

Numerical investigation of a novel blade for use in Vertical Axis Wind Turbines

by

Gareth Erfort



*Dissertation presented for the degree of Doctorate of
Philosophy in Mechanical Engineering in the Faculty of
Engineering at Stellenbosch University*

Supervisor: Prof. T.W Von Backström

Co-supervisor: Prof. G Venter

December 2019

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2019

Copyright © 2019 Stellenbosch University
All rights reserved.

Abstract

Numerical investigation of a novel blade for use in Vertical Axis Wind Turbines

G. Erfort

*Department of Mechanical and Mechatronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.*

Dissertation: PhDEng (Mech)

December 2019

Renewable energy prospects in South Africa have been growing thanks to the government's commitment to alternative energy sources. The country has committed to 36 projects ranging in size from 52 MW to 140 MW. South Africa's sole energy distributor has been implementing rolling black-outs due to unscheduled maintenance on their plants. This has resulted in average two hour periods of no power for citizens and companies alike. These entities have turned to storage and small scale renewable generation to tide them over during a black out. Within the country wind power is therefore being utilized on both a commercial and private scale. Vertical axis wind turbines have been identified for their applicability in large scale off shore wind farms and ability to operate in urban environments, as a future power generation technology.

At this time the vertical axis wind turbine is however not a common sight for power generation. Studies have indicated that a few inherent traits of this turbine design have hindered deployment due to increased manufacturing cost associated with their mitigation. The variation in torque generated during the course of rotation is an example. It can result in drive train stresses, and negatively affect the fatigue life of drive components.

This dissertation is aimed at reducing the variation in torque experienced by a straight bladed Darrieus turbine during operation. The study proposed a novel blade that would allow for adjustments to the forces experienced by the turbine during rotation.

A virtual laboratory was used to analyse the effect of the blade. An analytical model for a two-bladed H-rotor was implemented in Python and validated against published data. The model was based on the double multiple stream-tube (DMST) method as it provided fast accurate solutions. The blade is

designed to have an adaptive distortion on the upper surface. The distortion is able to change height and thereby control the tripping of the boundary layer from laminar to turbulent flow. Lift and drag coefficients for the blade were obtained through computational fluid dynamic (CFD) simulations in the open source software OpenFOAM. A transitional turbulence model based on momentum thickness and intermittency was implemented and adjusted to increase efficiency. A random forest surrogate model was used in optimization to determine the exact nature of the proposed distortion.

The blade design proved effective in reducing the torque ripple. Placement of the distortion was predominantly on the leading edge (LE) of the blade, where a small change in shape had the largest effect on the boundary layer. The optimized solution reduced the maximum possible torque that the turbine could achieve by synchronously increasing the drag experienced by the blade with the torque fluctuations. The reduction in ripple resulted in an increased life span of the drive train shaft by an estimated 36%. An equation relating the reduction in torque ripple to the reduction in coefficient of performance was identified.

Uittreksel

Numeriese ondersoek van 'n nuwe lem vir gebruik in Vertikale-As Windturbines

(“Numerical Investigation of a Novel Blade for use in Vertical Axis Wind Turbines”)

G. Erfort

*Departement Meganiese en Megatroniese Ingenieurswese,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.*

Proefskrif: PhDIng (Meg)

Desember 2019

Hernubare energievoorsigte in Suid-Afrika het verbeter danksy die regering se verbintenis tot alternatiewe energiebronne. Die land is verbind tot 36 projekte wat wissel van 52 MW tot 140 MW. Suid-Afrika se enigste energiever spreider is tans besig om kragonderbrekings te implementeer weens ongeskeduleerde instandhouding van aanlegte. Dit het burgers sowel as maatskappye blootgestel aan twee-uur lange kragonderbrekings. Hulle het hulle dus gewend na die stoor van energie en na kleinskaalse hernubare kragopwekking om hulle te help tydens beurtkrag. In die land word windkrag tans kommersiëel en privaat gebruik. Weens die toepaslikheid van die vertikale-as windturbine in grootskaalse afluende windplase, en die vermoë om in stedelike omgewings te funksioneer, word dit erken as 'n toekomstige kragopwekkingstechnologie.

Tans is die vertikale-as windturbine egter nie in algemene gebruik vir kragopwekking nie. Studies het aangedui dat 'n paar inherente eienskappe van hierdie turbine-ontwerp die implementering verhinder het as gevolg van verhoogde geassosieerde vervaardigingskoste. Die variasie in opgewekte wringkrag tydens die rotasie is 'n voorbeeld. Dit kan lei tot spanning in aandryfstelsels, en die vermoedidheidslewe van aandryfonderdele negatief beïnvloed.

Hierdie proefskrif is daarop gemik om die variasie in wringkrag wat 'n reguit-lem Darrieus-turbine ondervind te verminder. Dit studie stel 'n nuwe lem voor wat toelaat dat die kragte wat die turbine tydens rotasie ervaar, aangepas kan word.

'n Virtuele laboratorium is gebruik om die effek van die lem te analiseer. 'n Analitiese model vir 'n twee-lem H-rotor is geïmplementeer in Python, en

bevestig teen gepubliseerde data. Die model is gebaseer op die tweevoudige stroomstrook (DMST) metode, aangesien dit vinnige en akkurate oplossings verskaf. Die lem is ontwerp om 'n aanpasbare vervorming op die boonste oppervlak te hê. Die vervorming kan van hoogte verander en sodoende die oorgang van die grenslaag van laminêre tot turbulente vloei beheer. Hef- en sleurkoëffisiënte vir die lem is verkry deur middel van 'n oorgangsturbulensiemodel gebaseer op momentumdikte en onderbrokenheid, en is geïmplementeer en aangepas om doeltreffendheid te verhoog. 'n "Random forest" surrogaatmodel is gebruik in optimering om die presiese aard van die voorgestelde vervorming te bepaal.

Die lem ontwerp was effektief in die vermindering van die wringkrag rimpeling. Plasing van die vervorming was hoofsaaklik aan die voerpunt van die lem, waar 'n klein verandering in vorm die grootste effek op die grenslaag gehad het. Die geoptimeerde oplossing verminder die maksimum moontlike wringkrag wat die turbine kan behaal, deur die sleur wat die lem ervaar met die wringkrag-skommeling sinchronies te verhoog. Die vermindering in die rimpeling het gelei tot 'n verhoogde lewensduur van die aandryfas met 'n geskatte 36%. 'n Vergelyking wat die vermindering in wringkrag rimpeling met betrekking tot die vermindering in koëffisiënt van prestasie weergee, is geïdentifiseer.

Acknowledgements

I give thanks to God almighty for giving me patience and perseverance during the course of my study, furthermore I would also like to express my gratitude to

- My family and loved ones for their ever present support and understanding
- My **brothers** for their faith and encouragement along the way
- My supervisors for the guidance and insight during the course of this study
- My colleagues for dealing with my random questions
- The department for giving me the time and space to finish my studies
- Finally the Centre for High Performance Computing in Rosebank for the use of the world class cluster

Dedications

Life is a beautiful journey and I have been blessed to experience it abundantly
- Josh Luke Jansen

Contents

Declaration	i
Abstract	iii
Uittreksel	v
Acknowledgements	vii
Dedications	viii
Contents	ix
List of Figures	xii
List of Tables	xv
Nomenclature	xvi
1 Problem Definition	1
1.1 Objectives	2
1.1.1 Analytical model	2
1.1.2 Blade characteristics	2
1.1.3 Optimized operation	3
1.2 Report outline	3
1.3 Contributing publications	5
2 Literature Review	8
2.1 Vertical axis wind turbines	8
2.2 Notable differences	10
2.3 Aerodynamics in wind turbines	12
2.3.1 Vortex model	12
2.3.2 Actuator disk models	12
2.4 Aerodynamic response	15
2.4.1 Camber changes	16
2.4.2 Boundary layer alterations	18

<i>CONTENTS</i>	ix
2.5 Load reduction in turbines	20
2.6 Surface roughness	21
3 Blade Design	25
3.1 Biomimicry	25
3.2 Manufactured distortions	26
3.3 Proposed blade	27
4 Numerical Modelling	30
4.1 Virtual laboratory	31
4.2 CFD model	32
4.2.1 OpenFOAM	32
4.2.2 Mesh development	33
4.2.3 Model initialization	42
4.2.4 Boundary layer	43
4.2.5 Reynolds-averaged Navier Stokes turbulence model	45
4.2.6 Boundary conditions	48
4.3 DMST model	50
5 Surrogate Modelling	56
5.1 Model selection	56
5.1.1 Random Forests	58
5.2 Optimization	60
5.2.1 Genetic algorithm	60
5.2.2 Breeder Genetic algorithm	61
6 Results	63
6.1 Optimization result	64
6.2 Trends in optimized results	67
6.3 Surrogate suitability	70
6.4 Effect of distortion	72
6.5 Turbulence model	72
7 Conclusion	78
7.1 Mesh	79
7.2 Turbulence	79
7.3 Surrogate model	80
Appendices	81
A Double Multiple Stream Tube Model	82
A.1 Analytical Model	82
A.1.1 Actuator Disk Model	82
A.1.2 Aerodynamic loading	84

B Surrogate Modelling	89
B.1 Response surfaces	89
B.2 Neural networks	90
B.3 Random forests	90
B.4 Support vector regression	91
B.4.1 Dual formulation	92
C Optimization	94
C.1 Introduction	94
C.2 Breeder Genetic algorithm(BGA)	94
C.2.1 Elites	95
C.2.2 Selection	95
C.2.3 Mutation	95
C.3 Combined schemes	96
D Life span calculations	97
D.1 Modified Goodman criterion	97
E Python code	99
List of References	121

List of Figures

1.1	Document outline	4
2.1	Basic concept behind the Savonius turbine taken from Islam <i>et al.</i> (2008)	9
2.2	Typical Darrieus configurations: (a) Full Darrieus, (b) H-Rotor, (c) "V", (d) Delta, (e) Diamond, (f) Giromill taken from Sutherland <i>et al.</i> (2012)	9
2.3	Vortex filament representation for a single blade element taken from Islam <i>et al.</i> (2008)	12
2.4	Stream tube model for wind turbine with enclosing control volume .	13
2.5	Inline cascade representation from Islam <i>et al.</i> (2008), blades are projected onto a single plane	15
2.6	DMST model showing upstream and downstream interference factors taken from Paraschivoiu and DELCLAUX (1983)	15
2.7	Angle of attack changes with changes to the camber taken from Griffin and McCoy (2008)	17
2.8	A micro-tab deployed at the trailing edge affects the flow patterns of the foil taken from Van Dam <i>et al.</i> (2007)	17
2.9	No applied jet vs applied jet stream taken from Shires and Kourkoulis (2013)	19
2.10	Angle of attack changes with change to boundary layer taken from Griffin and McCoy (2008)	19
2.11	Lift and Drag curves for Microtabs deployed on suction and pressure side of foil recreated from (Wilson <i>et al.</i> , 2009)	20
2.12	Power input and resultant power for a circulation controlled system recreated from Shires and Kourkoulis (2013)	21
2.13	Maximum lift coefficient for a NACA 0012 foil with various finishes as per Jones and Williams (1936)	22
2.14	Improvement in C_p due to surface roughness of a 2 bladed VAWT, taken from Howell <i>et al.</i> (2010)	23
3.1	Annotated image from Fish and Rohr (1999) showing turbulent versus laminar C_d values	25

3.2	Partially recreated image from Alam <i>et al.</i> (2010) for drag of various golf balls	27
3.3	The distortion to be implemented on a NACA0012 foil	28
3.4	The function governing distortion height	29
4.1	Work flow for virtual laboratory	31
4.2	C-MESH in wireframe view	33
4.3	C-MESH with boundary labels	34
4.4	A single block used to build a mesh in blockMesh taken from Green-shields (2015)	34
4.5	2D mesh as made up of blocks	35
4.6	Focused view of foil with additional 'block' on the upper surface . .	36
4.7	Flow diagram for mesh script	37
4.8	Changing height of distortion 15 mm wide at x/C of 0.15	39
4.9	Increased resolution on upper surface of foil	39
4.10	Mesh around the distortion	39
4.11	Calculation of face skewness	40
4.12	Velocity changes within the boundary layer	43
4.13	Illustration of boundary layer on a flat pate	44
4.14	Depiction of skin friction calculation	45
4.15	Turbulence intensity for VAWTs as reported by Pagnini <i>et al.</i> (2015)	48
4.16	C_l at Re 360 000 for NACA 0012	49
4.17	C_l at Re 700 000 for NACA 0012	49
4.18	C_d at Re 360 000 for NACA 0012	50
4.19	C_d at Re 700 000 for NACA 0012	50
4.20	Plan view of a blade in a VAWT as per Islam <i>et al.</i> (2008)	51
4.21	Interference Factors as determined by DMST analytical model, taken from Erfort <i>et al.</i> (2019b)	52
4.22	Interference Factors published in Paraschivoiu and DELCLAUX (1983)	53
4.23	Torque coefficients as determined by DMST analytical model, taken from Erfort <i>et al.</i> (2019b)	53
4.24	Torque coefficients published in Paraschivoiu and DELCLAUX (1983)	54
4.25	Torque ripple experienced by the VAWT during a single rotation . .	55
5.1	Neural Network	57
5.2	Support Vector Regression	57
5.3	Polynomial	57
5.4	Random forests	57
5.5	Surrogate model predictions for C_l	58
5.6	Surrogate model predictions for C_d	58
5.7	SVR surrogate for C_d used in Erfort <i>et al.</i> (2018b) showing a poor fit	59
6.1	The BSL torque ripple for a two bladed H-rotor	63

6.2	Comparison between BSL ripple and optimised solutions for group 4	66
6.3	Comparison between BSL ripple and optimised result from run 24	66
6.4	C_l for run 24	67
6.5	C_d for run 24	67
6.6	Reduction in ripple versus reduction in power coefficient	68
6.7	The change in distortion height for each run in group 1	69
6.8	The change in distortion height for each run in group 2	69
6.9	The change in distortion height for each run in group 3	70
6.10	The change in distortion height for each run in group 4	71
6.11	Coefficient of lift for surrogate and CFD	71
6.12	Coefficient of drag for surrogate and CFD	71
6.13	Coefficient of lift for surrogate and CFD	72
6.14	Coefficient of drag for surrogate and CFD	72
6.15	Coefficient of lift for turbulence models	73
6.16	Coefficient of lift for turbulence models	73
6.17	Skin friction coefficient on upper surface near LE	73
6.18	Comparison of relative pressure on foil surface for run 24, using $\gamma - \tilde{Re}_\theta$ and $k\omega$ SST models	74
6.19	Comparison of relative pressure on upper surface at the LE, using $\gamma - \tilde{Re}_\theta$ and $k\omega$ SST models	74
6.20	Turbulent kinetic energy around the distortion, based on $k\omega$ SST model	75
6.21	Turbulent kinetic energy around the distortion, based on $\gamma - \tilde{Re}_\theta$ model	76
A.1	Control volume used in actuator disk model	82
A.2	Physical model for VAWT	84
A.3	Relative velocity diagram of VAWT	85
A.4	Lift and drag vectors due to relative wind velocity	85
A.5	Decomposing normal and tangential coefficients into thrust	86
A.6	Streamtube based on airfoil shape	86
A.7	Relative velocity equations for quadrants	88
A.8	Normal and tangential equations for quadrants	88

List of Tables

2.1	Differences between the vertical axis and horizontal axis wind turbines	10
4.1	The working range of variables	38
4.2	Effect of refinement on RMSE between experiment and simulation .	41
4.3	Linear solver settings for steady state simulations	42
4.4	The overshoot in C_d predictions from CFD simulations	50
4.5	Parameters for DMST model input	52
5.1	Coefficient of determination for surrogate models	56
6.1	Design variables	64
6.2	Optimisation results	65

Nomenclature

Abbreviations

BGA	Breeder genetic algorithm
BSL	Base line model
CFD	Computational fluid dynamics
DILU	Diagonal incomplete-Cholesky
DMST	Double multiple streamtube
EA	Evolutionary algorithm
GAMG	Generalised geometric algebraic multi grid
HAWT	Horizontal axis wind turbine
LE	Leading edge
NN	Neural network
PBiCG	Preconditioned bi-conjugate gradient
RANS	Reynolds-averaged Navier Stokes
REIPPPP	Renewable Energy Independent Power Producers Procurement Program
RF	Random forest
RMSE	Root mean square error
SVR	Support Vector regression
TSR	Tip speed ratio
VAWT	Vertical axis wind turbine

Roman symbols

A	Area	$[\text{m}^2]$
a	Induction factor	$[\]$
c	Chord length	$[\text{m}]$
C_d	Coefficient of drag	$[\]$
C_l	Coefficient of lift	$[\]$
C_f	Skin friction coefficient	$[\]$
C_N	Coefficient of normal force	$[\]$
C_P	Coefficient of power/performance	$[\]$

C_Q	Coefficient of torque	[]
C_t	Coefficient of thrust	[]
C_T	Coefficient of tangential force	[]
D	Drag	[N]
f_p	Penalty function	[]
F_T	Tangential Force	[N]
G_N	Generational limit	[]
k	Turbulent kinetic energy	[m ² s ⁻²]
L	Lift	[N]
N	Number	[]
p	Pressure	[Pa]
P_0	Initial Population	[]
P_N	Population limit	[]
R	Radius	[m]
$\tilde{R}e_{\theta_t}$	Transition momentum thickness Reynolds number	[]
r	Pearson correlation coefficient	[]
t	Time	[s]
T	Thrust	[N]
y^+	Dimensionless wall distance	[]
U	Velocity	[m.s ⁻¹]
Tu	Turbulence intensity	[]
u	Velocity in x direction	[m.s ⁻¹]
v	Velocity in y direction	[m.s ⁻¹]
V_∞	Wind Velocity	[m.s ⁻¹]
V_a	Induction Velocity	[m.s ⁻¹]
V_T	Tangential Velocity	[m.s ⁻¹]
w	Velocity in y direction	[m.s ⁻¹]
W	Relative Velocity	[m.s ⁻¹]
X/c	Non-dimensional distance from leading edge	[]

Greek symbols

α	angle of attack	[°]
θ	Azimuth angle	[°]
Γ	Circulation per unit length	[]
ρ	Density	[kg.m ⁻²]
\dot{m}	Mass flow rate	[kg.s ⁻¹]
ω	Angular velocity	[rad.s ⁻¹]

\ddot{x}	Acceleration	[ms ⁻²]
θ	Rotation angle	[rad]
ω	Angular rotation	[rad.s ⁻¹]
λ	Tip speed ratio	[]
μ	Dynamic viscosity	[kg.m ⁻¹]
μ_T	Eddy viscosity	[kg.m ⁻¹]
ν	Kinematic viscosity	[m ² .s ⁻¹]
\tilde{Re}_{θ_t}	Momentum thickness Reynolds number	[]
γ	Intermittency	[]
λ_θ	Pressure gradient	[]
ϵ	Dissipation	[]
σ_k	Closure coefficient	[]
ζ	Boundary array	[]
η	Elite members of population	[]
Ω	Termination criteria	[]

Constants and symbols

π	ratio of circle's circumference to its diameter	
∇	gradient operator	[]
μ_m	Mutation rate	[]
μ_r	Recombination rate	[]

Vectors and Tensors

U_i	Time averaged velocity vector	[m.s ⁻¹]
τ_{ij}	Reynolds stress tensor	[m ² .s ⁻²]
S_{ij}	Strain rate tensor	[s ⁻¹]
u'_i	Instantaneous fluctuating velocity vector	[m.s ⁻¹]
x_i	Direction vector	[m]
\vec{x}_0	Starting vector	[m]
\vec{g}	Gradient function	[m]

superscripts

-	time average
---	--------------

Chapter 1

Problem Definition

A global drive towards renewable energy has bolstered wind turbine development and use. In 2016 a total of 54.6 GW of wind power was added globally to the supply of electricity. South Africa has reached the point of implementing wind energy on a commercial scale (Global-Wind-Energy-Council, 2017) and the Integrated Resource Plan is set to install 9 GW of capacity by the year 2030. According to the Independent Power Producers Procurement Program Unit (2017), the Renewable Energy Independent Power Producers Procurement Program (REIPPPP) has secured 1.4 GW of operational capacity for South Africa, with a further 1.9 GW already procured. All the projects to date have been onshore wind farms, but projections for the industry going forward indicate offshore farms as the growing sector.

Horizontal and vertical axis wind turbines represent the two categories of wind turbines. The dominant design for commercial scale energy production is the horizontal axis wind turbine (HAWT). However, according to Musgrove (1987) when wind farms go offshore the vertical axis wind turbine will play a more prominent role, due to their ability to scale up easier than a HAWT. More recently, Bravo *et al.* (2007) discussed the application of small wind turbines in an urban environment. These studies show the vertical axis wind turbine (VAWT) can play a role on both the commercial and private scale for electricity generation.

The merits of each type of turbine are summarized in chapter 2. The variation in torque transmitted through the power train of a wind turbine over time, termed as the torque ripple, is present in both horizontal and vertical axis wind turbines. In the former these fluctuations are predominantly caused by wind shear and tower shadow. In VAWTs the constantly changing angle of attack results in a prominent torque ripple. Such fluctuations reduce the life cycle of the equipment due to fatigue and add to poor power quality, like flicker and harmonics on the line (Reuter and Worstell, 1978; Sutherland *et al.*, 2012). Torque ripple during operation is highlighted as an obstacle in the deployment VAWTs and this work aims to address the issue by attempting to minimize these fluctuations through the implementation of a novel blade design.

1.1 Objectives

When connecting a wind farm to other power generators on a grid there are voltage quality concerns. There are international standards ensuring that the wind turbine output falls within acceptable levels. The international standard regarding wind turbines (IEC 61400-21) provides manufacturers with data on power quality characteristics. One of the factors mentioned in the IEC 61400-21 is the effect of voltage fluctuations and flicker. Voltage fluctuations may cause changes in the luminance of lighting and can be measured with a flicker meter. The voltage fluctuations are caused by changes in load or generation.

Looking at the turbine itself, large variations in torque also have a negative effect on the fatigue life of drive components such as shafts, couplings and transmissions. This could result in over-designed drive trains or equipment with reduced life spans. With this in mind the following research question was posed:

Can a blade with morphing capabilities reduce the alternating torque experienced by a straight-bladed vertical axis wind turbine during operation?

This work focused on a numerical study to design a morphing foil inspired by nature. The drag minimization concepts of boundary layer control as seen in swimming dolphins (Fish and Hui, 1991) and present on humpback whale pectoral fins (Rostamzadeh *et al.*, 2017) were applied to a morphing foil concept. The milestones used to answer the research question are given in the following sections.

1.1.1 Analytical model

An analytical VAWT model was implemented in open source software to provide torque as function of azimuthal position for various operating conditions. The VAWT characteristics during operation were based on the double multiple stream tube model developed by Paraschivoiu (1981), which uses the lift and drag information of the VAWT blades. The model was validated through comparison with published data.

1.1.2 Blade characteristics

The proposed blade had multiple configurations and therefore required a lengthy process to determine all the possible lift and drag combinations. A virtual laboratory was used to reduce the design time, through parallel computing. Computational fluid dynamic (CFD) modelling was used to obtain the lift and drag curves of the morphing foil. Multiple curves were generated for various Reynolds numbers and foil shapes. A transitional turbulence model was selected because of its ability to capture the tripping of the boundary layer. The

CFD model was compared to published experimental data, to ensure it was correctly implemented.

1.1.3 Optimized operation

To make use of the morphing nature of the blade an optimization routine was applied to identify the airfoil shape at various stages of a single VAWT rotation. Using the CFD data a surrogate model was created to provide lift and drag data as input to the analytical model. The surrogate model was essential for optimisation because of its stability and fast execution. The analytical model was optimised for a reduced torque ripple while maintaining a high power coefficient.

1.2 Report outline

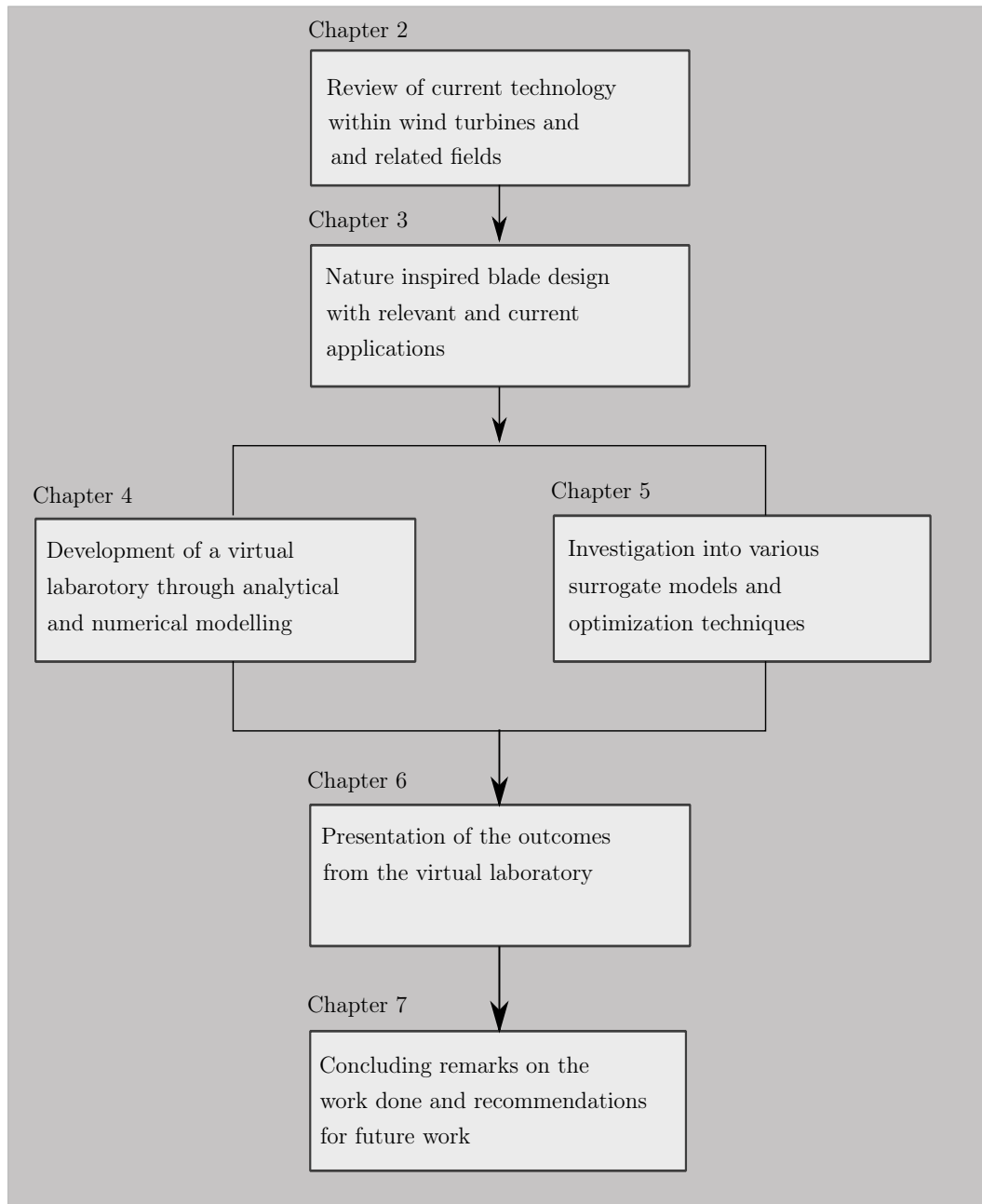
This document was written in a concise format. The fundamental ideas and theory behind turbulence modelling, CFD and surrogate modelling are not revised. Only key concepts in each field are discussed to provide insight and explain the reason for certain choices made through the study. Background information on the relevant topics is included in the appendices to provide the reader with additional information.

Chapter 2 consists of a literature review on the current landscape of wind turbine technology. This chapter also provides information on boundary layer manipulation in airfoil design and applications.

The inspiration behind the novel blade and its implementation is given in chapter 3. Chapter 4 discusses the numerical modelling undertaken during this work. This includes an analytical model for a VAWT and a discussion on the choice of turbulence model used in the CFD simulation. Details on the morphing capabilities of the foil and steps taken to correctly model it in CFD are also presented.

Chapter 5 serves as motivation for the choice of surrogate model and optimization technique. It highlights aspects considered during selection and evaluates the applicability of various models. For surrogate models this includes prediction accuracy. The rationale behind the optimisation technique is also provided. The results of optimisation performed on the surrogate model and verification through CFD modelling are provided in chapter 6. This chapter expands on the distortion geometry and operation for a reduced torque ripple during operation.

Finally in chapter 7 recommendations on future work and design approaches are provided. The methodology used in this work is represented visually in figure 1.1.

**Figure 1.1:** Document outline

1.3 Contributing publications

This section outlines the relevant peer-reviewed publications produced during this research. The list includes paper title, co-authors and their format. Additionally the subsections provide the abstract that accompanied each publication.

- A - Erfort, G., von Backström, T.W. and Venter, G. (2017, Published). **Numerical optimisation of a small-scale wind turbine through the use of surrogate modelling.** *Journal of Energy in Southern Africa*
- B - Erfort, G., von Backström, T.W. and Venter, G. (2018 July, Presented). **Fine tuning of the $\gamma - Re_\theta$ turbulence model using historical data sets.** In: Wan, P.D. (ed.), *The proceedings of the 13th OpenFOAM workshop (OFW 13)*
- C - Erfort, G., von Backström, T.W. and Venter, G. (2018, Accepted). **Reduction in the torque ripple of a vertical axis wind turbine through foil pitching optimization.** *Journal of Wind Engineering*
- D - Erfort, G., von Backström, T.W. and Venter, G. (2019, Accepted). **Numerically determined empirical relationships for a transitional turbulence model** *Journal of Applied Fluid Mechanics*

Numerical optimisation of a small-scale wind turbine through the use of surrogate modelling

Wind conditions in South Africa are suitable for small-scale wind turbines, with wind speeds below 7 m.s⁻¹. This investigation is about a methodology to optimise a full wind turbine using a surrogate model. A previously optimised turbine was further optimised over a range of wind speeds in terms of a new parametrisation methodology for the aerodynamic profile of the turbine blades, using non-uniform rational B-splines to encompass a wide range of possible shapes. The optimisation process used a genetic algorithm to evaluate an input vector of 61 variables, which fully described the geometry, wind conditions and rotational speed of the turbine. The optimal performance was assessed according to a weighted coefficient of power, which rated the turbine blades ability to extract power from the available wind stream. This methodology was validated using XFOIL to assess the final solution. The results showed that the surrogate model was successful in providing an optimised solution and, with further refinement, could increase the coefficient of power obtained.

Fine tuning of the $\gamma - Re_{\theta}$ turbulence model using historical data sets

The use of empirical turbulence models has been well documented in computational fluid dynamic simulations. The $\gamma - \tilde{Re}_{\theta_t}$ model, also known as the SST-transition model, proposed by Langtry and Menter, in particular has received much attention for being able to more closely replicate the pressure values on an airfoil surface as seen in experiments. The original empirical relations were based on observations by the authors, but the published relationships were developed to capture multiple geometries and experimental set-ups. This paper discusses an optimisation approach used to alter the empirical relationships to match an existing data set, captured prior to the model's development. Simulations were carried out using the open source CFD package openFOAM (Weller *et al.*, 1998). The new model coefficients are then compared to the standard formulation as well as the shear stress transport $k - \omega$ model ($k - \omega SST$). This work aims to show how the SST-transition model can be adapted for specific geometries using historical data sets.

Reduction in the torque ripple of a vertical axis wind turbine through foil pitching optimization

Vertical axis wind turbines have a place in the small scale renewable energy market. They are not currently implemented on a commercial scale but have found a niche space in urban areas. Here the turbulent wind conditions and limited space are more easily tapped into with a vertical axis wind turbine. However, the challenges facing these types of turbines have hampered deployment. One of these issues is the fluctuating torque experienced during operation, which can lead to over-designed power trains. Genetic and gradient based optimization is applied to an analytical model of a vertical axis wind turbine, in order to reduce the torque fluctuation while attempting to maintain a high power coefficient. The reduction in torque ripple is achieved through a sinusoidal pitching motion of the blades. The torque ripple can be reduced by 10 % with a similar reduction in power coefficient.

Numerically determined empirical relationships for a transitional turbulence model

Turbulence models in computational fluid dynamics (CFD) aim to capture a complex phenomenon through simplified mathematical models. The models themselves range in terms of application, complexity and methodology. This work looked at a transitional model for Reynolds averaged Navier Stokes equations. In particular the focus was on the correlation based intermittency and momentum thickness Reynolds number ($\gamma - \tilde{Re}_{\theta_t}$) model. The original model has high order correlations, that were determined and calibrated from flat plate tests of various pressure gradients. In this work the correlations were simplified

to reduce the number of calibration coefficients and help in understanding the effect of each parameter. Flat plate test data, from the European Research Community on Flow, Turbulence and Combustion (ERCOFTAC) T3A series, were used to verify the lower order approximations through OpenFOAM simulations. The open source CFD package OpenFOAM was used for its easy access to the base code. The reduced order model was then applied to a National Advisory Committee for Aeronautics (NACA) 0012 foil at a transitional Reynolds number of 360 000 as a means of validation. The reduced order, the original $\gamma - \tilde{R}e_{\theta_t}$ and the fully turbulent $k - \omega$ shear stress transport ($k - \omega$ SST) turbulence models are compared over a range of angles of attack to highlight the difference between models. The proposed model reduced the runtime of simulation by approximately 6%. The reduction in model coefficients meant a step by step adjustment could be implemented to increase model accuracy. In addition the adjusted model increased the accuracy of drag prediction on a NACA0012 airfoil, while maintaining a similar lift prediction as the original.

Chapter 2

Literature Review

Both VAWTs and HAWTs have been developed during the same time period with the latter eventually being favoured. Eriksson *et al.* (2008) cite Brothers (1998) in saying that the choice of HAWT over VAWT can be attributed to random selection in the early days of large scale development. Alternatively it could be argued that the HAWT was self starting and more efficient than the VAWT and thus for early development was the turbine of choice. In any event, encouraged by more funding, the study of HAWT technology advanced beyond its VAWT counterpart. According to Spera (1994) VAWTs can benefit most from improved aerodynamic performance and a reduction in manufacturing costs.

A VAWT operates in winds coming from any direction, but in doing so the blades experience a change of angle of attack (α) during rotation. The VAWT angle of attack is a function of both wind speed and azimuth angle (θ). Rotation can be divided into upstream and downstream halves. The upstream portion experiences undisturbed oncoming wind, while the downstream half sees a slower wind. This is because the flow has already been affected by passing through the upstream half of rotation. Due to the nature of α and different flow conditions between the upstream and downstream halves, the torque output is sinusoidal and referred to as a torque ripple. For power generation the torque input is ideally constant with negligible fluctuations. The following chapter discusses the different types of wind turbines, analytical models for VAWTs and a review of the technological trends in blade design.

2.1 Vertical axis wind turbines

Vertical axis wind turbines are so called because they rotate about a vertically orientated axis. This axis or tower is then connected to a generator either through direct drive or gearboxes. While many configurations are possible a VAWT is classified as either a Savonius or a Darrieus type. The Savonius turbine is momentum based and rotation is achieved by the change in momen-

tum between oncoming air and the paddle. A Darrieus turbine is lift based, where air flowing past the blade results in a lift force that causes rotation. Figure 2.1 shows a Savonius type while figure 2.2 provides examples of the Darrieus configuration

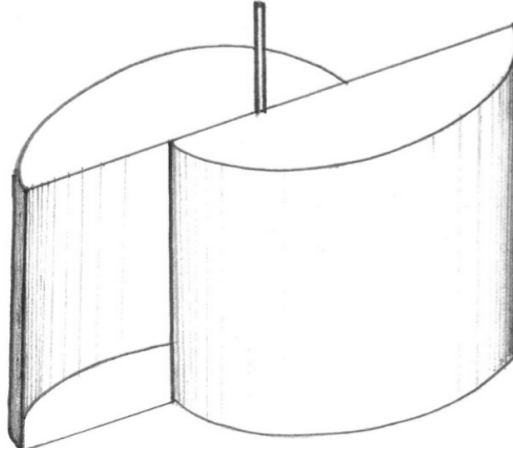


Figure 2.1: Basic concept behind the Savonius turbine taken from Islam *et al.* (2008)

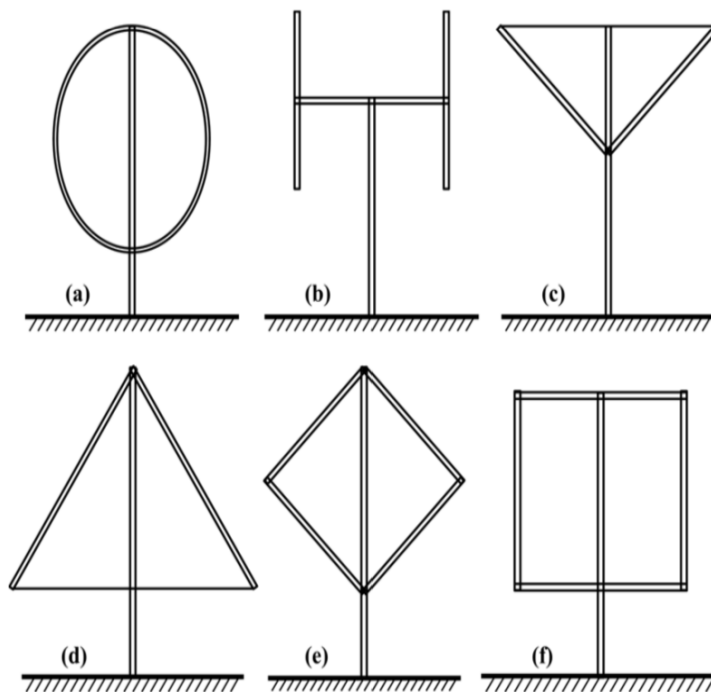


Figure 2.2: Typical Darrieus configurations: (a) Full Darrieus, (b) H-Rotor, (c) "V", (d) Delta, (e) Diamond, (f) Giromill taken from Sutherland *et al.* (2012)

As wind power moves offshore the turbines are growing larger and the HAWT is less suited for such operations. VAWTs are the more viable option for offshore wind farms, because of their design (Paquette and Barone, 2012). This is elaborated on in detail in the next section.

