Propeller Design Optimization for Tunnel Bow Thrusters in the Bollard Pull Condition

by

James R. Wilkins IV

B.S., Systems Engineering
United States Naval Academy, 2004

Submitted to the Department of Mechanical Engineering in Partial Fulfillment of the Requirements for the Degrees of

Naval Engineer

and

Master of Science in Mechanical Engineering

at the Massachusetts Institute of Technology
June 2012

Signature of Author_____
Department of Mechanical Engineering
May 16, 2012

Certified by_____
Chryssostomos Chryssostomidis
Doherty Professor of Ocean Science and Engineering
Professor of Mechanical and Ocean Engineering
Thesis Supervisor

Accepted by_____
David E. Hardt
Chair, Departmental Committee on Graduate Students
Department of Mechanical Engineering

Propeller Design Optimization for Tunnel Bow Thrusters in the Bollard Pull Condition

by

James R. Wilkins IV

Submitted to the Department of Mechanical Engineering on May 16, 2012 in Partial Fulfillment of the Requirements for the Degrees of Naval Engineer and Master of Science in Mechanical Engineering

Abstract

Tunnel bow thrusters are often used by large ships to provide low-speed lateral maneuverability when docking. Required to provide high thrust while essentially at a standstill, the design point for these thrusters is the bollard pull condition. Traditionally, the term bollard pull refers to the amount of force a tug can apply to a bollard when secured to a pier. Here, the bollard pull condition is used to describe a propeller with no flow over it except for that induced by its own rotation. Conventional propeller design is primarily performed for an optimal vessel speed or range of speeds. OpenProp, a propeller design code based on lifting line theory, is a numerical model capable of design and analysis of such propellers. It has been experimentally validated for standard design conditions in an external flow, but until now has been incapable of design with no external fluid velocity component applied. Recent updates to the model now allow for bollard pull design work. This project is the first application of the OpenProp model update. Propellers are designed for both open water and ducted (tunnel) applications in OpenProp. Propeller geometry design refinement by coupling MTFLOW, an Euler Equation viscous flow solver, with PBD-14, a lifting surface design program for marine propulsors is examined. An experimental apparatus is constructed to test the propeller designs and validate the OpenProp model. A range of off-design operating conditions are analyzed and results are presented.

Thesis Supervisor: Chryssostomos Chryssostomidis
Title: Doherty Professor of Ocean Science and Engineering
Professor of Mechanical and Ocean Engineering

Acknowledgements

First and foremost, I would like to thank my wife, Beth, for her patience and support through many long days and late nights of work. Her encouragement kept me going and the sacrifices she made so that I could succeed were a beautiful picture of her love and commitment. I would not have finished were it not for her partnership.

I give a special thank you to my daughter Lucy, and my son Rud, for all the smiles and hugs they gave me and understanding when I missed bed time.

I am grateful to Professor Chryssostomos Chryssostomidis for his interest, financial support, and insight.

I owe a debt of gratitude to Dr. Brenden Epps for his guidance, support, and willingness to put in many long days helping me at Sea Grant and several very late nights at the towing tank. Without his help, this project may not have been accomplished.

Finally, thanks to John Capomaccio, who performs a vital but often thankless job, and always had a friendly word on his late-night rounds.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

Introduction

The term bollard pull is used as a term referring to the total amount of thrust that a boat or ship can generate and sustain during a test for which the vessel is secured to a mooring bollard via a hawser with a load cell. It is a measure of that vessel's maximum pushing or towing power. Since the vessel is tied off to the pier and unable to move during the test, bollard pull in the context of propellers is used to refer to the condition of operating with no advance speed.

Propellers designed to work at bollard pull and low speeds of advance are generally used for slow moving vessels like tugs, pusher boats, and Autonomous Underwater Vehicles (AUVs) that do not have a requirement for high speed transit. They are also extensively used in tunnel bow thrusters, which are transversely oriented and located at the bow of a ship near the keel. A tunnel bow thruster is advantageous because it gives large ships a control point near the bow to assist with stationkeeping and maneuvering alongside a pier. They can eliminate or reduce the need for tugboat assistance when entering or leaving port, saving both time and money.

Current interest in tunnel bow thruster design is focused on the potential energy savings to be found by taking advantage of the inherent drive train efficiencies of a permanent magnet motor rim-drive thruster system. This type of system is more mechanically and electrically efficient than current systems which require either the motor itself or a large gearing mechanism to be physically located in the tunnel. A rim drive system eliminates or reduces the need for structural intrusions into the propeller tunnel which tend to block fluid flow. It requires, however, specialized bearings that currently remain in experimental stages before it will be feasible. Current work on such self-compensating, hydrostatic/hydrodynamic hybrid bearings is ongoing at MIT.

As a preliminary step toward future development of optimized rim drive tunnel thruster systems, and looking forward to when the hybrid bearings are ready for application, this thesis validates recent updates to the OpenProp suite of propeller design codes that allows propeller design for the bollard pull condition. Three propellers were designed and built using the new OpenProp code. Procedures for coupling a propeller blade design code, PBD-14.36, with MTFLOW, a Multipassage ThroughFLOW design and analysis program, were established and documented although these codes were not used for the present propeller designs due to time constraints on this thesis. Procedures for using a propeller test platform for bollard pull tests were established and performed for the each of the three propellers. This thesis presents the test results for the three propellers and the procedural documentation for future use of the test fixture and coupling of the design codes.

Chapter 1: Design for Bollard Pull

Current Design Practice

Current practices for designing propellers for bollard pull fall into one of three general categories or techniques:

1) Adapt an existing design
2) Lifting line theory plus lifting surface correction factors
3) Lifting line theory with coupled CFD – lifting surface design

The first is simply to use existing public or proprietary design geometry and adapt it to the vessel size and loading requirements by matching characteristics such as diameter and number of blades. This is the easiest and cheapest way to develop a propeller design, and is readily available from a wide variety of commercial marine propulsion retailers and fabricators. Since this method is a one-size-can-be-adapted-to-most method, it is not considered to an optimum design, but in many cases is "good enough," since the designs used usually have a long and proven track record.

The second method (recently demonstrated by Mertes et al [22]) relies on lifting line theory for the primary design. Mertes' process was to begin by designing a free propeller with ducted inflow and given available power and actual speed. Lifting surface correction factors [23]were then applied to the propeller geometry to account for known shortcomings of the lifting line model implementation. These steps were carried out iteratively through several cycles, varying blade outline, pitch, and camber. This method is widely used and well established due to its long, proven history [1]It provides a focused design that is individually tailored for each application. It has a high but slightly limited degree of accuracy due to its simplified treatment of the wake alignment and structure.

The third method (recently demonstrated by Ozdemir et al [27]) begins the design using lifting line theory to develop initial propeller design characteristics, then used CFD analysis coupled with lifting surface corrections to finalize the blade geometry. Given the number of blades, diameter, and design rotation speed, a lifting line method was used which prioritized the propeller blade area to minimize cavitation risk. Beam theory was applied to determine stresses and calculate appropriate thicknesses. The sections were kept symmetric with respect to thickness, chord, and diameter. Finally a commercially available CFD code (FLUENT 6.3.26) was used along with an unspecified lifting surface code developed at Yildiz Technical University's Department of Naval Architecture and Marine Engineering to perform surface modification. Applying CFD for detailed wake alignment and lifting surface modification is the most difficult and computationally intense method discussed here, but can provide a propeller design with better performance characteristics than the previous two methods.

Additional work is being done in an effort to more accurately characterize the hydrodynamic behavior of the wake behind a propeller at bollard pull conditions. El Lababidy et al have conducted a series of studies examining near wake conditions at true and near bollard pull conditions. Using a Dynamic Positioning (DP) thruster model in the Italian Ship Model Basin,

they first examined the wake using Stereo Particle Image Velocimetry (SPIV) [7] measuring vorticity and turbulence over a range advance coefficients at six downstream transverse planes. In a follow-up study [8], Laser Doppler Velocimetry (LDV) measurements were taken at the same conditions as the SPIV study and compared to the previous results. These results largely agreed, and several advantages and disadvantages of each system were identified and discussed. In a further study [9], a two-component LDV system was used to compare the performance of a DP thruster with and without a nozzle. These results are instructive for bollard pull propeller design and provide valuable insight into the near wake performance.

Propeller Characteristics

Propeller performance is traditionally described by three convenient factors, the thrust coefficient, K_T, the torque coefficient, K_Q, and the open water efficiency, η_O.

Open water efficiency relates the thrust power generated by the propeller to the torque power delivered by the propeller.

$$\eta_O = \frac{P_T}{P_Q} = \frac{1}{2\pi}\frac{Tv_A}{Qn} = \frac{1}{2\pi}\frac{JK_T}{K_Q}$$

Where T is the thrust generated by the propeller, Q is the torque delivered by the propeller, v_A is the speed of advance, and n is the rotational speed of the propeller.

The thrust coefficient relates the thrust generated by the propeller to the propeller speed and the diameter of the propeller.

$$K_T = \frac{T}{\rho n^2 D^4}$$

Similarly, the torque coefficient relates the torque applied to the propeller to the rotational speed and diameter of the propeller.

$$K_Q = \frac{Q}{\rho n^2 D^5}$$

A convenient way to characterize these parameters is to compare them against the advance coefficient, J, which relates the speed of advance to the propeller tip speed.

$$J = \frac{v_A}{nD}$$

Plotting them against J is a good way to see how a propeller will perform over its full range of operating conditions. This is very convenient for a traditional propeller design for a ship intended to make way through the water. For very slow or at zero speed (v_A=0) however, the efficiency is no longer useful as a performance metric since both efficiency and advance coefficient go to zero along with advance speed.

Quality Factor

In an effort to find an alternate means to characterize performance, the open water efficiency of a propeller can be compared to the ideal efficiency achieved by a theoretical "actuator disk" of matching diameter and an infinite number of blades. According to momentum theory, the maximum efficiency that is theoretically possible by such a propeller (η_i) can be related to the thrust loading coefficient (C_T).

$$\eta_i = \frac{2}{1+\sqrt{1+C_T}} = \frac{2}{1+\sqrt{1+\frac{8K_T}{\pi J^2}}}$$

Where:

$$C_T = \frac{T}{\frac{1}{2}\rho {v_A}^2 \frac{\pi}{4} D^2} = \frac{8K_T}{\pi J^2}$$

Defining the Quality Factor, QF, as the ratio of the openwater efficiency to the ideal efficiency and using the relationships defined above, the following relationships can be derived as demonstrated by Woud and Stapersma in *Design of Propulsion and Electric Power Generation Systems* [28].

$$Quality\ Factor = \frac{\eta_O}{\eta_i} = \frac{1}{2\pi}\frac{K_T J}{K_Q}\cdot\frac{1+\sqrt{1+C_T}}{2} = \frac{1}{2\pi}\frac{JK_T}{K_Q}\cdot\frac{1+\sqrt{1+\frac{8K_T}{\pi J^2}}}{2}$$
$$= \frac{1}{4\pi}\frac{K_T}{K_Q}\left(J+\sqrt{J^2+\frac{8}{\pi}K_T}\right)$$

From this result, when the speed of advance, and hence the advance coefficient, is zero, the quality factor still has a value and is a useful measure of performance for the bollard pull condition.

$$Quality\ Factor\ (J=0) = \frac{1}{4\pi}\frac{K_T}{K_Q}\sqrt{\frac{8}{\pi}K_T} = \frac{1}{\pi\sqrt{2\pi}}\frac{{K_T}^{1.5}}{K_Q}$$

Chapter 2: Design in OpenProp

OpenProp Description

OpenProp [20] is a collection of open-source propeller design codes developed by students and professors from MIT and Maine Maritime Academy. OpenProp is rooted in the simple propeller design code, PVL (Propeller Vortex Lattice) [18] and has since been upgraded in performance and capability. In 2007, the PVL codes (which were written in FORTRAN) were translated to Matlab M-code and called MPVL. A graphical user interface was then added to MPVL, increasing its usefulness for both novice and skilled propeller designers [3]. MPVL was also updated with the capability to develop a propeller's geometry sufficient for output to a 3D printer for rapid prototyping [5]. In 2008, the capability for ducted propeller design was introduced [26] as well as routines for cavitation analysis [24] and the code was renamed OpenProp. In 2009, a method for evaluating propeller performance in off-design conditions was developed [17], and the code was expanded to enable axial flow turbine design and analysis [16]. In 2010, a method for calculating blade stress implementing the ABS rules was developed [19] and subsequently applied to propeller design optimization [13][15]. Further details of the capabilities of OpenProp version 2.4 are given in the theory document [14].

The OpenProp codes [20][14] treat the propeller blade as a moderately loaded lifting line which sheds trailing vorticity along the local flow velocity streamlines. The shed vortex filaments are assumed to take a uniform diameter helical flow shape. The blade is considered as a series of 2D airfoil sections that are subsequently integrated into a whole. The wake is modeled with a vortex lattice method. The OpenProp algorithm optimizes the circulation distribution along the blade in such a way that the propeller generates a specified thrust while minimizing the torque required to produce it.

OpenProp Updates

A number of recent updates to the OpenProp code have been released in version 3.2 [14] (which succeeds the prior release version 2.4 discussed above). OpenProp employs a vortex lattice wake model in which the propeller is represented by a set of lifting lines partitioned into panels. Each panel is surrounded by a "horseshoe vortex," which consists of a vortex spanning the panel and one trailing off each panel endpoint. As it turns out, the assumed pitch angle for these trailing vortices drastically affects the numerical stability of the model. In OpenProp versions 2.4 and prior (as well as other lifting line codes, such as PLL [4], the pitch angle was interpolated from the inflow angle at the control point radii. Although this is physically realistic, it results in some mathematical inconsistency between the wake pitch angle and it's corresponding panel induction factors. This inconsistency led to numerical instability in heavily-loaded design conditions such as bollard pull.

Recent updates to OpenProp have overcome this inconsistency by altering the way the pitch of the trailing vortex filaments are defined. Rather than interpolating a pitch angle for the intersection of each panel and calculating the corresponding shed circulation as the difference between the adjacent panel circulations, the new model defines a single pitch angle for each panel. Two trailing vortices are shed at each panel abutment rather than one, with the circulation

of each shed vortex filament corresponding to its respective panel's bound circulation and the pitch analytically related to the pitch angle of the respective panel. This new formulation of the wake alignment algorithm introduced greater robustness to the model, enabling design optimization convergence over a much wider range of design conditions, including bollard pull.

Propeller Design

Propellers for this thesis were designed to test three aspects of OpenProp design capability. Propeller 1 tested the open (non-ducted) propeller design, Propeller 2 tested the ducted propeller design with a gap between the blade tip and the duct surface, and Propeller 3 tested the ducted propeller design with no gap between the blade tip and the duct surface. The only design input difference between Propellers 2 and 3 was that the Propeller 2 design radius input was equal to the duct surface radius of 9 inches, and the Propeller 3 radius input was 1/16 inch less than the duct surface radius of 9 inches.

The following procedure was used to conduct the design of the three propellers.

1. Determine initial design constraints. In this case, several design features were limited by the sensors and capability of the test fixture to be used for design validation and described in Chapter 4. In order to ensure a sufficient safety margin and maintain the ability to test the propellers at a wide range of off design points, the design thrust was set to 20 lbf. The hub diameter for each propeller was set to match the test fixture body diameter. The propeller and duct diameters were chosen to be reasonably sized compared to the test fixture diameter. The constraints applied for each propeller design are given in Table 1.

Design Constraint	Prop 1	Prop 2	Prop 3	Basis
Specified Thrust, T	20 lbf	20 lbf	20 lbf	Sensor limitation (50 lbf)
Max Torque, Q	3 ft-lbf	3 ft-lbf	3 ft-lbf	Sensor limitation (6 ft-lbf)
Hub Diameter, D_{hub}	3.5 in	3.5 in	3.5 in	Motor Housing
Propeller Diameter, D	12 in	8.9375 in	9 in	Reasonable Sizing
Duct Diameter	N/A	9 in	9 in	Reasonable Sizing
Max Propeller Speed, N	1500 rpm	1500 rpm	1500 rpm	Motor capability

Table 1: Parametric Design Constraints

2. Perform parametric design study. The OpenProp 3.2 optimization algorithm was run with the constraints determined in step 1, the settings given in Table 2, and varying the propeller speed and number of blades. Input scripts for each propeller are included in Appendix A.

Setting	Value	Notes
Vs	1	Used by OpenProp as a reference speed for normalization
XVA	0.001	$XVA = V_A/V_s$ Setting XVA approximately equal to zero is how bollard pull is implemented in OpenProp
Mp	20	Number of vortex panels over the radius
Np	20	Number of points over the chord for geometry plots
Tau	1	$Tau = T_{prop}/(T_{prop}+T_{duct})$ Tau equal to one indicates that no thrust is generated from the duct, which is the case for a straight tunnel with no curvature.
Viscous_flag	1	Viscous force effects were included in the algorithm.
Hub_flag	1	Hub effects were included in the algorithm.
Chord_flag	1	Chord optimization was included in the algorithm.
EppsOptimizer02_flag	1	The standard Epps optimization algorithm is used.

Table 2: OpenProp Design Settings

The design results (Figure 1 - Figure 3) clearly showed that the 5 bladed designs yielded the highest Quality Factors (as defined in Chapter 1). The design speeds (of 600, 900, and 900 rpm respectively) were chosen balancing a desire for the highest test speed before Quality Factor dropped too much, but low enough speed to enable testing over a wide range without overloading the sensors.

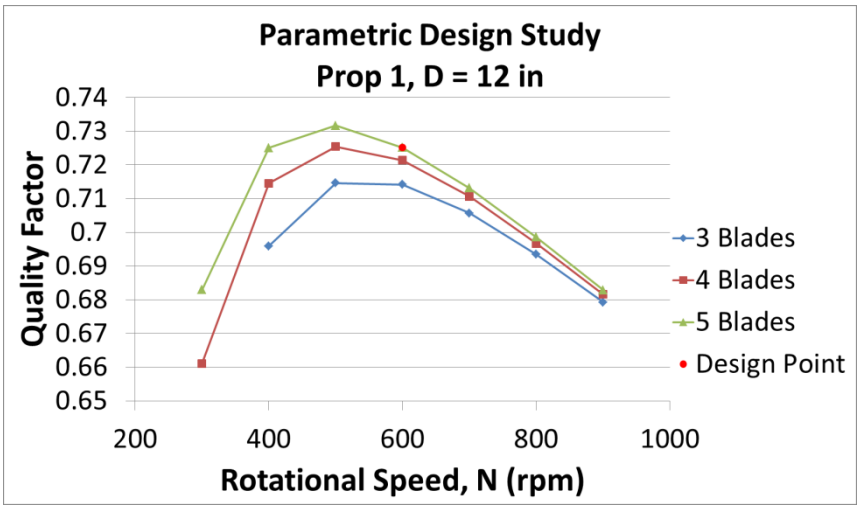

Figure 1: Prop 1 Parametric Design

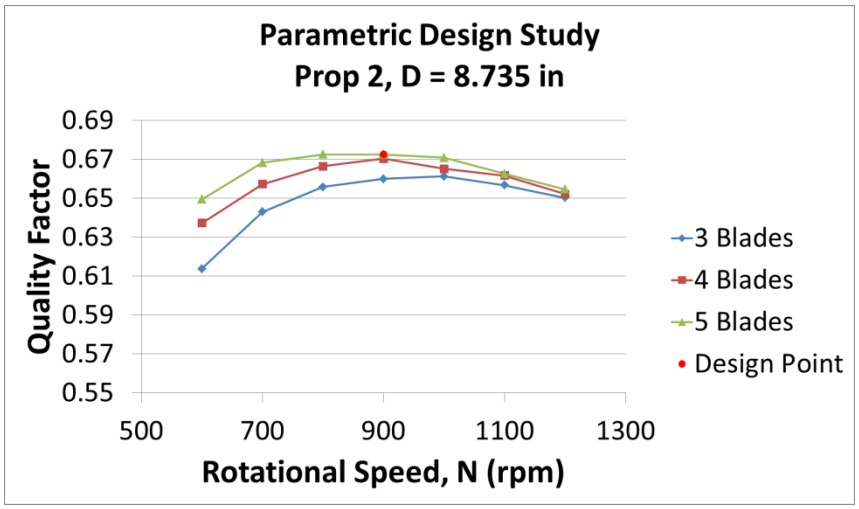

Figure 2: Prop 2 Parametric Design

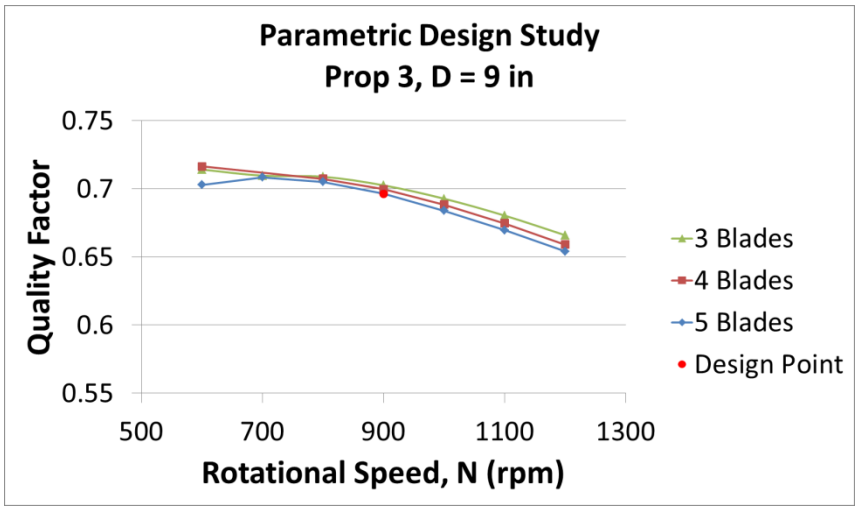

Figure 3: Prop 3 Parametric Design

3. Determine the blade geometry. Check the chord and thickness distribution generated for the selected design point. Ensure that the shape of the blade looks reasonable. For this project, Propeller 1 initially had unusually skinny blades (small chord lengths). In order to increase chord length, the maximum allowable lift coefficient along the blade was decreased and the design was run again to generate greater chord lengths and wider blades. While Propeller 2 and Propeller 3 maintained a large similarity to each other, the biggest difference came in the circulation distribution near the tip, which resulted in a change in tip geometry. For Propeller 2 (which had the small gap), the circulation (and hence loading) must go to zero at the tip. This tends to result in nicely rounded blade tips. For the Propeller 3 (which had no gap), the circulation is treated in the same manner as at the hub and a reflected circulation is assumed to continue into the duct wall, resulting in non-zero tip circulation and blades that widen at the tip. The final design parameters are given in Table 3. Figure 4 shows the chord and circulation distribution for each propeller and Figure 5 shows 2D and 3D images of the propellers.

Parameter	Prop 1	Prop 2	Prop 3	Description
Z	5	5	5	Number of blades
N	600	900	900	Rotation rate [rpm]
D	12	8.735	9	Diameter [in]
$\mathbf{V_s}$	1	1	1	Reference Velocity [m/s]
$\mathbf{D_{hub}}$	3.5	3.5	3.5	Hub diameter [in]
M	20	20	20	Number of panels
ρ	1000	1000	1000	Water density [kg/m^3]
$\mathbf{C_{L,max}}$	0.3	0.5	0.5	Maximum allowable lift coefficient

Table 3: Propeller Design Parameters

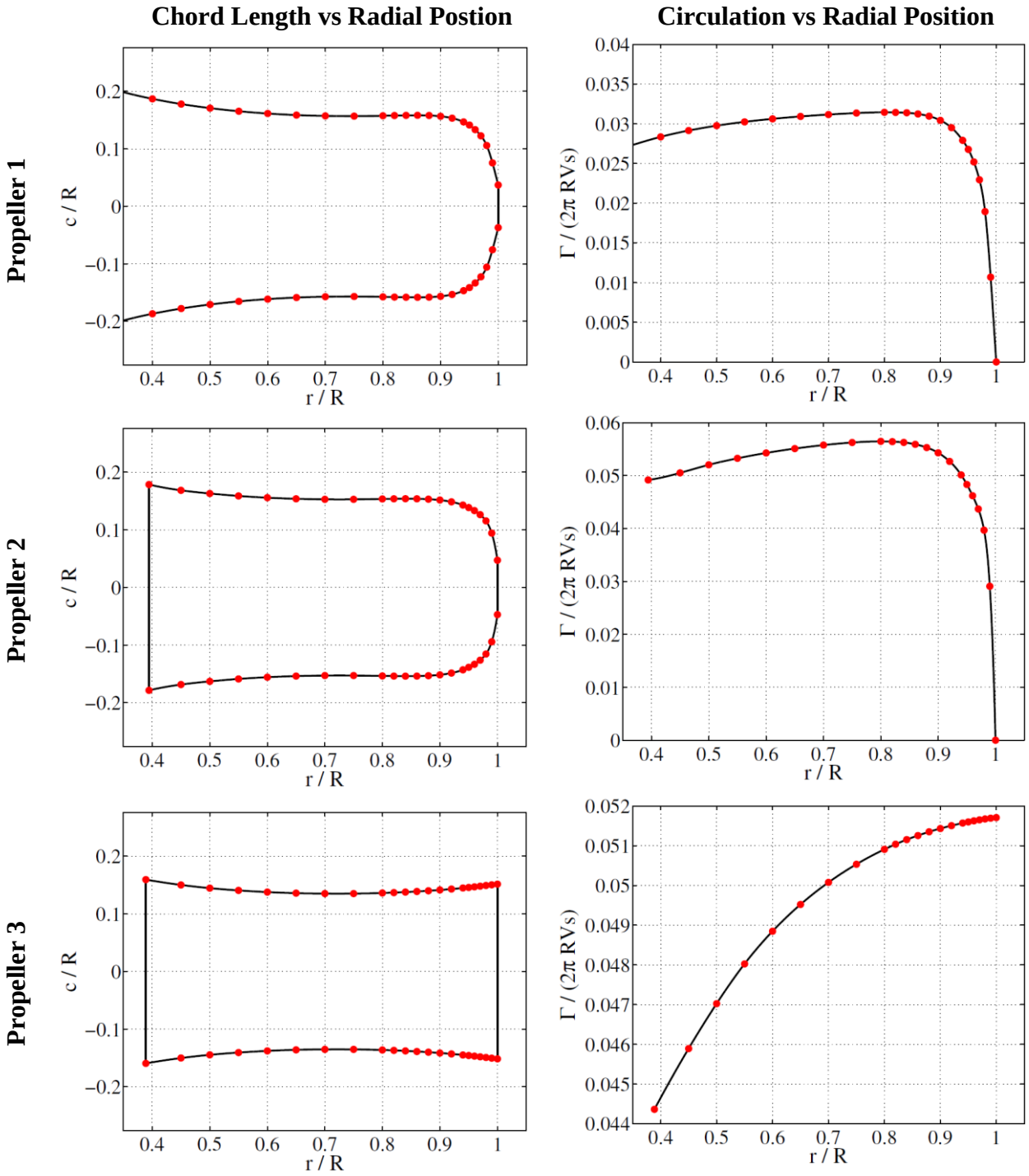

Figure 4: Propeller Chord and Circulation Plots

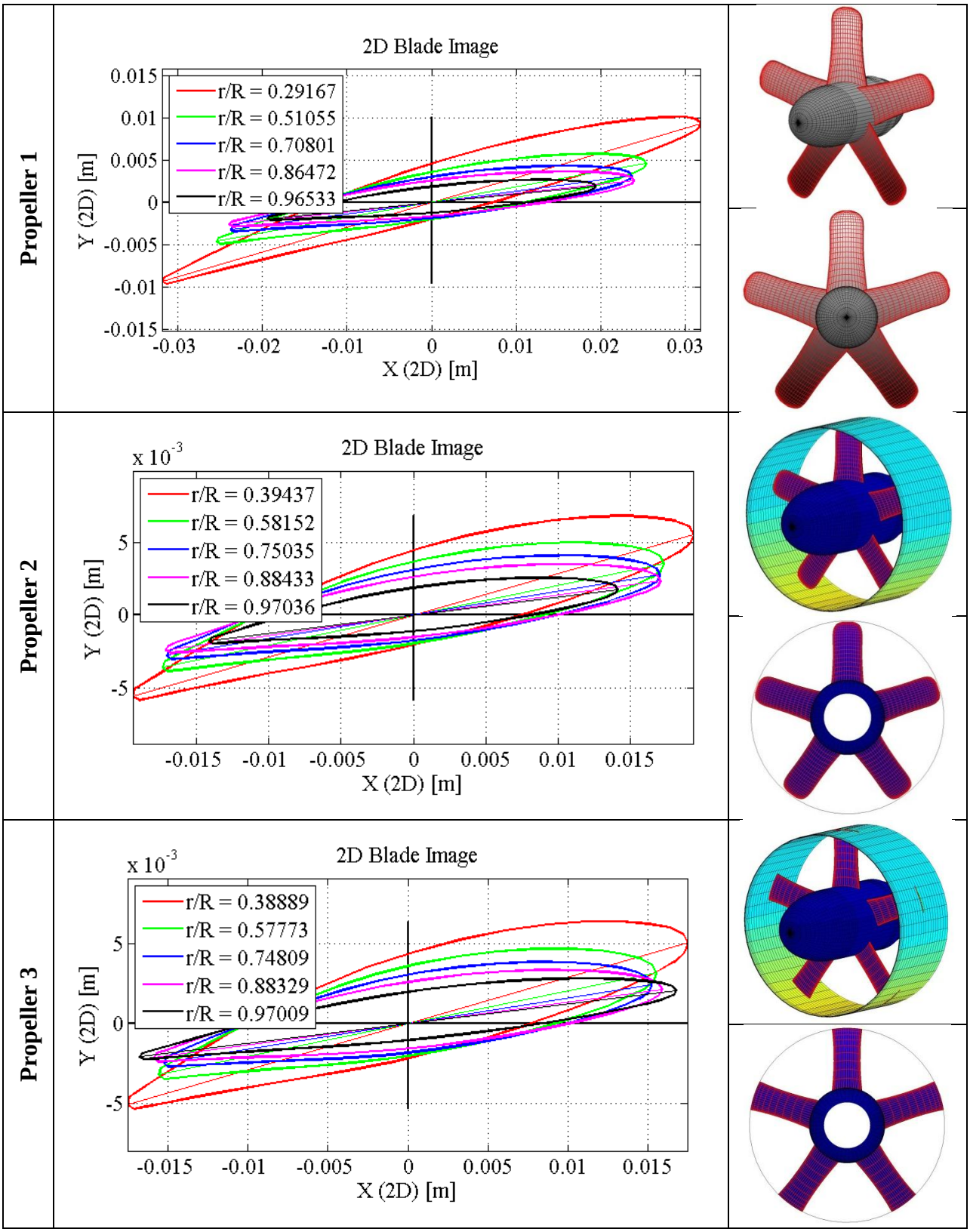

Figure 5: Propeller 2D and 3D Images

The original intent of this thesis was to continue the OpenProp design to advanced design of the propeller using PBD 14.36 (a 3D lifting surface Propeller Blade Design program) and MTFLOW (a Multi-passage ThroughFLOW fluid dynamics solver). A Matlab script for was written to interpolate PBD input parameters from OpenProp design data (and is included in Appendix A). Much work was put into coupling the two programs and developing the design process with them, but uncertainty arose as to the nature of the MTFLOW calculations for zero initial flow and whether they could adequately handle zero-flow conditions. Due to time constraints, design efforts with PBD and MTFLOW were suspended.

