Simulation of Morphing Blades for Vertical Axis Wind Turbines

Jennifer Tan

A Thesis
In the Department of

Mechanical, Industrial and Aerospace Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Applied Science in Mechanical Engineering at
Concordia University

Monreal, Quebec, Canada

August 2017

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Jennifer Tan**

Entitled: **Simulation of Morphing Blades for Vertical Axis Wind Turbines**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Mechanical Engineering)

complies with the regulation of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

_____ Chair
Dr. Mingyuan Chen

_____ Examiner
Dr. Samuel Li

_____ Examiner
Dr. Brian Vermeire

_____ Supervisor
Dr. Marius Paraschivoiu

Approved by: _____

Martin D. Pugh, Chair
Department of Mechanical and Industrial Engineering

_____ 2017 _____

Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Simulation of Morphing Blades for Vertical Axis Wind Turbines

Jennifer Tan

The simulation of flow through vertical axis wind turbine (VAWT) is characterized by unsteady flow where the blade experiences varying angles of attack and Reynolds number as it completes a cycle. Therefore, the lift generated also varies as a function of its rotational position relative to the incoming freestream velocity. In order to improve the performance of these turbines the blade can take advantage of smart materials developed for control surface actuation. The aim of this paper is to investigate the effect of morphing blades on the aerodynamic performance of the turbine blades. The study uses commercial software Ansys Fluent pressure-based solver to investigate the flow past the turbine blades by solving the 2D Unsteady Reynolds-Averaged Navier-Stokes (URANS) equations. In order to simulate the morphing blade for VAWT, a sliding mesh method is used to simulate the VAWT rotation while a user-defined function (UDF) is written for the blade morphing flexure motion. This entails the use of dynamic mesh smoothing to prevent the mesh from having negative cell volumes. Although the dynamic mesh strategy has been successful in preserving the cell quality, it has been shown that the proposed method of simulating the morphing blade on VAWT is inadequate due to unphysical solutions. Finally, the effect of morphing the blade is tested on a static airfoil case instead, where it is shown that stall is alleviated by morphing the blade trailing edge.

Acknowledgements

I would like to thank my supervisor Dr. Marius Paraschivoiu for giving me the opportunity to work on this thesis project, and for providing invaluable guidance and feedback. I could not have done as much without the support of my supervisor. I would like to extend my gratitude to Matin Komeili and Gabriel Naccache for their suggestions and counsel especially during the initial phases of working on the thesis topic. I would also like to thank my colleagues Spencer Foley, Dan McLean, Patrick Larin, Farbod Vakilimoghaddam, and Aierken Dilimulati for all the insightful discussions and discourses. I would also like to acknowledge Dr. Daniel Inman and Dr. Alexander Pankonien of University of Michigan for providing the Synergistic Smart Morphing Aileron blade profiles used for the study. I want to thank the exam committee members Dr. Brian Vermeire and Dr. Samuel Li. Finally, I would like to dedicate my thesis to my parents Rolando and Sylvia Tan, who have been patient with me all their life and supported me through thick and thin. My thanks also to my brothers Alexander, Randolph, Lawrence, and Reginald for all the love and support.

All the computations for the study were made possible through the supercomputer Briarée from l'Université de Montréal, managed by Calcul Québec and Compute Canada. The operation of this supercomputer is funded by the Canada Foundation for Innovation (CFI), the ministère de l'Économie, de la science et de l'innovation du Québec (MESI) and the Fonds de recherche du Québec - Nature et technologies (FRQ-NT).

Contents

1 Introduction **1**

1.1 Morphing Blade for Vertical Axis Wind Turbines 2

1.1.1 $\delta = +0.48°$ Blade Profile (almost symmetric blade) 3

1.1.2 $\delta = -13.37°$ Blade Profile (cambered blade) 4

1.1.3 $\delta = +6.98°$ Blade Profile (inverted-camber blade) 4

1.2 Problem Statment and Objective 4

1.3 Outline of the Study 5

2 Theory and Literature Review **6**

2.1 Airfoil Static Stall 6

2.2 Dynamic Stall in VAWT 7

2.3 Wake and Blade-Vortex Interaction 10

2.4 Effect of Blade Camber and Pitch 11

2.5 Deforming Mesh Methods 11

2.5.1 Linear Spring Analogy 12

2.5.2 Linear Elasticity 12

2.5.3 Laplace Diffusion 13

2.5.4 Transfinite Interpolation 13

2.5.5 Radial Basis Functions 13

2.5.6 Summary of Deforming Mesh Methods 14

3 Models and Methodology **15**

3.1 Governing Equations 15

3.2 Turbulence Model 17

3.3 Numerical Setup 18

3.3.1 Spatial Discretization 18

3.3.2 Pressure at Cell Faces 18

3.3.3 Pressure-Velocity Coupling 18

3.3.4 Temporal Discretization 19
3.4 Calculation of Forces 19
3.5 Grid Generation, Initial and Boundary Conditions 21
3.6 Simulation of Moving Grids 24
3.6.1 Governing Equations for Dynamic Mesh 24
3.6.2 Sliding Mesh and Non-conformal Cells 26
3.6.3 Dynamic Mesh Methods 27
3.6.4 User-Defined Function 28

4 Verification and Validation **31**
4.1 Grid and Time Step Sensitivity Analysis 31
4.2 Grid Resolution at Boundary Layer 33
4.3 Validation with Experiment 34
4.4 Choice of Temporal Discretization 36
4.5 Mesh Deformation Quality 38

5 Results and Discussion **40**
5.1 Airfoil Static Stall Angle Investigation 40
5.2 VAWT Fixed Profile Results 44
5.3 Morphing Trailing Edge at Static Stall Angle 49
5.4 Challenges of Morphing Case for VAWT 52
5.4.1 Implementing Morphing Blade for VAWT 52
5.4.2 Alternative Strategies for Morphing Blade on VAWT 57
5.4.3 Recommendations Outside of Fluent 60

6 Conclusion **62**

A Fourier Approximation of Airfoil **71**
A.1 Fourier approximation for the airfoil surfaces 71
A.2 21 Frames of SSMA actuation range 72

B User-Defined Function **80**

List of Figures

1.1 Synergistic Smart Morphing Aileron (SSMA) [52] 2
1.2 The trailing edge of the three blade profiles considered for the study 3

2.1 Deflection angle and shift in C_L curve [3] 7
2.2 Range of α in one cycle for various λ plotted against the azimuthal position 8
2.3 Dynamic C_L (left) and C_D (right) plotted against α for $2 < \lambda < 5$ [76] 9
2.4 Computed vortex trajectory downstream [2] 10

3.1 Domain geometry and boundary conditions 22
3.2 (a) Rotating zone, (b) sliding mesh, (c) mesh resolution on trailing edge . . . 22
3.3 Boundary conditions for the static airfoil cases 23
3.4 Sliding mesh and non-conformal interfaces [27] 26
3.5 Blade surface split into upper, lower, and tip for specifying coordinates . . . 28
3.6 Algorithm diagram for specifying node movement for blade flexure motion . 30

4.1 Grid sensitivity with C_P as objective variable 32
4.2 Temporal sensitivity with C_P as objective variable 33
4.3 Turbulent viscosity ratio near the wall 34
4.4 Validation of experimental case for $\lambda = 2$ with non-dimensional tangential force 35
4.5 Validation of experimental case for $\lambda = 2$ with non-dimensional normal force 35
4.6 C_P comparison for $\delta = +0.48°$ 37
4.7 Turbulent viscosity ratio (a) 2nd order (b) 1st order temporal for $\delta = +0.48°$ 37
4.8 (a) Initial mesh, (b) $\delta = +6.98°$, and (c)$\delta = -13.37°$ after defromation . . . 38
4.9 Orthogonality of the grids before (left) and after deformation (middle, right) 39
4.10 Skewness of the grids before (left) and after deformation (middle, right) . . . 39

5.1 Static airfoil $\delta = +0.48°$ convergence history, $\alpha = 10°$ (left), $\alpha = 20°$ (right) . 41
5.2 C_L and C_D vs angle of attack for the static airfoils 41
5.3 Pressure contour [Pa] for the static airfoils 43
5.4 (a) C_L, (b) C_D for the three blade profiles for $\lambda = 3.17$ 44

5.5 Pressure contours [Pa] for $\theta = 120°$ (left column) and $\theta = 300°$ (right column) 46
5.6 Vorticity contours for $\theta = 120°$ (left column) and $\theta = 300°$ (right column) . . 47
5.7 Coefficient of power C_P at $\lambda = 3.17$ for each fixed blade profiles 48
5.8 Coefficient of power C_P at $\lambda = 3.17$ of baseleine profile and envelope 48
5.9 $\delta = +0.48°$ morphed to $\delta = -13.37°$ for $\alpha = 18°$ (left), morphing phase (right) 50
5.10 Comparison of C_L, C_D values between $\delta = -13.37°$ morphed and $\delta = -13.37°$ 50
5.11 Static pressure [Pa] while morphing the blade from $\delta = +0.48°$ to $\delta = -13.37°$ 51
5.12 Streamlines on the profile before and after morphing 51
5.13 (a) C_L vs effective aoa_{eff} (b) aoa_{eff} vs θ from fixed profile results 53
5.14 C_P comparison for baseline profile, envelope, and morphing profile 54
5.15 Turbulent viscosity ratio for the morphing case on VAWT 54
5.16 Convergence history four cycles after morphing to $\delta = +6.98°$ 55
5.17 C_P comparison between morphed once and fixed profile of $\delta = +6.98°$ 55
5.18 Vorticity magnitude contours for (a) morphing case, (b) baseline fixed profile 56
5.19 Declaring C-mesh fluid zone as "deforming" in Fluent dynamic mesh zones . 57
5.20 Diagram of the alternate method I for VAWT morphing case 58
5.21 Diagram of the alternate method II for VAWT morphing case 59

A.1 Upper Fourier coefficients for the 21 frames of actuation range 71
A.2 Lower Fourier coefficients for the 21 frames of actuation range 72

List of Tables

1.1 Summary of the three blade profile characteristics 3

2.1 Comparison of common mesh deformation methods 14

3.1 Geometry and inlet conditions 23

4.1 Summary of grid and time step sensitivity analysis 32
4.2 Average C_P per cycle for 1st order and 2nd order temporal discretization . . 36

5.1 Average C_P comparison among blade profiles 45
5.2 Comparison of C_L and $C_D$values for morphed and static case 50
5.3 Blade profile morphed at specific azimuthal postions 52

Nomenclature

α	angle of attack
α_{eff}	effective angle of attack
$\Delta\theta$	azimuthal step size
Δt	time step size
δ	blade trailing edge deflection
ϵ	rate of dissipation of turbulent kinetic energy
λ	tip speed ratio; ratio of blade speed over free stream velocity
μ	molecular viscosity
μ_t	eddy viscosity
ν	kinematic viscosity
ω	turbulence frequency
∂V	bounding surface of control volume
ρ	density
τ_{ij}	Reynolds stress
θ	azimuthal location; $\theta = 0^\circ$ is the location perpendiclar to freestrem velocity
$\vec{u}$	velocity vector
A	area
C_D	coefficient of drag

C_L	coefficient of lift
C_M	coefficient of moment
C_P	coefficient of power
DES	Detached Eddy Simulation
F_N	normal force
F_T	tangential force
F_x	component of force in x-coordinate
F_y	component of force in y-coordinate
$FR1$	refers to the blade profile with $\delta = +6.98°$
$FR14$	refers to the blade profile with $\delta = +0.48°$
$FR21$	refers to the blade profile with $\delta = -13.37°$
k	turbulent kinetic energy
LES	Large Eddy Simulation
P	pressure
$RANS$	Reynolds-Averaged Navier-Stokes equations
Re	Reynolds number
S_{ij}	strain rate
T	torque
$URANS$	Unsteady Reynolds-Averaged Navier-Stokes equations
V	control volume
V_∞	freestream velocity
V_{rel}	relative velocity
x/c	ratio of x-coordinate to blade chord length
y/c	ratio of y-coordinate to blade chord thickness

Chapter 1

Introduction

Wind energy has been shown to be a viable means of alternative energy [67]. To extract energy from the wind, the kinetic energy from the wind velocity is converted by wind turbines into mechanical energy available for generators to convert to electricity. There are two types of wind turbines, the horizontal axis wind turbine (HAWT) and the vertical axis wind turbine (VAWT). Standard designs of VAWT uses symmetric airfoils which have the advantage of being independent on wind direction but could have issues with starting up. The power coefficient of HAWT is typically 16% higher than the VAWT [23] but is not practical to install in urban cities. While most wind energy production has been from HAWTs, there has been growing research interest in VAWT [74, 75]. There are two general forms of VAWT, the Darrieus or lift-based and the Savonius or drag-based. The Savonius turbine was invented earlier in 1922 while Darrieus was invented in 1931. Savonius turbine is better suited for tidal turbines but for wind turbines, the lift-based Darrieus typically produces better power generation potential [23].

The flow around a vertical axis wind turbine is complex and inherently unsteady due to the blade angle of attack α changing as a function of its azimuthal position θ. The lift generated therefore also varies as a function of the azimuthal position. Because the beneficial angle of attack is predominantly in the upwind half cycle of VAWTs, studies have been done to control the pitch angle or by using ailerons and flaps to modify the effective angle of attack of the blades in order to increase the power coefficient per cycle. However the control mechanism involved usually entail penalties in weight and structural complexity of the system. In order to counter these problems, smart materials were developed for control surface actuation. Some of the common smart materials used for controls purposes include shape memory alloys (SMA), piezoelectric stacks, and piezoelectric polymers. The problem with SMA is that although it has high actuation strain, its response time is slow; on the other

hand, piezoelectric materials have low actuation strain but high response time. Pankonien et al. [52] developed a Synergistic Smart Morphing Aileron (SSMA) that combines a SMA-driven hinge with a piezoelectric-driven flexure box that allows the morphing of blade profile with good actuation strain and response time. By introducing morphing mechanism on the blade, the idea is to modify the blade camber as a function of the blade position in a cycle in order to increase the range of favorable power production within a cycle. The study aims to investigate the effect of introducing variable camber to the VAWT by means of morphing the blade and simulating this unsteady phenomenon with computational fluid dynamics.

1.1 Morphing Blade for Vertical Axis Wind Turbines

The Synergistic Smart Morphing Aileron (SSMA) by Pankonien et al.[52] provides the morphing aileron mechanism considered for this study. The SSMA was originally developed for unmanned aerial vehicles and is intended to provide better aerodynamic performance for wide range of aircraft flight conditions. The SSMA reflex actuation is shown to be capable of mitigating flow separation near stall [52]. This ability to alleviate flow separation is beneficial for VAWT since this is one of the main limiting factors to the power generation of VAWT as it experiences higher range of angles of attack at low tip speed ratios λ.

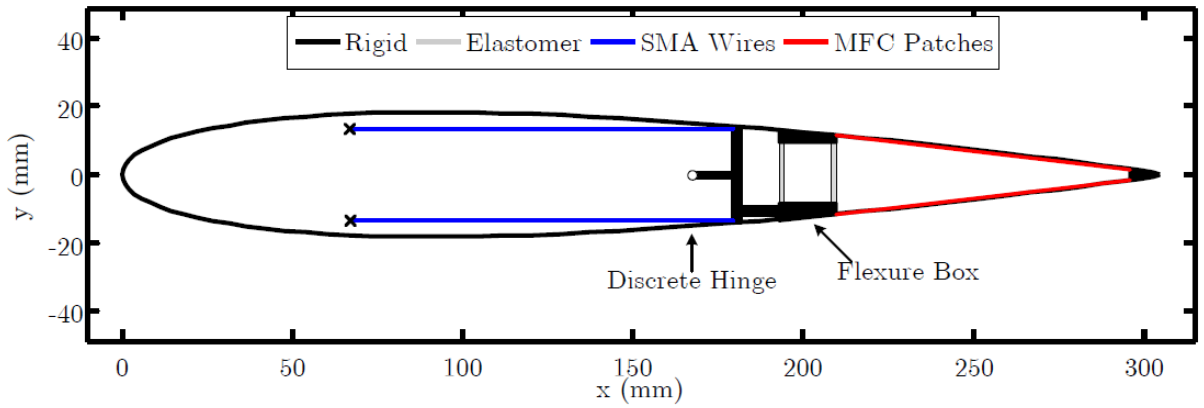


Figure 1.1: Synergistic Smart Morphing Aileron (SSMA) [52]

Figure 1.1 is a diagram of the SSMA mechanism which combines both shape memory alloy (SMA) and piezoelectric actuators (PZT) into a single morphing entity. While shape memory alloys have high actuation strain and blocked stress, it has a low frequency response. On the other hand, PZT has low actuation strain but high frequency response. SSMA therefore implements a SMA-driven hinge to provide the rotation that could resist the aerodynamic

loading while using Macro Fiber Composites (MFC) on the flexure box to provide the smooth conformal trailing edge profile at a faster response [52] .

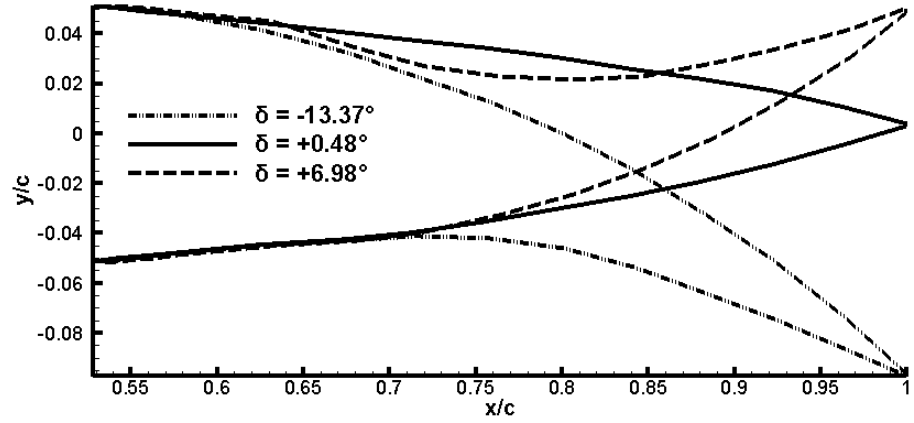


Figure 1.2: The trailing edge of the three blade profiles considered for the study

Table 1.1: Summary of the three blade profile characteristics

blade profile	chord length [m]	chord angle	angle of deflection δ
$\delta = +6.98°$	0.301064	$+2.82°$	$+6.98°$
$\delta = +0.48°$	0.304562	$+0.20°$	$+0.48°$
$\delta = -13.37°$	0.304683	$-5.52°$	$-13.37°$

The blade profile uses a standard NACA0012 leading edge while the data points for the morphed profiles are obtained from the prototype. The morphing control surface begins at 180 mm chord or x/c of 0.59. The two extremes with the highest deflection, $\delta = -13.37°$ and $\delta = +6.98°$, along with the profile that is closest to being a symmetric profile $\delta = +0.48°$ as shown in Fig.1.2 are considered for the study. Table 1.1 presents the blade profile characteristics. Although only three blade profiles are used for the study, there are 19 actuation profiles in between the two extreme actuation cases. These 19 frames of profiles are used during the process of morphing to ensure smooth transition. Including those used for the study, there are 21 blade profile frames in total.

1.1.1 $\delta = +0.48°$ Blade Profile (almost symmetric blade)

This blade profile is the one that most resembles the NACA0012 blade profile. It can be seen on Fig.1.2 that it is neither a completely symmetric nor straight blade and there is a slight deflection of $\delta = +0.48°$.

1.1.2 $\delta = -13.37°$ Blade Profile (cambered blade)

The negative deflection at the trailing edge has the effect of introducing positive incidence $+i$ on the chord line. The cambered blade profile allows the chord line to have a positive pitch angle relative to the zero incidence angle. Cambered airfoil is generally used to generate more lift. However, for vertical axis wind turbines, on the upwind half of the cycle, the positive pitch angle reduces the angle between the relative velocity and the chord line; on the downwind half because the angle of attack becomes negative (i.e. the relative velocity comes at a negative angle relative to the line tangent to the rotor rotation), the effective angle of attack between the relative velocity and the chord line is increased.

1.1.3 $\delta = +6.98°$ Blade Profile (inverted-camber blade)

This blade profile with positive trailing edge deflection has an inverted camber. This changes the chord line to have a negative incidence $-i$ with respect to the zero incidence angle. Inverted camber airfoil is generally used to decrease the C_L generated; however, for the case of vertical axis wind turbines, on the upwind half of a cycle the negative pitch angle from the inverse camber causes the effective angle of attack to increase as the angle between the relative velocity and the chord line is increased. On the other hand, on the downwind half, the effective angle of attack is reduced because the local angle of attack becomes negative on the downwind half cycle.

1.2 Problem Statment and Objective

The purpose of the study is to investigate the aerodynamic impact of introducing morphing blade on VAWT power generation. With this in mind, the study aims to investigate the methodologies of simulating a flow past VAWT with morphing blade within the context of using a commercial solver and addresses the capabilities and limitations. This includes the following objectives for the study:

1. application of the proper methodology for simulating morphing motion on VAWT,
2. investigation of the aerodynamic behavior of the three blade profiles at static condition and their performance for VAWT power generation,
3. simulation of the morphing blade for static case and VAWT case.

1.3 Outline of the Study

The study begins with Chapter 2 where the aerodynamic theory relevant to VAWT is reviewed and different mesh deformation methods are compared. Chapter 3 explains in detail the governing equations and simulation methodology used for the study. The first half of the methodology chapter introduces the Unsteady Reynolds-Averaged equations, the turbulence model used for the study, the discretization schemes used, and the grid generation as well as the boundary and initial conditions. The second half of the the methodology chapter describes how the morphing blade on VAWT is modelled; the sliding mesh method and the dynamic mesh method used is presented, as well as the user-defined function needed to specify the boundary motion of the morphing blade. Chapter 4 addresses the verification and validation of the methodologies; this chapter includes grid and temporal sensitivity analysis, as well as comparison of different turbulence models commonly used in VAWT literature with experimental data. Chapter 5 presents the results for both static airfoil and VAWT cases. The morphing case is applied to both the static and VAWT cases; the challenges and issues of implementing the morphing case on VAWT within the commercial solver is also addressed. Finally, Chapter 6 concludes the study by highlighting the contributions of the study and suggestion for future works.

Chapter 2

Theory and Literature Review

The first half of the chapter reviews the aerodynamic theory relevant to the performance of lift-based VAWT with special focus on dynamic stall and blade-vortex interaction. The effect of blade camber and pitch angle on VAWT performance is also explored. The last half of the chapter is dedicated to mesh deformation methods where each of their advantages and disadvantages are evaluated and compared.

2.1 Airfoil Static Stall

For a given free stream Reynolds number and Mach number, the lift and drag is only a function of the blade angle of attack [3].For symmetric airfoils, it produces no lift at zero angles of attack. Lift increases proportionally to the angle of attack until the blade exceeds the critical angle of attack. Higher than the critical angle of attack, the blade starts to stall where the lift starts to decrease. When separation occurs due to the high angle of attack, drag increases as well. At high angles of attack, it has been shown that RANS models are not as good at predicting leading edge separation compared to DES, LES or LES-RANS hybrid methods [18, 10]. High lift control surfaces like flaps and ailerons shift the C_L curve of an airfoil by introducing deflection angles on the blade chord. Fig.2.1 shows that by having a positive deflection, the maximum C_L is increased at the expense of having a lower stall angle. The opposite is true for negative deflection.

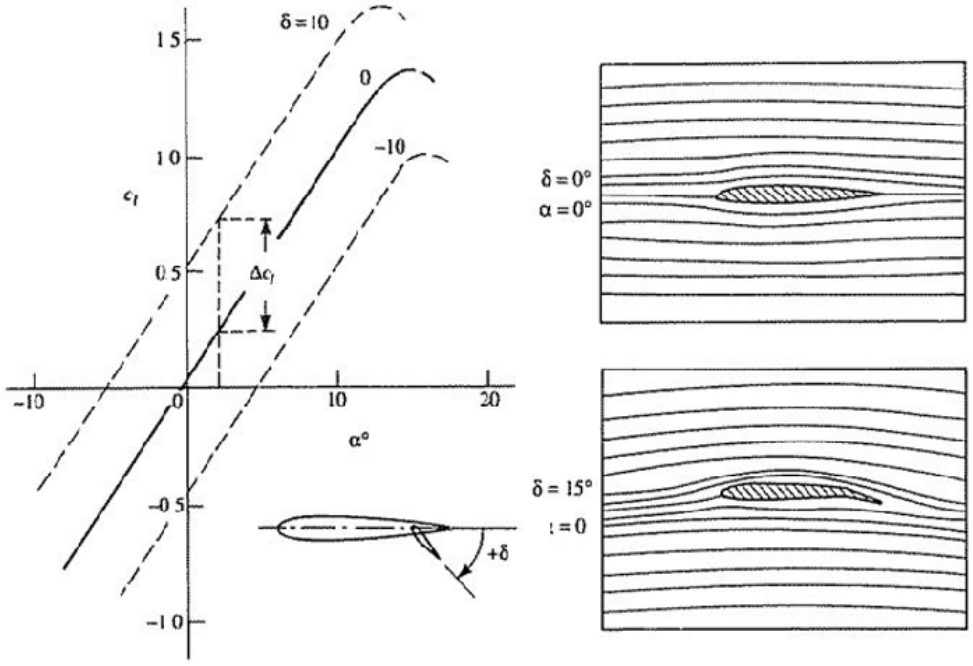


Figure 2.1: Deflection angle and shift in C_L curve [3]

2.2 Dynamic Stall in VAWT

The blade motion on a vertical axis wind turbine is similar to the case of an oscillating airfoil. This is because the blade experiences variations in angles of attack as the azimuthal angle changes. The ranges of angle of attack α experienced by the blade for each revolution is dependent on the value of the tip speed ratio λ and its azimuthal position θ.

For large tip speed ratios, the variation in α has a sinusoidal curve with a limited range of α. As the tip speed ratio decreases to $\lambda = 1$, the range of possible α increases and the peak value of α as experienced by the blade in one revolution also increases as can be seen in Fig.2.2.

Not only does the α change as a function of θ, but the local Reynolds number also varies for a given cycle due to variation in relative wind speeds. In the worst case where $\lambda = 1$, the local Reynolds number is 0 at the azimuthal position of $\theta = 0°$ [64]. When the blade angle of attack is higher than the static stall angle, the blade is likely to exhibit a dynamic stall behavior [46]. There is still contention on the ability of current CFD, especially RANS models, to simulate flow with high angle of attack [41]. Ferreira et al. [26] compared the influence of different turbulence models on the prediction of dynamic stall

with experimental data and found that DES is the closest to experiment and that URANS models are insufficient.

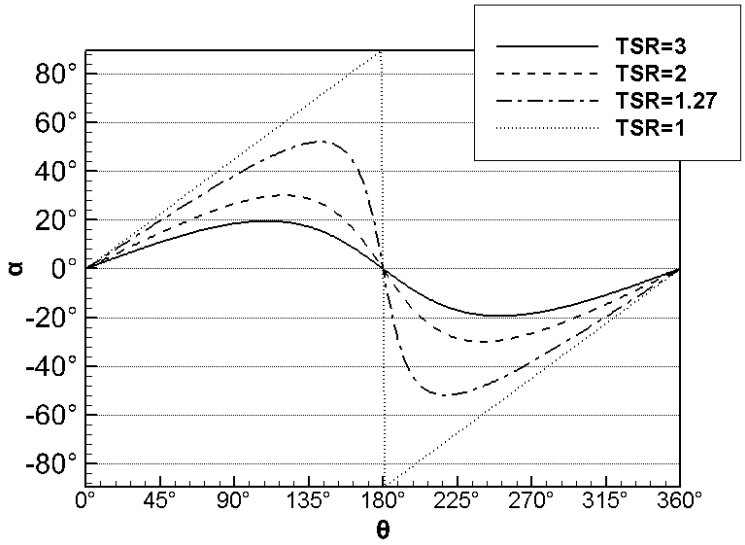


Figure 2.2: Range of α in one cycle for various λ plotted against the azimuthal position

In an experimental study conducted by Lanveville and Vittecoq [76] for a vertical axis wind turbine with two blades, it was shown that for $Re = 3.8x10^4$, dynamic stall behavior occurs for $\lambda < 4$. The dynamic C_L and C_D curves for different range of λ are shown in Fig.2.3. The positive α angles correspond to the upstream azimuthal positions while the negative angles corresponds to the downstream positions. The solid curves are the measured data while the dashed-curves represent the corrected coefficients that account for the wake-induced angle.

For azimuthal position in the range of $0° \leq \theta \leq 90°$, the blade motion is similar to an airfoil with oscillating pitch; it is at this region where the blade angle of attack exceeds the stall angle and therefore exhibits dynamic stall behaviors. For $\lambda \leq 3$ it can be seen that there is a sharp peak in C_L followed by a sudden decrease in C_L; this behavior represents the occurrence of deep dynamic stall. For$\lambda = 4$ and $\lambda = 5$, there is no apparent occurrence of dynamic stall since the maximum blade angle of attack does not exceed the stall angle.

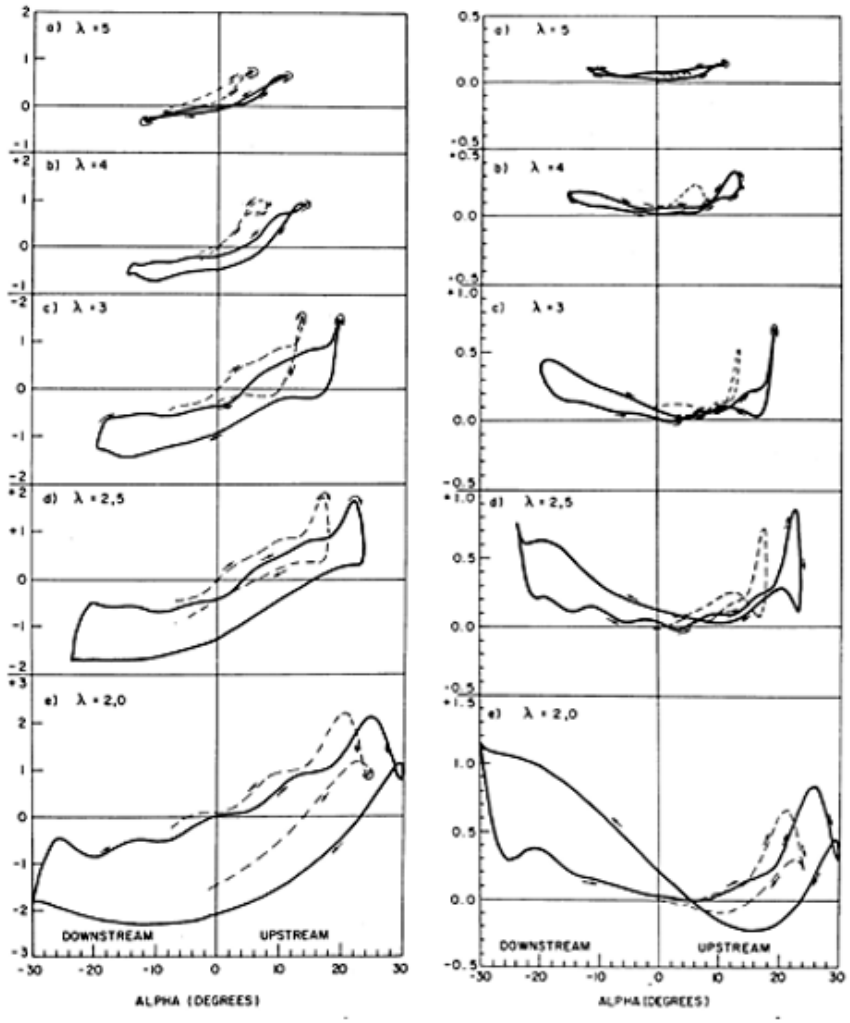


Figure 2.3: Dynamic C_L (left) and C_D (right) plotted against α for $2 < \lambda < 5$ [76]

For azimuthal position in the range of $90° \leq \theta \leq 180°$, the relative velocity of the blade decreases and the blade angle of attack starts to decrease from the maximum value. For this range of azimuthal position, dynamic stall occurs for $\lambda \leq 4$. Since the maximum blade angle of attack increases as the λ is decreased, dynamic stall behavior are more prevalent at low λ. However, Scheurich [64] indicated that dynamic stall may also occur at higher λ if the local blade angle of attack is increased due to the interaction between the blades and the vortices generated by the turbine.

