Autonomous Underwater Vehicle Propulsion Design

Richard S. Duelley

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
In partial fulfillment of the requirements for the degree of

Master of Science
In
Aerospace Engineering

Wayne L Neu
Michael Philen
Craig Woolsey

August 12, 2010

Blacksburg, Virginia

Keywords: AUV, OpenProp, Efficiency, Propulsion, Propeller, NeuMotor

Autonomous Underwater Vehicle Propulsion Design

Richard S Duelley

Abstract

The goal of this design process was to achieve the most efficient propulsive system for the candidate autonomous underwater vehicle (AUV) as possible. A mathematical approach, using fundamental motor equations and derived quantities, was used to characterize and select an efficient brushless electric motor for the propulsion system. A program developed at MIT, Massachusetts Institute of Technology, called OpenProp versions 1 and 2.3 was utilized to design a custom propeller that maximizes the efficiency of the system.

A brushless electric motor was selected for the candidate AUV based on a survey of available off the shelf motors and a mathematical characterization process. In parallel with the motor characterization a propeller design was optimized using OpenProp v1 to perform a parametric analysis. OpenProp v2.3 was then used to design a unique propeller for the selected motor. The propeller design resulted in a final propeller with an efficiency of 79.93%. The motor characterization process resulted in two candidate motors being selected, the NeuMotor 1925-3Y and NeuMotor 1521-10.5Y, for in house testing and evaluation. A total propulsive system efficiency of between 44% and 46% was achieved depending on which motor is selected for the final design.

Contents

List of Figures

List of Tables

List of Appendices

Chapter 1 - Introduction

The process described herein is a continuation and utilization of the process developed by James A Schultz [1]. This design, however, is not limited to off the shelf components. The motor selection process is similar to the process described by Schultz but the propeller is a completely custom design. The goal of this design process was to achieve the most efficient propulsive system possible. The equations presented by Schultz and confirmed by Hendershot and Miller [5] were used to characterize and select a brushless electric motor. A program developed at MIT called OpenProp [2] was utilized to design the custom propeller. First a parametric design was undertaken with OpenProp v1 to determine a range of design RPM and possible propeller diameters. A max propeller diameter rule, based on experience with Virginia Tech's fleet of 475 vehicles, is also proposed. A unique propeller design was then created using OpenProp v2.3 based on the parametric propeller results and the motor characterization results. The unique propeller design and motor selection process are both undertaken in parallel, results from the motor characterization process are required to complete the unique propeller design and results from the unique propeller design are required to complete the motor selection process.

This document is meant to be used as a guide for future designs of similar requirements and is written as such starting with the Motor Selection process below. Throughout this document the following design criteria for a proposed candidate AUV are referenced and used to make the final design decisions.

- Vehicle Diameter = 6.9 inches
- Vehicle Trust Required, found by CFD [3] = 8.67 N
- Vehicle Drag, found by CFD [3] = 8.11 N
- Vehicle Speed = 2.0 m/s

Chapter 2 - Motor Selection

2.1 Motor Classification

The two most common types of electric motors are brushed and brushless. A brushed motor has stationary contacts that transfer the electrical energy to the coils as the motor turns. A brushless motor uses an electronic controller to alternate power to several different groups of coils, called phases, which are housed within the motor. There are several key advantages to using a brushless motor design over an equivalent brushed design. A brushless motor is more efficient, lasts longer (no brushes to wear out) and produces no ionizing sparks when compared to an equivalent brushed electric motor. The motor selection process presented here was written with brushless motors in mind.

2.1.1 Physical

There are two main types of brushless electric motors, an interior-rotor and an exterior-rotor. An exterior-rotor motor consists of a series of magnets that rotate around an internal set of coils. A motor with an interior-rotor is the exact opposite and instead has a cluster of magnets that rotate inside the coils as shown below in Figure 2-1. A more detailed discussion of the physical traits of a wide variety of motors is presented by Shultz [1] and Hendershot and Miller [5]. Both interior and exterior-rotor motors were considered in the process described below.

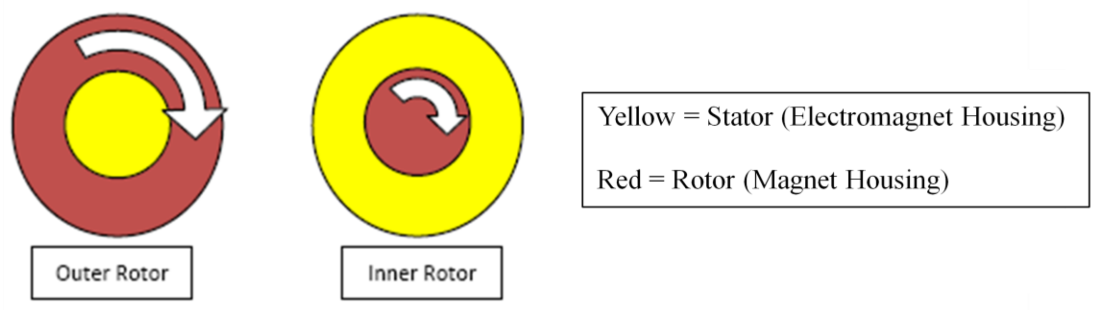

Yellow = Stator (Electromagnet Housing)
Red = Rotor (Magnet Housing)
Outer Rotor
Inner Rotor

Figure 2-1: Interior-Rotor vs Exterior-Rotor [1]

2.1.2 Mathematical

In order to determine if a motor is appropriate for the application in question the motor needs to be classified mathematically. This mathematical classification is faster and cheaper than acquiring each motor and testing them individually. Brushless electric motors are classified by their manufacturer by three motor constants, RPM (rotations per minute) per volt, K_v , the resistance of the motor in ohms, R_m , and the no load current in amps, I_o. Using these three constants and the fundamental motor equations the current at max efficiency, $I_{\eta,\mathrm{max}}$, the current at max power, $I_{\mathrm{P,max}}$, the maximum efficiency achievable by the motor, η_{max} , the *RPM* at max efficiency and the torque provided at maximum efficiency, Q at $I_{\eta,\mathrm{max}}$, can be derived and used to characterize a brushless electric motor. The standard, fundamental motor equations presented below, Equations 2-1 and 2-2, were derived from empirical motor data and Ohm's law [e.g., 5]. In the equations below I is the current drawn by the motor in amps, V is the voltage input to the motor, and Q is the torque in Newton-meters (N-m).

$$RPM = K_v(V - IR_m) \tag{2-1}$$

$$Q = K_q(I - I_o) \tag{2-2}$$

The quantity K_q is the torque constant in (N-m) per amp and is directly related to K_v. For brushless electric motors it can be shown that K_q is defined by Equation 2-3 as shown in Hendershot and Miller [5].

$$K_q = \frac{30}{\pi K_v} \tag{2-3}$$

Equation 2-4 is the definition of power out of a rotational system, Equation 2-5 is the definition of electrical power and Equation 2-6 is the definition of radial velocity, ω, where RPS is the rotations per second.

$$P = Q\,\omega \tag{2-4}$$

$$P = VI \tag{2-5}$$

$$\omega = 2\pi RPS = 2\pi RPM/60 \tag{2-6}$$

The above fundamental motor equations can be utilized to derive an expression for the efficiency of the motor, η_m . The efficiency is defined as the mechanical power out over electrical work in as shown in Equation 2-7. The *RPM* and *Q* appearing in 2-7 can be written in terms of the motor characteristics, voltage and current using 2-1 and 2-2. Once simplified Equation 2-7 yields the final definition of the motor efficiency shown in Equation 2-8.

$$\eta_m = \frac{2\pi Q * RPM}{60\,VI} = \frac{2\pi\left[K_q(I - I_o)\right]\left[K_v(V - IR_m)\right]}{60VI} \tag{2-7}$$

$$\eta_m = \frac{(V - IR_m)(I - I_o)}{VI} \tag{2-8}$$

In order to find the current at max efficiency, $I_{\eta,\max}$, the derivative of Equation 2-8 was taken with respect to the current, I, set equal to zero, Equation 2-9, and then solved for the current, I , resulting in Equation 2-10. Equation 2-10 is one of the key equations used in classifying an electric motor and gives the important motor characteristic current at max efficiency, $I_{\eta,\max}$.

$$\frac{d\eta_m}{dI} = \frac{-(I^2R_m - I_oV)}{I^2V} = 0 \tag{2-9}$$

$$I_{\eta,\max} = \sqrt{\frac{\mathrm{V}I_o}{R_m}} \tag{2-10}$$

The torque at maximum efficiency can then be found using this current in 2-2. By solving Equation 2-5 for the current, I , and substituting into Equation 2-8, the same process used to derive Equation 2-10 can be used to derive the equation for the current at maximum power output, $I_{\mathrm{P,max}}$, Equation 2-13. $I_{\mathrm{P,max}}$ is not specifically used in this motor selection process but it can be an important value if your system is based on maximum power output and not maximum efficiency.

$$P_{electrical} = \frac{(V - IR_m)(I - I_o)}{V\,I} = \frac{\left(V - \frac{P}{V}R_m\right)\left(\frac{P}{V} - I_o\right)}{V(P/V)} \qquad 2\text{-}11$$

$$\frac{dP_{electrical}}{dP} = \frac{-(2\,P\,R_m - (I_oR_m + V)V)}{V^2} \qquad 2\text{-}12$$

$$I_{\mathrm{P,max}} = \frac{V + R_m * I_o}{2 * R_m} \qquad 2\text{-}13$$

The maximum efficiency attainable by the motor, η_{max} , is derived by substituting Equation 2-10 into Equation 2-8 which yields Equation 2-14. $RPM\ at\ I_{\eta,\mathrm{max}}$ is found by substituting $I_{\eta,\mathrm{max}}$, Equation 2-10, into the fundamental RPM equation, Equation 2-1. The resulting RPM, Equation 2-15, is the RPM that is used in the unique propeller design process, see Chapter 3.4.

$$\eta_{max} = \left(1 - \sqrt{\frac{I_o * R_m}{V}}\right)^2 \qquad 2\text{-}14$$

$$RPM\ at\ I_{\eta,\mathrm{max}} = K_v * \left(V - I_{\eta,\mathrm{max}} * R_m\right) \qquad 2\text{-}15$$

K_q, Equation 2-3, is another key value and is used to determine if the motor can generate enough torque to turn the system. Not only must the torque required to turn the propeller be taken into account but the additional torque in the system caused by bearings, seals or any other external loads must also be considered. The torque provided by the motor must be calculated at the current draw of the motor at maximum efficiency, Equation 2-16. Equation 2-16 is derived by substituting Equation 2-10 into the fundamental motor Equation 2-2.

$$Q\ at\ I_{\eta,\mathrm{max}} = K_Q * \left(I_{\eta,\mathrm{max}} - I_o\right)\ \text{in N-m} \qquad 2\text{-}16$$

A set of example calculations is given below in Table 1, orange cells denote inputs and yellow cells denote outputs.

Table 2-1: Sample motor calculations

Motor Name: NeuMotor 1925 3Y	$I_{\eta,max}$ = 4.83 amp
R_m = 0.18 ohm	$I_{P,max}$ = 39.03 amp
I_o = 0.3 amp	η_{max} = 0.8796
V = 14 volt	At $I_{\eta,max}$ RPM = 1785.75
K_V = 136 RPM/volt	At $I_{\eta,max}$ Q (N-m) = 0.3181
K_q = 0.07022 (N-m)/amp	in-oz = 45.04

It is convenient to program all of these calculations into an Excel spreadsheet. Once the equations are programmed in it is straight forward to calculate these values and compare a vast array of electric motors in a relatively short span of time.

No system is perfect, so if the torque provided by the motor exceeds or is less than the required torque to turn the system the above efficiency calculations will not be accurate. If the motor does not provide enough torque to turn the system at maximum efficiency all is not lost. The motor may still be able to turn the system it will just turn said system at a slightly lower RPM and some efficiency will be lost. This is exactly what was encountered with the NeuMotor 1521-10.5Y, see Chapter 5.2 for details. Also, if the motor in question provides more torque than required then it will spin the system slightly faster than the RPM at max efficiency predicts. This off design efficiency will also provide a less optimistic and more realistic efficiency number for the overall system. Thus it is also important to be able to analyze the off ideal, or off design point efficiency of a motor. One does not need to do this for every motor analyzed but it should be done once the preliminary motor selection is complete and the field of motor candidates has been thinned out. The NeuMotor 1925-3Y, shown in Table 2-1, will be used to illustrate the process.

The first step is to determine the torque required, Q_{req}, to turn the system, which is the sum of the torque required to turn the propeller, Chapter 3.3, and any torque added to the system by bearings, seals, Chapter 5.1, or any other external factors. Then utilize the following equations to determine the off efficiency of the motor at the specific torque required. Equation 2-9 is a modified form of Equation 2-8 where I_{req} is the input current, I, in Equation 2-8.

$$I_{req} = (Q_{req}/K_q) + I_0 \tag{2-19}$$

$$\eta_m = (V - I_{req} * R_m) * (I_{req} - I_o) / (V * I_{req}) \tag{2-20}$$

Sample results are shown in Table 2-2 below based on the 1925-3Y. It is interesting to note that, in this case, despite the 1925-3Y being oversized for this example scenario the effect on the efficiency is relatively small with an efficiency reduction of around 3%. The off efficiency RPM was found by utilizing Equation 2-1 with the I_{req} found using Equation 2-19.

Table 2-2: Off design point calculations

Total Torque Required (in-oz)	21.43
Total Torque Required (N-m)	0.1513
Current Required (amp), I	2.456
Watts at 14 volts	34.38
η at above current required at 14V	0.8501
RPM at 14 volts	1843.87

Chapter 3 Propeller Design

3.1 Physical

A propeller is defined by several key features. The first is the radius or distance from the center of the hub to the tip of the blade. The propeller diameter is based on the diameter of the vehicle the propulsion system is being designed for. It is a good rule of thumb to set the maximum propeller diameter for a small AUV to be no more than 85% of the diameter of the vehicle itself. So if the vehicle has a diameter of 6.9 inches the maximum propeller diameter allowed by this rule would be approximately 5.86 inches. This size restriction is to help mitigate possible ventilation of the propeller as the AUV dives or operates on the surface. Ventilation is when the propeller blade draws air from the surface into the blades; this causes a reduced load on the propeller and a significant reduction in available trust. Ventilation may even cause the AUV to be unable to dive or maneuver effectively while on the surface.   This rule was developed from experience gained with Virginia Tech's fleet of 475 AUVs [11]. The minimum diameter depends on the shaft and hub size and is arbitrary. In general a larger diameter propeller and slower RPM will yield a more efficient propeller than a smaller high RPM propeller. A compromise needs to be made between efficiency and practicality, if the propeller is too large it will ventilate near the surface and reduced thrust will result, which can cause the vehicle to not be able to dive or maneuver effectively on the surface. The next propeller characteristic is the pitch and is the measure of how far the propeller would move forward in one revolution if it was moving through a solid. The chord is the distance from the leading edge to the trailing edge measured with a straight line at a given station along the propellers radius. The thickness of the propeller is also a defining feature and is an important feature when the manufacturability of the propeller is in question. Again a compromise needs to be made when choosing the chord and thickness. From a hydrodynamic standpoint, the ideal propeller would be infinitely thin. This ideal propeller is impractical and impossible to manufacture. One must compromise between durability, manufacturability and efficiency.

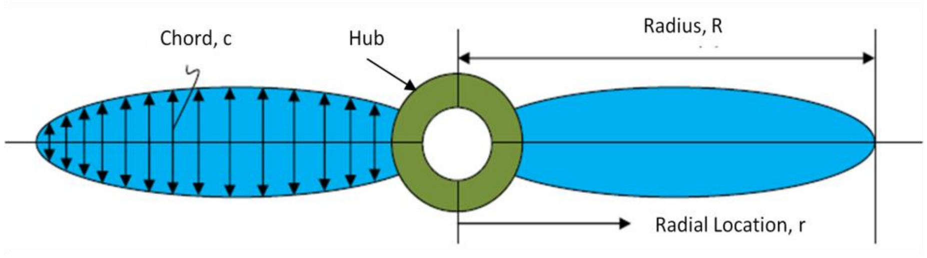


Figure 3-1: Propeller nomenclature [1]

Rake is the angle the propeller blade makes with the centerline of the hub. This can be a forward or rearward angle, a rearward rake is shown in Figure 3-2. If ventilation of the propeller is a concern, like when a small AUV dives for example, adding rearward rake in the propeller can help mitigate the ventilation.

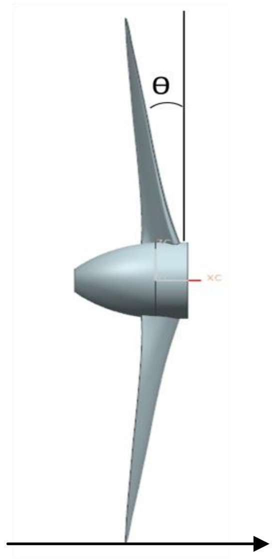


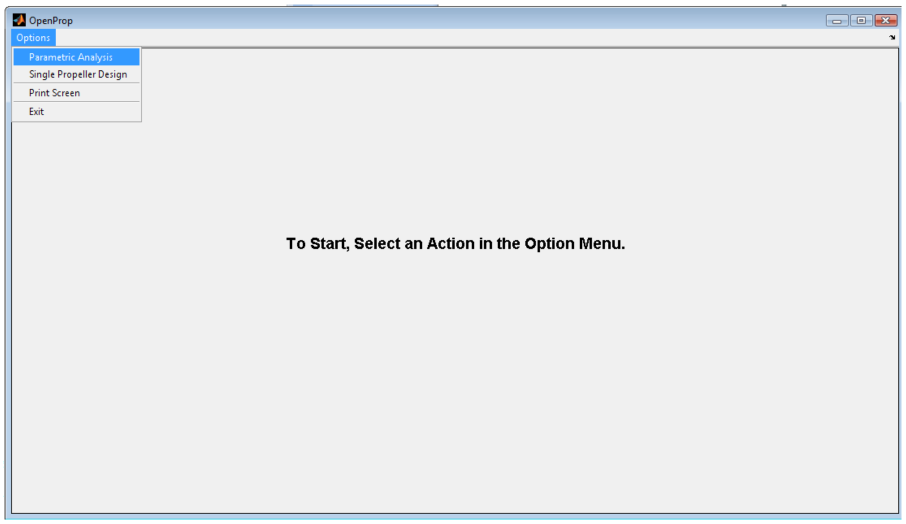
Figure 3-2: Propeller rake, θ

3.2 OpenProp v1 Parametric Design

OpenProp v1 is the first generation of MITs propeller design MATLAB algorithm that utilizes a numerical lifting line theory to predict propeller performance. A detailed discussion of OpenProp can be found in the OpenProp v2.3 Theory Document found in the Reference file included in the v2.3 code download [2] or any of the other documents on the main OpenProp Wiki page. Version 1 was used for the parametric analysis due to its simple graphical user interface, the more advanced capabilities of version 2.3 were not necessary for this aspect of the design process. Chapter 3.2 is a step by step breakdown of how OpenProp v1 was utilized.