2.2 Notable differences

There are a few fundamental differences in the way vertical axis and horizontal axis turbines are constructed and operated. The various differences are highlighted in table 2.1 and expanded upon thereafter.

Table 2.1: Differences between the vertical axis and horizontal axis wind turbines

Factor	VAWT	HAWT
Wind direction	Omni-directional	Head on
Wind conditions	Varying	Consistent
Machinery location	Base	Tower top
External aid	Yes	No
Blade	Simple to complex	Detailed and complex
Forces	Tension, fatigue and bending moments	Root bending moments, fatigue
Torque	Sinusoidal	Constant
Noise	Quiet	Noticeable
Energy conversion effectiveness	\lll Betz limit	75-80% of Betz limit

A VAWT accepts wind from any direction, eliminating the need for a yaw system as is present on the HAWT. This also means it has more placement options, such as areas with varying wind directions and odd gusts. The generator for the VAWT is located at the base of the structure which has knock-on effects for its design as well as the tower design. The generator can be designed for efficiency over space concerns while the tower no longer needs to support a top heavy design as in the case of a HAWT. Additionally the lower centre of gravity in a VAWT makes it attractive in offshore wind farms using floating pedestals. A direct drive gearbox and generator is possible with VAWTs as their size is no longer an issue, as in the HAWT nacelle construction. All of these machinery concerns also play a part in planned maintenance which is again easier in the case of a VAWT.

The blade design of a HAWT is no simple due to the changing airfoil shapes in the span wise direction. This is made additionally harder with the

adjustable pitch of the blades. The full Darrieus turbine has curved blades which in their own right are complex in construction. The long slender blades are more easily damaged in transport. The simpler H-rotor blades usually have a single profile along their entire length but are also fragile in transport.

There is stress at the blade root of a HAWT due to the cantilever design. The design results in root stresses due to the changing direction of blade weight vector relative to the structure (blade rotation) and aerodynamic loading. This stress is a major contributor to the size limitations of current HAWTs. The VAWT does not suffer the same issue but it has its own stresses to deal with. The H-rotor blade has periodical loading due to wind shear and bending moments as a result of the centrifugal forces. Both turbines have bending moments in their blades, limiting their size, but to different degrees. The cantilever designed VAWT, with spindle towers, do have stresses in the tower base but these are due to the cyclic aerodynamic forces, whereas the HAWTs have periodic stresses due to gravity.

The HAWT provides a relatively constant torque but the VAWT has a problematic torque ripple. The ripple is the result of the changing angle of attack due to azimuth angle and by the downstream blades operating in lower energy wind.

HAWTs are self starting while VAWTs need external assistance to reach their design tip speed ratio. HAWTs use blade pitching for power control; this is not the case in Darrieus turbines. VAWTs are self limiting with a peak power production at design point and a reduction in power as wind speed passes this point.

The VAWT is seen as a less noisy turbine, because of the location of the machinery and the fact that the noise is directly related to tip speed ratio. The tip speed ratio of a VAWT is lower than that of a HAWT and the generator of the VAWT is on the ground reducing the distance the sound can travel (Eriksson *et al.*, 2008).

The Betz limit is the theoretical maximum of kinetic energy that a wind turbine can extract from a stream tube equal in size to the rotor swept area. The HAWTs in use today are much closer to this limit than any VAWT in the current market.

In summary the better efficiency and self starting nature of the horizontal axis turbine allowed for faster development and gains in the commercial sector. As wind farms move offshore (Bilgili *et al.*, 2011) the turbines are expected to grow in size, beyond the limits of the HAWT, and according to Siddiqui *et al.* (2015) this is where VAWTs can play an important role. It seems VAWTs can be used on a large scale offshore or on small scale in urban environments. Their versatility in application is hindered only by the lack of study in the field.

2.3 Aerodynamics in wind turbines

Broadly speaking the VAWT analytical models can be divided into two groups; the vortex and actuator disk models. The latter is further divided into a cascade approach or the momentum method. Each of these models is reviewed for its merits and appropriateness for the application being considered.

2.3.1 Vortex model

This model makes use of potential lines that determine the velocity field around the turbine as influenced by the vorticity in the wake. Vortex filaments, seen in figure 2.3, are determined from airfoil coefficients, relative flow velocity and angle of attack. This model is computationally expensive compared to others.

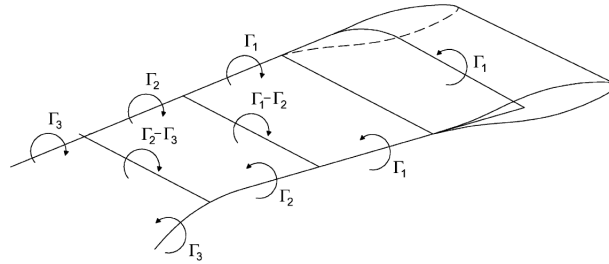


Figure 2.3: Vortex filament representation for a single blade element taken from Islam *et al.* (2008)

2.3.2 Actuator disk models

Figure 2.4 is an illustration of the actuator disk model used for wind turbine modelling. This model assumes the following:

- Homogeneous, steady state, incompressible flow
- No drag due to friction
- Infinite number of blades
- Uniform thrust of the disk area
- Non-rotating wake
- Pressure far upstream and far downstream are equal to ambient pressure

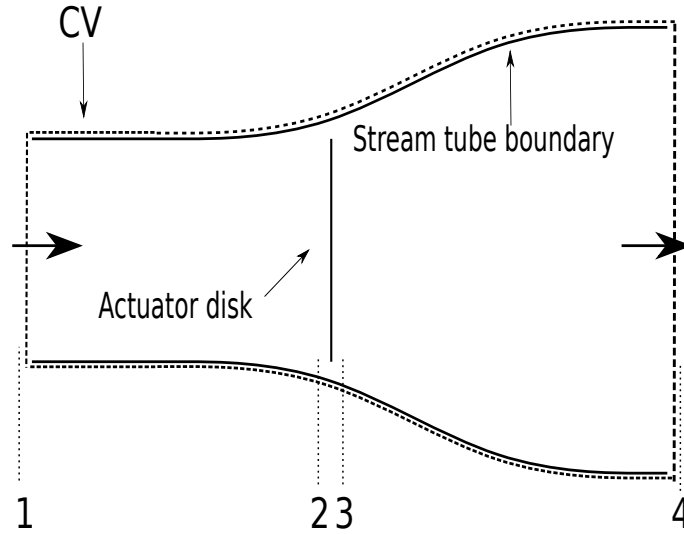


Figure 2.4: Stream tube model for wind turbine with enclosing control volume

The disk creates a pressure discontinuity in the stream tube at the actuator disk. Thrust is then the stream wise force as a result of the pressure drop

$$T = \Delta p A \quad (2.3.1)$$

The velocity behind the disk is less than upstream since the thrust is positive. Applying Bernoulli's equation before and after the disk gives:

$$p_1 + \frac{1}{2}\rho V_1^2 = p_2 + \frac{1}{2}\rho V_2^2 \quad (2.3.2)$$

$$p_3 + \frac{1}{2}\rho V_3^2 = p_4 + \frac{1}{2}\rho V_4^2 \quad (2.3.3)$$

We can then express the pressure drop at the disk as

$$\Delta p = \frac{1}{2}\rho(V_1^2 - V_4^2) \quad (2.3.4)$$

Using the conservation of momentum and the control volume in figure 2.4 we can derive another expression for T

$$T = \rho V_2 A (V_1 - V_4) \quad (2.3.5)$$

Using equations 2.3.1, 2.3.4 and 2.3.5 allows an expression for wind velocity at the blade

$$V_2 = \frac{V_1 + V_4}{2} \quad (2.3.6)$$

Defining an induction factor a as the fractional decrease in wind velocity between upstream and the rotor:

$$a = \frac{V_1 - V_2}{V_1} \quad (2.3.7)$$

Finally with equations 2.3.6 and 2.3.7 the velocity downstream can be defined in terms of the upstream velocity and the induction factor

$$V_4 = (1 - 2a)V_1 \quad (2.3.8)$$

Through appropriate substitution for velocities and the use of equation 2.3.5 an expression for the coefficient of thrust can be defined, as

$$C_t = \frac{T}{\frac{1}{2}\rho AV^2} \quad (2.3.9)$$

Expressions for the lift and drag coefficients can be defined in a similar fashion and are written below:

$$C_l = \frac{L}{\frac{1}{2}\rho AV^2} \quad (2.3.10)$$

$$C_d = \frac{D}{\frac{1}{2}\rho AV^2} \quad (2.3.11)$$

2.3.2.1 Cascade model

Taken from the field of turbo machinery, here the regular spacing of the blades is used to re-imagine the turbine in a single plane, as shown in figure 2.5. Wake and free stream velocity relationships are determined by the Bernoulli equation. Each element of the blade has aerodynamic characteristics, obtained through the actuator disk model. The flow conditions of the reference blade, labelled 1 in figure 2.5, are assumed to be the same on blades 2 and 3. This process is repeated for all azimuthal positions.

This model is capable of handling high tip speed ratios (TSR) and solidity, and its accuracy is improved by including dynamic stall effects and flow curvature.

2.3.2.2 Momentum models

These models equate the aerodynamic force on the blades with the rate of change of momentum of the air. The momentum equations are inadequate for large TSR, making the models unsuitable at this range (Islam *et al.*, 2008). Originally the proposed model was based on a single streamtube for the entire turbine, but to improve its accuracy this was replaced with multiple streamtubes. These tubes would encapsulate an entire blade at various azimuthal positions. To accurately capture the upstream and downstream interaction Paraschivoiu (1981) proposed the double multiple stream tube model (DMST), illustrated in figure 2.6. Here the stream tubes are divided between up and down stream sections with the upstream tube exit velocity forming the inlet velocity for the downstream tube. This model provided a better correlation for the local aerodynamic blade forces as seen in experiments.

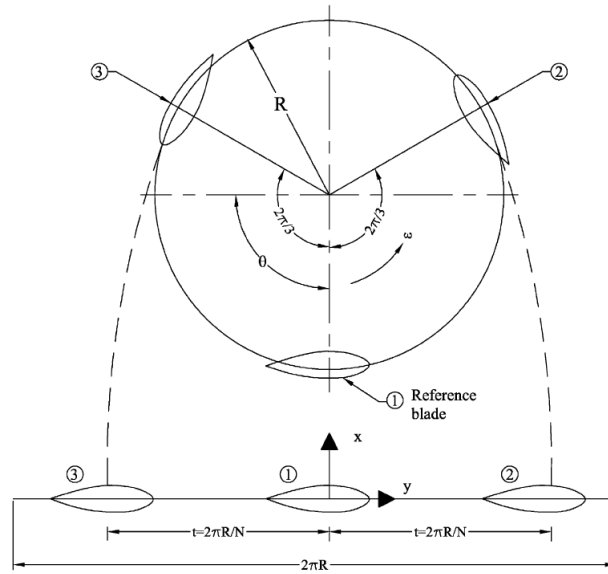


Figure 2.5: Inline cascade representation from Islam *et al.* (2008), blades are projected onto a single plane

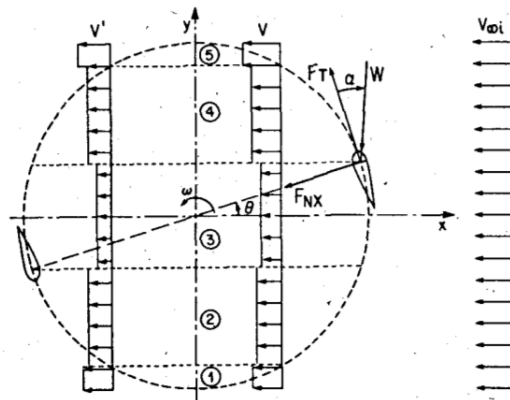


Figure 2.6: DMST model showing upstream and downstream interference factors taken from Paraschivoiu and DELCLAUX (1983)

2.4 Aerodynamic response

A wind turbine, based on its geometry and the wind conditions, has an optimal TSR where it extracts the most power. A HAWT is designed to experience a uniform loading along the entire blade span for a range of tip speed ratios. A VAWT blade sees a change in angle of attack as it rotates and this results in a variation in the torque it produces. By changing the forces seen on the blade the torque output can be controlled.

Studies have been conducted on devices or methods that can be used to alleviate load on wind turbine blades (Wilson *et al.*, 2009; Griffin and McCoy,

2008; Van Dam *et al.*, 2007; Mayda *et al.*, 2005; Krawczyk *et al.*, 2013; Berg *et al.*, 2009; Beyene and Peffley, 2007). The load alleviation is done by altering the aerodynamic response of the system for a given flow field. According to Griffin and McCoy (2008) the devices investigated in the aforementioned papers can be split into two categories :

1. Those affecting camber changes
2. Those affecting the boundary layer around an airfoil

Griffin and McCoy (2008) looked at active aerodynamic devices and a controlled retractable blade rotor, which showed cost saving implications. Wilson *et al.* (2009) investigated the performance of several types of trailing edge devices (micro-tabs, morphing trailing edges and conventional trailing edge flaps) and showed that in combination with current pitch control methodologies, advanced independent flap control is useful in load reduction. Many such devices have been suggested but some require new control techniques or are impractical due to their energy requirements (Van Dam *et al.*, 2007).

The following sections explain the workings of both methods for adapting the aerodynamic response curves in detail.

2.4.1 Camber changes

Camber changes can be brought about through the following devices:

- Flaps
- Ailerons
- Gurney flaps
- Trailing edge wedges
- Micro tabs
- Morphing wings
- Circulation control

Camber changes shift the whole C_l vs α curve upwards and left, illustrated in figure 2.7, providing more lift at lower angles and increasing the maximum lift coefficient.

When these devices are enabled, an increase in lift coefficient is accompanied by an increase in the drag coefficient. When the device causes a decrease in lift there is an associated decrease in drag but by a smaller margin than seen with the increased lift configuration (Griffin and McCoy, 2008). Flaps and the likes have been studied in depth, while micro tabs and morphing wings have only recently been discussed in literature.

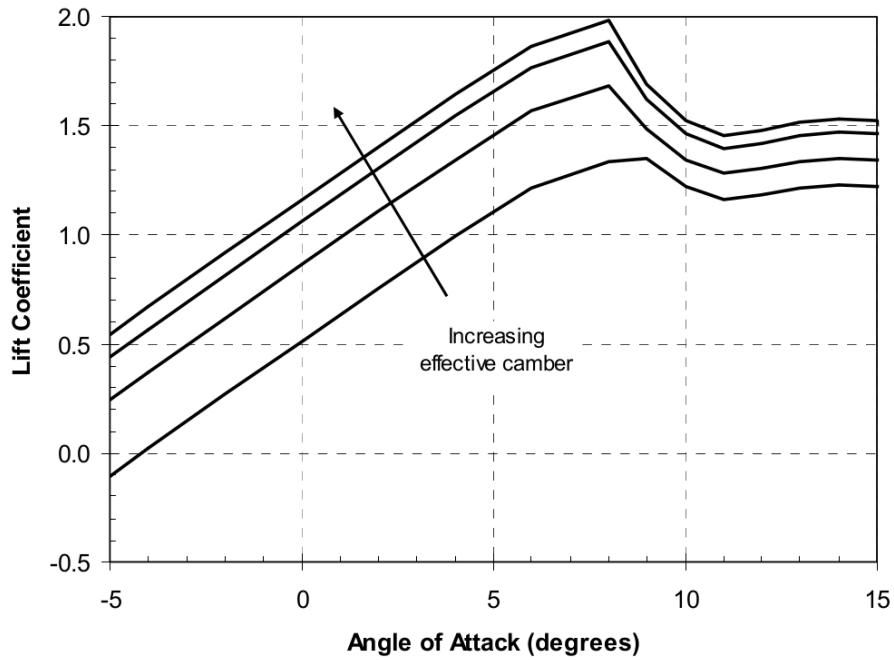


Figure 2.7: Angle of attack changes with changes to the camber taken from Griffin and McCoy (2008)

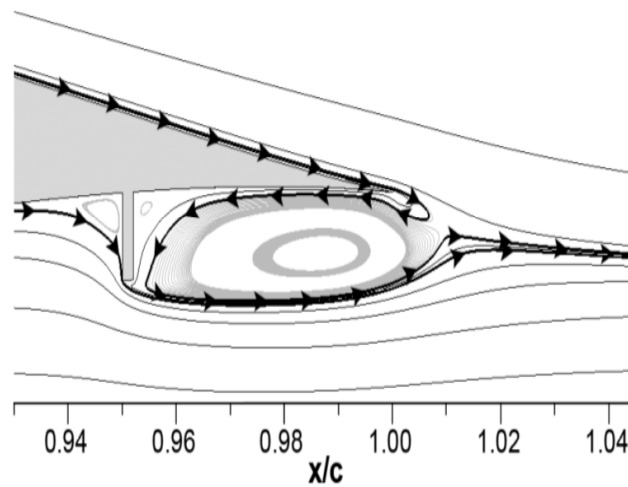


Figure 2.8: A micro-tab deployed at the trailing edge affects the flow patterns of the foil taken from Van Dam *et al.* (2007)

Micro tabs are usually on-off devices located at the trailing edge of an aerofoil. Figure 2.8 shows a deployed tab and its influence on the flow. The tab has a height approximately equal to the boundary layer thickness. It shows significant increase in C_l and relatively small increases in C_d when placed and sized appropriately (Wilson *et al.*, 2009; Van Dam *et al.*, 2007). Mayda *et al.*

(2005) conducted a computational study into micro tabs with perforations, serrations and spacing as variables, showing that a change in solidity ratio of the tab would allow for control of the loading on a blade.

2.4.1.1 Morphing foil

The term morphing is used when describing the change in shape of a continuous surface, such as the morphing skins discussed by Thill *et al.* (2008) for aerodynamic applications. The paper looked at materials, structures and concepts that could achieve deformation for aircraft. The reasons for development of a morphing foil stem from the simple idea that if a foil can adapt its characteristic lift to drag ratio, for a change in Reynolds number, the overall turbine would increase in efficiency (Beyene and Peffley, 2007). Lachenal *et al.* (2013) discussed various approaches to morphing a blade while Beyene and Peffley (2007) looked at the importance of material selection associated with a morphing blade. The latter also discussed the drawbacks of previously suggested design enhancements. The drawbacks included the limited effect on the partial load condition and the negative effect on peak efficiency. A fluid structure interaction simulation on the effects of a morphing wing versus a traditional blade was conducted by Krawczyk *et al.* (2013). This was done to see if the range over which the turbine blade operates could be expanded. The paper showed a morphing blade to be more efficient at part loads but its flexibility hampered performance at the design point, corroborating the sentiment of Beyene and Peffley (2007).

2.4.1.2 Circulation control

This form of camber control is best highlighted in figure 2.9 which indicates how the streamlines passing over the upper half of a foil become entrained by a jet stream coming from the trailing edge.

The effect is to cause the flow to behave as if the camber line of the foil has been altered. In the case of circulation control it uses the Coanda effect to manipulate flow at the trailing edge (Shires and Kourkoulis, 2013).

2.4.2 Boundary layer alterations

The idea with devices for boundary layer control is to adjust the kinetic energy in the boundary layer and thereby delay separation. These devices include but are not limited to:

- Vortex generators
- Leading edge slats
- Trip slats

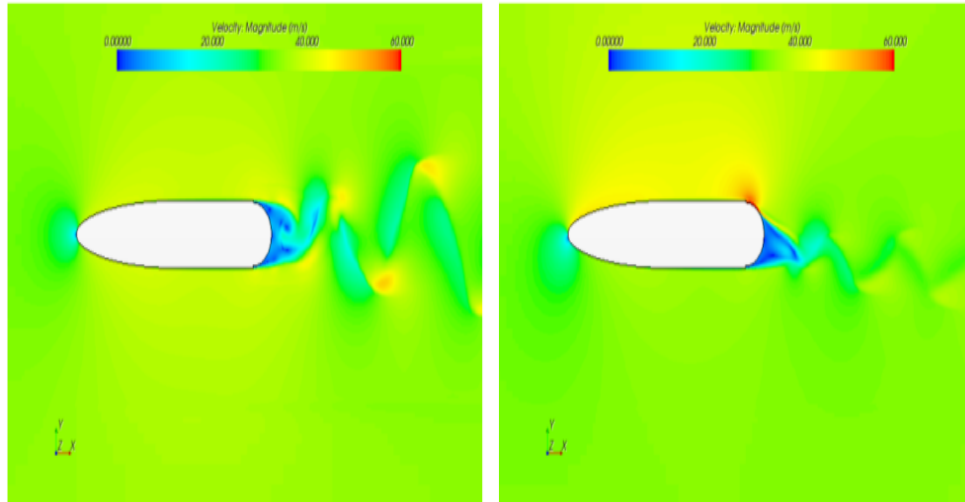


Figure 2.9: No applied jet vs applied jet stream taken from Shires and Kourkoulis (2013)

The effect is seen at high angles of attack and is not evident in the linear portion of the curve. Figure 2.10 shows how the addition of boundary layer augmentation increases the maximum lift coefficient.

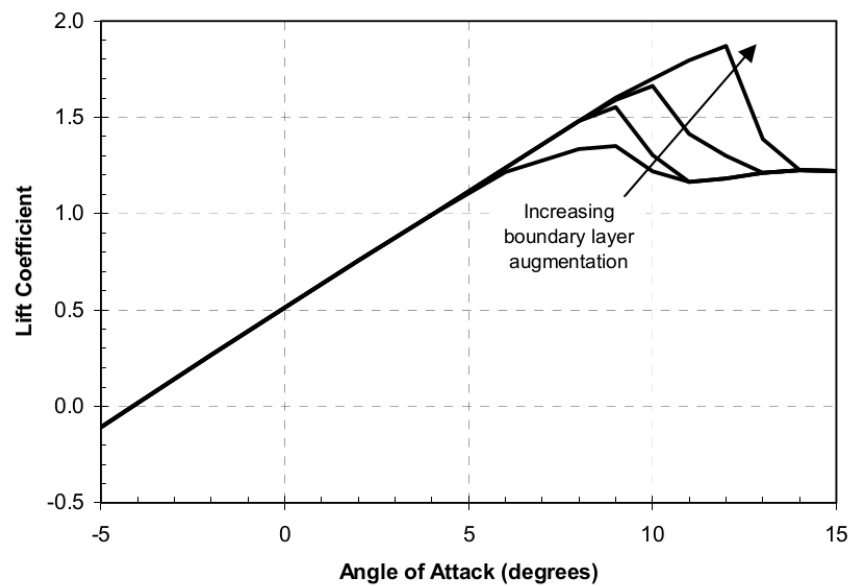


Figure 2.10: Angle of attack changes with change to boundary layer taken from Griffin and McCoy (2008)

There is an opportunity for these techniques to reduce loads but it will be highly complex due to the inherent non-linearity. Alternatively the blade

could be redesigned to operate closer to stall to better take advantage of these devices. Designing the blade for operation in this region may lead to poor aerodynamic efficiency and reduce the energy capturing capabilities. Variable speed and pitch turbines have negligible gains from boundary layer changes, as they operate within the linear region of figure 2.10, and these effects are more prominent post stall.

2.5 Load reduction in turbines

For the micro tabs of a height approximately equal in boundary layer thickness there is a noticeable difference in the C_d and C_l curves. Figure 2.11 shows the effect on these curves as reported in experiments by Wilson *et al.* (2009). A micro tab on the pressure side of the airfoil provides increased lift and drag for positive angle of attack. In negative angles these tabs reduce the lift available, when compared to a nominal airfoil. When placed on the suction side of the airfoil the tabs also increase the drag effect but by a smaller margin than seen on the pressure side.

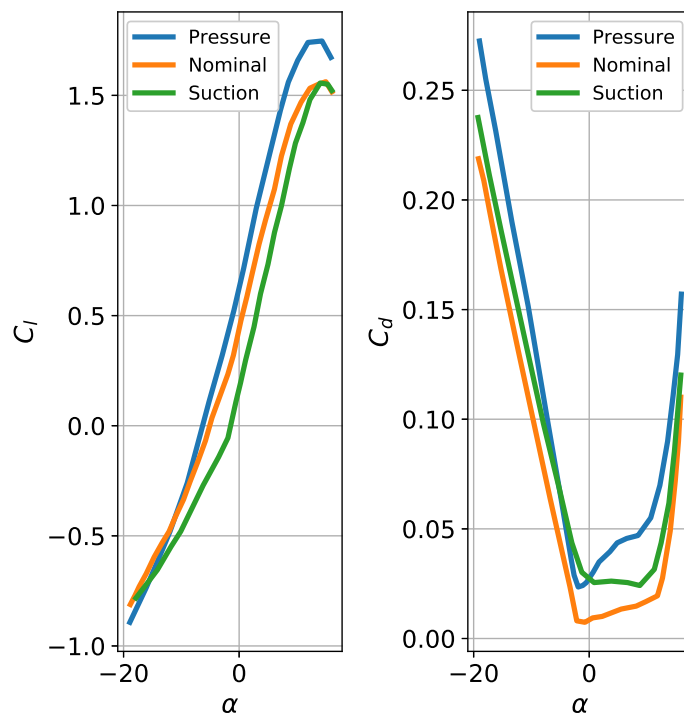


Figure 2.11: Lift and Drag curves for Microtabs deployed on suction and pressure side of foil recreated from (Wilson *et al.*, 2009)

The adjusted curves would then translate to an adjusted torque output, but exact data was not provided in the article reviewed.

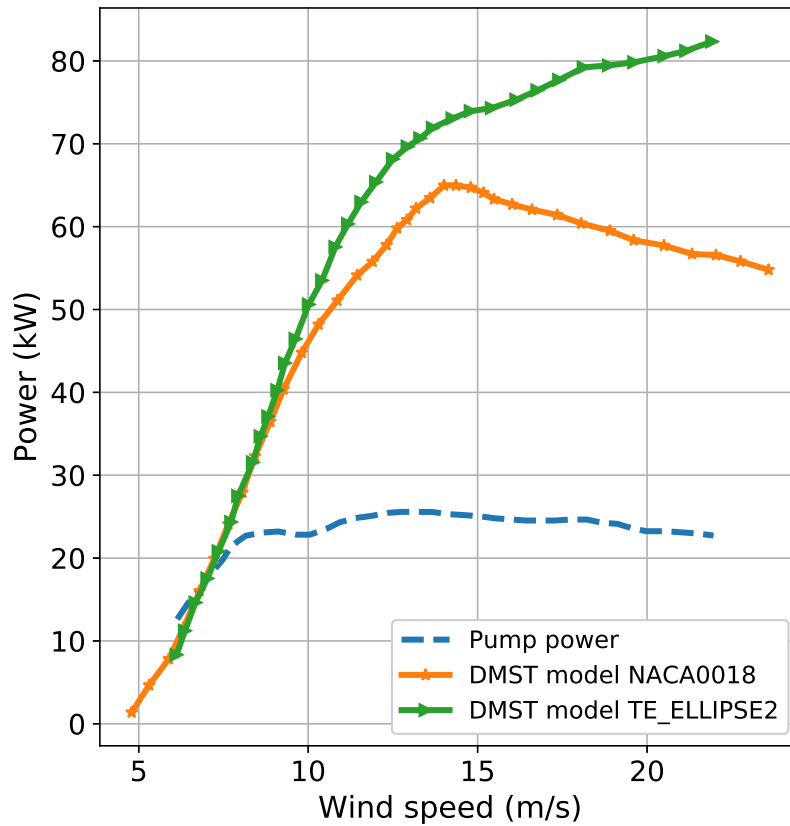


Figure 2.12: Power input and resultant power for a circulation controlled system recreated from Shires and Kourkoulis (2013)

Shires and Kourkoulis (2013) investigated circulation control to find out how much power the jet nozzle would deliver to the system and what the corresponding increase in power output would be. The results are shown in figure 2.12 where it can be seen that for approximately 20 kW input the power output only shows an increase close to 10 kW. The system makes an apparent net gain in wind speeds greater than 20 $\text{m}\cdot\text{s}^{-1}$ where the information was not accurate.

2.6 Surface roughness

According to Bons *et al.* (2001), typical production line standards for wind turbine components ensure a surface roughness less than 1 μm . The technical report by Jones and Williams (1936) explained that for the NACA foil 0012,

surface roughness on the scale ($10\ \mu\text{m} - 25\ \mu\text{m}$) has a large effect on maximum lift and drag values at high Reynolds numbers. In particular after a certain Reynolds number the maximum lift remains almost constant, as seen in figure 2.13, in comparison to the non-roughed foil which has an increasing trend.

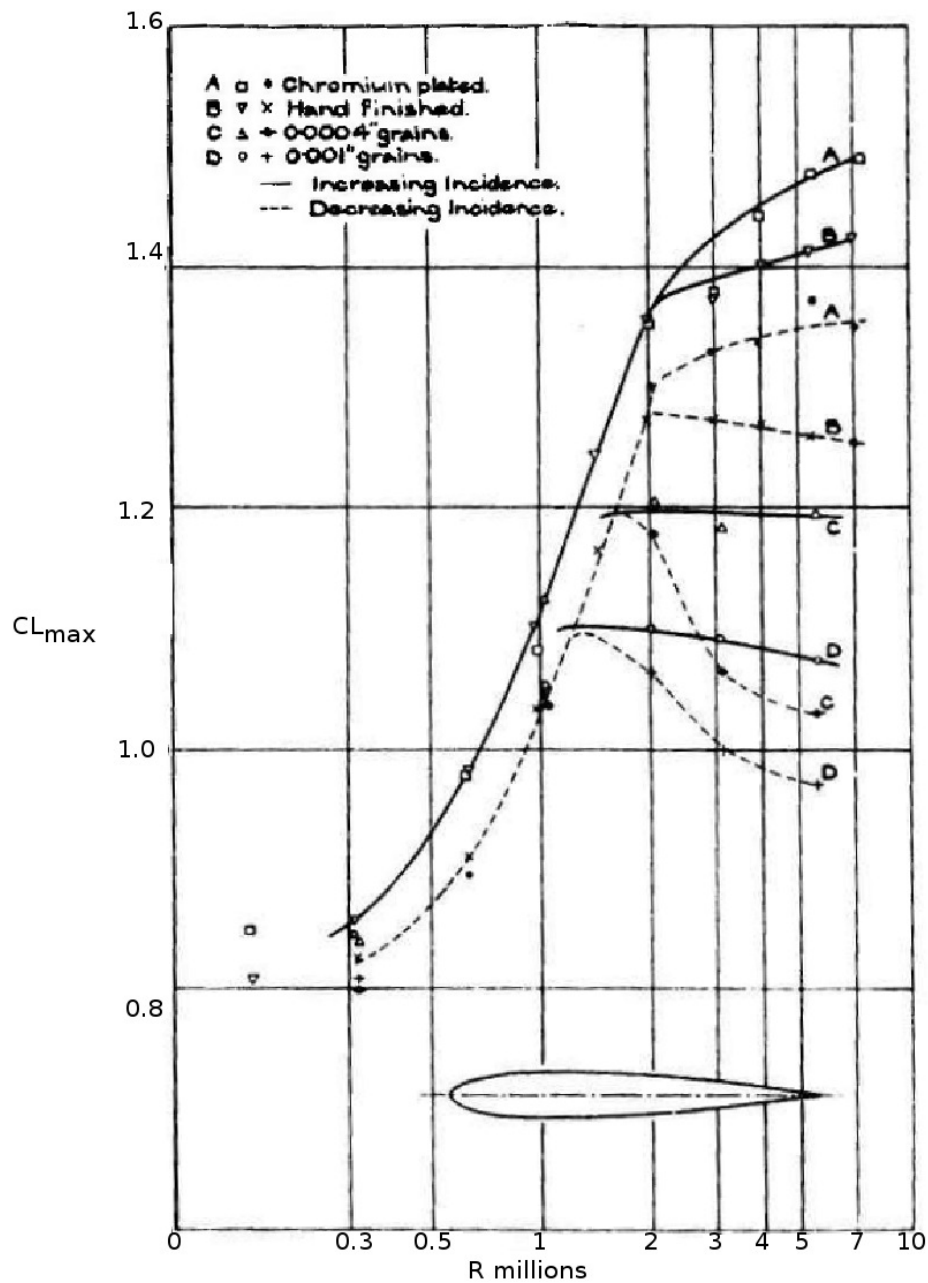


Figure 2.13: Maximum lift coefficient for a NACA 0012 foil with various finishes as per Jones and Williams (1936)

Back *et al.* (2012) report that surface roughness, on the scale of μm , has been shown to affect the performance (in terms of loading and loss) of cascade compressors. The article presented data describing the effect of roughness location and severity.

Chakroun *et al.* (2004) did a study on the surface roughness effects on symmetrical foils. In general increases in roughness showed a decrease in lift and increase in drag coefficients. The tests were conducted on roughness from $200\ \mu\text{m}$ up to $2\ \text{mm}$ on various locations of the foil. Most notable differences in lift and drag were seen at low angles of attack and the $2\ \text{mm}$ roughness on the leading edge (LE) showed a delay in the onset of stall. Howell *et al.* (2010) show that surface roughness could actually improve the performance coefficient (C_p) of small VAWTs at low tip speed ratios. The effect on the performance coefficient curve for both rough and smooth surfaces is presented in figure 2.14.

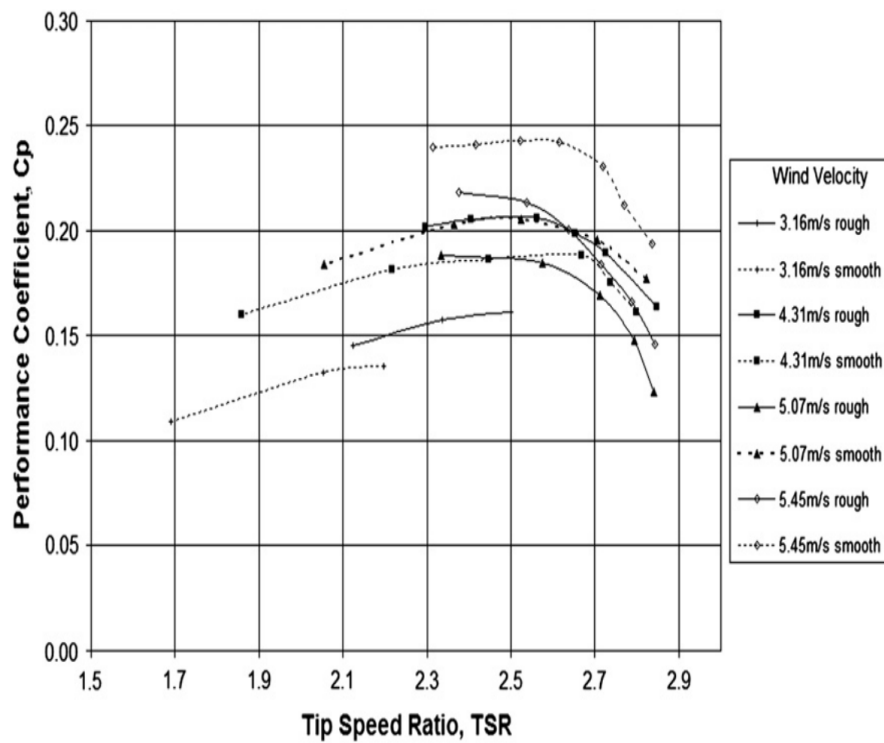


Figure 2.14: Improvement in C_p due to surface roughness of a 2 bladed VAWT, taken from Howell *et al.* (2010)

Summary

There are multiple analytical models for VAWTs, each with their own merits. The DMST model used in this work was selected for its ability to closely predict the aerodynamic loading on the blade of a turbine. A review of the current trends in morphing foils indicated minor adjustments, within the boundary layer, could delay stall and alter the lift and drag curves of a foil. The alteration of the boundary layer characteristics through minimal shape change was of particular interest. Surface roughness studies showed that changes on the millimetre scale could affect the lift and drag curves in multiple ways and thus influenced the morphing capabilities examined in this study.

