The Pennsylvania State University

The Graduate School

Department of Aerospace Engineering

PERFORMANCE STUDY OF A SMALL SCALE WIND TURBINE

A Thesis in

Aerospace Engineering

by

Anagha Ray

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2014

The thesis of Anagha Ray was reviewed and approved* by the following:

Dennis McLaughlin
Professor of Aerospace Engineering
Thesis Co-Advisor

Susan Stewart
Research Associate Aerospace Engineering and Architectural Engineering
Thesis Co-Advisor

George A. Lesieutre
Professor of Aerospace Engineering
Head of the Department of Aerospace Engineering

*Signatures are on file in the Graduate School

ABSTRACT

Wind Energy at The Pennsylvania State University is an exciting field of research. The Skystream 3.7 manufactured by Southwest Windpower is a 2.4kW system used for this research. The system is located at the Sustainability Experience Center that serves as a research facility for students. The purpose of this study is to do a comprehensive study of power performance and wind turbine aerodynamics using the Skystream 3.7. Data recorded from this 2.4 kW rated small scale wind turbine consisted of generator output power and RPM, acquired simultaneously with wind speed and wind direction in the immediate vicinity of the turbine. The resolution of the collected data is of 3 second intervals and this data is processed and averaged for analysis. The experimental data is compared with computed predictions using the code WT_Perf developed by the National Renewable Energy Laboratory (NREL). The study was performed over a wide range of both wind speed and turbine RPM comprising a comprehensive aerodynamic study of the performance of the wind turbine. The performance in power generation from experimental data compares favorably with manufacturer's published data.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

a	axial flow induction factor
a'	tangential flow induction factor
A	area at actuator disc
A_d, A_R	rotor swept area
A_∞, A_W	Upstream and downstream stream-tube cross-sectional areas
B	Number of blades
c	blade chord
C_d	drag coefficient
C_l	lift coefficient
C_p	pressure coefficient
C_P	power coefficient
C_Q	torque coefficient
C_T	thrust coefficient
C_x	coefficient of tangential forces
C_y	coefficient of axial forces
D	drag force; rotor diameter
f	tip loss factor
F	force
F_b	Prandtl's tip loss correction factor
H	hub height
L	lift force
$\dot{m}$	mass flow rate
N	number of blades
p	static pressure

p'	pressure drop across rotor plane
P	aerodynamic power
Q	rotor torque
r	radius of blade element or point on blade
R	blade tip radius
Re	Reynolds number
RPM	Rotations per minute
T	rotor thrust; temperature
t	time (sec)
U_∞ ,U_1	free stream velocity
U_2	velocity before rotor disc
U_3	velocity after rotor disc
U_R ,U_d	velocity at the rotor disc
U_w	velocity in the far wake
U_T	Tangential velocity component
V	wind velocity relative to a point on a rotating blade
α	angle of attack
β	inclination of local blade chord to rotor plane (i.e., blade twist plus pitch angle)
ϕ	flow angle of resultant velocity V to rotor plane
λ	tip speed ratio
ρ	air density
σ	blade solidity
ω	angular frequency (rad/s)
Ω	rotational speed of rotor

ACKNOWLEDGEMENTS

I would like to thank my thesis advisors, Dr. Dennis McLaughlin and Dr. Susan Stewart for giving me the opportunity to work with them, for guiding me with their experience on the research. I would especially like to thank them for their constant support and motivation throughout the Masters program.

I would also like to thank Brian Wallace for building a platform for this research and for giving me his guidance. I would like to thank all students at The Pennsylvania State University who helped me in various stages of the research.

Chapter 1

Introduction

The purpose of this study is to take experimental data from a distributed wind turbine deployed on the Penn State campus and compare it with computational predictions for the turbine using industry standard analyses and procedures which are not commonly used for small/distributed wind applications. The recorded experimental data are analyzed and compared to that of predicted data developed using software that uses blade element momentum theory for performance prediction. This study is done to compare the data from the wind turbine, with that of predicted data and manufacturer's data, that will help us to get a better understanding of the wind turbine's performance. The wind turbine used for this study is the Skystream 3.7 which is a three bladed, 2.4 kW model.

Wind Energy Background

The term “small scale wind turbine” characterizes wind turbines that are scaled such that they produce electricity for households or small businesses. The International Electrotechnical Commission defines that a wind turbine with a rotor swept area smaller than 200 m^2, equating to a rated power of approximately 50 kW and generating at voltage below 1000 V AC and 1500 V DC would classify as small scale wind turbine. However the widely accepted range for small scale wind turbines is a rated power of less than 100 kW.[1]

Brief history of Small Scale Wind Energy

The power of the wind has been used to drive windmills for centuries to grind grains and to pump water. The earliest wind turbines used to generate electricity were in the early 1900's. By the 1930's wind turbines were used commonly in farms in the United States. The generated electricity made life easier for these farmers who started using it for radio's and other household conveniences.(2) The use of wind power however declined after 1950's due to electricity being supplied to rural houses through the power grid.

Present market for small scale wind energy

We have, however, lately understood the need to harness renewable sources of energy as our coal and gas reserves are declining. Renewable energy is not only a non-depleting source but it is also a clean form of energy. There are no harmful by-products or wastes produced for generating energy. The rising cost of energy and other environmental factors have caused the rise in interest for wind energy. Commercial wind farms have been most responsible for the growth of wind energy, however small wind energy is still on the rise because of convenience in installation and savings made in money because the excess energy produced by the wind turbine can be fed back to the electrical grid.

Today there is an 18% increase in the global small scale use of wind energy, compared to 2011.(1) Although most of the growth is in China, USA and the UK, better governmental policies and more exposure would help spread small wind energy use to other countries. This would especially be helpful in rural settings of developing countries where wind is a good natural resource and the power supply is erratic and expensive. Therefore, for countries that have smaller

consumption needs and have a significant wind resource, installing wind energy could be a cost effective solution.

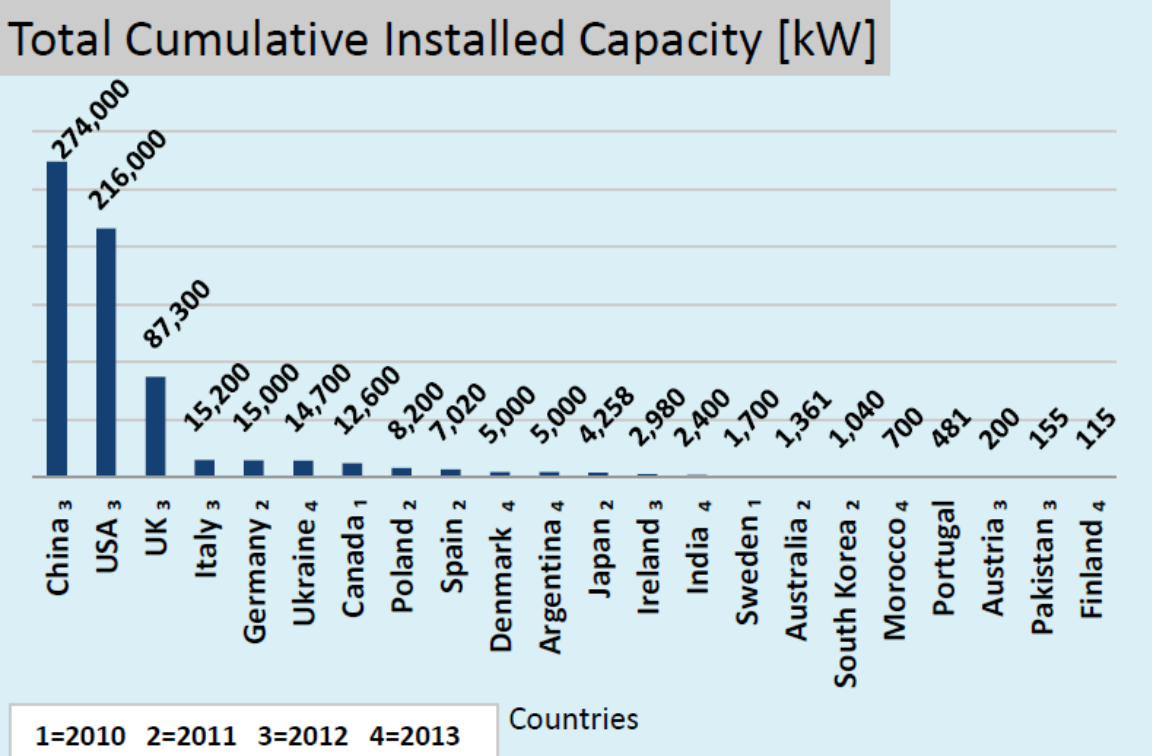


Figure 1-1. Distribution of installed capacity for small wind energy for each country(1)

The market today is trending towards wind farms with large scale wind turbines. However small scale wind turbines can have various independent applications in particular at remote locations with a good wind resource and some of them being; farms, water pumping, remote telecom stations, small businesses, recreational and fishing boats, residential use and research in remote locations.

Wind is a renewable source of energy. Therefore if a small wind turbine is installed, wind being a constantly replenished natural resource, it would reduce our dependence on coal and natural gas and could be a cost effective substitute.

Understanding the wind resource

Evaluating the wind resource of a location is always beneficial for understanding the performance of a proposed wind turbine installation. For large scale projects, this involves the installation of meteorological towers which collect data from a series of anemometers at several heights for a period of several years. However important this may be for large scale wind projects, it is not cost effective for small scale wind turbines. This is because the tools and the manpower required for this would be more than the cost of the wind turbine itself.

The use of wind maps could be an alternative. Wind maps show the wind resource of an area at heights ranging from 30 m to 80 m. Many small scale wind turbines have a height below 30 m and therefore although an alternative resource, it may not prove very effective. The Skystream wind turbine at Penn State reaches a height of 21 m. Below is a wind resource map of the United States at 30 m height.

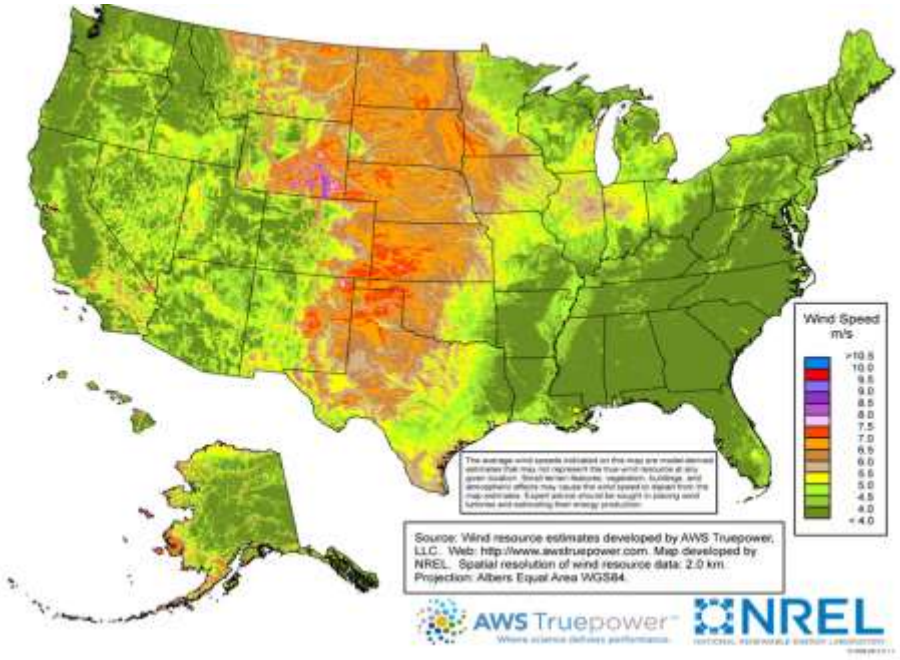


Figure 1-2. Wind resource map at 30 m height

Research Background

With an increasing demand for clean and affordable energy it is estimated that there would be an estimated growth in use of wind energy of 20% from 2015 to 2020 [1] .With increasing prices of fossil fuels and global warming becoming a major concern, it is imperative that researchers draw attention to the prospects of renewable, cost effective sources of energy; small scale wind energy being one of them.

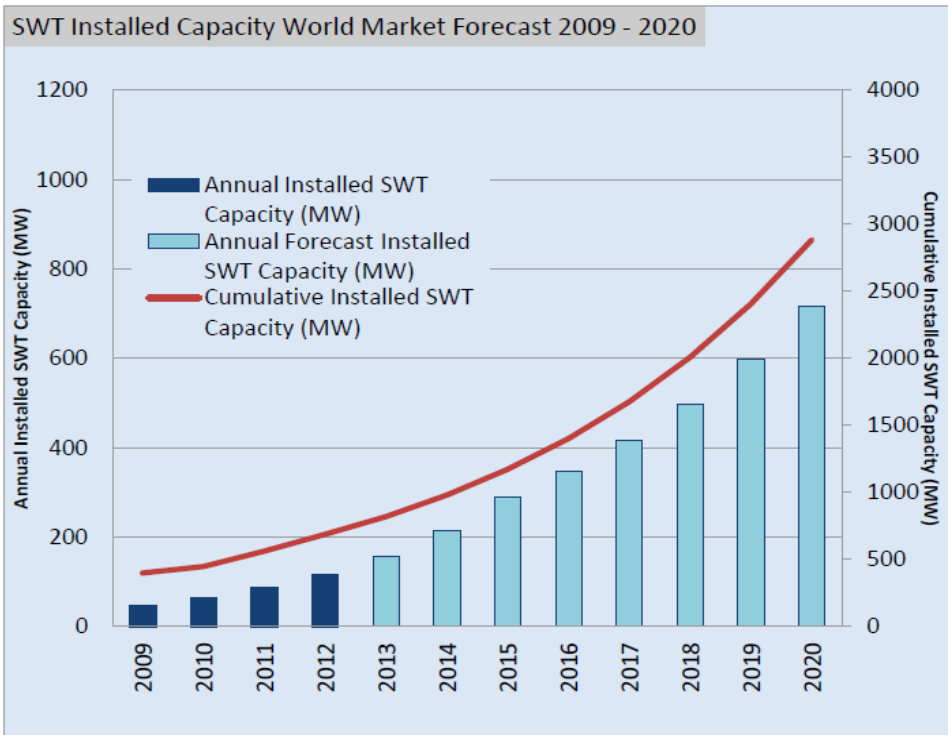


Figure 1-3. Small scale wind turbine installed capacity forecast[1]

As we can see, wind energy after solar, is the fastest growing renewable energy source in the world today. This growth and demand for wind energy is much required in the world today due to our depleting fuel resources. Many universities in the United States understand the scientific need to promote wind energy and are conducting research and awareness programs to promote it. Some universities and research groups focus at small scale wind energy while others

on 'Big wind'. Appalachian State University with the help of North Carolina Wind Energy is attempting to create awareness among people through their vast range of installed small wind turbines at their 'Wind Research and Demonstration Site'.[7] Michigan State University on the other hand is trying to conduct research on different blade designs for small scale wind turbines[8] and Clarkson University is doing extensive research with turbine solidity and blade number optimization for small scale turbines.[9] The National Renewable Energy Laboratory (NREL) conducts research and testing for performance on several small wind turbines, turbine development and takes up field verification projects. [10] NREL continues to play an important role in the market growth for small wind turbines.

The Pennsylvania State University is one such research university that understands this global need and the Department of Aerospace Engineering is trying to increasingly evolve with understanding small wind energy technology. The Skystream 3.7 is a part of Penn State's Center for Sustainability wind research facility.

When research with small wind energy is published it will support and change government policies and will help to drive the market. Research also helps to provide an educated workforce that would further help in the advancement of small scale wind energy.

Research Scope

Penn State University attempts at building a platform for research and development through collaboration between the Sustainability Experience Center and the Aerospace Engineering department's wind research program. Various graduate students have conducted research on micro size wind turbines like the Sunforce 400 and Allpower 400 [11] and small scale turbines like Whisper 500 [12].

The research done on this project is performance testing on the Skystream 3.7 using industry standard techniques. This research helps us to understand power performance of the wind turbine with collected data from the Skystream at The Pennsylvania State University facility. Similar tests have been done on the Skystream 3.7 by WindTest Inc. where the sampling rate was 1 min and the averaging interval for data processing was 10 min [15]. The Skystream wind turbine at our location has a sampling rate of 3 s and our research has a wide data processing scope with averaging intervals of 30 s, 60 s and 10 min. Data is also processed without averaging and compared with averaged data, which has shown some differences, discussed later in the paper. Different parameters from the wind turbine are compared for one day, one month and for one year to get a closer as well as a broader look at how the wind turbine performs. Finally these experimental results are compared against predicted data using WT_Perf and manufacturer's data.

A detailed wind resource assessment was not conducted prior to installation at our site, as would be the case with most independent buyers. The findings from our research might help many independent buyers to understand and better predict the performance of a proposed small wind turbine.

This research also helps to better understand the aerodynamics of a small scale wind turbine through the computations done using WT_Perf. For this, the Skystream blade geometry and airfoil were measured. The National Renewable Energy Laboratory (NREL) has previously conducted computational models and wind tunnel tests on the Skystream, citing the use of an S822 airfoil for the Skystream blade [10]. Our results show that S823 airfoil was used on the blade along with S822 airfoil. The findings have been discussed further in this paper.

The results of the current study are discussed in detail in the sections below, showing strong correlation between the experimental performance of the wind turbine from Penn State, the computed data, and the manufacturer's published power curve.

Objective of research

The study done on the Skystream 3.7 small scale wind turbine will help us to understand the general performance of a wind turbine at a regular site, where a detailed wind resource study has not been conducted before installation. We have several objectives to study such data:

- To study the collected experimental data from the wind turbine. Plot and analyze the wind data as well as turbine performance data using industry standards for one day, one month and one year.
- Compute using WT_Perf software the performance of the wind turbine. The computing tools also help us understand the nature of the aerodynamic loads around the blade.
- Compare power performance data from computed data, experimental data and manufacturer's published power curve.

Chapter 2

Brief Theory of Horizontal Axis Wind Turbine Aerodynamics

A wind turbine is a device that produces energy by extracting kinetic energy from the wind. The wind drives the blades of the turbine that rotate and produce mechanical energy. The generator converts the mechanical energy to electrical energy. Since the wind drives the turbine blades, the turbine performance is understood by analyzing the aerodynamic forces generated by the wind on the blades.

A simple model called an actuator disc model is used to understand the energy extraction process, calculate the power output and the thrust on the rotor. In this model a circular disc is considered instead of a turbine, through which the airstream flows with a velocity U_∞, and there is a pressure drop p_u to p_d as shown in Figure 2-1. This model assumes that there is no drag and that the flow is incompressible and flowing at steady state. [5]

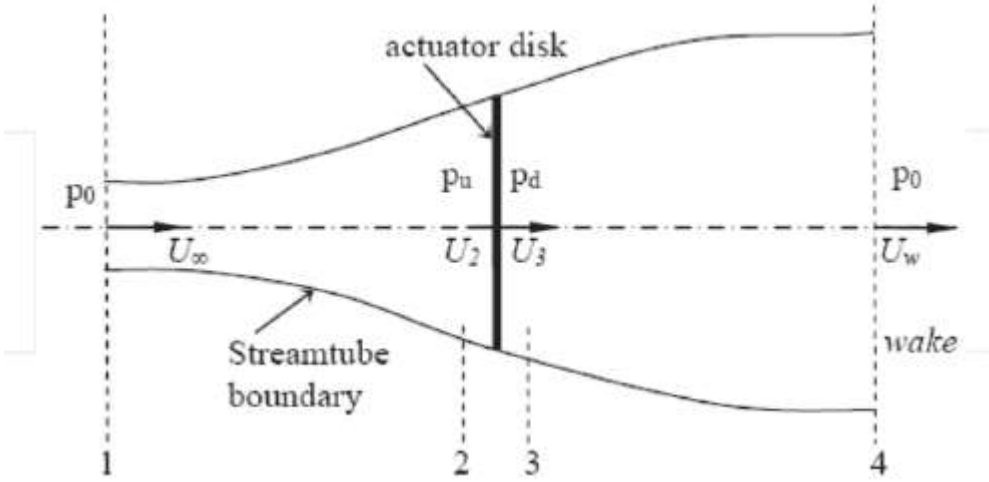


Figure 2-1. Actuator disc model [5]

There are four sections in the diagram above; 1 is the free stream region, 2 is the region before the blades, 3 is the region after the blade and 4 is the wake region. Therefore the mass flow equation through the stream tube is:

$$\rho\, A_\infty\, U_\infty = \rho\, A_d\, U_d = \rho\, A_w\, U_w \qquad (1)$$

Assuming continuous velocity of wind through the disc,

$$U_R = U_2 = U_3 \tag{2}$$

And the steady mass flow rate through the actuator disc is,

$$\dot{m} = \rho\, A\, U_R \tag{3}$$

where ρ is the density of the air, A is the area at the actuator disc

Thrust is the change in momentum across the blades,

$$T = \dot{m}\,(U_\infty - U_w) \tag{4}$$

Applying Bernoulli's equation between station 1 and 2 and between 3 and 4 we get,

$$p_d + \frac{1}{2}\rho U_R^{\,2} = p_0 + \frac{1}{2}\rho U_w^{\,2} \tag{5}$$

$$p_0 + \frac{1}{2}\rho U_\infty^{\,2} = p_u + \frac{1}{2}\rho U_R^{\,2} \tag{6}$$

From equations (5) and (6) , we get the pressure drop across rotor plane as,

$$(p_u - p_d) = p' = \frac{1}{2}\rho(U_\infty^{\,2} - U_w^{\,2}) \tag{7}$$

Since thrust on the actuator disc is the sum of the forces on each side,

$$T = Ap' = \frac{1}{2}\rho A(U_\infty^{\,2} - U_w^{\,2}) \tag{8}$$

Combining equations (3) ,(4) and (10) we get,

$$U_R = \frac{U_\infty + U_w}{2} \tag{9}$$

Axial induction flow is explained as,

$$a = \frac{U\infty - U_R}{U\infty} \tag{10}$$

Power is defined as a product of force and velocity, therefore,

$$P = \frac{1}{2}\rho(U_\infty^{\,2} - U_w^{\,2})\, U_R \tag{11}$$

Introducing axial induction factor in the equation we get,

$$P = \frac{1}{2}\,\rho\, A\, U_\infty^{\,3}\; a\,(1-a)^2 \tag{12}$$

Expressing the power and thrust in dimensionless forms gives us,

$$C_P = \frac{P}{\frac{1}{2}\rho\pi R^2 U_\infty^3} \quad (13)$$

$$C_T = \frac{T}{\frac{1}{2}\rho\pi R^2 U_\infty^2} \quad (14)$$

Defining the tip speed ratio we get,

$$\lambda = \frac{\Omega R}{U_\infty} \quad (15)$$

We can also write the power coefficient as,

$$C_P = 4a(1-a)^2 \quad (16)$$

The maximum value of C_P can be obtained by differentiating the equation and equating it to zero,

$$\frac{dC_p}{da} = 4a(1-a)(1-3a) = 0 \quad (17)$$

$$\text{Therefore } a = \frac{1}{3} \quad \text{and} \quad C_{Pmax} = \frac{16}{27} = 0.593 \quad (18)$$

The above result tells us that in case of an ideal rotor, if operated such that the velocity at the rotor is 2/3 of the free stream velocity, then it would be operating at C_{Pmax}

The axial thrust on the disc can be rewritten as,

$$T = 2A\rho a(1-a)U_\infty^2 \quad (19)$$

Substituting equation (19) into equation (14) we get the thrust coefficient for an ideal wind turbine,

$$C_T = 4a(1-a) \quad (20)$$

The figure below illustrates the variation of power and thrust coefficients with a.

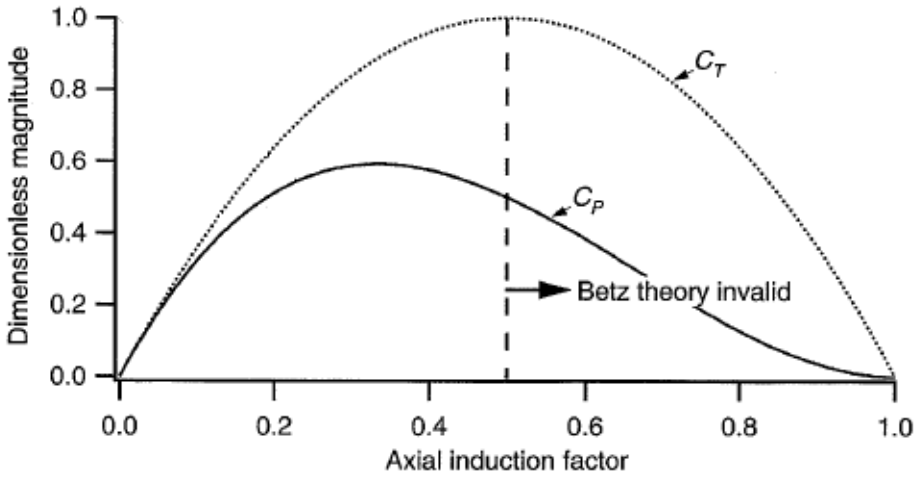


Figure 2-2. non-dimensional thrust and power coefficient variation with axial flow induction factor

We get the maximum value of C_T equal to 1 at a=0.5. The problem is that when a $\geq$ 0.5 then the wake velocity value, given by $(1-2a)U_\infty$ becomes zero or negative then the momentum theory stops applying.

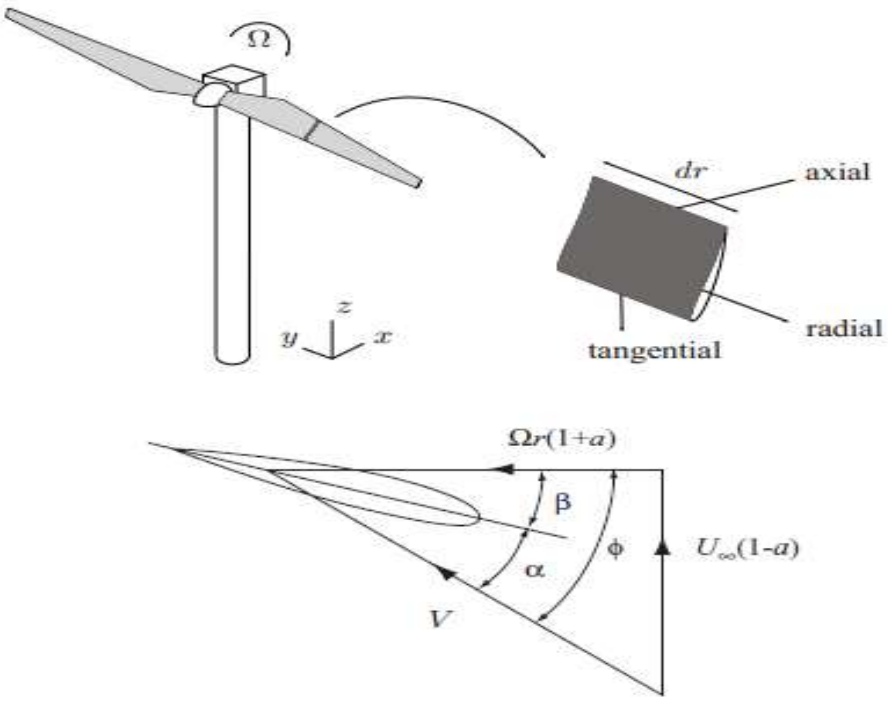


Figure 2-3. Nomenclatures for wind turbine aerodynamics

The wind turbine consists of a number of blades that converts wind energy into rotational energy. The torque on the turbine blade is caused by the wind passing over it. An opposing torque is then exerted on the wind by the blades that creates a wake. Therefore there is a presence of a tangential component of velocity that is due to this wake. [3] Figure 2-3 diagrammatically helps to understand the different velocity components.

We know that $U_R = U_\infty(1-a)$ from the definition of a. The blades rotate with an angular velocity Ω leading to a tangential component Ωr. We also know now that there is a tangential velocity due to the reaction torque acting on the flow which we define as U_T. [3] This helps us to define the tangential flow induction factor as

$$a' = \frac{U_T}{\Omega \mathrm{r}} \tag{21}$$

The net velocity V as seen on Figure 2-3 is,

$$V = \sqrt{(U_\infty(1-a))^2 + (\Omega r(1+a'))^2} \tag{22}$$

This velocity is at an angle φ to the free stream

$$tan\ \varphi = \frac{U_\infty(1-a)}{\Omega r(1+a')} \tag{23}$$

From Figure 2-3 we can see that the angle of attack $\alpha = \varphi - \beta$, where β is defined as the pitch angle of the blades.

