



8-2012

Computational Analysis of a Wing Oscillator

Ryne Derrick Radermacher

Western Michigan University, rhyno466@gmail.com

Follow this and additional works at: http://scholarworks.wmich.edu/masters_theses



Part of the [Aerodynamics and Fluid Mechanics Commons](#)

Recommended Citation

Radermacher, Ryne Derrick, "Computational Analysis of a Wing Oscillator" (2012). *Master's Theses*. 30.
http://scholarworks.wmich.edu/masters_theses/30

This Masters Thesis-Open Access is brought to you for free and open access by the Graduate College at ScholarWorks at WMU. It has been accepted for inclusion in Master's Theses by an authorized administrator of ScholarWorks at WMU. For more information, please contact maira.bundza@wmich.edu.



COMPUTATIONAL ANALYSIS OF A WING OSCILLATOR

by

Ryne Derrick Radermacher

A Thesis

Submitted to the

Faculty of the Graduate College

in partial fulfillment of the

requirements for the

Degree of Master of Science of Engineering (Mechanical)

Department of Mechanical and Aeronautical Engineering

Advisor: William W. Liou, Ph.D.

Western Michigan University

Kalamazoo, Michigan

August 2012

THE GRADUATE COLLEGE
WESTERN MICHIGAN UNIVERSITY
KALAMAZOO, MICHIGAN

Date 7/10/12

WE HEREBY APPROVE THE THESIS SUBMITTED BY

RYNE DERRICK RADERMACHER

ENTITLED COMPUTATIONAL ANALYSIS OF A WING OSCILLATOR

AS PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

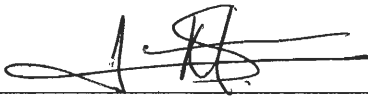
DEGREE OF Master of Science in Engineering (Mechanical)

Mechanical and Aeronautical Engineering
(Department)


Thesis Committee Chair

Mechanical Engineering
(Program)


Thesis Committee Member


Thesis Committee Member

APPROVED


Dean of The Graduate College

Date August 2012

COMPUTATIONAL ANALYSIS OF A WING OSCILLATOR

Ryne Derrick Radermacher, M.S.E.

Western Michigan University, 2012

An analysis of a newly proposed wind power extraction device called a wing oscillator was performed using computational fluid dynamics (CFD). A wing oscillator is a device that consists of two airfoils attached to a central pivot by means of a frame or shaft. Oscillatory motion is produced by controlling the angle of attack of the airfoils as a fluid flow is passed over the device. A robust mathematical scheme was developed to investigate the performance of the wing oscillator and was hooked to the CFD software package FLUENT through User Defined Functions (UDFs). Using the dynamic meshing analysis methods available in FLUENT in conjunction with the developed mathematical scheme, a time accurate dynamic analysis of the wing oscillator was performed. Post processing of the various analyses was completed using the built in FLUENT post processing functions as well as reading the solution output data into a custom MATLAB code. A parametric study was performed using a combination of different system variables such as maximum system angle, airfoil angle of attack, and spring constant. The aerodynamic characteristics of the various cases were analyzed and the system performance was compared to analyze the effect of the individual parameters.

© 2012 Ryne Derrick Radermacher

TABLE OF CONTENTS

LIST OF TABLES.....	iii
LIST OF FIGURES.....	iv
CHAPTER 1 – INTRODUCTION AND BACKGROUND	1
CHAPTER 2 – NACA 0012 VALIDATION	4
2.1 – INTRODUCTION	4
2.2 – MESHING.....	4
2.3 – SOLUTION SETUP	8
2.4 – VALIDATION RESULTS.....	10
CHAPTER 3 – NACA 0015 VALIDATION	17
3.1 – INTRODUCTION	17
3.2 – VALIDATION RESULTS.....	18
CHAPTER 4 – DYNAMIC ANALYSIS	20
4.1 - GEOMETRY.....	20
4.2 – MESHING.....	21
4.3 - FLOW DRIVEN THEORY	25
4.3.1- MATHEMATICAL SCHEME.....	26
4.3.2- USER DEFINED FUNCTIONS.....	28
4.4 – VERIFICATION.....	34
4.5- DYNAMIC SOLUTION SETUP	38
4.6 – RESULTS.....	40
4.6.1 – BASE CASE RESULTS	40
4.6.2 – PARAMETRIC STUDY.....	47
CHAPTER 5- CONCLUSIONS.....	58
5.1 – RECOMMENDATIONS AND FUTURE WORK	59
REFERENCES.....	61

LIST OF TABLES

1- 0012 Validation Velocity Components	10
2- Grid Independence Validation	12
3- 0015 Validation Velocity Components	18
4- Verification Parameters.....	34
5-Error Verification	36
6- Force Verification.....	37
7- Base Case System Parameters	41
8- Parametric Study System Parameters	47
9- Parametric Study Period and Frequency	48

LIST OF FIGURES

1- Wing Oscillator (1)	2
2- NACA 0012 Airfoil Validation	5
3- 0012 Blocking Scheme	6
4- 0012 Far- Field Mesh	7
5- 0012 Near- Airfoil Mesh.....	8
6- Boundary Layer Validation.....	11
7- Boundary Layer Selection Line.....	12
8- Grid Independence Pressure Validation	13
9- 0012 Lift Validation (16)	13
10- 0012 Drag Validation (16).....	14
11- 0012 Cp Validation (17)	15
12- 0015 Far Field Mesh.....	17
13- 0015 Lift Validation (16)	18
14- 0015 Drag Validation (16).....	19
15- Wind Tunnel Model Geometry	21
16- Workbench Geometry	23
17- Edge Sizing Definitions.....	24
18- Wind Tunnel Mesh.....	24
19- Position Verification.....	35
20- System Angle Verification.....	35
21- Error Verification	36
22- Angular Velocity Verification	37
23- Airfoil Movement t=1s.....	41
24- Airfoil Movement t=2.5s.....	42
25- Base Case Wake Shedding After AoA Change	42
26- Base Case Lift Force	43
27- Base Effective AoA.....	44

LIST OF FIGURES CONTINUED

28- Base Torque	44
29- Base Case Drag Force.....	45
30- Degraded Mesh.....	46
31- Base Case Final Velocity Magnitude	47
32- Parametric System Angle.....	48
33- Parametric Angular Velocity	49
34- Parametric Torque	51
35- Parametric Effective AoA	52
36- Parametric Spring Torque.....	53
37- Parametric Error	54
38- AoA Separation	55
39- Base Separation	55
40- AoA Drag Force	56
41- Spring Separation.....	57

CHAPTER 1 – INTRODUCTION AND BACKGROUND

The concept of extracting power from the wind has been around for hundreds of years in the various forms of windmills that have evolved throughout the years into highly engineered wind turbines. Recently, these wind turbines have been at the forefront of the green energy movement and have thus seen tremendous research and modern implementation. Current wind turbines can be found in two basic forms. The first form is the traditional windmill style that employs a horizontal rotational axis, which is commonly referred to as Horizontal Axis Wind Turbines (HAWT). The second form is a more modern design whereby the rotational axis is vertical (VAWT).

Although many recent research efforts have been focused on these existing wind turbine designs, there are still some inherent problems that have yet to be overcome (1). First, due to the rotational nature of the motion of the turbines there are many dynamic loads interacting with the blades and structure. These can come from the body weight of the blades themselves as the blades move through the rotation and the direction of the resolved force changes, or the dynamic centrifugal force caused by varying wind speeds. All of this dynamic loading can cause fatigue in the system, which could lead to failure of the wind turbine or power generation components.

Some wind turbines have the ability to control the pitch of the blades to adjust for varying wind speed, however, others do not offer this functionality and thus cannot be operated in all wind conditions. If the wind speed becomes too high, these static blade wind turbines could over-spin which could cause damage or failure to the system. Therefore, some wind turbines are equipped with a brake that can stop the rotational motion and subsequent power generation. This stoppage of power generation is inherently inefficient and a large problem if society would ever rely on these wind turbines for power generation.

An issue presented by Shimizu with these conventional wind turbines is the aerodynamic noise caused by the blades as they rotate through the air at high speed. This noise is especially problematic with large scale HAWT that can demonstrate very high blade tip speeds. Another issue Shimizu presented was that at low Reynolds numbers, the efficiency of these conventional designs can suffer due to the designs being susceptible to laminar separation. (2)

The current research represents an attempt to overcome some of these issues with the traditional designs. The current design is based off the design that was proposed by Liu and consists of two NACA 0015 airfoils attached to a central pivot by means of a frame or shaft. A spring used for system damping is attached to the frame at the central pivot. Rotation about this central pivot is achieved by controlling the angle of attack of the leading and trailing airfoils while airflow, wind, is passed over the device. When a maximum system angle is reached, the AoA of the airfoils is reversed and the device begins to rotate in the opposite direction. In this way, oscillatory motion is achieved and repeated to generate power. Some of the advantages of

this system are the simplicity, adaptability, and lack of significant differential loadings. The design is simple in that elaborately tuned airfoils are not needed; a symmetrical airfoil with a constant span wise chord is preferred. The system is adaptable, meaning it will scale easily depending on the application, and could even potentially be used in rivers. The fact that there is no significant span wise differential loading and the two airfoils are countering each other means the frame support system can be simple. (1)

To extract power from this system, a transmission system that includes a linear gear crank, gearbox and generator is attached near the pivot point. The purpose of this linear gear system and gearbox is to convert the oscillator motion into rotational motion in one direction, which is a requirement for the generator. This transmission system will not be considered in this study, as it would introduce significantly more design questions that are beyond the scope of this study. Figure 1 below shows a conceptual mockup of the wing oscillator.

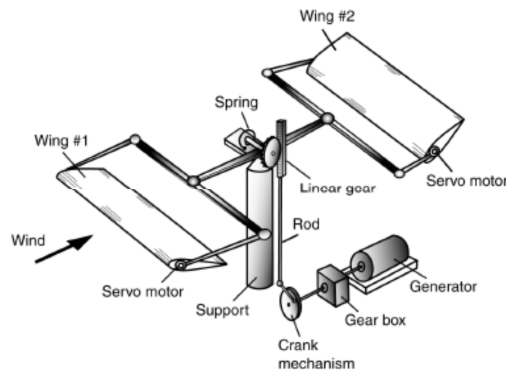


Figure 1- Wing Oscillator (1)

The concept of capturing wind energy by use of an oscillating airfoil has been explored by McKinney whose focus was a single airfoil oscillating vertically through a complex electro-mechanical system. McKinney's Wingmill design overcame some limitations of previous similar oscillating designs that focused more on flapping wing motion similar to a bird. The Wingmill employed rectangular planform that utilized whole wing motion to eliminate the need for any span wise twisting. These features minimized differential span loading and allowed for a more simplified analytical analysis approach before any experimentation was conducted. The experimental results provided an efficiency of 28.3% and concluded that the motion and power generation come mainly from the normal forces rather than the leading edge suction over the airfoil. McKinney used a NACA 0012 airfoil and kept the maximum angle of attack (AoA) at 15.5 degrees, which is normally just beyond the static stall angle for the 0012. It was found, however, that there was a possibility of stall delay due to the dynamic motion of the airfoil. (3)

Shimizu used a multi-objective design study to analyze a large number of CFD simulations of an oscillating airfoil. This multidimensional analysis led to the optimization of both power output and efficiency of what Shimizu referred to as a flapping wing power generator. The CFD was conducted using an unsteady compressible finite difference Navier-

Stokes code. The two dimensional, ten chord far field computational domain captured the NACA 0012 airfoil with a C- type grid of 280 by 80 points with 200 points residing on the airfoil. The study showed a tradeoff between power output and efficiency with a maximum efficiency shown to be 48.6%. The lower bound of the efficiency was 36.5% with values for different motions lying all throughout the bounds. The study also concluded that when efficiency was the emphasized design objective, the heaving or vertical translation should be small with a high oscillation frequency. (2)

Kinsey explored extracting wind power from an oscillating airfoil through a parametric study, which focused on the pitching amplitude and nondimensional frequency. The computational analysis was completed using a NACA 0015 airfoil with an unstructured, rotating, non- conformal mesh in FLUENT. This mesh did not have to deform through the analysis; instead, the mesh rotated along the non- conformal boundary to control the pitching while the heaving was controlled through dynamically changing the inlet velocity magnitude and direction. This dynamic scheme allowed for second order temporal discretization that helped decrease the numerical diffusion caused by larger time steps. The highest numerical efficiency was found to be approximately 34% at a nondimensional frequency of .15 and a pitching amplitude of 70-80 degrees. (4)

Due to the wing oscillator having two airfoils in the flow, it is necessary to assume there will be an interaction between the wake from the leading airfoil and the trailing airfoil. Although Michelsen used steady positions, the observed effect was that lift and drag would both slightly decrease for the trailing airfoil with the greatest decrease being observed when the leading airfoil was slightly above the trailing airfoil. The conclusion for the decreased forces was that the leading airfoil created a downwash that was experienced by the trailing airfoil. Michelsen also observed that the higher the AoA of the leading airfoil, the greater the effect on the trailing airfoil due to a larger wake being created. (5)

In the current research, a dynamic analysis of this wing oscillator is performed using the computational fluid dynamics software package FLUENT. The analysis is a two dimensional turbulent, transient, dynamic mesh analysis, with the mesh motion driven by a mathematical scheme which uses the calculated forces acting on the airfoils in the computational domain at every time step. User Defined Functions (UDFs), which are written in the C programming language and linked into FLUENT, perform the calculations that drive the motion and incorporate the mathematical scheme. By varying the system variables, many different cases were analyzed and compared.

The CFD solution setup and meshing parameters that were used for the dynamic analysis were validated through a steady state analysis and compared to published experimental data. The dynamic analysis scheme was verified using a combination of a MATLAB test program and a calculations test program written in the C programming language. All of the results presented in the current research were post processed using a custom written MATLAB code.

CHAPTER 2 – NACA 0012 VALIDATION

2.1 – INTRODUCTION

The validation of a NACA 0012 airfoil was performed to show the computational setup and methodologies provided an accurate solution. These methodologies and settings could then be applied to other geometries with confidence in an accurate solution being calculated. The NACA 0012 was chosen for the initial validation due to the widely available experimental data to compare with the computational results and the similarity to the NACA 0015 airfoil that is used on the wing oscillator. A boundary layer validation was completed to show that the meshing scheme would resolve the boundary layer thoroughly as required by the turbulence model. The coefficient of lift (Cl) is compared to the airfoil AoA and the coefficient of drag (Cd) is compared to Cl. This data was available experimentally, thus it was convenient to reproduce these data sets computationally. The coefficient of pressure (Cp) at 5 degrees AoA was compared to experimental data. A grid independence study was also completed.

2.2 – MESHING

The meshing scheme used for this validation was a two dimensional fully structured C-type far field grid. The meshing software used was ICEM CFD, which provides extensive structured meshing controls. The first step in the meshing process is to import the curves that make up the top and bottom surfaces of the airfoil. The curves are created from a set of coordinate points. The coordinate point files for NACA airfoils were generated through an online application called JavaFoil, which generated the airfoil coordinate files for any number of specified points (6). A validation of the JavaFoil points was completed using MATLAB and the airfoil points generated through JavaFoil were compared to the airfoils calculated by the NACA equation. Equation 1 is shown below and has been modified from the original NACA equation to provide a closed trailing edge, by changing the fourth order coefficient such that all the coefficients sum to zero (7).

$$y = \pm \frac{0.12}{0.2} (0.2969\sqrt{x} - 0.126x - 0.35160x^2 + 0.2843x^3 - 0.1036x^4)$$

Equation 1- 0012 Airfoil (7)

The results of the comparison are shown in the Figure 2 below. Note that the airfoil is distorted from its actual shape due to the scaling of the axes. The 300 point JavaFoil approximation matches the curve created by the NACA equation over the entirety of the chord length. Minimal error can be noticed along the top and bottom surfaces from the maximum thickness to just before the trailing edge. This error was concluded to be within the acceptable range and the 300 point coordinate file to be sufficient in approximating the airfoil shape.

It was found experimentally through a series of poor results that the internal ICEM formatted point data imported did not properly resolve the high curvature leading edge region of the airfoil. This error was present when coordinate files of 100 points to 10,000 points were

used. The fact that the error was present in such a wide range of coordinate file sizes lead to the conclusion that the error was coming from the way ICEM splined the curves through the points, rather than the coordinate files. To overcome the poor curves create by ICEM, the coordinate files were instead imported into SolidWorks. Solidworks was able to create curves without the spline error using the same 300-point coordinate file. These curves could then be exported by SolidWorks as IGES geometry files and imported into ICEM without the leading edge spline error. The airfoil curves were of unit length with the leading edge located at the origin and the trailing edge extending in the positive X direction.

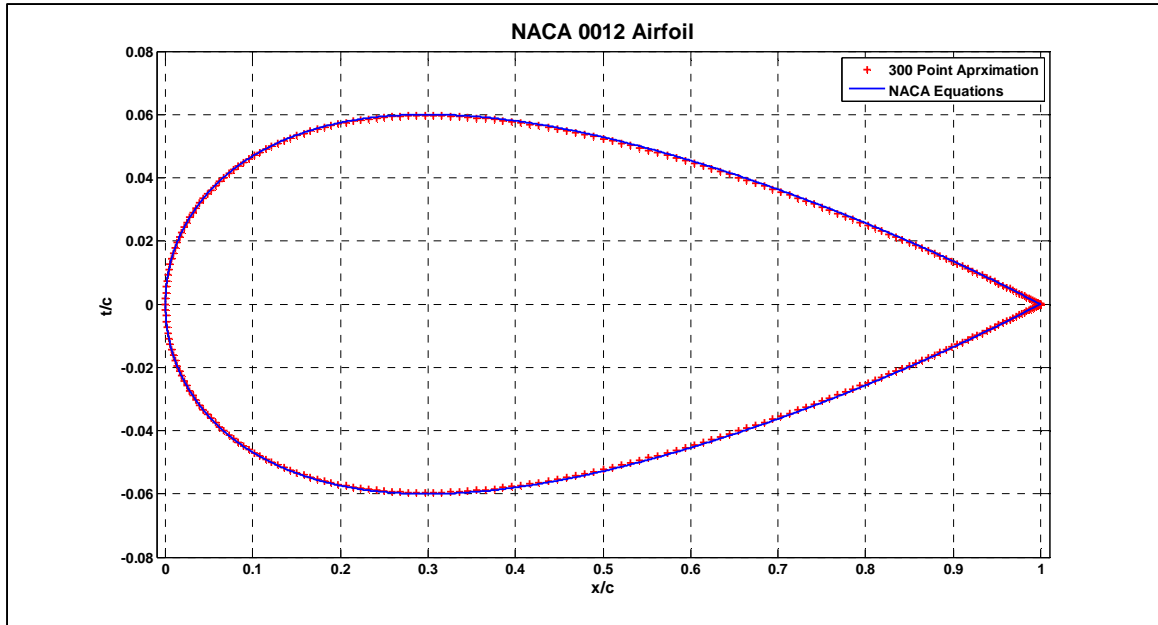


Figure 2- NACA 0012 Airfoil Validation

The computational domain was a C-type far field extending 12 chord lengths from the airfoil in all directions. The domain consists of four edges, a curved edge in front of the leading edge of the airfoil, top and bottom edges that are parallel to the chord line of the airfoil, and a vertical edge behind the trailing edge. The generation of these edges comes from first defining corner points and then connecting them with either an arc or straight edges. The top and bottom edges are combined with the curved edge to form an inlet part, and the vertical edge forms the outlet part. The grouping of these edges into parts is important for the boundary condition definition phase. A fluid surface part is created between the extents of the domain and overlaps the airfoil curves.

Once the geometry is created and grouped into the appropriate parts, the blocking meshing scheme can be implemented. ICEM uses a blocking scheme that breaks the domain into smaller blocks that are associated to the geometry and provide precise control of the mesh. The initial blocking is a large rectangle that encompasses the entire computational domain. The edges of this rectangle are associated to the edges of the domain. This association means the mesh definition of the block will be used on the associated edges. In this manner, a straight

block edge can define the meshing parameters on a curved geometry edge and the generated mesh will follow the contour of the curve, not the block edge.

The initial block needs to be split into a number of smaller blocks in order to capture the airfoil geometry. The first split is called an orthogonal, or Ogrid, split and is used to define mesh orthogonally to the airfoil. This orthogonality to the airfoil is used to minimize mesh distortion and helps create a structured boundary layer with sufficient resolution. The Ogrid split creates multiple blocks, the most important of which is used to define the airfoil. Mesh is created through all blocked areas of the domain, so the block that defines the airfoil is deleted. The top and bottom edges of the deleted block are associated to the top and bottom of the airfoil respectively. The two block corner points near the trailing edge are combined and the resulting triangular block is collapsed such that all that remains is an edge extending from the trailing edge of the airfoil to the outlet domain edge. Other, linear, splits were made in the blocks to create edges to control the mesh orientation and sizing in critical locations. The final blocked geometry is shown in Figure 3 below.

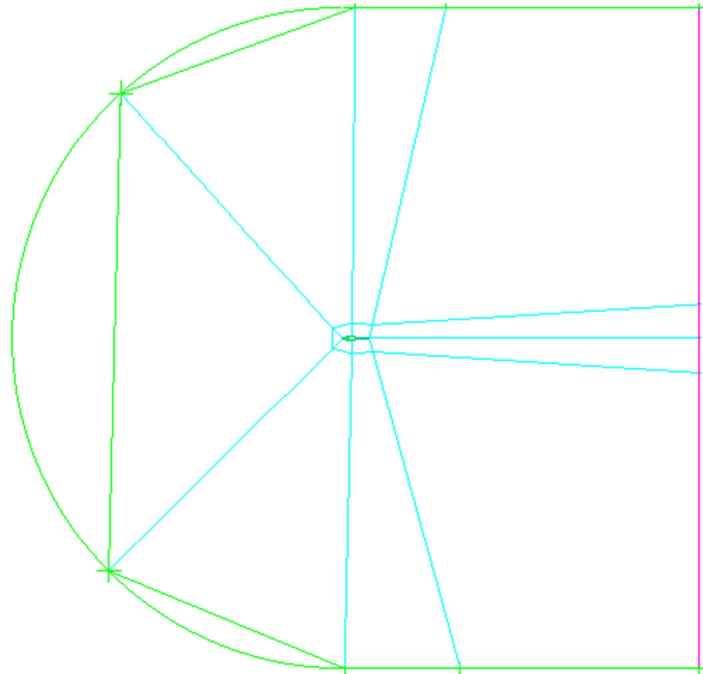


Figure 3- 0012 Blocking Scheme

The blue lines in Figure 3 above represent block edges that are not associated to any geometry. The straight green lines near the inlet, the green curve, are also blocked edges, but are associated to the inlet geometry. This color scheme provides a visual representation of what edges are associated to geometry, which can be important if the geometry is complex. The outlet is represented by the vertical pink line and has blocked edges associated with it, but they are not visible due to overlapping. Care was taken to ensure the blocked edges extending from the airfoil to the edges of the domain were as perpendicular as possible to the airfoil surface to

ensure the cells in the near wall region are as rectangular as possible. Uniform and rectangular near wall cells are necessary to provide an accurate boundary layer computation.

Mesh sizing was then applied to the blocked edges. The most critical area for the mesh sizing is the near wall region of the airfoil. The turbulence model requires the nondimensional wall distance, y^+ , of the first cell to be on the order of 1 and the boundary layer must be resolved by at least 10-20 cells (8) (9). Meeting these requirements will be proven and validated later in this chapter, but the y^+ requirement is heavily based off the initial cell height and the boundary layer resolution is based heavily off the growth rate of the near airfoil cells. Through experimentation, an initial cell height of 0.00015 meters allowed the solution to meet the y^+ requirement. The cell growth rate in the near wall region was set to 1.05, which was the smallest growth rate ICEM was capable of producing for this particular mesh. The total number of cells making up the airfoil surface was 235 for the base mesh and was varied for the grid independence study shown later in this chapter.