3.2.1 Inputs

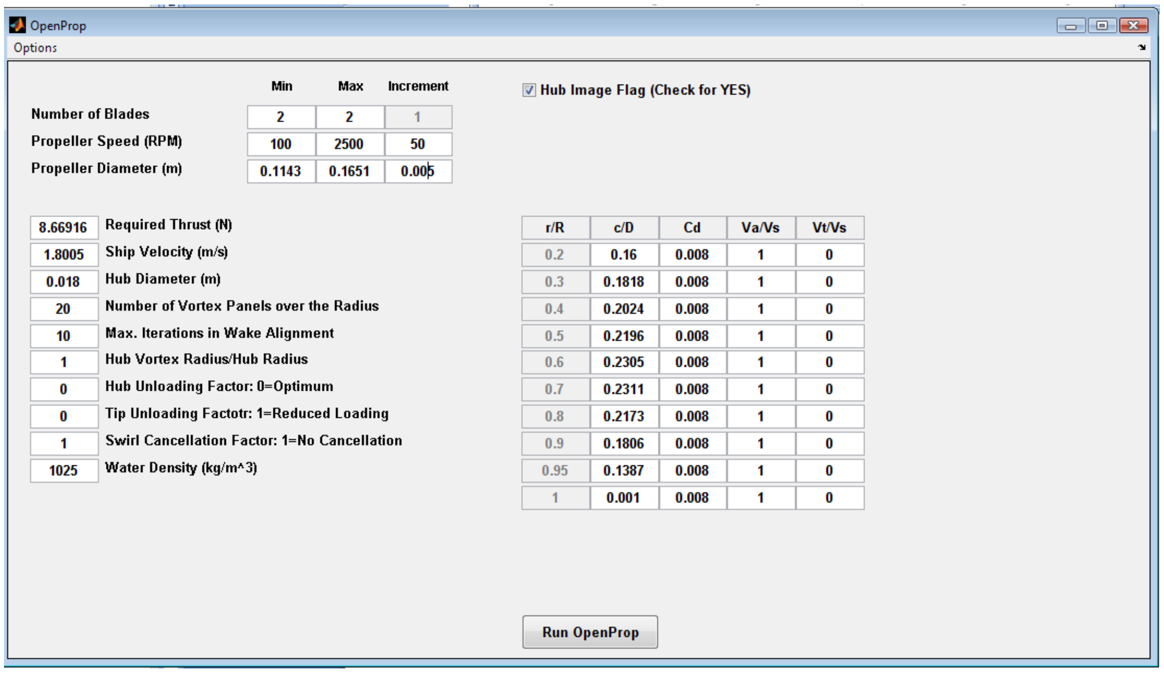
After running OpenProp v1 an intro screen appears, the Parametric Analysis option was utilized for this portion of the propeller design process.



To Start, Select an Action in the Option Menu.

Figure 3-3: OpenProp v1 starting screen

All length units are in meters and the hub diameter must be at least 15% of the smallest diameter in the Propeller Diameter Range. The Water Density is can be changed for a variety of applications like fresh water operation or, in this case, sea water operation. One can also specify the desired Number of Blades, Propeller Speed, Required Thrust, Ship Velocity, and the range of Propeller Diameters that are being considering. At this stage all other inputs were left as the OpenProp defaults. The numbers entered in Figure 3-4 are applicable to vehicle characteristics provided in the Introduction. Depending on the power of the computer being used and other inputs OpenProp may take several minutes to run.



[✓] Hub Image Flag (Check for YES)

	Min	Max	Increment
Number of Blades	2	2	1
Propeller Speed (RPM)	100	2500	50
Propeller Diameter (m)	0.1143	0.1651	0.005

8.66916	Required Thrust (N)
1.8005	Ship Velocity (m/s)
0.018	Hub Diameter (m)
20	Number of Vortex Panels over the Radius
10	Max. Iterations in Wake Alignment
1	Hub Vortex Radius/Hub Radius
0	Hub Unloading Factor: 0=Optimum
0	Tip Unloading Factotr: 1=Reduced Loading
1	Swirl Cancellation Factor: 1=No Cancellation
1025	Water Density (kg/m^3)

r/R	c/D	Cd	Va/Vs	Vt/Vs
0.2	0.16	0.008	1	0
0.3	0.1818	0.008	1	0
0.4	0.2024	0.008	1	0
0.5	0.2196	0.008	1	0
0.6	0.2305	0.008	1	0
0.7	0.2311	0.008	1	0
0.8	0.2173	0.008	1	0
0.9	0.1806	0.008	1	0
0.95	0.1387	0.008	1	0
1	0.001	0.008	1	0

Figure 3-4: OpenProp v1 Input screen

3.2.2 Results

The parametric analysis algorithm outputs a figure that shows an estimation of efficiency vs. propeller diameter. At first glance the graph may look hectic; this is usually caused by low, below 500 RPM, propeller speeds. To clean up the figure these extraneous results were suppressed using MATLAB's Plot Browser function.

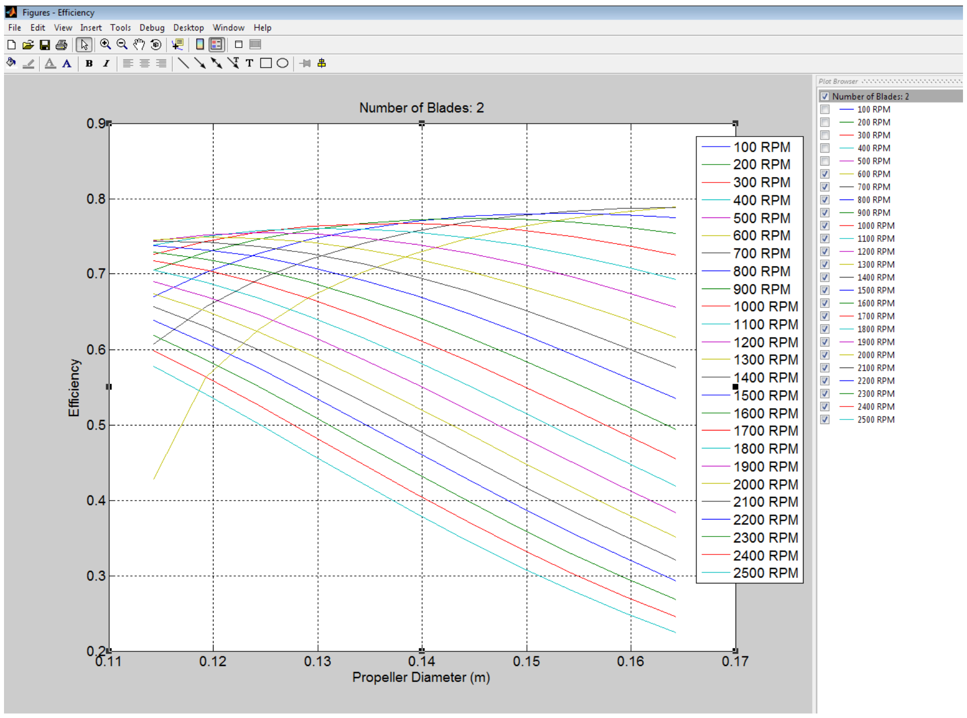


Figure 3-5: Results after clean up

MATLAB reuses colors so it can be difficult to determine which lines correspond to which RPM. The Plot Browser was used to check which lines correspond to which RPM and to eliminate undesirable solutions. Figure 3-5 illustrates that the slower the propeller spins the more efficient it will be but again a compromise needs to be made. It is difficult to find a brushless electric motor that can spin efficiently at such low RPMS, which corresponds to low K_v values, thus the choice of propeller speed cannot be based just on the results shown in Figure 3-5 but also must be based on available electric motors. The plot also makes it clear that the efficiency goes up as the propeller increases in diameter for the lowest RPM values, below 1000. After 1000 RPM is exceeded there is a point of diminishing returns and then a loss of efficiency as the blade increases in size. So let's assume that during the search for a motor it was determined that a motor that operated below 1500 RPM and above 2000 RPM at its maximum efficiently are impractical for the application in question or simply cannot be acquired. Thus any results not in the above mentioned range can be eliminated as shown in Figure 3-6. Using the vehicle diameter given in Chapter 1 it was determined that the maximum propeller diameter would be set to 0.12 meters or approximately 4.73 inches. This maximum diameter was chosen in order to mitigate ventilation of the propeller while on the surface and because the parametric plot produced with OpenProp shows the efficiency dropping significantly when the propeller diameter exceeds 0.12 meters, Figure 3-6. Figure 3-6 shows that we can expect a propeller efficiency of around 65% to 74%. It is important to note that this efficiency prediction is based on the default propeller geometry provided by OpenProp. The parametric design was then run again, setting the max propeller diameter at 0.12 meters and a much lower minimum propeller diameter, the results are shown in Figure 3-7.

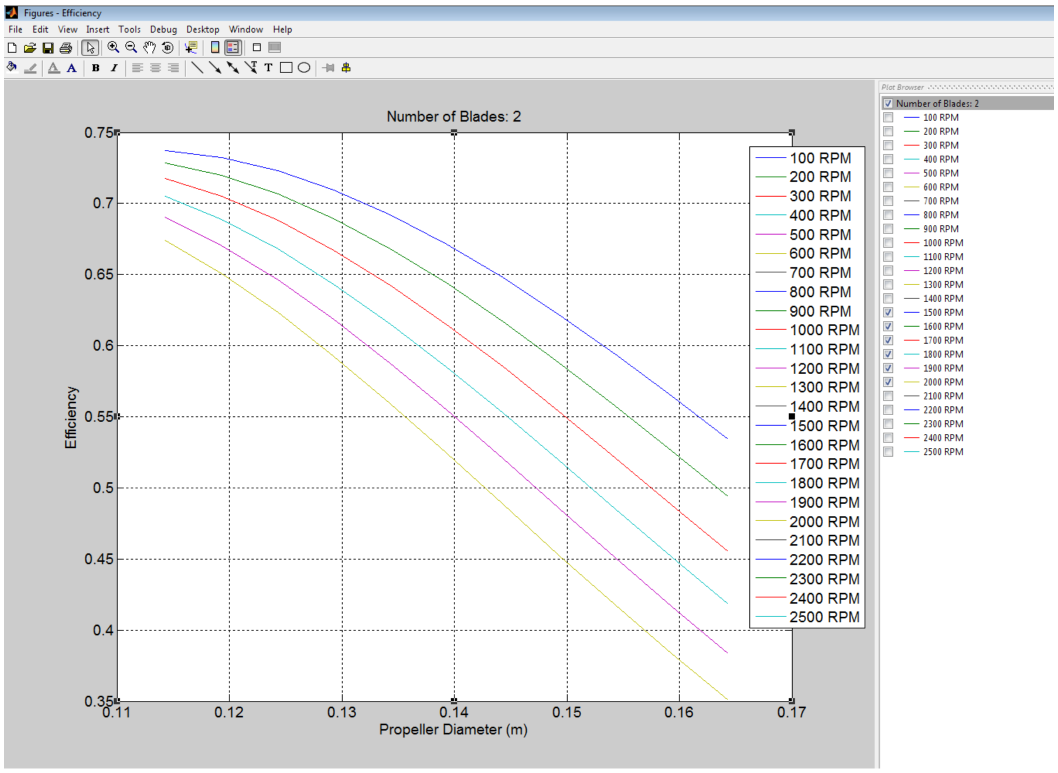


Figure 3-6: Results after elimination of undesired results

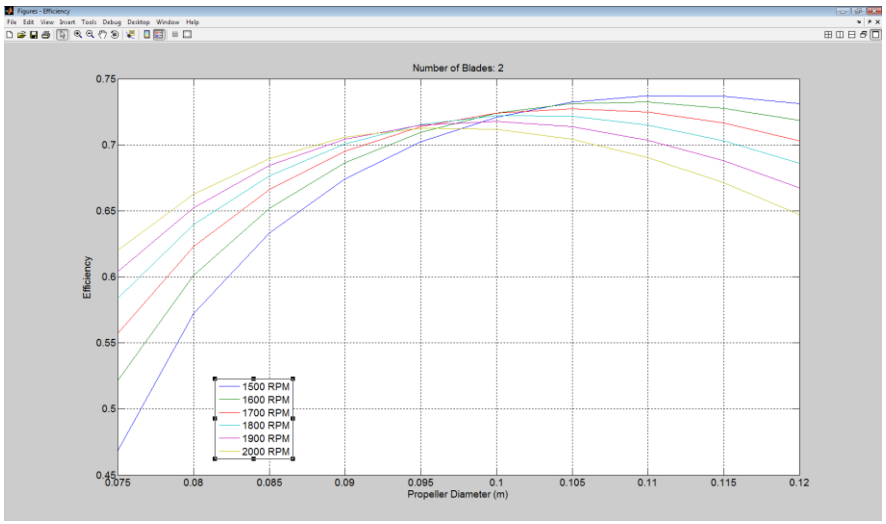


Figure 3-7: Second parametric run

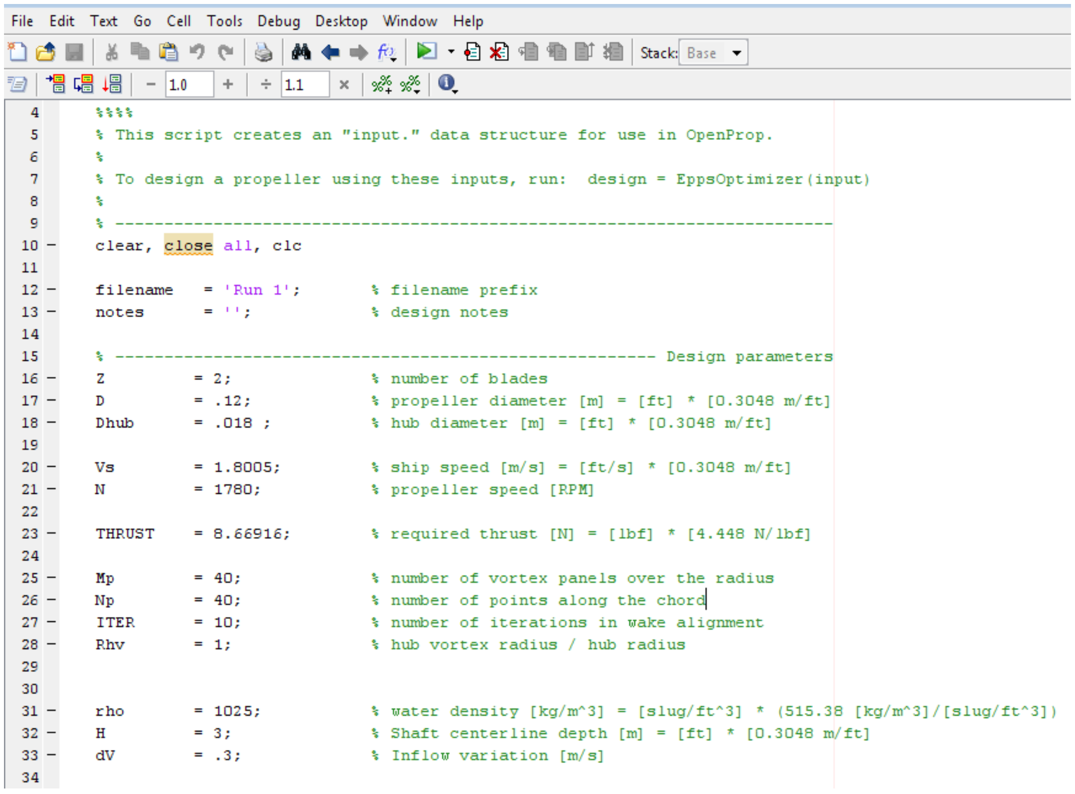
From the results shown in Figure 3-7 the ideal propeller set up using the default propeller geometry is a propeller with a diameter of between 0.11 and 0.12 meters spinning at 1500 RPM.

3.3 OpenProp v2.3 Unique Propeller Design

OpenProp v2.3 does not contain a graphical user interface (GUI), like version 1, and is a text based program. The downloadable program files [2] contain several example inputs that one can use to develop their own scripts. Below is a walkthrough and examples of how the code was modified and used is this design process to design the candidate AUV's propeller. The final propeller was designed to operate at 1850 RPM based on the available NeuMotor 1925-3Y. Appendix A contains the final input m file in its entirety.

3.3.1 Input File, Rough Propeller Design

The input m file, Appendix A, is similar to the GUI of OpenProp v1 and calls for most of the same inputs. Lines 1-33 of the m file or lines 1-36 of Appendix A are the basic inputs like those listed in Chapter 3.2 and are self explanatory. Lines 36-51 of the m file or lines 39-71 of Appendix A are more advanced options and contain options for defining the thickness, chord/diameter distribution, axial inflow velocities, max thickness/chord distribution and rake/diameter distribution. Only the options that were modified are listed above, the other options provided in the input file were left at the default values. The drag coefficient, line 45 of Figure3-9, of 0.0080 is an OpenProp default value that approximates the drag of the various cross sections of the propeller blade. After reviewing the reference material provided on the OpenProp Wiki page [2] it was discovered that the 0.0080 approximation was acceptable for blade lift coefficients of 0.2 to 0.5. As can be seen in Appendix H the final propeller design only has lift coefficient values that fall into the 0.2-0.5 region over half of its length. In order to determine the effect of this high lift coefficient the drag coefficient values from the root to ½ of the radius was doubled and then tripled. This change in drag coefficient resulted in minimal changes in efficiency, ±1%. Thus it was determined that the XCD has minimal effect on the design and was left at its default value. Lines 64-77of the m file contains several other options that were left in the default configuration and were not used in this design process.

