WORCESTER POLYTECHNIC INSTITUTE

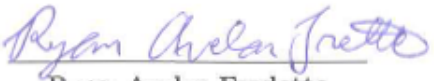
Scale-Model Testing of Tethered Undersea Kites for Power Generation

A Thesis Research Report
Submitted to the Faculty
of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the
Degree of Masters of Science
In Mechanical Engineering

SUBMITTED BY:



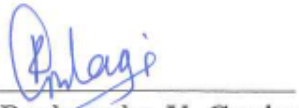
Ryan Avelar Fredette
June 2015

APPROVED BY:

Prof. David J. Olinger
Thesis Advisor

Prof. Anthony Linn
Committee Member

Prof. Seongkyun Im
Committee Member



Prof. Raghvendra V. Cowlagi
Graduate Committee Rep.

Abstract

This research focuses on studying the feasibility of tethered undersea kite (TUSK) systems for power generation. Underwater tethered kite systems consist of a rigid wing that moves in a circular or figure-8 path below the surface. The tether can connect to a platform mounted either on the surface or anchored to the seafloor. On the kite is a turbine that extracts energy from the kite's forward motion, which has the potential to be several times the current velocity. This speed multiplication combined with the density increase of water as opposed to air is one of the main benefits of this class of systems over wind turbines. A scale-model TUSK kite was designed. Testing was conducted in a water flume at Alden Research Labs (ARL). Model scale factors were determined from a real world prototype TUSK system currently in commercial development. The scale-model kite was primarily constructed out of ABS plastic using 3D printing rapid prototyping methods. Other components of the system were either repurposed from prior projects or constructed with traditional methods. Testing was conducted at current speeds of 0.15 m/s, 0.31 m/s, and 0.46 m/s; kite pitch angles of 80°, 85°, and 90°; and over circular and figure 8 trajectory shapes. Data collected included the azimuth and declination angles of the rigid tether as well as the power output of the generator on board the kite. Filtering techniques were employed on the data to generate graphs of kite position, velocity, and output for analysis. Relationships between current velocity, kite velocity, kite pitch angle, and power output have been measured. Inaccuracies in the model and areas for improvement in future work have been identified.

Acknowledgments

This work would not have been possible without the financial support of the National Science Foundation Energy for Sustainability Program under grants CBET-1033812 and CBET-1336130.

Alden Research Lab was an instrumental partner in the completion of this project, and I would like to thank them for the use of their water flume facilities.

I am indebted to my colleagues Kevin Arruda, Richard Eberheim and Brad Mello for their assistance with design input, construction, and testing.

I would also like to thank Professor Olinger as my project advisor for his efforts in guiding my work and revising my writing.

Contents

List of Figures

List of Tables

Nomenclature

V_{kite}	Kite Speed
$V_{current}$	Current Speed
b	Kite Wingspan
A_{kite}	Kite Wing Area
ζ	Kite Pitch Angle
L_t	Tether Length
D_t	Tether Outside Diameter
θ	Tether Declination Angle
Φ	Tether Azimuth Angle
r	Blade Section Radius
λ_r	Local Tip Speed Ratio
Φ	Angle of Relative Wind
c	Turbine Blade Chord Length
θ_p	Section Pitch Angle
θ_t	Blade Twist Angle
V_t	Turbine Inlet Velocity
B	Number of Turbine Blades
$\theta_{p,0}$	Blade Twist Angle at Tip
ω	Revolutions per Minute
R	Turbine Radius
α	Airfoil Section Angle of Attack
C_l	Airfoil Section Lift Coefficient
I_S	Measured Current
V_{out}	Voltage Output To Arduino
R_C	Internal Current Sensor Resistance
R_S	Shunt Resistor Value
R_L	Output Resistor Value

1 Introduction

1.1 Basic TUSK Concept

Ocean and tidal currents are rarely used as a renewable energy resource. One example of this kind of resource is the Agulhas Current off the coast of South Africa, studied by Moodley, et al [1]. Of the four sites studied in the cited paper, the greatest power potential was found to be 1913 GWh per year. Another area that has been studied is the section of the Gulf Stream off the coast of Florida [2]. The energy potential of that area was estimated at 219 GWh per year. Even given the potential generation capability of those sites and others, no large-scale harvesting technologies have been installed anywhere in the world. Tethered Undersea Kite (TUSK) systems, as proposed by Landberg [3], might remedy this shortcoming and take advantage of the potential of tidal and ocean currents. In a TUSK system, a kite is connected to a platform by a flexible tether. This platform can be fastened to the seafloor for applications in shallower water, or float on the surface for deep water sites. Figure 1 shows a typical arrangement of the components in a TUSK system.

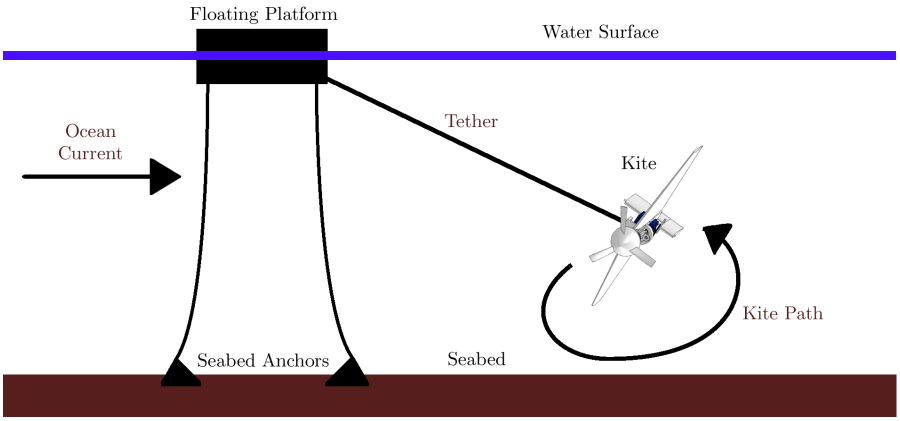


Figure 1: Overview of a general TUSK system and its basic components.

The kites used in these applications are rigid wings similar to airborne gliders. These wings use the passing current to generate forces that move the kite along a path across

the current flow. The exact path that the kite follows is determined by the position of various control surfaces that influence the motion of the kite about its pitch, roll, and yaw axes, as shown in figure 2.

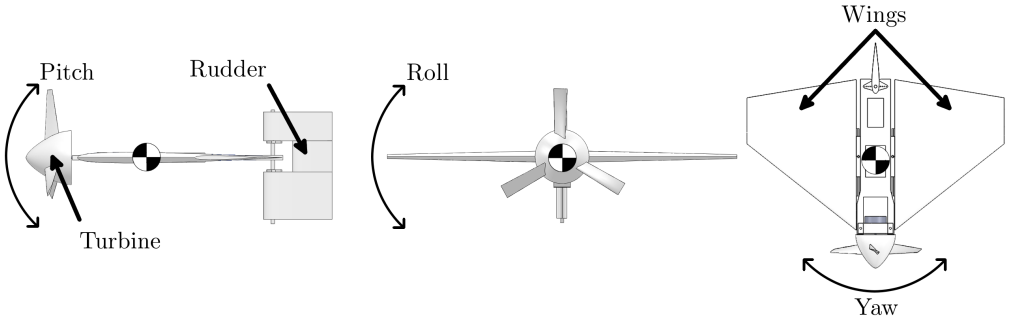


Figure 2: Illustration of pitch, roll, and yaw axes, along with rotation centers.

There are two main methods of extracting power from a kite system, termed GroundGen and FlyGen. In a GroundGen system, the kite tether is wound around a spool on the ground attached to the axle of a generator. This requires that the kite follow a “pumping" trajectory. As the kite moves downwind in a circular or figure-8 path, it spools out the tether and spins the generator. When a maximum tether length is reached the angle of the kite is changed to reduce the hydrodynamic forces on it, allowing the kite to be reeled back in using a smaller amount of energy than was generated during the reel out phase. Since the generation equipment is on the ground in this configuration, the kite can be made of light flexible fabric. This can reduce the risk of damage from a kite crash due to winds dropping below minimum values required to sustain flight.

A FlyGen system, which is studied in this thesis, places a turbine and generator on the kite itself. In this configuration, the base station can be mechanically simpler because the tether is a constant length and does not need to alternately retract and extend. Simpler position control algorithms can be employed because the kite does not move downwind. The same circular or figure-8 shapes are flown, but with a constant tether length. However, FlyGen configurations usually employ rigid kites that are heavier

than their flexible counterparts. Thus, the risk of damage during a crash is increased as compared to GroundGen setups. Another disadvantage of FlyGen configurations is the losses incurred by transmitting the power down the tether. To reduce transmission losses voltages have to be very high, which necessitates large transformers. Given their weight and size, there is no practical way to locate these transformers on the kite itself. Therefore, the power coming down the tether is at a lower voltage than is optimal for the tether length distance between the kite and a step-up transformer on the ground. A GroundGen system could locate this transformer at the generator to minimize these losses.

Loyd [4] first studied crosswind kite power, and a short summary of his findings is presented here. The amount of power that can be extracted from a fluid flow is found by

$$P = C_P \frac{1}{2} \rho A V_{current}^3 \tag{1.1}$$

where ρ is the density of the fluid, A is the area swept by the turbine, and C_P is a power coefficient. This holds for fixed turbines and kites that remain stationary with respect to the ground. With cross-current motion, the optimal theoretical speed achievable [4] is a function of the lift to drag ratio of the kite and tether combination, given by

$$V_{kite} = \frac{2}{3} \frac{C_L}{C_D} V_{current} \tag{1.2}$$

Because power is being extracted by the kite rather than a fixed turbine, the speed found with equation 1.2 can be substituted into 1.1 to obtain

$$P_{kite} = C_P \frac{1}{2} \rho A V_{kite}^3 \tag{1.3}$$

Since V_{kite} will always be higher than $V_{current}$, this demonstrates the origin of the power advantage of kites over fixed turbines. An example will be presented to demonstrate

the magnitude of this advantage. If $V_{current}$ is taken to be 2 m/s and the $\frac{L}{D}$ is equal to 8, V_{kite} will be 10.67 m/s according to equation 1.2. Since the proportion between crosswind power and stationary power goes by velocity cubed, the kite will produce $\left(\frac{V_{kite}}{V_{current}}\right)^3 = \left(\frac{10.67}{2}\right)^3 = 151.8$ times more power than a fixed turbine with the same swept area.

Other considerations can also highlight advantages of airborne and waterborne kites over fixed, bladed turbines. Because of the relationship between speed and aerodynamic force, the outer portion of a fix turbine blade's span contributes the majority of the power produced by the turbine. The inner portions of the blade, along with the hub and associated support structure, comprise the majority of the weight and complexity of the device but offer little to the power output. This material volume and weight also negatively impact the environmental footprint of the turbine by virtue of their usage of raw material, fabrication, and transportation resources. With a kite system, these components are all replaced with a single lightweight tether between the kite and a base station. If the kite in this example were made the same wing span as the existing turbine's outer blade length, and flown in a circle of the same radius, it would produce significantly more power than the original turbine while being a lighter, smaller system. A literature review of studies of AWE systems will be given later in this section.

1.2 Power Estimates

The basic equations used to predict kite power have already been presented in section 1.1. Loyd [4] and Diehl [5] derived a more realistic upper limit of possible power output from any kite flying in any fluid flow.

$$P \leq \frac{2}{27}\,\rho\,A\,V_{current}^3\,C_R\left(\frac{C_R}{C_D}\right)^2 \tag{1.4}$$

where $C_{D_{power}}$ is the drag coefficient due to energy production by the kite ande

$$C_R = C_L \sqrt{1 + \left(\frac{C_D + C_{D_{power}}}{C_L}\right)^2} \tag{1.5}$$

For a TUSK system, values of $C_L = 1$ and $C_D = 0.1$ can be assumed to be realistic. Best performance is attained when $C_{D_{power}} = \frac{C_D}{2}$ [4,5], so $C_{D_{power}} = 0.05$. Using equation 1.5, yields $C_R = 1.01$, making $\frac{C_R}{C_D} = 10.1$. The area of a typical small TUSK kite is A = 11 m^2, operating in $V_{current} = 1.5$ m/s with a water density of $\rho = 998$ kg/m^3. Using equation 1.4 and the listed parameters results in a power output of 282 kW per kite.

Equations 1.4 and 1.5 hold if the total aerodynamic force is in line with the current direction, the only loss is due to the drag of the wing, and the speed of the wing is $V_{kite} = \frac{2}{3}\frac{C_R}{C_D} V_{current}$. At high $\frac{L}{D}$ and low “extra" drag, which are good assumptions for cross-current kites, $C_R \simeq C_L$ and the kite speed goes to

$$V_{kite} = \frac{2}{3}\frac{C_L}{C_D} V_{current} \tag{1.6}$$

The force exerted by a kite is not always aligned with the wind direction, a condition that is required to achieve optimal power output. The power reduction due to cosine losses is found by

$$P_{cosine} = V_{current}\, F_{kite}\, \cos\theta \tag{1.7}$$

where F_{kite} is the aerodynamic force generated by the kite, and θ is the angle between the flow direction and the tether. Figure 3 shows the arrangement of the forces that lead to cosine losses. The angle of the tether away from the horizontal is a major contributor to cosine losses.

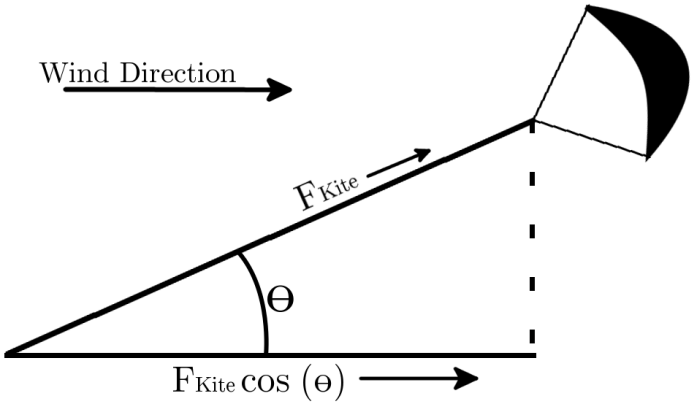


Figure 3: Illustration of tether angle leading to cosine losses.

1.3 Advantages of TUSK vs AWE

TUSK systems are similar to airborne wind energy (AWE) systems, which use a tethered kite in air. AWE systems are being considered as an alternative to traditional wind turbines. Both types use kites tethered to base stations flying in cyclical paths to generate power. Since the development of AWE systems is more advanced than TUSK systems, lessons learned from AWE can guide TUSK development. Even given the similarities between the two classes of systems, there are distinct advantages associated with using kites underwater. The 800 times increase in density of water over air allows for increased power output from a TUSK system, as described in equation 1.3. For optimal performance, kites need to operate where there is consistent wind which translates into altitudes of about 500 meters. Depending on the location of the kite farm, this could negatively affect air traffic. Also, maximum power is produced when the kite is moving across the current, thus necessitating that the trajectory sweep very wide angles covering a large area. All of the land below the kite's path needs to be kept clear in the event of a failure resulting in a kite crash, meaning that this land can't be used for another purpose due to access restrictions. Siting a kite farm underwater does not incur these space penalties. As with conventional wind turbines, there is a chance of public opposition

to AWE systems due to their appearance and effect on nearby neighborhoods. TUSK systems would likely have a smaller visible impact with surface platforms over the horizon offshore. Ocean currents are more stable in direction and intensity than wind. Greater stability in the resource being harvested will result in greater stability and predictability of the power the kite farm delivers to the grid. Typical velocity profiles of wind and water currents result in another advantage of TUSK over AWE systems. Ocean currents have their highest velocities near the ocean surface while winds are slow near the earth, as seen in figure 4. This allows a TUSK system to operate with a lower angle between the horizon and the tether, resulting in lower cosine losses. Cosine losses were described in equation 1.7 earlier in this section.

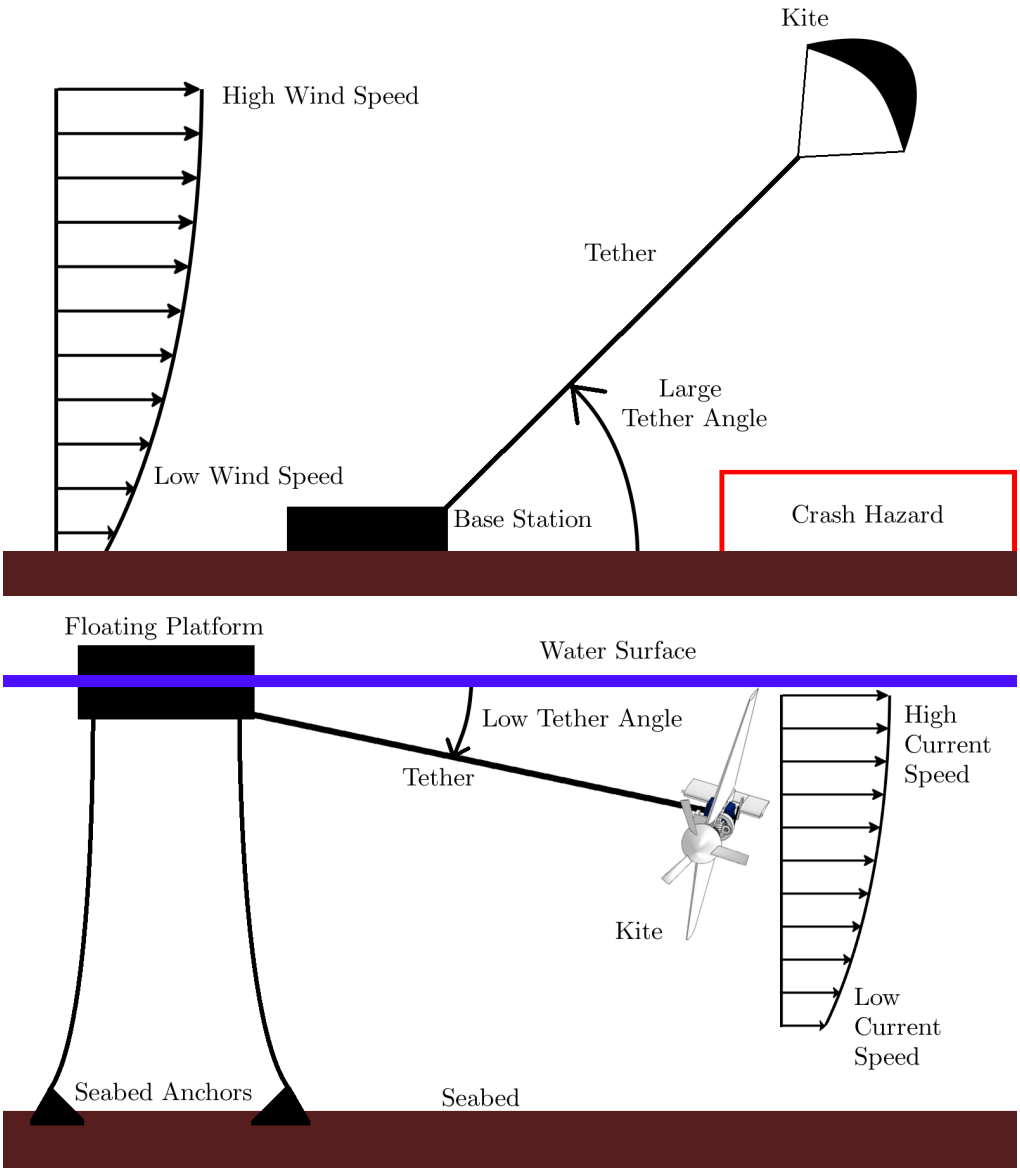


Figure 4: Comparison of velocity profiles and tether angles of AWE and TUSK systems.

1.4 Possible Disadvantages of TUSK

Given the corrosive nature of seawater operating environments, material selection for various TUSK system components is important. Materials will have to be lightweight, water resistant, and easy to manufacture to make TUSK systems practical. Controlling the complex figure 8 or circular kite motions necessary to achieve high power output will

be important, and kite control systems have been identified as the enabling technology for AWE and TUSK systems [6–9]. Any system installed long term in the ocean has to be designed to withstand storms. Most of the damaging wave action due to a storm occurs at the surface, potentially affecting the platforms more than the kites, which can remain submerged in relative safety. The array of platforms associated with a surface-anchored TUSK farm would have to be sited as to not interrupt existing shipping lanes, but kites with submerged anchors could operate far enough below the surface for shipping to pass harmlessly overhead. In the high hydrodynamic loading situations that TUSK devices would be designed to operate at, there exists the possibility of cavitation on various lifting surfaces. The effects of this phenomenon and a method for its prediction have been studied by Wang and Olinger [10].

Negative impacts on marine life could also occur during the operation of TUSK farms. Kites and high tension tethers moving through the water at several meters per second, carrying turbines spinning at hundreds of RPM, could present hazards to sea life swimming through the area where these kites are deployed. Acoustic disruptions will also be generated by the operation of kites, also impacting marine life. Farm sites need to be chosen with care to mitigate these environmental effects.

1.5 Previous Work

Kites as a power source are not a new idea. Loyd first studied [4] and patented [11] the concept of crosswind kite power in 1980 and 1981. His work set the stage for various theoretical studies of the dynamics of AWE kite systems [6, 12–14], and the availability of global wind resources [15–17].

Most of the work done since then has been focused on airborne kites, of which there are a variety of concepts currently undergoing development. Representative of the GroundGen category is EnerKite's [18] system. Their system has been demonstrated

operating autonomously during adverse weather conditions, indicating the feasibility of real world deployment. However, tests have shown that the materials involved in the construction of the kite have a lifetime of only about 100 hours. Research by this firm continues to pursue a solution for this problem. SkySails' [19] AWE system is aimed at providing tractive effort for surface ships as well as electricity generation. Tests of their hardware have shown that a kite designed for a load of 320 kN translates into 60-70% of the normal cruising power requirement of a typical 130 m long marine vessel. A novel aspect of their solution is the launch and recovery system for the kite. In a shipboard application, an extendable arm is mounted near the bow of the vessel. This arm can retrieve the reefed kite from a compartment below the deck and hold it in such a way that it can inflate and catch enough wind to take off. The process can be reversed to stow the kite back below decks. Makani Power [20] has enjoyed recent success with their rigid wing FlyGen design. Their prototype 7, flown from 2011 to 2013, produced an average of 20 kW that peaked at 30kW at a measured wind speed of 10 m/s. Makani has indicated that the measured performance of their system is above what analytical models predict, which has been attributed to wind shear between the sensor on the ground and the location of the wing. This shear was not directly measured during their testing. Because this system is a rigid wing with onboard turbines and generators it gains the capability to be self-launching by feeding power to the generators, which spin the propellers to lift the wing off the base station and begin forward motion. The above work on AWE systems is only a sampling of the work currently being undertaken in the field. A wide variety of configurations have been tested and their performance characterized in [21–23].

While airborne kites have been studied extensively, TUSK systems have rarely been studied. Landberg patented the concept of a crosscurrent underwater kite [3] in 2007. That patent became the basis of the Deep Green system being developed by Minesto AB [24], which has been supported by further research [25, 26]. Deep Green is designed

to operate in currents of 1.2 - 2.5 m/s, depths between 60 and 120 m and produce about 90MW with a 100 device farm. They have not yet reached that level of output, but tests have been performed in a real ocean environment since 2011 and a sub-scale prototype began testing in 2013 [27].The similarity in concept of the Deep Green design to the work in this thesis led to using Minesto's DG-8 kite of 8 meters span as a point of comparison. The details of the comparisons made are presented in section 2.3. Another TUSK system is being developed by HydroRun [28]. The kites developed by this company do not have turbines attached, but rather operate using the GroundGen model. Their website indicates that these are aimed at river sites and will be able to avoid obstacles with the help of sonar and cameras on board each kite. Performance is advertised as a 40 kW output at 2.5 m/s of current speed, with an operating range of 1.5 - 8 m/s of current speed. However, this design does not leverage the advantages intrinsic in attaching a turbine to the kite. One possibility of the reasoning behind this decision was eliminating the high speed turbine blades to reduce the danger to river-dwelling marine life.