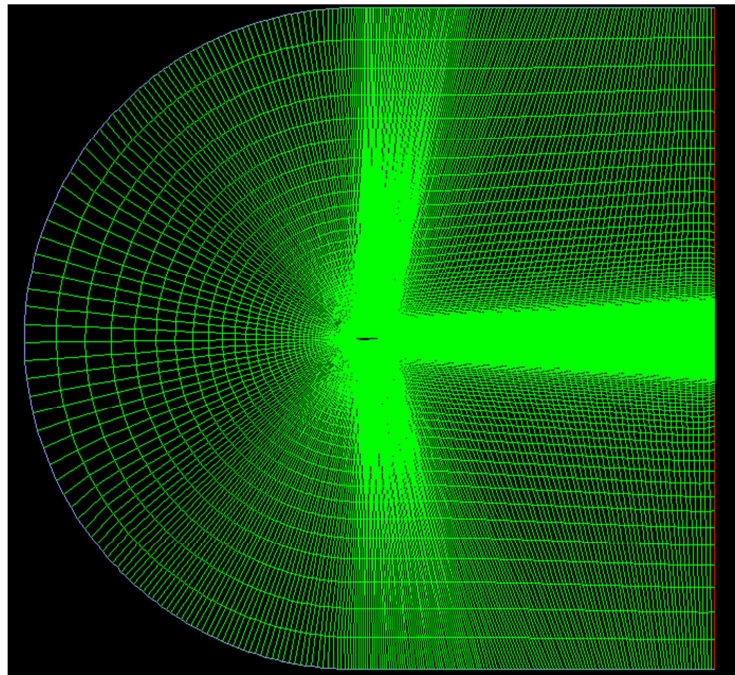


Figure 4- 0012 Far- Field Mesh

The rest of the mesh sizing came from attempting to create smooth transitions of mesh size between the blocks. Smooth transitions are required in order for the computation to provide accurate results. Having too fast of transitions or major discontinuities in cell size can introduce errors into the computation, such as numerical diffusion. One of the most important regions to have smooth transitions is, again, the near airfoil region and the vicinity moving away from the airfoil. This was accomplished by controlling the number of cells on the blocked edges moving away from the airfoil, as well as matching the cell sizes on adjacent block edges and using a growth rate of 1.1.

The mesh sizing in the wake region behind the airfoil is strongly influence by the boundary layer sizing. For computational validity, it is important for the wake region to be well resolved. The boundary layer spacing was slightly diffused to a uniform spacing on the outlet. This seemed to reduce computational errors that would occur by having so many high aspect ratio cells located in the wake, as well as allowing the ICEM smoothing algorithm to properly smooth the mesh, instead of tangling the cells when no diffusion was used.

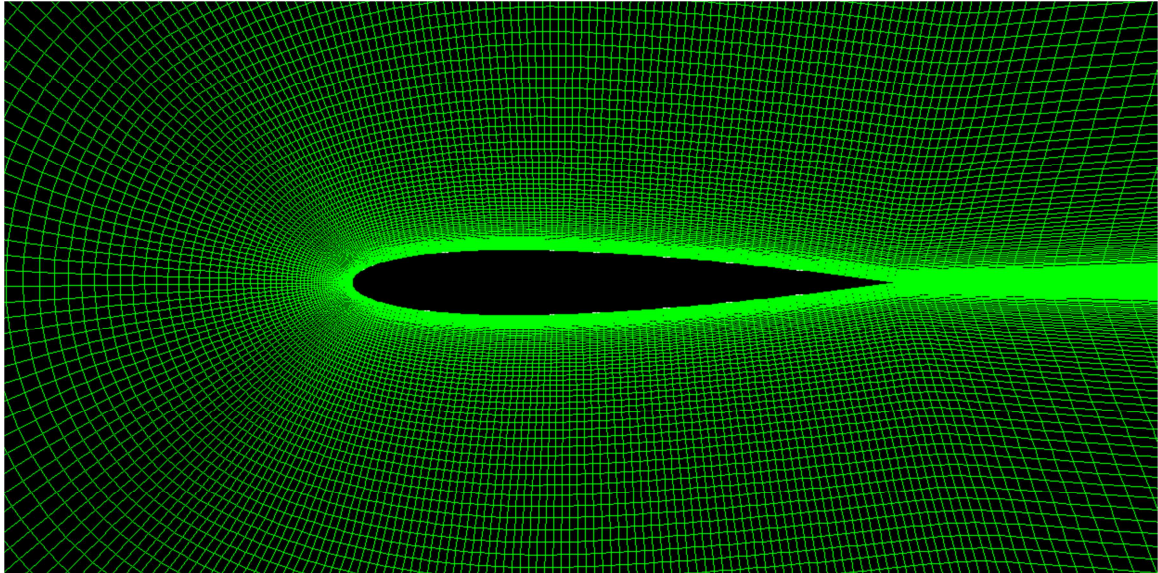


Figure 5- 0012 Near- Airfoil Mesh

The total number of cells created for the base validation mesh was 43,279. After the mesh was created, a Laplacian smoothing algorithm was employed for 10 iterations. This smoothing further assisted the smooth cell transitions and added some slight curvature to the mesh in regions when two blocks would come together at an angle. The final base mesh and near airfoil region meshes are shown in Figure 4 and Figure 5 above respectively.

2.3 – SOLUTION SETUP

The solution setup used in the validation is critical because a similar setup was used for the dynamic analysis. The validation cases were run using the steady, two dimensional, double precision, pressure based FLUENT solver. To decrease the computational time required for each case, parallel computation was used. Steady state was chosen over transient because the experimental data came from wind tunnel testing and was not presented as time dependent. Steady state calculations also reduce computational time.

One of the most important aspects of the computational setup is the choice in turbulence model. A turbulence model is required as it is unfeasible to resolve all of the true turbulent scales in space and time. A model is used to approximate, or filter out, parts of the turbulence. This is accomplished by using an averaging scheme called Reynolds Averaging, and when applied to the governing equations, is called the Reynolds Averaged Navier Stokes (RANS)

equations. RANS modeling eliminates the turbulent structures from the computation and averaged pressure and velocities can be obtained. The RANS model, however, adds unknown terms to the transport equations, which must be resolved by adding a turbulence model. This is generally referred to as turbulence closure.

The turbulence model used in the validation, and subsequently the dynamic analysis, was the k-omega shear- stress transport (SST) model. This robust model is actually a combination of the standard k-omega model and the k-epsilon model. This combination takes advantage of the strengths of both models. The standard k-omega model provides an accurate formulation in the near airfoil region, while the standard k-epsilon model is more accurate in the far field regions. This blending makes the k-omega SST model valid for a wide range of flows and is recommended for airfoils. The k-epsilon model must first be transformed into a k-omega formulation, and then a blending function is used to activate the k-omega model in the near wall region and the transformed k-epsilon in the far field region. When using this model, the flow is assumed to be turbulent everywhere, and does not provide a transition from laminar to turbulent flow. This assumption has the potential of causing some error between the computational and experimental data. A turbulent boundary layer plays a significant role in the formulation of drag, thus the error can be most noticed in the comparison of drag forces. (9)

The fluid was set to air with a density of 1.225 kilograms per meter cubed and a viscosity of $1.7894E^{-5}$ kilo grams per meter second. These values represent the FLUENT defaults and were used because the exact experimental conditions were unknown. The inlet of the computational domain was set to as a velocity inlet with a velocity magnitude of 2.485 meters per second. This velocity provided a chord based Reynolds number of approximately 170,000, which corresponded to the experimental data. To control the AoA of the validation cases, the velocity magnitude was split into the corresponding X and Y components. These components are shown in Table 1 below. The turbulent boundary conditions at the inlet were a turbulent intensity of 2% and a turbulent viscosity ratio of 10. These values were within the reasonable range suggested by FLUENT and were not defined due to experimental data matching as the experimental conditions were not stated (8). The outlet of the domain was set as a pressure outlet with default boundary conditions. The airfoil was set as a no- slip wall such that a boundary layer would form over the surface.

In order for FLUENT to calculate the solution coefficients, the reference values used in the calculations must be properly set. FLUENT can compute these reference values by specifying the appropriate part from which to calculate and the reference zone to where the values should be applied. The airfoil was the part selected, and the main reference value calculated from it was the reference, or chord, length. The fluid zone was specified as the reference zone, as this was the only zone created in the meshing scheme.

α	Vx	Vy
-2	2.483486	-0.08673
0	2.485	0
2	2.483486	0.086725
4	2.478947	0.173345
5	2.475544	0.216582
6	2.471387	0.259753
8	2.460816	0.345845
10	2.447247	0.431516
12	2.430697	0.516661

Table 1- 0012 Validation Velocity Components

The spatial discretization of all the transport equations was set to second order upwind scheme. FLUENT recommends this second order scheme to decrease the numerical diffusion error that is more likely to occur with a first order scheme. Additionally, the second order scheme was recommended for flows that are not aligned with the grid, which is the case for the higher angle of attack validation cases as well as the dynamic analysis where an unstructured mesh was used. The pressure- velocity coupling scheme was the FLUENT default SIMPLE scheme, and was recommended for steady state flows. (8)

The convergence criterion for all transport equation scaled residuals was set to 10^{-4} and was met by all of the equations for each validation case. This 10^{-4} is an order of magnitude lower than what was recommended by FLUENT for most cases (8). The solution was always initialized using the boundary conditions at the inlet. The solutions were iterated until convergence, with a maximum number of iterations set at 50,000, which was never reached. The number of iteration required for convergence varied by case, but generally fell below 10,000.

2.4 – VALIDATION RESULTS

Resolving the boundary layer of the airfoil with 10 to 20 cells with an initial y^+ on the order of 1 was a requirement of the k-omega SST turbulence model. The definition of y^+ is given in Equation 2 below and represents the non-dimensional wall distance.

$$y^+ = \frac{yu_*}{\nu}$$

Equation 2- Y Plus (10)

$$u_* = \sqrt{\frac{\tau_w}{\rho}}$$

Equation 3- Friction Velocity (10)

Where y is the cell wall distance, u_* is the friction velocity, and ν is the dynamic viscosity. The τ_w is the wall shear stress and ρ is the air density. Notice that the y^+ quantity is not only based off the cell distribution, but it is also a function of the flow itself through the friction velocity. This function of the flow makes meshing for a specific y^+ value somewhat of a guess and check process.

The validation of the boundary layer is accomplished by plotting the y^+ value against the ratio of the free stream velocity, U , and the friction velocity. The results of the boundary layer validation are shown in Figure 6 below. The initial cell y^+ is approximately 0.85, which is acceptably on the order of 1. Care was taken to select a region of the airfoil where there was not a significant velocity gradient and that the selection plane was perpendicular to the airfoil surface. The location of the boundary layer selection line is shown in Figure 7 below.

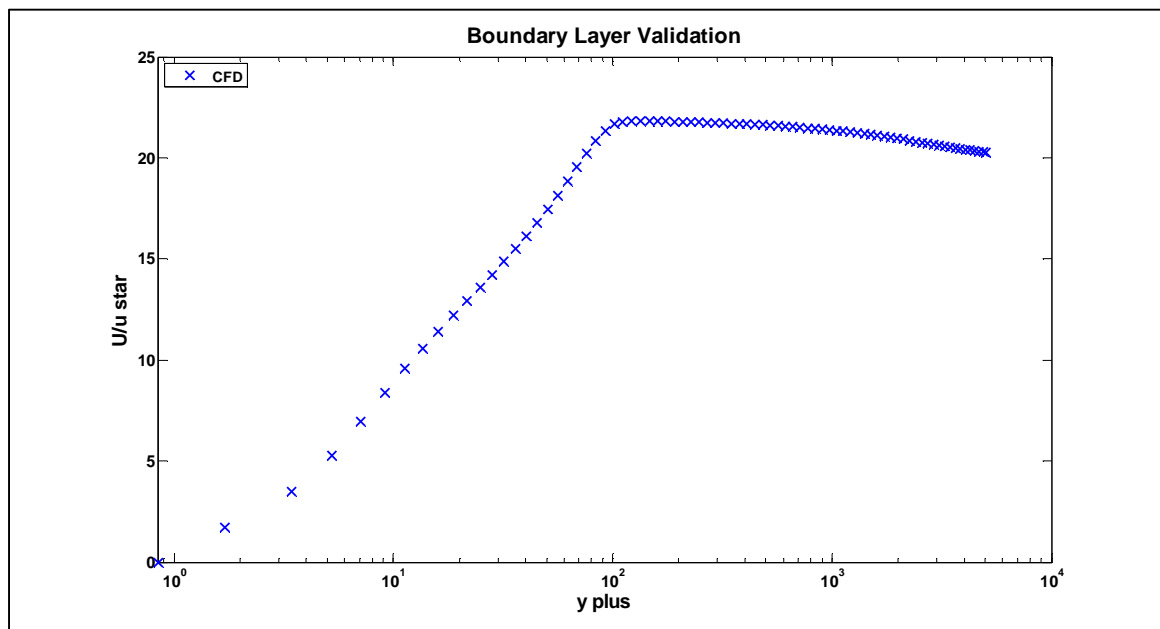


Figure 6- Boundary Layer Validation

At a y^+ of approximately 100, the validation data can be seen to level out, which indicates the edge of the boundary layer, where the local velocity is nearly the same as the free stream velocity. There are approximately 24 cell data points before the leveling out of the data, which leads to the conclusion that the boundary layer is resolved with more than the required amount of points. The ample amount of boundary layer points combined with the initial y^+ on the order of 1 allows for the conclusion that the meshing scheme used provides a sufficiently resolved and valid boundary layer.

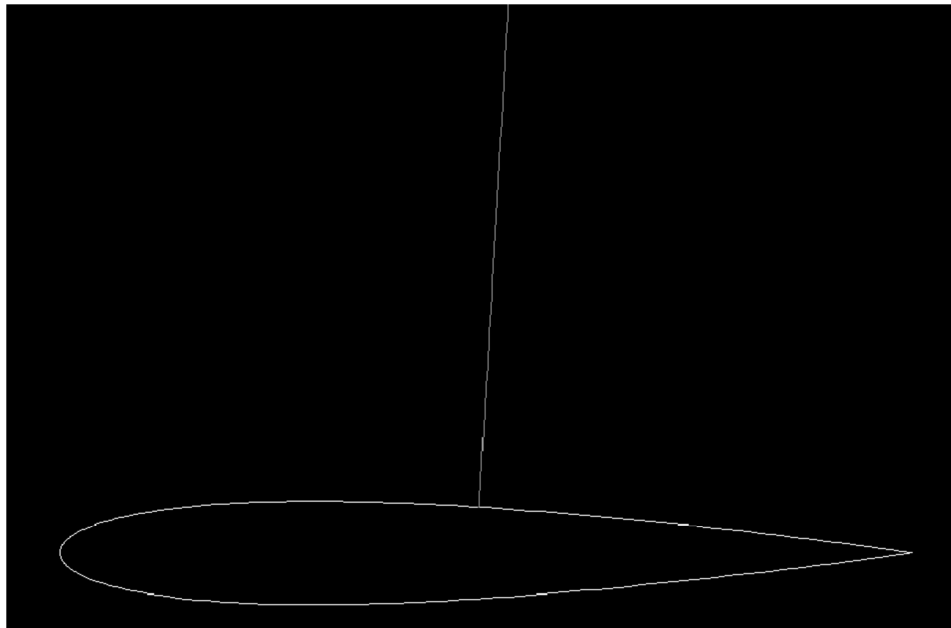


Figure 7- Boundary Layer Selection Line

A grid independence study was completed to validate that the mesh sizing was not having an influence on the solution result. Three different mesh sizings were used, one more coarse and one more fine than the base, medium, mesh. The three meshes were run at an AoA of 5 degrees using the computational setup described earlier in this chapter. The only mesh sizings that were changed from mesh to mesh were the number of cells on the airfoil, which had an effect on the total number of cells. Table 2 outlines the different mesh sizings and the force results of the study is shown below.

	Cells	Airfoil Cells	Cl	Cd
Coarse	25344	140	0.492876	0.016802
Medium	43279	235	0.491980	0.016797
Fine	68034	345	0.491528	0.016824

Table 2- Grid Independence Validation

As Table 2 shows, the total number of cells ranged from 25,344 to 68,034 with the number of cells comprising the airfoil surface ranging from 140 to 345. The lift and drag coefficients did not vary any significant amount from the coarse to fine mesh cases. This lack of any variation allows for the conclusion that the solution result is independent of the mesh and that the medium base mesh and sizing methodology is acceptable.

To further demonstrate grid independence, the pressure coefficient as a function of airfoil chord location has been plotted in Figure 8 below. Figure 8 is slightly ambiguous, but this is due to all of the validation data producing an identical pressure coefficient curve. This convergence of validation data further leads to the conclusion of grid independence. A further pressure curve validation is explored later in this chapter.

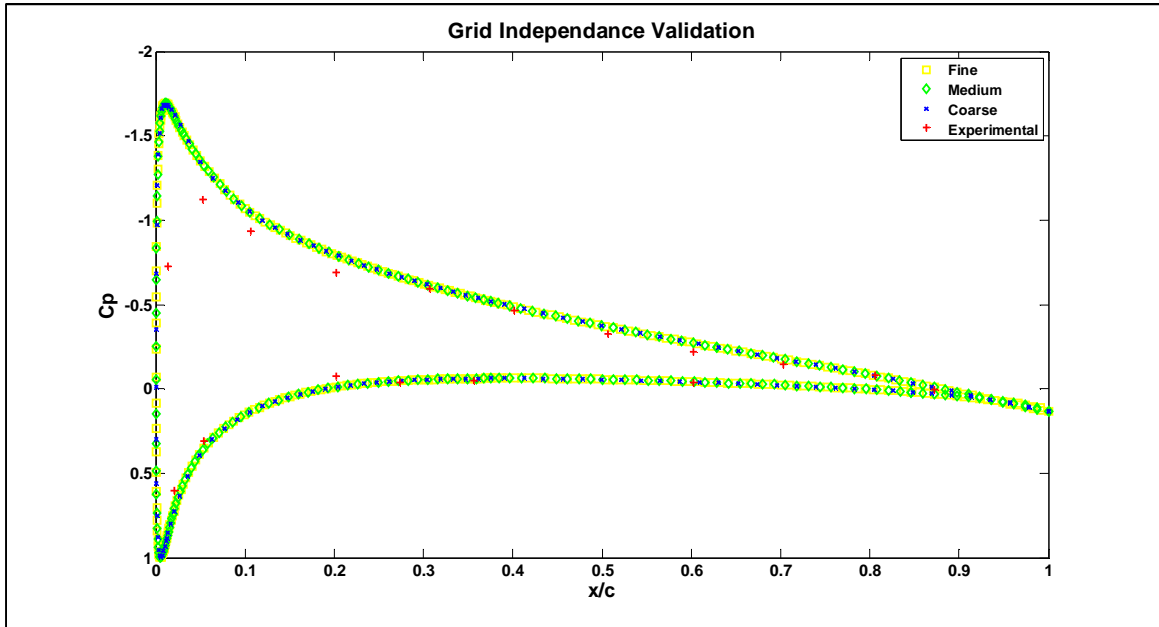


Figure 8- Grid Independence Pressure Validation

Since the boundary layer requirements for the turbulence model have been validated and the base mesh has been shown to provide a mesh independent solution, the force and pressure validation data can be presented with confidence. The first validation is that of the lift coefficient. The validation data has been plotted against the experimental data and is shown in Figure 9 below.

The validation solution matches well with the experimental data until approximately an AoA of 9 degrees. The error at the higher AoA is due to the CFD solution not properly predicting the separation that leads to the stalling of the airfoil. This could be due to a limitation in the

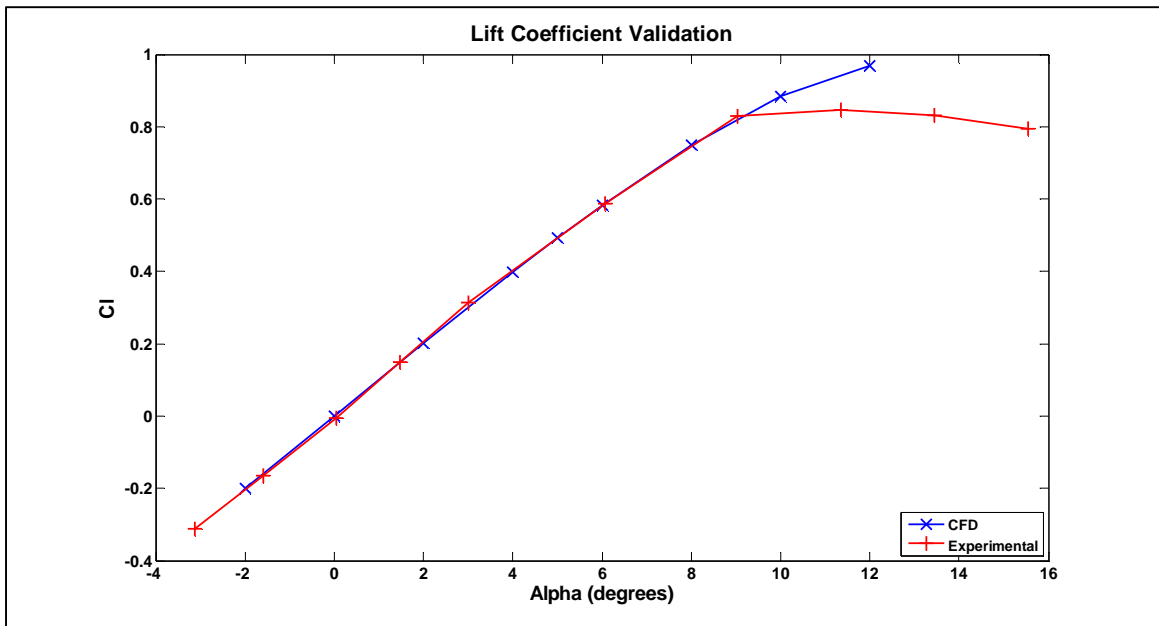


Figure 9- 0012 Lift Validation (16)

turbulence model not being accurate in the stall regime for steady state calculations. Although there is the discrepancy at higher AoA, the majority of the lift curve is shown to match the experimental data with very slight variations that could have been caused by the recording of the experimental data.

The drag coefficient validation is shown in Figure 10. First, the fluctuations and poor resolution of the experimental data were present in the original data and is not an error attributed to the re-presenting of the data. Despite the fluctuations in the experimental data, the trend of validation data follows the experimental data until a C_l of proximately 0.7. The zero lift error in the drag values is approximately 10%, which is acceptable due to the challenges of accurately modeling drag through CFD. This discontinuity at the higher C_l values can be attributed to the CFD solution not being stalled because the error occurs in the same region as the error in the lift curve above.