2.3 Wake and Blade-Vortex Interaction

Dynamic stall occurs in unsteady flow conditions when the airfoil goes beyond the static stall angle [46]. During dynamic stall, flow separation bubble occurs in the leading edge which temporarily increases lift until vortices from the separation bubble are convected downstream to the trailing edge, causing a sharp decline in the lift. A clockwise vortex is formed from the leading edge adjacent to the counterclockwise vortex on the suction surface. The roll-up of the leading edge vorticity is shed in discontinuous manner along the wake, however the vortices from the suction side rolls-up at the trailing edge and the vortex gets dragged along the wake [11]. Dynamic stall causes two opposite-rotating vortices to be shed; the first vortex is formed from the leading edge bubble that rolls-up and rotates in the same direction as the rotor while another vortex rotates in the opposite direction and is shed at the trailing edge due to the roll-up from the blade surface aft of the leading edge; it is the counter-clockwise vorticity that is shed from the trailing edge roll-up that gets shed downstream and causes blade-vortex interaction [11].

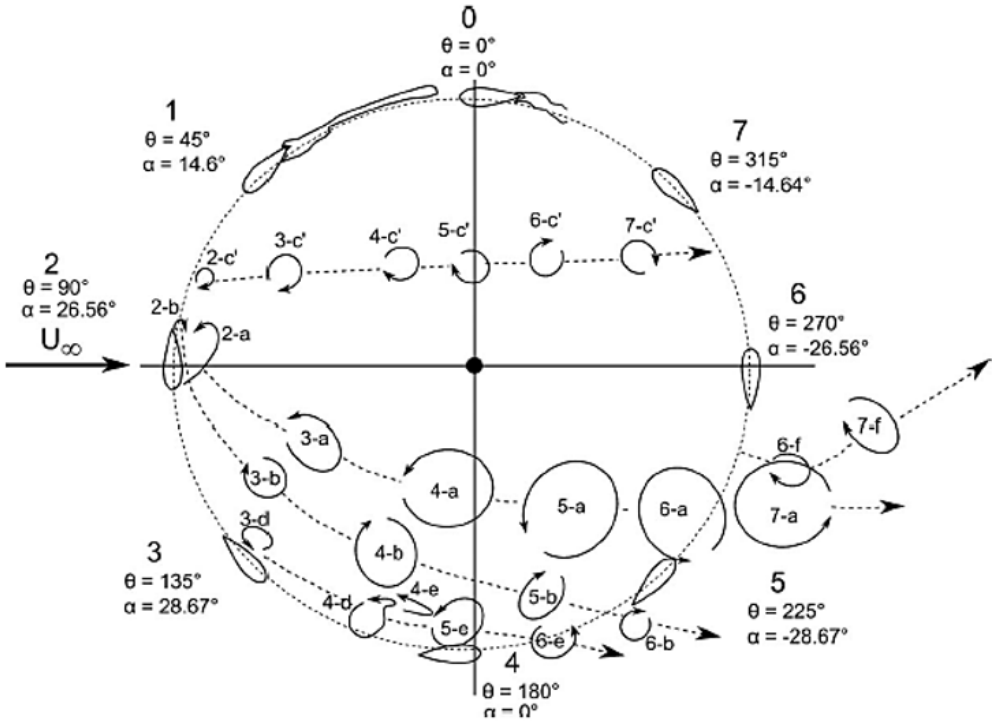


Figure 2.4: Computed vortex trajectory downstream [2]

Amet et al. [2] studied the 2D blade-vortex interaction for a straight two-bladed turbine using $k-\omega$ turbulence model and showed on Fig.2.4 the computed trajectories of vortices using Q criterion for the case of $\lambda = 2$. According to the computed trajectory, there are

three regions of particular importance: $\theta = 0^\circ$ to 45°, $\theta = 90^\circ$ to 270° and $\theta = 315^\circ$, where the blade interacts with its own vortices. The induced velocity on the blade was calculated using Kelvin vortex model and the results showed that there is a 21% induced velocity from vortices while the other vortices have weaker effects [2] . It is therefore important to take into account the effects of blade-vortex interaction especially in the lower half region of the azimuthal blade positions for low λ. Amet et al. [2] also performed the same study for $\lambda = 7$ and found that there is only weak shedding of vortices in the upstream half of the turbine while on the downstream half, flow is attached.

2.4 Effect of Blade Camber and Pitch

Walters et al. [77] compared the power performance of VAWT to HAWT and suggested that VAWT performance could still be improved by optimizing blade angle, blade profile, and turbine solidity. By introducing flaps to VAWT, the C_L/C_D ratio and turbine blade lift curve can be improved, however the drag increases much higher at flap deflections larger than 20 degrees, and thus C_L/C_D ratio starts to decrease beyond this deflection angle [33]. This means that flap angles below this value can help in self-starting capabilities of the VAWT while higher flap deflection could aid in VAWT breaking mechanism. Paillard et al. [51] showed that for a Darreius tidal turbine, there is an increase in C_P from 0.28 to 0.43 by using optimal variable pitch. Paraschivoiu et al. [53] noticed a 30% increase in annual energy by using polynomial optimal pitch control. Cambered airfoils on VAWT have been shown to have the ability to self-start at the expense of having lower peak efficiency compared to straight bladed airfoils [6]. Chen and Kuo [15] have also shown that the larger the camber of the blade, the better it is at self-starting. Rezaeiha et al. [59] found that having pitch angle of -2° at $\lambda = 4$ increases the C_P by 6.6%. Wolff et al. [80] performed 2D RANS on morphing turbine airfoil with a deformable grid and found that there is a phase shift between the deflection and the lift. They also noticed that while defecting the trailing edge at angles of attack near stall, there is an overshoot above the steady state lift coefficient.

2.5 Deforming Mesh Methods

Mesh deformation and automated remeshing have been utilized for shape optimization and rapid prototype design modifications without having to manually create a new mesh. In the case of unsteady simulation of flow past moving bodies, remeshing can be computationally costly so it is preferred to have a robust mesh deformation method that preserves cell quality as much as possible. There is currently no other work in the literature on morphing blade for

the purpose of investigating its unsteady effect on VAWT; however, there has been successful implementation of mesh deformations on grids with high aspect ratio in the region of moving boundary [82, 42, 63] meant for viscous flow. There are inherent complications for simulating morphing blades for viscous flow due to very large aspect ratio typically used on the boundary layer; thus, the method of smoothing out the morphing motion becomes critical to ensure the non-negative cell volumes when deforming the mesh. The following section reviews the most common mesh deformation methods in the literature.

2.5.1 Linear Spring Analogy

The spring analogy for moving the mesh was first proposed by Batina [5] where Hooke's law is used to model the node displacement and static equilibrium is obtained when the force at each node is zero. The parameter that affects the node displacement is the stiffness coefficient used in the Hooke's law equation and the Drichlet boundary conditions are the known boundary displacements [30, 49]. The spring analogy is one of the simplest to implement but is not robust [62]. Farhat [25] augmented the method by introducing torsional springs between adjacent edges to prevent cells from intersecting during rotational motion. Other modifications to the linear spring analogy include semi-torsional spring [7], ortho-semi-torsional spring approach [43], and ball-vertex method [9]. Among these methods, it was shown that the orth-semi-torsional approach is the most robust [43].

2.5.2 Linear Elasticity

For this method, the node displacements are calculated by solving for the linear elasticity equations where the mesh modulus of elasticity E and mesh Poisson's ratio ν are the parameters that are used to control the node displacements. The idea behind this method is to find the optimized E and ν to allow node displacement without invalidating the cells. These methods have been successfull in implementing a dynamic mesh that preserves the boundary layer for visous flow calculations [82, 83]. Karman [35] set ν as constant and allowed E to be equal to the cell aspect ratio in order to increase the stiffness in the boundary layer where cells have high aspect ratio. Mavriplis [82] used a method where E is a function of the distance to the boundary or inversely proportional to cell volume. Hsu [32] used a two-step approach wherein the equations are solved with $E = 1$ in the first step, the mesh strain energy density from this first step is then used to compute the E for the next step. In another study, Yang and Mavriplis [83] calculated the optimal E distribution using adjoint-based optimization and has shown that this method is able to avoid negative cell volumes even for highly stretched mixed element grids which are predominanly used for viscous flow

calculations.

2.5.3 Laplace Diffusion

The Laplace Diffusion method diffuses the node movements by solving for the Laplace equation. The diffusion coefficient γ is the parameter that controls the node displacement. Crumpton and Giles [17] suggested a smoothing equation based on the Laplace equation with the coefficient of thermal conductivity inversely proportional to cell volume while Löhner [40] used the γ as a function of the distacne from the moving boundary. The disadvantage of solving a linear Laplace equation is that the mesh deformation components are solved independently; therefore, if the boundary motion moves only in the x-coordinate, the interior nodes will move only along the x-coordinate [32]. A variation of the method is proposed where the diffusion coefficient is raised to an exponent. This modified Laplace Diffusion coefficient has been shown to improve the capability of handling larger deformations [12].

2.5.4 Transfinite Interpolation

For structured grids, the most common method is based on transfinite interpolation (TFI) with blending functions [72]. In the TFI method, the node displacement is equal to the moving boundary multiplied by a scaling factor which depends on the distance of the nodes to the boundary [13, 21, 81]. The main disadvantage of TFI is that it does not account for cells intersecting and overlapping without augmenting it with additional smoothing operators, and is predominantly limited to structured grids with small deformations.

2.5.5 Radial Basis Functions

Instead of solving for physical-based equations like Hooke's Law or Laplace's equation, the radial basis function (RBF) method transfers the boundary displacement to the interior nodes through an interpolation function. The RBF interpolation method is known to give high quality grids that preserve the cell orthogonality close to the deforming boundary [20] which makes it ideal for grids with high aspect ratio close to the wall. Bos [8] performed a mesh motion that combines translation and rotation with a 2D block, where the rotation was done for 57.3° and 180°. The quality of mesh deformation is compared using Laplace, solid body rotation (SBR) stress equation (a variant of linear elasticity equation), and RBF interpolation method and the study showed that Laplace performed the worst with 0.09 average skewness and 20.1 average non-orthogonality while RBF has average skewness of 0.051 (-41%) and average non-orthogonality of 19.0(-6%); the SBR stress gave the same result as the Laplace

within 6% of difference. The deformed mesh using RBF interpolation was able to rotate by 180° without invalidating the mesh. The disadvantage of RBF is the computational cost; however, there has been numerous studies that have suggested augmentations to improve its computational efficiency. Most notable is the inclusion of greedy algorithm by Rendall and Allen [57, 56] and a gradient-based algorithm by Jakobsson and Amoignon [34].

2.5.6 Summary of Deforming Mesh Methods

The main concern when choosing a mesh deformation method is the computational cost and robustness of preserving mesh quality. Selim and Koomullil [66] summarized in Table 2.1 some of the more common mesh deformation methods wherein n_e is the number of edges, n_v is the number of vertices, n_b is the number of boundary nodes, and n_s is the number of selected boundary nodes.

Table 2.1: Comparison of common mesh deformation methods

method	advantages	disadvantages	complexity
Linear spring	simple to implement	intersecting and overlapping elements	$O(n_e^3)$
Torsional spring	robust mesh quality preservation	computationally expensive	$O(n_e^3 + n_v^3)$
Linear elasticity	computationally feasible	need to optimize E and ν to avoid cell invalidation	$O(n_e log n_e)$
Laplace	computationally efficient	works for single frequency deformations only	$O(n_v)$
TFI	simple and efficient	intersecting and overlapping elements	$O(n_v)$
RBF	robust mesh quality preservation	computationally expensive	$O(n_b^3)$

Both the linear elasticity and RBF interpolation methods have been shown to be most robust [82, 8, 66]. Other less robust methods can be augmented to increase mesh quality preservation at the cost of higher computational cost. Samareh [63] showed that the spring analogy method can preserve the mesh quality in the viscous boundary layer for both unstructured and structured grids by adding quaternions. Maruyama [44] used Laplace smoothing with quaternions that allows for the preservation of mesh orthogonality for 2D and 3D mesh undergoing large deformations; however, the quaternion method with Laplacian equation is at least one order of magnitude more CPU intensive than the RBF-based methods. For 3D grids or mesh of large sizes, the linear spring, torsional spring, linear elasticity, and Laplace methods become computationally expensive as their complexity scales with the number of element vertices or edges. The RBF method, when combined with the greedy algorithm on the other hand, the order of complexity is reduced greatly with $O(n_s^3)$ where $n_s < 5\% \, n_b$. The mesh deformations that rely on interpolation have the advantage of not requiring node connectivity information and therefore have less memory requirements.

Chapter 3

Models and Methodology

This chapter covers the governing equations and the methodologies used to solve these equations. The governing equations are the 2D incompressible Unsteady Reynolds Averaged Navier-Stokes (URANS) and the turbulence model $k-\omega$ SST which are discretized and solved using finite volume approach and are all done through the commercial solver ANSYS Fluent. This section also addresses the numerical setup used for the study by considering best practices from literature. The resolution of the grid and time step used for the study is discussed as well. The chapter then looks into the methods of moving the mesh, smoothing the mesh motion, as well as the algorithm written for the user-defined function to specify the blade profile coordinates.

3.1 Governing Equations

The governing equation for incompressible unsteady viscous flow is the 2D Navier-Stokes equation which is composed of the continuity equation and momentum equations.

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \tag{3.1}$$

$$\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)=-\frac{\partial P}{\partial x}+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)+\rho g_x \tag{3.2}$$

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)=-\frac{\partial P}{\partial y}+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right)+\rho g_y \tag{3.3}$$

Using the general variable of interest ϕ, the transport equation can be written as

$$\rho\frac{\partial \phi}{\partial t}+\rho\nabla\cdot(\phi\vec{u})=\nabla\cdot(\Gamma\nabla\phi)+S_\phi \tag{3.4}$$

where the first term is the rate of change, second term is the convective term; on the right hand of the equation is the diffusion term and the source term respectively. Integrating the transport equation above gives the integral form appropriate for the finite control volume method.

$$\int_V \rho \frac{\partial \phi}{\partial t} dV + \int_V \rho \nabla \cdot (\phi \vec{u}) dV = \int_V \nabla \cdot (\Gamma \nabla \phi) dV + \int_V S_\phi dV \tag{3.5}$$

Using the Gauss Divergence Theorem, the convection and diffusion terms can be rewritten in the form where the integrals are taken over the control surface instead of the whole volume

$$\frac{\partial}{\partial t} \int_V \rho \phi dV + \int_{\partial V} \rho \phi \vec{u} d\vec{A} = \int_{\partial V} (\Gamma \nabla \phi) d\vec{A} + \int_V S_\phi dV \tag{3.6}$$

Where $(\rho \phi \vec{u})$ is the convective flux and $(\Gamma \nabla \phi)$ is the diffusive flux. The equation above shows that the fluid property within the control volume is conserved.

For turbulent flows, the velocity is characterized by chaotic fluctuations in time. The Reynolds decomposition [58] decomposes the instantaneous Navier-Stokes equations into time-averaged and fluctuating terms. Using the tensor notation where velocity components are expressed in $i = 1, 2, 3$, the velocity and scalar quantities can be decomponed into mean and fluctuating components.

$$u_i = \overline{u_i} + u_i^{'} \tag{3.7}$$

$$\overline{u_i} = \frac{1}{\Delta t} \int_t^{t+\Delta t} u_i dt \tag{3.8}$$

Substituting Eq.3.7 and the equivalent form for scalar terms into the instantaneous Navier-Stokes equations and taking the time-average yields the following set of equations called the Reynolds-averaged Navier-Stokes (RANS) equations.

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{3.9}$$

$$\rho \frac{\partial u_i}{\partial t} + \rho \frac{\partial (u_i u_j)}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j} \left(2\mu S_{ij} - \rho \overline{u_i^{'} u_j^{'}} \right) \tag{3.10}$$

where μ is the molecular viscosity and S_{ij} is the strain rate which is defined as

$$S_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \tag{3.11}$$

The extra term $-\rho\overline{u_i' u_j'}$ from the Reynolds decomposition is referred to as Reynolds stress, and additional equations are required to close the system of equations. The manner in which this closure is achieved depends on the turbulence model used. For this study, the $k-\omega$ SST turbulence model is used for the closure.

3.2 Turbulence Model

The Reynolds stress τ_{ij} is computed using the Boussinesq approximation [79] where the upper case is used to denote time-averaged quantities.

$$\tau_{ij} = -\rho\overline{u_i' u_j'} = 2\mu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_i}{\partial x_i} \right) - \frac{2}{3}\rho k \delta_{ij} \tag{3.12}$$

where μ_t = eddy viscosity
k = turbulent kinetic energy = $\frac{1}{2}(\overline{u'^2} + \overline{v'^2} + \overline{w'^2})$
δ_{ij} = Kronecker delta ($\delta_{ij} = 1$ if $i = j$, and $\delta_{ij} = 0$ if $i \neq j$)

The eddy viscosity μ_t is a function of the velocity scale and the length scale, and the turbulence models differ in the way that the velocity scale and length scale are computed. This study uses the $k-\omega$ SST as the turbulence model, which was suggested by Menter [47] due to the inability of $k-\epsilon$ to account for adverse pressure gradients in the boundary layer [78]. The $k-\omega$ SST uses a hybrid model wherein $k-\epsilon$ turbulence model is used in the freestream and transitions into the $k-\omega$ model at near wall [48]. The $k-\epsilon$ model [39] uses the turbulent kinetic energy k, and the rate of dissipation of turbulent kinetic energy ϵ for the eddy viscosity, while the $k-\omega$ model uses the turbulence frequency $\omega = \epsilon/k$ instead. The k-equation for the $k-\omega$ model [79] are

$$\rho\frac{\partial k}{\partial t} + \rho\nabla\cdot(k\vec{u}) = \nabla\cdot\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\nabla(k)\right] + \eta\left(2\mu S_{ij}\cdot S_{ij} - \frac{2}{3}\rho k\frac{\partial U_i}{\partial x_j}\delta_{ij}\right) - \beta^*\rho k\omega \tag{3.13}$$

Near the wall, the ϵ-equation is switched to a ω equation by substituting $\epsilon = k\omega$

$$\begin{aligned}\rho\frac{\partial \omega}{\partial t} + \rho\nabla\cdot(\omega\vec{u}) = \nabla\cdot&\left[\left(\mu + \frac{\mu_t}{\sigma_{\omega,1}}\right)\nabla(\omega)\right] \\ &+ \eta\left(2\rho S_{ij}\cdot S_{ij} - \frac{2}{3}\rho\omega\frac{\partial U_i}{\partial x_j}\delta_{ij}\right) - \beta_2\rho\omega^2 + 2\frac{\rho}{\sigma_{\omega,2}\omega}\frac{\partial k}{\partial x_k}\frac{\partial \omega}{\partial x_k}\end{aligned} \tag{3.14}$$

The last term on the right hand side is the added source term that is due to the $\epsilon = k\omega$ used on the diffusion term in the ϵ-equation; where σ_k, $\sigma_{\omega,1}$, $\sigma_{\omega,2}$, η, β^*, β_2 are the model constant coefficients. Finally, to account for the instabilities that might arise from using the $k-\epsilon$ in the free stream and using $k-\omega$ near the wall, a blending function is used to have a smooth transition between the two models.

3.3 Numerical Setup

The commercial solver ANSYS Fluent 14.5.7 is used to solve the governing equations. Fluent uses finite volume method with a cell-centered co-located scheme where the velocity and pressure values are both stored in the cell center. The following sections show how the governing equations are discretized and solved.

3.3.1 Spatial Discretization

The diffusion terms are always discretized using central differencing scheme which are 2nd order accurate. The convection term is discretized using 2nd order upwind. The 2nd order upwind is chosen because although the 1st order upwind is most stable, it is known to give rise to numerical diffusion or false diffusion [55], while higher order schemes like QUICK and Third-Order MUSCL scheme are more unstable and could give oscillations in solutions. Therefore a good balance between stability and accuracy is achieved by using 2nd order upwind. The gradients are calculated using the Least Squares Gradient reconstruction.

3.3.2 Pressure at Cell Faces

The pressure values are needed at cell faces; however, since Fluent uses cell-centered co-located scheme, pressure values are known only at the cell center. To address this, PRESTO (Pressure Staggering Option) scheme considers a staggered control volume to compute the staggered pressure. By shifting the faces to the adjacent cell centers, pressure at the faces can be obtained.

3.3.3 Pressure-Velocity Coupling

By default, FLUENT's pressure-based solver is a segregated solver where velocity and pressure have to be solved separately and iterated until the continuity is satisfied. The most common method of pressure-velocity coupling is the SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) which was developed by Pantakar [54], and its other variants

like SIMPLER, SIMPLEC, and PISO. While the SIMPLE algorithm has the advantage of requiring less memory, it has the disadvantage of having a slower rate of convergence as velocity and pressure are solved sequentially. This is why the COUPLED method is used for the study. Compared to SIMPLE and PISO, the COUPLED algorithm is shown to have a convergence rate that is independent of grid size [4], whereas especially in the case of SIMPLE algorithm, the convergence rate is highly dependent on the grid size used. Moreover, SIMPLE algorithm may sometimes require the use of an under-relaxation factor for the pressure correction term to stabilize the solution. The disadvantage of the COUPLED algorithm is that it requires 3 to 4 times more memory than SIMPLE [27]. Finally, the coupled AMG (Algebraic Multi-Grid) with F-cycle is used to solve for the coupled system of equations for the velocity and pressure values.

3.3.4 Temporal Discretization

Temporal discretization in Fluent's pressure-based solver can be explicit or implicit, 1st order or 2nd order. Based on linear stability analysis, explicit schemes are limited to the CFL criteria [16] while implicit schemes are not. Both the implicit backward Euler 1st order and 2nd order methods are considered in the validation and verification section of the study. The reason both are considered was because the 2nd order implicit was initially the preferred method as it allows for larger time step size compared to a 1st order time integration. However, non-physical solutions where large pressure gradients exist in mesh interfaces were observed when using this 2nd order scheme in conjunction with dynamic mesh. The chapter on validation and verification shows the difference in the solutions when using 1st order and 2nd order implicit backward Euler. All the values for the results chapter uses 1st order implicit.

3.4 Calculation of Forces

Fluent can monitor the lift, drag, and moment coefficients during a transient simulation. However, the drag and lift are monitored in Fluent with fixed force vectors. Lift coefficient by default is monitored with force vector $\langle 0, 1\rangle$ while drag coefficient has the force vector$\langle 1, 0\rangle$. However, this formulation is appropriate only for translational motion were the force vectors remain constant. Because the blade is rotating, the appropriate force vectors changes in time according to the azimuthal location of the blade. By using the constant $\langle 0, 1\rangle$ and$\langle 1, 0\rangle$ force vectors, Fluent is effectively monitoring the coefficient of forces in the x and y direction instead of the lift and the drag coefficients. In order to calculated the lift and drag coefficients,

the following transformations are performed on the monitored F_x and F_y coefficients.

$$F_N = F_y \cos\theta - F_x \sin\theta \tag{3.15}$$

$$F_T = F_y \sin\theta + F_y \cos\theta \tag{3.16}$$

The angle of attack is positive in the upstream half of a cycle but becomes negative in the downstream half where $\theta > 180°$, this effect on the lift and drag equation is taken into account by the sine function. After calculating for the normal and axial components, these forces are then transformed to components or the force normal (lift) and parallel (drag) to the local relative velocity.

$$L = F_N \cos(\alpha + i) - F_T \sin(\alpha + i) \tag{3.17}$$

$$D = F_N \sin(\alpha + i) + F_T \cos(\alpha + i) \tag{3.18}$$

The angle of attack α is the angle between the relative velocity V_{rel} and the blade chord. The local angle of attack is a function of both the azimuthal position and the λ. For symmetric airfoils, the angle of attack is calculated as

$$\alpha = \arctan\left(\frac{\sin\theta}{\cos\theta + \lambda}\right) \tag{3.19}$$

The incidence angle i takes into account the induced change in angle of attack due to the blade camber. The lift coefficient and drag coefficient are the calculated as

$$C_L = \frac{L}{0.5\rho_\infty V_{rel}^2 c} \tag{3.20}$$

$$C_D = \frac{D}{0.5\rho_\infty V_{rel}^2 c} \tag{3.21}$$

Where ρ_∞ is the freestream density and c is the blade chord length. There is a decrease in the free stream velocity downwind after the blade upstream hits the incoming flow; however, for simplicity, the calculation of the of relative velocity assumes that free stream velocity in the downwind half cycle is the same as the free stream velocity in the upwind cycle. The normal force is important for stress and aerodynamic calculations. For the purpose of the study, only the tangential force is relevant. The tangential force is responsible for producing the torque generated by the vertical axis wind turbine. The coefficient of torque or moment

coefficient is calculated from the tangential force as

$$C_M = \frac{F_T * r}{0.5\rho V_\infty^2 Ar} = \frac{T}{0.5\rho V_\infty^2 Ar} \tag{3.22}$$

Since $P = T\omega$, substituting torque T from the C_M into power coefficient C_P equation gives the relationship between the C_P and the tip speed ratio $\lambda = r\omega/V$ as

$$C_P = \frac{P}{0.5\rho V_\infty^3 A} = \frac{(C_M * 0.5\rho V_\infty^2 Ar)(\omega)}{0.5\rho V_\infty^3 A} = C_M \frac{r\omega}{V_\infty} = C_M \lambda \tag{3.23}$$

The C_P is used to measure the amount of power generated by the VAWT; it is the main variable to consider when implementing the morphing blade, and C_P is also used as the objective variable for convergence.

3.5 Grid Generation, Initial and Boundary Conditions

The computational domain is shown in Fig.3.1 and is made up of three zones – the outer domain, the rotating zone, and the C-mesh zone. The outer domain is a stationary zone for the far field flow. The boundaries on the left, on the top, and bottom sides have a distance of 20 times the diameter Di of the rotating zone, while the right boundary has a distance of 40 times the rotating zone diameter to prevent the wake from affecting the boundary conditions. Wake interference downstream is minimal at approximately 5 times the diameter of the turbine [4]. While the rotating zone has a diameter of 6m, the rotor diameter is 5.395m or rotor radius of 2.6975m. The initial blade profile used for the grid generation is $\delta = +0.48°$, the mesh for the two other profiles are not generated manually and was deformed using to generate their respective meshes. The x-coordinate distance from the leading edge to the trailing edge for all blade profiles varies from 0.3007-0.30561m. The quarter chord location is approximated by 0.25 of 0.303m. The leading edge of the blade profile is at the origin $(0, 0)$ while the axis of rotation is located at $(0.07575, -2.6975)$. With this, the quarter chord length of the blade is vertically aligned with the axis of rotation. Spalart and Rumsey [69] suggests $\nu_t/\nu \approx 2\text{x}10^{-7}Re$ for the inflow condition for most external flows. Figure 3.2 shows the sliding mesh interface and the fully structured C-mesh zone. The nodes that will be flagged for deformation are all localized in the C-mesh zone as this saves computational cost; instead of having to smooth out the node positions of the whole domain, only the nodes inside the C-mesh zone are deformed. The first cell height from the blade surface is $y = 1.99\text{x}10^{-5}$m or $6.56\text{x}10^{-5}$ the chord length. The cells on the boundary layer

have large aspect ratio but in the region towards the tip of the trailing edge, the nodes are distributed in such a way that aspect raio is kept close to 1. Having cells with high aspect ratio at the tip is not recommended with the diffusion-based method as it causes negative cell volumes on the tip of the blade when deforming the mesh. The computational domain has a total of 384,770 cells, or 241,356 nodes, with 680 nodes on the blade. The local Reynolds number experience by the blade varies as a function of azimuthal position as the relative velocity changes throughout the cycle. For the inlet free stream velocity of 8 m/s, blade speed of 90rpm and blade chord length of 0.303m, the range of Reynolds number for the blade is between $Re = 3.5\text{x}10^5$ to $Re = 6.8\text{x}10^5$; thus having an average of $Re = 5.2\text{x}10^5$.

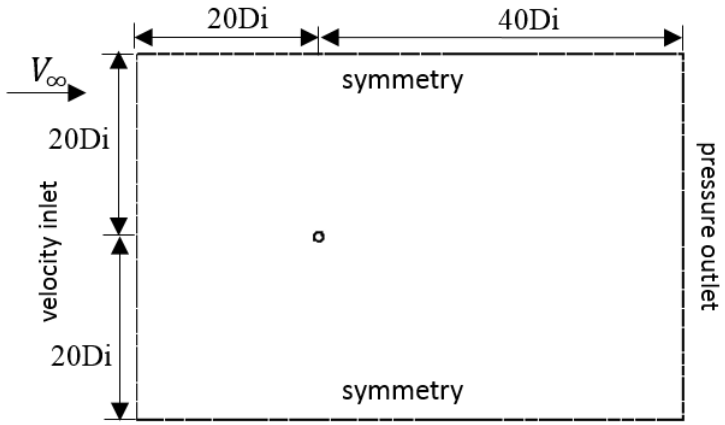


Figure 3.1: Domain geometry and boundary conditions

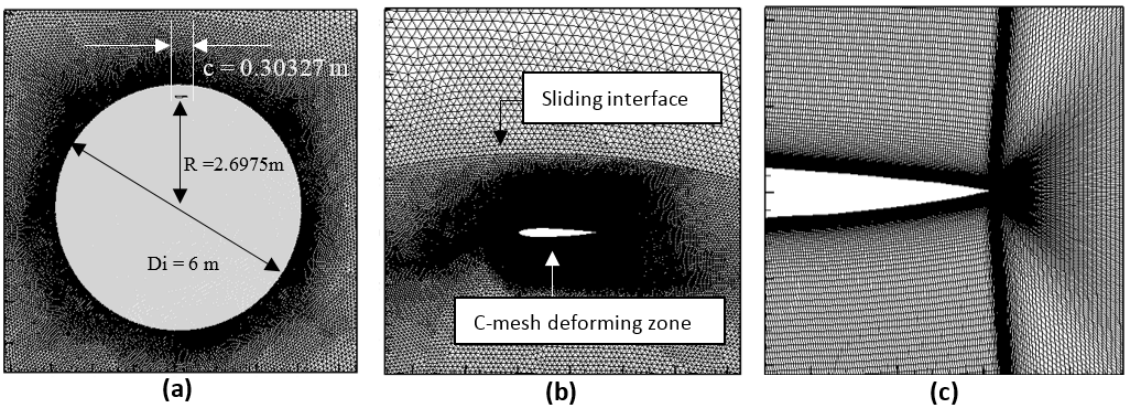


Figure 3.2: (a) Rotating zone, (b) sliding mesh, (c) mesh resolution on trailing edge

The turbulence is modeled using $k-\omega$ SST with an inlet turbulent intensity of 0.05% and inlet turbulent viscosity ratio of 0.1 for the turbulence inlet conditions. The Spalart-Allmaras model [68] and the SST transition model [37] are two other commonly used turbulence model in the VAWT literature [4, 31], therefore these models are compared with the $k-\omega$ SST in the chapter on validation of the methodology.

Table 3.1: Geometry and inlet conditions

rotor diameter	5.395m
rotating zone diameter	6.000m
rotor angular velocity	90rpm
blade chord (baseline)	0.303m
inlet velocity	8 m/s
turbulent viscosity ratio	0.1
turbulent intensity %	0.05

The spatial discretization used is a second order upwind scheme; all the momentum terms and the turbulence terms are discretized in the second order. Since implicit scheme is used for the temporal discretization, it does not need to satisfy the CFL criterion [55]. However, in order to make sure the highly unsteady flow phenomenon is properly accounted for, the time-step size used for the simulation is 0.0001s or azimuthal step of $\Delta\theta = 0.054°$.

To ensure the accuracy of the solution, the absolute convergence criteria is set to $1x10^{-5}$ for both the continuity and momentum residuals. It is expected that this level of accuracy cannot always be obtained but to ensure the residual is as close to $1x10^{-5}$ as possible, an inner loop of 50 iterations per time step is used. A conservative approach of no more than 0.1% difference in Cp between the subsequent cycles is suggested by Balduzzi et al.[4]. For this study, the simulation is carried out for multiple cycles until the average C_P per successive cycle do not differ by more than 0.1%. The simulation for the deforming case is carried out for one cycle using the converged baseline fixed blade result as initial condition.