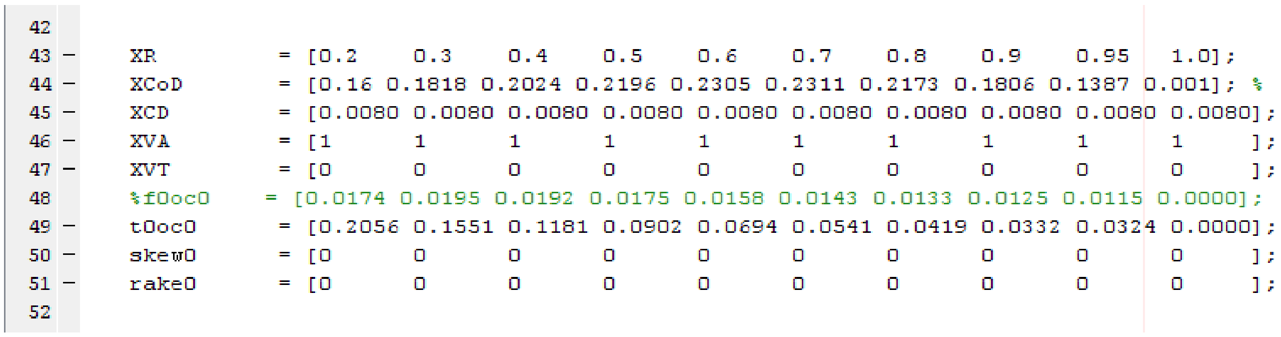


```
 4     %%%%
 5     % This script creates an "input." data structure for use in OpenProp.
 6     %
 7     % To design a propeller using these inputs, run:  design = EppsOptimizer(input)
 8     %
 9     % ---------------------------------------------------------------------------
10 -   clear, close all, clc
11
12 -   filename    = 'Run 1';        % filename prefix
13 -   notes       = '';             % design notes
14
15     % ----------------------------------------------------------- Design parameters
16 -   Z          = 2;               % number of blades
17 -   D          = .12;             % propeller diameter [m] = [ft] * [0.3048 m/ft]
18 -   Dhub       = .018 ;           % hub diameter [m] = [ft] * [0.3048 m/ft]
19
20 -   Vs         = 1.8005;          % ship speed [m/s] = [ft/s] * [0.3048 m/ft]
21 -   N          = 1780;            % propeller speed [RPM]
22
23 -   THRUST     = 8.66916;         % required thrust [N] = [lbf] * [4.448 N/lbf]
24
25 -   Mp         = 40;              % number of vortex panels over the radius
26 -   Np         = 40;              % number of points along the chord
27 -   ITER       = 10;              % number of iterations in wake alignment
28 -   Rhv        = 1;               % hub vortex radius / hub radius
29
30
31 -   rho        = 1025;            % water density [kg/m^3] = [slug/ft^3] * (515.38 [kg/m^3]/[slug/ft^3])
32 -   H          = 3;               % Shaft centerline depth [m] = [ft] * [0.3048 m/ft]
33 -   dV         = .3;              % Inflow variation [m/s]
34
```

Figure 3-8: Example selection of code from Input m file

3.3.2 Input File Modifications, Optimized Propeller Design

In order to achieve the most efficient propeller design possible it is desirable to create a propeller with a high aspect ratio. When OpenProp v2.3 is run with its default chord/diameter or XCoD, the code creates a low aspect ratio propeller. In order to create this high aspect ratio propeller one must modify the XCoD matrix, line 44 of the actual m file or lines 54-55 of Appendix A. OpenProp breaks the propeller into 10 stations along its radius, m file line 43 or lines 52-53 in Appendix A. The propeller shape can be defined by changing the appropriate XCoD values. The inputs used for this example are shown below, Figure 3-9, lines 37-51 were left at the default values, and lines 16-33 of Figure 3-8 are based on the design criteria listed in Chapter 1. A 3-D image, which is created with OpenProp see Chapter 3.3.4, of the resulting propeller is shown in Figure 3-10.



```
42
43 -    XR          = [0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    0.95   1.0];
44 -    XCoD        = [0.16 0.1818 0.2024 0.2196 0.2305 0.2311 0.2173 0.1806 0.1387 0.001]; %
45 -    XCD         = [0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080];
46 -    XVA         = [1      1      1      1      1      1      1      1      1      1     ];
47 -    XVT         = [0      0      0      0      0      0      0      0      0      0     ];
48      %f0oc0     = [0.0174 0.0195 0.0192 0.0175 0.0158 0.0143 0.0133 0.0125 0.0115 0.0000];
49 -    t0oc0       = [0.2056 0.1551 0.1181 0.0902 0.0694 0.0541 0.0419 0.0332 0.0324 0.0000];
50 -    skew0       = [0      0      0      0      0      0      0      0      0      0     ];
51 -    rake0       = [0      0      0      0      0      0      0      0      0      0     ];
52
```

Figure 3-9: Default propeller geometry inputs

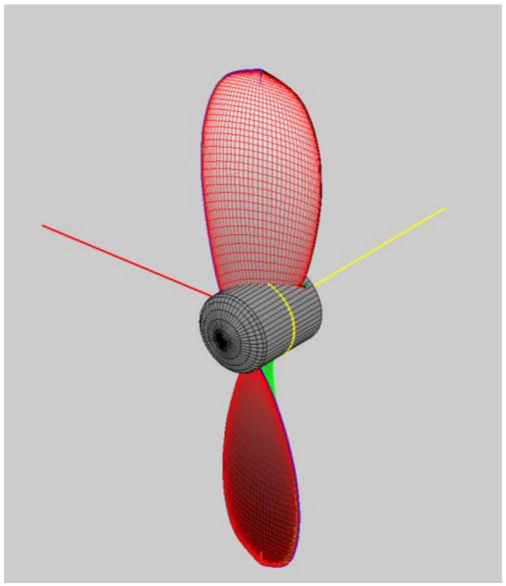


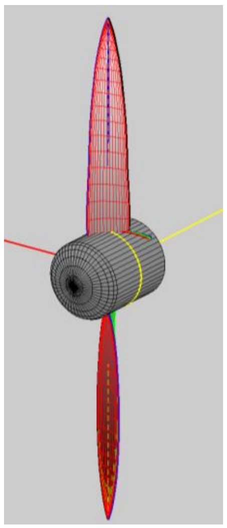
Figure 3-10: Default propeller results 3D blade image

This default propeller result is far from optimized and yields a propeller with an efficiency of 68.9%. This efficiency is optimistic because this is the propeller's open water efficiency but the propeller is in fact behind the hull of the AUV and thus does not actually see the full 2 m/s ship speed. There are two ways to take the hull of the vehicle in to account. One is to use a thrust deduction fraction, , which, along with Equation 3-1, can be used to estimate a new input Ship Velocity. This ship velocity is more characteristic of what the propeller actually sees and will yield a more realistic propeller design and efficiency estimation. In Equation 3-1, V_S is the old Ship Velocity of 2 m/s and V_A is the new Ship Velocity to be input into the input m file.

$$V_A = V_S(1 - w) \tag{3-1}$$

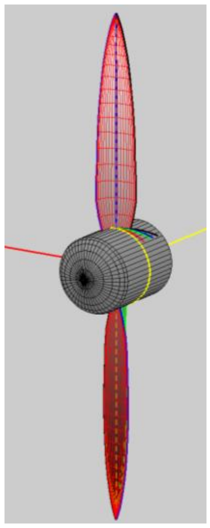
The second and most accurate way is to input the axial inflow variation as found by CFD [3] and keep the ship velocity at 2 m/s. OpenProp then uses this inflow variation to effectively model the varying velocities seen by the propeller caused by the hull of the vehicle. In order to create a more realistic propeller with a less optimistic efficiency estimate this axial inflow variation caused by the wake of the vehicle should be taken into account. The efficiency may drop as much as 5% when the axial inflow variation, XVA, is added into the calculation. For the case above, after the code was run again with the axial inflow variation added to the input file the resulting propeller efficiency was 65.7% and the resulting propeller looked identical to the one shown in Figure 3-10. In order to improve on this efficiency the first geometry input that was modified was the cord to diameter ratio, XCoD. The XCoD basically just changes the blade shape. By changing the XCoD the propeller efficiency can be improved by 10% or more. For this example we will start by including the CFD axial inflow results for the AUV vehicle [3]. Appendix B contains the formula provided by the CFD results and the points generated for use in the OpenProp input file.

In order to fully optimize the design the XCoD needs to be altered until a point of diminishing returns in efficiency is found. This was done by a simple trial and error methodology keeping one key fact in mind; a slim and smooth propeller will yield the most efficient design as long as cavitation is not present. Below are several iterations that looked promising and are just a small selection of geometries that were tested.



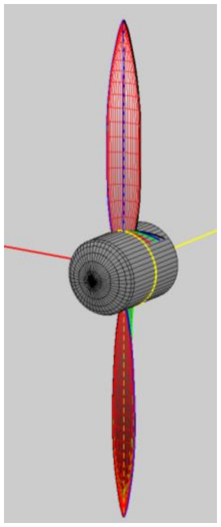
Open Prop Points, x/r	XCoD
0.2	0.0800
0.3	0.0770
0.4	0.0730
0.5	0.07180
0.6	0.0680
0.7	0.0600
0.8	0.0500
0.9	0.0320
0.95	0.0200
1	0.0010
Efficiency	78.73

Figure 3-11: Example 1 chord modification



Open Prop Points, x/r	XCoD
0.2	0.0650
0.3	0.0770
0.4	0.0730
0.5	0.07180
0.6	0.0680
0.7	0.0600
0.8	0.0500
0.9	0.0320
0.95	0.0200
1	0.0010
Efficiency	78.74

Figure 3-12: Example 2 chord modification



Open Prop Points, x/r	XCoD
0.2	0.0530
0.3	0.0620
0.4	0.0650
0.5	0.0660
0.6	0.0670
0.7	0.0610
0.8	0.0500
0.9	0.0310
0.95	0.0200
1	0.0010
Efficiency	78.83

Figure 3-13: Example 3 chord modification

Out of the three propellers shown above example 3, Figure 3-13, had the highest efficiency at 78.83%. This increase in efficiency over the starting 65.7% was achieved just by altering the XCoD of the input file. Every time to code is re-run OpenProp re-optimizes the blade section angles and the blade thickness profile to achieve an optimize blade shape for the design conditions provided.

The above propellers have succeeded in achieving the goal of high efficiency numbers but there is one big problem with them, they are difficult, if not impossible to manufacture as they are. The OpenProp default Thickness profile includes infinitely small, sharp leading and trailing edges. This particular problem is, however, an easy fix. The default Thickness distribution is option '1' in line 38 of the m file and line 42 of Appendix A. Option 1 yields the 2-D cross section with sharp leading and trailing edges shown in Figure 3-14. To set OpenProp v2.3 to design a blade with a leading and trailing edge with a radius just change option '1' to option '4.' Option '4' yields the 2-D cross section shown in Figure 3-15, this change resulted in no change to the predicted efficiency.

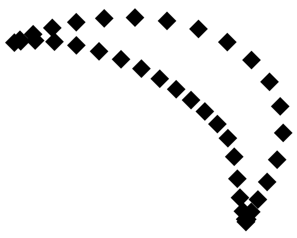


Figure 3-14: OpenProp thickness option 1

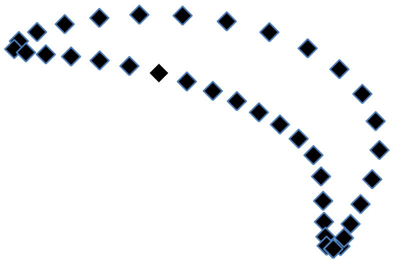


Figure 3-15: OpenProp thickness option 4

The next problem that needs to be addressed is the thickness of the blades themselves. The default thickness profile yields a thin, structurally unsound propeller that may break and/or bend when loaded or handled roughly. At this stage it is convenient to have access to some type of rapid prototyping machine to check the manufacturability and structural integrity of the propeller. The default thickness profile, t0oc0, was modified and the thickness was increased incrementally until an acceptable design was found. A comparison of the default values and the first iteration of the modified propeller thickness profile is shown in Table 3-1. The values shown in Table 3-1 are thickness over chord ratios and are found on lines 66-67 of Appendix A or line 49 of the m file. This modified propeller was built on an Alaris30 rapid prototyping machine [6]. The resulting propeller is shown in Figure 3-16. Notice that the left tip of the propeller is drooping slightly, this is not by design, and is caused by the lack of significant blade thickness.

Table 3-1: Thickness over Chord (t0oc0) profile example 1

XR	Default t0oc0	Modified t0oc0
0.2	0.2056	0.3056
0.3	0.1551	0.2551
0.4	0.1181	0.2181
0.5	0.0902	0.1902
0.6	0.0694	0.1694
0.7	0.0541	0.1541
0.8	0.0419	0.1419
0.9	0.0332	0.1332
0.95	0.0324	0.1324
1.0	0.0000	0.0000

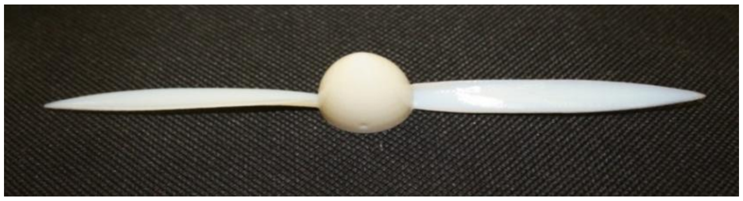


Figure 3-16: Prototype propeller 1

After learning that the propeller thickness and chord needs to be increased in order to achieve a structurally sound design a second prototype propeller was created. This propeller also incorporates a slight rearward rake, defined as the rake over the diameter. A comparison of the default chord distribution, thickness and rake to the second prototype design is shown in Table 3-2. A picture and side profile CAD drawing of the second prototype propeller is provided in Figure 3-17.

Table 3-2: 2nd Prototype propeller t0oc0, XCoD and Rake modifications

XR	Default t0oc0	2nd Modified t0oc0	Default XCoD	2nd Prototype XCoD	Rake
0.2	0.2056	0.3256	0.2056	0.0650	0
0.3	0.1551	0.2651	0.1551	0.0650	0.005
0.4	0.1181	0.2251	0.1181	0.0665	0.01
0.5	0.0902	0.1952	0.0902	0.0660	0.015
0.6	0.0694	0.1784	0.0694	0.0670	0.02
0.7	0.0541	0.1591	0.0541	0.0610	0.025
0.8	0.0419	0.1469	0.0419	0.0500	0.03
0.9	0.0332	0.1382	0.0332	0.0310	0.035
0.95	0.0324	0.1374	0.0324	0.0200	0.037
1.0	0.0000	0.0000	0.0000	0.0010	0.04

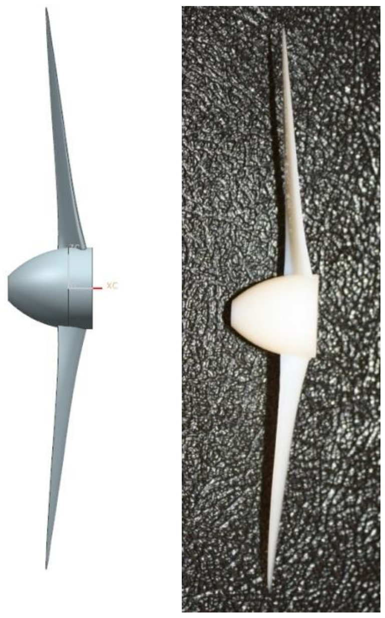


Figure 3-17: 2nd Prototype propeller

The process of adjusting the thickness and chord may require several prototype propellers to be created and scrutinized until a satisfactory propeller is developed. In this case 7 separate propeller thickness profiles were manufactured and scrutinized before a propeller was selected. The thickness, chord distribution and rake of the final propeller are provided below. In order to further increase the durability of the final propeller the blades were further thickened and the chord distribution of the tip was also increased, these inputs are shown in Table 3-3.

Table 3-3: Final Propeller Geometry Inputs

XR	Final t0oc0	Final XCoD	Rake
0.2	0.4606	0.0650	0
0.3	0.4001	0.0650	0.005
0.4	0.3601	0.0655	0.01
0.5	0.3302	0.0660	0.015
0.6	0.3034	0.0670	0.02
0.7	0.2841	0.0650	0.025
0.8	0.2719	0.0600	0.03
0.9	0.2632	0.0450	0.035
0.95	0.2624	0.0330	0.037
1.0	0.0000	0.0010	0.04

3.3.3 Run Script

The OpenProp v2.3 Run Script is an outline of commands that are used to manipulate the OpenProp v2.3 source code. Appendix C contains the Run Script used for the final propeller design in its entirety. The m-file itself is commented in detail and is straightforward in its use. Each command is discussed in detail in Chapter 3.3.4. The Run Script described herein is a modified version of the one provided by the original examples contained in the OpenProp v2.3 downloadable code package.

3.3.4 Results and Off Design Analysis

The following results were taken from one iteration of the propeller design for illustration purposes only, and full text files are provided for the final AUV propeller design in Appendix E-H. This section is meant to be a brief overview of the various plots and text files that are generated by OpenProp v2.3.

After an input file is constructed and the EppsOptimizer is run there are several options in the Run Script that may be used to analyze and obtain visual and statistical representations of the designed propeller. The first option that was used is the ‘Make_Reports(pt)’ command. This command generates three text based reports as well as a MATLAB figure called the Graphical Report that summarizes several propeller statistics. The first generated text file is a summary of the input file and is named prefix_Input.txt, where the prefix is defined by the Input.m file, which can be found on line 11 of the m file or line 13 of Appendix A under the option ‘filename’. The next text file is named prefix_Output.txt and is a summary of the outputs provided by the OpenProp v2.3 code. This file summarizes all of the propeller constants and other important statistics like the efficiency and coefficients of torque, thrust and advance. The last file generated is the prefix_Performance.txt file. This file contains many exacting details on the propeller itself including total inflow velocity, section lift coefficient, and undisturbed flow angle all of which are measured along defined stations along the radius of the propeller. Only the Output file was used in this design process, the other information is interesting to have but not necessary. A single figure called ‘Graphical Report’ is also produced and is shown in Figure 3-18. At the top of the figure is a summary of the key propeller constants and the efficiency. It is important to note that none of the generated figures are automatically saved and must be individually saved and named.

The next option in the Run Script is the 'Geometry_Original(pt)' command, this runs the default geometry script that comes in the OpenProp v2.3 download. Note that the file was renamed from its default name of 'Geometry.' This generates several plots and a single text file. This command also generates a Rhino CAD input file as well as a SolidWorks CAD input file. The text file that is generated is named prefix_Geometry.txt and summarizes the physical geometry of the propeller including statistics like section pitch/diameter and chord-length/diameter ratios. This command also generates two figures, a 2-D and a 3-D Blade Image as shown in Figures 3-19 and 3-20. The 2-D Blade Image figure shows cross section blade shapes along the radius of the propeller blade. The 3-D Blade Image figure is an interactive MATLAB figure where the blade image can be rotated and zoomed so the blade can be scrutinized from all angles. The 3-D image was particularly helpful in determining errors in the blade shape, for example odd twists and lumps in the blade were noticed in several design iterations. These errors can be contributed to XCoD input errors.