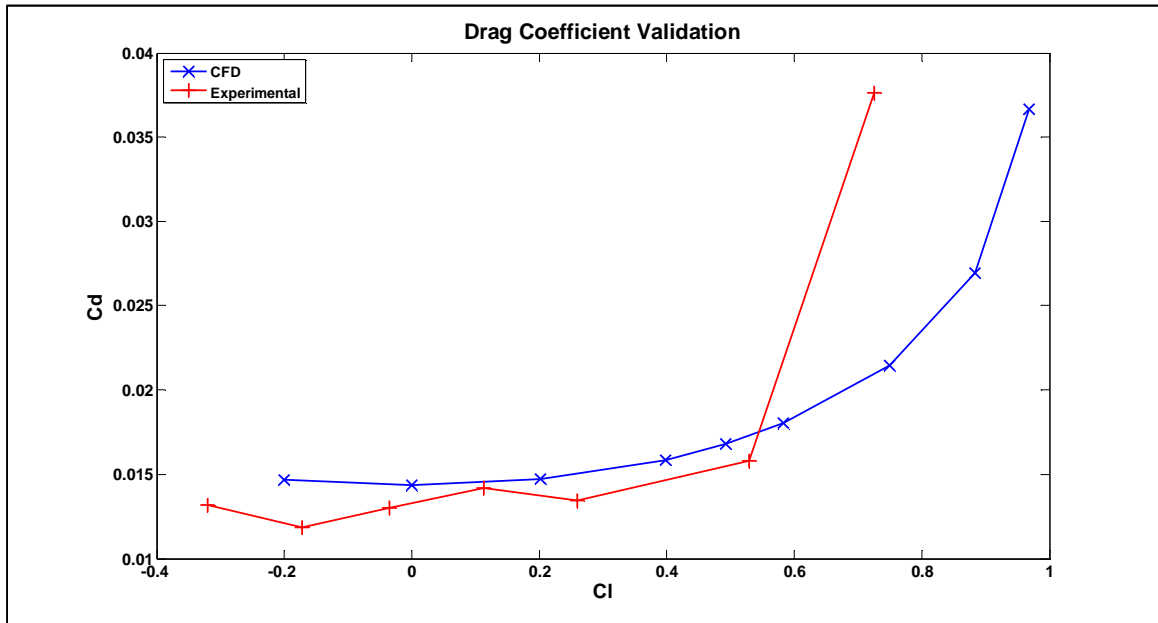


Figure 10- 0012 Drag Validation (16)

By comparing the validation and experimental results for the force values through Figure 9 and Figure 10, it can be concluded that the CFD solution is providing sufficiently accurate results for a majority of the AoA range. The error at high AoA and C_l values comes from limitations in the turbulence model when dealing with separating flow and stall conditions.

The final validation performed on the NACA 0012 airfoil was to compare the airfoil pressure coefficient from the CFD solution to that of the experimental data and is plotted in Figure 11. The AoA used in the C_p validation was 5 degrees so that there would be a distinguishable difference in the upper and lower surface pressures of the airfoil, as well as there being experimental data available for this AoA.

It should be noted that the Y axis has been reversed, such that the negative pressures are on the top. This was done to better represent the physical nature of the pressure, because the upper surface of the airfoil experiences the lower pressure. One thing to notice in the figure above is the stagnation pressure coefficient has been modeled almost perfectly, as the highest possible pressure is the pressure at the stagnation point and has a coefficient value of 1. This can be seen in the lower left corner of the graph where the validation solution maximizes at a C_p of 1. The location of the stagnation pressure is the lower, positive pressure side, leading edge which is expected from a positive AoA.

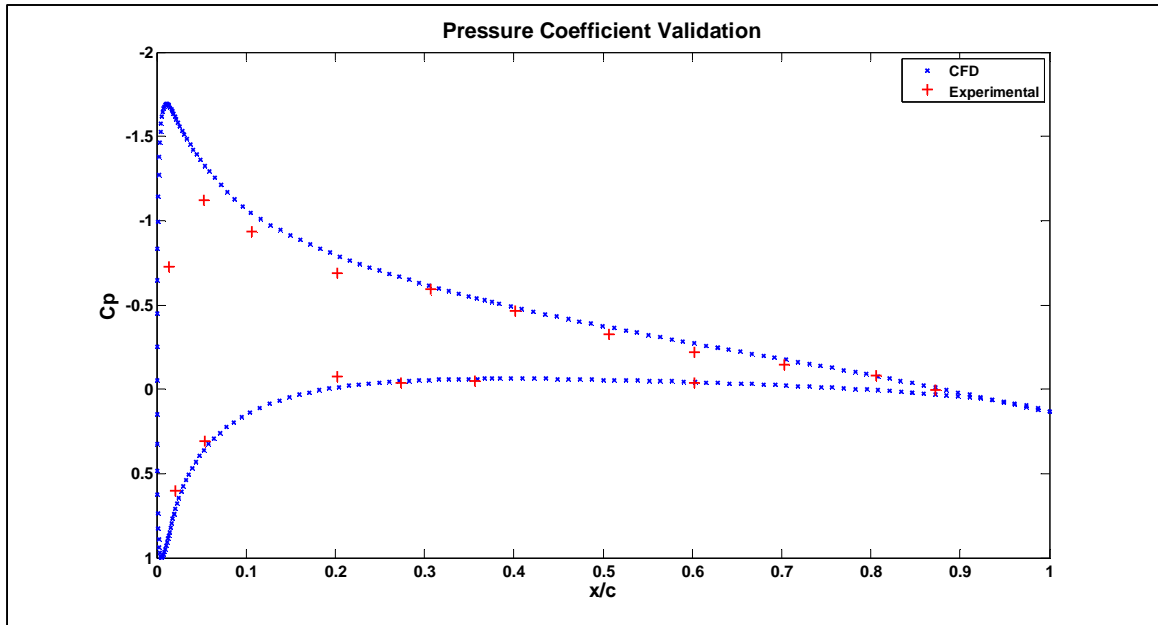


Figure 11- 0012 Cp Validation (17)

The lower surface C_p from the validation solution matches the experimental data well from the leading edge to the trailing edge. There is a slight error at an x/c of 0.2, but this appears to be an error in the experimental data because at an x/c of approximately 0.275 the validation and experimental data match again. The upper surface C_p values match well from an x/c of 0.3 to the trailing edge. From the leading edge to the x/c location 0.3, the validation solution seems to have slightly over predicted the negative C_p as compared to the experimental data. The resolution of the experimental data is such that the true extent of the over prediction cannot be fully realized. At an x/c of approximately 0.025, the maximum leading edge suction occurs and there is no experimental data in this region. It is entirely possible that if experimental data was available for this region it would show a similar spike, but there is just no way of knowing. Although the C_p is slightly over predicted on the upper surface just behind the leading edge, the error is small and with the rest of the validation C_p curve matching the experimental data, it can be concluded that the pressure prediction is sufficiently accurate in providing a valid solution.

In conclusion, the boundary layer requirements from the turbulence model have been met, the solution has been shown to be independent of the grid, and the airfoil forces and pressure distribution have been validated against experimental data. Although there is some slight error in the force validations due to turbulence model behavior, the error occurs at the extremes of AoA and the majority of the lift and drag curves match the experimental data. Further, the error displayed by the C_p distribution at 5 degrees AoA is minimal and no discernable error is present in the lift curve at 5 degrees. This is an important validating fact for the C_p data, because the lift is mostly calculated by integrating the pressure distribution. Each of these individual validations helps lead to an overall conclusion that the meshing scheme derived for the NACA 0012 airfoil and the solution setup used in FLUENT provide an overall accurate and valid result.

CHAPTER 3 – NACA 0015 VALIDATION

3.1 – INTRODUCTION

The NACA 0015 airfoil was selected for use in the wing oscillator due to its slightly larger thickness over the NACA 0012. This increased thickness was beneficial to the initial design prototype and will be discussed in chapter 4. The 0015 is not as widely used as the 0012 for research and thus there is a limited amount of experimental data available in a Reynolds number regime that is similar to the wing oscillator, and was the reason for the thorough 0012 validation. The only experimental data available for the 0015 is the lift and drag curves at a Reynolds number of 166,000.

The meshing scheme used for the 0015 airfoil was identical to the 0012 scheme described in the previous chapter in every way, so much so that the 0012 ICEM file was converted to a 0015 by deleting the 0012 geometry and importing the 0015 airfoil curves. These 0015 curves were then associated to the existing blocking and mesh sizing. Due to the increased thickness of the 0015, the mesh sizing of adjacent edges between the blocks on the upper and lower airfoil surfaces had to be re-matched slightly. The number of cells on the airfoil was 235 and the total number of cells was 43,279. Both of these values are identical to the 0012 mesh.

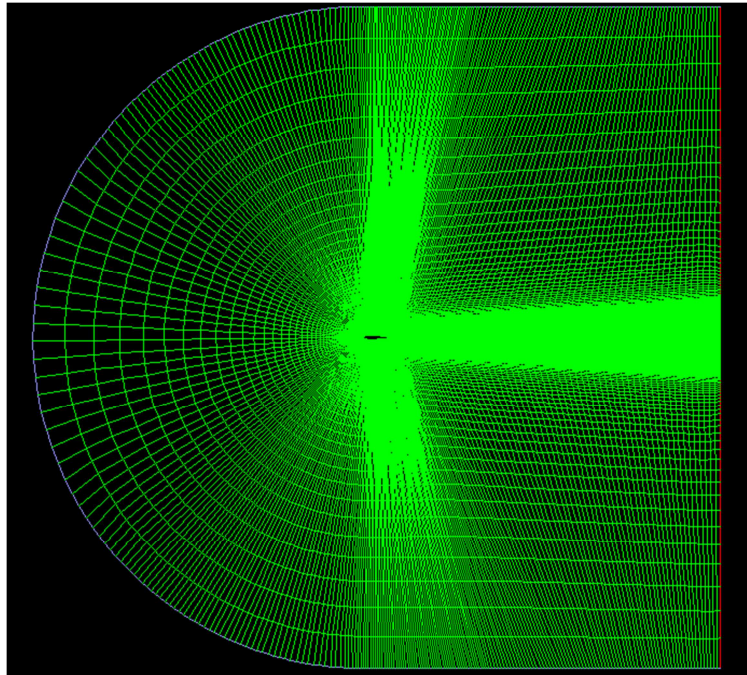


Figure 12- 0015 Far Field Mesh

The FLUENT solution setup was also identical, where by the 0012 case was loaded and the 0015 mesh imported into the case. Due to all of the boundaries being identically named, the 0012 boundary conditions were read onto the 0015 boundaries without issue. Although the case was converted from the 0012 to the 0015 without error, all of the settings were checked manually. The only difference in the solution setups was that a different velocity magnitude was

required due to the slightly lower Reynolds number. To meet the 166,000 Reynolds number from the experimental data, the velocity magnitude was set to 2.425 meters per second. The X and Y velocity components used to control the angle of attack are given in Table 3 below.

α	Vx	Vy
-2	2.423523	-0.08463
0	2.425	0
2	2.423523	0.084631
4	2.419093	0.169159
6	2.411716	0.253482
8	2.4014	0.337495
10	2.388159	0.421097
12	2.372008	0.504186

Table 3- 0015 Validation Velocity Components

3.2 – VALIDATION RESULTS

As mentioned above, the 0015 experimental data was limited to the lift and drag curves. The lift curve is given in the Figure 13 below. As with the 0012 airfoil, the validation data matches well with the experimental data for a majority of AoA. The discrepancy, again, is in the higher AoA region, however the error is smaller than the 0012 case. This is due to the 0015 not stalling until approximately 11 degrees and the CFD solution not stalling at all. Right until this 11 degree stalling point, the validation data matches the experimental data almost perfectly. The only other error occurs at approximately 6 degrees and appears to be an error in the

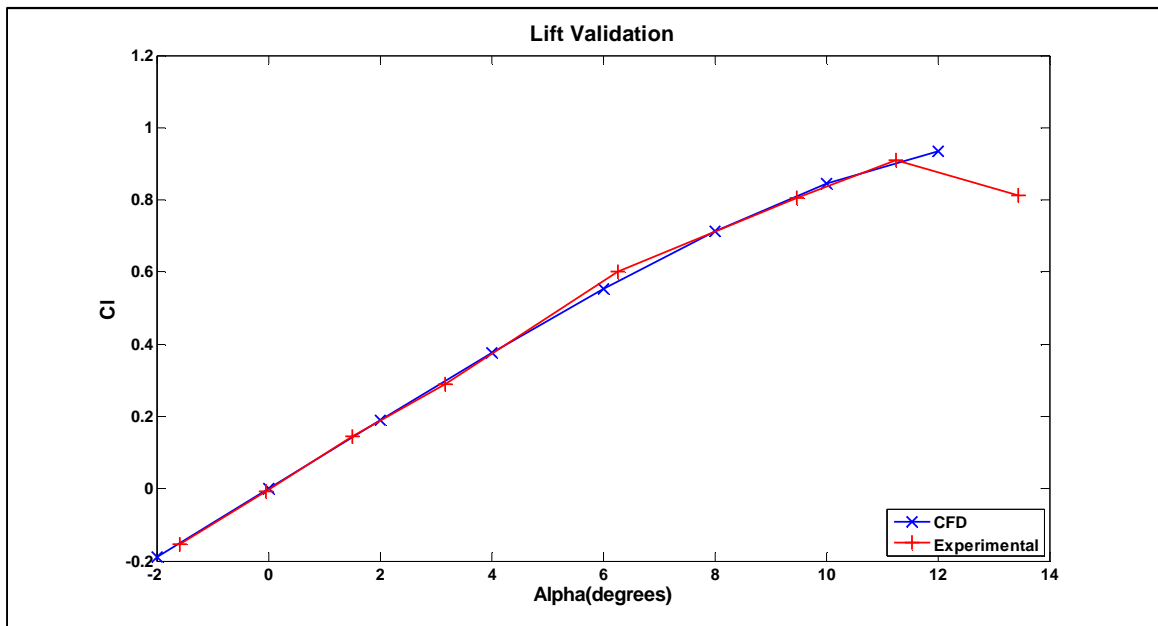


Figure 13- 0015 Lift Validation (16)

experimental data as the experimental data point does not lie in a straight line with the previous and next points. Overall, the error is minimal and the error in the stalling AoA region can be attributed to a turbulence model limitation.

The drag validation curve is given in Figure 14 below. The 0015 drag validation matches the experimental data better than the 0012 drag validation. This is partially due to the lack of severe data fluctuations as seen in the 0012 data. The only peculiar experimental data point is the zero lift data point, which shows a dip from the smooth trend normally observed. Due to this dip the error at zero lift is approximately 67%. This error seems to over represent the overall error as the trend through most of the drag curve is decreasing error. At a Cl of 0.6 the error is approximately 19% and at the last data point, the highest Cl, there error is just under 1%. Due to the decreasing error through the curve, and the fact that properly modeling drag in CFD solutions is difficult, the error displayed is considered acceptable.

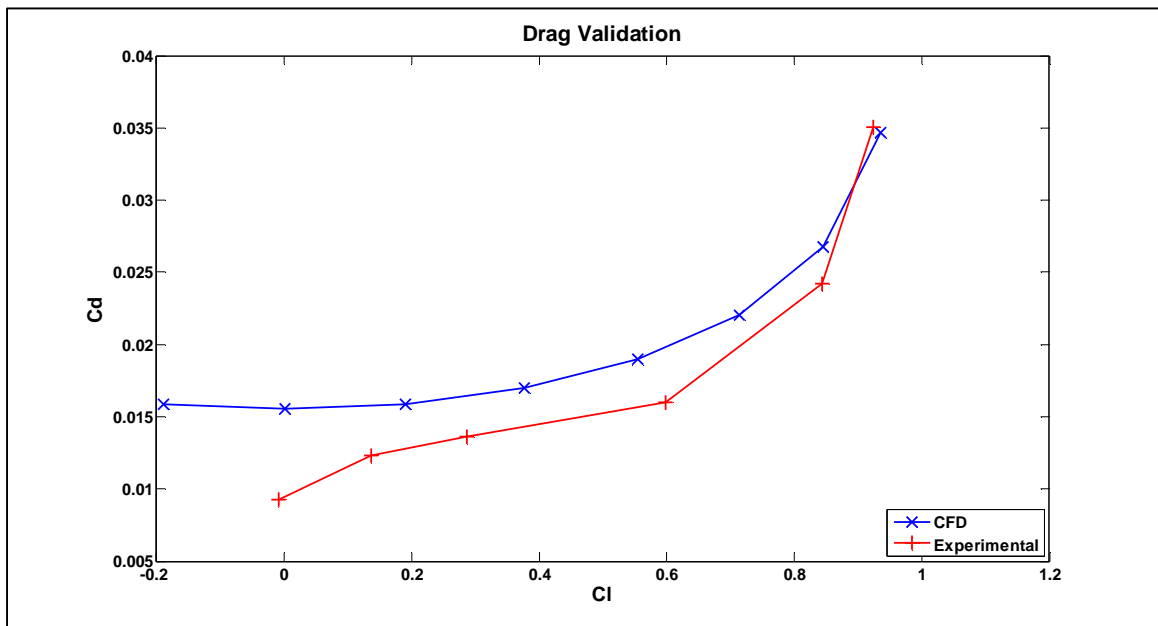


Figure 14- 0015 Drag Validation (16)

The 0015 validation data has been shown to match or follow the trend of the experimental data and has been concluded to be sufficiently accurate and valid. The lift validation is accurate for almost the entire lift curve, and the drag curve error decreases as the Cl increases. A larger area of error is the stalling AoA region, which was the same for the 0012 validation, and has been attributed to a limitation in the turbulence model handling separated flow. Another large area of error is the drag at the zero lift condition. This error is suspected to be due to an error in the experimental data as that data point seems to be below the trend of the other points. The experimental data was also suspected to be slightly erroneous in the 0012 validation, but the low zero lift drag data was not present in the 0012 validation.

CHAPTER 4 – DYNAMIC ANALYSIS

4.1 - GEOMETRY

In order to predict the performance of the wing oscillator, a dynamic computational analysis was carried out. This analysis was modeled after a proposed prototype that was designed for wing tunnel testing in Western Michigan University's Advanced Design Wind Tunnel. Care was taken to ensure the computational analysis would model as closely as possible the wind tunnel testing conditions and geometry. As with the validation cases, only two dimensional analyses were conducted due to the limited three dimensional aspects of the flow and lack of any major span wise varying geometry with regards to the wing oscillator.

The Advanced Design Wind Tunnel has a rectangular test section with a height of .812 meters and a length of 2.438 meters. The depth of the wind tunnel is 1.143 meters, but is not used due to the two dimensional nature of the analysis. From the control room for the wind tunnel, the airflow travels from left to right, thus a convenient reference frame is that the X axis is positive to the right, which allows for an always positive value for velocity and the y axis is positive upwards towards the top of the test section. This is also the same reference frame used by FLUENT, which is beneficial because no geometry transformation is necessary and the computational model can be easily related back to the real world wind tunnel. (11)

The wing oscillator prototype was designed specifically for use in the Advanced Design Wind Tunnel and has been scaled from the original conceptual design from Liu, which is shown in Figure 1 (1). The airfoils chosen for the design are designated as NACA 0015 with a chord length of .1524 meters. The airfoil pivot points are located at the quarter chord, or .0381 meters from the leading edge, and on the chord line through the middle of the airfoil thickness. The quarter chord was chosen because it is roughly the aerodynamic center of the airfoil and thus the pitching moment will remain the same as the angle of attack changes. This was a design feature used for convenience when implementing a control system for the angle of attack. The distance between the two airfoil pivot points on the ADWT wing oscillator prototype is 1.1473 meters, giving each of the airfoils a moment arm of .7366 meters. The prototype was designed to be centered in the wind tunnel such that the main system pivot point is located precisely in the middle of the test section. A frame consisting of metal or plastic tubing would be used to connect the airfoils to the pivot point, which would be located on a stationary frame with bearings to reduce frictional losses in the oscillatory motion. The airfoil angle of attack control system would consist of either electronically controlled servos or a mechanical system. The final design of the control system was not completed; however it is not critical to the computational analysis to have its design finalized as no part of that system is included in the analysis.

The computational domain consists of four walls representing the wind tunnel test section and the two airfoils. All of the other components were excluded from the analysis because of the two dimensional nature of the analysis. Excluding the support frames also

reduces the computational cost as resolving the flow over the frame bodies is not necessary. For computational convenience, the main system pivot point was centered at the origin, $(0,0)$, of the computational domain. Using the reference coordinate system above, the inlet and outlet of the test section were located at -1.219 and 1.219 meters respectively and aligned parallel to the y axis. The top and bottom of the wind tunnel, noted as walls in the computational domain, were located at $.406$ and $-.406$ meters respectively, and aligned parallel to the y axis. In this manner, the computational domain was completely restricted to the xy plane, thus satisfying the two dimensionality of the analysis. The airfoils were set in the domain such that the angle of attack pivot points for the leading and trailing airfoils are at $-.7366$ and $.7366$ meters. A diagram of the geometry is given in Figure 15 below where the blue lines represent the wing oscillator

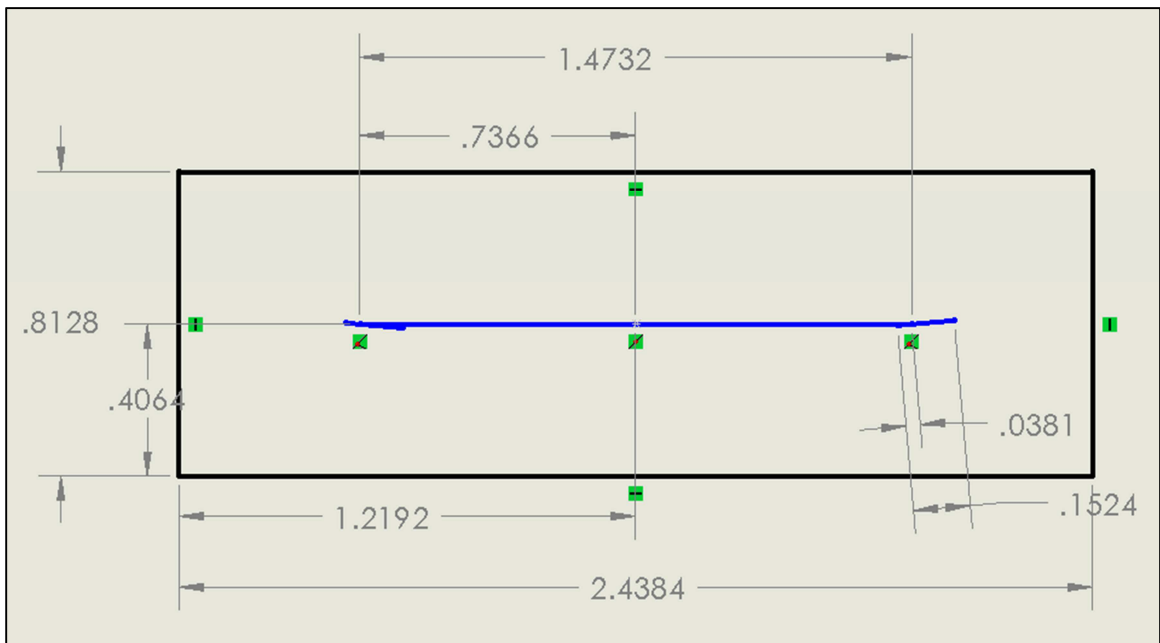


Figure 15- Wind Tunnel Model Geometry

frame and airfoil chord lines and the black lines represent the wind tunnel boundaries.

4.2 – MESHING

The meshing scheme implemented for the computational domain was a fully unstructured, triangle based mesh. This particular scheme was chosen for a number of reasons. First, due to the need for the airfoils to traverse the computational domain, a structured scheme such as that used in the validation cases was not convenient. As the airfoils heave vertically, the domain between the airfoils would get significantly shifted at an angle due to the structured requirement of the cells to keep reference to the airfoils. The small boundary layer sized cells in this region would become ill aligned to the flow direction and could potentially cause numerical errors or convergence problems. By using an unstructured triangular cell scheme, the cells in

this region do not need to deform themselves to stay referenced to the space between the airfoils and would only need to move or deform if a moving body came through the region. Second, the FLUENT dynamic meshing parameters are designed to be used on unstructured triangular meshes. This includes both the spring based smoothing deformation methods, as well as the dynamic re- meshing methods. These specific parameters are described further below.