Instead, the OpenProp designs were adjusted with standard lifting surface corrections [23]. 3D geometries for the corrected designs of each propeller were generated in OpenProp and transferred to Solidworks using a macro written for this purpose. Solidworks was used to take the blade shape designed in OpenProp, model it around a hub, and export it in a format suitable for 3D printing. Full propeller design parameter and geometry tables are included in Appendix A.

A detailed account of the PBD/MTFLOW coupling process and descriptions of the subroutines is included as Appendix B. A sample copy of all the PBD/MTFLOW input files generated to design Propeller 1 is included as Appendix C

Chapter 3: Test Fixture

Motor and Motor Controller

This project relied heavily on the work done by Ketcham for his 2010 M.S. Thesis [19]. Ketcham designed and built a test fixture for marine propellers and turbines. This apparatus consists of a DC brushless motor which drives a shaft with an integrated dual strain gauge torque and thrust sensor. The motor, a Parker kit motor (model K089300-7Y2), also has a built-in Hall Effect sensor to provide a feedback signal. The motor, sensors, and sensor amplifiers are enclosed in a single housing with a watertight penetration for the drive shaft and a non-watertight penetration that extends through a pipe structure that serves as the cable access point and the support system for the assembly (Figure 6). The motor is driven and controlled by a test fixture designed by students at the University of Maine using a Copley Xenus XTL-230-40 amplifier/motor controller. The test fixture has the operational limits given in Table 4. A photograph and schematic of the electrical components of the motor controller are reproduced from Ketcham's thesis as Figure 8 and Figure 7. Commands to the motor controller are delivered via USB connection from a computer running Copley CME2 software.

Figure 6: Motor Assembly

Limits	Value	Basis
Torque	6 ft-lbs	Sensor limitation
Thrust	50 lbf	Sensor limitation
RPM	1500 rpm	Peak capability of motor
Current	18 amps	Peak capability of motor
Voltage	240 V-AC 300 V-DC	Required supply voltage Maximum controller output voltage

Table 4: Test Fixture Limitations *(Courtesy Ketcham)*

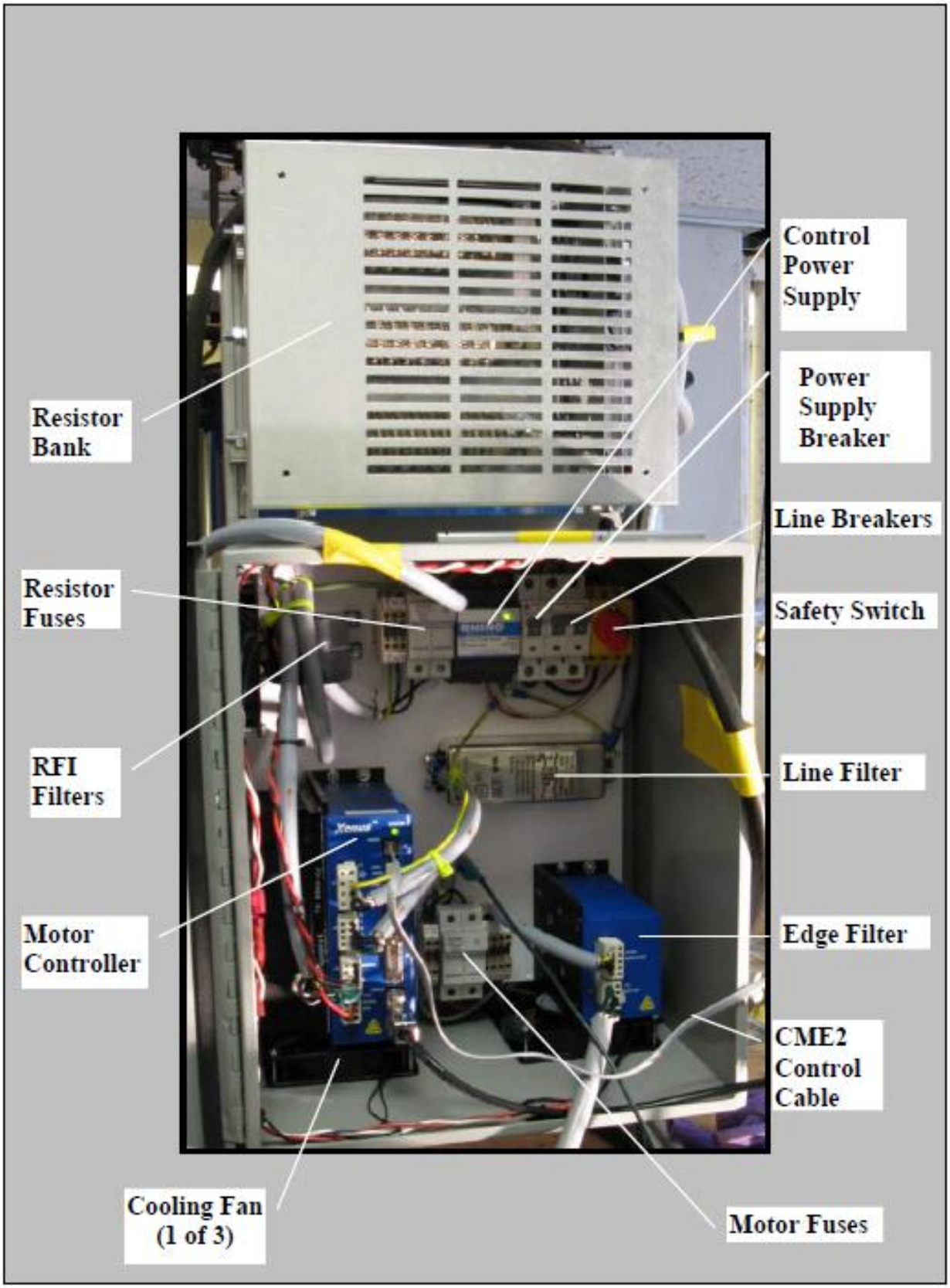

Figure 7: Electrical Components (*Courtesy Ketcham*)

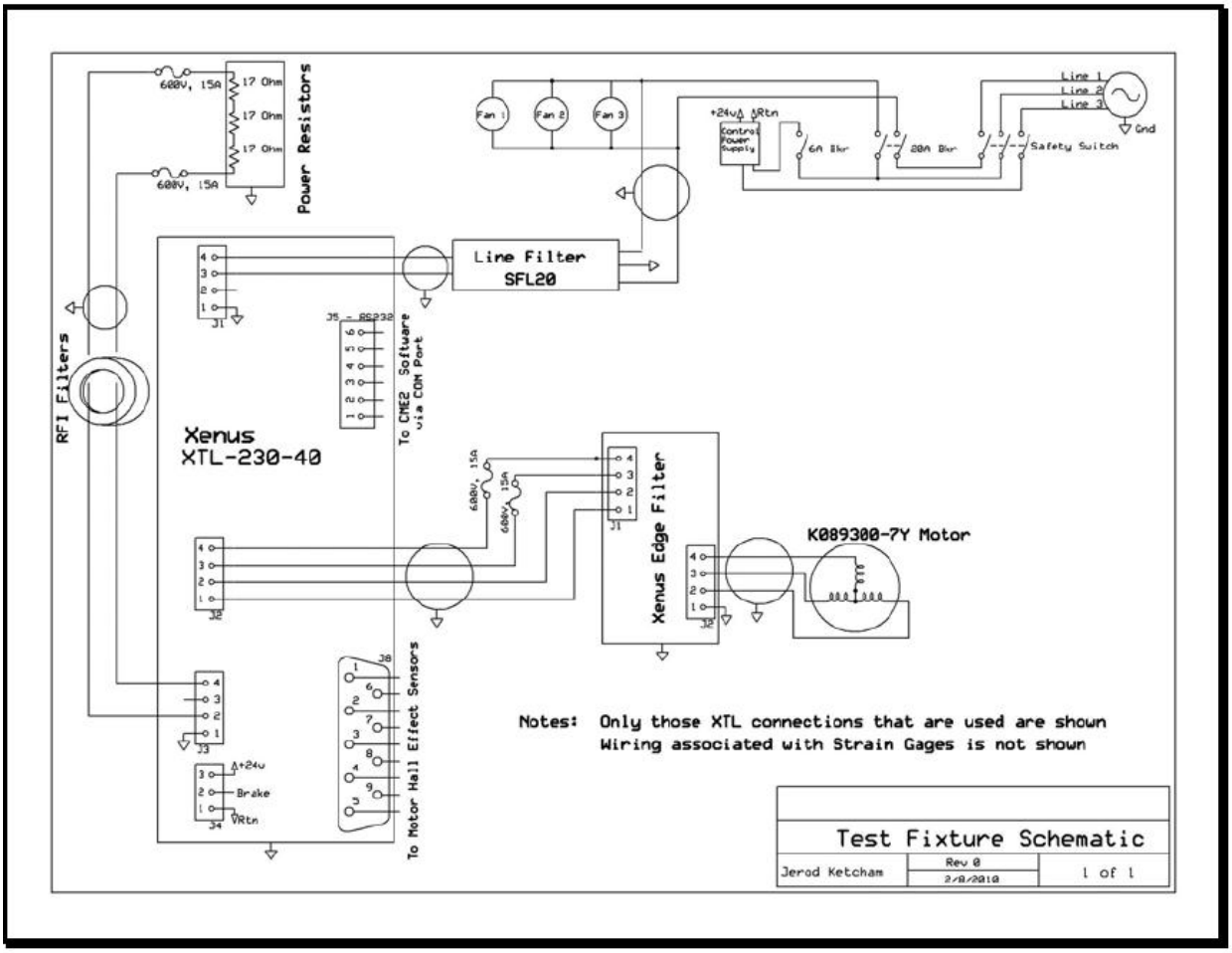

Figure 8: Schematic of Enclosure Electrical Components (*Courtesy Ketcham*)

CME2 Software Description and Use

Copley Controls CME2 software is used to program the Xenus amplifier and generate the command signals for control input to the motor. Prior to use, the amplifier must be loaded with the correct motor characteristic parameters. It must also be calibrated to set the appropriate feedback and control gains to achieve the desired response profile. For this project, the motor was operated in the software programmed velocity control mode. The gains were adjusted to develop a response profile with a quick response time and minimal overshoot. This resulted in high torque and thrust transient values when a velocity change was ordered, but ensured the system reached steady state operation quickly for data acquisition. Motor and amplifier parameters can be saved to and stored on both the Xenus amplifier flash memory and to the local hard drive of the computer operating CME2 as *.ccm and *.ccx files. The *.ccm and *.ccx files used for this project are included in the electronic materials library for this thesis. Additionally, the motor parameters and gain values used are summarized in Table 5. Step by step instructions for amplifier setup and initialization are included as Appendix D.

Motor Parameters					
Manufacturer	Parker		**Back emf Constant**	112.09	V/krpm
Units	Metric		**Resistance**	2.97	ohms
Model Number	K089300-7Y2		**Inductance**	10.98	mH
Motor Inertia	3.601	$kg{\cdot}cm^2$	**Hall Count Multiplier**	1	
Number of Poles	12		**Counts per Rev**	36	
Peak Torque	17.95	N·m	**Brake/Stop Delay Time**	10	ms
Continuous Torque	7.13	N·m	**Brake/Stop Activation Velocity**	1	rpm
Velocity Limit	1500	rpm	**PWM Delay Brake/Stop Response Time**	100	ms
Torque Constant	0.927	N·m/Apk			

Gains	
Vp	32000
Vi	32000
Cp	5000
Ci	1000
Output Filter FC	200

Velocity Parameters		
Velocity Tracking Window	1500	rpm
Velocity Tracking Time	100	ms
Acceleration Limit	5000	rps^2
Deceleration Limit	5000	rps^2

Table 5: CME2 Motor and Amplifier Parameters

CME2 usage notes:

A potentially hazardous aspect of the system as configured is that on startup, the amplifier defaults to its enabled configuration. This means that if a non-zero command velocity was the last one input prior to shutdown, the system will be energized and begin moving as soon as the system is next powered on. In order to avoid this, care must be taken to first power on only the motor controller's single, power supply breaker, then open CME2 and verify the command state prior to activating the double line breakers as shown in Figure 7, above. This prevents the control power from being sent to activate the motor inadvertently.

The system has both a hardware enable/disable capability and a software enable/disable capability. The hardware enable/disable is determined by the position of the double line breaker as described above. The software enable/disable button is located on the amplifier control panel in CME2. Additionally, when CME2 is the active window, hitting the F12 key serves as a quick disable shortcut.

When the amplifier basic setup is configured for software programmed velocity control, the main CME2 window displays a feedback control loop with a "Programmed Velocity" block. When the amplifier is fully enabled, motor control is commanded simply by clicking on the "Programmed Velocity" block and entering in the desired velocity as shown in .Figure 9.

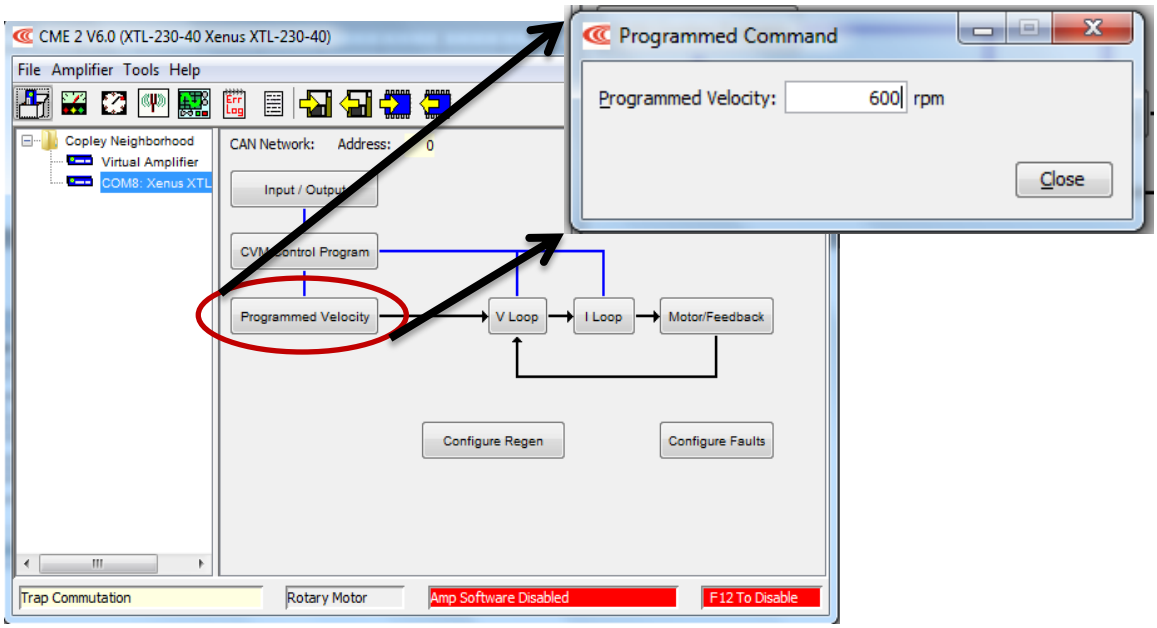

Figure 9: Inputting Command Velocity in CME2

Thrust and Torque Sensor

Sensor/Amplifier Description

The thrust and torque sensor installed in the test fixture consists of two strain gauges; one for each measurement. They are paired with ruggedized RaeTech Motorsports programmable amplifiers with an output range of 0.1 – 4.9 V. A wiring diagram for the sensor input/output connections is included at the end of this chapter with the equipment arrangement diagram.

Calibration

Thrust calibration was first performed by suspending a series of known weights from the end of the propeller shaft and measuring the output voltage with Logger Pro 3.5.0 software via a Vernier Lab Pro data acquisition unit. The first calibration runs were done with a sensor supply voltage of +5V DC. A linear relationship was confirmed and calibration constants were calculated. The resulting constants were valid but did not initially match the Ketcham calibration.

Torque calibration was then attempted by changing the control style of the motor to hold a commanded position. A 2 ft bar was fixed to the shaft for application of a series of torque values. The motor does not have an accurate position encoder however, and relied on the Hall Effect sensor output for position monitoring. As a result, when the rotor position was only slightly perturbed, the system began oscillating out of control while trying to return to its starting point. This made a direct torque calibration highly impractical. The original torque calibration performed by Ketcham was completed before the sensor was installed in the text fixture and closed down for waterproofing. In order to avoid the need to open the motor housing and risk the watertight integrity of the fixture, additional thrust calibration runs were completed in an

attempt to match Ketcham's calibration factor. Matching calibration slope values were found (within approximately 1%) when the supply voltage to the sensor amplifiers was +8V DC. A significant difference in the offset was observed between the measured calibration and the Ketcham data, however. This was attributed as most likely the effect of the dynamic seals surrounding the propeller shaft which contribute to its support. Similarly, local zero offsets were observed to fluctuate throughout the test series, so care was taken to record the relevant zero point for each measurement over the entire series. These local zero values were ultimately used to modify their respective data point calibrations. It was assumed that since Ketcham's thrust sensor calibration was still valid, so was the torque calibration. As such, Ketcham's calibration constants were used during data processing for translating output thrust and torque voltage into Newtons and Newton-meters. The calibration constants are shown in Table 6. These constants are only valid when +8V DC is supplied to both sensors. The matching thrust calibration data is shown in Figure 10. For consistency, both of the Ketcham calibration constants were used for subsequent data processing.

With +8 VDC supply to sensors	Ketcham Values	Measured Values	Difference (%)
Thrust Calibration Constant [N/V]	-120.62	-121.63	0.8303
Torque Calibration Constant [N·m/V]	3.5161		

Table 6: Thrust and Torque Calibration Constants

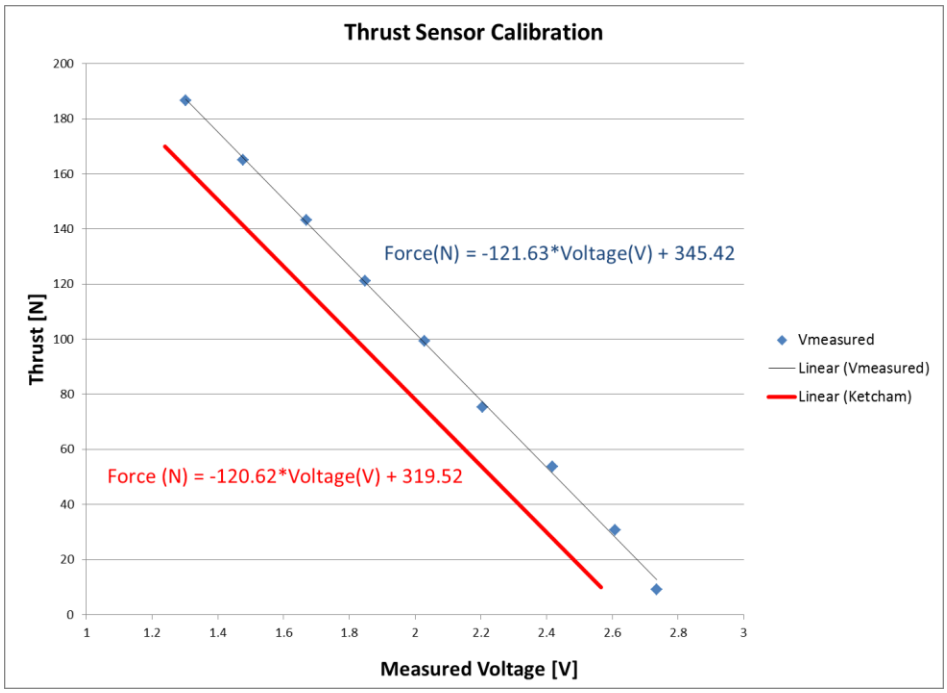

Figure 10: Thrust Sensor Calibration

Data Acquisition

Thrust and torque sensor data was acquired using a Vernier Lab Pro (Figure 11) and the accompanying Logger Pro 3.5.0 software. A Vernier flow rate sensor (Figure 12) was used to measure induced flow. Designed for use with the Vernier Lab Pro, the flow rate sensor did not require an external power supply. Additionally, when used with Logger Pro 3.5.0, the output values have units of meters per second, so calibration was not required.

Figure 11: Vernier Lab Pro

Figure 12: Vernier Flow Rate Sensor

Framing and Support Structure

A support structure to suspend the motor in the towing tank was constructed from 80/20 aluminum framing. An existing roller-style hanging structure was borrowed from an unused carriage at the towing tank and adapted for use with the 80/20 frame. The motor support bar was clamped to the frame at two points by stanchion clamps. This allowed for the vertical position and orientation of the motor to be adusted for the different diameter propellers between test runs. Two mounting frames for the tunnel duct where designed and cut by waterjet from a sheet of 0.5 inch ABS plastic. The duct consisted of a 33 inch length of 9 inch diameter PVC ducting. The duct was non-cambered. Figure 13 and Figure 14 show the frame and tunnel as designed in Solidworks for both the open and ducted propeller configurations.

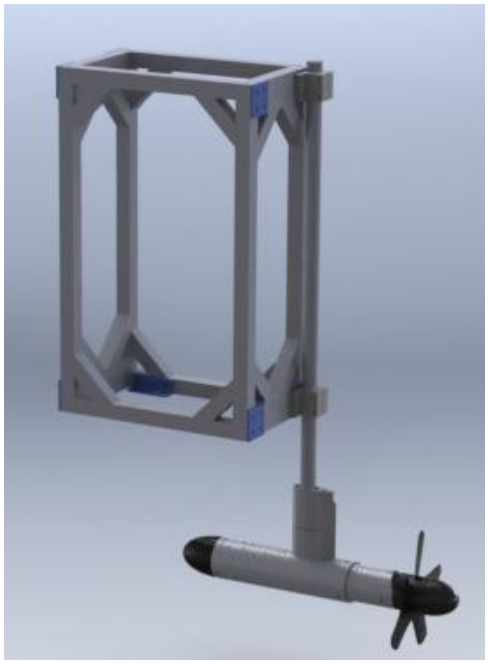

Figure 13: Open Propeller Frame Arrangement

Figure 14: Ducted Propeller Frame Arrangement

Equipment Arrangement

The test equipment for running the experiments was arranged according to layout in Figure 15. The motor is controlled through user inputs to CME2. CME2 initializes the controller amplifier settings with motor characteristics and control gains. It is used in the velocity control mode to order a commanded velocity. The controller applies power to the motor and maintains the commanded velocity via feedback from the Hall Effect sensor. An 8V DC power supply provides power to the thrust and torque sensors. The thrust and torque sensor output is read by a Vernier Lab Pro data acquisition unit which delivers their respective voltage output values to the computer running Vernier Logger Pro 3.5.0.

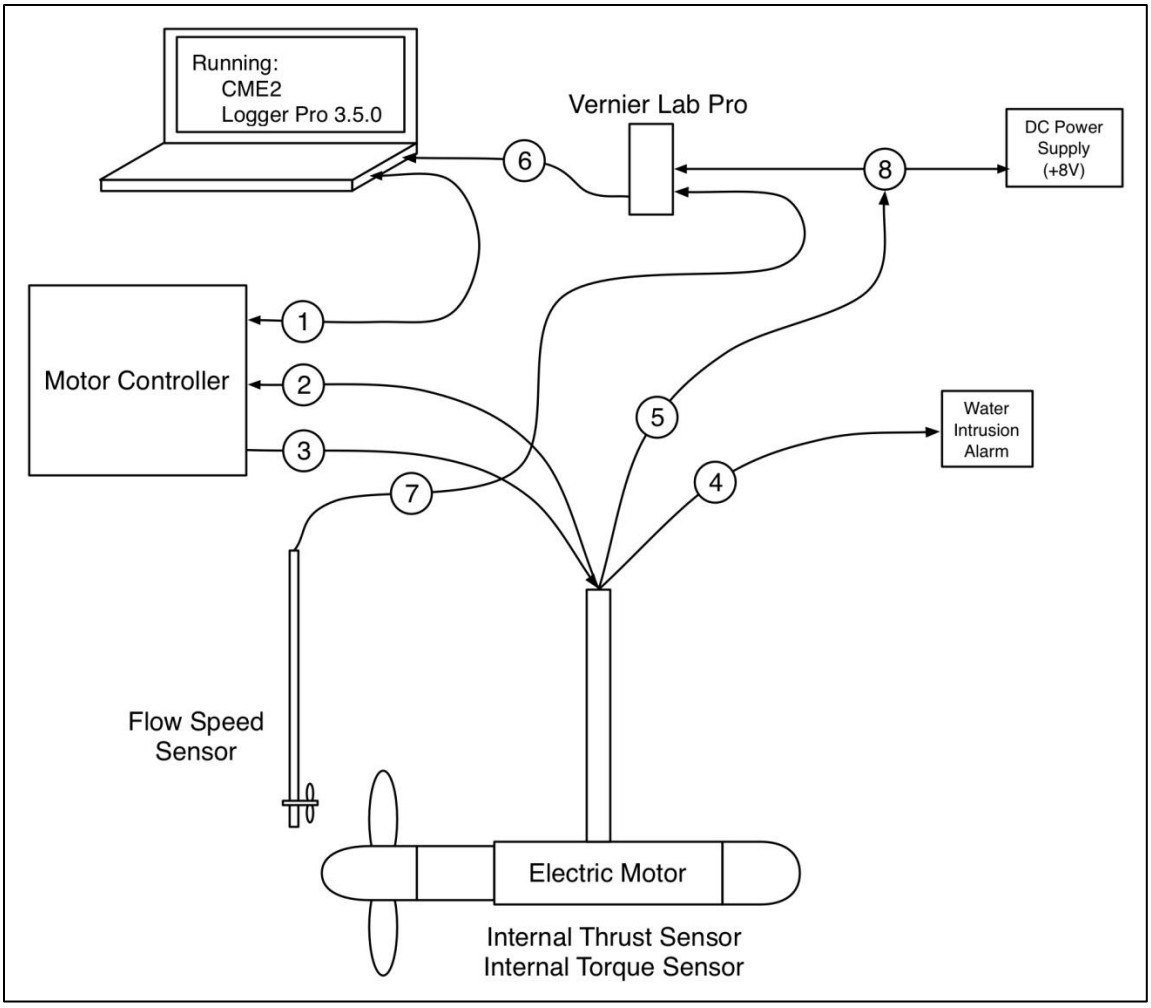

Figure 15: Test Equipment Arrangement

Figure 15 Legend:

1. Gray Ethernet USB cable connection for communication between the computer running CME2 and the motor controller.
2. Black Ethernet cable with an RJ45 connector for Hall Effect Sensor output from the motor to the motor controller.
3. Black 4-wire bundled cable to provide control power to the motor.
4. Red 2-wire (twisted pair) connection to water intrusion alarm.
5. Black 8-wire cable for sensor input and output.
6. Gray USB cable connecting the Logger Pro data acquisition unit to the computer.
7. Black signal transfer cable with Logger Pro input connection.
8. See Figure 16 for wiring diagram and color scheme detail of sensor hookup.