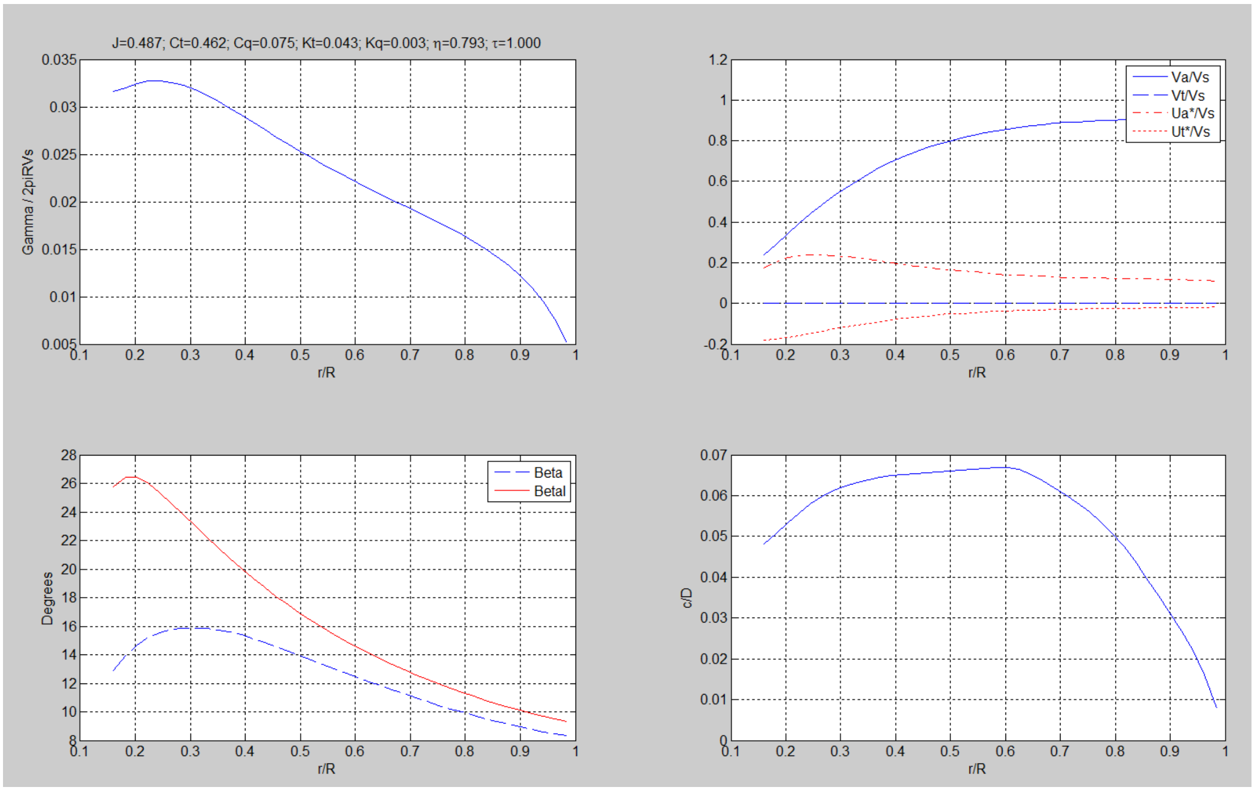


Figure 3-18: Graphical report

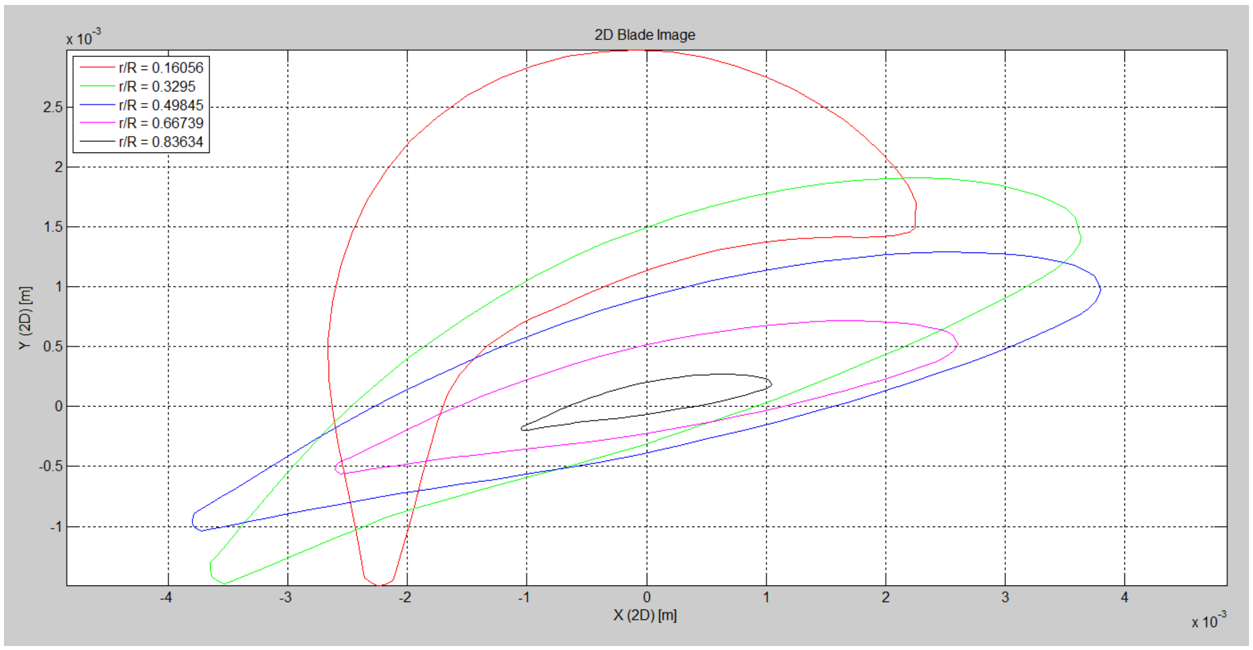


Figure 3-19: 2-D blade image

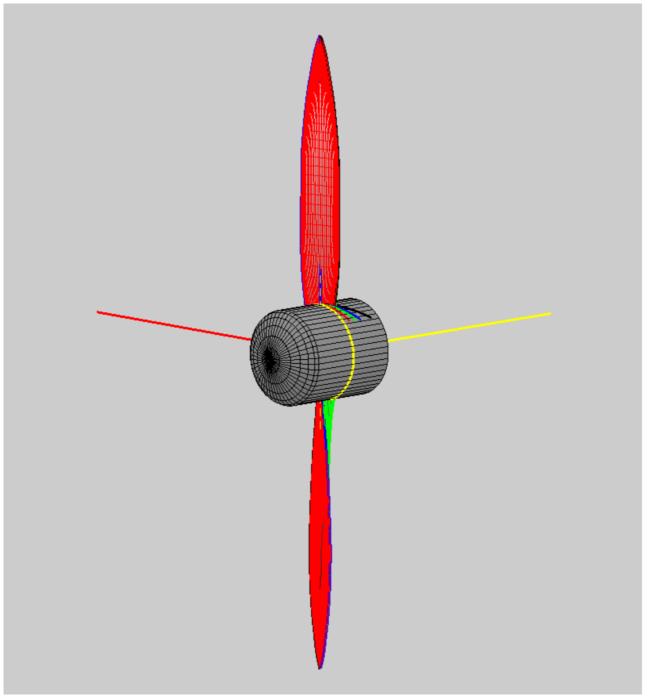


Figure 3-20: 3-D blade image

There is one geometry option available in the downloadable OpenProp v2.3 code package. The original code is described above and the a second geometry command file named Geometry(pt), was created by modifying the original geometry m file. This m file was modified to produce a series of tab delineated text files that can be used with practically any CAD program. Each file contains a series of coordinates that define a single cross sectional profile of one of the propeller blades. All of the files together define a single blade that can then be rotated to generate a propeller with the desired number of blades. The number of points along the chord and the number of stations used along the radius are defined by the Mp and Np values of the Input file, lines 24-25 of the actual m file or lines 26-27 of Appendix A. This modified geometry file is provided in Appendix I.

Once a propeller is developed it is important to also analyze its off design performance in case a mission or adverse conditions requires the vehicle run outside of its intended operating point. OpenProp v2.3 is able to generate the propeller performance curves that show the off design point performance of the propeller design. The 'Analyze off-design states' section in the Run Script, lines 51-55 of the m file or lines 50-54 of Appendix C, can be used to generate the off design performance data. In order to visualize this data the 'Plot Off Design Results' section, lines 58-78 of the m file or lines 57-83 of Appendix C, can be used to generate a plot similar to the one shown in Figure 3-21. In order in increase or decrease the range analyzed change the Js_all option to the desired range, line 52 of the m file or line 51 of Appendix C. An example of the resulting plot is shown below in Figure 3-21. It is important to note that the legend has been modified from its default setting to more clearly describe the results.

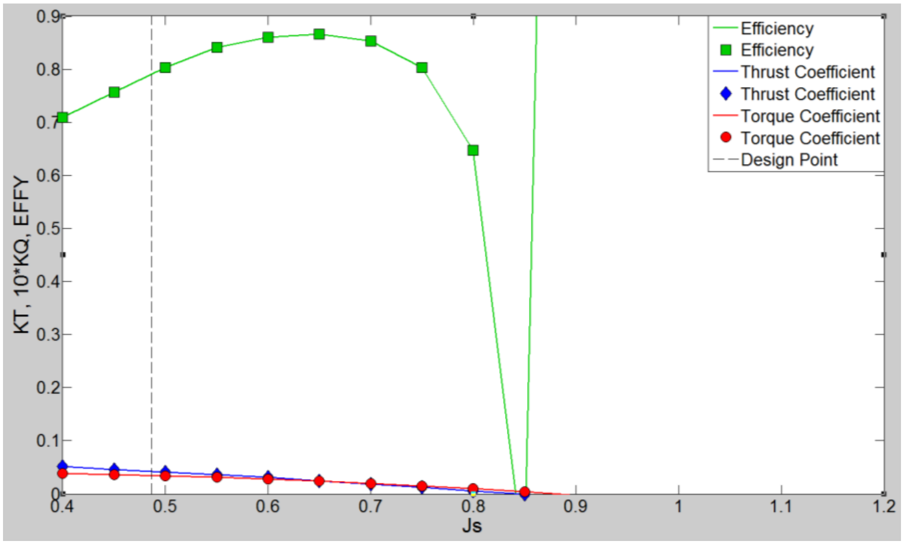


Figure 3-21: Propeller performance curves

The next check that should be done is a cavitation check. 'Cav_CavitationMap(pt)' was run to generate a cavitation map as shown in Figure 3-22. The green marks on the cavitation map, Figure 3-22, denote areas with no cavitation while red marks would denote areas that are predicted to cavitate at the designated operating point. OpenProp runs this calculation at the specified center shaft depth of 3 meters, which is the OpenProp default value and can be found on line 34 of the input m file or line 36 of Appendix A. In this case no cavitation was predicted.

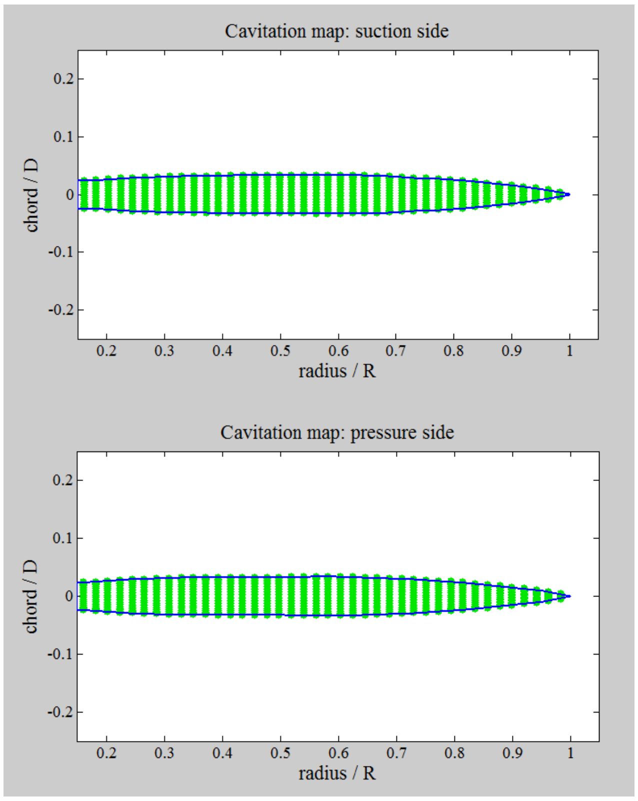


Figure 3-22: Cavitation map

Chapter 4 - Propeller and Motor Optimization

The propeller and motor optimization is an iterative process and both are dependent on each other. The motor and propeller process is also dependent on the rest of the system including, seals, bearings, shafts and any other electronics that may be required in the system. Each component can be changed independently but are all dependent on one another. One must be diligent in their book keeping to make sure nothing is left out of the system so everything will come together as expected and to allow for the highest efficiency value and most accurate possible efficiency estimation possible.

After the motor calculations have been done, as in Table 2-1, a few key points are screened before a custom propeller design is attempted. In order to make the system as efficient as possible it is desirable for the propeller to spin as slowly as possible, so a low RPM is desirable. So if two motors are compared, Motor 1 spins at 5,000 RPM and Motor 2 spins at 2,000 RPM and their efficiencies are similar then Motor 2 would be more desirable because it would lead to a more efficient propeller design and thus a more efficient system overall. The propeller efficiency is not the only factor in this decision and every aspect of the propulsion system must be taken into account. For example, it is likely that Motor 2 will generate more torque than Motor 1, if this torque is not required then Motor 1 may actually lead to a more efficient system because the system would operate closer to Motor 1's maximum efficiency while Motor 2 would be operating off of its designed max efficiency. Also, if the RPM at max efficiency was 10,000 RPM and the shaft seal being utilized in the system is rated to only 5,000 RPM, it is obvious that the proposed motor does not meet the desired specifications. Since the design space is not limited to off the shelf propellers the motor selection process is iterative where motors are analyzed and then propellers are designed for the specific $I_{\eta max}$ RPM result of the specific motor being looked at in order to get an estimate of the total propulsive efficiency. This propeller design is not meant to be an optimized design but is instead just a rough estimation using OpenProp's default settings, see Chapter 3.4.1. Once an estimated propeller is created a total torque required by the system can be estimated. The motor can then be analyzed at this required torque and its actual performance can be determined as per the example in Table 2-2. This off design efficiency is the value that should be used in the final efficiency calculation. Table 4-1 below is a summary of various efficiencies that were used in this design process. Some are rule of thumb values while others are determined through mathematical means like the motor efficiency for example. The hull efficiency depends on the wake fraction and thrust deduction fraction of the vehicle. In the case of this design the wake fraction, *w*, and thrust deduction fraction, *t*, were determined by CFD [3] but were not utilized.

Table 4-1: Summary of efficiencies

Electronic Speed Controller, η_e	0.97 [8]
Hull, η_H	Calculation
Seal, η_{seal}	Experiment
Propeller, η_P	OpenProp
Motor, η_M	Calculation

Equation 4-1 can then be used to determine the hull efficiency, η_H. The CFD work determined that the value of w is 0.3 and the value of t is 0.06 for the candidate AUV. If the wake behind the vehicle is taken into account during the propeller design, as was done in this study, then this efficiency calculation is unnecessary.

$$\eta_{\mathrm{H}} = (1-t)/(1-w) \tag{4-1}$$

Chapter 5 – Component Testing

5.1 Seal and Motor Testing

In order to validate manufacturer claims and boost confidence in the proposed shaft seals for the AUV the seals were thoroughly tested. The motor shaft seals were tested at a wide variety of pressures from running depth, 50psi, to the maximum expected operational depth, 750psi. The original seals provided by BAL [4] were found to have a slow leak at maximum pressure. This was reported to the company and they delivered a handful of redesigned seals for testing. These were put through the same test regime as the previous faulty seals and were found to perform within specifications. Both sets of test results are presented for comparison purposes.

Two different tail designs were considered, a ‘wet’ tail and a ‘dry’ tail. The dry tail is completely sealed off and the components are run in air and there is a pressure differential equal to the water pressure at depth. In the ‘wet’ tail the drive components are isolated from the vehicle and are submersed in oil. A flexible membrane is use to equalize the pressure between the surrounding water and the oil inside of the tail. This effectively leads to no pressure differential across the shaft seals of the propulsion system and maneuvering fins. In the dry configuration the added torque on the shaft by the seal increases as the outside pressure increases. For the wet configuration there is no pressure difference so the added torque is constant across all outside pressures and is equivalent to the running torque at zero depth.

5.1.1 Dry Tail Seal Testing

Four different seals were tested in this experiment. Each seal company specified high tolerance gland dimensions for proper performance of their particular seal. In order to accommodate the four different seals in the same test apparatus multiple interchangeable Seal Housings were manufactured, one for each seal, as shown in Figure 5-1.

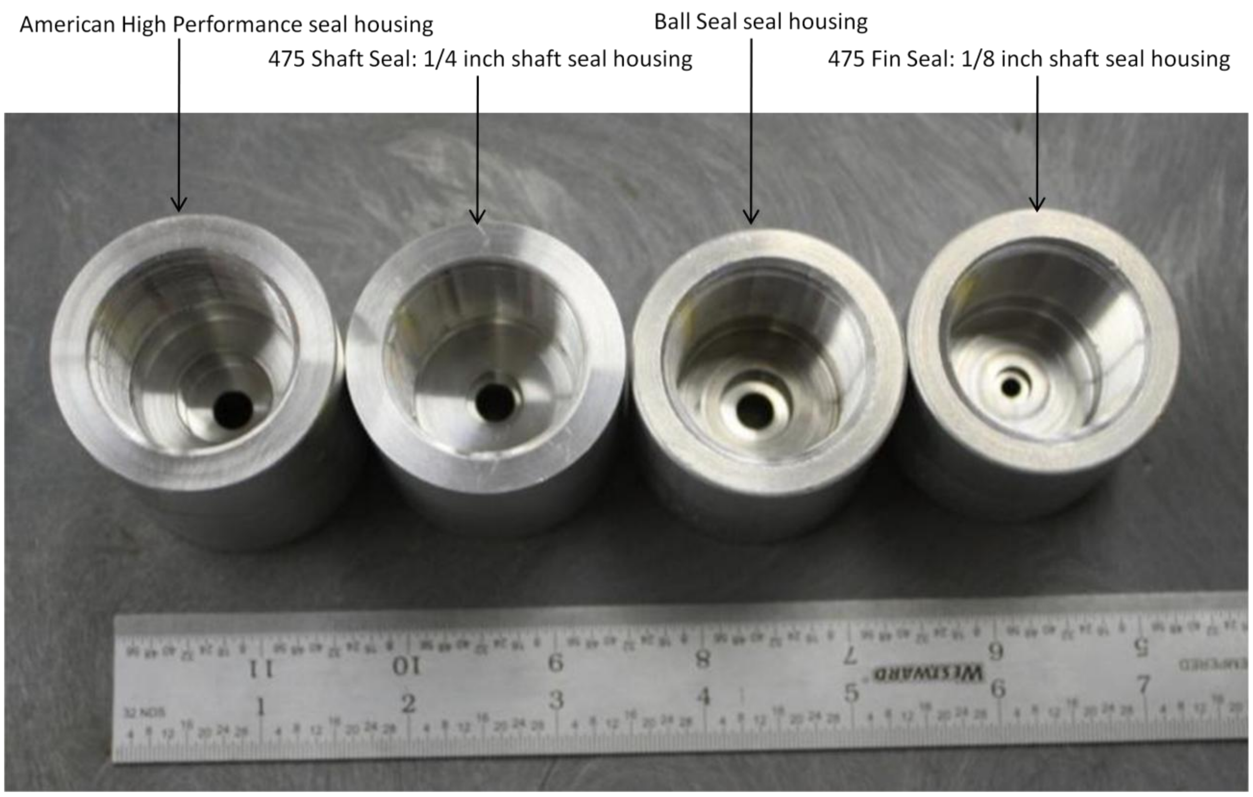

American High Performance seal housing
475 Shaft Seal: 1/4 inch shaft seal housing
Ball Seal seal housing
475 Fin Seal: 1/8 inch shaft seal housing

Figure 5-1: Seal Housings

The American High Performance Seal proved to be too delicate and repeatedly fell apart upon installation. The seal was deemed too fragile for practical use and was removed from testing. The 475 Shaft Seal is an off the shelf shaft seal that is currently in use on Virginia Tech's 475 fleet of AUVs. This seal is a standard shaft seal purchased from McMaster-Carr.com part number 13125K65. The 475 Fin Seal is also from McMaster-Carr.com and has a part number 13125K63; this seal is also the proposed fin seal for the candidate AUV. The BAL Seal is a custom seal manufactured to the candidate AUV operating conditions by a company called BAL Seal Engineering Inc [4]. Two different prototype seals were developed by BAL.

