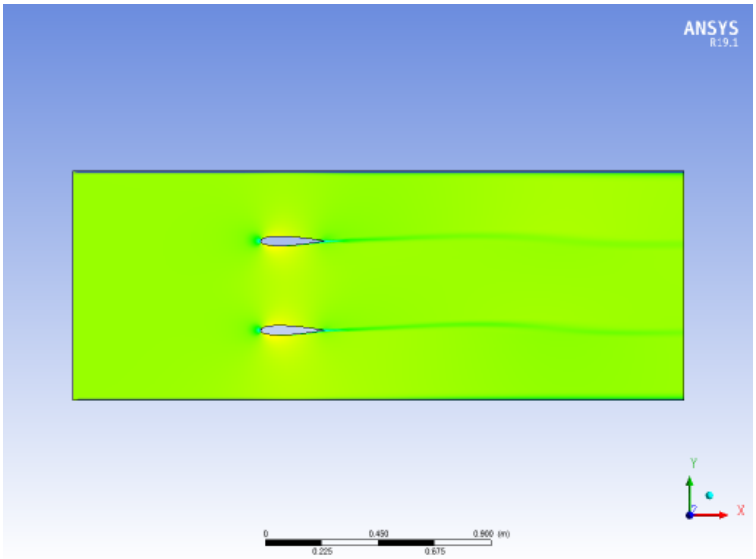

Figure 4.51: Unsteady Simulation: Three-Degree Oscillation, High k at $\frac{T}{2}$ (Snapshot)

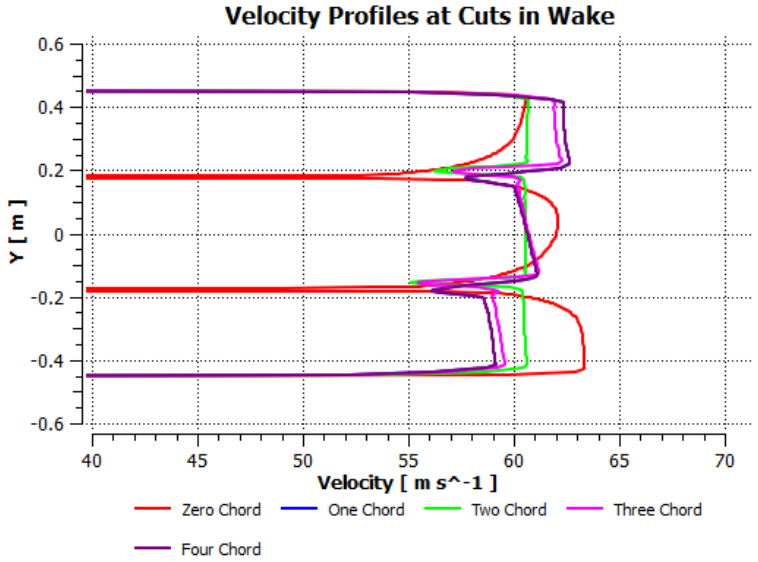

Figure 4.52: Unsteady Simulation: Three-Degree Oscillation, High k at $\frac{T}{2}$ (Chart)

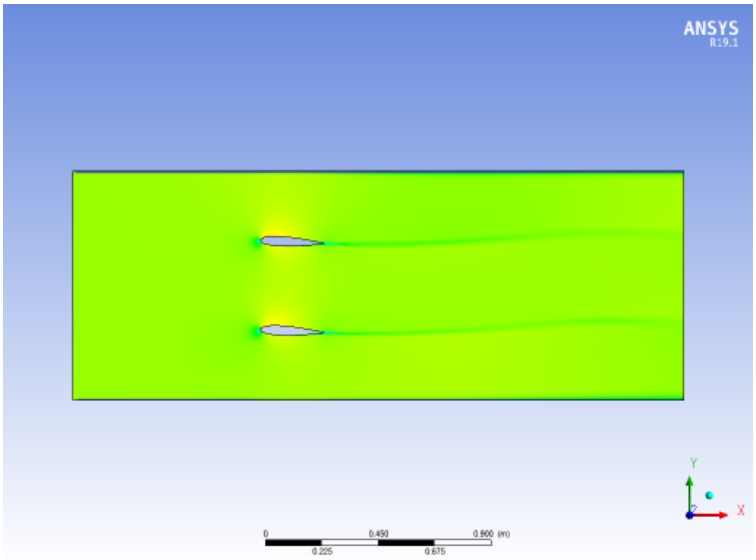

Figure 4.53: Unsteady Simulation: Three-Degree Oscillation, High k at $\frac{3T}{4}$ (Snapshot)

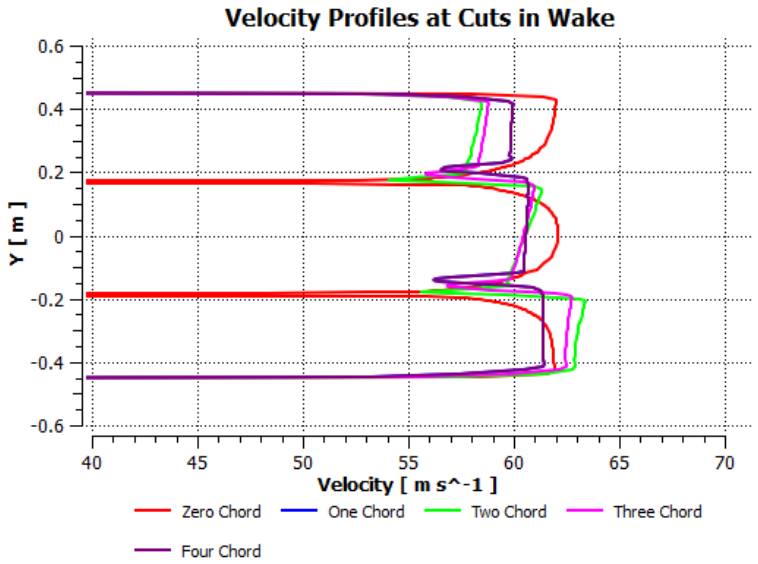

Figure 4.54: Unsteady Simulation: Three-Degree Oscillation, High k at $\frac{3T}{4}$ (Chart)

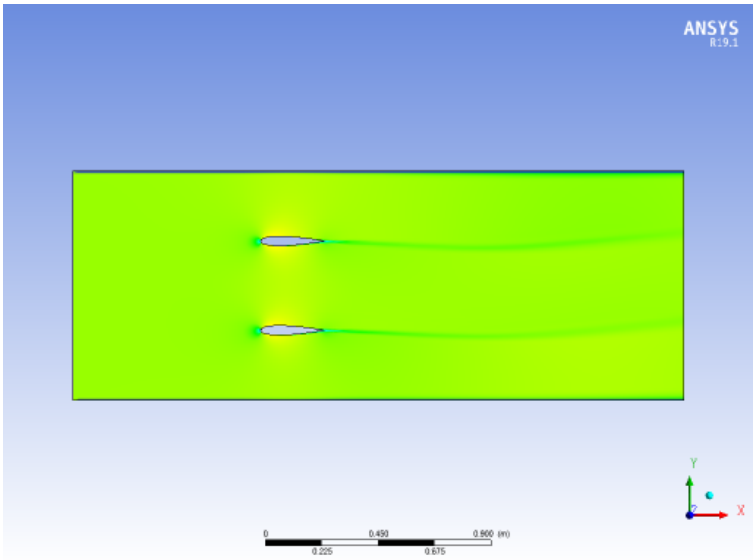

Figure 4.55: Unsteady Simulation: Three-Degree Oscillation, High k at T (Snapshot)

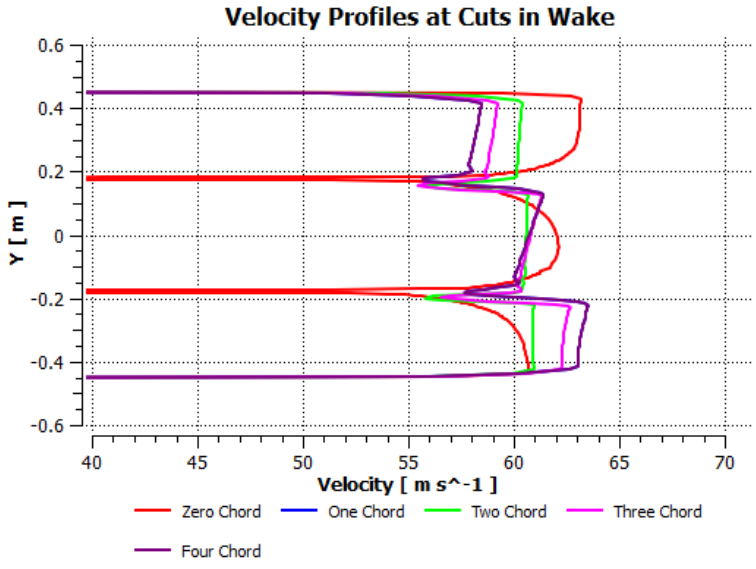

Figure 4.56: Unsteady Simulation: Three-Degree Oscillation, High k at T (Chart)

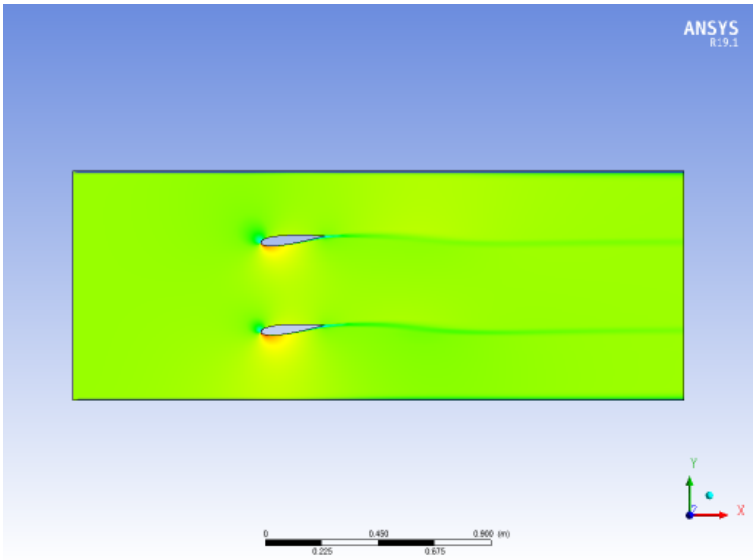

Figure 4.57: Unsteady Simulation: Six-Degree Oscillation, High k at $\frac{T}{4}$ (Snapshot)

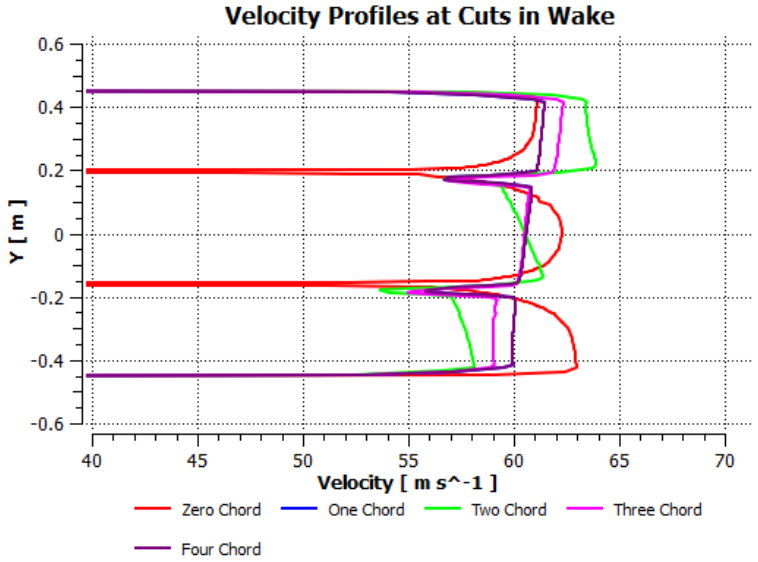

Figure 4.58: Unsteady Simulation: Six-Degree Oscillation, High k at $\frac{T}{4}$ (Chart)

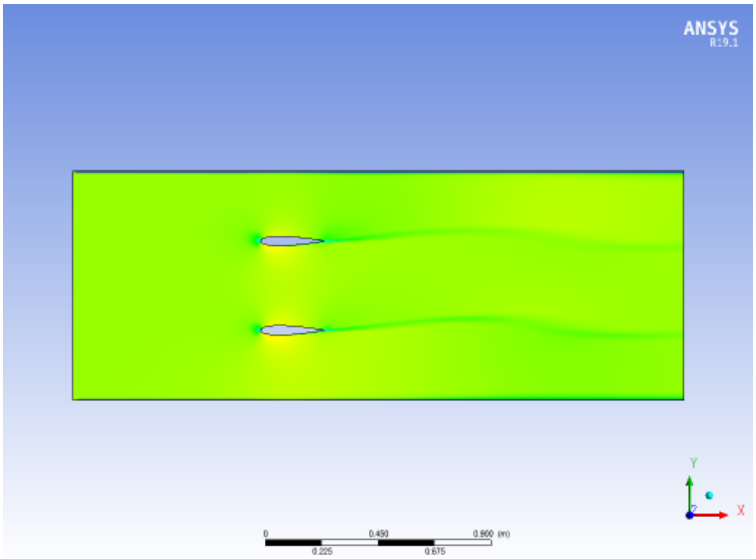

Figure 4.59: Unsteady Simulation: Six-Degree Oscillation, High k at $\frac{T}{2}$ (Snapshot)

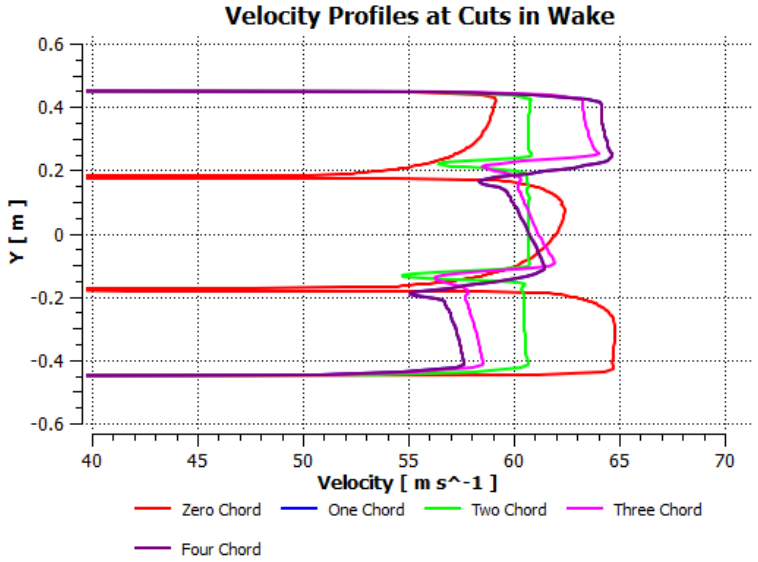

Figure 4.60: Unsteady Simulation: Six-Degree Oscillation, High k at $\frac{T}{2}$ (Chart)

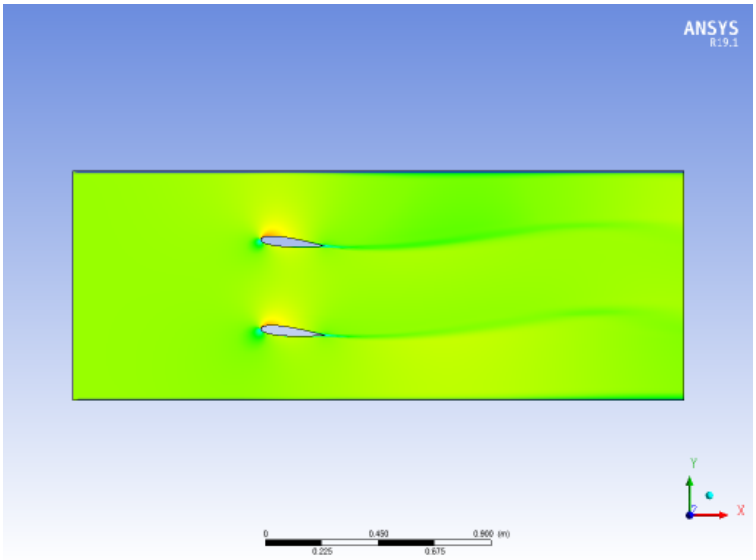

Figure 4.61: Unsteady Simulation: Six-Degree Oscillation, High k at $\frac{3T}{4}$ (Snapshot)

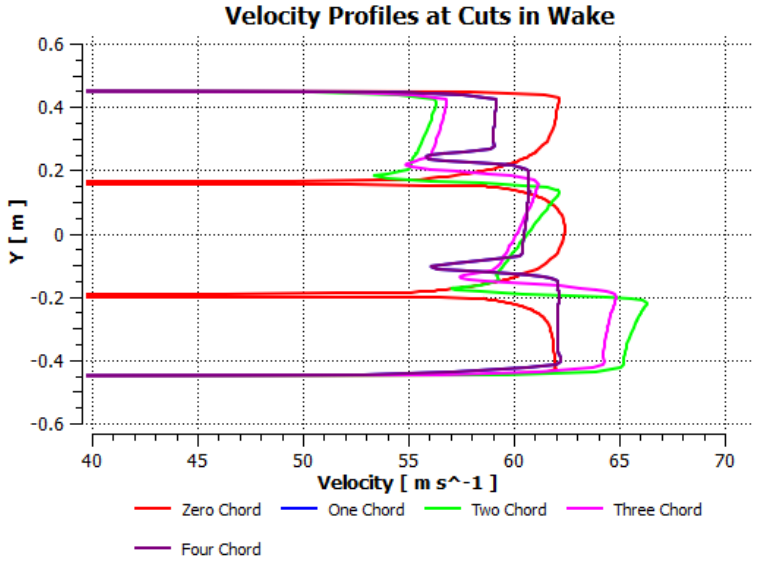

Figure 4.62: Unsteady Simulation: Six-Degree Oscillation, High k at $\frac{3T}{4}$ (Chart)

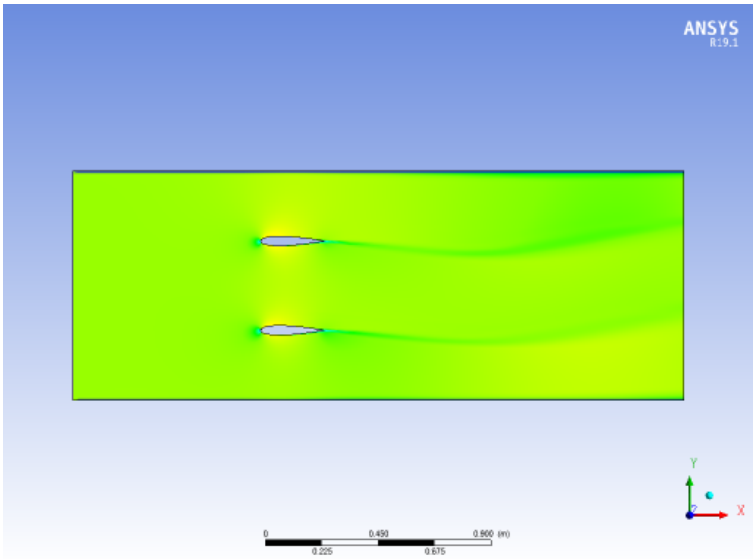

Figure 4.63: Unsteady Simulation: Six-Degree Oscillation, High k at T (Snapshot)

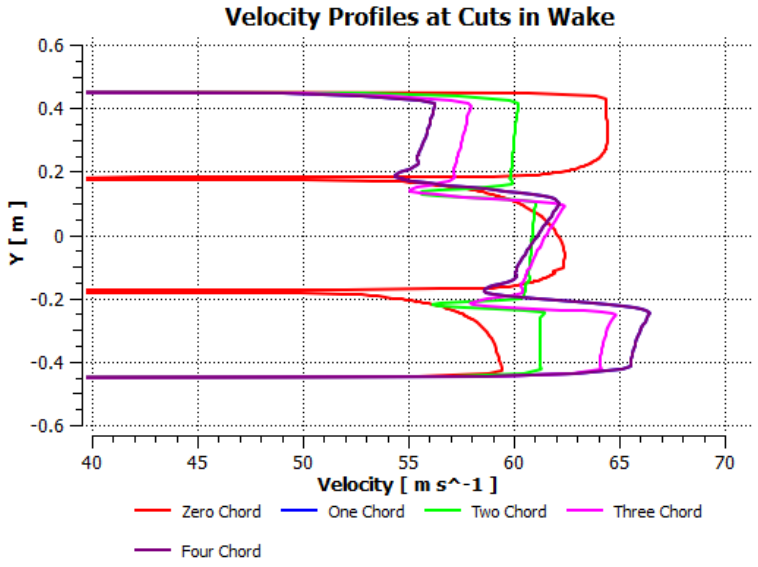

Figure 4.64: Unsteady Simulation: Six-Degree Oscillation, High k at T (Chart)

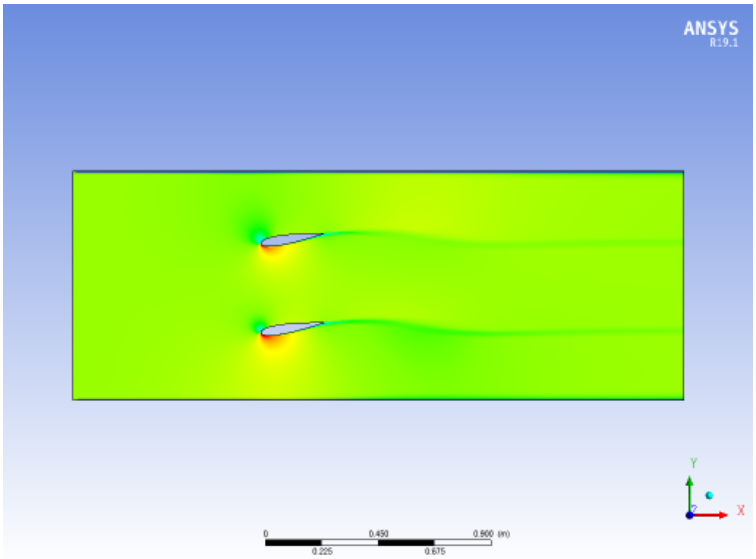

Figure 4.65: Unsteady Simulation: Nine-Degree Oscillation, High k at $\frac{T}{4}$ (Snapshot)

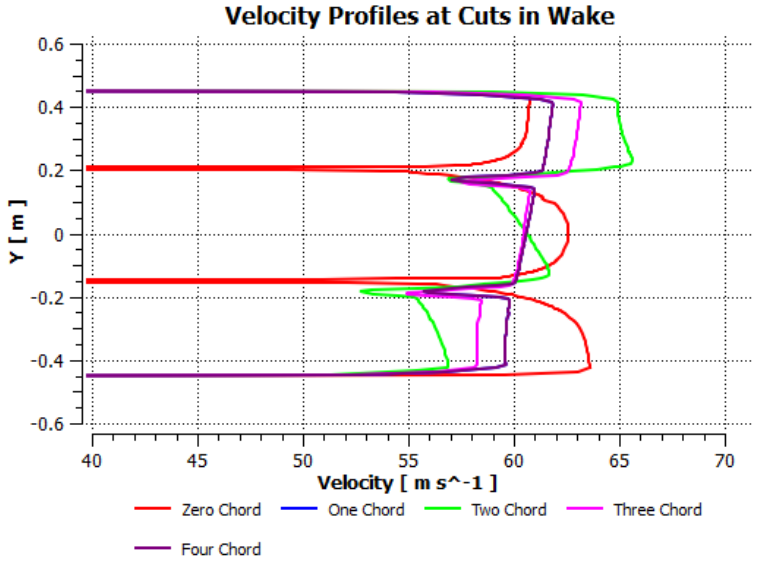

Figure 4.66: Unsteady Simulation: Nine-Degree Oscillation, High k at $\frac{T}{4}$ (Chart)

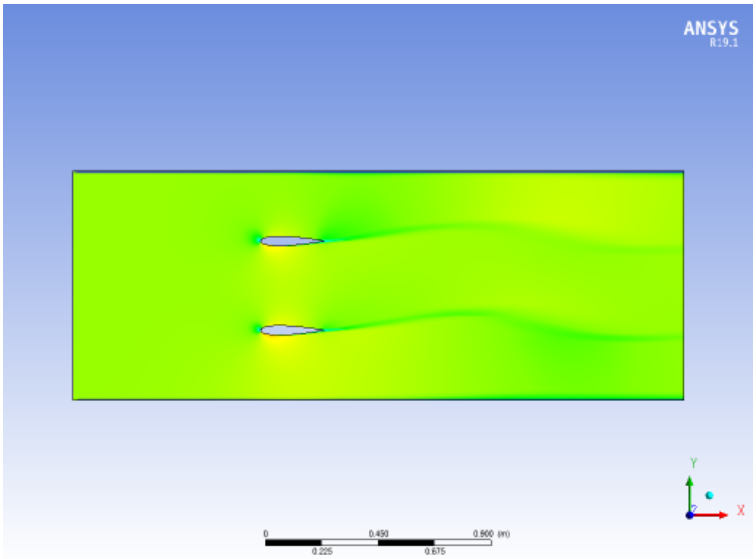

Figure 4.67: Unsteady Simulation: Nine-Degree Oscillation, High k at $\frac{T}{2}$ (Snapshot)

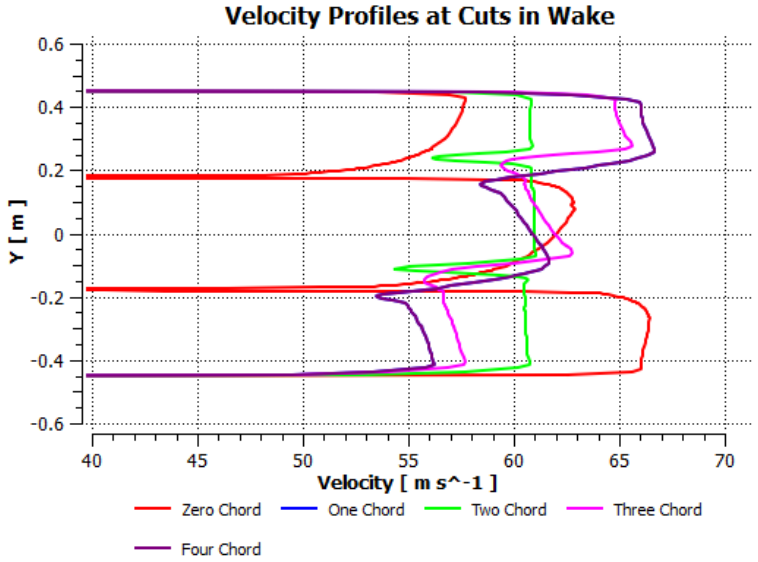

Figure 4.68: Unsteady Simulation: Nine-Degree Oscillation, High k at $\frac{T}{2}$ (Chart)

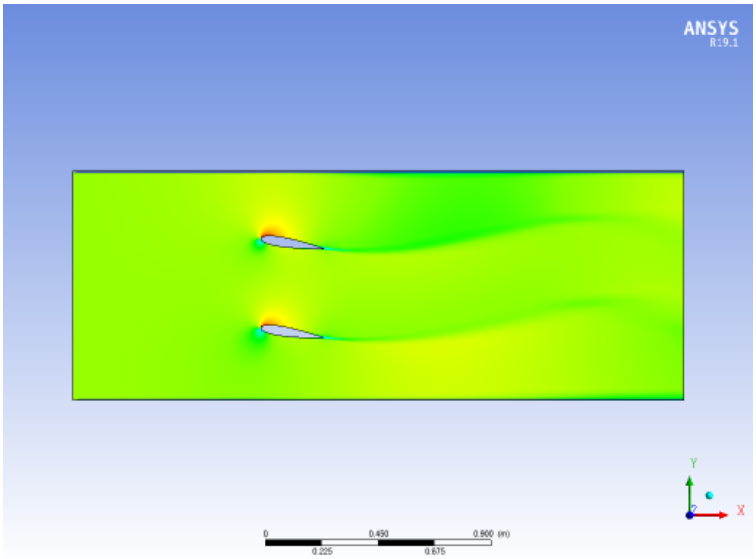

Figure 4.69: Unsteady Simulation: Nine-Degree Oscillation, High k at $\frac{3T}{4}$ (Snapshot)

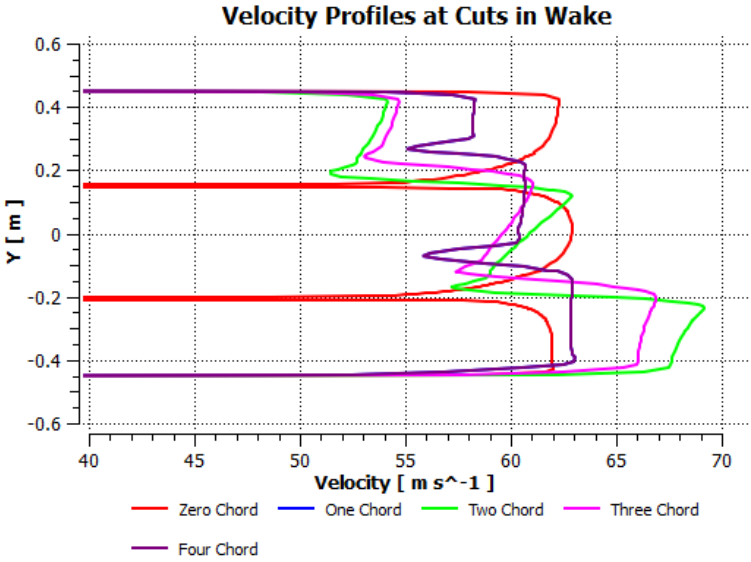

Figure 4.70: Unsteady Simulation: Nine-Degree Oscillation, High k at $\frac{3T}{4}$ (Chart)

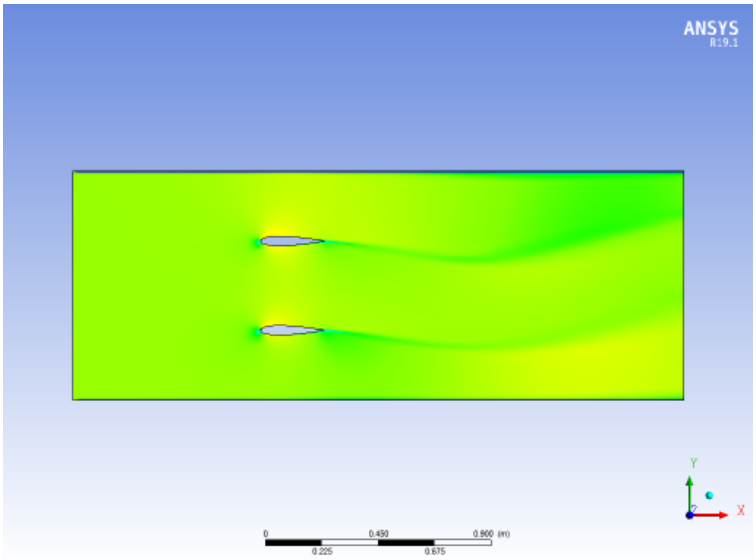

Figure 4.71: Unsteady Simulation: Nine-Degree Oscillation, High k at T (Snapshot)

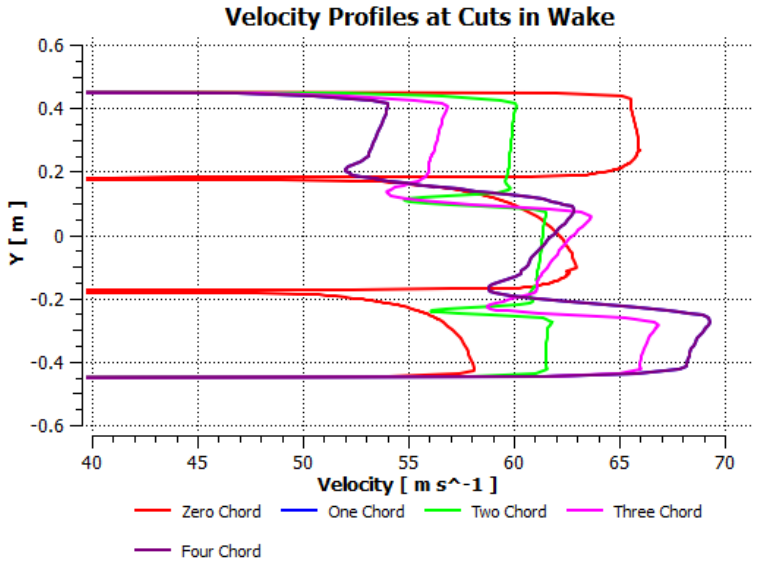

Figure 4.72: Unsteady Simulation: Nine-Degree Oscillation, High k at T (Chart)

The high reduced-frequency simulations demonstrated significantly different flow field behavior than the low reduced-frequency simulations. The maximum velocity experienced at

the trailing edge of the gust vanes occurred when the airfoils are at zero AOA, corresponding to the $0.5T$ and T timesteps. Such a result differed from the steady-state and low reduced-frequency unsteady simulations, where the maximum velocity occurred at the maximum amplitude of oscillation. The maximum velocity measured at the trailing edge of the airfoils for the high reduced-frequency case was also slightly less than the maximum velocity measured in the same spot for the steady-state case. Additionally, the velocities measured at cross-sectional cuts in the wake fluctuated at a greater magnitude than the velocities measured in the steady-state simulations. At the $0.75T$ and T timesteps, the maximum velocity at two chord lengths from the trailing edge and four chord lengths from the trailing edge respectively were higher than the maximum velocity measured at the trailing edge cut.