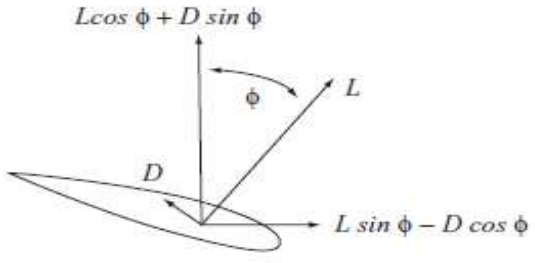


Figure 2-4. Forces on wind turbine airfoil

Figure 2-4 show us the forces on an airfoil on a wind turbine blade. The thrust and torque coefficients from the diagram as seen are,

$$C_T = C_L \cos\varphi + C_D \sin\varphi \quad (24)$$

$$C_Q = C_L \sin\varphi - C_D \cos\varphi \quad (25)$$

In Blade Element momentum theory the blade is divided into small elements and the performance is calculated by a summation of the forces on each element on the blade. Most wind turbine rotor design code are based on this method. Using this method we find the expressions of a and a' where

$$a(r) = \frac{1}{\frac{4\sin^2\varphi}{\sigma C_T}+1} \quad (26)$$

$$a'(r) = \frac{1}{\frac{4\sin\varphi\cos\varphi}{\sigma C_Q}-1} \quad (27)$$

Solidity σ is defined as the swept area occupied by the turbine blades,

$$\sigma = \frac{cB}{2\pi r} \quad (28)$$

where c is the chord, *B* id the number of blades and *r* is the rotor radius.

The tangential and axial flow induction factors are used in an iterative algorithm such that the power coefficient can be found. The algorithm is described below:

1. Estimated values for a and a' are used to start ;e.g. $a = a' = 0$.
2. $\varphi(r)$is calculated with given values for $U_\infty$and Ω (or λ), by using equation (23).

3. The local angle of attack, given by $\alpha(r) = \varphi - \beta$ is determined since the pitch angle β is given.
4. The values of C_L and C_D as a function of α are found from an external source (Xfoil is a good code to get these values).
5. The thrust and torque coefficients are computed from equations (24) and (25).
6. The values of a and a' are updated to equations (26) and (27) and if the values are not converged then the algorithm returns to step 2.

After the algorithm converges the values of a and a' are used to find the power coefficient from the equation

$$C_P = \frac{B\lambda}{\pi}\int_0^1 C_Q(r')[(1-a)^2 + (\lambda r')^2(1+a')^2]c'(r')\, r'\, dr' \quad (29)$$

where $r' = r/R$, $c' = c/R$

Up until now, the tip losses at the blade have not been accounted for. The tip losses are due to the interaction of the shed vortices with the blade flow near the tip. The pressure difference between the upper and lower surface of the blade causes the air to flow from the lower to the upper surface and then radially inward as seen in Figure 2-5. [(4)]

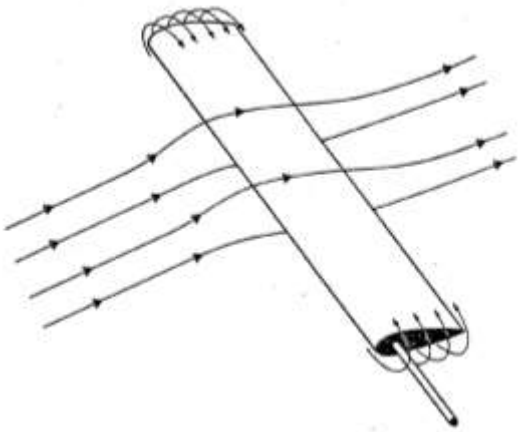


Figure 2-5. Tip loss flow diagram[(4)]

The tip losses have a great effect on the overall performance despite having a small chord at the tip, because of its distance from the hub.[4] To account for the loss at the tip, Prandtl derived a factor called the Prandtl's tip loss correction factor, that approximates this radial flow effect as can be seen in Figure 2-5.

Prandlt's tip loss correction factor is defined as follows:

$$F_P = \frac{2}{\pi}\cos^{-1}(e^{-f}) \tag{30}$$

$$where, \qquad f = \frac{B}{2}\frac{R-r}{rsin(\varphi)}$$

Incorporating Pradtl's tip loss correction factor the equations for *a* and *a'* are:

$$a = \frac{1}{\left(\frac{4F_P sin^2(\varphi)}{\sigma C_y}+1\right)} \tag{31}$$

$$a' = \frac{1}{\left(\frac{4F_P sin(\varphi)\cos(\varphi)}{\sigma C_x}-1\right)} \tag{32}$$

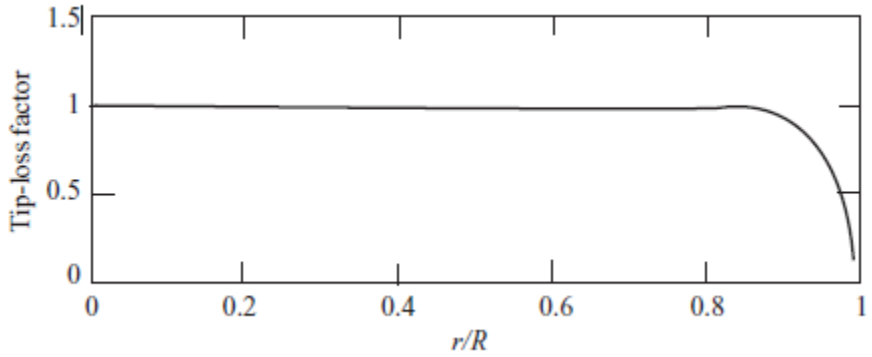


Figure 2-6. Spanwise variation of tip loss factor for a blade with uniform circulation

Equation (31) is only valid until a value of a less than 0.2, above which the momentum theory breaks down. Glauert's correction factor helps us to understand the effects after that as seen in Figure 2-7.[4]

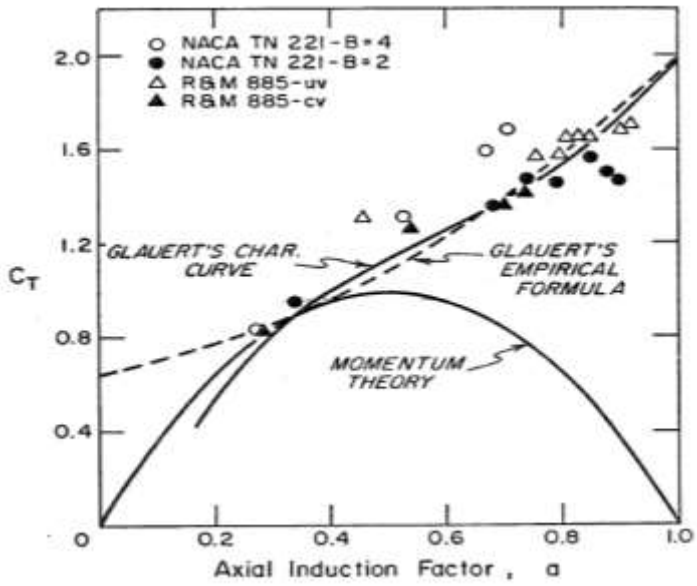


Figure 2-7. Performance at Windmill brake state[4]

The Glauert's empirical formula is,

$$a = \frac{1}{2}\Big(2 + K(1-2a_c) - \sqrt{(K(1-2a_c)+2)^2 + 4(Ka_c^2 - 1}\Big)) \quad (33)$$

where, $K = \frac{4Fsin^2(\varphi)}{\sigma C_y}$ and axial induction factor, $a_c > 0.2$

An example of power coefficient and torque coefficient with respect to λ, calculated by the Blade element momentum theory model is shown in Figure 2-8.

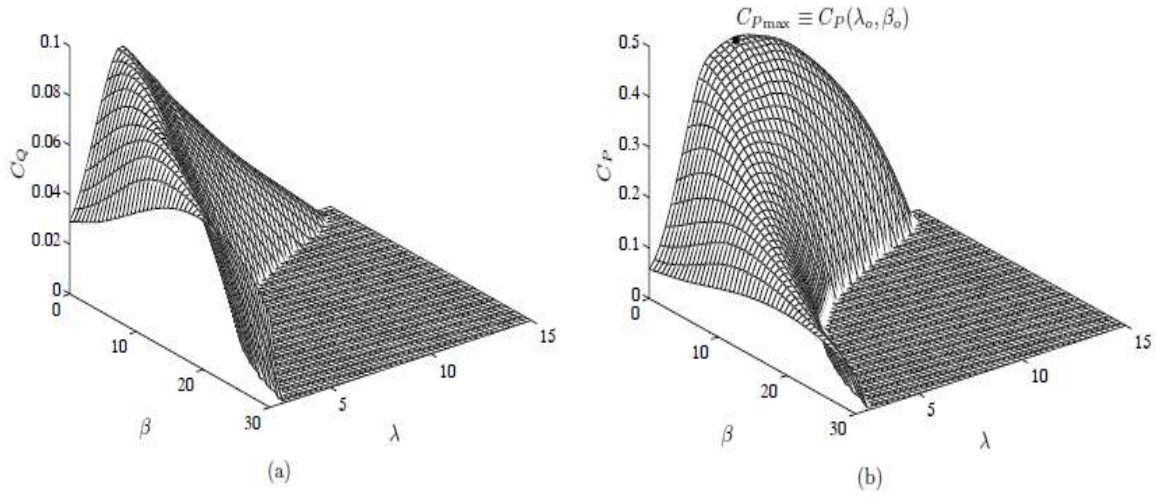


Figure 2-8. Typical variations of (a)Torque coefficient and (b) power coefficient with λ[6]

Chapter 3

Skystream Wind Turbine Systems

The skystream wind turbine is a product of Southwest Windpower. The blades for this wind turbine have a diameter of 3.7m and have a slight twist, taper and sweep to it. The unique design with swept blades and a controller that restricts high tip speed ratio's enables the wind turbine to generate power with reduced noise. The skystream has a neodymium magnet based slot-less alternator which is designed to produce energy at low speeds.

The Sksytream wind turbine at The Pennsylvania State University has a LabVIEW software system that has been developed by Brian Wallace. This provides the user with an interface to see how the turbine is generating power. This software also helps to save data from the wind turbine in files for each day. This software gets data from the Skystream's data acquisition system as shown in

Figure 3-1.

The upper part of

Figure 3-1 shows the turbine performance data being transmitted using a ZigBee system via a radio transmitter. It is a built in feature of the wind turbine that easily transmits data to a computer with a software called a Skyview which is also provided with the wind turbine. The meteorological (MET) data shown in the lower half of the diagram comes from the anemometers and wind veins installed on the wind turbine. This data is transferred to the computer using an ethernet cable through the NRG WiFi cDAQ , which is then processed by LabVIEW.

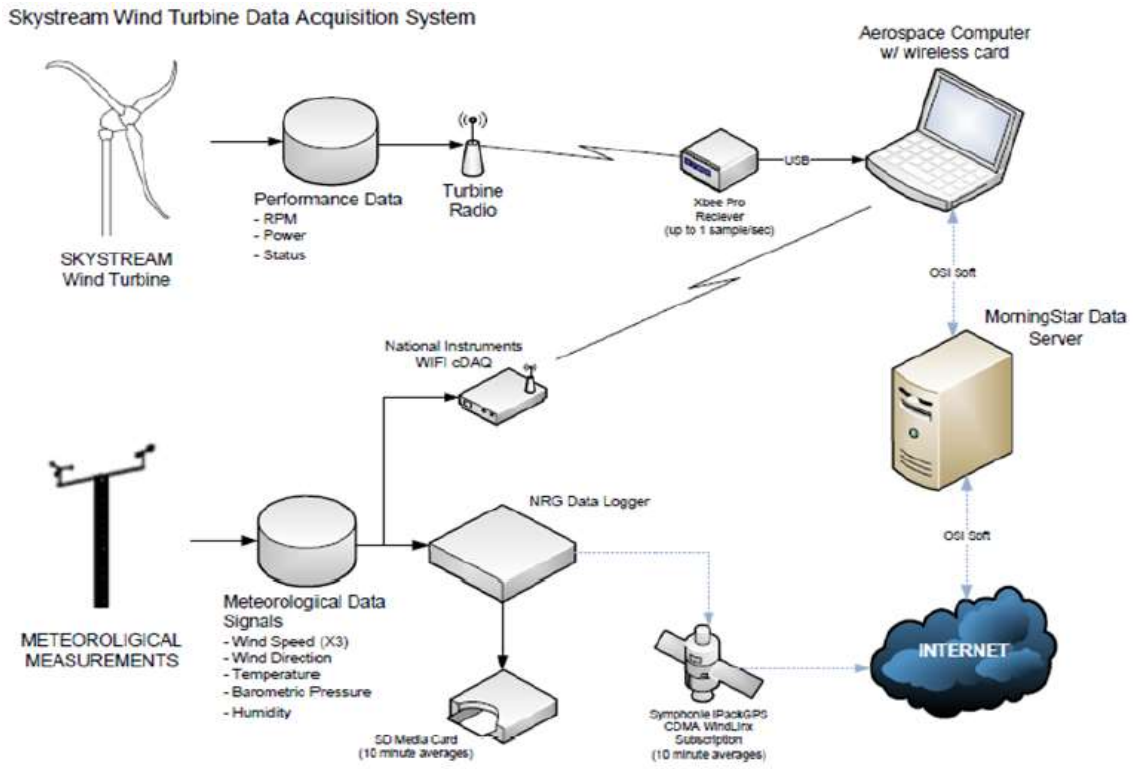


Figure 3-1. Skystream data acquisition system[14]

The meteorological data like the wind speed, wind direction and ambient temperature is measured using our instruments for higher data accuracy and research purpose. Three Cup Anemometers are placed at two different heights as shown in Figure 3-2. One directional vane is used to measure the wind direction and a temperature sensor gives the ambient temperature. All the sensors are plugged in respective channel ports on a data logger, from which the signals are transmitted through an ethernet cable. All of the sensors, and the data logger, were originally a part of a National Renewable Energy Laboratory (NREL) MET tower.[14] Figure 3-2 below shows the arrangement of the MET systems on the wind turbine.

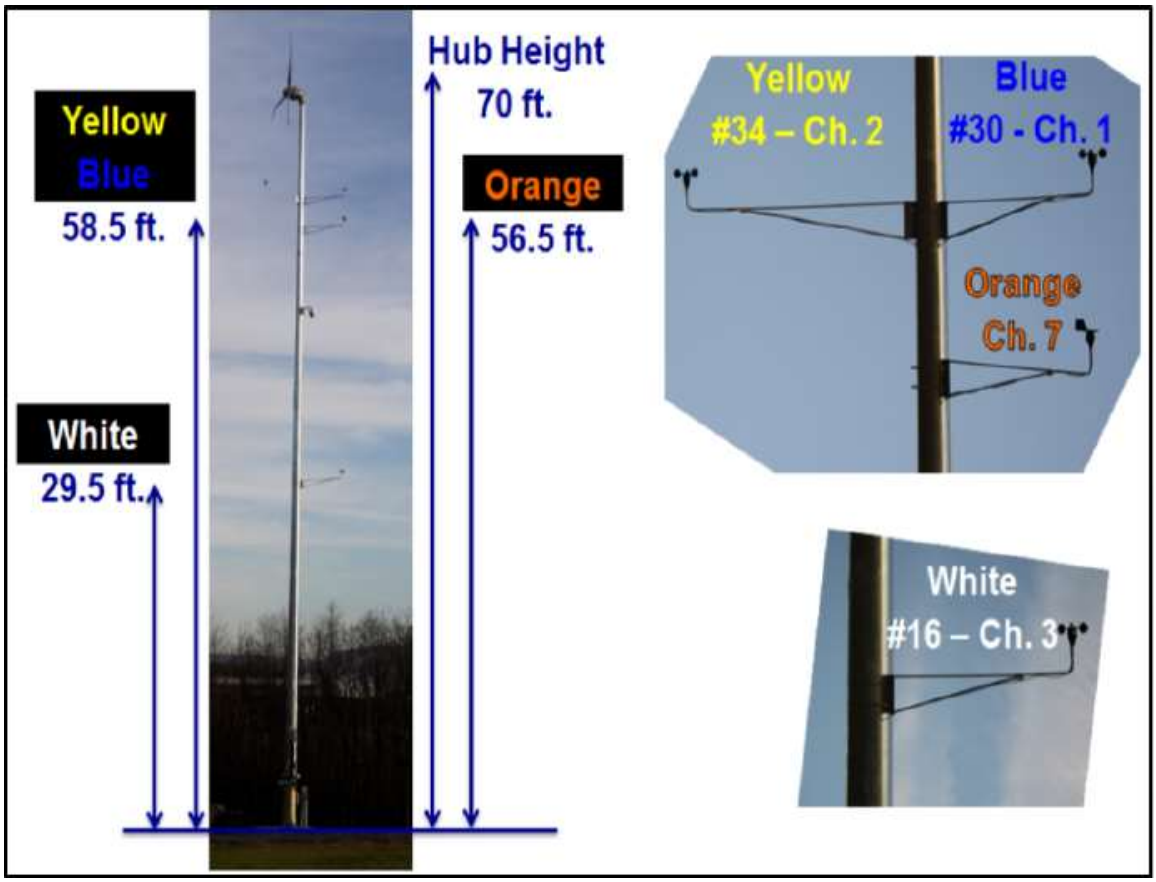


Figure 3-2. MET systems position on Skystream[14]

Chapter 4

Computational Performance Prediction for Skystream

The purpose of this study is to compare experimental data from Skystream 3.7 with that of predicted data. The data prediction not only helps us to understand the power performance but also helps us get a closer look at the aerodynamic loads acting on the blade at various configurations. The data is predicted using a software that uses blade element momentum theory for performance prediction. For computing the performance of the wind turbine, lift and drag characteristics of the airfoil used are needed along with the blade geometry specifications. This section explains all processes and procedures taken to predict the performance for Skystream 3.7 wind turbine.

Rotor and Airfoil Specifications

To get a better understanding of the aerodynamic properties of the Skystream, a blade was acquired to study its profile. The Skystream blade is stated to be 3.7 m in diameter. The blade on first look can be seen to have a sweep, taper and slight twist.

Figure 4-1. Picture of the Skystream blade

Process involved in scanning the blade

The measurement of the blade was performed by dividing it into 12 sections. The blade sections were aligned such that the divisions were perpendicular to the Z axis as shown in Figure 4-3. This alignment procedure was used for better aerodynamic prediction since the blade has a sweep.

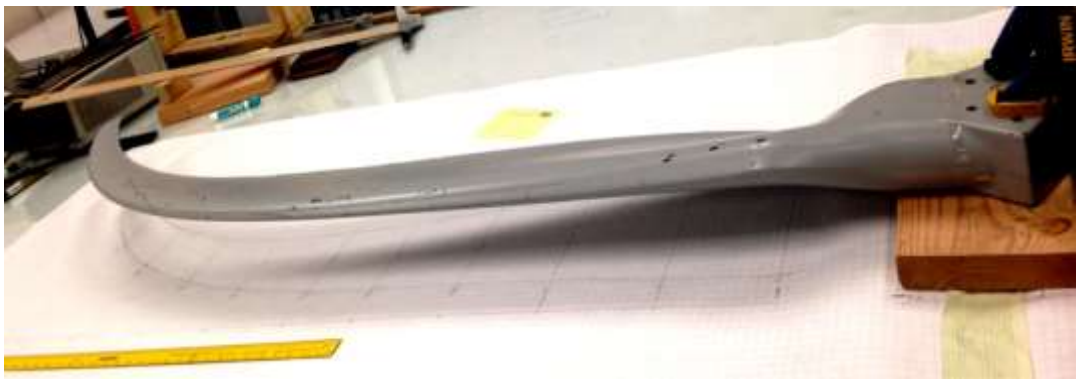


Figure 4-2. The blade being scanned on the graph sheet

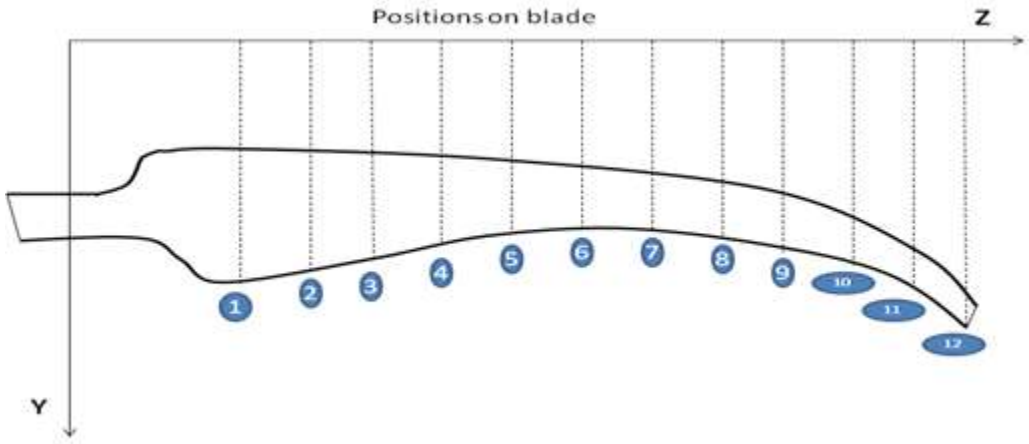


Figure 4-3. Diagram depicting how the blade was scanned

The Skystream blade's chord, pitch angle, taper and sweep were determined by manually measuring them on a large graph sheet. The blade was clamped onto a large table with a graph sheet underneath it. A simple but careful pencil trace gave us the 2-D image of the blade. It was then divided into 12 sections as depicted in Figure 4-3 above. The leading edge and trailing edges for each element section on the blade were marked on the blade. The height of the leading edge and trailing edge from the graph sheet on the table was measured with an error of +/- 2mm.

The chord length of the blade was found using the Pythagoras theorem. The base of the triangle was the difference in height of the leading and trailing edges (Δx). The perpendicular of the triangle was the distance between the LE and TE on the graph paper. The hypotenuse of this triangle gave us the chord for each of the 12 sections on the blade. The pictorial description can be seen in the diagram below.

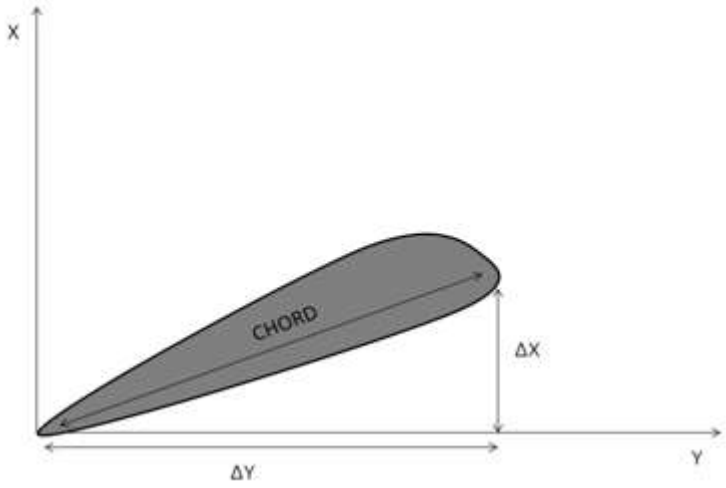


Figure 4-4. Figure explaining calculation of chord length

The pitch angle and chord for each section of the blade were determined from the triangles as shown in Figure 4-4 . Δx on the diagram is the difference in height between the leading edge and the trailing edge. Δy is the distance between the leading edge and the trailing edge for each section. Using Pythagoras theorem the blade chord was determined. The pitch angle was found by

finding the inverse tangent of Δx/Δy.

The taper was seen as the difference in the chord lengths. The sweep is the distance of the leading edge from the z-axis. The plot below shows the variation in chord, twist and sweep for the blade.

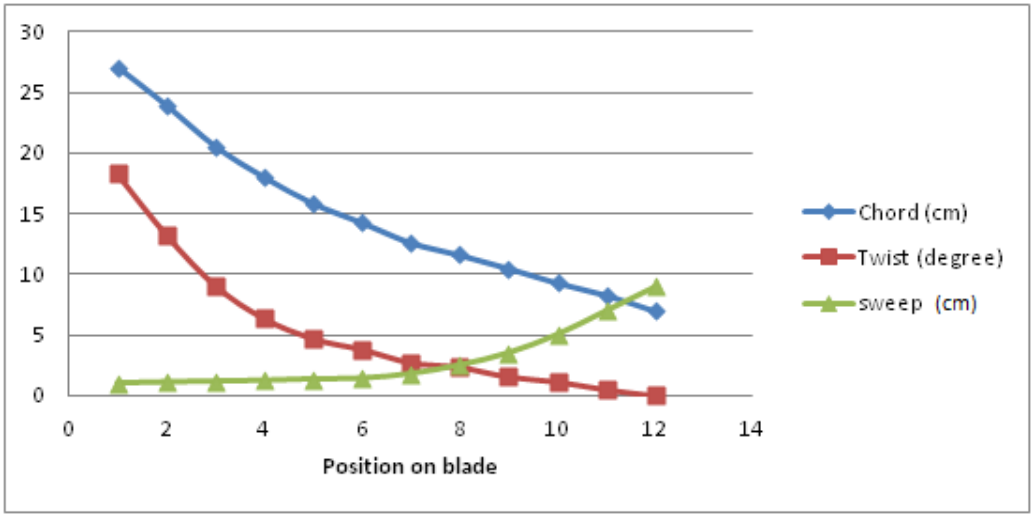


Figure 4-5 Blade geometry distribution along the radius

Table 0-1 Twist and chord distribution for each element along the radius

Relm	Twist (deg)	Chord (cm)
0.179	18.301	27.069
0.269	13.281	23.940
0.359	9.096	20.559
0.441	6.375	18.011
0.524	4.704	15.853
0.593	3.814	14.282
0.676	2.726	12.614
0.752	2.468	11.611
0.834	1.364	10.503
0.903	1.232	9.302
0.959	2.070	8.305
1.000	0.000	7.000

Next, the airfoil shape was required to determine the aerodynamic properties of lift and drag using a code called Xfoil. The airfoil shapes for the Skystream was found by scanning three sections of the blade. The scanning was done by using a profile gauge also called a contour duplication gauge.

It is a device with thin plastic/steel pins that are set against each other in a frame. The pins are always perfectly perpendicular to the frame, and when pressed against an object the pins take the cross-sectional shape of the object surface. The contour can then be traced. Figure 4-6 and Figure 4-7 below show the device and the traced profile shapes.

Figure 4-6 Tool used for scanning the blade profile

The airfoil shapes were scanned and digitized to get the coordinates of the airfoil, which would help us determine which airfoils were used to design the blade. The airfoil shapes from the profiler were traced onto a paper, scanned and digitized using a matlab code called 'digitize2' which is included in the Appendix for reference. The airfoil shapes were compared to check for uniformity across the length of the blade, to see if the same airfoil is used at all the sections. The report from NREL[7] says the S822 was selected to be the airfoil for Skystream blade. The profiles that were scanned was used to compare with other documented airfoil shapes. After comparison

the results were plotted as seen in Figure 4-7. It is desired that the blade tip airfoil be thin airfoils with high lift to drag ratio. For the blade chord thicker airfoils are desired for structural support and high C_{Lmax} values so that the blade root is not under stall. It can be inferred from the diagram below that that S823 airfoil was used at the root of the blade and S822 airfoil towards the tip. At the intermediate sections the two airfoils are blended.

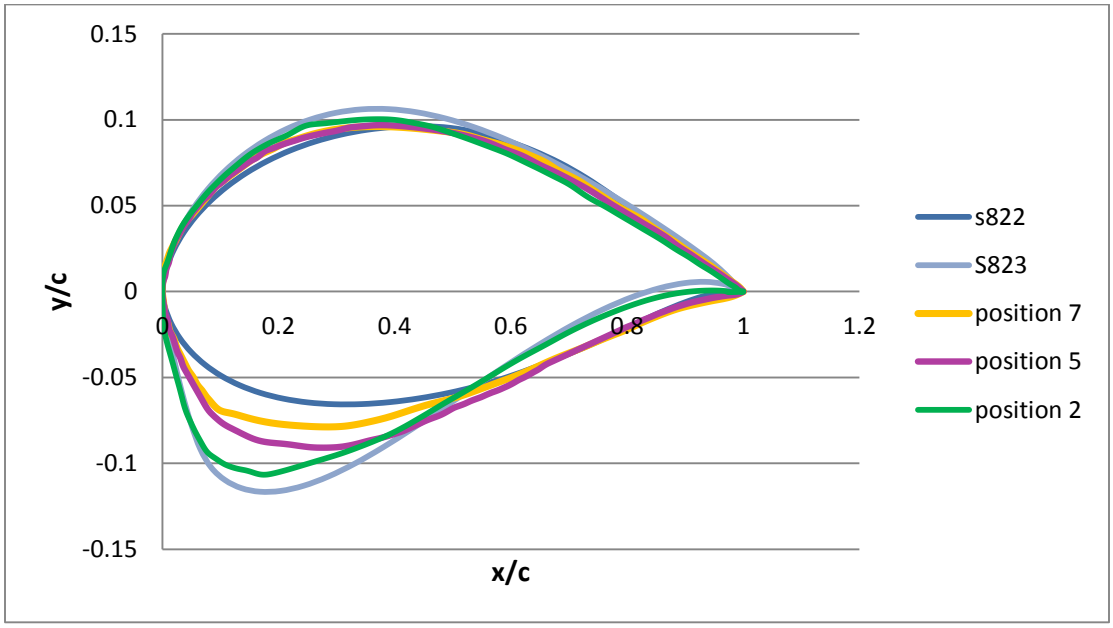


Figure 4-7 Airfoil profile shapes (axes are not to the same scale)

The aerodynamic properties of lift and drag for the airfoils were found by using the blade's coordinates. A coordinate file containing x and y coordinates for every point on the airfoil was made. The file has a .dat extension to be able to input into Xfoil. This file is included in the Appendix. To correctly make any airfoil coordinate file the first step is to non-dimensionalize the coordinates by the chord length of the airfoil. The data should be ordered such that it goes from the trailing edge (1,0) along the upper surface to the leading edge (0,0) and

back to the trailing edge (1,0) along the lower surface. The coordinates are arranged with the same number of decimal places and a one line header is used for the name of the coordinate file.