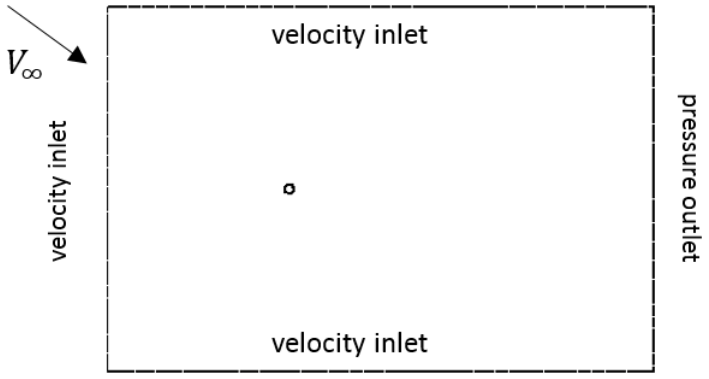


Figure 3.3: Boundary conditions for the static airfoil cases

For the static airfoil case, the same grid is used but with different boundary conditions and initial conditions. The boundary condition is shown in Fig.3.3. The velocity inlets are given the free stream condition. The static airfoil cases are performed for different angles of attack and so the horizontal and vertical components of the velocities are specified at the boundary conditions. The angles of attack for the static cases are meant to simulate the flow past the airfoil in the upwind half of the cycle; therefore as the angles of attack are specified at the inlet, the y-component of the inlet velocity will have a direction pointing downwards.

3.6 Simulation of Moving Grids

The moprhing blade for VAWT requires grid motion during simulation. The grid motion involves (i) the turbine rotation, and (ii) morphing flexure motion. In order to model the turbine rotation, the sliding mesh method is employed, while for the morphing flexure of the blade, the dynamic mesh method is employed. The sliding mesh method is preferred for the simulation of VAWT because this allows the unsteady effects of the VAWT to be simulated while keeping the computational cost reasonable. The sliding mesh moves the mesh together as a rigid body and all cells within the specified zone move with the same velocity; thus, no relative motion within the nodes occur and dynamic mesh is not required. On the other hand, the morphing of the blade requires the nodes on the blade to move to new coordinates with different relative velocities. To avoid high cell skewness or negative cell volumes, dynamic mesh motion is required for this method.

3.6.1 Governing Equations for Dynamic Mesh

The governing equations used for the simulation of moving grids require the Arbitrary Lagrangian-Eulerian (ALE) formulation of the transport equation,

$$\frac{d}{dt}\int_V \rho\phi dV + \int_{\partial V} \rho\phi\left(\vec{u}-\vec{u}_g\right)\cdot d\vec{A} = \int_{\partial V} \Gamma\nabla\phi\cdot d\vec{A} + \int_V S_\phi dV \tag{3.24}$$

where V = arbitrary moving control volume
∂V = bounding surface of control volume
$\vec{u}$ = velocity vector
$\vec{u}_g$ = mesh velocity

The first two terms in the equation are treated diffently from Eq.3.5. The time derivative now has to account for the change in cell volume and the convective flux has to account for the grid velocity. Using a first-order backward difference, the time derivative term is

computed as

$$\frac{d}{dt}\int_V \rho\phi dV = \frac{(\rho\phi V)^{(n+1)} - (\rho\phi V)^n}{\Delta t} \tag{3.25}$$

$$V^{(n+1)} = V^n + \frac{dV}{dt}\Delta t \tag{3.26}$$

Eq.3.27 must be satisfied in order to satisfy the geometric conservation law (GCL) [71].

$$\frac{d}{dt}\int_V dV - \int_{\partial V} \vec{u}_g \cdot \vec{A} = 0 \tag{3.27}$$

The change in control volume with respect to time is then computed from

$$\frac{dV}{dt} = \int_{\partial V} \vec{u}_g \cdot \vec{A} = \sum_j^{nf} \vec{u}_{g,j} \cdot \vec{A}_j = \sum_j^{nf} \frac{\delta V_j}{\Delta t} \tag{3.28}$$

where j subscript refers to the j face area vector; nf is the number of faces on each control volume, and δV_j is the volume swept out by face j for Δt.

The sliding mesh motion is a special case of dynamic mesh wherein the mesh moves as a rigid body motion. It is governed also by Eq.3.24; however since mesh motion is rigid, $dV/dt = 0$ and thus,

$$\frac{d}{dt}\int_V \rho\phi dV = \frac{V[(\rho\phi)^{(n+1)} - (\rho\phi)^n]}{\Delta t} \tag{3.29}$$

$$V^{(n+1)} = V^n \tag{3.30}$$

$$\sum_j^{nf} \vec{u}_{g,j} \cdot \vec{A}_j = 0 \tag{3.31}$$

The GCL condition can be interpreted as a stamement that any arbitrary mesh motion should not introduce any disturbance to a uniform flow [45]. The manner in which temporal integration is carried out in the Arbitrary Lagrangian-Eulerian formulation of moving mesh must satisfy the geometric conservation law in order to preserve the order of accuracy of a time-integration scheme meant for fixed grids [36, 24, 45].

3.6.2 Sliding Mesh and Non-conformal Cells

The sliding mesh is used to simulate the unsteady flow past the rotating turbine. To setup the sliding mesh, two separate zones are created with ICEM, each zone having its own face. The interface between the two face zones are created in Fluent. Figure 3.4 shows how fluent creates the interior face that connects the two face zones. The intersection of the two faces generates the faces a-d, d-b, b-e, and e-c, Fluent groups them into interior zone that allows flux to be passed on from zone 1 to zone 2. When the flux is computed across the interface to cell IV for example, the interior face d-b and b-e are used to calculate the flux instead of face D-E. At each time step, the mesh is updated and the non-conformal interface is also updated. It should be noted that the mesh motion must "slide" along the interface face for the flow to be able to go across the zones as any interface that is not in contact is treated as wall [27]. The sliding mesh method does not automatically guarantee flux conservation [28, 70] but is commonly used in unsteady rotating simulations and have been shown to be in good agreement with experimental results [38, 4].

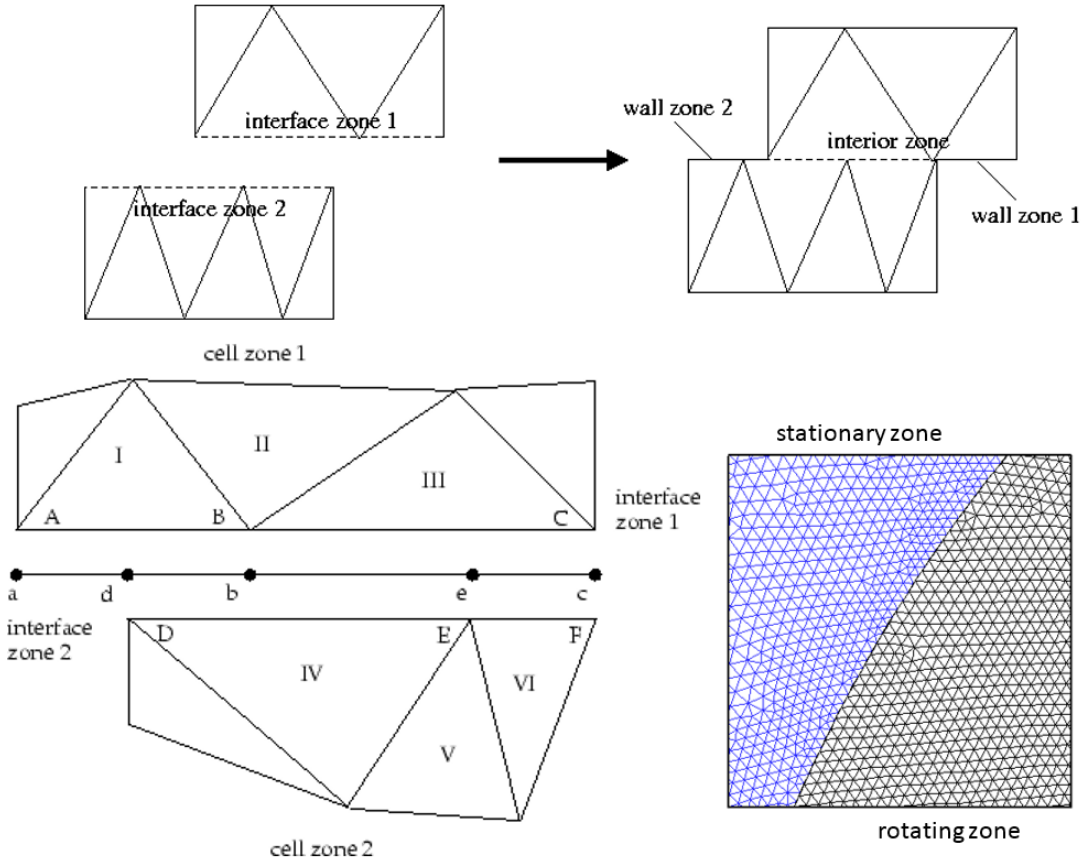


Figure 3.4: Sliding mesh and non-conformal interfaces [27]

3.6.3 Dynamic Mesh Methods

Fluent has two available methods of dynamic mesh motion, (i) spring-based and (ii) diffusion-based. The spring-based is the same as the linear spring analogy method in the literature review chapter while the diffusion-based is the same as the Laplace method. Using Fluent's diffusion-based smoothing as opposed to Fluent's spring-based smoothing assures a higher quality of grid deformation [27, 66]. Fluent diffusion-based dynamic mesh motion solves for the Laplace equation

$$\nabla \cdot (\gamma \nabla \vec{u}) = 0 \tag{3.32}$$

in which $\vec{u}$ is the mesh displacement velocity, and γ is the diffusion coefficient. The Dirichlet boundary conditions are the coordinates given to the blade surface. Fluent allows the γ to be calculated either inversely proportional to boundary distance or inversely proportional to cell volume. For the study, diffusion coefficient is calculated as a function of the boundary distance as it allows for the boundary layer resolution to be preserved while most of the displacement motion is absorbed in nodes further away from the blade. .

$$\gamma = \frac{1}{d^a} \tag{3.33}$$

wher a is the diffusion parameter. As the first cell height is very small for the grid used in the study, the validity of the cell, whether it will have negative cell volume or not, is highly dependent on this parameter. The diffusion parameter used for the simulation is $a = 1.105$; this value has been tested for morphing the mesh with added 20 interpolated subframes between each 21 frame profiles. This value is dependent on the initial mesh topology and it has also been found that adding more frames during morphing does not guarantee this parameter will not invalidate the mesh. Fluent diffusion-based deformation uses finite element method to solve for the Laplace equation where $\vec{u}$ obtained for each node. The nodes are then moved with the displacement velocity $\vec{u}$ and node position is updated as

$$\vec{x}_{new} = \vec{x}_{old} + \vec{u}\Delta t \tag{3.34}$$

It should be noted that the mesh deformation does not check for the quality nor the validity of the cells after deformation. If the mesh is invalidated in the process, the solver will be unable to calculate the solution for the next time step and abort.

3.6.4 User-Defined Function

In order to morph the blade, a user-defined-function (UDF) is needed to specify the boundary motion of the profile deformation. As shown in Fig.3.5, the node coordinates on the blade surface are grouped into upper, lower, and tip where the nodes are given new coordinates based on Fourier series expansion representation of experimental data points; there are total of 21 frames for the actuation range of SSMA which can be found in Appendix A. The Fourier coefficients are different for upper and lower surface, while tip is calculated as a straight line that joins the last nodes on upper and lower. Each surface is linked to their own DEFINE_GRID_MOTION function.

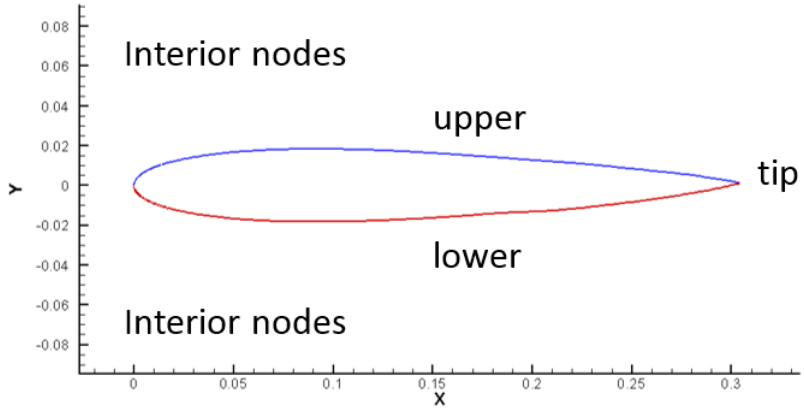


Figure 3.5: Blade surface split into upper, lower, and tip for specifying coordinates

The leading edge geometry of the blade profile is kept the same and so the blade is morphed only for nodes that have x-coordinates of x $>$ 0.1609m or at approximately half the chord length; anything before this value remains the same as the standard NACA0012.The UDF algorithm is shown in Fig.3.6 and the code is presented in Appendix B. All coordinates are calculated on the azimuthal position $\theta = 0°$ coordinate space and are transformed in their corresponding azimuthal coordinate space at the point of deformation.The deformation is flagged off by default and is only flagged at certain azimuthal positions since the blade is not morphing continuously throughout the whole cycle. The sequence of execution for Fluent's dynamic motion is as follows: DEFINE_GRID_MOTION gets called first and Laplace equation is solved for the dynamic mesh, cell zone motion increments the whole rotating zone by $\Delta\theta°$ with the sliding mesh method, then DEFINE_ADJUST is called at the beginning of each iteration. The subroutine that calculates the new coordinate positions are executed within DEFINE_ADJUST so that at next time step the DEFINE_GRID_MOTION can assign the new coordinates to the blade. Since DEFINE_ADJUST function gets called at the beginning of each iteration, a conditional statement must be included so that the

necessary subroutine gets executed only once per time step.

Although there are 21 frames of coordinate data available, subframes can be added between each frames through linear interpolation. The addition of subframes serves as a practical approach to control the speed of deformation without changing the time step size. This is because the node coordinates can be modified only at each time step before calculating the solution. As an example, if only 21 frames are used for a time step size of 0.0001s, the blade will be morphing for a mere 0.00021s for the whole actuation range. For the simulation of the morphing blade, 20 subframes are added between each frames; this is the maximum subframes that can be added without invalidating the mesh. Although the diffusion-based method handles the interior mesh motion, it cannot prevent the deterioration of cell quality. This deterioration is exacerbated as the number of incremental displacement increases. In order to ensure the quality of the grids are not deteriorated every cycle, the initial grid coordinates are stored for all nodes inside the C-mesh zone, and these coordinates are retrieved instead of recalculated when the target profile cycles back to the initial profile. In fact, the whole range of actuation cannot be obtained without invalidating the mesh unless the initial mesh is retrieved as this frame is traversed during actuation.

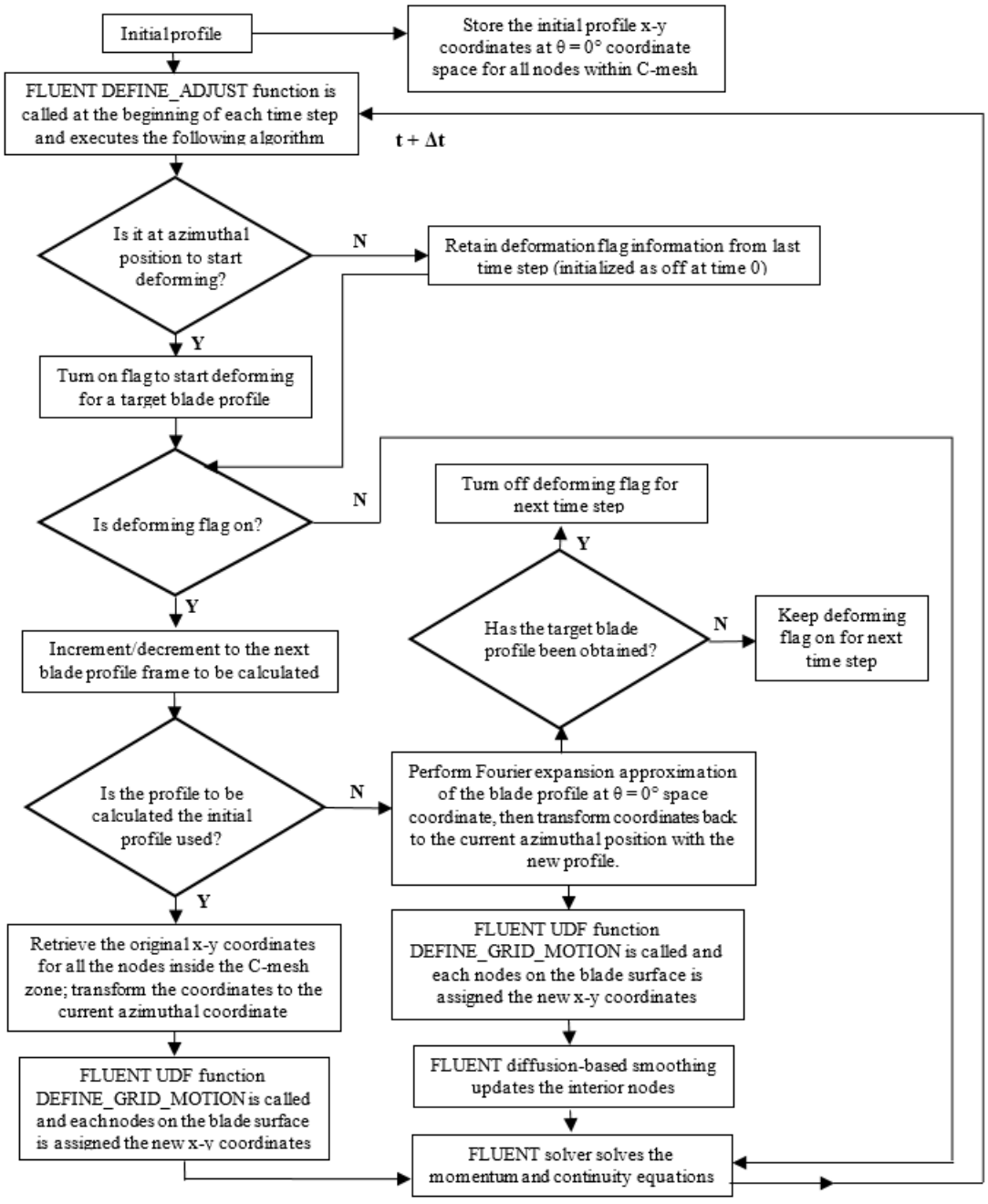


Figure 3.6: Algorithm diagram for specifying node movement for blade flexure motion

Chapter 4

Verification and Validation

Verification and validation of the model and methods used in the study is a means to quantify and be aware of the uncertainties involved. The verification step is meant to check if the equations and model are being solved correctly, while validation is a means to check whether the equations being solved are representative of the physical reality [50].

4.1 Grid and Time Step Sensitivity Analysis

The verification of the CFD method employed is done by performing grid sensitivity and time step size sensitivity analysis in order to assess the discretization error of the CFD method used for the study. The spatial and temporal sensitivity analysis are performed for the same geometry and boundary conditions as the one used for the study but with varying mesh resolution and time step size. The grid refinement factor is based on the minimum refinement factor when performing Grid Convergence Index (GCI) according to Roache [61], which is based on Richardson's extrapolation (h-extrapolation) method [60]. For unstructured grids, the grid refinement index r is calculated from the number of elements used in the finer grid N_1 and coarser grid N_2 where D is the number of dimensions on the domain.

$$r = \left(\frac{N_1}{N_2}\right)^{1/D} \tag{4.1}$$

The grid is refined by 1.3 in each dimensions; since it is a 2D mesh, this means the total cell count increases by approximately 1.32. It should be noted however, that the first cell height from the airfoil wall and the number of boundary layers are kept the same for all grids after the simulation results from $G2$ has shown that this first cell height and boundary layer gives the desired y+ value and resolves the boundary thickness as it is shown in the

next section. The grid sizes are summarized in Table 4.1, where the $G1$ is the finest mesh; therefore $N_1 > N_2 > N_3$. Two time step sizes are used, $\Delta t = 0.0003$s or $\Delta\theta = 0.162°$ and $\Delta t = = 0.0001$s or $\Delta\theta = 0.054°$. The C_P is the objective variable used to check the convergence where each simulation is run on multiple cycles until the average C_P of two successive cycles do not differ by more than 0.1% as recommended [4]. The results for the grid and time step size convergence are shown in Fig.4.1 and Fig.4.2. The difference between the grid and time step refinements are clearly seen in the azimuthal region of 90° to 180° where stall is expected to occur. It can be observed that the time step size of $\Delta\theta = 0.054°$ is necessary for the study; $G2$ is used instead of $G1$ to reduce computational cost. The trend in mesh convergence study is similar to what Zadeh et al. [84] have seen wherein coarser grids are shown to give lower values in C_P for 2D VAWT simulations.

Table 4.1: Summary of grid and time step sensitivity analysis

grid name	cell count N	boundary layers	nodes on blade	ave y+	$C_P(\Delta\theta = 0.162°)$	$C_P(\Delta\theta = 0.054°)$
G3	227,977	50	523	≤ 1	0.245855	0.231899
G2	384,768	50	680	≤ 1	0.251283	0.242973
G1	658,549	50	884	≤ 1	0.278331	0.243167

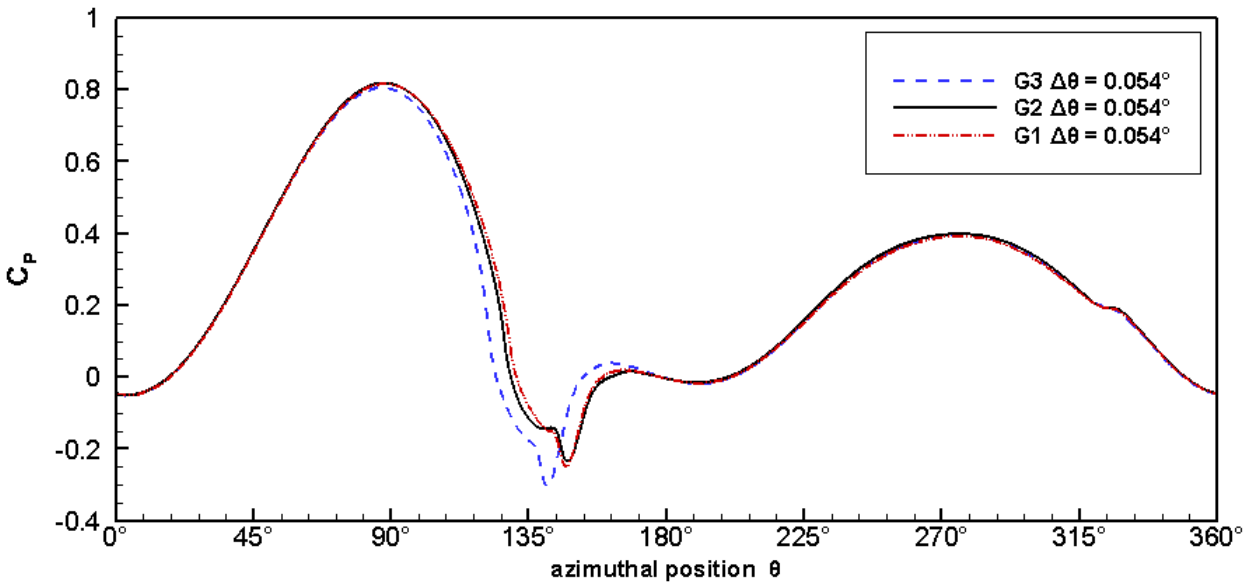


Figure 4.1: Grid sensitivity with C_P as objective variable

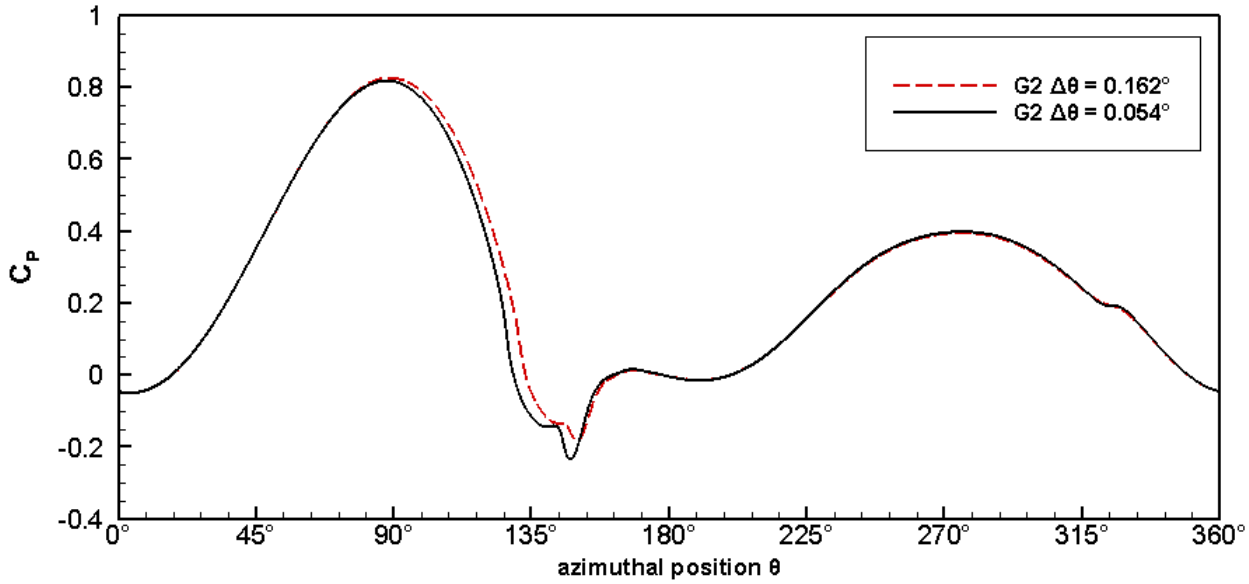


Figure 4.2: Temporal sensitivity with C_P as objective variable

4.2 Grid Resolution at Boundary Layer

The boundary layer is characterized by highly viscous flow where there are large velocity gradients in the direction normal to the blade surface. The no-slip condition on the blade surface means that velocity is zero relative to the blade surface; however, flow conditions at the outer edge of the boundary layer thickness is the same as the flow conditions if it the flow was inviscid [3]. The wall shear stress and boundary layer thickness are important parameters for the modelling of turbulent flows. The accuracy of the turbulence model is dependent on how well the boundary layer is resolved. An important parameter is the $y+$ which is a measure of the first cell height from the wall, and to be as accurate as possible while using $k-\omega$ SST, a $y+$ of less than one is required [47]. When the grid was generated, the first cell height had to be approximated with a desired y+ of 1 or less using the equation for y+ [79],

$$y+ = \frac{\rho u^* y}{\mu} \tag{4.2}$$

where y is the first wall height and u^* is the friction velocity which can be calculated from the wall shear stress τ_w.The wall shear stress requires the skin friction coefficient in which for the calculation of the wall distance is approximated using the Schlichting skin friction correlation [65].

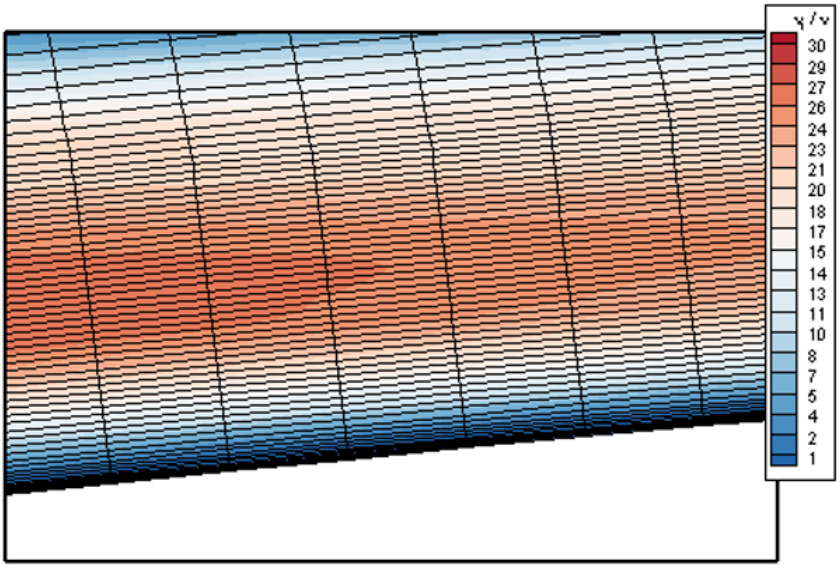


Figure 4.3: Turbulent viscosity ratio near the wall

Figure 4.3 shows the turbulent viscosity ratio near the surface of the blade. The boundary layer is shown to be well resolved for $k-\omega$ SST turbulence model as the region near the wall is laminar. As the normal distance moves further away from the wall from the buffer region to the logarithmic region, the turbulent viscosity ratio gradually increases before going back to being laminar [79]. The region of maximum turbulent viscosity is expected to be in the middle of the boundary layer; thus the boundary layer edge can be approximated as twice that height. The simulation result gives a y+ on the blade that ranges from 0.1 to 1.2 depending on the azimuthal position and the location on the blade surface, but the average y+ throughout the cycle is less than 1.

4.3 Validation with Experiment

After verifying that the equations are being solved in the proper manner, it is now required to check that the equations are representative of reality. In order to validate the method and setup used for the simulation of vertical axis wind turbine, the benchmark test case of Castelein et al. [14] is compared with the simulation results run on Fluent. The test case geometry uses 2-bladed NACA0018 profiles with turbine blade chord of 0.06m, span length of 1m, and rotor radius of 0.5m operating at $\lambda = 2$ with free stream velocity of 10.2 m/s. The experimental measurements are acquired by Particle Image Velocimetry (2C-PIV) at mid-span of the turbine, allowing it to be comparable to a 2D simulation. It should be noted however that the tower in the middle of the turbine rotor is not included in the simulation, which might explain discrepancies in the results.

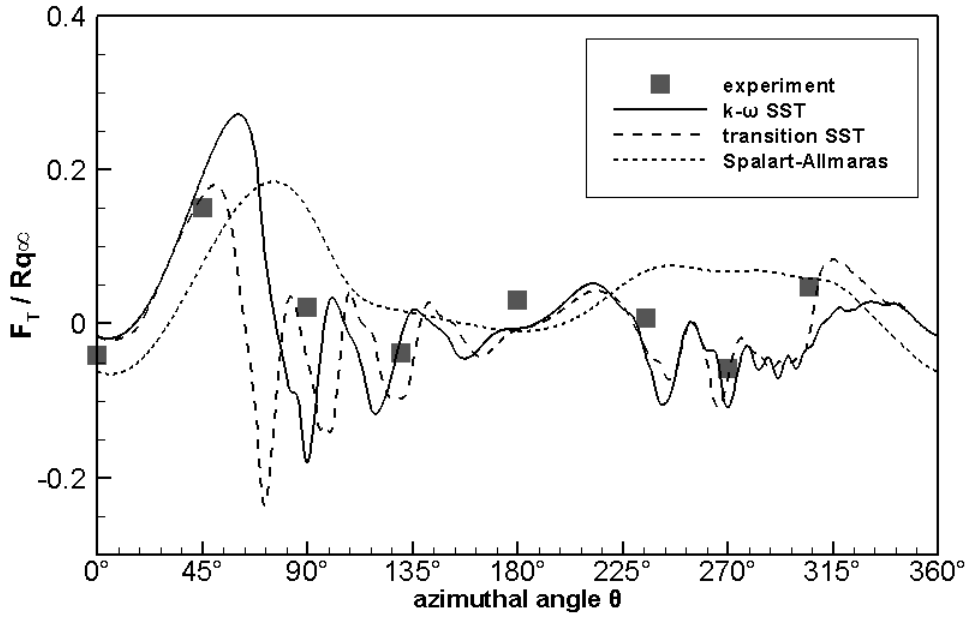


Figure 4.4: Validation of experimental case for $\lambda = 2$ with non-dimensional tangential force

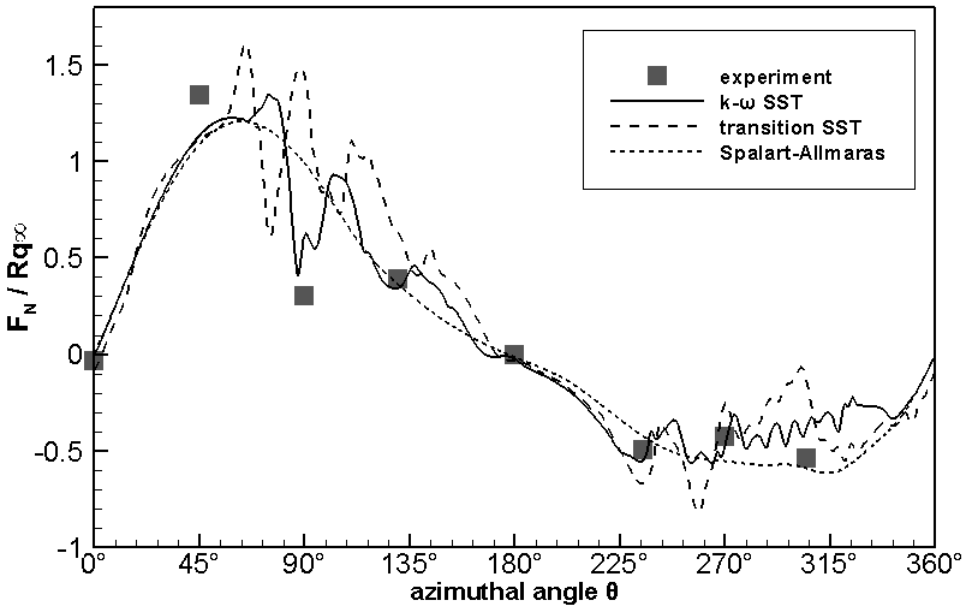


Figure 4.5: Validation of experimental case for $\lambda = 2$ with non-dimensional normal force

The numerical curves of the non-dimensional tangential and normal force are compared with the experimental results in Fig.4.4 and Fig.4.5, respectively. The region between 90° to 180° is the expected region of deep stall which might not be properly characterized with the URANS models. The $k - \omega$ SST has the highest peak tangential force but agrees well with the experiment throughout the rest of the cycle. Transition SST has a similar trend as

the $k-\omega$ SST model except it has a lower peak value. The Spalart-Allmaras model fared poorly especially as the stall is clearly not captured. For the normal force the $k-\omega$ SST gives the closest result with the experiment. The experiment was chosen since it exhibits dynamic stall and is at a Reynolds number comparable with the one being used in the study and can be compared with a 2D case. It can be argued that Transition SST gives better agreement with experimental result however, since this study deals with a slightly higher tip speed ratio of 3.17, it has been decided that the two equation $k-\omega$ SST gives close enough result, as the four equation transition SST is more costly to compute.