Chapter 3

Blade Design

3.1 Biomimicry

The blade presented here, for use in the VAWT is inspired by drag reduction techniques in nature, in particular the mechanism used by swimming dolphins to reduce their drag. In Fish and Hui (1991) the various myths and proposals surrounding these techniques are discussed. In Gray (1936), the theory of a turbulence free boundary layer was postulated as the reason for dolphin swimming efficiency. Attempts to verify the laminar theory included viscous dampening through compliant surfaces, namely that the skin acted to passively damp the Tollmein-Schlichting waves that add turbulent fluctuations (Fish and Battle, 1995). The attempt was ultimately unsuccessful but did manage to reduce the C_f in some instances.

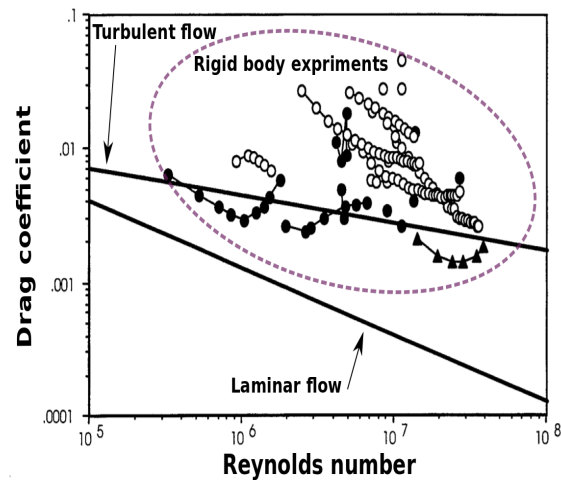


Figure 3.1: Annotated image from Fish and Rohr (1999) showing turbulent versus laminar C_d values

This laminar theory was disputed in Fish and Rohr (1999) through experiments on flat plates and surrogate rigid bodies. Figure 3.1 shows that drag values over a flat plate for turbulent boundary flow more closely match the experimental results for rigid body models of dolphins and other cetaceans. In experiments involving a tripping wire there was no reported difference in drag coefficients for the surrogate bodies. This indicated the presence of a turbulent boundary layer. Such layers are noted to delay separation which, in turn, minimizes drag. In Fish and Battle (1995) the pectoral fins on humpback whales showed that LE tubercles affected lift and drag characteristics during swimming. By inducing a turbulent boundary layer the separation point can be delayed further downstream along an airfoil surface. The mechanism for tripping the boundary layer has been shown to be small adjustments to the surface of a rigid body. Finally, placement of these distortions should be near the leading edge to effect change along the surface of the airfoil further downstream.

3.2 Manufactured distortions

The use of surface distortions for flow manipulation was identified in golf ball design. Alam *et al.* (2010) concluded that the dimpled surface of the ball significantly affects the drag characteristics of the ball. The dimples act to reduce the pressure drag by reducing the wake behind the ball. This is a result of the turbulent boundary layer entraining the flow as it passes the surface of the golf ball. These effects allow the ball to travel farther due to its reduced drag at low speeds.

Figure 3.2 shows the reduced drag for various golf balls in comparison with the smooth surface of a squash ball. Since each manufacture has a specific pattern and arrangement, the effects on the ball characteristics also vary. Transition occurs earlier in golf balls than it does for the smooth squash ball.

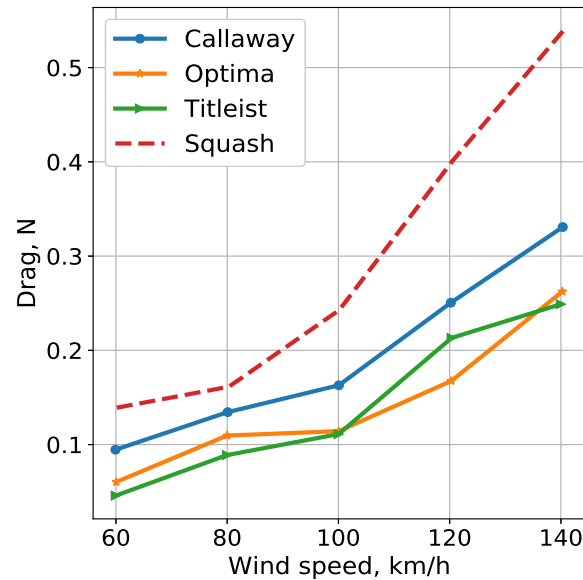


Figure 3.2: Partially recreated image from Alam *et al.* (2010) for drag of various golf balls

3.3 Proposed blade

Studies have shown dolphins induce a turbulent boundary layer to reduce their drag while swimming. Cutaneous ridges have been shown to affect the dynamics of vortex filaments resulting in a reduction of the skin friction coefficient. The humpback whale has tubercles on its pectoral fins that positively influence their swimming ability. Finally the golf ball design includes a tripping mechanism to affect the boundary layer and thus maximise its flight potential. These factors influenced the blade design as follows:

- Implementation of active surface distortion, capable of changing height, based on the control seen in dolphins.
- Placement of the distortion is envisaged for the upper surface of a foil, near the leading edge after investigating the pectoral fins on humpback whales.
- The distortion is shaped as a half circle protruding from the foil surface. The golf ball studies indicated different shapes have different effects and the simple half circle provides a good baseline geometry.

The examples discussed earlier all have multiple distortions over the base geometry. This is a preliminary investigation into the effects of a single adaptive distortion. The use of multiple distortions of fixed height is documented and this study is more concerned with the ability of the distortion to vary in height.

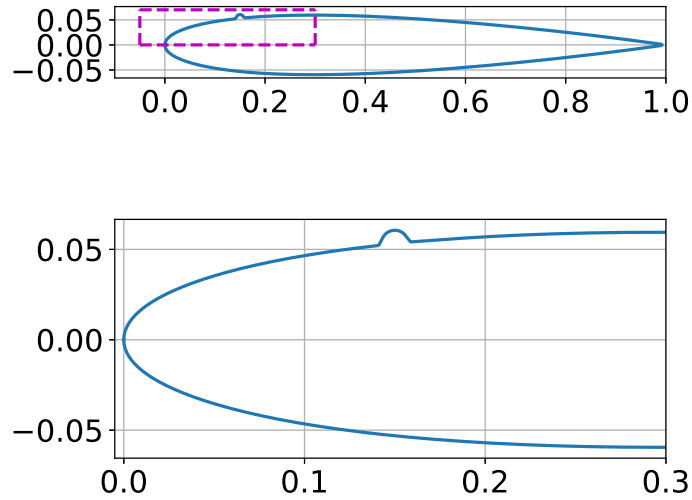


Figure 3.3: The distortion to be implemented on a NACA0012 foil

CFD simulations on a smooth blade showed the transition point to be near the apex for 0° angle of attack at the lowest predicted blade Reynolds number using the analytical model. The transition point moves forward as the angle of attack increases so $0.2X/c$ was set as the upper limit, in terms of location, for the study. Figure 3.3 illustrates the size of a distortion relative to the foil. An airfoil with a chord length of 1 m, a distortion located at 0.15 m from the LE and with a height of 7.5 mm is shown. The magenta box is then focused on in the lower portion of the figure.

A single distortion would be placed between the leading edge and $0.2 X/c$ on the upper surface, the height varies as a function of azimuthal position of the blade. The exact location is unknown and determined through optimization. The variation in height conformed to a sine wave function, whose phase, frequency and amplitude are also determined through optimization. Figure 3.4 shows the expected variation in height as a function of azimuthal position. The function is limited to have a maximum height of half the width of the distortion and can not retract past the foil surface.

The simple sine function was chosen for its simple implementation through a cam following mechanism. The dependence on azimuthal position to determine height introduces a yaw system for the VAWT. Previous work on pitch variation, by Paraschivoiu *et al.* (2009), showed an increase in C_P while Kirke (2011) and Miao *et al.* (2012) showed it could improve the self starting characteristics. All the pitch variation techniques required a yaw system to ensure

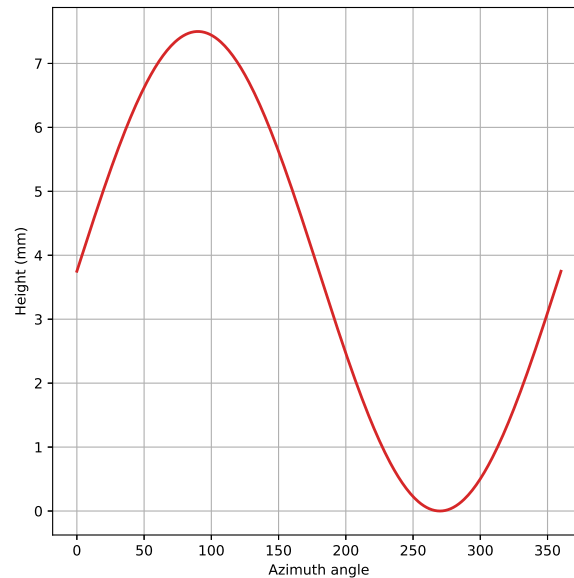


Figure 3.4: The function governing distortion height

the pitch varied according to azimuthal position. These authors have demonstrated that the removal of the omni-directional capabilities of a VAWT is acceptable given an improvement on another design characteristic.

Summary

Nature has provided a minimalist approach to adjust lift and drag characteristics for an aerodynamic body. The use of a turbulent boundary layer to delay separation has proven effective for dolphins and golf balls. Based on the CFD of a smooth foil and influenced by the study on whale pectoral fins, the location of the distortion was bound between the nose and $0.2 X/c$. A simple equation governing the variation in height has been suggested. Requiring the VAWT to implement a yaw system was deemed acceptable given previously published work.

Chapter 4

Numerical Modelling

Published data has shown that pitch variation on a VAWT can achieve both increases in coefficient of power (C_P) and result in self starting, as discussed in section 3.3. Not much work has addressed the torque ripple during operation. A study was conducted to see if pitching the foil during operation could also reduce the torque ripple and results are provided in Erfort *et al.* (2018b). This study showed that gains in ripple reduction were equal to the loss in power coefficient.

As the torque output from a VAWT is directly related to the lift and drag characteristics of the turbine blade, it was then proposed that changes to the blade shape during operation could also have an effect on the torque ripple. Morphing blades were discussed in section 2.4.1.1; however adaptations to the foil were complex. Looking at the dimpled surface of a golf ball, it was postulated that a single adaptive dimple on a foil could result in early tripping of the boundary layer, thus delaying separation. Having control of the separation point through minimal structural changes could then be translated into a customized lift and drag response curve.

Different approaches were investigated to obtain lift and drag values for the new foil. The selection of a suitable numerical tool can greatly affect the design and optimization outcome of a study. While Morgado *et al.* (2016), reported that XFOIL outperformed CFD in overall predictive capabilities, it was found that the shape of the proposed foil was beyond the capabilities of the code.

In the next section details of the turbulence modelling, morphing foil and data collection are discussed. High fidelity CFD modelling was done to obtain lift and drag curves for each foil shape. These curves were in turn used to develop a surrogate model that would be optimised.

4.1 Virtual laboratory

Numerical work was conducted to explore the viability of virtual laboratories. Experimental work is the backbone of engineering and provides the overwhelming majority of data on which research is based. With the advent of high powered computers it is possible to study complex phenomena and large scale problems to a high degree of accuracy. With this in mind it was envisioned that wind tunnel testing would be replaced by CFD. Figure 4.1 provides the work flow for both the experimental route and the numerical approach.

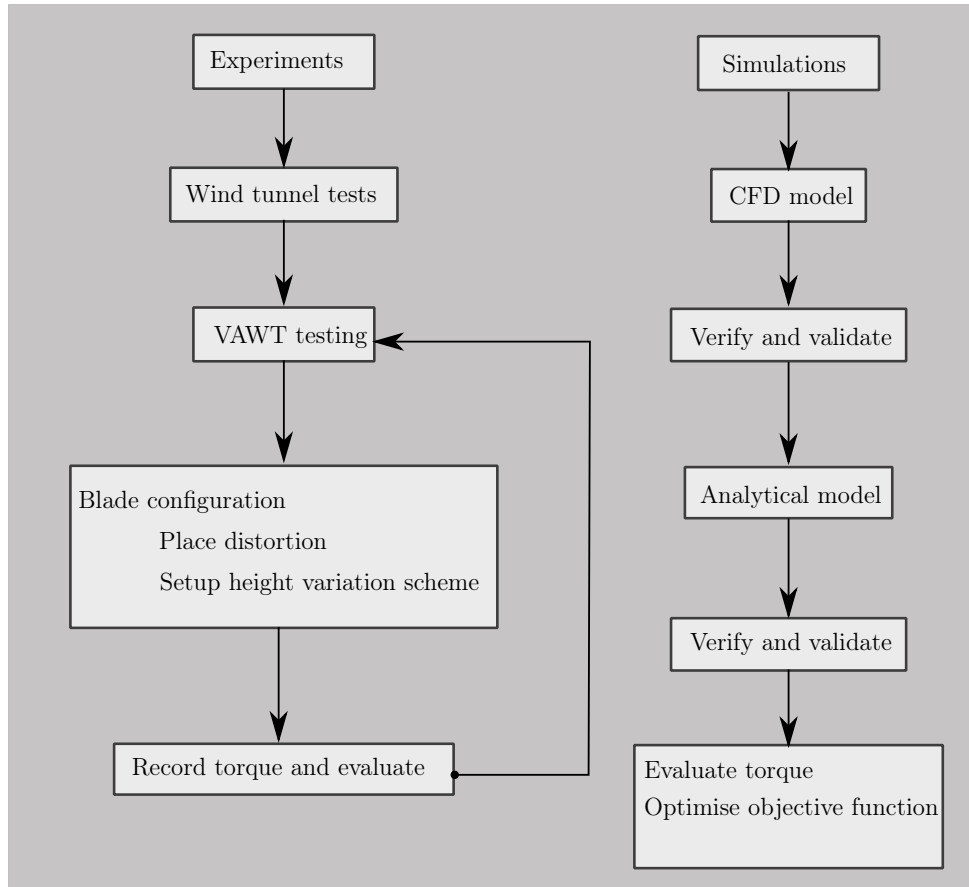


Figure 4.1: Work flow for virtual laboratory

To conduct experiments on the torque ripple of a VAWT would require wind tunnel testing, and for each proposed blade design a new test would need to be conducted. The loop between evaluation and testing is a resource intensive process both in terms of time and materials. The virtual laboratory route is not a simpler one from the outset. A high fidelity CFD model is required, with verified and validated results, to provide inputs for the analytical model. Similarly the approach for an analytical model also required verification and

validation. The CFD simulations were conducted for each blade design. The use of multiple workstations meant more than one design could be tested at a time. The analytical model for the VAWT was used to provide the torque ripple for each variation in blade design. This meant testing a new VAWT required only changes to the geometry settings in the script.

The virtual laboratory has its own drawbacks. In this instance mesh sensitivity and the turbulence model presented the largest hurdles. The mesh was based on the tool blockMesh (discussed in 4.2.2). The area of interest for each proposed distortion varied along chord length therefore local refinement of that area was required. In addition turbulence model selection played an important role. Since the boundary layer was being investigated, a model that could capture turbulent and laminar flow was important. A fully turbulent model was less susceptible to both mesh and boundary conditions. While a transitional turbulence model showed a large sensitivity to free stream turbulence intensity and mesh refinement.

4.2 CFD model

4.2.1 OpenFOAM

This study focused on the use of open source software. The open source package OpenFOAM, developed from Weller *et al.* (1998), was selected for this work due to its ease of access to the base code. Commercial CFD packages are industry accepted but limit user interaction with the code to ensure its functionality. OpenFOAM provides the user with direct access to the source code allowing for adjustments to all aspects of the package. OpenFOAM stands for open field operation and manipulation; it uses the finite volume method and is written in C++.

In the finite volume method the governing equations for conservation (Navier-stokes) are represented in integral form. These integrals are solved over a finite volume, the centre of which stores a variable value. The faces of these volumes are shared with neighbouring cells or a domain boundary. The value on the domain boundary has to be specified and all internal cell values can be determined through interpolation. OpenFOAM always uses a 3D mesh but for a 2D simulation the cells in the third axis are 1 unit thick and the boundaries are prescribed as empty. This allows OpenFOAM to treat the simulation as two dimensional. A 2D mesh was suitable for the purposes of this study since the DMST model portions the blade into elements. The assumption is that no flow takes place across the element boundaries, allowing for a 2D analysis. The simulations were run as steady state since the analytical model, discussed in section 4.3, was designed to use lift and drag coefficients at each stream tube location. This meant that the analytical model did not take into account the shape of the airfoil before or after the calculation point.

4.2.2 Mesh development

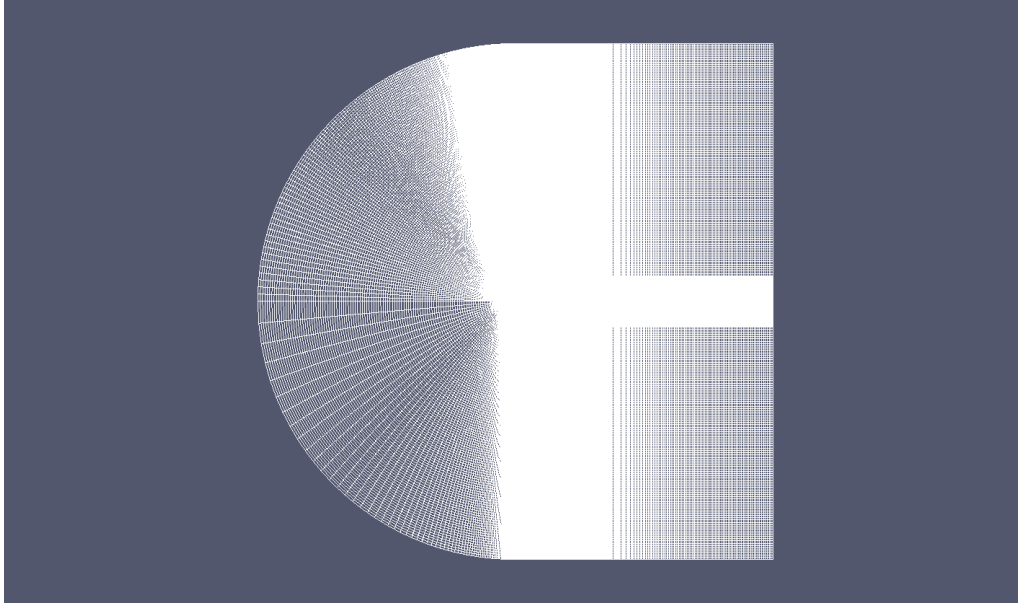


Figure 4.2: C-MESH in wireframe view

A C-mesh, as depicted in figure 4.2, was used in this study to enable changing the angle of attack through a velocity vector instead of rotating the mesh. The labels for the boundary patches are given in figure 4.3. The airfoil is only just visible in figure 4.3 at the centre of the image. The detail is lost when viewing the entire domain given the dimensions of the airfoil. It had a chord length of 1 m with a centre point at $[0,0]$. The remaining boundary faces were set 20 chord lengths from the centre of the mesh. This was done to ensure up stream and down stream effects were adequately catered for, while trying to ensure a small overall domain as laid out by Gebreslassie *et al.* (2012). The type of patches prescribed on each boundary and their associated values are set out in section 4.2.6.

In keeping with the open source theme of the study, the built-in meshing tool blockMesh was used. OpenFOAM comes packaged with another mesh program called snappyHexMesh which was investigated but ultimately not used. SnappyHexMesh uses a hexahedral background mesh in conjunction with a **STL** file to create 3D meshes. The user has a plethora of options allowing control of the cell splitting for refinement in local regions, surface snapping with layer addition and finally mesh quality control. Proficiency with snappyHexMesh enables users to create highly refined and structured meshes.

BlockMesh is the original mesh generation tool in OpenFOAM. A user breaks the domain up into hexhedra, as seen in figure 4.4. This shape has 6

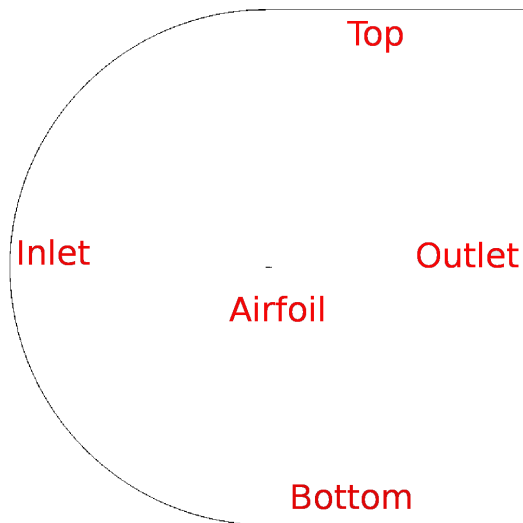


Figure 4.3: C-MESH with boundary labels

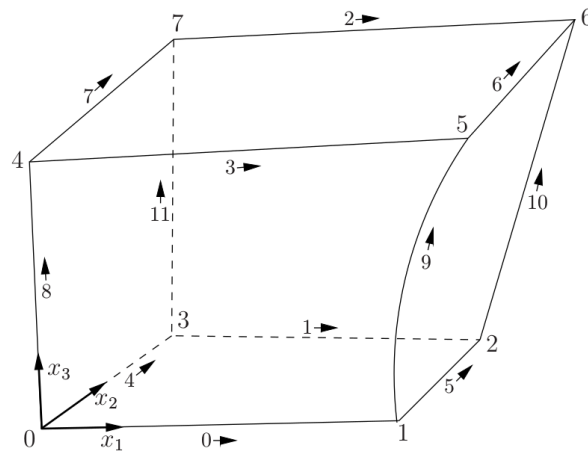


Figure 4.4: A single block used to build a mesh in blockMesh taken from Greenshields (2015)

faces and eight edges. The user provides the number of cells in each primary axis (x_1, x_2, x_3) and the grading along each edge. A user creates a text file comprising of points, blocks, edges and patches. The patch entry specifies a name for a collection of faces. The faces are not necessarily adjacent to each other but share a boundary condition. It is important to plan a mesh when using this tool. Logical ordering of the points and blocks allow a user to generate a script rather easily. Placement of the blocks in regions of interest also enable local refinement at a later stage. The simple description and flexibility

of blockMesh makes it easy to use in a parametric study through scripting.

4.2.2.1 BlockMesh scripting

Figure 4.5 illustrates how the mesh was decomposed into blocks, each labelled in blue. The black lines are the edges of the blocks and the red lines show the airfoil shape. There is an additional block on the upper surface of the foil. This block was added to provide increased resolution around the proposed distortion. Figure 4.6 shows the entire domain. The foil is not visible in this

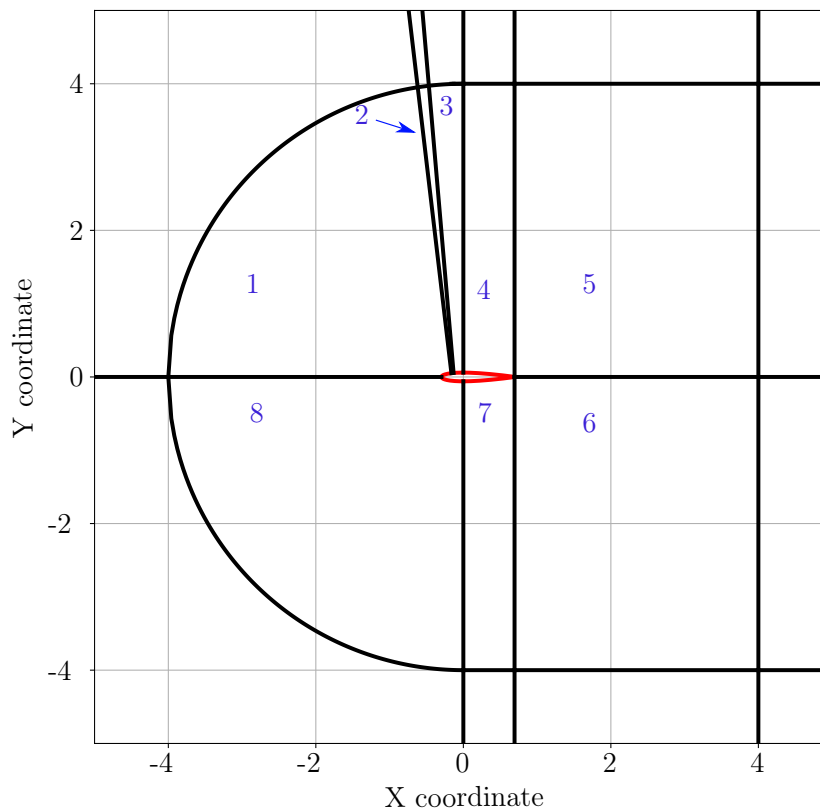


Figure 4.5: 2D mesh as made up of blocks

figure due to its size relative to the domain. The final mesh comprises of individual point specifications, to build twenty blocks.

As mentioned earlier the specification for a block not only requires points, but the number of cells in the primary axes and grading, along every edge. Grading in OpenFOAM is determined through a geometric progression dictated by the setting of the number of cells (N) along an edge length (L_1) and

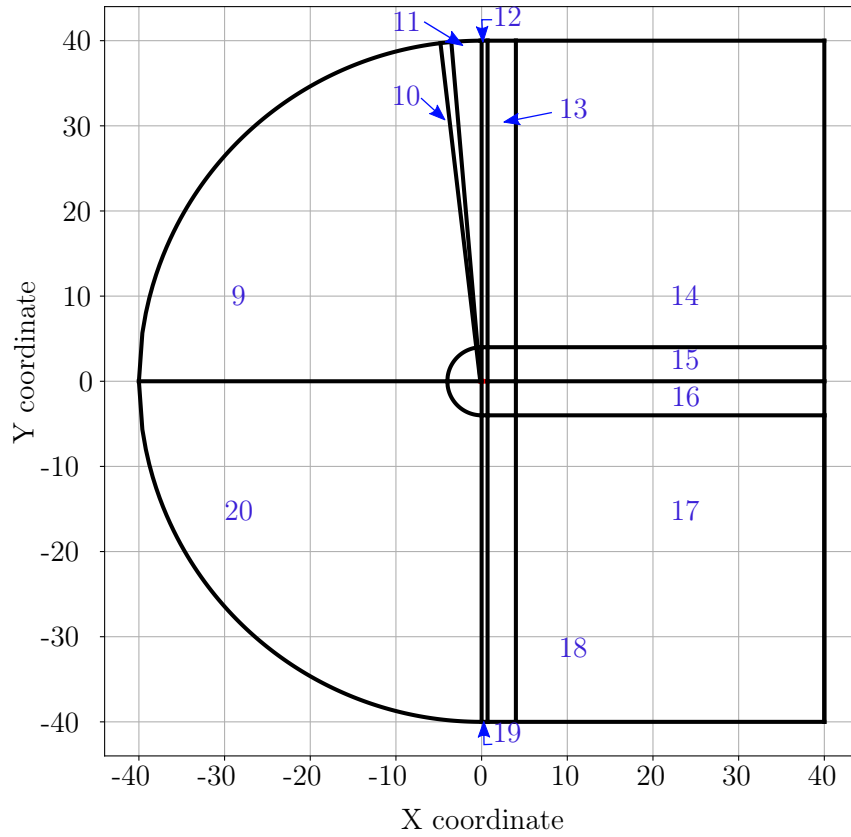


Figure 4.6: Focused view of foil with additional 'block' on the upper surface

the size ratio R between the first and last cell. The size of the smallest cell is given by

$$L_c = L_1 \frac{r - 1}{\beta r - 1} \quad (4.2.1)$$

where r is the size ratio between neighbouring cells as defined by

$$r = R \frac{1}{n - 1} \quad (4.2.2)$$

with

$$\beta = \begin{cases} R & \text{for } R > 1 \\ 1 - r^{-n} + r^{-1} & \text{for } R > 1 \end{cases} \quad (4.2.3)$$

Due to the number of meshes being produced through the course of the study a script was written to minimize the number of cells in the domain. The process is outlined below and presented as a flow chart in figure 4.7.

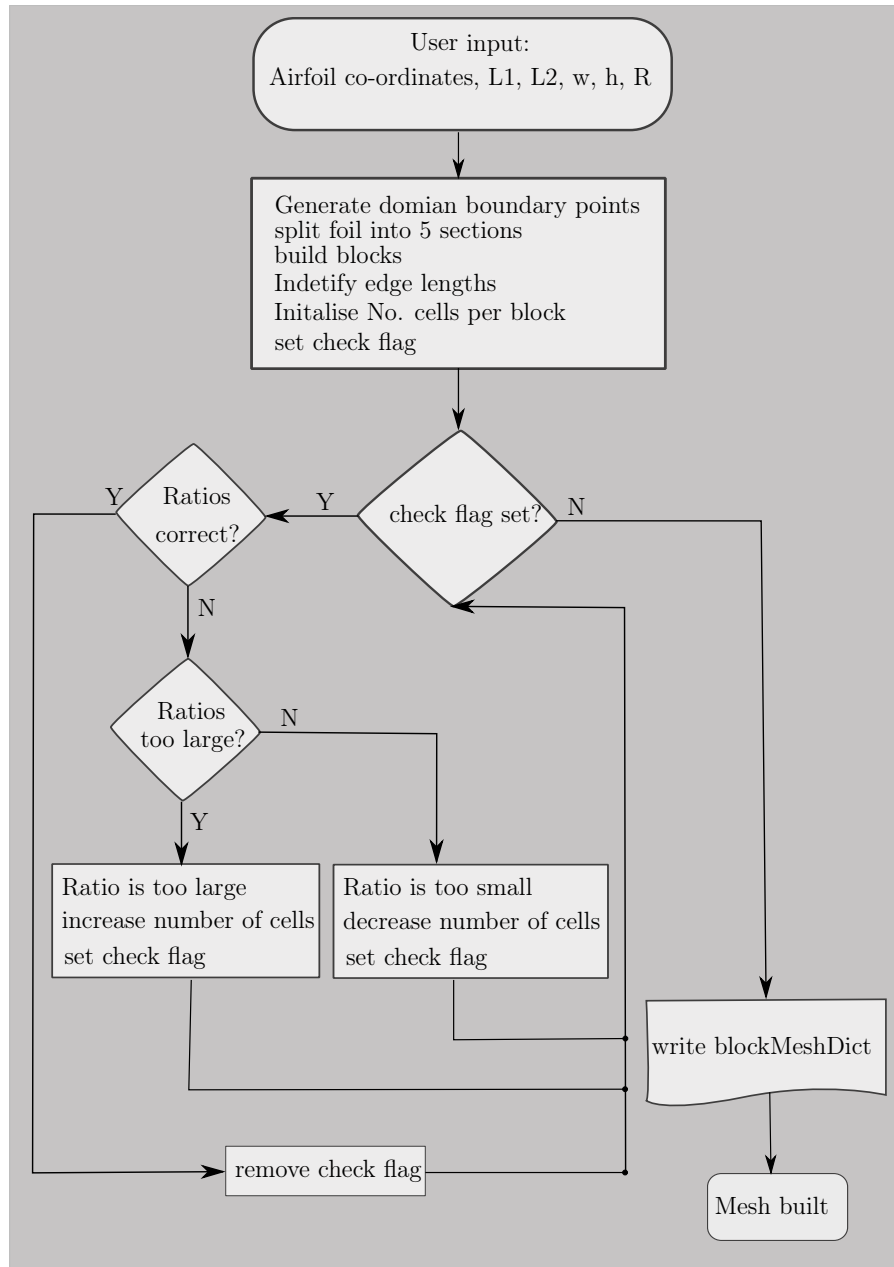


Figure 4.7: Flow diagram for mesh script

1. The user provides mesh co-ordinates; two points on either side of the distortion, an array of widths for the corner cells of each block, similarly an array of heights in the same order. Finally the acceptable ratios, as judged by the user, for the edges.
2. The user provides the domain points and splits the foil into five pieces. Once all the points are connected into blocks, a list of edge lengths is created. These are used in conjunction with the user inputs on width

and height to determine the initial ratios. These initial values are based on experience and are hard coded for each block.

3. A flag is set before going into a while loop.
4. The ratios are compared to the user limits and cells are either added or removed from blocks within the loop.
5. Once within tolerance of the ratios, the flag is removed and the code exits the loop
6. A blockMeshDict file is written with all the necessary entries (points, blocks and edges)

The check for ratios happens at each block. To ensure continuity in the mesh a cell at the corner of a block needs to line up with its neighbour in the adjacent block. If the ratio and cell count change in a block it has knock-on effects for its neighbour. To adequately account for these effects manually and design a mesh for each different airfoil would be prohibitive in nature. This made the script not only a useful tool for keeping the mesh small and of good quality, but also essential to build the various meshes used in this study.

The surface distortion was envisioned as a half circle, with diameter equal to the distortion width, protruding from the foil. At maximum height, equal to the radius, fixed points were assigned to the left and right of the distortion for placement of fillet radii. These were added to ensure a smooth transition from foil to distortion. These fillet radii were stationary as the distortion decreased in height. By fixing their position the slope changes between distortion width were accounted for.

Figure 4.8 shows the distortion at $0.15 X/c$, a non-dimensional distance from the LE based on chord length, with a width of 15 mm. The blue lines show how the height of the distortion varied per mesh. Each distortion and therefore mesh has three variables; width, height and location. Table 4.1 displays the variables covering the optimization workspace in this study. The height and width of the distortion were expressed in terms of chord length. The location for each distortion was defined in terms of distance from the LE of the foil.

Table 4.1: The working range of variables

Variable	Range	Unit
Re	$3.4e^5; 7e^5$	-
α	-10 ... 10	degrees
width	[0.5, 0.1, 0.15, 0.2]	%C
height	[50, 40, 34, 28, 24, 0]	%w
location	[0.5, 0.1, 0.15, 0.2]	%C from LE

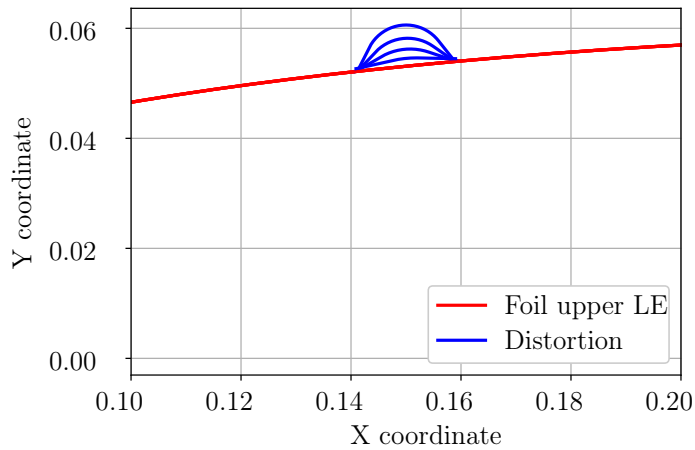


Figure 4.8: Changing height of distortion 15 mm wide at x/C of 0.15

A total of 96 meshes were built to cover the ranges described in table 4.1. The combination of variables listed in table 4.1 resulted in a single data point. Each of these meshes was used at two different Reynolds numbers with 1° increments in angle of attack from 1 - 10. This provided the data set upon which the surrogate model was trained. In total 1920 data points for C_l and C_d were created.

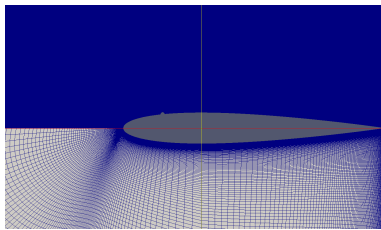


Figure 4.9: Increased resolution on upper surface of foil

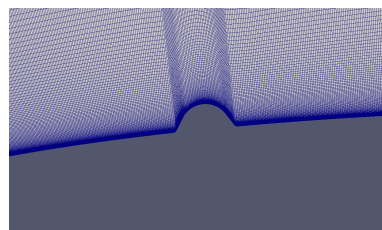


Figure 4.10: Mesh around the distortion

Figure 4.9 shows the results of the script. There are noticeably more cells on the upper surface of the foil as this is where the distortion was placed. Figure 4.10 shows the additional cells near the wall, to give detailed information on the boundary layer activities.

4.2.2.2 Mesh quality

The checkMesh function in OpenFOAM was used to determine mesh quality based on factors such as aspect ratio, skewness and orthogonality. These factors are described below and are important for a stable and accurate solution.