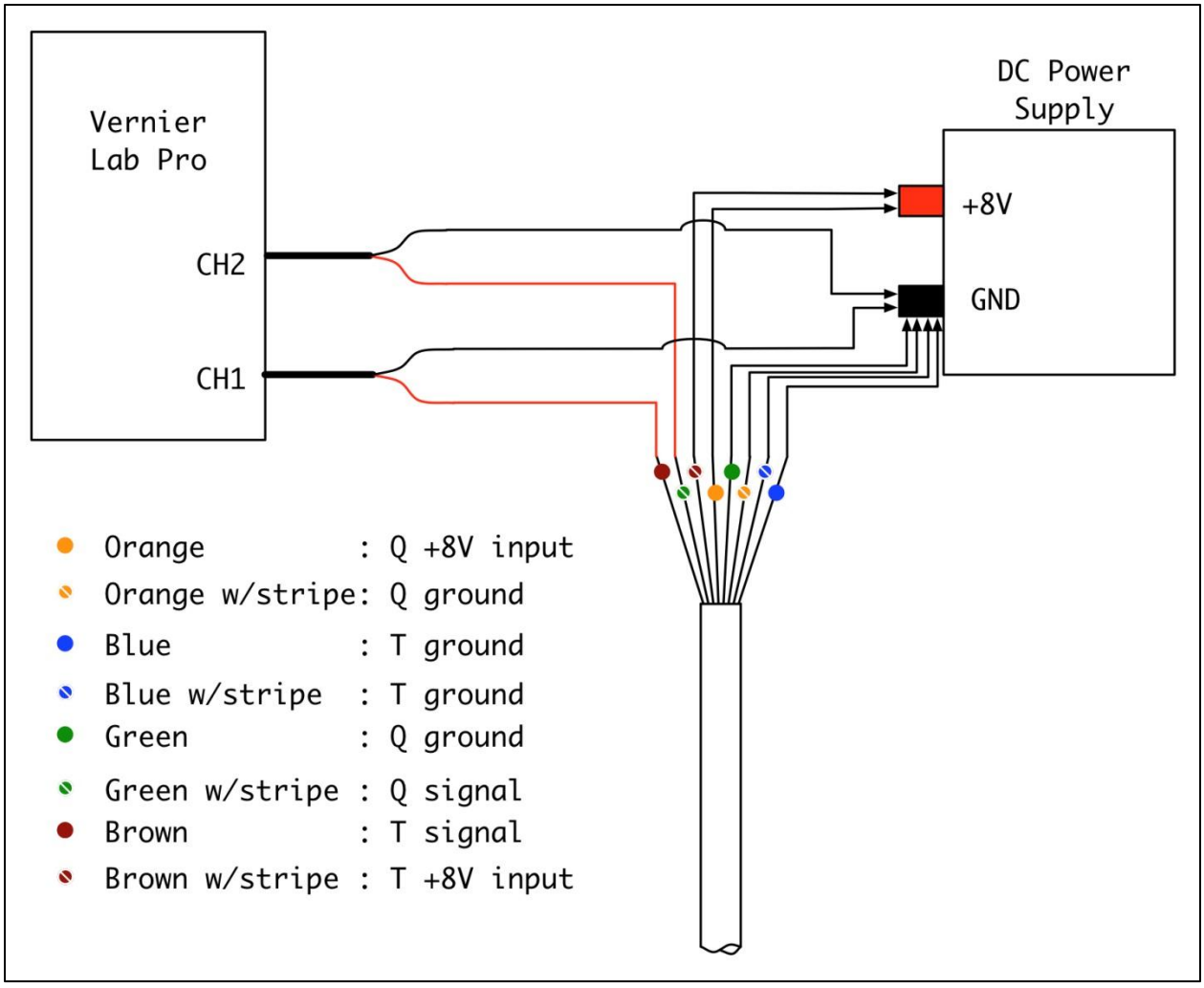

Figure 16: Sensor Connection Wiring Diagram Detail

Chapter 4: Experimental Procedure

Testing was conducted at the MIT Towing Tank and at the MIT Propeller Tunnel. At the towing tank, all three propellers were tested operating from 0 rpm up to 300 rpm greater than their respective design speeds. The bare hub friction torque and induced flow speeds for each propeller was also measured.

The Towing Tank has the following characteristics:

Length:	100 ft
Width:	8 ft 6 in
Depth:	3 ft 4 in

Experiments were conducted to measure the following properties:

Thrust and torque applied to the propeller shaft.
Flow speed induced by the propeller.
Hub drag from induced flow.
"No-load" friction torque.

Test results for all experiments were recorded in spreadsheets or as raw data files readable by Logger Pro 3.5.0. All data files discussed herein are included with the electronic materials accompanying this thesis.

Towing Tank Tests

Most of a day was spent setting up the equipment and suspending the frame and motor from the towing tank rail. Due to its weight, the frame and motor were particularly difficult to put in place. Two people were required to get it set up correctly. The borrowed carriage which suspended the frame from the rail was designed to glide smoothly along the length of the towing tank and as such, did not have a built in braking system. To conduct the bollard pull propeller tests, the frame was required to remain stationary, so an improvised system using materials available on site was devised to anchor the frame in place. The motor was initially set up with the bare hub installed and the test fixture in the arrangement for testing Propeller 1, with the motor positioned as low as possible beneath the frame and with the drive shaft oriented opposite to the frame as shown in Figure 13. In this condition, the center of the hub was 18 inches below the free surface.

Table 7 lists the test speeds for which data was recorded for each propeller in both the forward and reverse directions over the course of the experiments. Positive rotation is defined in the direction that produces forward thrust.

Motor Test Velocity (rpm)			
Bare Hub	**Prop 1**	**Prop 2**	**Prop 3**
0	0	0	0
±50	±50	±50	±50
±100	±100	±100	±100
±150	±150	±150	±150
±200	±200	±200	±200
±250	±250	±250	±250
±300	±300	±300	±300
±400	±400	±400	±400
±500	±500	±500	±500
±600	±600	±600	±600
±700	±700	±700	±700
±800	±800	±800	±800
±900	±900	±900	±900
±1000	+1000	±1000	±1000
±1100		±1100	±1100
±1200		±1200	±1200
±1300		+1300	+1300
±1400		+1400	+1400
±1500			

Table 7: Motor Test Velocities

The first tests conducted were to characterize the friction torque generated by running a bare hub at every expected test condition. The bare hub steady state data was recorded in Logger Pro for each speed for approximately 10 seconds. The sampling rate was set to 250 Hz, the maximum rate at which Logger Pro was capable of recording. The logger Pro statistics function was used following each run to calculate the average voltage output from the torque sensor over the 10 seconds. This value was then recorded into an excel spreadsheet for data compilation. The spreadsheet is named "120422_Bare_Hub_Torque.xls" and is included with the electronic materials for this thesis. A sample complete data set for N = +500rpm is saved as "500rpm_run1_example_data.cmbl" and also included with the electronic materials.

Once the bare hub testing was complete, the bare hub was removed from the motor and Propeller 1 was installed.

The following procedure was followed to measure thrust and torque for each propeller:

1. Ensure CME2 is open, software and hardware enabled, and the programmed velocity control is set to 0 rpm.
2. Ensure Logger Pro 3.5.0 is open, receiving data from the sensor, the sampling rate is set to 250 Hz, and continuous sampling is activated.
3. Begin recording data in Logger Pro.
4. Wait approximately 10 seconds to capture 0 rpm data.

5. Input the test speed in CME2 while observing thrust and torque response. If the test speed is greater than 600 rpm, incrementally step up the speed until the test speed is reached in order to reduce transient loads on the sensor.
6. Allow time for the transient response to settle, then capture approximately 10 seconds of steady state values.
7. Set test speed to 0 rpm. If the test speed is greater than 600 rpm, incrementally step down the speed in order to reduce transient loads on the sensor.
8. Allow time for the transient response to settle, then capture approximately 10 seconds of 0 rpm data.
9. Use the statistics function of Logger Pro to find the average steady state value for thrust and torque and enter it into the test spreadsheet.
10. Save the raw data captured by Logger Pro as a *.cmbl file[1].
11. Repeat steps 3-9 until data has been collected for the full range of speeds.

Individual Logger Pro data files with the raw sensor voltage output data were recorded and saved with the following naming convention:

Positive rotation:	prop**x**_run**y**_**zzz**rpm.cmbl
Negative rotation:	prop**x**_run**y**_n**zzz**rpm.cmbl
Where:	**x** = propeller number, **y** = run number, **zzz** = rpm value

The test series for Propeller 1 was begun and partially completed with the propeller hub center 18 inches below the free surface. As the test speed increased however, large surface effects began to manifest in the form of vortices connecting to the surface a short distance upstream of the propeller, creating a whirlpool effect which (if allowed to continue long enough) introduced air into the propeller flow stream and wake. To compensate, the propeller was lowered to a depth of 24 inches below the free surface. This depth was chosen to find a balance between the influence of surface and bottom effects on the propeller's performance. The extra space between the propeller flow stream and the surface provided a greater buffer from surface effects. While the same whirlpools would still form at the new depth given a long enough run duration, it allowed steady state operations to be reached and measurements recorded at each run speed before the vorticity could build up significantly enough to have a noticeable effect. Following the depth adjustment, the test series was restarted from the beginning.

Data was recorded in an excel spreadsheet entitled "120423_Prop1_Experimental_Data.xls." A single full run measuring thrust and torque at each test speed was completed the first day. Testing was suspended for the night following completion of the test run.

It was originally intended to remove the motor from the water and not leave it submerged in the tank for extended periods of time, especially when unattended. The setup had proven so difficult earlier in the day however, that it was determined that the best course of action would be to leave the motor in place and trust that the seals would hold. Power was disconnected overnight in case of leaks.

A leak test conducted upon return showed no water intrusion had occurred overnight.

[1] Raw data files for individual test speeds were not saved for the first two runs with Propeller 1.

Testing of Propeller 1 was resumed with a repeat of the previous day's tests. The speed range was limited to 900 rpm in the positive direction for this run due to the visible and audible stresses on the propeller at 1000 rpm.

When the second test run was complete, Propeller 1 was removed from the motor and the blades sanded down to smooth out some remaining irregularity on the blade surfaces and ensure uniformity. Some minor affect to the torque measurements was expected as a result of the reduced friction coefficient. No significant difference was discernible upon testing, however.

Propeller 1 was returned to the motor and a third test run was performed for the full range of speeds.

Following the third run of Propeller 1 tests, the test fixture was reset to hold Propeller 2 as shown in Figure 14. Propeller 1 was replaced with Propeller 2. The tunnel support mounts were attached to the frame and the tunnel was slid into place over the propeller. Careful alignment of the propeller inside the tunnel was accomplished by hand before the stanchion clamps were tightened down to secure the motor in place. The alignment required two people in order to ensure there was no physical contact between the blade tips and the inner duct surface; one to submerge in the tank for a close view and to perform manual adjustment, the other to operate the motor at very slow speeds for observation.

Two complete data sets were taken for Propeller 2 operating over the speed ranges shown in Table 7 and following the specified test procedure.

Testing for the day concluded following the completion of the Propeller 2 test runs. As before, the motor was left submerged in the towing tank overnight after being disconnected from power.

A leak test was conducted again before testing resumed. As before, no water intrusion was detected.

The frame was reset to hold Propeller 3. The tunnel was removed for easier access to the propeller, then Propeller 2 was swapped out with Propeller 3 before the tunnel was re-installed. The propeller alignment inside the tunnel was conducted as before.

A full range of data measurements were taken for Propeller 3 following the specified test procedure.

Following completion of the thrust and torque measurement for each of the three propellers, measurements to characterize the flow speed induced by each propeller were taken using the Vernier Flow Speed sensor introduced in Chapter 4. Figure 17 and Figure 18 show flow speed sensor placement for each of the test fixture propeller configurations.

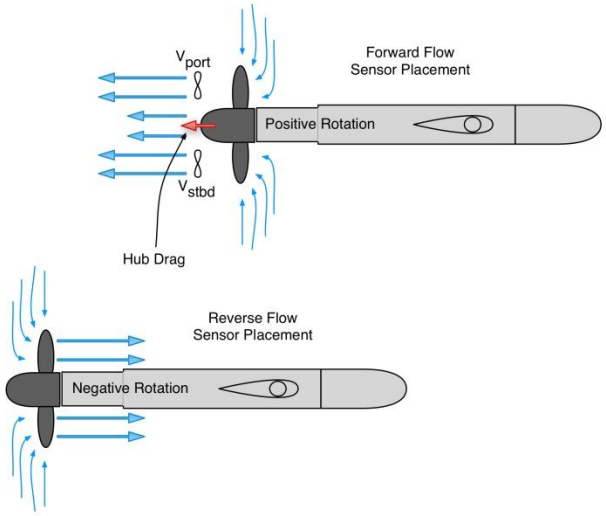

Figure 17: Flow Speed Sensor Placement for Propeller 1

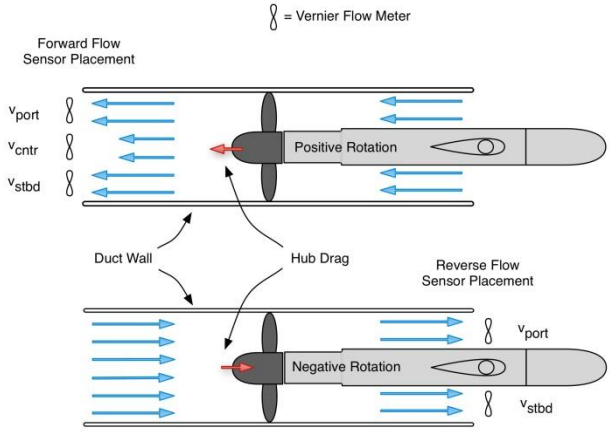

Figure 18: Flow Speed Sensor Placement for Propellers 2 and 3

The following procedure was used to capture the flow speed data.

1. Position flow speed sensor in desired location.
2. Ensure CME2 is open, software and hardware enabled, and the programmed velocity control is set to 0 rpm.
3. Ensure Logger Pro 3.5.0 is open, receiving data from the sensor, the sampling rate is set to 10 Hz, and continuous sampling is activated.
4. Begin recording data in Logger Pro.
5. Wait approximately 10 seconds to capture 0 rpm data.
6. Increment the speed input in CME2 while observing the flow speed response.
7. Record the time of each speed change (used for reading the raw data during processing).
8. Allow time for the transient response to settle, then capture approximately 10 seconds of steady state values.
9. Repeat steps 6-8 until data has been collected for the full range of speeds.

Once each run was complete, the Logger Pro statistics function was used to find the average flow speed recorded for each propeller speed. A sample data file with averages taken is shown in Figure 19. The results were compiled in an Excel spreadsheet entitled "120424 Flow Speed Data.xlsx." Raw data files were saved according to the following naming convention.

Filename: prop**x**_velprof_**yyy**rpm_**zzz**.cmbl
Where : **x** = propeller number,
yyy = pos or neg (rotation),
zzz = port/stbd/cntr

Flow speed measurements began with Propeller 3 since it was already on the motor, followed by tests for Propeller 2, then Propeller 1.

After flow rate had been measured for Propeller 1 with positive rotation, an attempt to measure the induced flow upstream of the propeller resulted in the sensor inadvertently being pulled into the path of the propeller blades. This immediately sheared off all five blades, terminating the test run.

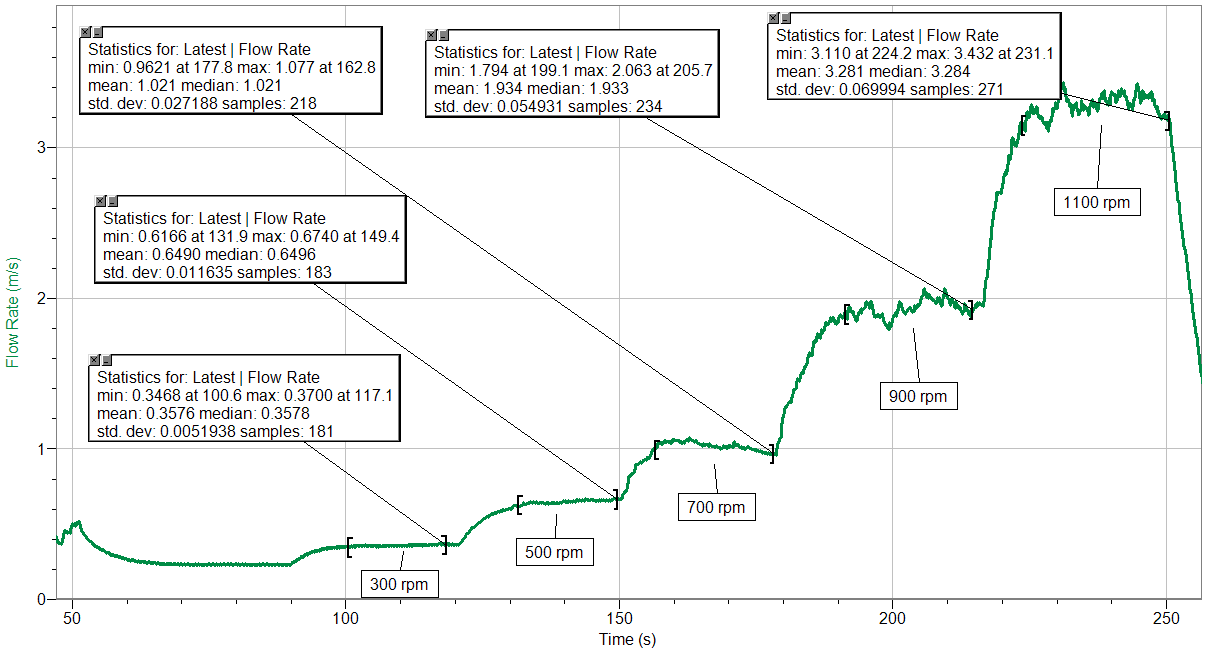

Figure 19: Flow Speed Data for Propeller 3 with Negative Rotation

Testing Notes

Flow speed sensor problems

Care was required to ensure that the cable from the flow speed sensor was kept out of the water. Due to insufficient cable shielding an interference problem was discovered when the cable was in contact with the water and the motor was in operation. When both conditions were met, the interference caused the sensor reading to saturate at its maximum value of 4 m/s. Simply keeping the wire suspended over the water was sufficient to bypass the interference.

Prop change-out difficulties

A manufacturing defect on one of the inner surfaces of Propeller 3 left a lump of additional plastic material that interfered with the position of the washer which held the propeller in place on the drive shaft. When the nut was fastened down tightly over the washer to secure the propeller, the washer deformed the plastic and became entrapped in the extra material. This had no effect on the operation of the propeller during testing, but it caused great difficulty in removing the propeller. While attempting to remove the propeller from the motor, one of the blades was inadvertently snapped off. Fortunately, testing for that propeller had already been completed, but this did serve to render further testing impossible even if desired.

Propeller Tunnel Tests

The MIT Propeller Tunnel at the Marine Hydrodynamics Laboratory was used to measure the hub drag acting on the propeller hub over the range of induced flow speeds. The propeller tunnel is a variable pressure, recirculating water system which uses an impeller to generate flow speeds up to 10 m/s and a flow stabilizer to ensure uniform flow.

The motor with the bare hub attached was installed in the propeller tunnel supported by the tunnel's dynamometer test platform (Figure 20). The thrust and torque sensors were connected with the power supply and Lab Pro as before. Since motor operation was not required, the motor controller was not set up.

Tunnel flow is controlled by setting the impeller frequency. Drag measurements were taken from the hub split over two runs in each direction. The first run measured drag at odd impeller frequencies and the second measured drag at even frequencies. Raw data files were saved according to the following naming convention.

Filename: barehub_run**x**_**yy**hz.cmbl
Where: **x** = run number, **yy** = impeller frequency

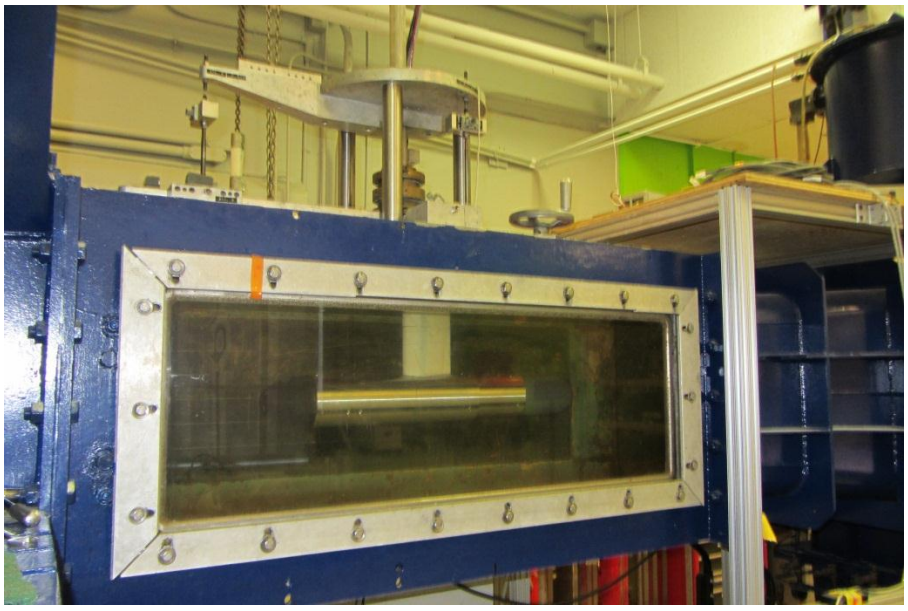

Figure 20: Motor Installed in MIT Propeller Tunnel

Chapter 5: Results

Summary

The testing described in the previous chapter measured the thrust and torque for each of the designed propellers over a wide range of operating conditions. Measurements to characterize the secondary loads on the propellers were also recorded. Table 8 summarizes the tests for which data was collected over the course of experimentation.

Testing Summary				
	Bare Hub	**Prop1**	**Prop2**	**Prop3**
Thrust		X	X	X
Torque	X	X	X	X
Induced Flow		X	X	X
Drag	X			

Table 8: Summary of Tests Conducted

The measurement goal of the tests was to characterize the thrust generated by the propeller and the torque delivered by the propeller over the range of operating conditions. The thrust measurement had two components; the thrust generated by the propeller, and the drag felt on the hub due to the flow induced by the propeller. Figure 21 shows the sign conventions for the following relationship between propeller thrust, measured thrust, and hub drag:

$$T_{propeller} = T_{measured} + T_{hub\ drag}$$

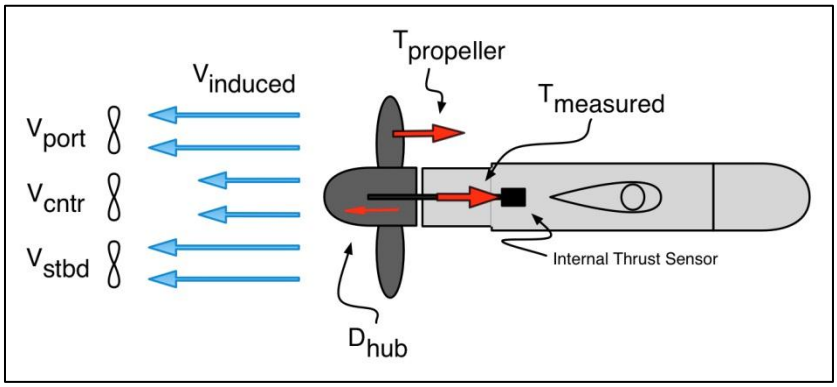

Figure 21: Thrust Data Sign Convention (all quantities positive as shown)

The torque measurement also had two components; the torque delivered by the propeller, and the friction torque required to overcome the motor's inherent resistance to rotation from internal bearings and seals. Figure 22 shows the sign conventions for the following relationship between propeller torque, measured torque, and friction torque:

$$Q_{propeller} = Q_{measured} - Q_{friction}$$

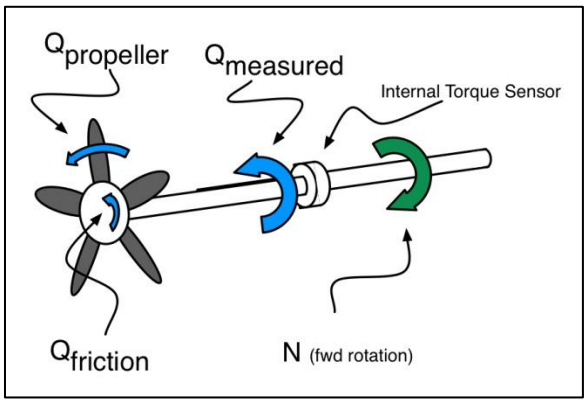

Figure 22: Torque Data Sign Convention (all quantities positive as shown)

Reference Point Error

Raw data was collected in the form of DC voltage output values from the internal thrust and torque sensors in the motor. The average value over approximately 10 seconds of steady state operation was taken at each test speed to represent that speed's measured response value. The response values were plotted in Excel for data visualization.

Data was converted from sensor voltage output to thrust and torque in units of Newtons and Newton-meters using Ketcham's calibration constants as verified in Chapter 4. Since the calibration was a linear relationship for both thrust and torque, the conversion was done by comparing the measured voltage to the corresponding zero thrust or torque voltage output and multiplying the difference by the calibration constant. Wide fluctuations in the zero values however, particularly with the thrust sensor output, were observed throughout the tests. These fluctuations tended to cause large discrepancies between runs in the calculated measurements. The zero value was observed to shift significantly over the course of a series of runs and even over just one test speed such as the shift shown in Figure 23, which displays the sensor voltage output for Propeller 2 being tested at 50 rpm. The red Potential 1 line shows the thrust sensor data, while the blue Potential 2 line displays the torque sensor data.

Before the test speed was commanded for this sample, the average zero speed voltage was 2.94621 V. The 50 rpm command speed was entered at approximately the 2.5 second mark and returned to the 0 rpm command speed at approximately 18 seconds. Following the run, the average zero speed voltage was 3.03266 V. The same effect can be seen in the torque data.

This effect was also observed prior to the experiments when testing the thrust sensor by hand with the motor powered off. Applying a 5-10 pound load to the shaft by hand was enough to cause an observable shift in the sensor zero value. This indicates that the zero shift is not due to residual flow passing the propeller after it is stopped and rather that the zero shift is either due to the stiction in the seals or an electrical issue.

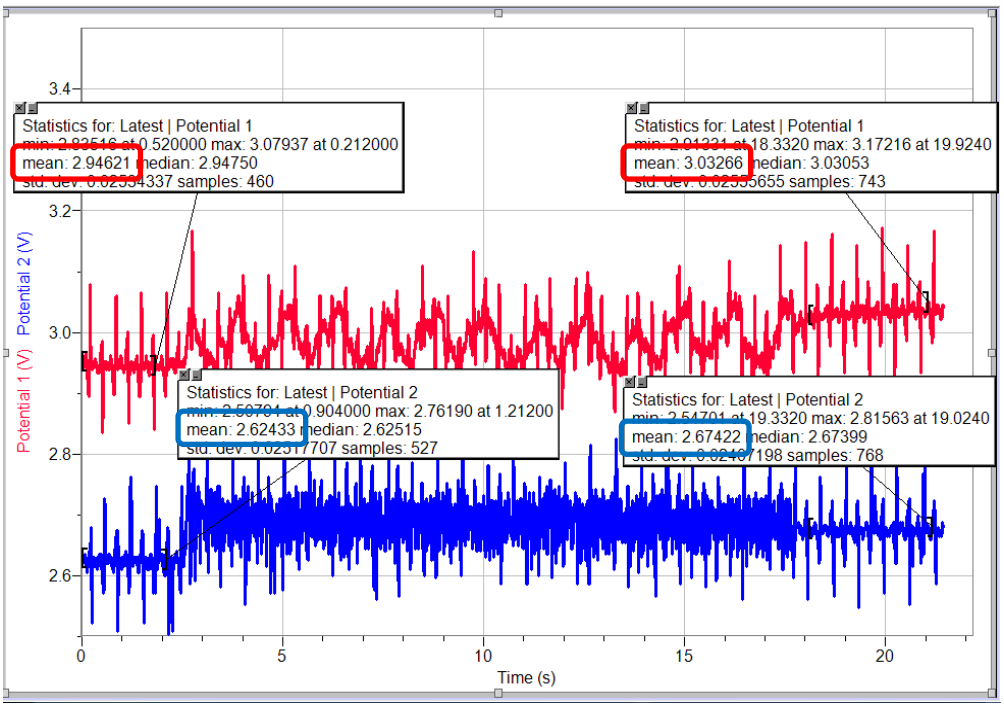

Figure 23: Zero Speed Voltage Shift Example

It is suspected that this zero shift issue is due in large part to an aspect of the mechanical design of the motor. As designed, the rotor is not firmly fixed to the motor mount. The shaft to which the propeller is mounted is driven by a rotor which is supported by tapered roller bearings. These bearings allow a small amount of play in the axial position of the rotor within the motor housing. As loads are applied to the shaft in differing directions, the position of the rotor/shaft/sensor structure shifts slightly. These motions tended to shift the zero reading, particularly on the thrust sensor. As can also be seen there are periodic small spikes in the thrust and torque signals even at zero speed. These small spikes appear to be an effect of the manner in which the motor controller maintains a zero speed command. Since the controller was operating in a velocity control mode relying on Hall Effect sensors for feedback rather than in a position control mode, it was unable to maintain a "locked" or "frozen" motor condition. Instead, a recurring series of very small motions generated enough velocity feedback information for the motor to maintain what appeared to be a stopped position, but really was oscillating with very small motions. Together, these affects tended to shift the zero readings from run to run, particularly on the thrust sensor.