One of the negatives of using this unstructured mesh scheme is that in the near wall region, the mesh quality is much harder to control. This control issue makes it more difficult to resolve the boundary layer as required by the turbulence models. In order to get an acceptable resolution of the flow structure in the boundary layer region, the number of cells on the airfoil region had to be increased significantly over the number used in the validation cases. This lack of cell control also comes from the fact that the cell spacing can only be defined at the boundaries. This limitation has the potential to leave parts of the domain less defined than desired due to the growth of the mesh away from the boundaries leaving too large of cells in certain flow regions. This is not as much of a problem when using structured meshes because artificial edges with spacing definition can be placed to control the mesh density in specific regions of the domain.

The meshing software used for the unstructured meshing was the ANSYS Workbench. Originally ICEM was used, however it was concluded that for two dimensional unstructured domains the Workbench approach allowed for more control of the growth parameters and overall higher quality meshes. The Workbench approach also allowed for better integration with FLUENT and testing of the meshes was faster as FLUENT could be started and the test mesh automatically imported through the Workbench software. Also, the geometry described above was easily created through workbench, which allowed for easier modification to the base geometry airfoil angle of attacks which were used in different testing conditions.

The first step was to create the base computational domain using the Workbench Geometry modeler. As described above, the computational domain consisted of four walls defining the rectangular test section and the two airfoils. The four walls were simply created from straight edges connecting the corner points, which were defined from the known geometry points described above. The airfoils were imported in a similar fashion as the validation cases. The 0015 airfoil was imported as an IGES file that was created from a coordinates text file in Solidworks. Because the imported airfoil shape was of unit length, it needed to be appropriately scaled to the dimension of the prototype. In addition to the scaling, the airfoil needed to be translated into the base position and then copied so that two airfoils existed in the domain. The second airfoil shape then needed to be translated into position as well. In addition to creating these parts in the Geometry Modeler, the airfoils were given an initial angle of attack by rotating the airfoils about the airfoil pivot points which again were located at the quarter chord. This was accomplished by creating local coordinate systems to use as the rotational axes of the airfoils. This initial angle of attack had to be changed at this level of the meshing scheme if different angle of attack parameters were desired for testing. The initial angle of attack chosen for initial mesh testing was set at negative and positive five degrees for

the leading and trailing airfoils respectively. The final geometry setup step was to create a surface for the fluid domain itself. This surface included the rectangle formed by the free walls, inlet, and outlet with the airfoil sections cut out as these are represented as solids and do not need to be meshed. Figure 16 shows the geometry in Workbench, including the initial airfoil geometry before scaling, translation, and rotation. Note that although the initial airfoil geometry remains in the Workbench file, it has no effect on the meshing because it is not defining any cut outs of the fluid surface.

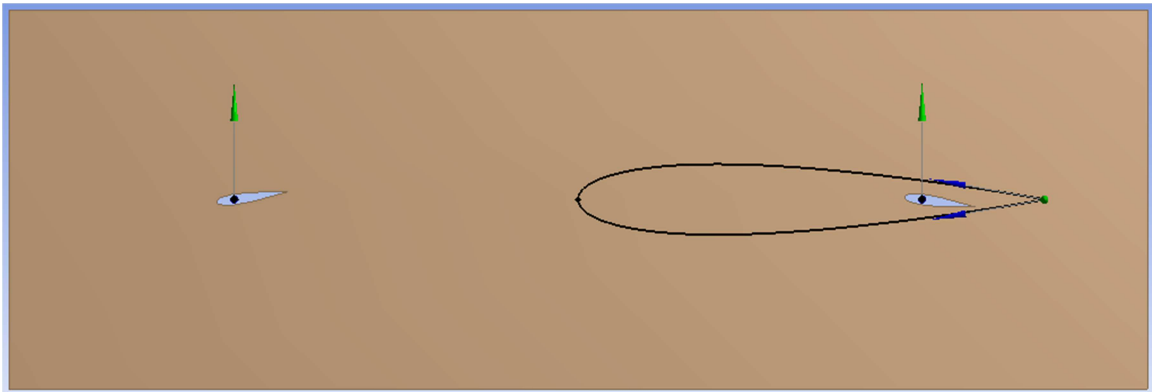


Figure 16- Workbench Geometry

Once the geometry was finished being created with the Geometry Modeler, the ANSYS Meshing software is loaded and the geometry data is read. The first meshing step is to create named sections for the different geometry edges and boundaries. This naming helps perform two tasks. First, by giving the edges and boundaries names, specifying selections for meshing parameters is extremely simplified within the Meshing software itself. Second, by choosing appropriate names for the edges and boundaries FLUENT can automatically set boundary conditions. By default FLUENT uses the simple wall type boundary condition, but if the inlet and outlets have names similar to *inlet* and *outlet* FLUENT will set those edges to more appropriate boundary conditions. The naming process is accomplished in the Meshing software by selecting the desired edge and creating a named selection. The named selections used in the current meshing scheme are leading and trailing for the leading and trailing airfoils respectively, inlet and outlet for the inlet and outlet edges of the test section, and free walls for top and bottom edges of the test section. With the exception of the free walls, in which the top and bottom edge were grouped, all of the named selections consist of only one edge. The free walls could be grouped because the boundary conditions applied to both the top and the bottom are identical.

After naming the parts appropriately, the meshing parameters could be set and created. First, the unstructured meshing method must be set to all triangles, as opposed to triangle dominated or quadrilateral. The all triangles method was chosen because the FLUENT dynamic meshing is best suited to these types of unstructured meshed. FLUENT is capable of some dynamic meshing with quadrilateral cells, but the re-meshing scheme is only valid for triangular cells. Some of the parameters associated with the meshing type are general global parameters such as the maximum and minimum mesh sizes, the general transition and growth rate, and the

general amount of smoothing. For all of the unstructured meshes used in this current study, the maximum and minimum cell sizes were .02 and .0001 meters respectively. The general



Figure 17- Edge Sizing Definitions

transition rule was set to slow with a default growth rate of 1.2. The mesh smoothing parameter was set to high, which was the maximum available through the Ansys Meshing software.

The specific meshing parameters used were edge sizing definitions on the airfoils and outlet as shown in Figure 17. All of the specifications used on these edges were defined by the number of divisions as opposed to specific sizing requirements like the global parameters. Besides defining the number of divisions on the edges other meshing options were selected, such as the behavior of the sizing definition and the local growth rate of the cells away from the edge. The behavior of the sizing is a parameter that can be set to either soft or hard. When soft is selected, the meshing algorithm has more control over the edge sizing depending on the proximity of that edge to other geometric and meshing features and definitions. When hard is selected, the number of divisions cannot be changed by the meshing algorithm. The hard parameter was only used on the outlet edge in order to ensure the mesh in the region behind the trailing airfoil was resolved well up to the outlet. The number of divisions on the outlet was defined as 50, which in combination of the small sizing on the trailing airfoil, gave sufficient mesh resolution, which can be seen in Figure 18 below.

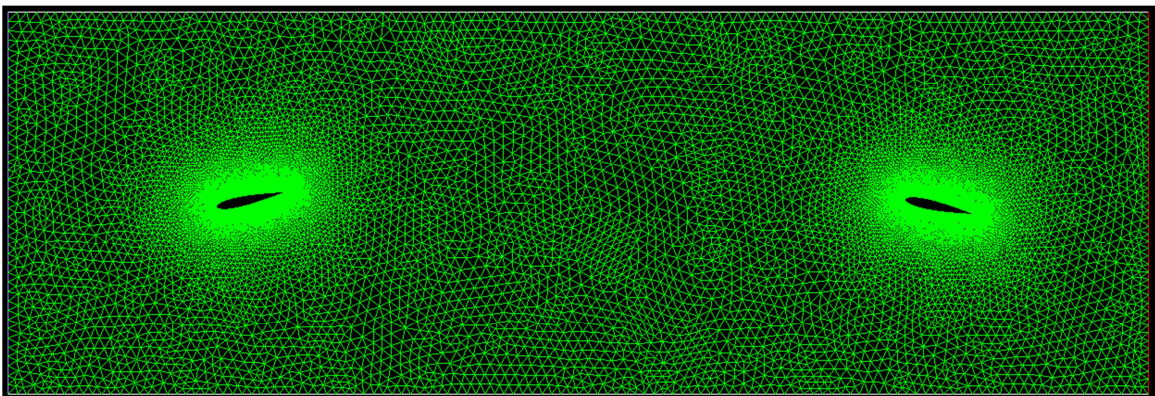


Figure 18- Wind Tunnel Mesh

The airfoils were technically defined as two separate edges that shared common leading and trailing edge points. It follows that, when defining the local meshing parameters on the airfoils each of the edges receives identical definitions. This means that the number of divisions specified for each of the airfoils is actually being defined for both the top and bottom surfaces of the airfoils. For the base cases used, the number of divisions used was 500 per airfoil side, giving a total of 1000 cells per airfoil. This number was determined through experimentation in order to get the airfoil y^+ values to be on the order of 1, which was required by the turbulence model used (8). The cell growth rate used for the airfoil edges was 1.075, which was similar to the growth rate used in the validation cases, yet provided a reasonable number of total cells. The total number of cells used for the base meshes was approximately 80,000. This is an approximate number because although identical meshing parameters were used for the meshes with different initial angles of attack, the meshing algorithm might not grow the mesh in quite the same fashion and the total number of cells could vary slightly.

4.3 - FLOW DRIVEN THEORY

There were two choices for driving the system motion for the dynamic analysis. The first would be to specify the airfoil motion as a function of time, and the second would be to have the flow over the airfoils drive the motion. While exploring the first option, immediate problems arose. By specifying the motion, the system output could already be calculated and there would be no point in performing a CFD analysis of the system. Another problem would be in determining an appropriate specified function. The best way would be to create a function from experimental data, but this again defeats the purpose of performing a CFD analysis. These problems quickly lead to the conclusion that a flow driven dynamic analysis should be explored and conducted if feasible.

Determining the feasibility of the flow driven solution was conducted by examining the computational tools available. FLUENT is capable of dynamic analyses through the use of its built in dynamic meshing tools and User Define Functions (UDFs). By linking UDFs that describe the desired motion to FLUENT and then defining the moving boundaries and the re-meshing parameters, an analysis with moving bodies can be performed. It became immediately clear that the UDF approach would be necessary. Simple UDFs were written to explore their use and implementation, as well as investigating the use of the dynamic mesh tools. After the initial exploration, it was determined that by implementing a mathematical scheme which describes the system motion in FLUENT through UDFs and the built in dynamic meshing tools, a flow driven dynamic analysis would be feasible.

4.3.1- MATHEMATICAL SCHEME

The mathematical scheme used to define the motion, which is passed to FLUENT through UDFs, is based upon rotational motion. The rotational motion scheme was used because the system as a whole is based on the airfoils, and other system components, rotating around a single pivot. The governing differential equation for rotational motion is given below in Equation 4.

$$\frac{d^2\theta}{dt^2} = \frac{d\omega}{dt} = \frac{\tau}{I}$$

Equation 4- Rotational Motion Governing Equation (12)

Where θ is the system angle, t is the time, ω is the angular velocity, τ is system torque, and I is the system moment of Inertia. The left hand side second order differential represents the angular acceleration of the system and is equivalent to the first derivative of the angular velocity. For the implementation of this math scheme, it is more important to calculate the angular velocity and use that value throughout the following calculations. First, however, it is necessary to linearize the differential equation. This can easily be accomplished using a first order approximation, which will be sufficient for the analysis. A derivation of this is given in the following equations.

$$d\omega = \frac{\tau}{I} dt$$

$$\Delta\omega = \frac{\tau}{I} \Delta t$$

$$\omega_n = \omega_{n-1} + \frac{\tau_n}{I} \Delta t = \omega_{n-1} + \Delta\omega_n$$

Equation 5- Linearization of the Governing Equation

Where Δ represents the change in a variable and n represents the current time step. This linear approximation simply states that the angular velocity at a given time step is equal to the angular velocity at the previous time step plus the current change in the angular velocity. In this fashion, the differential equation is calculating only the change in the angular velocity by knowing the torque produced by the system at the current time step and the moment of inertia of the system. The moment of inertia is constant and an approximate value of 1 was used after a rough estimation was completed for the wing oscillator prototype. The torque, however, is constantly changing as the flow over the airfoils change with the dynamic motion. The torque at any given time is the sum of the reaction forces from the airfoils acting perpendicular to the frame connecting the airfoils and passing through the pivot point, multiplied by the distance from the airfoil pivot point to the central system pivot point. The airfoil angle of attack pivot point was chosen because it is at this point where the airfoils are connected to the system, thus the reaction forces must be resolved at this location.

To resolve the reaction forces such that they are parallel to the frame, trigonometric principles must be used on the two airfoil forces, lift and drag. The lift force is defined as the force acting perpendicular to the fluid flow and drag is defined as the force acting parallel to the fluid flow. For this dynamic case, these directions were simply resolved due to the geometry and reference frame chosen such that the fluid flow can only be in the X direction. This means that the lift force will always be acting in the Y direction and the drag force acting in the X direction. In addition to the forces on the airfoils, the model has been designed to have a torsional spring applied to the central system pivot for damping purposes. This spring adds a torque to the system proportionally with respect to the spring constant and system angle. A derivation of the torque from the airfoil forces follows.

$$\tau = R_l r - R_t r + S$$

Equation 6- System Torque

$$R_l = L_l \cos(\beta) + D_l \sin(\beta)$$

Equation 7- Leading Airfoil Reaction Force

$$R_t = L_t \cos(\beta) + D_t \sin(\beta)$$

Equation 8- Trailing Airfoil Reaction Force

$$S = C_s * \beta$$

Equation 9- Torsional Spring Torque

Where the subscripts l and t represent the leading and trailing airfoils respectively, r represents the distance from the airfoil angle of attack pivot point to the central system pivot point, β represents the system angle of attack, S is the spring torque, and C_s is the torsional spring constant. The subtraction in the first equation comes from the trailing airfoil having a negative moment arm with respect to the system, because to get from that airfoil pivot to the system pivot the distance would be in the negative X direction.

Although the system motion is based off these rotational principles, it was found to be more convenient to use Cartesian displacements to approximate the rotational motion. It was more convenient based on how the UDF macro passed information to FLUENT. The UDF principles will be explained later in this chapter; however, the Cartesian approximations will be derived here. The angular velocity, which is solved out of the linearized differential equation, needs to be resolved into the X and Y velocities for the airfoil as follows.

$$V = \omega r$$

Equation 10- Tangential Velocity

$$V_Y = V \cos(\beta)$$

$$V_x = V \sin(\beta)$$

Equation 11- Velocity Components

Where V is the tangential velocity at the airfoil pivot point, and the X and Y subscripts represent the Cartesian velocity direction. These equations are valid for both the leading and trailing airfoils and the positive and negative portions get resolved through the signs of the tangential velocity and trigonometric functions. These Cartesian approximations are similarly accurate to the linearized differential equation because the motion is again going to be based on the size of the time step. This approximation will be shown to not introduce significant error into the system later in this chapter.

4.3.2- USER DEFINED FUNCTIONS

As mentioned above, FLUENT requires UDFs that define the desired motion in order to drive the built in dynamic meshing capabilities and tools. FLUENT requires the UDFs to be written in the C programming language, and provides a significant amount of pre-written macros which can be linked to FLUENT to accomplish a number of common tasks. These C files, or source files, must be either compiled or interpreted before being linked to FLUENT in a functional manner. The choice of compilation or interpretation is based on the type of macro used and is specified in the supplied UDF manual. The dynamic meshing macros all require compiling, which can be accomplished through the FLUENT Graphical User Interface (GUI) menus. When compiling through the GUI, only the source files need to be specified as long as no proprietary header files are called in the source code. Additionally, a UDF library name can be specified if the default, *LIBUDF*, is not satisfactory. When the source files and library name are defined, the library can be built. This building process is the automated compiling of the source code files. It is imperative that the case, or mesh, file that will reference the UDF library be in the same directory as the library folder. Once the library is built, it can be loaded and the UDFs become available for use within FLUENT. It should be noted that once a UDF library is built, it can be loaded again at any time, thus it is not necessary to compile every time a UDF is to be used, provided that the library has been built with the current UDF source code. The UDF macro used for the flow driven dynamic scheme derived above is the *DEFINE_CG_MOTION* macro. The macro has six arguments and is called as follows:

`DEFINE_CG_MOTION(name,dt,vel,omega,time,dtime)`

The *name* argument is the name of the function which is specified by the user. The names used in the dynamic analysis were *leading* and *trailing*. The *dt* argument is a dynamic thread pointer which stores the dynamic meshing parameters. The *vel* and *omega* arguments specify the linear velocity and angular velocity arrays respectively. The arrays are zero based so 0 represents the X direction, and 1 represents the Y direction. The *time* and *dtime* arguments are the current time and the time step respectively. Only the *vel* and *omega* arguments get

passed from the UDF to FLUENT at each time step, however the *name* gets passed to FLUENT by loading the UDF library. For this dynamic analysis, two UDF source codes were used, one for each of the leading and trailing airfoils, and was necessary due to the nature of the DEFINE_CG_MOTION macro only being able to define motion of one airfoil at a time. These UDF source codes are extremely similar and perform almost all of the same calculations, but are different in that they send opposite motion commands to FLUENT. (13)

In addition to the macro, FLUENT allows access to many solution variables through the use of functions and looping. Using these functions and variables in combination with the basic mathematic functions common to C programming, almost any calculation can be carried out in the UDF and fed back into FLUENT. Recall from above that the goal of the governing mathematical scheme is to use the system torque to calculate the angular velocity, which is then converted to a Cartesian approximation of the rotational motion. The Cartesian velocity components are fed into FLUENT using the *vel* array in the dynamic mesh macro above. (13)

It is immediately imperative then, that the system torque must be calculated first. Recall that the torque on each airfoil is a trigonometric combination of the lift and drag forces which are in themselves a combination of pressure and viscous forces. Both the pressure and viscous forces must be calculated in order to properly resolve the total system torque. These can be calculated by performing summation loops over the cell faces making up the airfoils. Within the summation loops, solution variables on the airfoil faces can be called and calculated upon. The pressure force can be calculated, in both the lift and drag directions, by calling the pressure on the cell face and multiplying by the face projected area in either the *Y* for lift, or *X* for drag directions. The viscous forces can be calculated by calling the wall shear stress force acting in either the *X* or *Y* directions. These two forces can then be added to calculate the total force in either direction. The function that calculates calls the cell face pressure is:

$$F_P(f,t1) * A[0]$$

Where *F_P* is the face pressure variable with the *f* denoting a face pointer to the surface zone with the *t1* pointer. The *A[0]* is the projected area in the *O* or *X* direction with is multiplied by the face pressure to give the pressure force. (13) The viscous force is called through the following function:

$$F_STORAGE_R_N3V(f,t1,SV_WALL_SHEAR)[0]$$

This function is not defined in the FLUENT UDF manual and was only referenced through an online CFD forum, thus the way this particular function calls the wall shear stress is somewhat speculative. The *F_STORAGE_R_N3V* is presumably a variable storage array in which the shear stress is called out through the *SV_WALL_SHEAR*. The rest of the arguments are the same as the pressure calculation. Both of these functions are looped over all of the faces of the airfoil zones, which were defined through the *t1* pointer. These pointers, defined as *t1* and *t2* for the leading and trailing airfoils respectively, are defined at the mesh generation level and can be

found in the boundary conditions definition screen in the FLUENT GUI under zone ID and are set through the following:

```
Domain *d = Get_Domain(1);
```

```
Thread *t1 = Lookup_Thread(d,5);
```

```
Thread *t2 = Lookup_Thread(d,6);
```

The first function sets the domain pointer, d , which was simply 1 because there was only one fluid domain. The next two functions set the $t1$ and $t2$ thread pointers by defining the zone ID, 5 and 6, of the leading and trailing airfoils respectively. These zone values can be found in the boundary conditions definition page in the FLUENT GUI. In the UDFs these threads need to be defined before the looping summations of the airfoil forces. Two summation loops were needed in the UDFs, one for each airfoil because they have different zone pointers and the loops can only accept one pointer at a time. Once the forces are summed into the total lift and drag forces for each airfoil the perpendicular resultant force can be calculated using Equation 7 and Equation 8. Now the torque can be calculated and fed into the linearized differential equation and the new angular velocity is calculated and can be converted to the tangential velocity. Recall that it is from this tangential velocity the Cartesian velocity approximations are calculated and fed into FLUENT to move the airfoils. (13)

In addition to the governing mathematical scheme calculations there are many other necessary calculations. These included the tracking of variables for the output data files and conditions that control the airfoils' angle of attack. The most crucial of these variables is the system angle of attack, β . This quantity is used significantly throughout the mathematical scheme and needs to be updated at every time step as the system angle will change every time there is a vertical displacement. The calculation is based off the vertical, Y_l , position of the leading airfoil and the distance from the airfoil pivot to the system pivot point through Equation 12.

$$\beta = \sin^{-1}\left(\frac{Y_l}{r}\right)$$

Equation 12- System Angle of Attack

In order to perform this calculation, the Y_l value must be calculated. The positions of the airfoils are tracked by adding the displacements at each time step to the location at the previous time step. The current displacement is found by multiplying the Cartesian velocity components by the time step size, which results in a distance and is demonstrated in the following example given in Equation 13.

$$\phi_n = \phi_{n-1} + \frac{d\phi_n}{dt} * \Delta t$$

Equation 13- Variable Tracking

Where \square represents any of the four position variables and the differential represents the Cartesian velocity. For both the system angle and the position variables, an initial value must be supplied to the UDF at the source code level. Care was taken to ensure that these values were consistent with the mesh that would be used. It was found that an initial system angle of zero was ideal because it simplified both the meshing scheme and the initial variable values specified in the UDF.

One of the most important features of the UDFs is the portion that controls the airfoils' angle of attack. If this portion is not included, the analysis would just drive the airfoils in one direction until they overlapped with the top and bottom boundaries of the domain, which would cause the analysis to fail. The point at which to begin changing the angle of attack of the airfoils comes from selecting a maximum, or minimum, vertical limit. Although this location can be chosen as any vertical location, for this analysis it was chosen based off a maximum desired system angle. If the airfoil passes this maximum, the angle of attack will begin to reverse until it has reached the specified angle. This process repeats when the airfoil reaches the other maximum on the other side of the X axis, and the desired oscillatory motion is produced. This process is accomplished in the UDF source code using two *if* statements, one for the upper limit and one for the lower limit. The actual angle of attack change velocity is sent to FLUENT using the *omega[2]* array in the dynamic mesh macro. This angular velocity will act about the center of gravity which is specified in the dynamic mesh GUI screen and is tracked by FLUENT as the airfoils translate through the domain. The 2 index specifies that the rotation will be about the Z , or out of plane, axis. (13)

In order to be able to post process the analysis, data files from each of the UDFs were written to the FLUENT working folder as the analysis progressed. The data files, named *data.txt* and *data2.txt* for the leading and trailing airfoils respectively, are tab delimited text files that were appended with the current time step variable values after every time step. Some of the variable outputs included in both of the data files were the current solution time, the airfoil force breakdowns and totals, the system angle of attack, and the system angular velocity. Each data file also included the velocity and position of the airfoil that corresponded to that data file, as well as the tracking of the angle of attack and the rate of change of that angle of attack. The only variables exclusive to one data file were the torque, power, and spring torque outputs and were included only in the *data.txt* leading airfoil file. These were only added to the leading airfoil output file because they were not originally included and it had already been determined that the two UDFs were performing and outputting identical calculations. The actual post processing of these data files will be discussed later in this chapter..

The last consideration that went into writing the UDFs comes from the necessity of the analysis to be performed using parallel computation. In parallel computation, the computational domain is split into a certain number of partitions and FLUENT performs the computations on these partitions simultaneously, thus reducing the computational time required to get a solution. The number of partitions used is based off the number of parallel computing processes used by FLUENT. Parallel computing was required in order to minimize the amount of computing

time required for each analysis. By reducing the amount of computing time used for each analysis, more cases with varying parameters could be run, thus providing a wider range of test data for comparison.