- **Aspect ratio:** The ratio of length to width of a cell, ideally as close to 1 as possible.
- **Non orthogonality:** Angle between the face normal of adjacent cells and the vector between their centres
- **Face skewness:** Describes the distortion of the cells by providing a ratio between lengths f and \vec{d} , as shown in figure 4.11. \vec{d} is the distance between neighbouring cell centres and C is the centre of their shared face. f is the perpendicular distance between C and \vec{d} .

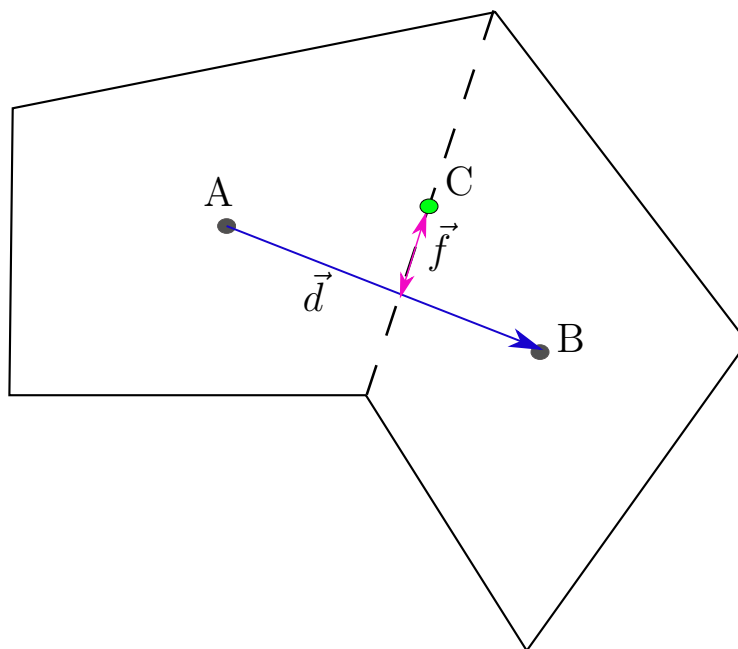


Figure 4.11: Calculation of face skewness

The OpenFOAM default settings check that the mesh has a 70° limit on non-orthogonality, 1000 for aspect ratio and finally a skewness value of 4. After each mesh generation the tool checkMesh confirmed its suitability. The skewness for all 96 meshes on average was around 1.2 with a maximum non-orthogonality of 67. The aspect ratio for all meshes never exceeded 700.

4.2.2.3 Mesh independence

Due to the Python script developed for this study, mesh independence took on a different format. Traditionally a mesh refinement is considered by increasing cell count and reducing cell size, until a parameter of choice no longer shows significant change in value. This was not a valid approach in this study due to the mesh generation script. The mesh utilized in this study was adjusted through a ratio refinement within each block of the mesh. Thus a statement on coarse or fine mesh no longer applied to the mesh as a whole but rather to individual zones of the domain. The independence will be presented in terms of local refinement and cell count with the C_l curve over the working range being the parameter under scrutiny. To deal with the angles of attack variation it was decided to use a the Pearson correlation coefficient (r) and root mean square error (RMSE) between two curves. r measures the linear relationship between variables on a scale between -1 and 1, where -1 indicates an inverse correlation and zero implies no correlation. This measure is useful to judge the similarity in trends between curves. While the curves may follow the same trend their values may differ significantly and this is where RMSE is used as an additional measure. It provides a measure of how far removed our curves are from each other. For each mesh α would be varied between 1 and 10 to generate a C_l curve that would be compared against the experimental data from Sheldahl and Klimas (1981). Results of the mesh independence study are given in table 4.2.

Table 4.2: Effect of refinement on RMSE between experiment and simulation

Refinement	Value	Cell count	r	RMSE
LE width ratio	1.01	966 000	0.9989	0.0368
LE width ratio	1.1	912 900	0.9996	0.0144
Fixed cell height after TE	$1e^{-5}$	912 900	0.9943	0.0335
Ratio mid span	1.5	859 500	0.9970	0.0620

As ratios are required for both height and width, only one aspect was adjusted at a time. On the LE, where the distortion would be placed, it was assumed a finer mesh would be needed to capture the boundary layer activity. Increasing the ratio results in a coarser mesh. The decrease in ratio on the LE showed an improvement of fit between simulation and experimental results. In the wake region the cells were originally set to grow proportionally as the

distance from the trailing edge increased. Enforcing a more uniform cell height in the wake region lowered the quality of fit. In an effort to reduce the number of cells the ratio at the centre of the foil was increased. This too resulted in a worse fit and the original cell expansion after the trailing edge was adopted.

4.2.3 Model initialization

The discretization schemes were all chosen for second order accuracy. The self-filtering central differencing scheme is part of the normalised variable diagram family of differencing schemes and is available in OpenFOAM. Studies have shown the choice of discretization scheme can affect the strength of separation on the leading edge of buildings (Cowan *et al.*, 1997). The following factors come into play when selecting a linear solver:

- **Preconditioner:** Aids in faster propagation of information through the mesh, by speeding up convergence.
- **Smoother:** While technically not a solver it is a means of removing a part of the residual error. Selection is based on the characteristics of the matrix being solved (symmetric, diagonally dominant, and so forth)

Pressure was solved with the generalised geometric algebraic multi grid (GAMG) solver which has its own preconditioning. Velocity, turbulent kinetic energy (k) and turbulent dissipation (ω) used a sparse linear solver with smoothing. The momentum Reynold thickness (Re_θ) and intermittency (γ) were solved with the preconditioned bi-conjugate gradient (PBiCG) technique with diagonal incomplete-Cholesky (DILU) preconditioning. Table 4.3 shows the settings utilised for the linear solvers as well as the relaxation factors. The under relaxation values were within the limits proposed by Barron and Salehi Neyshabouri (2003).

Table 4.3: Linear solver settings for steady state simulations

Parameter	Preconditioner	Solver	Smoother	Relaxation
Pressure	N/A	GAMG	Gauss Siedel	0.3
ω	-	Smooth	Gauss Siedel	0.5
Velocity	-	Smooth	Gauss Siedel	0.7
k	-	Smooth	Gauss Siedel	0.7
Re_θ	DILU	PBiCG	-	0.7
γ	DILU	PBiCG	-	0.7

To speed up the solution and initialise the internal mesh potentialFOAM was run before the simpleFOAM solver.

The models had a fine resolution near the wall and a y^+ value of less than 1 along the surface of the blade. Since the boundary layer effect was in

question, apart from a $y^+ < 1$, it was ensured that at least 10 cells were within the boundary layer. This enabled closer monitoring of the effect of the surface distortions.

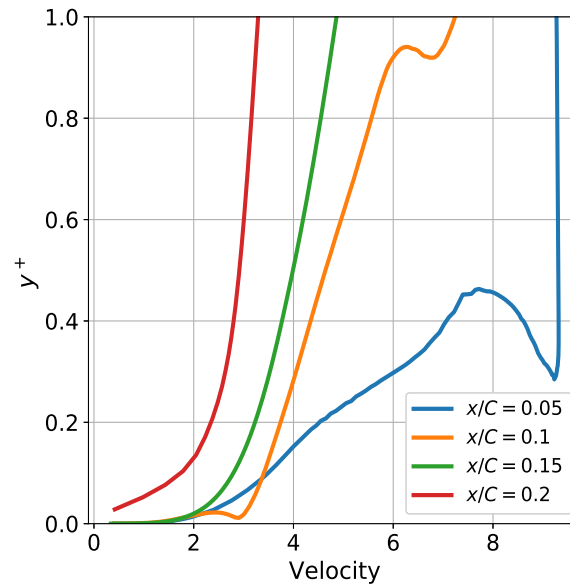


Figure 4.12: Velocity changes within the boundary layer

Figure 4.12 shows the resolution obtained for a NACA 0012 foil at 10 degrees angle of attack. Each line represents a different location on the upper surface. With the fine resolution in the boundary layer the velocity changes along the surface are more accurately represented. The figure indicates that more variations are present in the boundary layer as we travel upstream along the foil surface. As x/C increases the velocity profile stabilizes. The blue line, closest to the leading edge, shows a drop in y^+ . This is due to a sudden drop in streamwise velocity due to the formation of a recirculation zone.

4.2.4 Boundary layer

The thin layer where velocity increase from zero at the wall to 0.99 of the freestream velocity is termed the boundary layer. Within this zone the viscosity and wall friction have a strong influence compared to the frictionless flow further from the wall. The size of this layer decreases with an increase in Reynolds number. Figure 4.13 shows how layer thickness (δ) increases while travelling along the wall in the downstream direction.

Within this layer there is a strong velocity gradient normal to the wall. This in turns allows the viscosity to influence the flow through shear stress

$$\tau = \mu \left(\frac{\partial U}{\partial y} \right) \quad (4.2.4)$$

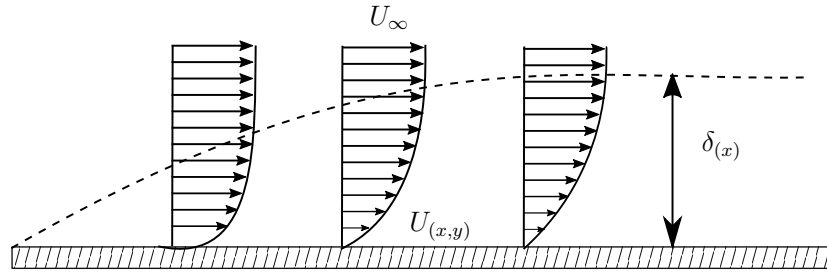


Figure 4.13: Illustration of boundary layer on a flat plate

Assuming δ to be much smaller than the airfoils characteristic length l and no slip at the wall ($u, v = 0$ at $y = 0$), the boundary layer thickness can be estimated as

$$\frac{\delta}{l} \approx \sqrt{Re} \quad (4.2.5)$$

More specifically for flat plate flow with zero incidence we can rewrite the boundary layer thickness as

$$\delta = 5 \sqrt{\frac{\nu x}{U_\infty}} \quad (4.2.6)$$

Another meaningful measurement of the boundary layer is displacement thickness

$$\delta_1 = \int_{y=0}^{\infty} \left(1 - \frac{u}{U_\infty}\right) dy \quad (4.2.7)$$

which gives the distance external streamlines are shifted due to the boundary layer. Finally by comparing the momentum in the boundary layer with that determined by potential flow theory

$$\rho \int_{y=0}^{\infty} u (U_\infty - u) dy \quad (4.2.8)$$

a momentum thickness can be defined

$$\theta_{BL} = \int_{y=0}^{\infty} \frac{u}{U_\infty} \left(1 - \frac{u}{U_\infty}\right) dy \quad (4.2.9)$$

These additional measurements provide more insight as to the effect on the flow. Separation is when the slow moving fluid in the boundary layer is transported into freestream flow. Schlichting and Kestin (1968) define the point of

separation as “*the limit between forward and reverse flow ... in the immediate neighbourhood of the wall*” written as

$$\left(\frac{\partial U}{\partial y}\right)_{y=0} = 0 \quad (4.2.10)$$

From this definition we can say separation is where transition starts. During transition the flow exhibits both laminar and turbulent characteristics intermittently. The next useful definition is for skin friction or viscous drag

$$D_f = b \int_{s=0}^l \tau \cos \phi \, ds \quad (4.2.11)$$

where b is the height of the cylindrical body; ϕ is the angle between oncoming flow and the tangent line to the surface, and s is measured along the surface.

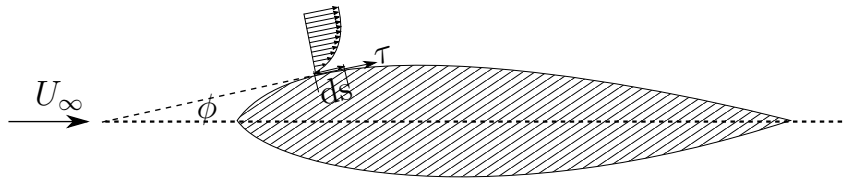


Figure 4.14: Depiction of skin friction calculation

Equation 4.2.11 can be used to calculate drag in both laminar and turbulent flow. Transition in the boundary layer from laminar to turbulent flow is marked by a dramatic increase in skin friction. Turbulent boundary layer flow calculations are based on the integral forms of the momentum and energy equations. They use equation 4.2.9 as the characteristic boundary layer dimension and relate the velocity profile to the pressure gradient through a shape factor

$$H = \frac{\delta_1}{\theta_{BL}} \quad (4.2.12)$$

This theory is applied in the turbulence model selection discussed in the next section.

4.2.5 Reynolds-averaged Navier Stokes turbulence model

Reynolds-averaged Navier Stokes (RANS) equations for turbulence modelling are based on the statistical approach expressing all relevant quantities as the sum of the mean and fluctuating parts (Wilcox *et al.*, 1998). The discussion below is limited to incompressible flow. The equations are given with time derivatives but these terms were not used in the CFD, as it was solved in a

steady state. Equations 4.2.13 and 4.2.14 represent the RANS in their familiar format.

$$\frac{\partial U_i}{\partial x_i} = 0 \quad (4.2.13)$$

$$\rho \frac{\partial U_i}{\partial t} + \rho U_j \frac{\partial U_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_i} (2\mu S_{ij} - \rho \bar{u}'_j \bar{u}'_i) \quad (4.2.14)$$

Holding the Boussinesq approximation as true, the Reynolds stress tensor is then defined as

$$\tau_{ij} = -\rho \bar{u}'_j \bar{u}'_i \quad (4.2.15)$$

The symmetric tensor τ_{ij} introduces additional unknown quantities. The averaging process increases the number of unknowns without providing additional equations such that the system is no longer closed. In turbulence modelling the aim is to provide approximations for the unknown correlations in terms of known properties. Using the definition of specific turbulence kinetic energy as below

$$k = \frac{1}{2} \bar{u}'_i \bar{u}'_i \quad (4.2.16)$$

the transport equation for k is then

$$\underbrace{\rho \frac{\partial k}{\partial t}}_{\text{Unsteady}} + \underbrace{\rho U_j \frac{\partial k}{\partial x_j}}_{\text{Convection}} = \underbrace{\tau_{ij} \frac{\partial U_i}{\partial x_j}}_{\text{Production}} - \underbrace{\rho \epsilon}_{\text{Dissipation}} + \frac{\partial}{\partial x_j} \left[\underbrace{\mu \frac{\partial k}{\partial x_j}}_{\text{Molecular}} - \underbrace{\frac{1}{2} \rho \bar{u}'_j \bar{u}'_i \bar{u}'_i}_{\text{Turbulent transport}} - \underbrace{\bar{p}' \bar{u}'_j}_{\text{Pressure}} \right] \quad (4.2.17)$$

On the left-hand side are the unsteady and convection terms. On the right the molecular diffusion term is known. However the production, dissipation, turbulent transport and pressure diffusion term are unknown. Equation 4.2.15 can be re-written as

$$\tau_{ij} = 2\mu_T S_{ij} - \frac{2}{3} \rho k \delta_{ij} \quad (4.2.18)$$

providing the production term. Previous work showed that the turbulent transport and pressure diffusion term can be re-written as

$$\frac{1}{2} \rho \bar{u}'_j \bar{u}'_i \bar{u}'_i + \bar{p}' \bar{u}'_j = -\frac{\mu_T}{\sigma_k} \frac{\partial k}{\partial x_j} \quad (4.2.19)$$

Equation 4.2.17 is then re-written as

$$\rho \frac{\partial k}{\partial t} + \rho U_j \frac{\partial k}{\partial x_j} = \tau_{ij} \frac{\partial U_i}{\partial x_j} - \rho \epsilon + \frac{\partial}{\partial x_j} \left[(\mu + \mu_T / \sigma_k) \frac{\partial k}{\partial x_j} \right] \quad (4.2.20)$$

and finally the dissipation as

$$\epsilon \approx k^{3/2} l \quad (4.2.21)$$

leaving l as the only remaining unknown quantity. Two equation models give us a means of calculating k and a variable based on either length scale or

dissipation. This study looked at the model proposed by Menter (1994), which makes use of both ϵ and the specific dissipation rate ω

$$\mu_T \approx \frac{\rho k}{\omega} \quad (4.2.22)$$

at different regions within the flow field. The blending function as defined in Menter and Esch (2001) dictated the turbulent dissipation. The commonly referred to $k - \omega$ SST turbulence model formed the basis for the transitional model utilized in this study. The $\gamma - \tilde{R}e_\theta$ is based on $k - \omega$ SST, but includes two more transport equations. This transition turbulence model put forward by Langtry *et al.* (2006) relied on empirical relationships, developed through wind tunnel testing, to accurately model the turbulence in the boundary layer and incorporate transitional flow. The model was based entirely on local variables, namely a transition momentum thickness Reynolds number ($\tilde{R}e_{\theta_t}$) and intermittency (γ). The model uses two additional transport equations, for intermittency and a transition onset criterion. This model does not try to describe the physical process but instead is based on empirically determined relationships. Upon publication, Langtry and Menter did not release their original correlations. However, in Langtry and Menter (2009) three empirical relationships are given in piecewise equations. These published relations were for a wide range of geometries and flow conditions. In Erfort *et al.* (2018a) the complex relationships were converted to simplified mathematical equations. The simplification involved replacing the piecewise equations with single functions. This reduced the run time by approximately 6 %. This also allowed for easier adjustments to the relationships, for a more accurate fit to experimental data, as the effect of each function coefficient was more apparent. Details on the model adjustments are given in Erfort *et al.* (2019a).

4.2.5.1 Turbulence Intensity

The $\gamma - \tilde{R}e_\theta$ model is sensitive to a freestream turbulence intensity (Tu). According to Sunderland *et al.* (2013) a VAWT can experience between up to 15 % Tu over the working range of velocities. The values experienced by a VAWT in an urban environment as reported by Pagnini *et al.* (2015), decreases with increasing wind speed but can be as high as 45 % as shown in figure 4.15.

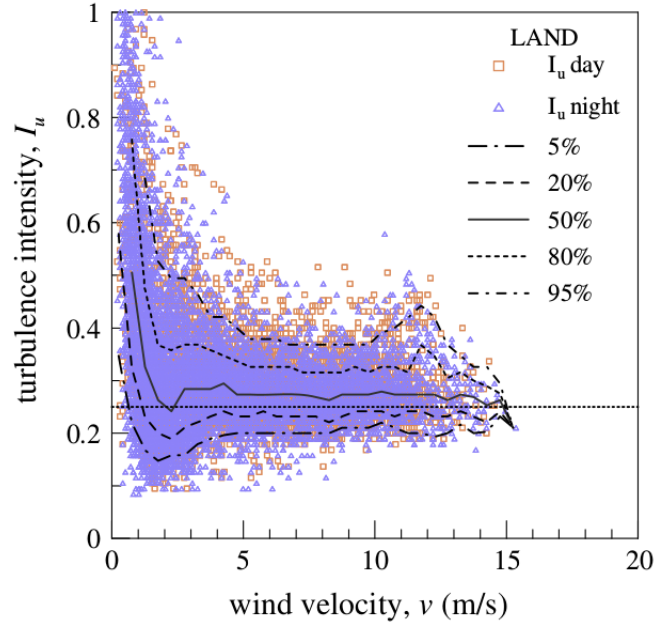


Figure 4.15: Turbulence intensity for VAWTs as reported by Pagnini *et al.* (2015)

4.2.6 Boundary conditions

Every boundary needs to be specified at the start of a simulation, either as Dirichlet, Neumann or a combination of the two. The pressure for all the boundaries, except the airfoil, was set as **FREESTREAMPRESSURE** conditions. This is a boundary condition in OpenFOAM that switches between a fixed value or zero gradient, depending on the sign of the flux through the patch. On the airfoil a Neumann condition of zero gradient was applied. The pressure on both the inlet and outlet was set as **FREESTREAMPRESSURE** patches. The velocity values were based on the angle of attack and again the freestream condition was applied to all boundaries. The turbulent kinetic energy value was determined from

$$k = \frac{3}{2}(Tu|U|)^2 \quad (4.2.23)$$

where Tu is the freestream turbulence intensity. This equation was applied to the inlet patch. The airfoil had a Dirichlet condition of almost zero. The remaining patches had a zero gradient value. The turbulent dissipation for the domain patches was calculated using

$$\omega = \frac{\sqrt{k}}{l} \quad (4.2.24)$$

and the mixing length l was set as 0.07 of the characteristic length of the foil, as per Wilcox *et al.* (1998). The blade values for ω were set using a floor

limiting value as laid out in Spalart and Rumsey (2007) and Hellsten (1998).

$$\omega = \frac{5U}{c} \quad (4.2.25)$$

According to the original paper detailing the $\gamma - \tilde{Re}_\theta$, the boundary condition for γ on the inlet was 1 while at the wall it was set to zero gradient. The transition momentum thickness was determined from the empirical equations based on the freestream turbulence intensity and pressure gradient.

$$Re_{\theta_t} \begin{cases} [1173.51 - 589.428Tu + \frac{0.2196}{Tu^2}] F(\lambda_\theta) & Tu \leq 1.3 \\ 331.5 [Tu - 0.5658]^{-0.671} F(\lambda_\theta) & Tu > 1.3 \end{cases} \quad (4.2.26)$$

$$F(\lambda_\theta) = \begin{cases} 1 - [-12.986\lambda_\theta - 123.66\lambda_\theta^2 - 405.689\lambda_\theta^3] \exp(-[\frac{Tu}{1.5}]^{1.5}) & \lambda_\theta \leq 0 \\ 1 + 0.275[1 - \exp(-35\lambda_\theta)] \exp(\frac{-Tu}{0.5}) & \lambda_\theta > 0 \end{cases} \quad (4.2.27)$$

The pressure gradient (λ_θ) was assumed to be zero when using these equations, to allow for an initial estimate. For validation of the turbulence model and mesh refinement, lift coefficients from the CFD were compared to the Sheldahl and Klimas (1981) data set at a Reynolds number of 360 000 and 700 000. Plots of the results from CFD overlaid on experimental data are given in figures 4.16 and 4.17.

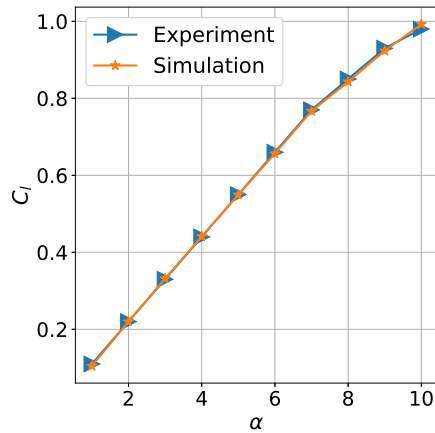


Figure 4.16: C_l at Re 360 000 for NACA 0012

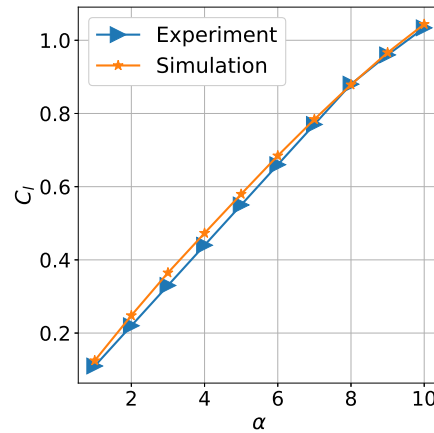


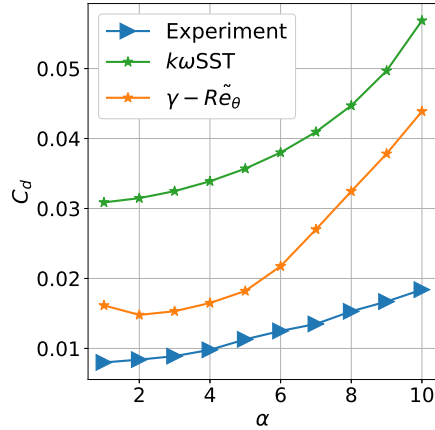
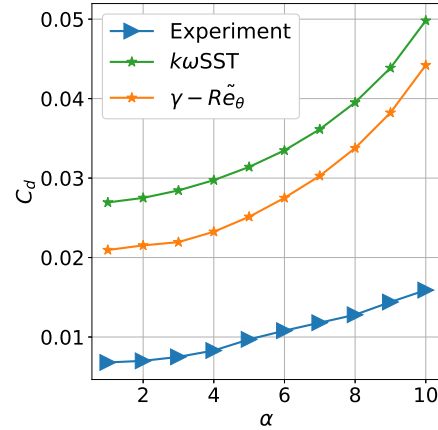
Figure 4.17: C_l at Re 700 000 for NACA 0012

Capturing the C_d was more difficult, but the transition model proved more effective than a fully turbulent model. Both turbulence models over predicted the drag coefficient. The transitional model was closer in its predictions and the relative overshoot of both models is given in table 4.4. As Reynolds number increased the error in C_d prediction from both models increased, with more severity in the transitional turbulence model.

Table 4.4: The overshoot in C_d predictions from CFD simulations

Model	Overshoot (%)	
	Re 360 000	Re 700 000
$k\omega$ SST	237	246
$\gamma - \tilde{R}e_\theta$	118	188

A comparison between experimental and simulation results are given in figures 4.18 and 4.19 illustrating the difference in C_d predictions.

**Figure 4.18:** C_d at Re 360 000 for NACA 0012**Figure 4.19:** C_d at Re 700 000 for NACA 0012

4.3 DMST model

The double multiple stream tube analytical model was selected to perform calculations on the VAWT setup. As mentioned in section 2.3.2.2 the model has been noted to provide good correlation of aerodynamic forces and can be solved relatively fast. This model uses lift and drag coefficients per stream tube to determine forces on the blade and the resulting VAWT motion. Looking at the Darrieus turbine, expressions for angle of attack (α) can be derived as a function of azimuthal position (θ) and TSR.

Figure 4.20 shows the cross-sectioned view of a H-rotor blade during rotation. The wind speed (V_∞) is in the y direction, θ is measured from the x axis and rotating about a point with an angular rotation (ω). The induced velocity (V_a) is in the same direction as V_∞ and due to rotation the foil has a tangential velocity (V_T).

$$V_T = \omega R \quad (4.3.1)$$

$$W = V_a - V_T \quad (4.3.2)$$

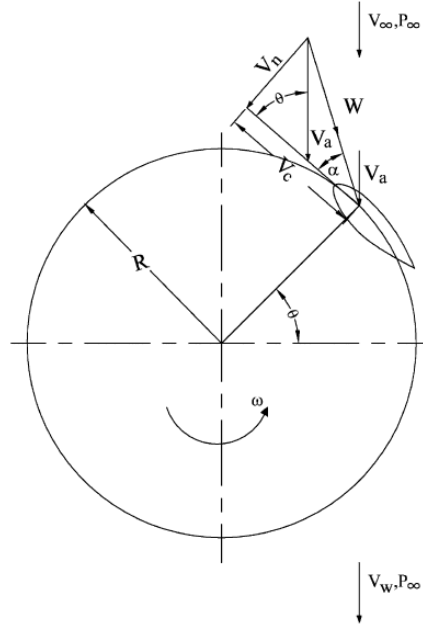


Figure 4.20: Plan view of a blade in a VAWT as per Islam *et al.* (2008)

By splitting the V_a into normal and tangential components the relative velocity can be expressed as:

$$W = \sqrt{(V_a \sin \theta)^2 + (V_a \cos \theta + \omega R)^2} \quad (4.3.3)$$

W can be expressed in terms of wind speed by dividing by V_∞ and using a from equation 2.3.7. Similarly α can be expressed as a function of θ using trigonometry:

$$\alpha = \arctan \left[\frac{(1-a) \sin \theta}{(1-a) \cos \theta + \lambda} \right] \quad (4.3.4)$$

where λ is the tip speed ratio of the turbine:

$$\lambda = \frac{V_T}{V_\infty} \quad (4.3.5)$$

With the aforementioned information and the blade characteristics for lift and drag, equations for the tangential and normal forces on the VAWT can be defined as

$$C_T = C_l \sin \alpha - C_d \cos \alpha \quad (4.3.6)$$

$$C_N = -C_l \cos \alpha - C_d \sin \alpha \quad (4.3.7)$$

To determine the torque experienced by the VAWT, information on the ambient wind conditions, geometry of the turbine and lift as well as drag characteristics of the blade being used need to be provided. As there is no open source

standardized code for the DMST model a verification and validation process was followed as outlined in Erfort *et al.* (2018b). In that work the trends in induction and torque coefficient for the python model, figures 4.21 and 4.23, were compared to published data, figure 4.22 and 4.24. Thereafter the was VAWT configured as per table 4.5.

Table 4.5: Parameters for DMST model input

Parameter	Value
Rotor diameter	6m
Chord	0.2m
Airfoil	NACA0015
Number of blades	2
Stream tubes	51
TSR	5.3

With this setup a direct comparison for C_P could be made with Paraschivoiu *et al.* (2009).

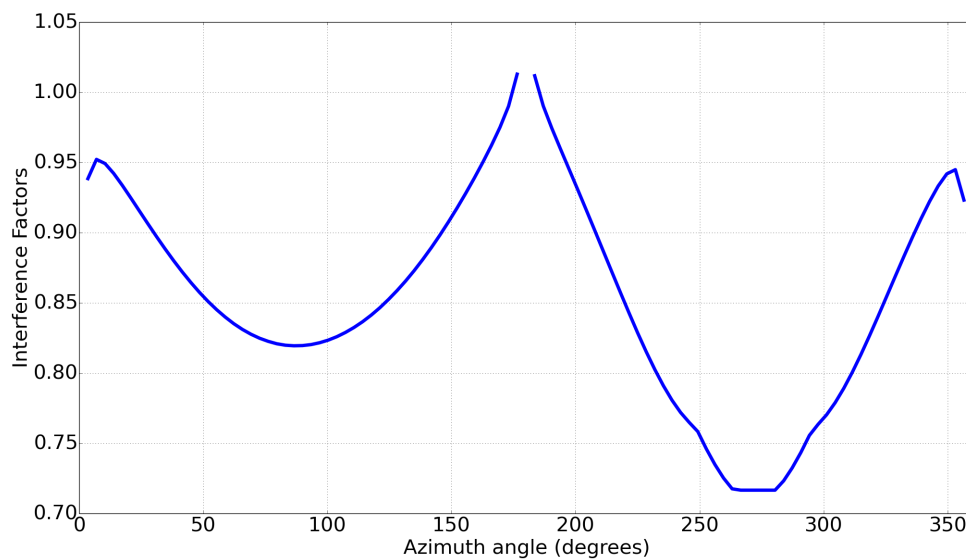


Figure 4.21: Interference Factors as determined by DMST analytical model, taken from Erfort *et al.* (2019b)

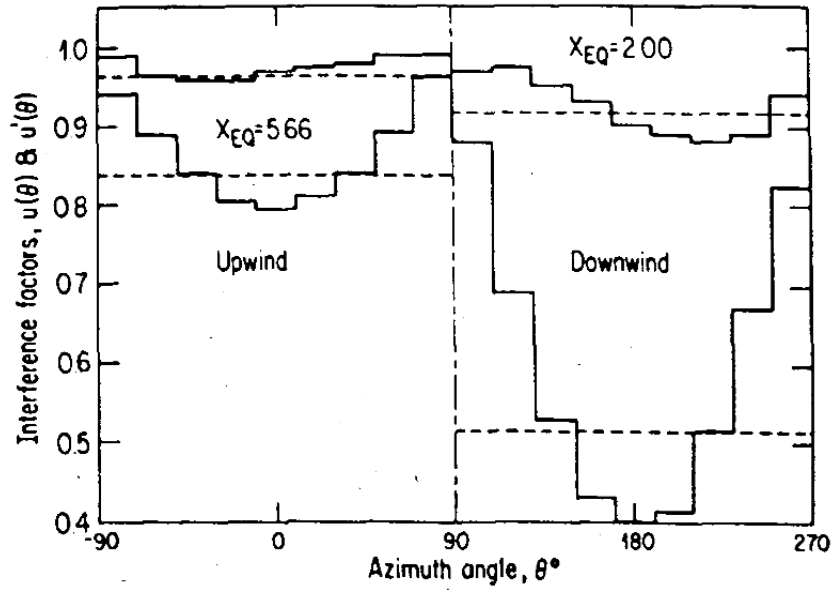


Figure 4.22: Interference Factors published in Paraschivoiu and DELCLAUX (1983)

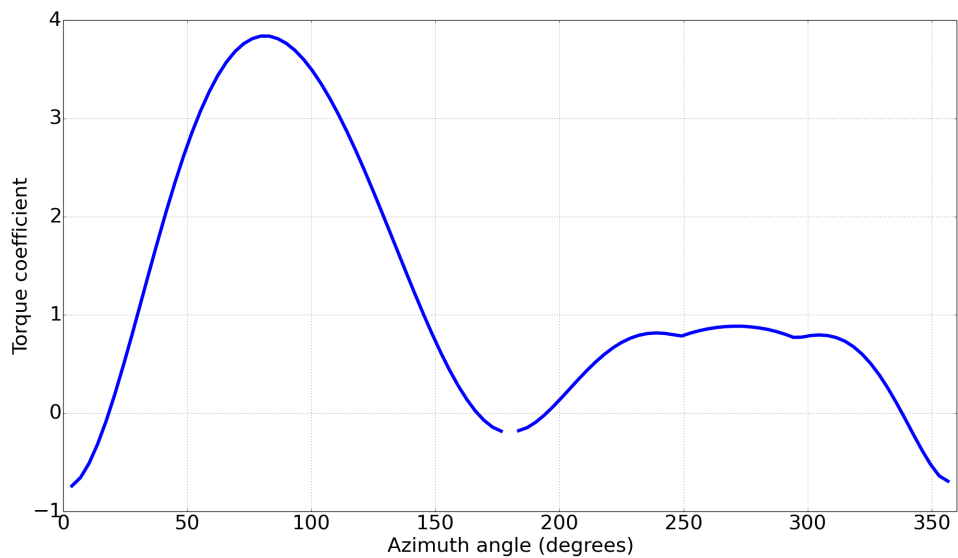


Figure 4.23: Torque coefficients as determined by DMST analytical model, taken from Erfort *et al.* (2019b)

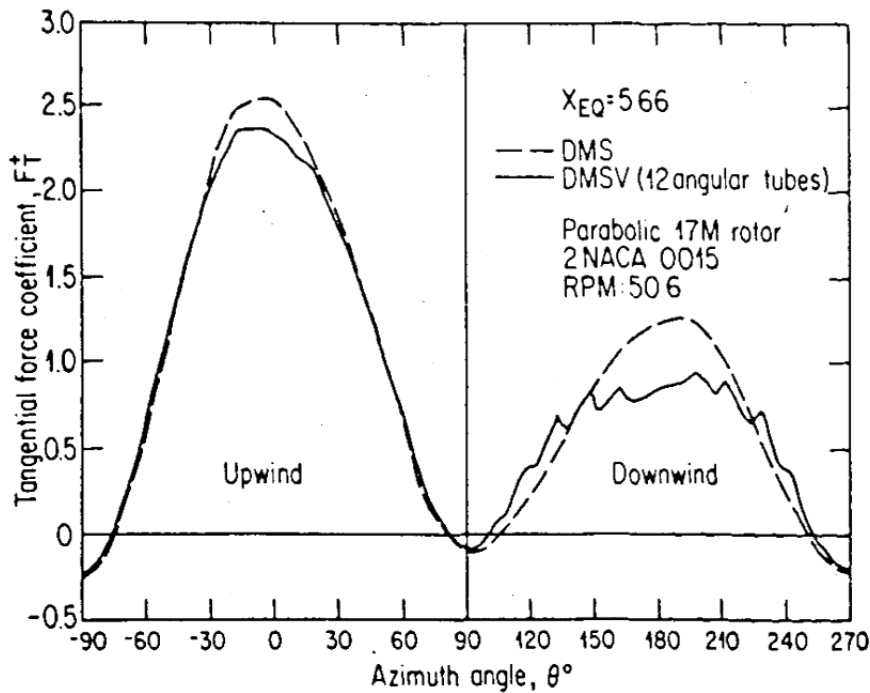


Figure 4.24: Torque coefficients published in Paraschivoiu and DELCLAUX (1983)

The VAWT investigated in this study was a two bladed H-rotor configuration, as per table 4.5. The only difference being the airfoil of choice. For this study a NACA 0012 for reasons discussed in section 4.2.5. Figure 4.25 shows the torque ripple predicted by Xfoil and the surrogate model (discussed in section 5.1) used in this work. This ripple is for the original foil shape and provided the baseline against which all optimisation was compared. The overall shape of the ripples is similar, with a Pearson correlation coefficient (r) of 0.998. The major concern for this study is the distance between the mean and peak of the torque coefficient.