The six-degree oscillation case showed similar results as the three-degree oscillation case with greater magnitudes of disturbance. The maximum velocity at the trailing edge was seen at the $0.5T$ and T timesteps, as is in the three-degree oscillation case. The highest magnitude velocity measured on the trailing edge cut for the high reduced-frequency six-degree oscillation case was also lower in magnitude than the respective velocity measured in the steady-state case. However, the velocities measured at cuts in the wake for the high reduced-frequency simulation were higher than those measured in the low reduced-frequency simulation.

The high reduced-frequency nine-degree oscillation case continued the trends of the corresponding six-degree oscillation and three-degree oscillation cases. Once again, the maximum velocity at the trailing edge cut occurred at the $0.5T$ and T timesteps. Also, the maximum trailing edge velocity was less in magnitude than the maximum trailing edge velocity measured during the steady-state case with nine-degrees perturbation. However, the nine-degree oscillation case differed from the six-degree oscillation and three-degree oscillation case in that for every timestep considered, the maximum velocity at the trailing edge was of lesser magnitude than the maximum velocity of at least one cross-section cut in the wake.

The high reduced-frequency simulations differed considerably from the low reduced-frequency simulations. The high reduced-frequency simulations demonstrated that steady-

state or quasi-steady analysis techniques would not be accurate for the given problem. An unsteady analysis is necessary to correctly capture the flow field created by the gust generator in the wind tunnel at higher reduced frequencies, especially when using larger amplitudes of oscillation.

Returning to the comparison of CFD results for the free airfoil and the corresponding Theodorsen solution, small differences in force predictions were observed and the difference between the geometries of the cases (thick airfoil versus infinitely thin airfoil) were noted. For potential fully-attached flow, airfoil thickness (in the no-walls case) affects the flow field but not the net force and moment on the airfoil. CFD simulations, accounting for viscous effects, make it possible to study the flow field in search of better understanding the prediction differences discussed above.

4.7.3 Flow Separation Evaluation

As the following figure shows, some flow separation at the trailing edges of the airfoils is present. At high angles of attack significant flow separation can impact the flow field created by the gust vanes and needs to be adequately modeled.

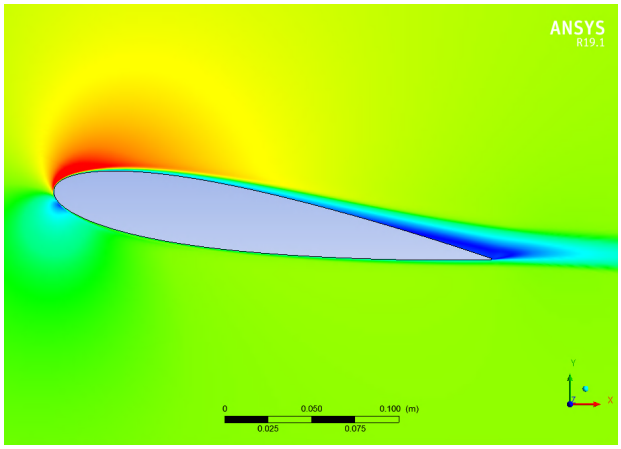

Figure 4.73: Close-In Flow Snapshot: Full Airfoil at Nine-Degree Oscillation, High k at $3/4T$

For the nine-degree oscillation in the high reduced frequency case and at the airfoil's

maximum amplitude, there is a small amount of separation at the trailing edge. That may explain the small differences between CFD and Theodorsen results. Overall,however, the amount of flow separation does not appear sufficiently significant to require additional investigation. The nine-degree oscillation in the high reduced frequency case leads, in the cases studied in this work, to the most severe flow separation. Therefore, flow separation, because of the small angles of attack involved, was not deemed a major issue for any of the simulations considered here.

4.8 Interaction Between the Gust Vanes and a Wing Mounted Behind Them

The gust generation system's effect on a single test wing mounted behind the vanes was analyzed. The lift-curve slope of the single test airfoil without the influence of the gust vanes was compared to the lift-curve slope of the test airfoil when the test airfoil was located behind the gust vanes.

Figure 4.74: Test Configuration Example - 1 Chord Length at wing 6 Degrees AOA

The gust vanes were stationary and set at zero degrees AOA. The test airfoil was simulated at 0 degree, 3 degree, and 6 degree AOA with the test airfoil rotated about the 1/4-chord. The test airfoil was a NACA 0012 airfoil. The chord length of the test airfoil was 6.55 in. The location of the test airfoil relative to the test vanes was varied. The test airfoil was located one chord length, two chord-lengths, and three chord-lengths behind the gust vanes

along the centerline of the tunnel. The chord length used to set the position of the test airfoil was the test airfoil's chord length. The dynamic pressure was 2130 Pa.

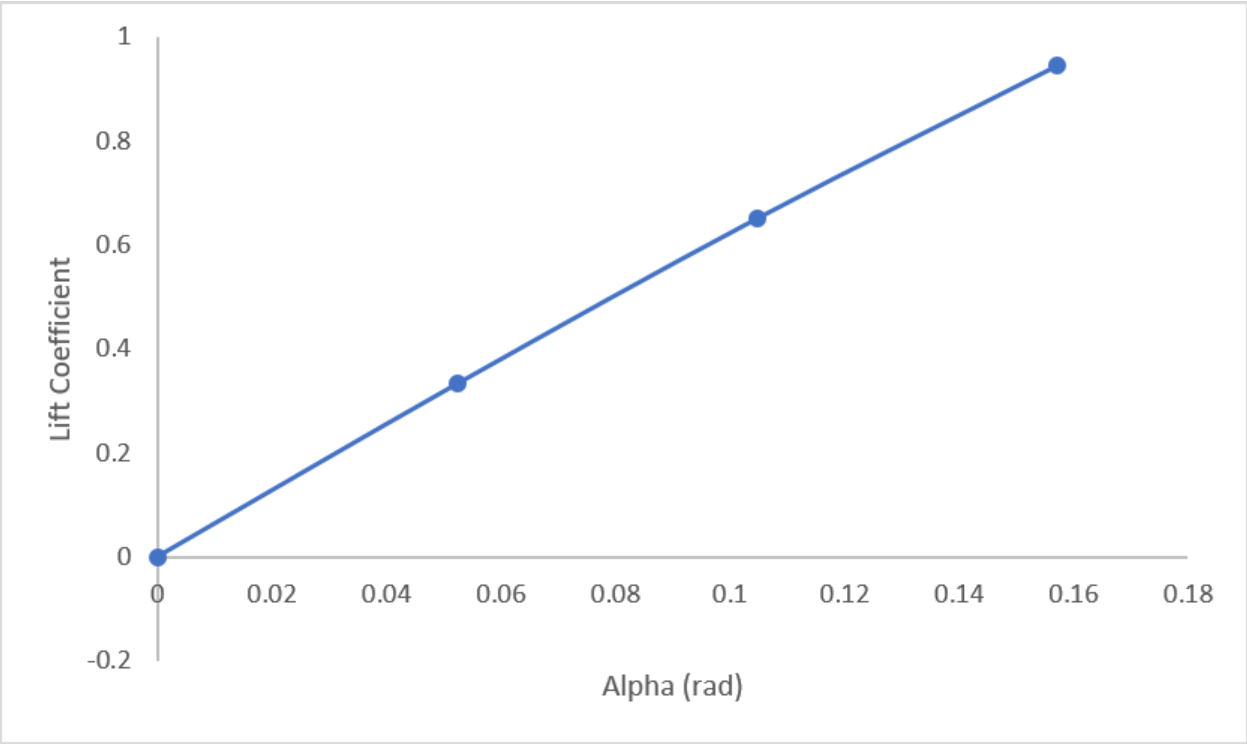

Figure 4.75: Single Wing Lift-Curve Slope with No Gust Generator

The lift-curve slope calculated for the single airfoil in the wind tunnel without the influence of the gust vanes was $C_{L_\alpha} = 6.22$ 1/rad. The calculated lift-curve slope differed from the free-airfoil theoretical value of 2π by 2.01 percent.

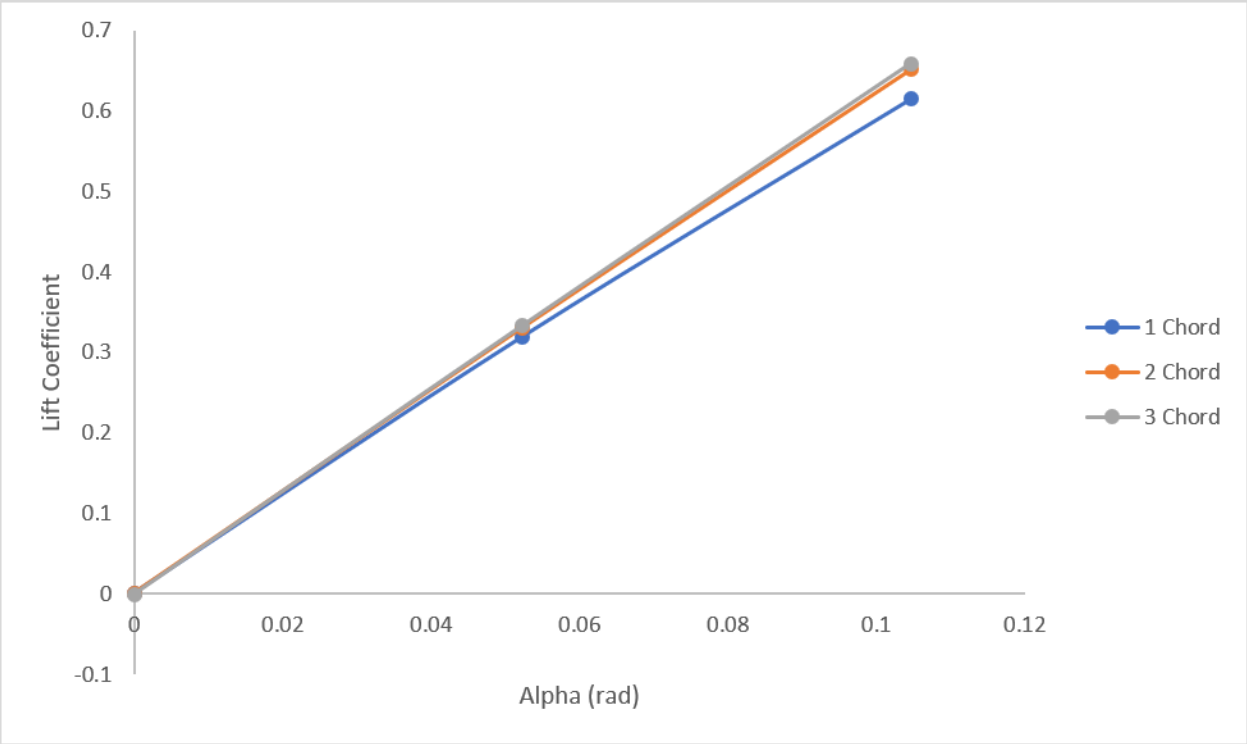

Figure 4.76: Single Wing Lift-Curve Slope Under the Influence of the Gust System

The single wing with no gust vanes case was compared against the single wing with the gust system cases. The lift-curve slope of the wing located one chord length behind the gust vanes was calculated to be 5.87 1/rad. The measured lift-curve slope differed from the free-wing theoretical value of 2π by 13.15 percent. The lift-curve slope of the wing located two chord lengths behind the gust vanes was found to be 6.22 1/rad. The measured lift-curve slope differed from the theoretical free-wing value of 2π by 2.01 percent. The lift-curve slope of the wing located three chord lengths behind the gust vanes was found to be 6.30 1/rad. The measured lift-curve slope differed from the theoretical free-wing value of 2π by 0.53 percent.

Studying the four cases indicated that the presence of the gust vanes ahead (at zero vanes AOA) can have a significant impact on the lift produced by the test airfoil. When the test airfoil is within at least one chord length, 6.55 in, of the gust vanes, the lift produced by the

airfoil was diminished by more than 10 percent. Given the range of the other three estimates, it was not possible, based on the present studies, to find trends in the lift of the test airfoil as the test airfoil was moved further away from the gust system. More analysis would be necessary. It is clear, however, that when a wind tunnel model is placed too close to the gust vanes, the model and vanes would affect one another aerodynamically.

4.9 Unsteady Gust System-Test Wing and Tail System Interaction Study

The effect of the oscillating gust generator on a wind tunnel test object was studied further. The vanes were oscillated at three amplitudes: 3 degrees, 6 degrees and 9 degrees. Also, the vanes were oscillated at two frequencies: 1 Hz and 24 Hz. The effects of the changing amplitudes and frequencies on the lift force experienced by the test object was considered. The test object was based on an aeroelastic model previously used to test active control methods at the University of Washington aimed at gust suppression [28]. The test object was a wing-tail configuration, which in the 2D case was modeled as two airfoils. The airfoils were NACA 0012 airfoils. The chord length of the wing was 6.55 inch and the chord length of the tail was 5 inch.

The test airfoils were 27 in apart as measured from the 1/4-chord of the wing to the 1/4-chord of the tail. The test object remained stationary throughout the simulation. The lift forces experienced by the test wing and the test tail separately were calculated for each case.

4.9.1 Low Reduced-Frequency (1 Hz, 60 m/sec) Cases

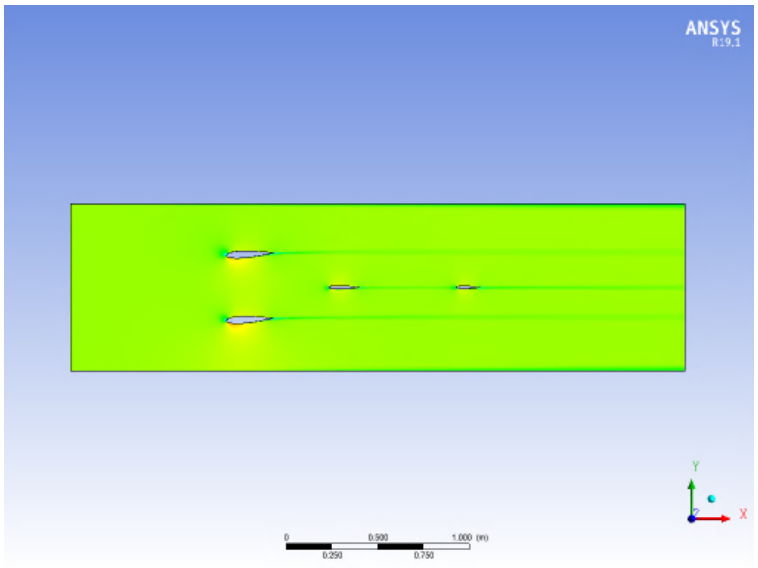

Figure 4.77: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 3 Degree, Low k Oscillation at $t = 1/4T$

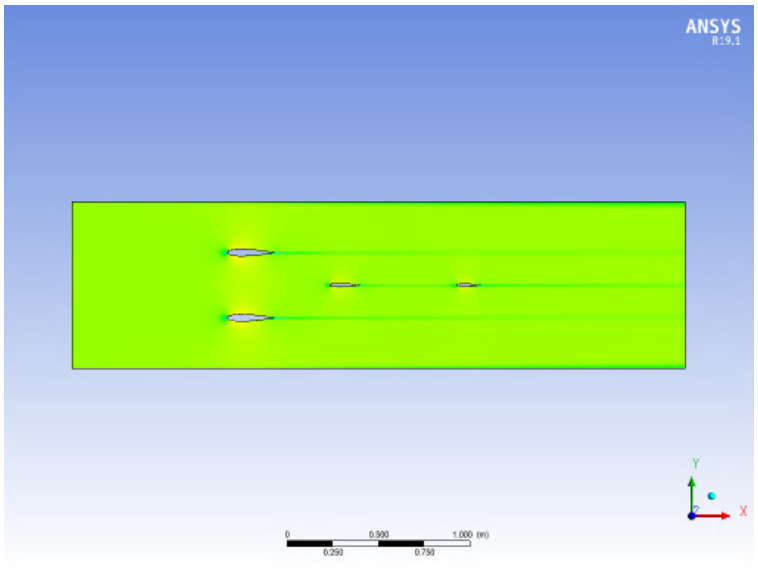

Figure 4.78: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 3 Degree, Low k Oscillation at $t = 1/2T$

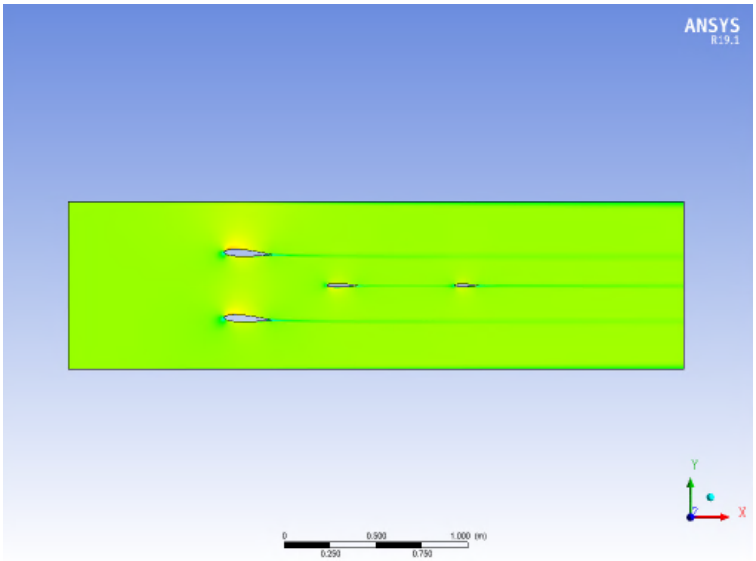

Figure 4.79: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 3 Degree, Low k Oscillation at $t = 3/4T$

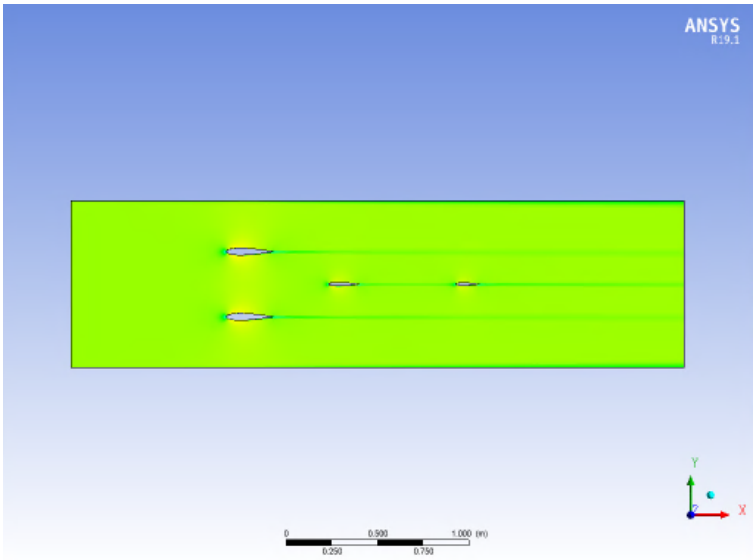

Figure 4.80: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 3 Degree, Low k Oscillation at $t = T$

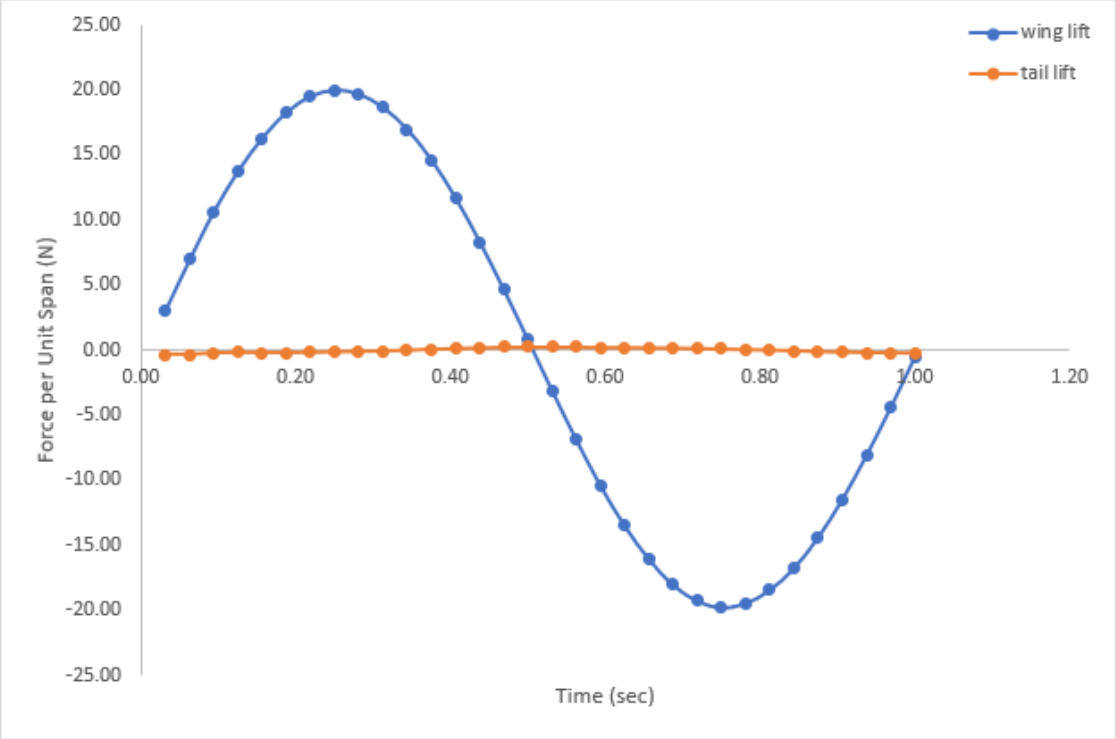

Figure 4.81: Unsteady Gust System-Test Object Interaction: Lift Forces on Wing and Tail for 3 Degree, Low k Oscillation

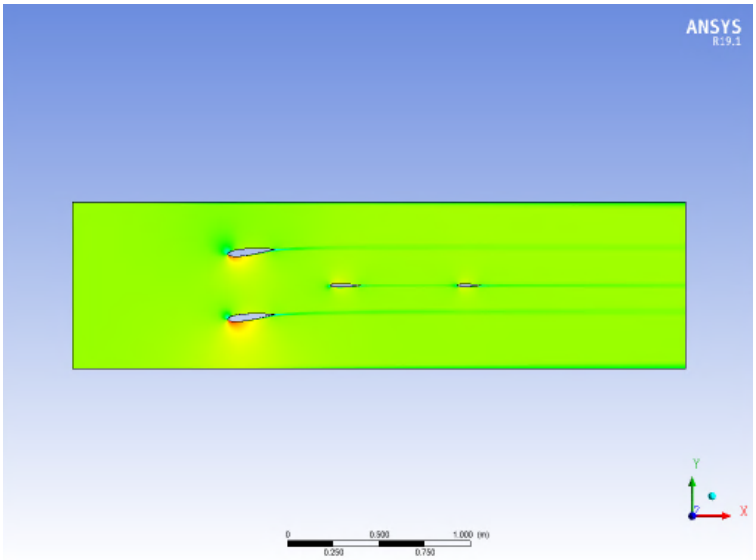

Figure 4.82: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 6 Degree, Low k Oscillation at $t = 1/4T$

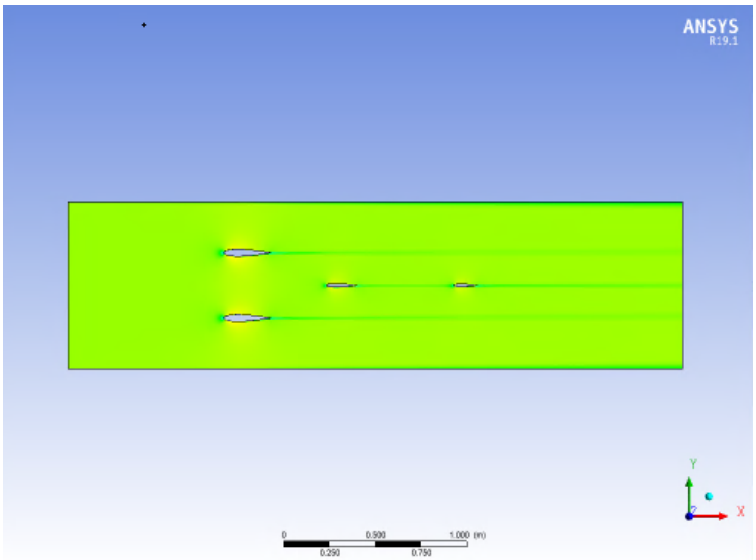

Figure 4.83: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 6 Degree, Low k Oscillation at $t = 1/2T$

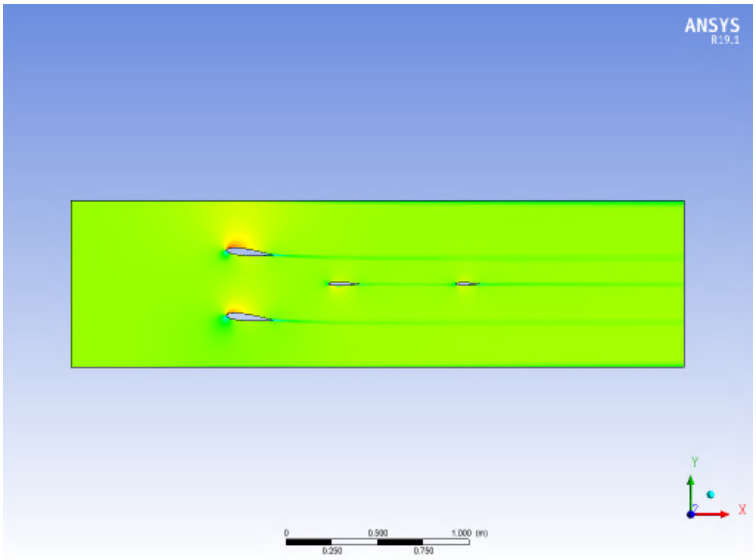

Figure 4.84: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 6 Degree, Low k Oscillation at $t = 3/4T$