4.4 Choice of Temporal Discretization

The difference between the solution for 2nd order temporal discretization and 1st order temporal discretization are summarized in Table 4.2. The C_P curve difference for $\delta = +0.48°$ is shown in Fig.4.6; the main difference is in the region where stall occurs. As it can be seen from Fig.4.7, the turbulent viscosity ratio in the first order is more diffusive; however, looking at Fig.4.6, the difference in C_P between the two methods is almost negligible but this difference in solution is more apparent for $\delta = +6.98°$ which experiences higher effective local angles of attack compared to the two other profiles. Although 2nd order implicit was used initially for all simulations, there were unphysical flow behavior when deforming mesh was used where large pressure gradients occur in the interface between the deforming region and non-deforming region, which was not observed for 1st order implicit. Moreover, when remeshing is required, Fluent automatically limits the choice of temporal discretization to 1st order. It is uncertain what causes the numerical error when using 2nd order implicit together with dynamic mesh motion in Fluent; however the rest of the simulation results in the Results and Discussion section are performed with 1st order implicit in time due to this reason.

Table 4.2: Average C_P per cycle for 1st order and 2nd order temporal discretization

profile	1st Order	2nd Order	relative difference
$\delta = +6.98°$	0.14388	0.15313	6.0454%
$\delta = +0.48°$	0.24297	0.24833	2.1573%
$\delta = -13.37°$	0.24368	0.24325	0.1773%

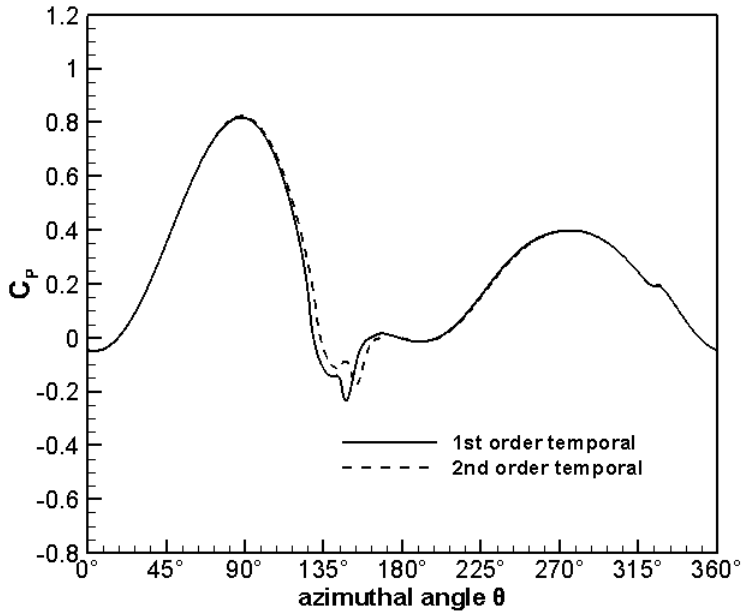


Figure 4.6: C_P comparison for $\delta = +0.48°$

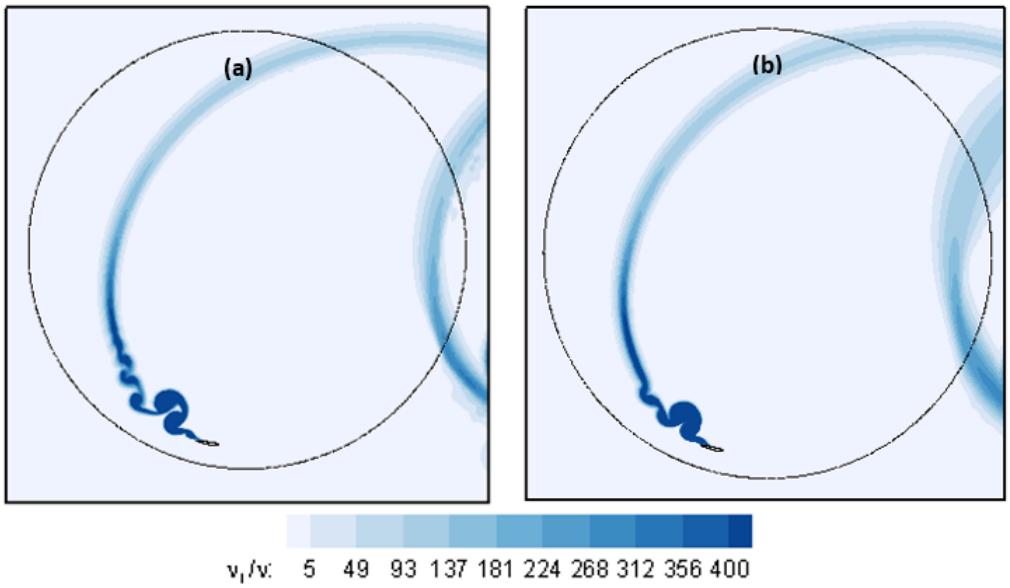


Figure 4.7: Turbulent viscosity ratio (a) 2nd order (b) 1st order temporal for $\delta = +0.48°$

4.5 Mesh Deformation Quality

The quality of the grid is important as it affects the solution convergence rate as well as the accuracy of the solution. Figure 4.8 shows the mesh before and after it is deformed, and it becomes apparent that there is a loss in orthogonality after grid deformation. To assess the impact of dynamic mesh on the grid quality, two mesh quality metrics are used, the orthogonality and skewness.

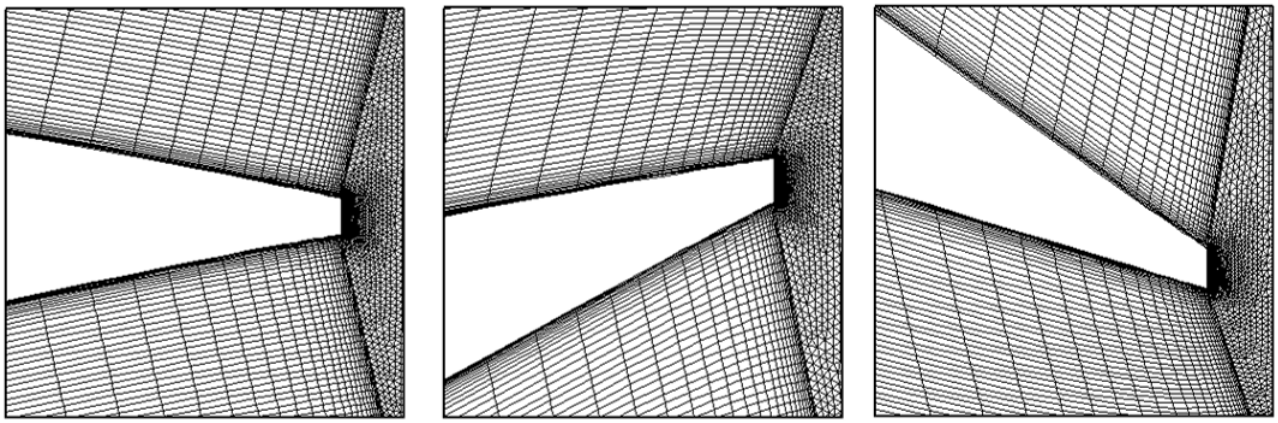


Figure 4.8: (a) Initial mesh, (b) $\delta = +6.98°$, and (c)$\delta = -13.37°$ after defromation

The mesh orthogonality for the grids before and after dynamic meshing is shown in Fig.4.9. The initial orthogonality on the upper and lower surface of the blade have high orthogonal grids, but the grid downstream of the blade has lower orthogonality, which could have been avoided with a better meshing strategy. Nevertheless, the most critical region is the boundary layer on the blade surface, which can be seen to possess 0.80-0.95 orthogonality. However, for the grids that have been deformed, close to the trailing edge tip, the grid orthogonality drops to almost 0.5. This effect is more severe for the case of $\delta = -13.37°$ which is expected since the blade tip deflection is more severe than the case of $\delta = +6.98°$. The grid skewness is presented in Fig.4.10 where it can be seen that skewness is increased on the grid surrounding the trailing edge close to the tip. Again we see that the effect is more severe for the case of $\delta = -13.37°$.

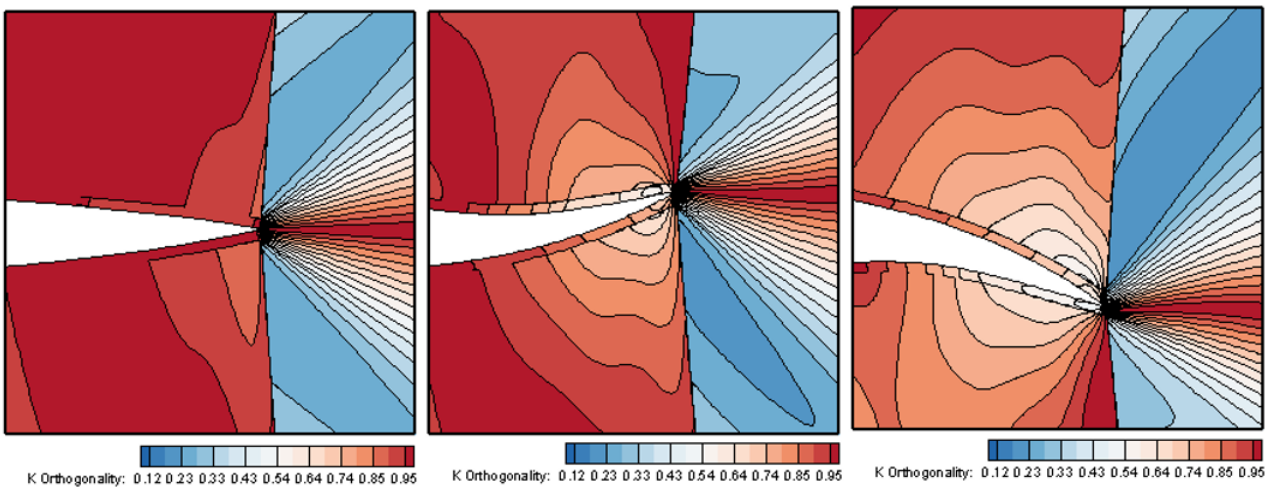


Figure 4.9: Orthogonality of the grids before (left) and after deformation (middle, right)

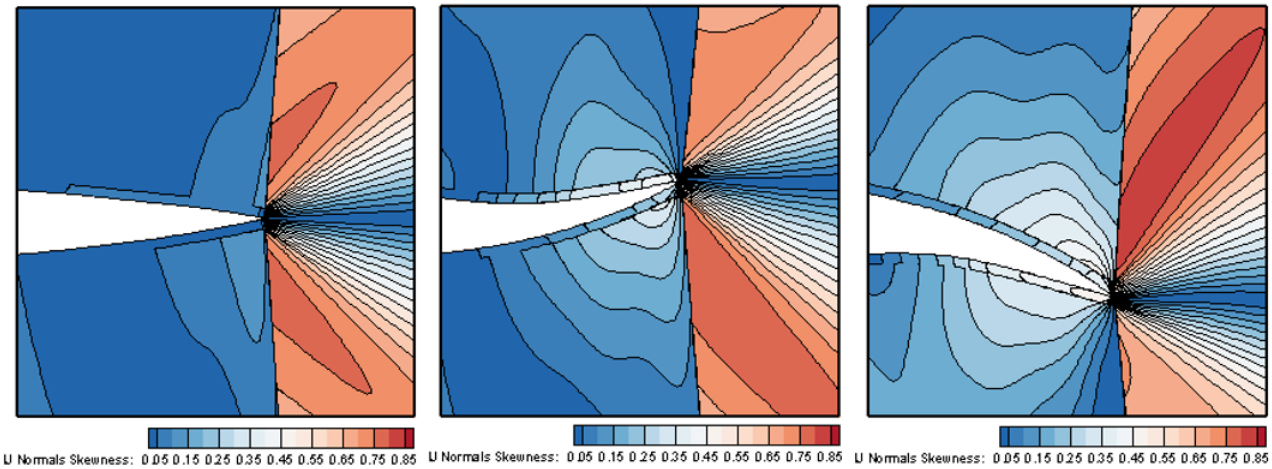


Figure 4.10: Skewness of the grids before (left) and after deformation (middle, right)

Chapter 5

Results and Discussion

This section first covers the static airfoil analysis to assess the static stall angle of the airfoil and to get an insight on how they should behave in VAWT cycle. The three profiles are then simulated for the rotating VAWT case and their C_P are compared in the upwind and downwind half cycles. These results form a basis for the morphing case. The morphing blade is applied on the static case and it is shown how the C_L and C_D curves can be modified by morphing the blade. The morphing blade is then implemented on the VAWT case where the challanges and limitations of the commerical software is discussed. The chapter closes by suggesting future works for the simulation of morphing VAWT.

5.1 Airfoil Static Stall Angle Investigation

The static analysis of the three airfoil profiles will give an insight as to how each profile behaves at different angles of attack. For the static case, the same grid is used as the one for the VAWT simulation; however, in this case the boundary conditions and initial conditions are different. There is no sliding mesh employed for the static case; the velocity used for the inlet condition is the VAWT average local relative velocity on the blade which in this case is 25 m/s and therefore Re = 5.18 x 10^5. The angle of attack is specified so that the flow hits the blade from above as this is the case for the VAWT local angle of attack in the upwind half cycle (positive lift points toward rotor center). Further details on the boundary conditions and initial conditions for the static case are specified in the models and methodology chapter.

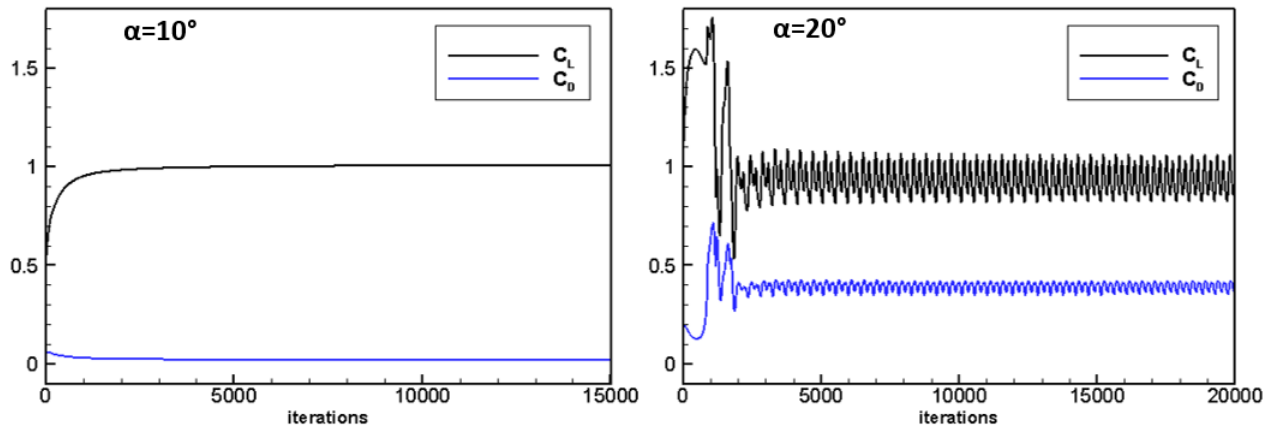


Figure 5.1: Static airfoil $\delta = +0.48°$ convergence history, $\alpha = 10°$ (left), $\alpha = 20°$ (right)

The convergence history of C_L and C_D for two angles of attack are shown in Fig.5.1. For low angles of attack the lift and drag are easily obtained; however, at high angles of attack, as is shown for $\alpha = 20°$, C_L and C_D both oscillates due to separation and vortex shedding. In the case of high angles of attack, both lift and drag are calculated by taking the average of the periodic solution.

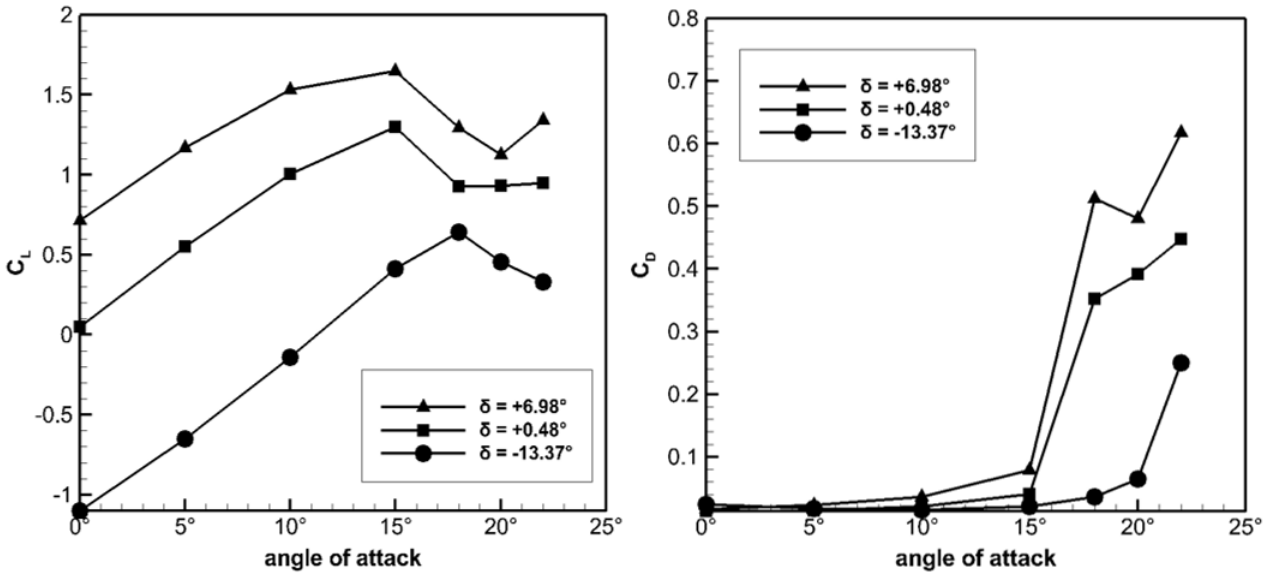


Figure 5.2: C_L and C_D vs angle of attack for the static airfoils

The static lift and drag coefficients for the three airfoil profiles are shown in Fig.5.2. The profile with $\delta = +0.48°$ is close to NACA0012 and is shown to start stalling at an angle of attack between 15° and 18°. To give an idea of NACA0012 C_L and stall angle range, at $Re = 10^6$ for NACA0012, the stall angle is known to be approximately $\alpha = 16°$ with $C_{L,max} = 1.6$ [1]. Wind tunnel results from ONERA for NACA0012 for $Re = 5$ x 10^5 suggests that onset of stall occurs at an angle of attack closer to $\alpha = 12°$ [73].

Within expectations, $\delta = +6.98°$ profile has a higher lift curve due to the camber while $\delta = -13.37°$ profile has lower C_L curve but higher stall angle due to the inverse camber. The drag curves are shown in Fig.5.2, where it can be seen that drag coefficient is almost constant at low angles of attack but rapidly increases at higher angles of attack. At high angles of attack where separation occurs, drag coefficient is very unstable and difficult to measure accurately especially with RANS [22, 29].

Figure 5.3 shows the static pressure contour of the three blade profiles at three different angles of attack. The maximum local effective angle of attack experienced by the blade for the VAWT case having $\lambda = 3.17$ is in the range of $10° \leq \alpha \leq 22°$ for the three blade profiles. It is worth noting that at $\alpha = 0°$, the suction side and pressure sides are inverted for $\delta = -13.37°$, thus it has a negative lift at low angles of attack. At $\alpha = 20°$, the $\delta = -13.37°$ is the only profile that does not have vortex shedding. This indicates that among the three blade profiles, the $\delta = -13.37°$ is the least likely to experience dynamic stall or blade-vortex interactions when the blades are implemented for VAWT.

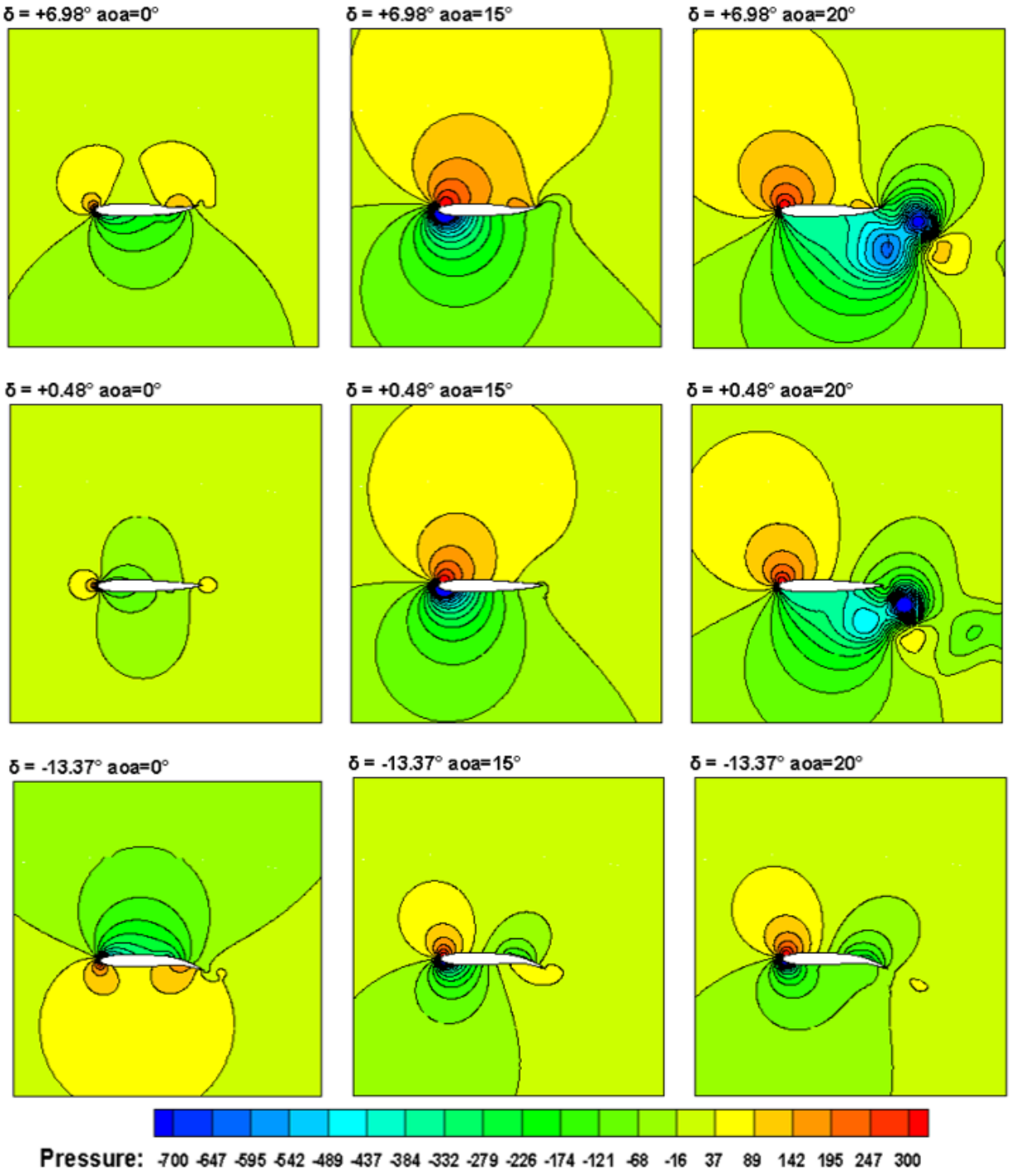


Figure 5.3: Pressure contour [Pa] for the static airfoils

5.2 VAWT Fixed Profile Results

In order to better understand the flow behavior of each blade profiles for the VAWT case, the dynamic lift and drag coefficients are plotted against blade angles of attack as presented in Fig.5.4. Both the lift and drag coefficients are non-dimensionalized using the relative velocity instead of the free stream inlet velocity.

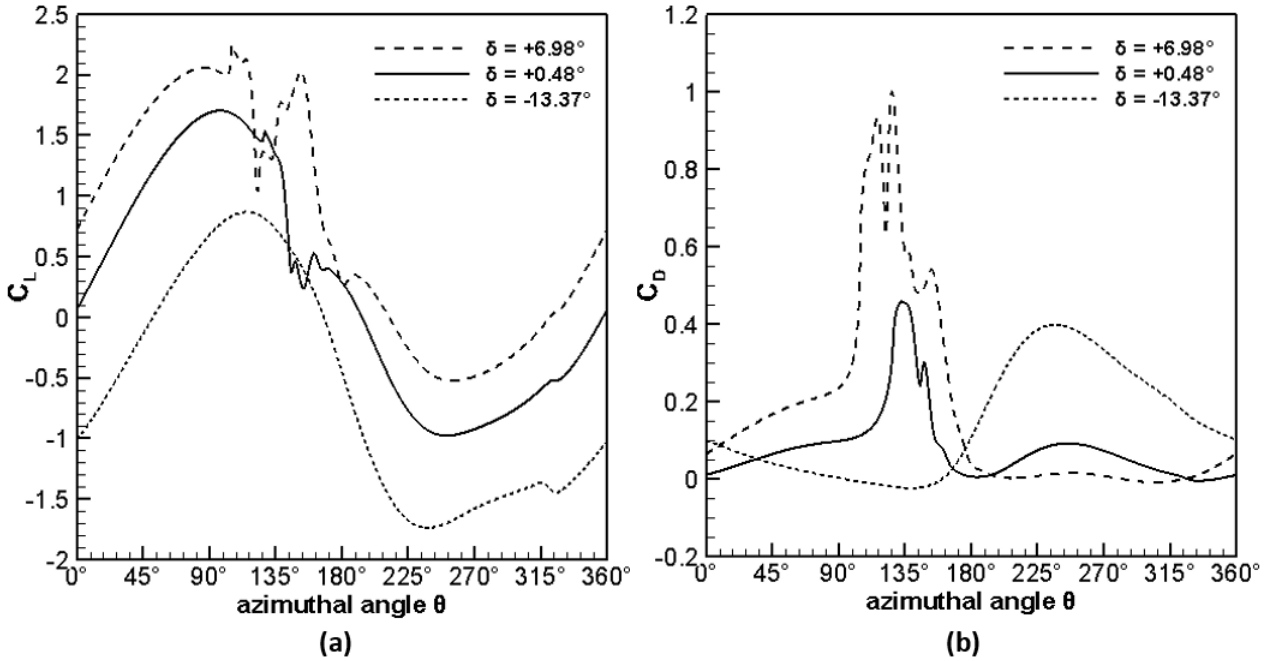


Figure 5.4: (a) C_L, (b) C_D for the three blade profiles for $\lambda = 3.17$

The cambered profile $\delta = +6.98°$ generates the most lift upwind but also incurs the highest drag; on the other hand, the cambered profile $\delta = -13.37°$ shows much of the lift it generates is on the downwind side. The sharp decrease in lift during dynamic stall is preceded by an increase in lift due to the separation bubbles that occur when the critical angle of attack is exceeded. Drag is noticeably highest for the cambered profile $\delta = +6.98°$. Whilst the cambered profile $\delta = +6.98°$ produced the most lift, it also approaches stall the soonest; This corresponds to the azimuthal position close to $\theta = 100°$; as also noted by Laneville and Vittecoq [76], this is the region where a symmetric airfoil is likely to stall for $\lambda = 3$; past this azimuthal angle, the cambered profile $\delta = +6.98°$ does not produce substantial lift. The opposite is true for the cambered profile $\delta = -13.37°$, where it barely generates any lift on the upwind, while except for the observed lift increase on the first quarter of the downwind cycle.

The static pressure contour plots for azimuthal locations $\theta = 120°$ and $\theta = 300°$ are shown in Fig.5.5. At $\theta = 120°$, $\delta = +6.98°$ is already experiencing dynamic stall, while it doesn't occur yet for $\delta = +0.48°$; as for $\delta = -13.37°$ pressure gradient is very small between suction and pressure side. On the contrary, for $\theta = 300°$, $\delta = -13.37°$ is the blade profile that generates the most pressure gradient. These results are all consistent with the lift and drag coefficients presented in the previous section. Figure 5.6 shows the vorticity contours of the three profiles. Both $\delta = +6.98°$ and $\delta = +0.48°$ has vortex shedding that occurs in the region $90° \leq \theta \leq 180°$ but the $\delta = -13.37°$ does not shed any vortices throughout the whole azimuthal range. It is worth noting that the vortex shed downstream does not hit the same blade in the same cycle or in the next cycle. This is not the case if there is more than one blade on the turbine.

The coefficient of power C_P for all three blade profiles are plotted for comparison in Fig.5.7, while Fig.5.8 shows how much more C_P could be generated from the envelope of the individual blade profiles. The individual C_P curves show that the cambered profile $\delta = +6.98°$ gives a much better C_P on the upwind while the cambered profile $\delta = -13.37°$ performs better on the downwind half. This trend is similar to what Danao et al. [19] found when they compared the effects of camber and reverse cambered LS0421 blade on the C_P of VAWTs. Table 5.1 shows the average C_P for each blade profiles for the upwind, downwind, and for the whole cycle. The C_P values are averaged from the last simulation cycle, where convergence criteria is reached only when C_P average of subsequent cycles do not vary for more than 0.1%.