A pressure chamber, with gauge and valve system was then manufactured as shown in Figure 5-2. The pressure chamber consists of a schedule 40 standard wall aluminum threaded pipe with a diameter of 1 inch and a length of 6 inches. The end that connects the valve and gauge assembly is a high-pressure aluminum female reducing coupling that reduces the 1 inch NPT to a ½ inch NPT. A reducing ½ inch to ¼ inch high pressure brass fitting is attached to the aluminum female reducing coupling. A high pressure brass tee fitting was then used to attach the gauge and two way high pressure purge valve. A length of abrasion resistant PTFE hose rated to 3000psi was used to connect the pressure chamber to an air cylinder and regulator that was used to set the pressure of the system during testing.

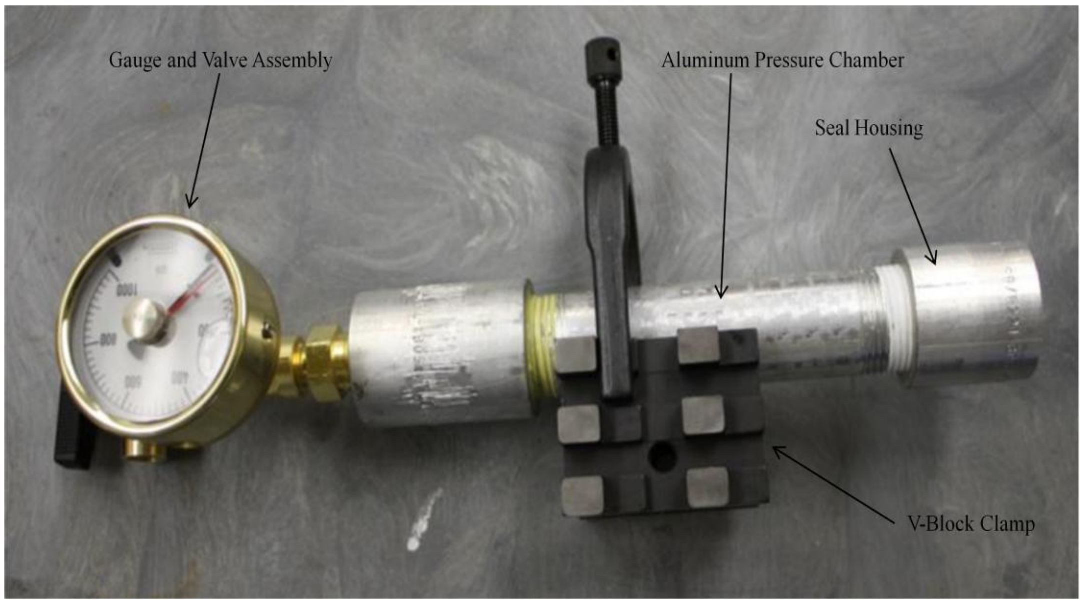


Figure 5-2: Pressure Chamber

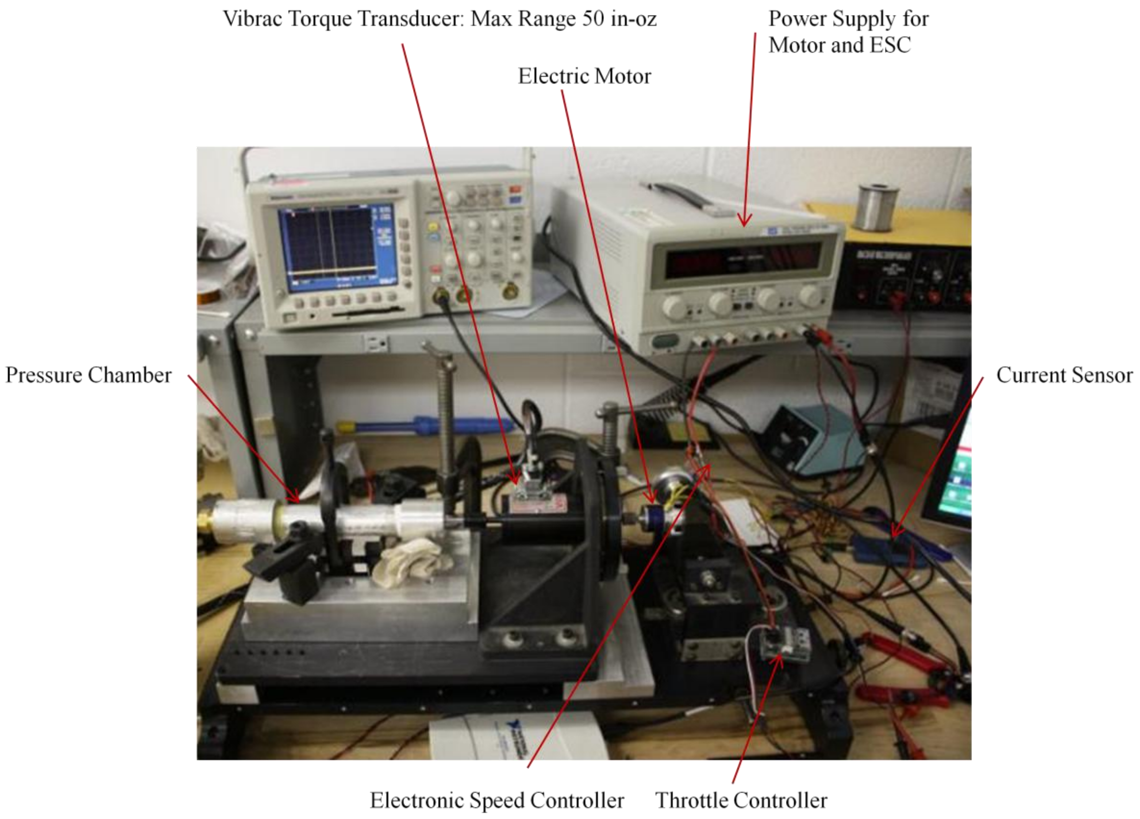

Vibrac Torque Transducer: Max Range 50 in-oz
Power Supply for Motor and ESC
Electric Motor
Pressure Chamber
Current Sensor
Electronic Speed Controller
Throttle Controller

Figure 5-3: Seal torque test set up

Figure 5-3 shows the completed experimental set up. The pressure chamber was mounted to a V-Block that was then mounted to a custom aluminum base plate via C-clamps that allowed precise alignment of the shaft protruding from the pressure chamber and the shaft of the Vibrac 50 in-oz torque transducer. The Series 1 Vibrac torque transducer is rated to a maximum torque of 50 in-oz and is capable of surviving a 100% overload without failure. The accuracy of Vibrac transducer is +/-1% of the span and is capable of handling up to 10,000 RPM [9]. A power supply was initially used to power the drive motor but it was discovered that it could not provide adequate amperage to the motor at the higher test pressures. A Lithium Polymer battery was used instead of the power supply for the duration of the experiment. The motor used was a brushless Hyperion Z-2213-24 exterior-rotor.

The test results for the first prototype BAL seal are shown in the upper plot in Figure 5-4. The breakaway torque for the existing 475 fin seal is also included in Figure 5-4. All tests were run at 2000 RPM. Breakaway torque is the impulse required to start the shaft turning and the running torque is the measured torque after the system has reached a steady state at 2000 RPM.

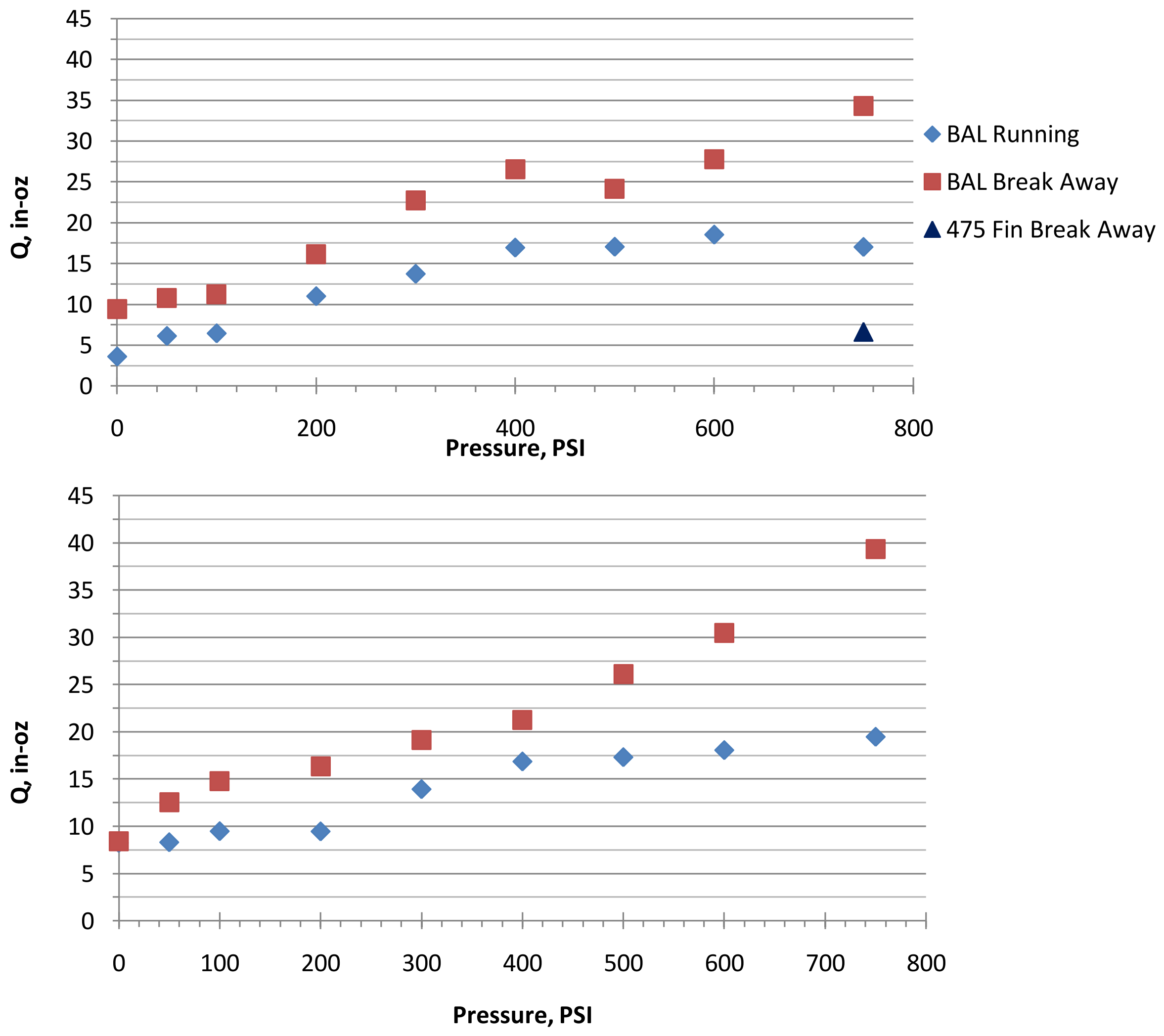


Figure 5-4: Prototype BAL seal torque test results, prototype 1 top, prototype 2 bottom

The second prototype BAL seal was tested using the same experimental set up as the first but this seal was found to add more torque to the system than the first prototype seal. The original motor used in the testing setup was unable to turn the system at the higher test pressures. Modifications to the test stand were manufactured in order to mount a larger motor. A NeuMotor 1925-1Y was used to complete the testing. The torque results are shown in the bottom plot in Figure 5-4. The second prototype BAL seal running torque was found to be 8.3 in-oz, at the most, at the expected average operating depth of 10 meters. In order to provide a factor of safety in the system the 50psi results were used in all calculations.

5.1.2 Oil-Filled Tail Testing

The next experimental test that was undertaken was to determine the possible losses caused by an oil filled tail. Since an oil-filled tail equalizes with the outside pressure there is effectively no pressure difference across the shaft seal. Thus no pressure chamber was required and the test was run at atmospheric pressure. A simple PVC pipe chamber was manufactured to house the oil and motor components. Power and communication wires are allowed to pass out of the chamber through two vertical protruding pieces. The chamber and its components are shown in Figure 5-5 through 5-7. A NeuMotor 1925-1Y interior-rotor and a Hyperion Z-2213-24 exterior-rotor motors were used in the test. The results of the testing are summarized in Table 5-1. The exterior-rotor style motor was shown to have significant losses caused by the oil and significant stirring of the oil was observed while. The losses of the interior-rotor motor were much less and no stirring of the oil was observed. The oil used was Carnation Light Mineral Oil, the product number is 1067-6 [10]. This oil is the exact oil that would be used in an actual oil filled tail configuration. The 1925-1Y is also dimensional identical to the proposed 1925-3Y.

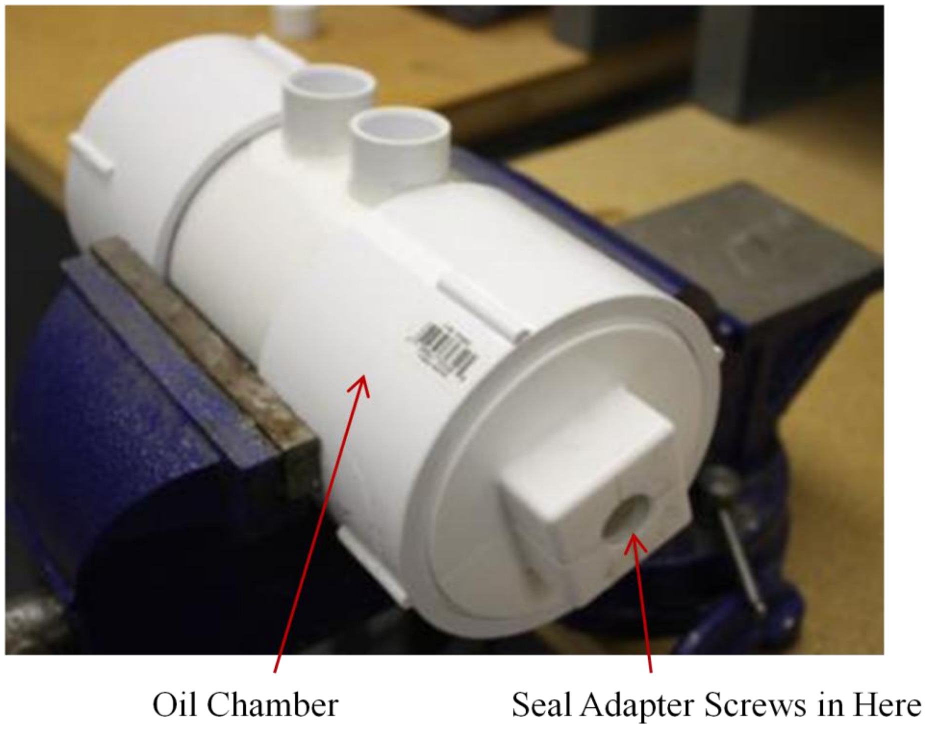


Figure 5-5: Oil test chamber

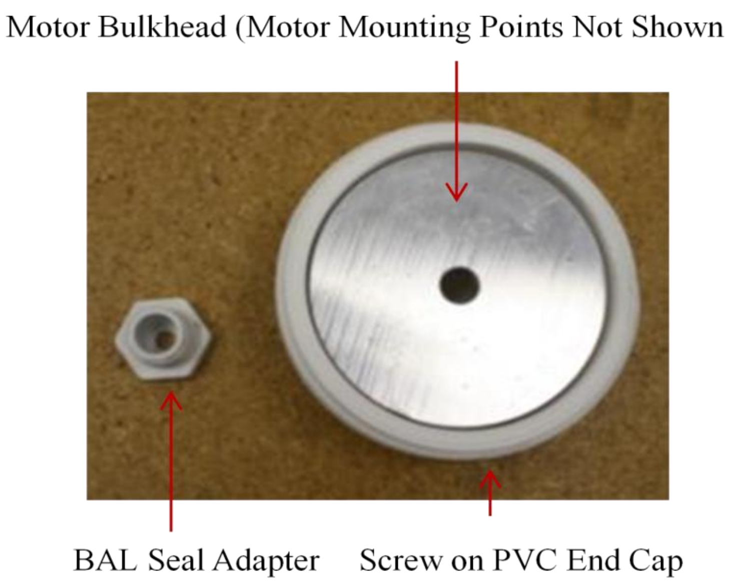


Figure 5-6: Oil chamber end cap, motor bulkhead and seal adapter

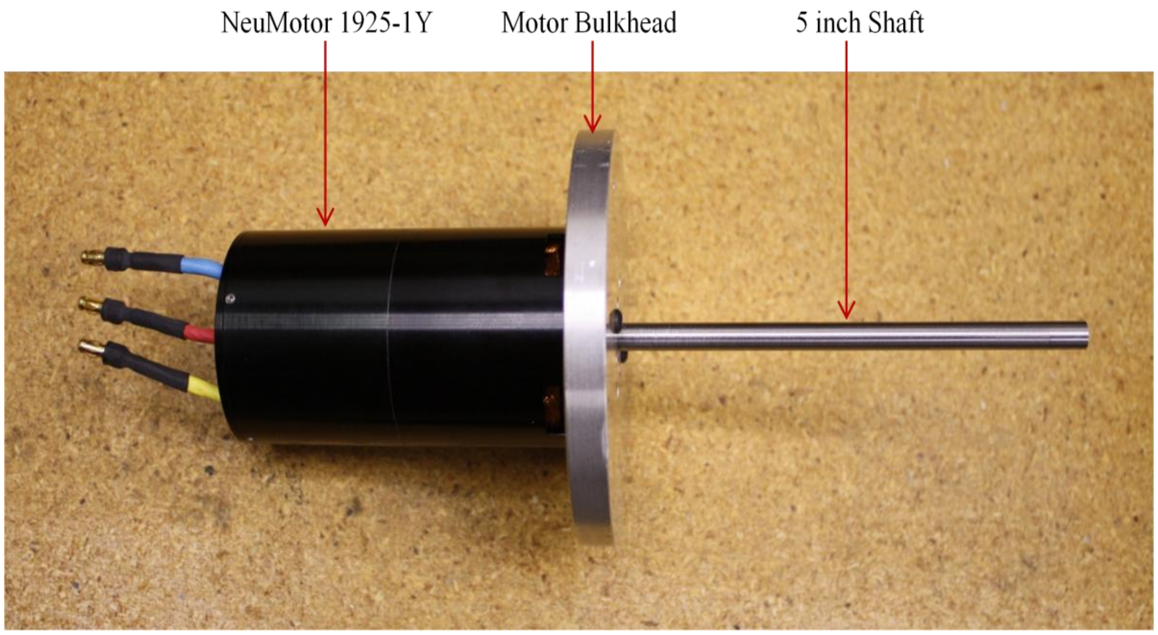


Figure 5-7: NeuMotor 1925-1Y mounted to motor bulkhead

Table 5-1: Oil filled test results

Hyperion HP-Z2213-24, Out Runner

No Oil			Oil Filled		
Run	Voltage	Amp Drawn	Run	Voltage	Amp Drawn
1	14	1.13	1	14	2.41
2	14	1.11	2	14	2.39
3	14	1.16	3	14	2.43
4	14	1.14	4	14	2.36
5	14	1.18	5	14	2.38
	Average	1.144		Average	2.394