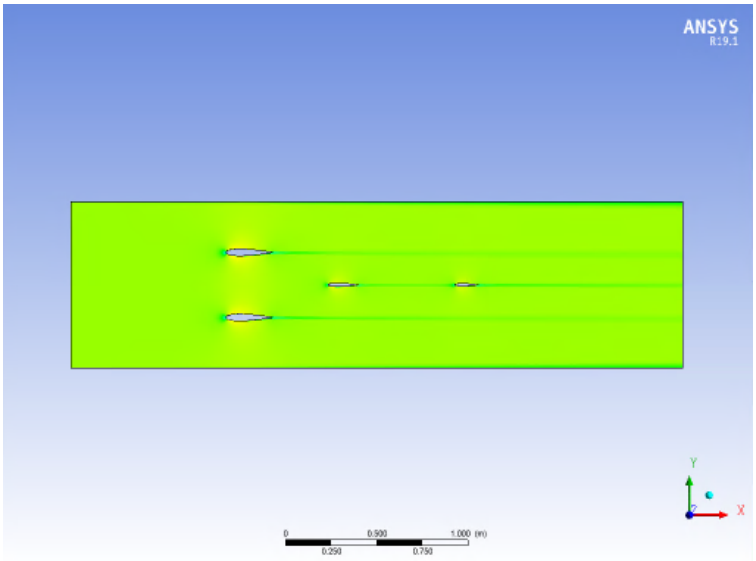

Figure 4.85: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 6 Degree, Low k Oscillation at $t = T$

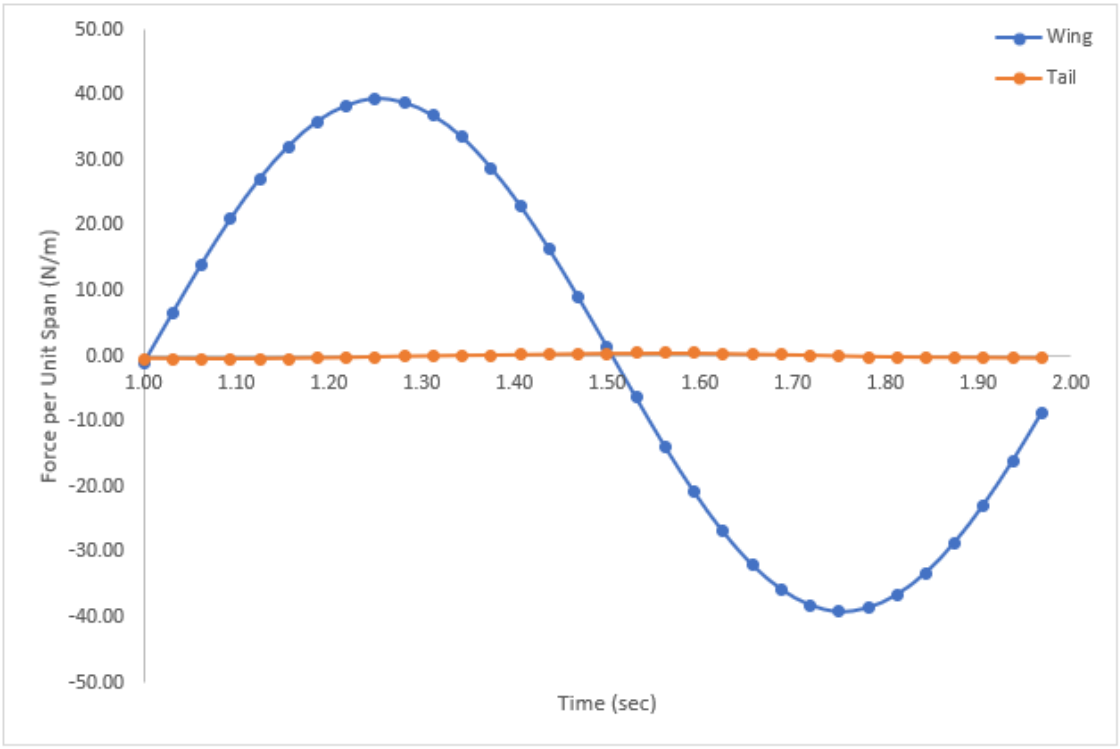

Figure 4.86: Unsteady Gust System-Test Object Interaction: Lift Forces on Wing and Tail for 6 Degree, Low k Oscillation

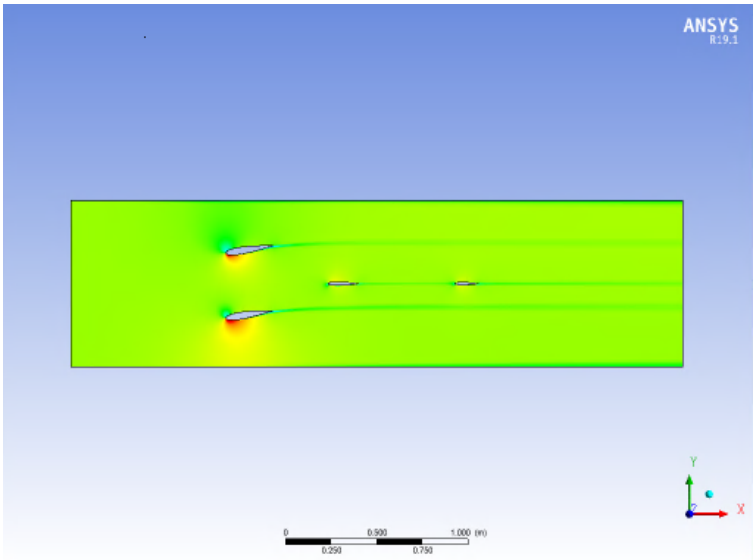

Figure 4.87: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 9 Degree, Low k Oscillation at $t = 1/4T$

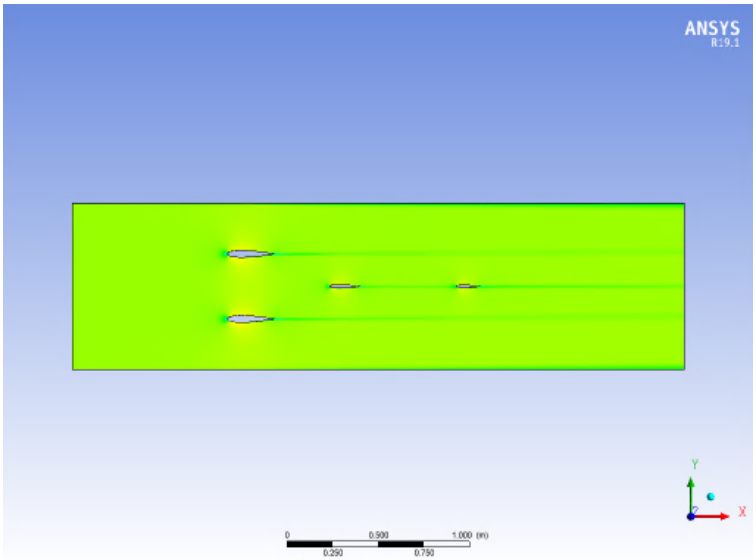

Figure 4.88: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 9 Degree, Low k Oscillation at $t = 1/2T$

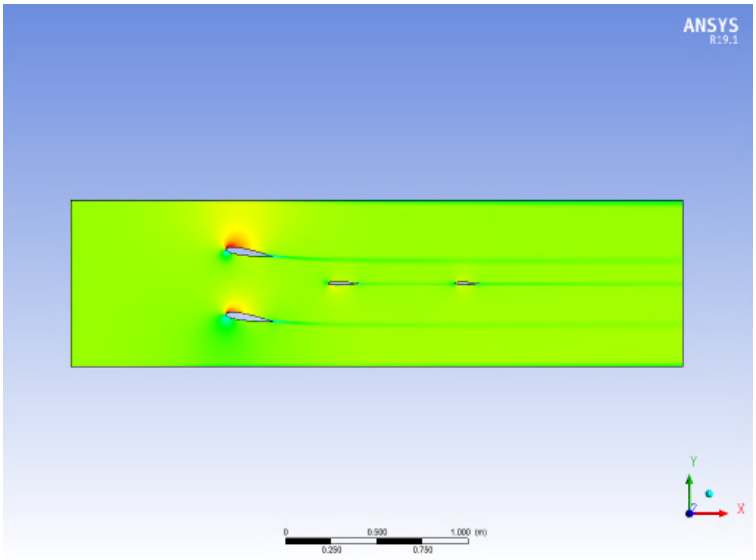

Figure 4.89: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 9 Degree, Low k Oscillation at $t = 3/4T$

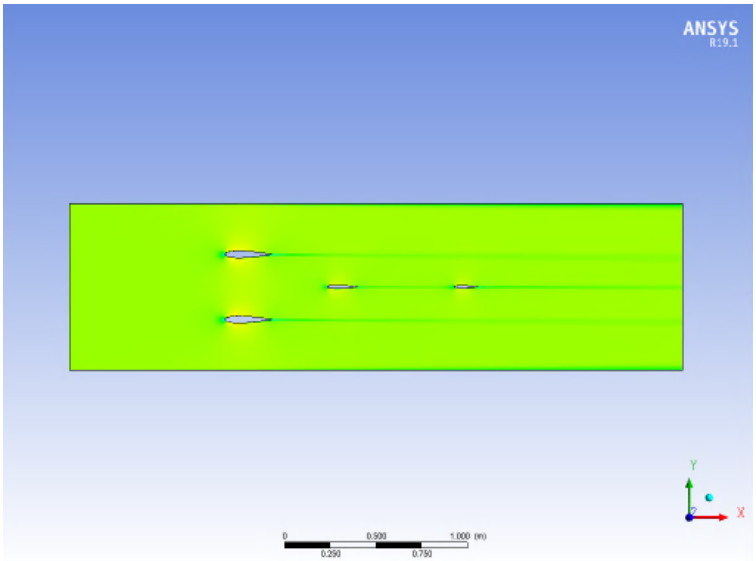

Figure 4.90: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 9 Degree, Low k Oscillation at $t = T$

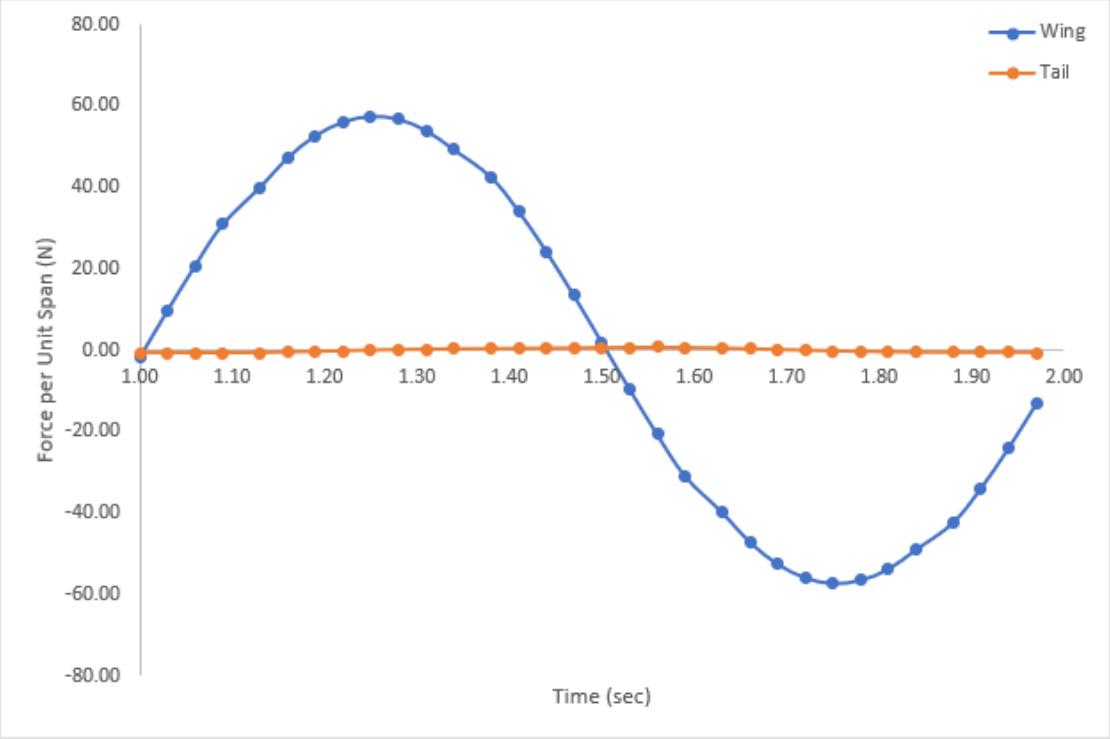

Figure 4.91: Unsteady Gust System-Test Object Interaction: Lift Forces on Wing and Tail for 9 Degree, Low k Oscillation

The low reduced-frequency simulations for each amplitude produced similar results. For each case, the lifting force experienced by the wing was approximately 180 degrees out of phase with the motion of the airfoil. The wing experienced an increase in lift as the gust vanes pitched down and vice versa. The magnitude of the lifting force on the wing increased in a generally linear pattern as the amplitude of oscillation of the gust vanes increased at a rate of $\sim 6.7 N/degree$. The lifting force felt by the tail was very small for all cases. Such a result was expected. For low-frequency simulations, as shown in previous studies, gust effects produced by the gust vanes dissipated quickly with distance into the wake. The flow field generally returns to the freestream flow field by the time the tail is encountered. Therefore, the uniform flow field-symmetric airfoil interaction resulted in little to no lifting force on the tail. Additionally, the tail was covered by the direct wake of the wing. The tail saw

a slower flow speed than it would have seen if not covered by the wake of the wing. The reduced flow speed also may have impacted the lifting force experienced by the tail. Such results highlight the value of the quasi-steady simulations of the low-reduced-frequency cases. As documented previously, such conditions can be modeled with reasonable accuracy using steady-state techniques.

4.9.2 High Reduced-Frequency (24 Hz, 60 m/sec) Cases

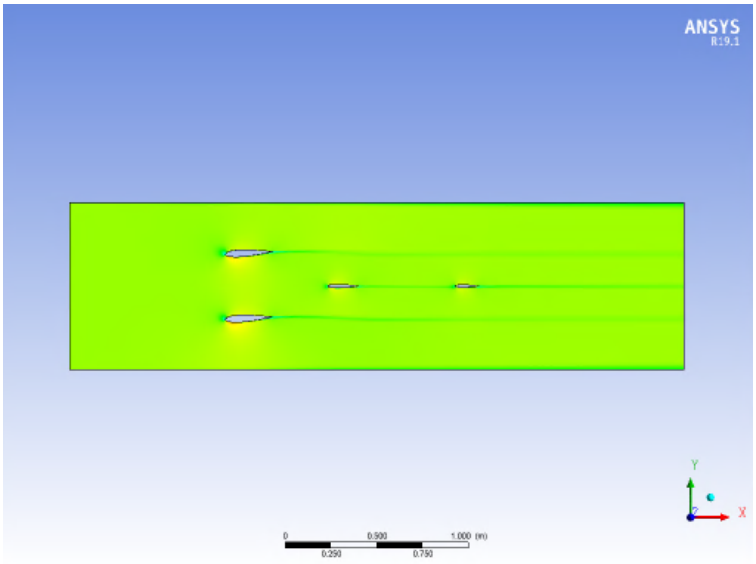

Figure 4.92: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 3 Degree, High k Oscillation at $t = 1/4T$

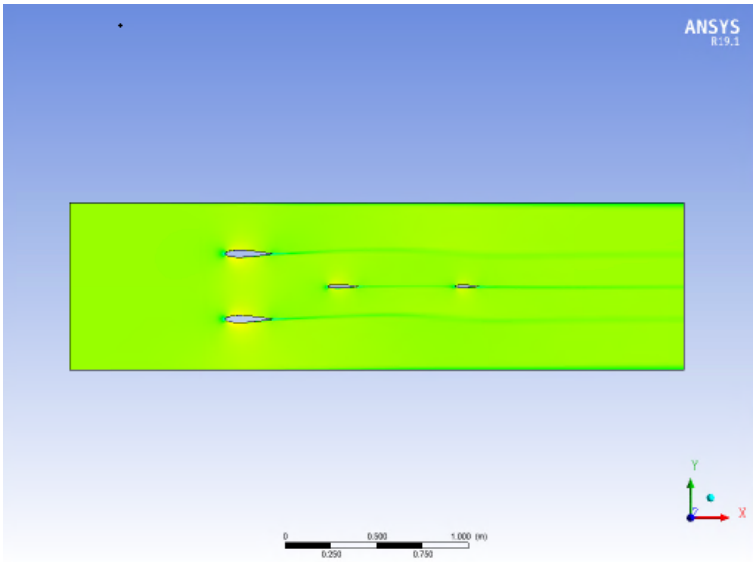

Figure 4.93: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 3 Degree, High k Oscillation at $t = 1/2T$

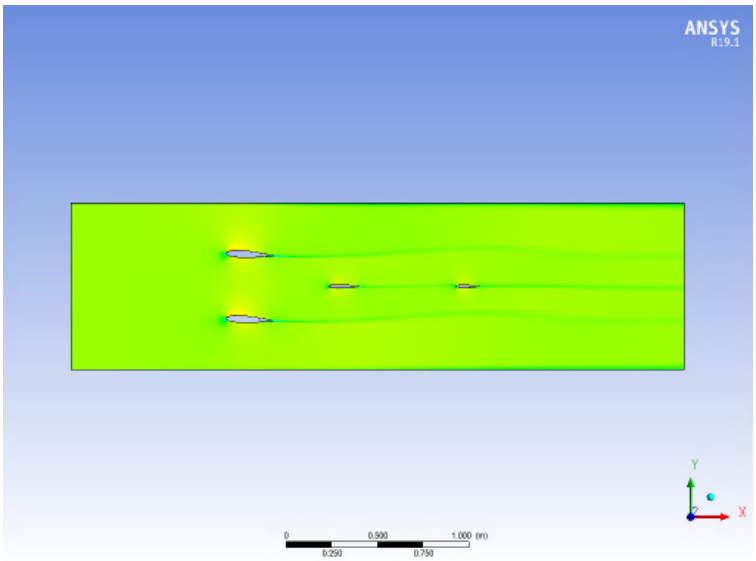

Figure 4.94: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 3 Degree, High k Oscillation at $t = 3/4T$

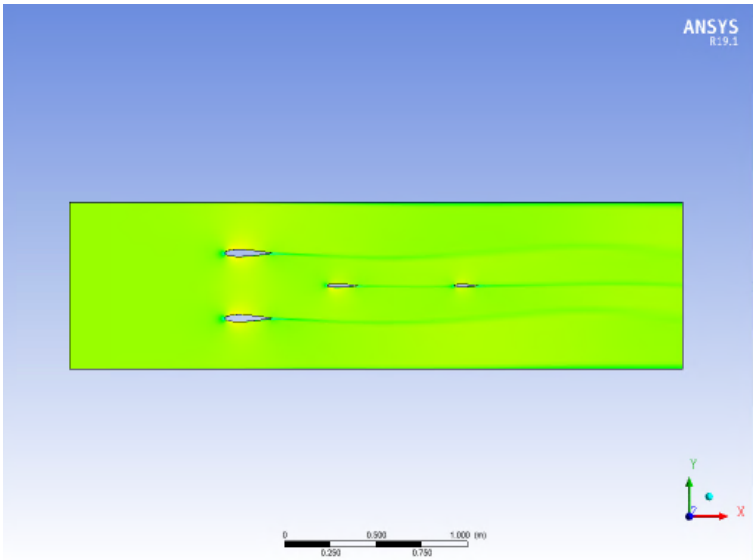

Figure 4.95: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 3 Degree, High k Oscillation at $t = T$

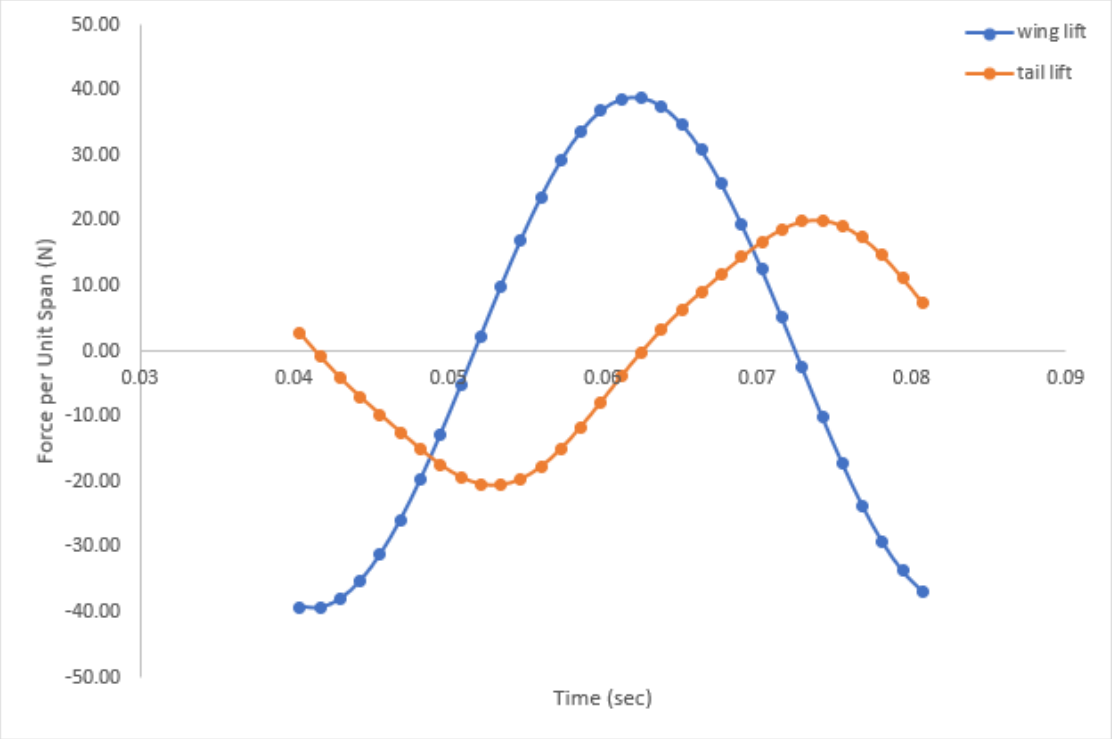

Figure 4.96: Unsteady Gust System-Test Object Interaction: Lift Forces on Wing and Tail for 3 Degree, High k Oscillation

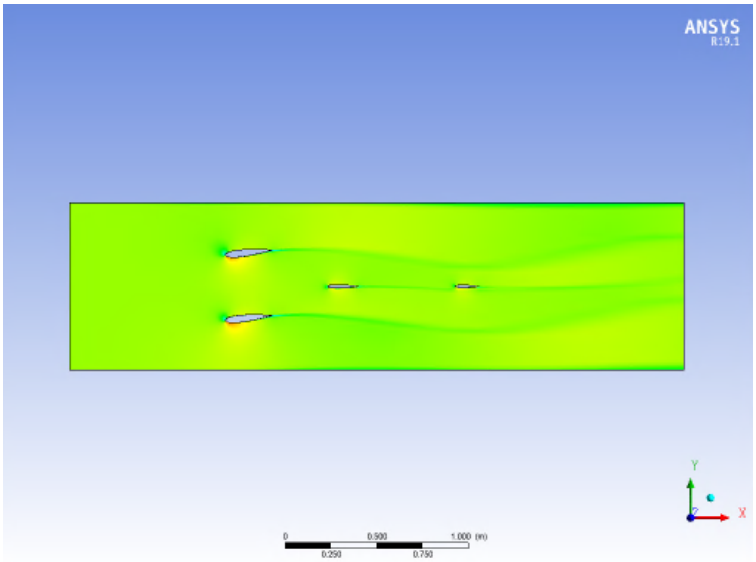

Figure 4.97: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 6 Degree, High k Oscillation at $t = 1/4T$

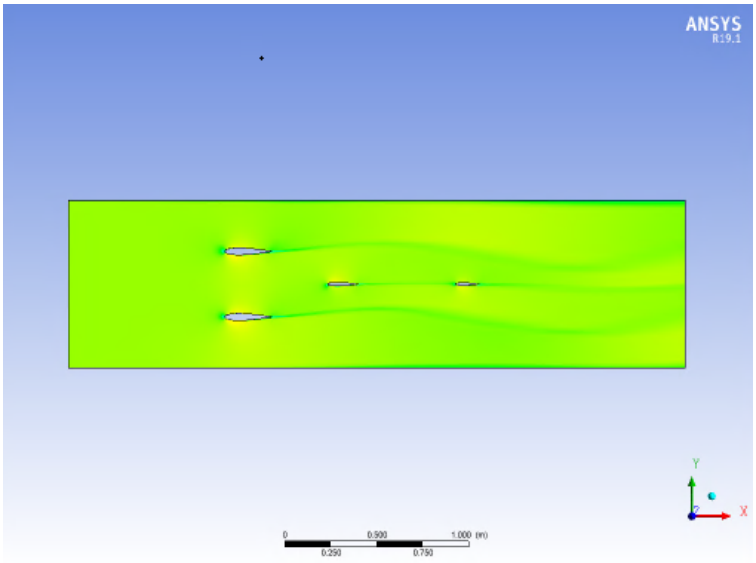

Figure 4.98: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 6 Degree, High k Oscillation at $t = 1/2T$

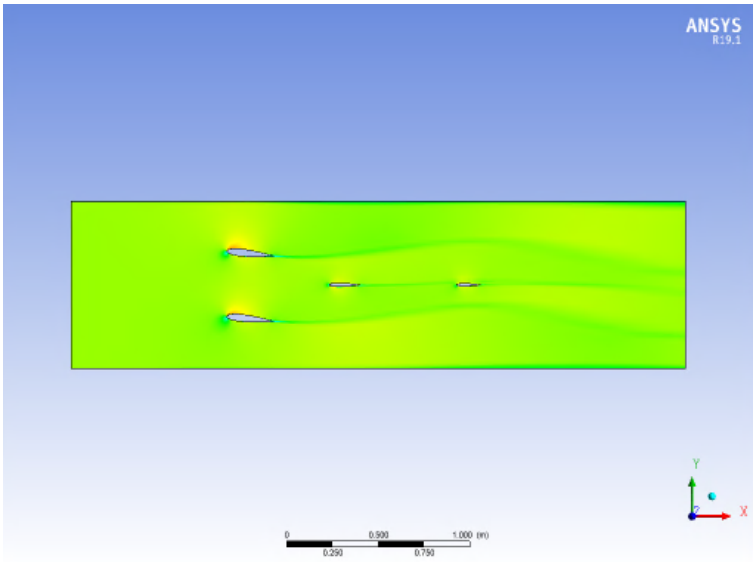

Figure 4.99: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 6 Degree, High k Oscillation at $t = 3/4T$

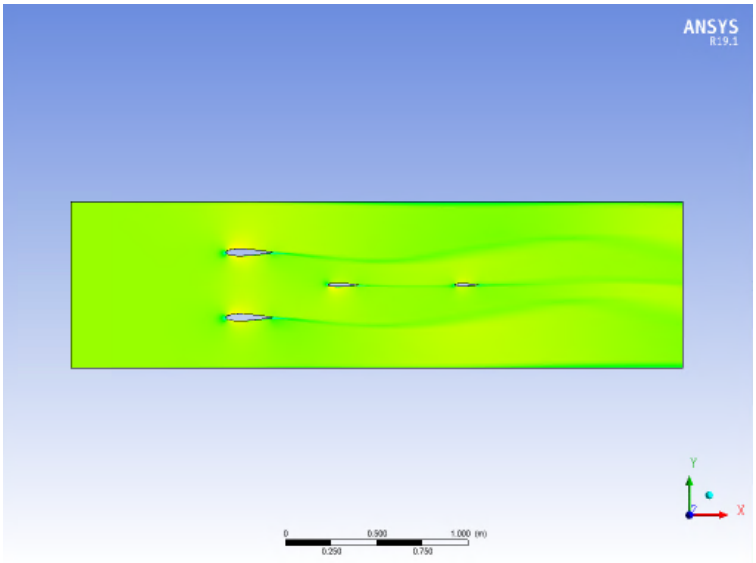

Figure 4.100: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 6 Degree, High k Oscillation at $t = T$

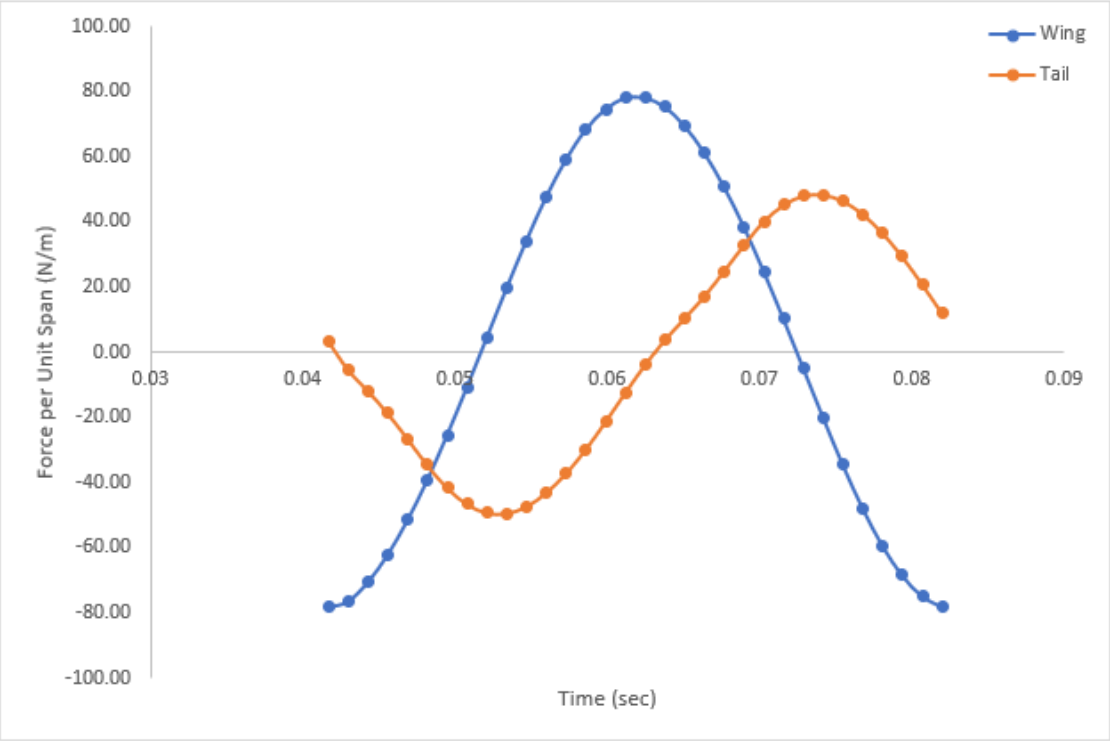

Figure 4.101: Unsteady Gust System-Test Object Interaction: Lift Forces on Wing and Tail for 6 Degree, High k Oscillation