Performance prediction with scanned data

Xfoil

XFoil is a tool used for analyzing subsonic airfoils, given the 2-D airfoil coordinates and Reynolds number.

A coordinate file for the airfoil shape is used by Xfoil to get the Lift and Drag coefficients for any desired range of angle of attacks. For a range of Reynolds number from 0.1 x 10^6 to 0.3 x 10^6 the C_L and C_D data were obtained for each of the airfoils.

The output we get from XFoil is used to make the airfoil file with C_L and C_D data that is used by WT_Perf. WT_Perf is a code developed by NREL which is based on blade element momentum theory. It helps us get steady state rotor performance with root and tip losses.

Aerodynamic properties of airfoil closer to root (position 2) of the blade at 0.25 million is shown in the plots below.

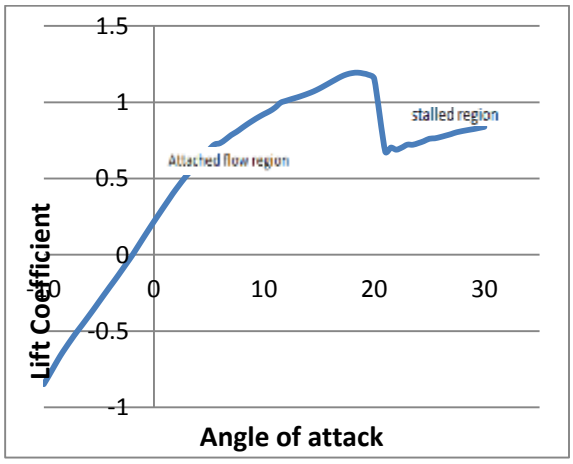


Figure 4-8.Lift coefficient curve for airfoil at position 2 on blade

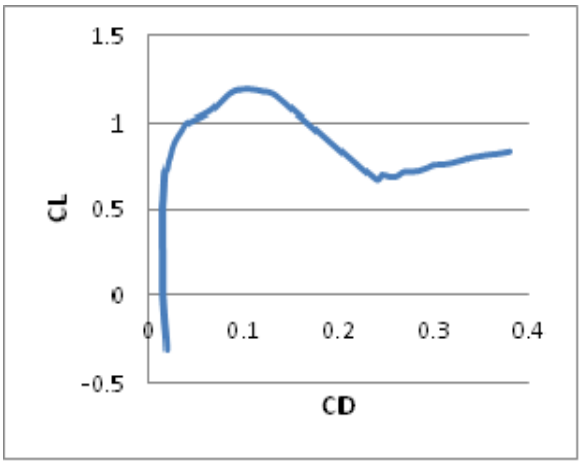


Figure 4-9.Lift Vs Drag curve for airfoil at position 2 on blade

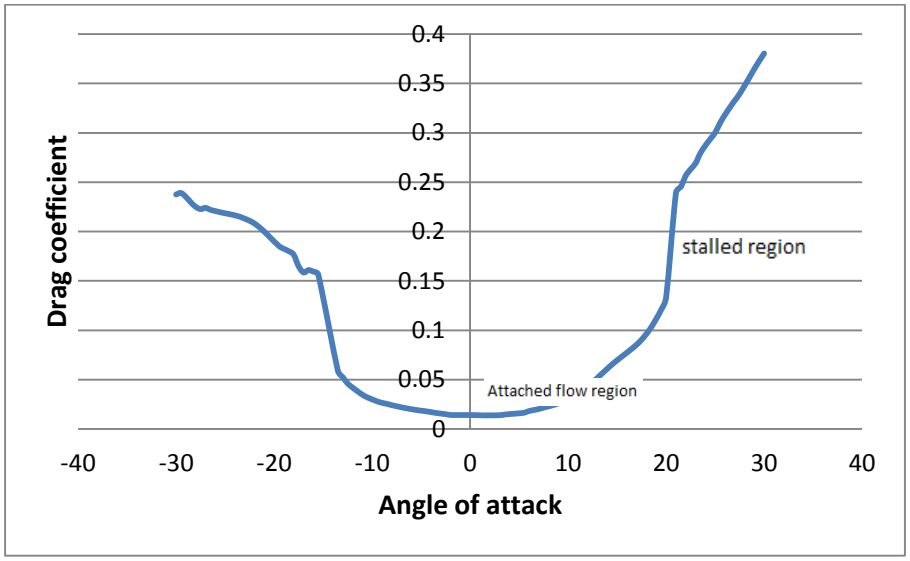


Figure 4-10 Drag Coefficient curve for airfoil at position 2 on blade

Aerodynamic properties of airfoil closer to the tip (position 7) of the blade at Reynolds number 0.26 million is shown in the plots below.

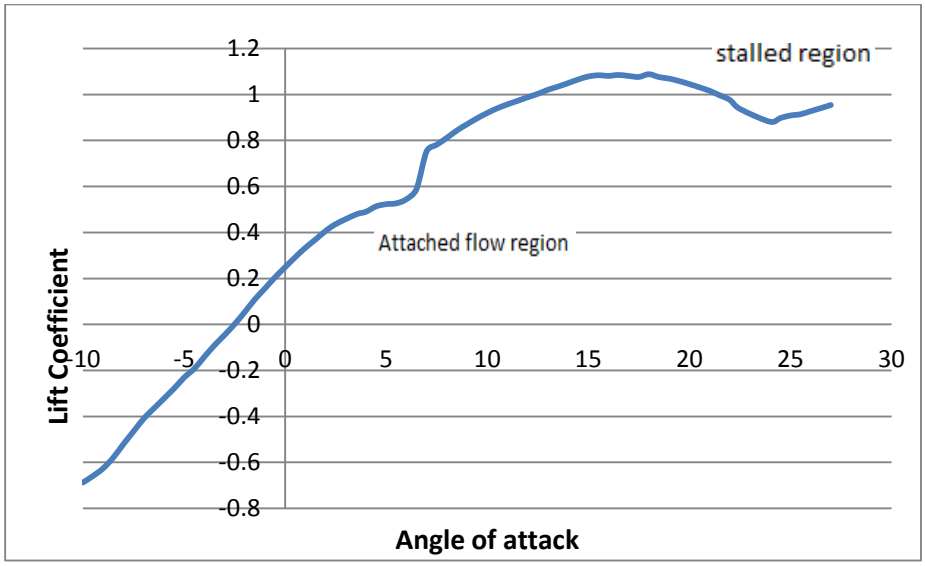


Figure 4-11 Lift Coefficient curve for airfoil at position7 on blade

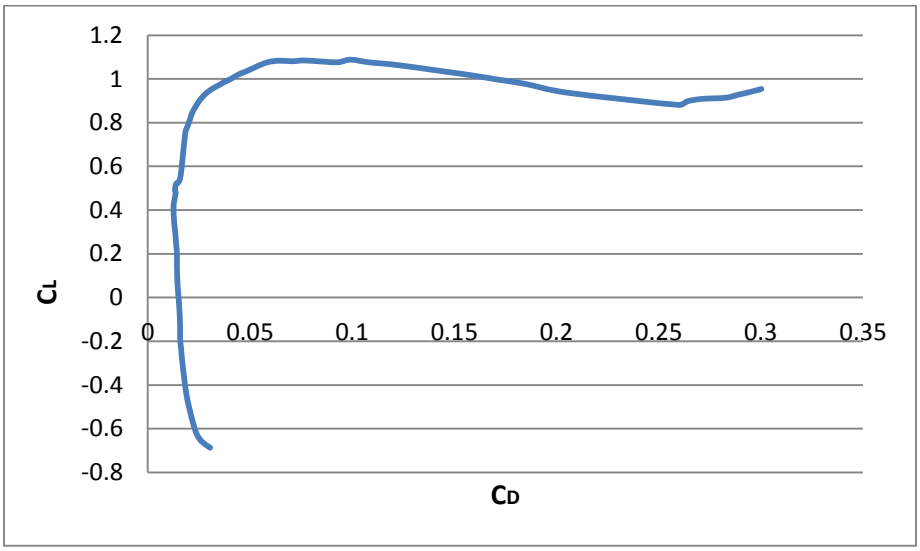


Figure 4-12 Lift Vs Drag curve for airfoil at position 7 on blade

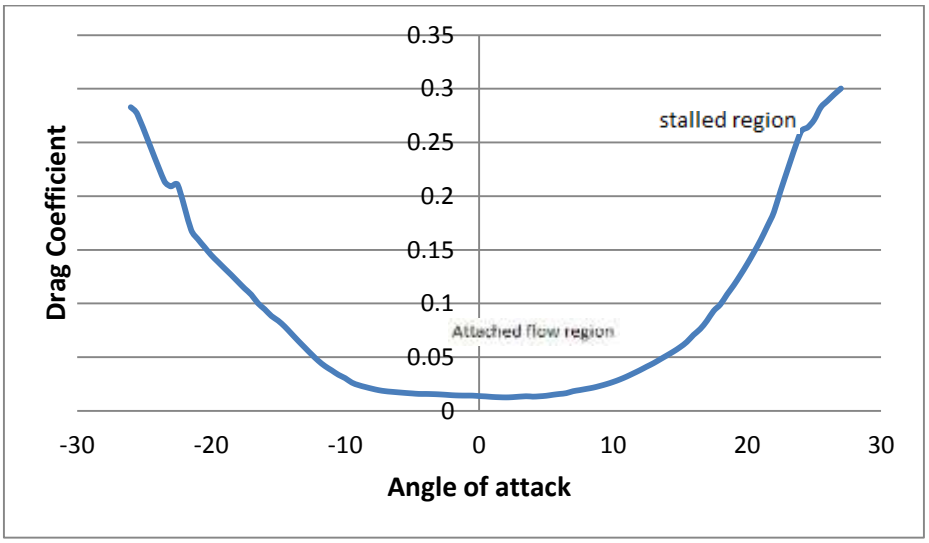


Figure 4-13 Drag coefficient curve for airfoil at position 7 on blade

The wind turbine airfoils are best chosen such that the airfoils at the blade root have a high C_{Lmax} and are thick for structural support. At the tip, thin airfoils are preferred with a high lift to drag ratio.

The properties of a good wind turbine airfoil is that it must have a maximum lift to drag ratio close to its C_{Lmax}. It is also desired that there be a flat C_L vs alpha curve after C_{Lmax} has been reached, so that there is a smaller peak in the load during a gust of wind. From the plots above it can be seen that the airfoils fairly satisfy the criteria for a good wind turbine airfoil. From the above plots for both of the airfoils we can see that the stall angle is approximately 18 degrees.

WT_Perf

In order to run WT_Perf an airfoil input file is made in .dat format. The data needed for this airfoil file is obtained using a spreadsheet called AirfoilPrep. AirfoilPrep is an extremely useful preprocessing tool to run codes like WT_Perf and AeroDyn. It extrapolates 2-D airfoil data to high angles of attack and also, when required, modifies 2-D airfoil data to account for 3-D effects. This airfoil file contains values of C_L and C_D for a range of angles of attacks from -180 to +180 degrees for a set of Reynolds numbers. The format of this airfoil file depends on the version of WT_Perf and can be obtained from the sample files available from the downloaded files. The airfoil file for the Skystream in this study is included in Appendix.

WT_Perf is a wind turbine performance prediction code formerly released by the National Renewable Energy Laboratory (NREL). It is based on blade-element-momentum theory (BEM). An input file is used to run the WT_Perf executable. The input file contains the blade geometry, turbine data, aerodynamic data and other user specified parameters of the model and algorithm. The output from WT_Perf gives us the prediction for power, torque, thrust and C_P produced at the range of specified wind speeds, RPMs and pitch. Other properties along the blade (e.g. angle of attack, axial induction factor, C_L, C_P, Thrust, Torque, power, Reynolds number) are

also obtained which help us understand their distribution on the blade. The WT_Perf user guide is a very useful source to learn using the code.[13]

Output and Discussion

The generated results from WT_Perf help us understand the aerodynamics around the blade. The results obtained also help in computing the performance of the wind turbine.

The plots below show the variation of Reynolds number along the blade. We can see that the Reynolds number distribution is fairly constant along the blade at different tip speed ratios. The slight variation at the root at 100 RPM is because in the velocity of the blade, $\Omega r(1+a')$ the value or 'r' is low at the root compared to that at the tip. Therefore the relative velocity varies more. This is shown in Figure 4-14. However at high RPMs the effect of r is very low as compared to the effect of Ω and the relative velocity does not vary much.

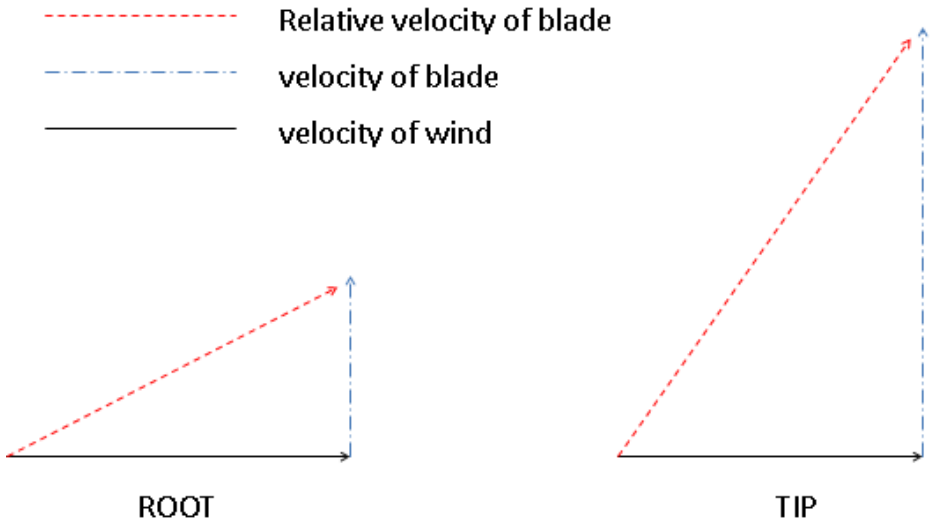


Figure 4-14. Velocity triangle for wind turbine blade

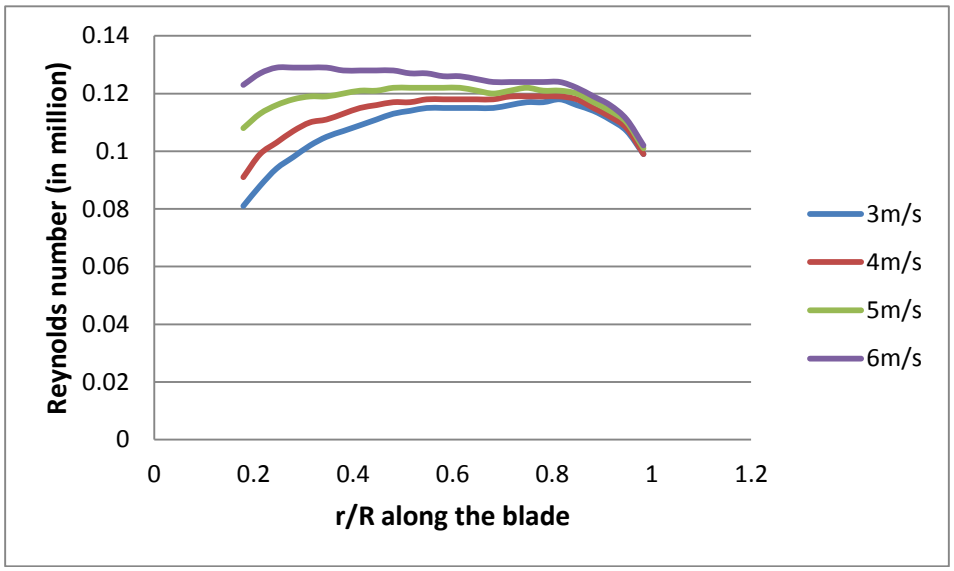


Figure 4-15. Reynolds number distribution at 100 RPM

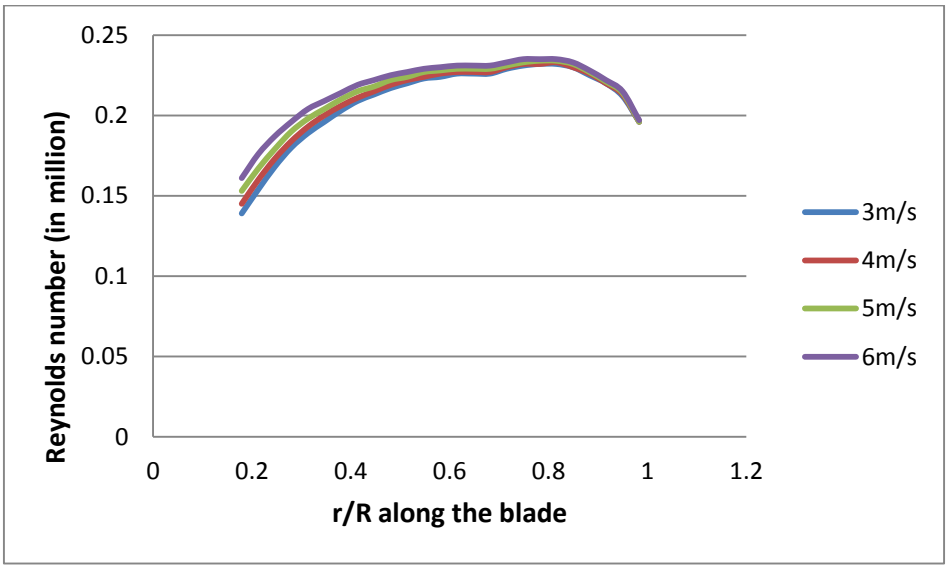


Figure 4-16. Reynolds number distribution at 200RPM

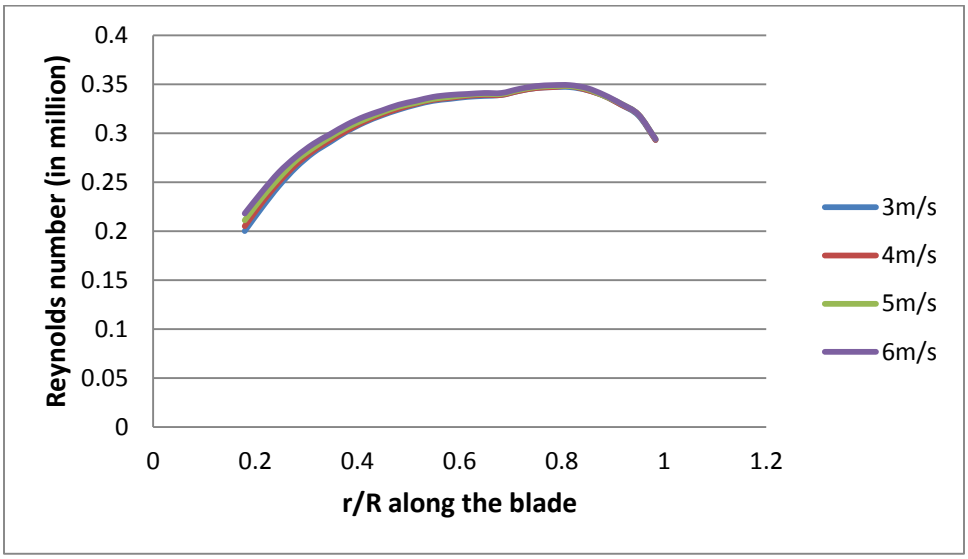


Figure 4-17. Reynolds number distribution at 300 RPM

With the scanned blade geometry and the airfoil properties generated from XFoil, WT_Perf was run to get the analysis of loads and its performance. The output was generated at a range of wind speeds from 3 m/s – 6 m/s and at RPMs from 100-300.

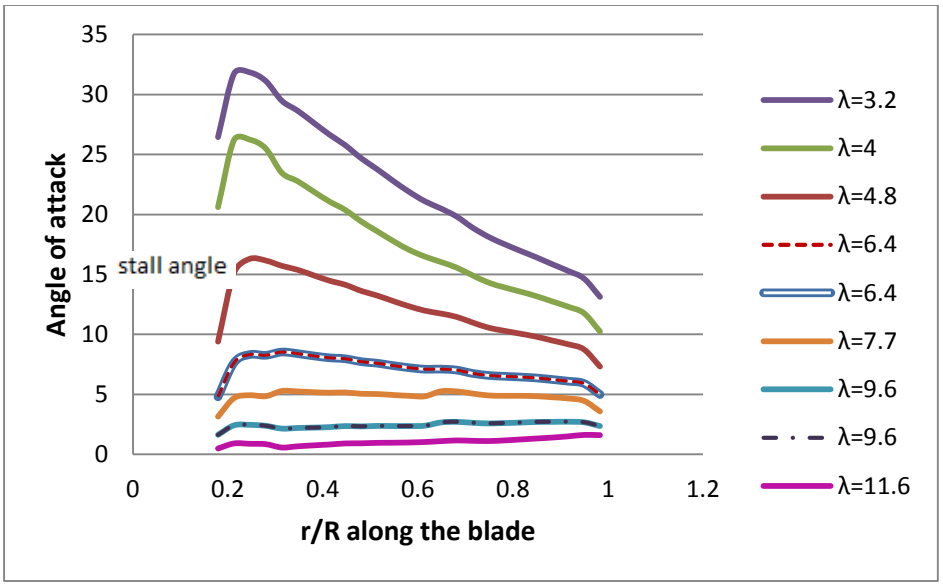


Figure 4-18. Angle of attack distribution at varying tip speed ratio

From Figure 4-1818 we can see that the root of the blade is stalled for λ=4 and for λ=3.2 about 70% of the blade is under stall conditions.

The values for the velocities and tip speed ratios for the corresponding RPM are listed below.

Table 0-2. Tip speed values from corresponding velocity and RPM

RPM	v (m/s)	λ
100	3	6.4
100	4	4.8
100	5	4.0
100	6	3.2
200	4	9.6
200	5	7.7
200	6	6.4
300	5	11.6
300	6	9.6

The plots below show the variation in lift coefficient along the blade. It is desired to have a higher lift coefficient at the root and low lift coefficient at the tip. This is desired as we want the airfoil at the root to have a high C_{LMax} value.

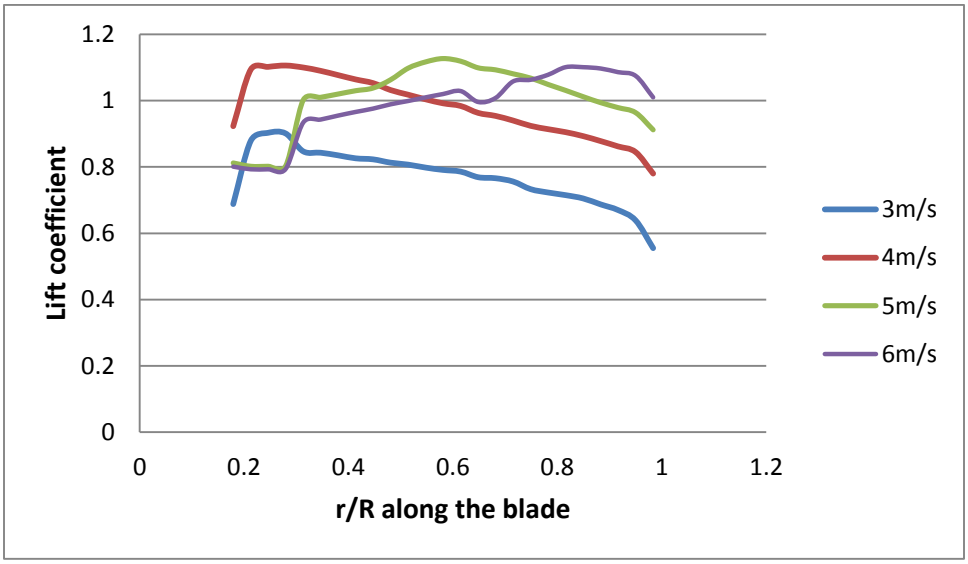


Figure 4-19. Lift distribution along blade at 100 RPM

In Figure 4-19, for 5 m/s and 6 m/s, which corresponds to λ=3.2 and λ=4, we can see the irregular lift distribution due to the blade stall regions.

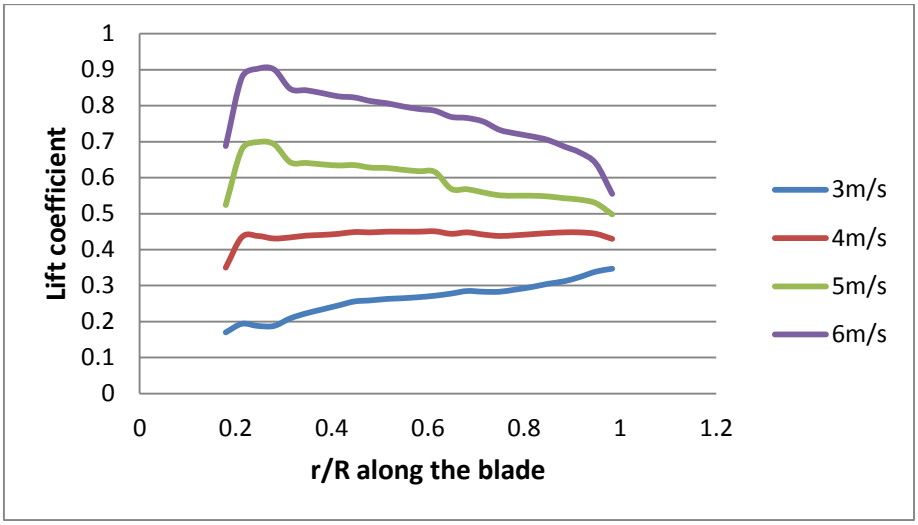


Figure 4-20. Lift distribution along blade at 200 RPM

The low values of lift coefficient in Figure 4-200 and Figure 4-21 is because parts of the blade or the whole blade is in negative angles of attack due to high RPM and low wind speeds. High RPM's under low wind speeds also leads to low lift coefficient at the root and higher lift coefficient at the tip. This is undesired as this would lead to bending and higher stress loads at the tip.

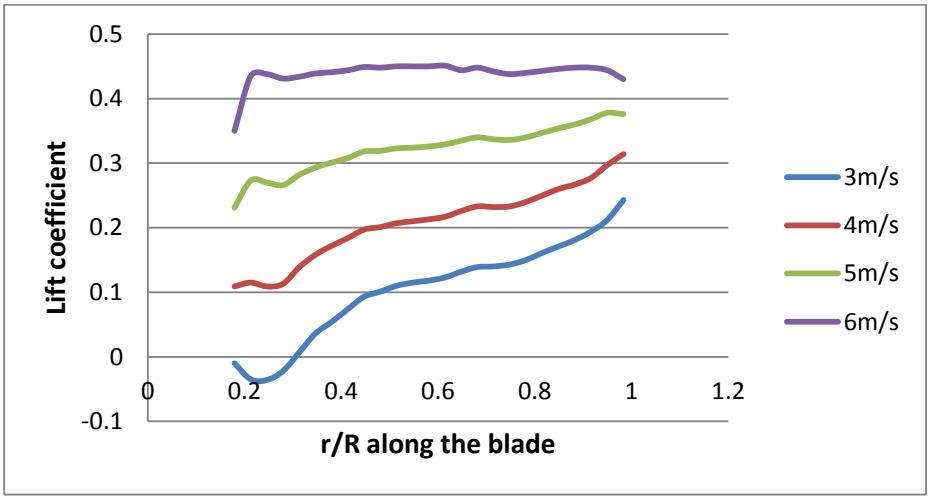


Figure 4-21. Lift distribution along blade at 300 RPM

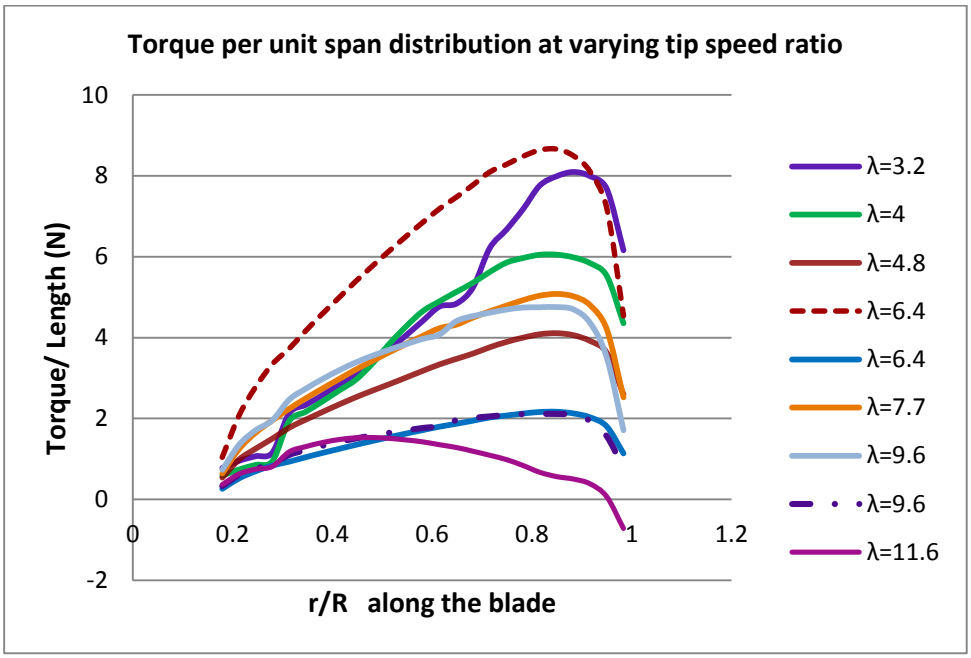


Figure 4-22. Torque per unit span at varying tip speed ratio

The effect of stall can be clearly seen in the above Figure 4-222. For regions of that blade that are under stall at λ=3.2 and λ=4 we can see that torque is not linear. The more flat distribution of torque at higher tip speed ratios can be attributed to the increased effect of drag which reduces torque as the square root of the local speed ratio. This effect of drag at low angles of attack causes significant loss of power which can be seen at high tip speed ratio.