References [9,10,29,30] on TUSK systems have been recently published or submitted by our research group at Worcester Polytechnic Institute. Also of interest to this work is an undergraduate project completed concurrently with the efforts here [31]. The aim of the undergraduate work was the preliminary design and fabrication of a scale-model TUSK kite and the results are discussed in section 2.4.

1.6 Project Goals and Contribution

The work carried out here is an early step towards a higher level of testing performed in the various projects on AWE systems. Future work will eventually attain results similar to [32], with similar test setup and control systems.

The main goals of the project work presented here were to

- Improve an existing scale-model TUSK system design [31]

- Conduct scale model experiments of a TUSK system in a water flume
- Obtain experimental data, including kite power output and trajectory
- Provide a preliminary scale-model TUSK system for future work including kite control system studies

Most of the testing work on TUSK systems has not yielded published results, most likely due to intellectual property concerns. This work is one of the first known that aims to publicly publish test results and performance data. Other work being undertaken by graduate students at WPI includes studies of kite control and trajectory optimization schemes, and the model developed in the course of this work is intended to serve as a platform for future testing of those schemes.

2 Methods

2.1 Outline

This methods section presents the procedures and techniques used to conduct scale-model underwater kite testing at Alden Research Labs in Holden, MA. The design of the scale-model TUSK system is presented first, followed by a discussion of scaling laws and techniques used to process data after collection.

2.2 General System Overview

Overall, the test system consists of five main subsystems: the kite, the turbine, the tether, the gimbal, and the electronics. Each of these subsystems will be discussed in more detail below. Figure 5 shows the entire underwater kite system.

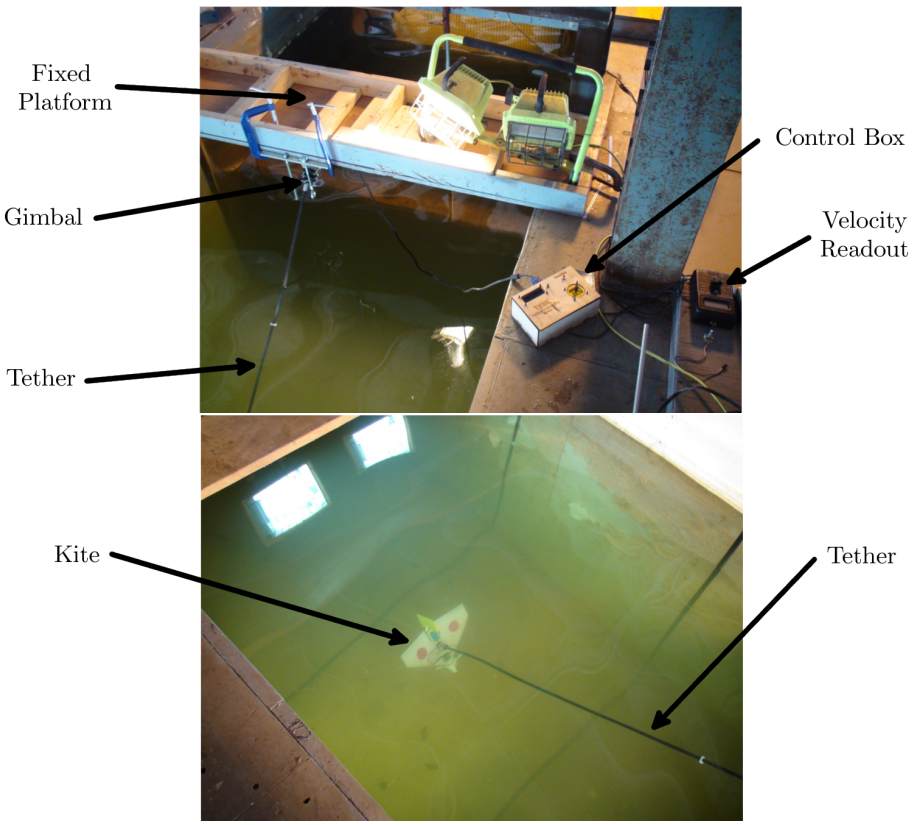


Figure 5: Installed system configuration in the Alden Research Lab 6ft x 6ft water flume.

The gimbal and electronics were located above the water mounted to a structure fixed relative to the flow. The tether extended from the gimbal into the water to hold the kite and turbine components.

Table 1: Important Geometric Dimensions for Scale-Model TUSK Testing

Dimension	**Symbol**	**Value**
Tether Length	L_t	1.969 m
Tether Diameter	D_t	0.012 m
Kite Wing Area	A_{kite}	0.047 m^2
Kite Root Chord	-	0.224 m
Kite Tip Chord	-	0.068 m
Kite Span	b	0.382 m
Turbine Blade Radius	R	0.074 m

Table 1 shows the most important geometric dimensions of the scale-model system

used for testing.

2.3 Model Scaling

To provide a comparison to a real-world application, scale factors were calculated for several important parameters of the model: weight, current speed, and power output. The Deep Green DG-8 from Minesto was the point of reference for this exercise, referred to as the prototype with P subscripts. Subscript M refers to the scale-model kite. From publicly available data on Minesto's website [33], the DG-8 weighs 2 metric tons, has a wingspan $b_P = 8$ m, and generates 120 kW for a current range of 1.2 - 1.8 m/s. Normally, scaling of winged vehicles references the chord length of the wings to decide on a scale factor. However, the elliptic planform shape of the Minesto kite is different from the straight tapered planform shape of the model developed here. No detailed data is provided on the chord distribution of the Minesto wing, so finding an average chord to compare to the model was not feasible. Therefore, span b was used as a scaling parameter. The scaling factor was then found by

$$SF = \frac{b_P}{b_M} = 20.9 \tag{2.1}$$

This factor was then employed in various equations to evaluate how closely the as-tested model matched the prototype. A series of targets was developed by scaling the prototype down to the model. How the model compared to these targets is discussed in section 3.6.

The first parameter scaled was the current speed. To find this scale factor, the relationship between hydrodynamic and buoyancy forces was used as a starting point, since these are the two main forces acting on the kite during its motion.

$$\frac{F_{hydro}}{F_{bouy}} \propto \frac{V_{current}^2 b^2}{b^3} = \frac{V_{current}^2}{b} \tag{2.2}$$

This leads to the following relationship

$$\frac{V^2_{current_P}}{b_P} = \frac{V^2_{current_M}}{b_M} \tag{2.3}$$

$$V_{current_P} = V_{current_M}\sqrt{\frac{b_P}{b_M}} = V_{current_M}\sqrt{SF} \tag{2.4}$$

Due to conditions discussed in section 2.9, the baseline test current speed, $V_{current_M}$, was set to 0.46 m/s. Plugging in the scale factor shows that the prototype kite current speed is $V_{current_P} = 2.1$ m/s. At that speed, the DG-8 is advertised to produce 120 kW [33]. Scaling the power began with the general power equation, introduced as equation 1.3 and reproduced here.

$$P_{kite} = \frac{1}{2}\,\rho\,A\,C_P\,V^3_{kite} \tag{2.5}$$

A major assumption applied in this situation was that both the model and prototype shared the same power coefficient, or C_P. That allowed for rewriting the equation as

$$\frac{P_M}{\frac{1}{2}\rho_M V^3_{kite_M} A_M} = \frac{P_P}{\frac{1}{2}\rho_P V^3_{kite_P} A_P} \tag{2.6}$$

and simplifying to

$$P_M = P_P \frac{\rho_M}{\rho_P}\left(\frac{V_{kite_M}}{V_{kite_P}}\right)^3 \frac{A_M}{A_P} \tag{2.7}$$

Both the model and prototype operate in water, hence the density ratio $\frac{\rho_M}{\rho_P}$ reduces to 1. The only data known about the prototype that could be related to the model were the wingspan and power output. Therefore, the velocity and area terms were be changed to powers of length using

$$\frac{V_{current_M}}{V_{current_P}} = \left(\frac{b_M}{b_P}\right)^{\frac{1}{2}} \tag{2.8}$$

obtained from 2.2 and

$$\frac{A_M}{A_P} = \frac{b_M^2}{b_P^2} \tag{2.9}$$

which resulted in

$$P_M = P_P \left(\frac{L_M}{L_P}\right)^{\frac{3}{2}} \frac{{L_M}^2}{{L_P}^2} \tag{2.10}$$

which simplifies to

$$P_M = P_P \left(\frac{1}{SF}\right)^{3.5} \tag{2.11}$$

The target power output for the model was then found by plugging the prototype parameters and scale factor into equation 2.11.

$$P_M = 1.2 \times 10^5 \; Watts \left(\frac{1}{20.9}\right)^{3.5} = 2.875 \text{ Watts} \tag{2.12}$$

Since weight scales by length cubed, the weight target is related to the prototype weight by the cube of the scale factor

$$\frac{W_P}{W_M} = SF^3 \tag{2.13}$$

then

$$W_M = \frac{W_P}{SF^3} = \frac{2000 \text{ kg}}{20.9^3} = 0.219 \text{ kg} \tag{2.14}$$

Proper weight scaling is often difficult to achieve in scaled models with large scale factors SF. The scale-model kite weight was 0.43 kg, which is nearly twice the predicted weight of 0.219 kg. In this particular model no concerted effort was made to keep weight down, so there is room for improvement. However, construction via 3D printing can only go so far in terms of weight reduction, so future efforts may have to switch manufacturing methods to match weight predictions.

In summary, the various calculated scale parameters are given in table 2.

Table 2: Scale Factor Summary

Variable	Model	Prototype	Scale Factor: Prototype / Model
Span b	0.382 m	8 m	$SF = \frac{b_P}{b_M}$
$V_{current}$	0.46 m/s	2.1 m/s	$\sqrt{SF}$
Power	2.875 W	120 kW	$(1/SF)^{3.5}$
Weight	0.430 kg	2000 kg	SF^3

2.4 Kite Design

As mentioned in the introduction, a prior undergraduate student team developed a design for a TUSK test kite. Their design served as a starting point for this work, but some significant improvements were made to the configuration. One driving tenet of the undergraduates' design was the assumption that the generator had to operate in a water-free environment, and the resulting nacelle and sealing features made the kite large and heavy. One major improvement in the new kite was the removal of this nacelle upon finding out that the generator did not actually have to be waterproofed. Also, the type of generator was changed to one that requires less torque to start spinning. Figure 6 shows the configuration at the end of the previous team's work.

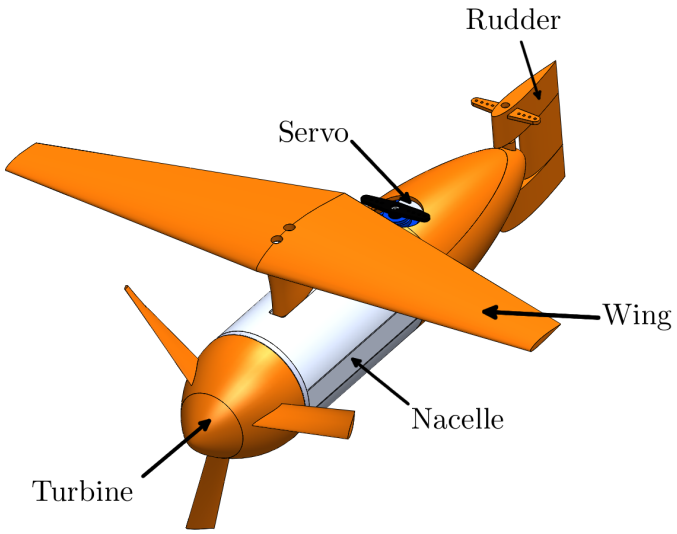


Figure 6: Kite design produced by undergraduate student team.

After redesign, figure 7 shows the final configuration of the kite.

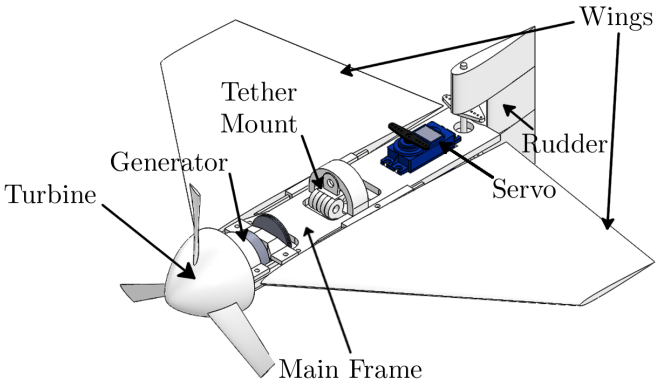


Figure 7: New scale model kite design used for testing.

The main frame was the base component that all of the other components were attached to. Its major features are shown in figure 8.

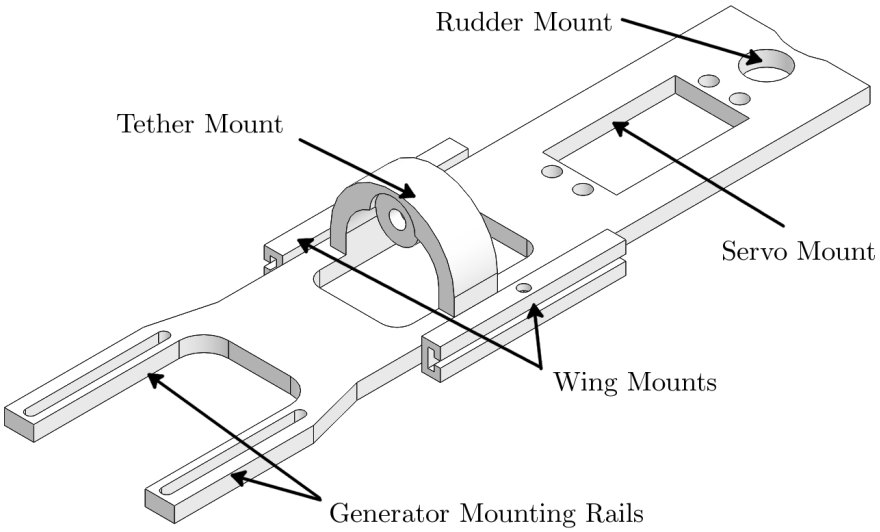


Figure 8: Main kite frame configuration.

Electrical generation was provided by a brushed DC motor, sold as the Wind Energy Generator from Kid Wind [34]. The generator was mounted at the forward end of the frame between a pair of slotted rails. It was clamped in place by half-round hoop like structures that grip the housing. The turbine was attached to the generator shaft with a standard propeller collet found on radio control aircraft propellers, as seen in figure 9. [35]

Figure 9: Propeller collet for attaching turbine to generator shaft.

Rudder control was provided by a HiTec HS-5646WP [36], seen in figure 10 [37] which is a commercial off the shelf waterproof servo, mounted in a cutout at the aft end of the frame. The rudder itself was mounted just behind the servo in a block of low-friction UHMW plastic acting as a bearing for the rudder shaft.

Figure 10: Radio control servo for controlling rudder angle.

The pitch and roll angles with respect to the tether were set by the tether mounting arrangement partially integrated into the main frame. This allowed the pitch and roll with respect to the tether to be locked in place during testing. Because the kite was firmly attached to the end of the tether, the entire tether rotates along with the kite during yawing motions. The tether was supported by bearings at the gimbal, as described in section 2.6. Wing attachment was provided by a pair of tongue and slot arrangements along the sides of the frame. The intention of this design was to allow for adjusting the wings fore and aft to correct any balance issues that may have arisen or replacement of the wings altogether.

The 1.83 meter width of the Alden testing site constrained the wingspan of the model to $b_M = 0.382$ m. Due to the width and depth of the water channel any turns the kite made were restricted to a certain radius. If the wingspan extended too far towards the center of the turn, the effective velocity at the inner wingtip would be very low. Since it was impossible to change the size of the test channel, a limit was placed on kite wingspan. To maximize the wing area provided by a limited span with a reasonable aspect ratio, a delta planform was chosen. A symmetrical NACA 0005 airfoil was chosen for the initial prototypes, because of its ease of manufacture and integration with the mounting system design. Airfoil thickness was chosen for similar reasons, allowing adequate room for structural elements at the root of the wing. The root chord of the wing is 0.224 m

and the tip chord is 0.068 m. The basic planform of the kite is displayed in figure 11.

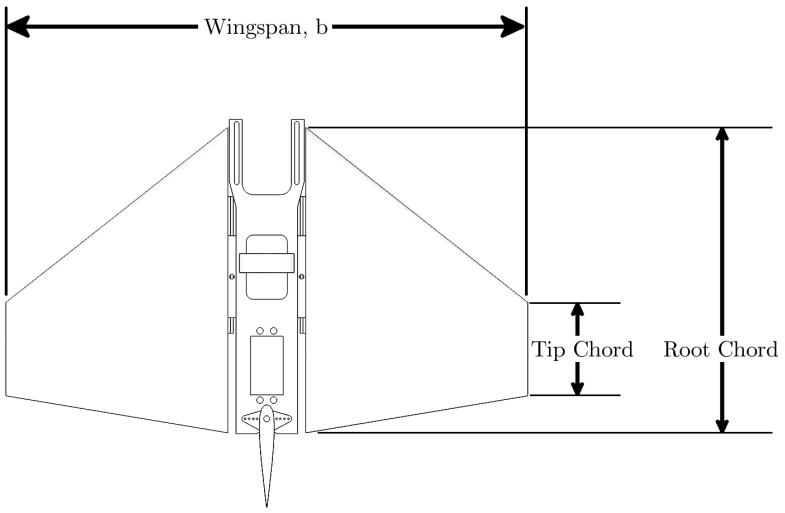


Figure 11: Kite wing planform view showing overall wing dimensions.

The tether itself was a hollow carbon fiber tube of length $L_t = 1.969$ m with an outside diameter $D_t = 1.27$ cm. Electrical wiring for the servo and generator were run through the inside.

2.5 Turbine Design

Designing the turbine was one of the first tasks undertaken in the overall design process. It was assumed that testing would occur at the flume's highest speed of 1 m/s. However, as described in section 2.9 the actual test speeds did not reach this value. Matching the turbine design to the final current speed is a subject for future work.

The flow velocity at the turbine V_t was assumed to be equal to the velocity of the kite $V_t = V_{kite}$. In addition to assuming a current speed, an assumption of the kite lift to drag ratio $\frac{L}{D}$ was also required. This parameter was not known at the time of turbine design, but early configurations were not expected to perform well aerodynamically. Thus, a value of $\frac{L}{D} = 4$ was chosen for the kite. Using this information, the speed of the kite and

therefore the flow through the turbine, could be found with equation 1.2, reproduced here

$$V_{kite} = \frac{2}{3}\frac{L}{D}V_{current} = 2.7\ m/s \tag{2.15}$$

The manufacturer of the generator [38] indicates that 1000 RPM is the most efficient rotational speed for power generation. Thus, that value was chosen for the target turbine RPM. Manufacturing considerations limited the airfoil choices to thin low camber options. Under these considerations, a NACA 0015 was chosen for the root and a NACA 0012 was chosen for the tip, with a linear interpolation of thickness along the blade span. These airfoils are illustrated in figure 12.

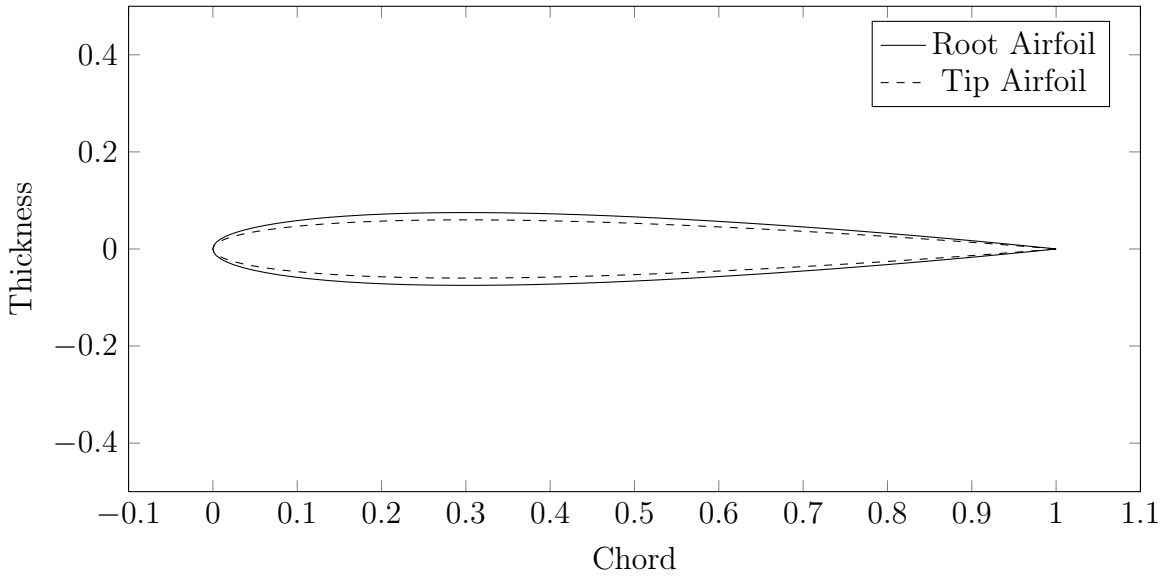


Figure 12: Comparison of root and tip airfoils used in the turbine.

In terms of airfoil performance, a lift coefficient C_l of 1 at an angle of attack α of 7 degrees could be reasonably assumed to exist along the entire blade length. Twist angle at the tip of the blade θ_{p_0} was set to zero. The radius R of 0.074 m was carried over from the previous undergraduate kite design [31], giving a turbine area of 0.0172 m^2. A three bladed configuration, where $B = 3$, was chosen due to its polar moment of inertia

characteristics. When experiencing yawing motions, a three bladed rotor experiences less reaction torque counteracting the yawing motion as compared with smaller or larger numbers of blades. This reaction is irrespective of the turbine's azimuthal position, making the response constant as the turbine spins. By following the process for designing an ideal turbine rotor with wake rotation for optimum power output as given in Manwell et al [39], the chord and blade twist distribution could be determined. The process began with finding the angular velocity of the turbine ω in radians per second, resulting in $\omega = 104.7$ rad/s. Next, the tip speed ratio was calculated with

$$\lambda = \frac{\Omega\, R}{V_{kite}} = 2.869 \tag{2.16}$$

Once the parameters that applied to the entire turbine were found, more relevant information could be calculated at intervals along the blade span. For this turbine, each parameter was found at 10 locations $\frac{r}{R}$ along the blade from 0.1 to 1.0, in increments of 0.1. Angles are defined between blade chord and the rotational axis, with the rotational axis being 0°. This is illustrated in figure 13.

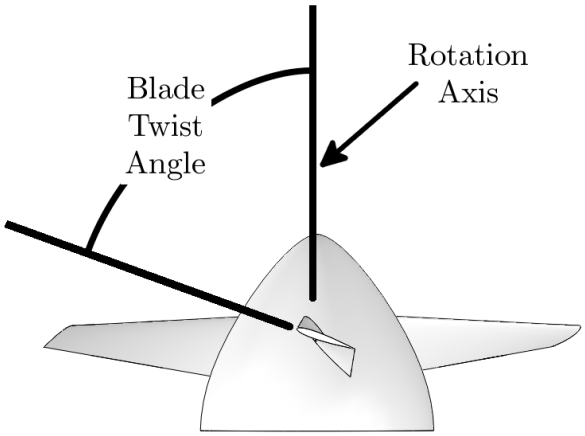


Figure 13: Illustration of turbine blade twist angle reference.