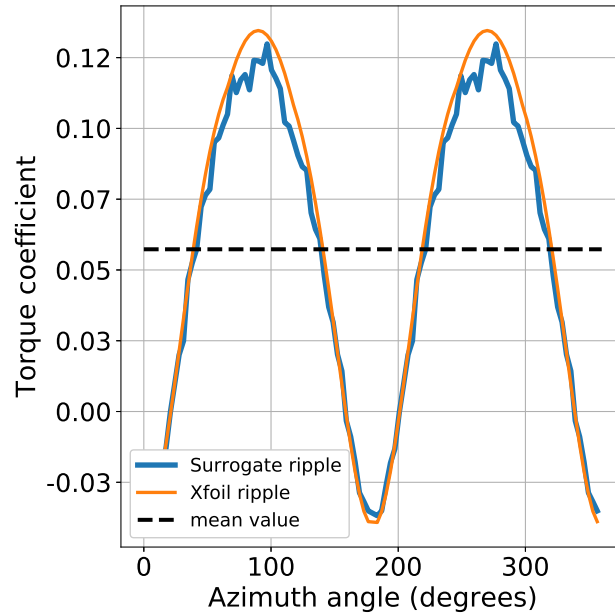


Figure 4.25: Torque ripple experienced by the VAWT during a single rotation

Summary

A discussion on the merits of a virtual laboratory are given in section 4.1. Based on the variation in blade shape and optimization goals it was decided that numerical methods were more appropriate for this study. An analytical model was implemented for the VAWT using the double multiple stream tube approach. This model verified in Erfort *et al.* (2018b) for use in predicting the torque ripple, provides evidence that the results obtained numerically will be close to experimental results. Actual wind tunnel testing is planned for future work. OpenFOAM was selected as the tool for obtaining lift and drag coefficients for the analytical model. A Python script was written to efficiently build a mesh based on airfoil and distortion geometry. The script ensured a high quality mesh and was used to minimize the number of cells in the domain. To investigate the boundary layer activity a transitional turbulence model was used. The model was verified in Erfort *et al.* (2018a), through a comparison with experimental data. A total of 84 meshes were generated and testing of the workspace described in table 4.1 resulted in 1920 individual data points. This included the full range of α and the bounding Reynolds numbers as provided by the analytical model.

Chapter 5

Surrogate Modelling

5.1 Model selection

The Python module Scikit-learn, developed by Pedregosa *et al.* (2011), was used to build the surrogate models. Four different surrogate modelling techniques were tested on the same data set at various sample sizes. Increasing the number of samples in a data set incrementally allowed for a simple analysis of model suitability. The training error increases as the samples increase due to the model having to fit more points. The testing error decreases as the model gains more knowledge of the work space. According to Abu-Mostafa (2018) the difference between testing and training error is termed the generalization error. The aim is to decrease the overall testing error while also minimizing the generalization error. Figures 5.1 to 5.4 represent the performance of the models based on their default formulation.

Table 5.1: Coefficient of determination for surrogate models

Model	R^2
Neural Network	0.6
Polynomial	0.88
Support Vector Regression	0.87
Random Forest	0.95

Table 5.1 provides the coefficient of determination (R^2) for tested surrogate models. R^2 was used a measure because it quantifies the certainty we have in predictions using a surrogate model. Neural networks (NN) require a lot of data and are used for highly non linear regression. These produced the worst testing error, which was ascribed to model complexity. The increase in testing error is usually accompanied by a decrease in training error due to over fitting. Figure 5.1 shows that both training and testing error increased after a certain point.

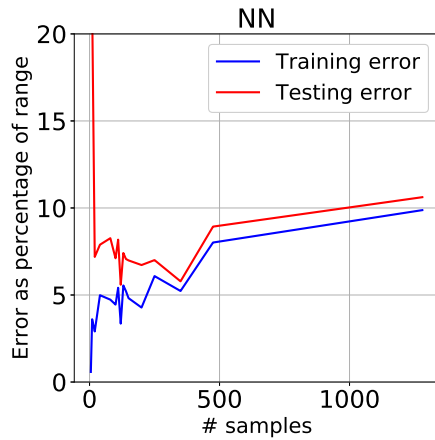


Figure 5.1: Neural Network

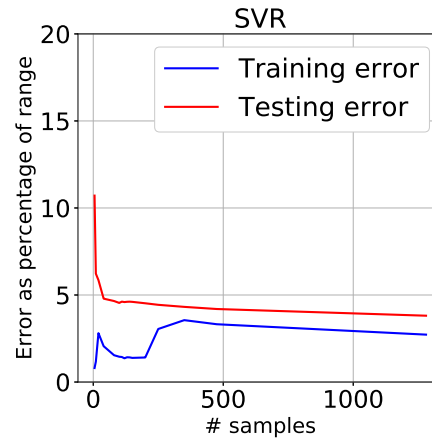


Figure 5.2: Support Vector Regression

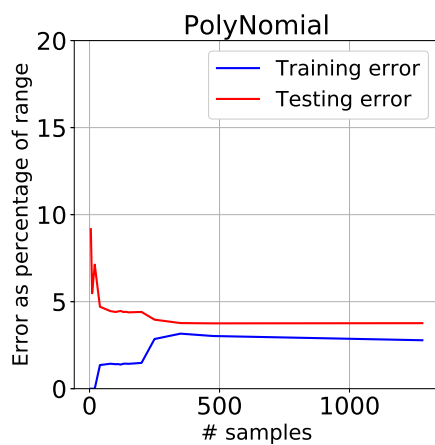


Figure 5.3: Polynomial

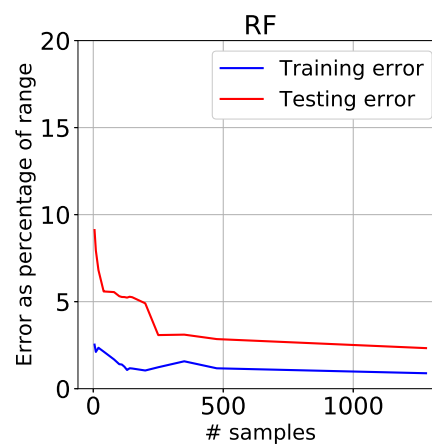


Figure 5.4: Random forests

This is attributed to the use of generic (default settings within the module) neural network settings such as number of layers, nodes, activation function and solver. Support vector regression (SVR) and a polynomial fit fared equally well achieving a testing error of just less than 5%. Adjustment of the specific settings for a surrogate model, like those described for neural networks, for increased accuracy is called hyper tuning. This can be a lengthy process and is usually done heuristically unless performed by an expert in the subject, as such only one model was chosen for hyper tuning. In the case of support vector regression models hyper tuning involved adjusting the penalty function, kernel and its associated parameters. Random forests was selected for hyper tuning as it out performed the other models, with the highest R^2 and lowest testing error. This type of surrogate model was hyper tuned by adjusting the number of trees in the forest and the depth of each tree.

5.1.1 Random Forests

Random forests (RF) are part of the ensemble group of learning methods. Ensemble methods are learning algorithms that use multiple base learners to inform on the final prediction result. In the case of Random Forests the base learner is a decision tree. This decision tree works through sequential identification and segregation of sub groups so as to reduce the error in prediction of a dependent variable. Meaning the algorithm take the whole set and divides into groups such that members have more in common with each other than another group. The newly formed groups are then subjected to the same process till the differentiation between group members is below a certain threshold. The tree is grown based on a random vector, which is introduced to minimize correlation and maintain strength. According to Breiman (2001) the introduction of randomness provides the following advantages:

- Accuracy is comparable to adaptive boosting techniques
- Robust to outliers and noise
- Faster than bootstrap aggregation (bagging)

The model implemented here used bootstrapping for development. Bootstrapping involves building a new data set of the same size as the original through replacement. For example with a sample set of 100 points a new set is created, using the original data set but with the duplication of some points. Thus a new distribution of data is provided with the same underlying information. The new data set results in a new decision tree. Hyper tuning on the number of trees and depth of a tree was carried out using three-fold cross validation.

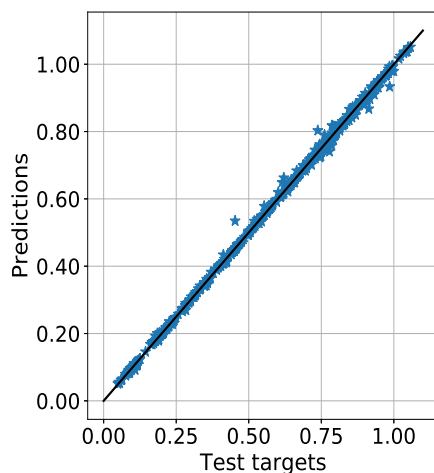


Figure 5.5: Surrogate model predictions for C_l

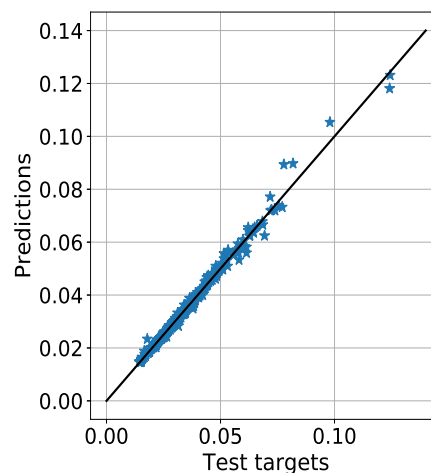


Figure 5.6: Surrogate model predictions for C_d

The coefficient of determination for the drag surrogate model was 0.97 and 0.99 for the lift surrogate. Figures 5.5 and 5.6 show the prediction versus target values for the testing samples of each surrogate post hyper tuning. It is clear the drag surrogate has trouble predicting higher drag values and achieved a good correlation of determination due to the number of samples in the lower region.

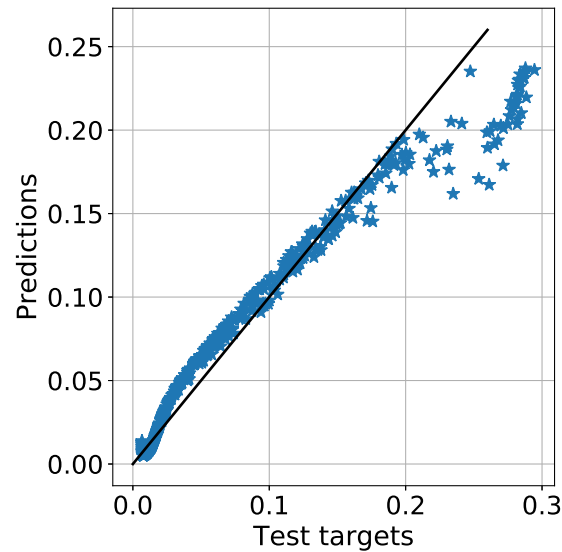


Figure 5.7: SVR surrogate for C_d used in Erfort *et al.* (2018b) showing a poor fit

A hyper tuned SVR model was used to capture drag in Erfort *et al.* (2018b) and showed a similar problem; see figure 5.7. In light of previous experience the RF model was found acceptable.

5.2 Optimization

Classical or gradient based optimization is very useful for a continuous, smooth design space and is efficient in high dimensionality. The optimizer needs a starting point, \vec{x}_0 , which is evaluated through the objective function, $f(\vec{x})$. As the name implies the gradient of the objective function at any point within the design space is also needed:

$$\vec{g} = \nabla f(\vec{x}) \quad (5.2.1)$$

The gradient is important as it directs the optimiser towards the next function evaluation point.

$$x_{i+1}^{\vec{}} = \vec{x}_i + J \vec{g} \quad (5.2.2)$$

The step size between \vec{x}_i and $x_{i+1}^{\vec{}}$ is determined by the function J which is unique for each algorithm and can be a function or constant value (Salomon, 1998). As the algorithm is based on gradient information it may encounter local minima or maxima that can stop the optimiser's exploration of the design space prematurely. Techniques have been developed to combat local minima. Through the operator J a degree of randomness could be included to counter weak minima or a more complex physics based model could be implemented to overcome more serious local gradients. There is also the option of multiple random starts which then allows for new paths to be taken through the design space (Cox *et al.*, 2001).

A very different approach to optimisation involves evolutionary algorithms (EAs) which have been described as a stochastic approach. These algorithms mimic the natural process of evolution to produce results that undergo fitness-based selection, where the quality fitness is measured through the objective function and its associated constraints. EAs are capable of handling both continuous and discrete design spaces but are not well suited to high dimensional problems and are inherently unconstrained algorithms. Within the field of EAs a further sub-division is made between evolution strategies, genetic algorithms and evolutionary programming (Back, 1996).

5.2.1 Genetic algorithm

Evolution is described as the adaptation of an organism to best suit its environment. Genetic algorithms use this analogy by replacing the environment with the optimization problem. The degree to which an organism is suited for the environment is determined through the objective function. An organism is made up of a unique genetic code which, in programming terms, means our sample is made up of a unique vector of data (\vec{x}_i). The members of the population being evaluated are organisms of the same genetic composition but with unique values. Their compositions are side constrained with upper and lower bounds for each vector component. The solution has constraints applied

through a penalty factor, that adjusts the objective function value to inform the algorithm of a non feasible solution.

$$P_0 = [\vec{x}_0, \vec{x}_1, \dots, \vec{x}_n] \quad (5.2.3)$$

The method to generate the starting population is algorithm independent. Once established the selection criteria (S) are applied. A subset (χ) of the initial population based on specific selection criteria is then created.

$$\chi = S(P_i) \subset P_i \quad (5.2.4)$$

The subset helps create the next generation through mutation or recombination (\bowtie).

$$P_{i+1}[k] = \chi[i] \bowtie \chi[j] \quad i, j, k \in \mathbb{I} \quad (5.2.5)$$

This process is repeated until the termination conditions are met (Tomassini, 1999).

5.2.2 Breeder Genetic algorithm

The breeder genetic algorithm (BGA) is described by Schlierkamp-Voosen and Mühlenbein (1993) as a genetic algorithm based on an artificial selection process analogous to the one followed by farmers. It is a combination of evolutionary strategies and genetic algorithms. The next generation is populated through the offspring of the top percentage of the previous population. The top performers, elite members, provide offspring through any number of recombination or mutation techniques. Additionally a subset of the elites is put into the new population ensuring the next generation never does worse than its predecessor. Further more a BGA can include more than one genetic operator, meaning both mutation and recombination can be used to create children. Full details of this approach are given in appendix C.

This study implements a BGA in Python based on the approach described above.

$$BGA = (f(x), \vec{x}, \zeta, G_N, P_N, E, \bowtie_m, \bowtie_r, \eta, \Omega) \quad (5.2.6)$$

ζ represents the upper and lower boundaries for the genetic make-up of each individual. This ensured that $x[i]$, for example, remained within the design space by enforcing the boundaries on each member of the population. G_N and P_N are the maximum number of generations and individuals in a population as stipulated by the user. $\bowtie_{(r,m)}$ represents the recombination and mutation rates respectively. η provides the number of elites to select from each population and Ω is the termination criterion.

The code used the seed vector and created $P_N - 1$ number of individuals through random mutation for a P_0 . Each member of the population is then confirmed to be within ζ ; if they are not, the offending component in the vector is limited to the boundary value. A function evaluation is carried out for every

member of the population and the results stored for inspection. The top E candidates are selected for breeding, they along with their offspring, populate the next generation. Portioned recombination was carried out by α_r . The offspring in the new generation are then mutated by α_m . In this particular code α_m was an adaptive mutation factor based on the population's general fitness level. The optimisation loop described earlier is repeated while the number of generations is less than G_N and the improvement between, the population as a collective, two consecutive generations is greater than Ω . The following conditions were used in this study:

- Random seed vector containing five values; amplitude, phase and frequency of the sine wave controlling the height of the distortion and finally the width and location of the distortion
- All variables were scaled between 0 and 1 prior to use in the BGA
- The BGA was allowed a maximum of 50 generations
- Within each generation there were 50 individuals
- The mutation and selection rate were 0.5 and 0.15 respectively
- The top five members of any generation formed the elite class
- The improvement between generations had to be less than $1e^{-4}$ to terminate early

Summary

Due to the complex relationship, and assumed equally variable and oscillatory environment being investigated, a genetic algorithm was selected to perform optimization. This was to combat possible discontinuities and local minima while effectively sampling the entire region. The design space was replicated with a surrogate model based on ensemble techniques. Shortcomings from experience with SVR encouraged investigation into alternative models. The random forest model proved the most accurate regression technique for the data set.

Chapter 6

Results

This chapter describes the culmination of the numerical and surrogate modelling. The analytical model had 102 data points, specific azimuth angle and Reynolds number, during a single rotation (see section 4.3 for details). The optimizer gave a height for the distortion at each of these points based on the Reynolds number and angle of attack as determined by the analytical model. Additionally the location and width of the distortion were also sought by the algorithm. The baseline model (BSL) for the VAWT showed a peak value for the torque coefficient, 30 % larger than the average.

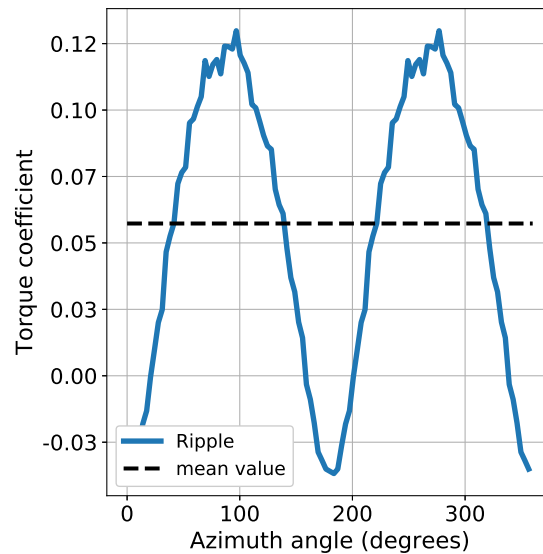


Figure 6.1: The BSL torque ripple for a two bladed H-rotor

The objective was to reduce the peak torque while maintaining an acceptable power output. The C_P value is directly related to the area under the ripple allowing the focus to be on the torque curve for both the objective and constraints.

6.1 Optimization result

The BGA was run, with random starts (random seed vector), multiple times for a given C_P constraint. Seven runs were conducted per group, and each group was defined by a constraint on the C_P value. The constraint was applied as an allowable percentage drop in C_P . This meant the BGA had to reduce the maximum swing between the mean and peak of the ripple while ensuring the final C_P value was greater than or equal to the constraint.

The function being minimized was

$$f(x) = |\overline{C_Q} - C_{Q_{max}}| + f_p \quad (6.1.1)$$

where C_Q is the torque coefficient as determined by the DMST model described in 4.3. The design variables adjusted by the optimizer are listed in table 6.1.

Table 6.1: Design variables

Variable	Range	
	Upper	Lower
Location	$0.2X/c$	$0.05X/c$
Width	20mm	5mm
Phase	π	$-\pi$
Frequency	10	0.1
Amplitude	$0.5 \times \text{width}$	0

The distance from the leading edge and height of the distortion along with the phase frequency and amplitude of the sine wave controlling its motion made up the design variables. These variables were scaled between 0 and 1 before being used in the optimizer. A penalty function (f_p) based on a goal C_P^* was used to create different optimisation groups.

$$f_p = \begin{cases} 1000 \times C_P & < C_P^* \\ 0 & \geq C_P^* \end{cases} \quad (6.1.2)$$

If the solution had a C_P value less C_P^* the function value received a penalty factor significantly larger than viable solutions and was thus thrown out by the optimiser.

Table 6.2 provides the geometry and reduction on torque ripple for each optimisation run. There are seven runs in a group, within each group the C_P constraint was fixed. The different groups were limited to a three, eight, twelve and seventeen percentage drop in C_P respectively. Group four has the widest spread in results and figure 6.2 shows the resulting torque curves for each run. The optimizer attempts to create a plateau where there once were peaks. In doing so the area under the curve is reduced and thus the C_P value decreases.

Table 6.2: Optimisation results

Run No.	Width (mm)	Location (X/c)	Reduction (%)	
			Ripple	C_P
1	5.3	0.07	5.63	3.01
2	6.5	0.06	5.63	3.01
3	5.3	0.07	5.63	3.01
4	5.0	0.06	5.63	3.01
5	5.2	0.06	5.63	3.01
6	5.1	0.07	5.63	3.01
7	5.2	0.05	5.63	3.01
8	5.2	0.07	12.74	8.03
9	5.3	0.05	12.99	7.73
10	18.5	0.05	12.94	7.95
11	5.0	0.05	12.13	7.61
12	19.8	0.05	11.48	6.70
13	5.2	0.05	5.63	3.01
14	6.1	0.13	3.62	7.56
15	20.0	0.05	19.06	10.74
16	12.4	0.06	6.81	3.42
17	12.8	0.07	13.93	11.69
18	19.5	0.05	9.96	9.73
19	19.7	0.05	14.83	11.18
20	20.0	0.05	11.37	6.66
21	6.2	0.05	15.65	11.20
22	7.5	0.07	23.99	16.51
23	19.7	0.05	17.30	13.27
24	10.0	0.11	25.65	17.02
25	5.5	0.05	5.63	3.01
26	14.7	0.06	13.52	16.69
27	7.3	0.07	5.63	3.01
28	18.8	0.06	16.67	14.88

A closer look is taken for run 24, the solution with the largest ripple reduction, highlighting what the optimizer is doing to the lift and drag values to achieve this levelling out. Figure 6.3 shows the optimized torque ripple as well as the mean torque coefficient decrease. In this instance the torque ripple was reduced by 25% for an associated reduction in C_P of only 17%.

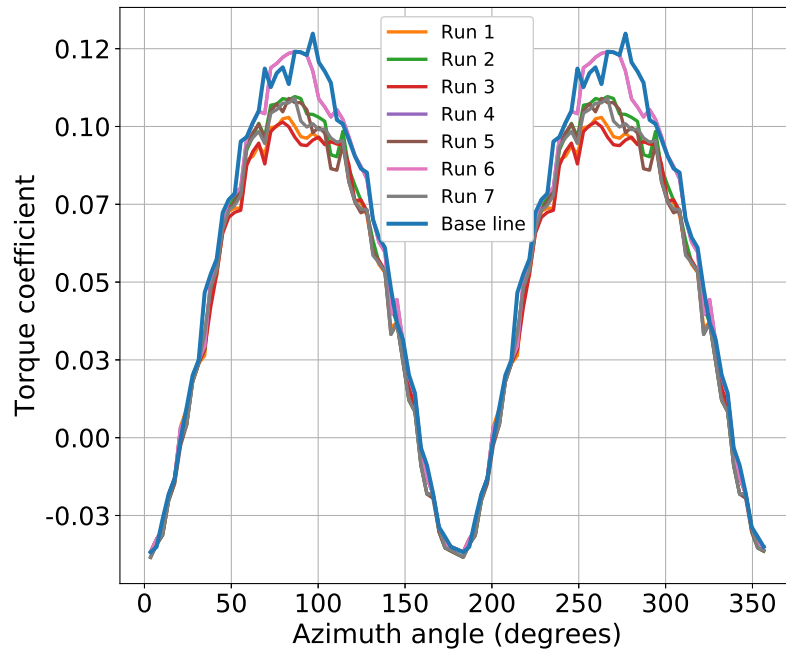


Figure 6.2: Comparison between BSL ripple and optimised solutions for group 4

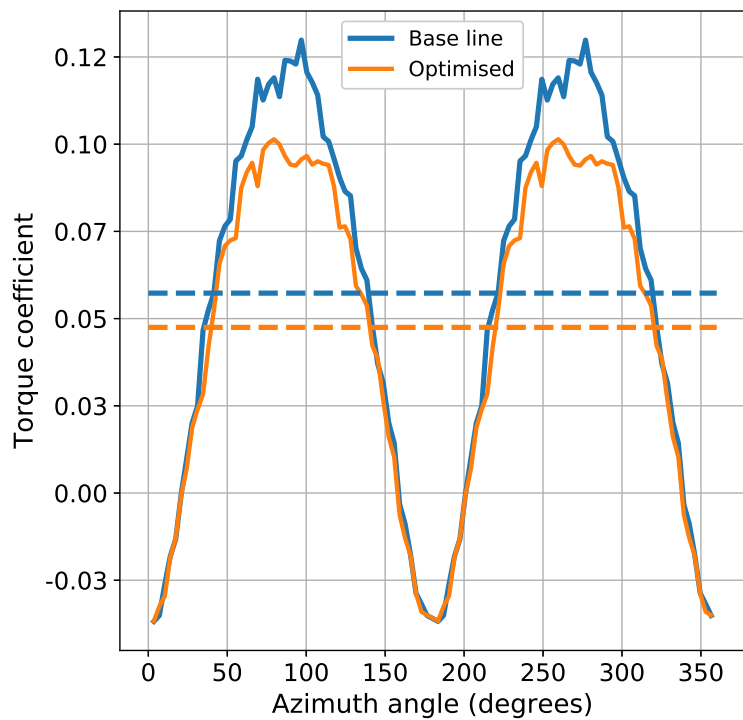
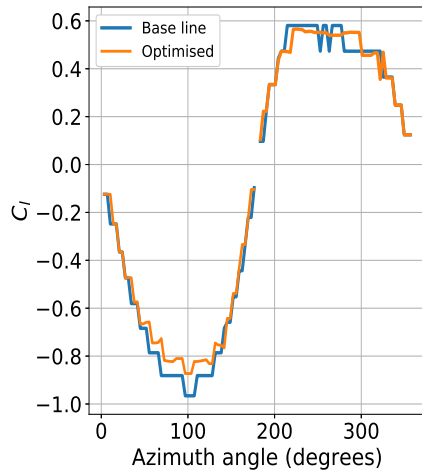
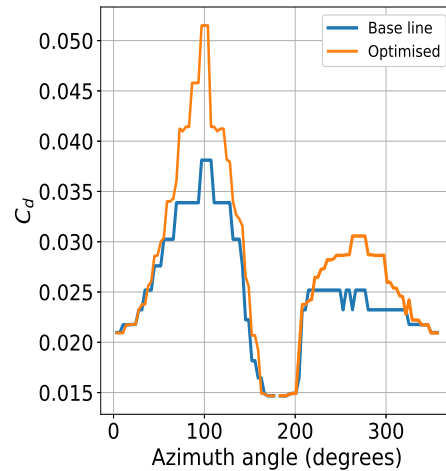


Figure 6.3: Comparison between BSL ripple and optimised result from run 24

Figure 6.4: C_l for run 24Figure 6.5: C_d for run 24

The changes in ripple are due to the effect of the distortion on the lift and drag characteristics of the foil. Figure 6.4 shows how these traits differ due to the inclusion of an adjustable distortion. In the upwind zone minimal changes exist between the BSL and optimised result, with the optimiser actually choosing to lower the maximum achievable C_l . However, the C_d graph shows that the cause for reduction in power was the large increase in drag throughout the rotation. The increase in drag was expected as any alterations to the initially smooth surface were reported in literature (see Back *et al.* (2012)) to always increase drag.

6.2 Trends in optimized results

Reduction in torque ripple versus C_P is plotted in figure 6.6. Four groups of optimisation were run with seven restarts in each group. Looking at the best performers in each group a polynomial fit can be made for the relationship between ripple reduction and power coefficient.

$$y = -0.035x^2 + 2.145x - 0.722 \quad (6.2.1)$$

Equation 6.2.1 provides a quick calculation on how much we can reduce the torque ripple for a given reduction in C_P . The reduction in ripple is larger than the corresponding change in C_P . The optimal solutions showed a larger spread as the C_P constraint was relaxed. Group three was allowed to drop the C_P value by 12 %, but run 16 only managed to reduce the torque ripple by 6.8 % with a corresponding drop in C_P of 3.4 %. This indicated that the optimizer got stuck in a local minima, thus supporting the approach taken of using random seed vectors. In figure 6.6 group one shows only a single point. However, while each run gave the same objective function value and

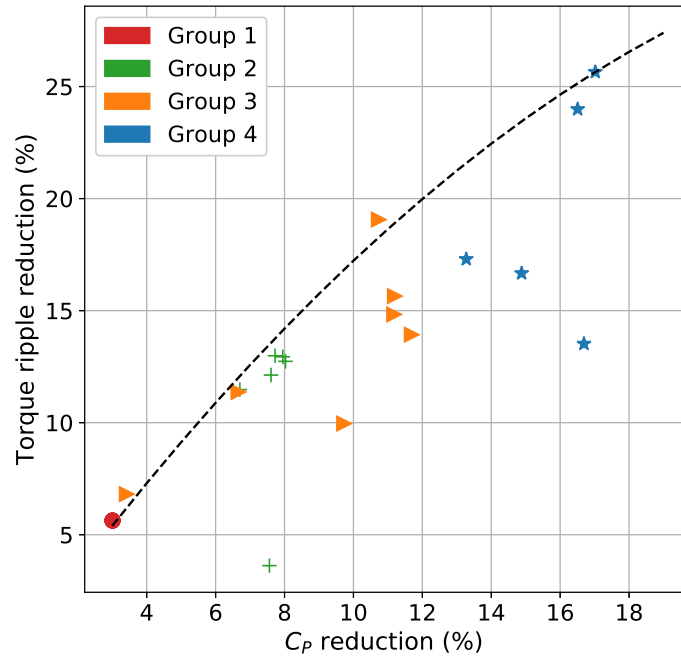


Figure 6.6: Reduction in ripple versus reduction in power coefficient

corresponding C_P reduction, each solution showed a different combination of distortion width, location and height variation.

Figure 6.7 shows that the amplitude, frequency and phase of the distortions vary between each run. In addition the combination of location and width for each solution are also unique. This may however be an artefact of the surrogate. All of these solutions are less than 1 mm in height and the data set used to build the surrogate did not include distortions of this height. The figure suggests that any change to the smooth surface will result in minimum of 3 % drop in C_P .

Looking at group two results, figure 6.8, indicates a limitation in the height of a distortion to ensure a drop in C_P of less than 8 %. The outlier with a height of 2.7 mm produced the lowest reduction in torque ripple. The frequency of the sine wave seems to be related to the frequency of the BSL torque ripple, see figure 6.9 for the results of group 3. Again a single outlier in terms of frequency is shown, while the majority of results only have 2 peaks.

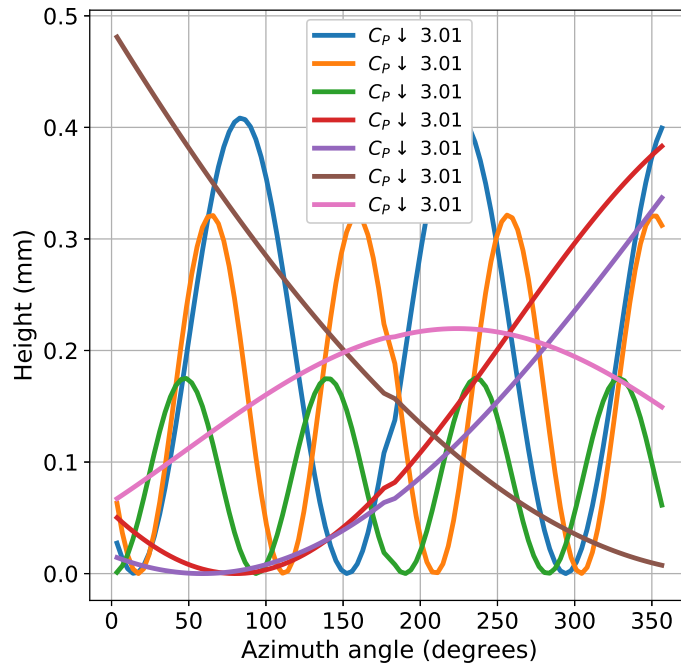


Figure 6.7: The change in distortion height for each run in group 1

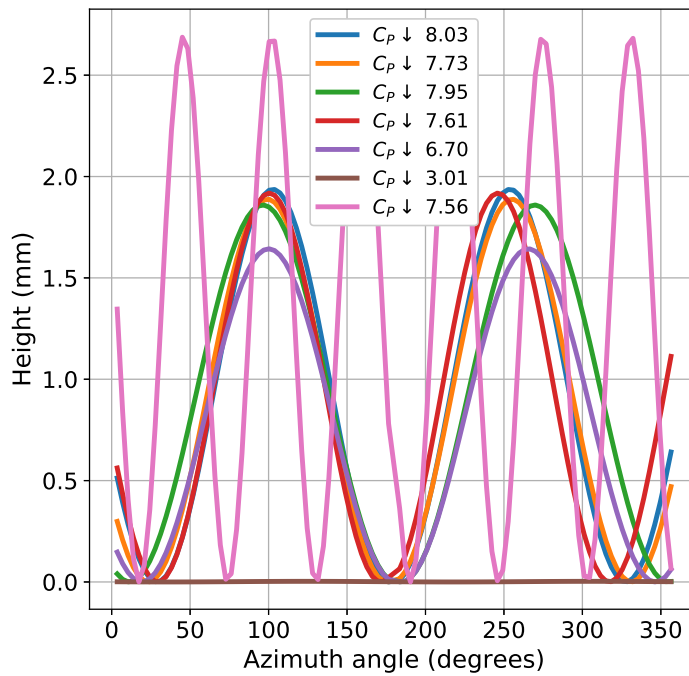


Figure 6.8: The change in distortion height for each run in group 2

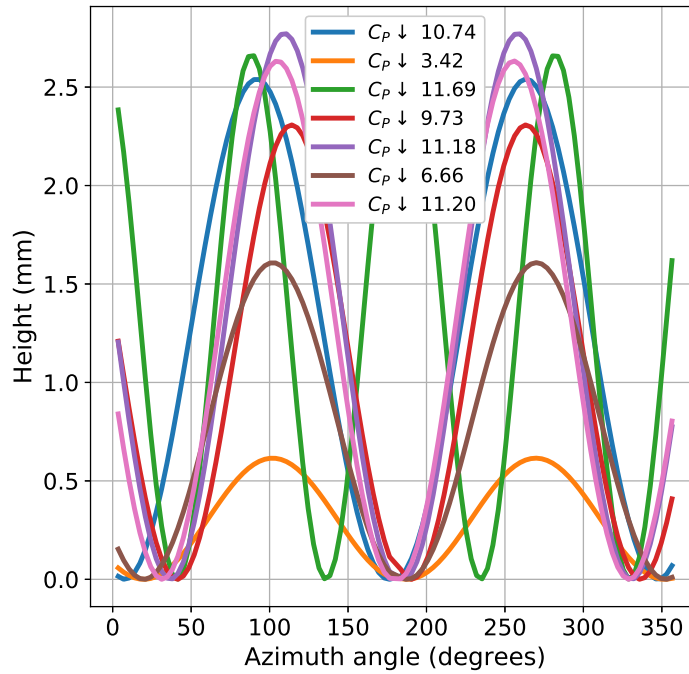


Figure 6.9: The change in distortion height for each run in group 3

Figure 6.10 shows that distortions are maximised close to 100 and 270 degrees azimuth angle. This is where the original model had its largest peak. In creating the largest displacement at this rotational position the maximum achievable lift is diminished. This coupled with the increased drag causes a reduction in torque as per equation 4.3.6.

The generational limit was not reached in any of the simulations with convergence being found within 35 generations. The optimiser tended towards placing the distortions close to the leading edge, with the majority of final solutions as reported in table 6.2 showing the distortion location before $0.8 x/C$. Incoming flow sees the most disturbance closest to the LE and these effects are transported along the foil surface. Thus a distortion in this region would have a larger impact.

6.3 Surrogate suitability

A comparison of the surrogate generated lift and drag values is made with OpenFOAM to confirm the validity of the results. Run 24, with a 10 mm width distortion at $0.11 x/C$, provided the comparison data from the surrogate model. Of the original 102 data points from a single rotation, 78 were used to test in OpenFOAM. These points all had a distortion height of 1 mm or greater. The 78 points, and therefore meshes, were then run at the Reynolds number and angle of attack as prescribed by the optimiser.