NeuMotor 1925-1Y, In Runner

No Oil			Oil Filled		
Run	Voltage	Amp Drawn	Run	Voltage	Amp Drawn
1	14	0.77	1	14	1.10
2	14	0.77	2	14	1.10
3	14	0.78	3	14	1.09
4	14	0.78	4	14	1.10
5	14	0.77	5	14	1.09
	Average	0.774		Average	1.096

Hyperion Out Runner Difference = 1.25 amp
NeuMotor In Runner Difference = 0.322 amp

Chapter 6 Final Propulsion Design

6.1 Final Motor Selection and Propeller Design for Dry Tail

At the time of writing two motors have been selected and ordered from the company NeuMotor [7], the 1925-3Y (now called the 1924-3Y by NeuMotor) and the 1521-10.5Y. The 1521 series motor is not listed on the NeuMotor website and was discovered after requesting a chart that outlined what custom motors they could wind using the 1500 series chassis. The chart provided by NeuMotor is provided in Appendix D. These two motors were analyzed as per Chapter 2, the results of this analysis are summarized in Tables 6-1 and 6-2. Both motor selections recommended are capable of turning the system at all pressure values measured and shown in Chapter 5 Figure 5-5. The torque required used for these off efficiency calculations was 21.43 in-oz. The torque required was found by adding the torque imparted to the shaft by the seal, 8.3 in-oz, to the torque required to turn the final propeller design, 13.13 in-oz. We are assuming the 8.3 in-oz added by the seal will be correct even though the experiment was run at 2000 RPM and the actual system is turning at around 1850 RPM.

Table 6-1: Summary of off motor characteristics

Motor	Rm	Io	Kv	Max Eff	Q Max Eff	RPM Max Eff
1521-10.5Y	0.689	0.19	149	81.595	16.11	1884.29
1925-3Y	0.18	0.3	136	87.965	45.05	1785.75

Table 6-2: Motor off design characteristics

Motor	RPM Off Eff	Motor Off Eff
1521-10.5Y	1823.99	80.93
1925-3Y	1843.87	85.01

The proposed motors have nearly identical performance characteristics for the proposed application. The 1925 is about 4% more efficient than the 1521 but the 1521 is a smaller motor and would remove about 5 oz off of the total system weight. In order to select a final motor both motors were ordered for in house testing and evaluation, which is ongoing. Conveniently both motors have identical mounting options.

The final AUV prototype propeller is shown in Figure 6-1. The final propeller design was made at a diameter of 0.12 meters and included an increased tip chord and thickness to help mitigate tip flexion when the propeller is under load. In order to validate the diameter chosen the propeller was redesigned using a diameter of 0.13, 0.11, and 0.10 meters. All alternative diameters yielded a propeller with efficiency lower than that of the final propeller designed with a diameter of 0.12 meters. A 3-D drawing of the final propeller is provided in Figure 6-3. The final propellers thickness, chord distribution and rake inputs are shown in Table 6-3, the input file is shown in Appendix A in its entirety.

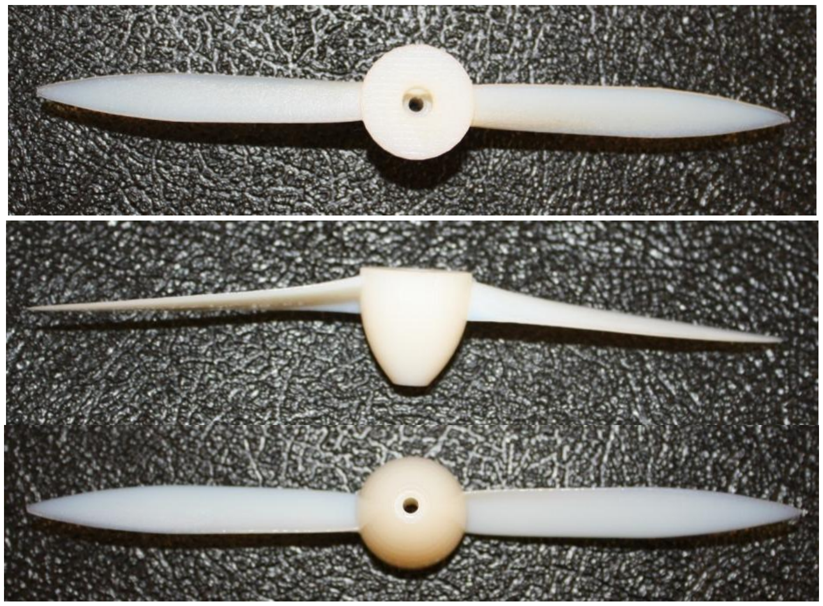


Figure 6-1: Final Propeller Prototype

Table 6-3: Final Propeller Prototype Geometry

XR	Final t0oc0	Final XCoD	rake0
0.2	0.4606	0.0650	0
0.3	0.4001	0.0650	0.005
0.4	0.3601	0.0655	0.010
0.5	0.3302	0.0660	0.015
0.6	0.3034	0.0670	0.050
0.7	0.2841	0.0650	0.025
0.8	0.2719	0.0600	0.030
0.9	0.2632	0.0450	0.035
0.95	0.2624	0.0330	0.037
1.0	0.0000	0.0010	0.040

The propeller performance curves and other OpenProp v2.3 outputs, as described in Section 3.3.4, are shown below in Figures 6-2 through 6-7. The various output text files are provided, in their entirety, in Appendix E through H.

From the tests described in Chapter 5 the running torque of the seal at the candidate AUV's operating pressure of 50psi was found to be 8.3 in-oz at 2000 RPM. Using this result the power absorbed by the seal was determined to be 11.31 watts for the 1925-3Y and 11.19 watts for the 1521-10.5Y using Equation 6-1 where n is the rotations per second of the system and Q is the torque.

$$P = Q2\pi n \tag{6-1}$$

The power into the propeller can also be found by using Equation 6-1. We know from OpenProp that the torque required to turn the propeller is 13.13 in-oz and that the design point was 1850

RPM. Using these quantities the power into the propeller was found to be 17.97 watts. We also know that, at this design point, the 1925-3Y requires 34.38 watts and the 1521-10.5 requires 35.73 watts to turn the system. Assuming that the controller has an efficiency of 97% [8] the total power into the system is 35.44 watts for the 1925-3Y and 36.83 watts for the 1521-10.5Y.

We know that the vehicle has a drag of 8.11 Newtons at 2 meters per second. Using Equation 6-2, where T is the drag in this case and V is the velocity of the vehicle, it was determined that the effective power is 16.22 watts for the vehicle to travel at 2 meters per second.

$$P = T * V \tag{6-2}$$

By dividing this effective power by the total power into the system the total efficiency of the system can be calculated to be 45.76% for the 1925-3Y and 44.03% for the 1521-10.5Y.

A summary of the final predicted propulsive efficiency and other statistics is provided in Table 6-4 and 6-5. Table 6-6 is a summary of the data points used to create the K_T and K_Q plots in Figures 6-4 and 6-5. Figure 6-6 shows no cavitation to be predicted for the propeller at this design point.

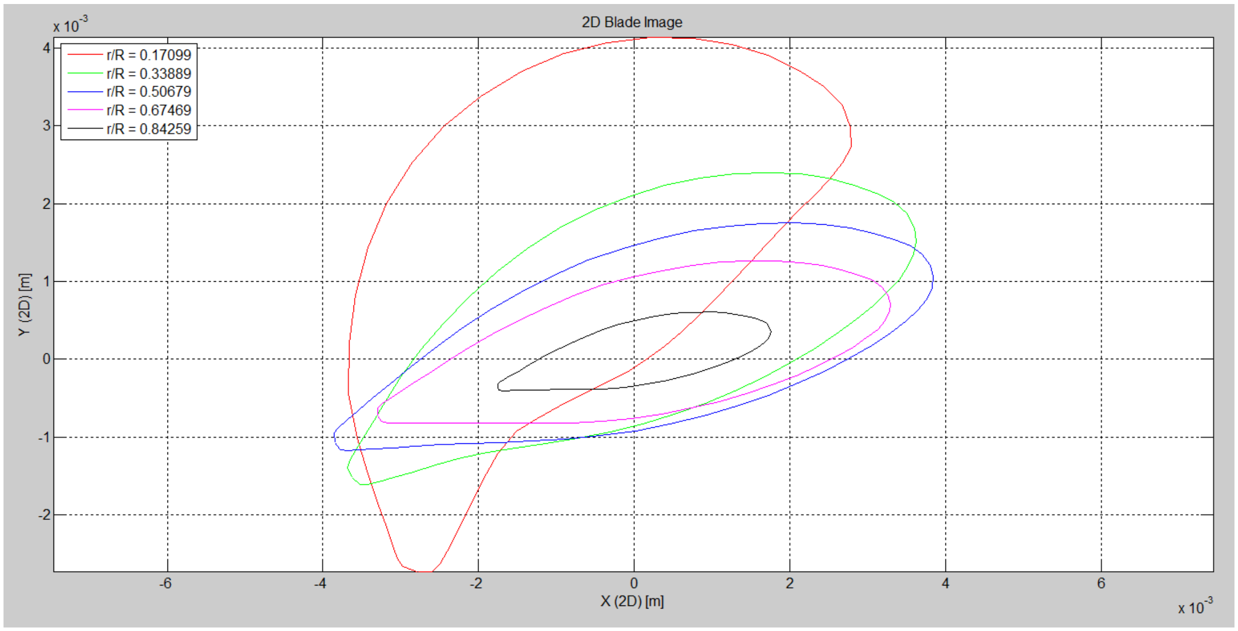


Figure 6-2: Final AUV 2-D propeller image

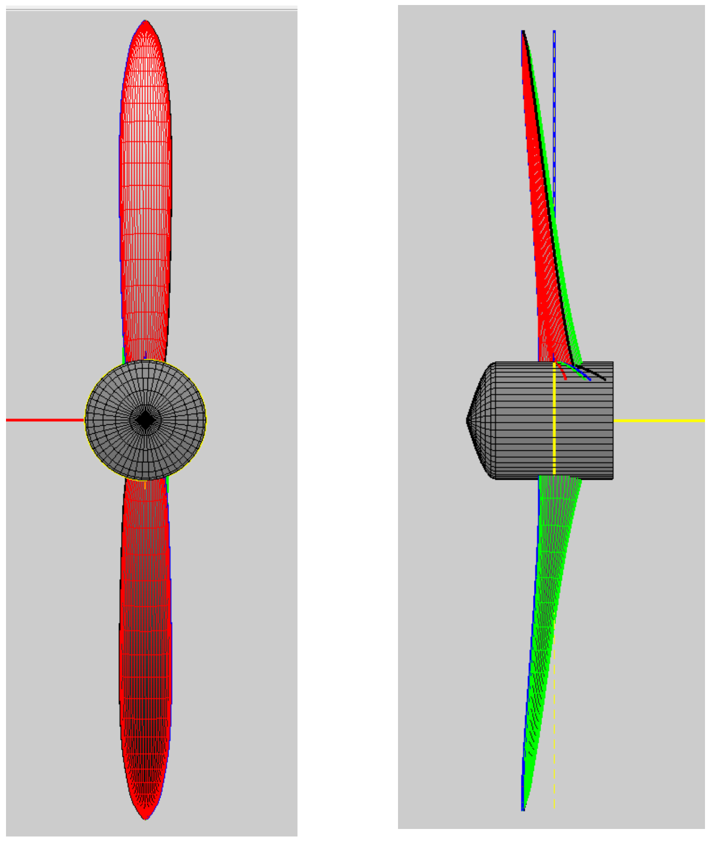


Figure 6-3: Final AUV 3-D propeller image

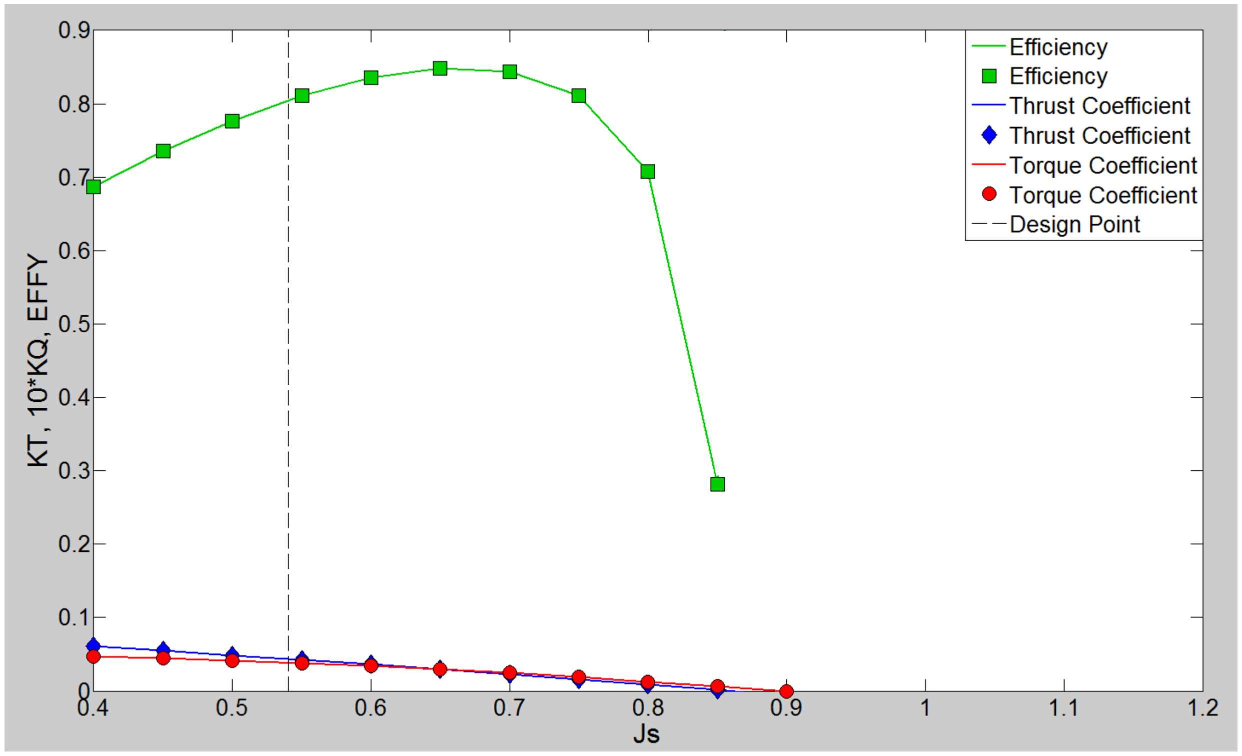


Figure 6-4: Final AUV propeller performance curves

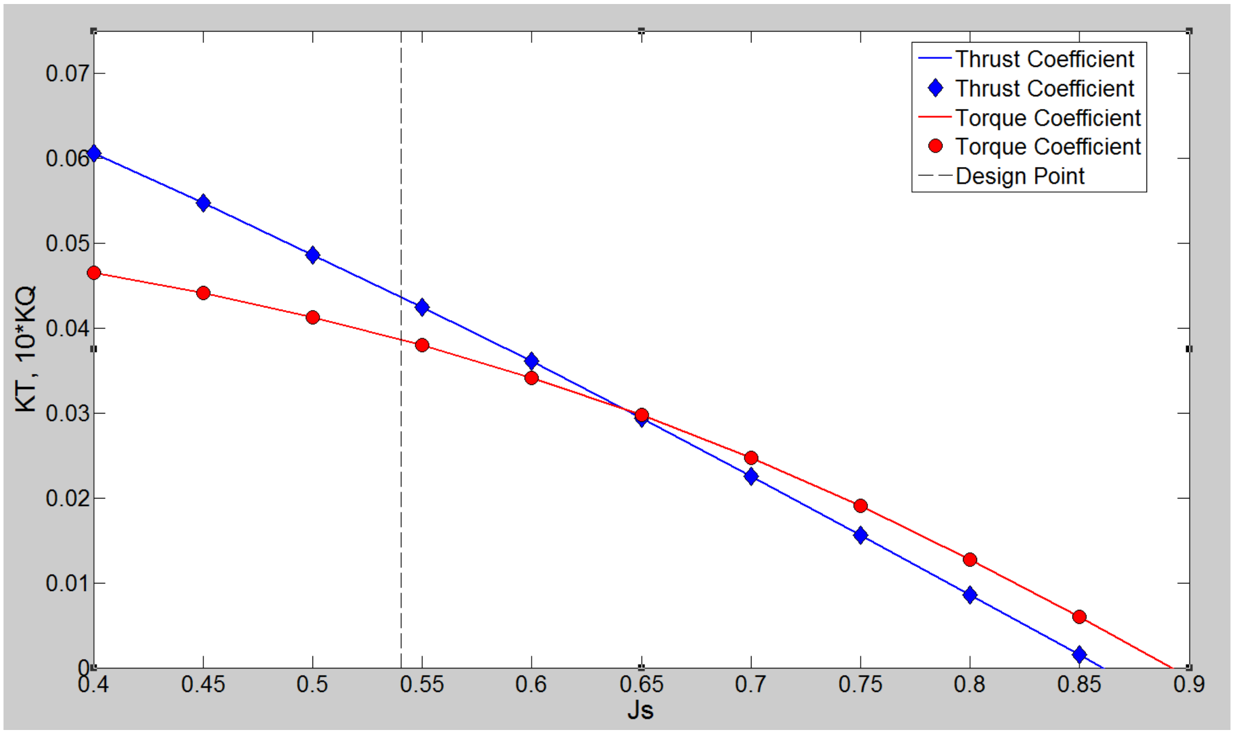


Figure 6-5: Kt and Kq performance curves

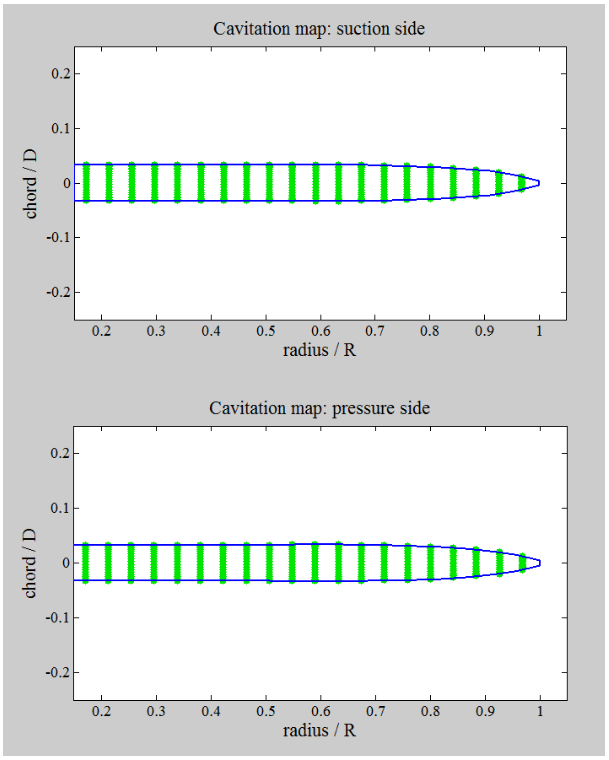


Figure 6-6: Final AUV propeller cavitation map

Table 6-4: Motor power consumption summary

Motor	RPM Max Eff	RPM Off Eff	Motor Off Eff	Watts at 14V	Current Drawn, amp
1521-10.5Y	1884.2	1823.99	80.93	35.73	2.55
1925-3Y	1785.75	1843.87	85.01	34.38	2.45