The shifted zero values resulted in data that appeared to be very imprecise. However, there was a remarkable consistency in the output voltages of the sensors any time the motor was maintaining a commanded velocity 50 rpm or greater. Therefore, it was concluded that the measured voltages under load were highly accurate but care was required in the data processing to accurately account for the zero voltages.

Data Calculations

To account for the shifting zero values and take advantage of the consistent low speed data, it was assumed that when the propeller operated at the slowest test speed of 50 rpm, both the thrust

and torque loads were so low as to be negligible. As such, the more consistent 50 rpm test values were used as the new reference zero points. This reduced the standard deviation of the zero value data sets by approximately 60%. Using this formulation, the following thrust and torque results were measured for each propeller.

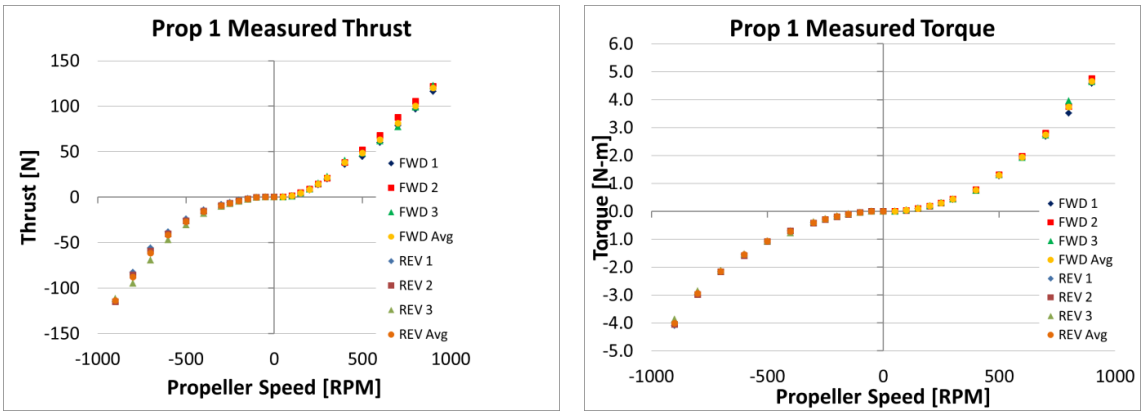

Figure 24: Propeller 1 Thrust and Torque Measurement

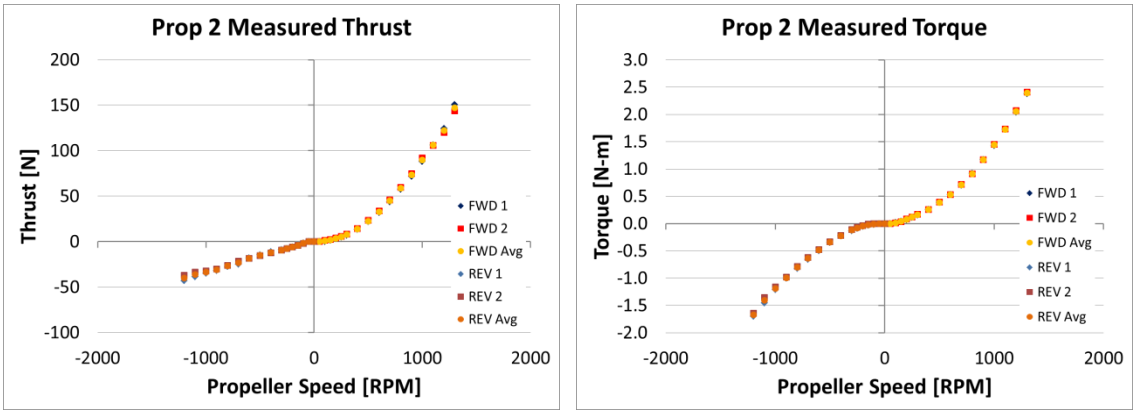

Figure 25: Propeller 2 Thrust and Torque Measurement

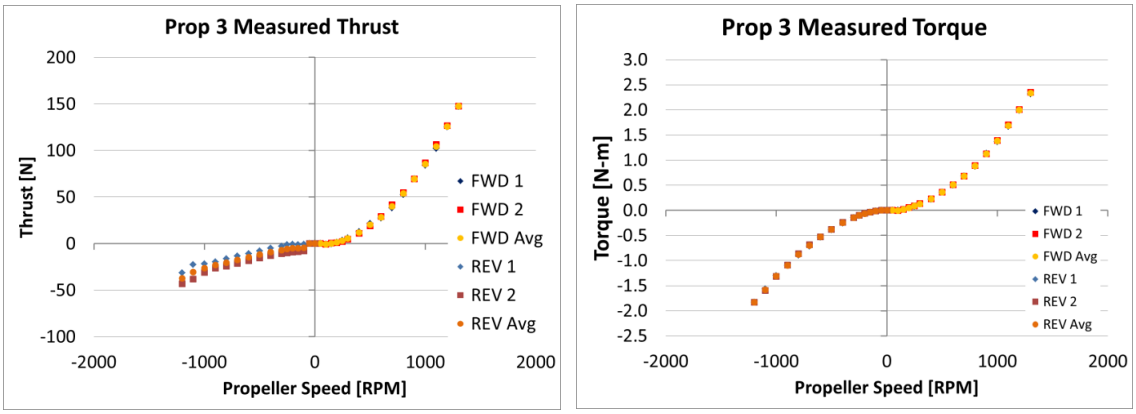

Figure 26: Propeller 3 Thrust and Torque Measurement

The following calculations were performed for each propeller at each test speed to find the propeller thrust and torque from the measured data.

Calculate the measured thrust from the sensor voltage output.

.

$$T_{measured} = (V_{measured} - V_{zero}) \cdot a_T$$

Where a_T = 120.62 N/V, , the calibration constant from Table 6.

Calculate the induced flow speed. Since the induced flow was not measured for every test point, a linear trend line was fit to the flow speed data and used to interpolate the induced flow (Figure 27-Figure 29).

$$v_{induced} = m_{best\ fit} \cdot N + b_{best\ fit}$$

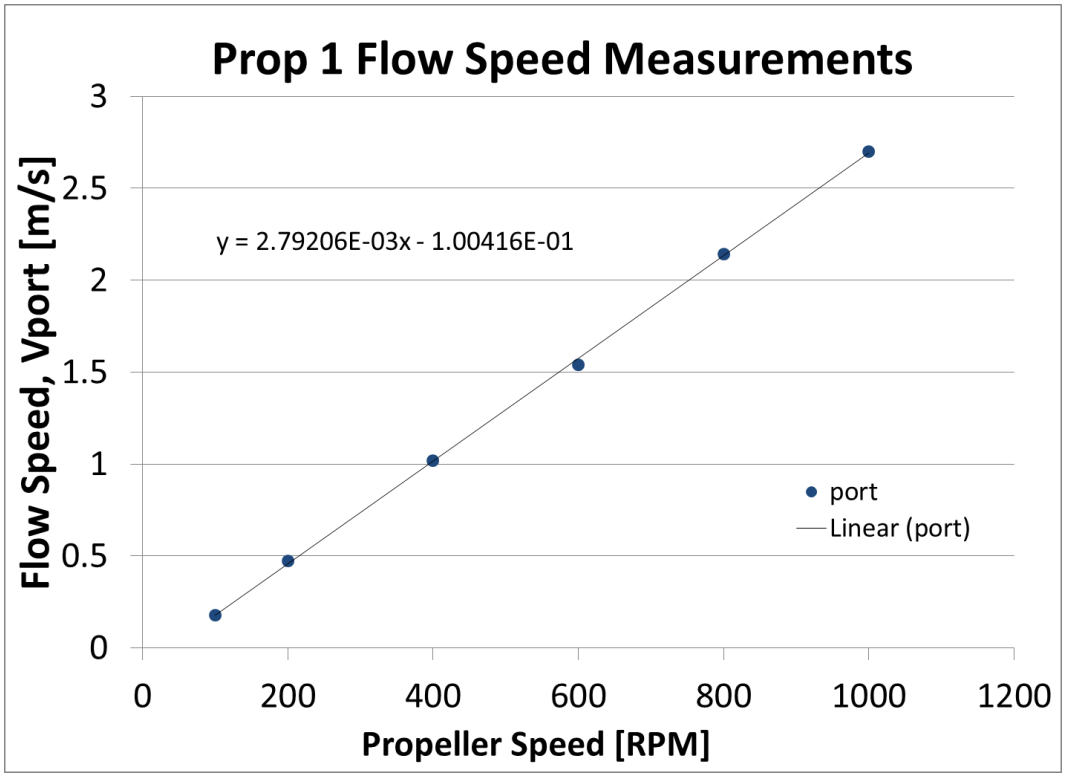

Figure 27: Prop 1 Flow Speed Measurement

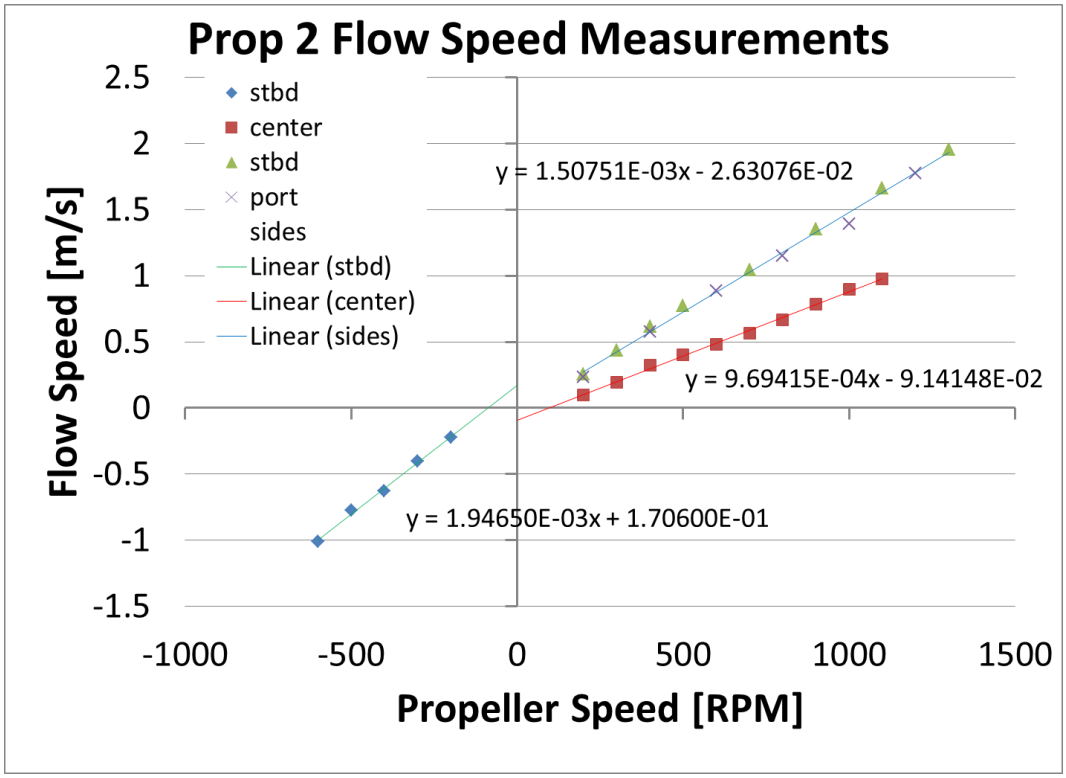

Figure 28: Prop 2 Flow Speed Measurement

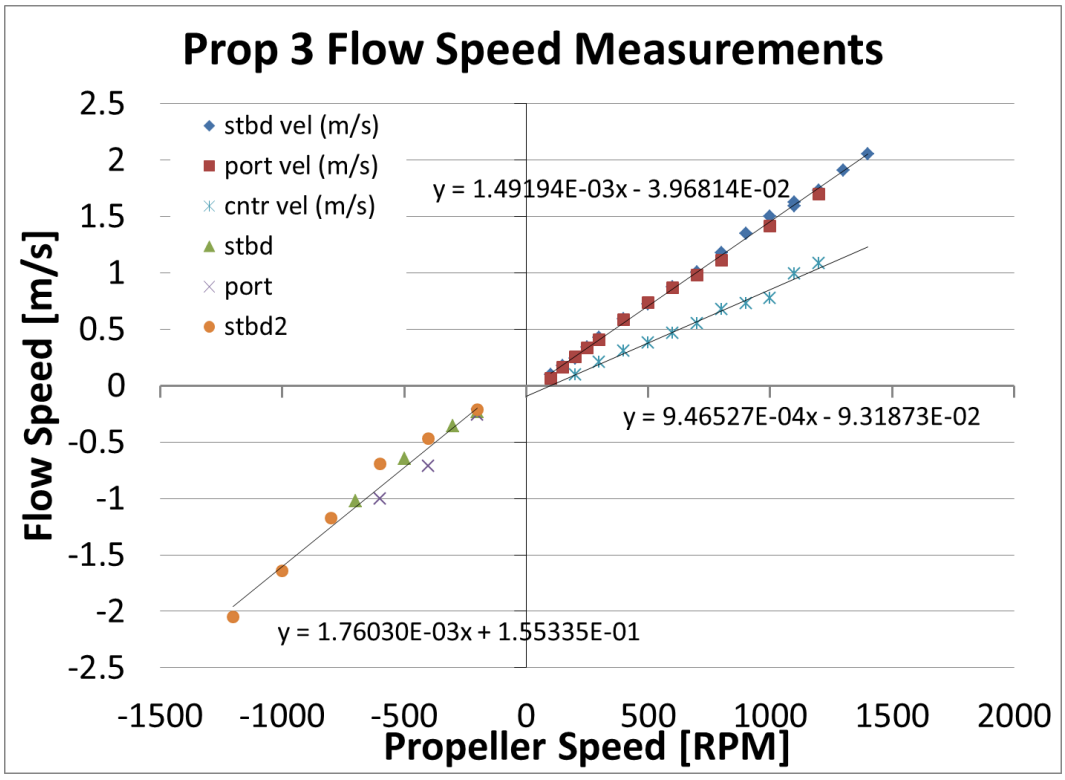

Figure 29: Prop 3 Flow Speed Measurements

Calculate the hub drag. Quadratic trendlines (shown in Figure 30) were fit to the propeller tunnel drag vs flow speed data and used to interpolate the hub drag.

$$D_{hub} = c_2 \cdot v_{induced}{}^2 + c_1 \cdot v_{induced} + c_0$$

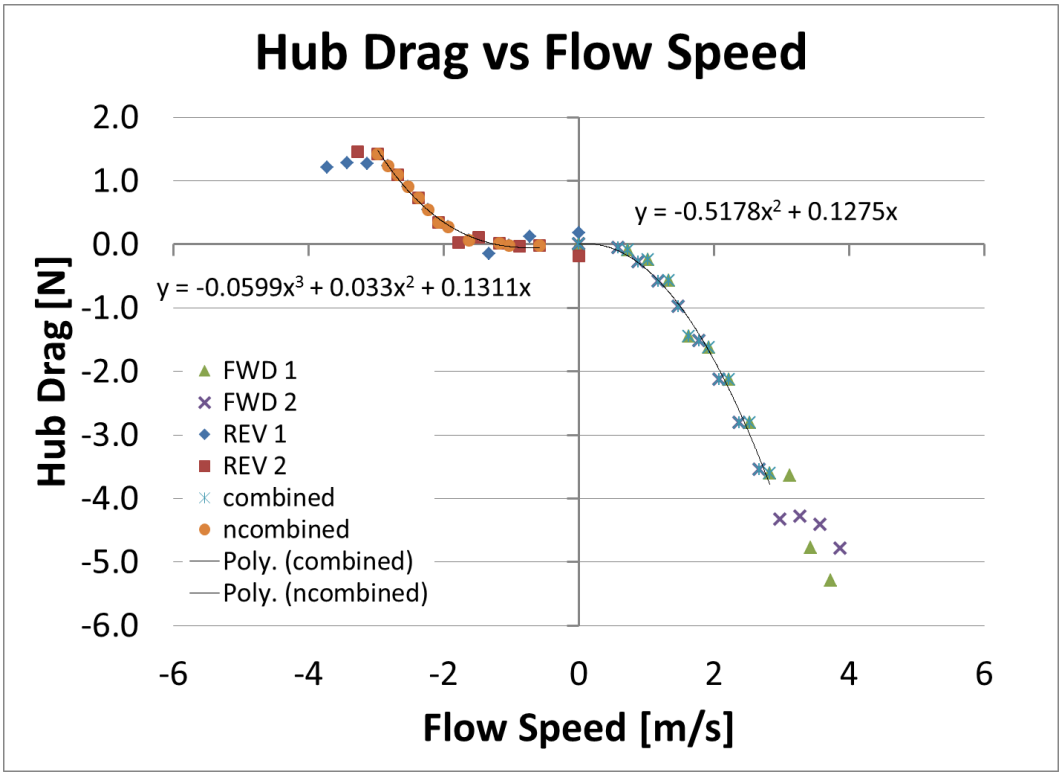

Figure 30: Hub Drag Measurements

Calculate the propeller thrust.

$$T_{propeller} = T_{measured} + D_{hub}$$

Calculate the measured torque from measured voltages.

$$Q_{measured} = (V_{measured} - V_{zero}) \cdot a_Q$$

Where a_Q = 3.5161 N·m/V, the calibration constant from Table 6.

Calculate the measured friction torque (Figure 31).

$$Q_{friction} = \left(V_{measured,bare\ hub} - V_{zero}\right) \cdot a_Q$$

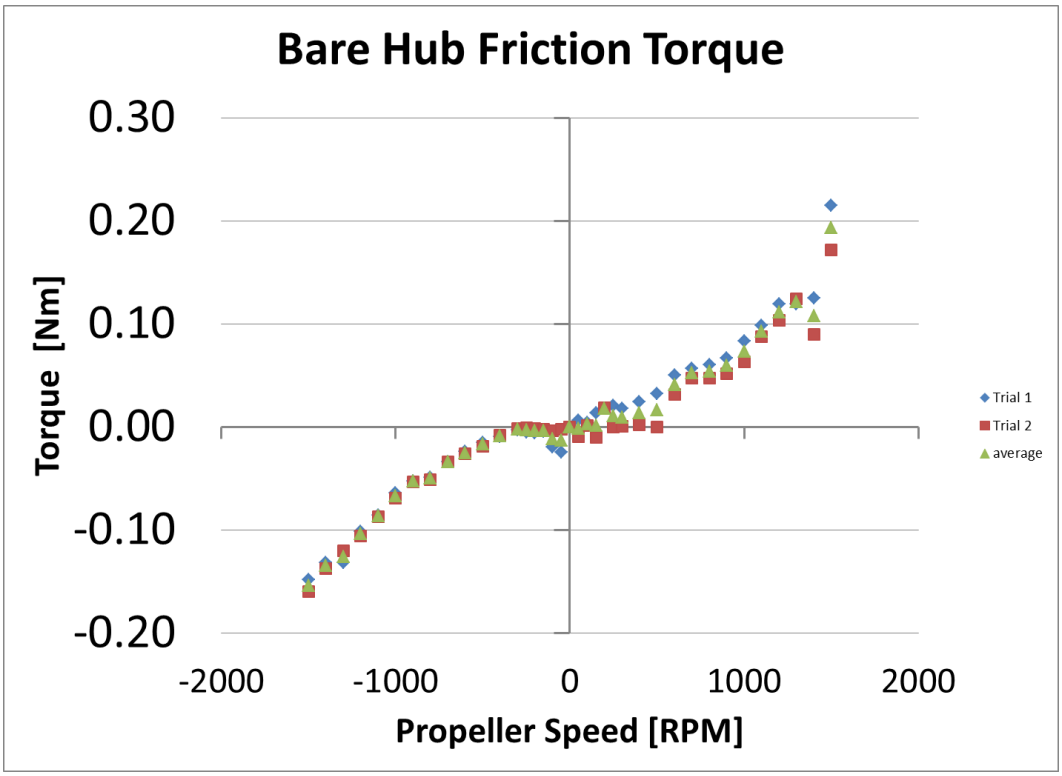

Figure 31: Bare Hub Friction Torque

Calculate the propeller torque

$$Q_{propeller} = Q_{measured} - Q_{friction}$$

Calculate the experimental K_T

$$K_T = \frac{T_{propeller}}{\rho {n_P}^2 D^4}$$

Calculate the experimental K_Q

$$K_Q = \frac{Q_{propeller}}{\rho {n_P}^2 D^5}$$

Calculate the experimental Quality Factor.

$$QF = \frac{1}{\pi\sqrt{2\pi}} \frac{{K_{T\,experimental}}^{1.5}}{K_{Q\,experimental}}$$

Figure 32 and Figure 33 show the results of these calculation as well as original design thrust and torque points for each propeller.

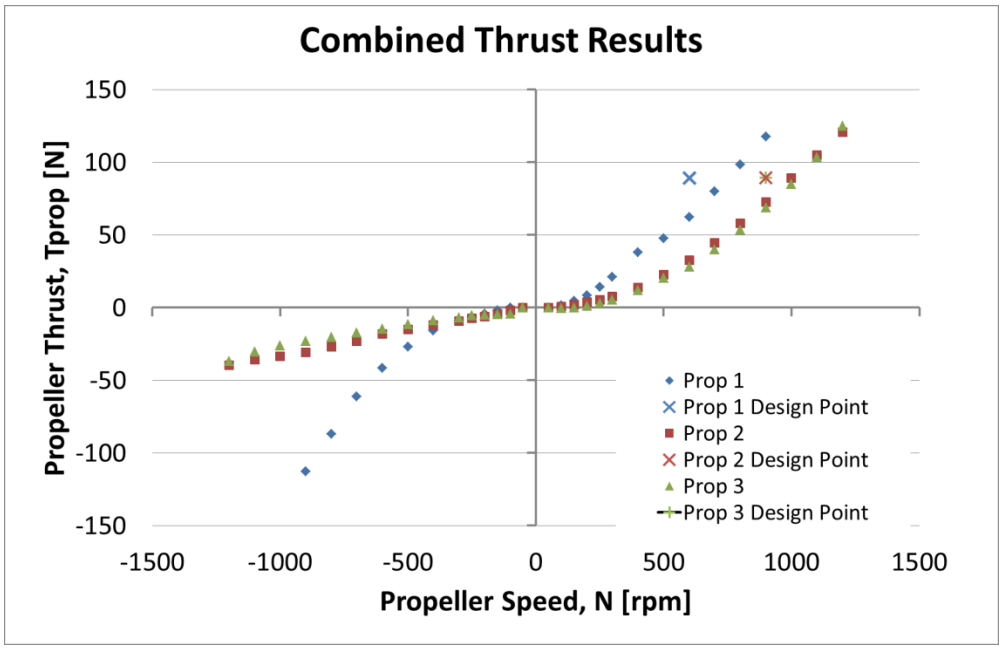

Figure 32: Combined Propeller Thrust Results

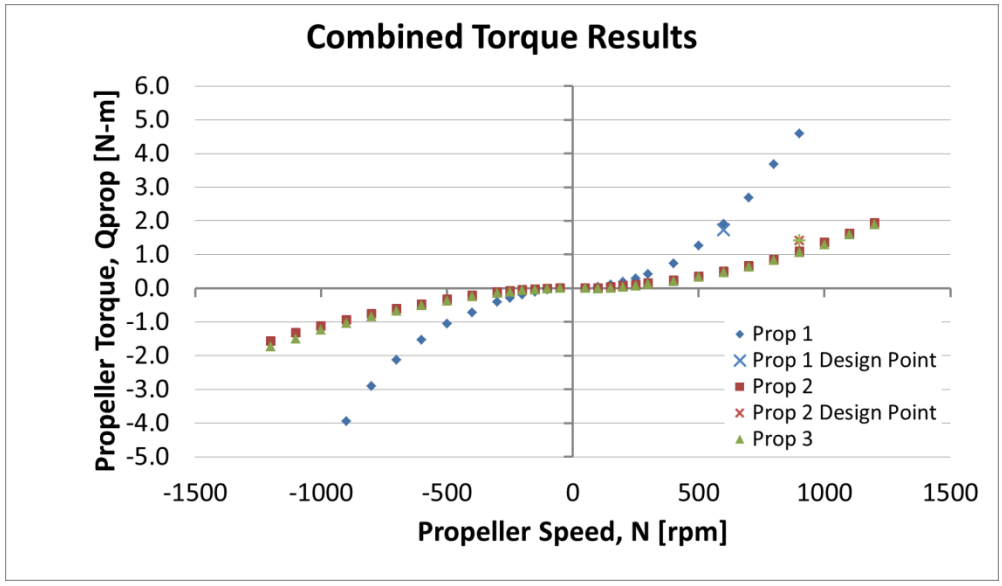

Figure 33: Combined Propeller Torque Results

The results at the design point are summarized in Table 9. All three propeller designs have measured torque results very close to the values designed for in OpenProp. All three also come reasonably close to the designed thrust values. The propeller 1 thrust is within 70%, while propellers 2 and 3 generated 82% and 77% of the design thrust of 88.96 N.

At Design Operating Condition				
		As Designed	As Measured	% Error
Prop 1	Prop Speed [rpm]	600	600	
	Thrust [N]	88.96	62.35	30%
	Torque [N-m]	1.72	1.91	-11%
	KT	0.103	0.0723	30%
	KQ	0.0065	0.0072	-12%
	QF	0.645	0.33999	47%
Prop 2	Prop Speed [rpm]	900	900	
	Thrust [N]	88.96	72.6	18%
	Torque [N-m]	1.42	1.10	23%
	KT	0.153	0.12530	18%
	KQ	0.0108	0.008409	23%
	QF	0.700	0.6698	4%
Prop 3	Prop Speed [rpm]	900	900	
	Thrust [N]	88.96	68.88	23%
	Torque [N-m]	1.41	1.06	25%
	KT	0.1448	0.11229	22%
	KQ	0.010049	0.007546	25%
	QF	0.69632	0.63323	9%

Table 9: Design Result Comparison

In general, the ducted propellers performed closer to the design condition than the open propeller. If this is a side effect of OpenProp wanting a uniform inflow for the calculations to work well, then it makes sense since the open propeller hardly pulled anything in uniformly. Lots of the water came in from the edge with a radial component to its velocity rather than directly into the propeller face. It would make sense then that the ducted propellers performed better in this regard because even though they don't have a speed of advance assisting with flow, their induced flow is restricted to the axial and tangential components and guided by the duct itself into a somewhat uniform inflow to the propeller..

Chapter 6: Conclusions and Recommendations

Conclusions

This thesis leveraged the work of two ongoing Sea Grant projects to design and test a series of propellers to operate in the bollard pull condition. Recent updates to OpenProp strengthening the theoretical model and making the optimization algorithm more robust allowed propeller design in operating conditions that previously would have crashed the code. Additionally, the Ketcham test fixture provided a convenient platform for testing the propellers in an attempt to validate the OpenProp design characteristics.

A parametric design of three different propellers was completed in OpenProp. These designs were imported to PBD-14.36 and MTFLOW. The coupling process for these two programs was developed and codes were adapted to cause the coupled programs to run a complete design loop.

Three propellers were built and tested on the Ketcham test fixture in the MIT Towing Tank and Propeller Tunnel. The results showed general agreement with the predicted design performance, but underperforming the design by an average of roughly 20%. It is well known that lifting line geometry predictions yield propellers that underperform. Even though the propellers were built from the Morgan (1968) lifting surface geometry corrections, it is not surprising that these one-size-fits-all geometry corrections underperformed in the relatively severe bollard pull design conditions .In spite of this underperformance, the method described and demonstrated herein is a useful tool for application to prototyping and design of tunnel bow thrusters and AUV thrusters.

Recommendations for Future Work

The updates to OpenProp are mostly successful in that they allow the optimizer to converge under conditions that it was unable to handle before. Some work remains though to determine the reason for the discrepancies discovered.

Several areas present themselves as good opportunities for further work improving OpenProp's capability:

- Generate and export the blade geometry as a B-spline polygon, ready for direct importing to PBD 14.36.
- Improve robustness of off-design analysis module for use evaluating propellers operating at a very low speed of advance or bollard pull. Currently the off-design analysis module can only converge at the design point. Any perturbation off the design point causes the analyze.m code to crash.
- Symmetric blade design for efficient forward and reverse operation. By their nature, bow thrusters and AUV maneuvering thrusters need to operate bi-directionally. This thesis measured the results of reverse operation, but the propellers were only designed for forward operation, and had no specific predictions for performance in the reverse direction.
- Adaptation for a hubless propeller design where the blades are rooted to a rim drive type motor system.

PBD 14 and MTFLOW are very capable programs. The ability to smoothly relate OpenProp geometry for coupled propeller design and analysis would be very beneficial for future projects. Additional work remains in determining their ability to handle the bollard pull (zero inflow) condition.