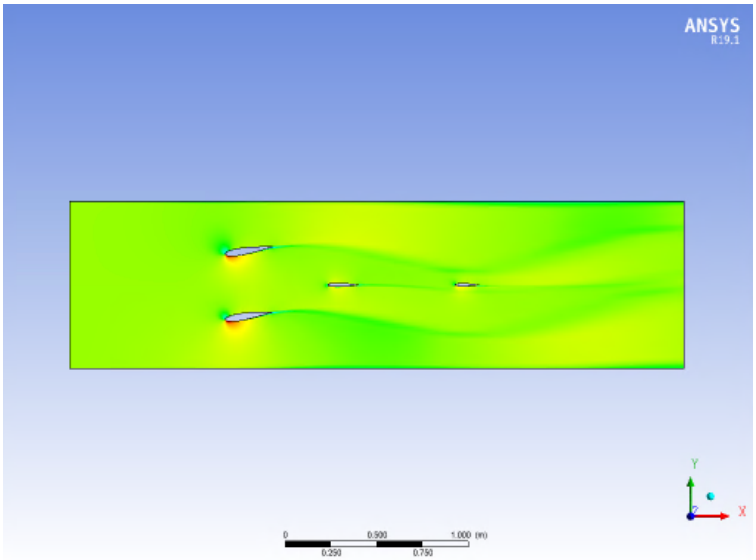

Figure 4.102: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 9 Degree, High k Oscillation at $t = 1/4T$

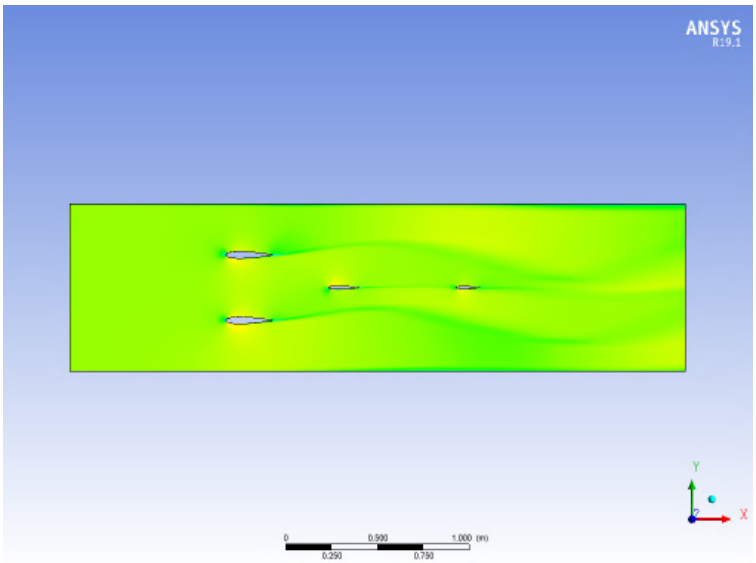

Figure 4.103: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 9 Degree, High k Oscillation at $t = 1/2T$

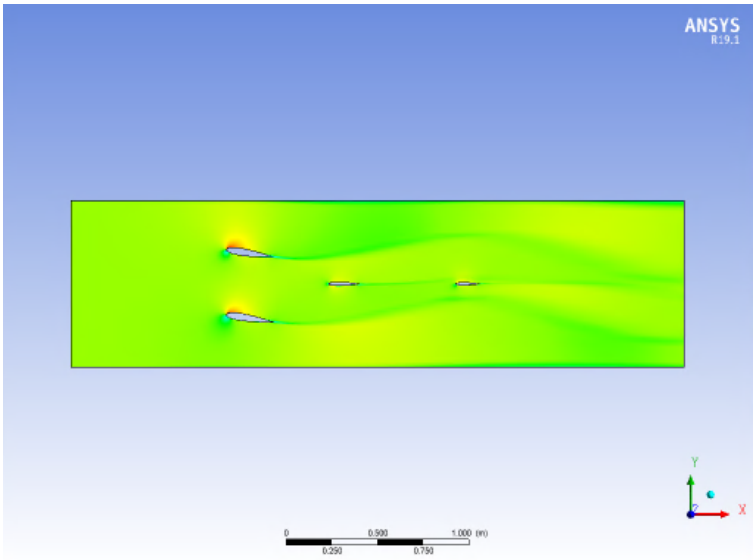

Figure 4.104: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 9 Degree, High k Oscillation at $t = 3/4T$

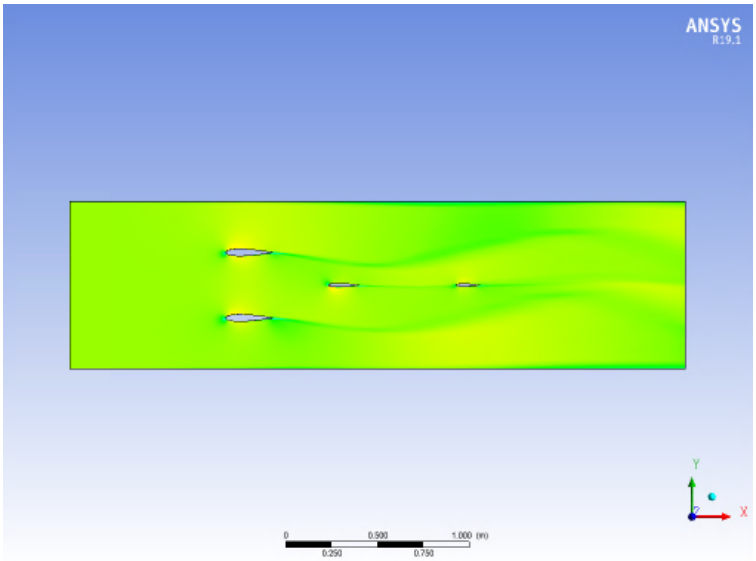

Figure 4.105: Unsteady Gust System-Test Object Interaction: Flow Snapshots for 9 Degree, High k Oscillation at $t = T$

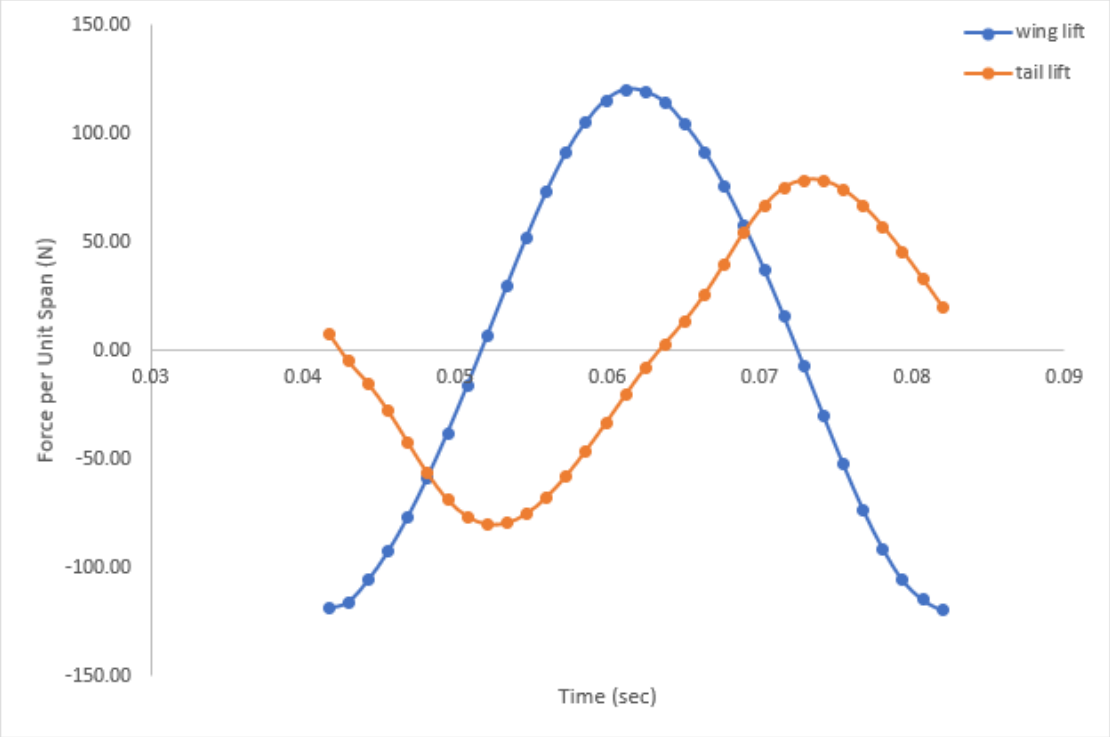

Figure 4.106: Unsteady Gust System-Test Object Interaction: Lift Forces on Wing and Tail for 9 Degree, High k Oscillation

The high-frequency simulations displayed similar results for all amplitudes of oscillation considered. The lifting force on the wing increased generally linearly with the increase in magnitude of oscillation at a rate of $\sim 13.3 N/deg$, like the trend in the low-frequency cases. However, the magnitude of the lifting forces on the wing in the high-frequency cases was about twice the magnitude of the lifting force on the wing in the low-frequency cases. Additionally, the lifting force experienced by the wing was approximately 90 degrees out of phases with the motion of the gust vanes, compared to the 180-degree phase difference for the low-frequency cases. The lifting force on the tail was significant in the high-frequency cases and differed dramatically from the low-frequency cases. The lifting force on the tail was approximately in-phase with the gust vane oscillation, a difference of about 90 degrees from the lift force experienced by the wing. The magnitude of the lifting force on the tail increased

generally linearly, in a similar manner as the lifting force on the wing, as the magnitude of oscillation increased at a rate of $\sim 10N/degree$. The differences in the lifting forces on the tail between the high-frequency and low-frequency cases are most likely due to the increased convection of gusting action into the wake. From previous simulations it is known that the higher frequency cases dissipate gusts at a further distance into the wake. The tail was, therefore, exposed to greater flow velocities and gust AOAs, which resulted in greater lifting forces. A similar effect can be seen on the wing. The elevated lifting forces on the wing for the high-frequency cases, when compared to the low-frequency cases, are most likely due to the increased velocity and gust AOA seen by the wing.

Chapter 5

THE MARCHING VORTEX LATTICE METHOD - THEORY

A Vortex Lattice Method (VLM) for simulating the unsteady behavior of an incompressible flow field in a low-speed wind tunnel was investigated as a potential supplement to or replacement, for fast simulations, of the computational analysis using CFD. The VLM used for this work was derived based on previous work by [13] and [9].

In a methodology laid out by [13] a single thin airfoil in a 1-D uniform freestream is modeled. The airfoil and the wake are partitioned into a number of elements (for 2D, "boxes" or "panels"). A point vortex is placed at the 1/4-chord of each element and a collocation point is placed at the 3/4-chord of each element on the airfoil. "At these [collocation] points, we require the velocity induced by the discrete vortices to equal the downwash arising from the unsteady motion of the airfoil" [13]. Aerodynamic boxes (or panels) also cover the wake behind the airfoil from the trailing edge (TE) to a large distance (in terms of chord lengths) behind the TE. There are M vortices on the airfoil in [13]. N is the total number of vortices. The rest of the vortices are in the wake. Downwash boundary conditions are imposed on the aerodynamic boxes on the airfoil.

$$w_i^{n+1} = \sum_{j=1}^{N} K_{ij}\Gamma_j^{n+1} \quad for \quad i = 1 : M \tag{5.1}$$

Where w_i^{n+1} is the downwash at the 3/4-chord of each element i at time step n+1, Γ_j^{n+1} is the circulation of each vortex at time step n+1, The function K_{ij} is the influence of vortex j on box i [13]. It is called called the kernel function and is given by:

$$K_{ij} = \frac{1}{2\pi R}\frac{x_j - x_i}{R} \tag{5.2}$$

Where x_j is the z-coordinate of the j-th vortex, x_i is the z-coordinate of the i-th collocation point and R is the distance between the respective vortex and collocation point. Conservation of circulation is imposed, in accordance with Helmholtz's theorem, in order to model the first vortex in the wake, which must reflect the difference between the total circulation around the airfoil in two consecutive time steps.

$$\Gamma_{M+1}^{n+1} = -\sum_{j=1}^{M}(\Gamma_j^{n+1} - \Gamma_j^n) \tag{5.3}$$

Finally, the vorticity in the wake is convected away from the airfoil in a stepwise manner, where the distance between wake elements is the distance covered by the flow over the time step used for the simulation.

$$\Gamma_i^{n+1} = \Gamma_{i-1}^n \quad for \quad i = M+2, N \tag{5.4}$$

The preceding equations are compiled into a matrix equation which is solved for the strength of all vortices as they evolve from time step to time step.

$$\mathbf{A\Gamma^{n+1}} + \mathbf{B\Gamma^n} = \mathbf{w^{n+1}} \tag{5.5}$$

$$\mathbf{\Gamma^{n+1}} = \mathbf{A^{-1}}[\ \mathbf{w^{n+1}} - \mathbf{B\Gamma^n}] \tag{5.6}$$

Where text in bold letters denotes matrices [13].

The method of [9], published earlier, is simplier than [13] in that it considers the entire airfoil as one element, with a single vortex at the 1/4-chord and a single downwash collocation point at the 3/4-chord of the airfoil. Reference [9] determines the lift and pitching moment on the airfoil for each time step.

$$N_j = N_{cj} + N_{aj} \tag{5.7}$$

where

$$N_{cj} = \rho U_\infty \Gamma_j \tag{5.8}$$

$$N_{aj} = \frac{1}{4}\rho\pi c^2(\ \ddot{h}_j + U_\infty \dot{\theta}_j) \tag{5.9}$$

and

$$M_j = \frac{1}{4} c N_{cj} + M_{aj} + M_{\dot{\theta} j} \tag{5.10}$$

where

$$M_{aj} = -\frac{1}{128}\rho\pi c^4 \ddot{\theta}_j \tag{5.11}$$

$$M_{\dot{\theta} j} = -\frac{1}{16}\rho\pi c^3 U_\infty \dot{\theta}_j \tag{5.12}$$

The matrix approach used by [13] was combined with the simplified, single element per airfoil approach used by [9] in the present work. One vortex was located at the airfoil's 1/4-chord with n vortices considered in the wake. The lift and moment data were calculated using the equations presented in [9].

The work of [13] and [9] was generalized to account for wind tunnel wall effects. The effect of the walls was modeled using mirror images of all vortices (airfoil-bound and those in the wake) in accordance with the method described by [32]. The mirroring attempts to enforce the no-downwash boundary condition at the walls along the length of the airfoil and the wake. In the present work in-section vortices at any location with respect to the walls are mirrored - first by the first wall, with subsequent mirrors of the mirrored vortices in the other wall, and so on, as well as mirror vortices first in the second wall, to be then mirrored by the first wall, and so on).

If we follow the work of [32] the wind tunnel and airfoil in the marching vortex model are considered in the x-z plane with the airfoil's 3/4-chord at the origin and the wake extending

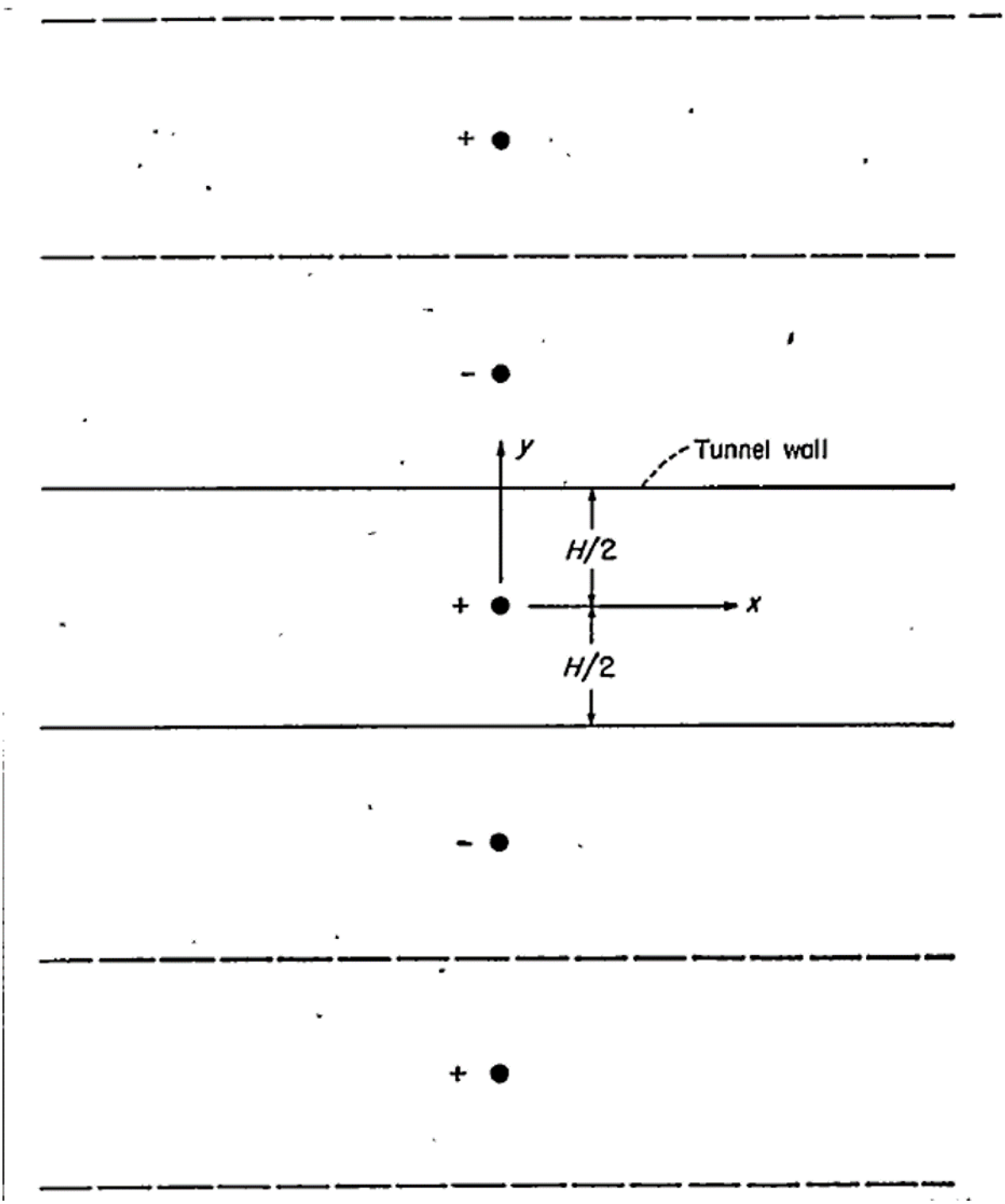

Figure 5.1: Mirroring Example from Runyan and Watkins for a Single Vortex at the Test Section Center Line

along the line $z = 0$ in the positive x-direction. Consider a vortex at the airfoil's 1/4-chord, at the coordinate $(x, z) = (-b, 0)$ where b is the half-chord of the airfoil. A vortex of equal magnitude but opposite sign, called the mirrored vortex, is placed above the top wall of the wind tunnel. The top wall is equidistant from the original vortex and the mirrored vortex with both vortices along the same $x = -b$ vertical line. The mirrored vortex is mirrored about the bottom wall in a similar manner, with another vortex created of equal magnitude but opposite sign of the mirrored vortex. The mirroring continues with vortices being dispersed further in the positive z-direction and negative z-direction along the same $x = -b$ line. The same method described for the top wall mirroring is repeated in a similar fashion for the bottom wall as well. The entire process explained above is also completed for each vortex in the wake.

For example, for a tunnel that is $2H$ wide (H from centerline to each wall where centerline is at $z = 0$) with an airfoil located $\overline{z}$ above centerline (at $z = \overline{z}$), the vortex z-locations would be as described in the table below, organized based on mirrored vortices starting about the top wall followed by mirrored vortices starting about the bottom wall. The mirrored vortices' x-coordinate would be the same as the original vortex's x-coordinate.

Figure 5.2: Vortex Location and Sign Example

z-Location	Vortex Sign Compared to Original Vortex
$2H - \overline{z}$	Negative
$-4H + \overline{z}$	Positive
$6H - \overline{z}$	Negative
$-8H + \overline{z}$	Positive
$-2H - \overline{z}$	Negative
$4H + \overline{z}$	Positive
$-6H - \overline{z}$	Negative
$8H + \overline{z}$	Positive

A simulation considering N vortices in the wake results in $(N+1)$x$(2*M+1)$ vortices, where M is the number of mirroring repetitions completed for a given vortex per wall. However, there are only N+1 distinct vortices. The strength of each mirrored vortex is the same magnitude as the strength of the preceding vortex, just with the opposite sign. Such a result is important as it is used to more efficiently consider a large number of mirrored vortices. As stated in [32], theoretically an infinite number of mirrors would be needed to perfectly simulate the no-downwash boundary condition on the walls. However, an infinite number of mirrors is impossible and impractical. A large number of mirrors was utilized instead.

The matrix approach of [13] was modified to account for the mirrored vortices. In [13]'s formulation, only one kernel function, which describes the distance of the vortex to the collocation point, is used per vortex. In the VLM with mirroring simulation, multiple kernel functions are applied to each in-section vortex because of the need to add the effects of all out of section mirrored (reflected) vortices, each with its (x,z)-location coordinates and correct sign. Since the magnitude of the vortex is constant for each vertical set of vortex-mirrored vortices combinations, the kernel functions were modified to account for distance from the vortex to the airfoil's collocation point and for the desired sign of the vortex. The kernel functions for each vertical set of vortices were summed together to return the equation to the original form described by [13] with [9]'s simplifications:

$$w^{n+1} = \sum_{j=1}^{N} \overline{K_j} \Gamma_j^{n+1} \tag{5.13}$$

where, with M reflections in each wall for each in-section vortex (on airfoil or wake):

$$K_{kj} = \frac{1}{2\pi R} \frac{x_j - x_i}{R} \tag{5.14}$$

$$\overline{K_j} = \sum_{k=1}^{2M+1} K_{kj} \tag{5.15}$$

Where x_j is the z-coordinate of the j-th vortex, x_i is the z-coordinate of the i-th collocation point and R is the distance between the respective vortex and collocation point. The strength

of the vortices was then solved for in the same manner as before:

$$\mathbf{A\Gamma^{n+1}} + \mathbf{B\Gamma^{n}} = \mathbf{w^{n+1}} \tag{5.16}$$

$$\mathbf{\Gamma^{n+1}} = \mathbf{A^{-1}}[\ \mathbf{w^{n+1}} - \mathbf{B\Gamma^{n}}] \tag{5.17}$$

The final capability added to the VLM simulation in order to model the gust generation system in the wind tunnel more accurately was the consideration of two airfoils in the VLM simulation of two gust vanes (while the VLM approaches discussed by [13] and [9] only considered a single airfoil). To allow studies of gust vanes and test wings in the wind tunnel, the method was extended to include multiple airfoils.

In the case of two airfoils the equations used to determine the strength of the vortices were modified as follows. The vortices of the two airfoils, airfoil A or the top airfoil (Γ_A) and airfoil B or the bottom airfoil (Γ_B), were concatenated, as well as the downwash on the two airfoils.

$$\left\{\Gamma\right\} = \begin{Bmatrix} \Gamma_A \\ \Gamma_B \end{Bmatrix} \tag{5.18}$$

$$\left\{w\right\} = \begin{Bmatrix} w_A \\ w_B \end{Bmatrix} \tag{5.19}$$

Kernel functions were used to relate the vortices produced by one airfoil to the same airfoil's downwash and to relate the vortices produced by one airfoil to the other airfoil's downwash. Specifically, kernels relating the vortices of airfoil A to the downwash on airfoil A ($K_{A\Rightarrow A}$), kernels relating the vortices of airfoil A to the downwash on airfoil B ($K_{A\Rightarrow B}$), kernels relating the vortices of airfoil B to the downwash on airfoil B ($K_{B\Rightarrow B}$), and kernels relating the vortices of airfoil B to the downwash on airfoil A ($K_{B\Rightarrow A}$) were defined. The A and B matrices were also modified and expanded to account for the cross-influence effect of the airfoils on each other.

$$\mathbf{A} = \begin{bmatrix} K_{AA_1} & K_{AA_2} & K_{AA_3} & K_{BA_1} & K_{BA_2} & K_{BA_3} \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ K_{AB_1} & K_{AB_2} & K_{AB_3} & K_{BB_1} & K_{BB_2} & K_{BB_3} \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.20}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.21}$$

For the case where two consecutive vortices in the wake were considered, variations of overall circulation in each airfoil propagated to the wake behind it. The time evolution of the strengths of the vortices on both airfoils were solved step by step in the same way as before using the modified matrices described above.

$$\mathbf{A\Gamma^{n+1}} + \mathbf{B\Gamma^{n}} = \mathbf{w^{n+1}} \tag{5.22}$$

$$\mathbf{\Gamma^{n+1}} = \mathbf{A^{-1}}[\ \mathbf{w^{n+1}} - \mathbf{B\Gamma^{n}}] \tag{5.23}$$

Finally, the mirroring capability was included in the double airfoil VLM simulation as well. The same methodology used to mirror a single airfoil was used to mirror two airfoils individually. The mirrored vortices' contributions were considered on the airfoil which created the vortex as well as the other airfoil using the technique described above.

Chapter 6

THE TIME MARCHING VORTEX LATTICE METHOD - RESULTS

Time marching VLM simulations with and without the effect of walls were carried out. The VLM simulation involving no wall effects was compared to the known Theodorsen solution for validation. The VLM simulation involving wall effects was compared to the Theodorsen solution as well. The effectiveness of the mirrored vortices in enforcing the no-downwash boundary condition along the walls was evaluated. For both cases, the sensitivity of the simulation to changes in various parameters was considered. All simulations were completed at 60 m/s using a thin airfoil with a chord length of 10 in.

6.1 VLM Simulation without Mirroring

The time marching VLM simulation was compared the Theodorsen solution to validate the simulation in the no-walls case. A frequency of 1 Hz and a maximum amplitude of 5 deg with 200 time steps per period (at at speed of 60 m/sec this corresponds to a reduced frequency of 0.0133).

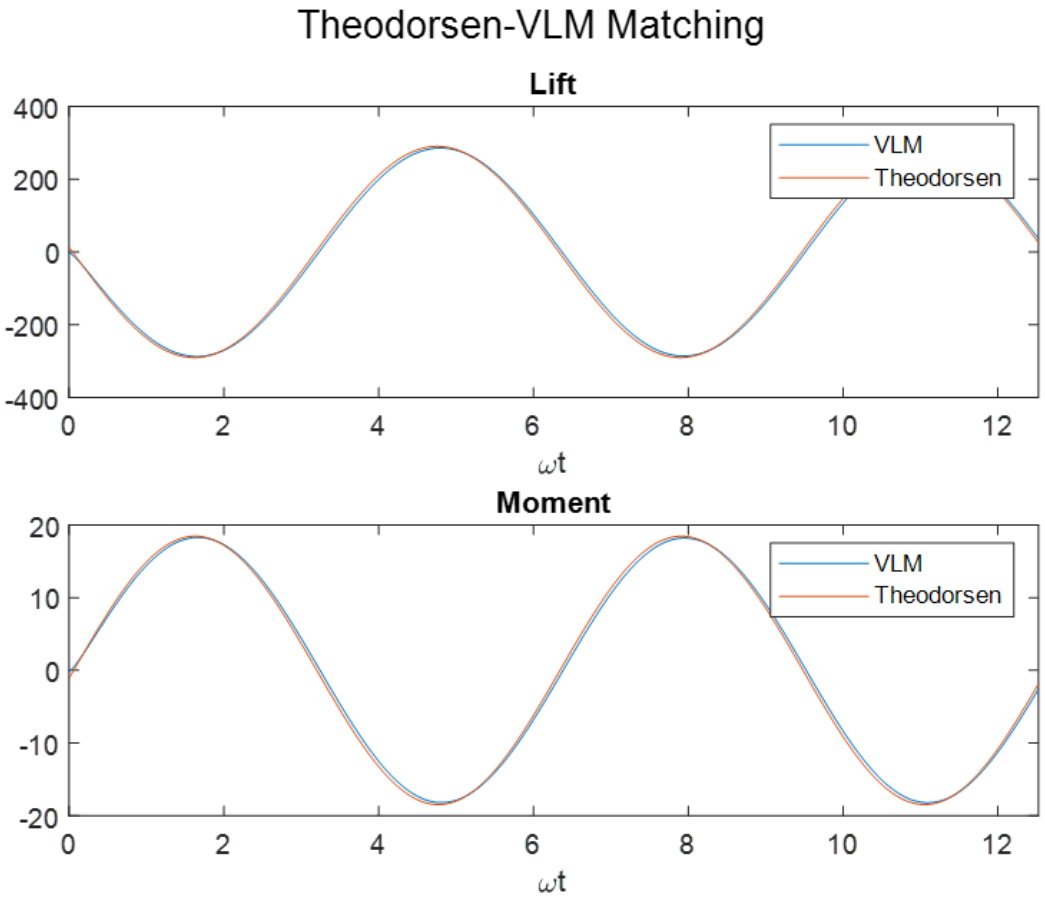

Figure 6.1: Marching VLM Single-Airfoil Simulation vs. the Theodorsen solution - No Walls

As the figure shows, the agreement between the Theodorsen and marching VLM solutions is excellent for both lift and pitching moment time histories.