Before explaining the parallel considerations needed in the UDFs, it is important to have at least a brief understanding of how FLUENT handles the parallel calculations. Each of the aforementioned domain partitions gets loaded into a different compute process called a compute node. These nodes perform the same calculations simultaneously on the different partitions. There is also a host process, which acts as the interface between the FLUENT GUI and the compute nodes. The host sends the commands it receives from FLUENT to one of the compute nodes, which in turn passes the commands to the rest of the nodes. The nodes are connected virtually through a communication, or message passing system that is used to pass overlapping data and synchronization information. (13)

Problems arise for UDFs in parallel processing because both the host and node can execute commands and calculations as specified by the UDF. An example of this being a problem is the printing of messages. If the UDF is not written to specify that only the host should print messages to the FLUENT console, the message will print as many times as there are nodes. To overcome this FLUENT provides special macros and compiler directives in order to parallelize the UDFs. These compiler directives specify whether a section of code should be executed by a serial process, a host process, or nodal processes. By properly implementing these directives, the UDF can be run in serial or parallel mode. Specifying what directive to use for each part of the UDF code is challenging and takes considerable understanding of the both the desired calculations and how FLUENT will execute those calculations. (13)

One of the main areas of the UDF codes used in this implementation is the calculation of the airfoil forces. Recall that these calculations were based on an integration of the force values on the surface of the airfoils, and that with parallel processing the domain is split into partitions, each being calculated by a different compute node. If parts of the airfoil reside in different partitions, the force integration has to get the force integrations from all of the partitions and add them together in order to have the entire force represented. This can be accomplished relatively easily by taking advantage of the compiler directives and special parallel macros. (13)

First, the compiler directive that states the code should be ignored by the host process is called as shown below. The *!* is the C programming directive for *not*, which in this instance will direct the code to be executed by anything but the host process, leaving only serial and node processes as desired. This directive acts as a special type of *if* statement, such that it needs termination at the end of the code.

```
#if !RP_HOST
```

```
Code to be executed by either the serial or node processes
```

```
#endif
```

Within the cell face integration loops, a parallel macro is called to check if the node is the principle compute node for each particular face. This is important because the cells at the boundaries of the partitions would be integrated twice if the primary node check was not implemented. It should be noted that this principle face check is acceptable for use by a serial process because it always returns a value of *true* as there is only one mesh partition. The macro is given below where *f* is a face pointer and *t1* is again the surface zone pointer. This macro acts exactly like a normal *if* statement in that the brackets around the conditional code are required. (13)

```
if PRINCIPAL_FACE_P(f,t1)
{
    Force summation code
}
```

At this point in the UDF, each node has a force value from the integration loop and these individual forces need to be summed to get the total force acting on the airfoil. This is accomplished by using a global summation macro within a compiler directive that only executes on nodes. An example of this code is given below where the *TotalVariable* is being computed through the sum of the *variable* across all compute nodes through the PRF_GRSUM1 macro. (13)

```
# if RP_NODE
    TotalVariable = PRF_GRSUM1(variable)
# endif
```

Because all of these calculations have been executed by compute nodes, the force data needs to be sent to the host to be used in the calculation of the mathematical scheme, and is accomplished using another parallel macro designed to transfer data between the nodes and host. An example of this data passing is given below where *double* is the type of data and *N* is the number of variables to be passed. It should be noted that when running serial calculations, this part of the code is ignored by FLUENT. The host process then performs all of the calculations necessary to get the Cartesian velocities and sends those values back to the nodes to move the airfoils for each time step. The host process also handles the writing of the output data files. (13)

```
node_to_host_double_N(N variables);
```

By implementing the mathematical scheme into the UDFs using the above methodologies, the leading and trailing airfoil UDFs can be used on either Windows or Linux and be run in either serial or parallel. The system parameters can be easily changed at the beginning of the UDFs to allow different cases to be created and run efficiently.

4.4 – VERIFICATION

Ideally, the verification of the CFD solutions and implemented mathematical scheme would be conducted by comparing solution output, both steady state and dynamic, to experimental data. Unfortunately, there is no experimental data to compare the CFD output to, even for the steady state case. All of the published experimental data for tandem airfoil setups were too dissimilar to the current research to be able to make a proper verification. Due to the unknown effect of the wind tunnel walls in the CFD solution, verifying a single airfoil in the dynamic computational domain would not be a valid comparison. The dynamic CFD solution setup and mesh sizing are very similar to that of the validation cases presented in the previous two chapters, which helps substantiate the validity of the dynamic solutions.

In order to ensure the math scheme derived above would provide the desired motion of the system, and subsequently be properly implemented within the FLUENT UDFs, verification codes were written. The first code was a MATLAB code that implemented the mathematical scheme and provided some sample system outputs. The goal of this code was to show that for a given constant torque and moment of inertia, the scheme would provide rotational motion and constant angular acceleration. The goal of the second code, which was written in C, was to ensure the implementation and calculation of the linearized governing differential equation in the programming language required for UDFs would provide the same outputs as the MATLAB verification code. The comparison parameters between the MATLAB and C codes are shown in Table 4 below, where I is the moment of inertia, τ is the torque, dt is the time step, and N is the number of time steps.

Parameter	Value
I	1
τ	3.2
dt	0.1
N	5

Table 4- Verification Parameters

The first verification output is the position plot of the airfoil pivot points and can be seen in Figure 19 below. By examining the movements of the airfoils, it is qualitatively suggested that the mathematical scheme is producing rotational motion about the origin. The rotation is in the positive direction, which is correct based on the positive constant inputs. The leading airfoil is moving upward, while the trailing airfoil is seemingly moving completely opposite, as it should. The positions seem to be moving farther apart increasingly fast, which is what is expected from a constant angular acceleration.

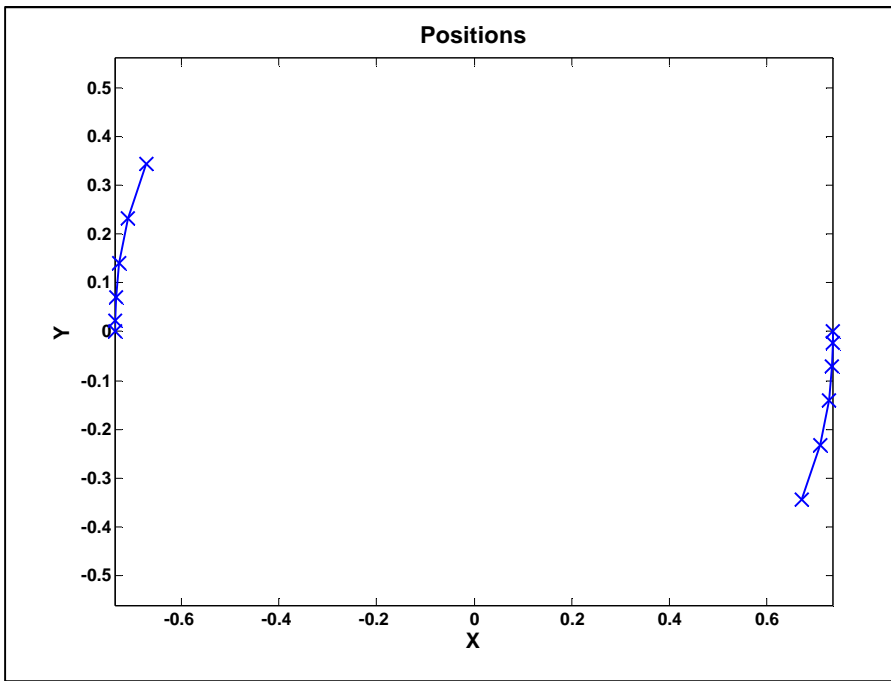


Figure 19- Position Verification

The constant angular acceleration is supported in Figure 20, which depicts the system angle, β as a function of time step. This same trend of increasingly larger changes in system angle is much more clearly visible in Figure 20 than Figure 19. Qualitatively, by comparing the final system angle to the angle between the leading airfoil and the X axis at the final time step in the previous figure, the 28° calculated value seems accurate.

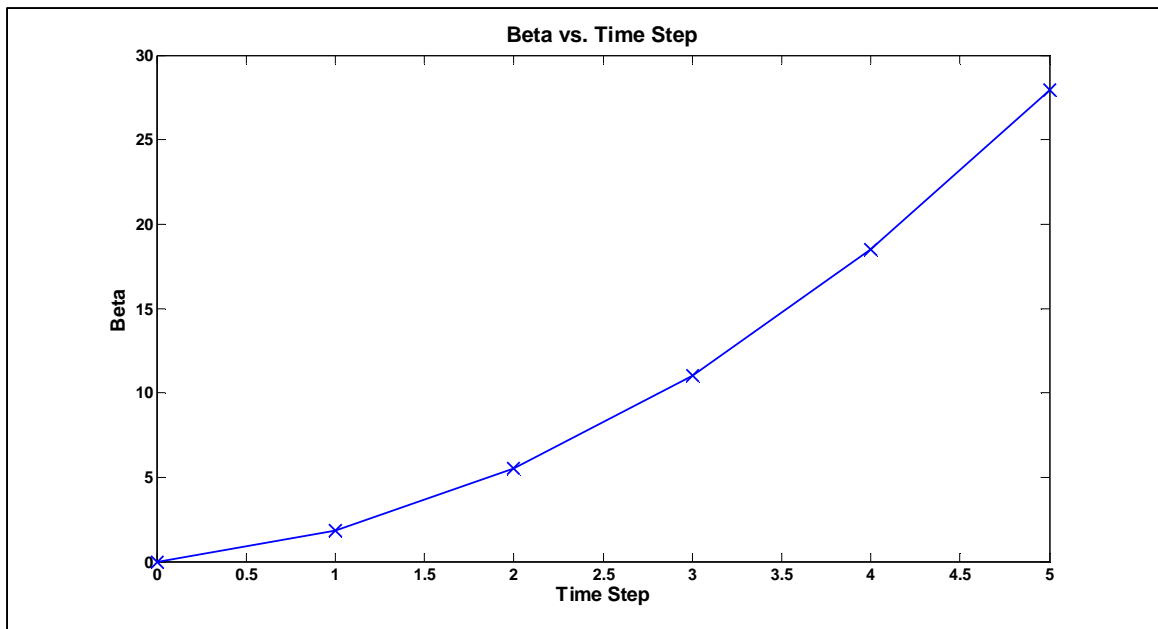


Figure 20- System Angle Verification

Due to the rotational motion being approximated by the Cartesian displacements, it was important to measure the error introduced into the system by this approximation. This was accomplished by looking at the distance between the airfoil pivot points which, again, are the *center of gravity* points of which FLUENT will move the airfoils. If the motion was purely rotational, the distance between these points would remain constant because the points would remain equidistant from the origin. The percent error of the current pivot point distance with respect to the starting distance was calculated and plotted in Figure 21 below.

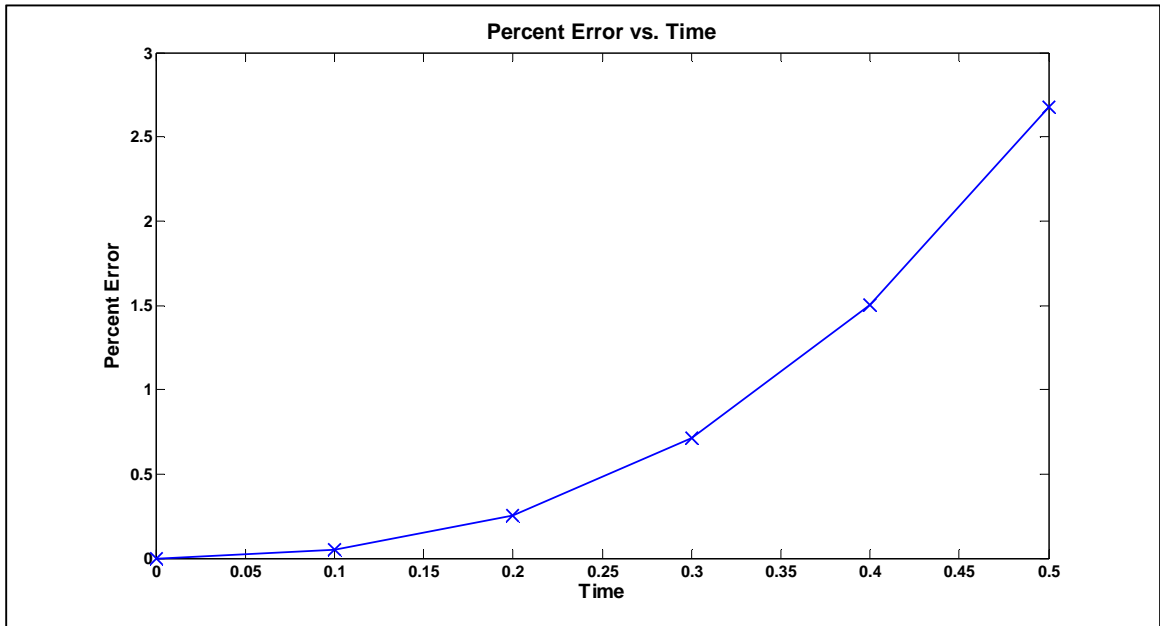


Figure 21- Error Verification

The accumulated error over the .5 seconds was less than 3%, which was due to the rather large time step chosen for this verification. The expected behavior of this approximation, however, is that the error should decrease when the time step is decreased. This is because the Cartesian approximation is based heavily on the time step, much like the linearized governing differential equation. This behavior is further explored in Table 5 below.

dt	N	% Error
0.1	5	2.6794
0.01	50	0.2096
0.001	500	0.0204

Table 5-Error Verification

As expected, the percent error decreased as the time step was decreased. The decrease in percent error was approximately an order of magnitude for every order of magnitude decrease in time step. From this investigation, it was concluded that the error introduced by the Cartesian approximation was small and should not introduce any noticeable motion degradation in the dynamic solution if the time step was kept small.

The final output was the angular velocity, ω , and was plotted with the output from the C verification code. As shown in Figure 22 below, the angular velocity increases linearly and the outputs from both the MATLAB and C codes match perfectly. The constant angular velocity that was suspected in the previous figures is clearly shown here through the linearly increasing angular velocity. The fact that the C code provides identical output as MATLAB gives confidence to the calculations being performed by FLUENT through the UDFs.

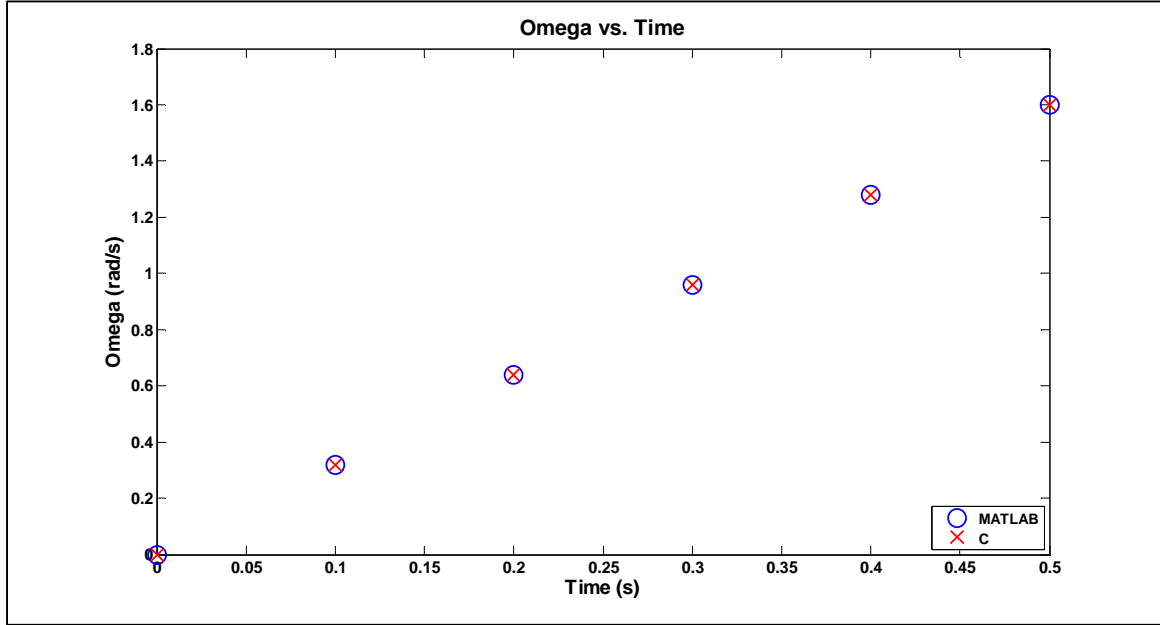


Figure 22- Angular Velocity Verification

One of the most crucial calculations performed by the UDFs is the calculation of the forces acting on the airfoils. The entire mathematical scheme is based off these forces creating the driving torque on the system. Due to this importance, it was crucial to verify the UDF force calculations and compare those forces to the ones FLUENT can provide through the GUI (8). Table 6 below shows the UDF forces compared to the FLUENT forces taken from the last time step of a sample case.

Force	FLUENT	UDF	Abs Error	% Error
X Pressure	0.13865	0.13724	0.00141	1.01954
X Viscous	0.03435	0.03405	0.00030	0.87384
X Total	0.17300	0.17129	0.00171	0.99061
Y Pressure	-0.80052	-0.78885	0.01167	-1.45764
Y Viscous	0.00162	0.00148	0.00014	8.56002
Y Total	-0.79890	-0.78737	0.01153	-1.44324

Table 6- Force Verification

The highest percent error of approximately 8.6% comes from the smallest valued force, which was the Y direction viscous force. This force also had the smallest absolute error, which

leads to the conclusion that although this force had the highest percent error the calculation is sufficiently accurate. One of the sources in error for these calculations could be rounding error through the UDF integration loops and secondary additions. In addition, the method FLUENT used to calculate the forces is unknown and the method used internally could differ from the UDF implementation.

It has been shown that the derived mathematical scheme provides the desired rotational motion through an approximate Cartesian displacement method. This approximation introduces sufficiently minimal error into the motion provided a time step is small. Both the MATLAB and C verification codes provide an identical and accurate calculation of angular velocity through the linearized differential equation. The force calculation from the UDFs has been verified against the built in force reporting from FLUENT by showing the percent error between the two values is minimal. All of these factors contribute to the conclusion that the mathematical scheme and implementation within FLUENT are valid.

4.5- DYNAMIC SOLUTION SETUP

The solution setup for the dynamic analysis was very similar to that of the validation cases as suggested to keep the CFD solutions as valid as possible. The main difference is the addition of the top and bottom wind tunnel walls and the dynamic meshing parameters as well as the transient formulation. FLUENT version 14 was used for the dynamic analysis and was loaded using the two dimensional, double precision, pressure solver. After loading, or compiling, the desired UDF library the mesh file can be read into FLUENT. The only major difference in the mesh files used for the various cases was the initial airfoil AoA. Care was taken to ensure the AoA coded into the UDF was the same as that of the mesh file. The mesh was checked for errors and the domain was reordered to optimize the computational performance. This reordering process rearranges the cell locations in the memory such that neighboring cells are near each other, which decreases the bandwidth of the mesh in memory and allows for a more efficient computation. The turbulence model used was again the k-omega SST model that was explained in detail in the validation section above.

The boundary conditions were kept constant through all of the test cases to keep the predicted performance focus on the system parameters. The wind tunnel inlet was set as a velocity inlet with a 5 meters per second airflow in the positive X direction. This velocity was chosen to reflect the most probable experimental wind tunnel velocity and represented a Reynolds number of approximately 52,000. The turbulent boundary conditions at the inlet were the same as the ones used in the validation case, whereby the turbulent intensity was set at 2% and the turbulent viscosity ratio was set at 10. These values again came from the FLUENT user's guide, as no experimental values had been determined due to the lack of wind tunnel testing. If experimental data was ever collected, the turbulent parameters of the flow should be implemented in the CFD analysis. The wind tunnel outlet was set as a pressure outlet. The top

and bottom wind tunnel walls were set as slip walls such that no boundary layer would build near them. This was accomplished by setting the specified shear pressure to 0. This slip wall condition was done to minimize computational cost as the near wall mesh would not have to be refined to meet the requirements for the turbulence model and resolve a boundary layer. The airfoils were set as no slip walls because the boundary layer is important in resolving the viscous forces acting on the airfoil. (8)

Some of the exact values of the dynamic meshing parameters were dependent on the mesh sizing and scale information of the individual meshes used, however, the methodologies used to set these parameters applied generally to all of the meshes. These values also did not vary a great deal due to the identical sizing parameters used in the mesh generation. The two dynamic mesh methods employed were the smoothing and remeshing schemes. The smoothing scheme idealizes the cells between moving boundaries as springs. The cell movement is based on how much displacement force is applied through a spring constant and boundary displacement. The spring constant used in the dynamic analysis was set to 0.0001, which represents a strong spring and helps the boundary nodes keep their shape. In order to prevent negative cells due to the mesh folding onto itself, the boundary node relaxation was kept at the default value of 1. Through experimentation, better quality meshes were created if the convergence tolerance was decreased an order of magnitude to 0.0001 and the number of iterations was increased to 150. (8)

The local cell remeshing method was also used at every time step. The default mesh scale values were used as a starting point for the remeshing parameters. The maximum size was decreased by an order of magnitude and the cell skewness was set to 0.4 because the default values for these parameters resulted in poor mesh quality. The sizing function was used in addition to the remeshing size parameters. The sizing function is used to assist remeshing by adding cell size distribution criteria to the remeshing parameters. The default resolution of 3 was used and the maximum variation and rate values of -1 and 0.99 respectively were used. The variation parameter is how large the cells can be with respect to the closest boundary cell, so by setting it at -1 the size of the cells was minimized. The rate parameter controls how rapidly the cells grow away from the boundary and the 0.99 setting used leads to the slowest transition. These sizing function parameters helped minimize the amount of mesh degradation as the airfoils oscillate and the solution progresses. (8)

The last dynamic mesh setting was creating the appropriate zone conditions. For the leading and trailing airfoils, the motion type was set to rigid body and the movements were hooked to appropriate UDFs. The initial center of gravity locations were set to an X value of -0.7366 and 0.7366 meters for the leading and trailing airfoils respectively. These values were based off the geometric locations of the airfoil pivot points at a 0 AoA. The adjacent cell size was specified as 0.0001 as this value was close to the near wall cell size, if not smaller. The fluid zone was set to be deforming and used similar size parameters to what was used in the local cell remeshing method. The deforming zone was selected to have both the spring based smoothing and remeshing applied. (8)

All of the spatial discretization schemes were set to second order, as was used in the validation cases. The pressure- velocity coupling scheme was changed to the Pressure-Implicit with Splitting of Operators (PISO) scheme as was recommended by FLUENT for all transient analyses. The PISO scheme is stable over a larger range of time steps for both pressure and momentum than the SIMPLE scheme used in the validation cases. The transient discretization used was a first order implicit scheme. The solution was initialized from the velocity inlet, which gave the flow field an initial velocity of 5 meters per second. This simulates the wing oscillator being released from a locked position after the flow is up to a steady speed. (8)

4.6 – RESULTS

All cases presented in this section were run for 500 time steps using a step size of .01 seconds, which gave an ending analysis time of 5 seconds. This total time was generally sufficient to allow the system to oscillate at least twice and settle into a more steady state oscillation, which was desired in order to obtain proper data. This time step was chosen to both provide significant analysis resolution as the system oscillates, as well as to avoid any re-meshing errors due to large body displacements. There was no stability requirement for the time step size due to FLUENT using a fully implicit computational scheme (8).