Sea Grant is a great resource for and repository of existing programs which provide a good launch point to facilitate future research and projects. This thesis would not have been possible without the resources and support of the Sea Grant staff and students.

References

[1] Celik, F., Guner, M., "An Improved Lifting Line Model for the Design of Marine Propellers," Marine Technology, Vol. 43, pp. 100-113, April 2006

[2] Chrisospathis, A., Kerwin, J.E., Taylor, T. E., "PBD-14.36: A Coupled Lifting-Surface Design/Analysis Program for Marine Propulsors," MIT Department of Ocean Engineering, June 2001.

[3] Chung, H., "An Enhanced Propeller Design Program Based on Propeller Vortex Lattice Lifting Line Theory," Masters Thesis, Massachusetts Institute of Technology, June 2007.

[4] Coney, W. B., "A Method for the Design of a Class of Optimum Marine Propulsors," Massachusetts Institute of Technology PhD dissertation, Cambridge, MA, 1989.

[5] D'Epagnier, K. P., "A Computational Tool for the Rapid Design and Prototyping of Propellers for Underwater Vehicles," Masters Thesis, Massachusetts Institute of Technology and the Woods Hole Oceanographic Institution, September 2007.

[6] Drela, M., "A User's Guide to MTFLOW 1.2: Multi-passage ThroughFLOW Design/Analysis Program,"MIT Fluid Dynamics Research Laboratory, November 1997.

[7] El Lababidy, S. A., Bose, N., Liu, P., Walker, D., Di Felice, F., "Experimental Analysis of the Near Wake from a Ducted Thruster at True and Near Bollard Pull Conditions Using Stereo Particle Image Velocimetry (SPIV)," Journal of Offshore Mechanics and Arctic Engineering, Vol. 127, pp. 191-196, August 2005.

[8] El Lababidy, S. A., Bose, N., Liu, P., Di Felice, F., "Detailed Analysis of the Wake of a DP Thruster Emphasizing Comparison Between LDV and SPIV Techniques," Journal of Offshore Mechanics and Arctic Engineering, Vol. 128, pp. 81-88, May 2006.

[9] El Lababidy, S. A., Bose, N., Liu, P., Walker, D., "Dynamic Positioning Thruster Near Wake Hydrodynamic Characteristics at Near Bollard Pull," Journal of Ship Research, Vol. 53, No. 1, pp. 48-58, March 2009.

[10] Epps, B., "OpenProp v2.3 Theory Document," available at http://openprop.mit.edu, 2010

[11] Epps B., J. Chalfant, K. Flood, A.H. Techet, R.W. Kimball, and C. Chryssostomidis, "OpenProp: an open-sourced parametric design and analysis tool for propellers," Grand Challenges in Modeling and Simulation, Istanbul, Turkey, July 14, 2009.

[12] Epps, B. P., "An impulse framework for hydrodynamic force analysis: fish propulsion, water entry of spheres, and marine propellers, Ph.D. thesis, MIT, February 2010.

[13] Epps B., O. V´ıquez, and C. Chryssostomidis, “Dual-operating-point blade optimization for high-speed propellers,” 11th International Conference on Fast Sea Transportation (FAST 2011), Honolulu, Hawaii, USA, September 2011.

[14] Epps B., “OpenProp v2.4 theory document,” MIT Department of Mechanical Engineering Technical Report, December 2010.

[15] Epps B., J. Ketcham, and C. Chryssostomidis, “Propeller blade stress estimates using lifting line theory,” Grand Challenges in Modeling and Simulation, Ottawa, Canada, July 14, 2010.

[16] Epps B., M. Stanway, and R. Kimball, “OpenProp: an open-source design tool for propellers and turbines,” SNAME Propellers and Shafting, Williamsburg, VA, September 16, 2009.

[17] Flood, K. M., “Propeller Performance Analysis Using Lifting Line Theory,” Masters Thesis, Massachusetts Institute of Technology

[18] Kerwin, J. E., “Hydrofoils and Propellers,” Massachusetts Institute of Technology Course 2.23 Notes, 2007.

[19] Ketcham, J. W., “Design, Build and Test of an Axial Flow Hydrokinetic Turbine with Fatigue Analysis,” Masters Thesis, Massachusetts Institute of Technology, June 2010.

[20] Kimball, R.W., Epps, B.P., "OpenProp v2.4 propeller/turbine design code," http://openprop.mit.edu, 2010.

[21] Laskos, D., “Design and Cavitation Performance of Contra-rotating Propellers,” Masters Thesis, Massachusetts Institute of Technology, June 2010.

[22] Mertes, P., Heinke, H., “Aspects of the Design Procedure for Propellers Providing Maximum Bollard Pull,” Proceedings of the International Tug and Salvage Convention, 2008.

[23] Morgan, W. B., Silovic, V., Denn, S. B., “Propeller Lifting-Surface Corrections,” Trans. SNAME, 76, 1968.

[24] Peterson, C. J., “Minimum Pressure Envelope Cavitation Analysis Using Two-Dimensional Panel Method,” Masters Thesis, Massachusetts Institute of Technology, June 2008

[25] Polsenberg, A. M., Kerwin, J. E., Taylor, T. E., “A Manual for the Coupling of PBD-14 and DTNSAX: A Coupled Lifting-Surface Design/Analysis Technique for Marine Propulsors,” MIT Marine Hydrodynamics Laboratory, April 2000.

[26] Stubblefield, J. M., "Numerically-based Ducted Propeller Design Using Vortex Lattice Lifting Line Theory," Masters Thesis, Massachusetts Institute of Technology, June 2008

[27] Ozdemir, Y. H., Bayraktar, S., Yilmaz, T., Guner, M., "Determining Optimum Geometry for a Bow Thruster Propeller,"Royal Institute of Naval Architects, 8th Symposium on High Speed Marine Vehicles, May 2008

[28] Woud, H. K., Stapersma, D., "Design of Propulsion and Electric Power Generation Systems," IMarEST, 2008.

APPENDIX A

OpenProp Input Files

Propeller 1

```
%                    ------------------------------------------------------------
prop1_open_inputs.m
% Created: 5/28/09, Brenden Epps, bepps@mit.edu
% Modified: 12/14/11, Kip Wilkins, wilkinsj@mit.edu

% This script creates an "input." data structure for use in OpenProp.
%
%   To   design   a   propeller   using   these   inputs,   run:      design   =
EppsOptimizer(input)
%
% ------------------------------------------------------------------------
clear, close all, clc,

addpath ./SourceCode ../SourceCode ../../SourceCode
clr,
% set(0,'DefaultFigureWindowStyle','docked')
% set(0,'DefaultFigureWindowStyle','normal')

filename   = 'prop1';   % filename prefix
notes      = '';            % design notes

% ------------------------------------------------------- Design parameters
Z          = [5];           % number of blades
D          = [12] *0.0254;  % (12 in) propeller diameter [m] (Note: 39.37 in/m
)
N          = [600];         %propeller speed [RPM]
THRUST     = [20]   /0.2248;  % choose 20 lbf (mid range of sensor limits) [N]
(0.2248 lb/N)



Dhub        = [3.5]*0.0254;         % (3.5 in) hub diameter [m] determined by
Ketcham apparatus (must be greater than 0.15*D)
rho        = 1000;          % water density [kg/m^3]
Vs         = 1;             % ship velocity [m/s]
n          = N/60;           % propeller speed [rev/s]
Js         = Vs./(n.*D);

Mp         = 20;            % number of vortex panels over the radius
Np          = 20;              % Number of points over the chord for geometry
plots [ ]
ITER       = 3000;           % number of iterations

% ------------------------------------------------------- Duct parameters
% Inputs for no duct:  Duct_flag = 0; TAU = 1; Rduct_oR = 1; CDd = 0;
TAU         = 1.0;          % thrust ratio
Rduct       = 0;            % duct radius [m]
Cduct       = 0;            % duct chord length [m]
CDd         = 0;            % duct viscous drag coefficient
```

```
% ------------------------------------------------------------------------ Flags
Propeller_flag  = 1;      % 0 == turbine, 1 == propeller
  Viscous_flag  = 1;       % 0 == viscous forces off (CD = 0), 1 == viscous
forces on
      Hub_flag  = 1;      % 0 == no hub, 1 == hub
     Duct_flag  = 0;      % 0 == no duct, 1 == duct
    Chord_flag  = 1;
     Plot_flag  = 0;      % 0 == do not display plots, 1 == display plots

EppsOptimizer02_flag = 1;   % standard propeller optimization algorithm
EppsOptimizer23_flag = 0;    % more robust but slower propeller optimization
algorithm

Make_SWrks_flag = 1;


% ------------------------------------------------ Blade 2D section properties
LSGeoCorr  = 'none';
Meanline   = 'NACA a=0.8 (modified)';            % Meanline type  (1 == NACA
a=0.8, 2 == parabolic)
Thickness  = 'NACA 65A010 (Epps modified)';           % Thickness form (1 ==
NACA 65A010, 2 == elliptical, 3 == parabolic)

XR      = [0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000
0.9000    0.9500    0.9800    0.9900    1.0000];  % radius / propeller radius
% XR        = [0.2910     0.3500     0.4000     0.5000     0.6000     0.7000
0.8000     0.9000     0.9500     0.9800     0.9900     1.0000];   % radius /
propeller radius
% XCoD     = [0.1740     0.2280     0.2750     0.3130     0.3380     0.3480
0.3340     0.2810     0.2190     0.1530     0.1150     0.0010];   % chord /
diameter  (12/6/2011 NOTE: finite chord at tip: 0.0010)
% XCoD     = [0.2222     0.1883     0.1683     0.1567     0.1499     0.1471
0.1473    0.1429    0.1246    0.0910    0.0665    0.0000];
% XCoD     = [0.3135     0.2580     0.2244     0.2050     0.1937     0.1887
0.1890    0.1881    0.1696    0.1282    0.0958    0.0000];
% XCoD     = [0.2611     0.2147     0.1868     0.1706     0.1612     0.1571
0.1573    0.1565    0.1412    0.1066    0.0793    0.0000];  % with 1.2*XCLmax
XCoD    = [0.2035    0.1852    0.1748    0.1606    0.1528    0.1501    0.1519
0.1534    0.1396    0.1061    0.0791    0.0000];

% XR         = [0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    0.95
0.9750 0.9875 1.0];     % radius / propeller radius
% XCoD          = [0.2002 0.2274 0.2531 0.2743 0.2878 0.2913 0.2817 0.2410
0.1962 0.1480 0.0988 0.0259];  % chord / diameter (factor of 1.25 was chosen
to make t0oc look good)
XCD         = [1      1      1      1      1      1      1      1      1
1      1      1]*0.008; % section drag coefficient
XVA         = [1      1      1      1      1      1      1      1      1
1      1      1]; % axial      inflow velocity / ship velocity
XVT         = [0      0      0      0      0      0      0      0      0
0      0      0]; % tangential inflow velocity / ship velocity
skew0       = [0      0      0      0      0      0      0      0      0
0      0      0]; % skew [deg]
rake0       = [0      0      0      0      0      0      0      0      0
0      0      0]; % rake / diameter
```

```
XVA = 0.001*XVA;  % for bollard pull cases
% XVA = 0*XVA;  % for bollard pull cases

% Max lift coefficient
% Values used prior to 2/9/2012 and for ducted prop:
%    XCLmax = 0.5 + (0.2-0.5)/(1-XR(1)) * (XR-XR(1));   % XCLmax == 0.5 at
XR(1) and 0.2 at tip
% Values used after 2/9/2012 for non-ducted prop:
    XCLmax = 0.25 + (0.1-0.25)/(1-XR(1)) * (XR-XR(1));   % XCLmax == 0.25 at
XR(1) and 0.1 at tip
XCLmax = 1.2*XCLmax;
% XCLmax = 0.25
% ------------------------------------------------ Compute derived quantities
R       = D/2;                          % propeller radius [m]
Rhub    = Dhub/2;                       % hub radius [m]
Rhub_oR = Rhub/R;
L       = pi/Js;                          % tip-speed ratio


% % Previous Thickness profile (see notes below)
% TTRF  = 0.5;  % Tip Thickness Reduction Factor == modified thickness at tip
/ baseline thickness at tip
% XRmax = 0.8;  % maximum XR for which thickness reduction is less than 1%
% HTTR  = 3.5;                           % Hub-Tip Thickness Ratio = t0(hub) /
t0(tip)
% t0tip = 0.00254;                % [m] == 0.254 mm = 0.1 inch, max thickness
at tip section
% t0oc0 = t0tip*(HTTR - (HTTR-1).*(XR-Dhub/D)/(1-Dhub/D))./(XCoD*D) .* (1-(1-
TTRF)*exp(-4.6*(1-XR)/(1-XRmax)));

% % taken from other parametric study example (openprop v2.4)
% t0oc0      = [0.2056 0.1551 0.1181 0.0902 0.0694 ...
%                0.0541 0.0419 0.0332 0.0324 0.0000];          % max section
thickness / chord

% Blade thickness profile
t0hub = 0.250*0.0254; % [m] == 0.250 inch (for model), max thickness at hub
section
t0tip = 0.150*0.0254; % [m] == 0.150 inch (for model), max thickness at tip
section
t0tpm = 0.080*0.0254; % [m] == 0.080 inch (for model), modified tip thickness
XRmax = 0.80;               % maximum XR for which thickness reduction is less
than 1%
TTRF   = t0tpm/t0tip;          % Tip Thickness Reduction Factor == modified
thickness at tip / baseline thickness at tip
HTTR   = t0hub/t0tip;         % Hub-Tip Thickness Ratio          == t0(hub) /
t0(tip)

t0        =  t0tip*(HTTR  -  (HTTR-1).*(XR-Rhub_oR)/(1-Rhub_oR))    .*  (1-(1-
TTRF)*exp(-4.6*(1-XR)/(1-XRmax)));
t0oD  = t0/D;
t0oc0 = t0oD ./ XCoD;

% %Pre-design thickness and chord inputs
% fig; plot(XR,t0/0.0254)
%
% fig; plot(XR,XCoD)
```

```
%
% fig; plot(XR,t0oc0), axis([0 1 0 0.5])


Xt0oD = t0oD;


% ========================================================================
% ================================================ Pack up input variables
input.part1      = '------ Performance inputs ------';
input.Z          = Z;           % [1 x 1], [ ] number of blades
input.N          = N;           % propeller speed [RPM]
input.D          = D;           % propeller diameter [m]
input.Vs         = Vs;          % [1 x 1], [m/s] ship speed
input.Js         = Js;          % [1 x 1], [ ] advance coefficient, Js =
                                Vs/nD = pi/L
input.L          = L;           % [1 x 1], [ ] tip speed ratio, L = omega*R/V
input.THRUST     = THRUST;      % [1 x 1], [ ] desired thrust
input.rho        = rho;         % [1 x 1], [kg/m^3] water density

input.part2      = '------ Geometry inputs ------';
input.Mp         = Mp;          % [1 x 1], [ ] number of blade sections
input.Np         = Np;          % [1 x 1], [ ] number of points along the
                                  chord
input.R          = R;           % [1 x 1], [m] propeller radius
input.Rhub       = Rhub;        % [1 x 1], [m] hub radius
input.XR         = XR;          % [length(XR) x 1], [ ] input
                                  radius/propeller radius
input.XVA        = XVA;         % [length(XR) x 1], [ ] input axial inflow
                                velocity  at XR
input.XVT        = XVT;         % [length(XR) x 1], [ ] input swirl inflow
                                  velocity  at XR
input.XCD        = XCD;         % [length(XR) x 1], [ ] input drag
                                  coefficient       at XR
input.XCoD       = XCoD;        % [length(XR) x 1], [ ] input chord /
                                  diameter       at XR
input.Xt0oD      = Xt0oD;       % [length(XR) x 1], [ ] input thickness /
                                   diameter   at XR
input.skew0      = skew0;       % [length(XR) x 1], [ ] input skew  [deg]
                                   at XR
input.rake0      = rake0;       % [length(XR) x 1], [ ] input rake X/D
                                   at XR
input.Meanline   = Meanline;    % 2D section meanline  flag
input.Thickness  = Thickness;   % 2D section thickness flag
input.LSGeoCorr  = LSGeoCorr;
input.XCLmax     = XCLmax;

input.part3      = '------ Computational inputs ------';
input.Propeller_flag  = Propeller_flag; % 0 == turbine, 1 == propeller
input.Viscous_flag    = Viscous_flag;   % 0 == viscous forces off (CD = 0), 1
                                          == viscous forces on
input.Hub_flag        = Hub_flag;       % 0 == no hub, 1 == hub
input.Duct_flag       = Duct_flag;      % 0 == no duct, 1 == duct
input.Plot_flag       = Plot_flag;      % 0 == do not display plots, 1 ==
                                        display plots
input.Chord_flag      = Chord_flag;     % 0 == do not optimize chord lengths,
                                           1 == optimize chord lengths
```

```
input.ITER            = ITER;           % [ ] number of iterations
input.EppsOptimizer02_flag = EppsOptimizer02_flag;
input.EppsOptimizer23_flag = EppsOptimizer23_flag;
input.Make_SWrks_flag      = Make_SWrks_flag;


input.part4      = '------ Duct inputs ------';
input.TAU        = TAU;          % [1 x 1], [ ] propeller thrust / total
                                    thrust
input.Rduct      = Rduct;        % [1 x 1], [m] duct radius
input.Cduct      = Cduct;        % [1 x 1], [m] duct chord length
input.CDd        = CDd;          % [1 x 1], [ ] duct drag coefficient


% ---------------------------- Pack up propeller/turbine data structure, pt
pt.filename = filename; % (string) propeller/turbine name
pt.date     = date;     % (string) date created
pt.notes    = notes;    % (string or cell matrix)   notes
pt.i        = input;    % (struct) input parameters
pt.d        = [];       % (struct) design conditions
pt.g        = [];       % (struct) design geometry
pt.s        = [];       % (struct) off-design state analysis

% ------------------------------------------------------------ Save input data
save OPinput pt

clear, clc,

load OPinput pt

pt.i
```

Propeller 2

```
%                    ------------------------------------------------------
prop2_duct_gap_inputs.m
% Created: 5/28/09, Brenden Epps, bepps@mit.edu
% Modified: 12/14/11, Kip Wilkins, wilkinsj@mit.edu
%
% Ducted Propeller design with a 1/16 in gap between propeller blade tip
% and duct wall.
%
% This script creates an "input." data structure for use in OpenProp.
%
% To be called by Wilkins_usage_duct.m
%
% --------------------------------------------------------------------------
clear, close all, clc,

addpath ./SourceCode ../SourceCode ../../SourceCode

% set(0,'DefaultFigureWindowStyle','docked')

filename   = 'prop2';   % filename prefix
notes      = '';            % design notes

% ------------------------------------------------------ Design parameters
% Z         = [3 4 5];             % number of blades
% D          = [9 10 11 12] *0.0254;   % (approx 9 in) propeller diameter [m]
(Note: 39.37 in/m )
% N         = [300 400 500 600 700 800 900];         %propeller speed [RPM]
% THRUST     = [10 20 30]    /0.2248;   % choose 30 lbf (mid range of sensor
limits) [N] (0.2248 lb/N)

Z         = [5];           % number of blades
D          = [9-0.125] *0.0254;  % (approx 9 in) propeller diameter [m] (Note:
39.37 in/m )
N         = [900];         %propeller speed [RPM]
THRUST     = [20]    /0.2248;  % choose 20 lbf (mid range of sensor limits) [N]
(0.2248 lb/N)



Dhub        = [3.5]*0.0254;         % (3.5 in) hub diameter [m] determined by
Ketcham apparatus (must be greater than 0.15*D)
rho       = 1000;          % water density [kg/m^3]
Vs        = 1;             % ship velocity [m/s]
n         = N/60;           % propeller speed [rev/s]
Js        = Vs./(n.*D);

Mp        = 20;            % number of vortex panels over the radius
Np          = 20;              % Number of points over the chord for geometry
plots [ ]
ITER      = 3000;            % number of iterations

% ------------------------------------------------------ Duct parameters
% % Inputs for no duct:
% Duct_flag = 0; TAU = 1; Rduct_oR = 1; CDd = 0;
```

```
% TAU        = 1.0;          % thrust ratio
% Rduct      = 0;            % duct radius [m]
% Cduct      = 0;            % duct chord length [m]
% CDd        = 0;            % duct viscous drag coefficient

% Inputs for duct:
TAU        = 1.0;          % thrust ratio
Rduct      = [9/2] *0.0254;          % duct radius [m]
Cduct      = 8*Rduct;          % duct chord length [m]
Xduct      = 2*Rduct;          % duct position [m]  Xduct = 0 is default
CDd        = 0.00;        % duct viscous drag coefficient


% ------------------------------------------------------------------------ Flags
Propeller_flag  = 1;      % 0 == turbine, 1 == propeller
  Viscous_flag  = 1;        % 0 == viscous forces off (CD = 0), 1 == viscous
forces on
      Hub_flag  = 1;      % 0 == no hub, 1 == hub
     Duct_flag  = 1;      % 0 == no duct, 1 == duct
    Chord_flag  = 1;
     Plot_flag  = 0;      % 0 == do not display plots, 1 == display plots

EppsOptimizer02_flag = 1;   % standard propeller optimization algorithm
EppsOptimizer23_flag = 0;    % more robust but slower propeller optimization
algorithm

Make_SWrks_flag = 1;

% ------------------------------------------------ Blade 2D section properties
Meanline    = 'NACA a=0.8 (modified)';            % Meanline type  (1 == NACA
a=0.8, 2 == parabolic)
Thickness   = 'NACA 65A010 (modified)';           % Thickness form (1 == NACA
65A010, 2 == elliptical, 3 == parabolic)

XR       = [0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000
0.9000    0.9500    0.9800    0.9900    1.0000];  % radius / propeller radius
% XCoD      = [0.1740     0.2280     0.2750     0.3130     0.3380     0.3480
0.3340     0.2810     0.2190     0.1530     0.1150     0.0010];   % chord /
diameter  (12/6/2011 NOTE: finite chord at tip: 0.0010)
% found optimal shape with chord optimization for best combination (Z=5,
N=900):  rerunning to verify design
XCoD     = [0.1980    0.2512    0.2221    0.2038    0.1948    0.1913    0.1920
0.1896    0.1734    0.1445    0.1219    0.0000];



% XR          = [0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    0.95
0.9750 0.9875 1.0];     % radius / propeller radius
% XCoD          = [0.2002 0.2274 0.2531 0.2743 0.2878 0.2913 0.2817 0.2410
0.1962 0.1480 0.0988 0.0259];  % chord / diameter (factor of 1.25 was chosen
to make t0oc look good)
XCD         = [1      1      1      1      1      1      1      1      1
1      1      1]*0.008; % section drag coefficient
XVA         = [1      1      1      1      1      1      1      1      1
1      1      1]; % axial      inflow velocity / ship velocity
XVT         = [0      0      0      0      0      0      0      0      0
0      0      0]; % tangential inflow velocity / ship velocity
```

```
skew0       = [0       0       0       0       0       0       0       0       0
0       0       0]; % skew [deg]
rake0       = [0       0       0       0       0       0       0       0       0
0       0       0]; % rake / diameter

XVA = 0.001*XVA;  % for bollard pull cases
% XVA = 0*XVA;  % for bollard pull cases

% Max lift coefficient
% Values used prior to 2/9/2012 and for ducted prop:
  XCLmax = 0.5 + (0.2-0.5)/(1-XR(1)) * (XR-XR(1));  % XCLmax == 0.5 at XR(1)
and 0.2 at tip
% Values used after 2/9/2012 for non-ducted prop:
%     XCLmax = 0.25 + (0.1-0.25)/(1-XR(1)) * (XR-XR(1));  % XCLmax == 0.25 at
XR(1) and 0.1 at tip
% XCLmax = XCLmax * 4/5;
% XCLmax = 0.25;
% ------------------------------------------- Compute derived quantities
R       = D/2;                          % propeller radius [m]
Rhub    = Dhub/2;                       % hub radius [m]
Rhub_oR = Rhub/R;
L       = pi/Js;                          % tip-speed ratio



% Blade thickness profile
t0hub = 0.250*0.0254; % [m] == 0.400 inch (for model), max thickness at hub
section
t0tip = 0.150*0.0254; % [m] == 0.150 inch (for model), max thickness at tip
section
t0tpm = 0.080*0.0254; % [m] == 0.080 inch (for model), modified tip thickness
XRmax = 0.80;               % maximum XR for which thickness reduction is less
than 1%
TTRF   = t0tpm/t0tip;          % Tip Thickness Reduction Factor == modified
thickness at tip / baseline thickness at tip
HTTR   = t0hub/t0tip;         % Hub-Tip Thickness Ratio          == t0(hub) /
t0(tip)

t0        =  t0tip*(HTTR  -  (HTTR-1).*(XR-Rhub_oR)/(1-Rhub_oR))    .*  (1-(1-
TTRF)*exp(-4.6*(1-XR)/(1-XRmax)));
t0oD0 = t0/D;
t0oc0 = t0oD0 ./ XCoD;

% %Pre-design thickness and chord inputs
% fig; plot(XR,t0/0.0254)
%
% fig; plot(XR,XCoD)
%
% fig; plot(XR,t0oc0), axis([0 1 0 0.5])


Xt0oD = t0oD0;



% ========================================================================
% ============================================== Pack up input variables
```

```
input.part1      = '------ Performance inputs ------';
input.Z          = Z;           % [1 x 1], [ ] number of blades
input.N          = N;           % propeller speed [RPM]
input.D          = D;           % propeller diameter [m]
input.Vs         = Vs;          % [1 x 1], [m/s] ship speed
input.Js           = Js;            % [1 x 1], [ ] advance coefficient, Js =
Vs/nD = pi/L
input.L          = L;           % [1 x 1], [ ] tip speed ratio, L = omega*R/V
input.THRUST     = THRUST;      % [1 x 1], [ ] desired thrust
input.rho        = rho;         % [1 x 1], [kg/m^3] water density

input.part2      = '------ Geometry inputs ------';
input.Mp         = Mp;          % [1 x 1], [ ] number of blade sections
input.Np           = Np;            % [1 x 1], [ ] number of points along the
chord
input.R          = R;           % [1 x 1], [m] propeller radius
input.Rhub       = Rhub;        % [1 x 1], [m] hub radius
input.XR               =  XR;                % [length(XR)  x  1],  [  ]  input
radius/propeller radius
input.XVA          = XVA;          % [length(XR) x 1], [ ] input axial inflow
velocity  at XR
input.XVT          = XVT;          % [length(XR) x 1], [ ] input swirl inflow
velocity  at XR
input.XCD            = XCD;             % [length(XR)  x  1],  [  ]  input  drag
coefficient       at XR
input.XCoD           = XCoD;           % [length(XR)  x  1],  [  ]  input  chord  /
diameter       at XR
input.Xt0oD        = Xt0oD;         % [length(XR) x 1], [ ] input thickness /
diameter   at XR
input.skew0        = skew0;         % [length(XR) x 1], [ ] input skew  [deg]
at XR
input.rake0         = rake0;          % [length(XR)  x  1],  [  ]  input  rake  X/D
at XR
input.Meanline   = Meanline;    % 2D section meanline  flag
input.Thickness  = Thickness;   % 2D section thickness flag
input.XCLmax     = XCLmax;

input.part3      = '------ Computational inputs ------';
input.Propeller_flag  = Propeller_flag; % 0 == turbine, 1 == propeller
input.Viscous_flag    = Viscous_flag;   % 0 == viscous forces off (CD = 0), 1
== viscous forces on
input.Hub_flag        = Hub_flag;       % 0 == no hub, 1 == hub
input.Duct_flag       = Duct_flag;      % 0 == no duct, 1 == duct
input.Plot_flag         = Plot_flag;        % 0 == do not display plots, 1 ==
display plots
input.Chord_flag      = Chord_flag;     % 0 == do not optimize chord lengths,
1 == optimize chord lengths
input.ITER            = ITER;           % [ ] number of iterations
input.EppsOptimizer02_flag = EppsOptimizer02_flag;
input.EppsOptimizer23_flag = EppsOptimizer23_flag;
input.Make_SWrks_flag      = Make_SWrks_flag;


input.part4      = '------ Duct inputs ------';
input.TAU          = TAU;            % [1 x 1], [ ] propeller thrust / total
thrust
input.Rduct      = Rduct;        % [1 x 1], [m] duct radius
```

```
input.Cduct      = Cduct;       % [1 x 1], [m] duct chord length
input.Xduct      = Xduct;
input.CDd        = CDd;         % [1 x 1], [ ] duct drag coefficient


% ---------------------------- Pack up propeller/turbine data structure, pt
pt.filename = filename; % (string) propeller/turbine name
pt.date     = date;     % (string) date created
pt.notes    = notes;    % (string or cell matrix)   notes
pt.i        = input;    % (struct) input parameters
pt.d        = [];       % (struct) design conditions
pt.g        = [];       % (struct) design geometry
pt.s        = [];       % (struct) off-design state analysis

% ------------------------------------------------------------ Save input data
save OPinput pt

clear, clc,

load OPinput pt

pt.i
```