Table 5.1: Average C_P comparison among blade profiles

blade profile	$C_{P,ave}$ upwind	$C_{P,ave}$ downwind	$C_{P,ave}$ per cycle	% change in C_P
$\delta = +6.98°$	0.251871	0.035881	0.143876	-40.78%
$\delta = +0.48°$	0.287725	0.198296	0.242973	-
$\delta = -13.37°$	0.095039	0.392273	0.243679	+0.29%
envelope	0.413283	0.393675	0.403482	66.06%

In the upwind half, the baseline profile $\delta = +0.48°$ generates the most power, while on the downwind half $\delta = -13.37°$ profile gave the highest power. Although the $\delta = +6.98°$ profile has the highest peak C_P, it has the highest loss of C_P due to stall, therefore lowering its average in the upwind half. The average C_P per cycle is highest for the $\delta = -13.37°$ profile but it is only 0.29% higher than the baseline profile. Nevertheless, $\delta = +6.98°$ produced the highest peak C_P per cycle; therefore, if the $\delta = +6.98°$ is utilized only for the upwind half while adopting the $\delta = -13.37°$ before the $\delta = +6.98°$ profile stalls, the C_P could be

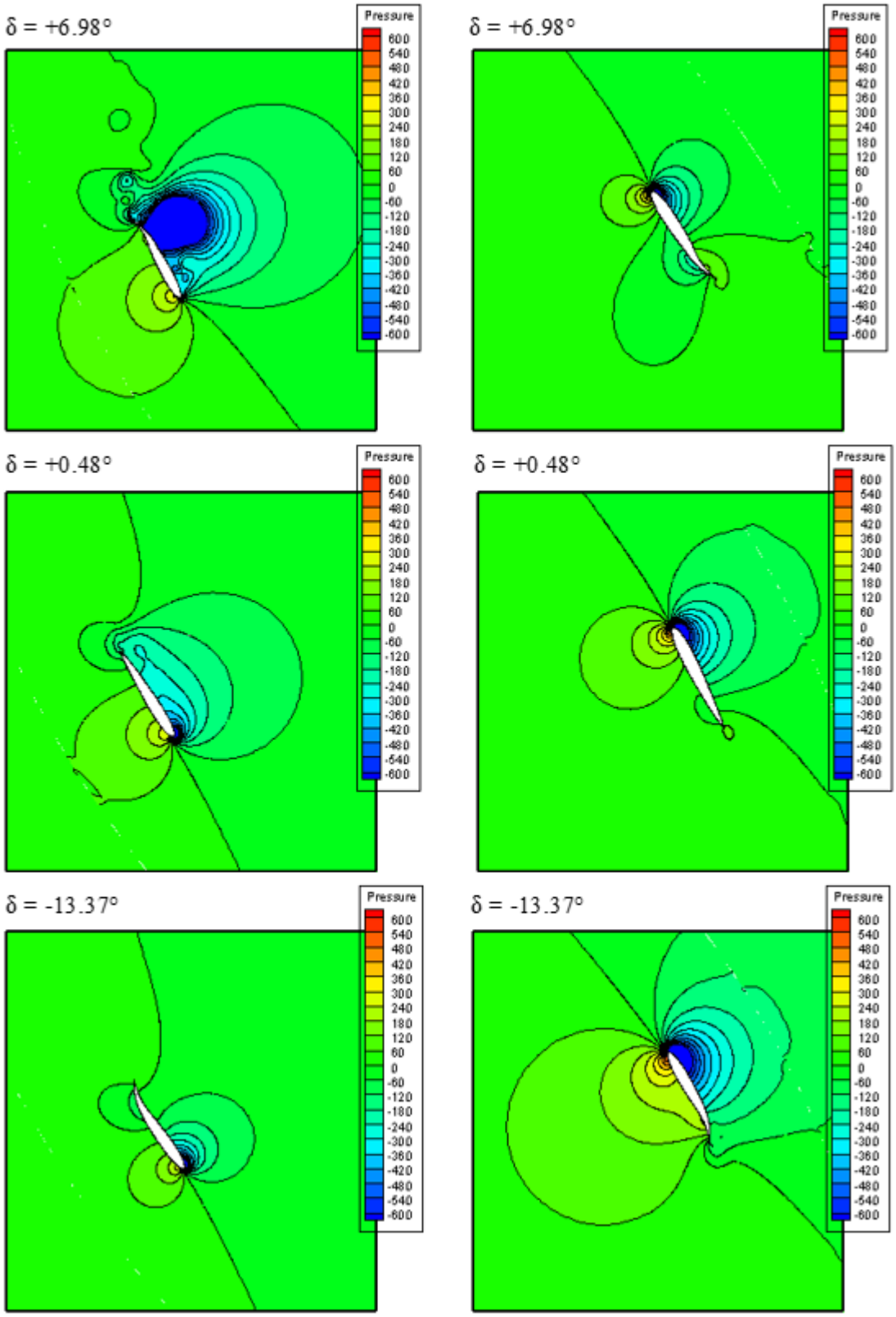


Figure 5.5: Pressure contours [Pa] for $\theta = 120°$ (left column) and $\theta = 300°$ (right column)

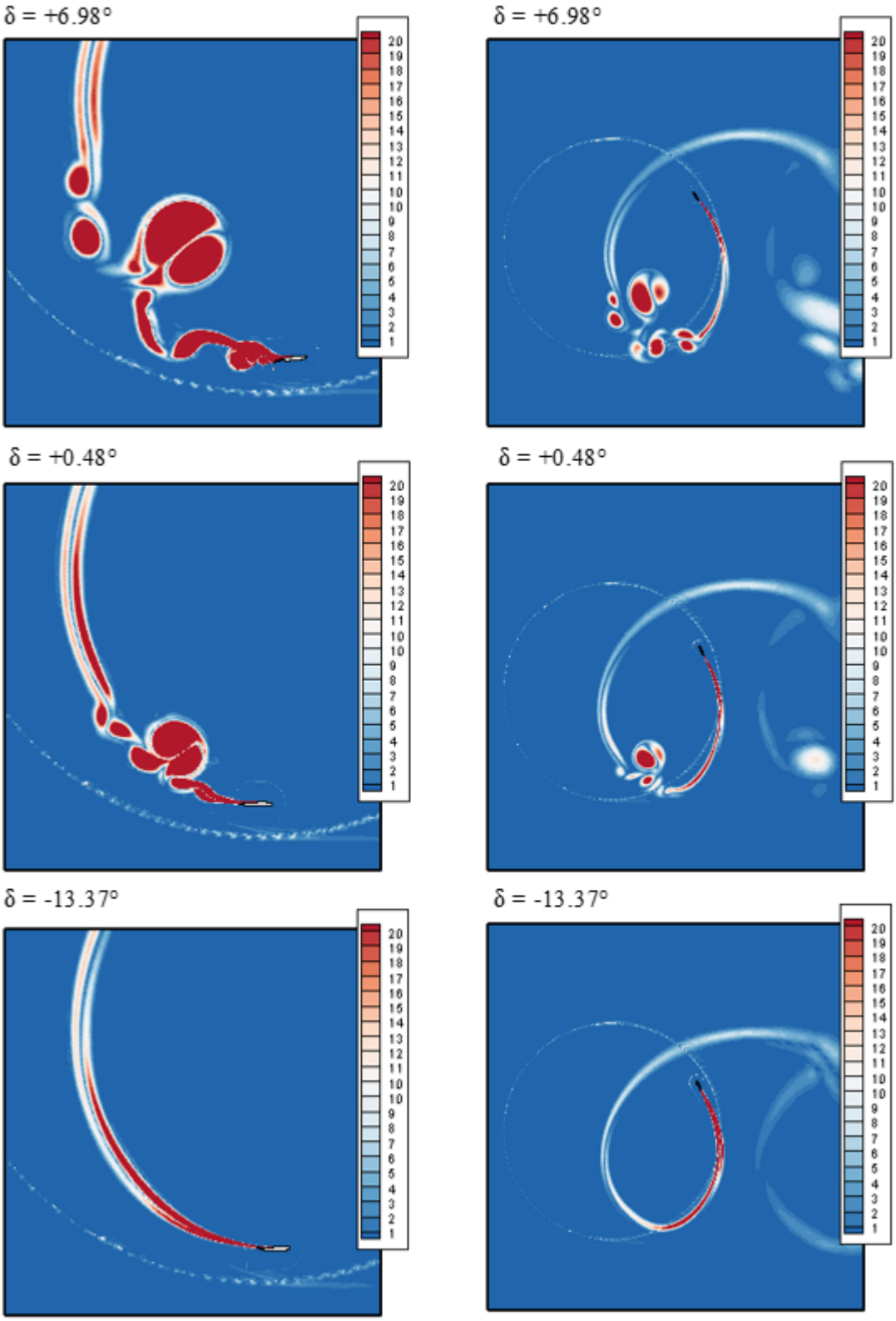


Figure 5.6: Vorticity contours for $\theta = 120°$ (left column) and $\theta = 300°$ (right column)

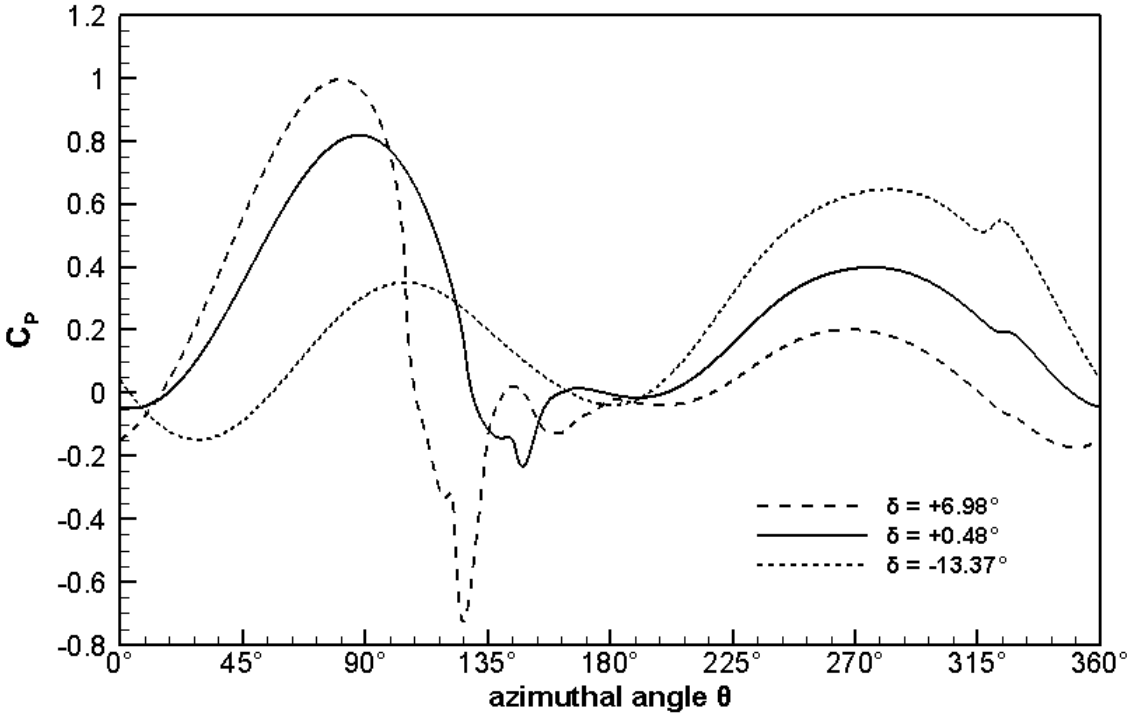


Figure 5.7: Coefficient of power C_P at $\lambda = 3.17$ for each fixed blade profiles

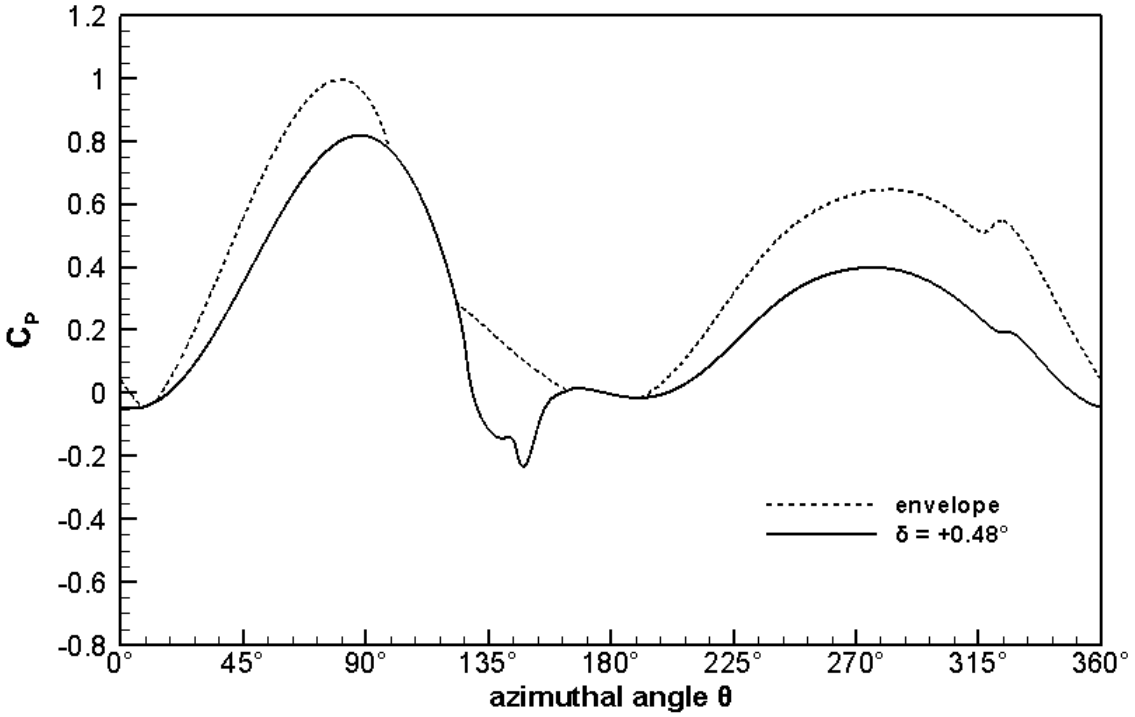


Figure 5.8: Coefficient of power C_P at $\lambda = 3.17$ of baseleine profile and envelope

optimized. The envelope C_P is the maximum that can be obtained when the maximum values for C_P are obtained from each blade profile; although it is not feasible to simply take the envelope C_P and expect the morphing result to be the same, it gives an upper bound to the possible C_P increase of 66.06%. The results from the aerodynamic performance of each individual blade shall be a basis for the deforming blade simulation. In particular, the blade will be morphed to give the profile that gives the closest possible result to the envelope C_P.

5.3 Morphing Trailing Edge at Static Stall Angle

To investigate the capability of morphing the trailing edge to prevent stall, the case is investigated for the benchmark profile with flow conditions based on the local angle of attack and local relative velocity at the azimuthal position of $\theta = 100°$. The flow conditions are chosen based on this location as it is the location at which local angle of attack approaches $\alpha = 18°$, which is the static stall angle for the benchmark profile. The local velocity of 25.3m/s is the relative velocity at $\theta = 100°$. The same mesh used for the VAWT case is used for the static morphing airfoil case but with different initial and boundary conditions. The details of the static airfoil setup can be found in the models and methodology chapter. Figure 5.9 shows the time history of C_L and as it is morphed from $\delta = +0.48°$ to $\delta = -13.37°$. The figure on the right shows the close up history in the duration of morphing.

The blade is morphed at $t = 0.600$s and can be identified from the time history where large jumps in solution occurs. The total duration of morphing is 0.014s but it takes a while for the solution to converge to the same solution of the static $\delta = -13.37°$. The comparison of the C_L and C_D between the morphed and static $\delta = -13.37°$ is presented in Fig.5.10 and summarized in Table 5.2 where it shows that the difference between the solution is very small.

Figure 5.11 and Fig.5.12 show the static pressure contour as the baseline blade profile $\delta = +0.48°$ is morphed to $\delta = -13.37°$. As the effective angle of attack is being decreased by morphing the blade, the region of high pressure is also seen to be decreasing as the blade trailing edge is morphed to $\delta = -13.37°$. From the static results of morphing the airfoil, it was shown that the stall can be delayed; however, it takes a couple of time step to converge to the new solution after morphing. The airfoil stopped morphing at $t = 0.614$s but has yet to reach a converged solution until $t = 0.9$s; for a time step size of 0.0001s, this is equivalent to 2860 time steps.

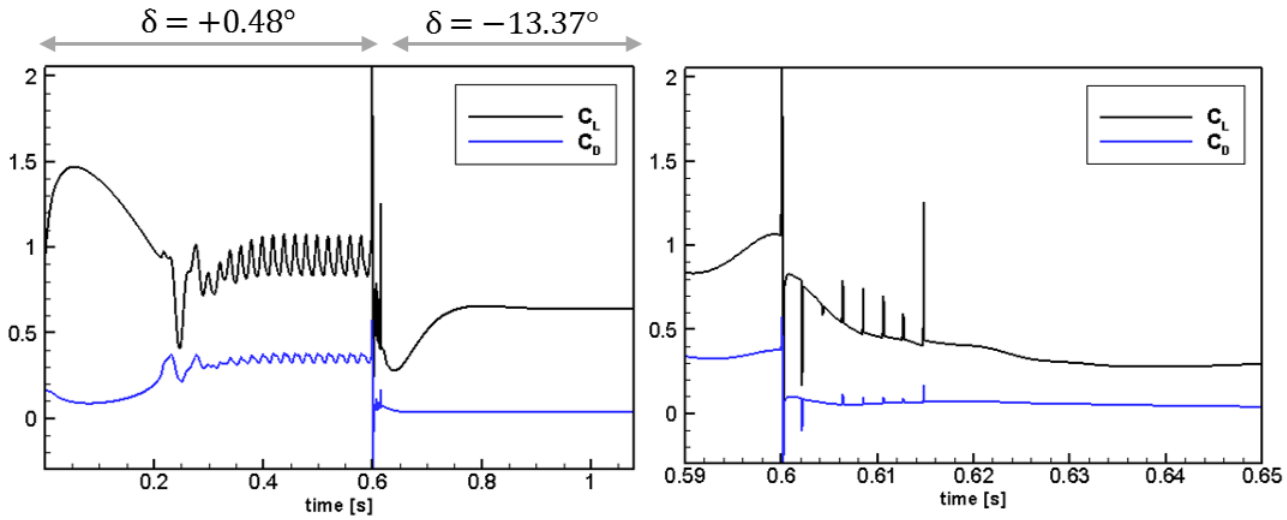


Figure 5.9: $\delta = +0.48°$ morphed to $\delta = -13.37°$ for $\alpha = 18°$ (left), morphing phase (right)

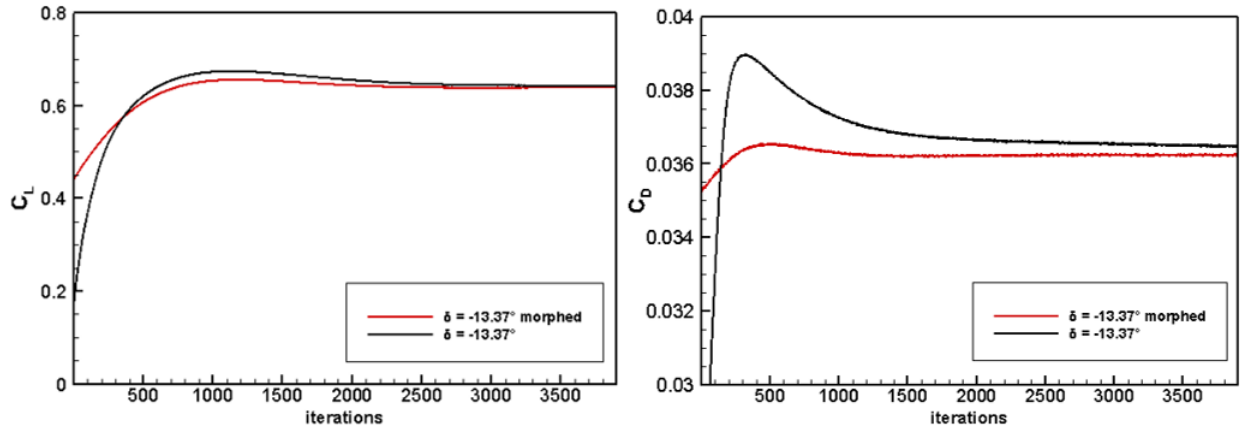


Figure 5.10: Comparison of C_L, C_D values between $\delta = -13.37°$ morphed and $\delta = -13.37°$

Table 5.2: Comparison of C_L and $C_D$values for morphed and static case

variable	morphed $\delta = -13.37°$	static $\delta = -13.37°$	relative difference
C_L	0.63851	0.64045	0.3023%
C_D	0.03625	0.03630	0.1427%

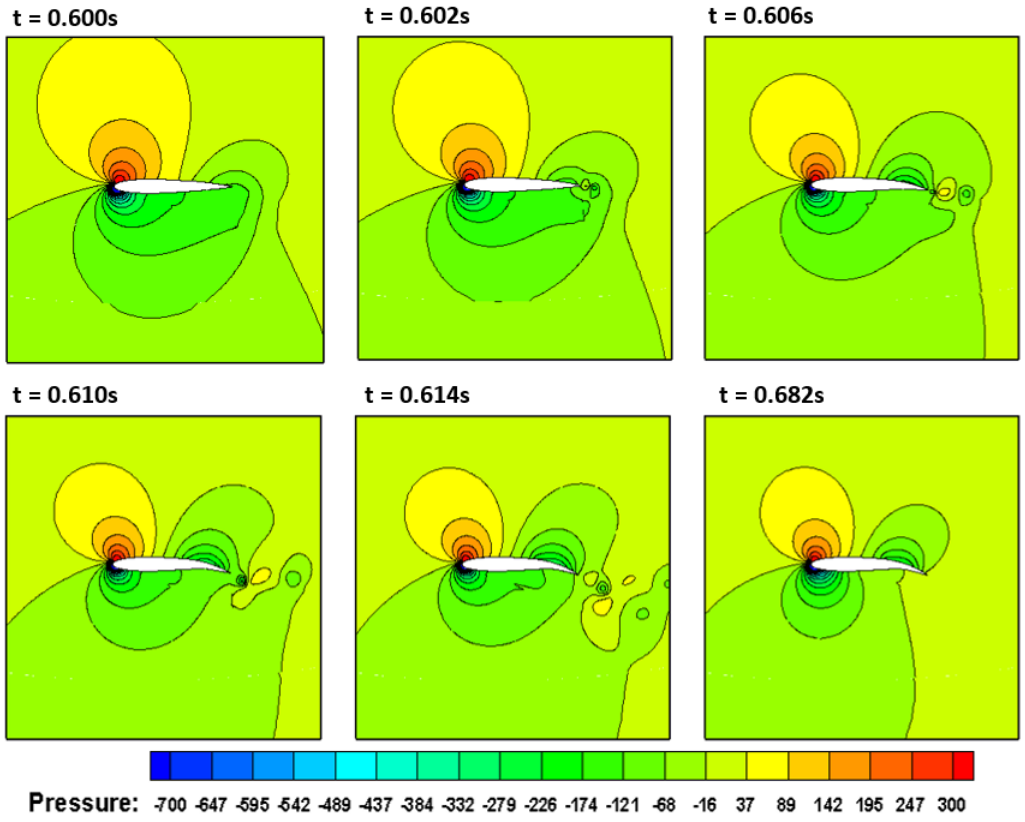


Figure 5.11: Static pressure [Pa] while morphing the blade from $\delta = +0.48°$ to $\delta = -13.37°$

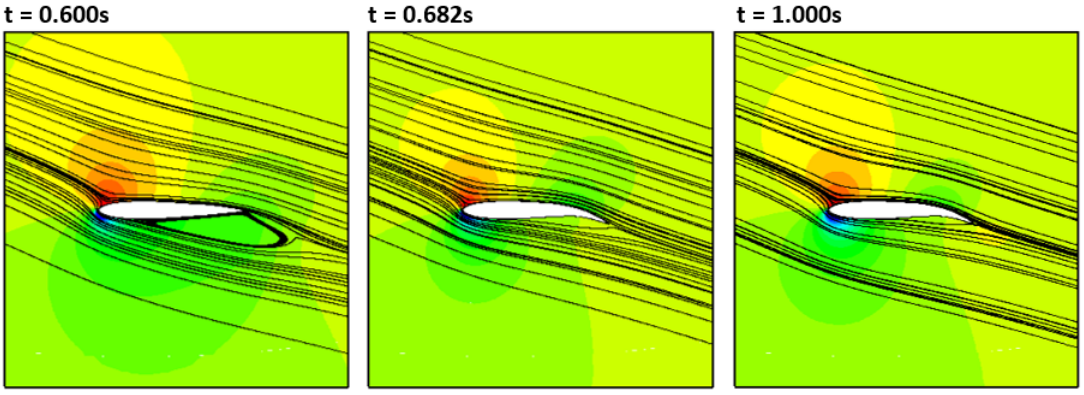


Figure 5.12: Streamlines on the profile before and after morphing

5.4 Challenges of Morphing Case for VAWT

To simulate the morphing blade on the VAWT, both the sliding mesh and dynamic mesh smoothing is utilized. This method is preferred since the dynamic mesh smoothing only has to smooth out the node motion of the morphing flexure motion, the blade rotation is moved as a bulk body motion by giving the angular velocity to the whole rotating zone which is specified through Cell Zone Conditions in Fluent. This chapter shows the results obtained from morphing the blade for VAWT. It is shown that the results obtained from using this method is unphysical, and alternate methods have been considered and discussed in the chapter but no adequate solution has been resolved for the morphing VAWT case. Finally, the issues and challenges encountered are summarized and discussed for future works within and outside of the context of using Fluent.

5.4.1 Implementing Morphing Blade for VAWT

The effect of implementing a morphing aileron on VAWT blades is considered by deforming the blade profiles at specific azimuthal positions as obtained from the fixed profile C_P envelope results in the previous chapter. However, because $\delta = +0.48°$ is only utilized in a small region and morphing the blade takes time to reach the desired profile, it is decided that the blade should be deformed only twice for each cycle. The static morphing case has shown that stall could be delayed by morphing the blade camber; therefore, the upwind half will be utilizing the $\delta = +6.98°$ profile and before it reaches the static stall angle, the blade will be morphed to $\delta = -13.37°$. Table 5.3 shows the specific azimuthal positions with the corresponding blade profile. Figure 5.13 is the C_L loop for the three fixed profile blades as obtained from the results in the previous chapter, the marking shows how the C_L is expected to change as the effective angle of attack changes in Fig.5.13 when the blade shifts to the curve of one profile to another.

Table 5.3: Blade profile morphed at specific azimuthal postions

azimuthal position	profile	morphing interval $\Delta\theta°$ (Δt)
$350° \leq \theta \leq 90°$	$\delta = +6.98°$	$22.68°(0.0420s)$
$90° \leq \theta \leq 350°$	$\delta = -13.37°$	$22.68°(0.0420s)$

Compared to the C_L loop of the baseline profile, the morphing blade is expected to have a larger C_L since morphing the blade allows the blade to shift to a larger range of C_L path as seen in Fig.5.13 when one follows the path of the markings on the curve. However it

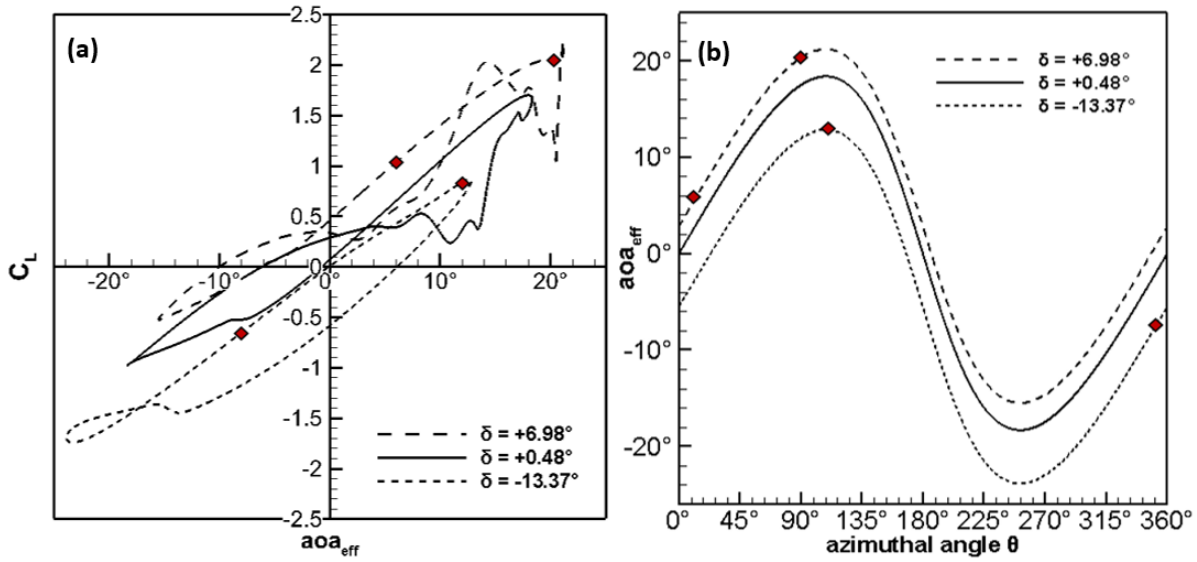


Figure 5.13: (a) C_L vs effective aoa_{eff} (b) aoa_{eff} vs θ from fixed profile results

is unclear what is the effect of changing the blade aoa_{eff} on the flow filed. The blade is morphed twice per cycle, as the results from the fixed profile shows that substantial amount of power is extracted from the $\delta = +6.98°$ profile in the upwind half and $\delta = -13.37°$ in the downwind half. The fixed blade profile of $\delta = +6.98°$ had the highest peak C_P but also stalled at an earlier azimuthal location due to the camber increasing the effective angle of attack. By morphing the trailing edge at the onset of stall close to 90°, the aim is to try to sustain the peak as much as possible before stall occurs. The results from morphing can be seen on Fig.5.14. The point of deformation can be identified in the C_P curve as these are also the regions where small oscillations can be observed. It appears that by morphing the trailing edge, the peak and the dip in C_P both are accentuated. Even though the blade has stopped morphing by the time the blade reaches the azimuth angle of $\theta = 13°$, the C_P generated is significantly larger in the region 45°-90°. The blade morphs again at $\theta = 90°$ but stops morphing by $\theta = 113°$, and again C_P is substantially different in the region that follows, this time with lower C_P at the dip and then a higher peak in the downwind half. However the increase in C_P in the downwind half is cut short as the blade encounters the region of large turbulence due to the vortex from upstream that was caused by morphing the blade at $\theta = 90°$ as in Fig.5.15.

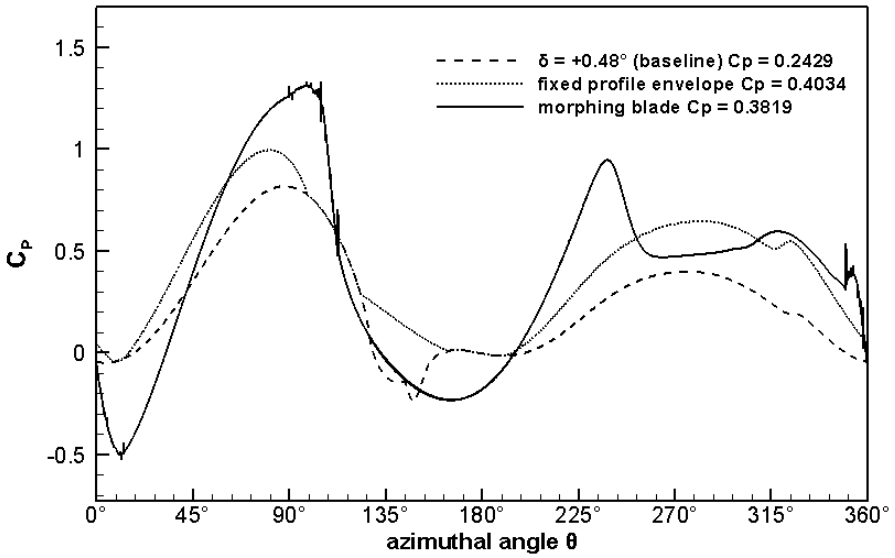


Figure 5.14: C_P comparison for baseline profile, envelope, and morphing profile

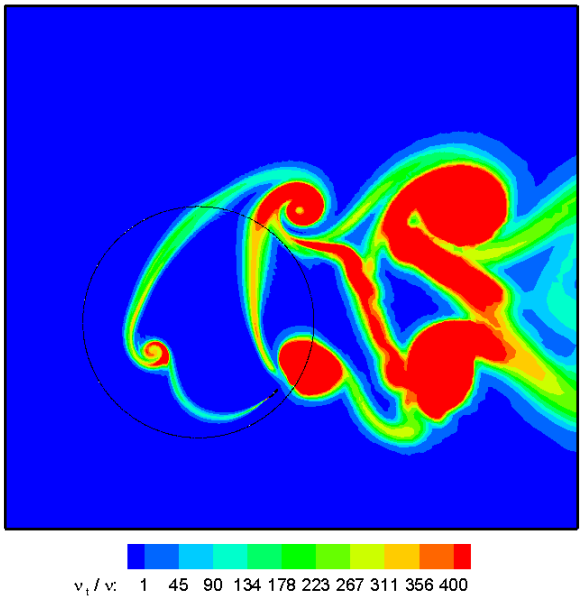


Figure 5.15: Turbulent viscosity ratio for the morphing case on VAWT

In order to check the validity of the solution for the morphing case, the solution from the fixed profile case of $\delta = +0.48°$ was morphed once to the blade profile $\delta = +6.98°$ while running the simulation and kept that profile to run at least four cycles. The time history of coefficient of power for this case is shown in Fig.5.16 The small section with oscillation in the beginning is the point where the morphing occurs and while the dynamic mesh is still enabled in Fluent, the node points are not being changed after the blade has done

morphing to the $\delta = +6.98°$ profile as the flag is turned off to prevent node motion from being assigned. However, after four cycles, Fig.5.17 shows that the C_P result is remains substantially different from the fixed profile case.