The first quantity was the blade section radius r, which was found with $r = \frac{r}{R} \cdot R$. Local tip speed ratio λ_r was calculated with

$$\lambda_r = \lambda \frac{r}{R} \tag{2.17}$$

The angle of relative wind Φ_{wind} represents the angle at which the apparent velocity would be striking a blade with zero angle of attack [39].

$$\Phi_{wind} = \frac{2}{3} \arctan\left(\frac{1}{\lambda_r}\right) \tag{2.18}$$

The chord length distribution of the blade [39] was found with

$$c_r = \frac{8\pi r(1 - \cos(\Phi_{wind}))}{B\, C_l} \tag{2.19}$$

Section pitch angle was found with [39]

$$\theta_p = \Phi_{wind} - \alpha \tag{2.20}$$

and blade twist angle was found with [39]

$$\theta_t = \theta_p - \theta_{p_0} \tag{2.21}$$

Combining the above analysis with the earlier airfoil choice enabled final design of the turbine's chord length and blade twist angle θ_t distributions along the blades.

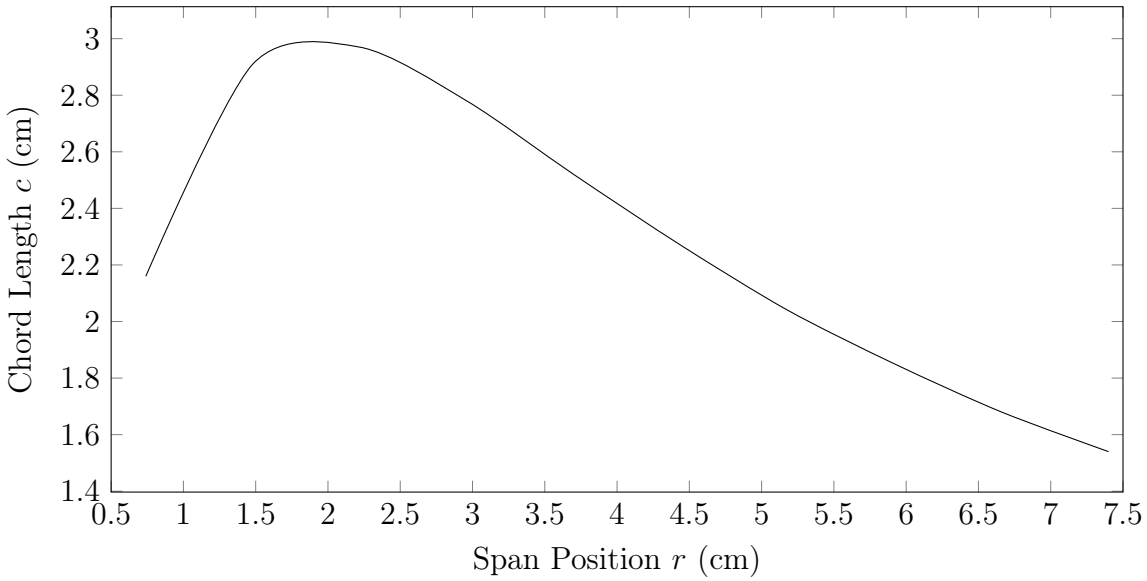


Figure 14: Chord distribution of scale-model TUSK turbine blades.

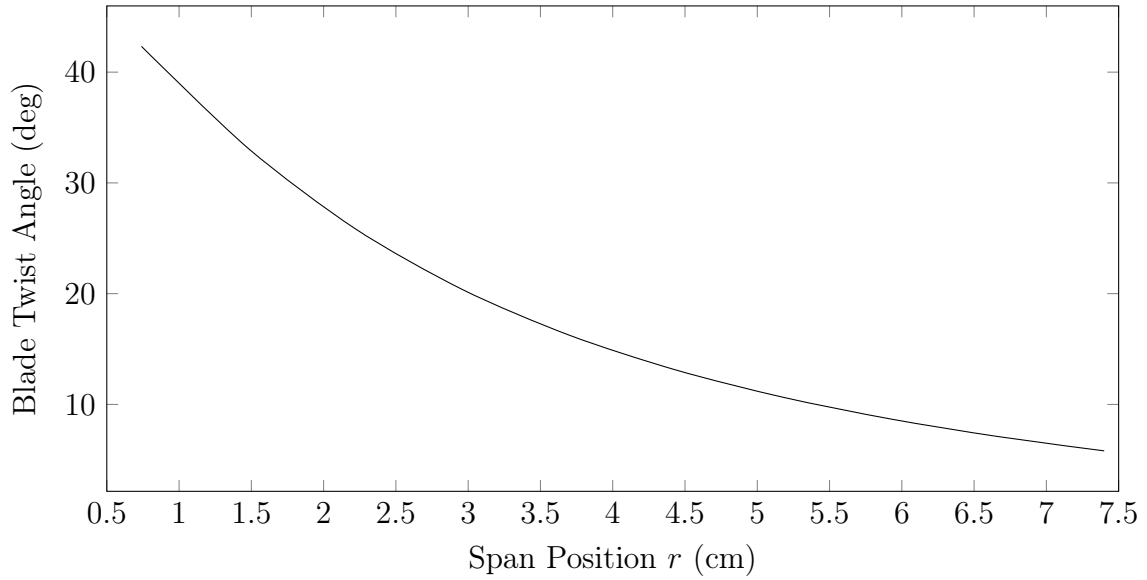


Figure 15: Twist angle distribution of scale-model TUSK turbine blades.

A summary of the design parameters of the turbine is presented in the table 2.5.

Table 3: Initial Turbine Design Specifications

Parameter	**Description**	**Value**
V_k	Kite Apparent Velocity	2.7 m/s
B	Number of Blades	3
$\theta_{p,0}$	Blade Twist Angle at Tip	0 deg
ω	Revolutions per Minute	1000
R	Turbine Radius	0.074 m
α	Airfoil Section Angle of Attack	7 deg
C_l	Airfoil Section Lift Coefficient	1

2.6 Gimbal Assembly

The device used to support the end of the tether above the water was a gimbal from a previous airborne kite project designed and constructed by Arruda [40]. It used potentiometers to measure the tether declination θ and azimuth Φ angles. Modifications had to be made to pass electrical connections through the rotating tether, so a slip ring was added. This required some redesign of the existing components to accommodate the slip ring, which was performed by Mello [41]. A new cross-member and slip ring bracket were manufactured to integrate the new parts into the existing assembly. Connecting the kite to the gimabl was a carbon fiber rod 1.969 meters in length. Clamping shaft collars were employed to retain the tether in the gimbal and retain the kite on the free end of the tether. The radial loads from the tether were held by a standard ball bearing pressed into the center block behind the slip ring. The mechanical limits of the gimbal are shown below.

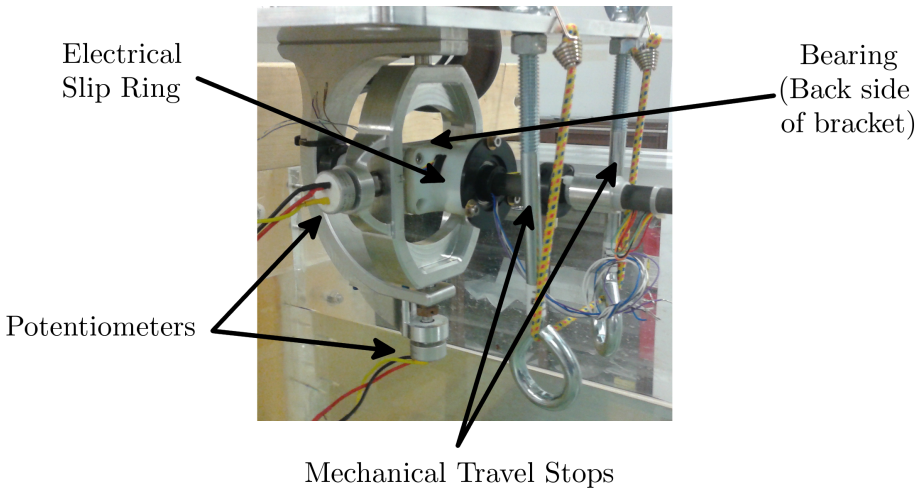


Figure 16: Construction details of support gimbal.

Table 4: Gimbal Mechanical Limits

Travel Direction	Minimum	Maximum
Declination	9.5°	34°
Azimuth	-19.5°	19.5°
Tether Rotation	Unlimited	Unlimited

2.7 Kite Control and Data Acquisition

One of two main functions of the electronics and associated software was to provide a manual control system for the kite. As described in section 2.4, the rudder was controlled by a standard radio control servo. These use the principle of pulse width modulation (PWM) to control their position. Figure 17 represents a standard servo PWM signal, with time on the horizontal axis and voltage on the vertical axis. The lowest voltage is 0 volts, and the highest is 5 volts. Making the high pulses longer commands the servo in one direction, and shorter pulses command it in the opposite direction. The processor was programmed to take input from a joystick, and convert it to a signal to be sent to the rudder servo to change its position and change the yaw angle of the kite.

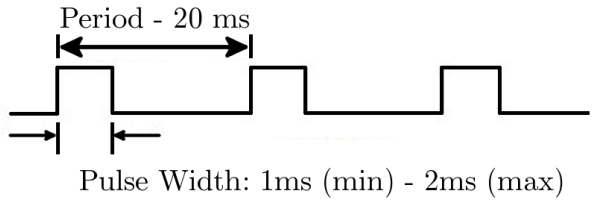


Figure 17: Illustration of pulse width modulation signal.

Also included in the system was a display screen to read system status. During data recording, the parameters being written to the memory card, the voltage of the batteries powering the whole system, and the file name currently being written were all displayed. When not recording, a menu system activates. These menus allowed for calibrating the end points of the servo travel and setting the name of the file to record next. The menu system was controlled from the same joystick that controlled the servo during data collection mode, and a simple push button was used to switch between modes. An overview of this system is provided in figure 18.

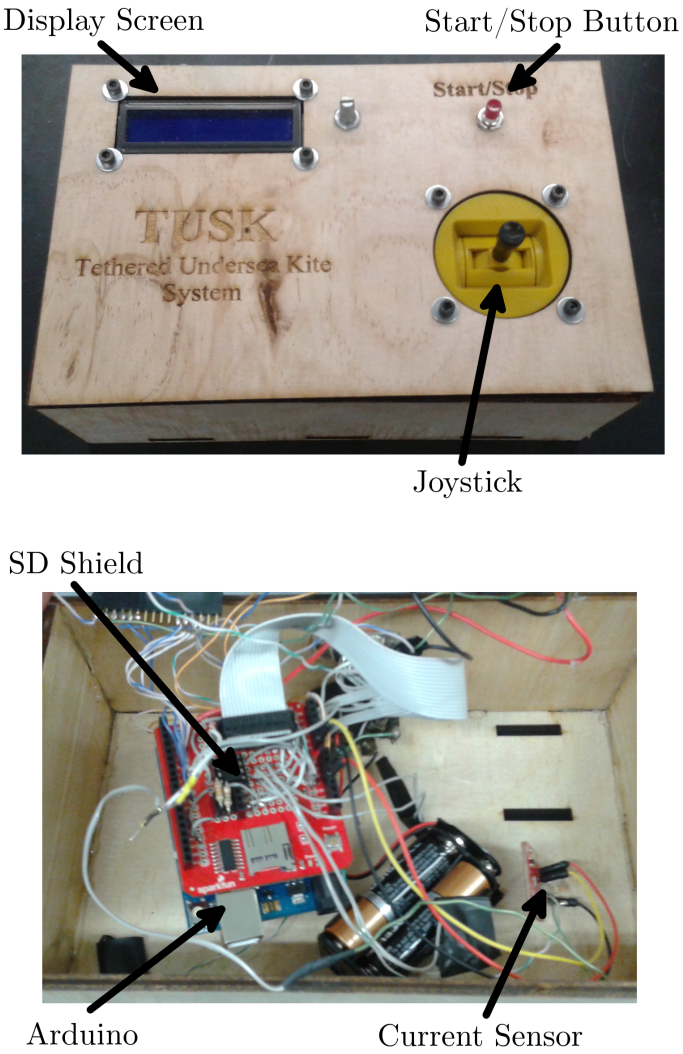


Figure 18: Overview of hardware used for data collection and kite control.

The main electronic components required to operate the kite and collect data were an Arduino Uno [42], a pair of potentiometers, a current sensor, and an SD shield [43]. Processing power for the whole system was provided by an Arduino circuit board. This particular system uses a 10 bit analog to digital converter, delivering a precision of 0.004 volts. Only analog sensors were used in this system, each delivering an output voltage between 0 and 5 volts. A pair of potentiometers mounted on the gimbal measured tether azimuth and declination. Tether rotation was not measured in the experiments. Since the mechanical motion of the gimbal did not reach the ends of the electrical travel of

the potentiometers, the signal from them had to be amplified. A Texas Instruments LM 358 op amp chip [44] mounted on the prototyping area of the SD shield performed this function before passing the amplified signal on to the Arduino itself to be recorded. Measurement of power from the motor was effected by a commercially available current sensing circuit board based on the INA169 microchip [45]. Using a series of resistors, this sensor converts current into a voltage that is then sent to the Arduino. Since the current is not measured directly, the relationship between current and voltage needs to be established. Documentation from the vendor provides this relationship as

$$I_S = \frac{V_{OUT} \cdot R_C}{R_S \cdot R_L} \tag{2.22}$$

where I_S is the current being measured, V_{out} is the voltage output to the Arduino, R_C is the constant internal resistance of the chip, R_S is the shunt resistor value of 10 Ω and R_L is the output resistor value of 10 $k\Omega$. Plugging in these values yields the relationship between measured voltage and current.

$$I_S = \frac{V_{OUT} \cdot 1k\Omega}{10\Omega \cdot 10k\Omega} \tag{2.23}$$

Now that the current is known, the power can be calculated using

$$P = I^2 \, R \tag{2.24}$$

where P is power, I is current, and R is resistance. The resistance of the circuit was found to be $R = 370$ Ω. Plugging the Arduino's measurement precision of 0.004 volts into equation 2.23 yields a precision of 5.92×10^{-7} watts for the power measurement. One of the benefits of the Arduino platform is its expandability, which was leveraged when designing the data acquisition portion of the hardware. This SD shield is a circuit board following a standard physical layout that plugs directly into the top of the Arduino. On

the shield is a slot for plugging in an SD card, which was where the data from the rest of the system was stored. This data was collected at a rate of one data point per 35 milliseconds for the full duration of testing.

2.8 Test Site Layout

The test site for the experiments was Alden Research Labs in Holden, MA. A water flume was used, measuring 1.83 meters wide and 1.83 meters deep, with a test section 3.7 meters long. The maximum test section velocity was $V_{current_{max}} = 1.0$ m/s, which was assumed to be uniform throughout the testing domain. Test hardware was mounted on wooden platforms extending across the width of the flume. Video cameras mounted in various locations around the test area, as seen in figure 19. One was on a tripod facing into the viewing window on the side of the flume, another was mounted to the gimbal facing downstream the water, and the third was a underwater GoPro was at about 1 meter depth facing upstream.

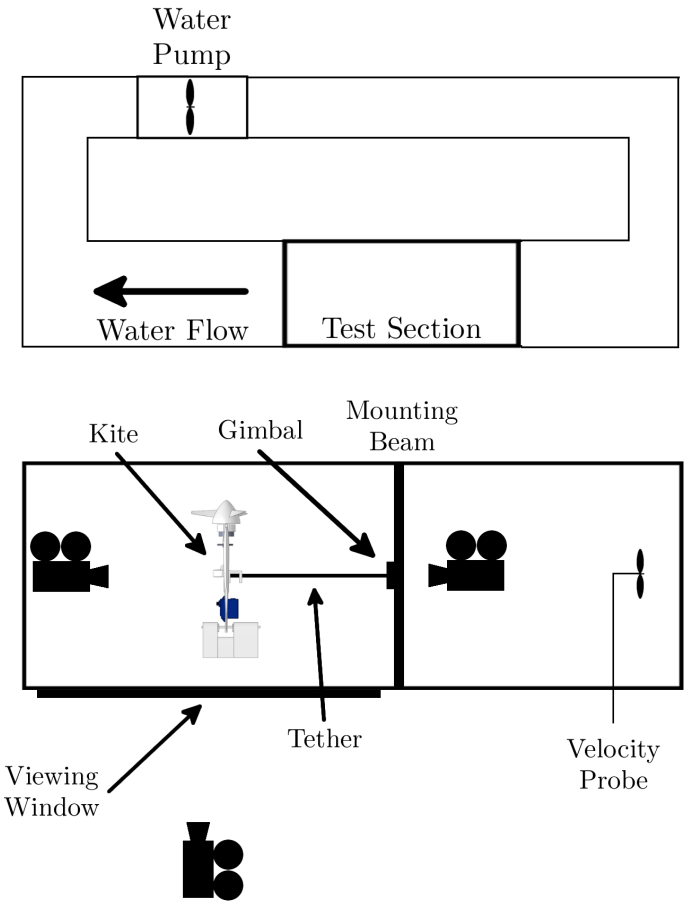


Figure 19: Schematic description of Alden test site flume configuration.

Photographs of the test site are included in figure 20. The viewing window seen in the second image is located to the right in the first image.

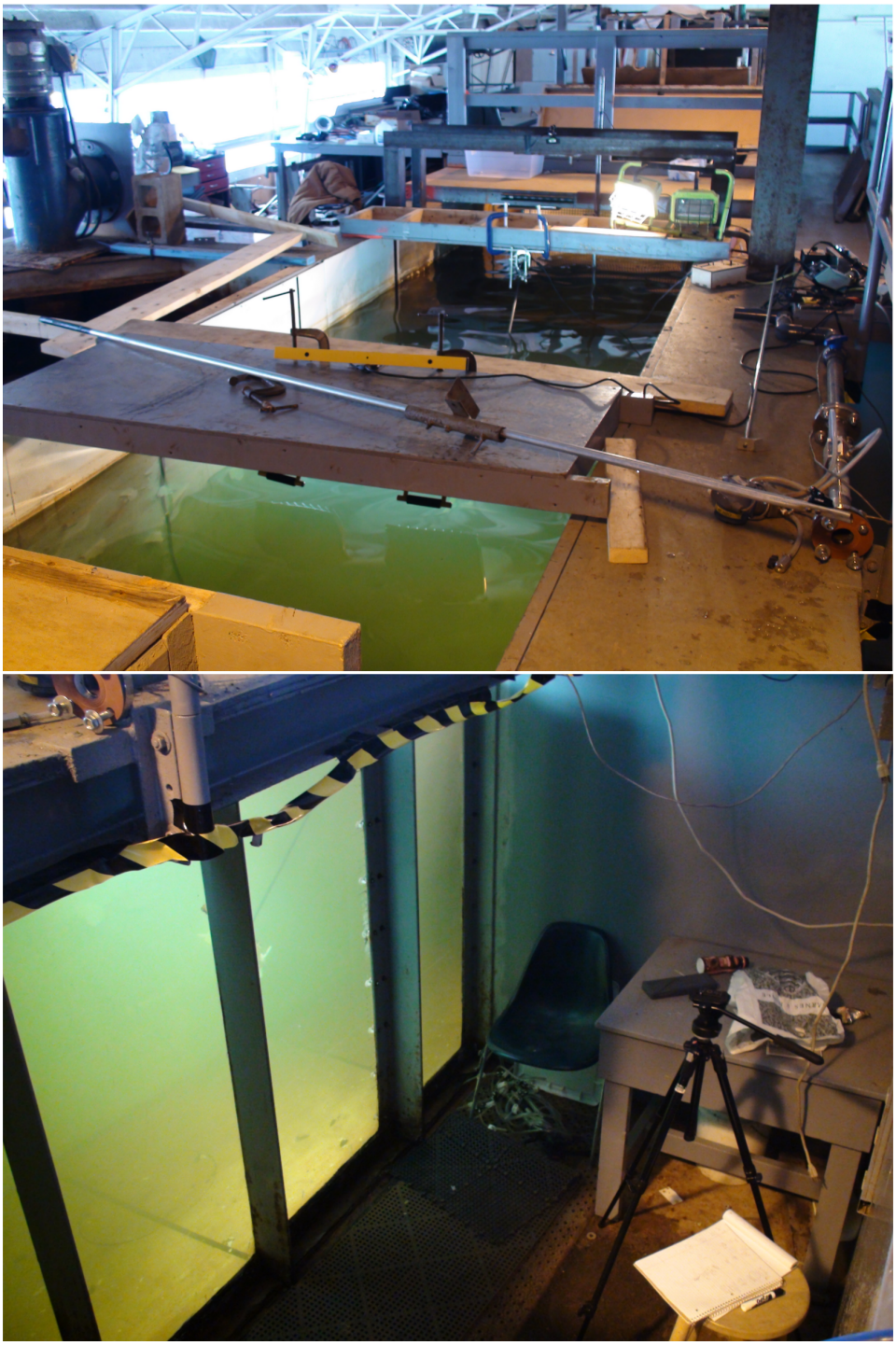


Figure 20: View of Alden test site flume and viewing window.

Flume velocity was measured with a Swoffer Instruments 2100-151 [46], with ± 0.003 m/s accuracy. This velocity probe consisted of a small impeller suspended from a rod

into the flow. This impeller was located at a depth of 0.6 meters and 1.2 meters upstream of the gimbal location on the centerline of the flume. Every ten seconds, the display screen was updated with an average of the velocity since the last update. The velocity was allowed to stabilize for at least two updates before trusting the velocity reading and conducting tests at a given condition.

2.9 Testing Procedures

A common set of steps was followed before and during every test condition.

1. Set flow velocity
2. Note conditions of upcoming test
3. Start cameras recording
4. Begin data recording
5. Push kite to begin motion
6. Maintain desired kite motion for at least 30 seconds
7. Stop data recording and cameras

As mentioned earlier, the flume can produce a $V_c = 1$ m/s. However, all testing was conducted at $V_c = 0.5$ m/s or less. At higher speeds, the structural integrity of the kite and tether became a concern due to visible deflections. Even though the kite was originally designed for $V_c = 1$ m/s flow speed, useful data could still be gained by running it at lower speeds. Three speeds were chosen to observe the effect on power output: $V_c = 0.15$ m/s, 0.31 m/s, and 0.46 m/s. The static pitch angle of the kite, ζ, was also varied between tests. The datum for this angle measurement was the centerline of the tether. When the longitudinal axis of the kite was perpendicular to the axis of the tether, the pitch angle was considered to be $\zeta = 90°$. Pointing the nose closer to the tether reduced this angle and pointing it away increased it, as seen in figure 21. Tests were run with pitch angles set of $\zeta = 80°$, $85°$, and $90°$.

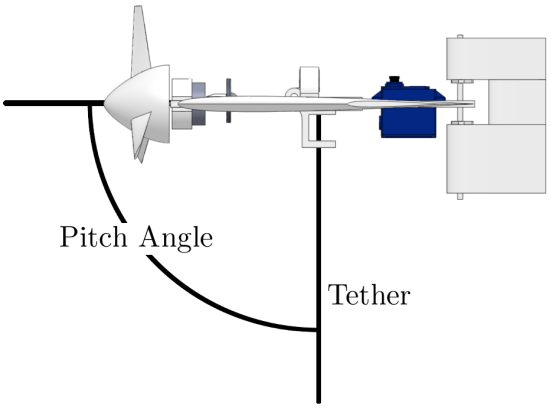


Figure 21: Illustration of kite pitch angle reference.