Propeller 3

```
%                    ------------------------------------------------------------
prop3_duct_no_gap_inputs.m
% Created: 5/28/09, Brenden Epps, bepps@mit.edu
% Modified: 12/14/11, Kip Wilkins, wilkinsj@mit.edu
%
% Ducted Propeller designed with zero gap between propeller blade tip
% and duct wall, but the duct diameter for design is reduced by 1/16 in
% from the actual.
%
% This script creates an "input." data structure for use in OpenProp.
%
% To be called by Wilkins_usage_duct.m
%
% ------------------------------------------------------------------------
clear, close all, clc,

addpath ./SourceCode ../SourceCode ../../SourceCode
clr,
% set(0,'DefaultFigureWindowStyle','docked')

filename   = 'prop3';   % filename prefix
notes      = '';            % design notes

% ------------------------------------------------------- Design parameters
% Z         = [3 4 5];             % number of blades
% D          = [9 10 11 12] *0.0254;  % (approx 9 in) propeller diameter [m]
(Note: 39.37 in/m )
% N         = [300 400 500 600 700 800 900];         %propeller speed [RPM]
% THRUST     = [10 20 30]    /0.2248;   % choose 30 lbf (mid range of sensor
limits) [N] (0.2248 lb/N)

Z         = [5];           % number of blades
D          = [9] *0.0254;  % (approx 9 in) propeller diameter [m] (Note: 39.37
in/m )
N         = [900];         %propeller speed [RPM]
THRUST     = [20]    /0.2248;  % choose 20 lbf (mid range of sensor limits) [N]
(0.2248 lb/N)



Dhub        = [3.5]*0.0254;         % (3.5 in) hub diameter [m] determined by
Ketcham apparatus (must be greater than 0.15*D)
rho        = 1000;          % water density [kg/m^3]
Vs         = 1;             % ship velocity [m/s]
n          = N/60;           % propeller speed [rev/s]
Js         = Vs./(n.*D);

Mp         = 10;            % number of vortex panels over the radius
Np          = 20;              % Number of points over the chord for geometry
plots [ ]
ITER       = 3000;           % number of iterations

% ------------------------------------------------------- Duct parameters
% % Inputs for no duct:
```

```
% Duct_flag = 0; TAU = 1; Rduct_oR = 1; CDd = 0;
% TAU        = 1.0;          % thrust ratio
% Rduct      = 0;            % duct radius [m]
% Cduct      = 0;            % duct chord length [m]
% CDd        = 0;            % duct viscous drag coefficient

% Inputs for duct:
TAU        = 1.0;          % thrust ratio
Rduct      = [9/2] *0.0254;          % duct radius [m]
Cduct      = 8*Rduct;          % duct chord length [m]
Xduct      = 2*Rduct;          % duct position [m]  Xduct = 0 is default
CDd        = 0.00;        % duct viscous drag coefficient


% ------------------------------------------------------------------- Flags
Propeller_flag  = 1;      % 0 == turbine, 1 == propeller
  Viscous_flag  = 1;       % 0 == viscous forces off (CD = 0), 1 == viscous
forces on
      Hub_flag  = 1;      % 0 == no hub, 1 == hub
     Duct_flag  = 1;      % 0 == no duct, 1 == duct
    Chord_flag  = 1;
     Plot_flag  = 0;      % 0 == do not display plots, 1 == display plots

EppsOptimizer02_flag = 1;   % standard propeller optimization algorithm
EppsOptimizer23_flag = 0;    % more robust but slower propeller optimization
algorithm

Make_SWrks_flag = 1;


% ------------------------------------------- Blade 2D section properties
Meanline    = 'NACA a=0.8 (modified)';            % Meanline type  (1 == NACA
a=0.8, 2 == parabolic)
Thickness   = 'NACA 65A010 (modified)';           % Thickness form (1 == NACA
65A010, 2 == elliptical, 3 == parabolic)

XR      = [0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000
0.9000    0.9500    0.9800    0.9900    1.0000];  % radius / propeller radius
% XCoD     = [0.1740     0.2280     0.2750     0.3130     0.3380     0.3480
0.3340     0.2810     0.2190     0.1530     0.1150     0.0010];   % chord /
diameter  (12/6/2011 NOTE: finite chord at tip: 0.0010)
% found optimal shape with chord optimization for best combination (Z=5,
N=900):  rerunning to verify design
XCoD    = [0.1980    0.2512    0.2221    0.2038    0.1948    0.1913    0.1920
0.1896    0.1734    0.1445    0.1219    0.0000];



% XR         = [0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    0.95
0.9750 0.9875 1.0];     % radius / propeller radius
% XCoD          = [0.2002 0.2274 0.2531 0.2743 0.2878 0.2913 0.2817 0.2410
0.1962 0.1480 0.0988 0.0259];  % chord / diameter (factor of 1.25 was chosen
to make t0oc look good)
XCD         = [1      1      1      1      1      1      1      1      1
1      1      1]*0.008; % section drag coefficient
XVA         = [1      1      1      1      1      1      1      1      1
1      1      1]; % axial      inflow velocity / ship velocity
```

```
XVT         = [0       0       0       0       0       0       0       0       0
0       0       0]; % tangential inflow velocity / ship velocity
skew0       = [0       0       0       0       0       0       0       0       0
0       0       0]; % skew [deg]
rake0       = [0       0       0       0       0       0       0       0       0
0       0       0]; % rake / diameter

XVA = 0.001*XVA;  % for bollard pull cases
% XVA = 0*XVA;  % for bollard pull cases

% Max lift coefficient
% Values used prior to 2/9/2012 and for ducted prop:
  XCLmax = 0.5 + (0.2-0.5)/(1-XR(1)) * (XR-XR(1));  % XCLmax == 0.5 at XR(1)
and 0.2 at tip
% Values used after 2/9/2012 for non-ducted prop:
%     XCLmax = 0.25 + (0.1-0.25)/(1-XR(1)) * (XR-XR(1));  % XCLmax == 0.25 at
XR(1) and 0.1 at tip
% XCLmax = XCLmax * 4/5;
% XCLmax = 0.25;
% ------------------------------------------- Compute derived quantities
R       = D/2;                          % propeller radius [m]
Rhub    = Dhub/2;                       % hub radius [m]
Rhub_oR = Rhub/R;
L       = pi/Js;                          % tip-speed ratio


% Blade thickness profile
t0hub = 0.250*0.0254; % [m] == 0.400 inch (for model), max thickness at hub
section
t0tip = 0.150*0.0254; % [m] == 0.150 inch (for model), max thickness at tip
section
t0tpm = 0.080*0.0254; % [m] == 0.080 inch (for model), modified tip thickness
XRmax = 0.80;               % maximum XR for which thickness reduction is less
than 1%
TTRF   = t0tpm/t0tip;          % Tip Thickness Reduction Factor == modified
thickness at tip / baseline thickness at tip
HTTR   = t0hub/t0tip;         % Hub-Tip Thickness Ratio          == t0(hub) /
t0(tip)

t0        =  t0tip*(HTTR  -  (HTTR-1).*(XR-Rhub_oR)/(1-Rhub_oR))    .*  (1-(1-
TTRF)*exp(-4.6*(1-XR)/(1-XRmax)));
t0oD0 = t0/D;
t0oc0 = t0oD0 ./ XCoD;

% %Pre-design thickness and chord inputs
% fig; plot(XR,t0/0.0254)
%
% fig; plot(XR,XCoD)
%
% fig; plot(XR,t0oc0), axis([0 1 0 0.5])


Xt0oD = t0oD0;
```

```
% ========================================================================
% ================================================ Pack up input variables
input.part1      = '------ Performance inputs ------';
input.Z          = Z;           % [1 x 1], [ ] number of blades
input.N          = N;           % propeller speed [RPM]
input.D          = D;           % propeller diameter [m]
input.Vs         = Vs;          % [1 x 1], [m/s] ship speed
input.Js           = Js;            % [1 x 1], [ ] advance coefficient, Js =
Vs/nD = pi/L
input.L          = L;           % [1 x 1], [ ] tip speed ratio, L = omega*R/V
input.THRUST     = THRUST;      % [1 x 1], [ ] desired thrust
input.rho        = rho;         % [1 x 1], [kg/m^3] water density

input.part2      = '------ Geometry inputs ------';
input.Mp         = Mp;          % [1 x 1], [ ] number of blade sections
input.Np           = Np;            % [1 x 1], [ ] number of points along the
chord
input.R          = R;           % [1 x 1], [m] propeller radius
input.Rhub       = Rhub;        % [1 x 1], [m] hub radius
input.XR               = XR;                % [length(XR) x 1], [ ] input
radius/propeller radius
input.XVA          = XVA;          % [length(XR) x 1], [ ] input axial inflow
velocity  at XR
input.XVT          = XVT;          % [length(XR) x 1], [ ] input swirl inflow
velocity  at XR
input.XCD            = XCD;             % [length(XR) x 1], [ ] input drag
coefficient       at XR
input.XCoD           = XCoD;           % [length(XR) x 1], [ ] input chord /
diameter       at XR
input.Xt0oD        = Xt0oD;         % [length(XR) x 1], [ ] input thickness /
diameter   at XR
input.skew0        = skew0;         % [length(XR) x 1], [ ] input skew  [deg]
at XR
input.rake0         = rake0;          % [length(XR) x 1], [ ] input rake X/D
at XR
input.Meanline   = Meanline;    % 2D section meanline  flag
input.Thickness  = Thickness;   % 2D section thickness flag
input.XCLmax     = XCLmax;

input.part3      = '------ Computational inputs ------';
input.Propeller_flag  = Propeller_flag; % 0 == turbine, 1 == propeller
input.Viscous_flag    = Viscous_flag;   % 0 == viscous forces off (CD = 0), 1
== viscous forces on
input.Hub_flag        = Hub_flag;       % 0 == no hub, 1 == hub
input.Duct_flag       = Duct_flag;      % 0 == no duct, 1 == duct
input.Plot_flag         = Plot_flag;       % 0 == do not display plots, 1 ==
display plots
input.Chord_flag      = Chord_flag;     % 0 == do not optimize chord lengths,
1 == optimize chord lengths
input.ITER            = ITER;           % [ ] number of iterations
input.EppsOptimizer02_flag = EppsOptimizer02_flag;
input.EppsOptimizer23_flag = EppsOptimizer23_flag;
input.Make_SWrks_flag      = Make_SWrks_flag;

input.part4      = '------ Duct inputs ------';
input.TAU          = TAU;           % [1 x 1], [ ] propeller thrust / total
thrust
```

```
input.Rduct      = Rduct;       % [1 x 1], [m] duct radius
input.Cduct      = Cduct;       % [1 x 1], [m] duct chord length
input.Xduct      = Xduct;
input.CDd        = CDd;         % [1 x 1], [ ] duct drag coefficient


% ---------------------------- Pack up propeller/turbine data structure, pt
pt.filename = filename; % (string) propeller/turbine name
pt.date     = date;     % (string) date created
pt.notes    = notes;    % (string or cell matrix)   notes
pt.i        = input;    % (struct) input parameters
pt.d        = [];       % (struct) design conditions
pt.g        = [];       % (struct) design geometry
pt.s        = [];       % (struct) off-design state analysis

% ------------------------------------------------------------ Save input data
save OPinput pt

clear, clc,

load OPinput pt

pt.i
```

Interpolate for PBD Script

```
%----------------------------------prop1_Interpolate_OpenProp_to_PBD.m
% Created:   03/20/2012, Brenden Epps, bepps@mit.edu
%                     and Kip Wilkins, wilkinsj@mit.edu
addpath ./SourceCode
clr,
load prop1_OpenProp

%  Interpolate OpenProp design data for PBD14 admin input file
d = pt.d;
g = pt.g;
g1 = pt.g1;
g2 = pt.g2;

 XR = [d.Rhub_oR,0.35:0.05:0.8,0.82:0.02:0.94,0.95:0.01:1.0];
XXR = linspace(d.Rhub_oR,1,1000);



% Circulation distribution
 XG = pchip([d.RC,1],[d.G,0],XR);  % inputs for PBD

XXG = spline(XR,XG,XXR);  % how PBD will interpolate XG
XPG =  pchip(XR,XG,XXR);  % how I would  interpolate XG

fig; title('XG')
    plot(XXR,XXG,'k',d.RC,d.G,'.b',XR,XG,'.r')
    plot(XXR,XPG,'m--')



 % thickness/diameter
 XT = pchip(d.RC,d.t0oD,XR);

XXT = spline(XR,XT,XXR);

Dinch = pt.i.D/0.0254;

fig; title('t0 (inch)')
    plot(XXR,XXT*Dinch,'k', d.RC,d.t0oD*Dinch,'.b', XR,XT*Dinch,'.r')

    ylim = get(gca,'ylim');
    set(gca,'ylim',[0,ylim(2)])



%  Interpolate C/D data for D2X input file
CoDtip = XT(end)/0.20;
 XCoD1 = InterpolateChord([d.RC,1],[d.CoD,CoDtip],XR);
 XCoD2 =            pchip([d.RC,1],[d.CoD,CoDtip],XR);

%  XCoD1 = InterpolateChord(d.RC,d.CoD,XR); % yields zero   chord at the tip
%  XCoD2 =            pchip(d.RC,d.CoD,XR); % yields finite chord at the tip
```

```
XXCoD1 = spline(XR,XCoD1,XXR);
XXCoD2 = spline(XR,XCoD2,XXR);

fig; title('XCoD')
    plot(XXR,XXCoD2,'k',XR,XCoD2,'.r')
    plot(XXR,XXCoD1,'g--')
    plot(d.RC,d.CoD,'.b')

    ylim = get(gca,'ylim');
    set(gca,'ylim',[0,ylim(2)])


% thickness / chord
 Xt0oc = XT ./ XCoD2;

XXt0oc = spline(XR,Xt0oc,XXR);

fig; title('t0oc')
    plot(XXR,XXt0oc,'k',d.RC,d.t0oc,'.b',XR,Xt0oc,'.r')

    ylim = get(gca,'ylim');
    set(gca,'ylim',[0,ylim(2)])


% drag coefficient
CDRAG = pchip(d.RC,d.CD,XR);
CLIFT = pchip(d.RC,d.CL,XR);


% induced axial velocity data
 XUA =  pchip(d.RC,d.UASTAR,XR);
XXUA = spline(XR,XUA,XXR);

fig; title('XUA')
    plot(XXR,XXUA,'k',d.RC,d.UASTAR,'.b',XR,XUA,'.r')


% induced tangential velocity data
 XUT = pchip(d.RC,d.UTSTAR,XR);
XXUT = spline(XR,XUT,XXR);

fig; title('XUT')
    plot(XXR,XXUT,'k',d.RC,d.UTSTAR,'.b',XR,XUT,'.r')




% Lift coefficient
fig;  title('CL')
    plot(XXR,pchip(d.RC,d.CL,XXR),'m--', d.RC,d.CL,'.b'),


%  Interpolate camber data for D2X input file
 XF = pchip(g.RG,g.f0oc,XR);
XXF = spline(XR,XF,XXR);

fig; title('XF')
```

```
    plot(XXR,XXF,'k',g.RG,g.f0oc,'.b',XR,XF,'.r')

% Camber and pitch with Morgan1968 lifting surface geometry corrections
XF2 = pchip(g2.RG,g2.f0oc,XR);
XP2 = pchip(g2.RG,g2.PoD ,XR);


%  Interpolate pitch data for D2X input file
 XP = pchip(g.RG,g.PoD,XR);
XXP = spline(XR,XP,XXR);

fig; title('XPoD')
    plot(XXR,XXP,'k',g.RG,g.PoD,'.b',XR,XP,'.r')


  Pd = atand(g.PoD./(pi.*g.RG));
 XPd = atand(   XP./(pi.*  XR));
XXPd = atand(  XXP./(pi.* XXR));

fig; title('pitch (deg)')
    plot(XXR,XXPd,'k',g.RG,Pd,'.b',XR,XPd,'.r')



% Data matrix for "prop.par" parameter input file for "d2x"
format long g
[XR', XCoD2', 0*XT', XF', XPd', 0*XR', 0*XR']


% Data matrix for "prop.adm" admin file for "PBD"
format short
[XG;
 XR;
 XT;
 CDRAG;
  XUA;
2*XUA;
  XUT;
2*XUT]


% Design parameter table for Kip's thesis:
[XR', 0.001*XR'./XR', 0*XR', XUA', XUT', XG', CLIFT', CDRAG']


% Geometry table for Kip's thesis:
[XR', XCoD2', XT', 0*XR', 0*XR', XF', XP', XF2', XP2']



%%
% Figures for Kip's thesis:


% Circulation distribution
fig;
    plot(XXR,XXG,'k')
```

```
    plot(XR,XG,'.r','markersize',24)

    set(gca,'FontName','Times','FontSize',22)
    xlabel('r / R','FontName','Times','FontSize',24)
    ylabel('\Gamma / (2\pi RVs)','FontName','Times','FontSize',24)

    set(gca,'Xtick',[0.3:0.1:1],'xlim',[0.35 1.05],'ylim',[0 0.04])

    saveas(gcf,'fig_prop1_G_vs_XR','epsc')



%  Chord distribution
fig;
    plot(XXR(1)*[1 1],XXCoD2(1)*[-1 1],'k')
    plot(XXR(end)*[1 1],XXCoD2(end)*[-1 1],'k')
    plot(XXR,XXCoD2,'k')
    plot(XR,XCoD2,'.r','markersize',24)

    plot(XXR,-XXCoD2,'k')
    plot(XR,-XCoD2,'.r','markersize',24)

    ylim = get(gca,'ylim');
    set(gca,'ylim',[-0.2,0.2],'xlim',[0.35 1.05])
    set(gca,'Xtick',[0.3:0.1:1])
    axis equal


    set(gca,'FontName','Times','FontSize',22)
    xlabel('r / R','FontName','Times','FontSize',24)
    ylabel('c / R','FontName','Times','FontSize',24)


    saveas(gcf,'fig_prop1_G_vs_CoD','epsc')
```

Design Parameter Tables

Propeller 1 Design Parameters							
$\frac{r}{R}$	$\frac{v_A}{v_s}$	$\frac{v_T}{v_s}$	$\frac{{u_A}^*}{v_s}$	$\frac{{u_T}^*}{v_s}$	$\frac{\Gamma}{2\pi R v_s}$	C_L	C_D
0.2917	0.0010	0.0000	0.7394	-0.2051	0.0261	0.2794	0.0080
0.3500	0.0010	0.0000	0.7828	-0.1962	0.0273	0.2663	0.0080
0.4000	0.0010	0.0000	0.8037	-0.1778	0.0283	0.2550	0.0080
0.4500	0.0010	0.0000	0.8183	-0.1622	0.0291	0.2437	0.0080
0.5000	0.0010	0.0000	0.8296	-0.1490	0.0298	0.2325	0.0080
0.5500	0.0010	0.0000	0.8383	-0.1376	0.0302	0.2213	0.0080
0.6000	0.0010	0.0000	0.8452	-0.1277	0.0306	0.2100	0.0080
0.6500	0.0010	0.0000	0.8506	-0.1190	0.0309	0.1988	0.0080
0.7000	0.0010	0.0000	0.8550	-0.1114	0.0312	0.1875	0.0080
0.7500	0.0010	0.0000	0.8586	-0.1046	0.0313	0.1763	0.0080
0.8000	0.0010	0.0000	0.8616	-0.0986	0.0314	0.1650	0.0080
0.8200	0.0010	0.0000	0.8627	-0.0963	0.0314	0.1605	0.0080
0.8400	0.0010	0.0000	0.8637	-0.0942	0.0314	0.1560	0.0080
0.8600	0.0010	0.0000	0.8646	-0.0922	0.0312	0.1515	0.0080
0.8800	0.0010	0.0000	0.8654	-0.0902	0.0310	0.1470	0.0080
0.9000	0.0010	0.0000	0.8662	-0.0883	0.0304	0.1425	0.0080
0.9200	0.0010	0.0000	0.8669	-0.0865	0.0295	0.1380	0.0080
0.9400	0.0010	0.0000	0.8672	-0.0847	0.0279	0.1335	0.0080
0.9500	0.0010	0.0000	0.8672	-0.0838	0.0268	0.1313	0.0080
0.9600	0.0010	0.0000	0.8672	-0.0829	0.0252	0.1290	0.0080
0.9700	0.0010	0.0000	0.8672	-0.0820	0.0229	0.1268	0.0080
0.9800	0.0010	0.0000	0.8671	-0.0811	0.0189	0.1245	0.0080
0.9900	0.0010	0.0000	0.8670	-0.0801	0.0107	0.1223	0.0080
1.0000	0.0010	0.0000	0.8669	-0.0792	0.0000	0.1200	0.0080

Table 10: Propeller 1 Design Parameters

Propeller 2 Design Parameters							
$\frac{r}{R}$	$\frac{v_A}{v_s}$	$\frac{v_T}{v_s}$	$\frac{{u_A}^*}{v_s}$	$\frac{{u_T}^*}{v_s}$	$\frac{\Gamma}{2\pi R v_s}$	C_L	C_D
0.3944	0.0010	0.0000	1.0790	-0.2869	0.0492	0.4271	0.0080
0.4500	0.0010	0.0000	1.1219	-0.2789	0.0505	0.4063	0.0080
0.5000	0.0010	0.0000	1.1502	-0.2618	0.0520	0.3875	0.0080
0.5500	0.0010	0.0000	1.1676	-0.2426	0.0533	0.3687	0.0080
0.6000	0.0010	0.0000	1.1819	-0.2265	0.0543	0.3500	0.0080
0.6500	0.0010	0.0000	1.1938	-0.2123	0.0551	0.3312	0.0080
0.7000	0.0010	0.0000	1.2036	-0.1996	0.0558	0.3125	0.0080
0.7500	0.0010	0.0000	1.2117	-0.1882	0.0562	0.2938	0.0080
0.8000	0.0010	0.0000	1.2184	-0.1779	0.0565	0.2750	0.0080
0.8200	0.0010	0.0000	1.2208	-0.1741	0.0564	0.2675	0.0080
0.8400	0.0010	0.0000	1.2230	-0.1704	0.0563	0.2600	0.0080
0.8600	0.0010	0.0000	1.2250	-0.1669	0.0559	0.2525	0.0080
0.8800	0.0010	0.0000	1.2268	-0.1634	0.0553	0.2450	0.0080
0.9000	0.0010	0.0000	1.2283	-0.1600	0.0543	0.2375	0.0080
0.9200	0.0010	0.0000	1.2298	-0.1568	0.0527	0.2300	0.0080
0.9400	0.0010	0.0000	1.2314	-0.1538	0.0501	0.2225	0.0080
0.9500	0.0010	0.0000	1.2323	-0.1524	0.0483	0.2188	0.0080
0.9600	0.0010	0.0000	1.2331	-0.1510	0.0462	0.2150	0.0080
0.9700	0.0010	0.0000	1.2339	-0.1496	0.0437	0.2113	0.0080
0.9800	0.0010	0.0000	1.2347	-0.1483	0.0397	0.2075	0.0080
0.9900	0.0010	0.0000	1.2356	-0.1470	0.0291	0.2038	0.0080
1.0000	0.0010	0.0000	1.2364	-0.1457	0.0000	0.2000	0.0080

Table 11: Propeller 2 Design Parameters

Propeller 3 Design Parameters							
$\frac{r}{R}$	$\frac{v_A}{v_s}$	$\frac{v_T}{v_s}$	$\frac{{u_A}^*}{v_s}$	$\frac{{u_T}^*}{v_s}$	$\frac{\Gamma}{2\pi R v_s}$	C_L	C_D
0.3889	0.0010	0.0000	1.0863	-0.2720	0.0444	0.4292	0.0080
0.4500	0.0010	0.0000	1.1242	-0.2533	0.0459	0.4063	0.0080
0.5000	0.0010	0.0000	1.1456	-0.2363	0.0470	0.3875	0.0080
0.5500	0.0010	0.0000	1.1603	-0.2187	0.0480	0.3687	0.0080
0.6000	0.0010	0.0000	1.1723	-0.2038	0.0488	0.3500	0.0080
0.6500	0.0010	0.0000	1.1822	-0.1907	0.0495	0.3313	0.0080
0.7000	0.0010	0.0000	1.1902	-0.1791	0.0501	0.3125	0.0080
0.7500	0.0010	0.0000	1.1969	-0.1686	0.0505	0.2938	0.0080
0.8000	0.0010	0.0000	1.2024	-0.1593	0.0509	0.2750	0.0080
0.8200	0.0010	0.0000	1.2043	-0.1558	0.0510	0.2675	0.0080
0.8400	0.0010	0.0000	1.2060	-0.1524	0.0512	0.2600	0.0080
0.8600	0.0010	0.0000	1.2075	-0.1492	0.0513	0.2525	0.0080
0.8800	0.0010	0.0000	1.2088	-0.1460	0.0514	0.2450	0.0080
0.9000	0.0010	0.0000	1.2101	-0.1429	0.0514	0.2375	0.0080
0.9200	0.0010	0.0000	1.2113	-0.1401	0.0515	0.2300	0.0080
0.9400	0.0010	0.0000	1.2125	-0.1373	0.0516	0.2225	0.0080
0.9500	0.0010	0.0000	1.2131	-0.1360	0.0516	0.2188	0.0080
0.9600	0.0010	0.0000	1.2137	-0.1347	0.0516	0.2150	0.0080
0.9700	0.0010	0.0000	1.2142	-0.1335	0.0517	0.2112	0.0080
0.9800	0.0010	0.0000	1.2148	-0.1322	0.0517	0.2075	0.0080
0.9900	0.0010	0.0000	1.2154	-0.1311	0.0517	0.2037	0.0080
1.0000	0.0010	0.0000	1.2159	-0.1299	0.0517	0.2000	0.0080

Table 12: Propeller 3 Design Parameters

Propeller Geometry Tables

Propeller 1 Geometry					No Lifting Surface Correction		With Lifting Surface Correction	
$\frac{r}{R}$	$\frac{c}{D}$	$\frac{t_0}{D}$	$skew$	$rake$	$\frac{f_0}{c}$	$\frac{P}{D}$	$\frac{f_0}{c}$	$\frac{P}{D}$
0.2917	0.2177	0.0208	0.0000	0.0000	0.0186	0.2688	0.0218	0.2777
0.3500	0.1983	0.0201	0.0000	0.0000	0.0177	0.2807	0.0183	0.2889
0.4000	0.1868	0.0196	0.0000	0.0000	0.0170	0.2851	0.0167	0.2923
0.4500	0.1777	0.0190	0.0000	0.0000	0.0162	0.2881	0.0159	0.2942
0.5000	0.1706	0.0184	0.0000	0.0000	0.0155	0.2905	0.0154	0.2955
0.5500	0.1652	0.0178	0.0000	0.0000	0.0147	0.2924	0.0148	0.2964
0.6000	0.1612	0.0172	0.0000	0.0000	0.0140	0.2938	0.0142	0.2973
0.6500	0.1586	0.0166	0.0000	0.0000	0.0132	0.2950	0.0136	0.2983
0.7000	0.1571	0.0160	0.0000	0.0000	0.0125	0.2958	0.0130	0.2993
0.7500	0.1568	0.0154	0.0000	0.0000	0.0117	0.2965	0.0126	0.2999
0.8000	0.1573	0.0148	0.0000	0.0000	0.0110	0.2970	0.0124	0.3005
0.8200	0.1577	0.0145	0.0000	0.0000	0.0107	0.2971	0.0125	0.3007
0.8400	0.1580	0.0142	0.0000	0.0000	0.0104	0.2972	0.0126	0.3010
0.8600	0.1581	0.0139	0.0000	0.0000	0.0101	0.2973	0.0128	0.3013
0.8800	0.1579	0.0135	0.0000	0.0000	0.0098	0.2974	0.0130	0.3017
0.9000	0.1564	0.0130	0.0000	0.0000	0.0095	0.2974	0.0134	0.3021
0.9200	0.1532	0.0124	0.0000	0.0000	0.0092	0.2974	0.0138	0.3026
0.9400	0.1466	0.0116	0.0000	0.0000	0.0089	0.2973	0.0143	0.3031
0.9500	0.1412	0.0111	0.0000	0.0000	0.0087	0.2972	0.0146	0.3033
0.9600	0.1333	0.0105	0.0000	0.0000	0.0086	0.2971	0.0149	0.3036
0.9700	0.1225	0.0098	0.0000	0.0000	0.0084	0.2969	0.0153	0.3038
0.9800	0.1057	0.0090	0.0000	0.0000	0.0083	0.2967	0.0156	0.3040
0.9900	0.0754	0.0082	0.0000	0.0000	0.0081	0.2965	0.0160	0.3043
1.0000	0.0370	0.0074	0.0000	0.0000	0.0080	0.2963	0.0164	0.3000