The power is also effected at low tip speed ratio due to the stalled regions on the blade, which can be seen in Figure 4-23 below. Comparing the results from Figure 4-244 and Figure 4-23 it can be inferred that the optimum tip speed ratio for the Skystream wind turbine is between 6 and 8 for best power performance.

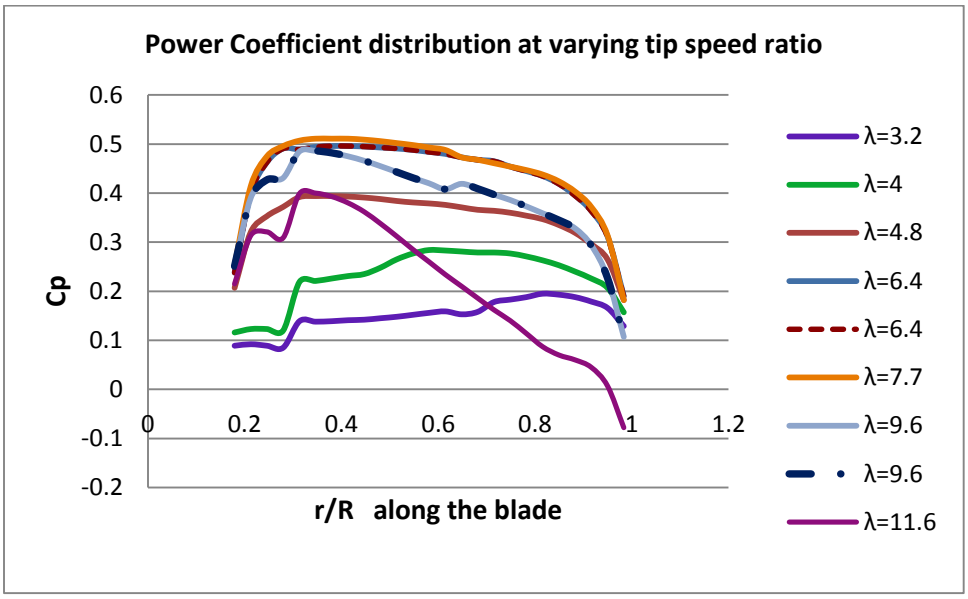


Figure 4-23. Power Coefficient at varying tip speed ratio

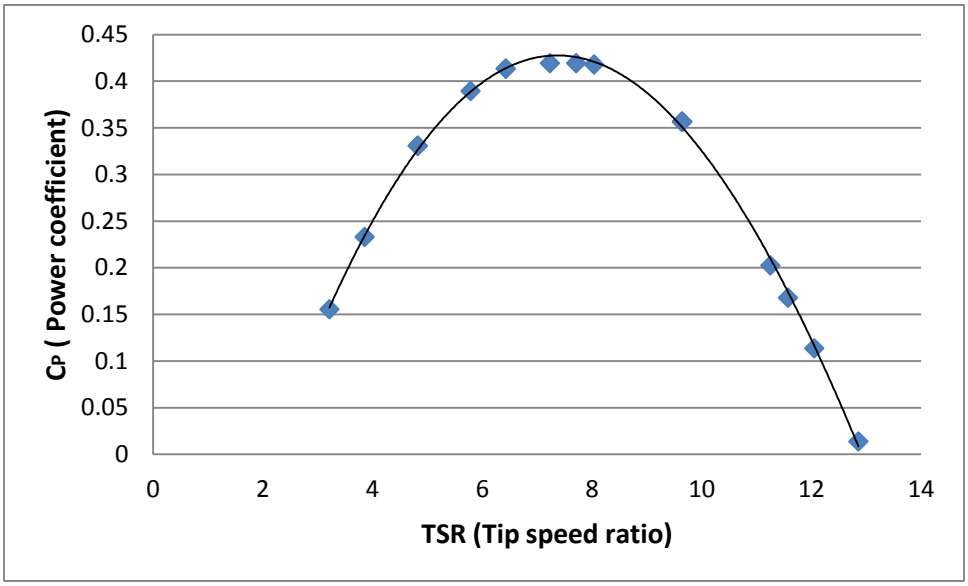


Figure 4-24. Non- dimensional power curve

The Cp vs TSR plot above (Figure 4-244) tells us that the blade is designed for optimum performance when the tip speed ratio is between 6 and 7. This result concurs with the results from Torque distribution, Cp distribution and C_L distribution along the blade.

Chapter 5

Skystream Experimental Data Collection and Data Processing

The Skystream wind turbine operates and produces data via its Skyview data acquisition system and software every day and at all times. Having such a large amount of data gives us the ability to analyze its power performance; as a sample small scale wind turbine at a site which had not been studied prior to installation, as would be the case for most small scale wind turbine owners for small businesses, farms, schools etc. Penn State has one such wind turbine from which we collect data for analysis and also serves as a small alternative power source.

Structure and format of collected data

The data from the wind turbine are stored in per day files which are in text format. They all bear the name swwtMMDDYYYY where MM stands for the month, DD for the day and YYYY for the year of the collected data. The file contains comma separated values of different meteorological, operational and performance related data at 3 second intervals. Thus the data acquisition of the Skystream wind turbine is every 3 seconds. These are files with large amounts of data with no header information on what each number signifies. These files will be referred to as the 'Raw Data'. Therefore the first step for data analysis was to determine the headers for each number. Below is listed what each column in the data file signifies-

1. Serial number -Looks like " DECI: +STAT turb=109725,vers=2.02 140597731,2506870"
2. Inv Time -time reported by inverter

3. watt-hours -energy production tracked by the sensors in DAQ
4. Voltage In (volts) - NA
5. Voltage DC Bus -NA
6. Voltage L1 (volts) -grid voltage
7. Voltage L2 (volts) -grid voltage
8. Voltage rise (volts) -voltage rise between L1 and L2
9. min v from rpm -NA
10. Current out (amps) -current generated bu generator
11. Power out (watt) -net power of DAQ and generator
12. Power reg (watt) -power from generator
13. Power max (watt) -controller maximum power production (variable)
14. Line Frequency (Hz) - NA
15. Inverter Frequency (Hz) -NA
16. Line Resistance (ohm) -NA
17. RPM -rotations per minute
18. Windspeed (ref) (m/s) -an integer value of windspeed from power
19. TargetTSR -Controller mandated tip speed ratio
20. Ramp RPM -NA
21. Boost pulsewidth -NA
22. Max BPW -NA
23. Current amplitude -amplitude of AC current
24. T1(F) -temperature from temperature sensor
25. T2(F) - temperature from temperature sensor
26. T3(F) - temperature from temperature sensor
27. Event count -count of events

28. Last event code -code of last event
29. Event status -NA
30. Event value -NA
31. Turbine status -NA
32. Grid status -NA
33. System status -NA
34. Slave Status -NA
35. Access Status -NA
36. Timer -NA
37. Wind 1 (m/s) -windspeed in one direction at 58 ft
38. Wind 2 (m/s) -windspeed in another direction at 56 ft
39. Wind 3 (m/s) -windspeed at 28ft
40. Wind direction -with respect to magnetic north
41. Temperature(°C) -of the day
42. Barometric Pressure(Hg)- of the day
43. Relative Humidity -humidity in percentage
44. date time -date of the day and time (in hours and minutes)

For a lot of the values 'NA' is specified, it is because we do not know much about those parameters.

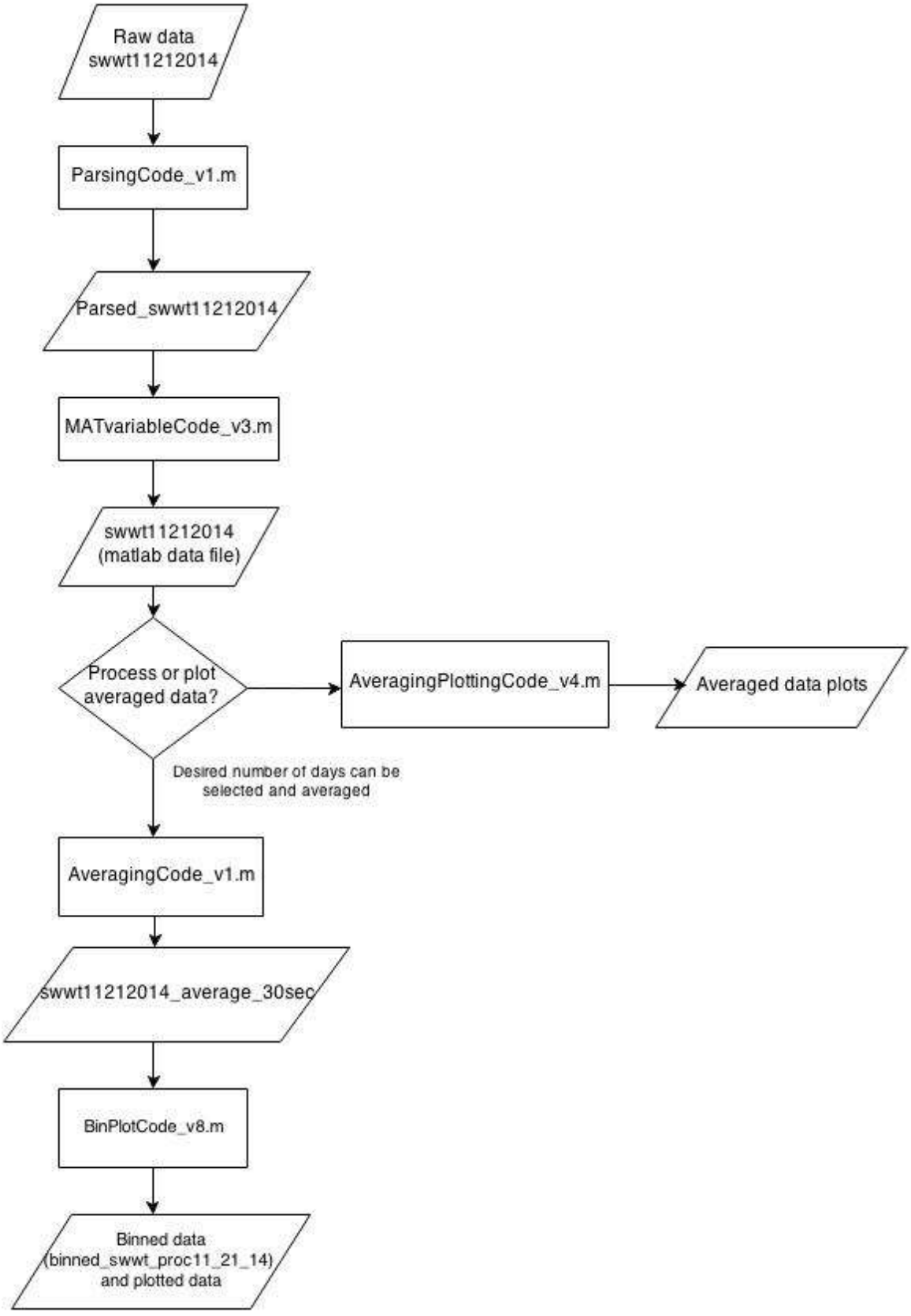
Data processing method

The 'Raw data' is called so since they have not been processed and are collected raw from the wind turbine. Therefore for analyzing the data, these data are run through MATLAB codes written by a colleague (Brian Wallace). The data is first run through a code called the 'ParsingCode'. Parsing means to analyze data minutely, therefore this code looks at every row and removes all extra characters that sometimes comes in the beginning of a data series. This data is saved with a name 'parsed_swwtMMDDYYYY'. After this process the data is converted to MATLAB data files and to do this process it is run through a code called the 'MATvariableCode'. This data is called Matlab variable files and are named the same as the raw files, except they are MATLAB files and not text files like the raw data and they now include the header information to describe the details of the data.

The next steps of processing are done to analyze the data using industry standards. The MATLAB variable files are first averaged by using an 'AveragingCode'. The data can be averaged at any desired averaging interval in multiples of three seconds. For our study we have averaged data for 30 seconds, 60 seconds and 10 minute intervals. Some of the averaged data is plotted and analyzed in the 'Experimental Results' section below. We then bin our data for a desired time period. Data binning is a process of combining data into small clusters. Our data is binned according to wind speed in bins of 0.5 m/s. This process also makes it easier to plot large amounts of data. The data is binned and plotted using the code 'BinPlotCode'. Below is a flow chart that diagrammatically shows that process.

A software called Windographer was also used as a tool to understand large sets of data. It also helped us crosscheck the results that we got from our processing codes.

The Flow chart below shows the different steps used for formatting the data that is obtained from the Skystream wind turbine, operational at The Pennsylvania State University.


Raw data
swwt11212014
ParsingCode_v1.m
Parsed_swwt11212014
MATvariableCode_v3.m
swwt11212014
(matlab data file)
Process or plot
averaged data?
AveragingPlottingCode_v4.m
Averaged data plots
Desired number of days can be
selected and averaged
AveragingCode_v1.m
swwt11212014_average_30sec
BinPlotCode_v8.m
Binned data
(binned_swwt_proc11_21_14)
and plotted data

Experimental Results

The data from the wind turbine for one day has been analyzed in this section. The different plots from monthly, daily and one year data has been compared for a smaller as well as a broader look at how the wind turbine performs. The plots in this section have been generated using the MATLAB codes, 'AveragingPlottingCode' and the 'BinPlotCode'.

First, the wind turbine data from the Skystream for one day was plotted for a closer look at how the weather on each day affects the power production. The selected day is a cold snowy winter day of December 30th, 2013.

Figure 5-1 and Figure 5-2 below show that there is a variation of wind speed and power on this day. From the former plot we can see that the wind speeds are lower in the evening as compared to morning and afternoon. From the power plot in Figure 5-2 we can see that the wind turbine is producing power in the morning and afternoon only. We can conclude by a closer look at the figures that the wind turbine produces power only when the wind speed is some value above 3 m/s. The correlation from these two plots also shows that the anemometer readings are correct and in accordance with the power generation.

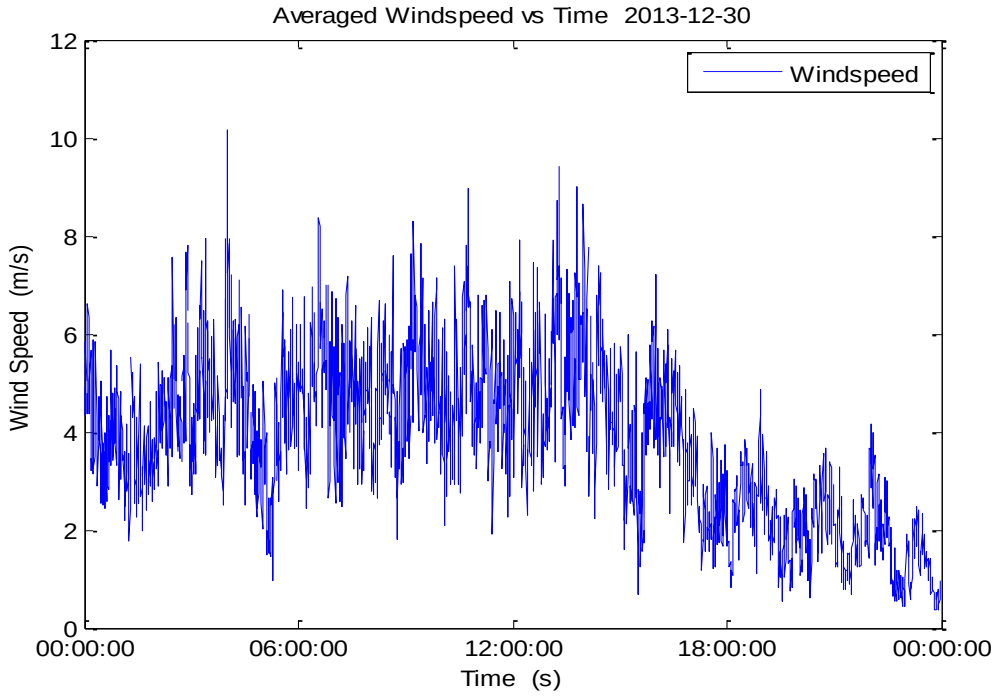


Figure 5-1. 30 second averaged wind speed vs. time for 12-30-2013

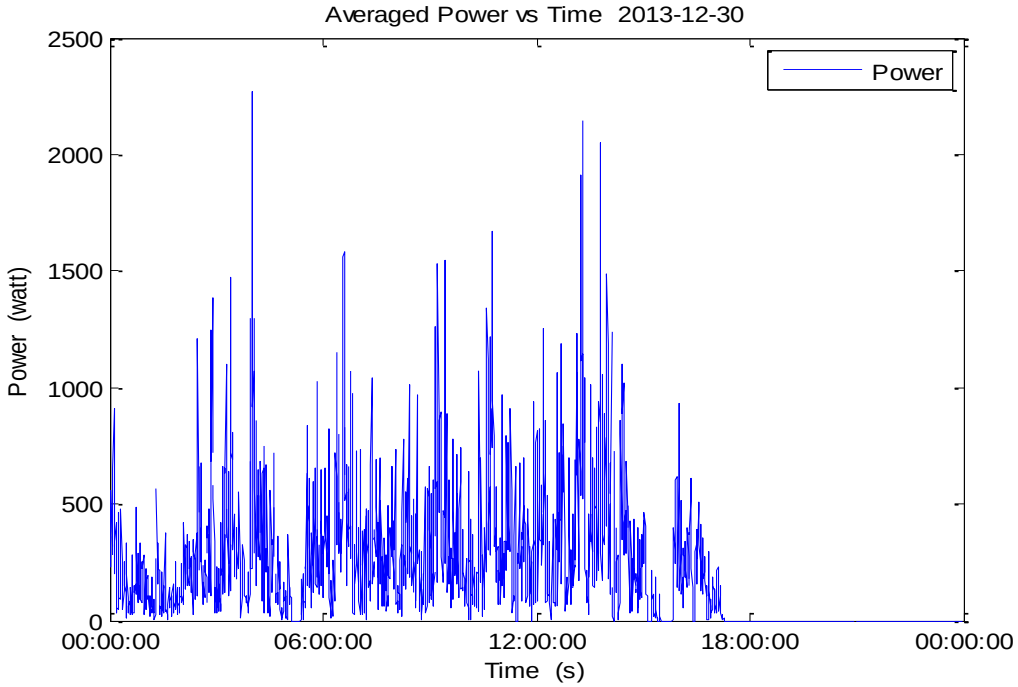


Figure 5-2. 30 second averaged power vs. time for 12-30-2013

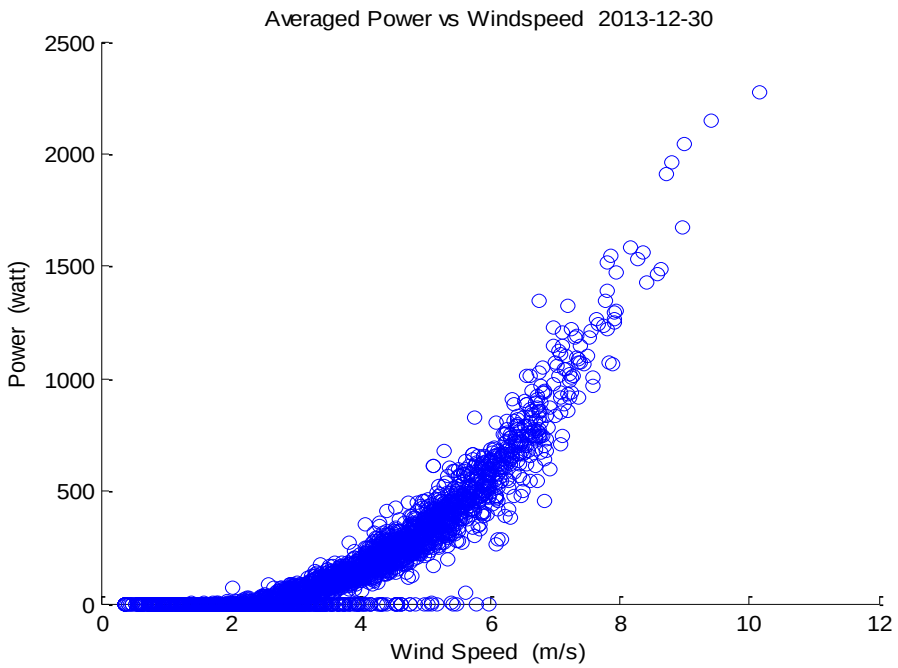


Figure 5-3. 30 second averaged power vs. wind speed for 12-30-2013

Figure 5-3 above shows the power produced for a range of wind speeds. We see that the power produced is sometimes zero even at higher wind speeds than 3 m/s. When observing the behavior of the turbine, it has been noted that sometimes it takes a higher wind speed to initiate start up, depending on the orientation of the blades with respect to the wind direction. The same reason can be attributed to the large range of power values for a single value of wind speed. This could be due to a variety of reasons e,g gusts, turbulence, and the shift in the direction of the wind. The range of power values are narrower at higher values of wind speed because the frequency of higher wind speeds is much lower at this location.

Figure 5-5 which shows the variation of power with RPM. This plot has a narrower range of power for a value of RPM as compared to power vs wind speed plot in Figure 5-3. This is because the RPM of the blades directly correlates to the power generated by the wind turbine.

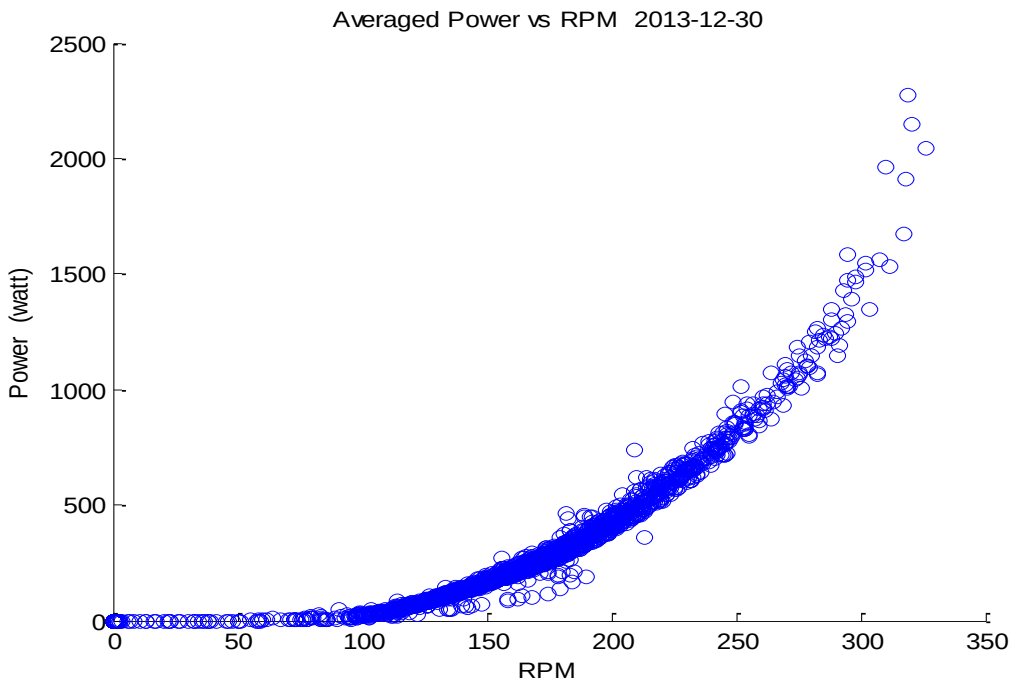


Figure 5-4. 30 second averaged power Vs RPM on 12-30-2013

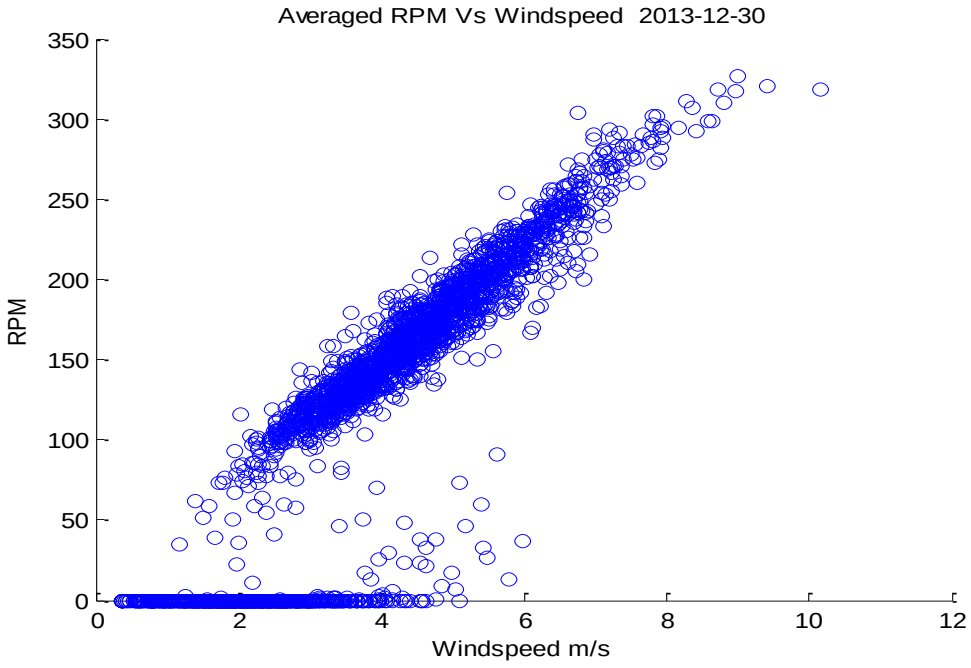


Figure 5-5. 30 second averaged RPM Vs wind speed on 12-30-2013

Figure 5-5, above, is useful as a sample to understand how the wind speed affects the RPM of the blades. As discussed in Figure 5-3, sometimes there is a start-up period in which the blades may be oriented in a direction misaligned with the wind, and a higher wind speed is required to begin generating rotation. This is why we see zero RPM values at wind speeds of 3-5 m/s and lower RPM values in the 3.5 – 6 m/s range for some cases than others.

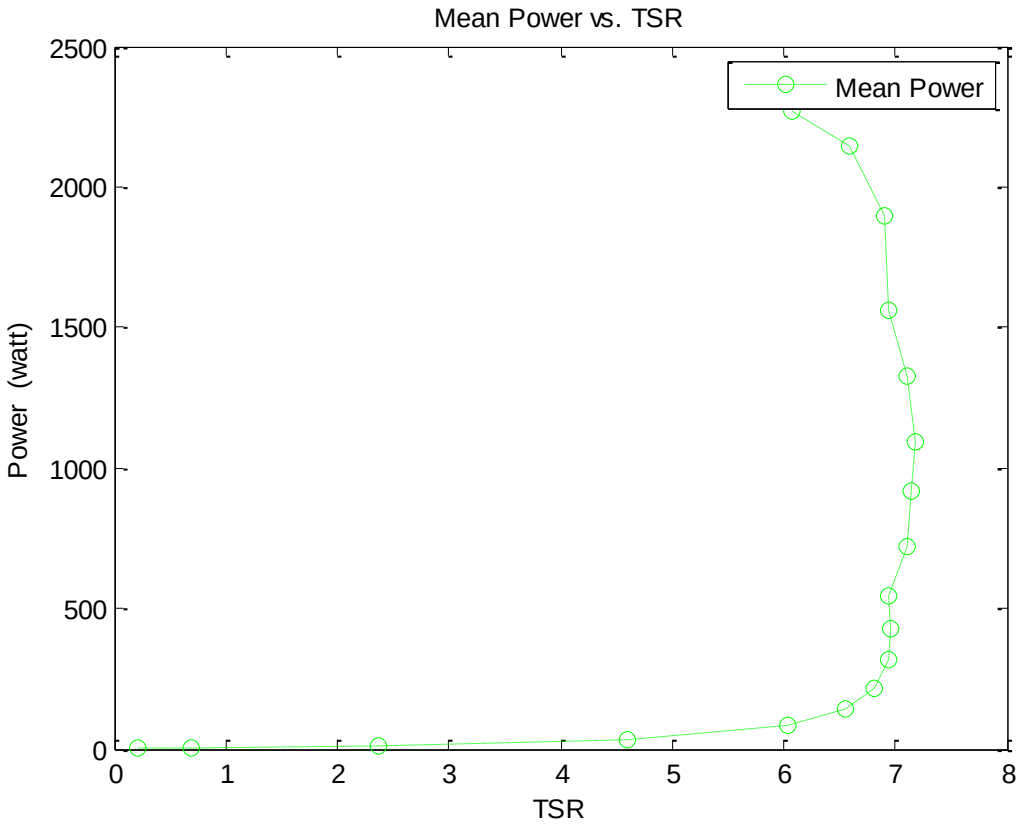


Figure 5-6. 30 second averaged Mean Power Vs Tip speed ratio for12-30-2013

Tip speed ratio is an important parameter to understand in studying the power of a wind turbine. In Figure 5-6 we see that the rotor produces power around a tip speed ratio of 7. This is usually a common value for optimal TSR for most wind turbines with 3 blades.