For each combination of pitch and speed, the kite was flown through a circular and figure 8 shaped path. This was done to investigate a possible change in performance dependent on the shape of the kite trajectory. For the two higher current speeds, the kite was held in position in the center of the flume with the nose pointing directly upstream. This was done to measure baseline power with no kite motion.

2.10 Data Post Processing

Data analysis was performed in MATLAB. The full, commented code used can be found in appendix 5.1. A baseline run with $V_c = 0.46$ m/s, $\zeta = 85°$ and a circular kite path will be used to discuss these techniques. The steps taken to post-process the collected data were:

1. Apply median filter to raw data
2. Apply coordinate transforms to match standard spherical coordinates
3. Transform spherical data to Cartesian for path visualization
4. Numerically differentiate tether angle data
5. Filter differentiated data
6. Compute velocity in spherical frame

Given the noise present in original data and noise introduced from numerical derivation, median filtering was employed with built-in MATLAB commands to smooth data for presentation and ease of analysis. The steps involved in median filtering, and an example of their application, are presented here.

1. Begin at first element in vector
2. Take sample size equal to window size
3. If sample extends beyond vector domain, add zeros
4. Sort values in sample from smallest to largest
5. Replace original value with the value in the center of sample
6. Move to next value in original vector

The above steps will now be demonstrated on the following vector, with a window size of 3.

$$\mathrm{A} = [5\ 7\ 4\ 6\ 10\ 2]$$

Taking the first element with a window size of 3 and padding empty positions with zeros:

$$0\ 5\ 7$$

The elements are already sorted from smallest to largest, so the median value is taken and added to a new vector.

$$\mathrm{F} = [5\ \ldots]$$

Moving to the next element of vector A gives:

$$5\ 7\ 4$$

Which sorts to:

$$4\ 5\ 7$$

The median of which is 5, generating the second value of the filtered vector.

$$F = [5\ 5\ \ldots]$$

The process is repeated for the third element of A.

$$7\ 4\ 6$$

$$4\ 6\ 7$$

$$F = [5\ 5\ 6\ \ldots]$$

When the operation is completed, the original and filtered vectors compare like so.

$$A = [5\ 7\ 4\ 6\ 10\ 2]$$

$$F = [5\ 5\ 6\ 6\ 6\ 2]$$

The effect of this filtering is demonstrated on baseline run data in figure 22, where it is observed that the filtering eliminates the sudden fluctuations (noise). This data has noise introduced from the numerical differentiation process, and is the best example of the effectiveness of this filtering.

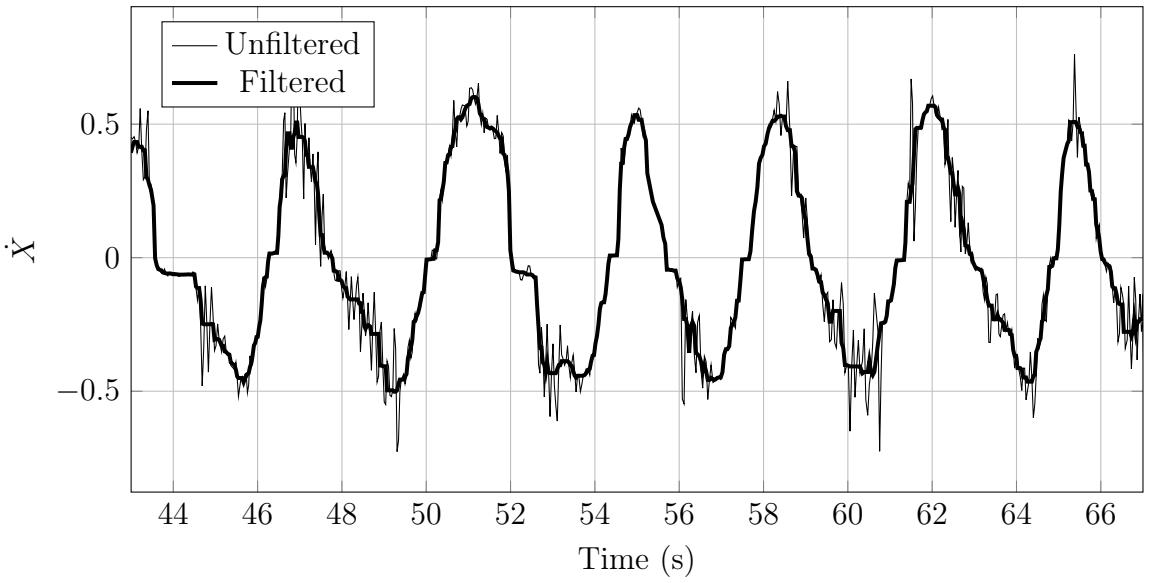


Figure 22: Illustration of median filter effect.

The tether declination and azimuth angles used in testing did not match up with the standard definition of spherical coordinates. Figure 23 shows the difference between the two systems.

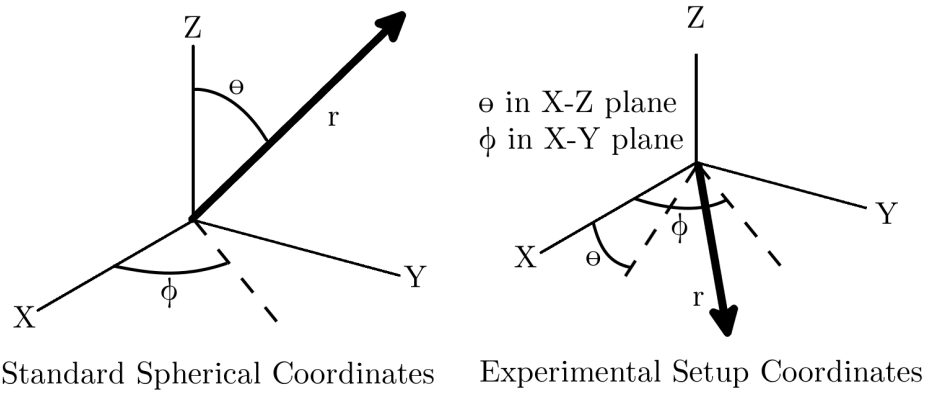


Figure 23: Illustration of differences between standard and experimental coordinate systems.

Transforms needed to be applied to the data before processing it with standard spherical coordinate equations. During testing, flow was along the positive x axis in standard spherical coordinates, and the positive y axis pointed at the viewing window

in the flume. The test equipment measured declination angle θ as zero in the x-y plane, while standard spherical coordinates have θ as zero along the positive z axis. Thus, 90 degrees had to be added to the measured data to transform it into the standard spherical system. The test equipment recorded azimuth angle Φ as positive towards the y axis, which corresponds with the standard definition. Radius r was fixed by the length of the tether for all tests as $r = 1.969$ m. Equations 2.25 and 2.26 show how the experimental data was modified before further analysis.

$$\theta_{Standard} = \theta_{Measured} + 90^{\circ} \tag{2.25}$$

$$\Phi_{Standard} = \Phi_{Measured} \tag{2.26}$$

After converting the angular data to the standard system, the kite position was transformed into cartesian coordinates for ease of plotting and visualization. The equations employed were

$$x_k = r\ sin(\theta)\ cos(\Phi) \tag{2.27}$$

$$y_k = r\ sin(\theta)\ sin(\Phi) \tag{2.28}$$

$$z_k = r\ cos(\theta) \tag{2.29}$$

where x_k, y_k, z_k is the kite tether mount position in cartesian coordinates. The components x_k, y_k, and z_k were then numerically differentiated to give the velocity components of the kite with respect to the gimbal, or the stationary laboratory frame. The standard central difference method was used on the majority of the data points, with forward and reverse difference methods applied near the ends of data vectors.

The water flow in the flume was assumed to be uniform throughout the domain of kite motion with a component only along the x axis. Thus, to convert the fixed laboratory

frame velocity to kite apparent velocity, the current flow velocity was added to V_{kite_x}. Once the addition was performed, total kite apparent velocity V_{kite} was found with

$$V_{kite} = \sqrt{V_{kite_x}^2 + V_{kite_y}^2 + V_{kite_z}^2} \tag{2.30}$$

Certain graphs and visualization methods worked best when the angular components of kite velocity were used as inputs. Even though the initial measurements were recorded in declination and azimuth angles, the angular velocity components were more conveniently obtained by converting the cartesian components into angular components. The equations used to transform into angular components are

$$\dot{r} = \frac{x_k V_{kite_x} + y_k V_{kite_y} + z_k V_{kite_z}}{\sqrt{x_k^2 + y_k^2 + z_k^2}} \tag{2.31}$$

$$\dot{\theta} = \frac{V_{kite_x} y_k - x_k V_{kite_y}}{x_k^2 + y_k^2} \tag{2.32}$$

$$\dot{\Phi} = \frac{Z(x_k V_{kite_x} + y_k V_{kite_y}) - (x_k^2 + y_k^2) V_{kite_z}}{(x_k^2 + y_k^2 + z_k^2)\sqrt{x_k^2 + y_k^2}} \tag{2.33}$$

Since the tether was a fixed length in all tests, $\dot{r} = 0$.

To accurately compare the power coefficient with other systems, equation 1.3 can be rearranged as

$$C_P = \frac{P}{\frac{1}{2}\,\rho\,V^3\,A_{turbine}} \tag{2.34}$$

where the kite area has replaced with the turbine area.

The performance of the kite with respect to the velocity cubed rule set forth by equation 1.3 was evaluated. By rearranging the equation as

$$\frac{P}{V^3} = \frac{\rho}{2}\,C_P\,A_{turbine} \tag{2.35}$$

and using the average C_P value for the whole run, the slope of the reference line can be found. The actual power data was then plotted on the same axes to compare.

As a method of roughly estimating the actual $\frac{L}{D}$ of the kite, equation 1.2 was employed, rearranged as

$$\frac{L}{D} = \frac{3}{2}\frac{V_{kite}}{V_{current}} \tag{2.36}$$

The actual speed and current speed were plugged in to find the $\frac{L}{D}$. The derivation of the original equation assumes that the kite is going at its optimal speed in a cross-current motion. However, generating a more accurate estimate of $\frac{L}{D}$ would involve either making direct lift and drag measurements or performing a detailed drag analysis of the design. Since the kite configuration was not optimized to reduce drag, the internal components were left exposed which created a situation that greatly complicated drag prediction. Equipment necessary for accurate drag measurement was not existant during this work and the development of such equipment was beyond the defined scope. Therefore, the described method provides the best estimate of $\frac{L}{D}$ found in the course of this project. Addressing these issues is a subject for future work.

3 Experimental Results

3.1 Baseline Circle

Presented here is a baseline case used for comparison to other test conditions. Table 5 shows important system parameters and test conditions, along with their values for the circular path baseline test.

Table 5: Baseline Circle Run Conditions

Parameter	**Value**
Current Speed	0.46 m/s
Kite Pitch Angle	85°
Desired Path Shape	Circle
Tether Length	1.969 m
Tether Diameter	0.012 m
Kite Wing Area	0.047 m^2
Kite Root Chord	0.224 m
Kite Tip Chord	0.068 m
Kite Span	0.382 m
Kite Weight	0.430 kg
Turbine Area	0.017 m

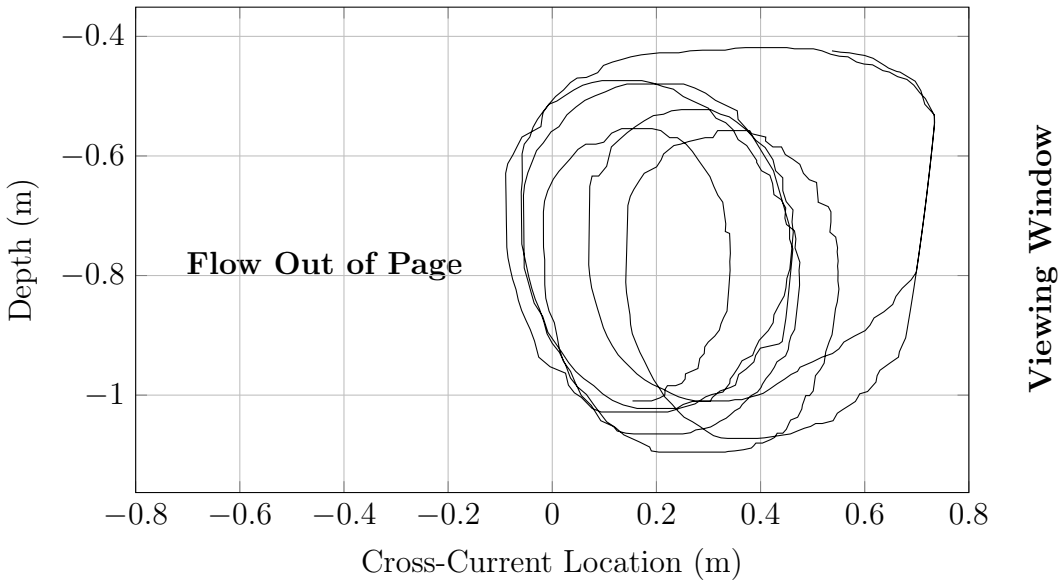


Figure 24: Downstream view of circular kite trajectory.

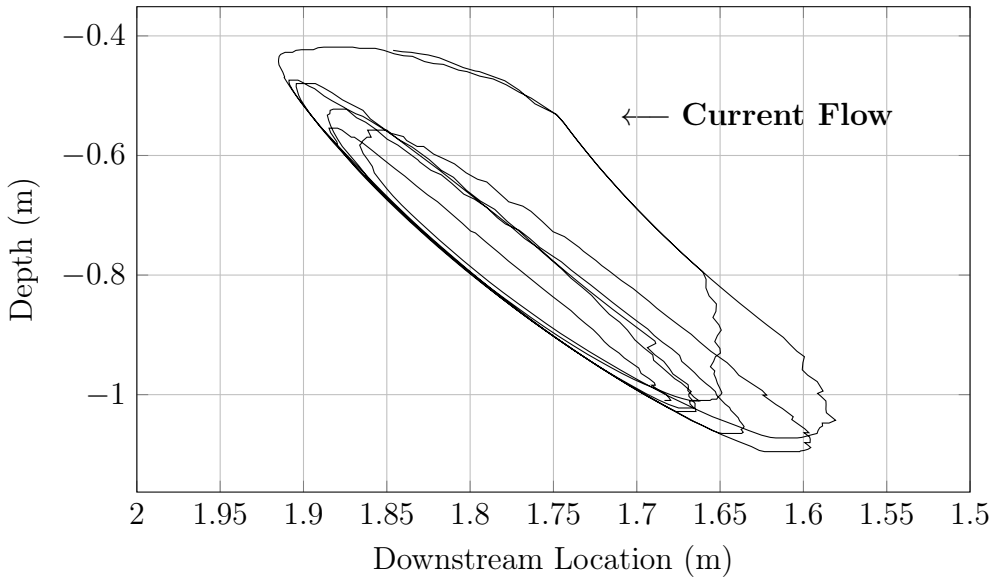


Figure 25: Window view of circular kite trajectory.

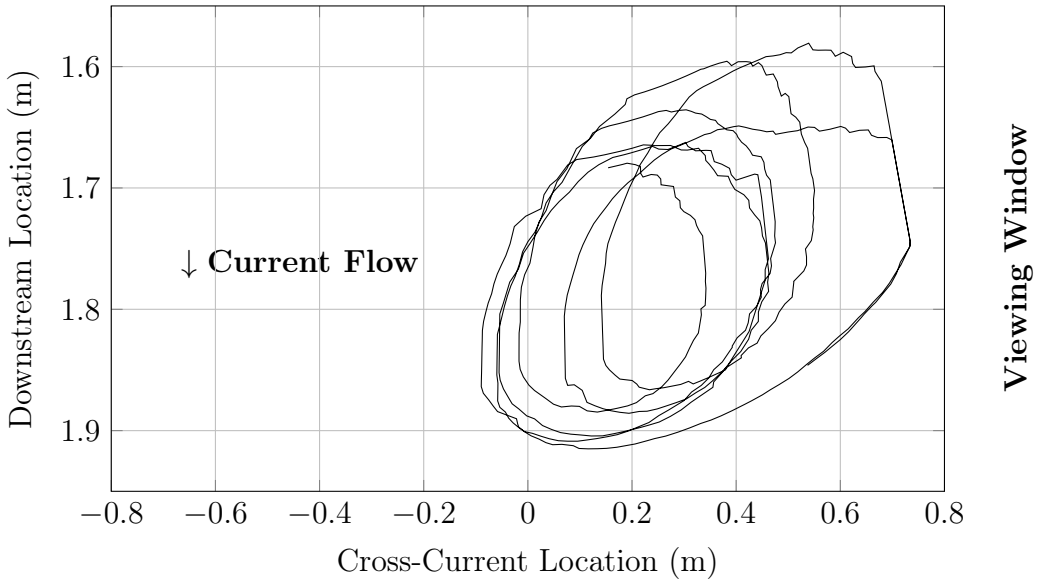


Figure 26: Overhead view of circular kite trajectory.

Figures 24, 25 and 26 show the kite's circular trajectory from three directions: downstream, the side viewing window, and directly overhead. The path shape variations are present because the trajectory was manually controlled by observing the kite's motion in real time. There was a definite tendency for the kite trajectory to favor the window

side of the flume, but its cause was not investigated as part of this testing.

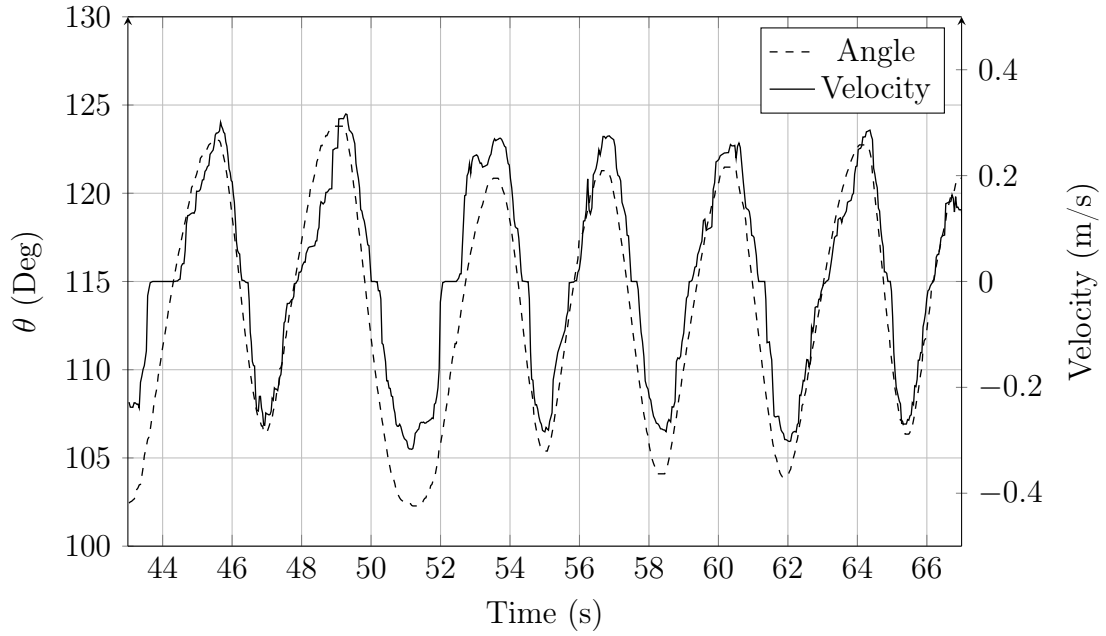


Figure 27: Declination angle and velocity in the gimbal reference frame.

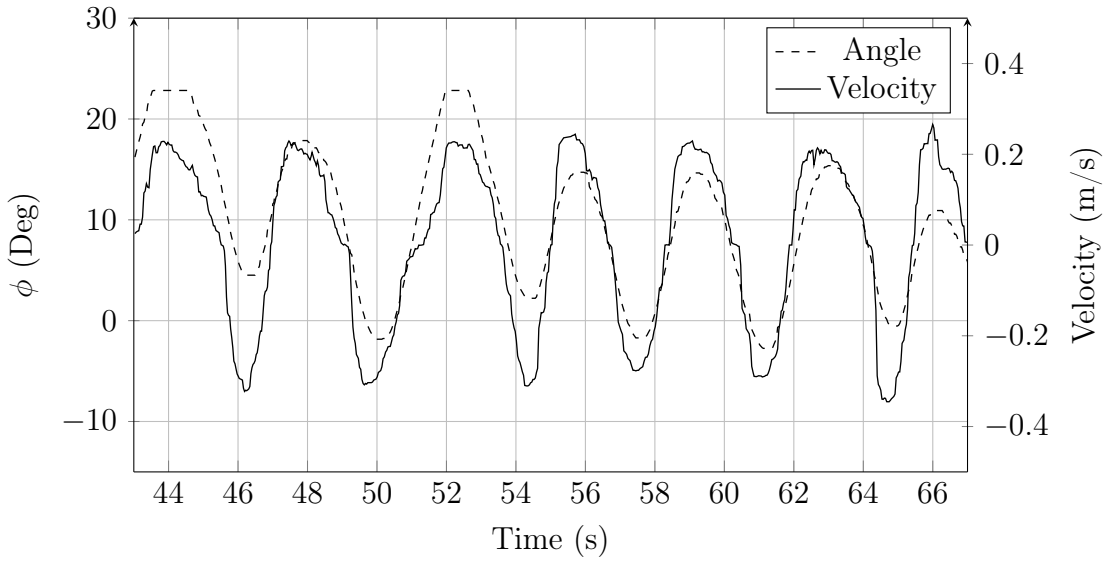


Figure 28: Azimuth angle and velocity in the gimbal reference frame.

Figures 27 and 28 show the declination and azimuth components of kite velocity with respect to the gimbal.

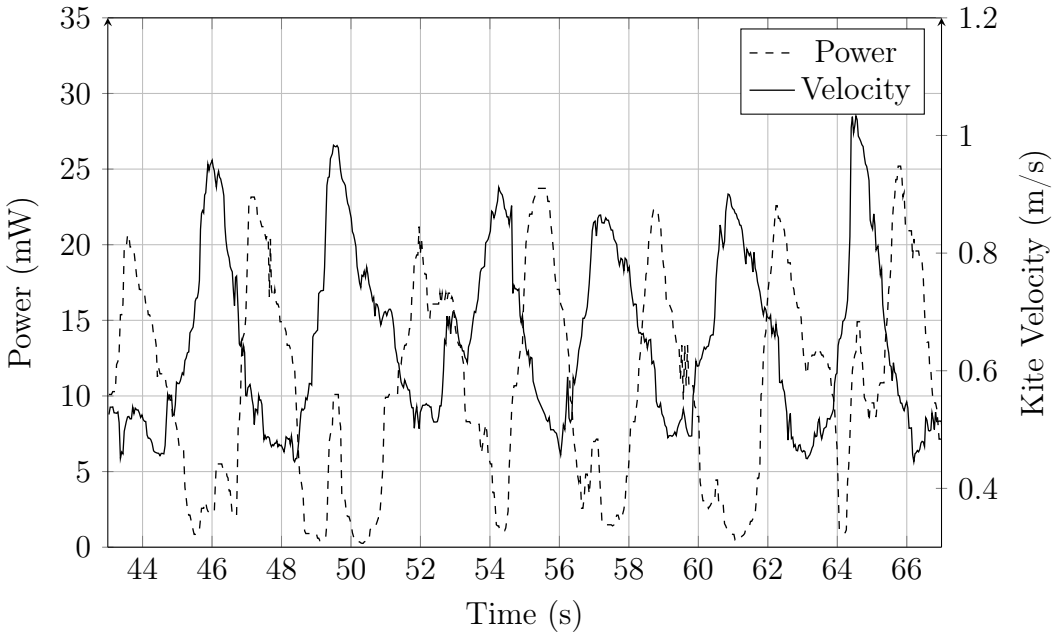


Figure 29: Power and V_{kite} in the kite body reference frame.