Table 13: Propeller 1 Geometry

Propeller 2 Geometry					No Lifting Surface Correction		With Lifting Surface Correction	
$\frac{r}{R}$	$\frac{c}{D}$	$\frac{t_0}{D}$	$skew$	$rake$	$\frac{f_0}{c}$	$\frac{P}{D}$	$\frac{f_0}{c}$	$\frac{P}{D}$
0.3944	0.1785	0.0282	0.0000	0.0000	0.0284	0.3567	0.0281	0.3666
0.4500	0.1684	0.0271	0.0000	0.0000	0.0270	0.3674	0.0265	0.3758
0.5000	0.1629	0.0262	0.0000	0.0000	0.0258	0.3739	0.0256	0.3811
0.5500	0.1586	0.0253	0.0000	0.0000	0.0245	0.3769	0.0247	0.3827
0.6000	0.1556	0.0243	0.0000	0.0000	0.0233	0.3795	0.0237	0.3843
0.6500	0.1537	0.0234	0.0000	0.0000	0.0220	0.3816	0.0226	0.3863
0.7000	0.1528	0.0225	0.0000	0.0000	0.0208	0.3834	0.0217	0.3883
0.7500	0.1528	0.0215	0.0000	0.0000	0.0195	0.3847	0.0208	0.3897
0.8000	0.1534	0.0205	0.0000	0.0000	0.0183	0.3856	0.0205	0.3909
0.8200	0.1537	0.0201	0.0000	0.0000	0.0178	0.3859	0.0205	0.3913
0.8400	0.1538	0.0196	0.0000	0.0000	0.0173	0.3861	0.0206	0.3918
0.8600	0.1537	0.0191	0.0000	0.0000	0.0168	0.3863	0.0209	0.3924
0.8800	0.1530	0.0186	0.0000	0.0000	0.0163	0.3864	0.0213	0.3930
0.9000	0.1515	0.0179	0.0000	0.0000	0.0158	0.3864	0.0218	0.3936
0.9200	0.1484	0.0170	0.0000	0.0000	0.0153	0.3865	0.0225	0.3945
0.9400	0.1428	0.0159	0.0000	0.0000	0.0148	0.3865	0.0233	0.3955
0.9500	0.1385	0.0152	0.0000	0.0000	0.0145	0.3865	0.0238	0.3960
0.9600	0.1332	0.0144	0.0000	0.0000	0.0143	0.3865	0.0242	0.3966
0.9700	0.1261	0.0133	0.0000	0.0000	0.0141	0.3866	0.0248	0.3973
0.9800	0.1154	0.0122	0.0000	0.0000	0.0138	0.3866	0.0253	0.3980
0.9900	0.0941	0.0108	0.0000	0.0000	0.0136	0.3866	0.0259	0.3987
1.0000	0.0472	0.0094	0.0000	0.0000	0.0133	0.3866	0.0264	0.3900

Table 14: Propeller 2 Geometry

Propeller 3 Geometry					No Lifting Surface Correction		With Lifting Surface Correction	
$\frac{r}{R}$	$\frac{c}{D}$	$\frac{t_0}{D}$	$skew$	$rake$	$\frac{f_0}{c}$	$\frac{P}{D}$	$\frac{f_0}{c}$	$\frac{P}{D}$
0.3889	0.1593	0.0278	0.0000	0.0000	0.0285	0.3518	0.0283	0.3618
0.4500	0.1501	0.0267	0.0000	0.0000	0.0270	0.3610	0.0266	0.3695
0.5000	0.1445	0.0258	0.0000	0.0000	0.0258	0.3654	0.0255	0.3724
0.5500	0.1405	0.0248	0.0000	0.0000	0.0245	0.3680	0.0246	0.3736
0.6000	0.1377	0.0239	0.0000	0.0000	0.0233	0.3700	0.0237	0.3749
0.6500	0.1358	0.0230	0.0000	0.0000	0.0220	0.3718	0.0226	0.3764
0.7000	0.1351	0.0221	0.0000	0.0000	0.0208	0.3731	0.0216	0.3779
0.7500	0.1351	0.0212	0.0000	0.0000	0.0195	0.3741	0.0209	0.3791
0.8000	0.1361	0.0202	0.0000	0.0000	0.0183	0.3748	0.0205	0.3800
0.8200	0.1368	0.0198	0.0000	0.0000	0.0178	0.3750	0.0205	0.3804
0.8400	0.1377	0.0193	0.0000	0.0000	0.0173	0.3751	0.0206	0.3808
0.8600	0.1387	0.0188	0.0000	0.0000	0.0168	0.3752	0.0209	0.3812
0.8800	0.1399	0.0183	0.0000	0.0000	0.0163	0.3752	0.0213	0.3817
0.9000	0.1413	0.0176	0.0000	0.0000	0.0158	0.3752	0.0218	0.3823
0.9200	0.1429	0.0167	0.0000	0.0000	0.0153	0.3751	0.0225	0.3831
0.9400	0.1447	0.0155	0.0000	0.0000	0.0148	0.3750	0.0233	0.3840
0.9500	0.1457	0.0148	0.0000	0.0000	0.0145	0.3750	0.0238	0.3845
0.9600	0.1468	0.0140	0.0000	0.0000	0.0143	0.3750	0.0242	0.3851
0.9700	0.1479	0.0132	0.0000	0.0000	0.0141	0.3749	0.0248	0.3857
0.9800	0.1490	0.0123	0.0000	0.0000	0.0138	0.3750	0.0253	0.3864
0.9900	0.1502	0.0114	0.0000	0.0000	0.0136	0.3750	0.0259	0.3871
1.0000	0.1514	0.0105	0.0000	0.0000	0.0133	0.3750	0.0264	0.3800

Table 15: Propeller 3 Geometry

APPENDIX B
Advanced Design with MTFLOW and PBD14

Introduction

One of the limitations of OpenProp design is that since it is based on a lifting line method for design it is not able to account for 3D lifting surface effects. As such, OpenProp is a good tool for a preliminary propeller blade design, but needs additional refinement for further development. Fortunately, previous work at MIT has resulted in two programs that when paired together are useful for solving this problem.

PBD-14 is a lifting surface design and analysis program for marine propulsors based on vortex lattice theory. It represents the blade shape as a B-spline polygon and through an iterative process aligns the blade surface such that the load distribution over the surface of the blade satisfies the kinematic boundary condition $V \cdot n=0$ in the presence of a given flow. If the effective inflow is known, PBD is capable of completing the design in a stand alone mode. PBD has no means of calculating the induced vorticity upstream of the propeller blade, however, so if the effective wake is not known beforehand, PBD requires coupling with a flow solver.

MTFLOW is a Multi-Passage ThroughFLOW design and analysis program that has been adapted to couple with PBD-14 . MTFLOW is a suite of codes that solve the Euler equations for an axisymmetric fluid flow domain. It is capable of accounting for the effects of swirl, heat addition, loss generation, and area blockage. Given a propeller circumferential mean force input, it can calculate the effect of the propeller to the total wake and deliver the new wake profile back to PBD-14.

Documentation for using both PBD-14.36 and MTFLOW is available in the form of user manuals ([2] and [6]) along with a coupling manual for using PBD-14 paired with DTNSAX [25] a different flow solver that wasn't evaluated for the project. There are several significant differences between the methods PBD uses to couple with DTNSAX and MTFLOW. The manuals have not been updated to match the most recent version of the codes yet, however, and getting the coupling between PBD-14 and MTFLOW to work properly required a lot of trial and error. This chapter attempts to summarize the use of PBD-14 with MTFLOW and lay out a simple structure for coupling the programs for propeller design.

The coupling method described herein can be used for non-zero advance speeds ($vA=1$). Unfortunately, due to time constraints, PBD/MTFLOW was not extended to the bollard pull condition, and this remains the subject of future work. The actual propellers used for this thesis were created from OpenProp using the Morgan 1968 lifting surface geometry corrections.

Overview

Both PBD-14 and MTFLOW consist of multiple individual codes that must be run successively in order to complete a full iteration of the design cycle. Figure 34 shows a flowchart of the major steps in the cycle along with a listing of the executables associated with each function. This discussion describes application of the propeller design function. Nearly all input files

described have detailed development in their respective users manuals. As such, the following descriptions are summaries intended to clarify the coupling procedure. Many of the executables generate output files that are not used here. Only the output files used for the propeller design are discussed.

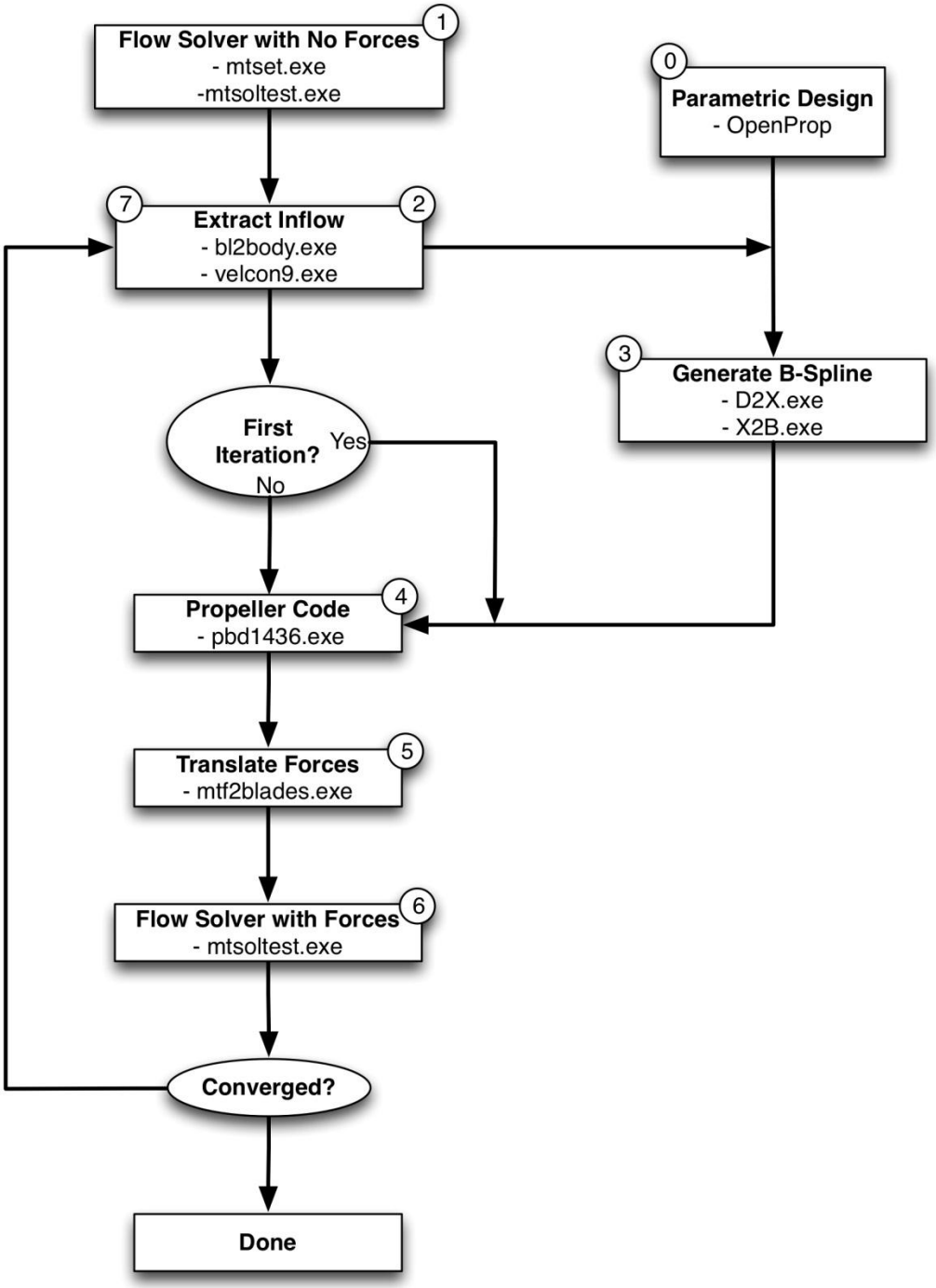

Figure 34: PBD-14.36 and MTFLOW Coupling Process.

The files passed to and from each block in Figure 34 are described in the following section.

User Guide

The codes for the versions of MTFLOW and PBD used for this project are FORTRAN codes compiled into executable files that are run from the Windows command prompt. Each code requires one or more user or code generated input files. Both mtset and mtsoltest require user interaction while running. The codes all require individual activation through the command prompt. Alternatively, a script for running through a complete loop is described at the end of this section. All input files except tdat can be opened, viewed, and edited using a basic text editor. tdat can be opened in the text editor but is unintelligible. A copy of all the input files generated for coupling this project's Propeller 1 design is included as Appendix B.

Step 0: Initial Propeller Design

This method requires a preliminary propeller design prior for input into PBD. The OpenProp parametric design described in Chapter 2 is one such method. A Matlab script to extract the required geometry information from OpenProp was developed for this thesis and is included in Appendix B. Other programs capable of providing the inputs required for the PBD admin file summarized in Step 4 and detailed in the PBD-14.36 user manual 50[2] would also work.

Step 1: Flow Solver with No Forces

mtset.exe

Input file:	walls (no file extension)
Output file:	tdat (no file extension)

mtset is the program that defines the fluid flow domain in which the propeller operates and for which MTFLOW calculates the wake profiles. It reads a user defined geometry file called "walls," which defines the axisymmetric boundaries and any elements within the domain, and generates the streamlines along which MTFLOW calculates the wake data. Within mtset the user has the option of refining the mesh characteristics and checking the input geometry for accuracy. mtset outputs a fluid domain state file, tdat, which defines the wake profile throughout the domain.

mtsoltest.exe

Input file:	tdat (no file extension)
Output files:	tdat (no file extension)
	OUTVEL.tec

mtsoltest is the primary MTFLOW calculation program. It reads the state file tdat, and ultimately updates tdat with the new domain characteristics when the user saves data. Within mtsoltest, the user can set or adjust calculation parameters, view plotted fluid properties, and execute the Euler solver calculations until the desired convergence has been reached. mtsoltest updates tdat with the new system state and generates the file OUTVEL.tec. OUTVEL.tec is a tecplot formatted file which describes the MTFLOW axial and tangential velocity output along the flow streamlines with x,y,z coordinates.

Step 2/7: Extract Inflow

bl2body.exe

Input files:	OUTVEL.tec
	mtcouple.inp
Output file:	VELJOIN.tec

bl2body is the first of two codes that translate the format of the velocity file. It reads the output from MTFLOW, OUTVEL.tec, and a user defined file, mtcouple.inp, which defines the Reynolds number and inlet Mach number. bl2body converts the coordinates and velocities in OUTVEL.tec from the MTFLOW format to the velcon9 format requirement and outputs the new file VELJOIN.tec.

velcon9.exe

Input file:	VELJOIN.tec
Optional input files:	velcon9.in
	velcon9.bat
Output file:	pbdin.vel

velcon9 reads VELJOIN.tec, the output from bl2body, and prompts the user to provide values for a series of domain and propeller definition characteristics. MTFLOW considers a much larger domain than what PBD needs for propeller design. velcon9 provides the user the option to limit the domain that PBD uses and requires the axial and radial coordinates for the propeller tip leading edge as a reference point. Alternatively, a batch file, velcon9.bat, can be activated, which runs velcon9.exe and calls velcon9.in to supply answers to all of the required user inputs. This greatly speeds up the iteration run time and allows for more smoothly cycling through the design loop. velcon9 outputs a file pbdin.vel which contains the user limited domain velocity information from MTFLOW in a format which can be recognized by both D2X and PBD.

Step 3: Generate B-Spline

D2X.exe

Input files:	prop.tog
	prop.par
	pbdin.vel
Output file:	prop.xyz

D2X (Design to XYZ coordinates) is the first of two programs designed to take a preliminary propeller design and generate a B-spline polygon for PBD to modify. It requires three input files. prop.tog is an administration file which defines some basic numerical procedure settings and propeller characteristics. prop.par is a propeller design parameter file which inputs a table of the design chord, thickness, camber, phi, skew, and rake data. pbdin.vel is the output from velcon9. D2X does not use the velocity data, but it does use the x,r data locations as streamlines upon which to build the blade geometry. D2X outputs prop.xyz which contains the blade geometry in Cartesian coordinates. Care must be taken to ensure that the leading edge tip of the blade represented in prop.xyz is exactly at x=0. If it is not, the reference x line coordinates must be adjusted and D2X rerun until x=0.

X2B.exe

Input file:	prop.xyz
Output file:	startblade.BSN

X2B reads the Cartesian geometry coordinates contained in propl.xyz and uses them to fit the B-spline surface and evaluate it for errors. It outputs the starting geometry B-spline polygon for the blade to be designed in PBD14.

Step 4: Propeller Code

pbd1436.exe

Input files:	prop.in
	startblade.BSN or PBDOUT.BSN
	pbdin.vel
Optional input file:	prop.adm
Output files:	PBDOUT.BSN
	PBDOUT.MTF

pbd1436 is the primary propeller blade design program. It does not require user interaction to run, but it does require significant input file development prior to running. pbd1436 reads three input files. prop.in is the main administration file for running PBD. It calls the other two input files as well as defines an array of other information required to run the program. The PBD-14.36 User Manual should be consulted for a detailed description of each element required in the input file. pbdin.vel is the velocity field output from MTFLOW that has been modified for input to PBD14. It is updated every design iteration but maintains the same filename. The first time PBD is run, the B-spline blade geometry generated by X2B should be used as the design blade. PBD adjusts the blade shape to match the wake conditions and generates a file PBDOUT.BSN. This file should be used in place of startblade.BSN for every subsequent run. A file PBDOUT.MTF is also generated which contains the information necessary to transfer the blade forces on the wake to MTFLOW.

Step 5: Translate Forces

mtf2blades.exe

Input file:	PBDOUT.MTF
	mtf2blades.in
Output file:	blades.xxx

mtf2blades translates PBDOUT.MTF, the blade force output file from pbd1436, using parameters contained in mtf2blades.in, into a format readable by MTFLOW. It is important to note that mtf2blades generates the output with the filename blades.xxx. mtsoltest will not recognize the file if it has any extension in the filename, however, so the file must be renamed to "blades" with no extension.

Step 6: Flow Solver with Forces

mtsoltest.exe

Input files:	tdat (no extension)
	blades (no extension)
Output files:	tdat (no extension)
	OUTVEL.tec

mtsoltest is run the same way as before. The difference is that it will also load the blade force file as long as it is stored in the active folder along with tdat. When the wake converges to a satisfactory criteria, write the data to the solution state file and close the program. This will update tdat and OUTVEL.tec.

Step 7: Extract Inflow

Step 7 is exactly the same as Step 2.

Repeat Steps 4-7 until convergence criteria is met.

Automating the coupling loop

It is quite tedious to complete more than one or two design iterations when individually running each subprogram. To facilitate a quicker design process, three batch file programs were adapted from previous work by Kerwin which can be used to cycle through the design process. This method requires the folder structure with user defined input files shown in Figure 35prior to use.

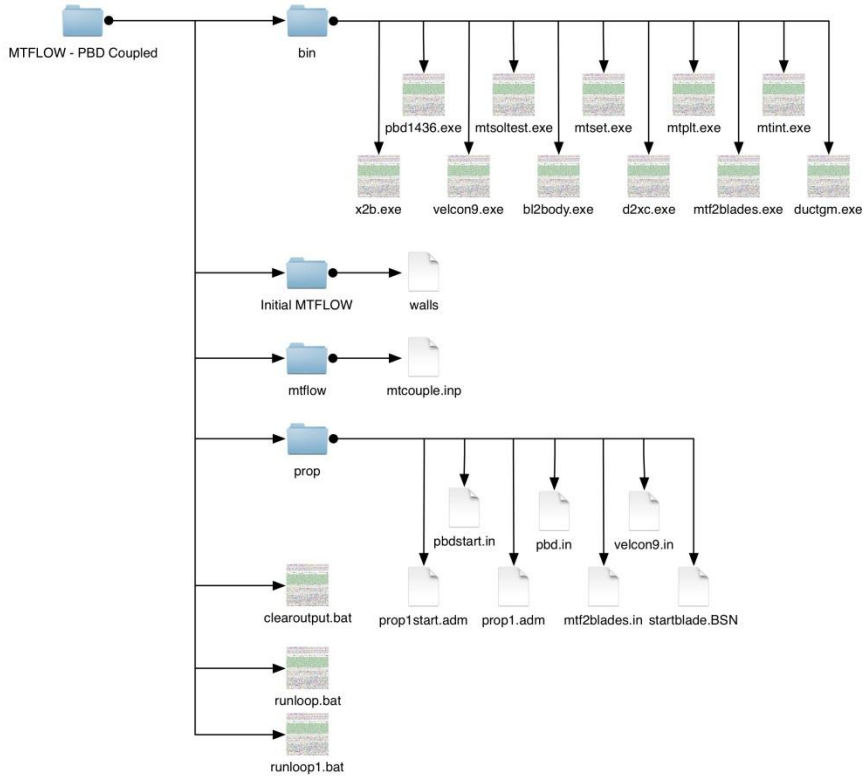

Figure 35: File structure for running MTFLOW/PBD batch files

Prior to running the batch files, steps 1-3 should be accomplished in the "Initial MTFLOW" folder, and all user defined input files should be generated and in place as shown in Figure 35.

For the first iteration, the batch file runloop1.bat should be run. This program copies the necessary velocity file output from the initial runs and calls pbdstart.in which in turn calls prop1start.adm and startblade.BSN as the required inputs. It then proceeds to cycle through steps 4-6. Step 6 runs mtsoltest, which requires user interaction. When the new state file is saved at the completion of step 6, the first iteration is complete. For all further iterations the batch file runloop.bat should be run. This program starts with step 7, before jumping to step 4, at which point it calls pbd.in, which in turn calls prop1.adm and PBDOUT.BSN. It then proceeds to run through steps 4-6 in the same manner as runloop1.bat. This process can be repeated until the desired results are achieved. A third batch file is include, clearoutput.bat, which simply deletes all of the generated output files from the loop, allowing a clean restart if desired.

The batch files, runloop1.bat, runloop.bat, and clearoutput.bat are included in Appendix A.

Some notes on using mtsoltest:

It will be obvious if the blades file loads properly or not. There is a loading message when it does, and there will be obvious flow influences in the plot window.

Sometimes the graphics in the plotting window become mirrored and/or flipped. One way to restore them is to maximize the plot window and then restore it to the original size.

Changing what variables/characteristics were plotted was very helpful for visual references as to what was happening during the calculation iterations within MTFLOW.

Questions remained about the correct calculation settings to use for bollard pull and zero flow starting condition. The explanations for settings in the user manual were not very clear for this particular case. The desired case was to set a starting condition with a mass flow at or very close to zero and solve for the final flow speed generated by the propeller forces acting on the fluid. Mass flow in MTFLOW is indirectly set by assigning the inlet Mach number, but it seemed as if none of the calculation options allow for updating the Mach number. If that is truly the case, then MTFLOW may not be able to determine the effective wake solution for a propeller operating in the bollard pull condition. One alternative that came to light following the testing described in Chapter 5 could be to use some of the measured fluid velocity data to generate a guess for the inlet Mach number.

APPENDIX C
PBD and MTFLOW Input and Batch Files

The following are copies of the user defined input files and batch files required to run a coupled design with PBD14.36 and MTFLOW. They can be implemented, written, and saved through the basic Windows text editor. Care should be taken that they are saved with the designated file extensions (or lack thereof). Windows defaults to saving files as a .txt file, which PBD and MTFLOW will not recognize.

walls

```
Open prop, expanded domain
-4.828 9.010 0.0 5.0
4.010       0.246
3.885       0.291
3.755       0.318
3.650       0.330
3.538       0.333
3.174       0.333
2.816       0.333
2.750       0.333
2.185       0.333
1.808       0.333
1.533       0.333
1.319       0.333
1.275       0.333
1.250       0.333
1.132       0.292
1.100       0.292
0.946       0.292
0.786       0.292
0.391       0.292
0.068       0.292
0.000       0.292
-0.207      0.292
-0.328      0.292
-0.385      0.290
-0.485      0.277
-0.595      0.247
-0.692      0.201
-0.760      0.147
-0.797      0.102
-0.817      0.063
-0.825      0.031
-0.828      0.000
```

mtcouple.inp

```
0                ! Reynolds number
0.01             ! inlet mach number
0.0              ! Vship (ignored)
0.000            ! x location of LE tip (ignored)
0.000            ! r location of LE tip (ignored)
xxx              ! MTFLOW case name     (it needs it)
                 ! pbd input file name  (ignored)
```

velcon9.bat

```
velcon9 <velcon9.in
```

velcon9.in

```
VELJOIN.tec
pbdin.vel
0      1
1
20
22
60
0
```

prop.tog

```
prop.tog
prop.par
pbd.vel
NINT1  NINT2   DIV                                  Numerical Procedure
15     15      2.0
DIAM   XLOC    RHOR
2.0    0.0     0.2917
ISPC    JSPC    NSEC    NPTS    ISURF    IDIR     Output parameters
1       1       20      15      0        0
V
0.001   0.1   0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  0.999
```

prop.par

```
TITLE=''prop.par''
VARIABLES = r/R c/D t/c f/c phi skew rake/d
ZONE T=''Input Blade Parameters'', I=24, F=POINT
0.29  0.2177      0     0.0186      16.3469     0     0
0.35  0.1983      0     0.0177      14.3220     0     0
0.40  0.1868      0     0.0170      12.7817     0     0
0.45  0.1777      0     0.0162      11.5174     0     0
0.50  0.1706      0     0.0155      10.4770     0     0
0.55  0.1652      0     0.0147      9.6037      0     0
0.60  0.1612      0     0.0140      8.8600      0     0
0.65  0.1586      0     0.0132      8.2195      0     0
0.70  0.1571      0     0.0125      7.6621      0     0
0.75  0.1568      0     0.0117      7.1725      0     0
0.80  0.1573      0     0.0110      6.7389      0     0
0.82  0.1577      0     0.0107      6.5791      0     0
0.84  0.1580      0     0.0104      6.4262      0     0
0.86  0.1581      0     0.0101      6.2798      0     0
0.88  0.1579      0     0.0098      6.1394      0     0
0.90  0.1564      0     0.0095      6.0045      0     0
0.92  0.1532      0     0.0092      5.8745      0     0
0.94  0.1466      0     0.0089      5.7485      0     0
0.96  0.1333      0     0.0086      5.6254      0     0
0.97  0.1225      0     0.0084      5.5649      0     0
0.98  0.1057      0     0.0083      5.5051      0     0
0.99  0.0754      0     0.0081      5.4462      0     0
1.00  0.0370      0     0.0080      5.3880      0     0
```

prop.par

```
PBD-14.3 admin file for Wilkins Thesis -- prop1 open propeller
PBDOUT.BSN                          :file name for blade b-spline net
pbdin.vel                           :file name for wake field
5     9 11                          :nblade, nkey, mkey
0     1                             :ispn (0=uniform,1=cos),iffixlat
10 1 2 3 4 5 6 7 8 9 10             :mctrp,mc(n)
5     0.0    0    0.0              :ihub, hgap, iduct, dgap
0     0                             :IHUB_SUBLAY, IDUC_SUBLAY
0     0                             :HDWAK, NTWAKE
0.0    0                            :Cq, MXITER
0.0    0.0                          :OVHANG(1), OVHANG(3)
24    -1    0    0                  :nx,ngcoeff,mltype,mthick
3                                   :IMODE
0    0.01    0.02                   :NWIMAX, RCZERO, RGROW   *nwimax
5    0.001    0.01    5.0    1      :niter,tweak,bulge,radwgt,nufix
0    1    1    8                   :I_BSHAPE,IHUB_SL,IDUC_SL,N_GENLINE
1    0.02                           :nplot,hubshk
0    0                              :NOPT, NBLK
0.32808 1.0    1.5  0.05            :ADVCO XULT XFINAL DTPROP

0.0261    0.0273    0.0283    0.0291    0.0298    0.0302    0.0306    0.0309
0.0312    0.0313    0.0314    0.0314    0.0314    0.0312    0.0310    0.0304
0.0295    0.0279    0.0268    0.0252    0.0229    0.0189    0.0107    0.0000

0.2917    0.3500    0.4000    0.4500    0.5000    0.5500    0.6000    0.6500
0.7000    0.7500    0.8000    0.8200    0.8400    0.8600    0.8800    0.9000
0.9200    0.9400    0.9500    0.9600    0.9700    0.9800    0.9900    1.0000

0.0208    0.0201    0.0196    0.0190    0.0184    0.0178    0.0172    0.0166
0.0160    0.0154    0.0148    0.0145    0.0142    0.0139    0.0135    0.0130
0.0124    0.0116    0.0111    0.0105    0.0098    0.0090    0.0082    0.0074

0.0080    0.0080    0.0080    0.0080    0.0080    0.0080    0.0080    0.0080
0.0080    0.0080    0.0080    0.0080    0.0080    0.0080    0.0080    0.0080
0.0080    0.0080    0.0080    0.0080    0.0080    0.0080    0.0080    0.0080

0.7394    0.7828    0.8037    0.8183    0.8296    0.8383    0.8452    0.8506
0.8550    0.8586    0.8616    0.8627    0.8637    0.8646    0.8654    0.8662
0.8669    0.8672    0.8672    0.8672    0.8672    0.8671    0.8670    0.8669

1.4789    1.5655    1.6074    1.6365    1.6592    1.6767    1.6904    1.7013
1.7101    1.7173    1.7233    1.7254    1.7273    1.7291    1.7308    1.7323
1.7337    1.7344    1.7344    1.7344    1.7344    1.7342    1.7341    1.7338

-0.2051   -0.1962   -0.1778   -0.1622   -0.1490   -0.1376   -0.1277   -0.1190
-0.1114   -0.1046   -0.0986   -0.0963   -0.0942   -0.0922   -0.0902   -0.0883
-0.0865   -0.0847   -0.0838   -0.0829   -0.0820   -0.0811   -0.0801   -0.0792

-0.4101   -0.3924   -0.3557   -0.3244   -0.2981   -0.2753   -0.2554   -0.2380
-0.2227   -0.2092   -0.1971   -0.1927   -0.1884   -0.1843   -0.1804   -0.1767
-0.1730   -0.1695   -0.1676   -0.1658   -0.1640   -0.1621   -0.1603   -0.1584
```

*Note: Each separate shaded (or not shaded) group of datapoints in the second portion of the file should be in extended along one line (row) in the actual file. There should be no separation between rows. This data properly displayed would look like an 8 row by 24 column table. This data is specific to the Propeller 1 design. Other designs may have more or fewer columns depending on the design specificity. See the PBD User Manual (Chrisospathis, 2001 for more detail).

mtf2blades.in

```
PBDOUT.MTF
0,1
blades.xxx
```

runloop1.bat

```
REM Batch file for first iteration of coupled run with rotor. Using
PBD1436.exe

cd initial mtflow
copy tdat ..\mtflow\tdat
copy OUTVEL.tec ..\mtflow\OUTVEL.tec

cd ..\mtflow
..\bin\bl2body
del ..\prop\veljoin.tec
copy veljoin.tec ..\prop

REM Change to rotor directory and run VELCON, PBD14 and MTF2BLADES:
cd ..\prop
..\bin\velcon9 <velcon9.in
..\bin\pbd1436 <pbdstart.in
..\bin\mtf2blades <mtf2blades.in

REM Change to mtflow directory, build blades.%1 and run MTSOL
cd..\mtflow
del blades
copy ..\prop\blades.xxx blades
..\bin\mtsoltest

cd ..
```

runloop.bat

```
REM Batch file for coupled run with rotor. Using PBD1436.exe
cd mtflow
..\bin\bl2body
del ..\prop\veljoin.tec
copy veljoin.tec ..\prop

REM Change to rotor directory and run VELCON, PBD14 and MTF2BLADES:
cd ..\prop
..\bin\velcon9 <velcon9.in
..\bin\pbd1436 <pbd.in
..\bin\mtf2blades <mtf2blades.in

REM Change to mtflow directory, build blades.%1 and run MTSOL
cd..\mtflow
del blades
copy ..\prop\blades.xxx blades
..\bin\mtsoltest

cd ..
```

clearoutput.bat

```
cd initial mtflow
del GridLower.dat
del GridTotal.dat
del OUTBL.tec
del OUTVEL.tec
del tdat
del mtgpar
del ORIGGRID.tec
del plot.ps

cd ..\mtflow
del blades
del VELJOIN.tec
del OUTVEL.tec
del tdat

cd ..\prop
del blades.xxx
del pbdin.vel
del PBDOUT.BSN
del PBDOUT.BUG
del PBDOUT.CBD
del PBDOUT.CLW
del PBDOUT.CMF
del PBDOUT.CMV
del PBDOUT.CP
del PBDOUT.CPV
del PBDOUT.GSP
del PBDOUT.IBG
del PBDOUT.KTQ
del PBDOUT.MTF
del PBDOUT.OBG
del PBDOUT.RDC
del PBDOUT.VCP
del VELJOIN.tec
```

Appendix D
Xenus Amplifier setup procedure

CME2 stores motor and amplifier parameters in a number of formats and locations. The amplifier has both flash and RAM memory. Flash memory is non-volatile and will store data until it is replaced by new data. RAM is volatile and will only hold data until the amplifier is powered down or reset. Motor parameters and basic setup information is always stored in flash memory. User inputs to amplifier settings such as gains and limits are set and changed in RAM but can be stored to flash. Operating information such as actual position and current are only stored in RAM and cannot be saved to flash. Any data that can be saved to the amplifier flash memory can also be backed up as files saved on the operating computer hard drive. The following are step by step instructions for initializing the motor controller to operate the Ketcham motor. The screenshots display the correct settings for motor parameters and control gains where indicated.