Additional studies examined the sensitivity of the simulation to changes in parameters such as the frequency of oscillation and the number of time steps per period.

The time stepping and wake length was determined based on the reduced frequency. For an airfoil, oscillating at some frequency, f Hz, corresponds to radial frequency $\omega = 2\pi f$ rad/sec. To capture the dynamic behavior by time stepping, use a stepping interval of $\Delta t = \frac{T}{N_T}$, where T is the period and N_T is the number of time steps per period.

Thus:

$$\Delta t = \frac{T}{N_T} = \frac{2\pi}{\omega \cdot N_T} \tag{6.1}$$

With N_{wake} vortices in the wake, and with the distance between consecutive vortices being $U \cdot \Delta t$, the length of the wake behind the airfoil is:

$$N_{wake} \cdot U \cdot \Delta t = N_{wake} \cdot U \cdot \frac{2\pi}{\omega \cdot N_T} \tag{6.2}$$

Normalizing the length of the wake by the chord of the wing $c = 2b$ leads to:

$$\frac{N_{wake} \cdot U \cdot \Delta t}{2b} = \pi \frac{N_{wake}}{N_T} \cdot \frac{U}{\omega \cdot b} = \pi \frac{N_{wake}}{N_T} \cdot \frac{1}{k} \tag{6.3}$$

Where k is the reduced frequency of oscillation.

For time marching simulations to be consistent, the number of vortices in the wake should be selected to keep the normalized wake length (wake length / chord) the same for the different reduced frequencies studied.

The sensitivity of the simulation to changes in reduced frequency, and subsequently step size and wake length, was investigated.

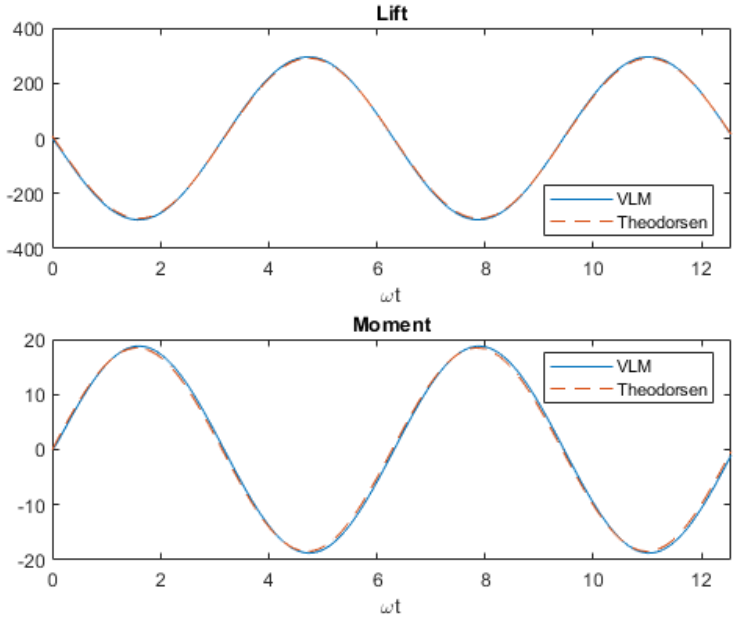

Figure 6.2: The Effect of the Oscillation Frequency: $k = 0.013$

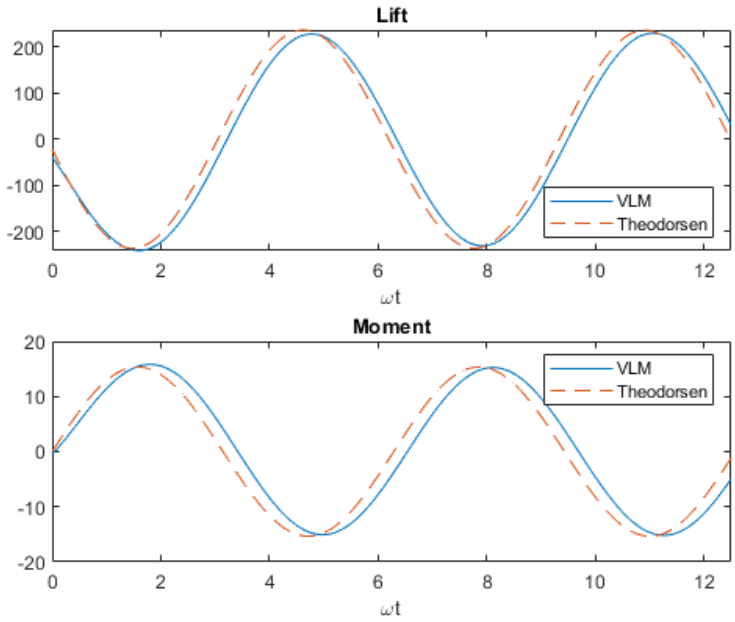

Figure 6.3: The Effect of the Oscillation Frequency: $k = 0.16$

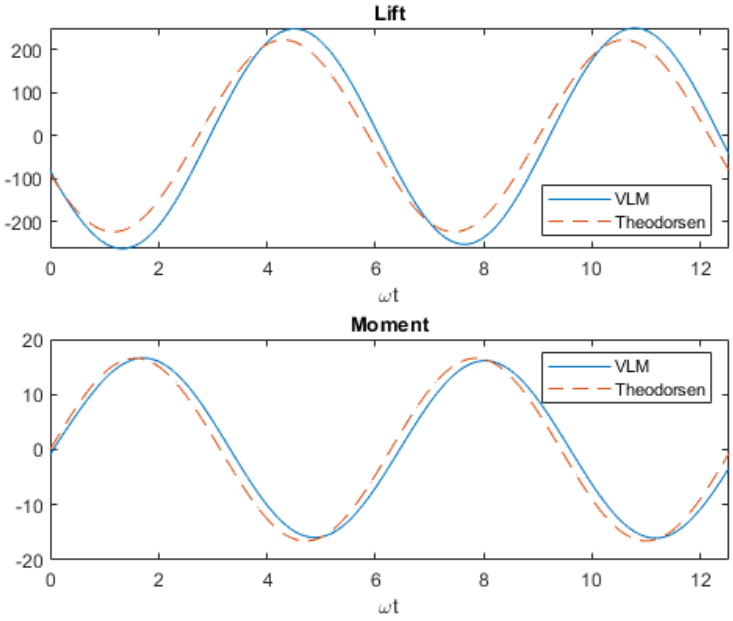

Figure 6.4: The Effect of the Oscillation Frequency: $k = 0.32$

The results shown in the figure above correspond to reduced frequencies of $k = 0.013$, $k = 0.16$, and $k = 0.32$. The main effect of the increasing frequency on the VLM predictions is a phase lag introduced into the prediction. As the frequency goes up, the phase lag seems to increase. For the high frequency case, the amplitude also seems to increase slightly compared to the Theodorsen solution. Some consideration may need to be taken for VLM simulations at higher frequencies. However, at low frequencies, the simulation behaves quite nicely.

A single frequency case, $f = 1Hz$, corresponding to a reduced frequency of $k = 0.013$, was analyzed for various number of time steps per period as well. As explained previously, step length and wake length are functions of oscillation frequency. However, step length, and subsequently wake length, can also be effected by changes to the number of time steps per period, N_T. Cases with $N_T = 20$, $N_T = 50$, and $N_T = 200$ were considered.

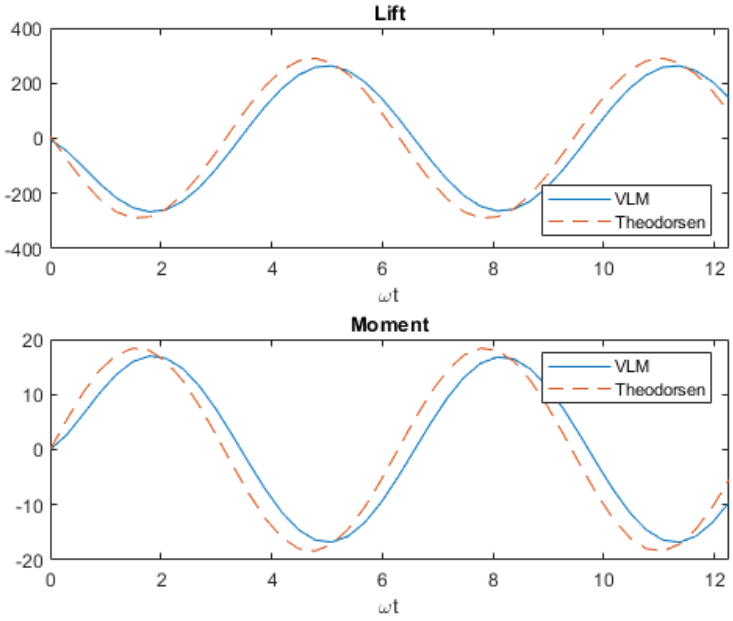

Figure 6.5: The Effect of the Number of Timesteps per Period: $N_T = 20$

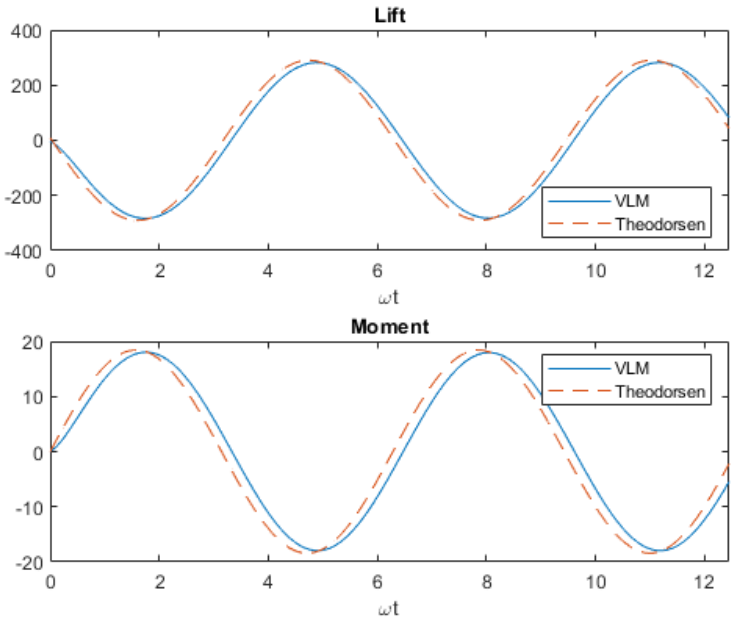

Figure 6.6: The Effect of the Number of Timesteps per Period: $N_T = 50$

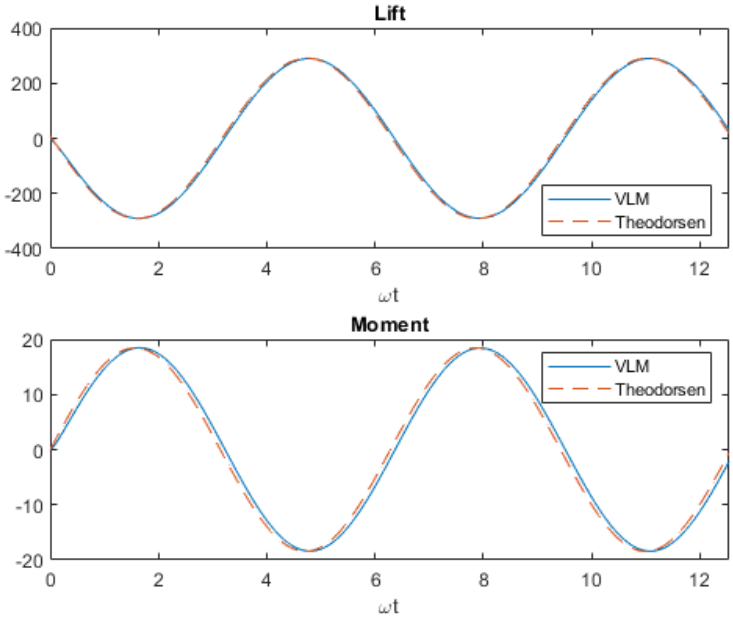

Figure 6.7: The Effect of the Number of Timesteps per Period: $N_T = 200$

The VLM simulations show a slight sensitivity to the number of time steps per period. As the number of time steps per period is decreased, the VLM simulation begins to underestimate and lag the Theodorsen solution slightly. However, the magnitude of both trends is small. Since the VLM code is computationally simple, it is relatively cheap in terms of time and computational energy to increase the number of time steps per period to a sufficient number.

6.2 Marching VLM Simulations with Walls and Mirroring

The VLM simulation with mirroring was evaluated next. The relationship of the wind tunnel simulation and the no-walls Theodorsen solution was studied and the impact of the proximity of the walls to the airfoil was analyzed. Also, the effectiveness of the mirroring in enforcing the no-downwash boundary condition along the wall was studied for various numbers of mirrored vortices. The sensitivity of the simulation to changes in parameters, specifically

number of vortices in wake and number of mirrored vortices was considered. Unless noted, the walls were 1.5 ft from the airfoil, in the case of a single airfoil at the center of the test section, in accordance with the dimensions of the UW 3x3 tunnel. The gust angle-of-attack (AOA) along the walls was also tracked for the baseline case to ensure the simulation sufficiently satisfied the no-downwash boundary condition. 10 mirrorred vortices for each wall were modeled in the baseline case.

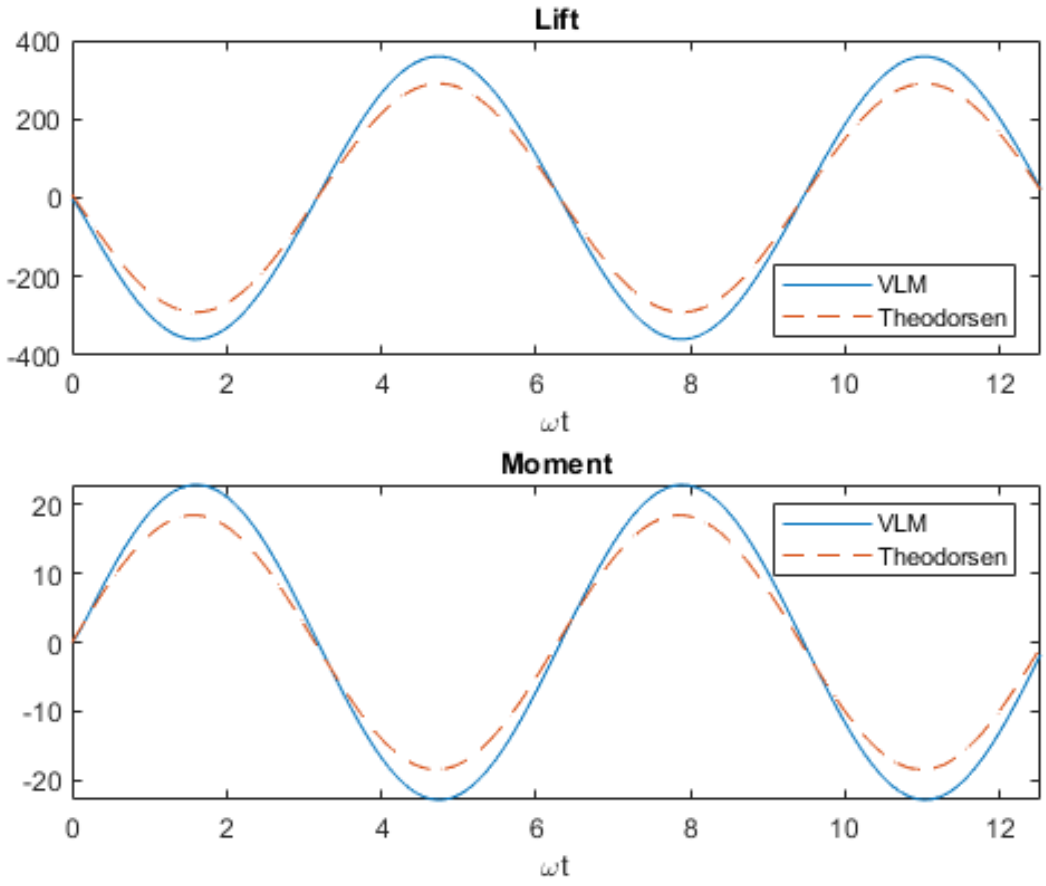

Figure 6.8: VLM Single-Airfoil Simulation Theodorsen vs. with Airfoil in Wind Tunnel

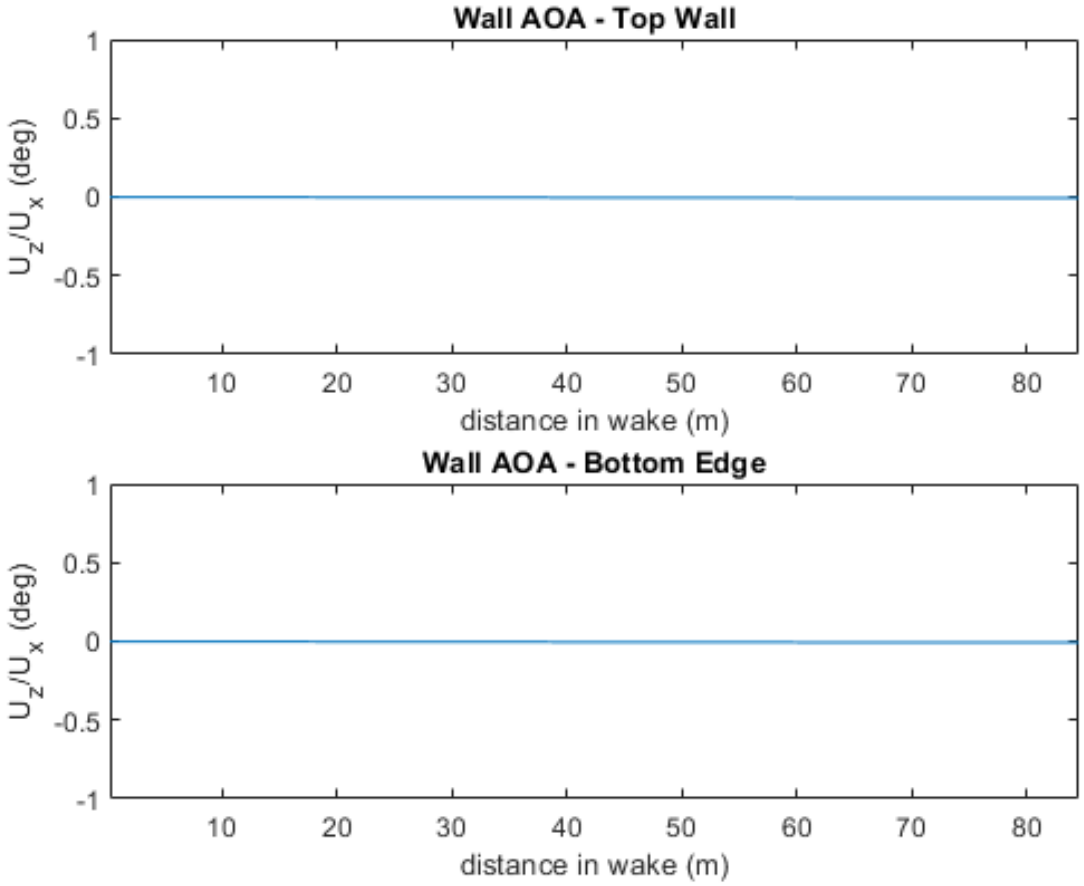

Figure 6.9: VLM Single-Airfoil Simulation Theodorsen vs. with Airfoil in Wind Tunnel: Wall AOA

The baseline case shows the influence of the walls on the lift and moment of the airfoil. The lift and moment determined by the VLM simulation are greater in magnitude than the lift and moment predicted by the Theodorsen solution. The VLM results are still in phase with the Theodorsen solution. Such a result is expected, as the walls were shown previously in CFD simulations to increase the lift on the airfoil. The wall AOA was also shown to be negligible for the length of the wake. Such a result confirms the effectiveness of the mirroring method in enforcing the correct boundary condition along the walls.

Moving the walls further and further away from the airfoil leads to the results in the

following figures:

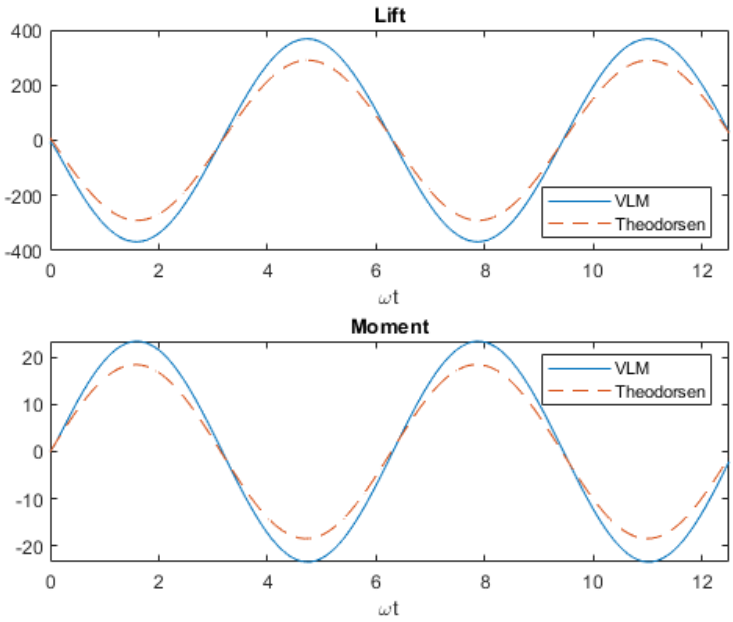

Figure 6.10: Effect of Walls on VLM Simulation with Mirroring: $H = 1.5ft$

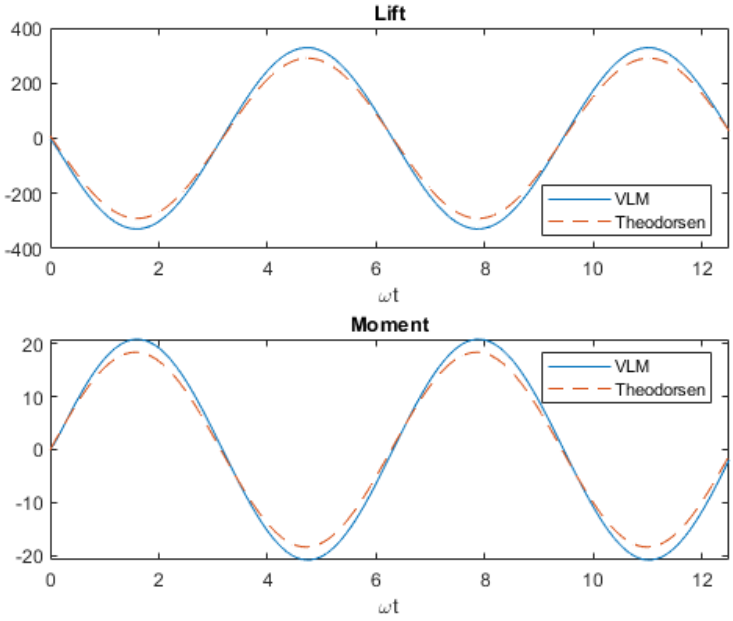

Figure 6.11: Effect of Walls on VLM Simulation with Mirroring: $H = 3ft$

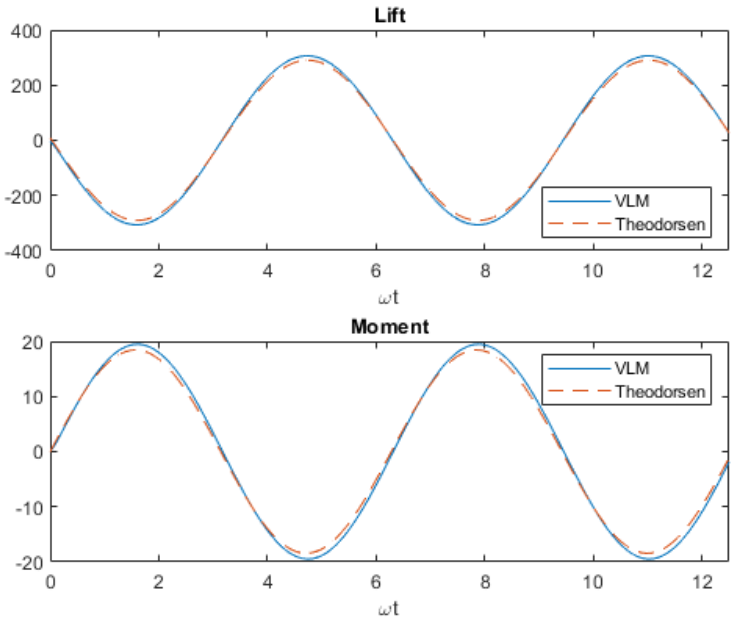

Figure 6.12: Effect of Walls on VLM Simulation with Mirroring: $H = 9ft$

The impact of the walls on the airfoil's lift and moment is evident in the VLM simulations. The closer the walls are to the airfoil, the greater the lift force experienced by the airfoil. Such results concur with the CFD results presented previously.

6.3 Boundary Condition Enforcement

The effectiveness of the airfoil and wake vortices mirroring in enforcing the desired boundary conditions along the walls of the wind tunnel is discussed in this section. The mirroring simulations were compared to the non-mirroring simulation for reference. The number of mirrored reflections was modified to determine the required number of mirrored vortices required for successful wall boundary condition enforcement. The angle of flow along the walls (AOA) was studied as function of position along walls and the number of mirror reflections used. The wall AOA was determined by dividing the vertical velocity (v) by the horizontal velocity (u) at various locations along the wall. Note the small scale for the gust AOA; even for no mirrored vortices or for a small number of mirrored vortices, the wall AOA is small.