Figure 29 shows time records of the V_{kite} with respect to the kite body frame and power versus time. A time lag between the velocity and the power curves is observed. The exact cause of these was not investigated in depth, but it is hypothesized that the mechanical rotational inertia of the turbine, along with varying turbine efficiency due to varying RPM, were the causes of the phase shift between velocity and power.

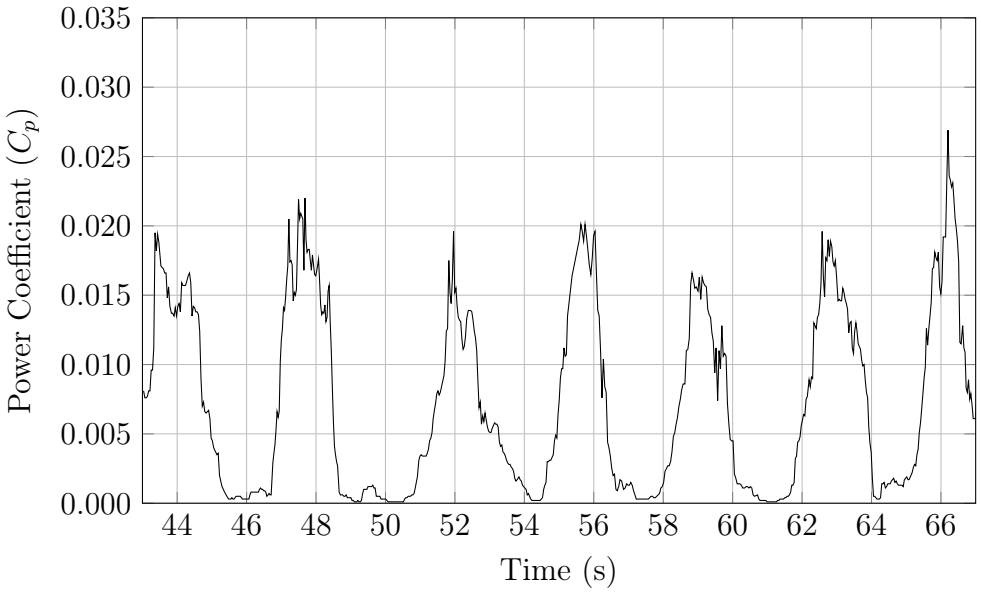


Figure 30: Kite power coefficient versus time.

Shown in figure 30 is the power coefficient, a measure of efficiency, as estimated with equation 2.34. During the entire test the turbine was extracting less than 3% of the available power in the flow, which is much less than the Betz limit of 59.3%.

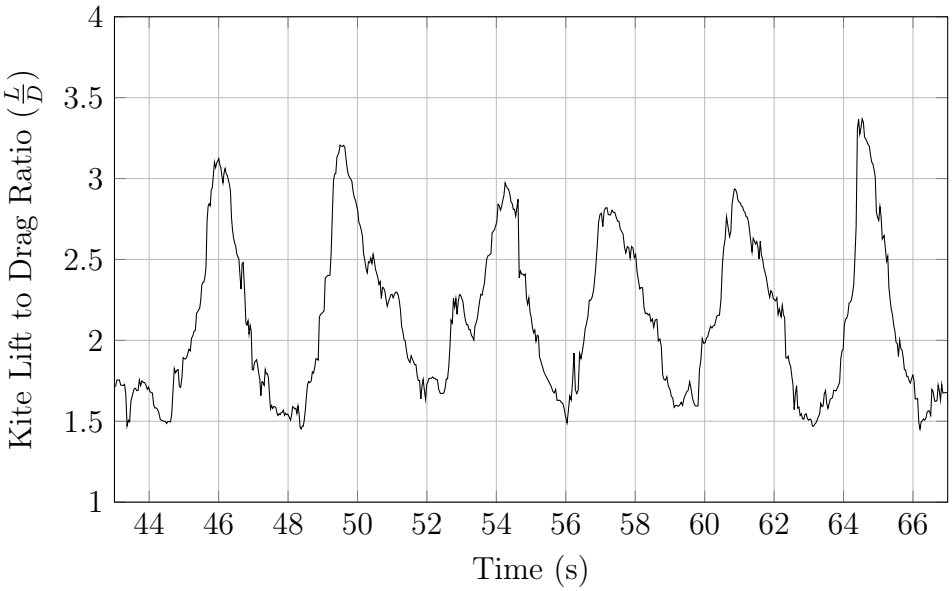


Figure 31: Kite lift to drag ratio versus time.

Figure 31 above shows the result of applying equation 2.36 to the velocity data from

the circular baseline. As discussed in section 2.5, an $\frac{L}{D}$ of 4 was chosen to conduct initial modeling of the turbine blades. The inaccuracies discussed in the derivation of equation 2.36 would lead to an under-prediction of $\frac{L}{D}$. Given that this graph shows a peak near 3.5, better modeling may show that was a reasonable estimate of the actual performance of the kite.

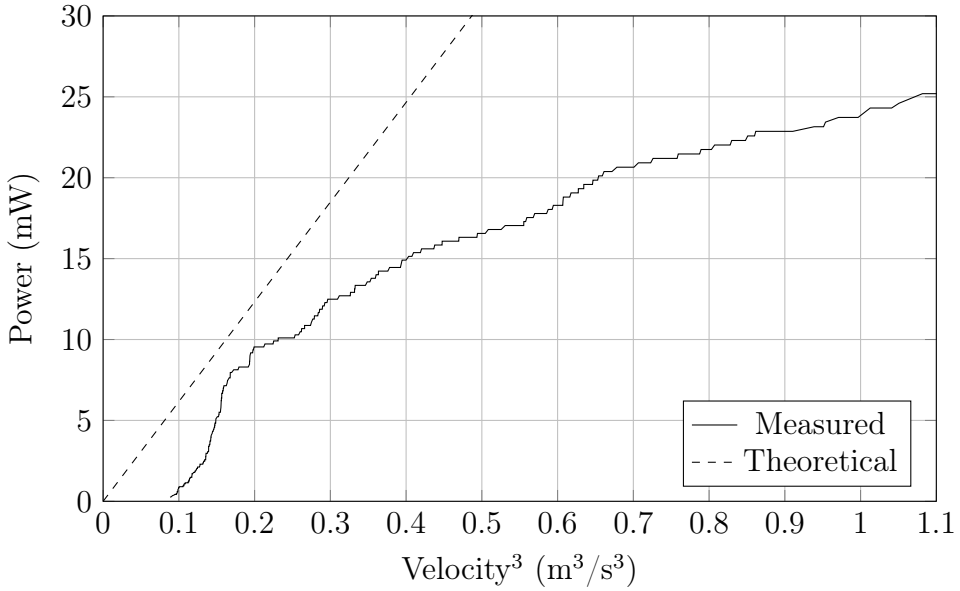


Figure 32: Comparison of theoretical velocity cubed behavior with actual performance.

Figure 32 shows a comparison between the ideal power vs velocity cubed line with a slope given by equation 2.35 and the actual performance of the kite. In the lower speed areas, the slope appears to be similar until 0.15 m/s or so. It then drops off and the kite performs worse than theory predicts.

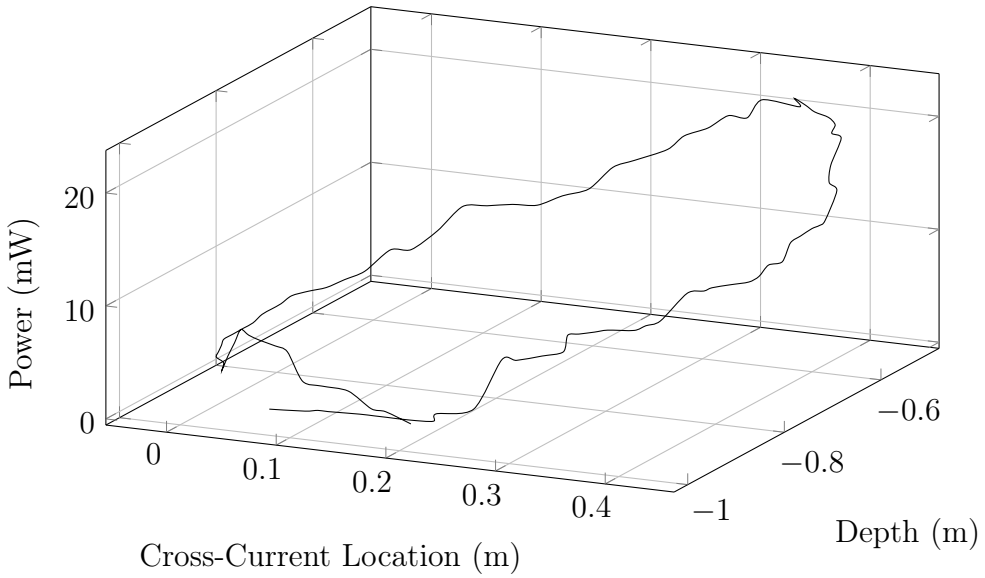


Figure 33: Power vs kite position in a single circular cycle.

Figure 33 shows the instantaneous power output as a function of position through a single loop of the circular path. The gradual increase and decrease is hypothesized to be a result of the phase shift observed and recorded in figure 29. Gravity had a significant effect on the kite's velocity, causing it to go faster on the downward portions of its trajectory. If the buoyancy of the kite were greater, it would act to offset gravity pulling the kite down and smooth out the velocity variations, particularly on circular trajectories. Considering the time lag between velocity and power previously observed, the power peak occurs near the surface as the kite continues around the trajectory.

Table 6: Baseline Circle Average Results

Parameter	**Value**
Average Power	10.3 mW
Average Velocity	0.65 m/s
Average L/D	2.15
Average C_p	0.0072

3.2 Baseline Figure 8

Presented here is another baseline case used for comparison to other test conditions. Table 7 shows important system parameters and test conditions, along with their values for the figure 8 path baseline test.

Table 7: Baseline Figure 8 Run Conditions

Parameter	Value
Current Speed	0.46 m/s
Kite Pitch Angle	85°
Desired Path Shape	Figure 8
Tether Length	1.969 m
Tether Diameter	0.012 m
Kite Wing Area	0.047 m^2
Kite Root Chord	0.224 m
Kite Tip Chord	0.068 m
Kite Span	0.382 m
Kite Weight	0.430 kg
Turbine Area	0.017 m

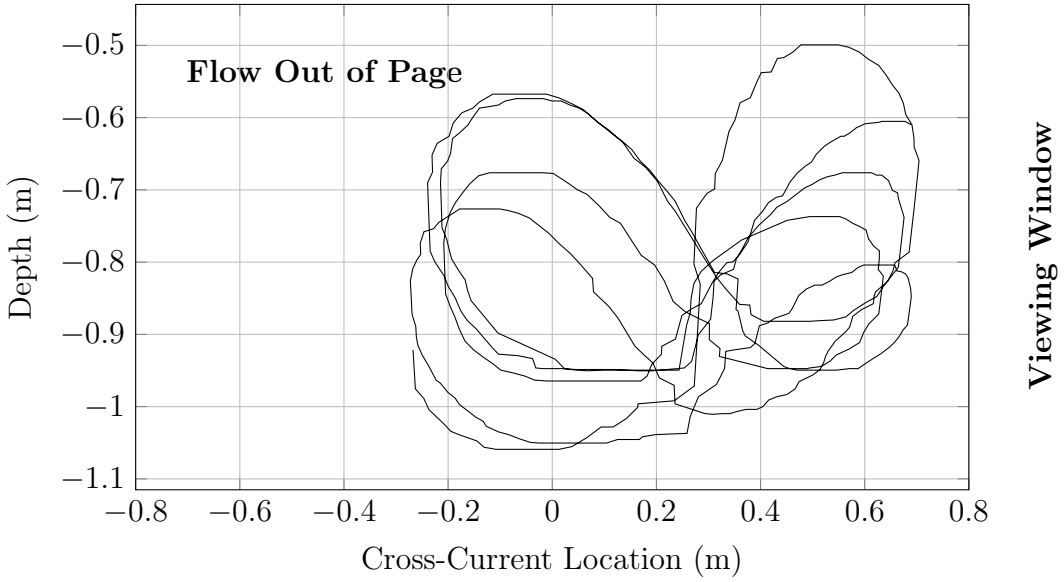


Figure 34: Downstream view of figure 8 kite trajectory.

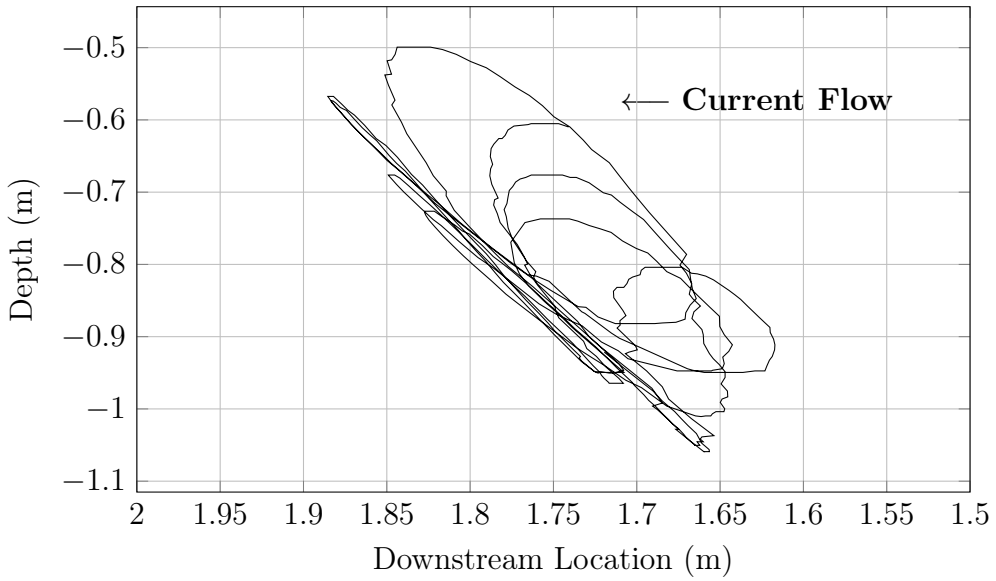


Figure 35: Window view of figure 8 kite trajectory.

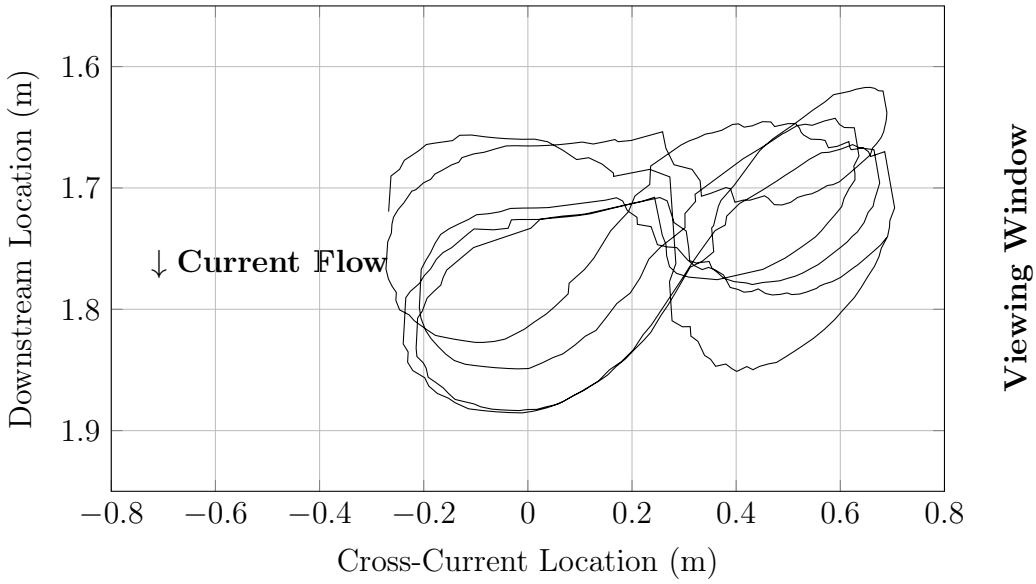


Figure 36: Overhead view of figure 8 kite trajectory.

Figures 34, 35 and 36 show the kite's figure 8 trajectory from three directions: downstream, the side viewing window, and directly overhead. The path shape variations are present because the trajectory was manually controlled by observing the kite's motion in real time. There was a definite tendency for the kite trajectory to favor the window

side of the flume, but its cause was not investigated as part of this testing.

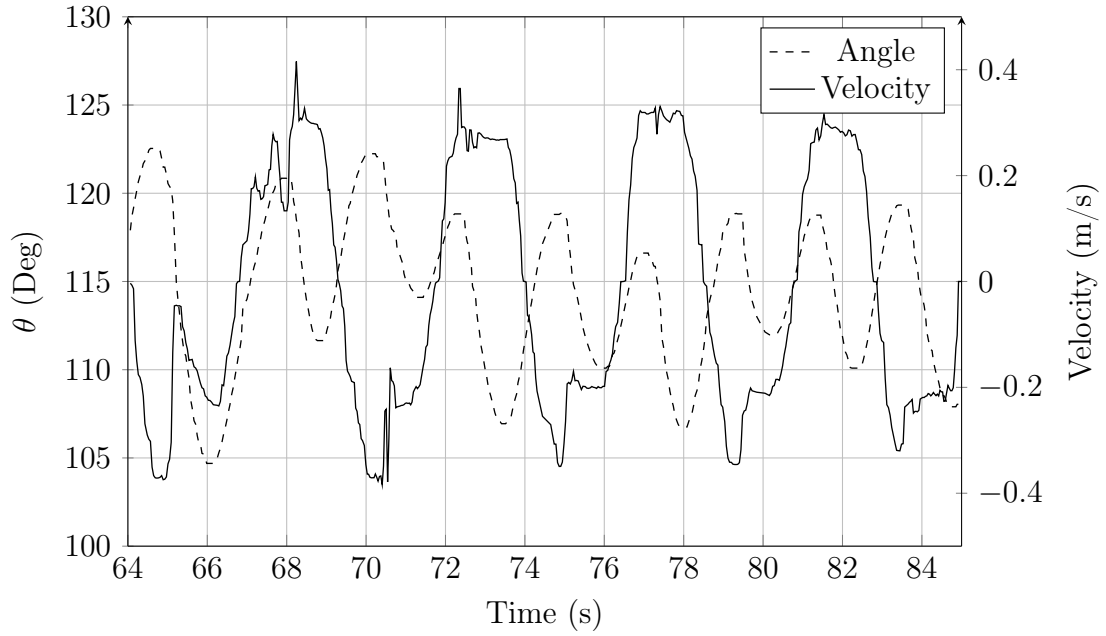


Figure 37: Declination angle and velocity in the gimbal reference frame.

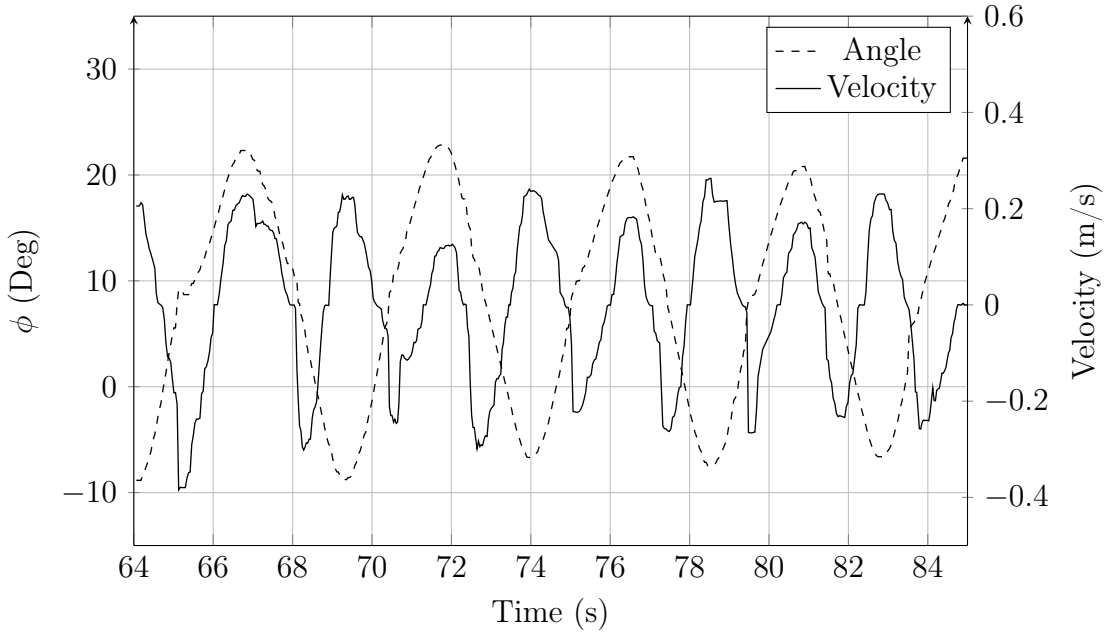


Figure 38: Azimuth angle and velocity in the gimbal reference frame.

Figures 37 and 38 show the declination and azimuth components of kite velocity with

respect to the gimbal.

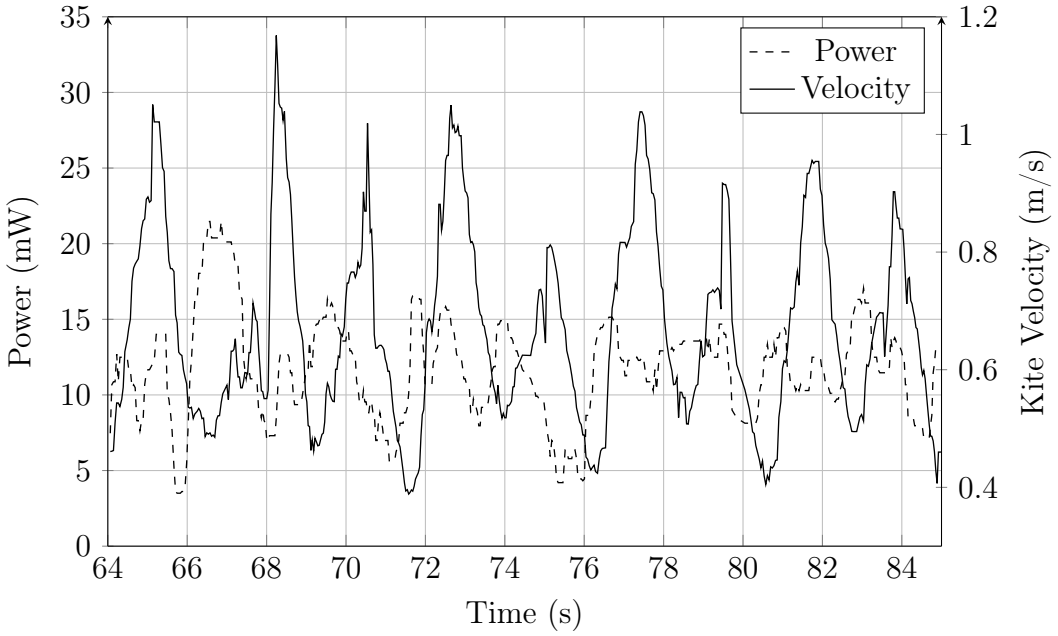


Figure 39: Power and V_{kite} in the kite body reference frame.

Shown in figure 39 is a plot of the V_{kite} with respect to the kite body frame and power with respect to time. The same time lag between V_{kite} and power observed in the circular test is also visible here.