Table 6-5: Summary of propulsion system efficiencies

Component	Efficiency
Electronic Speed Controller, η_e	0.97 [8]
Propeller, η_P	0.7993
Motor, η_M 1925-3Y	0. 8501
Motor, η_M 1521-10.5Y	0.8093

Total Efficiency with 1925-3Y	0.4576
Total Efficiency with 1521-10.5Y	0.4403

Table 6-6: Summary of K_T and $10*K_Q$ data points

X Value	K_T	$10*K_Q$
0.10	0.06058	0.04650
0.45	0.05470	0.04413
0.50	0.04857	0.04124
0.55	0.04244	0.03796
0.60	0.03610	0.03417
0.65	0.02940	0.02973
0.70	0.02255	0.02469
0.75	0.01559	0.01902
0.80	0.008554	0.01275
0.85	0.001502	0.005974

Chapter 7 Conclusions

A mathematical motor characterization process was proposed to characterize a brushless electric motor based on the three basic motor constants, K_v , R_m , and I_o provided by the manufacturer of the motors. A wide variety of electric motors were characterized using the proposed method and the field was narrowed down by using results from the motor characterization process along with the RPM range obtained from the parametric propeller analysis. Prototype propellers were then created to yield the highest efficiency possible based on the RPM range obtainable by the candidate motors and the RPM and diameter range predicted by the parametric propeller analysis. Once a unique propeller blade shape was finalized the manufacturability of the blade was taken into account. The chord distribution and thickness was then altered and prototype propellers were manufactured on an Alaris30 3D printer. This propeller prototyping process was repeated until a structurally acceptable propeller blade was created. A variety of seals were tested and the results, combined with the torque required to turn the propeller obtained from OpenProp, were used to determine the torque required to turn the candidate AUV's propulsion system. This result was then used to determine the off design motor characteristics of the candidate motors and a final motor selection was made. A final efficiency prediction was then made based on the predicted total power consumed by the system and the predicted effective power required.

The proposed design process is applicable to a variety of surface and under water propulsion design applications. Future work could be done to validate the process described by designing, building and testing a system and then comparing the predicted efficiency results with measured efficiency results from the prototype propulsion system.

Works Cited

[1] Schultz, J., “Anutonomous Underwater Vehicle (AUV) Propulsion System Analysis and Optimization,” MS Thesis (Online), Department of Aerospace and Ocean Engineering, Virginia Tech, 2009. And can be found at: http://scholar.lib.vt.edu/theses/available/etd-05252009230302/ unrestricted/Schultz_James_Thesis.pdf

[2] OpenProp (Online)
http://openprop.mit.edu/~openprop/wiki/index.php?title=Main_Page

[3] R. Coe, Personal Communication, March 23, 2010

[4] BAL Seal Engineering Inc
http://www.balseal.com/home

[5] J.R. Hendershot Jr and Miller, *Design of Brushless Permanent-Magnet Motors*. Magna Physics Publishing and Clarendon Press, Oxford New York, 1994.

[6] Alaris30 Rapid Prototype Machine (Online)
http://www.objet.com/3D-Printer/Alaris30/

[7] NeuMotor (Online)
http://www.neumotors.com/Site/Welcome.html

[8] Scorpion Calc v3.37 (Online)
http://www.scorpionsystem.com/downloads/

[9] Vibrac Torque Transducers (Online)
http://www.vibrac.com/products/transducers.aspx

[10] Ruger Chemical Product Catalog (Online)
https://www.echempax.com/ruger/dsp/pub/products/ProductCatalogSearch.dsp?pageType=Product&category=EE4X&type=&ctl_st=description_ASC

[11] Virginia Tech 475 AUV (Online)
http://www.ascl.ece.vt.edu/475AUV.html

Appendix A – Input Matlab m File for Final Propeller Design

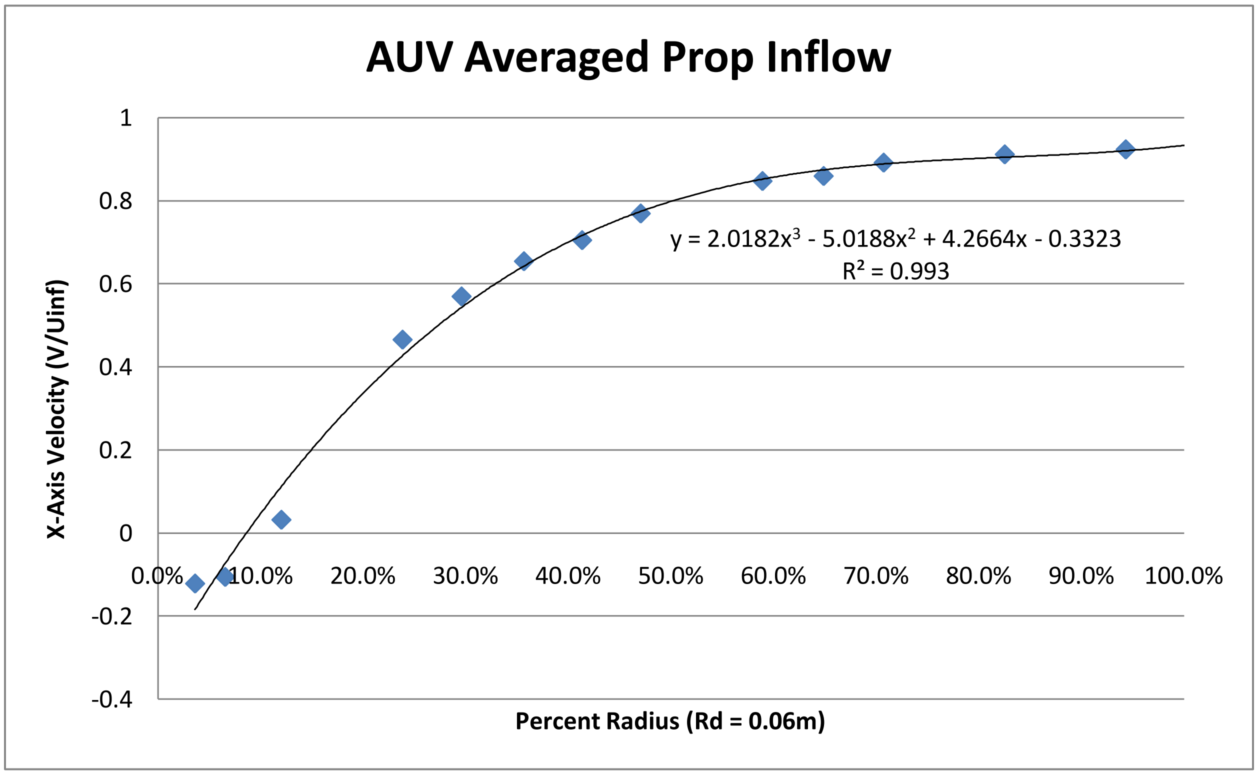
Due to text wrapping the line numbers are slightly off when compared to the m-file.

```
1   % ----------------------------------------------------------- Example_input.m
2   % Created: 5/28/09, Brenden Epps, bepps@mit.edu
3   % Modified: 2010, Richard Duelley, nifty@vt.edu, additional comments and
4   % organization
5   % This script creates an "input." data structure for use in OpenProp.
6   %
7   % To design a propeller using these inputs, run:  design =
8   EppsOptimizer(input)
9   %
10  % ---------------------------------------------------------------------------
11  clear, close all, clc
12
13  filename   = 'Run 1';        % filename prefix
14  notes      = '';             % design notes
15
16  % ------------------------------------------------------- Design parameters
17  Z          = 2;              % number of blades
18  D          = .12;            % propeller diameter [m] = [ft] * [0.3048 m/ft]
19  Dhub       = .018 ;          % hub diameter [m] = [ft] * [0.3048 m/ft]
20
21  Vs         = 2.0;            % ship speed [m/s] = [ft/s] * [0.3048 m/ft]
22  N          = 1850;           % propeller speed [RPM]
23
24  THRUST     = 8.66916;        % required thrust [N] = [lbf] * [4.448 N/lbf]
25
26  Mp         = 20;             % number of vortex panels over the radius
27  Np         = 28;             % number of points along the chord
28  ITER       = 10;             % number of iterations in wake alignment
29  Rhv        = 1;              % hub vortex radius / hub radius
30
31
32  rho        = 1025;           % water density [kg/m^3] = [slug/ft^3] * (515.38
33                                 [kg/m^3]/[slug/ft^3])
34  H          = 3;              % Shaft centerline depth [m] = [ft] * [0.3048
35                                 m/ft]
36  dV         = .3;             % Inflow variation [m/s]
37
38
39  % -------------------------------------------- Blade 2D section properties
40  Meanline   = 'NACA a=0.8';            % Meanline type  (1 == NACA a=0.8, 2 ==
41                                             parabolic)
42  Thickness  = 4;            % Number 4 gives a nice rounded leading and
43                                 trailing edge(1 == NACA 65A010, 2 ==
44                                 elliptical, 3 == parabolic, 4 == NACA 65A010
45                                 (modified))
46  alphaI     = 1.54;         % [deg] ideal angle of attack  (should match with
47                                 Meanline type)
48  CLI        = 1.0;         % [ ],  ideal lift coefficient (should match with
49                                 Meanline type)
50
```

```

XR          = [0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    0.95
1.0];    % radius / propeller radius
XCoD        = [0.065 0.065 0.0655 0.0660 0.0670 0.065 0.060 0.045 0.033
0.0010]; % chord / diameter
XCD         = [0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080
0.0080]; % section drag coefficient
XVA         = [0.336624      0.550666      0.70672      0.79875      0.857008
0.887554      0.902496      0.913942      0.922138      0.934     ]; % axial
inflow velocity / ship velocity, See AUVPropInFlowData Excel Sheet: By
Richard Duelley
XVT         = [0      0      0      0      0      0      0      0      0
0     ]; % tangential inflow velocity / ship velocity
%f0oc0    = [0.0174 0.0195 0.0192 0.0175 0.0158 0.0143 0.0133 0.0125 0.0115
0.0000]; % max section camber    / chord
t0oc0       = [0.4606 0.4001 0.3601 0.3302 0.3034 0.2841 0.2719 0.2632 0.2624
0.0000]; % max section thickness / chord
skew0       = [0      0      0      0      0      0      0      0      0
0     ]; % skew [deg]
rake0       = [0      0.005      0.01      0.015      0.02      0.025
0.03      0.035      0.037      0.04     ]; % rake / diameter


% ------------------------------------------------------------------- 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
     Wake_flag  = 0;      % 0 == Horseshoe(...,Wrench(...)), 1 ==
                             Wake_Horseshoe(...)
     Plot_flag  = 0;      % 0 == do not display plots, 1 == display plots
    Chord_flag  = 0;      % 0 == do not optimize chord lengths, 1 == optimize
                             chord lengths
Optimizer_flag  = 2;      % 1 == Lerbs optimizer, 2 == Epps optimizer
 Lagrange_flag  = 0;      % 0 == do not fix Lagrange multiplier, 1 == fix
                             Lagrange multiplier

Make2Dplot_flag = 1; % 0 == do not make a 2D plot of the results, 1 == make
                             plot
Make3Dplot_flag = 1; % 0 == do not make a 3D plot of the results, 1 == make
                             plot
Make_Rhino_flag = 1; % 0 == do not make Rhino files, 1 == make Rhino files

% ---------------------------------------------- Compute derived quantities
n       = N/60;                           % revolutions per second [rps]
R       = D/2;                            % propeller radius [m]
Rhub    = Dhub/2;                         % hub radius [m]
Rhub_oR = Rhub/R;
Js      = Vs/(n*D);                       % advance coefficient
L       = pi/Js;                          % tip-speed ratio
CTDES   = THRUST/(0.5*rho*Vs^2*pi*R^2);   % CT thrust coefficient required

dVs     = dV/Vs;                          % axial inflow variation / Vs
CDoCL   = mean(XCD)/CLI;
```

```

ALPHAstall = 8*pi/180;  % [rad], stall angle of attack - ideal angle of
                              attack

% ==========================================================================
% ================================================== Pack up input variables
input.filename   = filename;    % filename prefix for output files
input.date       = date;        % today's date

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;      % required thrust [N]
input.CTDES      = CTDES;       % [1 x 1], [ ] desired thrust coefficient

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.t0oc0      = t0oc0;       % [length(XR) x 1], [ ] input thickness /
                                  chord      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.ALPHAstall = ALPHAstall;  % [rad], stall angle of attack - ideal angle
                                  of attack
input.alphaI     = alphaI;      % [1 x 1], [deg] input ideal angle of attack
                                  at XR
input.CLI        = CLI;         % [1 x 1], [ ] input ideal lift coefficient
                                  at XR
input.CDoCL      = CDoCL;       % [1 x 1], [ ] blade section drag coefficient
                                  / lift coefficient

input.part3      = '------ Computational inputs ------';
input.ITER            = ITER;           % [ ] number of iterations
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.Wake_flag       = Wake_flag;      % 0 == Horseshoe(...,Wrench(...)),
                                          1== Wake_Horseshoe(...)
input.Optimizer_flag  = Optimizer_flag; % 1 == Lerbs optimizer, 2 == Epps
optimizer
input.Lagrange_flag   = Lagrange_flag;  % 0 == do not fix Lagrange
                                     multiplier, 1 == fix Lagrange multiplier
input.Make2Dplot_flag = Make2Dplot_flag;
input.Make3Dplot_flag = Make3Dplot_flag;
input.Make_Rhino_flag = Make_Rhino_flag;
nput.Rhv              = Rhv;         % [1 x 1], [ ] hub vortex radius / hub
radius

input.part4      = '------ Cavitation inputs ------';
input.rho        = rho;         % [1 x 1], [kg/m^3] fluid density
input.dVs        = dVs;         % [1 x 1], [ ] ship speed variation / ship
                                  speed
input.H          = H;           % [1 x 1]

input.part5      = '------ Duct inputs ------';

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

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

clear, clc, load OPinput

input
```

Appendix B – CFD Axial Inflow Results [3]



Model: AUV Speed: 2 m/s

Measurement Location: 1.90 m from nose of AUV, 1:30 (45°) and 4:30 (135°)

Max Radius: 0.06 m

Cells: ~2.5E6

Turbulence Model: SA (1st order)

Table B-1:Summary of CFD wake calculation results

Position in Disc[r] (m)-point 1 (m)	% Radius	1:30 Location, V (m/s)	1:30 Location, V/Uinf	4:30 Location, V (m/s)	4:30 Location, V/Uinf	**Average, V/Uinf**
0.06012	100.2%	1.8515	0.9257	1.8650	0.9325	**0.9291**
0.05659	94.3%	1.8379	0.9189	1.8552	0.9276	**0.9233**
0.05659	94.3%	1.8379	0.9189	1.8552	0.9276	**0.9233**
0.04951	82.5%	1.8082	0.9041	1.8372	0.9186	**0.9113**
0.04243	70.7%	1.7627	0.8813	1.8031	0.9015	**0.8914**
0.03892	64.9%	1.6952	0.8476	1.7413	0.8706	**0.8591**
0.03534	58.9%	1.6714	0.8357	1.71697	0.8584	**0.8470**
0.02822	47.0%	1.5189	0.7594	1.5570	0.7785	**0.7690**
0.02480	41.3%	1.3964	0.6982	1.4224	0.7112	**0.7047**
0.02140	35.7%	1.3045	0.6522	1.3114	0.6557	**0.6539**
0.01775	29.6%	1.1422	0.5711	1.1348	0.5674	**0.5692**
0.01430	23.8%	0.9325	0.4662	0.9282	0.4641	**0.4651**
0.0072164	12.0%	-0.06695	-0.03347	0.1928	0.09640	**0.03146**
0.003931	6.6%	-0.2775	-0.13876	-0.1472	-0.07363	**-0.1062**
0.002170	3.6%	-0.2441	-0.12205	-0.2441	-0.1220	**-0.12205**

Table B-2: CFD wake inputs used in OpenProp

Open Prop Points	Axial Inflow Velocity (From Formula)
0.2	0.3366
0.3	0.5506
0.4	0.7006
0.5	0.7987
0.6	0.8570
0.7	0.8875
0.8	0.9024
0.9	0.9139
0.95	0.9221
1	0.9340