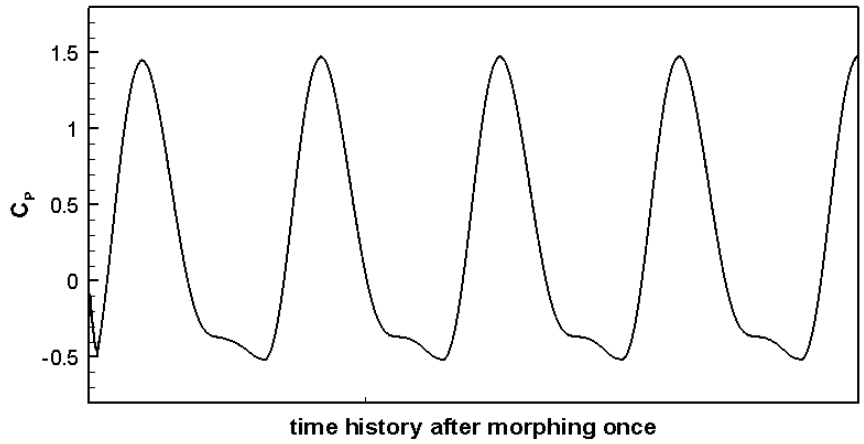


Figure 5.16: Convergence history four cycles after morphing to $\delta = +6.98°$

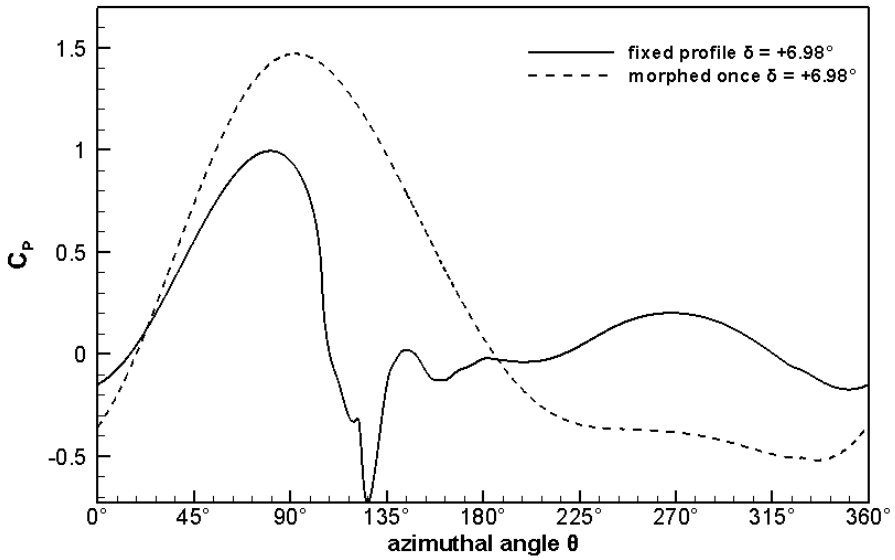


Figure 5.17: C_P comparison between morphed once and fixed profile of $\delta = +6.98°$

Looking at the vorticity contours as shown in Fig.5.18, one can see that there are some problems in the way vorticity, and therefore, the velocity gradients, are calculated within the C-mesh. The problem in the vorticity within the C-mesh is apparently addressed by declaring in Fluent dynamic mesh properties that the fluid in the C-mesh zone is a deforming body. However, in this case the source of error now comes from the interface between the sliding mesh and the rotating zone as shown in Fig.5.19.

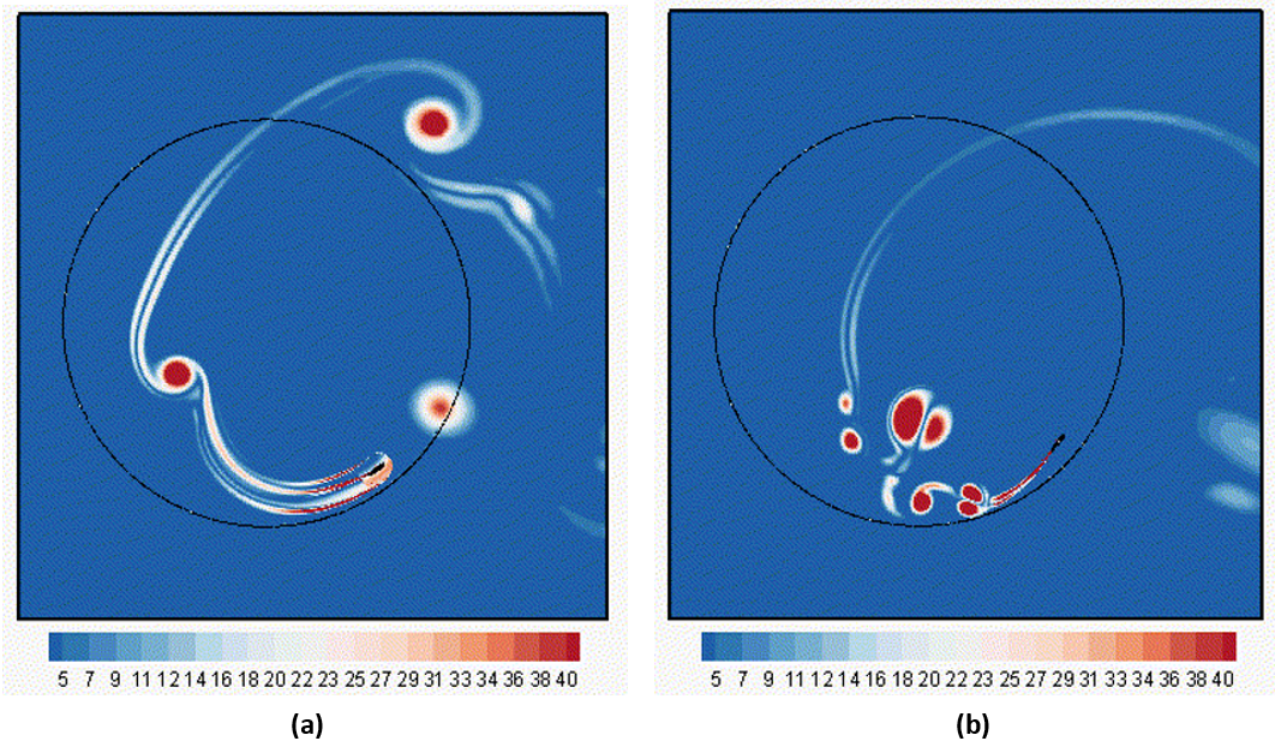


Figure 5.18: Vorticity magnitude contours for (a) morphing case, (b) baseline fixed profile

This error in solution occurs whenever sliding mesh is used at the same time as dynamic mesh. Different methods have been tested to alleviate this apparent bug in the commercial software. For example, the C-mesh zone was merged with the rotating zone and the whole rotating zone was allowed to deform; the problem became much worse, in this case the flow within the rotating zone are not being convected downstream, flow properties simply rotate along the rotating zone. To identify the problem that was causing this, the dynamic mesh is enabled in Fluent while the coordinates are given the same values as what they had, and this still cause the same problem in the vorticity similar to that in Fig.5.18 as long as this was done while sliding mesh is also active. When the sliding mesh was inactive, the dynamic mesh appear to work fine without the errors as is the case for the static morphing trailing edge in the previous chapter.

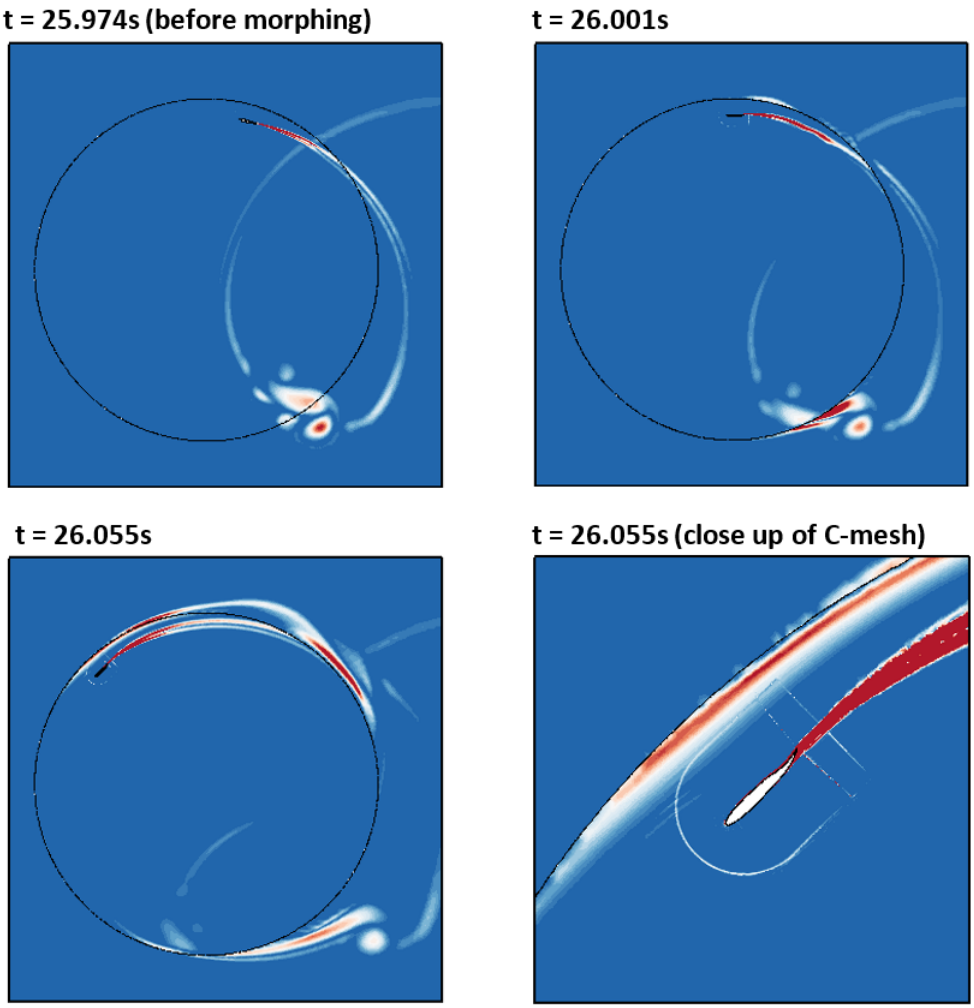


Figure 5.19: Declaring C-mesh fluid zone as "deforming" in Fluent dynamic mesh zones

5.4.2 Alternative Strategies for Morphing Blade on VAWT

The observations from the previous section seem to point to the error mainly being caused by the use of sliding mesh when morphing. The main idea for the alternate methods is to find another way to simulate the VAWT rotation motion without using sliding mesh motion. Two alternate methods within Fluent are considered and tested to see if the problem can be fixed.

Alternate Method I

The idea of this alternate method is to give a rigid body rotation only to the C-mesh zone and keep the other zones stationary. The region outside the C-mesh in Fig.5.20 will undergo mesh deformation, at the same time the internal nodes within the C-mesh will have mesh deformation due to the morphing flexure motion. There are two ways to define a rigid

body motion in Fluent, (i) using the Cell Zone Condition Mesh Motion, (ii), using the rigid body motion Dynamic Mesh Zone type and linking it to a UDF that contains the Fluent function DEFINE_CG_MOTION. One may consider to give a rotational velocity to the C-mesh through the Dynamic Mesh Zones specification instead of the Cell Zone Motion specification to simulate the turbine rotation, then implement the flexure motion and allow the nodes to be smoothed out. The difference between the Dynamic Mesh Zone and Cell Zone Motion rigid body specification is that for the former, dynamic mesh smoothing is performed on the cells adjacent to the rigid body motion, while for the latter specification it is used along with sliding mesh and therefore does not smooth out the nodes adjacent to the rigid body motion.

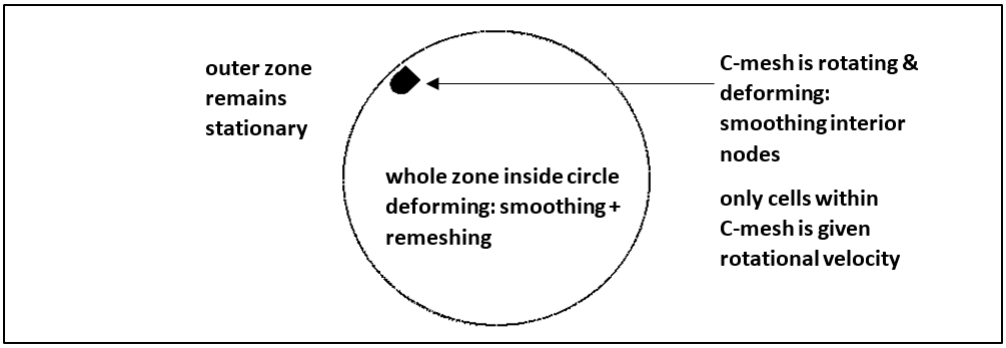


Figure 5.20: Diagram of the alternate method I for VAWT morphing case

For this method, in order to provide the VAWT motion, the DEFINE_CG_MOTION has to be used to provide the VAWT angular velocity; however, this requires that the C-mesh be declared as a Rigid Body within Fluent. This is in contradiction with the flexure motion that requires the C-mesh to be a deforming zone. The ideal solution to this problem is to have two phases of mesh motion within one time step: first specify that C-mesh be rigid body during the VAWT rotation phase, and then to allow it to be deformed and smoothed only during the flexure motion phase; however this option is not available in the commercial software. Another issue that this method could face is a problem during mesh deformation, In this case when the Dynamic Mesh Zone specification is used for the rotation motion, the node smoothing would have to occur within the C-mesh and outside the C-mesh. Fluent has to take care of two different zones, the zone outside of the C-mesh that needs mesh smoothing due to VAWT rotation, and zone inside the C-mesh due to the flexure motion. Although two different motions require different parameter for mesh smoothing, unlike remeshing parameters, there is only one global parameter that affects both zones for the diffusion-based smoothing.

Alternate Method II

Another method is to specify both the VAWT rotation motion and the morphing flexure on the airfoil surface and allow the rest of the mesh to be smoothed out by the dynamic mesh motion as in Fig.5.21. For this method, each node on the blade surface is given both the VAWT rotation motion and the flexure motion by linking a UDF through DEFINE_GRID_MOTION. Each node in the airfoil is rotated first by transforming the coordinates by $\Delta\theta$, and is then given the flexure motion on this new azimuth position, but all of these subroutines are contained and executed within the DEFINE_GRID_MOTION.

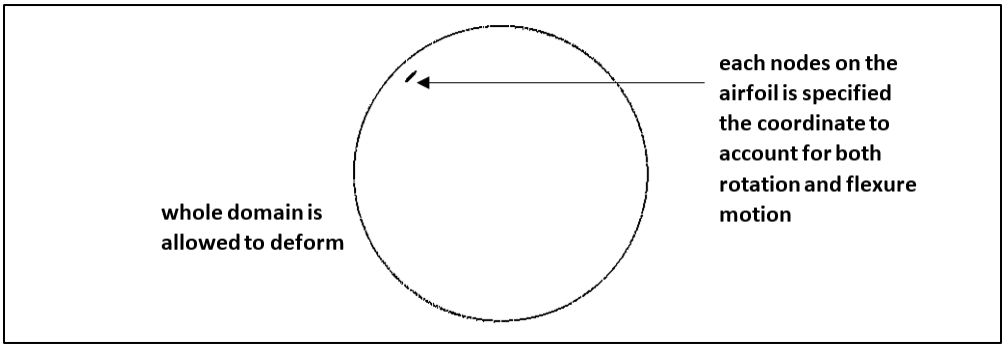


Figure 5.21: Diagram of the alternate method II for VAWT morphing case

The disadvantage in this method is that dynamic mesh smoothing is done after DEFINE_GRID_MOTION is called, therefore instead of having to only smooth out the node motion from the flexure alone, it now also has to smooth out the node motion from $\Delta\theta$ at the same time as the flexure motion; there is no rigid body motion phase in this method. Another major disadvantage in this method is that the cells adjacent to the airfoil wall have the smallest cell size in the whole grid, the cells adjacent to the wall therefore have to be able to accommodate the rotational increment at the same time as the flexure motion. For this method to even work, the time step size has to be reduced by at least 10 times so that $\Delta\theta$ can be reduced by at least 10 times. To test this method, the first step is to test the dynamic mesh motion with just the VAWT rotation without the morphing flexure motion. Because this method requires the whole domain to be remeshed, it is important to check if the solution is compromised. The idea is to run the VAWT simulation with dynamic mesh instead of using a sliding mesh and the results of this simulation should match the results when sliding mesh was used. However, this is not even possible to simulate since negative cell volumes are created right after the first time step. The local cell remeshing method on Fluent only applies to triangular and tetrahedral cell types [27]. This method could work if the cells adjacent to the airfoil walls are triangular cells, but this is not ideal and would have significantly changed the solution.

5.4.3 Recommendations Outside of Fluent

The implementation of morphing blade on VAWT shows the limitation of the commercial solver. The limitation presented by the commercial solver is two-fold, (i) inability to simulate sliding mesh simultaneously with deforming mesh, and (ii) the need for a better preservation of grid orthogonality and skewness during mesh deformation. The former can be addressed by using another solver that allows for the sliding mesh to be compatible with mesh deformation or start with a new grid and allow the whold domain to rotate instead of using a sliding mesh. For the latter issue, it can be said that the mesh deformation methods available in Fluent 14.5.7 are very basic compared to the methods mentioned in the literature review chapter.

It could be argued that the mesh deformation methods in Fluent are not meant to be used for visous flow calculations where high aspect ratio cells are predominant in the region near the wall. In order to prevent negative volumes, the node distribution towards the end of the trailing edge have to be prescribed so that the cells in this region have aspect ratio of close to 1 or negative cell volumes are generated. Moreover, the mesh quality degrades quickly as more subframes are added between the 21 profiles. This is because the deformation quality deteriorates as the number of incremental dispalcement increases. In fact, the whole actuation range of going from $\delta = -13.37°$ to $\delta = +6.98°$ cannot be achieved without having to retrieve the initial mesh of $\delta = +0.48°$ to restore the mesh quality everytime this profile is traversed during the actuation range. Having only the spring-based and diffusion-based deformation methods available, the parameters that control the mesh quality are either the stiffness factor or the diffusion coefficient. The literature review chapter presents augmentations to these methods that may improve cell quality during deformation like modyifying the diffusion coefficient to be raised to an exponent or the addition of quaternions to the spring-based method; however, none of these are available in the commercial software. Although the morphing case was shown to give good agreement with the static solution, the quality of the mesh near the wall, especially orthogonality and skewness leaves much room for improvement.

Outside the context of using the commercial solver Fluent, the most robust mesh deformation method would be either the linear elasticity method or the RBF interpolation method. The linear elasticity method could be used for 2D cases where computational costs are manageable. For 3D cases with large node counts, the RBF with augmented algorithms to reduce computational cost is ideal. One could argue to keep the spring-based or diffusion-based method and remesh when cell is invalidated; however, this would make

it computationally impractical especially for the case of VAWT where small time step and multiple cycles are required.

Chapter 6

Conclusion

The main purpose of the study is to simulate the flow past a morphing blade on VAWT. In line with this objective, the study investigates the lift and drag generated by the three blade profiles from the actuation range of the SSMA as well as their critical stall angle of attack. These results give a good insight as to how the blade profiles will behave for the VAWT case. Within the expected results, the $\delta = +6.98°$ produced the highest peak C_P in the upwind half and the $\delta = -13.37°$ generated the highest peak C_P in the downwind half. Taking the envelope of the three profiles gave an upper bound of 66% possible increase in average C_P per cycle. With this in mind, the SSMA morphing blade could be used to give the blade varying camber as a function of the azimuthal position with the purpose of increasing the C_P.

The morphing blade is first simulated for an unsteady static case to investigate the capability of simulating the effect of morphing on the C_L and C_D generated. It was shown that by morphing the blade from the baseline $\delta = +0.48°$ to $\delta = -13.37°$, the C_L and C_D were able to be decreased to the expected static values of $\delta = -13.37°$. This shows that there is a potential for the dynamic stall to be avoided and therefore mitigate the effect of vortex-blade interaction.

The implementation of the morphing blade on VAWT however, has shown to produce unphysical solutions with the method used in the study due to the apparent incompatibility due to the concurrent usage of sliding mesh along with dynamic mesh motion; further investigation is necessary to exactly locate the source of the error. With the literature review on mesh deformation methods in mind, the deforming mesh methods in Fluent is not robust especially within the context of solving the Navier-Stokes equations. Fluent only has the linear spring analogy and Laplace method for mesh deformation while there has

already been numerous studies on more robust and computationally efficient methods that are able to better preserve the quality of the mesh especially where high aspect ratio cells are concerned. This study concludes by highlighting the contribution of showing that blade camber has an important imact on the C_P curve of VAWT and while the implementation of morphing blade on VAWT in Fluent did not give acceptable results, the morphing blade was successfully implemented for static airfoil case and showed potential for future studies on its application for VAWT.

Bibliography

[1] I. H. Abbott and A. E. Von Doenhoff. *Theory of Wing Sections, Including a Summary of Airfoil Data, by Ira H. Abbott and Albert E. Von Doenhoff*. Dover Publications, 1959.

[2] E. Amet, T. Maitre, C. Pellone, and J.-L. Achard. 2d numerical simulations of blade-vortex interaction in a darrieus turbine. *Journal of fluids engineering*, 131(11), 2009.

[3] J. Anderson. *Fundamentals of Aerodynamics*. McGraw-Hill, Boston, 2001.

[4] F. Balduzzi, A. Bianchini, R. Maleci, G. Ferrara, and L. Ferrari. Critical issues in the cfd simulation of darrieus wind turbines. *Renewable Energy*, 85:419–435, 2016.

[5] J. T. Batina. Unsteady euler airfoil solutions using unstructured dynamic meshes. *AIAA journal*, 28(8):1381–1388, 1990.

[6] H. Beri and Y. Yao. Effect of camber airfoil on self starting of vertical axis wind turbine. *J. Environ. Sci. Technol*, 4(3):302–312, 2011.

[7] F. J. Blom. Considerations on the spring analogy. *International journal for numerical methods in fluids*, 32(6):647–668, 2000.

[8] F. M. Bos. Numerical simulations of flapping foil and wing aerodynamics: Mesh deformation using radial basis functions. 2010.

[9] C. L. Bottasso, D. Detomi, and R. Serra. The ball-vertex method: a new simple spring analogy method for unstructured dynamic meshes. *Computer Methods in Applied Mechanics and Engineering*, 194(39):4244–4264, 2005.

[10] M. Breuer, N. Jovicic, and K. Mazaev. Comparison of des, rans and les for the separated flow around a flat plate at high incidence. *International journal for numerical methods in fluids*, 41(4):357–388, 2003.

[11] G. Brochier, P. Fraunie, C. Beguiert, and I. Paraschivoiu. Water channel experiments of dynamic stall on darrieus wind turbine blades. 1986.

[12] C. Burg. Analytic study of 2d and 3d grid motion using modified laplacian. *International journal for numerical methods in fluids*, 52(2):163–197, 2006.

[13] C. Byun and G. P. Guruswamy. A parallel, multi-block, moving grid method for aeroelastic applications on full aircraft. *AIAA paper*, 98:4782, 1998.

[14] D. Castelein, D. Ragni, G. Tescione, C. S. Ferreira, and M. Gaunaa. Creating a benchmark of vertical axis wind turbines in dynamic stall for validating numerical models. In *Conference creating a benchmark of vertical axis wind turbines in dynamic stall for validating numerical models. American Institute of Aeronautics & Astronautics*, 2015.

[15] C.-C. Chen and C.-H. Kuo. Effects of pitch angle and blade camber on flow characteristics and performance of small-size darrieus vawt. *Journal of Visualization*, 16(1):65–74, 2013.

[16] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234, 1967.

[17] P. Crumpton and M. Giles. *Implicit time accurate solutions on unstructured dynamic grids*. Oxford University Computing Laboratory, Numerical Analysis Group, 1995.

[18] R. M. Cummings, J. R. Forsythe, S. A. Morton, and K. D. Squires. Computational challenges in high angle of attack flow prediction. *Progress in Aerospace Sciences*, 39(5):369–384, 2003.

[19] L. A. Danao, N. Qin, and R. Howell. A numerical study of blade thickness and camber effects on vertical axis wind turbines. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, 226(7):867–881, 2012.

[20] A. de Boer, M. van der Schoot, and H. Bijl. Mesh deformation based on radial basis function interpolation. *Computers & Structures*, 85(11):784–795, 2007.

[21] L. Dubuc, F. Cantariti, M. Woodgate, B. Gribben, K. Badcock, and B. Richards. A grid deformation technique for unsteady flow computations. *International Journal for Numerical Methods in Fluids*, 32(3):285–311, 2000.

[22] D. C. Eleni, T. I. Athanasios, and M. P. Dionissios. Evaluation of the turbulence models for the simulation of the flow over a national advisory committee for aeronautics (naca) 0012 airfoil. *Journal of Mechanical Engineering Research*, 4(3):100–111, 2012.

[23] S. Eriksson, H. Bernhoff, and M. Leijon. Evaluation of different turbine concepts for wind power. *Renewable and Sustainable Energy Reviews*, 12:1419–1434, 2008.

[24] S. Etienne, A. Garon, and D. Pelletier. Geometric conservation law and finite element methods for 3d unsteady simulations of incompressible flow. In *39th AIAA Fluid Dynamics Conference*, page 3571, 2009.

[25] C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer methods in applied mechanics and engineering*, 163(1-4):231–245, 1998.

[26] C. S. Ferreira, G. Van Kuik, G. Van Bussel, and F. Scarano. Visualization by piv of dynamic stall on a vertical axis wind turbine. *Experiments in Fluids*, 46(1):97–108, 2009.

[27] A. Fluent. 12.0 user's guide, 2009. *Fluent Inc., New Hampshire*.

[28] B. Francois, M. Costes, and G. Dufour. Comparison of chimera and sliding mesh techniques for unsteady simulations of counter rotating open-rotors. In *20th International Society of Air Breathing Engines Conference*, 2011.

[29] Y. Gan and G. Zha. Comparison of drag prediction using rans models and ddes for the dlr-f6 configuration using high order schemes. In *54th Aerospace Sciences Meeting, San Diego, CA, AIAA Paper*, volume 553, page 2016, 2016.

[30] P. Gnoffo. A finite-volume, adaptive grid algorithm applied to planetary entry flowfields. *AIAA Journal(ISSN 0001-1452)*, 21:1249–1254, 1983.

[31] R. Gosselin, G. Dumas, and M. Boudreau. Parametric study of h-darrieus vertical-axis turbines using urans simulations. *Paper CFDSC-2013*, 178:6–9, 2013.

[32] S. Hsu, C. Chang, and J. Samareh. A simplified mesh deformation method using commercial structural analysis software. In *Proceedings of the 10th AIAA/ISSMO multidisciplinary analysis and optimization conference. Albany, New York*, 2004.

[33] R. D. Ionescu, I. Szava, S. Vlase, M. Ivanoiu, and R. Munteanu. Innovative solution of vertical axis wind turbine, suitable for naval industry implementation (numerical methods and analythical calculus). *Procedia Technology*, 19:715–721, 2015.

[34] S. Jakobsson and O. Amoignon. Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization. *Computers & Fluids*, 36(6):1119–1136, 2007.

[35] S. L. Karman. Unstructured viscous layer insertion using linear-elastic smoothing. *AIAA journal*, 45(1):168, 2007.

[36] B. Koobus and C. Farhat. Second-order time-accurate and geometrically conservative implicit schemes for flow computations on unstructured dynamic meshes. *Computer Methods in Applied Mechanics and Engineering*, 170(1-2):103–129, 1999.

[37] R. B. Langtry and F. R. Menter. Correlation-based transition modeling for unstructured parallelized computational fluid dynamics codes. *AIAA journal*, 47(12):2894–2906, 2009.

[38] R. Lanzafame, S. Mauro, and M. Messina. 2d cfd modeling of h-darrieus wind turbines using a transition turbulence model. *Energy Procedia*, 45:131–140, 2014.

[39] B. E. Launder and D. B. Spalding. The numerical computation of turbulent flows. *Computer methods in applied mechanics and engineering*, 3(2):269–289, 1974.

[40] R. Löhner, C. Yang, and E. Onate. Viscous free surface hydrodynamics using unstructured grids. In *Proc. 22nd Symp. Naval Hydro*, pages 476–490, 1998.

[41] J. Leishman. Challenges in modelling the unsteady aerodynamics of wind turbines. *Wind Energy*, 5:85–132, 2002.

[42] X. Liu, N. Qin, and H. Xia. Fast dynamic grid deformation based on delaunay graph mapping. *Journal of Computational Physics*, 211(2):405–423, 2006.

[43] G. A. Markou, Z. S. Mouroutis, D. C. Charmpis, and M. Papadrakakis. The ortho-semi-torsional (ost) spring analogy method for 3d mesh moving boundary problems. *Computer Methods in Applied Mechanics and Engineering*, 196(4):747–765, 2007.

[44] D. Maruyama, D. Bailly, and G. Carrier. High-quality mesh deformation using quaternions for orthogonality preservation. *AIAA journal*, 2014.

[45] D. J. Mavriplis and Z. Yang. Construction of the discrete geometric conservation law for high-order time-accurate simulations on dynamic meshes. *Journal of Computational Physics*, 213(2):557–573, 2006.

[46] W. J. McCroskey. The phenomenon of dynamic stall. Technical report, National Aeronautics and Space Administration Moffett Field, CA Ames Research Center, 1981.

[47] F. R. Menter. Improved two-equation k-omega turbulence models for aerodynamic flows. 1992.

[48] F. R. Menter. Performance of popular turbulence model for attached and separated adverse pressure gradient flows. *AIAA journal*, 30(8):2066–2072, 1992.

[49] K. Nakahashi and G. Deiwert. Self-adaptive-grid method with application to airfoil flow. *AIAA journal*, 25(4):513–520, 1987.

[50] W. L. Oberkampf, M. Sindir, and A. Conlisk. Guide for the verification and validation of computational fluid dynamics simulations. *Am. Institute of Aeronautics and Astronautics*, 1998.

[51] B. Paillard, J. A. Astolfi, and F. Hauville. Uranse simulation of an active variable-pitch cross-flow darrieus tidal turbine: Sinusoidal pitch function investigation. *International Journal of Marine Energy*, 11:9–26, 2015.

[52] A. Pankonien, K. Duraisamy, C. Faria, and D. Inman. Synergistic smart morphing aileron: aerostructural performance analysis. *AIAA Adaptive Structures Conference*, January 2014.

[53] I. Paraschivoiu, O. Trifu, and F. Saeed. H-darrieus wind turbine with blade pitch control. *International Journal of Rotating Machinery*, 2009, 2009.

[54] S. Patankar. *Numerical heat transfer and fluid flow*. CRC press, 1980.

[55] R. H. Pletcher, J. C. Tannehill, and D. Anderson. *Computational fluid mechanics and heat transfer*. CRC Press, 2012.

[56] T. Rendall and C. Allen. Improved radial basis function fluid–structure coupling via efficient localized implementation. *International journal for numerical methods in engineering*, 78(10):1188–1208, 2009.

[57] T. C. Rendall and C. B. Allen. Efficient mesh motion using radial basis functions with data reduction algorithms. *Journal of Computational Physics*, 228(17):6231–6249, 2009.

[58] O. Reynolds. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Proceedings of the Royal Society of London*, 56(336-339):40–45, 1894.

[59] A. Rezaeiha, I. Kalkman, and B. Blocken. Effect of pitch angle on power performance and aerodynamics of a vertical axis wind turbine. *Applied Energy*, 197:132–150, 2017.