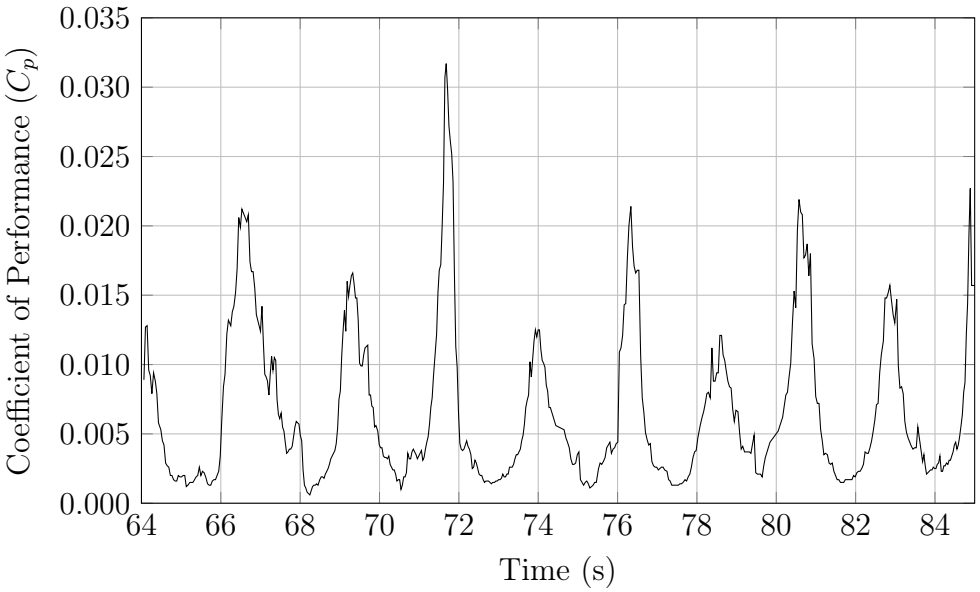


Figure 40: Kite power coefficient versus time.

Shown in figure 40 is the power coefficient as estimated with equation 2.34.

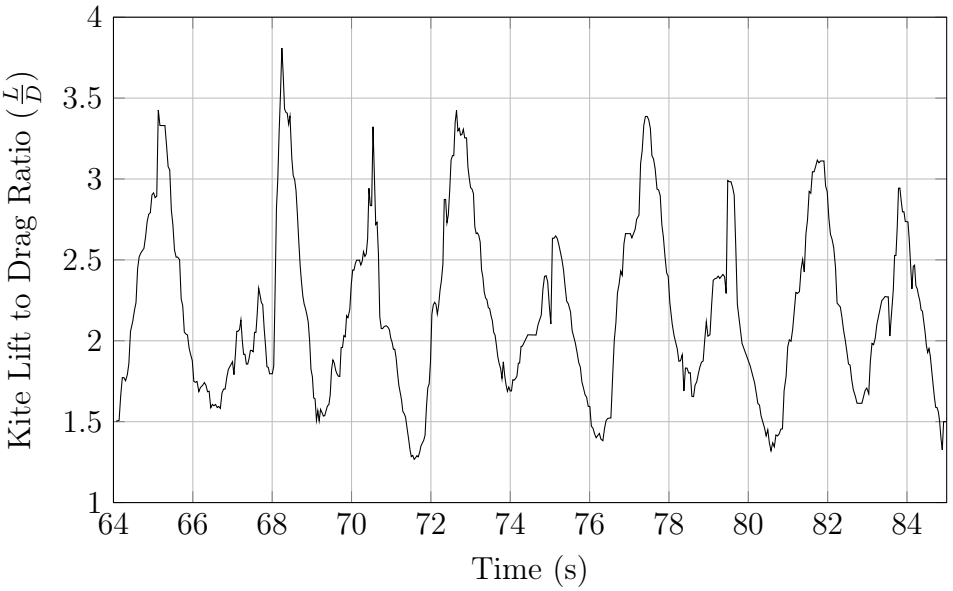


Figure 41: Kite lift to drag ratio versus time.

Figure 41 above shows the result of applying equation 2.36 to the velocity data from the circular baseline. Here, the $\frac{L}{D}$ peak is even higher than the circular case, further reinforcing the notion that 4 was a reasonable estimate for the final kite's $\frac{L}{D}$ value.

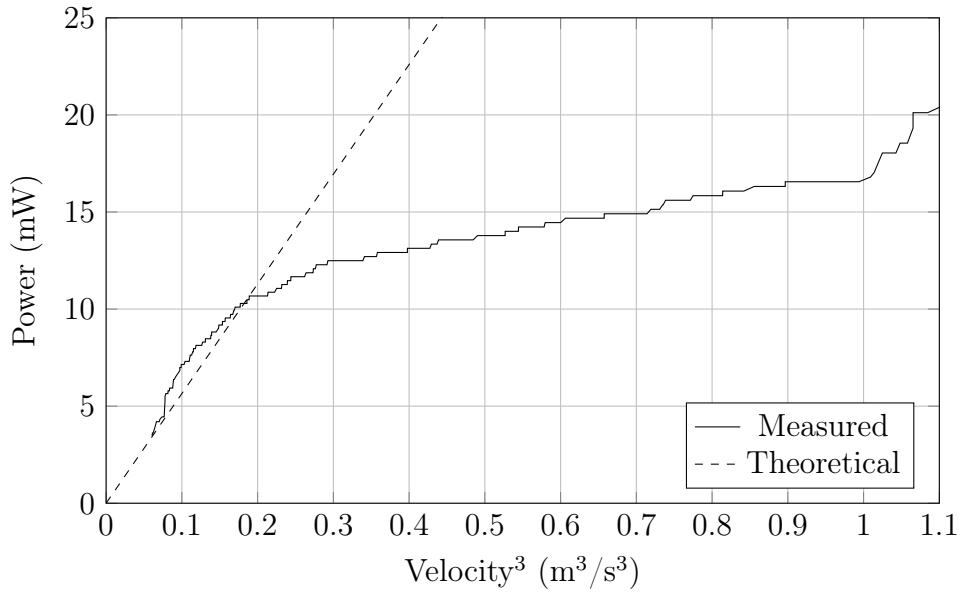


Figure 42: Comparison of theoretical velocity cubed behavior with actual performance.

Figure 42 shows a comparison between the ideal power vs velocity cubed line with a slope given by equation 2.35 and the actual performance of the kite. Performance in the figure 8 motion was slightly better than the circular motion, but both situations showed similar overall curve shapes.

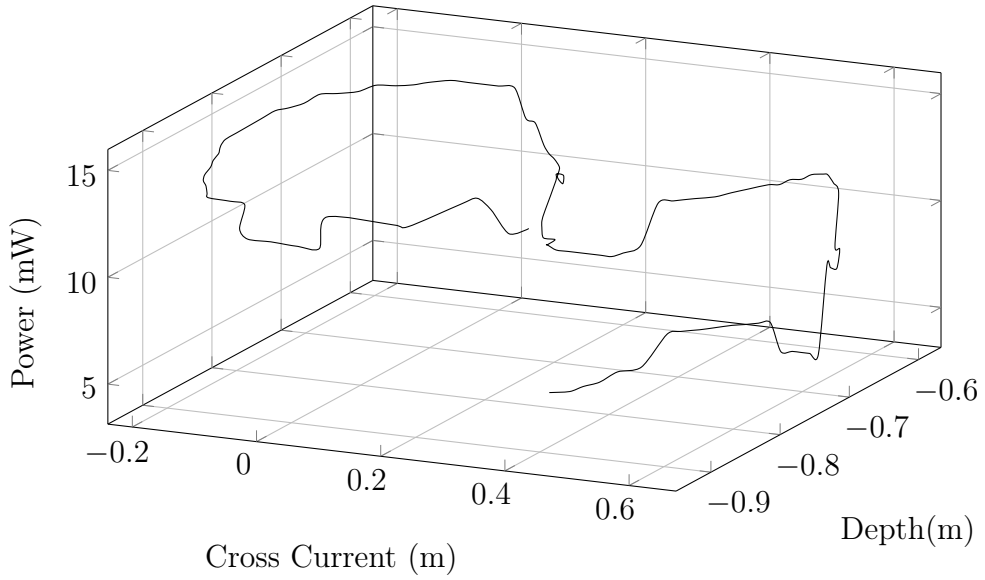


Figure 43: Power vs kite position in a single figure 8 cycle.

Figure 43 shows the instantaneous power output as a function of position through a single loop of the figure 8 path. The gradual increase and decrease is hypothesized to be a result of the phase shift observed and recorded in figure 29. Gravity had a significant effect on the kite's velocity, causing it to go faster on the downward portions of its trajectory. In the figure 8 case, these portions were in different locations around the trajectory as opposed to the circular case. Therefore, the power peaks occur in different locations as well.

Table 8: Baseline Figure 8 Average Results

Parameter	**Value**
Average Power	11.7 mW
Average Velocity	0.67 m/s
Average $\frac{L}{D}$	2.19
Average C_p	0.0024

3.3 Zero Kite Velocity Cases

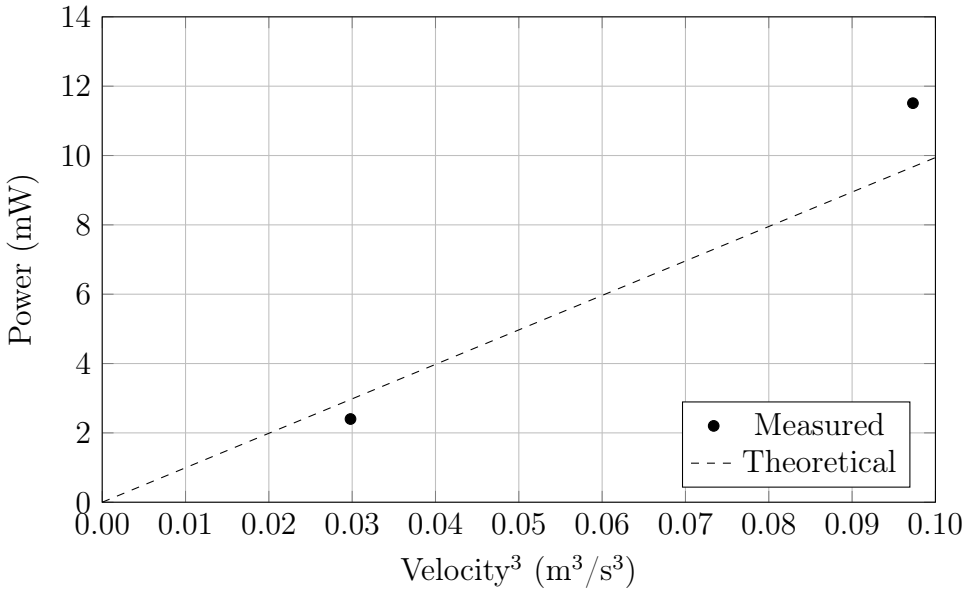


Figure 44: Comparison of theoretical velocity cubed behavior with actual performance with no kite motion.

As described in section 2.9 data was collected while the kite was held stationary at two current speeds, 0.31 m/s and 0.46 m/s. The average power produced was 2.40 mW and 11.51 mW, respectively. To evaluate these results against the power vs velocity cubed law, the same method used to generate figures 32 and 42 was used. To find the slope, the average $C_P = 0.0116$ was used. Both of the tested data points follow the theoretical line reasonably closely. To compare the performance between a moving kite and a stationary kite, the ratio $\frac{P_{moving}}{P_{stationary}}$ can be introduced. Of all the tests conducted at 0.31 m/s, the highest average power output was 17.58 mW, so $\frac{P_{moving}}{P_{stationary}} = 7.33$. At 0.46 m/s, there was less advantage to moving the kite. In that condition, the highest average power output was 19.23 m/s, so $\frac{P_{moving}}{P_{stationary}} = 1.09$. As stated elsewhere, this is a result of several factors currently reducing the performance of the kite.

3.4 Circular Path Performance Variation

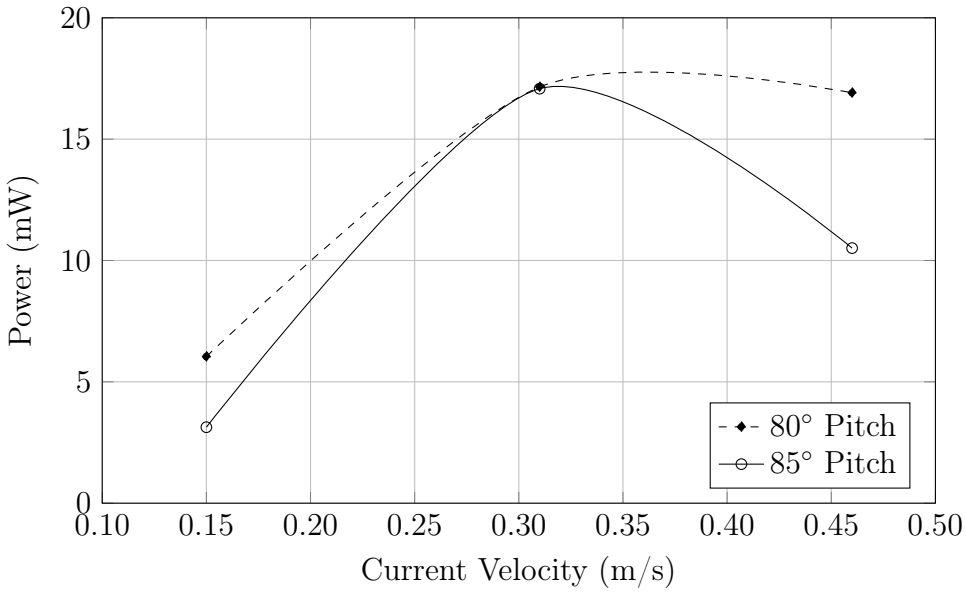


Figure 45: Current speed and pitch angle effect on power during circular motion.

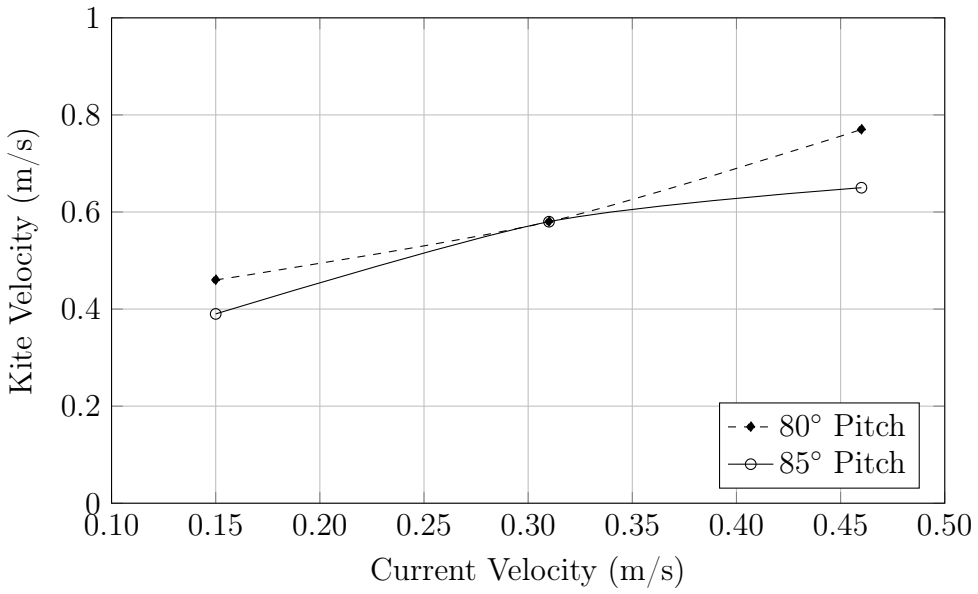


Figure 46: Current speed and pitch angle effect on V_{kite} during circular motion.

Figures 45 and 46 above show the variation in average V_{kite} and power output of the kite with respect to kite pitch angle, defined in figure 21, and current velocity. The kite was fairly sensitive to pitch angle, and good operation with reasonable V_{kite} only occurred for pitch angles of 80° and 85°. The tested range extended up to 90°, but that pitch angle resulted in the kite not operating at all.

It is not clear what caused the drop in output as $V_{current}$ was increased, and the investigation of this is a task for future work.

3.5 Figure 8 Path Performance Variation

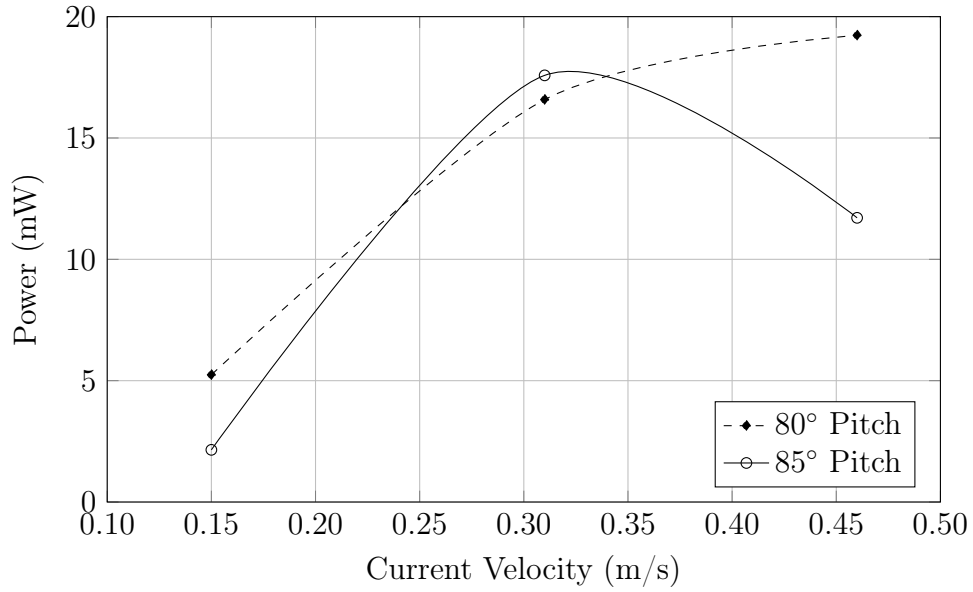


Figure 47: Current speed and pitch angle effect on power during figure 8 motion.

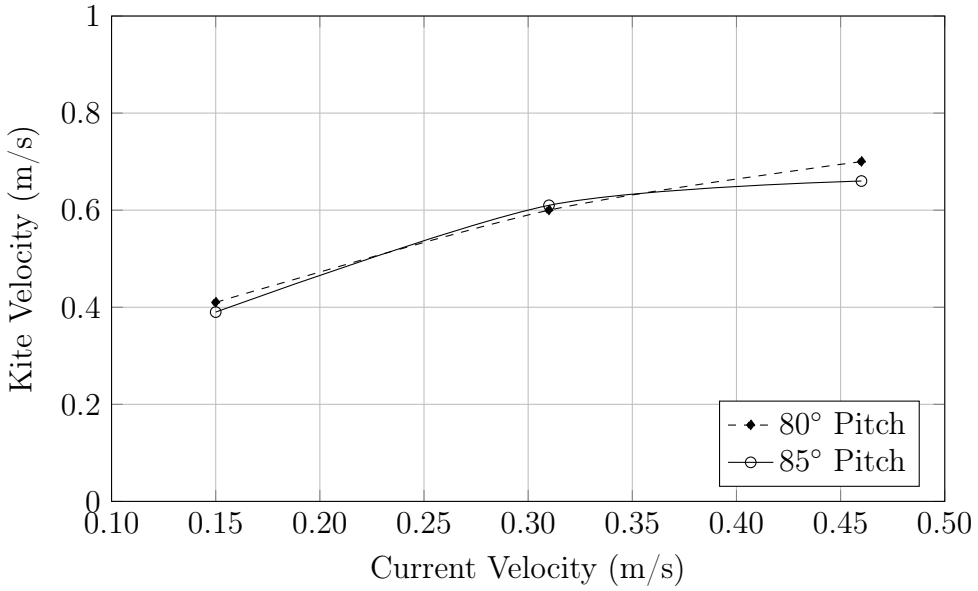


Figure 48: Current Speed and Pitch Angle Effect on V_{kite} during figure 8 motion.

Figures 47 and 48 above show the variation in average V_{kite} and power output of the kite with respect to kite pitch angle, defined in figure 21, and current velocity. In comparison

to the circular results, 80° of pitch at 0.46 m/s increased the power output by a noticeable margin over the same pitch angle at 0.31 m/s. Otherwise, the change from a circular path to a figure 8 path did not greatly affect the results, and the discussion above is applicable here.

3.6 Scaling Considerations

Now that the actual performance of the scale-model kite has been established, the results can be compared with the scale predictions made in section 2.3. The greatest difference between predicted and actual values was manifested in the power output, which was predicted to be $P_M = 2.875$ W. At its best operating conditions, the kite actually produced $P_{M_{actual}} = 17$ mW of peak instantaneous power. This represents a difference of two orders of magnitude. Several factors are hypothesized to be contributing to this disparity in predicted output and actual output. During the selection process of the generator, no attempt was made to properly scale the generator. There may also be scaling factors relating to transmission of power that were not investigated. Factors such as these were considered of lesser importance than verifying the functioning of the system as a whole, therefore the optimization of the electrical model was left to later investigation. Flume width may have also played into the reduced power generated. The power estimation assumed an optimal a steady state cross current motion, which was not achieved since the small flume test section width constrained the kite motion significantly. There is also significant room for improvement with regard to the shape of the kite and turbine.

4 Conclusion and Future Work

In this thesis research, a scale model of a Tethered Undersea Kite (TUSK) system was designed, constructed, and tested. Existing designs from WPI students were improved upon and replaced as necessary to ensure successful testing. This design effort encompassed all components of the system, including the kite; mechanical gimbal; controls; and data collection. The test equipment was able to measure the azimuth and declination angles of the rigid tether, and the power output of the generator on board the kite. This data was recorded for later analysis. Various sets of data were then plotted against each other to determine performance trends of the system at different operating conditions.

During testing at Alden Research Labs, the system was exposed to several current speeds. Initial plans called for current speeds as high as $V_{current} = 1$ m/s, but as speed was gradually increased the structural integrity of the kite came into question. It was then decided to stop increasing speed at $V_{current} = 0.46$ m/s. Multiple kite paths and pitch angles were tested at each current speed to determine the impact on the power output. Manual control systems were employed to guide the kite through circular and figure 8 path shapes. Pitch angle was static during tests and changed between tests.

Power output from the kite was much less than predicted from scaling calculations, but that can be attributed to several factors. One of the priorities of the kite design was ease of manufacture and configuration manipulation. The result of this was an open, unfaired design that allowed easy access to the components to make any unforeseen changes. However, exposing all of the components allowed them to make significant amounts of drag. Designing lightweight covers for the kite will likely significantly reduce the drag and lead to better performance. The turbine was designed with $V_{current} = 1.0$ m/s and $\frac{L}{D} = 4$ as estimates of nominal operating conditions. Given that the first condition was not met and the second is difficult to measure, the turbine may have been

operating significantly off of its design parameters. A new turbine design effort will be able to take advantage of this information to select more appropriate design conditions and ensure a higher performing turbine. Ease of manufacture was also paramount when designing the wings of the kite. Similarly to the turbine, making use of the results of this study will allow future design efforts to select more appropriate design parameters and limitations to optimize the hydrodynamic performance and increase power output. No outstanding issues were encountered with the data collection or manual kite control systems, and they will need few if any changes before being employed in future testing.