Motor Parameters

Motor parameter files are saved as *.ccm files. The motor parameters can be saved to or loaded from hard disk or amplifier flash memory by clicking the appropriate disk or chip icon on the bottom left corner of the window. Figure 36shows a screen shot of the correct settings for the Parker kit motor with model number K089300-7Y2. Additionally, the required motor parameters are shown in Table 16: CME2 Motor Parameters. The source for the parameter values is a motor data spreadsheet provided by Parker Hannifin Corporation.

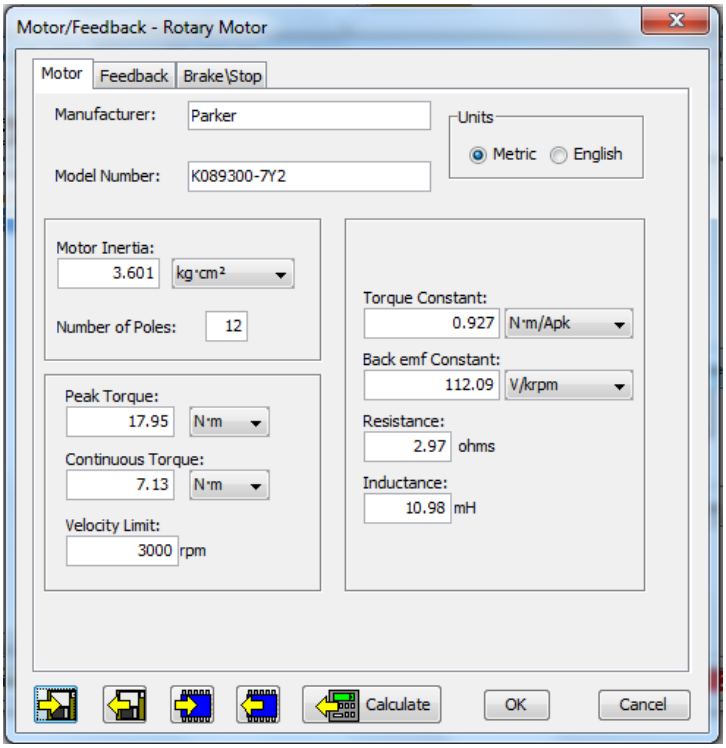

Figure 36: CME2 Motor Feedback Parameters

Figure 37shows the correct Hall Count Multiplier and Counts per Rev. values. Ketcham's lab notebook listed values to be HCM = 3 and CpR = 12, but these were incorrect. HCM = 1 and CpR = 36 were verified as the correct settings through operational tests by using an optical frequency counter to ensure the actual motor speed matched the commanded motor speed.

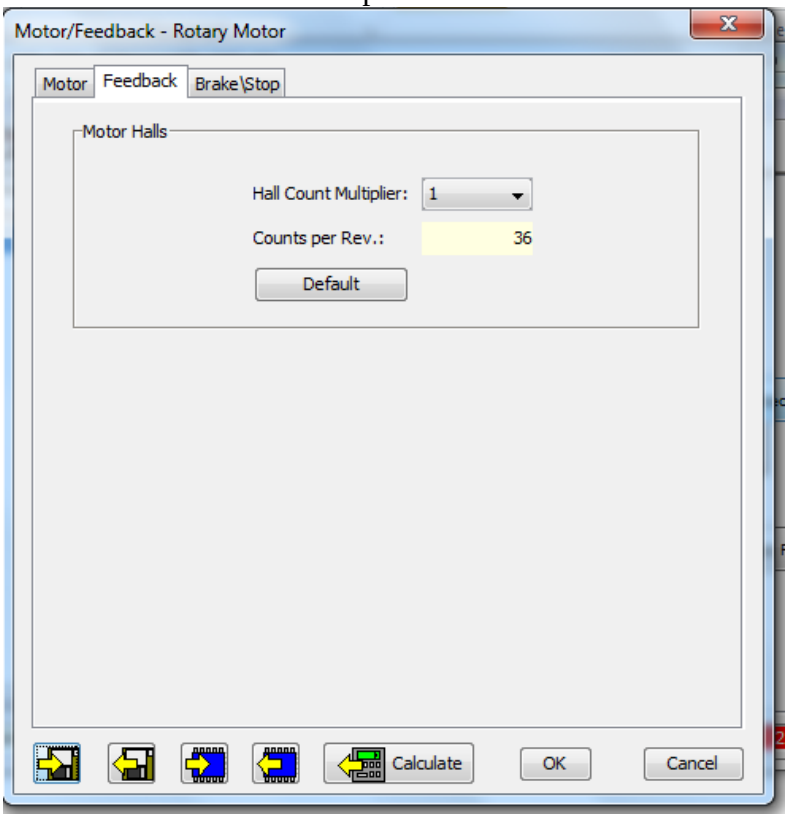

Figure 37: CME2 Feedback Parameters

Figure 38 shows the Brake/Stop settings. These were all left in the default configuration.

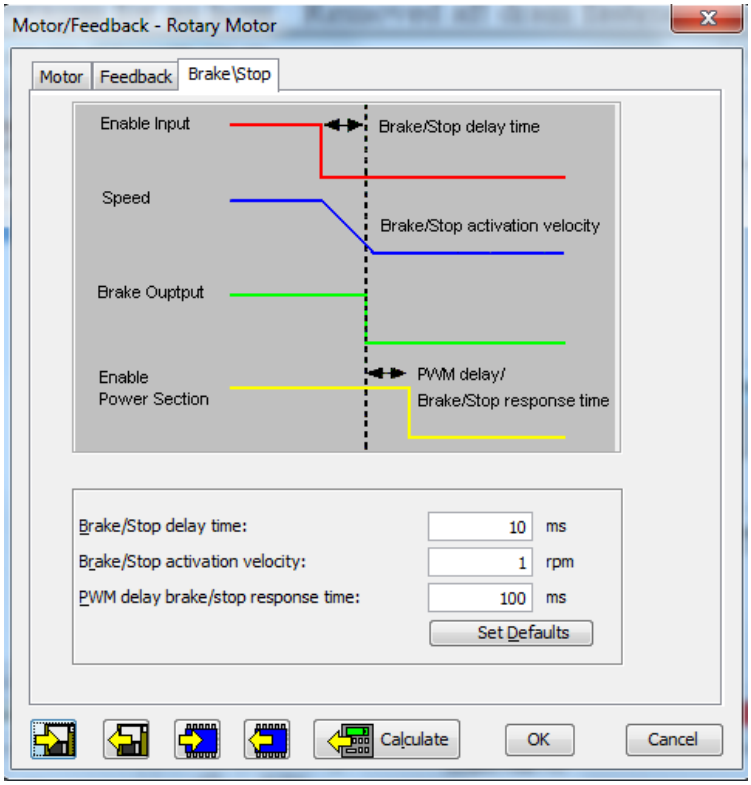

Figure 38: CME2 Brake/Stop Parameters

Motor Parameters					
Manufacturer	Parker		**Back emf Constant**	112.09	V/krpm
Units	Metric		**Resistance**	2.97	ohms
Model Number	K089300-7Y2		**Inductance**	10.98	mH
Motor Inertia	3.601	$kg{\cdot}cm^2$	**Hall Count Multiplier**	1	
Number of Poles	12		**Counts per Rev**	36	
Peak Torque	17.95	N·m	**Brake/Stop delay time**	10	ms
Continuous Torque	7.13	N·m	**Brake/Stop activation velocity**	1	rpm
Velocity Limit	1500	rpm	**PWM delay brake/stop response time**	100	ms
Torque Constant	0.927	N·m/Apk			

Table 16: CME2 Motor Parameters

Initial tuning for amplifier parameters.

In the Basic Setup menu, use the "Change Settings" button to set the operating mode to "Function Generator." The Function Generator is used for setting up and fine tuning the amplifier settings for the motor.

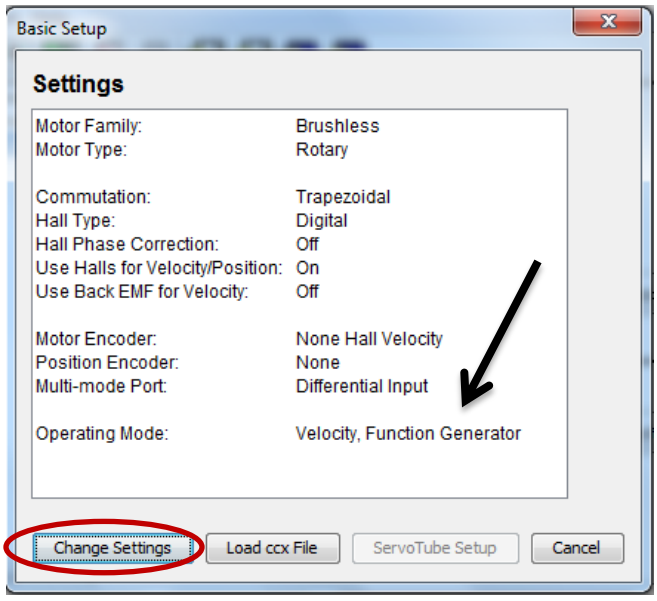

Figure 39: CME2 Basic Setup

Start by conducting manual or auto phasing of hall effect sensor: The values shown in Figure 40 aren't currently phased. The hall offset needs to be updated to match the black arrow to the red square.

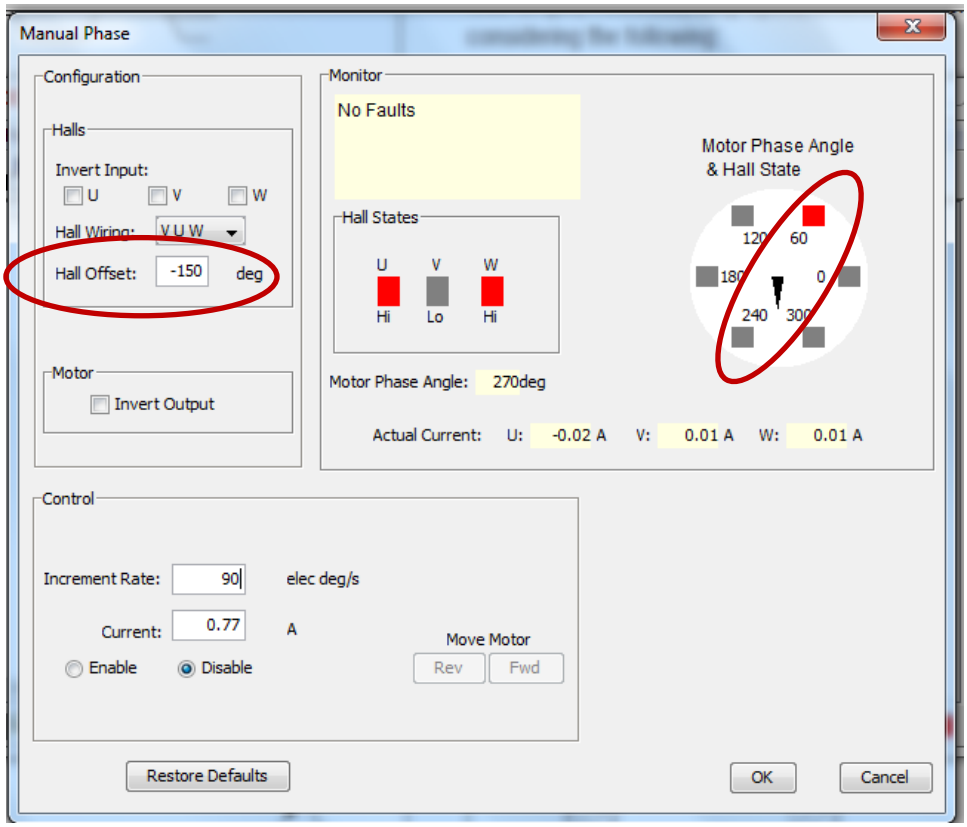

Figure 40: CME2 Manual Phasing

The function generator, accessed by the button circled in Figure 41, generates either a square or sine wave command velocity. When the function generator is active and the motor is fully enabled, use the scope function to observe motor response. Adjust the control-gain parameters (Vp, Vi, Cp, Ci) in the scope windows shown in Figure 42 to match the response to the commanded velocity until the desired response performance is achieved. Once the motor responds as desired, amplifier gain parameters can be saved to the amplifier flash memory or to the disk as a *.ccx file.

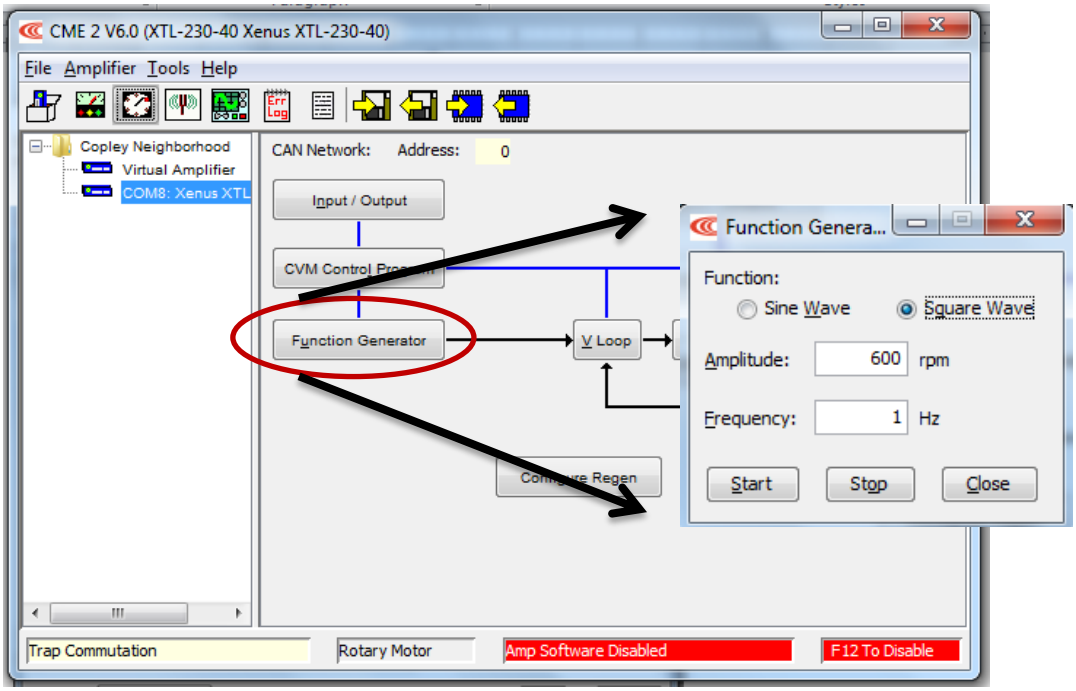

Figure 41: CME2 Function Generator

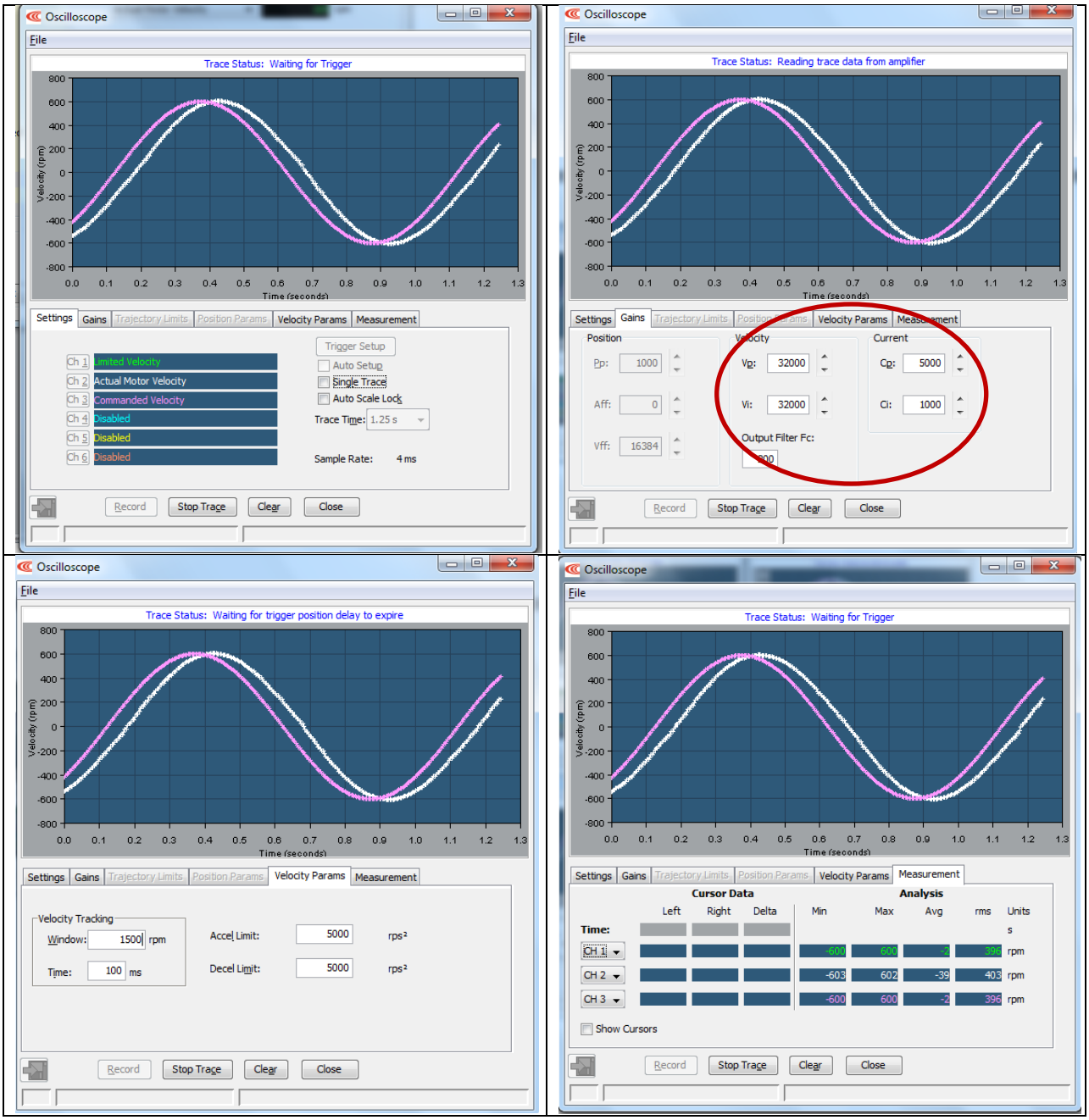

Figure 42: CME2 Scope Views

Scope data can be saved (as *.sco, *.txt, and *.xls files) by clicking on the disk in the bottom left corner of the window. Scope settings can be saved by clicking file→save settings (as *.scc file).

Operations

Once the control gains and velocity parameters are tuned, click “change settings” in the basic setup window to update the control input type to “software programmed” as shown in Figure 43

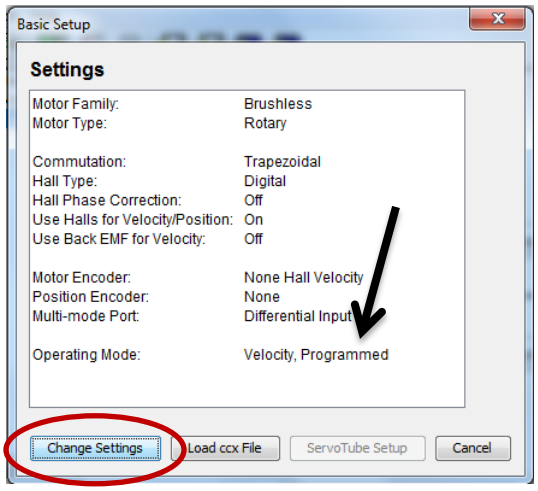

Figure 43: CME2 Basic Setup

Set desired velocity by clicking on the “Programmed Velocity” block on the feedback loop diagram shown in Figure 44.

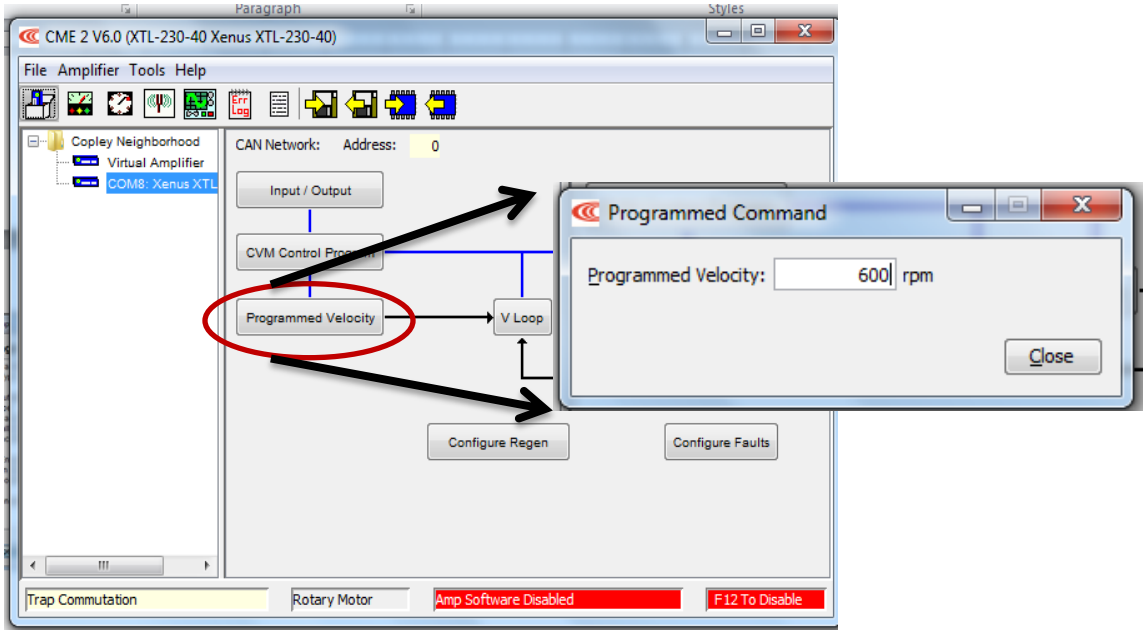

Figure 44: CME2 Programmed Velocity Control

Use the scope to monitor motor operation as desired. Figure 45shows the scope view of steady state operation at 600 rpm.

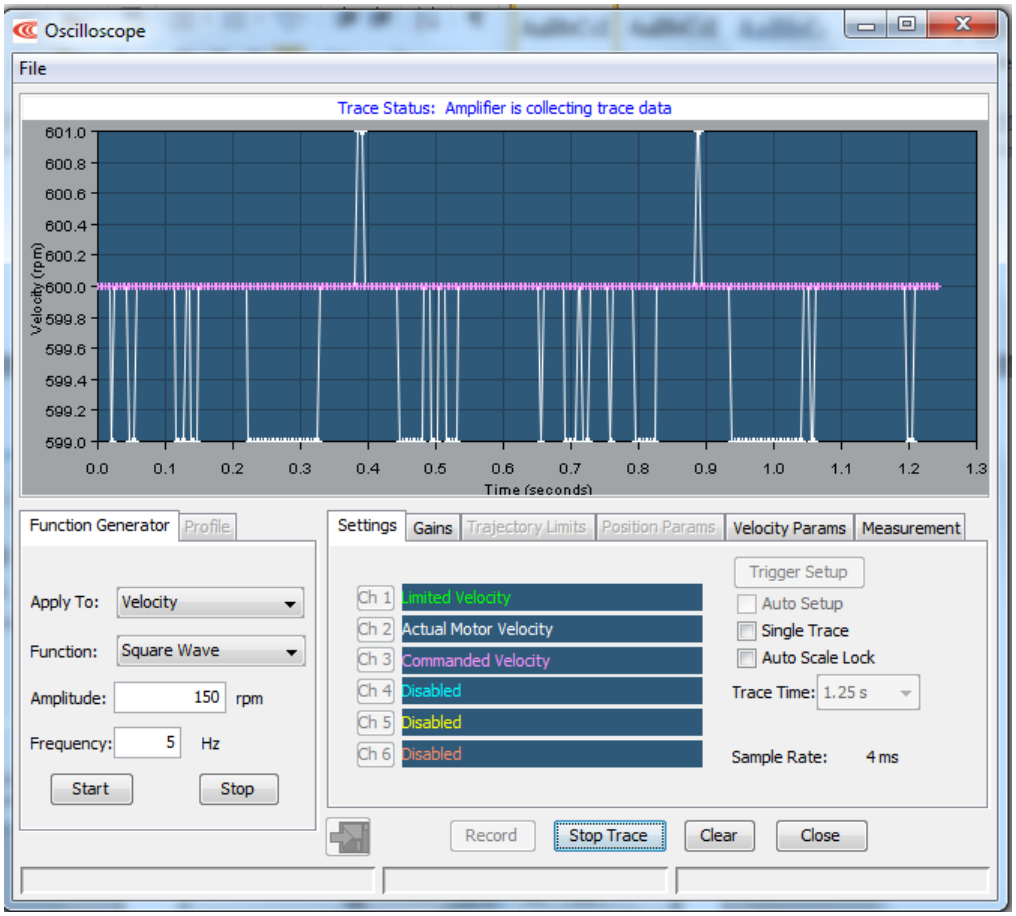

Figure 45: CME2 Scope view with motor operating at 600 rpm.