[60] L. F. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 210:307–357, 1911.

[61] P. J. Roache. Perspective: a method for uniform reporting of grid refinement studies. *Transactions-American Society of Mechanical Engineers Journal of Fluids Engineering*, 116:405–405, 1994.

[62] J. A. Samareh. Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization. *AIAA journal*, 39(5):877–884, 2001.

[63] J. A. Samareh. Application of quaternions for mesh deformation. 2002.

[64] F. Scheurich. *Modelling the aerodynamics of vertical-axis wind turbines PhD thesis*. PhD thesis, University of Glasgow, 2011.

[65] H. Schlichting, K. Gersten, E. Krause, and H. Oertel. *Boundary-layer theory*, volume 7. Springer, 1955.

[66] M. Selim and R. Koomullil. Mesh deformation approaches–a survey. *J Phys Math*, 7(181):2090–0902, 2016.

[67] E. Sesto and C. Casale. Exploitation of wind energy source to meet the world's electricity. *Journal of Wind Engineering and Industrial Aerodynamics*, 74-76:375–387, 1998.

[68] P. R. Spalart, S. R. Allmaras, et al. A one equation turbulence model for aerodynamic flows. *AIAA Paper*, pages 92–0439, 1994.

[69] P. R. Spalart and C. L. Rumsey. Effective inflow conditions for turbulence models in aerodynamic calculations. *AIAA journal*, 45(10):2544, 2007.

[70] R. Steijl and G. Barakos. Sliding mesh algorithm for cfd analysis of helicopter rotor–fuselage aerodynamics. *International Journal for Numerical Methods in Fluids*, 58(5):527–549, 2008.

[71] P. Thomas and C. Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA J*, 17(10):1030–1037, 1979.

[72] J. F. Thompson, B. K. Soni, and N. P. Weatherill. *Handbook of grid generation*. CRC press, 1998.

[73] W. Timmer. Aerodynamic characteristics of wind turbine blade airfoils at high angles-of-attack. In *3rd EWEA Conference-Torque 2010: The Science of making Torque from Wind, Heraklion, Crete, Greece, 28-30 June 2010*. European Wind Energy Association, 2010.

[74] W. Tjiu, T. Marnoto, S. Mat, M. H. Ruslan, and K. Sopian. Darrieus vertical axis wind turbine for power generation i: Assessment of darrieus vawt configurations. *Renewable Energy*, 75:50–67, 2015.

[75] W. Tjiu, T. Marnoto, S. Mat, M. H. Ruslan, and K. Sopian. Darrieus vertical axis wind turbine for power generation ii: Challenges in hawt and the opportunity of multi-megawatt darrieus vawt development. *Renewable Energy*, 75:560–571, 2015.

[76] P. Vittecoq. Dynamic stall: the case of the vertical axis wind turbine. *Journal of Solar Energy Engineering*, 108:141, 1986.

[77] R. Walters, J. Fanucci, P. Hill, and P. Migliore. Vertical axis wind turbine development: Executive summary. *Final Report, 1 Mar. 1976-30 Jun. 1977 West Virginia Univ., Morgantown. Dept. of Aerospace Engineering.*, 1979.

[78] D. C. Wilcox. Comparison of two-equation turbulence models for boundary layers with pressure gradient. *AIAA journal*, 31(8):1414–1421, 1993.

[79] D. C. Wilcox et al. *Turbulence modeling for CFD*, volume 2. DCW industries La Canada, CA, 1998.

[80] T. Wolff, B. Ernst, and J. R. Seume. Aerodynamic behavior of an airfoil with morphing trailing edge for wind turbine applications. In *Journal of Physics: Conference Series*, volume 524, page 012018. IOP Publishing, 2014.

[81] A. Wong, H. Tsai, J. Cai, Y. Zhu, and F. Liu. Unsteady flow calculations with a multiblock moving mesh algorithm. In *AIAA*, 2000.

[82] Z. Yang and D. J. Mavriplis. Unstructured dynamic meshes with higher-order time integration schemes for the unsteady navier-stokes equations. *AIAA paper*, 1222(2005):1, 2005.

[83] Z. Yang and D. J. Mavriplis. Mesh deformation strategy optimized by the adjoint method on unstructured meshes. *AIAA journal*, 45(12):2885, 2007.

[84] S. N. Zadeh, M. Komeili, and M. Paraschivoiu. Mesh convergence study for 2-d straight-blade vertical axis wind turbine simulations and estimation for 3-d simulations. *Transactions of the Canadian Society for Mechanical Engineering,*, 38(4), 2014.

Appendix A

Fourier Approximation of Airfoil

A.1 Fourier approximation for the airfoil surfaces

$$y = a_0 + a_1\cos(x \cdot ww) + b_1\sin(x \cdot ww) + a_2\cos(2x \cdot ww) + b_2\sin(2x \cdot ww)$$
$$+a_3\cos(3x \cdot ww) + b_3\sin(3x \cdot ww) + a_4\cos(4x \cdot ww) + b_4\sin(4x \cdot ww)$$
$$+a_5\cos(5x \cdot ww) + b_5\sin(5x \cdot ww) + a_6\cos(6x \cdot ww) + b_6\sin(6x \cdot ww)$$

UPPER	a0	a1	b1	a2	b2	a3	b3	a4	b4	a5	b5	a6	b6	ww
FR1	11.06	4.702	-0.605	0.0963	-0.054	-0.203	-0.115	-0.135	-0.164	-0.014	-0.022	0.0523	0.0334	0.039
FR2	11.12	4.456	-1.511	0.0496	0.0002	-0.201	0.0101	-0.178	-0.025	-0.005	-0.023	0.033	-0.045	0.038
FR3	11.21	4.275	-1.72	0.042	-0.014	-0.179	0.0462	-0.171	0.0082	-0.008	-0.026	0.0124	-0.052	0.0377
FR4	11.32	3.62	-2.548	0.1035	0.0505	-0.077	0.0846	-0.094	0.0894	-0.056	-0.02	-0.046	0.0011	0.0366
FR5	11.38	3.407	-2.339	0.1816	-0.074	-0.008	0.0578	-0.089	0.0912	-0.077	-0.005	-0.063	0.0181	0.0364
FR6	11.5	2.48	-3.086	0.2394	-0.176	-0.034	0.0049	-0.01	0.1005	-0.018	0.0857	0.0304	0.0292	0.035
FR7	11.63	1.912	-3.085	0.2002	-0.347	-0.115	0.0082	0.039	0.1156	0.0815	0.0784	0.0305	-0.019	0.034
FR8	11.45	2.149	-2.495	0.3074	-0.589	-0.299	-0.09	0.0008	0.1745	0.1279	0.0939	0.0484	-0.049	0.034
FR9	11.42	1.63	-2.598	-0.15	-0.852	-0.391	0.2391	0.173	0.1378	0.0977	-0.137	-0.066	-0.019	0.0326
FR10	10.11	0.081	-3.501	-1.236	2.508	0.1799	-1.335	0.0647	0.6127	-0.092	-0.185	0.0d0	0.0d0	0.0256
FR11	10.77	2.089	-2.547	-1.543	-0.809	0.29	0.8464	0.2082	-0.412	-0.247	0.061	0.064	0.0258	0.0304
FR12	8.192	4.055	-5.176	-3.428	3.373	1.819	-2.161	-0.758	1.227	0.2164	-0.57	0.0004	0.1317	0.0267
FR13	9.78	3.448	-3.472	-2.373	0.1361	1.071	0.5744	-0.243	-0.477	-0.053	0.2341	0.038	-0.055	0.0296
FR14	7.753	5.979	-4.307	-3.697	0.4255	1.928	0.7825	-0.637	-0.767	0.1024	0.458	0.0169	-0.126	0.0289
FR15	5.245	7.958	-7.368	-4.491	2.827	2.727	-0.824	-1.343	0.073	0.5459	0.1162	-0.127	-0.043	0.0278
FR16	-4.267	1.334	-24.64	9.477	8.888	-7.052	0.448	1.806	-2.501	0.2929	1.009	-0.184	-0.051	0.0232
FR17	-5.523	-6.861	-24.45	10.55	2.185	-3.79	3.208	-0.068	-1.919	0.5188	0.3346	-0.075	0.0558	0.0223
FR18	-1.331	-3.724	-20.06	6.972	2.768	-2.47	1.071	0.2617	-0.583	0.0d0	0.0d0	0.0d0	0.0d0	0.0242
FR19	-3.137	-2.229	-22.38	7.032	4.29	-2.938	0.496	0.4936	-0.563	0.0d0	0.0d0	0.0d0	0.0d0	0.0245
FR20	2.154	-4.198	-19.15	9.261	4.185	-5.751	1.02	2.168	-1.786	-0.366	1.043	0.0053	-0.264	0.0262
FR21	157.2	-46.89	279.6	-210.6	-41.76	29.98	-119.7	50.61	15.63	-5.49	14.55	-2.076	-1.077	0.0209

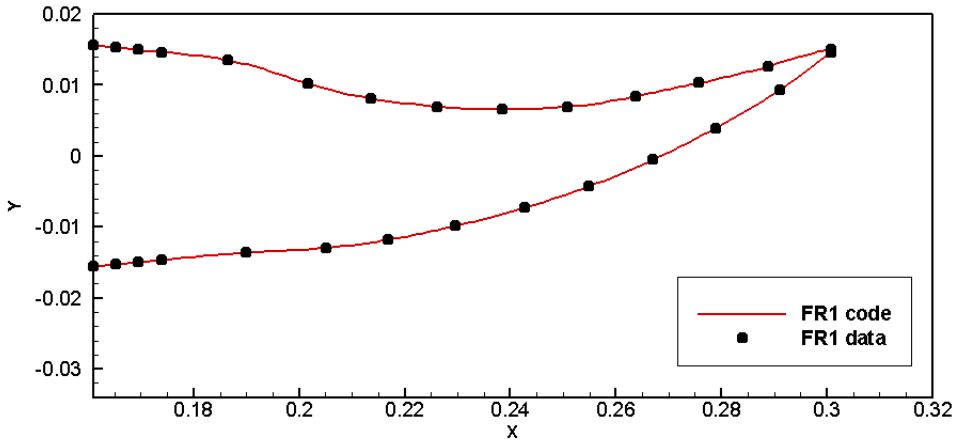
Figure A.1: Upper Fourier coefficients for the 21 frames of actuation range

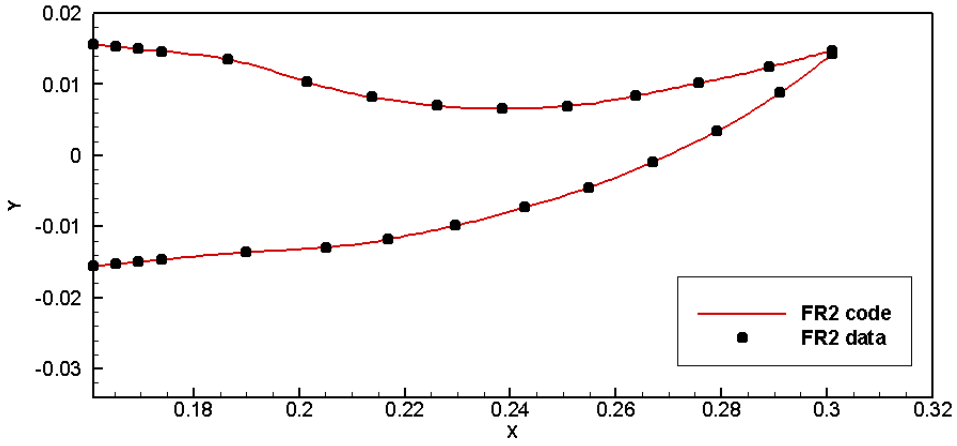
LOWER	a0	a1	b1	a2	b2	a3	b3	a4	b4	a5	b5	a6	b6	ww
FR1	-4.249	-12.74	4.657	6.147	0.0105	-2.498	-1.833	0.4049	1.256	0.0963	-0.521	-0.111	0.0984	0.0307
FR2	-4.342	-12.49	4.863	6.134	-0.187	-2.654	-1.654	0.5231	1.183	0.0429	-0.515	-0.098	0.1111	0.0305
FR3	-4.544	-12.47	4.051	5.996	0.532	-2.284	-2.085	0.2265	1.289	0.1888	-0.494	-0.134	0.0751	0.0308
FR4	-4.608	-12.16	4.708	5.878	-0.155	-2.439	-1.675	0.4132	1.163	0.0804	-0.497	-0.106	0.0945	0.0306
FR5	-4.751	-11.81	5.023	5.793	-0.475	-2.549	-1.417	0.5509	1.086	-0.004	-0.489	-0.079	0.109	0.0305
FR6	-4.988	-11.51	4.782	5.633	-0.264	-2.443	-1.51	0.4783	1.09	0.0348	-0.479	-0.089	0.1017	0.0305
FR7	-5.388	-11.42	3.326	5.287	0.9608	-1.685	-2.177	-0.091	1.158	0.3068	-0.374	-0.145	0.0188	0.0311
FR8	-5.8	-10.95	2.734	4.802	1.388	-1.165	-2.303	-0.364	1.061	0.3915	-0.256	-0.145	-0.034	0.0314
FR9	-5.962	-10.61	2.951	4.761	1.197	-1.313	-2.185	-0.274	1.064	0.3568	-0.297	-0.146	-0.015	0.0312
FR10	-6.541	-10.03	1.67	4.023	2.118	-0.398	-2.338	-0.748	0.7697	0.4639	-0.035	-0.11	-0.115	0.0318
FR11	-6.763	-9.364	3.239	4.397	0.7005	-1.425	-1.806	-0.074	0.9811	0.2546	-0.326	-0.125	0.0221	0.0311
FR12	-7.22	-8.487	3.835	4.105	-0.033	-1.591	-1.342	0.1683	0.8551	0.1167	-0.347	-0.089	0.052	0.0307
FR13	-7.921	-7.768	2.811	3.613	0.767	-1.029	-1.62	-0.203	0.7917	0.2692	-0.236	-0.11	-0.009	0.0312
FR14	-8.962	-6.336	2.373	2.854	0.9047	-0.627	-1.493	-0.32	0.6059	0.2771	-0.128	-0.092	-0.045	0.0314
FR15	-10.38	-4.518	1.541	1.636	1.259	0.1651	-1.209	-0.494	0.1701	0.2159	0.1188	-0.009	-0.101	0.0322
FR16	-12.22	-2.214	1.156	0.3001	0.919	0.5981	-0.407	-0.171	-0.237	-0.063	0.1356	0.0866	0.0117	0.0335
FR17	-13.69	-1.415	1.365	0.114	0.0873	0.0664	0.2151	0.0188	0.0254	-0.002	-0.01	0.0017	-0.043	0.0369
FR18	-15.62	2.35	3.223	-0.087	-1.01	-0.425	-0.094	0.116	0.2058	0.1143	-0.07	0.0041	-0.061	0.033
FR19	-17.15	5.102	2.967	-1.017	-1.439	-0.554	0.5606	0.4117	0.0381	-0.048	-0.196	-0.081	0.003	0.0318
FR20	-18.41	7.301	1.647	-2.074	-0.988	0.0009	0.9895	0.3277	-0.378	-0.198	-0.044	-0.018	0.073	0.0308
FR21	-19.45	8.746	2.032	-2.478	-1.541	-0.15	1.365	0.5221	-0.417	-0.274	-0.112	-0.029	0.103	0.031

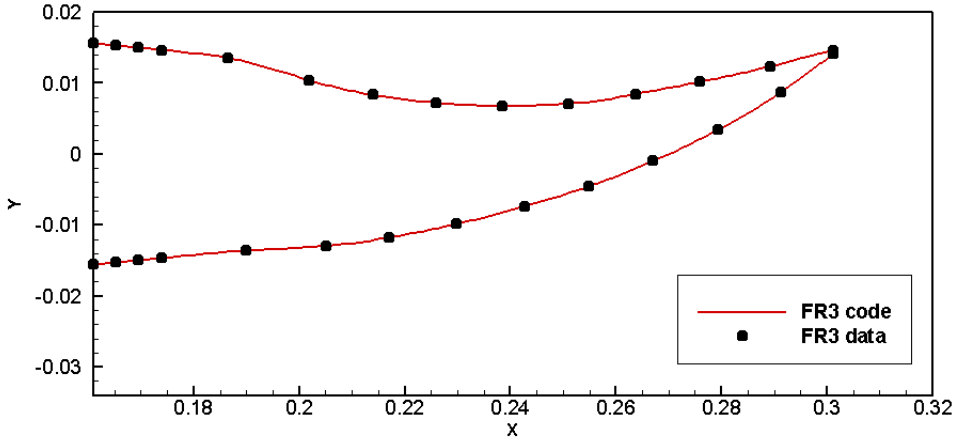
Figure A.2: Lower Fourier coefficients for the 21 frames of actuation range

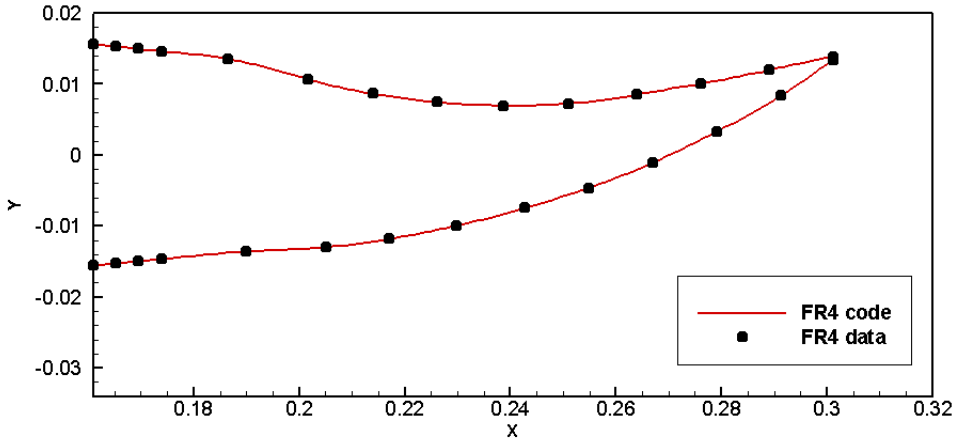
A.2 21 Frames of SSMA actuation range

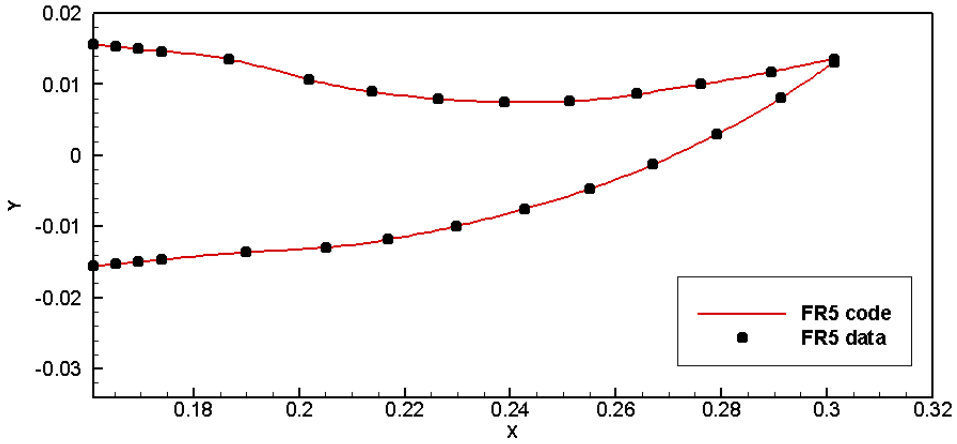
The data points for 21 actuation frames for the whole SSMA actuation range are from experimental data from Pankonien, et al.[52]. The surfaces are created using Fourier series approximation based on these data points.