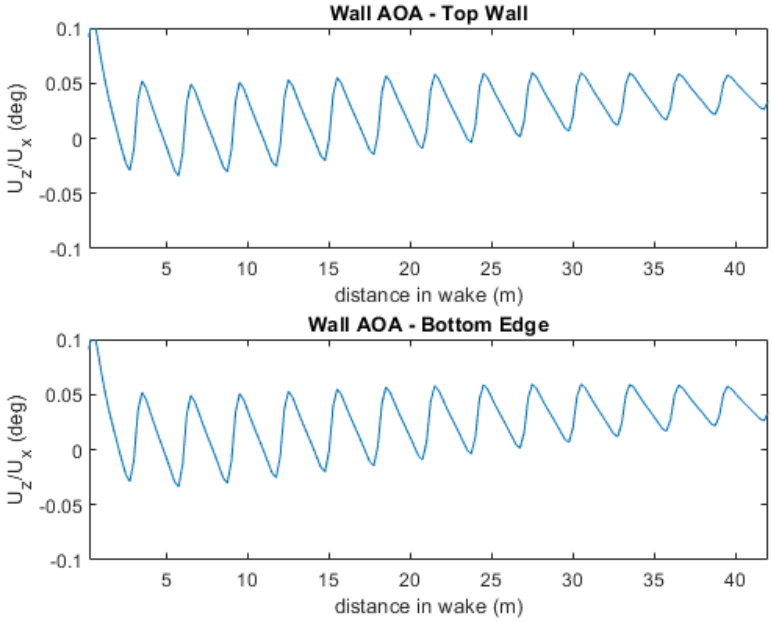

Figure 6.13: Wall Flow Angle for Various Numbers of Mirrors: No Mirrors

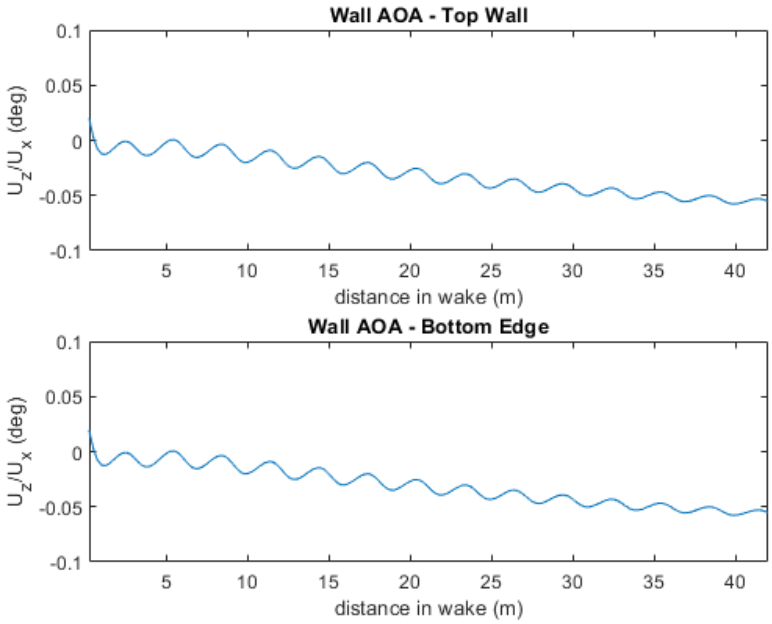

Figure 6.14: Wall Flow Angle for Various Numbers of Mirrors: $m = 1$

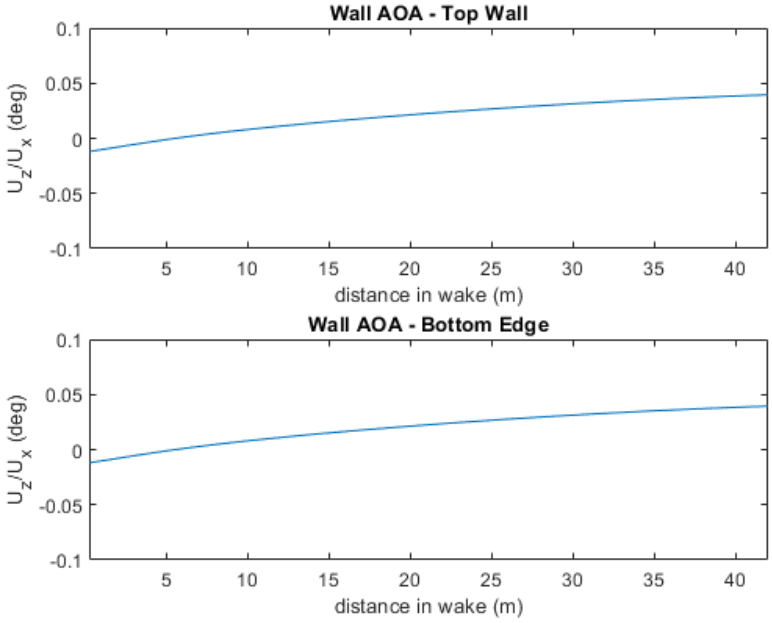

Figure 6.15: Wall Flow Angle for Various Numbers of Mirrors: $m = 10$

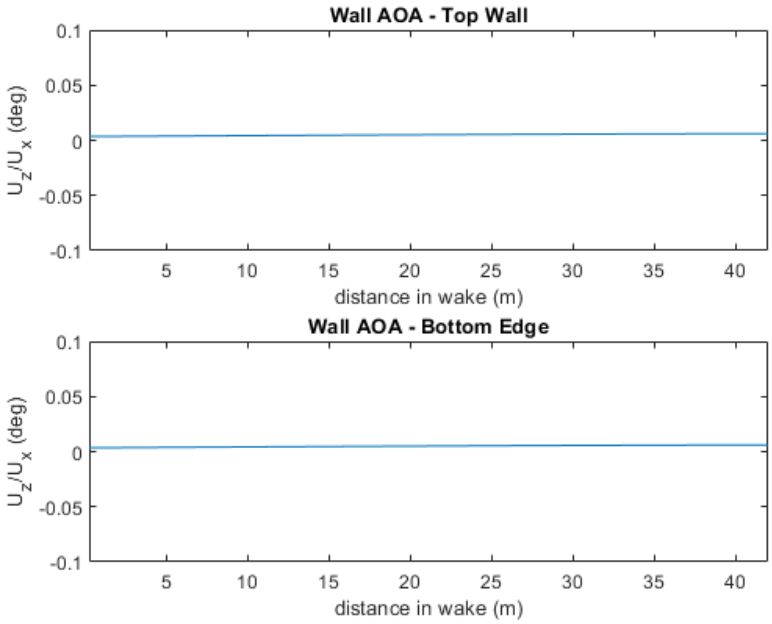

Figure 6.16: Wall Flow Angle for Various Numbers of Mirrors: $m = 100$

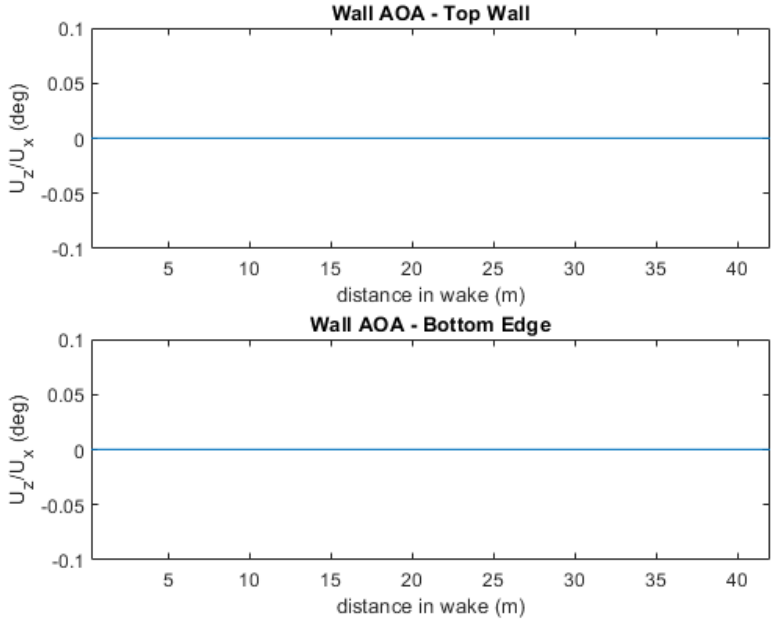

Figure 6.17: Wall Flow Angle for Various Numbers of Mirrors: $m = 500$

The wall AOA plots show the effectiveness of the mirroring technique. For the no-mirror case and the case involving only one set of mirrored vortices, the gust AOA is small but definitive. There is also clear oscillation in the gust AOA along the wall. For a small number of mirrored vortices ($m = 10$), the wall AOA shows slight deviation from zero. The boundary condition is not exactly enforced along large stretches of the walls but it is close to complete enforcement. Also, the oscillatory behavior is removed. As the number of mirrored vortices increases, the wall AOA decreases. For the 100 and 500 mirrored vortices cases, there is little to no discernable wall AOA. It appears the boundary condition is enforced across the entire domain. Such results indicate a larger number of vortices may more effectively enforce the boundary condition, as is expected. Nevertheless, the method is reasonably effective even for small numbers of vortices.

6.4 VLM Simulations with Mirroring - Sensitivity to Changes in Parameters

The sensitivity of the mirroring VLM simulation to changes in the number of mirrored vortices was investigated.

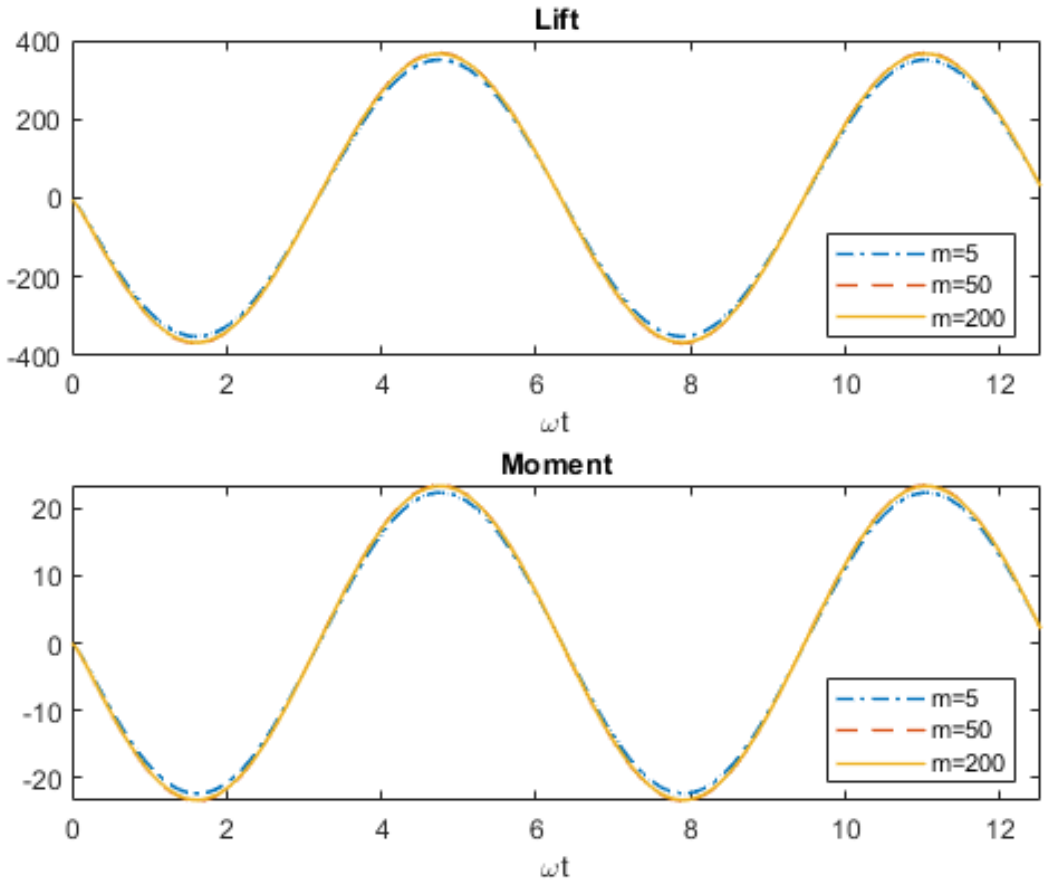

Figure 6.18: VLM Simulations with Mirroring: Sensitivity of Lift and Momemnt to Number of Mirrored Vortices

The lift and moment curves produced for various numbers of mirrored vortices appear largely the same. There does appear to be a slight difference in the magnitude of the maximum lift and moment experienced by the airfoil between the 5 mirrored vortices case and the other two cases. The 5 mirrored vortices case displays a lower maximum lift and

moment than the other two. Such results suggest a simulation using an insufficient number of mirrored vortices may fail to accurately simulate the added lift caused by the effect of the walls on the airfoil, even if the boundary condition is sufficiently enforced.

Chapter 7

MARCHING VORTEX LATTICE METHOD VS. CFD RESULTS - A COMPARISON

The marching VLM simulations were compared to the CFD simulations in order to assess the accuracy of the VLM in predicting the flow field created by the gust generation system. Single-vane and double-vane VLM simulations were compared to CFD single-vane simulations and double-vane simulations, respectively. The VLM simulation abilities to match the forces on the airfoils in the presence of the wind tunnel walls was evaluated. All simulations involved 100 mirrored vortices to simulate the effect of the tunnel walls, which were both located 1.5 ft from centerline in opposite directions. Simulations involved two frequencies of oscillation: 1 Hz (corresponding to a reduced frequency of 0.013) and 12 Hz (corresponding to a reduced frequency of 0.16), and an amplitude of oscillation of 6 deg.

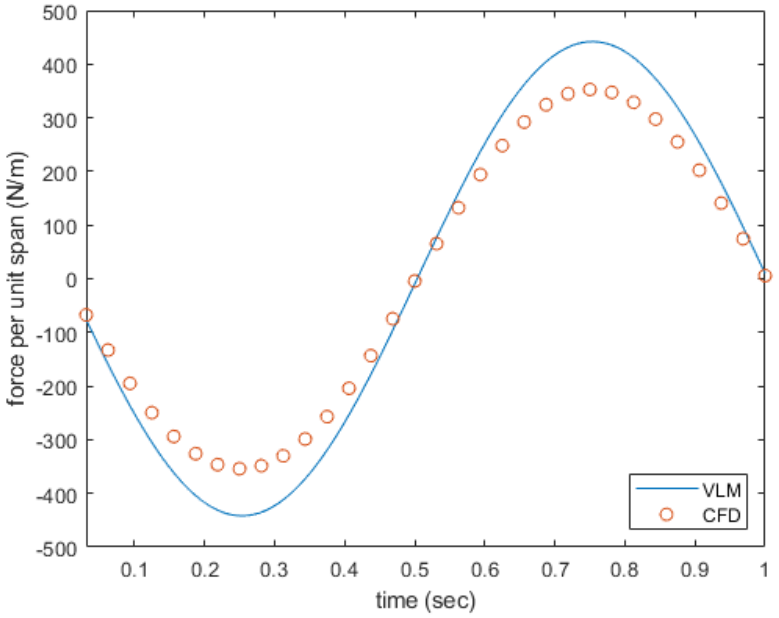

Figure 7.1: VLM-CFD Single Vane Comparison: $f = 1Hz$

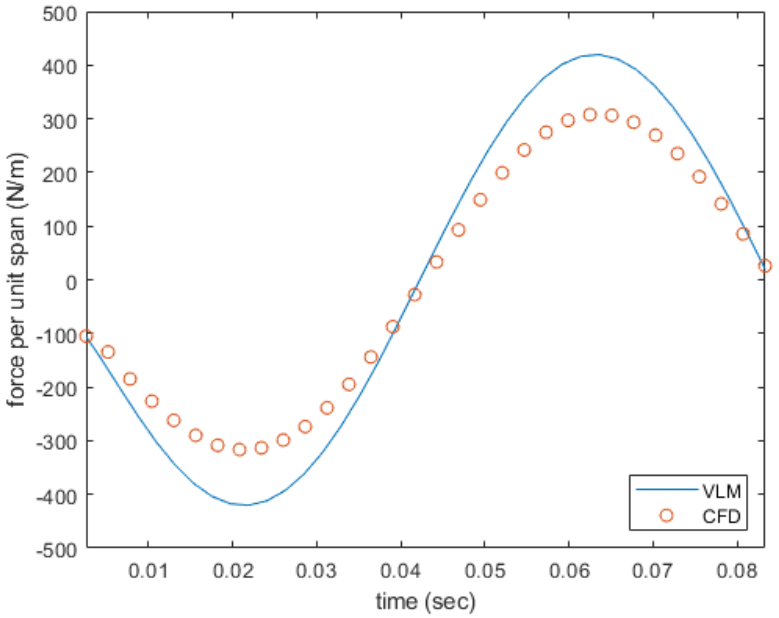

Figure 7.2: VLM-CFD Single Vane Comparison: $f = 12Hz$

The VLM simulations successfully matched the phase of the CFD solutions for each considered frequency. However, the amplitude of the VLM simulations was significantly greater than the amplitude of the CFD simulations. The VLM maximum lift force was approximately 28 percent greater than the CFD maximum lift force for both the $k = 0.013$ case and the $k = 0.16$ case. The results above suggested the VLM simulations were effective in matching the phase of the problem but overestimated the effect of the walls in increasing the lift on the airfoil.

7.1 Double-Vane Comparison

Two double-vane VLM simulations involving different reduced frequencies were compared to the double-vane CFD simulations.

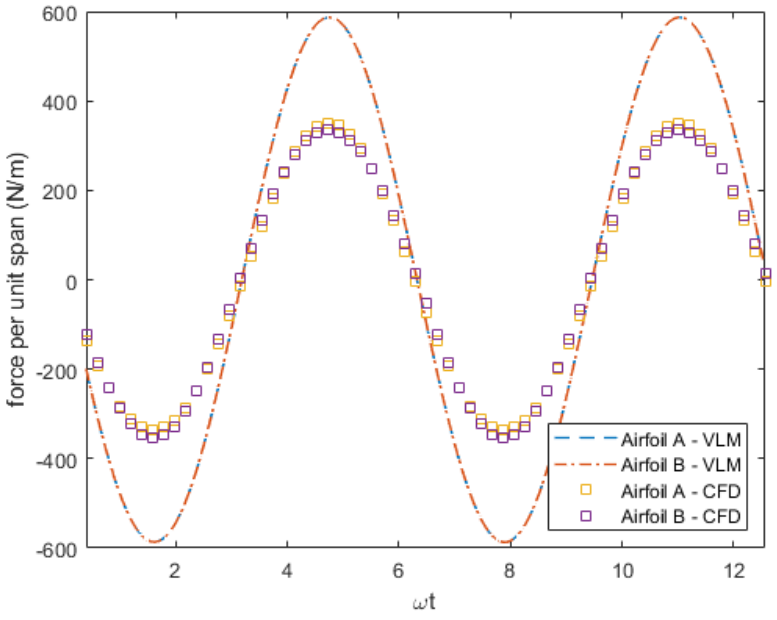

Figure 7.3: VLM-CFD Double Vane Comparison: $f = 1Hz$

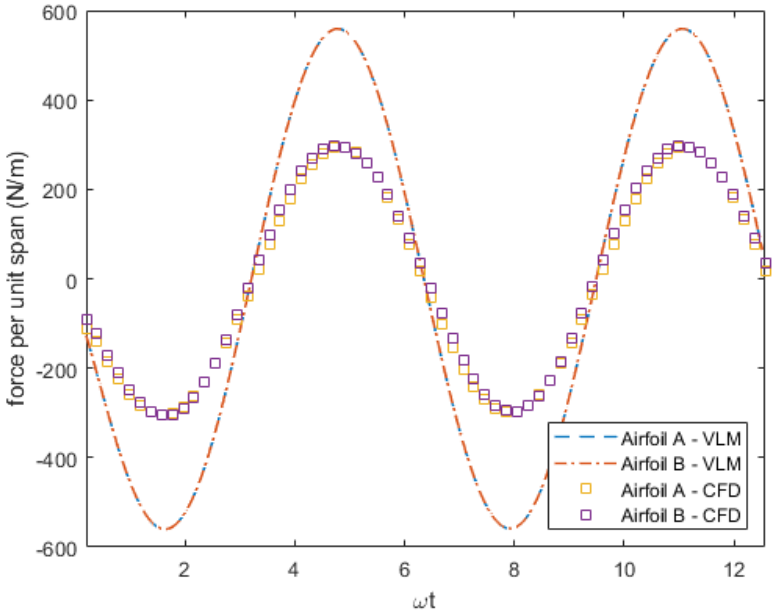

Figure 7.4: VLM-CFD Double Vane Comparison: $f = 12Hz$

The double-vane VLM simulations showed similar results as the corresponding single-vane VLM simulations when compared against CFD predictions. The VLM simulations accurately matched the phase of the CFD predictions. However, like with the single-vane simulations, the double-vane simulations demonstrated an overestimate in amplitude when compared to the CFD predictions. For both the $k = 0.013$ case and the $k = 0.16$ case, the VLM simulation predicted about 71 percent greater lift on both vanes than the CFD simulation predicted. Furthermore, the VLM simulations varied from the CFD simulations in that the VLM simulations showed no variance in force between the two airfoils at any part of the simulation. The CFD simulation showed differences in the forces felt by the airfoils throughout the oscillation. Such a result suggests the VLM simulations do not fully capture the physics of the mirroring situation at hand. Although the walls are being accounted for in the VLM simulation, as evidenced by the elevated lift when compared to the no-walls case, the full impact of the walls may not be accurately modeled.

7.2 Comparison between Walls and No-Walls Cases

Additional information about the accuracy of the VLM method in predicting the wind tunnel flow field can be gained from comparing the both VLM predictions and the CFD predictions for the respective walls and no walls cases. For the single-vane cases at the frequency, $f = 1Hz$, corresponding to a reduced frequency of $k = 0.013$, with an oscillatory amplitude of 5 deg, four strings of data were considered: VLM simulation with walls, VLM simulation without walls, CFD simulations with walls, CFD simulation without walls.

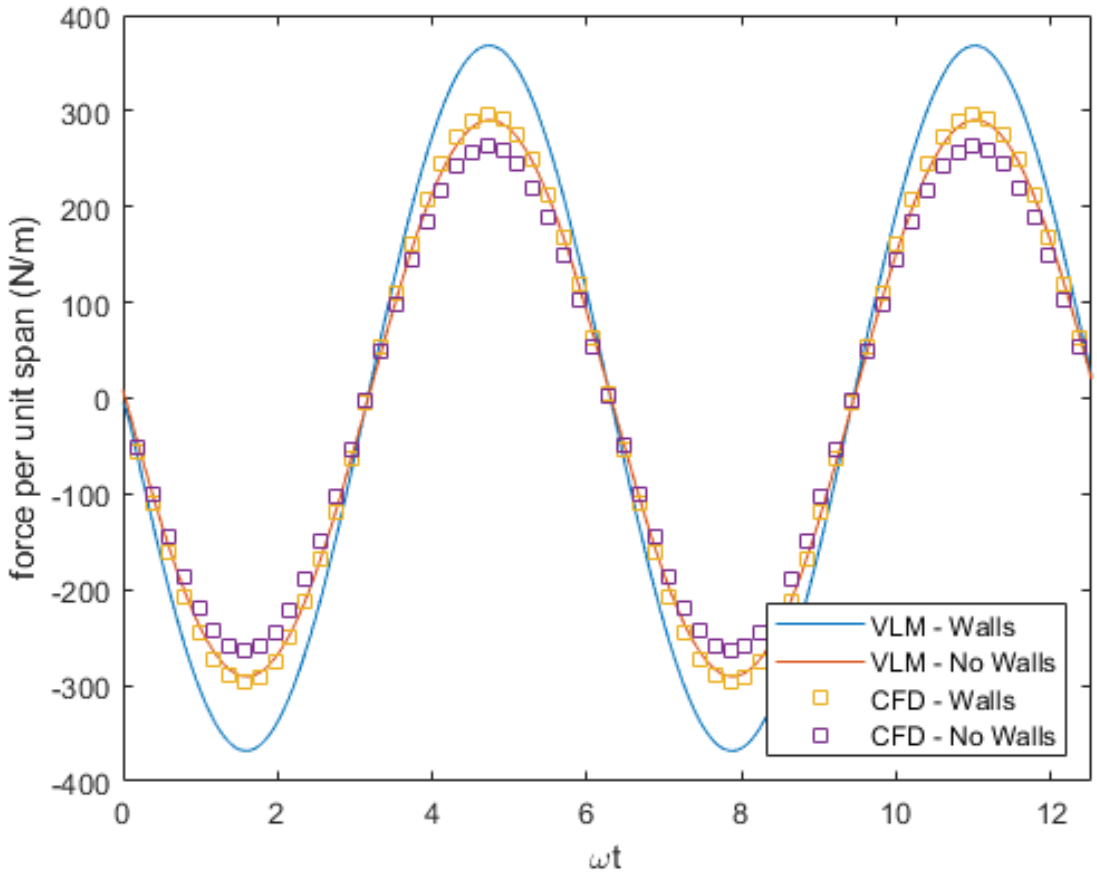

Figure 7.5: Comparison between Walls and No-Walls Cases

The walls and no-walls comparison of the two methods, VLM and CFD, show the VLM

mirroring method may be more accurate than it originally appeared. The maximum lift predicted in the VLM walls case was approximately 25 percent greater than the maximum lift predicted in the VLM no-walls case. The maximum lift predicted in the CFD walls case was approximately 13 percent greater than the maximum lift predicted in the CFD no-walls case. Such results indicate the approximately 28 percent difference between the VLM walls case and the CFD walls case is not a "one-to-one" comparison. The difference between the no-walls case and the walls case must be considered for each method individually before directly comparing the two methods. The variance in the differences between the CFD walls/no-walls comparison and the VLM walls/no-walls comparison shows the VLM simulation does not fully model that effect of the walls on the simulated flow. However, the variance in the differences between the two methods' individual comparisons show the mirroring method more effectively models the effect of the walls on the flow field than previously thought.

Chapter 8

CONCLUSION

The work described here focused on the numerical modeling of wind-tunnel gust generation systems, the flow fields they create, and their interactions with wind tunnel model wings placed behind them. A CFD-based simulation capability was developed. It was used to investigate the effects of wind tunnel walls, reduced frequency, gust vane oscillation amplitudes, and location of wing models in the test section on the gust vane and wing forces and on the resulting flow fields. The studies showed that at low reduced frequencies the flow field and forces were close to the ones predicted by steady analysis. New insights were gained regarding the effect of the distance between vanes, models, and the wind tunnel test section walls.

The work included development of a simple time marching vortex lattice method. In this modeling technique a single vortex is placed on a vane or an airfoil to capture changes in circulation around it, with a wake into which vortices are shed when circulation around airfoils varies with time, and those wake vortices propagate downstream at the speed of the incoming flow. The marching VLM model captured the no-walls case quite well, as shown by good correlation with the well-known Theodorsen solution and CFD solutions.

A mirroring technique was added to the VLM model to impose parallel flow along the walls of the test section. Its performance, regarding the modeling of the in-section wakes and all reflected mirror images was studied in an effort to develop modeling guidelines that would make this simple and fast modeling technique accurate. For cases where mirroring was used to simulate the effect of the walls on the airfoil, the VLM cases overestimated the effect of the walls on the airfoil when compared to the CFD predictions. For the single-vane case, the VLM simulations overshot the CFD ones by about 28 percent. In the double-vane

case, the forces on the vanes predicted by VLM are higher than the CFD ones by about 71 percent.

The VLM simulations were successful, though, in enforcing the no-downwash boundary condition along the walls and in matching the phase of the CFD simulations. Experimenting with time step size and number of wake vortices showed that there was sensitivity in matching the wall boundary conditions and in the predicted forces to variations in such parameters. More studies of the implementation of the VLM approach to this problem are required. The VLM simulations showed the potential of vortex methods in predicting 2D wind tunnel flow fields and such methods can be useful for predicting flow fields for gust vane design and gust vane / wind tunnel model interactions due to their relative speed and simplicity, with some correction for the magnitude of difference between CFD and VLM results. More work is still needed to refine the VLM so it will provide more accurate and actionable information.

There are multiple avenues for additional study on the present project. Both the VLM and CFD methods would benefit from experimental data for validation and comparison. Additionally, a move to include 3D effects would be helpful in furthering the realism of the simulations and would enable consideration of a greater number of models and situations.

BIBLIOGRAPHY

[1] W. E. A. Acum. *A Note on the Estimation of the Effect of Wind Tunnel Walls on the Forces on Slowly Oscillating Slender Wings*. 1965.

[2] ANSYS. Cfx steady theory guide, 2018.

[3] Yogesh Babbar, Vishvas S. Suryakumar, and Thomas W. Strganac. Experiments in aeroelastic response and control under gust. *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, May 2013.

[4] L. K. Barker and G. W. Sparrow. *Analysis of Effects of Spanwise Variations of Gust Velocity on a Vane-Controlled Gust-Alleviation System*. 1971.

[5] D. G. Bateman. Design and qualification of an upstream gust generator in a low-speed wind tunnel. *Aerospace Engineering Sciences Graduate Theses and Disserations*, 2017.

[6] Joseph Bicknell. Pulsed air gust generator. *Journal of Aircraft*, 8(9):741–743, 1971.

[7] Seung Ho Cho, Taehyoun Kim, Seung Jin Song, and Sang Joon Shin. Aeroelastic analysis of an isolated airfoil under a pulsating flow. *AIAA Journal*, 45(5):1000–1006, 2007.

[8] Russel M. Cummings, William H. Mason, Scott A. Morton, and David R. McDaniel. *Applied computational aerodynamics: a modern engineering approach*. Cambridge University Press, 2015.

[9] James Delaurier and James Winfield. Simple marching-vortex model for two-dimensional unsteady aerodynamics. *Journal of Aircraft*, 27(4):376–378, 1990.

[10] M Drela and H Youngren. Xfoil:subsonic airfoil development system, 2018.

[11] F. Fonte, L. Riccobene, S. Ricci, S. Adden, and M. Martegani. *Design, Manufacturing and Validation of a Gust Generator for Wind Tunnel Test of a Large Scale Aeroelastic Model*. 2016.

[12] L. C. Garby, A. M. Kuethe, and J. D. Schetzer. *The Generation of Gusts in a Wind Tunnel and Measurement of Unsteady Lift on an Airfoil*. 1957.

[13] Kenneth Hall. Eigenanalysis of unsteady flows about airfoils, cascades, and wings. *35th Structures, Structural Dynamics, and Materials Conference*, 32(12):2426–2432, 1994.

[14] N. D. Ham, P. H. Bauer, and T. L. Lawrence. *Wind Tunnel Generation of Sinusoidal Lateral and Longitudinal Gusts by Circulation Control of Twin Parallel Airfoils*. 1974.

[15] S. F. Harding and I. G. Bryden. Generating controllable velocity fluctuations using twin oscillating hydrofoils. *Journal of Fluid Mechanics*, 713:150–158, 2012.

[16] S. F. Harding, G. S. Payne, and I. G. Bryden. Generating controllable velocity fluctuations using twin oscillating hydrofoils: experimental validation. *Journal of Fluid Mechanics*, 750:113–123, 2014.

[17] Esteban Hufstedler and Beverley J. Mckeon. Isolated gust generation for the investigation of airfoil-gust interaction. *46th AIAA Fluid Dynamics Conference*, Oct 2016.

[18] Taehyoun Kim, Changho Nam, and Youdan Kim. Reduced-order aeroservoelastic model with an unsteady aerodynamic eigen formulation. *AIAA Journal*, 35:1087–1088, 1997.

[19] Paul M.g.j. Lancelot, Jurij Sodja, Noud P.m. Werter, and Roeland De Breuker. Design and testing of a low subsonic wind tunnel gust generator. *Advances in aircraft and spacecraft science*, 4(2):125–144, 2017.

[20] LearnCAE. Flow over an oscillating airfoil ansys cfx pitching mode, 2018.

[21] Eli Livne. *Vortex Mirrors in 2D Tunnel*. 2019.

[22] H. Mai, J. Neumann, and H. Hennings. Gust response: A validation experiment and preliminary numerical simulations. *IFASD*, 51, 2011.

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

[24] Jens Neumann and Holger Mai. Gust response: Simulation of an aeroelastic experiment by a fluid–structure interaction method. *Journal of Fluids and Structures*, 38:290–302, 2013.

[25] Nhan T. Nguyen, Juntao Xiong, and Nick B. Cramer. Simulation and modeling of flow generated by gust generator in a wind tunnel. *AIAA Scitech 2020 Forum*, May 2020.

[26] M. H. Patel and G. J. Hancock. A gust tunnel facility. *Aeronautical Research Council: Reports and Memoranda*, 1976.

[27] Charles Poussot-Vassal, Fabrice Demourant, Arnaud Lepage, and Dominique Le Bihan. Gust load alleviation: Identification, control, and wind tunnel testing of a 2-d aeroelastic airfoil. *IEEE Transactions on Control Systems Technology*, 25(5):1736–1749, 2017.

[28] Jake D. Quenzer, Alison Zongolowicz, Kimberly A. Hinson, Bijan Barzgaran, Eli Livne, Mehran Mesbahi, and Kristi Morgansen. Model for aeroelastic response to gust excitation. *AIAA Scitech 2019 Forum*, Jun 2019.