The main focus of future work will be using a large 20 foot wide by 10 foot deep flume at Alden Research Labs that allows for longer sustained cross current motion. Along with a new test site, improvements to the kite, turbine, and data collection systems should allow test results to more closely match real world performance. Replacing the rigid tether with a flexible tether is an important advancement in the fidelity of the model. This will also allow the kite to have 6 degrees of freedom. While being a closer simulation of real devices, kite control systems will have to be upgraded to handle 3 axis control duties. The data collection system will also have to be upgraded accordingly to gather 6 channels of data from the kite's position rather than the gimbal. Where numerical differentiation methods were used in this work, the application of gyroscopes and accelerometers will allow for direct measurement of angular rates. Also, adding the capability of measuring turbine RPM will allow more accurate measurements of efficiency and power output.

In terms of data analysis, there are improvements to be made with the techniques used. Better velocity and kite orientation data will allow better estimation of the velocity in the kite body frame. Along with this, an accurate measurement of the kite's angle of attack can be made. All of these improvements represent steps towards the model more closely replicating results from numerical simulation work currently under development at WPI.

References

[1] R. Moodley, M. Nthontho, S. Chowdhury, and S. Chowdhury, "A technical and economic analysis of energy extraction from the agulhas current on the east coast of south africa," *Power and Energy Society General Meeting*, pp. 1–8, 2012.

[2] A. Duerr and M. Dhanak, "An assessment of the hydrokinetic energy resource of the florida current," *IEEE Journal of Oceanic Engineering*, vol. 37, no. 2, pp. 281–293, 2012.

[3] M. Landberg, "Submersible plant," 2007. US Patent 8,246,293.

[4] M. Loyd, "Crosswind kite power for large-scale wind power production," *Journal of Energy*, vol. 4, no. 3, pp. 106–111, 1980.

[5] M. Diehl, "Airborne wind energy: Basic concepts and physical foundations," in *Airborne Wind Energy* (U. Ahrens, M. Diehl, and R. Schmehl, eds.), ch. 1, pp. 3–22, Springer, 2014.

[6] L. Fagiano, A. Zgraggen, M. Morari, and M. Khammash, "Automatic crosswind flight of tethered wings for airborne wind energy: Modeling, control design, and experimental results," *Control Systems Technology, IEEE Transactions on*, vol. 22, no. 4, pp. 1433–1447, 2014.

[7] L. Fagiano, A. Zgraggen, and M. Morari, "On modeling, filtering and automatic control of flexible tethered wings for airborne wind energy," in *Airborne Wind Energy* (U. Ahrens, M. Diehl, and R. Schmehl, eds.), pp. 167–180, Springer, 2014.

[8] G. Horn, S. Gros, and M. Diehl, "Numerical trajectory optimization for airborne wind energy systems described by high fidelity aircraft models," in *Airborne Wind Energy* (U. Ahrens, M. Diehl, and R. Schmehl, eds.), pp. 205–218, Springer, 2014.

[9] H. Li, D. Olinger, and M. Demetriou, "Control of a tethered undersea kite energy system using a six degree of freedom model," in *54th IEEE Conference on Decision and Control, Japan*, 2015.

[10] Y. Wang and D. Olinger, "Modeling and simulation of tethered undersea kites," in *34th International Conference on Ocean, Offshore, and Arctic Engineering*, 2015.

[11] M. Loyd, "Wind driven apparatus for power generation." `http://www.google.com/patents/US4251040`, 1981. US Patent 4,251,040.

[12] P. Williams, B. Lansdorp, and W. Ockles, "Optimal crosswind towing and power generation with tethered kites," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 1, pp. 81–93, 2008.

[13] L. Fagiano, M. Milanese, and D. Piga, "Optimization of airborne wind energy generators," *International Journal of Robust and Nonlinear Control*, vol. 22, pp. 2055–2083, 2011.

[14] W. Ockles, "Laddermill, a novel concept to exploit the energy in airspace," *Aircraft Design*, vol. 4, no. 2-3, pp. 81–97, 2001.

[15] M. Jacobson and C. Archer, "Saturation wind power potential and its implications for wind energy," *Proceedings of the National Academy of Sciences*, vol. 109, no. 39, pp. 15679–15684, 2012.

[16] K. Marvel, B. Kravitz, and K. Caldeira, "Geophysical limits to global wind power," *Nature Climate Change*, vol. 3, no. 2, pp. 118–121, 2013.

[17] C. Archer and K. Caldeira, "Global assessment of high-altitude wind power," *Energies*, vol. 2, no. 2, pp. 307–319, 2009.

[18] A. Bormann, M. Ranneberg, P. Kovesdi, C. Gebhardt, and S. Skutnik, "Development of a three-line ground-actuated airborne wind energy converter," in *Airborne Wind Energy* (U. Ahrens, M. Diehl, and R. Schmehl, eds.), pp. 427–436, Springer, 2014.

[19] F. Fritz, "Application of an automated kite system for ship propulsion and power generation," in *Airborne Wind Energy* (U. Ahrens, M. Diehl, and R. Schmehl, eds.), pp. 359–372, Springer, 2014.

[20] D. V. Lind, "Analysis and flight test validation of high performance airborne wind turbines," in *Airborne Wind Energy* (U. Ahrens, M. Diehl, and R. Schmehl, eds.), pp. 473–490, Springer, 2014.

[21] B. Lansdorp, R. Ruiterkamp, and W. Ockels, "Towards flight testing of remotely controlled surfkites for wind energy generation," *AIAA Paper*, vol. 6643, 2007.

[22] R. van der Vlugt, J. Peschel, and R. Schmel, "Design and experimental characterization of a pumping kite power system," in *Airborne Wind Energy* (U. Ahrens, M. Diehl, and R. Schmehl, eds.), pp. 403–425, Springer, 2014.

[23] M. Milanse, F. Taddei, and M. S., "Design and testing of a 60 kw yo-yo airborne wind energy generator," in *Airborne Wind Energy* (U. Ahrens, M. Diehl, and R. Schmehl, eds.), pp. 373–386, Springer, 2014.

[24] M. AB, "Minesto deep green technology." `http://minesto.com/deep-green/`, 2015.

[25] M. Jansson, "Hydrodynamic analysis and simulation of a tidal energy converter," *ISRN LUTMDN/TMHP–13/5283–SE*, 2013.

[26] I. Lazakis, O. Turan, and T. Rosendahl, "Risk assessment for the installation and maintenance activities of a low-speed tidal energy converter," 2012.

[27] M. AB, "Minesto deep green technology." http://minesto.com/technology-development/, 2015.

[28] HydroRun, "Hydrorun technology." http://hydrorun.ca/technology/, 2014.

[29] Y. Wang and D. Olinger, "Hydrokinetic energy from ocean currents using tethered underwater kites," in *34th International Conference on Ocean, Offshore, and Arctic Engineering*, 2015.

[30] D. Olinger and Y. Wang, "Hydrokinetic energy harvesting using tethered undersea kites," *submitted to the Journal of Renewable and Sustainable Energy*, 2015.

[31] N. Aye-Addo, J. O'Connor, and R. Perez, "Design of a scale-model tethered undersea kite for power generation," *Internal WPI Report*, 2014.

[32] R. Ruiterkamp and S. S., "Design and testing of a 60 kw yo-yo airborne wind energy generator," in *Airborne Wind Energy* (U. Ahrens, M. Diehl, and R. Schmehl, eds.), pp. 443–458, Springer, 2014.

[33] M. AB, "Minesto deep green data sheets." http://minesto.com/wordpress/wp-content/uploads/2014/09/dgtechnical-data-sheet11.pdf, 2015.

[34] "Kid wind generator." http://www.vernier.com/products/kidwind/wind-energy/parts/kw-gen/.

[35] "Propeller collet." http://www.hobbyexpress.com/images_products/1196_large.jpg.

[36] "Hs-5646wp servo." http://hitecrcd.com/products/servos/waterproof-servos-2/hs-5646wp-high-voltage-high-torque-programmable-digital-waterproof-servo/product.

[37] "Hs-5646wp servo image." http://www.modelsport.co.uk/_images/products/normal/hs5646wp.jpg.

[38] "Kid wind generator specifications." http://www.vernier.com/til/3155/?keyword=kw-gen.

[39] J. Manwell, J. McGowan, and A. Rogers, *Wind Energy Explained : Theory, Design and Application.* Wiley, 2009.

[40] K. Arruda, "A 2013 tethered glider for wind energy collection," *Internal WPI Report*, 2013.

[41] B. Mello, "Manufacture of a scale-model tethered undersea kite (tusk)," *Internal WPI Report*, 2014.

[42] "Arduino uno." http://www.arduino.cc/en/Main/ArduinoBoardUno.

[43] "Sparkfun microsd shield dev-12761." https://www.sparkfun.com/products/12761.

[44] "Operational amplifier chip." http://www.ti.com/product/LM358/description.

[45] "Sparkfun ina169 current sensor." https://learn.sparkfun.com/tutorials/ina169-breakout-board-hookup-guide/all.

[46] "Swoffer 2100-151 velocity probe." http://swoffer.com/products.htm.

5 Appendix

5.1 Appendix A - MATLAB Code for Data Post-Processing

```
%%
% This code takes a set of raw data in the MATLAB workspace
    and performs
% all of the data analysis desribed in the thesis report. It
    also outputs
% formatted data files for generating plots in LaTeX. The
    file paths in the
% writing loops at the end of the code will have to be
    changed to work on
% the particular computer being used to run this.
clc;
close all;
% Turn graphing and file writing on and off for
    troubleshooting.
graphToggle = input('Graph data this run? (1 = yes/0 = no) '
    );
writeToggle = input('Write data files this run? (1 = yes/0 =
    no) ');
%% Raw data intake and setup
% When the Arduino code was written, the names Pitch and Yaw
    were used, but
% these are not the correct names. They will be changed in
    this script so
% that "Pitch" data corresponds to declination angle, and "
    Yaw" data
% corresponds to azimuth angle.

% Begin by filtering raw angle data to remove noise
filtDec = medfilt1(Pitch_POT/100, 9); %Declination angles in
    degrees
filtAzm = medfilt1(Yaw_POT/100, 9); %Azimuth angles in
    degrees
filtPwr = medfilt1(MTR_PWR, 9); %Motor power in Arduino
    analog steps
%% Coordinate Transforms
% The coordinates used in actual testing do not match up
    with the standard
% definition of spherical coordinates. Transforms need to be
```

```
    applied to the
% data before processing it with standard spherical
    coordinate equations.

% The measured declination angle is zero in the x-y plane.
    Standard
% spherical coordinates have theta as zero along the
    positive z axis. Thus,
% 90 degrees has to be added to the measured data to
    transform it into the
% standard spherical system. This is the theta angle in
    standard spherical
% coordinates.

% During testing, flow was along the positive x axis in
    standard spherical
% coordinates, and the positive y axis pointed at the
    viewing window in the
% flume. Positive z was straight up, in accordance with the
    right hand
% rule. The test equipment recorded azimuth angles as
    positive towards the
% y axis, which is standard. This is the phi angle in
    standard
% spherical coordinates.

% Perform coordinate transforms in degrees
thetaDeg = filtDec + 90; %Theta (declination) angle in
    degrees
phiDeg = filtAzm; %Phi (azimuth) angle in degrees

% MATLAB performs all trig operations in radians, so the
    data is converted
% from degrees to radians
theta = thetaDeg*(pi/180); %Theta (declination) angles in
    radians
phi = phiDeg*(pi/180); %Phi (azimuth) angles in radians

% R in standard spherical coordinates corresponds to the
    length of the
% tether of the test rig. This was constanst throughout all
    tests and
% conditions.
tethL = 1.969; %Tether length of model in meters
```

```
r = tethL; %Change naming convention to match spherical
    coordinates

% For easier data visualization in MATLAB, the measured
    spherical
% coordinates are now transformed to Cartesian coordinates.
    Since the
% measured values were transformed to standard convention,
    the standard
% transform equations can be used.
kiteX = r.*sin(theta).*cos(phi); %Kite position along x axis
kiteY = r.*sin(theta).*sin(phi); %Kite position along y axis
kiteZ = r.*cos(theta); %Kite position along z axis
%% Kite position in Cartesian coordinates
if graphToggle == 1
    %Plot position data as X, Y, Z in a 3d plot
    figure(1)
    plot3(kiteX, kiteY, kiteZ)
    axis([1.5 2 -1 1 -1.2 -0.3])
    grid on
    title('3D Kite Position (Meters)')
    xlabel('Flow Direction (Downstream Positive)')
    ylabel('Cross-Flow (Window Positive)')
    zlabel('Depth (Negative Down)')
    %Plot 2-D projection in cross-flow vs depth plane
    figure(2)
    plot(kiteY, kiteZ)
    set(gca,'XDir','reverse');
    %axis([-1 1 0.35 1.2])
    grid on
    title('Kite Motion Projection on Y-Z Plane From Upstream
       ')
    xlabel('Cross-Flow (Window Positive)')
    ylabel('Depth')
end
%% Velocity Relative to Gimbal in Cartesian Coordinates
% The procedure for superimposing the current velocity on
    the kite velocity
% is simplified when performed in Cartesian coordinates, due
     to the current
% velocity field being given in Cartesian. After total
    velocity is found,
% azimuth and declination components will be found in
    spherical
```

```
% coordinates.

% A combination forward, reverse, and central difference
   methods were used
% to numerically differentiate the angle and time data.

% Differentiation of X

% Each successive data point is 1 data point greater than
   the last, so the
% delta is 1 for each step
dataStep = 1;

% Iterate through every element of the x position data
   vector
for a = 1: length(kiteX)
    if a < 3 %Perform a forward difference from first to
       third elements
        kiteXDiff(a,1) = (kiteX(a + dataStep) - kiteX(a))/
           dataStep; %Write the difference values to a new
           vector
    elseif a > (length(kiteX)-3) %Perform a backwards
       difference beginning 3 elements before the end of the
        vector
        kiteXDiff(a,1) = (kiteX(a) - kiteX(a - dataStep))/
           dataStep;
    else %Perform a central difference if the previous
       conditions fail, indicating a data point further than
        3 elements from either end
        kiteXDiff(a,1) = (kiteX(a + dataStep) - kiteX(a -
           dataStep))/(2*dataStep);
    end
end

% Differentiation of Y

% Iterate through every element of the y position data
   vector
for b = 1: length(kiteY)
    if b < 3 %Perform a forward difference from first to
       third elements
        kiteYDiff(b,1) = (kiteY(b + dataStep) - kiteY(b))/
           dataStep; %Write the difference values to a new
           vector
```

```
    elseif b > (length(kiteY)-3) %Perform a backwards
       difference beginning 3 elements before the end of the
        vector
        kiteYDiff(b,1) = (kiteY(b) - kiteY(b - dataStep))/
           dataStep;
    else %Perform a central difference if the previous
       conditions fail, indicating a data point further than
        3 elements from either end
        kiteYDiff(b,1) = (kiteY(b + dataStep) - kiteY(b -
           dataStep))/(2*dataStep);
    end
end

% Differentiation of Z

% Iterate through every element of the z position data
   vector
for c = 1: length(kiteZ)
    if c < 3 %Perform a forward difference from first to
       third elements
        kiteZDiff(c,1) = (kiteZ(c + dataStep) - kiteZ(c))/
           dataStep; %Write the difference values to a new
           vector
    elseif c > (length(kiteZ)-3) %Perform a backwards
       difference beginning 3 elements before the end of the
        vector
        kiteZDiff(c,1) = (kiteZ(c) - kiteZ(c - dataStep))/
           dataStep;
    else %Perform a central difference if the previous
       conditions fail, indicating a data point further than
        3 elements from either end
        kiteZDiff(c,1) = (kiteZ(c + dataStep) - kiteZ(c -
           dataStep))/(2*dataStep);
    end
end

% Differentiation of time

% Each successive data point is 1 data point greater than
   the last, so the
% delta is 1 for each step
deltaTime = 1;
for d = 1: length(MILLIS)
    if d < 3 %Forward difference at beginning of vector
```

```
        timeDiffMillis(d,1) = (MILLIS(d + deltaTime) -
           MILLIS(d))/deltaTime;
    elseif d > (length(MILLIS)-3) %Backwards difference at
       end of vector
        timeDiffMillis(d,1) = (MILLIS(d) - MILLIS(d -
           deltaTime))/deltaTime;
    else %Central difference elsewhere
        timeDiffMillis(d,1) = (MILLIS(d + deltaTime) -
           MILLIS(d - deltaTime))/(2*deltaTime);
    end
end
timeDiffSec = timeDiffMillis/1000; %Convert from
   milliseconds to seconds

% Now the above values are divided to find the dotted values
   .
kiteXDotUnFilt = kiteXDiff./timeDiffSec;
kiteYDotUnFilt = kiteYDiff./timeDiffSec;
kiteZDotUnFilt = kiteZDiff./timeDiffSec;

% Apply filtering to remove noise introduced by
   differentiation.
kiteXDot = medfilt1(kiteXDotUnFilt,9);
kiteYDot = medfilt1(kiteYDotUnFilt,9);
kiteZDot = medfilt1(kiteZDotUnFilt,9);

% The total kite velocity with respect to the gimbal is now
   found with the
% standard Cartesian velocity equation: V = sqrt(xDot^2 +
   yDot^2 +zDot^2)
kiteVelGim = sqrt(kiteXDot.^2 + kiteYDot.^2 + kiteZDot.^2);
%% Superposition of Current Velocity
% For simplicity, the current velocity was assumed to be
   uniform over the
% entire domain that the kite operated in. Tests were
   conducted at three
% different current speeds.
currentSpeedPrompt = input('Select Current Speed 1 = 0.15 m/
   s 2 = 0.31 m/s 3 = 0.46 m/s ');
if currentSpeedPrompt == 1
    currentSpeed = 0.15;
elseif currentSpeedPrompt == 2
    currentSpeed = 0.31;
elseif currentSpeedPrompt == 3
```

```
    currentSpeed = 0.46;
end

% Add the current velocity to the kite veclocity.
kiteXDotFlow = kiteXDot + currentSpeed;

% The total velocity of the kite with respect to the current
    is found with
% the same Cartesian formulation.
kiteVelFlow = sqrt(kiteXDotFlow.^2 + kiteYDot.^2 + kiteZDot
   .^2);
%% Conversion to Spherical Velocity
% The Cartesian velocity components will now be converted to
    their
% spherical counterparts. The conversion equations are as
   follows:
% rDot = (x*xDot+y*yDot+z*ZDot)/sqrt(x^2+y^2+z^2)
% thetaDot = (xDot*y-x*yDot)/(x^2+y^2)
% phiDot = (z*(x*xDot+y*yDot)-(x^2+y^2)*zDot)/((x^2+y^2+z^2)
   *sqrt(x^2+y^2))
for e = 1:length(kiteX)
    rDot(e,1) = ((kiteX(e)*kiteXDot(e))+(kiteY(e)*kiteYDot(e
       ))+(kiteZ(e)*kiteZDot(e)))/sqrt(kiteX(e)^2+kiteY(e)
       ^2+kiteZ(e)^2);
    thetaDot(e,1) = (kiteXDot(e)*kiteY(e)-kiteX(e)*kiteYDot(
       e))/(kiteX(e)^2+kiteY(e)^2);
    phiDot(e,1) = ((kiteZ(e)*(kiteX(e)*kiteXDot(e)+kiteY(e)*
       kiteYDot(e)))-((kiteX(e)^2+kiteY(e)^2)*kiteZDot(e)))
       /((kiteX(e)^2+kiteY(e)^2+kiteZ(e)^2)*sqrt(kiteX(e)^2+
       kiteY(e)^2));
end

% Total spherical velocity
sphVel = sqrt(thetaDot.^2+phiDot.^2);
%% Power Data
% Power from the motor was not measured directly, but rather
    with a current
% sensor. The total resistance of the system was measured as
    370 Ohms, and
% this was constant throughout all testing. Using the
   equation Power =
% Current^2 * Resistance, the power value can be found from
   the measured
% current. Information regarding the SparkFun INA169 current
```

```
    measurement
% board can be found at the following hyperlink:
% https://learn.sparkfun.com/tutorials/ina169-breakout-board
   -hookup-guide?_ga=1.33600098.2119036564.1426977964

% The board was unmodified from stock, so the tables
   regarding sensor
% accuracy from the above link apply without modification.

% Convert digital value to analog output of sensor
vOut = filtPwr.*(5/1024);
% Find current associated with Vout via given equation
current = (vOut.*1000)./(10*10000);
% Find power in watts with P = I^2 R
pwrWatt = 370.*(current.^2);
% Convert to milliwatts for more convenient analysis
pwrMilWatt = pwrWatt*1000;

% Find relationship between power and velocity^3
% Sort velocity and power values for proper plotting
kiteVelCubed = kiteVelFlow.^3;
sortKiteVelCubed = sort(kiteVelCubed);
sortPwrMilWatt = sort(pwrMilWatt);
%% Coefficient of Performance
% Power in Watts, area in m^2, density in kg/m^3, velocity (
   current
% reference) in m/s

% Kite area
kiteArea = 0.0474;

% Turbine area
turbArea = 0.0172;

% Water density
rhoH2O = 998;

for f = 1:length(pwrWatt)
    cP(f,1) = pwrWatt(f)/(0.5*rhoH2O*kiteVelFlow(f)^3*
       turbArea);
end

% Average value
avgCp = sum(cP)/length(cP);
```

```
%% Ideal Power vs Velocity Cubed
% Use average Cp and velocity cubed to demonstrate the
   theoretical
% relationship between power and v^3.
testVel = 0:0.001:max(kiteVelCubed);

% Multiplied by 1000 to display in milliWatts
thPwr = (avgCp*0.5*998*turbArea)*testVel*1000;
%% Other Data of Interest
% Average power in milliwatts
avgPwrMil = sum(pwrMilWatt)/length(pwrMilWatt);

%Peak power in milliwatts
peakPwrMil = max(pwrMilWatt);

% Average velocity (kite body frame reference)
avgVel = sum(kiteVelFlow)/length(kiteVelFlow);

% Kite L/D
for g = 1:length(kiteVelFlow)
    kiteLD(g,1) = (3/2)*(kiteVelFlow(g)/currentSpeed);
end

% Average value
avgLD = sum(kiteLD)/length(kiteLD);
%% General Data Exporting
% Create vector of real time values
realTime = MILLIS/1000;
% Query user for path shape
pathShape = input('Input path shape: 1 = Circle, 2 = Figure
   8 ');
%% Export Median Filter Demonstration Data
% Median filter demonstration data using Y velocity.
if pathShape == 1
    filtDemoData = [realTime,kiteYDotUnFilt,kiteYDot];
    if writeToggle == 1
        filtDemo = fopen('C:\Users\Ryan\Dropbox\TUSK\TUSK
           Project Report\Data\Methods\filtDemo.dat','w');
        fprintf(filtDemo, '%7s %7s %7s\r','time','unFilt','
           filt');
        fprintf(filtDemo, '%7.4f %7.4f %7.4f\r',filtDemoData
           ');
        fclose(filtDemo);
    end
```

```
end
%% Plot Power vs Position For One Loop
% Plot power vs position around one loop
% Circle: Day 3 Run 9 56.64 sec to 60.09 sec
% Figure 8: Day 3 Run 10 75.2 sec to 79.6 sec

if pathShape == 1
    % Find indicies of vectors that bound time interval of
       interest
    for h = 1:length(realTime)
        if realTime(h) < 56.5
            %disp('Below Value');
        elseif realTime(h) > 56.7
            %disp('Above Value');
            break
        else
            lowerLimitIndex = h;
        end
    end

    for i = 1:length(realTime)
        if realTime(i) < 59
            %disp('Below Value');
        elseif realTime(i) > 60.5
            %disp('Above Value');
            break
        else
            upperLimitIndex = i;
        end
    end
    % Take only the relevant portions of the position and
       power vectors
    shortKiteY = kiteY(lowerLimitIndex:upperLimitIndex);
    shortKiteZ = kiteZ(lowerLimitIndex:upperLimitIndex);
    shortPwr = medfilt1(pwrMilWatt(lowerLimitIndex:
       upperLimitIndex),9);
end

if pathShape == 2
    % Find indicies of vectors that bound time interval of
       interest
    for h = 1:length(realTime)
        if realTime(h) < 75.1
            %disp('Below Value');
```

```
        elseif realTime(h) > 75.3
            %disp('Above Value');
            break
        else
            lowerLimitIndex = h;
        end
    end

    for i = 1:length(realTime)
        if realTime(i) < 79.6
            %disp('Below Value');
        elseif realTime(i) > 79.7
            %disp('Above Value');
            break
        else
            upperLimitIndex = i;
        end
    end
    % Take only the relevant portions of the position and
       power vectors
    shortKiteY = kiteY(lowerLimitIndex:upperLimitIndex);
    shortKiteZ = kiteZ(lowerLimitIndex:upperLimitIndex);
    shortPwr = medfilt1(pwrMilWatt(lowerLimitIndex:
       upperLimitIndex),9);
end
%% Data Export for Results Sections
% Baseline data, depending on user input of path shape
if writeToggle == 1
    % If the baseline is a circular path, write all of the
       graphed data
    % vectors to this file.
    if pathShape == 1
        circBaseData = [realTime,kiteX,kiteY,kiteZ,thetaDeg,
           phiDeg,thetaDot,phiDot,kiteVelGim,kiteVelFlow,
           pwrMilWatt,sortKiteVelCubed,sortPwrMilWatt,cP,
           kiteLD];
        circBaseDataFile = fopen('C:\Users\Ryan\Dropbox\TUSK
           \TUSK Project Report\Data\Results\circBaseData.
           dat','w');
        fprintf(circBaseDataFile, '%7s %7s %7s %7s %6s %7s
           %7s %7s %7s %7s %7s %7s %7s %7s %7s %7s\r' ,'time
           ','kiteX','kiteY','kiteZ','thDeg','phDeg','thD','
           phD','kVG','kVF','pwr','kVC','pwrS','cP','L/D');
        fprintf(circBaseDataFile, '%7.4f %7.4f %7.4f %7.4f
```

```
        %8.4f %7.4f %7.4f %7.4f %7.4f %7.4f %7.4f %7.4f
        %7.4f %7.4f %7.4f\r', circBaseData');
    fclose(circBaseDataFile);
    % Write the non-graphed average data to a separate
        file.
    circAvgData = [avgPwr,avgVel,avgLD,avgCp];
    circAvgDataFile = fopen('C:\Users\Ryan\Dropbox\TUSK\
        TUSK Project Report\Data\Results\circAvgData.dat'
        ,'w');
    fprintf(circAvgDataFile,'%33s\r','Circular Baseline
        Average Figures');
    fprintf(circAvgDataFile,'%13s %5.4f %5s\r','Average
        Power',circAvgData(1),'Watts');
    fprintf(circAvgDataFile,'%16s %5.4f %3s\r','Average
        Velocity',circAvgData(2),'m/s');
    fprintf(circAvgDataFile,'%11s %5.4f\r','Average L/D'
        ,circAvgData(3));
    fprintf(circAvgDataFile,'%10s %5.4f','Average Cp',
        circAvgData(4));
    fclose(circAvgDataFile);
    % Write shortened data vectors to separate file
    shortCircData = [shortKiteY,shortKiteZ,shortPwr];
    shortCircDataFile = fopen('C:\Users\Ryan\Dropbox\
        TUSK\TUSK Project Report\Data\Results\
        shortCircData.dat','w');
    fprintf(shortCircDataFile,'%7s %7s %7s\r','kiteY','
        kiteZ','power');
    fprintf(shortCircDataFile,'%7.4f %7.4f %7.4f\r',
        shortCircData');
    fclose(shortCircDataFile);
    % Write theoretical power vs data to a separate file
    thPwrData = [testVel',thPwr'];
    thPwrDataFile = fopen('C:\Users\Ryan\Dropbox\TUSK\
        TUSK Project Report\Data\Results\thPwrDataCirc.
        dat','w');
    fprintf(thPwrDataFile,'%7s %7s\r','thVel','thPwr');
    fprintf(thPwrDataFile, '%7.4f %7.4f\r', thPwrData');
    fclose(thPwrDataFile);
% If the baseline is a figure 8 path, write all of the
    graphed data
% vectors to this file.
elseif pathShape == 2
    disp('Figure 8 Data')
    fig8BaseData = [realTime,kiteX,kiteY,kiteZ,thetaDeg,
```

```
    phiDeg,thetaDot,phiDot,kiteVelGim,kiteVelFlow,
    pwrMilWatt,sortKiteVelCubed,sortPwrMilWatt,cP,
    kiteLD];
fig8BaseDataFile = fopen('C:\Users\Ryan\Dropbox\TUSK
    \TUSK Project Report\Data\Results\fig8BaseData.
    dat','w');
fprintf(fig8BaseDataFile, '%7s %7s %7s %7s %8s %7s
    %7s %7s %7s %7s %7s %7s %7s %7s %7s %7s %7s\r' ,'time
    ','kiteX','kiteY','kiteZ','thDeg','phDeg','thD','
    phD','kVG','kVF','pwr','kVC','pwrS','cP','L/D');
fprintf(fig8BaseDataFile, '%7.4f %7.4f %7.4f %7.4f
    %8.4f %7.4f %7.4f %7.4f %7.4f %7.4f %7.4f %7.4f
    %7.4f %7.4f %7.4f\r', fig8BaseData');
fclose(fig8BaseDataFile);
% Write the non-graphed average data to a separate
    file.
fig8AvgData = [avgPwr,avgVel,avgLD,avgCp];
fig8AvgDataFile = fopen('C:\Users\Ryan\Dropbox\TUSK\
    TUSK Project Report\Data\Results\fig8AvgData.dat'
    ,'w');
fprintf(fig8AvgDataFile,'%33s\r','Circular Baseline
    Average Figures');
fprintf(fig8AvgDataFile,'%13s %5.4f %5s\r','Average
    Power',fig8AvgData(1),'Watts');
fprintf(fig8AvgDataFile,'%16s %5.4f %3s\r','Average
    Velocity',fig8AvgData(2),'m/s');
fprintf(fig8AvgDataFile,'%11s %5.4f\r','Average L/D'
    ,fig8AvgData(3));
fprintf(fig8AvgDataFile,'%10s %5.4f','Average Cp',
    fig8AvgData(4));
fclose(fig8AvgDataFile);
% Write shortened data vectors to separate file
shortFig8Data = [shortKiteY,shortKiteZ,shortPwr];
shortFig8DataFile = fopen('C:\Users\Ryan\Dropbox\
    TUSK\TUSK Project Report\Data\Results\
    shortFig8Data.dat','w');
fprintf(shortFig8DataFile,'%7s %7s %7s\r','kiteY','
    kiteZ','power');
fprintf(shortFig8DataFile,'%7.4f %7.4f %7.4f\r',
    shortFig8Data');
fclose(shortFig8DataFile);
% Write theoretical power vs data to a separate file
thPwrData = [testVel',thPwr'];
thPwrDataFile = fopen('C:\Users\Ryan\Dropbox\TUSK\
```

```
            TUSK Project Report\Data\Results\thPwrDataFig8.
            dat','w');
        fprintf(thPwrDataFile,'%7s %7s\r','thVel','thPwr');
        fprintf(thPwrDataFile, '%7.4f %7.4f\r', thPwrData');
        fclose(thPwrDataFile);
    end
end
```