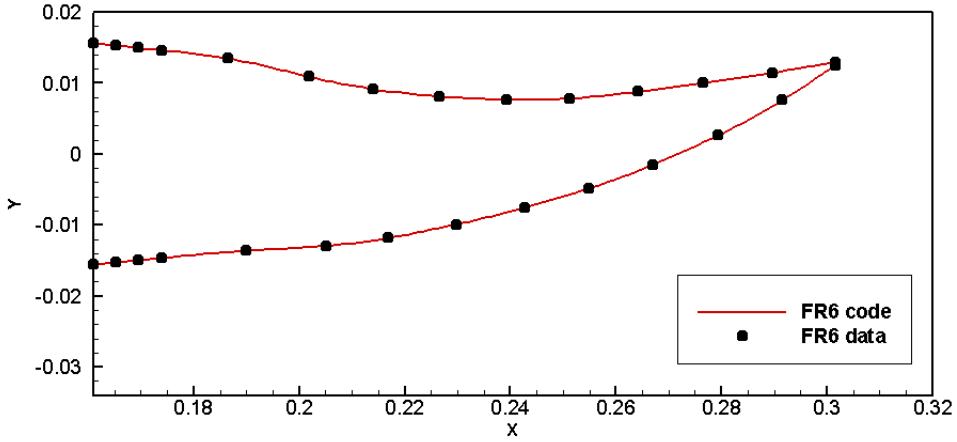


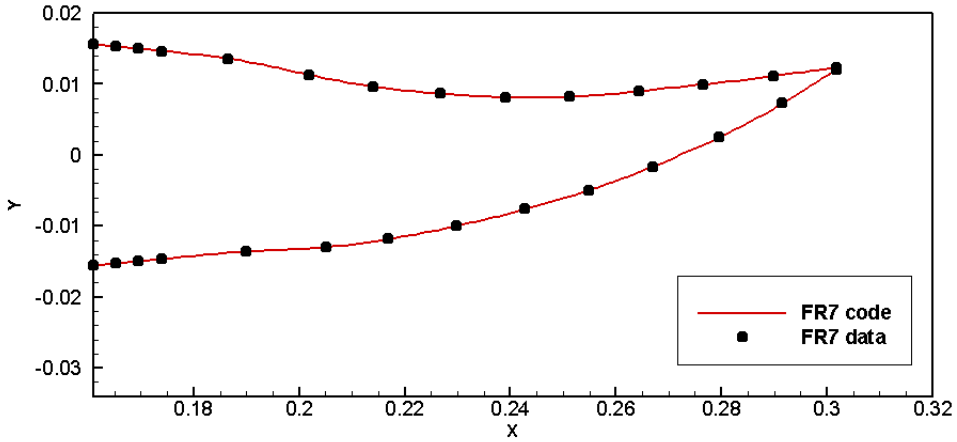

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR2 code
FR2 data

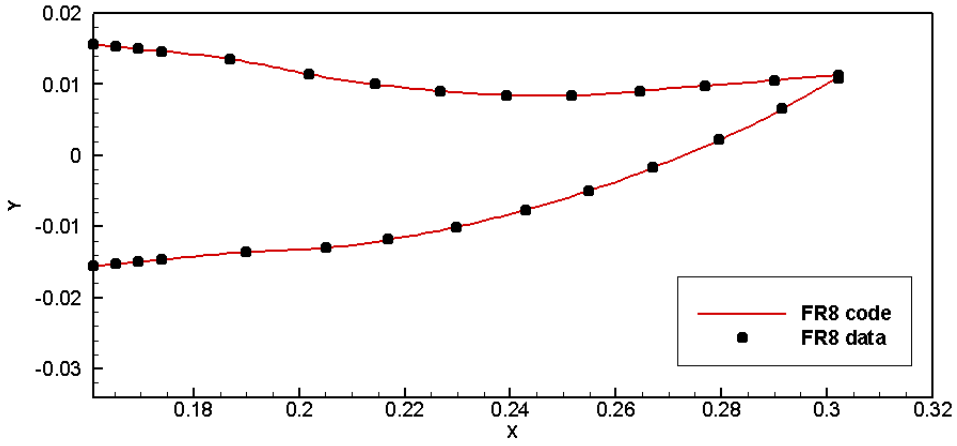

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR3 code
FR3 data

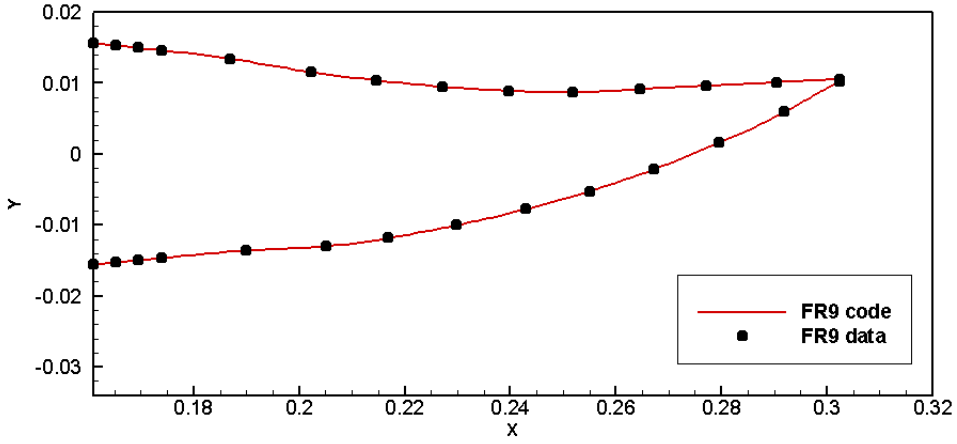

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR4 code
FR4 data

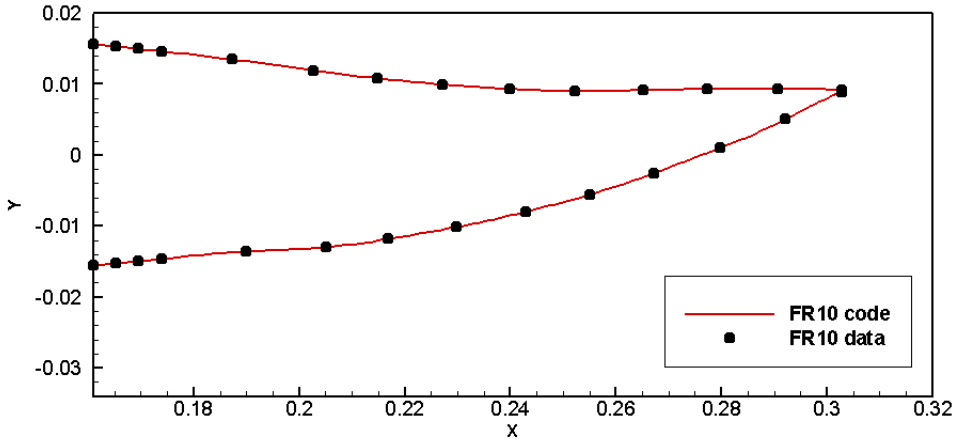

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR5 code
FR5 data

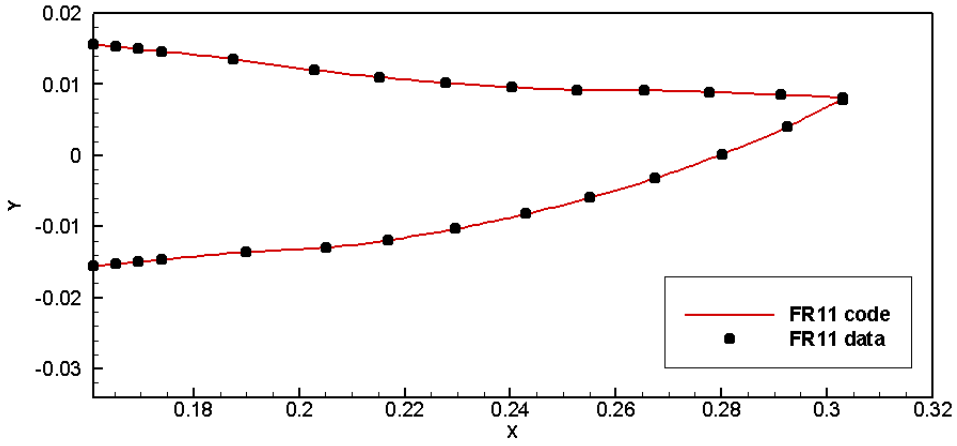

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR6 code
FR6 data

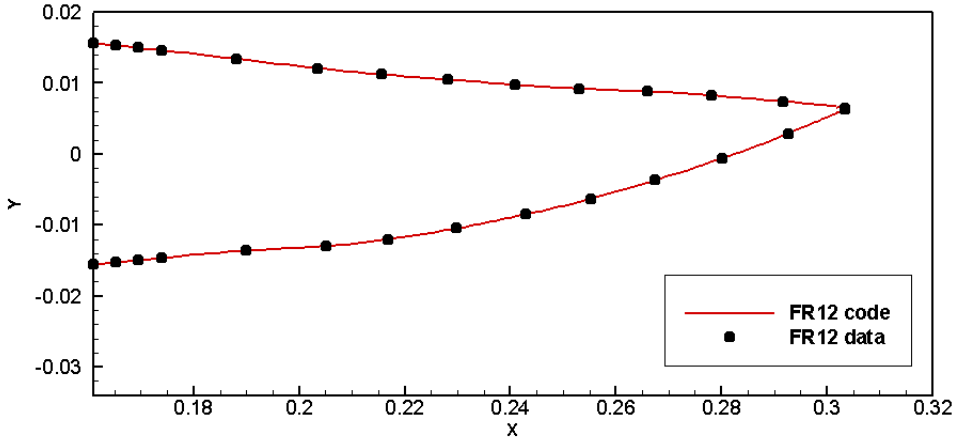

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR7 code
FR7 data

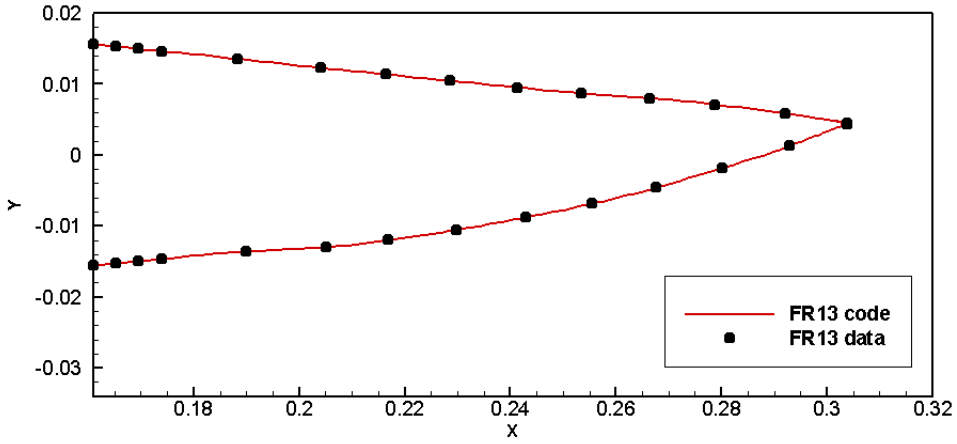

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR8 code
FR8 data

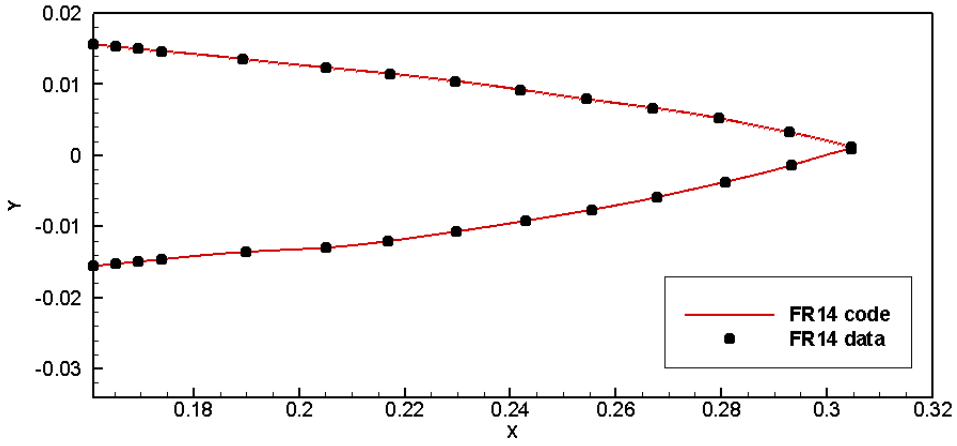

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR9 code
FR9 data

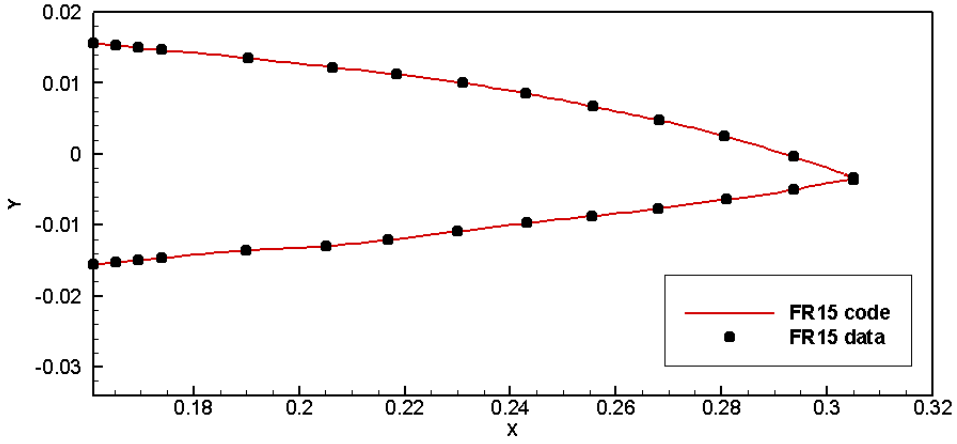

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR10 code
FR10 data

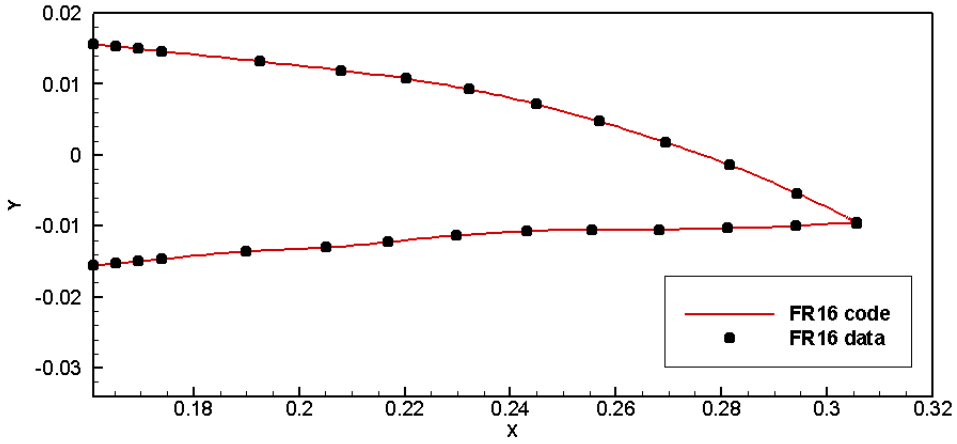

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR11 code
FR11 data

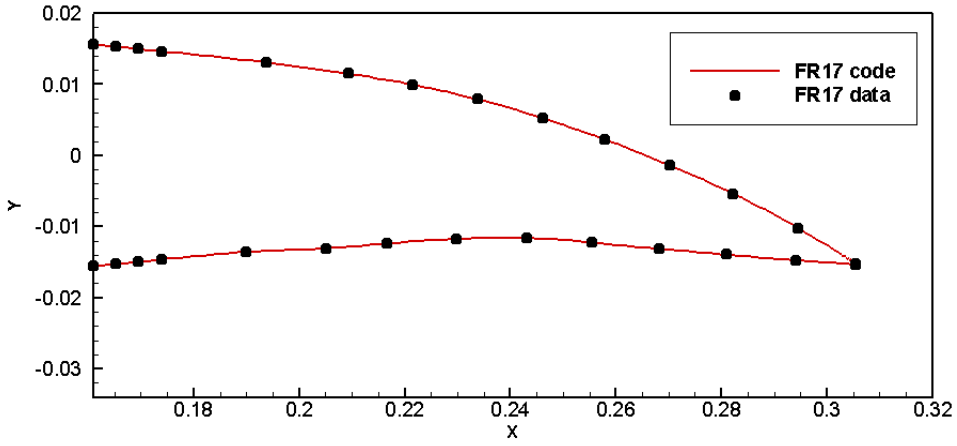

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR12 code
FR12 data

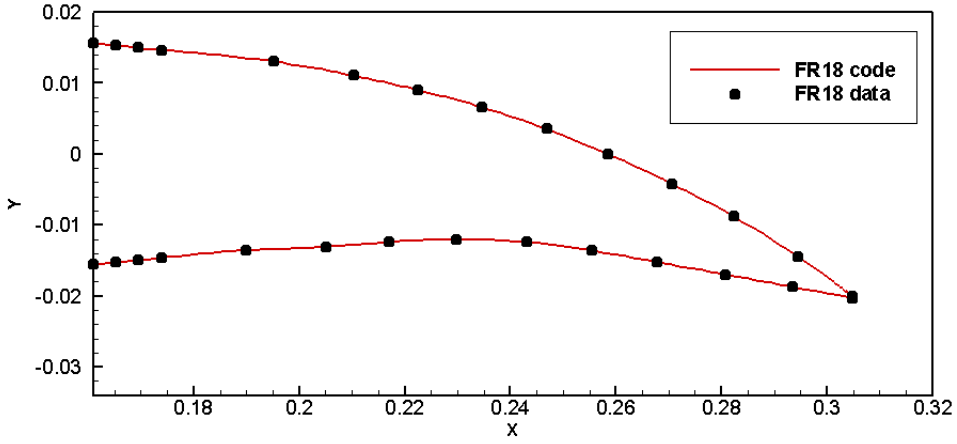

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR13 code
FR13 data

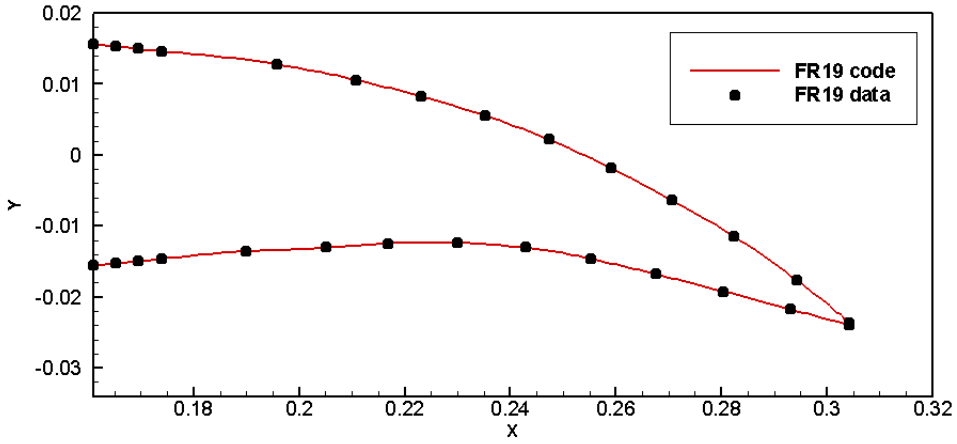

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR14 code
FR14 data

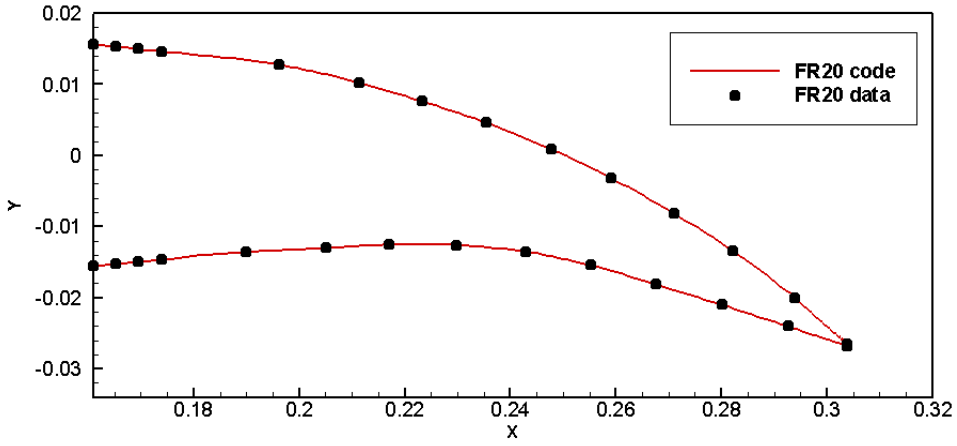

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR15 code
FR15 data

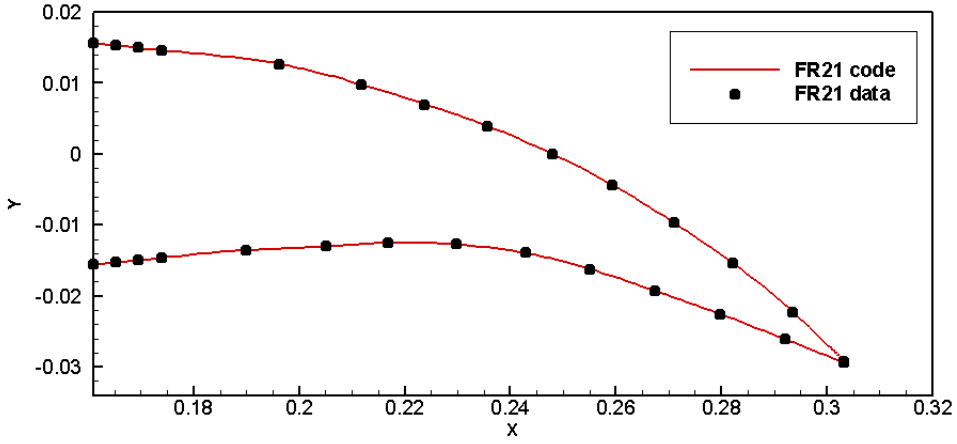

0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X
FR16 code
FR16 data


FR17 code
FR17 data
0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X


FR18 code
FR18 data
0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X


FR19 code
FR19 data
0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X


FR20 code
FR20 data
0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X


FR21 code
FR21 data
0.02
0.01
0
-0.01
-0.02
-0.03
Y
0.18
0.2
0.22
0.24
0.26
0.28
0.3
0.32
X

Appendix B

User-Defined Function

```
#include "udf.h"
#include "dynamesh_tools.h"
#include "math.h"
#include "para.h"
#include "prf.h"
#include "unsteady.h"
#include "string.h"

int rotating_case = 1;           //set to 0 for non-rotating, 1 for rotating
int continuousDeformingCase = 1; //set to 0 for deformation according to azimuthal location
int h = 14;                      //h is the frame profile h range from 1 to 21
int g = 1;                       //g = 1 for deflecting upward, g = -1 for deflecting downward
int b = 0;                       //b = subframes, b range from 0 to the specified subf value
int subf=20;                     //maximum subframes between each frame profiles
int zone_ID_upper = 33;          //zone ID of the thread, should be matched with Fluent bc
int zone_ID_lower = 34;
int zone_ID_tip = 35;
int zone_ID_cmeshborder = 36;
int zone_ID_int_cmesh_surface = 26;
int zone_ID_rotating_interface = 37;
int n_checkpt_nodes;
int n_nodes[3];
int a = 0;                       //indexing: a = 0 for upper, 1 for lower, 2 for tip
int last_ts = -1;
int initial_mesh = 0;
int ini_h = 0;
int ini_b = 0;
int targetframe = 1;
int keepframe = 1;            //flag 0 to calculate new coordinates
int call_grid_motion = 0;     //flag 1 to give the nodes new coordinates
int boundary_calculated = 0;  //flag 1 to indicate that node coordinates already calculated
```

```
double theta = 0;
double theta_point = 0;
double const rpm = 90;
double const centerOfRotationX = 0.07575;
double const centerOfRotationY = -2.6975;
Thread *airfoil[3];
Node *nodeID[3][9999];
Node *node_to_update[1000000];
Node *max_x_nodeID[2];
real nodePOS[3][2][999];
real tempxy[3][2][9999];      //to store all the nodes of upper,lower,tip in the 0 coordinate
real coord[1000000][2];       //to store all the initial node coordinates in the Cmesh only
real const xmin = 0.16094;    //the minimum relevant x-coord value for upper and lower in 0 coord
real xmax[21][20+1];          //to store max x-coord, second array size: subf+1

/*******************************************************
Convert the theta value to PI values 0 <= theta <= 360
*******************************************************/
double convert360(double theta)
{
   double theta_degree = theta*360/(2*M_PI);
   double value = 0;

   if (theta_degree>360)
   {
      if(fmod(theta_degree, 360) != 0)
      {
         value = theta_degree - 360*(int)(theta_degree/360);
      }
      else
      {
         value = 360;
      }
   }
   else
   {
      value = theta_degree;
   }

   return value;
}
```

```
/*****************************************************
Convert the theta value to PI values 0 <= theta <= 2PI
*****************************************************/
double convert2PI(double theta)
{
   double value = 0;

   if (theta>(2*M_PI))
   {
      if(fmod(theta, (2*M_PI)) != 0)
      {
         value = theta - (2*M_PI)*(int)(theta/(2*M_PI));
      }
      else
      {
         value = (2*M_PI);
      }
   }
   else
   {
      value = theta;
   }

   return value;
}

/*****************************************************************************
Transforms the coordinate space into clockwise of azimuthal angle
*****************************************************************************/
real transf_cw(real x, real y, double angle, int x_or_y, int print)
{
   real new_coord = 0;

   if (x_or_y == 0)
   {
      if (angle!=0){
         new_coord = centerOfRotationX+(x-centerOfRotationX)*cos(angle)
         +(y-centerOfRotationY)*sin(angle);
      }
      else{
         new_coord = x;
      }
   }
   else
   {
      if (angle!=0){
```

```
      new_coord = centerOfRotationY-(x-centerOfRotationX)*sin(angle)
      +(y-centerOfRotationY)*cos(angle);
    }
    else{
      new_coord = y;
    }
  }

  return new_coord;
}

/*****************************************************************************
Transforms the coordinate space into counter clockwise of azimuthal angle
*****************************************************************************/
real transf_ccw(real x, real y, double angle, int x_or_y, int print)
{
  real new_coord = 0;

  if (x_or_y == 0)
  {
    if (angle!=0){
      new_coord = centerOfRotationX+(x-centerOfRotationX)*cos(angle)
      -(y-centerOfRotationY)*sin(angle);
    }
    else{
      new_coord = x;
    }
  }
  else
  {
    if (angle!=0){
      new_coord = centerOfRotationY+(x-centerOfRotationX)*sin(angle)
      +(y-centerOfRotationY)*cos(angle);
    }
    else{
      new_coord = y;
    }
  }

  return new_coord;
}
```

```
/*****************************************************************************
Takes care of the looping of airfoil profiles from Fr1->Fr21 and Fr21->Fr1
*****************************************************************************/
void incrementframe()
{
   int c = 0;
   c = h % 21;

   if (c==0)
   {
         g = -1;
         b = subf;
         h+=g;
   }
   else if (c==1)
   {
      if(b==0)
      {
         g = 1;
         if (subf!=0){
            b+=g;
         }
         else{
            h+=g;
         }
      }
      else
      {
         b+=g;
         if (b>subf){
            h+=g;
            b = 0;
         }
      }
   }
   else
   {
      if (g > 0)
      {
         b+=g;
         if (b>subf)
         {
            h+=g;
            b = 0;
         }
      }
      else
```

```
      {
         if (b==0)
         {
            h+=g;
            b = subf;
         }
         else{
            b+=g;
         }
      }
   }
}

/*****************************************************************************
Calculates the y-coordinates of the nodes according to the azimuthal position 0
*****************************************************************************/
real get_ycoord_fourier(int frame, int a, real xcoord)
//subroutine omitted for brevity, refer to Appendix A

/*****************************************************************************
Store all the node coordinates of the initial Cmesh used in the simulation
*****************************************************************************/
void store_coord(int h, int b)
{
   Domain *domain;
   Thread *f_thread,*tf1,*tf2,*tf3,*tf4,*tf5;
   Thread *zone[5];
   Node *node;
   face_t f;
   int n, i, j;
   int m=0;
   int repeat = 1;
   int pt = h-1;
   int repeating_node = 0;

   domain = Get_Domain(1);

   tf1 = Lookup_Thread(domain,zone_ID_upper);
   tf2 = Lookup_Thread(domain,zone_ID_lower);
   tf3 = Lookup_Thread(domain,zone_ID_tip);
   tf4 = Lookup_Thread(domain,zone_ID_cmeshborder);
   tf5 = Lookup_Thread(domain,zone_ID_int_cmesh_surface);

   zone[0]=tf1;
   zone[1]=tf2;
   zone[2]=tf3;
   zone[3]=tf4;
```

```
   zone[4]=tf5;

   for(i=0;i<5;i++)
   {
      begin_f_loop(f,zone[i])
      {
         f_node_loop(f,zone[i],n)
         {
            node = F_NODE(f,zone[i],n);

            repeating_node = 0;
            for (j=0;j<m;j++)
            {
               if (node == node_to_update[j])
               {
                  repeating_node = 1;
                  break;
               }
            }

            if (repeating_node != 1)
            {
               node_to_update[m] = node;
               coord[m][0] = transf_cw(NODE_X(node),NODE_Y(node),theta,0,1);
               coord[m][1] = transf_cw(NODE_X(node),NODE_Y(node),theta,1,1);
               m++;
            }
         }
      }
      end_f_loop(f,zone[i]);
   }
   n_checkpt_nodes = m;
}

/*****************************************************************************************
Retrieve all the node coordinates of the initial Cmesh used in the simulation
*****************************************************************************************/
void retrieve_coord(int h, int b)
{
   Node *node;
   int m=0;
   double discrep=0;
   double theta_val=0;
   int pt = h-1;

   theta_val = convert2PI(theta);
```

```
   for (m=0; m<n_checkpt_nodes; m++)
   {
      node = node_to_update[m];

      NODE_X(node) = transf_ccw(coord[m][0],coord[m][1],theta,0,0);
      NODE_Y(node) = transf_ccw(coord[m][0],coord[m][1],theta,1,0);
   }
}

/*******************************************************************************
Calculates the node position on the upper, lower, tip surfaces of the airfoil
*******************************************************************************/
void calcul_boundary()
{
   Domain *domain;
   Node *node;
   Node *temp1;
   face_t f;
   Thread *tf1, *tf2, *tf3;
   int temp_int1, temp_int2;
   int repeat = 1;
   int i = 0, j = 0, k = 0, n = 0, m = 0, s = 0, d = 0, p = 0;
   double dps = dps = rpm*360/60;
   double rps = rpm*2*M_PI/60;
   real temp_real;
   real ratio[3][999];
   real current_max_x[2];
   real current_y_of_max_x[2];
   real global_current_max_x[2];
   real max = -9999;

   real yval1 =0.0;
   real yval2=0.0;

   real global_yu = 0;
   real global_yl = 0;

   real y_u = -99999;
   real y_l = 99999;
   real x_tip = 0;

   domain = Get_Domain(1);

   if (rotating_case!=0){
      theta = CURRENT_TIME*rps;
   }
   else{
```

```
   theta = 0;
}

tf1 = Lookup_Thread(domain,zone_ID_upper);
tf2 = Lookup_Thread(domain,zone_ID_lower);
tf3 = Lookup_Thread(domain,zone_ID_tip);

airfoil[0] = tf1;
airfoil[1] = tf2;
airfoil[2] = tf3;

xmax[0][0] = 0.3007;
xmax[1][0] = 0.3008553999999986;
xmax[2][0] = 0.3011335000000000;
xmax[3][0] = 0.3010631999999980;
xmax[4][0] = 0.3012042999999986;
xmax[5][0] = 0.3013326999999996;
xmax[6][0] = 0.3011037999999993;
xmax[7][0] = 0.3015872000000004;
xmax[8][0] = 0.3017690999999964;
xmax[9][0] = 0.3020107999999985;
xmax[10][0] = 0.3021791999999968;
xmax[11][0] = 0.3026812999999977;
xmax[12][0] = 0.3028716999999976;
xmax[13][0] = 0.3037125999999974;
xmax[14][0] = 0.3041798999999973;
xmax[15][0] = 0.3045871999999969;
xmax[16][0] = 0.3046234999999967;
xmax[17][0] = 0.3042084999999972;
xmax[18][0] = 0.3039504999999977;
xmax[19][0] = 0.3033040999999981;
xmax[20][0] = 0.3027535999999980;

#if !RP_HOST

d = h-1;

call_grid_motion = 0;

if (rotating_case!=0){
   Message0("\n ======== theta is %f rad or %f deg ========", theta, convert360(theta));
}

if (initial_mesh == 0)
{
   Message0("\n storing coordinates..");
   store_coord(h,b);
```

```
   ini_h = h;
   ini_b = b;
   Message0("\n ++++++++++ ini_h is %d, ini_b is %d ++++++++++ \n", h,b);
   initial_mesh = 1;
}

theta_point = convert360(theta);

if(continuousDeformingCase){
   keepframe = 0;
}

/*Specifies when to start deforming, and to which profile to deform to*/
if (!continuousDeformingCase)
{
   if ((theta_point<90) && ((90-theta_point) < (CURRENT_TIMESTEP*dps)))
   {
      targetframe = 21;
      keepframe = 0;
      Message0("theta = %f, theta_point = %f, start deforming \
      from %d to target frame %d", theta, theta_point, h, targetframe);
   }
   else if ((theta_point<350) && ((350-theta_point) < (CURRENT_TIMESTEP*dps)))
   {
      targetframe = 1;
      keepframe = 0;
      Message0("theta = %f, theta_point = %f, start deforming \
      from %d to target frame %d", theta, theta_point, h, targetframe);
   }
}

/*Only calculates new coordinates when deforming*/
if(keepframe == 0)
{
   n_nodes[0] = 0;
   n_nodes[1] = 0;
   n_nodes[2] = 0;

   /*Storing all the node coordinates on the upper,lower,tip surface in the transformed
   azimuthal angle = 0 space for calculation of new coordinates*/
   for(i=0;i<3;i++)
   {
      m=0;

      begin_f_loop(f,airfoil[i])
      {
         f_node_loop(f,airfoil[i],n)
```

```
      {
         if PRINCIPAL_FACE_P(f,airfoil[i])
         {
            node = F_NODE(f,airfoil[i],n);

            repeat = 1;

            if(((transf_cw(NODE_X(node),NODE_Y(node),theta,0,0))>=0.16094) ||
            (fabs(transf_cw(NODE_X(node),NODE_Y(node),theta,0,0)-0.16094)<FLT_EPSILON))
            {
               for(j=0;j<m;j++)
               {
                  if (node == nodeID[i][j])
                  {
                     repeat = 0;
                     break;
                  }
               }
               if (repeat != 0)
               {
                  nodeID[i][m] = node;
                  tempxy[i][0][m]=transf_cw(NODE_X(node),NODE_Y(node),theta,0,1);
                  tempxy[i][1][m]=transf_cw(NODE_X(node),NODE_Y(node),theta,1,1);
                  m++;
               }
            }
         }
      }
   }
   end_f_loop(f,airfoil[i]);
   n_nodes[i] = m;
}

/*Rearranging according to ascending x-coord for both upper and lower surface*/
for (i=0; i<2; i++)
{
   for (j=0; j<(n_nodes[i]-1); j++)
   {
      for (n=0; n<(n_nodes[i]-j-1);n++)
      {
         if (tempxy[i][0][n] > tempxy[i][0][n+1])
         {
            temp1 = nodeID[i][n];
            nodeID[i][n] = nodeID[i][n+1];
            nodeID[i][n+1] = temp1;
            temp_real = tempxy[i][0][n];
            tempxy[i][0][n] = tempxy[i][0][n+1];
```

```
            tempxy[i][0][n+1] = temp_real;
            temp_real = tempxy[i][1][n];
            tempxy[i][1][n] = tempxy[i][1][n+1];
            tempxy[i][1][n+1] = temp_real;
         }
      }
   }
}

/*Setting the maximum x-coord values for both upper and lower surface*/
for(i=0;i<2;i++)
{
   if(n_nodes[i]!=0){
      current_max_x[i] = tempxy[i][0][(n_nodes[i]-1)];
   }
}

/*Identifying the highest x-coord of the upper and lower surfaces among all compute nodes*/
for (i=0; i<2; i++)
{
   if (n_nodes[i]==0)
   {
      current_max_x[i] = -9999;
   }
   global_current_max_x[i] = PRF_GRHIGH1(current_max_x[i]);
}

/*Rearranging the nodes on the tip according to ascending y-coord*/
for (j=0; j<(n_nodes[2]-1); j++)
{
   for (n=0; n<(n_nodes[i]-j-1);n++)
   {
      if (tempxy[i][1][n] > tempxy[i][1][n+1])
      {
         temp1 = nodeID[2][n];
         nodeID[2][n] = nodeID[2][n+1];
         nodeID[2][n+1] = temp1;
         temp_real = tempxy[2][0][n];
         tempxy[2][0][n] = tempxy[2][0][n+1];
         tempxy[2][0][n+1] = temp_real;
         temp_real = tempxy[2][1][n];
         tempxy[2][1][n] = tempxy[2][1][n+1];
         tempxy[2][1][n+1] = temp_real;
      }
   }
}
```

```
/*Identifying the highest and lowest y-coord of the tip among all compute nodes*/
if(n_nodes[2]==0)
{
   tempxy[2][1][n_nodes[2]-1] = -9999;
   tempxy[2][1][0] = 9999;
}
current_y_of_max_x[0] = PRF_GRHIGH1(tempxy[2][1][n_nodes[2]-1]);
current_y_of_max_x[1] = PRF_GRLOW1(tempxy[2][1][0]);

/*Calculating the relative position of each node on the
surface based on the initial node distribution*/
for (a=0;a<3;a++)
{
   if (n_nodes[a]>0){

      for (i=0;i<n_nodes[a];i++)
      {
         node = nodeID[a][i];
         if((a==0)||(a==1))
         {
            ratio[a][i] = (transf_cw(NODE_X(node),NODE_Y(node),theta,0,1)-0.16094)/
            (global_current_max_x[a]-0.16094);
         }
         else
         {
            ratio[a][i] = (transf_cw(NODE_X(node),NODE_Y(node),theta,1,1)-
            current_y_of_max_x[1])/(current_y_of_max_x[0]-current_y_of_max_x[1]);
         }
      }
   }
}

incrementframe();
Message0("\n ******INCR/DECR @ THETA = %f, THETA_POINT = %f, TARGETFRAME is %d, h is %d, "
"b is %d, KEEPFRAME is %d******\n", theta, theta_point, targetframe, h, b, keepframe);

d = h-1; //to update d after incrementing frame

if ((h == ini_h) && (b == ini_b))
{
   Message0("\n ******** RETREIVING COORD FOR FRAME %d b = %d *********", h, b);
   retrieve_coord(h,b);
}
else
{
   call_grid_motion = 1;
```

```
if (b!=0)
{
   if (g<0){
      xmax[d][b] = xmax[d+1][0] + (xmax[d][0] - xmax[d+1][0])/
      (subf + 1)*(subf + 1 - b);
   }
   else{
      xmax[d][b] = xmax[d][0] + (xmax[d+1][0] - xmax[d][0])/(subf + 1)*b;
   }
}

for (a=0;a<2;a++)
{
   p = n_nodes[a];

   for (i=0;i<p;i++)
   {
      node = nodeID[a][i];

      nodePOS[a][0][i] = 0.16094 + ratio[a][i]*(xmax[d][b]-0.16094);

      if((nodePOS[a][0][i]-xmax[d][b])>FLT_EPSILON){
         Message(" WARNING! x exceeded the xmax for frame %d with \
         nodePOS = %f \n", d+1, nodePOS[a][0][i]);
      }

      if (b!=0)
      {
         if (g<0)
         {
            yval1 = get_ycoord_fourier(h+1,a,nodePOS[a][0][i]);
            yval2 = get_ycoord_fourier(h,a,nodePOS[a][0][i]);
            nodePOS[a][1][i] = yval1 + (yval2 - yval1) /
            (subf + 1)*(subf + 1 - b);
         }
         else
         {
            yval1 = get_ycoord_fourier(h,a,nodePOS[a][0][i]);
            yval2 = get_ycoord_fourier(h+1,a,nodePOS[a][0][i]);
            nodePOS[a][1][i] = yval1 + (yval2 - yval1) / (subf + 1)*b;
         }
      }
      else
      {
         nodePOS[a][1][i] = get_ycoord_fourier(h,a,nodePOS[a][0][i]);
      }
```

```
            if (fabs(nodePOS[a][0][i]-xmax[d][b]) < FLT_EPSILON)
            {
               x_tip = nodePOS[a][0][i];

               if (a==0){
                  y_u = nodePOS[a][1][i];
               }
               else{
                  y_l = nodePOS[a][1][i];
               }
            }
         }
      }

      global_yu = PRF_GRHIGH1(y_u);
      global_yl = PRF_GRLOW1(y_l);

      for (j=0;j<n_nodes[2];j++)
      {
         node = nodeID[2][j];
         nodePOS[2][0][j] = x_tip;
         nodePOS[2][1][j] = global_yl + ratio[2][j]*(global_yu-global_yl);
      }
   }

   if (!continuousDeformingCase)
   {
      if ((h==targetframe) && (b==0))
      {
         keepframe = 1;
      }
   }
 }
 #endif
}

/************************************************************
Fluent function that gets called once right after loading the UDF
************************************************************/
DEFINE_EXECUTE_ON_LOADING(executeOnLoading,justForExecuteOnLoading)
{
 calcul_boundary();
}
```

```
/************************************************************************
Fluent function that gets called at the beginning of each iteration
************************************************************************/
DEFINE_ADJUST(calcu_coord, domain)
{
   int curr_ts = N_TIME;
   //only executes at the beginning of each timestep
   if (last_ts != curr_ts)
   {
      calcul_boundary();
      last_ts = curr_ts;
   }
}

/*****************************************************************************
Fluent function that allows point by point prescription of node coordinates
*****************************************************************************/
DEFINE_GRID_MOTION(tip,domain,dt,time,dtime)
{
   Node *node;
   face_t f;
   Thread *tf3;
   int i=0 ,p = 0;

   if (call_grid_motion ==1)
   {
      domain = Get_Domain(1);
      tf3 = Lookup_Thread(domain,zone_ID_tip);
      airfoil[2] = tf3;
      a = 2;
      p = n_nodes[a];

      if (p>0)
      {
         SET_DEFORMING_THREAD_FLAG(THREAD_T0(airfoil[2]));

         for (i=0;i<p;i++)
         {
            node = nodeID[a][i];
            NODE_X(node) = transf_ccw(nodePOS[a][0][i],nodePOS[a][1][i],theta,0,0);
            NODE_Y(node) = transf_ccw(nodePOS[a][0][i],nodePOS[a][1][i],theta,1,0);
         }
      }
   }
}
```

```
/*****************************************************************************
Fluent function that allows point by point prescription of node coordinates
*****************************************************************************/
DEFINE_GRID_MOTION(upper,domain,dt,time,dtime)
{
   Node *node;
   face_t f;
   Thread *tf1;
   int i=0 ,p = 0;

   if (call_grid_motion == 1)
   {
      domain = Get_Domain(1);
      tf1 = Lookup_Thread(domain,zone_ID_upper);
      airfoil[0] = tf1;
      a = 0;
      p = n_nodes[a];

      if (p>0)
      {
         SET_DEFORMING_THREAD_FLAG(THREAD_T0(airfoil[a]));

         for (i=0;i<p;i++)
         {
            node = nodeID[a][i];
            NODE_X(node) = transf_ccw(nodePOS[a][0][i],nodePOS[a][1][i],theta,0,0);
            NODE_Y(node) = transf_ccw(nodePOS[a][0][i],nodePOS[a][1][i],theta,1,0);
         }
      }
   }
}

/*****************************************************************************
Fluent function that allows point by point prescription of node coordinates
*****************************************************************************/
DEFINE_GRID_MOTION(lower,domain,dt,time,dtime)
{
   Node *node;
   face_t f;
   Thread *tf2;
   int i=0 ,p = 0;

   if (call_grid_motion == 1)
   {
      domain = Get_Domain(1);
      tf2 = Lookup_Thread(domain,zone_ID_lower);
      airfoil[1] = tf2;
```

```
      a = 1;
      p = n_nodes[a];

      if(p>0)
      {
         SET_DEFORMING_THREAD_FLAG(THREAD_T0(airfoil[a]));

         for (i=0;i<p;i++)
         {
            node = nodeID[a][i];
            NODE_X(node) = transf_ccw(nodePOS[a][0][i],nodePOS[a][1][i],theta,0,0);
            NODE_Y(node) = transf_ccw(nodePOS[a][0][i],nodePOS[a][1][i],theta,1,0);
         }
      }
   }
}
```