The absolute convergence criterion for the scaled residuals at every time step was set to 10^{-4} and was met at almost every time step. This convergence value is an order of magnitude lower than what FLUENT recommended for most computations (8). While monitoring the convergence it was noticed that at a couple seemingly random time steps, the solution would not converge to the desired criterion. Generally, when this non- convergence occurred, the turbulence production residual, k , would converge to a value just decimal places above the criteria and the analysis would run to the designated maximum number of iterations per time step, which was set at 500. Due to the randomness of the non- convergence and the residual value of the variable converging so close to the desired criteria, it was determined that this non- convergence had a negligible effect of the solution. Typically, convergence occurred at each time step after less than 100 iterations.

4.6.1 – BASE CASE RESULTS

A base case was run with general parameters to show that the FLUENT and UDF setups and computations were providing the desired solution output, as up to this point the verifications were completed separately. These parameters used can be found in the Table 7 below, where AoA is the absolute AoA of the airfoils, β is the system angle at which the airfoils begins to reverse AoA, S is the spring constant, and dA is the number of seconds used to reverse the airfoil AoA. The first indication of the successful completion of the analysis was that the solution ran for all 500 time steps without error. Although a positive sign, completion of all the

desired time steps does not positively ensure the system worked as desired. For example, all time steps could be completed and the motion could be wrong and the airfoils could not move at all or move incorrectly. The second indication of successful completion is if the position of the airfoils after the final time step is different from the initial position. If calculation time permits, FLUENT allows for the recording of animations. These animations can be of any of the graphical outputs available in FLUENT, including pressure, velocity, or mesh. Further, partial solution case files can be saved at a desired interval of time steps and the airfoil positions at these intervals can be referenced. Figure 23 and Figure 24 show the movement of the airfoils at different time steps for the base case. Recall that the system pivot point was located in the center of the computation domain and that the airfoils should pivot about this point. By examining Figure 23 and Figure 24 qualitatively, the airfoils appear to be rotating about center of the computational domain.

AoA	β	S	dA
12	10	0.2	0.05

Table 7- Base Case System Parameters

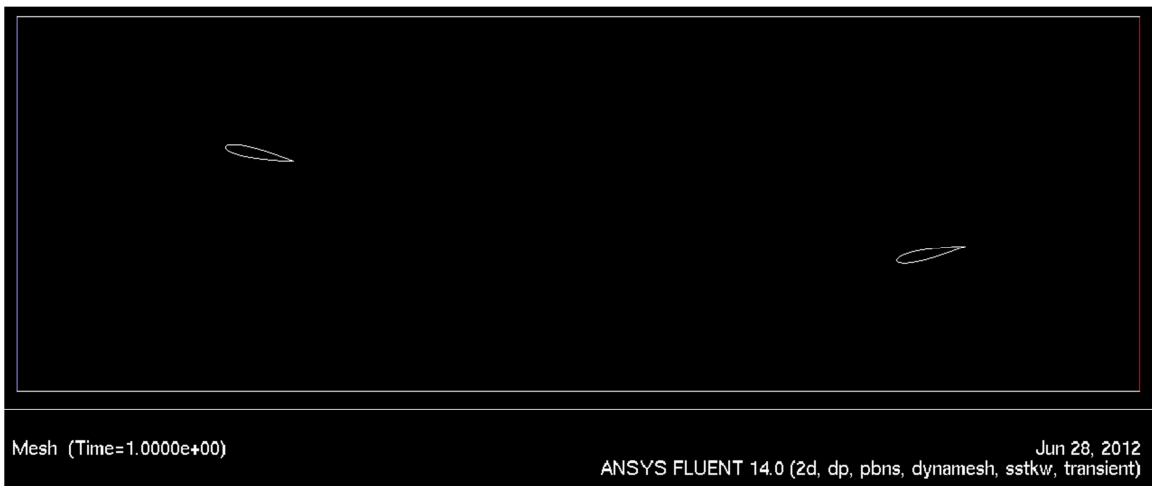


Figure 23- Airfoil Movement t=1s

If the time resolution of the saved intermediate case files is sufficient, or if animations are available, the aerodynamic characteristics of the flow can be examined. One of the more interesting characteristics of the flow is the interaction of the wake from the leading airfoil with the trailing airfoil and can be explored by analyzing the velocity magnitude contours. In general, the large distance between the leading and trailing airfoils has minimized the flow interaction, as the wake from the leading airfoil tends to dissipate and become less intense by the time the wake reaches the trailing airfoil. Alternatively, the large distance can have the opposite effect because it limits the vertical travel due to the wind tunnel walls keep the trailing airfoil in the vertical vicinity of the leading wake. If the airfoils were closer together, a larger system angle could be achieved, putting the trailing airfoil more out of the way of the intense wake at the extremes of vertical travel.



Figure 24- Airfoil Movement t=2.5s

For the base case, the wake interaction for a steady, not the initial transient movement, period of oscillation starts with the leading airfoil moving downward and the trailing airfoil moving upward with both passing through the X axis. At this point in the oscillation, the wake extending from the leading airfoil is small and the trailing airfoil is slightly interacting with an area of higher velocity in the near wall region. Additionally, the computational solution shows a higher velocity of the flow on the side of the airfoil in the direction of the vertical movement. It should be noted that the true interaction of the airfoils with the near wall region is not modeled accurately due to the slip wall boundary condition on the top and bottom walls of the wind tunnel, which prevents a boundary layer from building up to save computational time.

When the system reaches the maximum desired angle, the AoA of the airfoils reverse in 0.05 seconds as specified by the case parameter. This rapid change causes the area of high velocity that was on the movement side of the airfoil to be shed from the airfoils and propagate in the flow direction along the wind tunnel wall. While the high velocity is shed from the movement side, a lower velocity area is shed from the opposite side of the airfoil at approximately the same time the AoA change is completed. The low velocity wake shed from the leading airfoil travels in the flow direction where it dissipates and appears to have little

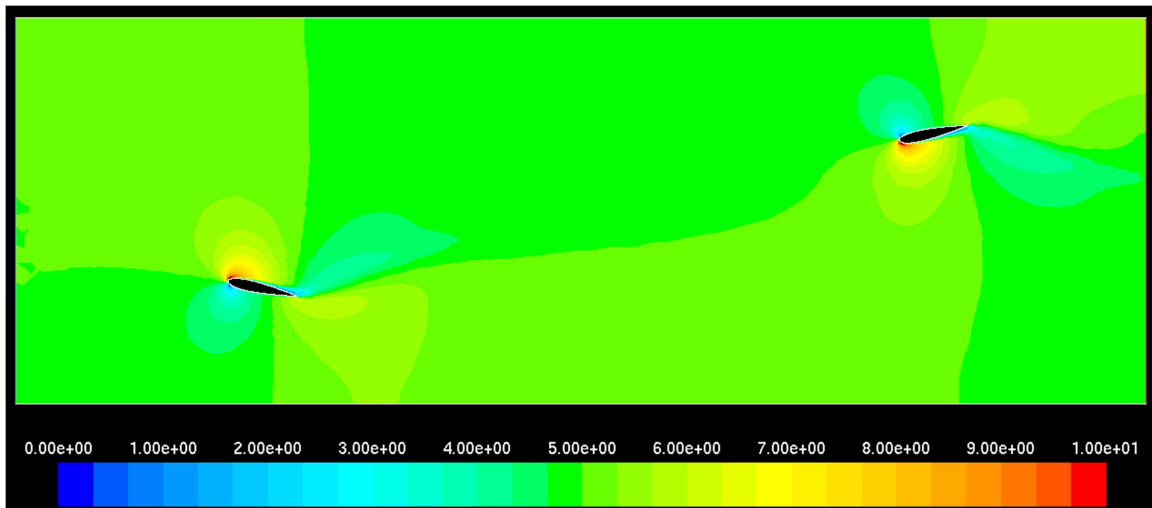


Figure 25- Base Case Wake Shedding After AoA Change

interaction with the trailing airfoil. Figure 25 shows the high and low velocity wakes being shed from the airfoils.

After the angle of attack changes, the system continues to rotate in the negative direction due to inertia. This opposite movement with respect to the absolute angle of attack causes the airflow to separate for approximately 0.2 seconds while the system changes the direction of rotation. Although separation generally means an airfoil is stalled and losing lift, the highest lift forces are experienced during this time. The peaks shown in Figure 26 correspond to the beginning of the separation, with the small minima peak immediately after occurring at the reattachment point. The somewhat flat areas after these peaks represent the general vertical travel of the airfoils. Because the highest lift force was experienced during the separation, the conclusion was drawn that there must be some dynamic stalling effects occurring, which prevents the significant loss of force normally attributed to a statically stalled airfoil. This conclusion is supported by the fact that Lee found similar phenomena when investigating the flow over oscillating airfoils (14). Dynamic stalling is a phenomena experienced by oscillating

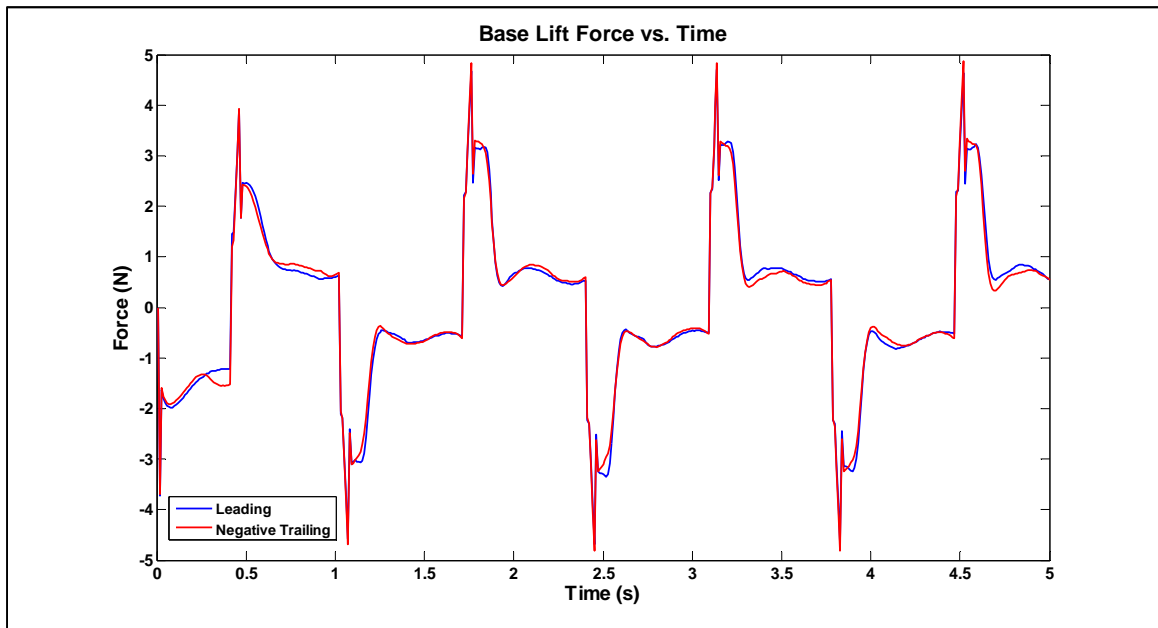


Figure 26- Base Case Lift Force

airfoils during a rapid AoA change. It is characterized by a vortex formed from the leading edge and causing a brief increase in lift (15). The lift increase generally lasts until the leading edge vortex is shed from the trailing edge. After the vortex is shed, the lift significantly decreases and the airfoil experience normal stalling characteristics.

The dynamic stall effect conclusion is supported by examining the effective AoA of the airfoils. The effective AoA is the true local AoA experienced by the airfoil and is the vector subtraction of the airfoil velocity from the flow velocity. The effective AoA for the leading airfoil is shown in Figure 27.

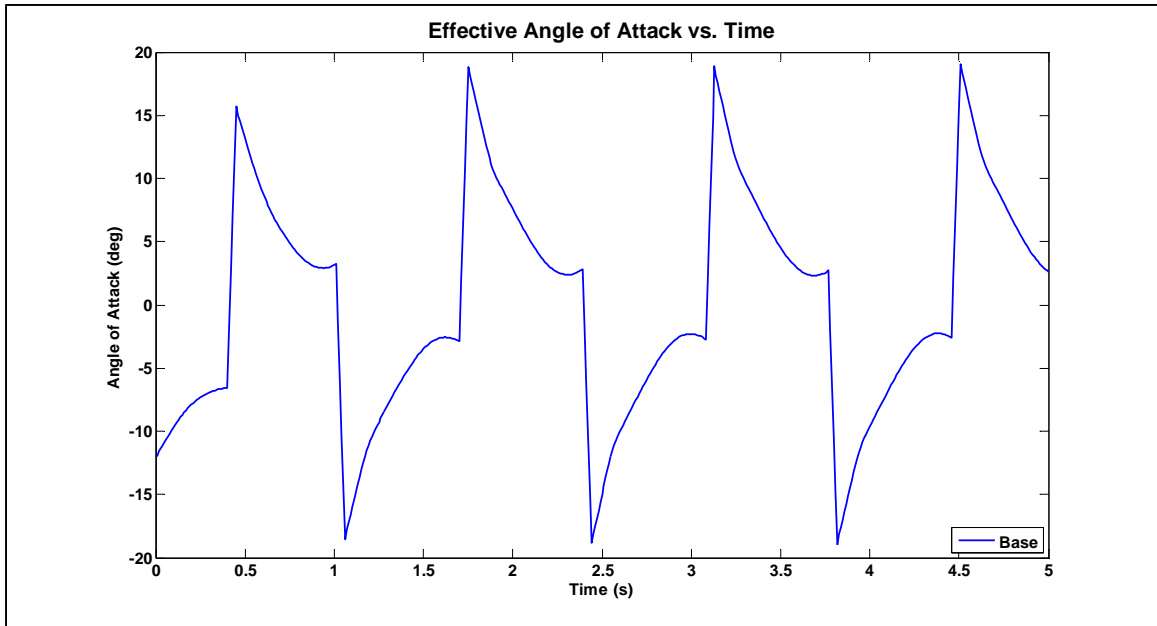


Figure 27- Base Effective AoA

Only the leading airfoil data is shown as the trailing airfoil has an identical, but opposite, effective AoA. At the time of the highest lift force, which was also the time of the separation, the effective AoA is at its maximum. For the base case, the maximum effective AoA is approximately 18 degrees. The static stall angle of attack for a NACA 0015 at a similar Reynolds number is approximately 13 degrees and marked by a significant drop in lift (16). This initial peak is caused by the continuation of the system rotation after the airfoil AoA has been changed. After the initial peak, the effective AoA decreases as the vertical velocity of the airfoil increases in the same direction of the lift force. The effective AoA continues to decrease until

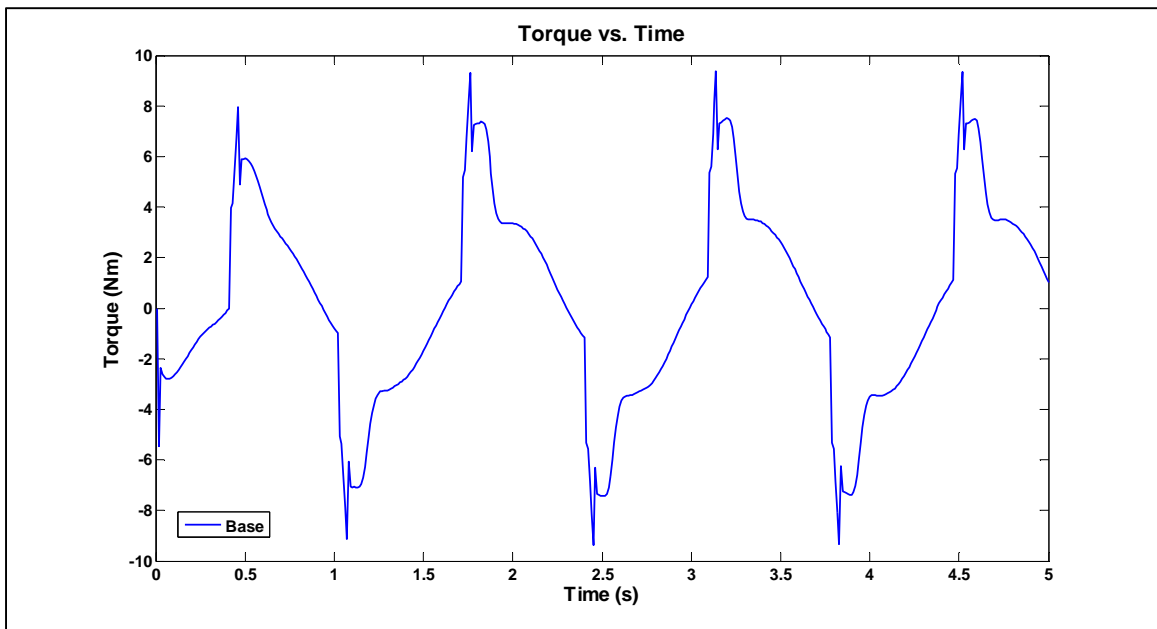


Figure 28- Base Torque

right before the airfoil AoA changes, where it actually increases slightly. This slight increase comes from the system torque no longer acting in the direction of rotation. This reversal in torque direction can be seen in Figure 28 and is the product of the system nearing its maximum travel, which causes the component of airfoil lift force, which is dominant over drag, acting perpendicular to the system to decrease and the spring force increasing as the system angle increases. In the base case, this area of torque reversal is small and seems to act as a damper as per the design inclusion of the spring.

The last aerodynamic effect of this separation is due to the low velocity created by the separated flow. Some of this low velocity area is shed into the wake during the flow reattachment process. This creates a second low velocity wake that interacts with the trailing airfoil slightly before the local AoA change occurs. By examining Figure 26 and Figure 29 the conclusion can be made that the wake interaction is minimal because generally no significant difference in whether the lift or drag forces is experienced. The largest difference can be seen in Figure 29 during the peaks in the drag force where the trailing airfoil tends to encounter a lower drag. This lower drag could possibly be attributed to the wake of the leading airfoil causing the trailing airfoil to experience a slightly lower flow velocity. This lower velocity has the potential for causing a slightly less severe separation, although this separation difference is not significantly pronounced qualitatively.

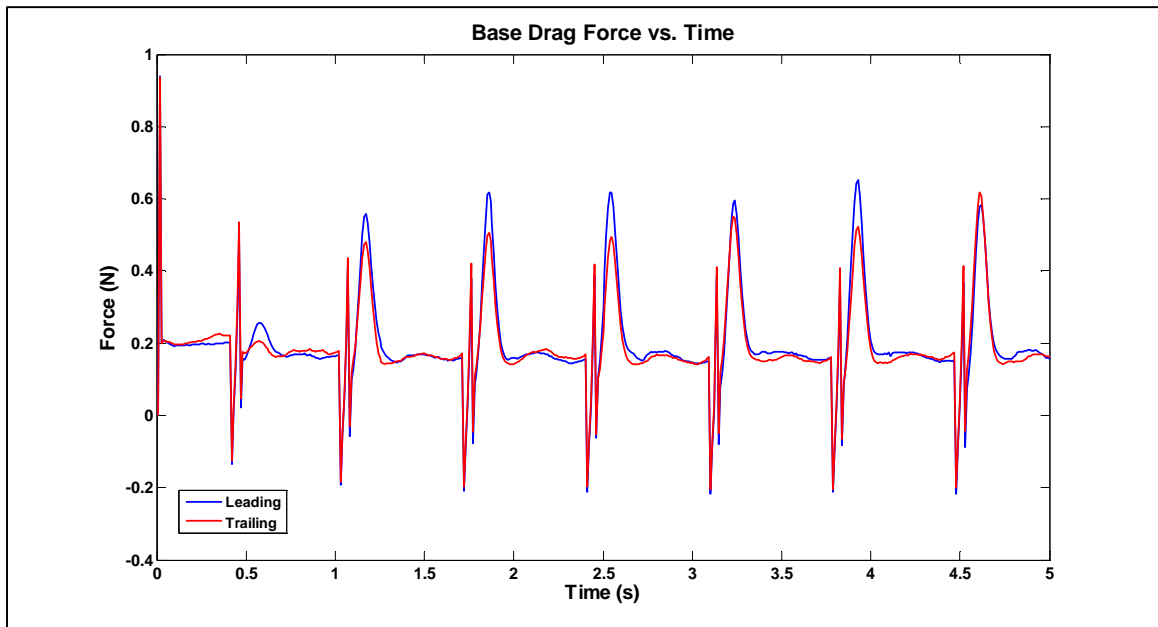


Figure 29- Base Case Drag Force

The peaks in drag coincide with the separation and are not consistent through every oscillation. This lack of consistency has two clear potential causes. First, because the separation is a very turbulent phenomenon, the randomness of the turbulence may be causing the separation to occur differently at the various oscillations. Additionally, the calculation could be running into some of the limitations of the turbulence model used, as the validation did not

cover separated flow. The second cause of the inconsistency could be due to the unknown flow interaction with the wind tunnel walls. Due to the movement of the airfoils, the proximity of the airfoils to the wind tunnel wall is at a maximum during the changing of the airfoil angle of attack. Without a rigorous experimental procedure and coupled with the exclusion of the wind tunnel wall boundary layer, the true effect of the near wall flow interaction is unknown.

One of the last interesting results from the base case is the behavior of the mesh throughout the analysis. Despite best efforts to control the mesh sizing and to keep the mesh resolution as close to the original mesh as possible, the mesh tends to degrade somewhat as the analysis progresses. The dynamic meshing parameters were selected to minimize the degradation and were chosen from experimentation and mostly qualitatively comparing the mesh resolution at the completion of the analysis. This degradation might be attributed to the near airfoil region coming in close proximity to the wind tunnel near wall region and the dynamic meshing parameters struggling to blend the mesh between the boundaries. Once the airfoils move away from the wall and back towards the center of the domain, the meshing parameters are unable to recapture the initial resolution of the near airfoil region. An example of the degraded mesh is shown in Figure 30 and can be compared to the initial mesh shown in Figure 18. The number of cells in the final mesh is approximately 30,000 compared to the 80,000 cells in the original mesh.

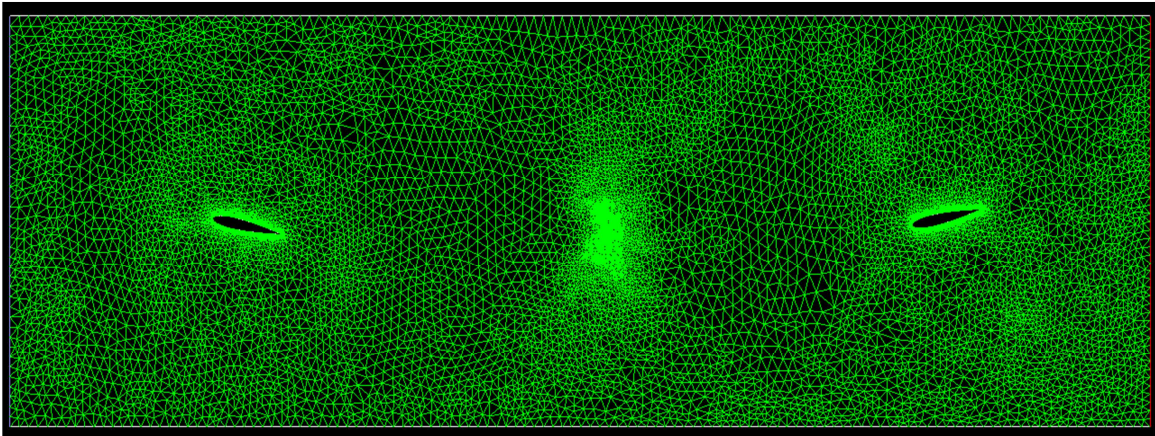


Figure 30- Degraded Mesh

Although the mesh does degrade, the solution converges at all the time steps and there does not appear to be any significant discontinuities when graphically plotting solution variables. Figure 31 shows the velocity magnitude contours that correspond to the degraded mesh shown in Figure 30. There is some slight jaggedness to the contour in front of the trailing airfoil, which corresponds to where the mesh is coarse. Although the degradation has been shown to not introduce significant error both qualitatively and quantitatively, the possibility still exists that there is some numerical diffusion occurring in the computational solution. However, without an experimental flow field to compare to, the full extent of the error is unknown.