5.2 Appendix B - Arduino Code

The following three files are the code that was uploaded to the Arduino to allow it to perform the functions described. These three files were written by Richard Eberheim.

TUSKLCD.ino

```
/*
This software is designed to turn an Arduino Uno into a controller and
    datalogger for the TUSK project
Data is collected from various sensors, time stamped, and logged to a file
     on the SD card.

A button switches the unit from configuration mode to run mode, during run
     mode a joystick controls the rudder.
  While running, data is logged to the SD card.  This data includes the
     position of the kite, the power generated and the rudder position along
     with a time stamp.
It displays an in-depth menu  on a 16x2 LCD for selecting runs and
    adjusting parameters.  The menu is navigated using a 2 axis joystick,
    where the Y axis is for selecting the option and the X-axis changes the
     value

LCD Menu Layout

Not Recording:
  -First Line:
    -Current Log File Info + Battery Value
  -Second Line: This will be the menu setup
  -Set Day
    -Increment through day number
  -Set Run
    -Increment through run number
  -Trim Servo
Recording:
  -First Line:
    Current Log File Information + Battery Value
  -Second Line
    Pot Pitch, Pot Yaw, Servo Value, Raw Power
*/


#define MENU_ITEM_TRIM 2
#define MENU_ITEM_CAL 3

#define MTR_PWR_PIN A0
#define POT_YAW_PIN A1
#define POT_PIT_PIN A2
#define JOY_VERT_PIN A3
#define JOY_HORZ_PIN A4
```

```
#define BATT_VOLT_PIN A5

#define RECORD_BTN_PIN 1
#define SERVO_PIN 6
#define SD_CS_PIN 8

#define LCD_RS_PIN 9
#define LCD_E_PIN 7
#define LCD_D4_PIN 5
#define LCD_D5_PIN 4
#define LCD_D6_PIN 3
#define LCD_D7_PIN 2

#include <LiquidCrystal.h>
#include <EEPROM.h>
#include <SD.h>
#include <Servo.h>
#include <SPI.h>

//Configure bounding values for the joystick analog input
const int joyVertMin = 236;
const int joyVertMax = 860;
const int joyVertCntr = 527;
const int joyHorzMin = 172;
const int joyHorzMax = 829;
const int joyHorzCntr = 517;
const int joyBumpDist = 100;

const int RS = 10; //INA 169 current sensor board Rs value (Ohms)
const long sysRes = 370L; //Total motor system resistance (including INA
    169 sensor) (Ohms)

LiquidCrystal lcd(LCD_RS_PIN, LCD_E_PIN, LCD_D4_PIN, LCD_D5_PIN, LCD_D6
    _PIN, LCD_D7_PIN); //Configures the LCD library
Servo rudder;

boolean isRunning = false; //Used to keep track if a run is in progress
int menuItem = 0; //The currently selected menu item
const int numMenuItems = 4; //The total number of menu items
long offset = 0; //The value for time offset from millis() recorded at the
     start of the run
int menuItemValues[numMenuItems]; //The settable value for each item in
    the menu
int lowStop = 0; //The lowest value the servo is allowed to go to

void setHighStop(int val){ //Sets the highest allowed value for the servo
  menuItemValues[numMenuItems - 1] = val;
}
void setLowStop(int val){ //Sets the lowest allowed value for the servo
  lowStop = val;
}
int getHighStop(){ //Returns the value for the high stop
```

```
 return menuItemValues[numMenuItems - 1];
}
int getLowStop(){ //Returns the value for the low stop
 return lowStop;
}

int prevMax = 95; //The stored value for the previous maximum value of the
    servo
int prevMin = 85; //The stored value for the previous minimum value of the
    servo

//This handles all of the joystick movements, both navigating the menus
   and controlling the servo
void inputJoystick(){
 if(isRunning){//If we are running, the joystick behaves as the controller
    for the Servo
  servoFromAnalog(analogRead(JOY_HORZ_PIN), getLowStop(), getHighStop(),
    false);
  long pitch = (long)(analogRead(POT_PIT_PIN))*100L;
  long yaw = (long)(analogRead(POT_YAW_PIN))*100L;
  long motorVolt = (long)(analogRead(MTR_PWR_PIN));

  pitch = map(pitch, 2000L, 76000L, 950L, 3400L);//Angle in 1/100 of a
    degree
  yaw = map(yaw, 11300L, 70900L, -1950L, 1950L);//Angle in 1/100 of a
    degree

  writeToFile(menuItemValues[0],menuItemValues[1],pitch,yaw,millis()-
    offset,motorVolt,rudder.read());//Here is the line that writes to the
    SD card.

  delay(1);
 }
  else{//If we are not running, use the joystick to move through the menus
   //Check to see if the user is commanding an upward movement in the
    menus
   if(analogRead(JOY_VERT_PIN) > joyVertCntr + joyBumpDist){
     while(analogRead(JOY_VERT_PIN) > joyVertCntr + joyBumpDist){
      delay(1);
     }
     menuUp();
   }//End Move Up If
   //Check to see if the user is commanding a downward movement in the
    menus
   if(analogRead(JOY_VERT_PIN) < joyVertCntr - joyBumpDist){
     while(analogRead(JOY_VERT_PIN) < joyVertCntr - joyBumpDist){
      delay(1);
     }
     menuDown();
   }//End Move Down If
```

```
    if(menuItem == MENU_ITEM_CAL){   //Checks to see if we are in the
    calibration menu
      int outVal = servoFromAnalog(analogRead(JOY_HORZ_PIN), 0, 180, false)
    ;
      if (outVal > prevMax){
       setHighStop(outVal);
       prevMax = outVal;
      }
      if (outVal < prevMin){
       setLowStop(outVal);
       prevMin = outVal;
      }
    }
    else{
      prevMax = 95;
      prevMin = 85;
      rudder.write(90 + menuItemValues[MENU_ITEM_TRIM]);
    //Check to see if the user is commanding a right movement in the menus
    if(analogRead(JOY_HORZ_PIN) > joyHorzCntr + joyBumpDist){
      while(analogRead(JOY_HORZ_PIN) > joyHorzCntr + joyBumpDist){
       delay(1);
      }
      changeCurrentMenuItem(-1);
    }//End Move Right If
    //Check to see if the user is commanding a left movement in the menus
    if(analogRead(JOY_HORZ_PIN) < joyHorzCntr - joyBumpDist){
      while(analogRead(JOY_HORZ_PIN) < joyHorzCntr - joyBumpDist){
       delay(1);
      }
      changeCurrentMenuItem(1);
    }//End Move Left If
    }
   }
}

//Configured Values

//Recording Values
int potYaw;
int potPitch;
int servoOutput;
int power;

//The SD card is not initialized on startup
boolean sdInitialized = false;

void loadInitialValues(){ //Loads the settings values from the EEPROM
  for(int i = 0; i < numMenuItems; i++){
    byte value = EEPROM.read(i);
    switch (i){
     case MENU_ITEM_TRIM:
       value = value - 127;
```

```
      break;
      case numMenuItems - 1:
       if (value == 0){
        value = 91;
       }
       byte value2 = EEPROM.read(i+1);
       if (value2 == 0){
        value2 = 90;
       }
       int val2 = value2;
       setLowStop(val2);
      break;
     }
     menuItemValues[i] = value;
   }
}

void storeValues(){ //Stores the settings values in the EEPROM
   for(int i = 0; i < numMenuItems; i++){
     byte value = menuItemValues[i];
      switch (i){
      case MENU_ITEM_TRIM:
        value = 0;
      break;
      case numMenuItems - 1:
       if (value == 0){
        value = 91;
       }
       if (getLowStop() == 0){
        setLowStop(90);
       }
       byte writeLowStop = getLowStop();
      EEPROM.write(i+1, writeLowStop);
      break;
     }
     EEPROM.write(i, value);
   }
}

//Shows a startup message while the Arduino starts up, allows everything
    to initialize and settle before doing anything
void startAndInitialize(int duration){
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("Starting up");
   delay(500);
   lcd.print(".");
   delay(500);
   lcd.print(".");
   delay(500);
   lcd.print(".");
   delay(250);
```

```
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Load EEPROM Vals");
  loadInitialValues();
  delay(500);
  initSDCard();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Welcome to TUSK");
  delay(2000);
}

int servoFromAnalog(int aIn, int outMin, int outMax, boolean useTrim){ //
    This returns a value in the specified range from the analog input range
    . It also enables the use of "trim" which is a slight offset of the
    servo from the center position
 int srvoVal = map(aIn, 172, 829, outMin, outMax);
 if (useTrim == true){
   srvoVal += menuItemValues[MENU_ITEM_TRIM];
 }
 srvoVal = constrain(srvoVal, outMin, outMax);
 rudder.write(srvoVal);
 return srvoVal;
}

//This checks to see if the system is in the "running". It also will
    check to see if the run button has been pressed and will switch modes
    accordingly.
void checkForRecord(){

 if (digitalRead(RECORD_BTN_PIN) == HIGH){//If the button has been pressed
    , switch the recording state
   storeValues();
   while(digitalRead(RECORD_BTN_PIN) == HIGH){
    delay(1); //Wait until it is released to prevent issues
   }
   if (isRunning == false){
    if(sdInitialized == false){//If the card has not been succesfully
   initialized yet
       initSDCard();//Try to initialize it
       if(sdInitialized == false){//If it is still not initializing, quit
         return;
      }
    }//End SD card check
    isRunning = true;
    lcd.clear();
    offset = millis();
   }
   else{
     isRunning = false;//Stop Running
     menuItemValues[1]++;
```

```
      lcd.clear();
    }
  }
}
int readBattVolt(){//Reads the battery voltage and puts it in the correct
    range
  int input = analogRead(A5);
  int out = map(input, 0, 1024, 0, 11000);
  //float voltage = out/10;
   return out;
}
void setup() {
  // put your setup code here, to run once:
  lcd.begin(16, 2);
  pinMode(RECORD_BTN_PIN, INPUT_PULLUP);
  pinMode(10, OUTPUT);
  pinMode(8, OUTPUT);
  rudder.attach(SERVO_PIN);
  rudder.write(90);
  startAndInitialize(5000);
  lcd.clear();
}

void loop() {
  // put your main code here, to run repeatedly:

  checkForRecord();//Check the run button and set the run state
  inputJoystick();// Reads the values from the joystick

  //the following code handles the setting of menu values
  if (menuItemValues[1] < 0){
   menuItemValues[1] = 0;
   storeValues();
  }
  if (menuItemValues[1] > 250){
   menuItemValues[1] = 250;
   storeValues();
  }


  if(menuItemValues[0] < 0){
   menuItemValues[0] = 0;
   storeValues();
  }
  if(menuItemValues[0] > 250){
   menuItemValues[0] = 250;
   storeValues();
  }


  if (menuItemValues[2] < -90){
     menuItemValues[2] = -90;
```

```
      storeValues();
  }
  if (menuItemValues[2] > 90){
      menuItemValues[2] = 90;
      storeValues();
  }
  //Clears the LCD
  if(isRunning){
    lcd.clear();
  }
  //Displays the correct values to the LCD
  displayFirstLine(menuItemValues[0], menuItemValues[1], readBattVolt());
  displaySecondLine();

  //Short delay for stability
  delay(1);
}
```

SDcontrol.ino

```
/*
This code handles all of the SD card specific functions needed by the TUSK
    project software
*/
//Handle all of the setup functions required to prepare the SD card
void initSDCard(){
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Init SD Card");
  pinMode(10, OUTPUT);
  delay(1000);
  if (!SD.begin(SD_CS_PIN)) {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("SD Init Failed");
    sdInitialized = false;
  }
  else{
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("SD Card OK");
    sdInitialized = true;
  }
  delay(1000);
}

//Writes the given set of values to a new line in the SD card
void writeToFile(int day, int runNum, long val1, long val2, long val3,
   long val4, long val5){
  /*String filename = "day";
```

```
filename = filename + String(day,DEC);
filename = filename + "run";
filename = filename + String(runNum,DEC);
filename = filename + ".csv";
*/
String filename = String("S" + String(day,DEC) + "R" + String(runNum,DEC
  ) + ".csv");
char w[filename.length()+10];
filename.toCharArray(w, filename.length()+10);
if(!SD.exists(w)){//If the file doesn't exist already, input the data
  header
   File dataFile = SD.open(w, FILE_WRITE);
   if (dataFile) {//Check to make sure the file is available
     dataFile.println("Pitch_POT,Yaw_POT,MILLIS,MIR_PWR,RUDD_POS");
     //dataFile.println(t);
     dataFile.close();
   }
   // if the file isn't available, display an error
   else {
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("FILE UNAVAILABLE");
     delay(2500);
     isRunning = false;
     return;
   }
}
File dataFile = SD.open(w, FILE_WRITE);

if (dataFile) {//Check to make sure the file is available, then write
  all of the values
   dataFile.print(val1,DEC);
   dataFile.print(",");
   dataFile.print(val2,DEC);
   dataFile.print(",");
   dataFile.print(val3,DEC);
   dataFile.print(",");
   dataFile.print(val4, DEC);
   dataFile.print(",");
   dataFile.println(val5,DEC);
   dataFile.close();
}
// if the file isn't available, display an error
else {
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("FILE ERROR");
   delay(2500);
   isRunning = false;
   return;
}
```

```
}
```

DisplayControl.ino

```
/*
This code handles all of the LCD control functions for the TUSK project
    software
*/
//The base strings to be displayed on the LCD
String menuItem0 = "Set Set: ";
String menuItem1 = "Set Run: ";
String menuItem2 = "Set Trim: ";
String menuItem3 = "Cal Srvo ";

//The array for containing the menu items
String menuItemNames[] = {menuItem0, menuItem1, menuItem2, menuItem3};

//Print the name and value for each menu item
void printMenuItem(String itemName, int value){
 lcd.setCursor(0,1);
 lcd.print(itemName);
 lcd.print(value);
}

//Strings for first line
String firstLineDay = "D";
String firstLineRun = " R";
String firstLineBattery = "mV";

//Write the correct values to the first line of the LCD, these are: day
    number, run number, and battery voltage
void displayFirstLine(int day, int run, int volts){
    lcd.setCursor(0,0);
    lcd.print(firstLineDay);
    lcd.print(day);
    lcd.print(firstLineRun);
    lcd.print(run);
    lcd.print(" ");
    lcd.print(volts);
    lcd.print(firstLineBattery);
}
//This will change value of the current menu item displayed
void changeCurrentMenuItem(int change){
  if(isRunning){//If we are running display the output values

  }
  else{//If we are not enable changing settings
  lcd.clear();
    if(menuItem == MENU_ITEM_TRIM){
       change = constrain(change, -4,4);
```

```
    }
    else{
     change = constrain(change, -1, 1);
    }
    if (menuItem != MENU_ITEM_CAL){
      menuItemValues[menuItem] = menuItemValues[menuItem] + change;
    }
}
}

//Moves the menu down to the next item
void menuDown(){
  lcd.clear();
  storeValues();
 if(menuItem < 3){
   menuItem++;
 }
 else{
  menuItem = 0;
 }
 if (menuItem == 3){
  delay(500);
 }
}

//Moves the menu up to the previous item
void menuUp(){
  lcd.clear();
  storeValues();
 if(menuItem > 0){
   menuItem--;
 }
 else{
  menuItem = 3;
 }
 if (menuItem == 3){
  delay(500);
 }
}

//Prints the values being read from the sensors to the LCD
void printOutputValues(){
  lcd.setCursor(0,1);
  lcd.print("M:");
  lcd.print(analogRead(MTR_PWR_PIN));
  lcd.print(" P:");
  lcd.print(analogRead(POT_PIT_PIN));
  lcd.print(" Y:");
  lcd.print(analogRead(POT_YAW_PIN));
}
```

```
//Displays the corect values on the second line of the LCD, that is the
    sensor values during a run and menu items when not running
void displaySecondLine(){
  if(isRunning){//If we are running display the output values
    printOutputValues();
  }
  else{//If we are not running display the menu
  if (menuItem != MENU_ITEM_CAL){
    printMenuItem(menuItemNames[menuItem], menuItemValues[menuItem]);
  }
    else{
     lcd.setCursor(0,1);
     lcd.print(menuItemNames[3]);
     lcd.print(getLowStop());
     lcd.print("/");
     lcd.print(getHighStop());
    }
   }
  }
```