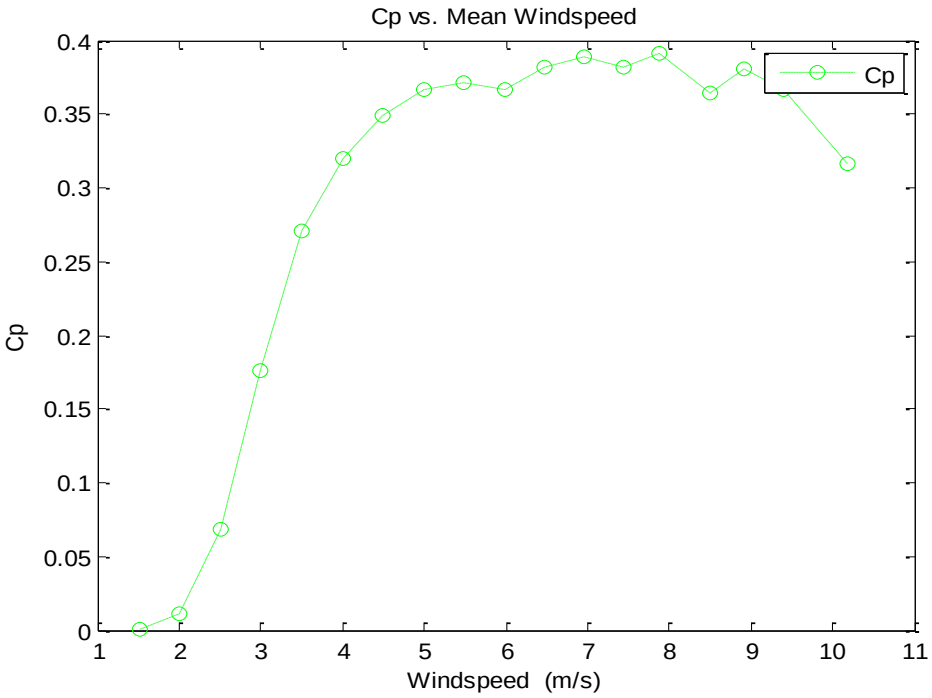


Figure 5-7. 30 second averaged power coefficient vs wind speed for 12-30-2013

Figure 5-7 helps us to see the correlation between power coefficient and wind speed. We see that the increase in wind speed from 3 m/s to 5m/s creates a large jump in the Cp value. For higher values of wind speed the Cp remains relatively constant until the controller steps in and we start to see a dip in Cp above approximately 8.5 m/s.

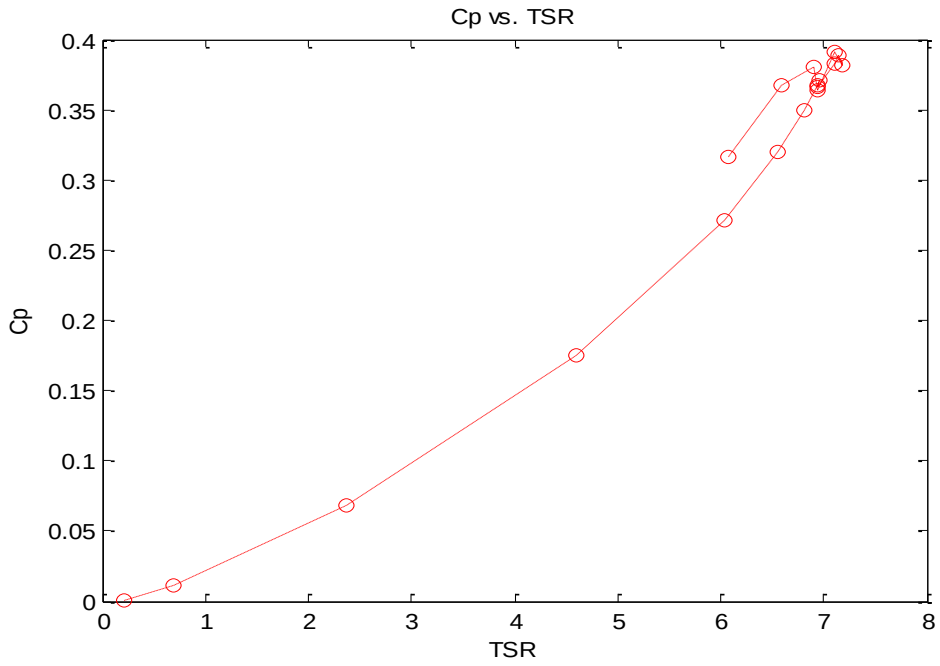


Figure 5-8. Power Coefficient Vs Tip speed ratio for 12-30-2013

Tip speed ratio is defined as the ratio of the speed of the rotor to the free stream velocity of the wind. When the rotor rotates slowly, a lot of wind is allowed to pass through without extracting much energy from it. If the tip speed ratio is too high, then the Cp value starts to reduce, thus there is an optimum value for tip speed ratio depending on the solidity and number of the turbine blades at which maximum Cp can be attained.

In Figure 5-8 we see C_p values increasing until a tip speed ratio of 7, at which point the plot takes a turn for lower TSR values and slightly higher C_p values before reducing. It can be inferred from this plot that the controller brakes the wind turbine to obtain lower TSR. This is probably done to maintain an optimum value of Tip speed ratio because at higher values the rotor appears as a large flat disc to the wind that creates excess drag and so as to not over speed the generator.

The next step to better analyze the data, is to compare certain plots for one day, one month and one year. The data for one day is averaged for 30 seconds, the one month data is averaged for 60 seconds and the one year data is averaged for 10 minutes. It is important to remind that the data collected is at 3 second intervals from the Skystream wind turbine.

The averaging done for our data is a simple average and not a weighted average, therefore the results are not to our desired accuracy. An example of this is further explained at the end of this section with Figure 5-22 and Figure 5-23. A weighted average would give more accurate results, but we are trying to explain the data with industry methods of processing, so we stick to simple averaging. In industry the data is usually averaged at 10 minute intervals and for a large amounts of data. However our averaging method is slightly different (single day data averaged for 30 seconds, one month data averaged for 60 seconds and one year data averaged for 10 minute) for more accurate results.

All four plots shown below depict the variation of Power with wind speed. The scatter plots shows the daily variation for two different days. Figure 5-10 and Figure 5-12 shows us the variation for one month and one year respectively. Note that the figures below don't have the same scale.

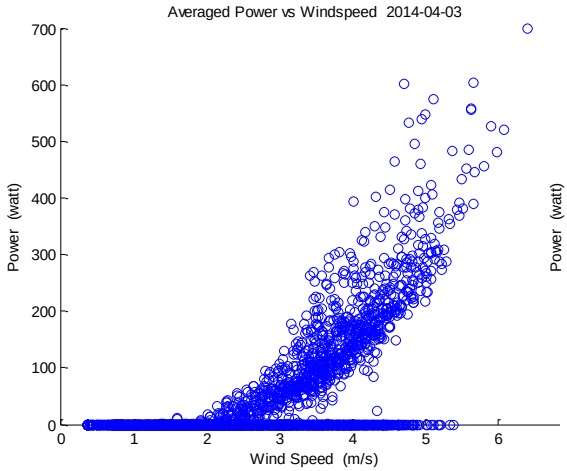


Figure 5-9. Power vs. wind speed variation for 04-03-2014

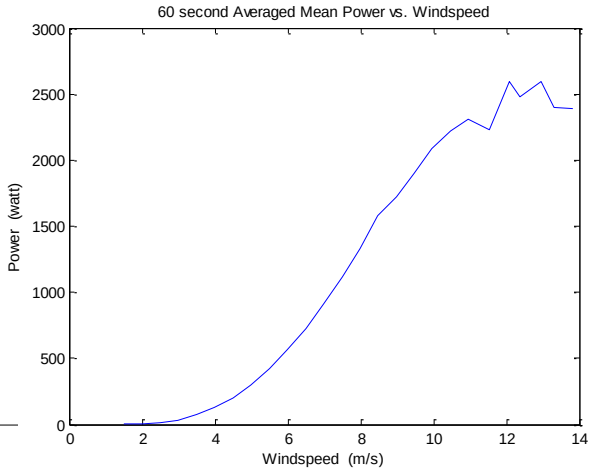


Figure 5-10.Power vs. wind speed variation for April 2014

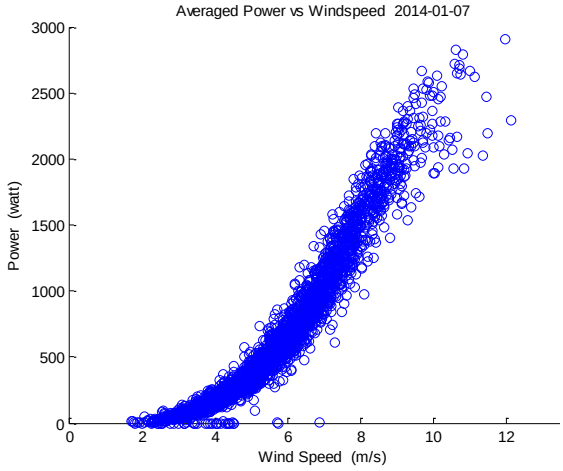


Figure 5-11. Power vs. wind speed variation for 01-07-2014

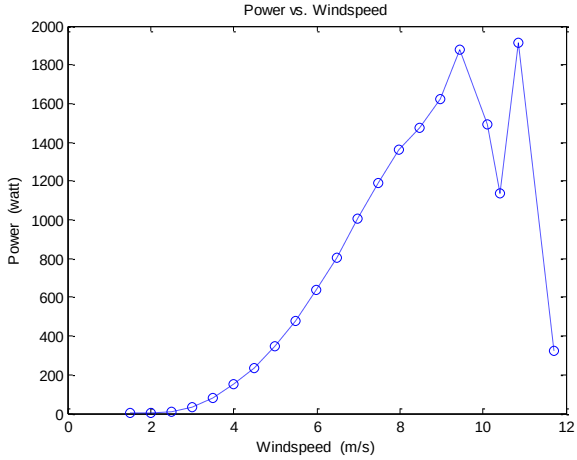


Figure 5-12. Power vs. wind speed variation for one year

Figure 5-9 is a scatter plot of the power generated at wind speeds throughout the day. We see that even at higher wind speeds the power generated is zero because the wind turbine was yawed at a different direction when the anemometer was still recording the wind speed data. This can be verified when we take a close look at Figure 5-13. The wind direction on the previous day and in the early hours on April 3rd changed from North West to North East in a few hours. The wind turbine must have taken a while to sense strong winds from NE and would have taken some

time to yaw towards the wind direction. Therefore when the strong winds close to 5 m/s were experienced (marked by the red line on the figure) the wind turbine was not oriented towards the wind and therefore no power was generated.

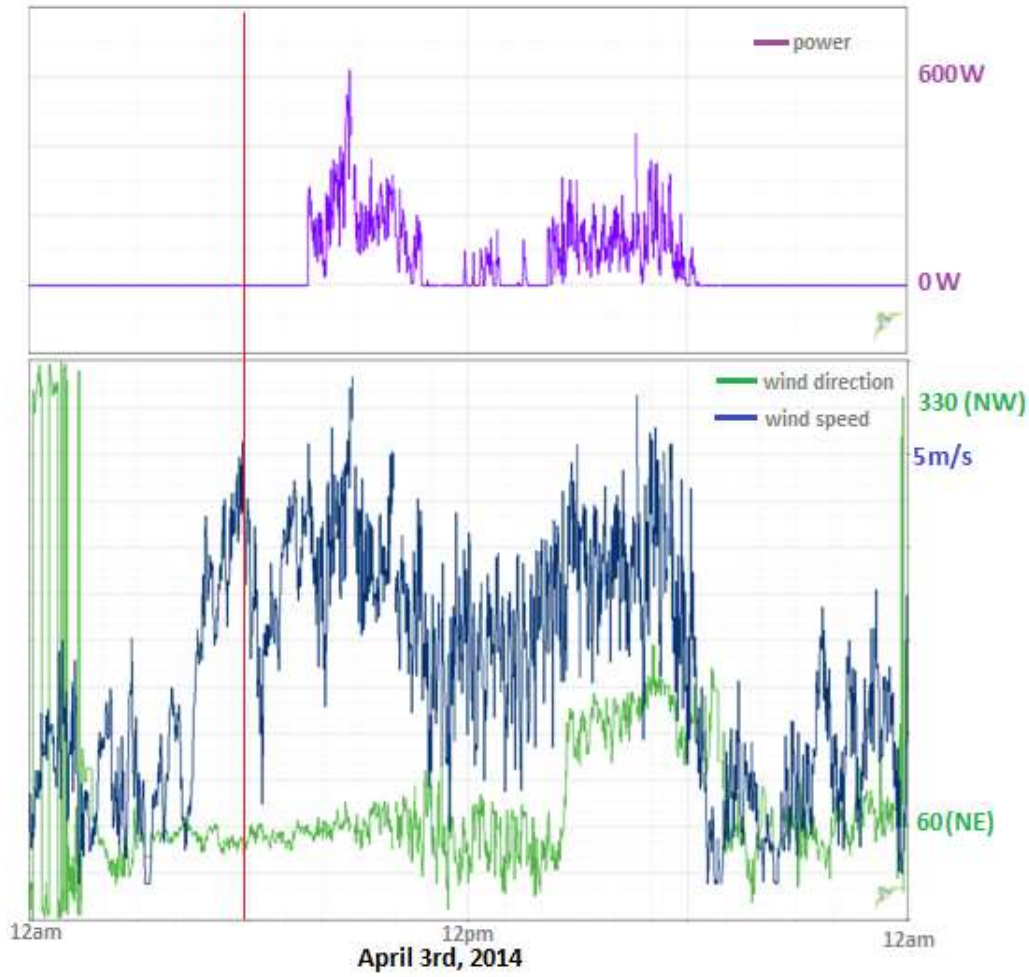


Figure 5-13. Power generated (in purple), Wind Direction (in green) and wind speed (in blue) for April 3rd, 2014 (whole day)

For lower values of wind speeds with zero power it would mean start up for the wind turbine took time to establish. Also we see power generation below cut-in wind speed, this would be because the wind turbine would already be rotating due to momentum, when the wind just dropped its speed. Figure 5-11 when compared with Figure 5-9 shows to be a much windier day. We see that there is very little startup speed required and above the cut-in speed of 3 m/s the wind

turbine is generating power well.

It is interesting to notice the dip and rise in the power scatter points at wind speeds above 10 m/s as can be seen the monthly and yearly power vs. wind speed plots. Comparing Figure 5-10 and Figure 5-12 we can see a very strong correlation in trend for the power produced at most values of wind speed. It is hypothesized that this variation is caused by the controller controlling the tip speed ratio and therefore the power generation starts to vary.

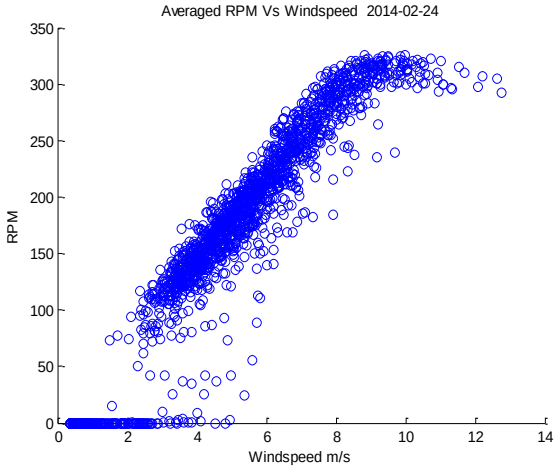


Figure 5-14. 30 sec averaged RPM vs. wind speed for 02-24-2014

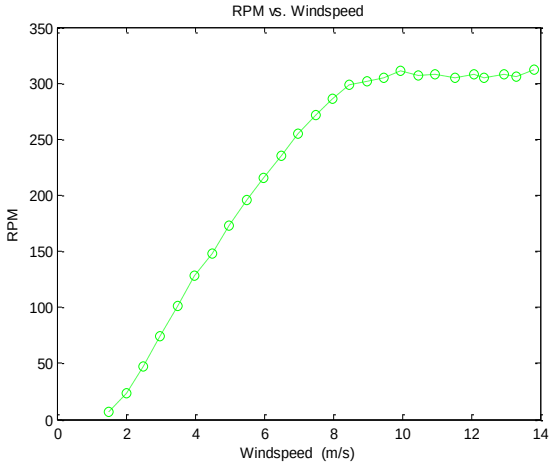


Figure 5-15.60 sec averaged RPM vs. wind speed for April 2014

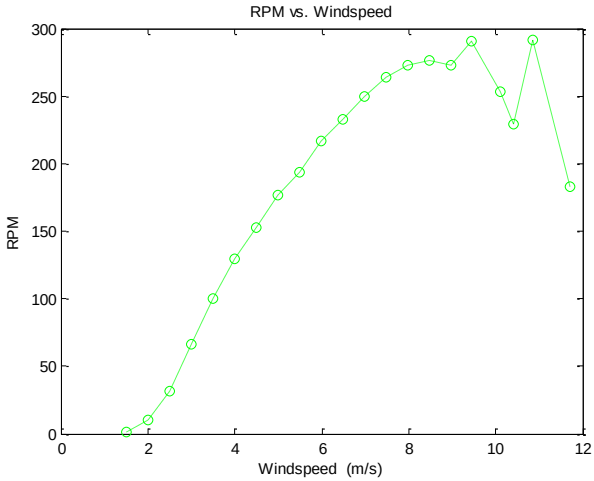


Figure 5-16. 10 min averaged RPM vs. wind speed for one year

Figure 5-14shows a very good response of RPM with wind speed. At very high wind speeds we slight curve and flattening in the RPM trend. This clearly shows that the RPM controller in the wind turbine reduces the RPM at high wind speeds. The same trend can be seen in the monthly data of April, 2014 in Figure 5-15. A closer look at Figure 5-15 shows that there is a dip in the RPM value slightly above 8 m/s, and then the RPM picks up again and becomes constant with a rise in wind speed. This can tell us that the controller steps in when the tip speed ratio goes above 7, which would correspond as an example, to a wind speed of 8.5 m/s and RPM above 310. The same trend in the wind speed and RPM can be seen in Figure 5-16 which is from the one year data. The effect of the controller can be more prominently seen in this figure. The dip and rise in the RPM value with increasing wind speeds above 8 m/s all correspond to the controller trying to maintain the tip speed ratio at 7.

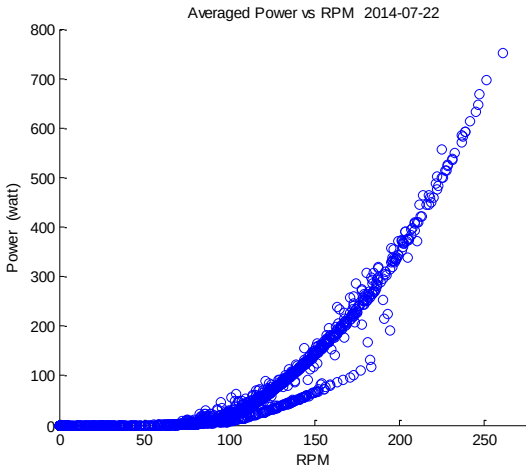


Figure 5-17. Power vs. RPM for 07-22-2014

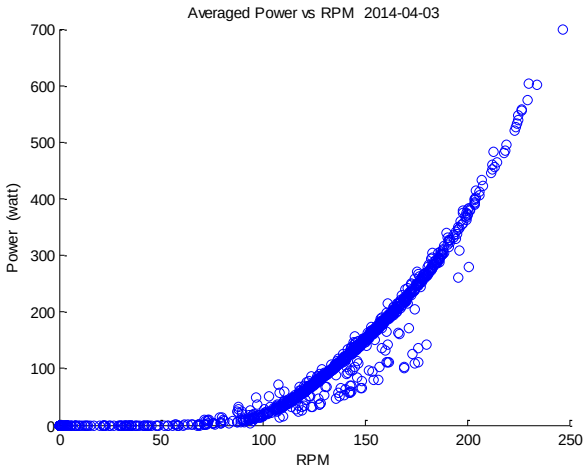


Figure 5-18. Power vs. RPM for 04-03-2014

In Figure 5-17and Figure 5-18 we can see a slight offshoot in the curve from 100 to 200 RPM where the power generation is lower than the main curve. With a closer look at the RPM vs. wind speed plots we can see that sometimes even at lower wind speeds of 2-3 m/s, the RPM is quite high. This would result in high tip speed ratio. From most of our Power vs. Tip speed ratio

plots we see that the maximum Tip speed ratio is 7. The higher values of TSR could be the reason for the controller to reduce the power and cause the off shoot in the Power vs. RPM plot.

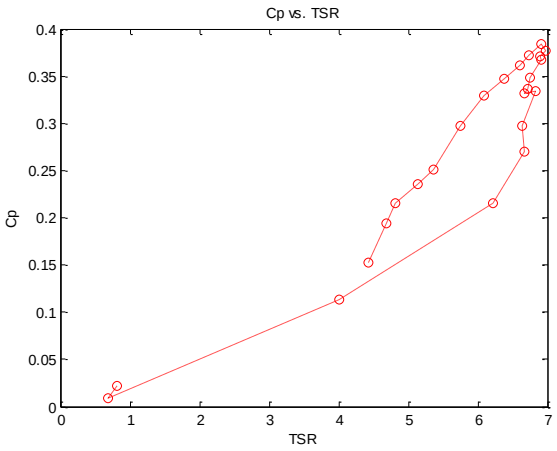


Figure 5-19. 30 sec averaged Cp vs. TSR 02-24-2014

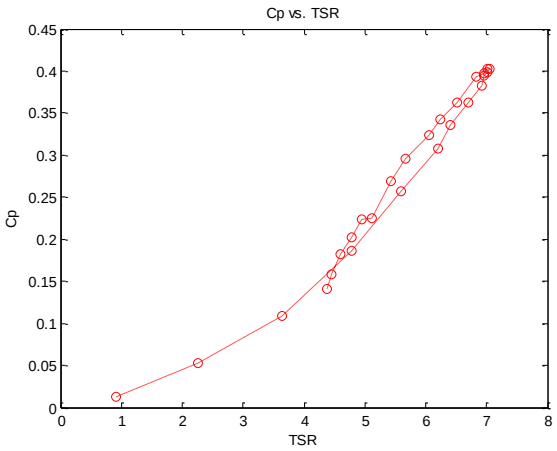


Figure 5-20. 60 sec averaged Cp vs. TSR for April 2014

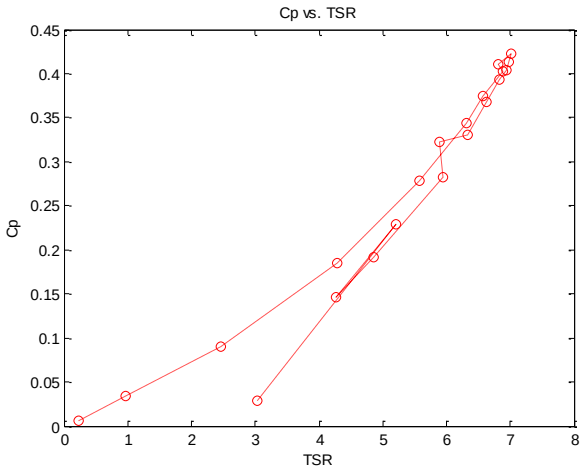


Figure 5-21. 10 minute averaged Cp vs. TSR for one year

The plots above show the variation of the power coefficient with the Tip speed ratio. Both the parameters are non-dimensional parameters which help us understand the performance better since it helps in comparing with all sizes of wind turbines also irrespective of the wind velocities that they experience.

Figure 5-19 is a good representation of Cp vs. TSR for one day. This day was a windy day with wind speeds ranging from 2 m/s -12 m/s, which helps us see a whole range of Cp values with varying TSR. It is interesting to notice that the controller brings down the tip speed ratio after reaching a maximum value of 7. The other two plots, Figure 5-20 and Figure 5-21 show the same phenomena where the TSR value is controlled to be under 7. The nature of the curve after that depends on the behavior of the controller. Further observation shows that the controller tries to maintain the same path in the return values of Cp after a TSR of 7. We however see a slight increase in the maximum Cp value between Figure 5-19, Figure 5-20 and Figure 5-21. This phenomena is not because the power produced is high, but because averaging the value of velocity yields higher Cp values, than instantaneous Cp. i.e. When the velocity is averaged for a certain period of time, the averaged value we get is lesser than the instantaneous value of wind speed, because the higher values of wind speed gets averaged out. The cubic value of this averaged wind speed gives a smaller velocity value (in the denominator) that makes a significant difference in the value of Cp.

$$Cp = \frac{P}{\frac{1}{2}\rho A V^3}$$

We can see this effect better in the plots below.

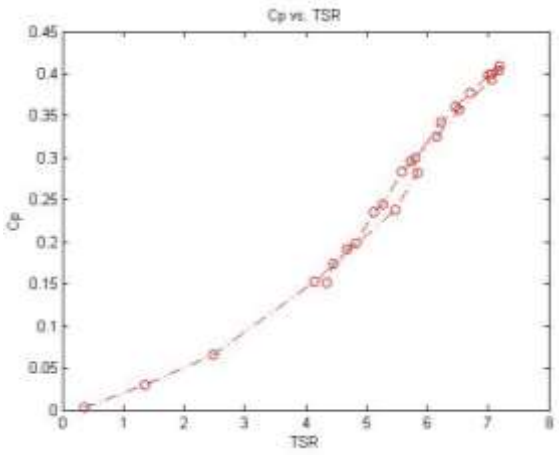


Figure 5-22. 60 second averaged Cp vs. TSR for December 2013

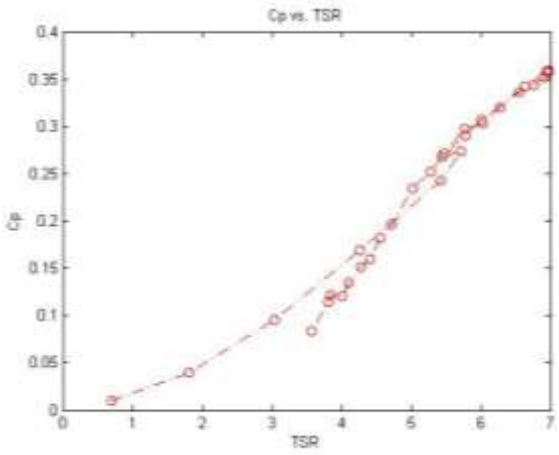


Figure 5-23. 3 second (not averaged) Cp vs. TSR for December 2013

Figure 5-22 is the 60 second averaged value for Cp vs. TSR for the month of December 2013 and Figure 5-23 shows the plot for the same month with data that is not averaged. The latter plot has a lower value of Cp, since the velocity data is not averaged. Figure 5-23 also extends to much lower Cp values because the data is not averaged out. We therefore prefer no averaging or smaller averaging intervals. Although for large amounts of data like one year, smaller averaging intervals make processing more difficult and therefore large averaging intervals are selected.

It is further important to note that the anemometer is placed at a height lower than the hub height. The recorded wind speed is therefore lower than the actual wind speed, which when considered makes a big difference in the calculated value of Cp.

The Skystream is well suited to produce power even at low wind speeds, so it can be placed at any location with moderate wind speed and some open space. Figure 5-24 shows an aerial view of the location of the Skystream at Penn State.

Figure 5-24. Map showing the position of the Skystream wind turbine at Penn State

Chapter 6
Experimental and Computational Comparisons

The purpose of this study is to understand the performance of the Skystream wind turbine. We have looked in detail into the experimental data from the wind turbine on the University Park campus at Penn State. We also analyzed what the performance of the wind turbine should be, by computing aerodynamic data acquired from measuring the wind turbine blade characteristics. The next step is to compare the two results and understand the reasons for variation in results.

Since performance is one of the main reasons for comparison we compared the dimensional and non-dimensional values of power.