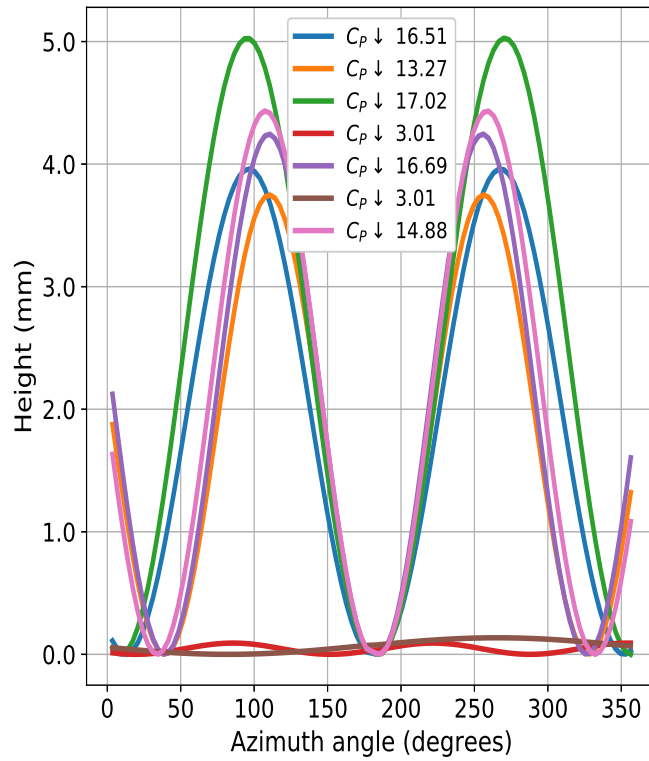


Figure 6.10: The change in distortion height for each run in group 4

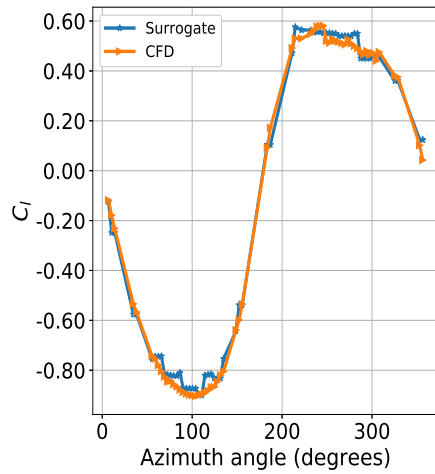


Figure 6.11: Coefficient of lift for surrogate and CFD

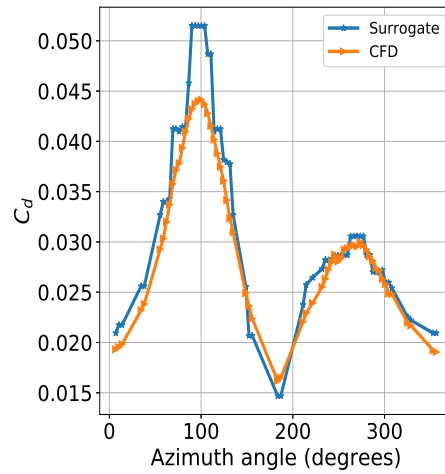


Figure 6.12: Coefficient of drag for surrogate and CFD

The results showed good agreement for the lift coefficient. There was some over estimating on the part of the surrogate but it achieved a coefficient of determination of 0.97 with a correlation of 0.99. This indicated the surrogate

model captured the trend very well and was able to predict the effect of an untested distortion with good accuracy.

The surrogate drag coefficient captured the correct trend but tended to over predict. The coefficient of determination in this case was only 0.89 and a correlation of 0.98. The lower drag values from the CFD infer a higher achievable C_P .

6.4 Effect of distortion

The addition of a distortion has been identified in literature to have an effect of the performance curves of an airfoil. The optimized solution determined that a variation in distortion height based on azimuth angle would provide the best result for ripple reduction and an acceptable C_P . Figure 6.13 illustrates the effect of varying the height by plotting C_l for three foils. A distortion of fixed height, a smooth foil and the optimized solution. There is no significant difference between the fixed and varying height foil. However, figure 6.14 shows a noticeable difference in the drag results. The variation in height ensures less drag earlier in rotation. This effect translates to a higher C_P than possible with a fixed height distortion.

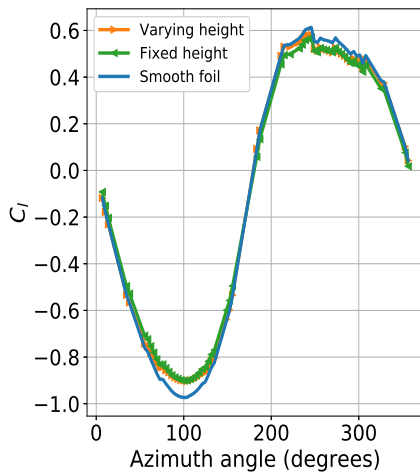


Figure 6.13: Coefficient of lift for surrogate and CFD

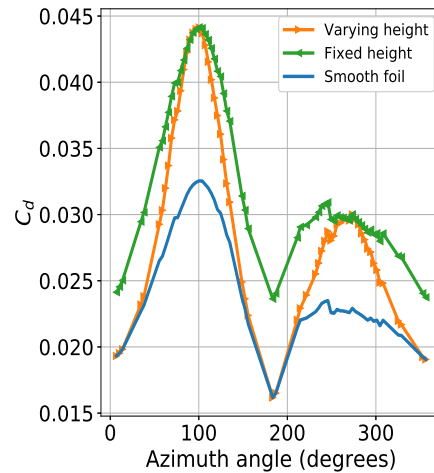


Figure 6.14: Coefficient of drag for surrogate and CFD

6.5 Turbulence model

Using a similar means of comparison as in section 6.3, lift and drag values for run 24 are plotted in figures 6.15 and 6.16. These results are obtained from a fully turbulent model and the transition model.

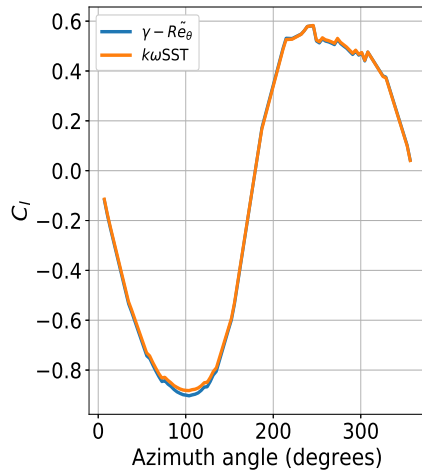


Figure 6.15: Coefficient of lift for turbulence models

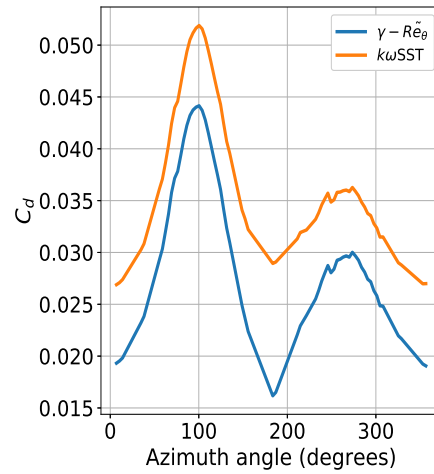


Figure 6.16: Coefficient of lift for turbulence models

The minor differences in lift coefficient are negligible but the difference in drag is substantial. As highlighted in section 6.4 this characteristic was important in estimating the performance of the VAWT. The increased drag is attributed to the higher skin friction coefficient (C_f) on the LE of the blade. This is shown in figure 6.17, where up to $0.05 X/c$, the fully turbulent model reports a significantly larger value.

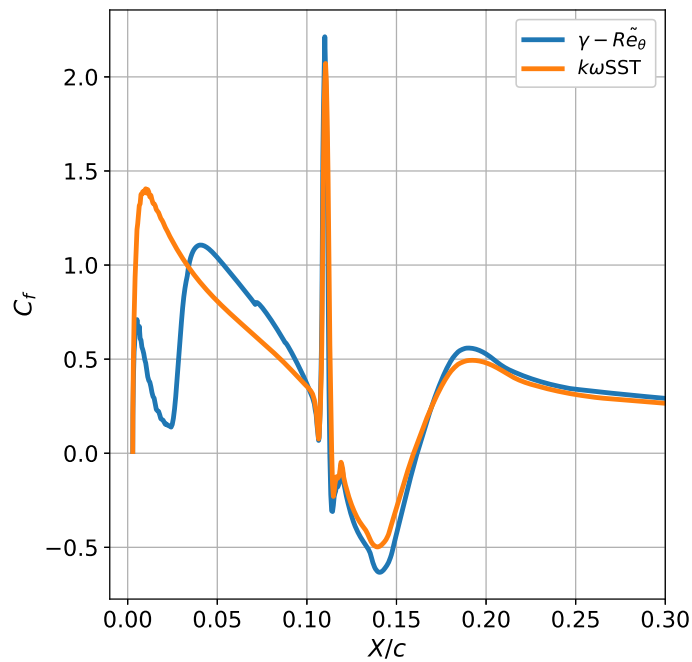


Figure 6.17: Skin friction coefficient on upper surface near LE

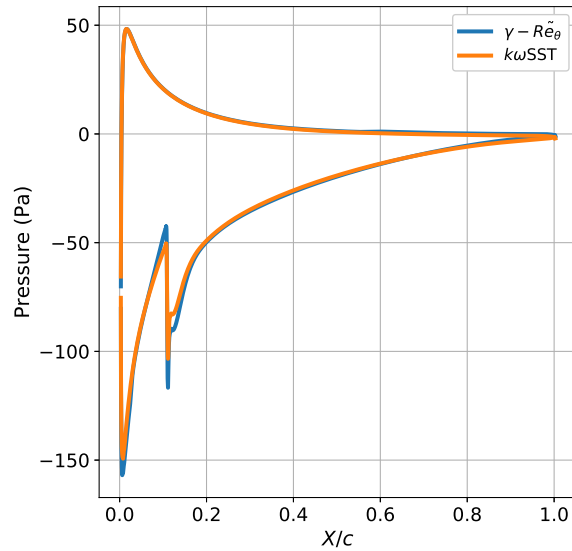


Figure 6.18: Comparison of relative pressure on foil surface for run 24, using $\gamma - \tilde{R}e_\theta$ and $k\omega$ SST models

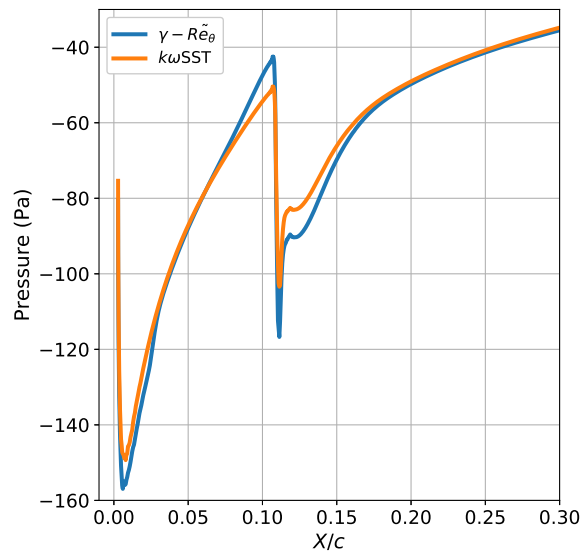


Figure 6.19: Comparison of relative pressure on upper surface at the LE, using $\gamma - \tilde{R}e_\theta$ and $k\omega$ SST models

Looking at the pressure on the foil surface, figures 6.18 and 6.19, the high pressure surface shows no discernible difference between models. The larger lift reported at 8° angle of attack is understandable based on the lower pressure on the upper surface. The $\gamma - \tilde{R}e_\theta$ model predicts increased suction near the LE and sustains this prediction post distortion.

Since the major modification between the turbulence models was only the

production in turbulent kinetic energy, a comparison of k for each model is given in figures 6.20 and 6.21. As expected the $k\omega$ SST model is fully turbulent prior to the distortion while k is still developing in figure 6.21.

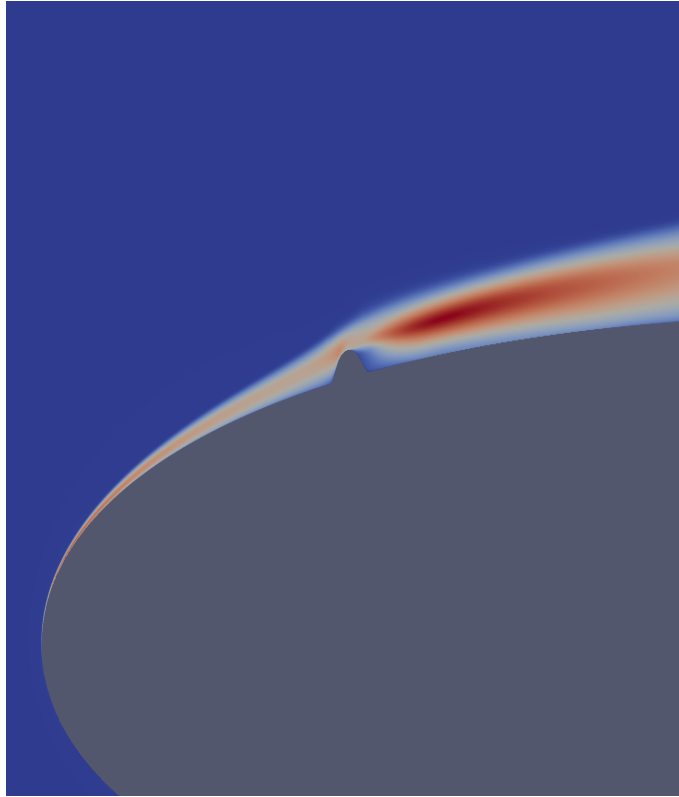


Figure 6.20: Turbulent kinetic energy around the distortion, based on $k\omega$ SST model

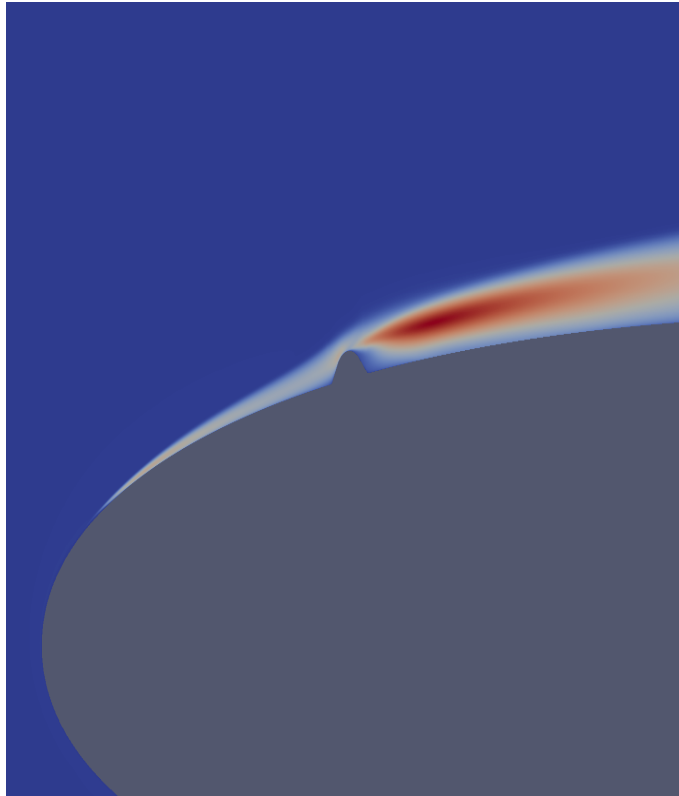


Figure 6.21: Turbulent kinetic energy around the distortion, based on $\gamma - \tilde{Re}_\theta$ model

Summary

The optimisation of the torque ripple led to the formation of a plateau in place of the original peaks. Surprisingly the rate of increase in torque coefficient was not affected. This meant that flattening the peaks reduced the area under the curve and therefore the C_P . The major influence of the distortion was to increase the drag, as shown in figure 6.5. C_P reduces linearly as the torque ripple is reduced. A distortion, whose maximum height coincides with the peak torque, results in the largest effect for ripple reduction. The surrogate model was effective in replicating the lift coefficients. It was less adequate in providing drag data. The turbulence model selection showed value in showing the tripping region on the foil surface.

Chapter 7

Conclusion

This study concluded that a morphing foil surface can reduce the torque ripple experienced by a VAWT during operation. Equation 6.2.1 provides a means of estimating the maximum reduction in torque ripple based on an acceptable loss in C_p . The summary of each chapter's contribution is listed below:

- A discussion on the current technology in wind turbines and boundary layer manipulation was given in chapter 2. Based on the literature uncovered and trends investigated, a morphing foil with minimal deformation would result in suitable changes in the lift and drag coefficients. Attempts to reduce the torque ripple by other means had treated the symptom and not the cause, like the compliant couplings described by Sutherland *et al.* (2012).
- The usefulness and applications of a virtual laboratory were discussed in chapter 4. A concerted effort was made to use open source code for each stage of the numerical modelling. Verification and validation of an analytical model for a VAWT implemented in Python were carried out in Erfort *et al.* (2018b). CFD simulations in OpenFOAM provided the lift and drag characteristics for the morphing foil once a suitable turbulence model was selected. Due to the high number of simulations performed, work was done to increase the efficiency of a transitional turbulence model through simplification of some key equations as described in Erfort *et al.* (2018a).
- A random forest surrogate model was chosen for use in a breeder genetic algorithm. The details investigated by the surrogate models and their suitability were given in chapter 5. Experience with surrogate modelling for lift and drag of simple foil was gained in Erfort *et al.* (2017), where SVR and Xfoil were used to build the surrogate. In the current work however the ensemble regressor outperformed neural networks, SVR and polynomial fits in modelling lift and drag for various foil configurations.

- The relationship between the drop in torque ripple and power coefficient is presented in chapter 6. This relationship was obtained by adjusting the penalty factor in the genetic algorithm. In every instance the C_P value drops to accommodate the drop in torque ripple. To compensate for the reduction in C_P the blade could be extended to occupy more volume and thereby achieve the same power output. This would of course require additional support. The results of the optimisation were verified by running the configuration through OpenFOAM. This analysis showed the surrogate to be effective in modelling a distortion at the LE of a foil.

The life span of a drive train shaft under the same loading conditions but with a reduced torque ripple, as predicted in this study, could be extended by 36%. This assumption is based on the Modified Goodman criterion for endurance limit. The physical mechanism for controlling the distortion height has been envisioned as a cam following system that would stretch the flexible skin of a foil. The use of a virtual laboratory facilitated in parallel testing of a variety of factors throughout the design process. However, to begin using the processing power of computers great care must be taken to ensure our mathematical models are sufficiently accurate to give realistic answers. The time needed to get the virtual laboratory ready for use may outweigh the time saved during the design process.

The major challenges faced in this work were:

1. Mesh accuracy and flexibility
2. Turbulence model selection and validation
3. Surrogate model accuracy

7.1 Mesh

In total 84 meshes, each consisting of more than 1 million cells, were used at various flow regimes to determine lift and drag coefficients of the distorted foil. A mesh scripting program was developed to facilitate in building acceptable meshes based on user input (see section 4.2.2). The description of a mesh built with blockMesh enables parametric studies of geometry to be carried out with ease. To reduce the number of cells and make use of grading in a mesh requires user oversight and is not simple. Open source meshing software is becoming more available but the learning curve remains a deterrent.

7.2 Turbulence

This study assumed boundary layer effects due to the distortion on the foil. A transitional turbulence model was capable of giving more accurate information

within the boundary layer. Implementation of such a model was simplified due to the OpenFOAM coding structure. Having experimental data on which to validate the mesh and turbulence model was vital before attempting new designs.

7.3 Surrogate model

When using an evolutionary algorithm like the BGA it is vital to have a stable robust and fast means of evaluating the objective function. The surrogate model replaced the CFD function of providing lift and drag coefficients during optimization. Ensuring its accuracy was important as the optimization would only be as effective as the information it received. More than one metric was used to determine the best model. The choice of these metrics was important and based on an understanding of the process being substituted.

Recommendations

The preceding sections described an investigation into a morphing foil design to reduce torque ripple of a VAWT. In the course of this areas were identified that should be expanded upon:

- The CFD model was developed and validated based on mesh refinement and appropriate turbulence model selection. However, it is further recommended to verify and validate it with wind tunnel experiments.
- Material selection for a flexible skin is required to ensure the material does not deform naturally but is compliant enough to allow for some stretch in length.
- The change in height of the distortion was modelled in stages. Each height was meshed individually and run with the steady state solver simpleFOAM. A dynamic mesh and transient simulation would be a more accurate representation of the envisioned system.
- The use of a single distortion proved effective, but the logical step forward would be investigating the use of more than one distortion and the resultant changes in lift and drag
- Instead of a time dependent variation a “smart” blade could use feedback from pressure on the foil surface to change shape thus removing the need for a yaw system.

Appendices

Appendix A

Double Multiple Stream Tube Model

A.1 Analytical Model

This section describes the maths behind the code used to build the analytical model. It uses BEM theory as applied to Vertical axis wind turbines. Section A.1.1 describes the actuator disk model based on conservation of energy, conservation of momentum and conservation of mass. Section A.1.2 then looks at the same situation from an aerodynamic loading approach. These two systems are used to obtain equations for thrust on the foil in a flow regime. These two equations are then compared to identify the induction values to be applied at any given position.

A.1.1 Actuator Disk Model

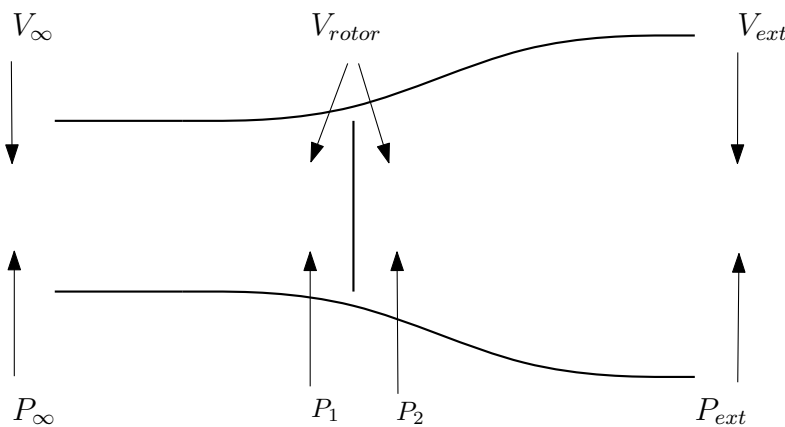


Figure A.1: Control volume used in actuator disk model

Figure A.1 represents a simplified model for flow past an airfoil. The disk is seen as an ideal actuator with a pressure drop before and after the blade. Upstream from the blade is the original wind speed and after is the reduced exit velocity of the wind. Using Bernoulli's equation for frictionless, incompressible expressions for the energy balance before (equation A.1.1) and after (equation A.1.2) the rotor can be written as

$$\frac{1}{2}\rho V_{\infty}^2 + P_{\infty} = \frac{1}{2}\rho V_{rotor}^2 + P_1 \quad (\text{A.1.1})$$

$$\frac{1}{2}\rho V_{rotor}^2 + P_2 = \frac{1}{2}\rho V_{ext}^2 + P_{ext} \quad (\text{A.1.2})$$

Pressure far downstream is equal to pressure far upstream, $P_{ext} = P_{\infty}$, the equation for the pressure drop across the disk is

$$P_1 - P_2 = \frac{1}{2}\rho (V_{\infty}^2 - V_{ext}^2) \quad (\text{A.1.3})$$

Assuming the disk to be the only energy absorber, the change in axial momentum is equal to the force on the rotor.

$$\dot{m} (V_{\infty} - V_{ext}) = \Delta P A \quad (\text{A.1.4})$$

Assuming the velocity at the rotor is a multiple of wind speed

$$V_a = V_{\infty} u \quad (\text{A.1.5})$$

and using equation A.1.3 equation A.1.4 can be rewritten as

$$\rho A V_{\infty} u (V_{\infty} - V_{ext}) = \frac{1}{2}\rho (V_{\infty}^2 - V_{ext}^2) A \quad (\text{A.1.6})$$

By cancelling and collecting terms the exit velocity can be stated in terms of the original velocity

$$V_{ext} = V_{\infty} (2u - 1) \quad (\text{A.1.7})$$

Looking at the force on the disk, the thrust is defined as the force in the direction of the original wind flow on the disk. Using the definition from equation A.1.7 and A.1.4, thrust is written as

$$T = \frac{1}{2}\rho [V_{\infty}^2 - (V_{\infty}(2u - 1))^2] A \quad (\text{A.1.8})$$

and reduces to

$$T = \frac{1}{2}\rho V_{\infty}^2 A 4u(1 - u) \quad (\text{A.1.9})$$

A.1.2 Aerodynamic loading

During a single rotation of a VAWT the angle of attack as seen by the airfoil changes in relation to the azimuthal angle. Figure A.2 shows the wind direction, rotational speed vector and azimuthal angle. The mathematical description below is valid for the upper right quadrant but similar proofs can be done for the remaining three quadrants.

To start off a few definitions are needed, the relative velocity of the wind to the blade is

$$\vec{W} = \vec{V}_\infty - \omega \vec{R} \quad (\text{A.1.10})$$

shown graphically in figure A.3. Note that $\omega \vec{R} == V_{rotor}$ defined in equation A.1.5 hereafter referred to as V_r . Rotor swept area is defined as

$$S = D \times h \quad (\text{A.1.11})$$

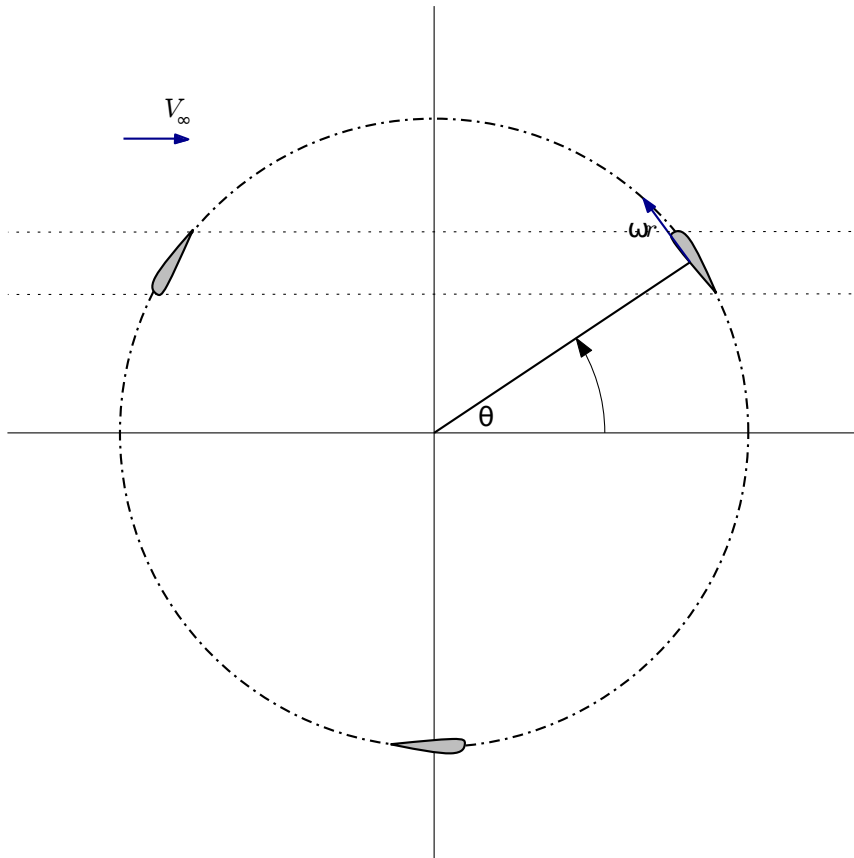


Figure A.2: Physical model for VAWT

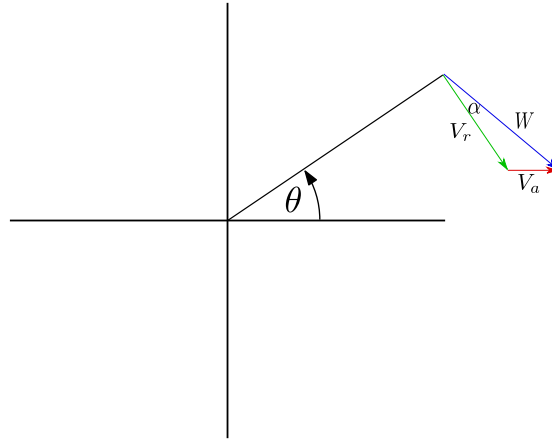


Figure A.3: Relative velocity diagram of VAWT

and finally two tip speed ratios

$$\lambda = \frac{V_r}{V_\infty} \quad (\text{A.1.12})$$

$$X = \frac{V_r}{V_a} \quad (\text{A.1.13})$$

λ is overall tip speed ratio and X is the local tip speed ratio. The magnitude of W and the angle of attack (α) are determined from the following equations

$$W = V_a \sqrt{X + 2X \sin \theta + 1} \quad (\text{A.1.14})$$

$$\alpha = \text{asin} \left[\frac{\cos(\theta)}{W} V_a \right] \quad (\text{A.1.15})$$

and based on figure A.3.

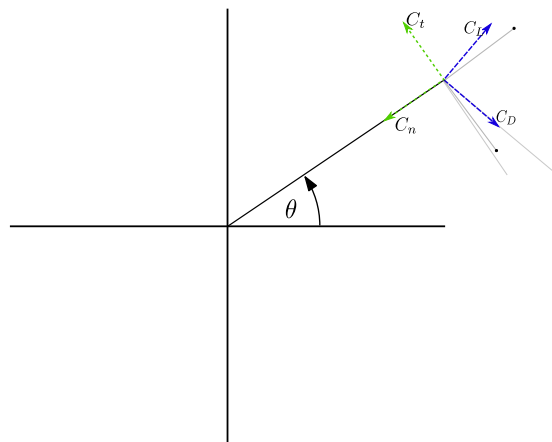


Figure A.4: Lift and drag vectors due to relative wind velocity

Figure A.4 highlights the axes used to define lift, drag, normal and tangential directions. Normalizing the lift and drag forces, their coefficients are then used to define the coefficients in the normal and tangential directions.

$$C_t = C_l \sin \alpha - C_d \cos \alpha \quad (\text{A.1.16})$$

$$C_n = -C_d \cos \alpha - C_l \sin \alpha \quad (\text{A.1.17})$$

By taking the x components of C_t and C_n the azimuthal angle can be used to define the thrust, as shown in figure A.5.

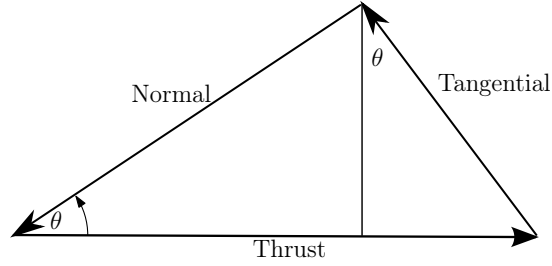


Figure A.5: Decomposing normal and tangential coefficients into thrust

By assuming h to be the depth of the foil thrust can be written as follows:

$$T = \frac{1}{2} \rho W^2 ch [-C_n \cos \theta - C_t \sin \theta] \quad (\text{A.1.18})$$

Figure A.6 is used to define the inlet area under consideration when calculating thrust

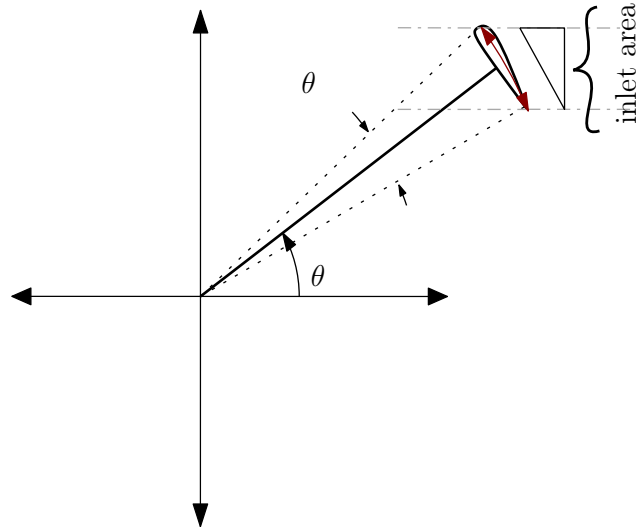


Figure A.6: Streamtube based on airfoil shape

The inlet area is based on azimuth angle and the geometric factor $\Delta\theta$ which is calculate form the following equation

$$\frac{\Delta\theta}{2} = \text{atan} \frac{c}{R} \quad (\text{A.1.19})$$

Normalizing equation A.1.18 results in

$$C_T = \left(\frac{W}{V_\infty} \right)^2 \frac{c}{R\Delta\theta} [-C_n \cos\theta - C_t \sin\theta] \frac{1}{|\cos\theta|} \quad (\text{A.1.20})$$

Normalizing the thrust from the our actuator disk model, equation A.1.9, and comparing it to equation A.1.20 results in

$$4u(1-u) = \frac{W^2}{V_\infty^2} \frac{c}{R\Delta\theta} \frac{-C_n \cos\theta - C_t \sin\theta}{|\cos\theta|} \quad (\text{A.1.21})$$

Average torque is defined as

$$T_{ave} = T_{instant} \frac{N \Delta\theta}{2 \pi} \quad (\text{A.1.22})$$

after where N is the number of blades. Using the average torque equation A.1.21 becomes

$$4u(1-u) = u^2 \frac{W^2}{V_a^2} \frac{Nc}{R2\pi} \frac{-C_n \cos\theta - C_t \sin\theta}{|\cos\theta|} \quad (\text{A.1.23})$$

Next a geometric factor

$$k = \frac{8\pi R}{Nc} \quad (\text{A.1.24})$$

is defined, which is based solely on the turbine geometry. Equation A.1.23 is then

$$k(1-u) |\cos\theta| \frac{1}{u} = \frac{W^2}{V_a^2} [-C_n \cos\theta - C_t \sin\theta] \quad (\text{A.1.25})$$

Finally letting the RHS of the equation equal $f(\theta)$ and solving for u

$$\frac{k|\cos\theta|}{f(\theta) + k|\cos\theta|} = u \quad (\text{A.1.26})$$

Note, in the upwind zone α is negative while downwind it has a positive sign. This explains the sign change in determining the normal and tangential coefficients as shown in Figure A.8.

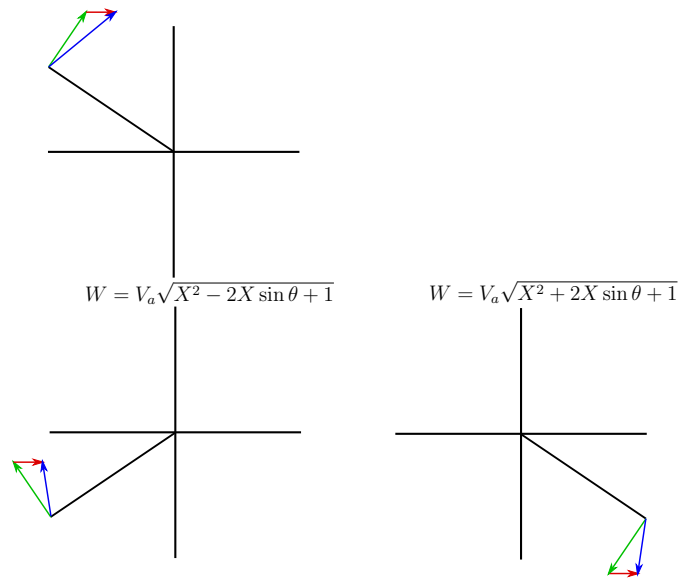


Figure A.7: Relative velocity equations for quadrants

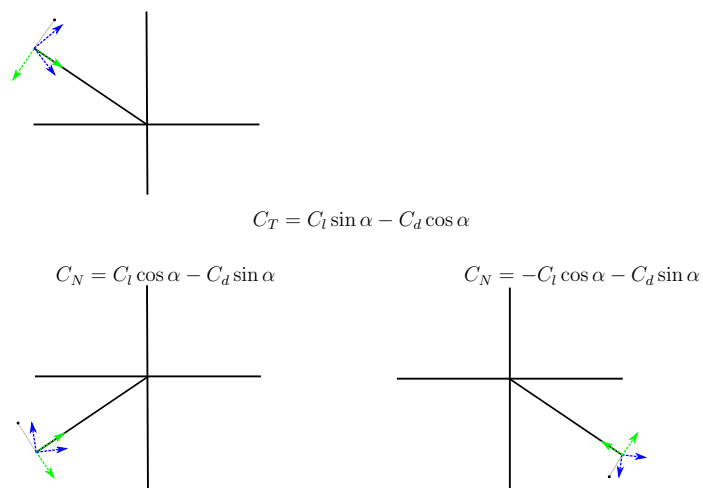


Figure A.8: Normal and tangential equations for quadrants

Appendix B

Surrogate Modelling

A surrogate model provides a fast, robust method of obtaining solutions to a more complex mathematical system. There are numerous regression techniques available and the choice for use is dependent on the problem at hand. In building such models it is important to have an understanding of the underlying trends so as to help identify which modelling technique is most applicable and will yield the best results. Additional factors to be considered include, but are not limited to, computational resources, accuracy and available data. The three techniques investigated in this project were response surfaces, support vector regression and neural networks.