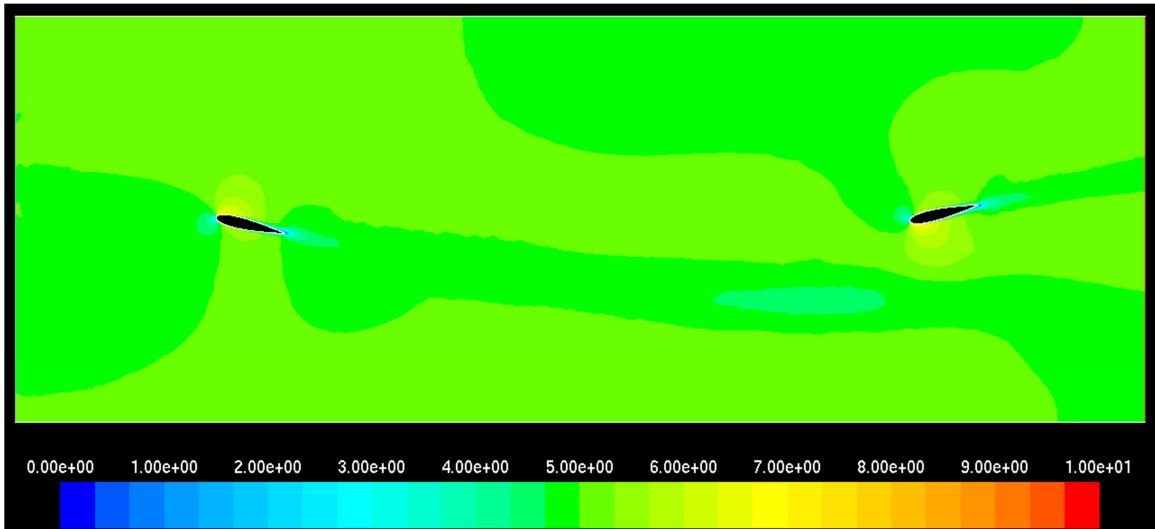


Figure 31- Base Case Final Velocity Magnitude

4.6.2 – PARAMETRIC STUDY

A parametric study was performed to examine impact on performance of the various system parameters. Each of the parameters was changed for one case and the results were compared. Including the base case, five cases were run using the parameters in the table below. The *AoA* case used a smaller absolute AoA and required a different mesh from the other cases as the absolute AoA can only be changed at the mesh generation level. Other than the one different mesh, all of the cases were set up identically to the base case and just used different UDF libraries that were compiled from UDF source codes employing the desired system parameters.

Case Name	AoA	β	S	dA
Base	12	10	0.2	0.05
AoA	8	10	0.2	0.05
Beta	12	15	0.2	0.05
Spring	12	10	0	0.05
dA	12	10	0.2	0.1

Table 8- Parametric Study System Parameters

To give a general idea of how these system parameters affect the system performance, the period and frequency of the cases were examined first. Table 9 below shows the data for all of the cases. Without examining any graphical output data, the results shown in Table 9 are what were expected. The *AoA* case had a longer period than the Base case due to the lower airfoil forces causing a slower system rotation. In fact, the base case had the highest frequency of all the cases, which happened to be a completely unintended consequence of the parameter choices. The beta case had a longer period due to the system having a larger allowed rotation before the reversal of the airfoil AoA's were reversed. Without a spring constant providing

dampening, the spring case had the longest observed period and was almost twice as long as the base case. The dA case had a very similar period to that of the AoA and beta cases. It should be noted that both the period and frequency are calculated from the last complete oscillation in an attempt to sample the most steady state data.

Case Name	Period	Frequency
Base	1.1934	0.8380
AoA	1.4033	0.7126
Beta	1.4999	0.6667
Spring	2.3769	0.4207
dA	1.4209	0.7038

Table 9- Parametric Study Period and Frequency

The period and frequency data can easily be supported by examining the plot of the system angle as shown in Figure 32 below. First, all of the 12 degree AoA cases exhibit identical initial behavior due to the spring effect near the 0 degree system angle being negligible. The spring cases begin to diverge from the other 12 degree AoA cases as the spring torque builds. The AoA case, with an 8 degree absolute AoA, initially starts slower than the other cases due to a lower system torque that will be shown later. This lower torque can be attributed to the lower airfoil force due to the reduced AoA.

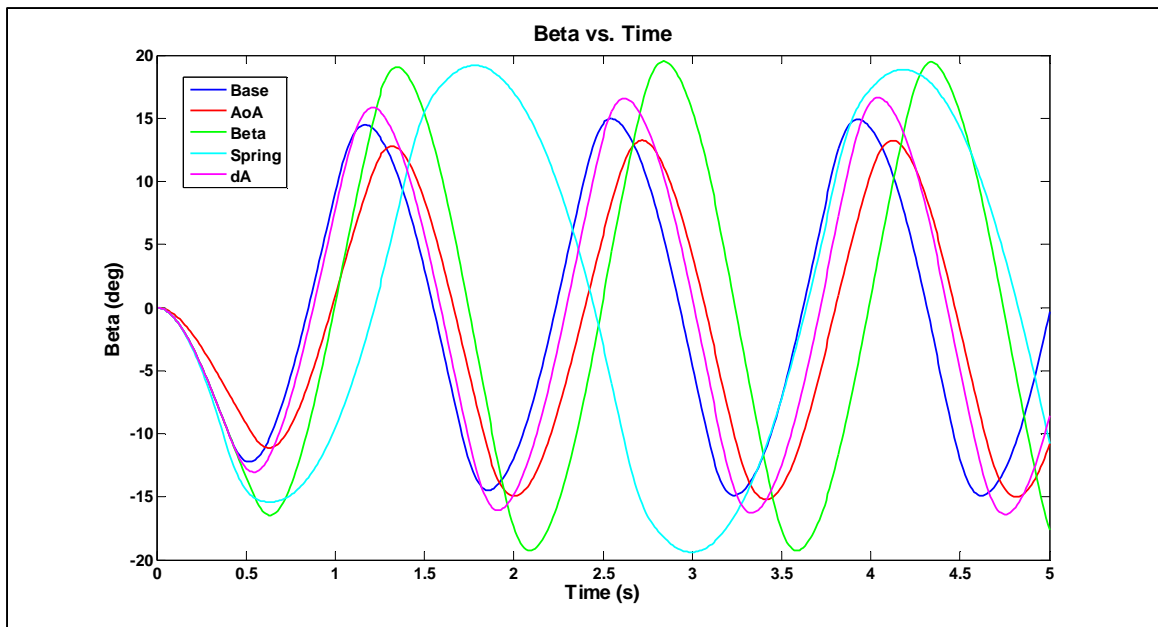


Figure 32- Parametric System Angle

The AoA cases takes the longest to reach the AoA reversal angle, which again is attributed to the lower force. This case also has the lowest maximum system angle and reaches just past the designated reversal angle of 10 degrees. Inertial effects account for the system angle minima, as the vertical velocity achieved with this case is smaller compared to the other

cases. This allows the system to reverse its rotation direction faster than other cases as the inertia is not as strong due to the lower vertical velocity.

The beta case, by parameter definition, had a larger maximum system angle. Interestingly, the overshoot of the reversal angle is less for the beta case compared to the base case, and exhibited an overshoot of approximately 4 degrees compared to the 4.5 degrees of the base case. It would be expected that the beta case would have a larger overshoot as the system would have more time to build angular velocity. Factors that could cause this lessened overshoot could be the effects of the effective airfoil AoA, reduced force projection acting in the torque direction, or higher spring torque dampening.

The spring case had the most interesting behavior and had a similar maximum angle as the beta case although the AoA reversal parameter was the same as the base case. As stated above, initially this case accelerated faster due to not having to fight the torque of the spring. When the reversal angle is reached, the spring case takes the longest out of all the cases to reverse the rotation direction. This behavior is characterized by a much shallower peak in system angle as well as a very high overshoot of approximately 9 degrees. The shallow peak and large overshoot are a direct result of not having the built up opposite torque dampening that is supplied in the other cases by the spring.

The dA case has a very similar behavior to the base case even though the airfoils reverse AoA in twice the time. This added time increased the maximum angle the system reached by approximately 1.25 degrees. The most effect this has was to give the dA case a phase offset from the base case, whereby the base case leads the dA case in time. This phase shift is increased every time an AoA reversal occurs.

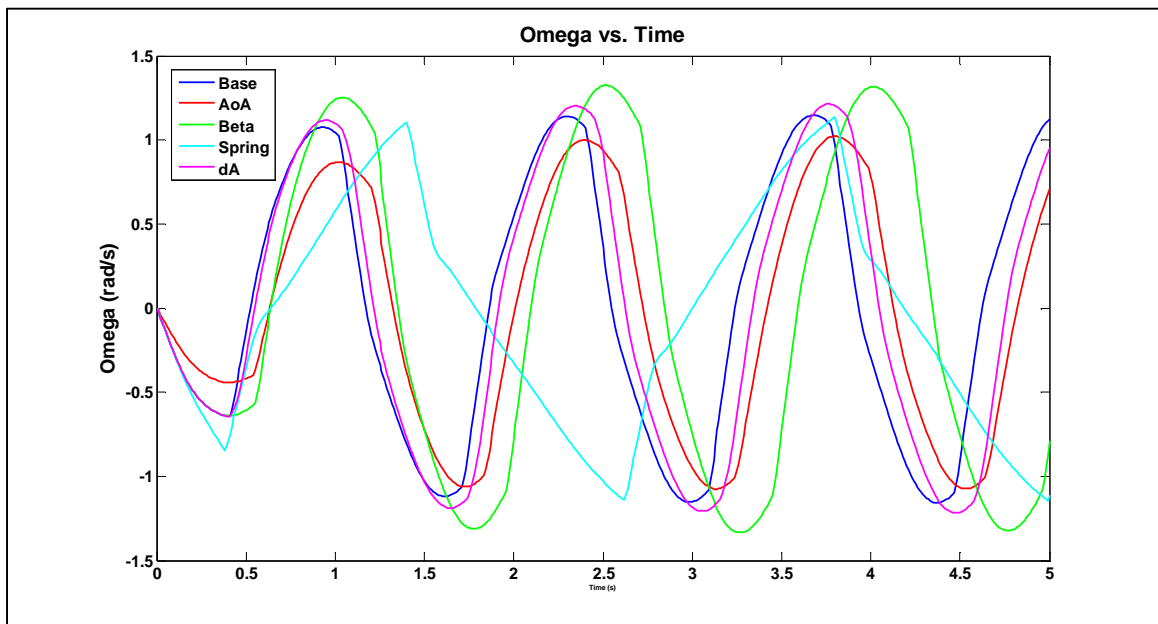


Figure 33- Parametric Angular Velocity

The identical initial behavior of the 12 degree AoA cases is again found in Figure 33, which depicts the system angular velocity as a function of time. Again, the spring case is the first to deviate from the other cases due to the lack of the restrictive spring torque. The remaining 12 degree cases maintain identical behavior until their respective AoA reversals, which occur at different times due to the parameter definition. In general, very similar behavior for all cases is realized in the angular velocity plot with respect to the system angle plot, due to the quantities being related. Equation 4 shows the relation as the velocity being the derivative of the system angle.

As with the system angle from Figure 32, the AoA case demonstrates the lowest angular velocity. This behavior was speculated previously and its confirmation gives merit to the behavioral conclusions listed above. Additionally, this supports the inertial effects minimizing the reversal angle overshoot because a lower rotational velocity would conclude a lower vertical velocity experienced by the airfoil. The beta case is another example of the similar velocity behavior with respect to the system angle. The increased rotational travel allows the system to accelerate to a slightly higher angular velocity than the base case. The same trends are displayed by the dA case as the beta case, although slightly less pronounced.

Besides diverging first from the other 12 degree cases, the spring case again exhibited interesting behavior. Instead of having smooth transitions at the velocity extremes like the other cases, the spring case had abrupt changes. These sharp peaks occur at the same time as the AoA reversal. The lack of any spring dampening means that the only torque available to drive the system comes from the airfoils. When the airfoils reverse AoA, the force provided by the airfoils abruptly changes and is the only contribution to the torque. This abrupt change in force is exaggerated by the effective angle of attack, which will be discussed later in this section. This rapid change is the reason for the shallow system angle peaks. Instead of gradually slowing the rotational acceleration, the spring case displays a rapid change in acceleration that has the effect of drawing out the maximum system angle.

The system torque plot, shown in Figure 34 below, clearly displays some of the phenomena discussed in the previous plots. Again, the initial transient torque region for the 12 degree cases remains identical until the first AoA reversal point. The phase shift with respect to the base case is easily demonstrated here for all of the cases. All of the cases with the exception of the spring case follow very similar trends throughout the oscillations. The maximum torque comes immediately after the AoA change as discussed earlier for the base case. The fluctuations in the torque after the maximums are likely due to the unsteady flow separation that is occurring due to the high effective AoA. This is supported by the fact that the AoA case does not have the last fluctuation dip as seen in the base, beta, and dA cases, and is a product of the AoA case experiencing lower separation than the other cases.

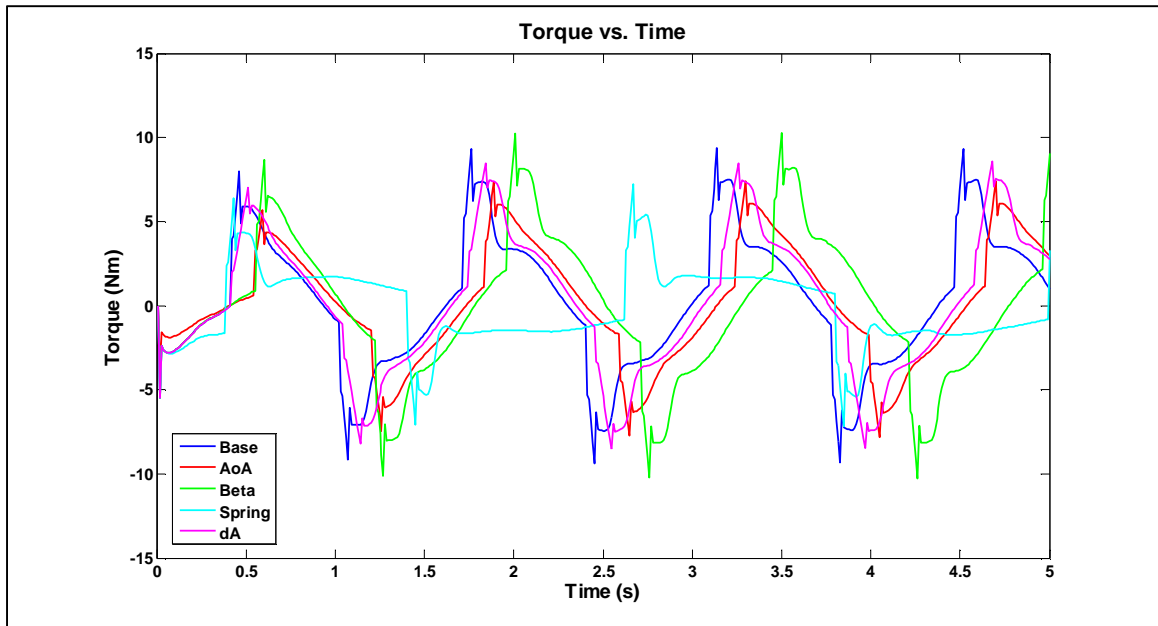


Figure 34- Parametric Torque

After the AoA reversal and the torque fluctuations, the system torque steadily decreases and even reaches negative values before the next airfoil AoA change. This steady decline is due to the spring using its built up torque, energy, to help reverse the rotational motion of the system, and the spring torque value going to zero when system passes through the X axis. The spring torque then begins to build its stored torque back up while resisting the system rotation and the torque supplied by the airfoils. The torque supplied from the airfoils also decreases through the oscillation due to the reduced effective AoA caused by the vertical velocity of the airfoil.

The torque fluctuations caused by the separation are also present for the spring case. What is not present is the steady decline in the torque after the fluctuations. As was suspected above, the declining torque was a product of the spring. This suspicion can clearly be concluded due to the lack of any significant decline when the spring is removed. The spring case does still display a slight decline, which is likely due to reduced airfoil force caused by a declining effective AoA.

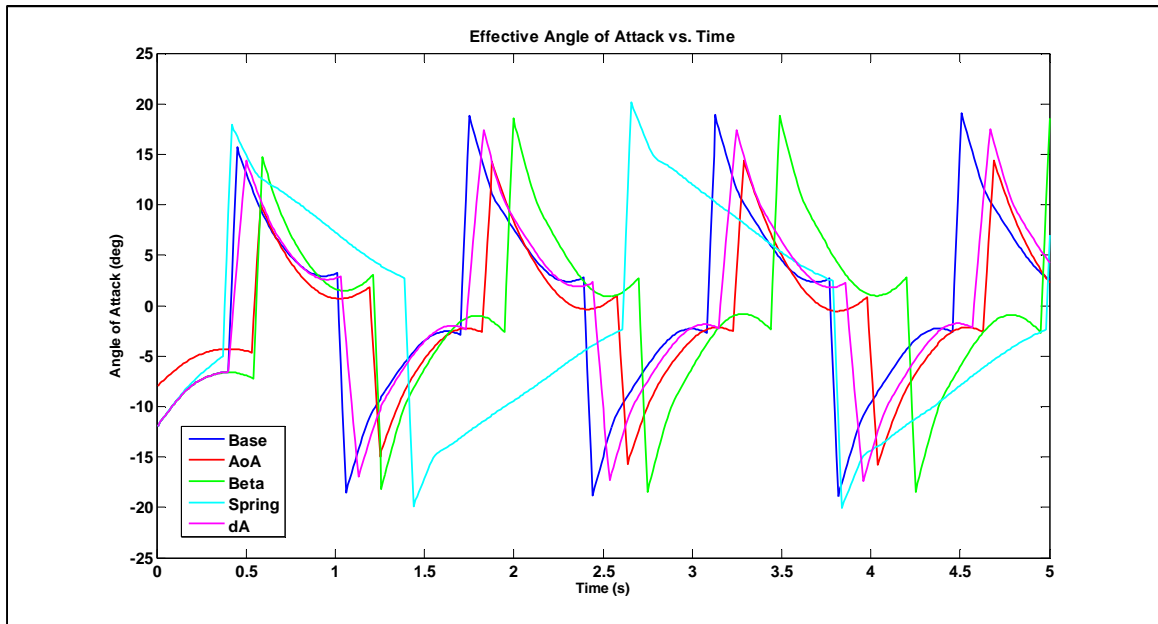


Figure 35- Parametric Effective AoA

Figure 35 above shows the effective AoA for all of the parametric cases. With the exception of the spring case, all of the cases follow the trend discussed for the base case. Initially, the cases being at their static AoA, but the effective AoA rapidly decreases due to the vertical motion induced by the rotation of the system. For the first time, the effect of the dA cases is noticeable through the reduced slope compared to the other cases just before the peaks. This indicates that the dA case took longer to reverse the airfoil AoA, which by the parameter definition is expected. The difference in the maximum peaks of the various 12 degree cases is due to the varying vertical velocity of the airfoils at the time of AoA reversal. Having a higher vertical velocity at the time of the AoA change causes the airfoils to experience a higher AoA because the vertical velocity is opposite the *direction* the airfoil wants to go with the new AoA. This conclusion is supported by examining Figure 33 and noticing the spring case has the highest system angular velocity at the time of airfoil AoA change and that it has the highest effective AoA. This trend remains consistent for all cases throughout the analysis. Naturally, the AoA cases have the lowest effective AoA, which is due to both having a lower absolute AoA as well as having the lowest vertical velocity at the time of AoA change.

The spring case is again the outlier in terms of trend consistency. After the AoA reversal, the spring case displays a more linear decline in effective AoA compared to the other cases. This can be attributed to there being no built up spring torque to help reverse the rotational motion immediately after the AoA reversal. This built up torque allows the other cases to have a higher initial vertical velocity and thus a lower effective AoA. Interestingly, the ending effective AoA before the airfoil AoA change is very similar for all of the cases, even though the system parameters varied widely. This ending value was also near 0, which would be very inefficient for the symmetrical NACA 0015 airfoils. This low effective AoA is part of the reason the torque goes negative before the AoA reversal, because the forces from the airfoils do not produce enough torque to overcome the spring torque.

The plot of the spring torque is as expected for all cases and is shown in Figure 36. The torque supplied by the spring has been defined to have a linear relationship with the system angle, β . The torque should be opposite in direction from the system angle deflection and should have a value of zero Newton meters when the system is parallel to the X axis. All of the cases, with the exception of the spring case due to the absence of a spring, show behavior that is consistent with the definition as identical spring constants were used. The only variations in the cases are from the various maximum system angles achieved, thus a higher spring torque, and the phase shifting from the base case due to the various behaviors of the cases.

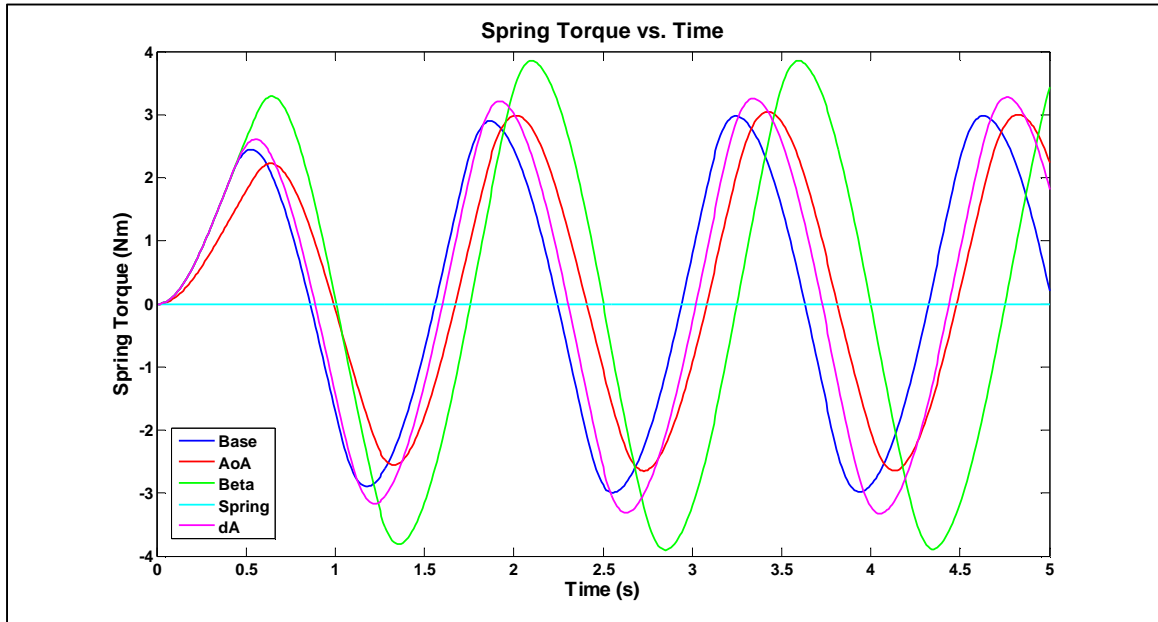


Figure 36- Parametric Spring Torque

In order to show that the mathematical scheme was not introducing significant error into the computation, an error was calculated. This error was calculated in the same manner as in the verification section above, such that the distance between the airfoil AoA pivot points should remain the same throughout the computation if true rotational motion is achieved and the computational distance is compared to the known original distance. Figure 37 below shows the percent error of the cases and follows the same trend predicted by the verification code presented in Figure 21 above. The error grows as the calculation progresses because the Cartesian velocity approximations continue to pull the airfoils away from each other slightly at every time step. Due to this growth, the largest error is displayed by every case at the final time step. The highest error of all the cases was just over 2% by the beta case. This error in the beta case is suspected to be due to having the most extreme movements out of all the cases. Due to the first order linear approximation for the Cartesian velocities, a larger velocity for a given time step would cause a large linear movement. Large linear movements naturally do not approximate rotational motion well, thus giving the beta case the largest error out of all the cases presented. The spring case has the lowest error of the parametric study cases, which is

likely due to the maximum system angular velocities being experienced for the shortest amount of time. Recall from Figure 33 above that the spring case demonstrated sharp peaks instead of gradual changes in velocity like those displayed in all of the other cases. In addition to the normal parametric study cases, an additional base case was run using a smaller time step to demonstrate the ability to minimize error. The time step for the base error case was half of the time step size used in the standard base case with a value of 0.005 seconds. This reduction in time step size naturally increased the number of time steps required to achieve the 5 second total analysis time. Reducing the time step in the computational analysis confirms the results shown in Table 5, such that by reducing the time step by half reduces the accumulated error by approximately half.