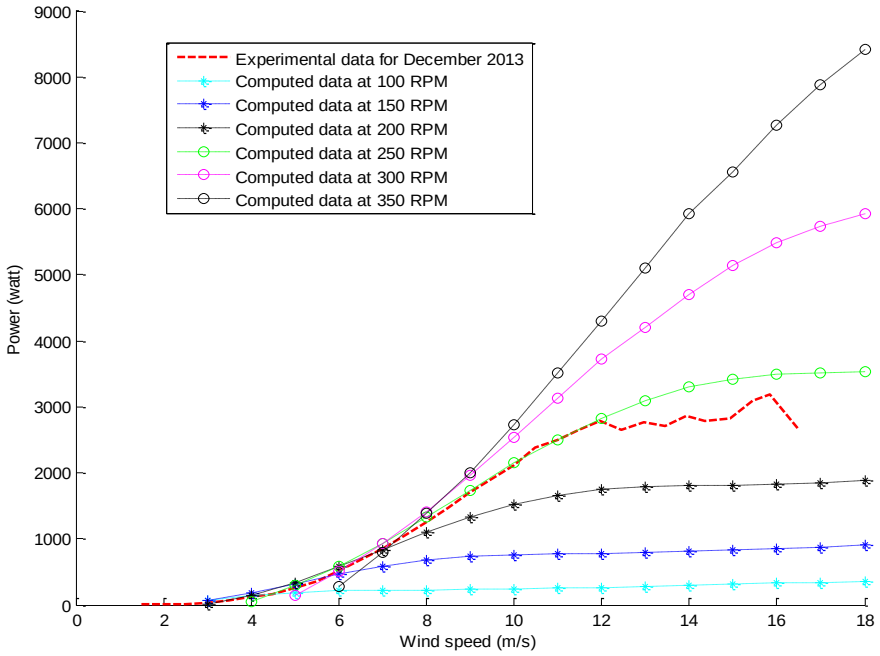


Figure 6-1. Comparison of Power vs. Wind Speed for the month of December 2013 with computed data

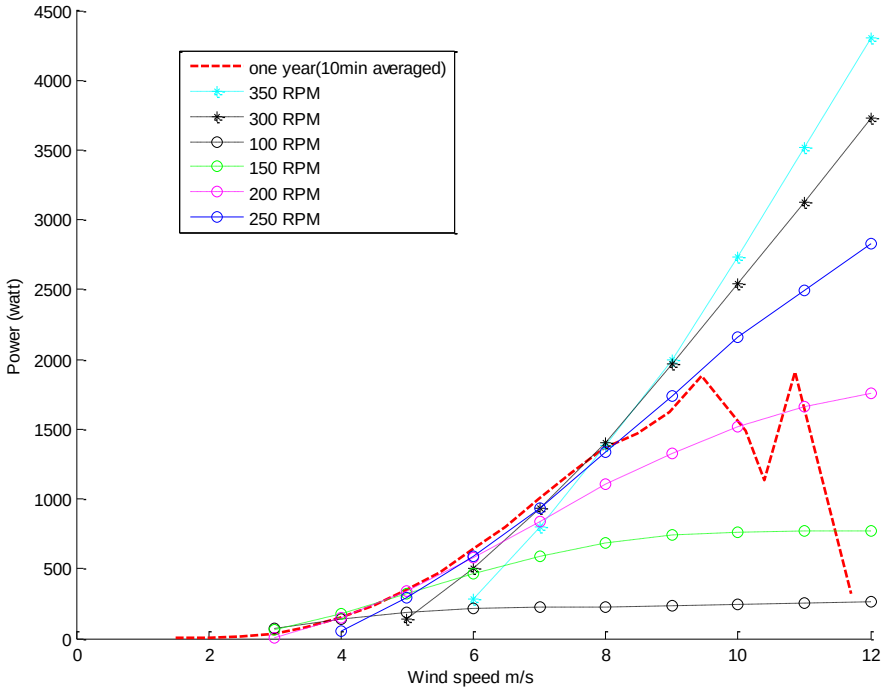


Figure 6-2. Comparison of Power vs. Wind Speed for one year (10 min averaged) with computed data

The first thing to notice when looking at Figure 6-1 and Figure 6-2 is that the experimental power data from December extends to higher wind speeds as compared to the one year data. This is because the power data in the one year comparison plot is 10 minute averaged data whereas the data for December is not averaged which means while averaging the data, higher values get averaged out.

The experimental and computed data for both the above plots are following the same trend until approximately 9 m/s after which the controller steps in to maintain a tip speed ratio of 7, because of which we see a change in the trend from the computed data.

A closer look at the two plots also show that the experimental data closely follows the computed curve for 250 RPM.

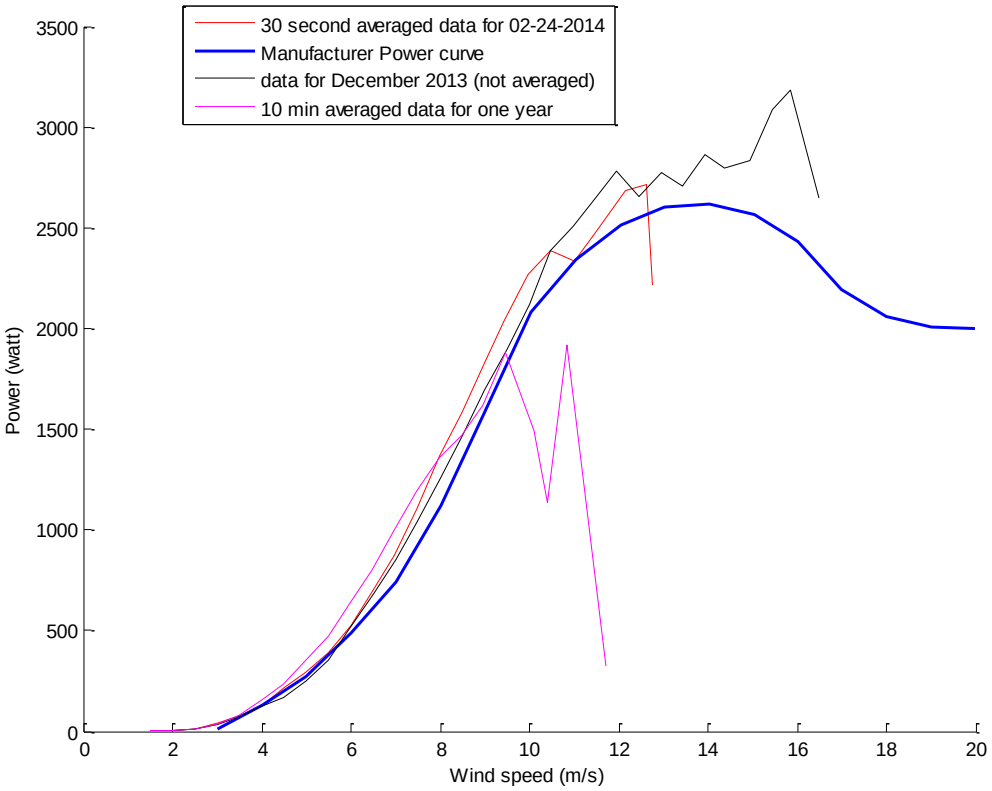


Figure 6-3. Experimental data compared with the manufacturer's power curve

Figure 6-3 shows that there is a close match between the manufacture's curve and the curves obtained from acquired data from the Penn State Skystream wind turbine from startup to about 9 m/s. Comparing December 2013 data, data from 24th February 2014 with the manufacturer data we see the closest correlation until 10 m/s after which the curve's irregularity is due to the controller maintaining the RPM to keep the tip speed ratio below 7.

However comparing the manufacturer's data with the one year's recorded data we see that the nature of the plot changes sharply after 9m/s. This could be due to various factors, like on

certain days the controller may have had some issues due to which power generated was low and due to averaging, the higher values of power on other days may have been averaged out.

The manufacturer's power curve gives an estimate of what the power performance should be like, however field test experimental data from the wind turbine, is more realistic since it has several other external factors that come into play. The nature of these differences in the power curves provides an opportunity to be explored in future work.

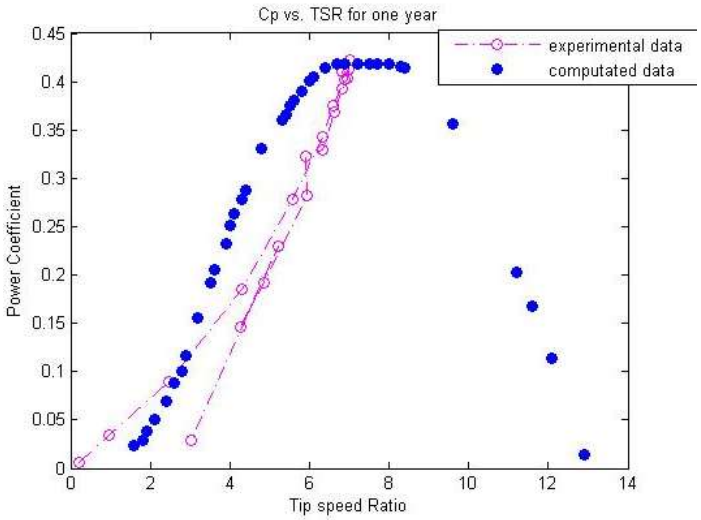


Figure 6-4 Cp vs. TSR comparison of computed data with one year(10 min averaged) experimental data

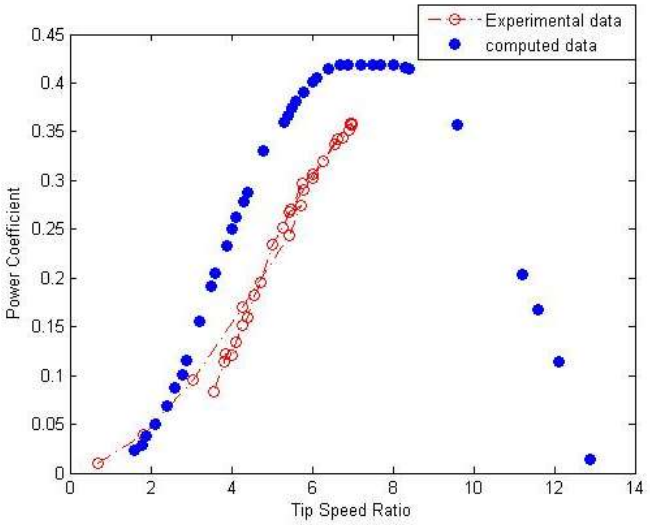


Figure 6-5. Cp vs. TSR comparison of computed data with data from December 2013

Figure 6-4 shows us the comparison of the experimental data of one year with the computed data. Although the curves do not overlay each other we see that up until a tip speed ratio of 7 the curves have a similar slope with a difference in Cp values of approximately 0.05-0.1 for a certain value of tip speed ratio. This difference in Cp value would be lower (i.e. the curves would be closer) if the generator efficiency was considered for the computed data. Figure 6-5 shows a similar data trend, however the maximum Cp value doesn't rise up to the computed maximum Cp as seen in Figure 6-4. This can be explained by referring to (page 66) where it is

explained that the calculated value of Cp becomes higher because we average the values of velocity. Therefore Figure 6-5 would be a more accurate representation of the Cp vs. TSR comparison plot. Another thing we notice from the above two plots is that the experimental curve turns back after reaching maximum Cp because the controller doesn't allow a tip speed ratio above 7. The experimental plots are made from binned data. The binned data is put into 0.5 m/s bins of wind speed. Therefore at high wind speeds and low tip speed ratios we see a return in the Cp curve.

However while comparing the two plots we must keep in mind that the computed data is calculated for steady wind conditions, whereas the experimental data from the Penn State Skystream does not operate at steady winds.

The computed data also assumes that the wind turbine is at sea level whereas the Skystream at Penn State is located at approximately 370 m above sea level. The figure below shows the comparison between power coefficient normalized to sea level and Cp recorded from wind turbine. The processed data was not normalized to sea level since we see a very small change in Cp value.

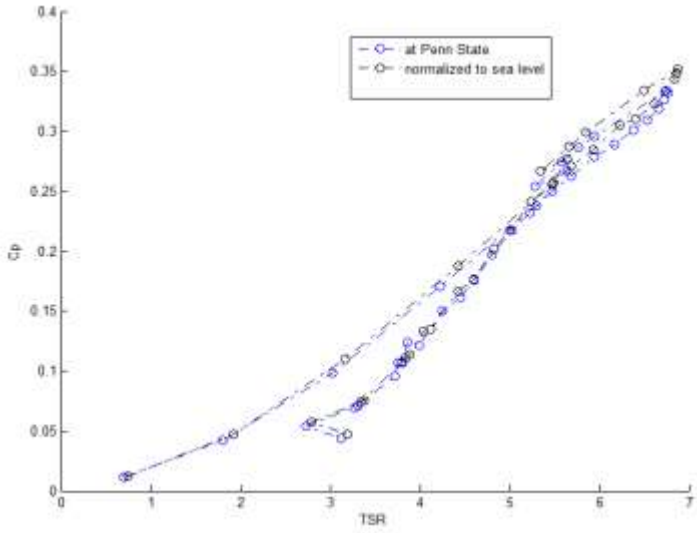


Figure 6-6. Comparison in Cp between data normalized to sea level and data at Penn State

We also hypothesize that the velocities recorded may not be accurate since the anemometer is not placed at hub height nor is it upstream from the turbine's location. The anemometer is placed 3.5 m below the hub height in the plane of rotation which could mean that the real velocities that the blade faces may be much higher.

Conclusion

This study analyzes the experimental as well as computational performance of the Skystream 3.7 wind turbine. The computed data not only helps us to predict the performance of the wind turbine but also helps us understand how the various aerodynamics loads effect the blade along its radius. After taking various measurements of airfoil profiles of the blade it was concluded that the S823 is the airfoil used at the root and S822 is used at the tip of the Skystream blade. Computing the rotor performance data using WT_Perf and plotting the different aerodynamic characteristics, it can be clearly seen that for the Skystream 3.7, the power coefficient is best between tip speed ratios of approximately 5 to 7. This explains why the Skystream has a controller that keeps the tip speed ratio under 7 at all times.

Analyzing the experimental data performance of the wind turbine using meteorological data versus time showed that the data being collected is accurate. This helped us validate that the data acquisition system was correctly gathering data. The experimental data analysis done for one day shows that the cut in wind speed for the wind turbine is 3 m/s . The data shows that at higher wind speeds the tip speed controller of the wind turbine starts to actuate which results in an irregularity in the amount of power being generated. We can conclude from the experimental data that the controller controls the tip speed by controlling the RPM at higher wind speeds. The variation in results is noticeable when comparing plots for one day, one month and one year data. This is because the nature of the wind is different at all times, while some days may be windy others may not be so. There are also differences in the daily, monthly and yearly data plots that could be because of the different averaging intervals used in the data processing. We specially see the difference in the power coefficient vs. tip speed ratio plots, where the maximum *Cp* value is higher for data plotted for a 60 second averaged interval as compared to the plot where data is not

averaged. For future work it would be interesting to do all data processing without averaging data or through weighted averages, so that more accurate results can be obtained.

Comparing the experimental and predicted data also shows some very exciting conclusions. Plotted power vs. wind speed data shows a very strong correlation between manufacturer's data and the experimental data from the Skystream. The curves closely follow the manufacturer's curve until a wind speed of 10 m/s after which the curve becomes erratic. The irregularity in the curve can be attributed to the controller stepping in to keep the tip speed ratio below 7. Comparison between the experimental data and computed data for *Cp* vs. TSR also follow a similar slope. The computed power coefficient values are higher as compared to the experimental data. We hypothesize that this difference could be because the computed data is measured at sea level whereas the skystream experimental data we have is from State College, Pennsylvania, which is 370 m above sea level. Another reason for this difference could be because the computed data is calculated for steady wind conditions, whereas the experimental data from the Penn State Skystream does not operate at steady winds.

Further studies on experimental data conducted by isolating periods of steady wind could give better results. In our study we assume that the wind speed that the anemometer records is the true wind speed at the hub height. It would be very educational to study the true nature of the wind speed upstream of the plane of rotation at the hub height and it would also help us get more accurate performance results.

Bibliography

1.Gsänger, S., & Pitteloud, J. (March 2014). "Small wind world report". Retrieved Sep 11,2014 from *'http://small-wind.org/wp-content/uploads/2014/03/2014_SWWR_summary_web.pdf'*

2. Chiras, Dan. "Power From the Wind" Retrieved Nov 22, 2014 from *'http://www.newsociety.com/var/storage/blurbs/9780865716179_intro.pdf'*

3. Sanderse, B. "Aerodynamics of wind turbine wakes." *Energy Research Center of the Netherlands (ECN), ECN-E–09-016, Petten, The Netherlands, Tech. Rep(2009).*

4. McCosker, John. *Design and Optimization of a Small Wind Turbine*. Diss. Rensselaer Polytechnic Institute, 2012.

5. Kulunk, Emrah. "Aerodynamics of wind turbines." *Fundamental and Advanced Topics in Wind Power* (1970).

6. Bianchi, Fernando D., Hernán De Battista, and Ricardo J. Mantz. "Robust Multivariable Gain-Scheduled Control of Wind Turbines for Variable Power Production." *International Journal of Systems Control* 1.3 (2010).

7. North Carolina Wind Energy. Applachian State University, Web http://wind.appstate.edu/programs/research-demonstration

8. Sprague, J., Huff, S., Solomon, K., & Waggy, M. Research and Development of Small Wind Turbine Blades. http://researchgroups.msu.edu/system/files/content/DevelopmentofSmallWindTurbineBlades.pdf.

9. "*Optimum Blade Numbers and Solidities for Small HAWTs*," Brown, M. M., and Visser, K.D., AIAA-2007-1370, 45th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV., January 2007.

10. Mendoza, I., and J. Hur. "Power Performance Test Report for the SWIFT Wind Turbine." *Contract* 303 (2012): 275-3000.

11. Kylie Flickinger (2013), "*Small Scale Horizontal Axis Wind Turbine Testing and Predictions*" Master's Thesis, The Pennsylvania State University, University Park

12. Brian Wallace(2011) , "*Development and validation of a wind generator field testing methodology"* Master's Thesis, The Pennsylvania State University, University Park

13. Platt, A. D., and M. L. Buhl Jr. *WT_Perf User guide for version 3.05. 00*. NREL, Tech. Report, 2012.

14. A. Jain, D. Chen, I. Patel and M. Washington; "*Skystream Wind Turbine Upgrades and Experiments*". Penn State , Report 2012

15. *"Power performance measurement of the Skystream 3.7 at Kaiser-Wilhelm-Koog, Germany according to IEC 61400-12-1 and BWEA"* by WindTest Inc. at Kaiser-Wilhelm-Koog, 2009-11-02

Appendix A

Guide to using Xfoil and WT_Perf

The document (created by Kylie Flickinger) below explains how to make an airfoil file and how to run XFoil:

1. Using a digitizing code find the x and y coordinates of your profile shape and copy them into excel
    a. Be sure to get many points on the leading edge as well as trailing edge.
    b. Also, be careful to be precise and attempt to get the profile as smooth as possible
2. Non-dimensionalize the data by the chord length of your profile shape.
3. Be consistent and keep only 5 decimal places for **ALL** x-y coordinates
4. Order your data of the Upper surface so that it starts at (1,0) this should be the location of your trailing edge point and goes to (0,0) the location of your leading edge point*
5. Order your data of the Lower Surface so that it starts at (0,0) and goes to (1,0)*
6. Combine the upper and lower surface data so that it goes from upper to lower (1,0)-(0,0)-(1,0)
7. You could also order it so that it goes down the lower surface first but it must start at the trailing edge go to the leading edge and back to the trailing. Direction doesn't matter
8. Open a new notepad file
9. Enter a one line header. This can be anything as long as it does not begin with a Fortran-readable pair of numbers so a name like "NACA 4412" will work but "44 12 NACA" will not. Also, there seems to be some problems with starting the name with T or F
10. Copy correctly ordered and non-dimensionalized data directly below the header. The x and y data should be separated by a tab and three spaces.
11. Save the notepad file as a .dat file and exit out of input file. Make sure this file is located in the same directory as x-foil
12. Open X-foil
13. Type <load> and hit enter
14. Type in file name of .dat file with the .dat extension
15. If warnings about excessive angles , which is caused by too few or poorly spaced points, type <pane> this will allow you to increase point density in areas of high curvature.
16. type in <oper> to change from the home menu to the operational menu

17. In order to toggle viscous mode type <visc> and hit enter. You will then be prompted to give a Reynolds number. Type desired Reynolds number and hit enter. You may use scientific notation if desired (ex. 10E6).
18. In order to save data you calculate from here on out you will need to turn on the point accumulation. Do this by typing <pacc> and hit enter. This will prompt you for a file name to save your polar information to. Pick a descriptive name and add <.txt> to the end. Hit enter. It will also ask you for a dump file name. This is unnecessary and you may just hit enter.
19. In order to accumulate polars type <aseq> and hit enter. It will then prompt you for the lower value of the desired angle of attack range, upper value of the angle of attack range and the degree increments. Type in angles in degree and hit enter each time.
20. Your polars will then be calculated. If your values do not converge you may not have a smooth enough profile shape still and will need to go back to the beginning and try to redo your digitizing of the profile

Image of an output file from XFoil:

XFOIL Version 6.94

Calculated polar for: Skystream Station 2

1 1 Reynolds number fixed Mach number fixed

xtrf = 1.000 (top) 1.000 (bottom)
Mach = 0.000 Re = 0.250 e 6 Ncrit = 9.000

alpha	CL	CD	CDp	CM	Top_Xtr	Bot_Xtr
-30.000	-1.0551	0.23755	0.23276	0.0445	1.0000	0.0223
-29.500	-1.0225	0.23908	0.23442	0.0432	1.0000	0.0234
-29.000	-1.0105	0.23504	0.23028	0.0390	1.0000	0.0236
-28.500	-1.0047	0.22958	0.22471	0.0340	1.0000	0.0250
-28.000	-0.9941	0.22499	0.22000	0.0297	1.0000	0.0253
-27.500	-0.9739	0.22272	0.21768	0.0266	1.0000	0.0260
-27.000	-0.9387	0.22400	0.21909	0.0255	1.0000	0.0276
-26.500	-0.9155	0.22203	0.21709	0.0228	1.0000	0.0279
-26.000	-0.8888	0.22076	0.21580	0.0204	1.0000	0.0280
-25.500	-0.8613	0.21957	0.21460	0.0182	1.0000	0.0281
-25.000	-0.8331	0.21847	0.21350	0.0160	1.0000	0.0282
-24.500	-0.8040	0.21753	0.21258	0.0139	1.0000	0.0284
-24.000	-0.7755	0.21639	0.21145	0.0118	1.0000	0.0285
-23.500	-0.7476	0.21503	0.21010	0.0096	1.0000	0.0287
-23.000	-0.7221	0.21302	0.20808	0.0072	1.0000	0.0290
-22.500	-0.6968	0.21092	0.20601	0.0048	1.0000	0.0293
-22.000	-0.6743	0.20806	0.20317	0.0021	1.0000	0.0296
-21.500	-0.6552	0.20427	0.19939	-0.0008	1.0000	0.0301
-21.000	-0.6384	0.19985	0.19497	-0.0039	1.0000	0.0306
-20.500	-0.6233	0.19498	0.19008	-0.0070	1.0000	0.0313
-19.500	-0.5937	0.18526	0.18066	-0.0131	1.0000	0.0457
-19.000	-0.5695	0.18247	0.17798	-0.0152	1.0000	0.0625
-18.500	-0.5435	0.18005	0.17561	-0.0171	1.0000	0.0688
-18.000	-0.5221	0.17652	0.17207	-0.0192	1.0000	0.0752
-17.500	-0.5366	0.16478	0.16023	-0.0249	1.0000	0.0875
-17.000	-0.5292	0.15851	0.15391	-0.0284	1.0000	0.0939
-16.500	-0.4830	0.16095	0.15640	-0.0278	1.0000	0.0978
-16.000	-0.4552	0.15935	0.15480	-0.0289	1.0000	0.1032
-15.500	-0.4336	0.15651	0.15196	-0.0304	1.0000	0.1101
-13.500	-0.8443	0.05910	0.05295	-0.0724	1.0000	0.1042
-13.000	-0.8555	0.05253	0.04622	-0.0750	1.0000	0.1061
-12.500	-0.8730	0.04608	0.03950	-0.0772	1.0000	0.1085
-12.000	-0.8751	0.04199	0.03545	-0.0778	1.0000	0.1113
-11.500	-0.8804	0.03820	0.03157	-0.0775	1.0000	0.1136
-11.000	-0.8978	0.03474	0.02797	-0.0752	1.0000	0.1156
-10.500	-0.9114	0.03202	0.02509	-0.0733	0.9958	0.1171
-10.000	-0.8485	0.03000	0.02309	-0.0820	0.9809	0.1211
-9.500	-0.7875	0.02801	0.02083	-0.0891	0.9661	0.1243
-9.000	-0.7242	0.02652	0.01931	-0.0939	0.9521	0.1282
-8.500	-0.6629	0.02543	0.01812	-0.0976	0.9373	0.1323

WT_Perf input with airfoil lift and drag information:

Below is just a snipped image of the file. But the lift and drag data for each Reynolds number goes from -180 to +180 degree angle of attack.

```
AeroDyn airfoil file.  Compatible with AeroDyn v13.0.
p2ClCd
FROM Anagha Ray aug 2014,aspect ratio=11
      4          Number of airfoil tables in this file
      0.1        Reynolds number in millions.  For efficiency, make very large if only one table.
      0.0        Control setting
     16.00      Stall angle (deg)
     -1.8263    Zero Cn angle of attack (deg)
      5.8575    Cn slope for zero lift (dimensionless)
      1.8224    Cn extrapolated to value at positive stall angle of attack
     -0.4272    Cn at stall value for negative angle of attack
      5.25      Angle of attack for minimum CD (deg)
      0.0215    Minimum CD value
-180.00    0.000   0.0924
-170.00    0.285   0.1304
-160.00    0.569   0.2398
-150.00    0.556   0.4070
-140.00    0.548   0.6112
-130.00    0.508   0.8269
-120.00    0.427   1.0272
-110.00    0.308   1.1866
-100.00    0.160   1.2846
 -90.00    0.000   1.3080
 -80.00   -0.160   1.2846
 -70.00   -0.308   1.1866
 -60.00   -0.427   1.0272
 -50.00   -0.508   0.8269
 -40.00   -0.548   0.6112
 -30.00   -0.556   0.4070
 -20.00   -0.569   0.2398
 -19.75   -0.531   0.2141
 -19.50   -0.512   0.2120
 -19.25   -0.501   0.2104
 -19.00   -0.502   0.2094
 -18.50   -0.474   0.2043
 -18.25   -0.461   0.2024
 -18.00   -0.453   0.2005
 -17.75   -0.459   0.2000
 -17.50   -0.444   0.1965
 -17.25   -0.427   0.1938
 -17.00   -0.414   0.1919
 -16.75   -0.409   0.1903
 -16.25   -0.391   0.1854
 -16.00   -0.377   0.1832
 -15.75   -0.371   0.1816
 -15.25   -0.364   0.1770
 -15.00   -0.348   0.1740
 -14.75   -0.335   0.1716
 -14.50   -0.326   0.1695
 -14.25   -0.368   0.1735
 -14.00   -0.337   0.1669
 -13.75   -0.315   0.1627
```