B.1 Response surfaces

Information for this section was taken from Meyers and Montgomery (2002). This methodology consists of exploring the space of independent variables, to develop an appropriate approximate relationship between said variables and a predefined response. Then undertaking optimization to adjust the response to better fit the variables. A response function is written in the form

$$\eta = f(x_1, x_2, x_3, \dots, x_j) \quad (\text{B.1.1})$$

and can have the independent variables redefined into coded variables, all having the same standard deviation and zero mean (values between -1 and 1), for efficiency. The exact relationship f is unknown and must be approximated. Typically a low order polynomial is fitted through least squares approximation to determine the coefficients. In equation B.1.2 the coefficients needing to be determined for a first order “main effects” model are represented by β_i

$$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \quad (\text{B.1.2})$$

It is termed main effects because it only uses the independent variables. A first order model is usually sufficient when trying to approximate only a small region

of space within the independent variables. This assumes little curvature within that region. If interaction between variables has to be accounted equation B.1.2 can be extended to

$$\eta = \beta_0 + \beta_1x_1 + \beta_2x_2 + \beta_{12}x_1x_2 \quad (\text{B.1.3})$$

Term 4 in equation B.1.3 can introduce some curvature but it is unlikely to be sufficient for many responses. The model is then taken into a 2nd order description with interaction as shown in the following equation

$$\eta = \beta_0 + \beta_1x_1 + \beta_2x_2 + \beta_{11}x_1^2 + \beta_{22}x_2^2 + \beta_{12}x_1x_2 \quad (\text{B.1.4})$$

In general a 2nd order model, seen in equation B.1.5, is popular due to its ability to take on a multitude of forms, the ease with which the coefficients can be calculated, the presence of a single optimum and previous experience indicating that it works well for solving real response surface problems.

$$\eta = \beta_0 + \sum_{j=1}^k \beta_jx_j + \sum_{j=1}^k \beta_{jj}x_j^2 + \sum_{i<j=2}^k \sum_{i=1}^k \beta_{ij}x_ix_j \quad (\text{B.1.5})$$

This methodology attempts to fit low order polynomials to data. The order of polynomial is selected based on assumptions of the underlying trend within the data. The coefficients are determined with a least squares fit.

B.2 Neural networks

Artificial neural networks are inspired by the way a brain operates. The interconnected neurons of our brain take information from neighbouring neurons causing it to react or not. In this predictive model artificial neurons perform a similar function. The problem with this approach is that it is generally considered a black box and understanding how the solution was obtained is not always possible.

The more common types of neural networks are feed forward. These consist of discrete layers of neurons each connected to the next. The input layer takes the supplied data and passes it unaltered to the next layer. Each layer takes the input and performs a calculation on the data, passing the output to the next layer. Each neuron has a weight and bias attached to the input. The products of inputs and weights are summed to give an output to either the next layer or the final solution. The output is typically a smooth function, to aid in the training of the model. Additionally we can use back propagation to adjust the weighting for each of neurons and reduce the error of the regressor.

B.3 Random forests

Information for this section was taken from Breiman (2001)

Random forests makes use of a collection of tree structured classifiers. Tree

classifiers proposed by Morgan and Sonquist (1963), try to minimize the difference within a group by subdividing the original set into smaller subgroups so that the variation in each is smaller than before. The implemented random forests are tree regressors but for the purposes of explanation the description below is based on classification.

With a set of data N of which each member (x) is labelled from the set $L_{1..N}$, the entropy of N is defined as

$$H(N) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 \dots - p_n \log_2 p_n \quad (\text{B.3.1})$$

where p_i is the portion of data with the label L_i . If subsets of N , $N_1 \dots N_m$, are created that each contain portions $q_1 \dots q_m$ the total entropy is defined as

$$H = q_1 H(N_1) + \dots + q_m H(N_m) \quad (\text{B.3.2})$$

A tree is said to contain decision and leaf nodes. The former directs the choices available and the latter proves an answer. To create the tree the following process is followed:

- a If all data within the group has the same label, create a leaf node and quit
- b If there are no more attributes on which to split a group, create a leaf node that predicts the most common label
- c If (a) and (b) are false, split the group so that equation B.3.2 is minimized and create a decision node based on the attribute selected
- d Repeat on each new subset

The random forest approach is to build multiple trees on the same data set and obtaining an answer based on the collective vote from all trees.

For regression purposes the random forest is made by growing trees on a random vector Θ such that the predictor $h(x, \Theta)$ outputs a numerical value instead of a label.

B.4 Support vector regression

Information for this section was taken from Smola and Schölkopf (2004)

Assume a training data set described as $\{(\vec{x}_1, y_1) \dots (\vec{x}_i, y_i)\}$, where \vec{x}_i is a vector of training points and y_i the corresponding target value. With support vector regression the function

$$f(x) = \langle \vec{w} \cdot \vec{x} \rangle + b \quad (\text{B.4.1})$$

is fitted so that the maximum deviation is ϵ and the gradient is as flat as possible. A small gradient implies a small \vec{w} allowing framing the problem as

a convex optimisation problem

$$\text{minimize} : \frac{1}{2} \|\vec{w}\|^2 \quad (\text{B.4.2})$$

$$\text{such that} \begin{cases} y_i - \langle \vec{w} \cdot \vec{x} \rangle - b \leq \epsilon \\ \langle \vec{w} \cdot \vec{x} \rangle + b - y_i \leq \epsilon \end{cases} \quad (\text{B.4.3})$$

assuming f exists for all pairs (\vec{x}_i, y_i) with ϵ precision. To compensate for the possibility of f not being feasible or just to allow for errors two new variables ξ_i, ξ_i^* are added. This adjusts B.4.3 to

$$\text{minimize} : \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^l (\xi_i + \xi_i^*) \quad (\text{B.4.4})$$

$$\text{such that} \begin{cases} y_i - \langle \vec{w} \cdot \vec{x} \rangle - b \leq \epsilon + \xi_i \\ \langle \vec{w} \cdot \vec{x} \rangle + b - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \quad (\text{B.4.5})$$

This formulation has a constant C , for setting the trade off between the flatness of f and the tolerance for deviations larger than ϵ .

B.4.1 Dual formulation

Dual formulation helps to deal with non-linear function approximations, but even linear functions are more easily solved using the dual formulation assuming the dimensionality of w is much higher than the observations.

$$\begin{aligned} L = & \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^l (\xi_i + \xi_i^*) - \sum_{i=1}^l (\eta_i \xi_i + \eta_i^* \xi_i^*) \\ & - \sum_{i=1}^l \alpha_i (\epsilon + \xi_i - y_i + \langle \vec{w} \cdot \vec{x} \rangle + b) \\ & - \sum_{i=1}^l \alpha_i^* (\epsilon + \xi_i^* - y_i - \langle \vec{w} \cdot \vec{x} \rangle - b) \end{aligned} \quad (\text{B.4.6})$$

The Lagrangian L has Lagrange multipliers $\eta_i, \eta_i^*, \alpha_i, \alpha_i^*$ implying in B.4.6

$$\eta_i, \eta_i^*, \alpha_i, \alpha_i^* \geq 0 \quad (\text{B.4.7})$$

this function has a saddle point with respect to the original variables (w, b, ξ_i, ξ_i^*) , thus the partial derivatives of L with respect to these variables turns to 0.

$$\frac{\partial L}{\partial b} = \sum_{i=1}^l (\alpha_i^* - \alpha_i) = 0 \quad (\text{B.4.8})$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^l (\alpha_i - \alpha_i^*) x_i = 0 \quad (\text{B.4.9})$$

$$\frac{\partial L}{\partial \xi_i^{(*)}} = C - \alpha_i^{(*)} - \eta_i^{(*)} \quad (\text{B.4.10})$$

If we then substitute B.4.8, B.4.9 and B.4.10 into B.4.5 we get the final dual formation:

$$\text{minimize : } \begin{cases} -\frac{1}{2} \sum_{i,j=1}^l (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \langle \vec{x}_i \cdot \vec{x}_j \rangle \\ -\epsilon \sum_{i=1}^l (\alpha_i + \alpha_i^*) + \sum_{i=1}^l y_i (\alpha_i - \alpha_i^*) \end{cases} \quad (\text{B.4.11})$$

$$\text{subject to } \sum_{i=1}^l (\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C] \quad (\text{B.4.12})$$

Equation B.4.12 allows for the complete description through w of training pattern x_i but we still have to allow for non-linearity.

Appendix C

Optimization

C.1 Introduction

Modelling is the process of defining variables, constraints and objectives for a given problem. This is the first step in optimization, which is the process of adjusting parameters of a system to achieve a certain objective and satisfy constraints. If a function f is made up of the vector \vec{x} , with constraints c , optimisation can be defined as

$$\min_{x \in \mathbb{R}^n} f(\vec{x}) \text{ subject to } \begin{cases} c_i(\vec{x}) = 0, i \in A \\ c_j(\vec{x}) \geq 0, j \in B \end{cases} \quad (\text{C.1.1})$$

the minimization (maximisation) of the function subject to constraints on its variables. In equation C.1.1 A is a set of indices for equality while B is a set of indices for inequality (Nocedal and Wright, 2006).

A further important distinction within optimisation is that of constrained and unconstrained optimization. In unconstrained optimization $A = B = \emptyset$. Constrained problems occur when the constraints play an important role in the design problem. Optimization can also involve side constraints, that are applied to the vector \vec{x} . These set limits on the components in \vec{x}

C.2 Breeder Genetic algorithm(BGA)

The information for this section was taken from (Schlierkamp-Voosen and Mühlenbein, 1993)

This type of genetic algorithm is based on rational or artificial selection as performed by human breeders. It can be viewed as a combination of evolution strategies and genetic algorithms. It is a search method applicable to both discrete and continuous functions.

$$BGA = (P_0, N, E, \aleph_m, \aleph_r, f(), \Omega) \quad (\text{C.2.1})$$

Equation C.2.1 describes the input requirements for a BGA. P_0 is the initial population, N is the population, E is the number of elites or truncation threshold, α_r is the selection rate, α_m is the mutation rate and Ω is the termination criteria. The following sections describe some of the more unique parameters of a BGA.

C.2.1 Elites

The elites represent the top percentage of the population which are randomly mated till the number of progeny plus the elite members equal the population size. This new population replaces the parent's population.

C.2.2 Selection

This is the operator that combines the genetic material of two parents. The selection method involves a choice of recombination scheme and the selection of which members are used for parental genetic material. If $\vec{x} = [x_1, x_2, \dots, x_n]$ and $\vec{y} = [y_1, y_2, \dots, y_n]$ are the parent vectors then the child $\vec{z} = [z_1, z_2, \dots, z_n]$ can be made up by using the following recombination schemes:

Discrete selection

A single parent is selected, with a 50 % probability, to pass on their genetic material

$$z_i = \{x_i\} \text{ or } \{y_i\} \quad (\text{C.2.2})$$

Extended intermediate selection

A line in the variable workspace is created to connect the parent vectors. The child is then made up by travelling along this line and stopping randomly (α_1) along the interval $[-0.25, 1.25]$. The interval allows for overshoot and therefore inclusion of new genetic material.

$$z_i = x_i + \alpha_1(y_i - x_i) \quad (\text{C.2.3})$$

Portioned selection

A fixed portion of consecutive variables $x_i \rightarrow x_{\alpha_2}$ is selected from one parent while the remaining variables of the child are populated from the second parent.

$$z = x[x_{1 \rightarrow \alpha_2}, y_{\alpha_2+1 \rightarrow n}] \quad \alpha_2 \in [1, j] \quad j < n \quad (\text{C.2.4})$$

C.2.3 Mutation

A member of the population has its genetic material mutated through random selection. The portion of material undergoing mutation is dictated by α_3 and

the mutation rate M can be a fixed value or adapt to the population status.

$$x_i = x[x_{1 \rightarrow \alpha_3} M_1, x_{\alpha_3 \rightarrow n}] \alpha_3 \in [1, n] \quad (\text{C.2.5})$$

Adaptive mutation

The mutation variable M can be adjusted based on the fitness of the population. This measure is found by taking the function value of the top half of the population and comparing it to the lower half. If the average of the upper half is greater than the average of the lower half the mutation rate is decreased.

$$M_{i+1} = M_i A \begin{cases} A > 1, \overline{f(x_{1 \rightarrow N/2})} < \overline{f(x_{N/2 \rightarrow N})} \\ A < 1, \overline{f(x_{1 \rightarrow N/2})} > \overline{f(x_{N/2 \rightarrow N})} \end{cases} \quad (\text{C.2.6})$$

C.3 Combined schemes

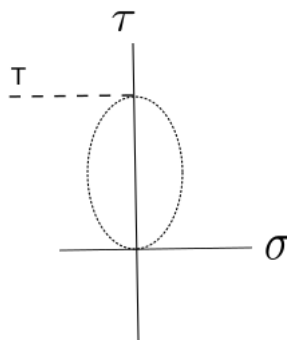
Mutation works on chance and is very effective in small populations. Recombination however, works on restricted chance. In shuffling around pieces of the existing population, the pieces that make up the optimum solution would need to be present within that population. The combination of both mutation and recombination has been shown to be synergistic. In large populations recombination will be the dominant factor until the homogeneity of the population increases.

Appendix D

Life span calculations

D.1 Modified Goodman criterion

Using Mohrs circle we can relate the torque experienced by a shaft directly to stress.



In keeping all other factors constant we can use the modified Goodman criterion to estimate the life cycle of the shaft

$$\sigma_a = \frac{\sigma_{max} + \sigma_{min}}{2} \quad (D.1.1)$$

$$\sigma_m = \frac{\sigma_{max} - \sigma_{min}}{2} \quad (D.1.2)$$

$$\frac{\sigma_a}{S_e} + \frac{\sigma_m}{S_{ut}} = \frac{1}{n} \quad (D.1.3)$$

also note that for an ultimate tensile strength less than 1400MPa the endurance limit can be estimated as

$$S_e \approx 0.5S_{ut} \quad (S_{ut} < 1400) \quad (D.1.4)$$

Using equations D.1.2 and D.1.3 and substituting the original torque (T_1)

$$\sigma_a = \frac{6T_1}{20} \quad (\text{D.1.5})$$

$$\sigma_m = \frac{4T_1}{20} \quad (\text{D.1.6})$$

and the reduced torque ($\frac{3T_1}{4}$)

$$\sigma_a = \frac{19T_1}{80} \quad (\text{D.1.7})$$

$$\sigma_m = \frac{11T_1}{80} \quad (\text{D.1.8})$$

we can define the equation D.1.3 in terms of the torque experienced in each case.

$$n_{org} = \frac{40S_{ut}}{14T_1} \quad n_{opt} = \frac{160S_{ut}}{41T_1}$$

We therefore get a 36% increase in life span

$$\frac{n_{org}}{n_{opt}} = \frac{41}{56} \quad (\text{D.1.9})$$

Appendix E

Python code

```

from pylab import *
import os
import sys
from NURB import *
import matplotlib.pyplot as plt
import scipy as sp
import stlout as stl
import foilpy as fp
import nacaFiles as nf
from scipy.optimize import curve_fit, leastsq, least_squares, fsolve,
    bisect
from scipy import interpolate
from scipy.spatial import distance
import time
import pandas as pd
import newHope as hope

def meshDict_Ch(C1,C2,C3,C4,R,l1,l2,width): #1 - 4 is Upper TE, Upper
    Le lower Le, lower TE
    sf = 10 #float(raw_input("inner shell scale: "))#10
    sf2 = 2 #float(raw_input("Outer shell scale: ")) #2
    alpha1 = sp.arcsin(C2[l1][1]/C2[l1][0])
    alpha2 = sp.arcsin(C2[l2][1]/C2[l2][0])
    '''trying to get perpendicular lines!!!'''
    mTE = -(C2[l1+1][0] - C2[l1-1][0])/(C2[l1+1][1] - C2[l1-1][1])
    cTE = C2[l1][1] - C2[l1][0]*mTE
    mLE = -(C2[l2+1][0] - C2[l2-1][0])/(C2[l2+1][1] - C2[l2-1][1])
    cLE = C2[l2][1] - C2[l2][0]*mLE
    '''y = mx + c, sub y of 40 to see projection'''
    interceptTEr = (-2*mTE*cTE - sp.sqrt((2*mTE*cTE)**2 -
        4*(1+mTE**2)*(cTE**2 - R**2)))/(2*(1+mTE**2))

```

```

interceptLExR = (-2*mLE*cLE - sp.sqrt((2*mLE*cLE)**2 -
    4*(1+mLE**2)*(cLE**2 - R**2)))/(2*(1+mLE**2))
interceptTExRsf = (-2*mTE*cTE - sp.sqrt((2*mTE*cTE)**2 -
    4*(1+mTE**2)*(cTE**2 - (R/sf)**2)))/(2*(1+mTE**2))
interceptLExRsf = (-2*mLE*cLE - sp.sqrt((2*mLE*cLE)**2 -
    4*(1+mLE**2)*(cLE**2 - (R/sf)**2)))/(2*(1+mLE**2))
pt10 = [interceptTExRsf,mTE*interceptTExRsf + cTE]
pt11 = [interceptLExRsf,mLE*interceptLExRsf + cLE]
pt22 = [interceptTExR,mTE*interceptTExR + cTE]
pt23 = [interceptLExR,mLE*interceptLExR + cLE]
points = [C1[0],C2[0],C2[11],C2[12],C3[0],C4[0]] #foil
Layer1 =
    [[R/sf,0],[R/sf,R/sf],[C1[0][0],R/sf],[C2[0][0],R/sf],pt10,pt11,\
[-R/sf,0],[C4[0][0],-R/sf],[C1[0][0],-R/sf],[R/sf,-R/sf]]
outside = [[R,0],[R,R/sf],[R,R],[R/sf,R],[C1[0][0],R],\
[C2[0][0],R],pt22,pt23,[-R,0],[C4[0][0],-R],[C1[0][0],-R]\
,[R/sf,-R],[R,-R],[R,-R/sf]]
TESTarray =
    sp.vstack([sp.array(points),sp.array(Layer1),sp.array(outside)])
lBump = curveL(C2[11:12])
lLEstart = curveL(C2[0:11])
lLEend = curveL(C2[12::])
lTE = curveL(C1)
lLE = curveL(C3)
arc11_12 = make_arc(TESTarray[11],TESTarray[12],R/sf)
arc23_24 = make_arc(TESTarray[23],TESTarray[24],R)
arc9_10 = make_arc(TESTarray[9],TESTarray[10],R/sf)
arc10_11= make_arc(TESTarray[10],TESTarray[11],R/sf)
arc22_23 = make_arc(TESTarray[22],TESTarray[23],R)
arc21_22 = make_arc(TESTarray[21],TESTarray[22],R)
arc24_25 = R*sp.pi/2
arc12_13 = R/sf*sp.pi/2
plt.plot(C2[0:11,0],C2[0:11,1], 'r')
plt.plot(C2[11:12,0],C2[11:12,1], 'y')
plt.plot(C2[12::,0],C2[12::,1], 'g')
input_width = 5e-4
guess1 = int(lBump/input_width) #number of cells so that width == 5e-5
LEU1 = 1300
LEU2 = 60
TEcells = 50
TEcellsL = 100
LECells = 40
yCells = 200
wake1 = 400
wake2 = 200
happy = 'n'

```

```

start = 'y'
number = 0
while happy == 'n':
    flag = 0
    one_1 = 1.0/sp.optimize.fmin(minimizer_first,1,\
args=(lLEend,LEU1,input_width),disp=0)[0]
    if (1.0/one_1)**(1.0/(LEU1-1)) > 1.1:
        LEU1 = LEU1 + 3
    print "-----increase-----"
    print "current 1_1: ", (1.0/one_1)**(1.0/(LEU1-1))
    flag = 1
    if (1.0/one_1)**(1.0/(LEU1-1)) < 0.991:
        LEU1 = LEU1 -1
    print "-----decrease-----"
    print "current 1_1: ", (1.0/one_1)**(1.0/(LEU1-1))
    flag = 1
    shellwidth = arc10_11/guess1
    zero_1 = 1.0/sp.optimize.fmin(minimizer_first,1,\
args=(arc11_12,LEU1,shellwidth),disp=0)[0]
    if (1.0/zero_1)**(1.0/(LEU1-1)) > 1.1:
        LEU1 = LEU1 + 3
    print "-----increase-----"
    print "current 0_1: ", (1.0/zero_1)**(1.0/(LEU1-1))
    flag = 1
    if (1.0/zero_1)**(1.0/(LEU1-1)) < 1:
        LEU1 = LEU1 -1
    print "-----decrease-----"
    print "current 0_1: ", (1.0/zero_1)**(1.0/(LEU1-1))
    flag = 1
    boundWidth = arc22_23/guess1
    zero_2 = 1.0/sp.optimize.fmin(minimizer_first,1,\
args=(arc23_24,LEU1,boundWidth),disp=0)[0]
    if (1.0/zero_2)**(1.0/(LEU1-1)) > 1.1:
        LEU1 = LEU1 + 3
    print "-----increase-----"
    print "current 0_2: ", (1.0/zero_2)**(1.0/(LEU1-1))
    flag = 1
    if (1.0/zero_2)**(1.0/(LEU1-1)) < 1:
        LEU1 = LEU1 -1
    print "-----decrease-----"
    print "current 0_2: ", (1.0/zero_2)**(1.0/(LEU1-1))
    flag = 1

five_1 = sp.optimize.fmin(minimizer_first,1,\
args=(lLEstart,LEU2,input_width),disp=0)[0]
if (five_1)**(1.0/(LEU2-1)) > 1.001:

```

```

LEU2 = LEU2 + 3
print "-----increase-----"
print "current 5_1: ", (five_1)**(1.0/(LEU2-1))
flag = 1
if (five_1)**(1.0/(LEU2-1)) < 0.995:
LEU2 = LEU2 -1
print "-----decrease-----"
print "current 5_1: ", (five_1)**(1.0/(LEU2-1))
flag = 1

five_2 = sp.optimize.fmin(minimizer_first,1,\
args=(arc9_10,LEU2,shellwidth),disp=0)[0]
if (five_2)**(1.0/(LEU2-1)) > 1.001:
LEU2 = LEU2 + 3
print "-----increase-----"
print "current 5_2: ", (five_2)**(1.0/(LEU2-1))
flag = 1
if (five_2)**(1.0/(LEU2-1)) < 0.993:
LEU2 = LEU2 -1
print "-----decrease-----"
print "current 5_2: ", (five_2)**(1.0/(LEU2-1))
flag = 1

four_2 = sp.optimize.fmin(minimizer_first,1,\
args=(arc21_22 ,LEU2,boundWidth),disp=0)[0]
if (four_2)**(1.0/(LEU2-1)) > 1.01:
LEU2 = LEU2 + 3
print "-----increase-----"
print "current 4_2: ", (four_2)**(1.0/(LEU2-1))
flag = 1
if (four_2)**(1.0/(LEU2-1)) < 0.991: #changed to 0.99287 for 0.05
    -loc 0.8-d, originally 0.993
LEU2 = LEU2 -1
print "-----decrease-----"
print "current 4_2: ", (four_2)**(1.0/(LEU2-1))
flag = 1

TEguess = int(floor(1TE/(input_width*five_1))) #cells in TE portion

noseCellWidth = input_width/one_1
thirteen_2 = sp.optimize.fmin(minimizer_first,1,\
args=(1LE,LECells,noseCellWidth),disp=0)[0]
if (thirteen_2)**(1.0/(LECells-1)) > 1.1:
LECells = LECells + 3
print "-----increase-----"
print "current 13_2: ", (thirteen_2)**(1.0/(LECells-1))

```



```

flag = 1
if (thirteen_2)**(1.0/(LECells-1)) < 1:
LECells = LECells - 1
print "-----decrease-----"
print "current 13_2: ", (thirteen_2)**(1.0/(LECells-1))
flag = 1

fifteen_2 = sp.optimize.fmin(minimizer_first,1,\
args=(1TE,TEcellsL,noseCellWidth*thirteen_2),disp=0)[0]
if (fifteen_2)**(1.0/(TEcellsL-1)) > 1.1 or
noseCellWidth*thirteen_2*fifteen_2 > input_width*five_1*1.1:
TEcellsL = TEcellsL + 3
print "-----increase-----"
print "current 15_2: ", (fifteen_2)**(1.0/(TEcellsL-1))
flag = 1
if (thirteen_2)**(1.0/(TEcellsL-1)) < 1:
TEcellsL = TEcellsL - 1
print "-----decrease-----"
print "current 15_2: ", (fifteen_2)**(1.0/(TEcellsL-1))
flag = 1
thirteen_1 = 1.0/sp.optimize.fmin(minimizer_first,1,\
args=(arc12_13,LECells,sp.absolute(TEStarray[13][0]
-TEStarray[14][0])/TEcellsL),disp=0)[0]
twelve_1 = 1.0/sp.optimize.fmin(minimizer_first,1,args=(arc24_25
,LECells,sp.absolute(TEStarray[25][0]
-TEStarray[26][0])/TEcellsL),disp=0)[0]

one_5 = sp.optimize.fmin(minimizer_first,1,\
args=(sp.absolute(TEStarray[11][1]
-TEStarray[3][1]),yCells,3e-5),disp=0)[0]
if (one_5)**(1.0/(yCells-1)) > 1.1:
yCells = yCells + 3
print "-----increase-----"
print "current 13_2: ", (one_5)**(1.0/(one_5-1))
flag = 1
if (one_5)**(1.0/(one_5-1)) < 1:
yCells = yCells - 1
print "-----decrease-----"
print "current 13_2: ", (one_5)**(1.0/(one_5-1))
flag = 1

thirteen_6 = 1.0/sp.optimize.fmin(minimizer_first,1,\
args=(sp.absolute(TEStarray[5][1]
-TEStarray[13][1]),yCells,3e-5),disp=0)[0]
if (1.0/thirteen_6)**(1.0/(yCells-1)) > 1.1:
yCells = yCells + 3

```

```

print "-----increase-----"
print "current 13_6: ", (1.0/thirteen_6)**(1.0/(yCells-1))
flag = 1
if (1.0/thirteen_6)**(1.0/(yCells-1)) < 1:
yCells = yCells -1
print "-----decrease-----"
print "current 13_6: ", (1.0/thirteen_6)**(1.0/(yCells-1))

three_6 = sp.optimize.fmin(minimizer_first,1,\
args=(sp.absolute(TESTarray[10][1]
      -TESTarray[2][1]),yCells,3e-5),disp=0)[0] #height!
five_6 = sp.optimize.fmin(minimizer_first,1,\
args=(sp.absolute(TESTarray[9][1]
      -TESTarray[1][1]),yCells,5e-5),disp=0)[0] #height!
seven_6 = sp.optimize.fmin(minimizer_first,1,\
args=(sp.absolute(TESTarray[8][1]
      -TESTarray[0][1]),yCells,7e-5),disp=0)[0] #height!

nine_1 = sp.optimize.fmin(minimizer_first,1,\
args=(sp.absolute(TESTarray[6][0]
      -TESTarray[0][0]),wake1,input_width*five_1),disp=0)[0]
if (nine_1)**(1.0/(wake1-1)) > 1.01:
wake1 = wake1 + 3
print "-----increase-----"
print "current 9_1: ", (nine_1)**(1.0/(nine_1-1))
flag = 1
if (nine_1)**(1.0/(nine_1-1)) < 1:
wake1 = wake1 - 1
print "-----decrease-----"
print "current 9_1: ", (nine_1)**(1.0/(nine_1-1))
flag = 1

eleven_1 = sp.optimize.fmin(minimizer_first,1,\
args=(sp.absolute(TESTarray[16][0]
      -TESTarray[6][0]),wake2,input_width*five_1*nine_1),disp=0)[0]
if (eleven_1)**(1.0/(wake2-1)) > 1.01:
wake2 = wake2 + 3
print "-----increase-----"
print "current 11_1: ", (eleven_1)**(1.0/(eleven_1-1))
flag = 1
if (eleven_1)**(1.0/(eleven_1-1)) < 1:
wake2 = wake2 - 1
print "-----decrease-----"
print "current 11_1: ", (eleven_1)**(1.0/(eleven_1-1))
flag = 1

```

```

nine_6 = sp.optimize.fmin(minimizer_first,1,\
args=(sp.absolute(TEStarray[8][1]
      -TEStarray[0][1]),yCells,5e-4),disp=0)[0] #height!

if flag == 1:
happy = 'n'
else:
happy = 'y'
number = number +1
print "-----number ", number
'''detmining points for arc'''
thetaTEsf = sp.arctan(pt10[1]/pt10[0])
thetaLEsf = sp.arctan(pt11[1]/pt11[0])
thetaBump = (thetaTEsf - thetaLEsf)/2 + thetaLEsf
thetaUP = (sp.pi/2 - thetaTEsf)/2 + sp.pi/2
'''need lengths to do grading!'''

B_dict={}
B_dict['header']='''/*-----* C++
  *-----*\
| ===== |
| |
| \\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox
| |
| \\ / O p e r a t i o n | Version: 2.2.2
| |
| \\ / A n d | Web: www.OpenFOAM.org
| |
| \\ / M a n i p u l a t i o n |
| |
\*-----
-----*/

FoamFile
{
version 2.0;
format ascii;
class dictionary;
object blockMeshDict;
}
// * * * * *
* * * * //
convertToMeters 1;
vertices
(
'''
B_dict['block'] = '''

```

```
blocks
(
'''
B_dict['edge'] = '''
edges
(
'''
B_dict['patch'] = '''
boundary
(
'''
B_dict['patchinlet'] = '''
inlet
{
type patch;
faces
(
(21 22 52 51)
(22 23 53 52)
(23 24 54 53)
(24 25 55 54)
);
}
'''
B_dict['patchoutlet'] = '''
outlet
{
type patch;
faces
(
(17 18 48 47)
(16 17 47 46)
(29 16 46 59)
(28 29 59 58)
);
}
'''
B_dict['patchblade'] = '''
bladeTop
{
type wall;
faces
(
(0 30 31 1)
(1 31 32 2)
(2 32 33 3)

```

```
(3 33 34 4)
);
}
bladeBot
{
type wall;
faces
(
(4 34 35 5)
(5 35 30 0)
);
}
'''
B_dict['patchSym'] = '''
Top
{
type patch;
faces
(
(18 19 49 48)
(19 20 50 49)
(20 21 51 50)
);
}

Bottom
{
type patch;
faces
(
(25 26 56 55)
(26 27 57 56)
(27 28 58 57)
);
}
'''

B_dict['patchfrontAndBack'] = '''
back
{
type empty;
faces
(
(24 23 11 12)
(12 11 3 4)
(23 22 10 11)

```

```
(11 10 2 3)
(22 21 9 10)
(10 9 1 2)
(21 20 8 9)
(9 8 0 1)
(20 19 7 8)
(8 7 6 0)
(19 18 17 7)
(7 17 16 6) //all above midline
(12 13 25 24)
(4 5 13 12)
(13 14 26 25)
(5 0 14 13)
(14 15 27 26)
(0 6 15 14)
(15 29 28 27)
(6 16 29 15)
);
}

front
{
type empty;
faces
(
(54 42 41 53)
(42 34 33 41)
(41 40 52 53)
(33 32 40 41)
(40 39 51 52)
(32 31 39 40)
(39 38 50 51)
(31 30 38 39)
(38 37 49 50)
(30 36 37 38)
(37 47 48 49)
(36 46 47 37) // all above midline
(54 55 43 42)
(42 43 35 34)
(55 56 44 43)
(43 44 30 35)
(56 57 45 44)
(44 45 36 30)
(57 58 59 45)
(45 59 46 36)
);
```

```

}
'''
B_dict['end'] = '''
mergePatchPairs
(
);'''
with open('blockMeshDict_bump', 'w') as w:
w.write(B_dict['header'])
cnt = 0
for i in TESTarray:
w.write('  {0} {1} 0) //{2}\n'.format(i[0],i[1],cnt))
cnt = cnt + 1
for j in TESTarray:
w.write('  {0} {1} 0.1) //{2}\n'.format(j[0],j[1],cnt))
cnt = cnt + 1
w.write(');')
w.write(B_dict['block'])
w.write('  hex (12 11 23 24 42 41 53 54) ({0} {1} 1) edgeGrading ({2}
    {3} {3} {2} 1 1 1 1 1 1 1)
    //{0}\n'.format(LEU1,100,zero_1,zero_2))
w.write('  hex (4 3 11 12 34 33 41 42) ({0} {1} 1) edgeGrading ({2}
    {4} {4} {2} {3} {3} {3} {3} 1 1 1 1)
    //{0}\n'.format(LEU1,yCells,one_1,one_5,zero_1))
w.write('  hex (11 10 22 23 41 40 52 53) ({0} {1} 1) edgeGrading (1 1
    1 1 1 1 1 1 1 1) //{2}\n'.format(guess1,100,))
w.write('  hex (3 2 10 11 33 32 40 41) ({0} {1} 1) edgeGrading (1 1 1
    1 {2} {3} {3} {2} 1 1 1 1)
    //{3}\n'.format(guess1,yCells,one_5,three_6))
w.write('  hex (10 9 21 22 40 39 51 52) ({0} {1} 1) edgeGrading ({2}
    {3} {3} {2} 1 1 1 1 1 1 1)
    //{4}\n'.format(LEU2,100,five_2,four_2))
w.write('  hex (2 1 9 10 32 31 39 40) ({0} {1} 1) edgeGrading ({2}
    {3} {3} {2} {5} {4} {4} {5} 1 1 1 1)
    //{5}\n'.format(LEU2,yCells,five_1,five_2,five_6,three_6))
w.write('  hex (9 8 20 21 39 38 50 51) ({0} {1} 1) edgeGrading (1 1 1
    1 1 1 1 1 1 1) //{6}\n'.format(TEguess,100,1))
w.write('  hex (1 0 8 9 31 30 38 39) ({0} {1} 1) edgeGrading (1 1 1 1
    {2} {3} {3} {2} 1 1 1 1)
    //{7}\n'.format(TEguess,yCells,five_6,seven_6))
w.write('  hex (8 7 19 20 38 37 49 50) ({0} {1} 1) edgeGrading ({2}
    {2} {2} {2} 1 1 1 1 1 1 1) //{8}\n'.format(wake1,100,nine_1))
w.write('  hex (0 6 7 8 30 36 37 38) ({0} {1} 1) edgeGrading ({3} {3}
    {3} {3} {2} {4} {4} {2} 1 1 1 1)
    //{9}\n'.format(wake1,yCells,seven_6,nine_1,nine_6))
w.write('  hex (7 17 18 19 37 47 48 49) ({0} {1} 1) edgeGrading ({2}
    {2} {2} {2} 1 1 1 1 1 1 1) //{10}\n'.format(wake2,100,eleven_1))

```

```

w.write(' hex (6 16 17 7 36 46 47 37) ({0} {1} 1) edgeGrading ({2}
    {2} {2} {2} {3} {3} {3} {3} 1 1 1 1)
    //11\n'.format(wake2,yCells,eleven_1,nine_6)) #above mid line
w.write(' hex (24 25 13 12 54 55 43 42) ({0} {1} 1) edgeGrading ({4}
    {3} {3} {4} 1 1 1 1 1 1 1)
    //12\n'.format(LECells,100,thirteen_2,thirteen_1,twelve_1))
w.write(' hex (12 13 5 4 42 43 35 34) ({0} {1} 1) edgeGrading ({5}
    {2} {2} {5} {3} {4} {4} {3} 1 1 1 1) //13\n'.\
format(LECells,yCells,thirteen_2,1.0/one_5,thirteen_6,thirteen_1))
w.write(' hex (25 26 14 13 55 56 44 43) ({0} {1} 1) edgeGrading (1 1
    1 1 1 1 1 1 1 1) //14\n'.format(TEcellsL,100,1))
w.write(' hex (13 14 0 5 43 44 30 35) ({0} {1} 1) edgeGrading (1 {2}
    {2} 1 {3} {4} {4} {3} 1 1 1 1)
    //15\n'.format(TEcellsL,yCells,fifteen_2,thirteen_6,1.0/seven_6))
w.write(' hex (26 27 15 14 56 57 45 44) ({0} {1} 1) edgeGrading ({2}
    {2} {2} {2} 1 1 1 1 1 1 1) //16\n'.format(wake1,100,nine_1))
w.write(' hex (14 15 6 0 44 45 36 30) ({0} {1} 1) edgeGrading ({2}
    {2} {2} {2} {3} {4} {4} {3} 1 1 1 1)
    //17\n'.format(wake1,yCells,nine_1,1.0/seven_6,1.0/nine_6))
w.write(' hex (27 28 29 15 57 58 59 45) ({0} {1} 1) edgeGrading ({2}
    {2} {2} {2} 1 1 1 1 1 1 1) //18\n'.format(wake2,100,eleven_1))
w.write(' hex (15 29 16 6 45 59 46 36) ({0} {1} 1) edgeGrading ({2}
    {2} {2} {2} {3} {3} {3} {3} 1 1 1 1)
    //19\n'.format(wake2,yCells,eleven_1,1.0/nine_6))
w.write(');')

w.write(B_dict['edge'])
w.write(' arc 9 10 ({0} {1} 0)\n'.\
format(R/sf*sp.cos(thetaUP),R/sf*sp.sin(thetaUP)))
w.write(' arc 10 11 ({0} {1} 0)\n'.\
format(R/sf*sp.cos(sp.pi-thetaBump),R/sf*sp.sin(sp.pi-thetaBump)))
w.write(' arc 11 12 ({0} {1} 0)\n'.\
format(R/sf*sp.cos(sp.radians(170)),R/sf*sp.sin(sp.radians(170))))
w.write(' arc 21 22 ({0} {1}
    0)\n'.format(R*sp.cos(thetaUP),R*sp.sin(thetaUP)))
w.write(' arc 22 23 ({0} {1}
    0)\n'.format(R*sp.cos(sp.pi-thetaBump),R*sp.sin(sp.pi-thetaBump)))
w.write(' arc 23 24 ({0} {1}
    0)\n'.format(R*sp.cos(sp.radians(170)),R*sp.sin(sp.radians(170))))
w.write(' arc 12 13 ({0} {1}
    0)\n'.format(R/sf*sp.cos(5*sp.pi/4),R/sf*sp.sin(5*sp.pi/4)))
w.write(' arc 24 25 ({0} {1}
    0)\n'.format(R*sp.cos(5*sp.pi/4),R*sp.sin(5*sp.pi/4)))

w.write(' arc 39 40 ({0} {1}
    0.1)\n'.format(R/sf*sp.cos(thetaUP),R/sf*sp.sin(thetaUP)))