[29] L. T. Redd, P. W. Hanson, and E. C. Wynne. *Evaluation of a Wind-Tunnel Gust Response Technique Including Correlations With Analytical and Flight Test Results.* 1979.

[30] R. H. Rhyne and R. Steiner. *Power Spectral Measurement of Atmospheric Turbulence in Severe Storms and Cumulus Clouds.* 1964.

[31] Sergio Ricci, Alessandro Gaspari, Luca Riccobene, and Federico Fonte. Design and wind tunnel test validation of gust load alleviation systems. *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, May 2017.

[32] H. L. Runyan and C. E. Watkins. *Considerations on the Effect of Wind-Tunnel Walls on Oscillating Air Forces for Two-Dimensional Subsonic Compressible Flow.* 1951.

[33] Aj Saddington, Mv Finnis, and K Knowles. The characterisation of a gust generator for aerodynamic testing. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 229(7):1214–1225, 2014.

[34] Salim Mohamed Salim and Kian Chuan Ong. Performance of rans, urans and les in the prediction of airflow and pollutant dispersion. *Lecture Notes in Electrical Engineering IAENG Transactions on Engineering Technologies*, page 263–274, Jun 2012.

[35] P. P. Sarkar and F. L. Haan. Design and testing of iowa state university's aabl wind and gust tunnel. *Iowa State University Department of Aerospace Engineering.*

[36] D. M. Tang, Paul G. A. Cizmas, and E. H. Dowell. Experiments and analysis for a gust generator in a wind tunnel. *Journal of Aircraft*, 33(1):139–148, 1996.

[37] Rami Veiberman, Martin Weiss, Moti Karpel, Federico Fonte, Lorenzo Travaglini, and Sergio Ricci. Aeroservoelastic response of nonlinear wind tunnel model to non-uniform gust field. *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2016.

[38] G. Warwick. Making bumps: Gusts generated at cruise speed in a wind tunnel could aid in certification and design. *Aviation Week and Space Technology*, page 48, Jul 2015.

[39] Andreas Wildschek, Rudolf Maier, Falk Hoffmann, Josef Steigenberger, Karl-Heinz Kaulfuss, Christian Breitsamter, Alexander Allen, Nikolaus Adams, Horst Baier, and Theodorus Giannopoulos. Wind tunnel testing of an adaptive control system for vibration suppression on aircraft. *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007.

[40] Kieran T. Wood, Ronald C. Cheung, Thomas S. Richardson, Jonathan E. Cooper, Oliver Darbyshire, and Clyde Warsop. A new gust generator for a low speed wind tunnel: Design and commissioning. *55th AIAA Aerospace Sciences Meeting*, May 2017.

[41] Heyong Xu, Shilong Xing, and Zhengyin Ye. Numerical study of an airfoil/rotating-slotted-cylinder based flutter exciter. *Journal of Aircraft*, 52(6):2100–2105, 2015.

[42] Yang Yang, Mingshui Li, Cunming Ma, and Shaopeng Li. Experimental investigation on the unsteady lift of an airfoil in a sinusoidal streamwise gust. *Physics of Fluids*, 29(5), 2017.

Appendix A

MATLAB CODE

Table of Contents


```
clear all
close all
clc
```

Basic Marching 2D Vortex Lattice Model %%

Geometry, Flow Characteristics and Initialization

```
%set up frequency of oscillation
f           =       1;
w_n         =       2*pi*f;

%define geometry and incoming flow field
b           =       .254/2; %half-chord length (c=2b) [m]
N_T         =       1; %number of time steps per period
x_first     =       10000*2*pi/(N_T*w_n); %b/2; %distance from
 trailing edge (TE) to first vortex
U_inf       =       60; %speed of incoming flow [m/s]
```

```
rho         =       1.185; %density of incoming flow [kg/m^3]
aa          =       .95; %releases wake gradually

%mirroring parameters
m           =       11; %number of vortex mirrors minus one
H           =       .4572; %height of half of windtunnel
z_bar       =       0; %distance single airfoil lies above centerline
 of wind tunnel

%multiple airfoil parameters
%note: airfoil A is upper airfoil and airfoil B is lower airfoil
z_bar_A = .1778; %z-coordinate of airfoil A w.r.t. centerline
z_bar_B = -.1778; %z-cooridnate of airfoil B w.r.t. centerline

%define time step
delta_t     =       2*pi/(N_T*w_n); %length of time step

%define length of wake
C           =       50*U_inf*delta_t/(2*b);
n           =       round(C*2*b/(U_inf*delta_t)); %number of vorticies
 considered in wake
%must use n = 65 if comparing to CFD results

%parameters for monitoring runs
check1      =       0;
check2      =       0;
check3      =       0;
check4      =       0;

%imaginary numbers
im          =       j;

%allocate size for future matricies
N_ci        =       zeros(n,1);
N_ai        =       zeros(n,1);
N           =       zeros(n,1);
M_ai        =       zeros(n,1);
M_ti        =       zeros(n,1);
M_c         =       zeros(n,1);
M           =       zeros(n,1);

%mirroring option: if mirror = 1, mirror; if mirror = 0, don't mirror
mirror                  =       1;
theodorsencomparison    =       1;
multiple                =       1;
```

Airfoil Oscillation

```
%set up frequency of oscillation
ts          =       1/f/(n/2);
t           =       0:ts:ts*n-ts;

%initialize airfoil plunging oscillation
```

```
h0          =       0;
p0          =       0;
h           =       h0*exp(im*(w_n*t+p0));
h_d         =       i*h0*w_n*exp(im*(w_n*t+p0));
h_dd        =       h0*w_n^2*(-exp(im*(w_n*t+p0)));
h           =       imag(h)';
h_d         =       imag(h_d)';
h_dd        =       imag(h_dd)';

%initialize airfoil rotating oscillation
alpha0      =       1*pi/180;
alpha       =       alpha0*exp(im*(w_n*t));
alpha_d     =       i*alpha0*w_n*exp(im*(w_n*t));
alpha_dd    =       alpha0*w_n^2*(-exp(im*(w_n*t)));
alpha       =       imag(alpha)';
alpha_d     =       imag(alpha_d)';
alpha_dd    =       imag(alpha_dd)';
```

Mirroring Option

```
if mirror   ==      1
```

Vortex Mirroring for Airfoil

```
d_airfoil_mirror_top    =   zeros(m-1,1);
d_airfoil_mirror_bottom =   zeros(m-1,1);
K_airfoil_mirror_top    =   zeros(m-1,1);
K_airfoil_mirror_bottom =   zeros(m-1,1);

for i = 2:m

    d_airfoil_mirror_top(i-1) = (-1)^(i)*((2*i-2)*H - z_bar);
    d_airfoil_mirror_bottom(i-1) = (-1)^(i+1)*((2*i-2)*H + z_bar);

end

for i=1:m-1
    if mod(i,2) == 0
        K_airfoil_mirror_top(i) = 1/(2*pi*((b)^2 +
 (d_airfoil_mirror_top(i)-z_bar)^2)^(1/2));
        K_airfoil_mirror_bottom(i) = 1/(2*pi*((b)^2 +
 (d_airfoil_mirror_bottom(i)-z_bar)^2)^(1/2));
    else
        K_airfoil_mirror_top(i) = -1/(2*pi*((b)^2 +
 (d_airfoil_mirror_top(i)-z_bar)^2)^(1/2));
        K_airfoil_mirror_bottom(i) = -1/(2*pi*((b)^2 +
 (d_airfoil_mirror_bottom(i)-z_bar)^2)^(1/2));
    end
end

K_airfoil_mirror    =
 [K_airfoil_mirror_top;K_airfoil_mirror_bottom];
```

Vortex Mirroring in Wake

```
d_wake_mirror_top   =   zeros(m-1,n);
d_wake_mirror_bottom =  zeros(m-1,n);

for i = 1:n
    for j = 1:m-1
        if mod(j,2) == 0
            d_wake_mirror_top(j,i) = -((d_airfoil_mirror_top(j)-
z_bar)^2 + (b/2 + x_first + U_inf*delta_t*(i-1))^2)^(1/2);
            d_wake_mirror_bottom(j,i) = -((d_airfoil_mirror_bottom(j)-
z_bar)^2 + (b/2 + x_first + U_inf*delta_t*(i-1))^2)^(1/2);
        else
            d_wake_mirror_top(j,i) = ((d_airfoil_mirror_top(j)-
z_bar)^2 + (b/2 + x_first + U_inf*delta_t*(i-1))^2)^(1/2);
            d_wake_mirror_bottom(j,i) = ((d_airfoil_mirror_bottom(j)-
z_bar)^2 + (b/2 + x_first + U_inf*delta_t*(i-1))^2)^(1/2);
        end
    end
end

K_wake_mirror_top   =   1./(2.*pi.*d_wake_mirror_top);
K_wake_mirror_bottom =  1./(2.*pi.*d_wake_mirror_bottom);
K_wake_mirror       =   [K_wake_mirror_top; K_wake_mirror_bottom];

K_mirror            =   [K_airfoil_mirror, K_wake_mirror];
```

Matrix Setup with Mirroring

```
%initialize gamma and downwash
gamma = zeros((n+1),(n+1)); %at step i=1, {gamma} = {0}
downwash = zeros((n+1),(n+1)); %at step i=1, {downwash} = {0}

%define gamma and downward velocity relationship on airfoil
K_11            =       1/(2*pi*b);

%define gamma and downward velocity relationship in wake
K_wake          =       zeros(1,n);

for i=1:n

    K_wake(i) = -1/(2*pi*(b/2+x_first+U_inf*delta_t*(i-1)));

end

%combine K_11 and K_wake with K_mirror
K_testsection = [K_11 K_wake];
K_comp = [K_testsection;K_mirror];

for i = 1:n+1

    K(i) = sum(K_comp(:,i));
```

```
end

%form matricies
%system will be in the form: [R]{gamma(i+1)}=[S]{gamma(i)}+[T(i+1)]
R = eye(n+1);
R(1,:) = K;
R(2,1) = 1; %[R] is like an identity matrix with the top row being
 made of Ks and the 2,1 entry equal to a 1

S = [zeros(1,n); eye(n)];
S = [S zeros(n+1,1)];%[S] is like an identity matrix with an extra row
 and column of zeros
```


Airfoil Behavior in Flow Field with Mirroring

```
lag = pi/2;

%solve for gamma
A = R\S; %{gamma(i+1)=[A]*{gamma(i)}+{u(i+1)}

%define loop
for i=1:n

    S(i,i)      =   0;
    S(i+1,i+1)  =   aa;

    %downwash
        %downwash(1,i+1) = U_inf*1; %downwash for every time other
 than i=1,n+1 equals U_inf (step input)
        %downwash(1,i+1) = U_inf*(h_d(i)/U_inf + alpha(i));
        downwash(1,i+1) = U_inf*alpha0*sin(w_n*t(i) -
 lag)+w_n*b*alpha0*cos(w_n*t(i) - lag);
        %downwash(1,2) = U_inf*alpha0;

    %calculate circulation
        gamma(:,i+1)=   A*gamma(:,i) + R\downwash(:,i+1); %solve for
 gamma(i+1)

    %lift (from DeLaurier)
        N_ci(i)     =   rho*U_inf*gamma(1,i+1);
        N_ai(i)     =   1/4*rho*pi*(2*b)^2*(h_dd(i) +
 U_inf*alpha_d(i));
        N(i)        =   N_ci(i) + N_ai(i);

    %moment (from DeLaurier)
        M_ai(i)     =   -(1/128)*rho*pi*(2*b)^4*alpha_dd(i);
        M_ti(i)     =   -(1/16)*rho*pi*(2*b)^3*U_inf*alpha_d(i);
        M_c(i)      =   1/4*(2*b)*N_ci(i);
        M(i)        =   M_c(i) + M_ai(i) + M_ti(i);

        check1 = check1 + 1;
```

```
end

gamma(1:n+1,:);
```

Speed Check Along Wall with Mirroring

142

```
x1 = zeros(n+1,1);
V_edge_top = zeros(n+1,1);
V_edge_bottom = zeros(n+1,1);

for kk = 2:n+1
        x1(kk) = x1(kk-1) + .25; %x_first+U_inf*delta_t*(kk-2);
end

%allocate sizes
r_airfoil_top = zeros(2*m-1,1);
r_airfoil_bottom = zeros(2*m-1,1);
K_airfoil_edge_top = zeros(2*m-1,1);
K_airfoil_edge_bottom = zeros(2*m-1,1);

%concatenate distances
z0 = [z_bar;d_airfoil_mirror_top;d_airfoil_mirror_bottom];
z1_top = H;
z1_bottom = -H;
d_airfoil_edge_top = [z_bar-z1_top;d_airfoil_mirror_top-
z1_top;d_airfoil_mirror_bottom-z1_top];
d_airfoil_edge_bottom = [z_bar-z1_bottom;d_airfoil_mirror_top-
z1_bottom;d_airfoil_mirror_bottom-z1_bottom];

for k = 1:n+1

x0 = zeros(1,n);

%find kernels for airfoil mirrors for both top and bottom wall
for i=1:length(d_airfoil_edge_top)

    r_airfoil_top(i) = ((b + x1(k))^2 +
 (d_airfoil_edge_top(i))^2)^(1/2);
    r_airfoil_bottom(i) = ((b + x1(k))^2 +
 (d_airfoil_edge_bottom(i))^2)^(1/2);

    if i == 1
        K_airfoil_edge_top(i) = 1/(2*pi*((b + x1(k))^2 +
 (d_airfoil_edge_top(i))^2)^(1/2));
        K_airfoil_edge_bottom(i) = 1/(2*pi*((b + x1(k))^2 +
 (d_airfoil_edge_bottom(i))^2)^(1/2));
    elseif (2<=i) && (i<=m)
        K_airfoil_edge_top(i) = -sign(d_airfoil_edge_top(i))*1/
(2*pi*((b + x1(k))^2 + (d_airfoil_edge_top(i))^2)^(1/2));
        K_airfoil_edge_bottom(i) = -sign(d_airfoil_edge_bottom(i))*1/
(2*pi*((b + x1(k))^2 + (d_airfoil_edge_bottom(i))^2)^(1/2));
    else
```

```
        K_airfoil_edge_top(i) = sign(d_airfoil_edge_top(i))*1/
(2*pi*((b + x1(k))^2 + (d_airfoil_edge_top(i))^2)^(1/2));
        K_airfoil_edge_bottom(i) = sign(d_airfoil_edge_bottom(i))*1/
(2*pi*((b + x1(k))^2 + (d_airfoil_edge_bottom(i))^2)^(1/2));
    end

end

%find kernels for wake mirrors for both top and bottom wall
d_wake_edge_top   =   zeros(2*m-1,n);
d_wake_edge_bottom =  zeros(2*m-1,n);

for i = 1:n
    x0(i) = b/2 + x_first + U_inf*delta_t*(i-1);
    for j = 1:2*m-1
        if j == 1
            d_wake_edge_top(j,i) = ((d_airfoil_edge_top(j))^2 + (b/2 +
 x_first + U_inf*delta_t*(i-1) - x1(k))^2)^(1/2);
            d_wake_edge_bottom(j,i) = ((d_airfoil_edge_bottom(j))^2 +
 (b/2 + x_first + U_inf*delta_t*(i-1) - x1(k))^2)^(1/2);
        elseif (2<=j) && (j<=m)
            d_wake_edge_top(j,i) = -
sign(d_airfoil_edge_top(j))*((d_airfoil_edge_top(j))^2 + (b/2 +
 x_first + U_inf*delta_t*(i-1) - x1(k))^2)^(1/2);
            d_wake_edge_bottom(j,i) = -
sign(d_airfoil_edge_bottom(j))*((d_airfoil_edge_bottom(j))^2 + (b/2 +
 x_first + U_inf*delta_t*(i-1) - x1(k))^2)^(1/2);
        else
            d_wake_edge_top(j,i) =
 sign(d_airfoil_edge_top(j))*((d_airfoil_edge_top(j))^2 + (b/2 +
 x_first + U_inf*delta_t*(i-1) - x1(k))^2)^(1/2);
            d_wake_edge_bottom(j,i) =
 sign(d_airfoil_edge_bottom(j))*((d_airfoil_edge_bottom(j))^2 + (b/2 +
 x_first + U_inf*delta_t*(i-1) - x1(k))^2)^(1/2);
        end
    end
end

K_wake_edge_top   =   1./(2.*pi.*d_wake_edge_top);
K_wake_edge_bottom =  1./(2.*pi.*d_wake_edge_bottom);

K_edge_top_matrix = [K_airfoil_edge_top,K_wake_edge_top];
K_edge_bottom_matrix = [K_airfoil_edge_bottom,K_wake_edge_bottom];

%assemble distances for Vx and Vz
x0 = [-b,x0];
r_top = [r_airfoil_top,abs(d_wake_edge_top)];
r_bottom = [r_airfoil_bottom,abs(d_wake_edge_bottom)];

%align kernels for calculation
K_edge_top_line = K_edge_top_matrix(1,:);
K_edge_bottom_line = K_edge_bottom_matrix(1,:);
gamma_wall = gamma(:,n+1);
```

```
for i = 2:2*m-1

    K_edge_top_line = [K_edge_top_line,K_edge_top_matrix(i,:)];
    K_edge_bottom_line =
 [K_edge_bottom_line,K_edge_bottom_matrix(i,:)];
    gamma_wall = [gamma_wall;gamma(:,n+1)];

end

%calculate speed along wall
V_edge_top_induced = K_edge_top_line'.*gamma_wall;
V_edge_bottom_induced = K_edge_bottom_line'.*gamma_wall;

%allocate sizes
Vx_top_indiv = zeros((n+1)*(2*m-1),1);
Vz_top_indiv = zeros((n+1)*(2*m-1),1);
Vx_bottom_indiv = zeros((n+1)*(2*m-1),1);
Vz_bottom_indiv = zeros((n+1)*(2*m-1),1);

%seperate speeds into X and Z components
for ii = 1:2*m-1
    for jj = 1:n+1
        Vz_top_indiv((n+1)*(ii-1)+jj,1) = -V_edge_top_induced((n
+1)*(ii-1)+jj)*(x1(k)-x0(jj))/r_top(ii,jj);
        Vx_top_indiv((n+1)*(ii-1)+jj,1) = V_edge_top_induced((n
+1)*(ii-1)+jj)*(z1_top-z0(ii))/r_top(ii,jj);
        Vz_bottom_indiv((n+1)*(ii-1)+jj,1) = -V_edge_bottom_induced((n
+1)*(ii-1)+jj)*(x1(k)-x0(jj))/r_bottom(ii,jj);
        Vx_bottom_indiv((n+1)*(ii-1)+jj,1) = V_edge_bottom_induced((n
+1)*(ii-1)+jj)*(z1_bottom-z0(ii))/r_bottom(ii,jj);
    end
end

%sum components together for collective speeds
Vx_top(k) = U_inf + sum(Vx_top_indiv);
Vz_top(k) = sum(Vz_top_indiv);
Vx_bottom(k) = U_inf + sum(Vx_bottom_indiv);
Vz_bottom(k) = sum(Vz_bottom_indiv);

check2 = check2 + 1;

end

%calculate angle of attack
AOA_edge_top = (Vz_top./Vx_top).*180./pi;
AOA_edge_bottom = (Vz_bottom./Vx_bottom).*180./pi;
AOA_edge = [AOA_edge_top', AOA_edge_bottom'];
V_Z = [Vz_top',Vz_bottom'];
maxAOA = max(abs(AOA_edge));

% % % %plot results
% % % figure(2)
% % % %sgtitle('Wall Angle of Attack Check')
% % % %subplot(2,1,1)
```

```
% % % plot(x1/(2*b)-b/2,AOA_edge(1:n+1,1))
% % % title('Wall AOA - Top Wall (t=T/4)')
% % % xlabel('distance along line from airfoil TE downstream in
 chords')%('distance in wake (m)')
% % % ylabel('U_z/U_x (deg)')
% % % %ylim([-.2 .2])
% % % %xlim([x1(2) x1(n)])
% % % xlim([1 6])
% % % %ylim([0 10*10^(-4)])
% % % % % % subplot(2,1,2)
% % % % % % plot(x1,AOA_edge(1:n+1,2))
% % % % % % title('Wall AOA - Bottom Edge')
% % % % % % xlabel('distance in wake (m)')
% % % % % % ylabel('U_z/U_x (deg)')
% % % % % % %ylim([-.2 .2])
% % % % % % %xlim([x1(2) x1(n)])

elseif mirror == 0
```

Matrix Setup without Mirroring

```
%initialize gamma and downwash
gamma = zeros((n+1),(n+1)); %at step i=1, {gamma} = {0}
downwash = zeros((n+1),(n+1)); %at step i=1, {downwash} = {0}

%define gamma and downward velocity relationship on airfoil
K_11 = 1/(2*pi*b);

%define gamma and downward velocity relationship in wake
K_wake =  zeros(1,n);

for i=1:n

    K_wake(i) = -1/(2*pi*(b/2+x_first+U_inf*delta_t*(i-1)));

end

%combine K_11 and K_wake
K = [K_11 K_wake];

%form matricies
%system will be in the form: [R]{gamma(i+1)}=[S]{gamma(i)}+[T(i+1)]
R = eye(n+1);
R(1,:) = K;
R(2,1) = 1; %[R] is like an identity matrix with the top row being
 made of Ks and the 2,1 entry equal to a 1

S = [zeros(1,n); eye(n)];
S = [S zeros(n+1,1)]; %[S] is like an identity matrix with an extra
 row and column of zeros
```

Airfoil Behavior in Flow Field without Mirroring


```
%solve for gamma
A = R\S; %{gamma(i+1)=[A]*{gamma(i)}+{u(i+1)}

%define loop
for i=1:n

    S(i,i)      =   0;
    S(i+1,i+1)  =   aa;

    %downwash
        %downwash(1,i+1) = U_inf*1; %downwash for every time other
 than i=1,n+1 equals U_inf (step input)
        %downwash(1,i+1) = U_inf*(h_d(i)/U_inf + alpha(i));
        downwash(1,i+1) =
 U_inf*alpha0*sin(w_n*t(i))+w_n*b*alpha0*cos(w_n*t(i));

    %calculate circulation
        gamma(:,i+1) = A*gamma(:,i) + R\downwash(:,i+1); %solve for
 gamma(i+1)

    %lift (from DeLaurier)
        N_ci(i) = rho*U_inf*gamma(1,i+1);
        N_ai(i) = 1/4*rho*pi*(2*b)^2*(h_dd(i) + U_inf*alpha_d(i));
        N(i) = N_ci(i) + N_ai(i);

    %moment (from DeLaurier)
        M_ai(i) = -(1/128)*rho*pi*(2*b)^4*alpha_dd(i);
        M_ti(i) = -(1/16)*rho*pi*(2*b)^3*U_inf*alpha_d(i);
        M_c(i) = 1/4*(2*b)*N_ci(i);
        M(i) = M_c(i) + M_ai(i) + M_ti(i);

        check3 = check3 + 1

end

gamma(1:n+1,:);

end
```

Theodorsen Solution

```
%define parameters
k = w_n*b/U_inf;
a = -1; %distance to rotation axis in units of b b/c focus is on 3/4-
chord
flap = 1; %no flap so c is at trailing edge
ndof = 2; %only consider plunge and AOA, not beta
[FF,GG] = TheodorsenFunction(k); %k is reduced frequency
T = Tfunctions(a,flap); %call Tfunctions for aero matrix to use
AA = AeroMatrix(k,a,flap,b,T,FF,GG,ndof);
```

```
AA = AA(1:ndof,1:ndof);

%initialize lift matrix
L_theodorsen = zeros(n,1);
M_theodorsen = zeros(n,1);

%calculate lift
LM = 1/2*rho*U_inf^2*AA*[h0;alpha0]; %check U_inf^2 or U_inf
L_amp_theodorsen = abs(LM(1));
L_phase_theodorsen = angle(LM(1));
M_amp_theodorsen = abs(LM(2));
M_phase_theodorsen = 0; %angle(LM(2));

for i = 1:n

    L_theodorsen(i) = L_amp_theodorsen*sin(w_n*t(i) +
 L_phase_theodorsen);
    M_theodorsen(i) = M_amp_theodorsen*sin(w_n*t(i) +
 M_phase_theodorsen);

end
```

plot and check

```
% % % figure(1)
% % % %sgtitle('Theodorsen-VLM Matching for Single Airfoil')
% % % subplot(2,1,1)
% % % plot(w_n*t,-N)%,'s')
% % % hold on
% % % plot(w_n*t,L_theodorsen,'--')
% % % hold off
% % % title('Lift')
% % % legend('VLM','Theodorsen','location','southeast')
% % % xlim([w_n*t(1) w_n*t(n)])
% % % %ylim([-700 700])
% % % xlabel('\omegat')
% % %
% % % subplot(2,1,2)
% % % plot(w_n*t,M)%,'s')
% % % hold on
% % % plot(w_n*t,M_theodorsen,'--')
% % % hold off
% % % title('Moment')
% % % legend('VLM','Theodorsen','location','northeast')
% % % xlim([w_n*t(1) w_n*t(n)])
% % % %ylim([-40 40])
% % % xlabel('\omegat')

%Lift = max(N)
% % % %% CFD Wall Effects
% % %
% % % %%%%%%%%%%%%%%%%%% for 5 AOA Case %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % %
```

```
% % % % % import excel sheet
% % % T1 =
 table2array(readtable('wall_effects_NACA0015.xlsx','Range','A1:C65'));
% % % time1 = T1(:,1);
% % % force_nowalls = T1(:,2);
% % % force_walls = T1(:,3);
% % %
% % % % plot and check
% % % figure(7)
% % % subplot(2,1,1)
% % % plot(w_n*t,-N)%,'s')
% % % hold on
% % % plot(w_n*t,L_theodorsen)
% % % hold off
% % % title('Lift')
% % % legend('VLM','Theodorsen','location','southeast')
% % % xlim([0 w_n*t(n)])
% % % xlabel('\omegat')
% % %
% % % subplot(2,1,2)
% % % plot(w_n*time1,force_walls)%,'s')
% % % hold on
% % % plot(w_n*time1,force_nowalls)
% % % hold off
% % % title('Lift')
% % % legend('Walls','No Walls','location','southeast')
% % % xlim([0 w_n*t(n)])
% % % xlabel('\omegat')
% % % ylabel('normalized lift force (N/m)')
% % %
% % % figure(8)
% % % plot(w_n*t,-N)
% % % hold on
% % % plot(w_n*t,L_theodorsen)
% % % hold on
% % % plot(w_n*time1,force_walls,'s')
% % % hold on
% % % plot(w_n*time1,force_nowalls,'s')
% % % hold off
% % % %title('Lift')
% % % legend('VLM - Walls','VLM - No Walls','CFD - Walls','CFD - No
 Walls','location','southeast')
% % % xlim([0 w_n*t(n)])
% % % xlabel('\omegat')
% % % ylabel('normalized lift force (N/m)')
```

VLM v. CFD

```
if theodorsencomparison == 1
```

Check VLM Against CFD

```
% import excel sheet
```

```
T1 = table2array(readtable('onevane.xlsx','Range','A1:B33'));
time1 = T1(:,1);
force1 = T1(:,2);

% % % % plot and check
% % % figure(3)
% % % %plot(t,L_theodorsen)
% % % %hold on
% % % % plot(t,-N)
% % % hold on
% % % plot(time1,force1,'o')
% % % hold off
% % % %title('Lift Comparison - Theodorsen, VLM and CFD')
% % % legend('VLM','CFD','location','southeast')
% % % xlim([time1(1) time1(end)])
% % % xlabel('time (sec)')
% % % ylabel('normalized lift force (N/m)')

% import excel sheet
T12 = table2array(readtable('onevane_12hz.xlsx','Range','A2:B34'));
time12 = T12(:,1);
force12 = T12(:,2);

% % % % plot and check
% % % figure(12)
% % % % plot(t,L_theodorsen)
% % % % hold on
% % % % plot(t,-N)
% % % hold on
% % % % plot(time12,force12,'o')
% % % hold off
% % % %title('Lift Comparison - Theodorsen, VLM and CFD')
% % % legend('VLM','CFD','location','southeast')
% % % xlim([time12(1) time12(end)])
% % % xlabel('time (sec)')
% % % ylabel('normalized lift force (N/m)')

Warning: Table variable names were modified to make them valid MATLAB
identifiers. The original names are saved in the VariableDescriptions
 property.

end
```