WT_Perf Input file

```
-----  WT_Perf Input File  ----------------------------------------
------------
WT_Perf skystream (Date 11/14/14) input file.  3 turbine (Non-dimen,
English, Space, BEM).
Compatible with WT_Perf v3.05.00a-adp.
-----  Input Configuration  ---------------------------------------
------------
```

```
False                Echo:                      Echo input
parameters to "<rootname>.ech"?
False                DimenInp:                  Turbine parameters
are dimensional?
True                 Metric:                    Turbine parameters
are Metric (MKS vs FPS)?
-----  Model Configuration  ----------------------------------------
------------
1                    NumSect:                   Number of
circumferential sectors.
5000                   MaxIter:                   Max number of
iterations for Newton's method to find induction factor.
25                   NSplit:                    Max number of splits
for binary search method
1.0e-5               ATol:                      Error tolerance for
induction iteration.
1.0e-5               SWTol:                     Error tolerance for
skewed-wake iteration.
-----  Algorithm Configuration  -----------------------------------
------------
True                 TipLoss:                   Use the Prandtl tip-
loss model?
True                 HubLoss:                   Use the Prandtl hub-
loss model?
True                 Swirl:                     Include Swirl
effects?
False                 SkewWake:                  Apply skewed-wake
correction?
True                 IndType:                   Use BEM induction
algorithm?
False                AIDrag:                    Use the drag term in
the axial induction calculation?
False                TIDrag:                    Use the drag term in
the tangential induction calculation?
True                 TISingularity:             Use the singularity
avoidance method in the tangential-induction calculation?
false                DAWT:                      Run Diffuser
Augmented Water Turbine Analysis? [feature not implimented yet]
False                Cavitation:                Run cavitation
check? if cavitation, output sevens, check 12 oclock azimuth
-----  Cavitation Model  ------------------------------------------
------------
101325               PressAtm:                  Air Atmospheric
Pressure, Pa units, absolute
2300                 PressVapor:                Vapor Pressure of
Water, Pa units, absolute
1.0                  CavSF:                     Cavitation safety
factor
0                WatDepth:                  Depth from water free
surface to mudline (tower base)
-----  Turbine Data  ----------------------------------------------
------------
```

```
3                    NumBlade:                  Number of blades.
1.8415                 RotorRad:                  Rotor radius
[length].
0.1624                  HubRad:                    Hub radius
[length or div by radius].
0.0                  PreCone:                   Precone angle,
positive downwind [deg].
0.0                  Tilt:                      Shaft tilt [deg].
0.0                 Yaw:                       Yaw error [deg].
11.586               HubHt:                     Hub height [length
or div by radius].
25                   NumSeg:                    Number of blade
segments (entire rotor radius).
   RElm      Twist      Chord  AFfile  PrntElem
  0.17915    18.292     0.1470     1      True
  0.21266    16.423     0.1406     1      True
  0.24616    14.554     0.1343     1      True
  0.27966    12.782     0.1278     1      True
  0.31317    11.218     0.1210     2      True
  0.34667     9.654     0.1141     2      True
  0.38018     8.387     0.1080     2      True
  0.41368     7.286     0.1024     2      True
  0.44718     6.258     0.0970     2      True
  0.48069     5.581     0.0922     2      True
  0.51419     4.904     0.0875     2      True
  0.54770     4.400     0.0832     2      True
  0.58120     3.968     0.0790     2      True
  0.61470     3.530     0.0752     2      True
  0.64821     3.090     0.0715     3      True
  0.68171     2.697     0.0681     3      True
  0.71522     2.531     0.0657     3      True
  0.74872     2.365     0.0633     3      True
  0.78222     2.059     0.0608     3      True
  0.81573     1.739     0.0584     3      True
  0.84923     1.472     0.0556     3      True
  0.88274     1.273     0.0525     3      True
  0.91624     0.999     0.0493     3      True
  0.94974     0.605     0.0460     3      True
  0.98325     0.202     0.0409     3      t True
-----  Aerodynamic Data  ----------------------------------------
------------
1.225                                Rho:                Air
density [mass/volume].
0.00001464                                KinVisc:
Kinematic air viscosity
0                                    ShearExp:           Wind shear
exponent (1/7 law = 0.143).
False                                     UseCm:              Are
Cm data included in the airfoil tables?
False                                  UseCpmin:           Are
Cp,min data included in the airfoil tables?
```

```
3                                       NumAF:              Number
of airfoil files.
"Airfoils/WindLite/p2ClCd.dat"  AF_File:            List of NumAF
airfoil files.
"Airfoils/WindLite/p5ClCd(1).dat"
"Airfoils/WindLite/p7ClCd.dat"
-----  I/O Settings  ------------------------------------------------
------------
False                UnfPower:                  Write parametric
power to an unformatted file?
True                 TabDel:                    When generating
formatted output (OutForm=True), make output tab-delimited (fixed-
width otherwise).
1                    ConvFlag:                  For non-converging
cases, 0 to output the result, 1 to output nines, 2 to output NaN
(safest).
True                 Beep:                      Beep on exit.
True                 KFact:                     Output dimensional
parameters in K (e.g., kN instead on N)
True                 WriteBED:                  Write out blade
element data to "<try5>.bed"?
False                InputTSR:                  Input speeds as
TSRs?
True                 OutMaxCp:                  Output conditions
for the maximum Cp?
"mps"                SpdUnits:                  Wind-speed units
(mps, fps, mph).
-----  Combined-Case Analysis  --------------------------------------
------------
0                    NumCases:                  Number of cases to
run.  Enter zero for parametric analysis.
WS or TSR   RotSpd   Pitch                      Remove following
block of lines if NumCases is zero.
-----  Parametric Analysis (Ignored if NumCases > 0)  ---------------
------------
3                    ParRow:                    Row parameter    (1-
rpm, 2-pitch, 3-tsr/speed).
2                    ParCol:                    Column parameter (1-
rpm, 2-pitch, 3-tsr/speed).
1                    ParTab:                    Table parameter  (1-
rpm, 2-pitch, 3-tsr/speed).
True                 OutPwr:                    Request output of
rotor power for formatted output?
True                 OutCp:                     Request output of Cp
for formatted output?
True                 OutTrq:                    Request output of
shaft torque for formatted output?
True                 OutFlp:                    Request output of
flap bending moment for formatted output?
True                 OutThr:                    Request output of
rotor thrust for formatted output?
```

```
0, 0, 1              PitSt, PitEnd, PitDel:     First, last, delta
blade pitch (deg).
100, 350, 50            OmgSt, OmgEnd, OmgDel:     First, last,
delta rotor speed (rpm).
3, 18, 1            SpdSt, SpdEnd, SpdDel:     First, last, delta
speeds.
```

There are two different WT_Perf Output files, one is with an extension .oup and the other is with an extension .bed. Below are the images of the output files since they are too big to include the whole file.

Sample .oup output file from WT_Perf

```
Results generated by WT_Perf (v3.05.00a-adp, 09-Nov-2012) for input file "try10.wtp".
Generated on 14-Nov-2014 at 18:49:07.
Input file title:
  WT_Perf skystream (Date 11/14/14) input file.  3 turbine (Non-dimen, English, Space, BEM).


================================================================================

Conditions leading to the maximum Cp:
  WndSp =   3.000 mps
  Pitch =   0.000 deg
  MaxCp =   0.414

--------------------------------------------------------------------------------
Power (kW) for Omega = 100 rpm.

WndSp   Pitch (deg)
(mps)     0.000
  3.000       0.073
  4.000       0.138
  5.000       0.190
  6.000       0.219
  7.000       0.225
  8.000       0.229
  9.000       0.236
 10.000       0.246
 11.000       0.256
 12.000       0.268
 13.000       0.281
 14.000       0.295
 15.000       0.310
 16.000       0.326
 17.000       0.343
 18.000       0.361

--------------------------------------------------------------------------------
Cp (-) for Omega = 100 rpm.

WndSp   Pitch (deg)
(mps)     0.000
  3.000  0.413604
  4.000  0.330639
  5.000  0.233087
  6.000  0.155521
  7.000  0.100721
  8.000  0.068683
  9.000  0.049685
 10.000  0.037657
 11.000  0.029497
 12.000  0.023765
 13.000  0.019593
```

Sample .bed output file from WT_Perf

This output file contains the aerodynamic data for every section (element) of the blade for each wind speed and RPM. The data from file shown below is usually export into Microsoft Excel for a better look at the data.

Blade-element data generated by WT_Perf (v3.05.00a-adp, 09-Nov-2012) for input file "try10.wtp".
Generated on 14-Nov-2014 at 18:49:07.
Input file title:
  WT_Perf skystream (Date 11/14/14) input file. 3 turbine (Non-dimen, English, Space, BEM).

Blade-element data for:
100   rpm Rotation Rate
0   deg Blade Pitch
3   m/s Wind Speed

Element	RElm	IncidAng	Azimuth	Loc	Vel	Re	Loss	Axial Ind.
	Tang. Ind.	Airflow Angle	AlfaD	Cl	Cd	Cm	Cpmin	CavNum
	Cav Thrust Coef	Torque Coef	Power Coef		Thrust/Len		Torque/Len	
	Power Converge							
(-)	(m)	(deg)	(deg)	(m/s)	(millions)	(-)	(-)	(-)	(deg)	(deg)	(-)
	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(N/m)	(N)	(kW)	(-)
1	0.330	18.29	0.0	4.36	0.081	0.529					
	0.430	0.161	23.09	4.80	0.688	0.022					
	0.000	0.000	0.000	F	0.531	0.207					
0.238	2.022	0.260	0.001	T							
2	0.392	16.42	0.0	4.97	0.088	0.792					
	0.325	0.106	24.06	7.64	0.879	0.023					
	0.000	0.000	0.000	F	0.703	0.293					
0.400	3.179	0.518	0.001	T							
3	0.453	14.55	0.0	5.54	0.094	0.913					
	0.282	0.075	22.88	8.32	0.903	0.024					
	0.000	0.000	0.000	F	0.748	0.293					
0.463	3.912	0.694	0.001	T							
4	0.515	12.78	0.0	6.11	0.098	0.969					
	0.269	0.057	21.04	8.26	0.901	0.024					
	0.000	0.000	0.000	F	0.769	0.273					

Appendix B

Codes used for processing data

Parsing Code: created by Brian Wallace

```
oclear all;
close all;
clc;
format long e;

% This code reads in raw data files from the Skystream wind turbine and
% removes blank and invalid data lines from the file.  The code will
output
% a new file with a parsed designation in the file name.

% Author: Brian Wallace {bdw5003@psu.edu)
% Date: 6/7/2013
% Version: ParsingCode_v1.m

disp(' ');
disp('**********************************************************************
***');
disp('Skystream Wind Turbine Data Parsing');
disp('**********************************************************************
***');
disp(' ');
disp('For every input, be aware to type capital letters where you have
to...');
disp(' ');

%% Locate user specified raw data acquisition files and create
directory for parsed text files

[FileName,PathName,FilterIndex] = uigetfile('*.txt','Select raw data
acquisition files to be parsed','MultiSelect','on');

%Create Directory Folder and set output file names.
parsed_directory = uigetdir('../','Select Directory to Save Parsed
Data');

N = length(FileName);

cd(PathName);

%% Load raw data acquisition file i and write only lines which begin
with D to the new parsed text files.
```

```
for i=1:N

    fid = fopen(char(FileName(i)));

    cd(parsed_directory)
    fid2 = fopen(['parsed_',char(FileName(i))],'a');

    while ~feof(fid)

        line = fgets(fid);

        if line(1)==char('D')

            fwrite(fid2, line);

        end

    end

    % Close both the raw and parsed text files.
    fclose(fid);
    fclose(fid2);

    cd(PathName);

end
```

MATvariableCode (to convert text data file from skystream to matlab file):

Created by Brian Wallace

```
clear all;
close all;
clc;
format long e;

% This code reads in user specified parsed data files from the
skystream
% wind turbine and generates MATLAB .mat files which can be used during
% data analysis.  The output files contain the same information as the
% parsed data files only in a more user friendly and compact format.

% Author: Brian Wallace {bdw5003@psu.edu)edited Anagha Ray
% Date: 6/7/2013
% Version: MATvariableCode_v3.m

disp(' ');
disp('**********************************************************************
***');
disp('Skystream Wind Turbine Data Pre-processing');
```

```
disp('**********************************************************************
***');
disp(' ');
disp('For every input, be aware to type capital letters where you have
to...');
disp(' ');

%% Locate user specified parsed data acquisition files and create
directory for MATLAB variable files

%Prompt the user to select the parsed data acquistion files; note
%filenames and pathnames
[FileName,PathName,FilterIndex] = uigetfile('*.txt','Select parsed data
acquisition files to be processed','MultiSelect','on');

%Prompt user for location to save MATLAB variable files
proc_directory = uigetdir('../','Select Directory to Save MATLAB
Variable Data Files');

%Extract number of files from FileName vector
N = length(FileName);

disp('  ');
disp('STARTING TO READ DATA FILES ');
disp('  ');



%% Loop through number of files, extract lines of data, write data to
MATLAB array variable
for i=1:N

    disp(' ');
    disp(['File ',num2str(i),' of ',num2str(N)]);
    disp(' ');


    % Change working directory to location of parsed data files
    cd(PathName)

    % Open parsed data file with index i
    fid = fopen(char(FileName(i)));

    Z = textscan(fid, 'DECI: +STAT turb=%d vers=2.02 %d %f %f %f %f %f
%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %s %s %f %f
%f %f %f %d %d %f %f %f %f %f %f %f %s ', 'delimiter', ',');

    % Close the file once reading of data lines is complete
    fclose(fid);

    data.time = datenum(Z{44},'mm/dd/yyyy HH:MM:SS PM');
    data.datetime = datestr(data.time);%realtime with date
```

```
data.realtime = datevec(data.time);
data.serial=Z{1};%serial number
data.invtime=Z{2}; %Inv Time
data.energy=Z{3};%watt-hours
data.voltin=Z{4};%Voltage In
data.voltDCbus=Z{5};%Voltage DC Bus
data.voltsL1=Z{6};%Voltage L1
data.voltsL2=Z{7};%Voltage L2
data.voltrise=Z{8}; %Voltage rise
data.minVrpm=Z{9};%min v from rpm
data.current=Z{10};%Current out
data.Pout=Z{11};%Power out
data.Preg=Z{12};%Power Reg
data.Pmax=Z{13};%Power max
data.lineFreq=Z{14};%Line Freq
data.InvFreq=Z{15}; %Inv Freq
data.LineRes=Z{16};%Line Resistance
data.rpm=Z{17};%RPM
data.windref=Z{18};%wind ref
data.targTSR=Z{19};%Target TSR
data.rampRPM=Z{20};%Ramp RPM
data.bootpulsw=Z{21};%Boot pulswidth
data.maxBPW=Z{22}; %Max BPW
data.currentAmplitud=Z{23};%current amplitude
data.T1=Z{24};%T1
data.T2=Z{25};%T2
data.T3=Z{26};%T3
data.EvCount=Z{27};%Event Count
data.LastEVcode=Z{28};%Last Event Code
data.EventStatus = Z{29};%Event Status
data.EventValue = Z{30};%Event Value
data.TurbineStatus=Z{31};%Turbine Status
data.GridStatus=Z{32};%Grid Status
data.SystemStatus=Z{33}; %System Status
data.SlaveStatus=Z{34};%Slave Status
data.AccessStatus=Z{35};%Access Status
data.timer=Z{36};%Timer
data.wind1=Z{37};%wind1
data.wind2=Z{38};%wind2
data.wind3=Z{39};%wind3
data.winddir=Z{40};%wind dir
data.tempr=Z{41};%temp
data.pressure=Z{42};%baro. pressure
data.humidity=Z{43};%relative humidity

% Generate a name string for the MATLAB .mat file
output = ['swwt',datestr(data.time(1),'yyyymmdd')];

% Change directory to location where .mat files will be saved
cd(proc_directory)

% Save .mat files to output directory
save(output, 'data');
dlmwrite('myfile.txt',data.wind1,'delimiter', '\t','-append');
```

```
    save('myfile.txt');

    clear data;

end

disp('END OF CODE');
```

Averaging code: created by Brian Wallace

```
clear all;
close all;
clc;
format long e;

% This code reads in MATLAB .mat variable files and computes averages
on
% these data over a user specified time interval.  The code will output
% results to a MATLAB .mat variable file with a "average_Nsec"
designation
% in the file name.

% Author: Brian Wallace {bdw5003@psu.edu)
% Date: 6/7/2013
% Version: AveragingCode_v1.m

disp(' ');
disp('**********************************************************************
***');
disp('Skystream Wind Turbine N-Second Data Averaging');
disp('**********************************************************************
***');
disp(' ');
disp('For every input, be aware to type capital letters where you have
to...');
disp('CAUTION: Average Interval Input must be a multiple of 3 sec');
disp(' ');

[FileName,PathName,FilterIndex] = uigetfile('*.mat','Select matlab
variable files to be processed','MultiSelect','on');

average_interval = input('Enter data resolution (averaging) interval in
seconds: ');
ave_num = datenum(00,00,00,00,00,average_interval);

%Create Directory Folder and set output file names.
proc_directory = uigetdir('../','Select Directory to save Averaging
Summary File');

N = length(FileName);
```

```
disp('  ');
disp('STARTING TO AVERAGE THE DATA ');
disp('  ');

for i=1:N

    disp(' ');
    disp(['File ',num2str(i),' of ',num2str(N)]);
    disp(' ');

    cd(PathName);

    load(char(FileName(i)),'data');

    day_num = floor(data.time(1));

    time_interval = day_num:ave_num:day_num+1;

    num_time_ints = length(time_interval);

    num_pts = floor(average_interval / 3);

    for j=1:num_time_ints-1

        int = find(data.time >= day_num + (j-1)*ave_num & data.time <
ave_num*j + day_num);
        ave.count(j) = length(int);

        if ave.count(j) >= num_pts

            ave.time(j) = ave_num*j+day_num;
            ave.Preg(j) = mean(data.Preg(int));
            ave.Pout(j) = mean(data.Pout(int));
            ave.rpm(j) = mean(data.rpm(int));
            ave.wind1(j) = mean(data.wind1(int));
            ave.wind2(j) = mean(data.wind2(int));
            ave.wind3(j) = mean(data.wind3(int));
            ave.STDPreg(j) = std(data.Preg(int));
            ave.STDPout(j) = std(data.Pout(int));
            ave.STDrpm(j) = std(data.rpm(int));
            ave.STDwind1(j) = std(data.wind1(int));
            ave.STDwind2(j) = std(data.wind2(int));
            ave.STDwind3(j) = std(data.wind3(int));
            ave.temprC(j) = mean(data.tempr(int));%Deg Celcius
            ave.temprK(j) = ave.temprC(j) + 273.15;%convert to Kelvin
            ave.pressureHG(j) = mean(data.pressure(int));%in. Hg
            ave.pressurePa(j) = ave.pressureHG(j)*3386.389;%convert to
Pa
            ave.humidity(j) = mean(data.humidity(int));%percentage

            %Algorithm to average standard deviation of the wind
direction [Source: Yamartino Method]
```

```
            sa = (1 / ave.count(j))*sum(sind(data.winddir(int)));
            ca = (1 / ave.count(j))*sum(cosd(data.winddir(int)));

            ave.winddir(j) = atan2(sa,ca)*180 / pi;

            if ave.winddir(j) < 0

                ave.winddir(j) = ave.winddir(j) + 360;

            end

        else

            ave.time(j) = ave_num*j+day_num;
            ave.Preg(j) = NaN;
            ave.Pout(j) = NaN;
            ave.rpm(j) = NaN;
            ave.wind1(j) = NaN;
            ave.wind2(j) = NaN;
            ave.wind3(j) = NaN;
            ave.STDPreg(j) = NaN;
            ave.STDPout(j) = NaN;
            ave.STDrpm(j) = NaN;
            ave.STDwind1(j) = NaN;
            ave.STDwind2(j) = NaN;
            ave.STDwind3(j) = NaN;
            ave.temprC(j) = NaN;
            ave.temprK(j) = NaN;
            ave.humidity(j) = NaN;
            ave.winddir(j) = NaN;
            ave.pressureHG(j) = NaN;
            ave.pressurePa(j) = NaN;
            ave.STDwinddir(j) = NaN;

        end

        clear D;
        clear delta;

    end

    %Calculate average air density by the Ideal Gas Law
    ave.airdensity = ave.pressurePa ./ (287.05*ave.temprK);

    %Calculate sea-level normalized wind speeds
    ave.wind1SL = ave.wind1.*(ave.airdensity / 1.225).^(1/3);
    ave.wind2SL = ave.wind2.*(ave.airdensity / 1.225).^(1/3);
    ave.wind3SL = ave.wind3.*(ave.airdensity / 1.225).^(1/3);

    output =
['swwt',datestr(data.time(1),'yyyymmdd'),'_average_',num2str(average_in
terval),'sec'];
```

```
    cd(proc_directory);

    save(output,'ave');

end

disp('END OF CODE');
```

Averaged data plotting Code:

```
% Author: Brian Wallace,Yande Liu,Anagha Ray
% Date: 7/21/2013
% Version: AveragingPlottingCode_v4.m

close all;
clear all;
clc;

format long e;

%% part1

[FileName,PathName,FilterIndex] = uigetfile('*.mat','Select parsed data
acquisition files to be processed','MultiSelect','on');
N = length(FileName);

% Select location where plots will be saved
proc_directory = uigetdir('../','Select Directory to save the plots');
disp('  ');
disp('Files loaded... ');
disp('  ');

disp('Please select the starting time (HH:MM), then hit Enter');
disp(' ');
disp('(If you want to see the 24-hour graph, type "24" and then hit
Enter)');

x = input(' ','s');
answer = strcmp (x,'24');
if (answer == 1)
    x = '00:00';
    y = '23:59';
else
    disp('Please select the ending time, then hit Enter');
    y = input(' ','s');
end

average_interval = input('Enter data resolution (averaging) interval in
seconds: ');
ave_num = datenum(00,00,00,00,00,average_interval);
```

```
%% part2
for i=1:N

        disp(' ');
        disp(['Plotting from file ',num2str(i),' of
',num2str(N)]);%disply progress

        cd(PathName)%change the directary
        load(char(FileName(i)))%load the .mat file


    %% set up the time interval

    date = datestr(data.time(1), 'yyyy-mm-dd');
    newTime1 = [date, ' ',x];
    newTime2 = [date, ' ',y];

    timeNum1 = datenum(newTime1,'yyyy-mm-dd HH:MM');
    %disp (datestr(timeNum1));

    timeNum2 = datenum(newTime2,'yyyy-mm-dd HH:MM');
    %disp (timeNum2);



    %% Find the starding point at the data.time vector.
    n = 1;
    while (data.time(n) < timeNum1 )
        n = n+1;
    end

    p = n;
    q = 1;

    while (data.time(p) <=timeNum2) %Find the correct time interval
        %disp(datestr(data.time(p)));
        data.selectedTime (q) = data.time(p);
        data.selectedPreg (q) = data.Preg(p);
        data.selectedRPM (q) = data.rpm(p);
        data.selectedWind1 (q) = data.wind1(p);
        data.selectedWindDir (q) = data.winddir(p);

        p = p+1;
        q = q+1;
    end

    %% data averaging
    day_num = floor(data.selectedTime(1));
```

```
    time_interval = day_num:ave_num:day_num+1;

    num_time_ints = length(time_interval);

    num_pts = average_interval / 3;

    for j=1:num_time_ints-1

        int = find(data.selectedTime >= day_num + (j-1)*ave_num &
data.selectedTime < ave_num*j + day_num);
        ave.count(j) = length(int);

        if ave.count(j) == num_pts
            ave.time(j) = ave_num*j+day_num;
            ave.Preg(j) = mean(data.selectedPreg(int));
            ave.rpm(j) = mean(data.selectedRPM(int));
            ave.wind1(j) = mean(data.selectedWind1(int));
         %Algorithm to average standard deviation of the wind direction
[Source: Yamartino Method]
            sa = (1 / ave.count(j))*sum(sind(data.winddir(int)));
            ca = (1 / ave.count(j))*sum(cosd(data.winddir(int)));

            ave.winddir(j) = atan2(sa,ca)*180 / pi;

            if ave.winddir(j) < 0

                ave.winddir(j) = ave.winddir(j) + 360;

            end

            eta = sqrt(1 - (sa^2 + ca^2));
            ave.STDwinddir(j) = asind(eta)*(1+((2/sqrt(3))-1)*eta^3);
            else

            ave.time(j) = ave_num*j+day_num;
            ave.Preg(j) = NaN;
            ave.wind1(j) = NaN;
            ave.rpm(j) = NaN;
         end
    end
%     %Calculate average air density by the Ideal Gas Law
%     ave.airdensity = ave.pressurePa ./ (287.05*ave.temprK);
%
%     %Calculate sea-level normalized wind speeds
%     ave.wind1SL = ave.wind1.*(ave.airdensity / 1.225).^(1/3);
%

        %% plotting
```

```
        %PowerGraph =['PowerRequired vs Time
',datestr(data.selectedTime(1),'yyyy-mm-dd')];%set up variables to name
the file
        %WindGraph =['Windspeed1 vs Time
',datestr(data.selectedTime(1),'yyyy-mm-dd')];%set up variables to name
the file
        PowerTime =['Averaged Power vs Time
',datestr(data.selectedTime(1),'yyyy-mm-dd')];%set up variables to name
the file
        WindTime =['Averaged Windspeed vs Time
',datestr(data.selectedTime(1),'yyyy-mm-dd')];%set up variables to name
the file
        WindPower =['Averaged Power vs Windspeed
',datestr(data.selectedTime(1),'yyyy-mm-dd')];%set up variables to name
the file
        PowerRPM =['Averaged Power vs RPM
',datestr(data.selectedTime(1),'yyyy-mm-dd')];%set up variables to name
the file
        RPMWind = ['Averaged RPM Vs Windspeed
',datestr(data.selectedTime(1),'yyyy-mm-dd')];
        %make the plots
        figure(1)
        %plot(data.selectedTime,data.selectedPreg)%realtime data plot
        plot(ave.time,ave.Preg)%average data plot
        datetick('x',13) %change the data from the serial date number
to date string
        title(PowerTime)
        legend('Power')
        xlabel('Time  (s)')
        ylabel('Power  (watt)')



        figure(2)
        %plot(data.selectedTime,data.selectedWind1)%realtime data plot
        plot(ave.time,ave.wind1)%average data plot
        datetick('x',13)%change the data from the serial date number to
date string
        title(WindTime)
        legend('Windspeed')
        xlabel('Time  (s)')
        ylabel('Wind Speed  (m/s)')



        figure(3)
        %plot(data.selectedTime,data.selectedWind1)%realtime data plot
        scatter(ave.wind1,ave.Preg)%average data plot
        title(WindPower)
        ylabel('Power  (watt)')
        xlabel('Wind Speed  (m/s)')
```

```
        figure(4)
        %plot(data.selectedTime,data.selectedWind1)%realtime data plot
        scatter(ave.rpm,ave.Preg)%average data plot
        title(PowerRPM)
        xlabel('RPM')
        ylabel('Power  (watt)')
        cd(proc_directory);%change the directary to the place where the
user wants to save the files

         figure(5)
        %plot(data.selectedTime,data.selectedWind1)%realtime data plot
        scatter(ave.wind1,ave.rpm)%average data plot
        title(RPMWind)
        xlabel('Windspeed m/s')
        ylabel('RPM')
        cd(proc_directory);%change the directary to the place where the
user wants to save the files

         saveas(1,PowerTime);
         saveas(2,WindTime);
         saveas(3,WindPower);
         saveas(4,PowerRPM);
         saveas(5,RPMWind);




end


close all;
disp(' ');
disp('**********************************************************************
***');
disp(' ');
disp('All the plots are finished...');
disp(' ');
disp('END OF THE CODE');
```