```



```

w.write(' arc 40 41 ({0} {1} 0.1)\n'.\
format(R/sf*sp.cos(sp.pi-thetaBump),R/sf*sp.sin(sp.pi-thetaBump)))
w.write(' arc 41 42 ({0} {1} 0.1)\n'.\
format(R/sf*sp.cos(sp.radians(170)),R/sf*sp.sin(sp.radians(170))))
w.write(' arc 51 52 ({0} {1}
    0.1)\n'.format(R*sp.cos(thetaUP),R*sp.sin(thetaUP)))
w.write(' arc 52 53 ({0} {1} 0.1)\n'.\
format(R*sp.cos(sp.pi-thetaBump),R*sp.sin(sp.pi-thetaBump)))
w.write(' arc 53 54 ({0} {1} 0.1)\n'.\
format(R*sp.cos(sp.radians(170)),R*sp.sin(sp.radians(170))))
w.write(' arc 42 43 ({0} {1}
    0.1)\n'.format(R/sf*sp.cos(5*sp.pi/4),R/sf*sp.sin(5*sp.pi/4)))
w.write(' arc 54 55 ({0} {1}
    0.1)\n'.format(R*sp.cos(5*sp.pi/4),R*sp.sin(5*sp.pi/4)))

w.write(' polyLine 0 1 (\n')
for i in xrange(1,len(C1)):
w.write('    ({0} {1} 0)\n'.format(C1[i][0],C1[i][1]))
w.write('    )\n')
w.write(' polyLine 1 2 (\n')
for j in xrange(1,11):
w.write('    ({0} {1} 0)\n'.format(C2[j][0],C2[j][1]))
w.write('    )\n')
w.write(' polyLine 2 3 (\n')
for k in xrange(11,12):
w.write('    ({0} {1} 0)\n'.format(C2[k][0],C2[k][1]))
w.write('    )\n')
w.write(' polyLine 3 4 (\n')
for l in xrange(12,len(C2)):
w.write('    ({0} {1} 0)\n'.format(C2[l][0],C2[l][1]))
w.write('    )\n')
w.write(' polyLine 4 5 (\n')
for m in xrange(1,len(C3)):
w.write('    ({0} {1} 0)\n'.format(C3[m][0],C3[m][1]))
w.write('    )\n')
w.write(' polyLine 5 0 (\n')
for n in xrange(1,len(C4)):
w.write('    ({0} {1} 0)\n'.format(C4[n][0],C4[n][1]))
w.write('    )\n')

w.write(' polyLine 30 31 (\n')
for i in xrange(1,len(C1)):
w.write('    ({0} {1} 0.1)\n'.format(C1[i][0],C1[i][1]))
w.write('    )\n')
w.write(' polyLine 31 32 (\n')
for j in xrange(1,11):

```

```

w.write('      ({0} {1} 0.1)\n'.format(C2[j][0],C2[j][1]))
w.write('      )\n')
w.write(' polyLine 32 33 (\n')
for k in xrange(11,12):
w.write('      ({0} {1} 0.1)\n'.format(C2[k][0],C2[k][1]))
w.write('      )\n')
w.write(' polyLine 33 34 (\n')
for l in xrange(12,len(C2)):
w.write('      ({0} {1} 0.1)\n'.format(C2[l][0],C2[l][1]))
w.write('      )\n')
w.write(' polyLine 34 35 (\n')
for m in xrange(1,len(C3)):
w.write('      ({0} {1} 0.1)\n'.format(C3[m][0],C3[m][1]))
w.write('      )\n')
w.write(' polyLine 35 30 (\n')
for n in xrange(1,len(C4)):
w.write('      ({0} {1} 0.1)\n'.format(C4[n][0],C4[n][1]))
w.write('      )\n')
w.write(');\n')
w.write(B_dict['patch'])
w.write(B_dict['patchinlet'])
w.write(B_dict['patchoutlet'])
w.write(B_dict['patchblade'])
w.write(B_dict['patchSym'])
w.write(B_dict['patchfrontAndBack'])
w.write(');\n')
w.write(B_dict['end'])
w.close()

def resNACA(res,m,p,t):
'''I have taken the above formula and made changes as done on the
turbulence modelling resource page.
the foil is run from 0 to 1.00893 chord length. Then scaled back down
to 1. This means we have a
perferct copy of the 4 digit foil but with a sharp trailed edge. In
math terms, i just scaled the
coefeceints by 1.008930411365'''
y_c = []
space = sp.linspace(0,1,res)
for x in reversed(space):
y_c.append([x, (t/(0.2*1.008930411365))*(0.298222773*sp.sqrt(x) -
0.127125232*x - 0.357907906*x**2 + 0.291984971*x**3 -
0.105174606*x**4)])
for x in space:
y_c.append([x, -(t/(0.2*1.008930411365))*(0.298222773*sp.sqrt(x) -
0.127125232*x - 0.357907906*x**2 + 0.291984971*x**3 -

```

```

    0.105174606*x**4)])

return sp.array(y_c)

def offset(coordinates, distance):
    '''project surface perpendicularly a set distance away'''
    coordinates = iter(coordinates)
    x1, y1 = coordinates.next()
    z = distance
    points = []
    for x2, y2 in coordinates:
        # tangential slope approximation
        try:
            slope = (y2 - y1) / (x2 - x1)
            # perpendicular slope
            pslope = -1/slope # (might be 1/slope depending on direction of
                travel)
        except ZeroDivisionError:
            continue
        mid_x = (x1 + x2) / 2
        mid_y = (y1 + y2) / 2

        sign = ((pslope > 0) == (x1 > x2)) * 2 - 1
        delta_x = sign * z / ((1 + pslope**2)**0.5)
        delta_y = pslope * delta_x

        points.append((mid_x + delta_x, mid_y + delta_y))
        x1, y1 = x2, y2
    return points

def make_circle(C,r):
    X = sp.linspace(C[0]-r,C[0] +r,1000)
    curve = []
    for x1 in X:
        curve.append([x1,sp.sqrt(r**2 - (x1 -C[0])**2) + C[1]])
    for x2 in X:
        curve.append([x2,-sp.sqrt(r**2 - (x2 -C[0])**2) + C[1]])
    return sp.array(curve)

def make_circle_t2(C,R,A1,A2):
    '''make an arc between A1 and A2'''
    theta = sp.linspace(A1,A2,360)

```

```

y = []
for i in xrange(len(theta)):
y.append([C[0]+R*sp.cos(theta[i]),C[1]+R*sp.sin(theta[i])])
return sp.array(y)

def find_r(A,B,C):
'''determine the radius of a circle based on two points and a
centre'''
radius = sp.sqrt(((A[0]-C[0])**2 + (A[1]-C[1])**2 + (B[0]-C[0])**2 +
(B[1]-C[1])**2)/2)
return radius

def f1(x,A,R):
y = sp.sqrt(R**2 - (x -A[0])**2) + A[1] -
(0.12/(0.2*1.008930411365))*(0.298222773*sp.sqrt(x) -
0.127125232*x - 0.357907906*x**2 + 0.291984971*x**3 -
0.105174606*x**4)
return y
def f2(x,A,R):
y = -sp.sqrt(R**2 - (x -A[0])**2) + A[1] -
(0.12/(0.2*1.008930411365))*(0.298222773*sp.sqrt(x) -
0.127125232*x - 0.357907906*x**2 + 0.291984971*x**3 -
0.105174606*x**4)
return y
def intersection(A,R):
'''find and intersection point'''
ans1 = bisect(f1,A[0]+R,A[0]-R,args=(A,R))
ans1y = (0.12/(0.2*1.008930411365))*(0.298222773*sp.sqrt(ans1) -
0.127125232*ans1 - 0.357907906*ans1**2 + 0.291984971*ans1**3 -
0.105174606*ans1**4)
ans2 = bisect(f2,A[0]-R,A[0]+R,args=(A,R))
ans2y = (0.12/(0.2*1.008930411365))*(0.298222773*sp.sqrt(ans2) -
0.127125232*ans2 - 0.357907906*ans2**2 + 0.291984971*ans2**3 -
0.105174606*ans2**4)
return [ans1,ans2]

def circl2ptsCt(Ct,A,B,R,number):
'''make a circle using 2 pts and their center'''
gamma = sp.arctan((Ct[1] - A[1])/(A[0] - Ct[0]))
theta1 = sp.pi*3/2 + gamma
thetap2 = sp.arctan((Ct[0] - B[0])/(Ct[1] - B[1]))
if thetap2 < 0:
theta2 = sp.pi + thetap2
else:
theta2 = thetap2
theta = sp.linspace(theta2,theta1,number)

```

```

y = []
for i in xrange(len(theta)):
y.append([Ct[0]-R*sp.sin(theta[i]),Ct[1]-R*sp.cos(theta[i])])
return sp.array(y)

def Le_y(x):
'''leadading edge coordinates'''
pt = (0.12/(0.2*1.008930411365))*(0.298222773*sp.sqrt(x) -
    0.127125232*x - 0.357907906*x**2 + 0.291984971*x**3 -
    0.105174606*x**4)
return pt

def projection_pt(C,r,grd,inter):
'''project a point from the foil to the bump'''
X = sp.linspace(C[0]-r,C[0] +r,100000)
V = []
for x1 in X:
V.append([x1,-sp.sqrt(r**2 - (x1 -C[0])**2) + C[1] - grd*x1 - inter])
Va = sp.array(V)
xx = sp.where(sp.absolute(Va[:,1]).min() ==
    sp.absolute(Va[:,1]))[0][0]
point = [X[xx],grd*X[xx] + inter]
return point

def tangetpts(C1,C2,r1,r2):
'''Obating tangets points for line connecting 2 circles. C2 is bigger
    circle, this assumes C1 always to the
    bottom left of C2. t1 for small circle T2 for big circle'''
C1C2d = sp.sqrt((C1[0] - C2[0])**2 + (C1[1] - C2[1])**2)
C1C2y = sp.absolute(C2[1] - C1[1])
C1C2x = sp.absolute(C2[0] - C1[0])
alpha = sp.arcsin(C1C2y/C1C2d)
beta = sp.arcsin(C1C2x/C1C2d)
gamma = sp.arcsin((r1+r2)/C1C2d)
if (C1[0] < C2[0]) and (C1[1] < C2[1]):
T2 = [C2[0] - r2*sp.cos(beta-gamma),C2[1] + r2*sp.sin(beta-gamma)]
T1 = [C1[0] + r1*sp.cos(beta-gamma),C1[1] - r1*sp.sin(beta-gamma)]
elif (C1[0] < C2[0]) and (C1[1] > C2[1]):
T2 = [C2[0] - r2*sp.sin(gamma-alpha),C2[1] + r2*sp.cos(gamma-alpha)]
T1 = [C1[0] + r1*sp.sin(gamma-alpha),C1[1] - r1*sp.cos(gamma-alpha)]
elif (C1[0] > C2[0]) and (C1[1] > C2[1]):
T2 = [C2[0] + r2*sp.sin(gamma-alpha),C2[1] + r2*sp.cos(gamma-alpha)]
T1 = [C1[0] - r1*sp.sin(gamma-alpha),C1[1] - r1*sp.cos(gamma-alpha)]
else:
T2 = [C2[0] + r2*sp.cos(beta-gamma),C2[1] + r2*sp.sin(beta-gamma)]
T1 = [C1[0] - r1*sp.cos(beta-gamma),C1[1] - r1*sp.sin(beta-gamma)]

```

```

return T1, T2

def findC(splineF,C,A,B,r,bump,leadingEdge,LE=True):
'''Create fillet radii. Spline is for offset radius, C is the centre,
    Aand B is there statr and end of this new bump
and finally r is bigger radius of offset BUMP'''
distanceFromCircle = 1e-3
if LE:
X1 = sp.linspace(A[0]-1e-3,C[0],20000)
offsetFOIL = []
offsetBUMP = circl2ptsCt(C,A,B,r+distanceFromCircle,20000) #i added
    distance to r to set it furtehr away from the main bump
for i in X1:
offsetFOIL.append([i,interpolate.splev(i,splineF)])
offsetFOILA = sp.array(offsetFOIL)
QQ = distance.cdist(offsetFOILA,offsetBUMP,'euclidean')
centre = [offsetBUMP[sp.where(QQ.min() ==
    QQ)[1][0]][0],offsetBUMP[sp.where(QQ.min() == QQ)[1][0]][1]]
print "LE circle C: ", centre
tan1,tan2 = tangetpts(centre,C,1e-3,r-1e-3) #changing r back to
    correct size
tan2ARRAY = sp.array([tan2,tan2])
LEc = make_circle(centre,1e-3)
int1 = distance.cdist(tan2ARRAY,bump,'euclidean')
int2 = distance.cdist(LEc,leadingEdge,'euclidean')
int2P = sp.where(int2.min() == int2)[0][0]
int1P = sp.where(int1.min() == int1)[1][0]
edgeint = LEc[int2P]
angle1 = sp.real(sp.arctan((centre[1] - tan1[1])/(tan1[0] -
    centre[0])))
angle2 = sp.real(sp.arctan((edgeint[0] - centre[0])/(centre[1] -
    edgeint[1])))
theta1 = 3*sp.pi/2.0 + angle2
theta2 = 2*sp.pi - angle1
arc = make_circle_t2(centre,1e-3,theta1,theta2-sp.radians(1))
return arc, sp.where(int1.min() == int1)[1][0]
else:
X1 = sp.linspace(C[0],B[0]+1e-3,20000)
offsetFOIL = []
offsetBUMP = circl2ptsCt(C,A,B,r+distanceFromCircle,20000) #i added
    distance to r to set it furtehr away from the main bump
for i in X1:
offsetFOIL.append([i,interpolate.splev(i,splineF)])
offsetFOILA = sp.array(offsetFOIL)
QQ = distance.cdist(offsetFOILA,offsetBUMP,'euclidean')

```

```

centre = [offsetBUMP[sp.where(QQ.min() ==
    QQ)[1][0]][0],offsetBUMP[sp.where(QQ.min() == QQ)[1][0]][1]]
print "TE circle C: ", centre
tan1,tan2 = tangetpts(centre,C,1e-3,r-1e-3) #changing r back to
    correct size
tan2ARRAY = sp.array([tan2,tan2])
LEc = make_circle(centre,1e-3)
int1 = distance.cdist(tan2ARRAY,bump,'euclidean')
int2 = distance.cdist(LEc,leadingEdge,'euclidean')
int1P = sp.where(int1.min() == int1)[0][0]
int2P = sp.where(int2.min() == int2)[0][0]
edgeint = LEc[int2P]
angle1 = sp.real(sp.arctan((centre[1] - tan1[1])/(centre[0] -
    tan1[0])))
angle2 = sp.real(sp.arctan((edgeint[0] - centre[0])/(centre[1] -
    edgeint[1])))
theta1 = sp.pi + angle1
theta2 = 3*sp.pi/2.0 + angle2
arc = make_circle_t2(centre,1e-3,theta1+sp.radians(1),theta2)
return arc, sp.where(int1.min() == int1)[1][0]

def make_bump(a,W,d,curve,data):
    '''make a bump base don width, and height d'''
    width = W
    depth = d*width/2.0 #% of radius to move away from original centre
    A = a # centre
    loc = sp.where(sp.absolute(curve[:,0] - A).min() ==
        sp.absolute(curve[:,0] - A))[0][0] #location on LE curve
    ct = curve[loc] #centre of curve for full radius use
    two = intersection(ct,width/2.0) #the two xpositions for intersection
    p1 = [two[0],Le_y(two[0])] #XY of intersection point
    p2 = [two[1],Le_y(two[1])] #XY of intersection point
    if p1[0] < p2[0]:
        ang = sp.arctan((p2[1] - p1[1])/(p2[0] - p1[0]))
    else:
        ang = sp.arctan((p1[1] - p2[1])/(p1[0] - p2[0]))
    gamma = 360 - 90 - 90 - sp.degrees(ang) #angle with x axis
    Ct_prime = [ct[0] - depth*sp.sin(sp.radians(gamma)),ct[1] -
        depth*sp.cos(sp.radians(gamma))] #new center based on d

    gradient = (Ct_prime[1] - ct[1])/(Ct_prime[0] - ct[0]) #m of straight
        line between centers
    intercept = Ct_prime[1] - gradient*Ct_prime[0] #c of straight line
        between centers
    new_r = find_r(p1,p2,Ct_prime) #this radius is found from two points
        and the centre

```

```

L2 = circl2ptsCt(Ct_prime,p1,p2,new_r,100000) #make arc given centre
    and two points
'''stage 2 is find circle center on either side'''
offsetBUMP = sp.array(offset(L2,-1e-3))
space2 = sp.linspace(L2[0][0]-0.002,L2[-1][0]+0.002,50) #setup x
    points for interp function
y_c2 = []
for x in space2: #cycle through points Build LE
y_c2.append([x, Le_y(x)])
newLE2 = sp.array(y_c2)
LE_off2 = sp.array(offset(newLE2,-1e-3)) #offset LE by 1e-3
DF = interpolate.splrep(LE_off2[:,0],LE_off2[:,1]) #create spline to
    fit the offset LE
smoothLE, bumpINTLE =
    findC(DF,Ct_prime,offsetBUMP[0],offsetBUMP[-1],new_r\
+1e-3,L2,curve,LE=True) #offset spline, bumpCT, start&end, offset
    radius, org bump and org LE
smoothTE, bumpINTTE =
    findC(DF,Ct_prime,offsetBUMP[0],offsetBUMP[-1],new_r\
+1e-3,L2,curve,LE=False) #offset spline, bumpCT, start&end, offset
    radius, org bump and org LE
L3 = sp.vstack([smoothLE,L2[bumpINTLE:bumpINTTE],smoothTE]) #make new
    smooth bump
proj = projection_pt(Ct_prime,new_r,gradient,intercept)

'''to get project to top of circle'''
mvProj = sp.array(proj) - sp.array(Ct_prime)
gamma = sp.arctan(mvProj[1]/mvProj[0])
rot = gamma + sp.pi
newProj = sp.array([new_r*sp.cos(rot),new_r*sp.sin(rot)])
mvBkProj = newProj + sp.array(Ct_prime)
plt.plot(mvBkProj[0],mvBkProj[1], 'cD')
plt.plot(proj[0],proj[1], 'c*')
plt.show()
'''end of proj2'''
print "D1: ", sp.linalg.norm(sp.array(ct) - sp.array(Ct_prime))
print "bubble H: ",sp.linalg.norm(mvBkProj - sp.array(ct))

q1 = L3[0] - CB #L2[0] - CB
q2 = L3[-1] - CB #L2[-1] - CB
intersection1 = sp.where((sp.sqrt(q1[:,0]**2 + q1[:,1]**2).min() ==
    sp.sqrt(q1[:,0]**2 + q1[:,1]**2)))[0][0]
intersection2 = sp.where((sp.sqrt(q2[:,0]**2 + q2[:,1]**2).min() ==
    sp.sqrt(q2[:,0]**2 + q2[:,1]**2)))[0][0]
newCB = sp.copy(curve)
part1 = newCB[:intersection2-1]

```



```

part2 = newCB[intersection1+1:]
L2p =L3[::-1] #reversed order of my curve
LE = sp.vstack([part1,L2p,part2])
return LE,p1,p2

if __name__ == "__main__":
single = resNACA(200000,0,0,0.12)
upper,lower = seperateUPDOWN(single)

DA,DB = seperateBot(lower) #curve with no overlapping points, to make
    stl LE then TE
CA,CB = seperateTop(upper)

location = 0.05
width = 0.005
depth = -0.001

output,a1,a2 = make_bump(location,width,depth,CB,df1)
LEL = []
for i in output[:,0]:
LEL.append([i,-Le_y(i)])
lel = sp.array(LEL)
DA1 = sp.copy(DA)
DB1 = sp.copy(DB)
CA1 = sp.copy(CA)

CB1 = sp.copy(output)
AoA = 0.0
motion = CA1[-1][0]
DA1[:,0] = DA1[:,0]-motion
DB1[:,0] = DB1[:,0]-motion
CA1[:,0] = CA1[:,0]-motion
CB1[:,0] = CB1[:,0]-motion

lel[:,0] = lel[:,0]-motion
lower_out = lel[:,0]
print 'value of depth: ', depth

rCA = CA[:,0]
rCB = CB[:,0]
rDA = DA1[:,0]
rDB = DB1[:,0]
rlower = lower_out[:,0]
'''---Thinsg to use CA1,CB1,lower_out,rDA1-----'''
CA1u = CA1
CB1u = CB1[1:]

```

```

DB1u = lower_out[1:-1]
DA1u = rDA[0:-1]
whole = sp.vstack([CA1u, CB1u, DB1u, DA1u])
'''get pts bump loc + distance L & R'''
plt.plot(CA1u[:,0], CA1u[:,1], 'r')
plt.plot(CB1u[:,0], CB1u[:,1], 'r')
plt.plot(DA1u[:,0], DA1u[:,1], 'r')
plt.plot(DB1u[:,0], DB1u[:,1], 'r')

loc = sp.where(sp.absolute(CB[:,0] - location).min() ==
               sp.absolute(CB[:,0] - location))[0][0] #location on LE curve
ct = CB[loc] #centre of curve for full radius use
two = intersection(ct,width) #the two xpositions for intersection
p1 = [two[0]-motion, Le_y(two[0])] #XY of intersection point
p2 = [two[1]-motion, Le_y(two[1])] #XY of intersection point
if p1[0] > p2[0]:
    TEloc = p1
    LEloc = p2
else:
    TEloc = p2
    LEloc = p1
locTE = sp.where(sp.absolute(CB1u[:,0] - TEloc[0]).min() ==
                 sp.absolute(CB1u[:,0] - TEloc[0]))[0][0]
locLE = sp.where(sp.absolute(CB1u[:,0] - LEloc[0]).min() ==
                 sp.absolute(CB1u[:,0] - LEloc[0]))[0][0]

hope.meshDict_Ch(CA1u, CB1u, DB1u, DA1u, 40, locTE, locLE, width)

```

List of References

- Abu-Mostafa, Y. (2018). Machine learning video library. online.
Available at: <https://work.caltech.edu/library/index.htm>
- Alam, F., Chowdhury, H., Moria, H., Brooy, R., Subic, A. *et al.* (2010). A comparative study of golf ball aerodynamics. In: *Proceedings of the 17th Australasian Fluid Mechanics Conference (AFMC)*, pp. 5–9.
- Back, S.C., Hobson, G.V., Song, S.J. and Millsaps, K.T. (2012). Effects of reynolds number and surface roughness magnitude and location on compressor cascade performance. *Journal of Turbomachinery*, vol. 134, no. 5, p. 051013.
- Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.
- Barron, R. and Salehi Neyshabouri, A.A. (2003). Effects of under-relaxation factors on turbulent flow simulations. *International journal for numerical methods in fluids*, vol. 42, no. 8, pp. 923–928.
- Berg, D.E., Wilson, D.G., Resor, B.R., Barone, M.F., Berg, J.C., Kota, S. and Ervin, G. (2009). Active aerodynamic blade load control impacts on utility-scale wind turbines. In: *Proc. of WindPower Conf. & Exhib.*
- Beyene, A. and Peffley, J. (2007 June). A morphing blade for wave and wind energy conversion. In: *Oceans 2007*, pp. 1–6.
- Bilgili, M., Yasar, A. and Simsek, E. (2011). Offshore wind power development in europe and its comparison with onshore counterpart. *Renewable and Sustainable Energy Reviews*, vol. 15, no. 2, pp. 905 – 915. ISSN 1364-0321.
- Bons, J.P., Taylor, R.P., McClain, S.T. and Rivir, R.B. (2001). The many faces of turbine surface roughness. In: *ASME Turbo Expo 2001: Power for Land, Sea, and Air*, pp. V003T01A042–V003T01A042. American Society of Mechanical Engineers.
- Bravo, R., Tullis, S. and Ziada, S. (2007). Performance testing of a small vertical-axis wind turbine. In: *Proceedings of the 21st Canadian Congress of Applied Mechanics (CANCAM07), Toronto, Canada, June*, pp. 3–7.
- Breiman, L. (2001). Random forests. *Machine learning*, vol. 45, no. 1, pp. 5–32.
- Brothers, C. (1998). Vertical axis wind turbines for cold climate applications. *Renewable Energy Technologies in Cold Climates, Montreal*.

- Chakroun, W., Al-Mesri, I. and Al-Fahad, S. (2004). Effect of surface roughness on the aerodynamic characteristics of a symmetrical airfoil. *Wind Engineering*, vol. 28, no. 5, pp. 547–564.
- Cowan, I.R., Castro, I.P. and Robins, A.G. (1997). Numerical considerations for simulations of flow and dispersion around buildings. *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 67, pp. 535–545.
- Cox, S.E., Haftka, R.T., Baker, C.A., Grossman, B., Mason, W.H. and Watson, L.T. (2001). A comparison of global optimization methods for the design of a high-speed civil transport. *Journal of Global Optimization*, vol. 21, no. 4, pp. 415–432.
- Erfort, G., von Backstrom, T.W. and Venter, G. (2017). Numerical optimisation of a small-scale wind turbine through the use of surrogate modelling. *Journal of Energy in Southern Africa*, vol. 28, no. 3, pp. 79–91.
- Erfort, G., von Backström, T.W. and Venter, G. (2018 Julya). Fine tuning of the $\gamma - re_{\theta}$ turbulence model using historical data sets. In: Wan, P.D. (ed.), *The proceedings of the 13th OpenFOAM workshop (OFW 13)*, pp. 433–435. Available at: <http://openfoamworkshop.org/proceedings/>
- Erfort, G., von Backström, T.W. and Venter, G. (2018 Septemberb). Reduction in the torque ripple of a vertical axis wind turbine through coupled optimization of the pitch angle. In: Tengen, P.T.B. (ed.), *The proceedings of the 11th South African Conference on Computational and Applied mechanics*, pp. 680–692.
- Erfort, G., von Backström, T.W. and Venter, G. (2019a). Numerically determined empirical relationships for a transitional turbulence model. *Journal of Applied Fluid mechanics*.
- Erfort, G., von Backström, T.W. and Venter, G. (2019b). Reduction in the torque ripple of a vertical axis wind turbine through foil pitching optimization. *Journal of Wind Engineering*, vol. pre-print.
- Eriksson, S., Bernhoff, H. and Leijon, M. (2008). Evaluation of different turbine concepts for wind power. *Renewable and Sustainable Energy Reviews*, vol. 12, no. 5, pp. 1419 – 1434. ISSN 1364-0321.
- Fish, F.E. and Battle, J.M. (1995). Hydrodynamic design of the humpback whale flipper. *Journal of Morphology*, vol. 225, no. 1, pp. 51–60.
- Fish, F.E. and Hui, C.A. (1991). Dolphin swimming—a review. *Mammal Review*, vol. 21, no. 4, pp. 181–195.
- Fish, F.E. and Rohr, J. (1999). Review of dolphin hydrodynamics and swimming performance. Tech. Rep., Space and naval warfare systems command San Diego CA.

- Gebreslassie, M.G., Tabor, G. and Belmont, M.R. (2012). Cfd simulations for sensitivity analysis of different parameters to the wake characteristics of tidal turbine. *Open Journal of Fluid dynamics*.
- Global-Wind-Energy-Council (2017 April). Global wind report: Annual market update 2017.
- Gray, J. (1936). Studies in animal locomotion: Vi. the propulsive powers of the dolphin. *Journal of experimental biology*, vol. 13, no. 2, pp. 192–199.
- Greenshields, C.J. (2015). Openfoam user guide.
- Griffin, D. and McCoy, T. (2008). Coe reductions through active aerodynamic control of rotor aerodynamics and geometry. Tech. Rep., National Renewable Energy Laboratory (NREL), Golden, CO.
- Hellsten, A. (1998). Some improvements in menter's k-omega sst turbulence model. In: *29th AIAA, Fluid Dynamics Conference*, p. 2554.
- Howell, R., Qin, N., Edwards, J. and Durrani, N. (2010). Wind tunnel and numerical study of a small vertical axis wind turbine. *Renewable energy*, vol. 35, no. 2, pp. 412–422.
- Islam, M., Ting, D.S.-K. and Fartaj, A. (2008). Aerodynamic models for darrieus-type straight-bladed vertical axis wind turbines. *Renewable and Sustainable Energy Reviews*, vol. 12, no. 4, pp. 1087–1109.
- Jones, R. and Williams, D. (1936). Effect of surface roughness on characteristics of aerofoils naca 0012 and raf 34. Tech. Rep., AERONAUTICAL RESEARCH COUNCIL LONDON (UNITED KINGDOM).
- Kirke, B. (2011). Tests on ducted and bare helical and straight blade darrieus hydrokinetic turbines. *Renewable Energy*, vol. 36, no. 11, pp. 3013–3022.
- Krawczyk, P., Beyene, A. and Macphee, D. (2013 January). Fluid structure interaction of a morphed wind turbine blade. *International Journal of Energy Research*, vol. 37, pp. 1784–1793.
- Lachenal, X., Daynes, S. and Weaver, P.M. (2013). Review of morphing concepts and materials for wind turbine blade applications. *Wind Energy*, vol. 16, pp. 283–307.
- Langtry, R., Gola, J. and Menter, F. (2006). Predicting 2d airfoil and 3d wind turbine rotor performance using a transition model for general cfd codes. In: *44th AIAA Aerodspace Sciences Meeting and Exhibit*.
- Langtry, R.B. and Menter, F.R. (2009 December). Correlation-based transition model for unstructured parallelized computational fluid dynamics codes. *AIAA journal*, vol. 47, no. 12, pp. 2894–2906.
- Mayda, E., Van Dam, C. and Yen-Nakafuji, D. (2005). Computational investigation of finite width microtabs for aerodynamic load control. *AIAA Paper*, vol. 1185, p. 2005.

- Menter, F. and Esch, T. (2001). Elements of industrial heat transfer predictions. In: *16th Brazilian Congress of Mechanical Engineering (COBEM)*, vol. 109, p. 650. sn.
- Menter, F.R. (1994). Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA journal*, vol. 32, no. 8, pp. 1598–1605.
- Meyers, R.H. and Montgomery, D.C. (2002). *Response Surface Methodology Process and Product Optimization Usind Designed Experiments*. JOHN WILEY and SONS.
- Miau, J., Liang, S., Yu, R., Hu, C., Leu, T., Cheng, J. and Chen, S. (2012 11). Design and test of a vertical-axis wind turbine with pitch control. In: *AEROTECH IV*, vol. 225 of *Applied Mechanics and Materials*, pp. 338–343. Trans Tech Publications.
- Morgado, J., Vizinho, R., Silvestre, M. and Páscoa, J. (2016). Xfoil vs cfd performance predictions for high lift low reynolds number airfoils. *Aerospace Science and Technology*, vol. 52, pp. 207–214.
- Morgan, J.N. and Sonquist, J.A. (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American statistical association*, vol. 58, no. 302, pp. 415–434.
- Musgrove, P. (1987). Wind energy conversion: Recent progress and future prospects. *Solar & Wind Technology*, vol. 4, no. 1, pp. 37 – 49. ISSN 0741-983X.
- Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.
- Pagnini, L.C., Burlando, M. and Repetto, M.P. (2015). Experimental power curve of small-size wind turbines in turbulent urban environment. *Applied Energy*, vol. 154, pp. 112–121.
- Paquette, J.A. and Barone, M.F. (2012). Innovative offshore vertical-axis wind turbine rotor project. Tech. Rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).
- Paraschivoiu, I. (1981). Double-multiple streamtube model for darrieus in turbines. In: *Wind turbine dynamics*, vol. 1, pp. 19–25.
- Paraschivoiu, I. and DELCLAUX, F. (1983). Double multiple streamtube model with recent improvements (for predicting aerodynamic loads and performance of darrieus vertical axis wind turbines). *Journal of Energy*, vol. 7, no. 3, pp. 250–255.
- Paraschivoiu, I., Trifu, O. and Saeed, F. (2009 05). H-darrieus wind turbine with blade pitch control. *International Journal of rotating Machinery*, vol. 2009.

- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830.
- Reuter, R.C. and Worstell, M.H. (1978). Torque ripple in a vertical axis wind turbine. Tech. Rep., SANDIA.
- Rostamzadeh, N., Kelso, R.M. and Dally, B. (2017). A numerical investigation into the effects of reynolds number on the flow mechanism induced by a tubercled leading edge. *Theoretical and Computational Fluid Dynamics*, vol. 31, no. 1, pp. 1–32.
- Salomon, R. (1998 July). Evolutionary algorithms and gradient search: similarities and differences. *Evolutionary Computation, IEEE Transactions on*, vol. 2, no. 2, pp. 45–55.
- Schlichting, H. and Kestin, J. (1968). *Boundary-layer theory*. sixth edn. McGraw-Hill.
- Schlierkamp-Voosen, D. and Mühlenbein, H. (1993). Predictive models for the breeder genetic algorithm. *Evolutionary Computation*, vol. 1, no. 1, pp. 25–49.
- Sheldahl, R.E. and Klimas, P.C. (1981). Aerodynamic characteristics of seven symmetrical airfoil sections through 180-degree angle of attack for use in aerodynamic analysis of vertical axis wind turbines. Tech. Rep., Sandia National Labs., Albuquerque, NM (USA).
- Shires, A. and Kourkoulis, V. (2013). Application of circulation controled blades for vertical axis wind turbines. *Energies*, vol. 6, pp. 3745–3763.
- Siddiqui, M.S., Rasheed, A., Kvamsdal, T. and Tabib, M. (2015). Effect of turbulence intensity on the performance of an offshore vertical axis wind turbine. *Energy Procedia*, vol. 80, pp. 312–320.
- Smola, A.J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, vol. 14, no. 3, pp. 199–222.
- Spalart, P.R. and Rumsey, C.L. (2007). Effective inflow conditions for turbulence models in aerodynamic calculations. *AIAA journal*, vol. 45, no. 10, pp. 2544–2553.
- Spera, D.A. (1994). *Wind turbine technology*. American Society of Mechanical Engineers.
- Sunderland, K., Woolmington, T., Blackledge, J. and Conlon, M. (2013). Small wind turbines in turbulent (urban) environments: A consideration of normal and weibull distributions for power prediction. *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 121, pp. 70–81.

- Sutherland, H.J., Berg, D.E. and Ashwill, T.D. (2012). A retrospective of vawt technology. Tech. Rep., Sandia National Laboratories.
- Thill, C., Etches, J., Bond, I., Potter, K. and Weaver, P. (2008). Morphing skins. *The aeronautical journal*, vol. 112, no. 1129, pp. 117–139.
- Tomassini, M. (1999). Parallel and distributed evolutionary algorithms: A review.
- Unit, I.P.P.P.P. (2017 June). Independent power producers procurement programme (ipp) an overview. Tech. Rep., IPPPP Unit.
Available at: <https://www.ipp-projects.co.za/Publications>
- Van Dam, C., Chow, R., Zayas, J. and Berg, D. (2007). Computational investigations of small deploying tabs and flaps for aerodynamic load control. In: *Journal of Physics: Conference Series*, vol. 75, p. 012027. IOP Publishing.
- Weller, H.G., Tabor, G., Jasak, H. and Fureby, C. (1998). A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in physics*, vol. 12, no. 6, pp. 620–631.
- Wilcox, D.C. *et al.* (1998). *Turbulence modeling for CFD*, vol. 2. DCW industries La Canada, CA.
- Wilson, D.G., Berg, D.E., Barone, M.F., Berg, J.C., Resor, B.R. and Lobitz, D.W. (2009). Active aerodynamic blade control design for load reduction on large wind turbines. In: *European Wind and Energy Conference & Exhibition 2009*, pp. 26–19.