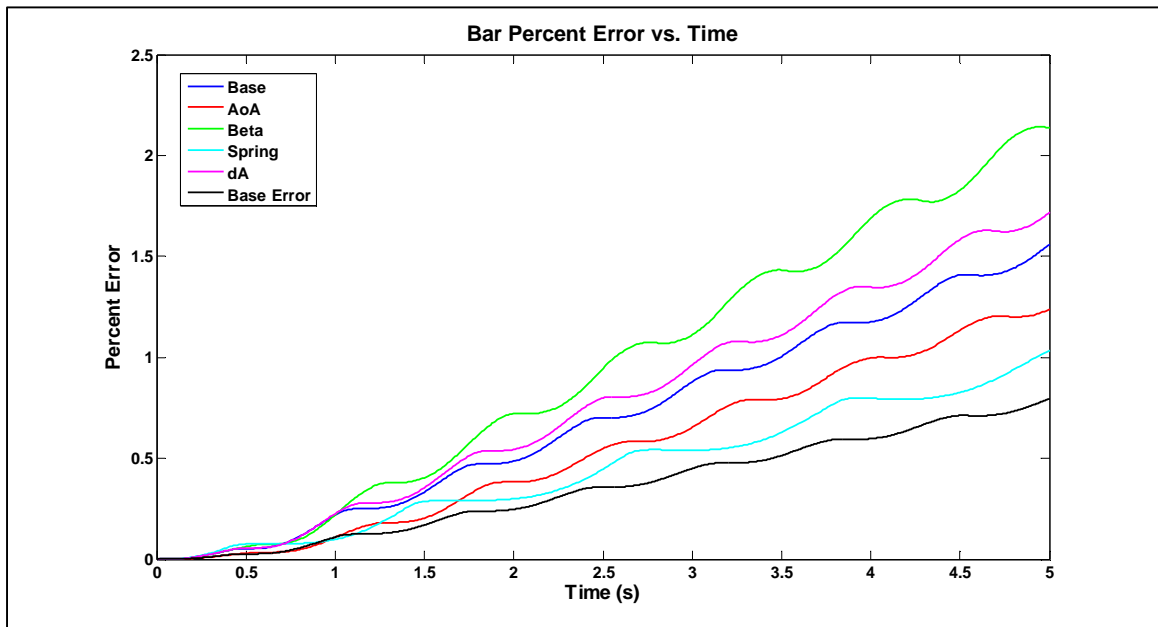


Figure 37- Parametric Error

In general, the error introduced in all of the cases is minimal and could potentially be reduced by reducing the time step size. The time step size used in this parametric study was chosen for computational time convenience as well as eliminating dynamic meshing errors, and the low error presented support the validity of the results obtained.

The aerodynamic characteristics of the cases presented in the parametric study are for the most part very similar to the base case. The AoA and the spring cases were the most varied and the differences will be discussed here. The other cases have very similar flow interactions between the leading and trailing airfoils. The largest difference is the larger vertical displacement seen in the beta and dA cases. This does help slightly minimize the wake interaction with the trailing airfoil, however, the difference is mostly negligible. The similarity does not come as a surprise after examining the above plots. The trends throughout all of the plots show very similar system response with the base, beta, and dA cases.

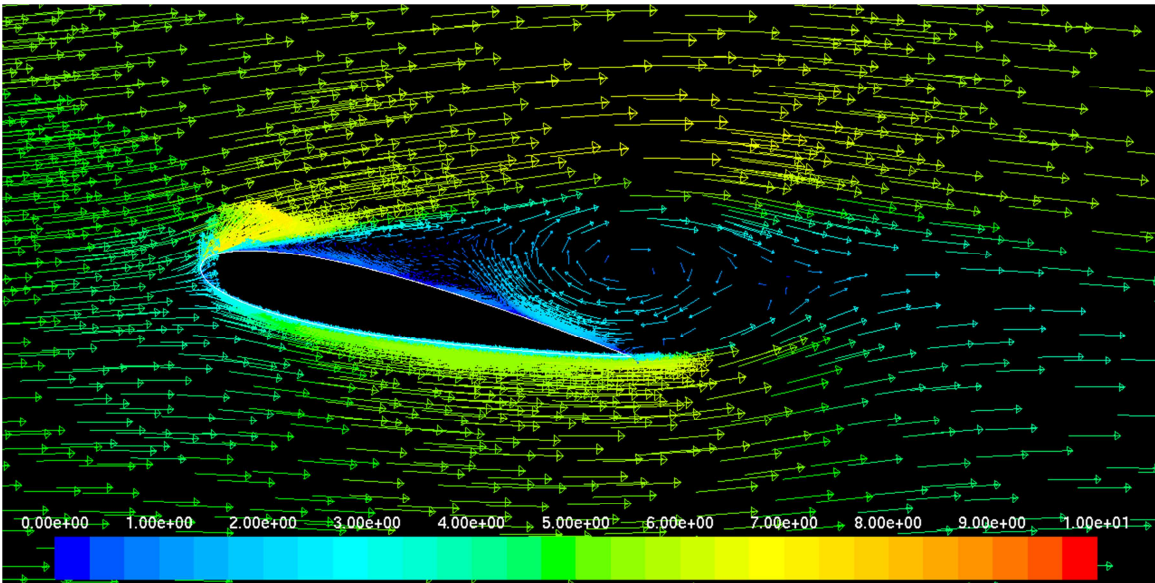


Figure 38- Base Separation

The AoA case is different from the base case in that all of the aerodynamic characteristics are lessened. First, the initial shedding of the high and low velocity wakes during the AoA reversal is much smaller and thus dissipates much quicker. The high velocity wake that travels along the wall is still present however. The separation seen in the AoA case is much less severe than the base case, due to both the lower vertical velocity of the airfoil at the time of the change and the lower airfoil AoA. The base separation is shown in Figure 38 and the AoA case separation is shown in Figure 39.

Care was taken to ensure the separation presented in Figure 38 and Figure 39 was captured at roughly the same time in the separation development, whereby the separation was approximately at its maximum and had not yet begun to shed off the trailing edge. In the base

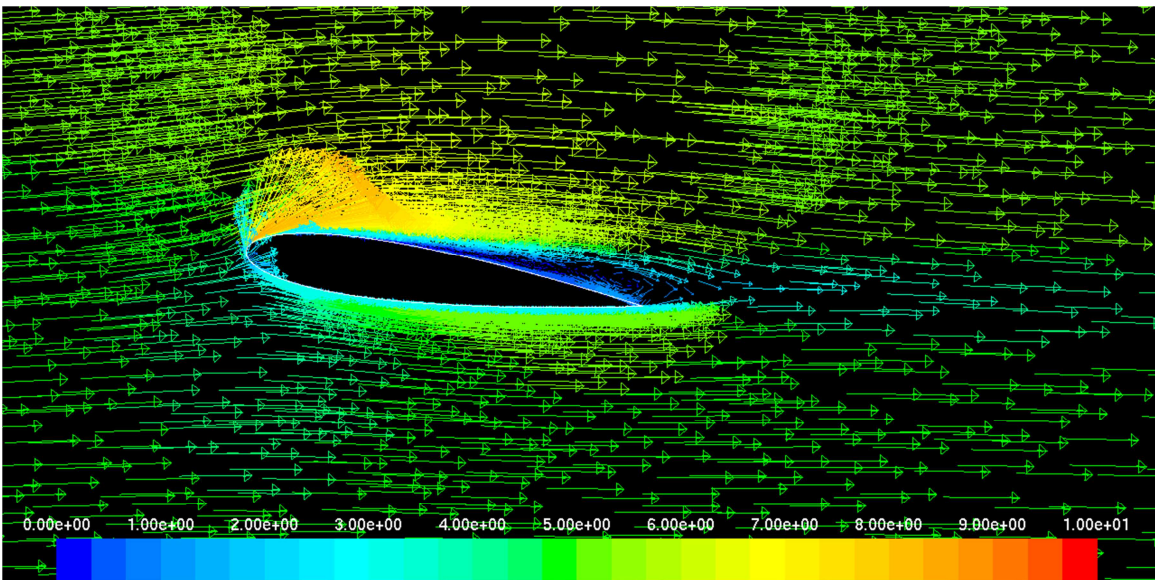


Figure 39- AoA Separation

separation, the computation predicts a very large vortex accompanied by two smaller vortices further towards the leading edge. The computation of the AoA case has predicted only one vortex, which happens to be much smaller and flatter than the base case. This smaller separation causes a much smaller wake to be created which happens to dissipate rather quickly.

Figure 40 shows the AoA case drag forces for the leading and trailing airfoils. Similar to the base case, the trailing airfoil experiences a slightly lower drag peak, although the difference between the two is smaller. Unlike the base case, the drag force is not symmetrical for the oscillation because during the upstroke, both airfoils experience a lower drag. This could be due

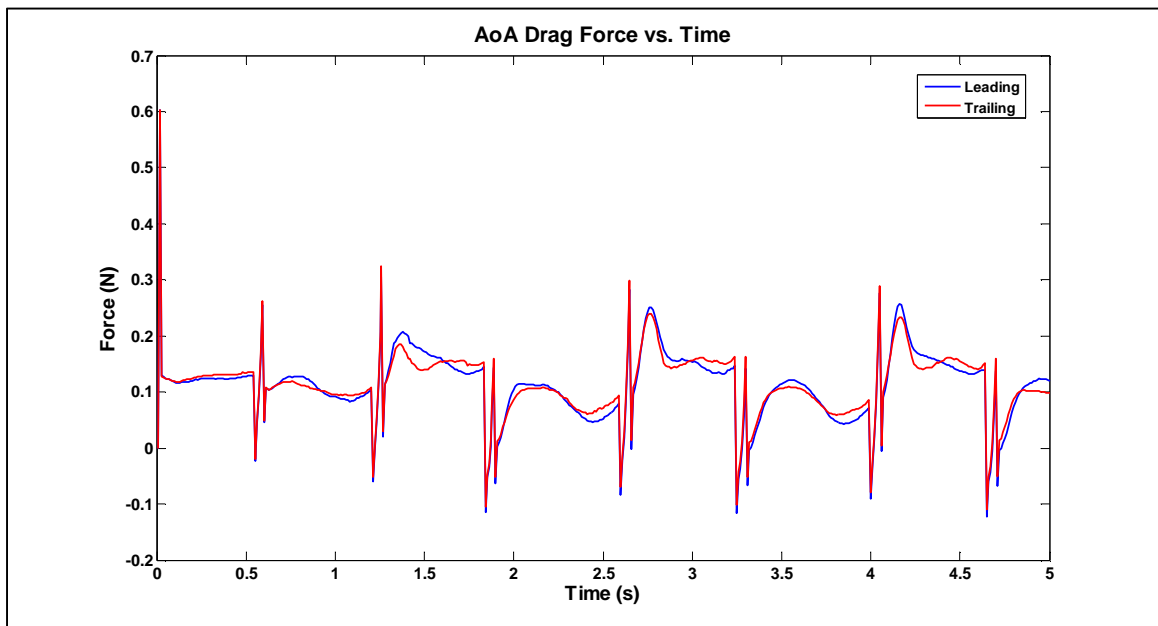


Figure 40- AoA Drag Force

to some wind tunnel wall interaction, or even some flow interaction that is not discernable qualitatively. The variation is quite small and suspected to be negligible due to the effect not being seen in the other results figures above.

The spring case had a very similar initial wake during the AoA reversal, such that both the low and high velocity regions reacted very similar to the base case. This similarity is likely caused by the cases have identical airfoil AoA change parameters, which explains the similarity in the other 12 degree AoA cases as well. Because there is no spring constant that builds up opposite torque to help reverse the system rotation, the airfoils continue to travel in the original rotational direction for longer than the base case. This phenomenon was well documented in the result figures above. This continuation of motion while having an opposite AoA causes a large separation to occur. Figure 41 shows the spring case separation. This separation is computationally predicted with a large vortex and one or two smaller vortices towards the leading edge of the airfoil like the base case. This separation, however, also has another vortex

near the trailing edge. The velocity magnitudes of these larger vortices are larger than the base case, thus helping validate the larger separation conclusion.

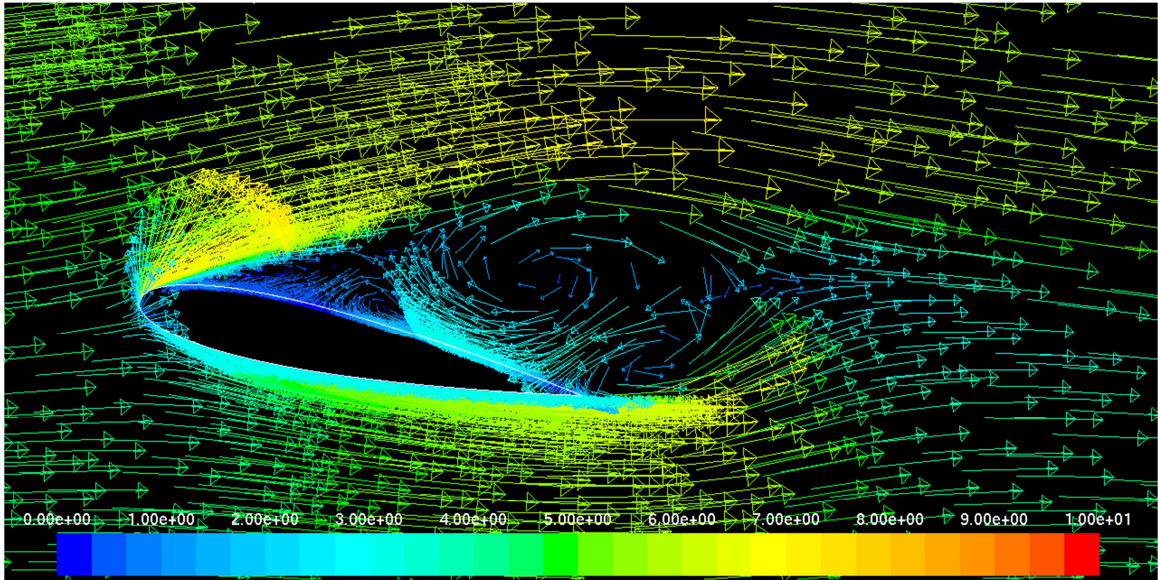


Figure 41- Spring Separation

CHAPTER 5- CONCLUSIONS

In conclusion, a computational solution setup using FLUENT has been developed and validated for NACA 0012 and NACA 0015 airfoils. The validation for both airfoils has shown grievance to published airfoil lift and drag data. Additionally, the pressure coefficient over the surface of the NACA 0012 airfoil has been validated against published pressure data. Similar meshing sizing schemes were used to create a scale wind tunnel based mesh for the wing oscillator prototype. A mathematical scheme was developed from the equations governing rotational motion to drive the wing oscillator motion in the computation. This mathematical scheme used the computed airfoil forces from the computational analysis to drive the motion. In this manner, the analysis was derived to provide a performance prediction that would be able to be compared to experimental data. The mathematical scheme was verified using both a MATLAB code and a C code to ensure the integrity of the calculation.

The mathematical scheme was hooked into FLUENT using UDFs that were written to accommodate parallel computation. The built in dynamic meshing parameters provided by FLUENT were used in conjunction with the UDFs to allow for the dynamic analysis of the wing oscillator. Numerous test cases were run to ensure correct performance of the computational analysis. Result cases were run using parallel computation with up to 16 partitions to reduce the computational time required for each case. Animation files were saved and rendered for each result case, as well as the saving intermittent case and data files for later static reference. With the completion of the aforementioned tasks, all of the project goals have been met.

The base results from the dynamic analysis were analyzed for aerodynamic characteristics. The base case demonstrated significant separation immediately after the airfoil AoA reversed direction near the extremes of travel. This separation caused a large wake to traverse the wind tunnel and interact with the trailing airfoil. Although somewhat dissipated, the leading airfoil wake had a small effect on the drag force experienced by the trailing airfoil. The conclusion of the wake interaction was that the effect was negligible for most of the oscillation period. This was likely due to the rather large distance between the leading and trailing airfoils. The possibility of some numerical diffusion occurring due to poor mesh resolution was considered. It was found that as the solution progressed, the mesh would degrade slightly despite the dynamic meshing parameters being selected to minimize cell growth and dissipation. The separation caused the forces on the airfoil to fluctuate as the flow separated and reattached while shedding the developed vortices. The effective AoA of the airfoils also had an effect on the forces produced by the airfoils. As the airfoils traversed the wind tunnel vertically, the vertical velocity reduced the effective angle of attack of the airfoils. This reduction in AoA reduced the amount of force produced by the airfoils. Aerodynamic characteristics that reduce the airfoil forces or system torque would have a negative effect on the power generation potential of the wing oscillator. Thus, tuning the system parameters to minimize the negative aerodynamic characteristics would be useful, which could include tuning the system to have the leading airfoil wake pass the trailing airfoil without causing an aerodynamic interaction.

A parametric study was conducted to investigate the effect on performance by the various system parameters. Most of the parameters seem to have an effect on the rotational travel of the system. The spring constant seemed to have a significant impact on the performance of the systems. At this point, it would be difficult to conclude whether these effects had a positive or negative effect on the system performance. In one aspect, the lack of a spring keeps the torque positive for the entire oscillation stroke, compared to the other cases that dipped into a negative torque region before the AoA reversal. Another aspect, however is that there is no built up opposite torque after the AoA reversal so the system continues to travel opposite of the airfoil *direction*. This causes a large separation of the flow from the airfoil surface, which causes large fluctuations in the airfoil forces.

The final conclusion of the computational analysis of a wing oscillator is that there is potential for power generation. The system has been shown to be self-driving though the above computational analysis. By modifying the system parameters, it may be feasible to tune the system to meet the needs of various flow conditions. It would be recommended to build a cost efficient prototype, similar to the geometry used in the dynamic case above, and perform some initial experimental analysis.

5.1 – RECOMMENDATIONS AND FUTURE WORK

One of the first recommendations for the current analysis would be to work on the mesh resolution and degradation issue. This recommendation might be limited by the tools available in FLUENT, however, there are some tools that exist that were not fully explored. The first tool would be the new diffusion dynamic meshing scheme found in FLUENT version 14. This method has not been explored due to it not being available in earlier FLUENT versions that were used to develop the dynamic meshing scheme used in this study. Secondly, the dynamic mesh adaptation functions have not been fully explored due to project time constraints. The goal of implementing these features would be to keep the mesh resolution similar to the initial mesh, as well as better capturing critical areas of the flow field like in the vicinity of the separation.

In the future, having experimental data to compare some of the dynamic results to would be imperative. Having experimental data to compare to would hopefully validate the numerical scheme as well as verify the results of the dynamic computation. At minimum, some steady state experimental data should be collected and compared with a steady state analysis using the wind tunnel based mesh. If a fully dynamic experimental setup was to be designed, the specific geometry should be used to create the mesh for the dynamic computation. The specific turbulent boundary conditions of the wind tunnel should also be added to the computational model. With or without a full experimental analysis, the transmission system to convert the oscillatory motion to pure rotational motion should be developed. A mathematical model of this transmission system should then be added to the calculations so that the power and efficiency of the wing oscillator could be predicted.

An active AoA control method should be developed to allow the wing oscillator to have more control over the effective AoA. This would more than likely greatly improve the performance of the wing oscillator by being able to produce a high torque throughout the entire oscillation. This method would also have the potential of being able to reduce the separation, which would reduce the force fluctuations. Implementing this active control would be a great task, as a full control system would have to be developed and subsequently implemented into the UDF code. This type of control would include a number of new tuning parameters, that coupled with a transmission system scheme would allow for the full investigation of maximizing power and efficiency.

Lastly, the UDF code should be modified to allow for even easier manipulation of the system parameters. This means that some of the calculations conducted externally of the UDF should be included such that the specific values of the variables can be calculated internally. This would greatly enhance both the usability and the efficiency of changing the parameters for various cases.

REFERENCES

1. **Liu, Tianshu.** *Wind Oscillator for Power Generation.* 20110064567A1 United States of America, 2009.
2. *Multiobjective Design Study of a Flapping Wing Power Generator.* **Shimizu, Eriko, Isogai, Koji and Obayashi, Shigeru.** 2008, Journal of Fluids Engineering, Vol. 130.
3. *The Wingmill: An Oscillating- Wing Windmill.* **McKinney, William and DeLaurier, James.** 2, 1981, Journal of Energy, Vol. 5.
4. *Parametric Study of an Oscillating Airfoil in a Power- Extraction Regime.* **Kinsey, T. and Dumas, G.** 6, 2008, AIAA Journal, Vol. 46.
5. *Low Reynolds Number Airfoil Performance Subjected to Wake Interference from an Upstream Airfoil.* **Michelsen, W.D. and Mueller, T.J.** 1987, AIAA.
6. **Hepperle, Martin.** Java Foil. [Online] January 27, 2007. <http://www.mh-aerotools.de/airfoils/javafoil.htm>.
7. **Abbot, Ira H. and von Doenhoff, A.E.** *Theory of Wing Sections.* New York : Dover Publications, 1959.
8. **FLUENT, ANSYS.** User's Guide. s.l. : ANSYS, Inc., 2012. Vol. 14.0.
9. —. Theory Guide. s.l. : ANSYS, Inc., 2012. Vol. 14.0.
10. **Tennekes, H: Lumley, John L.** *A First Course in Turbulence.* Cambridge, MA : MIT Press, 1972.
11. Applied Aerodynamics Laboratory. *Western Michigan University.* [Online] 2010. http://www.wmich.edu/mae/research_labs/applied_aerodynamics/advanced_wind_tunnel.php .
12. **Fitzpatrick, Richard.** Rotational Motion. *University of Texas Austin.* [Online] February 2, 2006. <http://farside.ph.utexas.edu/teaching/301/lectures/node97.html>.
13. **FLUENT, ANSYS.** UDF Manual. s.l. : ANSYS, Inc., 2012. Vol. 14.0.
14. *Investigation of Flow Over an Oscillating Airfoil.* **Lee, T. and Gerontakos, P.** 2004, Journal of Fluid Mechanics, Vol. 512, pp. 313-341.
15. *Unsteady Viscous Flow on Oscillating Airfoils .* **McCroskey, W. J. and Philippe, J. J.** 1, s.l. : AIAA Journal, 1975, Vol. 13.
16. **Jacobs, Eastman N. and Sherman, Albert.** *Airfoil Section Characteristics as Affected by Variations of the Reynolds Number .* s.l. : NACA, 1937. Report No. 586.

17. **Henga, Harwood A.** *Numerical Solution of Incompressible Flow Over Airfoils Near Stall.* s.l. : AIAA, 1982.