Binning and plotting Code :

```
clear all;
close all;
clc;
format long e;

% This code reads in N-Second MATLAB .mat variable files and bins
% these data based on wind speed.
% This code also generate figures for binned data

% Author: Brian Wallace {bdw5003@psu.edu),Anagha Ray ,Armstrong (Yande
Liu)
```

```
% Date: 7/28/2013
% Version: BinPlotCode_v6.m

disp(' ');
disp('**********************************************************************
***');
disp('Skystream Wind Turbine Data Binning');
disp('**********************************************************************
***');
disp(' ');


%% User interface prompts and vector initialization

% Prompt the user to select N-Second Averaged MATLAB variable files to
be
% included in this bin analysis
[FileName,PathName,FilterIndex] = uigetfile('*.mat','Select N-Second
Average matlab variable files to be included in Bins');

% Select location where bin summary data will be saved
proc_directory = uigetdir('../','Select Directory to save Bin Summary
File and Data Summary File');

N = length(FileName);

disp('  ');
disp('STARTING TO LOAD THE DATA FROM ALL FILES... ');
disp('  ');

% Set high and low limits of bin analysis in meter / sec.
threshold_low = 1.5;
threshold_high = 20;

% Set bin increment in m/s
increment = .5;
% row_inc = 1;

% Calculate the number of bins to be populated.
total_ind = ((threshold_high-threshold_low)/0.5)+1;

% % Calculate the offset index for the first bin
% offset_ind = (threshold_low/0.5)-1;

% Initialize the bin data vectors.  These vectors will accumulate data
from
% all MATLAB variable files used in this bin analysis.

bin.data.temprK = [];
bin.data.pressurePa = [];
bin.data.airdensity = [];
bin.data.wind1 = [];
```

```
bin.data.wind2 = [];
bin.data.wind3 = [];
bin.data.STDwind1 = [];
bin.data.STDwind2 = [];
bin.data.STDwind3 = [];
bin.data.Preg = [];
bin.data.Pout = [];
bin.data.STDPout = [];
bin.data.STDPreg = [];
bin.data.rpm = [];
bin.data.STDrpm = [];
bin.data.time = [];
bin.data.count = [];
bin.data.winddir = [];
bin.data.STDwinddir = [];
bin.data.wind1SL = [];
bin.data.wind2SL = [];
bin.data.wind3SL = [];



%% Read MATLAB variable files and save to data vectors

% Loop through matlab variable files
% for i=1:N

%     disp(' ');
%     disp(['File ',num2str,' of ',num2str(N)]);
%     disp(' ');

    % Change working directory to location of MATLAB variable file to
be
    % read.
    cd(PathName);

    % Load MATLAB variable "ave" from MATLAB variable file
    load(char(FileName),'ave');

    % Append MATLAB variable data from MATLAB variable file to the bin
data
    % vectors.

    bin.data.time = [bin.data.time,ave.time];
    bin.data.count = [bin.data.count,ave.count];
    bin.data.Preg = [bin.data.Preg,ave.Preg];
    bin.data.Pout = [bin.data.Pout,ave.Pout];
    bin.data.rpm = [bin.data.rpm,ave.rpm];
    bin.data.wind1 = [bin.data.wind1,ave.wind1];
    bin.data.wind2 = [bin.data.wind2,ave.wind2];
    bin.data.wind3 = [bin.data.wind3,ave.wind3];
    bin.data.STDPreg = [bin.data.STDPreg,ave.STDPreg];
    bin.data.STDPout = [bin.data.STDPout,ave.STDPout];
    bin.data.STDrpm = [bin.data.STDrpm,ave.STDrpm];
```

```
    bin.data.STDwind1 = [bin.data.STDwind1,ave.STDwind1];
    bin.data.STDwind2 = [bin.data.STDwind2,ave.STDwind2];
    bin.data.STDwind3 = [bin.data.STDwind3,ave.STDwind3];
    bin.data.temprK = [bin.data.temprK,ave.temprK];
    bin.data.pressurePa = [bin.data.pressurePa,ave.pressurePa];
    bin.data.airdensity = [bin.data.airdensity,ave.airdensity];
    bin.data.winddir = [bin.data.winddir,ave.winddir];
    bin.data.STDwinddir = [bin.data.STDwinddir,ave.STDwinddir];
    bin.data.wind1SL = [bin.data.wind1SL,ave.wind1SL];
    bin.data.wind2SL = [bin.data.wind2SL,ave.wind2SL];
    bin.data.wind3SL = [bin.data.wind3SL,ave.wind3SL];



disp('  ');
disp('STARTING TO BIN THE DATA... ');
disp('  ');

%% Wind 1 Bin Analysis

% Binning Sequence for Wind1 Anemometer
bin_cent = threshold_low:increment:threshold_high;

for j=1:total_ind

    int = find(bin.data.wind1 >= bin_cent(j)-0.25 & bin.data.wind1 <
bin_cent(j)+0.25);
    bin.count(j) = length(int);

    if bin.count(j) >= 1

        bin.wind1.temprK(j) = mean(bin.data.temprK(int));
        bin.wind1.wind1(j) = mean(bin.data.wind1(int));
        bin.wind1.STDwind1(j) = mean(bin.data.STDwind1(int));
        bin.wind1.Preg(j) = mean(bin.data.Preg(int));
        bin.wind1.Pout(j) = mean(bin.data.Pout(int));
        bin.wind1.STDPreg(j) = mean(bin.data.STDPreg(int));
        bin.wind1.STDPout(j) = mean(bin.data.STDPout(int));
        bin.wind1.airdensity(j) = mean(bin.data.airdensity(int));
        bin.wind1.count(j) = bin.count(j);
        bin.wind1.rpm(j) = mean(bin.data.rpm(int));
        bin.wind1.STDrpm(j) = mean(bin.data.STDrpm(int));
        bin.wind1.Cp(j) = bin.wind1.Preg(j) / (0.125 *
bin.wind1.airdensity(j) * bin.wind1.wind1(j) * bin.wind1.wind1(j) *
bin.wind1.wind1(j) * pi * 3.7 * 3.7);
        bin.wind1.TSR(j) = (((bin.wind1.rpm(j) * pi * 2) / 60) * 1.85)
/ bin.wind1.wind1(j);
    else

        bin.wind1.temprK(j) = NaN;
        bin.wind1.wind1(j) = NaN;
        bin.wind1.STDwind1(j) = NaN;
        bin.wind1.Preg(j) = NaN;
```

```
        bin.wind1.Pout(j) = NaN;
        bin.wind1.STDPreg(j) = NaN;
        bin.wind1.STDPout(j) = NaN;
        bin.wind1.airdensity(j) = NaN;
        bin.wind1.count(j) = NaN;
        bin.wind1.rpm(j) = NaN;
        bin.wind1.STDrpm(j) = NaN;
        bin.wind1.Cp(j) = NaN;
        bin.wind1.TSR(j) = NaN;
    end

end

% Binning Sequence for Wind1 Anemometer; Sea Level Corrected
bin_cent = threshold_low:increment:threshold_high;

for j=1:total_ind

    int = find(bin.data.wind1SL >= bin_cent(j)-0.25 & bin.data.wind1SL
< bin_cent(j)+0.25);
    bin.count(j) = length(int);

    if bin.count(j) >= 1

        bin.wind1SL.temprK(j) = mean(bin.data.temprK(int));
        bin.wind1SL.wind1SL(j) = mean(bin.data.wind1SL(int));
        bin.wind1SL.STDwind1(j) = mean(bin.data.STDwind1(int));
        bin.wind1SL.Preg(j) = mean(bin.data.Preg(int));
        bin.wind1SL.Pout(j) = mean(bin.data.Pout(int));
        bin.wind1SL.STDPreg(j) = mean(bin.data.STDPreg(int));
        bin.wind1SL.STDPout(j) = mean(bin.data.STDPout(int));
        bin.wind1SL.airdensity(j) = mean(bin.data.airdensity(int));
        bin.wind1SL.count(j) = bin.count(j);
        bin.wind1SL.rpm(j) = mean(bin.data.rpm(int));
        bin.wind1SL.STDrpm(j) = mean(bin.data.STDrpm(int));
        bin.wind1SL.Cp(j) = bin.wind1SL.Preg(j) / (0.125 *
bin.wind1SL.airdensity(j) * bin.wind1SL.wind1SL(j) *
bin.wind1SL.wind1SL(j) * bin.wind1SL.wind1SL(j) * pi * 3.7 * 3.7);
        bin.wind1SL.TSR(j) = (((bin.wind1SL.rpm(j) * pi * 2) / 60) *
1.85) / bin.wind1SL.wind1SL(j);
    else

        bin.wind1SL.temprK(j) = NaN;
        bin.wind1SL.wind1SL(j) = NaN;
        bin.wind1SL.STDwind1(j) = NaN;
        bin.wind1SL.Preg(j) = NaN;
        bin.wind1SL.Pout(j) = NaN;
        bin.wind1SL.STDPreg(j) = NaN;
        bin.wind1SL.STDPout(j) = NaN;
        bin.wind1SL.airdensity(j) = NaN;
        bin.wind1SL.count(j) = NaN;
        bin.wind1SL.rpm(j) = NaN;
        bin.wind1SL.STDrpm(j) = NaN;
```

```
        bin.wind1SL.Cp(j) = NaN;
        bin.wind1SL.TSR(j) = NaN;
    end

end

%% Wind 2 Bin Analysis

% Binning Sequence for Wind2 Anemometer
bin_cent = threshold_low:increment:threshold_high;

for j=1:total_ind

    int = find(bin.data.wind2 >= bin_cent(j)-0.25 & bin.data.wind2 <
bin_cent(j)+0.25);
    bin.count(j) = length(int);

    if bin.count(j) >= 1

        bin.wind2.temprK(j) = mean(bin.data.temprK(int));
        bin.wind2.wind2(j) = mean(bin.data.wind2(int));
        bin.wind2.STDwind2(j) = mean(bin.data.STDwind2(int));
        bin.wind2.Preg(j) = mean(bin.data.Preg(int));
        bin.wind2.Pout(j) = mean(bin.data.Pout(int));
        bin.wind2.STDPreg(j) = mean(bin.data.STDPreg(int));
        bin.wind2.STDPout(j) = mean(bin.data.STDPout(int));
        bin.wind2.airdensity(j) = mean(bin.data.airdensity(int));
        bin.wind2.count(j) = bin.count(j);
        bin.wind2.rpm(j) = mean(bin.data.rpm(int));
        bin.wind2.STDrpm(j) = mean(bin.data.STDrpm(int));

    else

        bin.wind2.temprK(j) = NaN;
        bin.wind2.wind2(j) = NaN;
        bin.wind2.STDwind2(j) = NaN;
        bin.wind2.Preg(j) = NaN;
        bin.wind2.Pout(j) = NaN;
        bin.wind2.STDPreg(j) = NaN;
        bin.wind2.STDPout(j) = NaN;
        bin.wind2.airdensity(j) = NaN;
        bin.wind2.count(j) = NaN;
        bin.wind2.rpm(j) = NaN;
        bin.wind2.STDrpm(j) = NaN;

    end

end

% Binning Sequence for Wind2 Anemometer; Sea Level Corrected
bin_cent = threshold_low:increment:threshold_high;
```

```
for j=1:total_ind

    int = find(bin.data.wind2SL >= bin_cent(j)-0.25 & bin.data.wind2SL
< bin_cent(j)+0.25);
    bin.count(j) = length(int);

    if bin.count(j) >= 1

        bin.wind2SL.temprK(j) = mean(bin.data.temprK(int));
        bin.wind2SL.wind2SL(j) = mean(bin.data.wind2SL(int));
        bin.wind2SL.STDwind2(j) = mean(bin.data.STDwind2(int));
        bin.wind2SL.Preg(j) = mean(bin.data.Preg(int));
        bin.wind2SL.Pout(j) = mean(bin.data.Pout(int));
        bin.wind2SL.STDPreg(j) = mean(bin.data.STDPreg(int));
        bin.wind2SL.STDPout(j) = mean(bin.data.STDPout(int));
        bin.wind2SL.airdensity(j) = mean(bin.data.airdensity(int));
        bin.wind2SL.count(j) = bin.count(j);
        bin.wind2SL.rpm(j) = mean(bin.data.rpm(int));
        bin.wind2SL.STDrpm(j) = mean(bin.data.STDrpm(int));

    else

        bin.wind2SL.temprK(j) = NaN;
        bin.wind2SL.wind2SL(j) = NaN;
        bin.wind2SL.STDwind2(j) = NaN;
        bin.wind2SL.Preg(j) = NaN;
        bin.wind2SL.Pout(j) = NaN;
        bin.wind2SL.STDPreg(j) = NaN;
        bin.wind2SL.STDPout(j) = NaN;
        bin.wind2SL.airdensity(j) = NaN;
        bin.wind2SL.count(j) = NaN;
        bin.wind2SL.rpm(j) = NaN;
        bin.wind2SL.STDrpm(j) = NaN;

    end

end

%% Wind 3 Bin Analysis

% Binning Sequence for Wind3 Anemometer
bin_cent = threshold_low:increment:threshold_high;

for j=1:total_ind

    int = find(bin.data.wind3 >= bin_cent(j)-0.25 & bin.data.wind3 <
bin_cent(j)+0.25);
    bin.count(j) = length(int);

    if bin.count(j) >= 1

        bin.wind3.temprK(j) = mean(bin.data.temprK(int));
```

```
        bin.wind3.wind3(j) = mean(bin.data.wind3(int));
        bin.wind3.STDwind3(j) = mean(bin.data.STDwind3(int));
        bin.wind3.Preg(j) = mean(bin.data.Preg(int));
        bin.wind3.Pout(j) = mean(bin.data.Pout(int));
        bin.wind3.STDPreg(j) = mean(bin.data.STDPreg(int));
        bin.wind3.STDPout(j) = mean(bin.data.STDPout(int));
        bin.wind3.airdensity(j) = mean(bin.data.airdensity(int));
        bin.wind3.count(j) = bin.count(j);
        bin.wind3.rpm(j) = mean(bin.data.rpm(int));
        bin.wind3.STDrpm(j) = mean(bin.data.STDrpm(int));

    else

        bin.wind3.temprK(j) = NaN;
        bin.wind3.wind3(j) = NaN;
        bin.wind3.STDwind3(j) = NaN;
        bin.wind3.Preg(j) = NaN;
        bin.wind3.Pout(j) = NaN;
        bin.wind3.STDPreg(j) = NaN;
        bin.wind3.STDPout(j) = NaN;
        bin.wind3.airdensity(j) = NaN;
        bin.wind3.count(j) = NaN;
        bin.wind3.rpm(j) = NaN;
        bin.wind3.STDrpm(j) = NaN;

    end

end

% Binning Sequence for Wind3 Anemometer; Sea Level Corrected
bin_cent = threshold_low:increment:threshold_high;

for j=1:total_ind

    int = find(bin.data.wind3SL >= bin_cent(j)-0.25 & bin.data.wind3SL
< bin_cent(j)+0.25);
    bin.count(j) = length(int);

    if bin.count(j) >= 1

        bin.wind3SL.temprK(j) = mean(bin.data.temprK(int));
        bin.wind3SL.wind3SL(j) = mean(bin.data.wind3SL(int));
        bin.wind3SL.STDwind3(j) = mean(bin.data.STDwind3(int));
        bin.wind3SL.Preg(j) = mean(bin.data.Preg(int));
        bin.wind3SL.Pout(j) = mean(bin.data.Pout(int));
        bin.wind3SL.STDPreg(j) = mean(bin.data.STDPreg(int));
        bin.wind3SL.STDPout(j) = mean(bin.data.STDPout(int));
        bin.wind3SL.airdensity(j) = mean(bin.data.airdensity(int));
        bin.wind3SL.count(j) = bin.count(j);
        bin.wind3SL.rpm(j) = mean(bin.data.rpm(int));
        bin.wind3SL.STDrpm(j) = mean(bin.data.STDrpm(int));

    else
```

```
        bin.wind3SL.temprK(j) = NaN;
        bin.wind3SL.wind3SL(j) = NaN;
        bin.wind3SL.STDwind3(j) = NaN;
        bin.wind3SL.Preg(j) = NaN;
        bin.wind3SL.Pout(j) = NaN;
        bin.wind3SL.STDPreg(j) = NaN;
        bin.wind3SL.STDPout(j) = NaN;
        bin.wind3SL.airdensity(j) = NaN;
        bin.wind3SL.count(j) = NaN;
        bin.wind3SL.rpm(j) = NaN;
        bin.wind3SL.STDrpm(j) = NaN;

    end

end

%% Save Files



output = ['binned_swwt_proc',datestr(now,'mm_dd_yy')];

cd(proc_directory);

save(output,'bin','FileName');
disp('**********************************************************************
***');
disp('You New Binned File Has Been saved')



%% Plotting figures
disp(' ')
disp(' ')
disp('Now, please select the options:');
disp(' ');
disp('1. Power vs. Windspeed ');
disp('2. Power Vs RPM');
disp('3. Cp Vs TSR');
disp('4. Weubull for windspeed');
disp('5. RPM Vs Windspeed');
disp('6. Mean and STD power Vs Windspeed');
disp('7. Cp Vs Windspeed');
disp('8. Mean power Vs TSR');
disp('9. I want to generate all of them');
disp('0. Close and quit');
disp('**********************************************************************
***');

answer = input(' ','s');
plot1 = strcmp (answer,'1');
```

```
plot2 = strcmp (answer,'2');
plot3 = strcmp (answer,'3');
plot4 = strcmp (answer,'4');
plot5 = strcmp (answer,'5');
plot6 = strcmp (answer,'6');
plot7 = strcmp (answer,'7');
plot8 = strcmp (answer,'8');
plot9 = strcmp (answer,'9');
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp('**********************************************************************
***');

%set up titles

MeanGraph =['Power vs Windspeed  ',datestr(now,'mm-dd')];%cannot
contain "." or "_"
STDGraph =['Power vs RPM  ',datestr(now,'mm-dd')];
BarGraph =['Weibull curve  ',datestr(now,'mm-dd')];
CpTSR =['Cp vs TSR  ',datestr(now,'mm-dd')];
TSRGraph = ['RPM Vs Windspeed',datestr(now,'mm-dd')];
STDMeanGraph=['STD and mean power Vs Windspeed',datestr(now,'mm-dd')];
PowerTSR =['power vs TSR  ',datestr(now,'mm-dd')];
CpWind =['Cp vs Windspeed  ',datestr(now,'mm-dd')];

if answer ~= '0'%OPTION PLOT FIGURES

disp('**********************************************************************
***');
    proc_directory = uigetdir('../','Select a New Directory to Save
Graphs');

    cd(proc_directory);%change the directary to the place where the
user wants to save the files
%options
    %1.Mean Power vs. Mean Windspeed
    if  (plot1 == 1 || plot9 == 1);
        figure(1);
        plot(bin.wind1.wind1,bin.wind1.Preg,'-.ob');
        title('Power vs. Windspeed');

        xlabel('Windspeed  (m/s)');
        ylabel('Power  (watt)');
        saveas(1,MeanGraph);
    end

    %2.Power vs. RPM
    if (plot2 == 1 || plot9 == 1);
```

```
        figure(2);
        plot(bin.wind1.rpm,bin.wind1.Preg,'-.ok');
        title('Power vs. RPM');

        xlabel('Windspeed  (m/s)');
        ylabel('Power  (watt)');
        saveas(2,STDGraph);
    end

    %3.Cp vs. TSR
    if  (plot4 == 1 || plot9 == 1);
        figure(4);
        plot(bin.wind1.TSR,bin.wind1.Cp,'-.or');
        title('Cp vs. TSR');

        xlabel('TSR');
        ylabel('Cp');
        saveas(4,CpTSR);
    end

   %4.Mean Windspeed vs Time, Bar Graph,
    if (plot3 == 1 || plot9 == 1);
        figure(3);
        bar(bin.wind1.wind1,bin.wind1.count);
        title('Mean Wind Distribution, Bar Graph ');
        xlabel('Windspeed  (m/s)');
        ylabel('Count');
        saveas(3,BarGraph);
    end

    %5.RPM vs. Windspeed
    if  (plot5 == 1 || plot9 == 1);
        figure(5);
        plot(bin.wind1.wind1,bin.wind1.rpm,'-.og');
        title('RPM vs. Windspeed');

        xlabel('Windspeed  (m/s)');
        ylabel('RPM');
        saveas(5,TSRGraph);
    end

    %6.mean and STD Power vs. Windspeed
    if  (plot6 == 1 || plot9 == 1);
        figure(6);
        plot(bin.wind1.wind1,bin.wind1.Preg,'-
.or',bin.wind1.wind1,bin.wind1.STDPreg,'-.og');
        title('Power vs. Windspeed');
        legend('STD Power','Mean Power');
        xlabel('Windspeed  (m/s)');
        ylabel('Power  (watt)');
        saveas(6,STDMeanGraph);
    end
```

```
    %7.Cp vs. Mean Windspeed
    if  (plot7 == 1 || plot9 == 1);
        figure(7);
        plot(bin.wind1.wind1,bin.wind1.Cp,'-.og');
        title('Cp vs. Mean Windspeed');
        legend('Cp');
        xlabel('Windspeed  (m/s)');
        ylabel('Cp');
        saveas(7,CpWind);
    end

    %8.Mean Power vs. TSR
    if  (plot8 == 1 || plot9 == 1);
        figure(8);
        plot(bin.wind1.TSR,bin.wind1.Preg,'-.og');
        title('Mean Power vs. TSR');
        legend('Mean Power');
        xlabel('TSR');
        ylabel('Power  (watt)');
        saveas(8,PowerTSR);
    end
end


%close all;
disp('.');
disp('.');
disp('.');
disp('.');
disp('.');
disp('.');
disp('.');
disp('Figures are made and saved in the target directory...');
disp('END OF CODE');


%     For multiple curves on the same plot
%     Plot4 = figure (1);
%
% % Create axes
% axes1 = axes('Parent',Plot4);
% hold(axes1,'all');
% % Create plot
%
plot(bin.wind1.wind1,bin.wind1.Preg,'Parent',axes1,'Marker','o','LineSt
yle','none',...
%     'DisplayName','Wind1');
%
plot(bin.wind2.wind2,bin.wind1.Preg,'Parent',axes1,'Marker','.','LineSt
yle','none',...
%     'DisplayName','Wind2');
%
plot(bin.wind3.wind3,bin.wind1.Preg,'Parent',axes1,'Marker','*','LineSt
yle','none',...
```

```
%     'DisplayName','Wind3');
% xlabel({'Windspeed (m/s)'});
% ylabel({'Power (watt)'});
% title({'Binned Windspeed vs. Power'});
% legend(axes1,'show');
```

Digitizing Code : By A. Prasad Original version created by J.D.Cogdell

```
function varargout = digitize2(varargin)

%DIGITIZE  digitize data from image.
%   DIGITIZE with no input or output arguments allows the user to
%   select an image file to load;  only IMREAD-compatible image
%   files are supported.  The function then prompts the user
%   to graphically identify the location of the origin and the X-
%   and Y- axes of the plot.  The user may then graphically select
%   an arbitrary number of data points from anywhere on the image
%   using the left mouse button.  Data acquisition is terminated
%   by clicking the right mouse button.  The function then prompts
%   the user to save the acquired data to file.
%
%   ACQDATA = DIGITIZE with one output argument returns the X- and
%   Y- values of the graphically selected data in the array ACQDATA.
%   The user is not prompted to save the data to file.
%
%   DIGITIZE(FILENAME) with one input argument FILENAME is used to
%   directly specify the image file to load.  As above, the user is
%   prompted to graphically set up the coordinate system and select
%   target data points.
%
% See also IMREAD, IMFINFO.

% Author(s): A. Prasad
% Original version created by J.D.Cogdell

% Check for proper number of input arguments
error(nargchk(0,1,nargin));

% Identify image filename
if (nargin == 0),
     [filename, pathname] = uigetfile( ...
           {'*.jpg;*.tif;*.gif;*.png;*.bmp', ...
        'All MATLAB Image Files (*.jpg,*.tif,*.gif,*.png,*.bmp)'; ...
        '*.jpg;*.jpeg', ...
        'JPEG Files (*.jpg,*.jpeg)'; ...
        '*.tif;*.tiff', ...
        'TIFF Files (*.tif,*.tiff)'; ...
        '*.gif', ...
        'GIF Files (*.gif)'; ...
        '*.png', ...
        'PNG Files (*.png)'; ...
        '*.bmp', ...
```

```
        'Bitmap Files (*.bmp)'; ...
        '*.*', ...
        'All Files (*.*)'}, ...
           'Select image file');
     if isequal(filename,0) | isequal(pathname,0)
      return
     else
      imagename = fullfile(pathname, filename);
     end
elseif nargin == 1,
     imagename = varargin{1};
     [path, file,ext] = fileparts(imagename);
     filename = strcat(file,ext);
end

% Read image from target filename
pic = imread(imagename);
image(pic)
FigName = ['IMAGE: ' filename];
set(gcf,'Units', 'normalized', ...
    'Position', [0 0.125 1 0.85], ...
    'Name', FigName, ...
    'NumberTitle', 'Off', ...
    'MenuBar','None')
set(gca,'Units','normalized','Position',[0   0 1   1]);

% Determine location of origin with mouse click
OriginButton = questdlg('Select the ORIGIN with left mouse button
click', ...
            'DIGITIZE: user input required', ...
            'OK','Cancel','OK');
switch OriginButton,
     case 'OK',
      drawnow
      [Xopixels,Yopixels] = ginput(1);
      line(Xopixels,Yopixels,...
           'Marker','o','Color','g','MarkerSize',14)
      line(Xopixels,Yopixels,...
           'Marker','x','Color','g','MarkerSize',14)
     case 'Cancel',
      close(FigName)
      return
end % switch OriginButton

% Prompt user for X- & Y- values at origin
prompt={'Enter the abcissa (X value) at the origin',...
        'Enter the ordinate (Y value) at the origin:'};
def={'0','0'};
dlgTitle='DIGITIZE: user input required';
lineNo=1;
answer=inputdlg(prompt,dlgTitle,lineNo,def);
if (isempty(char(answer{:})) == 1),
     close(FigName)
     return
```

```
else
    OriginXYdata = str2num(char(answer{:}));
end


% Define X-axis
XLimButton = questdlg(...
      'Select a point on the X-axis with left mouse button click ', ...
      'DIGITIZE: user input required', ...
      'OK','Cancel','OK');
switch XLimButton,
     case 'OK',
      drawnow
      [XAxisXpixels,XAxisYpixels] = ginput(1);
      line(XAxisXpixels,XAxisYpixels,...
           'Marker','*','Color','b','MarkerSize',14)
      line(XAxisXpixels,XAxisYpixels,...
           'Marker','s','Color','b','MarkerSize',14)
     case 'Cancel',
      close(FigName)
      return
end % switch XLimButton

% Prompt user for XLim value
prompt={'Enter the abcissa (X value) at the selected point'};
def={'1'};
dlgTitle='DIGITIZE: user input required';
lineNo=1;
answer=inputdlg(prompt,dlgTitle,lineNo,def);
if (isempty(char(answer{:})) == 1),
     close(FigName)
     return
else
     XAxisXdata = str2num(char(answer{:}));
end

% Determine X-axis scaling
Xtype = questdlg(...
      'Select axis type for absicca (X)', ...
      'DIGITIZE: user input required', ...
      'LINEAR','LOGARITHMIC','Cancel');
drawnow
switch upper(Xtype),
     case 'LINEAR',
      logx = 0;
      scalefactorXdata = XAxisXdata - OriginXYdata(1);
     case 'LOGARITHMIC',
      logx = 1;
      scalefactorXdata = log10(XAxisXdata/OriginXYdata(1));
     case 'CANCEL',
      close(FigName)
      return
end % switch Xtype
```

```
% Rotate image if necessary
% note image file line 1 is at top
th = atan((XAxisYpixels-Yopixels)/(XAxisXpixels-Xopixels));
% axis rotation matrix
rotmat = [cos(th) sin(th); -sin(th) cos(th)];


% Define Y-axis
YLimButton = questdlg(...
      'Select a point on the Y-axis with left mouse button click', ...
      'DIGITIZE: user input required', ...
      'OK','Cancel','OK');
switch YLimButton,
     case 'OK',
      drawnow
      [YAxisXpixels,YAxisYpixels] = ginput(1);
      line(YAxisXpixels,YAxisYpixels,...
           'Marker','*','Color','b','MarkerSize',14)
      line(YAxisXpixels,YAxisYpixels,...
           'Marker','s','Color','b','MarkerSize',14)
     case 'Cancel',
      close(FigName)
      return
end % switch YLimButton

% Prompt user for YLim value
prompt={'Enter the ordinate (Y value) at the selected point'};
def={'1'};
dlgTitle='DIGITIZE: user input required';
lineNo=1;
answer=inputdlg(prompt,dlgTitle,lineNo,def);
if (isempty(char(answer{:})) == 1),
     close(FigName)
     return
else
     YAxisYdata = str2num(char(answer{:}));
end

% Determine Y-axis scaling
Ytype = questdlg('Select axis type for ordinate (Y)', ...
         'DIGITIZE: user input required', ...
         'LINEAR','LOGARITHMIC','Cancel');
drawnow
switch upper(Ytype),
     case 'LINEAR',
      logy = 0;
      scalefactorYdata = YAxisYdata - OriginXYdata(2);
     case 'LOGARITHMIC',
      logy = 1;
      scalefactorYdata = log10(YAxisYdata/OriginXYdata(2));
     case 'CANCEL',
      close(FigName)
      return
```

```
end % switch Ytype

% Complete rotation matrix definition as necessary
delxyx = rotmat*[(XAxisXpixels-Xopixels);(XAxisYpixels-Yopixels)];
delxyy = rotmat*[(YAxisXpixels-Xopixels);(YAxisYpixels-Yopixels)];
delXcal = delxyx(1);
delYcal = delxyy(2);

% Commence Data Acquisition from image
msgStr{1} = 'Click with LEFT mouse button to ACQUIRE';
msgStr{2} = ' ';
msgStr{3} = 'Click with RIGHT mouse button to QUIT';
titleStr = 'Ready for data acquisition';
uiwait(msgbox(msgStr,titleStr,'warn','modal'));
drawnow

numberformat = '%6.2f';
nXY = [];
ng = 0;

fprintf(['\n INFO >> Click with RIGHT mouse button to QUIT \n\n']);
n = 0;
disp(sprintf('\n %s \n',' Index          X            Y'))

% %%%%%%%%%%%%%% DATA ACQUISITION LOOP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
while 1
    [x,y, buttonNumber] = ginput(1);
    xy = rotmat*[(x-Xopixels);(y-Yopixels)];
    delXpoint = xy(1);
    delYpoint = xy(2);
    if buttonNumber == 1,
        line(x,y,'Marker','.','Color','r','MarkerSize',12)
        if logx,
            x =
OriginXYdata(1)*10^(delXpoint/delXcal*scalefactorXdata);
        else
            x = OriginXYdata(1) + delXpoint/delXcal*scalefactorXdata;
        end
        if logy,
            y =
OriginXYdata(2)*10^(delYpoint/delYcal*scalefactorYdata);
        else
            y = OriginXYdata(2) + delYpoint/delYcal*scalefactorYdata;
        end
        n = n+1;
        xpt(n) = x;
        ypt(n) = y;
        disp(sprintf(' %4d         %f      %f',n, x, y))
        ng = ng+1;
        nXY(ng,:) = [n x y];
    else
        query = questdlg('STOP digitizing and QUIT ?', ...
            'DIGITIZE: confirmation', ...
```

```
            'YES', 'NO', 'NO');
        drawnow
        switch upper(query),
            case 'YES',
                disp(sprintf('\n'))
                break
            case 'NO',

        end % switch query
    end
end
```

Process used to convert matlab data to excel and then import into windographer

1. BinningCode_v2.m is used to produce the binned results.
2. The raw data from the binned results i.e bin.data is used for making the file
3. The each binned data we wish to add is put in a matrix 'M'(except Date and Time because they are string values).
4. This matrix is converted to a text file using matlab code: dlmwrite('textfil.txt', M')
5. Text file is converted to Excel file (comma delimited)
6. Date and Time is added to the excel file (copy and pasted from matlab date output). The date format compatible with windographer is taken from matlab.
7. Headings are added to each column of data (headings are in the same order as formed in the M matrix)
8. Excel file inputted into windographer and results obtained.