Modeling Multiple Airfoils in Wind Tunnel

```
if multiple == 1
```

Vortex Mirroring for Multiple Airfoil

```
%set aside space for kernels for airfoil A vortices affecting airfoil
 A
d_airfoil_mirror_top_A    =   zeros(m-1,1);
```

```
d_airfoil_mirror_bottom_A =   zeros(m-1,1);
K_airfoil_mirror_top_AtoA    =   zeros((m-1),1);
K_airfoil_mirror_bottom_AtoA =   zeros((m-1),1);

%set aside space for kernels for airfoil B vortices affecting airfoil
 B
d_airfoil_mirror_top_B    =   zeros(m-1,1);
d_airfoil_mirror_bottom_B =   zeros(m-1,1);
K_airfoil_mirror_top_BtoB    =   zeros((m-1),1);
K_airfoil_mirror_bottom_BtoB =   zeros((m-1),1);

%AtoB means airfoil A's vorticies affecting circulation on airfoil B
%BtoA means airfoil B's vorticies affecting circulation on airfoil A

%set aside space for kernels for airfoil A vortices affecting airfoil
 B
K_airfoil_mirror_top_AtoB    =   zeros((m-1),1);
K_airfoil_mirror_bottom_AtoB =   zeros((m-1),1);

%set aside space for kernels for airfoil B vortices affecting airfoil
 A
K_airfoil_mirror_top_BtoA    =   zeros((m-1),1);
K_airfoil_mirror_bottom_BtoA =   zeros((m-1),1);

for i = 2:m

    %z-coordinate of vortices from airfoil A w.r.t. centerline
    d_airfoil_mirror_top_A(i-1) = (-1)^(i)*((2*i-2)*H - z_bar_A);
    d_airfoil_mirror_bottom_A(i-1) = (-1)^(i+1)*((2*i-2)*H + z_bar_A);
    %z-coordinate of vortices from airfoil B w.r.t. centerline
    d_airfoil_mirror_top_B(i-1) = (-1)^(i)*((2*i-2)*H - z_bar_B);
    d_airfoil_mirror_bottom_B(i-1) = (-1)^(i+1)*((2*i-2)*H + z_bar_B);

end

for i=1:m-1
    if mod(i,2) == 0
        %kernels for each airfoil A's respective vortices
        K_airfoil_mirror_top_AtoA(i) = 1/(2*pi*((b)^2 +
 (d_airfoil_mirror_top_A(i)-z_bar_A)^2)^(1/2));
        K_airfoil_mirror_bottom_AtoA(i) = 1/(2*pi*((b)^2 +
 (d_airfoil_mirror_bottom_A(i)-z_bar_A)^2)^(1/2));
        %kernels for each airfoil B's respective vortices
        K_airfoil_mirror_top_BtoB(i) = 1/(2*pi*((b)^2 +
 (d_airfoil_mirror_top_B(i)-z_bar_B)^2)^(1/2));
        K_airfoil_mirror_bottom_BtoB(i) = 1/(2*pi*((b)^2 +
 (d_airfoil_mirror_bottom_B(i)-z_bar_B)^2)^(1/2));
    else
        %kernels for each airfoil A's respective vortices
        K_airfoil_mirror_top_AtoA(i) = -1/(2*pi*((b)^2 +
 (d_airfoil_mirror_top_A(i)-z_bar_A)^2)^(1/2));
        K_airfoil_mirror_bottom_AtoA(i) = -1/(2*pi*((b)^2 +
 (d_airfoil_mirror_bottom_A(i)-z_bar_A)^2)^(1/2));
        %kernels for each airfoil B's respective vortices
```

```
        K_airfoil_mirror_top_BtoB(i) = -1/(2*pi*((b)^2 +
 (d_airfoil_mirror_top_B(i)-z_bar_B)^2)^(1/2));
        K_airfoil_mirror_bottom_BtoB(i) = -1/(2*pi*((b)^2 +
 (d_airfoil_mirror_bottom_B(i)-z_bar_B)^2)^(1/2));
    end
end

for i=1:m-1
    if mod(i,2) == 0
        %kernels for airfoil B from airfoil A's vorticies
        K_airfoil_mirror_top_AtoB(i) = 1/(2*pi*((b)^2 +
 (d_airfoil_mirror_top_A(i)-z_bar_B)^2)^(1/2));
        K_airfoil_mirror_bottom_AtoB(i) = 1/(2*pi*((b)^2 +
 (d_airfoil_mirror_bottom_A(i)-z_bar_B)^2)^(1/2));
        %kernels for airfoil A from airfoil B's vortices
        K_airfoil_mirror_top_BtoA(i) = 1/(2*pi*((b)^2 +
 (d_airfoil_mirror_top_B(i)-z_bar_A)^2)^(1/2));
        K_airfoil_mirror_bottom_BtoA(i) = 1/(2*pi*((b)^2 +
 (d_airfoil_mirror_bottom_B(i)-z_bar_A)^2)^(1/2));
    else
        %kernels for airfoil B from airfoil A's vorticies
        K_airfoil_mirror_top_AtoB(i) = -1/(2*pi*((b)^2 +
 (d_airfoil_mirror_top_A(i)-z_bar_B)^2)^(1/2));
        K_airfoil_mirror_bottom_AtoB(i) = -1/(2*pi*((b)^2 +
 (d_airfoil_mirror_bottom_A(i)-z_bar_B)^2)^(1/2));
        %kernels for airfoil A from airfoil B's vortices
        K_airfoil_mirror_top_BtoA(i) = -1/(2*pi*((b)^2 +
 (d_airfoil_mirror_top_B(i)-z_bar_A)^2)^(1/2));
        K_airfoil_mirror_bottom_BtoA(i) = -1/(2*pi*((b)^2 +
 (d_airfoil_mirror_bottom_B(i)-z_bar_A)^2)^(1/2));
    end
end

%compile both top and bottom mirroring kernels for each airfoil
K_airfoil_mirror_AtoA      =
 [K_airfoil_mirror_top_AtoA;K_airfoil_mirror_bottom_AtoA];
K_airfoil_mirror_BtoB      =
 [K_airfoil_mirror_top_BtoB;K_airfoil_mirror_bottom_BtoB];
%account for cross effects
K_airfoil_mirror_AtoB      =
 [K_airfoil_mirror_top_AtoB;K_airfoil_mirror_bottom_AtoB];
K_airfoil_mirror_BtoA      =
 [K_airfoil_mirror_top_BtoA;K_airfoil_mirror_bottom_BtoA];
```

Vortex Mirroring in Wake

```
%set aside room for distance matricies
d_wake_mirror_top_AtoA   =   zeros((m-1),n);
d_wake_mirror_bottom_AtoA =  zeros((m-1),n);
d_wake_mirror_top_BtoB   =   zeros((m-1),n);
d_wake_mirror_bottom_BtoB =  zeros((m-1),n);

d_wake_mirror_top_AtoB   =   zeros((m-1),n);
```

```
d_wake_mirror_bottom_AtoB =  zeros((m-1),n);
d_wake_mirror_top_BtoA   =   zeros((m-1),n);
d_wake_mirror_bottom_BtoA =  zeros((m-1),n);

for i = 1:n
    for j = 1:m-1
        if mod(j,2) == 0
            %determine distances for airfoil A wake mirroring based on
 airfoil A contributions
            d_wake_mirror_top_AtoA(j,i) = ((d_airfoil_mirror_top_A(j)-
z_bar_A)^2 + (b/2 + x_first + U_inf*delta_t*(i-1))^2)^(1/2);
            d_wake_mirror_bottom_AtoA(j,i) =
 ((d_airfoil_mirror_bottom_A(j)-z_bar_A)^2 + (b/2 + x_first +
 U_inf*delta_t*(i-1))^2)^(1/2);
            %determine distances for airfoil B wake mirroring based on
 airfoil B contributions
            d_wake_mirror_top_BtoB(j,i) = ((d_airfoil_mirror_top_B(j)-
z_bar_B)^2 + (b/2 + x_first + U_inf*delta_t*(i-1))^2)^(1/2);
            d_wake_mirror_bottom_BtoB(j,i) =
 ((d_airfoil_mirror_bottom_B(j)-z_bar_B)^2 + (b/2 + x_first +
 U_inf*delta_t*(i-1))^2)^(1/2);
        else
            %determine distances for airfoil A wake mirroring based on
 airfoil A contributions
            d_wake_mirror_top_AtoA(j,i) = -
((d_airfoil_mirror_top_A(j)-z_bar_A)^2 + (b/2 + x_first +
 U_inf*delta_t*(i-1))^2)^(1/2);
            d_wake_mirror_bottom_AtoA(j,i) = -
((d_airfoil_mirror_bottom_A(j)-z_bar_A)^2 + (b/2 + x_first +
 U_inf*delta_t*(i-1))^2)^(1/2);
            %determine distances for airfoil B wake mirroring based on
 airfoil B contributions
            d_wake_mirror_top_BtoB(j,i) = -
((d_airfoil_mirror_top_B(j)-z_bar_B)^2 + (b/2 + x_first +
 U_inf*delta_t*(i-1))^2)^(1/2);
            d_wake_mirror_bottom_BtoB(j,i) = -
((d_airfoil_mirror_bottom_B(j)-z_bar_B)^2 + (b/2 + x_first +
 U_inf*delta_t*(i-1))^2)^(1/2);
        end
    end
end

for i = 1:n
    for j = 1:m-1
        if mod(j,2) == 0
            %determine distances for airfoil B wake mirroring based on
 airfoil A contributions
            d_wake_mirror_top_AtoB(j,i) = ((d_airfoil_mirror_top_A(j)-
z_bar_B)^2 + (b/2 + x_first + U_inf*delta_t*(i-1))^2)^(1/2);
            d_wake_mirror_bottom_AtoB(j,i) =
 ((d_airfoil_mirror_bottom_A(j)-z_bar_B)^2 + (b/2 + x_first +
 U_inf*delta_t*(i-1))^2)^(1/2);
            %determine distances for airfoil A wake mirroring based on
 airfoil B contributions
```

```
            d_wake_mirror_top_BtoA(j,i) = ((d_airfoil_mirror_top_B(j)-
z_bar_A)^2 + (b/2 + x_first + U_inf*delta_t*(i-1))^2)^(1/2);
            d_wake_mirror_bottom_BtoA(j,i) =
 ((d_airfoil_mirror_bottom_B(j)-z_bar_A)^2 + (b/2 + x_first +
 U_inf*delta_t*(i-1))^2)^(1/2);
        else
            %determine distances for airfoil B wake mirroring based on
 airfoil A contributions
            d_wake_mirror_top_AtoB(j,i) = -
((d_airfoil_mirror_top_A(j)-z_bar_B)^2 + (b/2 + x_first +
 U_inf*delta_t*(i-1))^2)^(1/2);
            d_wake_mirror_bottom_AtoB(j,i) = -
((d_airfoil_mirror_bottom_A(j)-z_bar_B)^2 + (b/2 + x_first +
 U_inf*delta_t*(i-1))^2)^(1/2);
            %determine distances for airfoil A wake mirroring based on
 airfoil B contributions
            d_wake_mirror_top_BtoA(j,i) = -
((d_airfoil_mirror_top_B(j)-z_bar_A)^2 + (b/2 + x_first +
 U_inf*delta_t*(i-1))^2)^(1/2);
            d_wake_mirror_bottom_BtoA(j,i) = -
((d_airfoil_mirror_bottom_B(j)-z_bar_A)^2 + (b/2 + x_first +
 U_inf*delta_t*(i-1))^2)^(1/2);
        end
    end
end
```


Mirroring Kernal Assembly

```
%determine kernels for A then assemble wake
K_wake_mirror_top_AtoA    =   -1./(2.*pi.*d_wake_mirror_top_AtoA);
K_wake_mirror_bottom_AtoA =  -1./(2.*pi.*d_wake_mirror_bottom_AtoA);
K_wake_mirror_AtoA        =   [K_wake_mirror_top_AtoA;
 K_wake_mirror_bottom_AtoA];
K_mirror_AtoA             =   [K_airfoil_mirror_AtoA,
 K_wake_mirror_AtoA];
%account for cross effects
K_wake_mirror_top_BtoA    =   -1./(2.*pi.*d_wake_mirror_top_BtoA);
K_wake_mirror_bottom_BtoA =  -1./(2.*pi.*d_wake_mirror_bottom_BtoA);
K_wake_mirror_BtoA        =   [K_wake_mirror_top_BtoA;
 K_wake_mirror_bottom_BtoA];
K_mirror_BtoA             =   [K_airfoil_mirror_BtoA,
 K_wake_mirror_BtoA];
%combine airfoil A and airfoil B contributions on airfoil A
K_mirror_A           =   [K_mirror_AtoA,K_mirror_BtoA];

%determine kernels for B then assemble wake
K_wake_mirror_top_BtoB    =   -1./(2.*pi.*d_wake_mirror_top_BtoB);
K_wake_mirror_bottom_BtoB =  -1./(2.*pi.*d_wake_mirror_bottom_BtoB);
K_wake_mirror_BtoB        =   [K_wake_mirror_top_BtoB;
 K_wake_mirror_bottom_BtoB];
K_mirror_BtoB             =   [K_airfoil_mirror_BtoB,
 K_wake_mirror_BtoB];
%account for cross effects
```

```
K_wake_mirror_top_AtoB   =   -1./(2.*pi.*d_wake_mirror_top_AtoB);
K_wake_mirror_bottom_AtoB =  -1./(2.*pi.*d_wake_mirror_bottom_AtoB);
K_wake_mirror_AtoB       =   [K_wake_mirror_top_AtoB;
 K_wake_mirror_bottom_AtoB];
K_mirror_AtoB            =   [K_airfoil_mirror_AtoB,
 K_wake_mirror_AtoB];
%combine airfoil A and airfoil B contributions on airfoil A
K_mirror_B           =   [K_mirror_AtoB, K_mirror_BtoB];
```


Kernal Development in Test Section

```
%define x-ccordinate for airfoil vortex
K_11            =       1/(2*pi*b);

%calculate x-coordinates for wake vorticies
K_wake          =       zeros(1,n);
for i=1:n

    K_wake(i) = -1/(2*pi*(b/2+x_first+U_inf*delta_t*(i-1)));

end

%determine inter-test section mirroring kernals
K_11_AtoB = 1/(2*pi*((z_bar_B-z_bar_A)^2 + (b)^2)^(1/2));
K_11_BtoA = 1/(2*pi*((z_bar_A-z_bar_B)^2 + (b)^2)^(1/2));

for i = 1:n

        %determine distances for airfoil A wake mirroring based on
 airfoil B contributions
        K_wake_AtoB(i) = -1/(2*pi*((z_bar_B-z_bar_A)^2 + (b/2 +
 x_first + U_inf*delta_t*(i-1))^2)^(1/2));

        %determine distances for airfoil B wake mirroring based on
 airfoil A contributions
        K_wake_BtoA(i) = -1/(2*pi*((z_bar_A-z_bar_B)^2 + (b/2 +
 x_first + U_inf*delta_t*(i-1))^2)^(1/2));

end

%combine K_11 and K_wake with K_mirror for both airfoils
%account for wakes of both airfoils
K_testsection_A = [K_11 K_wake K_11_BtoA K_wake_BtoA]; %use same
 spacing in wake for each airfoil
K_testsection_B = [K_11_AtoB K_wake_AtoB K_11 K_wake];

K_comp_A = [K_testsection_A;K_mirror_A];
K_comp_A(:,n+2) = -K_comp_A(:,n+2);
K_comp_B = [K_testsection_B;K_mirror_B];
K_comp_B(:,1) = -K_comp_B(:,1);


%sum each kernel for
```

```
for i = 1:2*(n+1)
    K_A(i) = sum(K_comp_A(:,i));
    K_B(i) = sum(K_comp_B(:,i));
end
```


Matrix Setup with Mirroring

```
%initialize gamma and downwash
gamma = zeros(2*(n+1),(n+1)); %at step i=1, {gamma} = {0}
downwash = zeros(2*(n+1),(n+1)); %at step i=1, {downwash} = {0}

%form matricies
%system will be in the form: [R]{gamma(i+1)}=[S]{gamma(i)}+[T(i+1)]
R = eye(2*(n+1));
R(1,:) = K_A;
R(2,1) = 1; %[R] is like an identity matrix with the top row being
 made of Ks and the 2,1 entry equal to a 1
R(n+2,:) = K_B;
R(n+3,n+2) = 1;

S = [zeros(1,2*(n+1)-1); eye(2*(n+1)-1)];
S = [S zeros(2*(n+1),1)];
S(n+2,n+1) = 0; %[S] is like an identity matrix with an extra row and
 column of zeros
S(n+1,n+1) = aa;
S(2*(n+1),2*(n+1)) = aa;
```

Airfoil Behavior in Flow Field with Mirroring

```
%solve for gamma
A = R\S; %{gamma(i+1)=[A]*{gamma(i)}+{u(i+1)}

%define loop
for i=1:n

    %downwash
        %downwash(1,i+1) = U_inf*1; %downwash for every time other
 than i=1,n+1 equals U_inf (step input)
        %downwash(1,i+1) = U_inf*(h_d(i)/U_inf + alpha(i));
        downwash(1,i+1) =
 U_inf*alpha0*sin(w_n*t(i))+w_n*b*alpha0*cos(w_n*t(i));
        downwash(n+2,i+1) = downwash(1,i+1);

    %calculate circulation
        gamma(:,i+1)=   A*gamma(:,i) + R\downwash(:,i+1); %solve for
 gamma(i+1)

    %lift for airfoil A(from DeLaurier)
        N_ci_A(i)     =   rho*U_inf*gamma(1,i+1);
        N_ai_A(i)     =   1/4*rho*pi*(2*b)^2*(h_dd(i) +
 U_inf*alpha_d(i));
        N_A(i)        =   N_ci_A(i) + N_ai_A(i);
```

```
    %moment for airfoil A(from DeLaurier)
        M_ai_A(i)     =   -(1/128)*rho*pi*(2*b)^4*alpha_dd(i);
        M_ti_A(i)     =   -(1/16)*rho*pi*(2*b)^3*U_inf*alpha_d(i);
        M_c_A(i)      =   1/4*(2*b)*N_ci_A(i);
        M_A(i)        =   M_c_A(i) + M_ai_A(i) + M_ti_A(i);

    %lift for airfoil B(from DeLaurier)
        N_ci_B(i)     =   rho*U_inf*gamma(n+2,i+1);
        N_ai_B(i)     =   1/4*rho*pi*(2*b)^2*(h_dd(i) +
 U_inf*alpha_d(i));
        N_B(i)        =   N_ci_B(i) + N_ai_B(i);

    %moment for airfoil B(from DeLaurier)
        M_ai_B(i)     =   -(1/128)*rho*pi*(2*b)^4*alpha_dd(i);
        M_ti_B(i)     =   -(1/16)*rho*pi*(2*b)^3*U_inf*alpha_d(i);
        M_c_B(i)      =   1/4*(2*b)*N_ci_B(i);
        M_B(i)        =   M_c_B(i) + M_ai_B(i) + M_ti_B(i);

        check4 = check4 + 1;

end

gamma;
```

plot and check

```
% % % figure(4)
% % % sgtitle('Theodorsen-VLM Matching for Double-Airfoil Gust
 System')
% % % % subplot(2,1,1)
% % % % plot(w_n*t,-N_A)%,'s')
% % % hold on
% % % % plot(w_n*t,-N_B)%,'o')
% % % hold on
% % % % plot(w_n*t,L_theodorsen)
% % % hold off
% % % title('Lift')
% % % legend('Airfoil A - VLM','Airfoil B -
 VLM','Theodorsen','location','southeast')
% % % xlim([w_n*t(1) w_n*t(n)])
% % % xlabel('\omegat')

% % % % subplot(2,1,2)
% % % % plot(w_n*t,M_A)%,'s')
% % % hold on
% % % % plot(w_n*t,M_B)%,'o')
% % % hold on
% % % % plot(w_n*t,M_theodorsen)
% % % hold off
% % % title('Moment')
% % % legend('Airfoil A - VLM','Airfoil B - VLM','Theodorsen')
% % % xlim([w_n*t(1) w_n*t(n)])
% % % xlabel('\omegat')
```

Compare Dual Vane Simulations to CFD Simulations

import excel sheet

```
T_2 =
 table2array(readtable('two_vanes_compare_2period.xlsx','Range','A1:C64'));
time_2 = T_2(:,1);
airfoil1_2 = T_2(:,2);
airfoil2_2 = T_2(:,3);

% % % figure(5)
% % % % plot(w_n*t,-N_A,'-')
% % % hold on
% % % % plot(w_n*t,-N_B,'-')
% % % % hold on
% % % % plot(w_n*t,L_theodorsen)
% % % hold on
% % % % plot(time_2*w_n,airfoil1_2,'s')
% % % hold on
% % % % plot(time_2*w_n,airfoil2_2,'s')
% % % hold off
% % % %title('Gust Generation System Simulations - Theodorsen, VLM and
 CFD')
% % % legend('Airfoil A - VLM','Airfoil B - VLM','Airfoil A -
 CFD','Airfoil B - CFD','location','southeast')
% % % xlim([w_n*time_2(1) w_n*time_2(end)])
% % % xlabel('\omegat')
% % % ylabel('normalized lift force (N/m)')

% import excel sheet
T_22 = table2array(readtable('twovane_12hz.xlsx','Range','A1:C65'));
time_22 = T_22(:,1);
airfoil1_22 = T_22(:,2);
airfoil2_22 = T_22(:,3);

% % % figure(15)
% % % % plot(w_n*t,-N_A,'-')
% % % hold on
% % % % plot(w_n*t,-N_B,'-')
% % % % hold on
% % % % plot(w_n*t,L_theodorsen)
% % % hold on
% % % % plot(time_22*w_n,airfoil1_22,'s')
% % % hold on
% % % % plot(time_22*w_n,airfoil2_22,'s')
% % % hold off
% % % %title('Gust Generation System Simulations - Theodorsen, VLM and
 CFD')
% % % legend('Airfoil A - VLM','Airfoil B - VLM','Airfoil A -
 CFD','Airfoil B - CFD','location','southeast')
% % % xlim([w_n*time_22(1) w_n*time_22(end)])
```

```
% % % xlabel('\omegat')
% % % ylabel('normalized lift force (N/m)')
```

```
Warning: Table variable names were modified to make them valid MATLAB
identifiers. The original names are saved in the VariableDescriptions
 property.
Warning: Table variable names were modified to make them valid MATLAB
identifiers. The original names are saved in the VariableDescriptions
 property.
```

```
end
```

Published with MATLAB® R2019a

158

Appendix B

MESHING CHECKLIST

1. Select "File:New Project" and name new project
2. Select "File:Import Model"
3. Choose .iges file from documents
4. Under "Import Model" menu, click "OK"
5. Under "Geometry" tab on upper task bar, select "Repair Geometry" (box with wrench icon)
6. Under "Repair Geometry" menu, click "OK"
7. Right click on "Parts", under "Model" tree, and select "Create Part"
8. Select part of model to be considered by clicking on yellow mouse icon in "Create Part by Selection" box
9. Name part as desired and press "OK"
10. Repeat process until all parts of model have been named
11. Under "Blocking" tab on upper task bar, select "Create Block" (see-through box with pencil icon)
12. Click on "Entities" selector (brown box with white mouse icon) in "Initialize Blocks" box

13. Under "Select geometry" pop-up, click on "Select all appropriate objects" (four black arrow icon)

14. Under "Blocking" tab on upper task bar, select "Split Block"

15. Under "Split Block" menu, click on white mouse icon corresponding to "Edge"

16. Select edge to be split and arrange blocking line as desired by dragging line across model

17. When split line is in correct position, click center button of mouse to accept location

18. Repeat process for additional split lines

19. Select "Ogrid Block" under "Split Block" menu

20. Click on "Select face" icon with plus sign and white mouse

21. Choose face of interest on model and select both front and back face by clicking on respective selection rectangle

22. Press center button of mouse to accept selection then press "OK"

23. Ensure "Points" is selected under "Geometry" in "Model" tree

24. Under "Blocking" tab on upper task bar, select "Associate" (yellow hand pointing icon)

25. Click on "Associate vertex" in "Edit Associations" box under "Blocking Associations" menu

26. Click on vertex of blocked rectangle then click on closest vertex on test object (airfoil in the given case)

27. Repeat for each vertex of the front and back face rectangles (eight associations total) for each test object

28. Under "Blocking" tab on upper task bar, select "Delete Block" (see-through box with red "X" icon)

29. Click on white mouse with brown box inside clear box icon and select block inside each test object

30. Press mouse center button to delete block(s)

31. Under "Blocking" tab in upper task bar, select "Associations"

32. In "Blocking Associations" menu, click on "Associate Edge to Curve"

33. Handling each test object individually, select the blocking lines visible inside the test object

34. Press the mouse center button to accept the selection

35. Select the corresponding boundaries of the test object

36. Press the mouse center button to accept the selection and associate the two sets

37. Repeat for all other test objects (if applicable)

38. Under "Blocking" tab on upper task bar, select "Pre-Mesh Param"

39. In "Pre-Mesh Params" menu click on "Edge Params" in "Meshing Parameters" box

40. In expanded "Pre-Mesh Params" menu, check "Copy Parameters" box and ensure "To All Parallel Edges" in "Copy" box is queued

41. Click on white mouse icon corresponding to "Edge" in "Meshing Parameters" box, select desired edge and denote number of nodes and spacing desired

42. Repeat for each edge as required

43. Under "Blocking" tab in upper task bar, click on "Edit Edge" (red pliers with wire icon)

44. In "Edit Edge" menu, click on "Split Edge" (axe with wire icon) in "Edit Edge(s)" box

45. Under "Model" tree in "Blocking" section, check "Pre-Mesh" box and click on "Re-mesh out of date parts"

46. Right click on "Pre-Mesh" and select "Convert to Unstruct Mesh"

47. Under "Output Mesh" tab in upper task bar, click on "Select solver" (red toolbox icon)

48. Set "Output Solver" option, in "Solver Setup" menu, to "ANSYS CFX"

49. Under "Output Mesh" tab in upper task bar, click on "Write input" (rubics cube with floppy disk icon)

50. Review output file, save project and press "Done"

Appendix C

STEADY STATE CFD CHECKLIST

1. Search "CFX" in "Search Windows" and click on "CFX"

2. In "CFX Launcher", choose "CFX-Pre"

3. Select File:New Case, pick "General" and press "OK"

4. On side menu, right click "Mesh"

5. Select "Import Mesh: ICEM CFD", search through documents and choose desired .cfx5 file (mesh file)

6. Double click "Default Domain" on side menu

7. In "Basic Settings" of "Default Domain" menu, set "Fluid 1 Material" to "Air at 25 C", set "Reference Pressure" to "0 atm"

8. In "Fluid Models" of "Default Domain" menu, set "Heat Transfer Option" to "None", set "Turbulence Option" to "Shear Stress Transport"

9. If necessary, right click on "Mesh" on side menu, select "Transform Mesh" and scale as desired

10. Right click on "Default Domain" on side menu, select "Insert" and choose "Boundary"

11. Name boundary as desired, set required "Boundary Type" and select corresponding "Location" from drop-down menu

12. For inlet, under "Boundary Details", set "Mass And Momentum Option" to "Normal Speed" and "Mass Momentum Normal Speed" to "60 m/s"

13. For outlet, under "Boundary Details", set "Mass And Momentum Option" to "Average Static Pressure" and "Mass Momentum Relative Pressure" to "99195 Pa"

14. For walls and airfoils, do not modify "Boundary Details"

15. Right click on "Interfaces" on side menu, select "Insert" and choose "Domain Interface"

16. Name interface as desired

17. Under Basic Settings, for "Interface Side 1 Region List", select top face and for "Interface Side 2 Region List", select bottom face

18. Set "Interface Models Option" to "Translational Periodicity"

19. Double click "Solver Control" on side menu

20. Under "Basic Settings", set "Turbulence Numerics Option" to "High Resolution", set "Convergence Control Max. Iterations" to "250", set "Convergence Criteria Residual Target" to "1.E-6"

21. Double click on "Analysis Type" on side menu and verify "Steady State" option is selected

22. Click on "Write Solver Input File" and save file in desired location

23. Open "CFX-Solver Manager" in "CFX Launcher"

24. In "CFX-Solver Manager", select "File:Define Run"

25. Click on yellow open folder icon corresponding to "Solver Input Control" and select desired file

26. In "Parallel Environment" box, set "Run Mode" to "IBM MPI Local Parallel" and set "Partitions" to "20" using plus button "Working Directory" in "Run Environment" box is correct then press "Start Run"

Appendix D

UNSTEADY CFD CHECKLIST

1. Execute steady state run

2. In "CFX-Pre", left click on "Expressions" on the left menu, select "Insert" and choose "Expression"

3. Name and define expressions as shown below:

   f=100 [Hz]

   Tp = 1/f

   ts = Tp/32

   ang = 10 [deg]

   ai = ang*sin(2*pi*t/Tp)

4. Double click on "Analysis Type" on the left menu and change "Option" to "Transient" in "Basic Settings"

5. Set "Time Duration Total Time" to "Tp*4" and set "Time Steps Timesteps" to "ts"

6. Double click on "Default Domain" on left menu

7. In "Basic Settings" of "Default Domain", set "Mesh Deformation Option" to "Regions of Motion Specified"

8. Press plus sign to expand "Mesh Motion Model"

9. In "Mesh Deformation" box, change "Mesh Stiffness Model" to "Value" and set "Mesh Stiffness" to "1 $[m^5 s^(- 1)]$/Volume of Finite Volumes"

10. Double click on "AIRFOIL" in "Default Domain" tree on left menu

11. Under "Boundary Details" of "AIRFOIL", set "Mesh Motion Option" to "Specified Location"

12. Define location of airfoil motion in "Location" box for each direction

13. For one airfoil motion:

    "X Component": Initial X*cos(ai) - Initial Y*sin(ai)

    "Y Component": Initial X*sin(ai) + Initial Y*cos(ai)

    "Z Component": Initial Z

14. Double click on red error messages in bottom dialogue box, press "Apply" to apply mesh motion changes

15. Double click on "Output Control" under "Solver" tree on left menu

16. Select "Trn Results" tab in "Output Control"

17. Next to "Transient Results" dialogue box, click on white paper with yellow star icon

18. In "Transient Results 1" box, change "Output Frequency Option" to "Every Timestep"

19. Select "Monitor" in "Output Control"

20. Next to "Monitor Points and Expressions" dialogue box, click on white paper with yellow star icon and name monitor point "AngleVariation"

21. In "AngleVariation" box, change "AngleVariation Option" to "Expression" and set "Angle Variation Expression Value" to "ai"

22. Select "Write Solver Input File" as before in steady-state simulation run

23. In "CFX-Solver Manager", define run

24. Load unsteady file into "Solver Input File" and change "Run Mode" and "Partitions" as before

25. Under Global Run Settings, select "Initial Values" tab and click on "Initial Values Specification" box

26. In "Initial Values 1 Settings" box, load the steady-state results (.res) file into the "File Name" dialogue box

27. Press "Start Run" to run simulation