Appendix C – Run Script Code

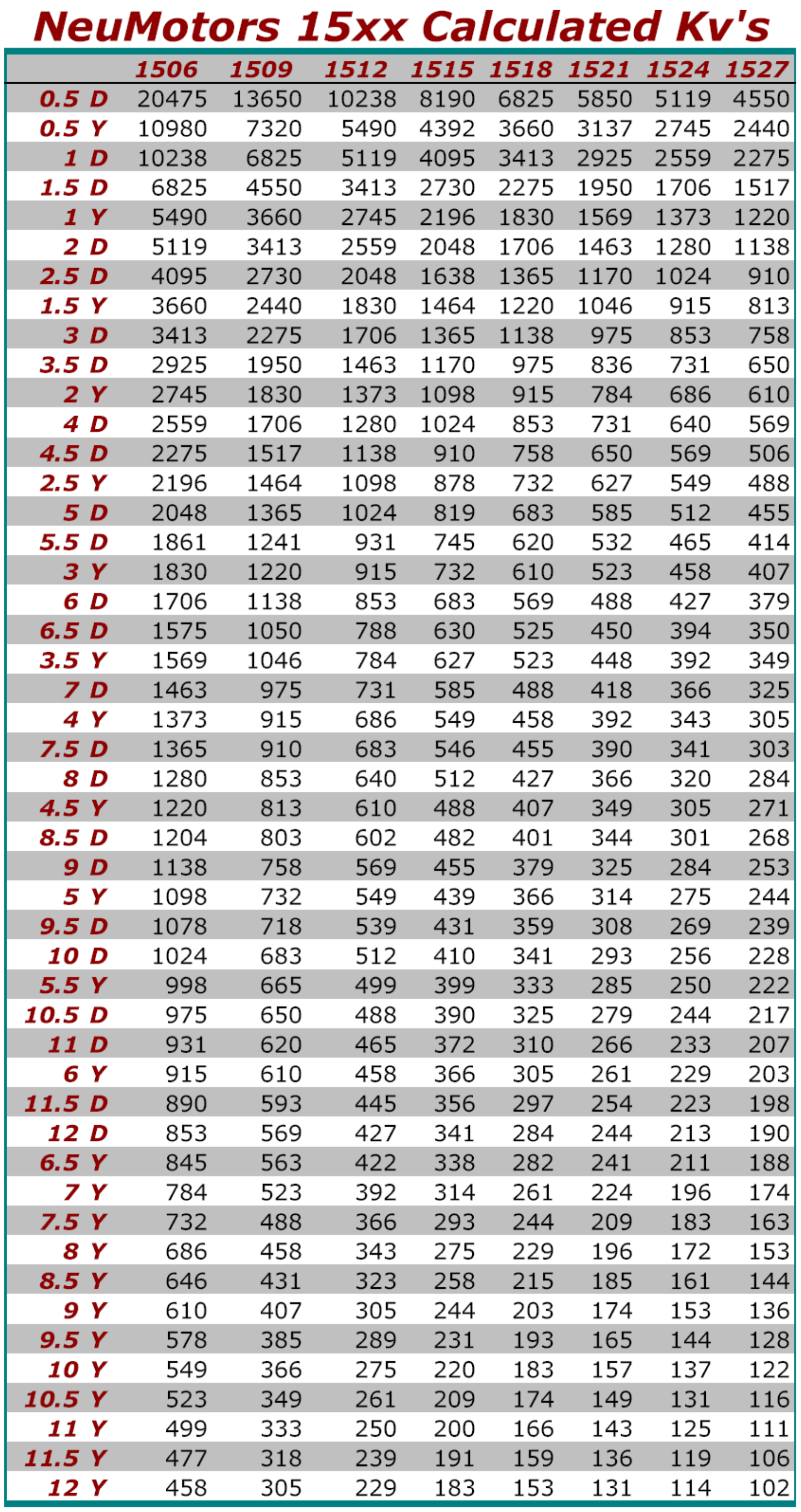
Due to text wrapping the line numbers are slightly off when compared to the m-file.

```
1   %%
2   %Modified and Additional Descriptive Comments: 2010, Richard Duelley
3   %Run the following commands to execute a single propeller design with
4   % OpenProp.
5   % Simply highlight desired operation and press F9
6   %%
7   addpath ../SourceCode
8
9   %%
10  % Single propeller design example:
11  clear, close all, clc,
12
13  % Load inputs:
14  Run1_input
15
16  pause(0.01)
17
18
19  %%
20  % Perform design optimization
21  pt.design = EppsOptimizer(input)
22
23  %Unsure of what this plot shows, not used in my design process
24  figure,
25      plot(pt.design.RC,pt.design.G,'.-b')
26      axis([0 1 0 0.03])
27      xlabel('r/R'), ylabel('G')
28
29  pause(0.01)
30
31
32  %%
33  % Create graphical and text reports, Creates an Input.txt, Output.txt,
34  % Performance.txt, and a Graphical Report
35  Make_Reports(pt)
36  %%
37  % Determine propeller geometry: Produces a series of tab delineated text
38  % files that can be imported to practically any CAD program.  Each file contains
39  % a series of coordinates that define a single cross sectional profile of
40  % one of the propeller blades.
41  pt.geometry = Geometry(pt)
42
43  %The modifications I did to the Geometry script in order to output text files for
44  %CAD drawings broke the 3D propeller model.  Just rerun using the original
45  %scrip to generate the 3D model if desired.  Make sure to save all files to
46  %another folder before running because it will overwrite any duplicate
47  %files.
48  pt.geometry = Geometry_Original(pt)
49  %%
50  % Analyze off-design states
51  Js_all= [1.05:-0.05:0.5]; % advance coefficient,  This defines the x axis of the
52                            %"Off-Design Results" plots below
```

```
LAMBDAall = pi./Js_all;           % tip-speed ratio
pt.states = Analyze(pt,LAMBDAall)

%%
%Plot Off-Design Results
figure, hold on,
    % Efficiency (green squares)
            plot(pt.states.Js,pt.states.EFFY,'-','LineWidth',2,'Color',[0 0.8 0])
    Heffy =
plot(pt.states.Js,pt.states.EFFY,'sk','MarkerSize',14,'LineWidth',1,'MarkerFaceCo
lor',[0 0.8 0]);

    % Thrust coefficient (blue diamonds)
            plot(pt.states.Js,pt.states.KT,'b-','LineWidth',2)
    Hkt   =
plot(pt.states.Js,pt.states.KT,'dk','MarkerSize',14,'LineWidth',1,'MarkerFaceColo
r','b');

    % Torque coefficient (red circles)
          plot(pt.states.Js,10*pt.states.KQ,'r-','LineWidth',2)
    Hkq =
plot(pt.states.Js,10*pt.states.KQ,'ok','MarkerSize',12,'LineWidth',1,'MarkerFaceC
olor','r');

    % Design point
    plot(pt.design.Js*[1 1],[0 2],'k--','LineWidth',1);

    xlabel('Js','Fontsize',24), ylabel('KT, 10*KQ, EFFY','Fontsize',24)
    axis([0.4 1.2 0 0.9])
    set(gca,'Fontsize',20)
    box on

%%
% Pefrorm cavitation analysis:
%%% Red indicates Cavitation, Blue indicates NO Cavitation.
Cav_CavitationMap(pt);

VLMbucket
```

Appendix D – NeuMotor Chart

This chart was provided, upon request, by the NeuMotor engineers. It summarizes all of the possible wind variations of the 1500 series of motors. The chart is preserved here for future reference.



NeuMotors 15xx Calculated Kv's

	1506	1509	1512	1515	1518	1521	1524	1527
0.5 D	20475	13650	10238	8190	6825	5850	5119	4550
0.5 Y	10980	7320	5490	4392	3660	3137	2745	2440
1 D	10238	6825	5119	4095	3413	2925	2559	2275
1.5 D	6825	4550	3413	2730	2275	1950	1706	1517
1 Y	5490	3660	2745	2196	1830	1569	1373	1220
2 D	5119	3413	2559	2048	1706	1463	1280	1138
2.5 D	4095	2730	2048	1638	1365	1170	1024	910
1.5 Y	3660	2440	1830	1464	1220	1046	915	813
3 D	3413	2275	1706	1365	1138	975	853	758
3.5 D	2925	1950	1463	1170	975	836	731	650
2 Y	2745	1830	1373	1098	915	784	686	610
4 D	2559	1706	1280	1024	853	731	640	569
4.5 D	2275	1517	1138	910	758	650	569	506
2.5 Y	2196	1464	1098	878	732	627	549	488
5 D	2048	1365	1024	819	683	585	512	455
5.5 D	1861	1241	931	745	620	532	465	414
3 Y	1830	1220	915	732	610	523	458	407
6 D	1706	1138	853	683	569	488	427	379
6.5 D	1575	1050	788	630	525	450	394	350
3.5 Y	1569	1046	784	627	523	448	392	349
7 D	1463	975	731	585	488	418	366	325
4 Y	1373	915	686	549	458	392	343	305
7.5 D	1365	910	683	546	455	390	341	303
8 D	1280	853	640	512	427	366	320	284
4.5 Y	1220	813	610	488	407	349	305	271
8.5 D	1204	803	602	482	401	344	301	268
9 D	1138	758	569	455	379	325	284	253
5 Y	1098	732	549	439	366	314	275	244
9.5 D	1078	718	539	431	359	308	269	239
10 D	1024	683	512	410	341	293	256	228
5.5 Y	998	665	499	399	333	285	250	222
10.5 D	975	650	488	390	325	279	244	217
11 D	931	620	465	372	310	266	233	207
6 Y	915	610	458	366	305	261	229	203
11.5 D	890	593	445	356	297	254	223	198
12 D	853	569	427	341	284	244	213	190
6.5 Y	845	563	422	338	282	241	211	188
7 Y	784	523	392	314	261	224	196	174
7.5 Y	732	488	366	293	244	209	183	163
8 Y	686	458	343	275	229	196	172	153
8.5 Y	646	431	323	258	215	185	161	144
9 Y	610	407	305	244	203	174	153	136
9.5 Y	578	385	289	231	193	165	144	128
10 Y	549	366	275	220	183	157	137	122
10.5 Y	523	349	261	209	174	149	131	116
11 Y	499	333	250	200	166	143	125	111
11.5 Y	477	318	239	191	159	136	119	106
12 Y	458	305	229	183	153	131	114	102

Appendix E – Input.txt

```
Run 1_Input.txt
OpenProp Input Table

Date and time: 2010-07-20 14:02:04

20          Number of Vortex Panels over the Radius
10          Max. Iterations in Wake Alignment
1           Hub Image Flag: 1=YES, 0=NO
0           Duct Flag:      1=YES, 0=NO
0.000       Duct Diameter
0.5         Hub Vortex Radius/Hub Radius
2           Number of Blades
0.541       Advance Coefficient Based on Ship Speed, Js
0.374       Desired Thrust Coefficient, Ct
1.000       Desired Thrust Ratio, tau
0.000       Duct Section Drag Coefficient, CDd
0           Hub Unloading Factor: 0 = optimum
0           Tip Unloading Factor: 1 = Reduced Loading
1           Swirl Cencellation Factor: 1 = No Cancellation

r/R       C/D       XCD     Va/Vs  Vt/Vs
0.20000  0.06500  0.00800    0.34  0.0000
0.30000  0.06500  0.00800    0.55  0.0000
0.40000  0.06550  0.00800    0.71  0.0000
0.50000  0.06600  0.00800    0.80  0.0000
0.60000  0.06700  0.00800    0.86  0.0000
0.70000  0.06500  0.00800    0.89  0.0000
0.80000  0.06000  0.00800    0.90  0.0000
0.90000  0.04500  0.00800    0.91  0.0000
0.95000  0.03300  0.00800    0.92  0.0000
1.00000  0.00100  0.00800    0.93  0.0000

r/R    [ ], input radial position / propeller radius.
c/D    [ ], input section chord-length / propeller diameter.
Cd     [ ], input section drag coefficient.
Va     [ ], input axial inflow velocity / ship velocity.
Vt     [ ], input tangential inflow velocity / ship velocity.
```

Appendix F – Output.txt

```
Run 1_Output.txt
OpenProp Output Table

Date and time: 2010-07-20 14:02:04

Js    = 0.5405
Ct    = 0.3739
Cq    = 0.0667
Kt    = 0.0429
Kq    = 0.0038
Cp    = 0.3876
VMIV  = 0.8286
Eff   = 0.7993
Tau   = 1.0000
Duct Circulation  = 0.0000

Output at the control points for the propeller

r/R          G        Va       Vt      Ua     Ua(ring)     Ut     Beta    BetaI   c/D        Cd
0.17099  0.030310  0.26417  0.0000  0.29950  0.00000  -0.17373  14.886  34.503  0.06500  0.00800
0.21296  0.030718  0.36771  0.0000  0.26510  0.00000  -0.15597  16.546  30.327  0.06500  0.00800
0.25494  0.030630  0.46201  0.0000  0.23559  0.00000  -0.13526  17.318  27.389  0.06500  0.00800
0.29691  0.029916  0.54506  0.0000  0.21044  0.00000  -0.11449  17.529  25.123  0.06500  0.00800
0.33889  0.028743  0.61863  0.0000  0.18912  0.00000  -0.09580  17.437  23.319  0.06512  0.00800
0.38086  0.027273  0.68262  0.0000  0.17110  0.00000  -0.07906  17.139  21.800  0.06539  0.00800
0.42284  0.025716  0.73180  0.0000  0.15587  0.00000  -0.06568  16.582  20.361  0.06561  0.00800
0.46481  0.024179  0.77133  0.0000  0.14289  0.00000  -0.05529  15.935  19.059  0.06580  0.00800
0.50679  0.022695  0.80356  0.0000  0.13155  0.00000  -0.04705  15.260  17.881  0.06605  0.00800
0.54877  0.021276  0.83105  0.0000  0.12111  0.00000  -0.04027  14.605  16.823  0.06657  0.00800
0.59074  0.019945  0.85310  0.0000  0.11203  0.00000  -0.03480  13.954  15.854  0.06698  0.00800
0.63272  0.018723  0.86917  0.0000  0.10540  0.00000  -0.03071  13.298  14.963  0.06670  0.00800
0.67469  0.017584  0.88176  0.0000  0.10020  0.00000  -0.02750  12.673  14.154  0.06572  0.00800
0.71667  0.016513  0.89068  0.0000  0.09692  0.00000  -0.02515  12.070  13.417  0.06447  0.00800
0.75864  0.015473  0.89711  0.0000  0.09660  0.00000  -0.02362  11.501  12.767  0.06264  0.00800
0.80062  0.014387  0.90258  0.0000  0.10031  0.00000  -0.02259  10.977  12.220  0.05995  0.00800
0.84259  0.013160  0.90734  0.0000  0.10907  0.00000  -0.02190  10.497  11.777  0.05507  0.00800
0.88457  0.011663  0.91195  0.0000  0.12396  0.00000  -0.02155  10.059  11.439  0.04790  0.00800
0.92654  0.009671  0.91794  0.0000  0.14600  0.00000  -0.02153  9.674  11.220  0.03965  0.00800
0.96852  0.006673  0.92605  0.0000  0.17625  0.00000  -0.02184  9.342  11.122  0.02415  0.00800
```

```
The propeller does not have a duct.

Js     [ ], advance coefficient.
Ct     [ ], required thrust coefficient.
Cp     [ ], power coefficient. Cp = Cq*pi/J.
Kt     [ ], thrust coefficient. Kt = Ct*Js^2*pi/8.
Kq     [ ], torque coefficient. Kq = Cq*Js^2*pi/16.
VMIV   [ ], volumetric mean inflow velocity / ship velocity.
Eff    [ ], efficiency = Ct*VMIV/Cp.
Tau    [ ], thrust ratio = propeller thrust / total thrust.

r/R    [ ], radial position of control points / propeller radius.
G      [ ], section circulation / 2*pi*R.
Va     [ ], axial inflow velocity / ship velocity.
Vt     [ ], tangential inflow velocity / ship velocity.
Ua     [ ], induced axial velocity / ship velocity.
```

Appendix G – Geometry.txt

```
Run 1_Geometry.txt
Propeller Geometry Table

Date and time: 20-Jul-2010

Propeller Diameter       = 0.1200 m
Number of Blades   = 2
Propeller Speed    = 1850 RPM
Propeller Hub Diameter   = 0.0180 m
Meanline  Type: NACA a=0.8
Thickness Type: NACA 65A010 (modified)

 r/R          P/D        Skew        Xs/D        c/D         f0/c        t0/c
0.1710       0.4356      0.0000      -0.0015     0.0650      0.1999      0.4818
0.2130       0.4509      0.0000      0.0006      0.0650      0.1609      0.4516
0.2549       0.4698      0.0000      0.0027      0.0650      0.1326      0.4245
0.2969       0.4882      0.0000      0.0048      0.0650      0.1103      0.4016
0.3389       0.5058      0.0000      0.0069      0.0651      0.0923      0.3828
0.3809       0.5216      0.0000      0.0090      0.0654      0.0774      0.3668
0.4228       0.5326      0.0000      0.0111      0.0656      0.0655      0.3526
0.4648       0.5410      0.0000      0.0132      0.0658      0.0560      0.3401
0.5068       0.5474      0.0000      0.0153      0.0661      0.0481      0.3283
0.5488       0.5523      0.0000      0.0174      0.0666      0.0414      0.3164
0.5907       0.5557      0.0000      0.0195      0.0670      0.0360      0.3055
0.6327       0.5581      0.0000      0.0216      0.0667      0.0317      0.2963
0.6747       0.5600      0.0000      0.0237      0.0657      0.0284      0.2882
0.7167       0.5613      0.0000      0.0258      0.0645      0.0257      0.2817
0.7586       0.5633      0.0000      0.0279      0.0626      0.0234      0.2764
0.8006       0.5673      0.0000      0.0300      0.0600      0.0216      0.2718
0.8426       0.5743      0.0000      0.0322      0.0551      0.0205      0.2673
0.8846       0.5851      0.0000      0.0343      0.0479      0.0199      0.2638
0.9265       0.6002      0.0000      0.0360      0.0396      0.0190      0.2628
0.9685       0.6240      0.0000      0.0380      0.0241      0.0206      0.2148

r/R    [ ], radial position of control points / propeller radius.
P/D    [ ], section pitch / diameter.
c/D    [ ], section chord-length / diameter.
fo/C   [ ], section camber / section chord-length.
to/C   [ ], section thickness / section chord-length.
```

Appendix H – Performance.txt

```
Run 1_Performance.txt
OpenProp Performance Table

Date and time: 2010-07-20 14:02:04

 r/R    V*    beta  betai      Gamma         CL           Sigma       dBetai
0.171  1.99  14.89  34.50      0.0229       2.944        63.335       11.76
0.213  2.51  16.55  30.33      0.0232       2.369        39.920       10.35
0.255  3.03  17.32  27.39      0.0231       1.953        27.262       9.14
0.297  3.56  17.53  25.12      0.0226       1.625        19.793       8.16
0.339  4.08  17.44  23.32      0.0217       1.359        15.050       7.35
0.381  4.60  17.14  21.80      0.0206       1.140        11.855       6.69
0.423  5.10  16.58  20.36      0.0194       0.965        9.624        6.14
0.465  5.60  15.94  19.06      0.0182       0.825        7.990        5.68
0.507  6.09  15.26  17.88      0.0171       0.709        6.751        5.28
0.549  6.58  14.60  16.82      0.0160       0.610        5.784        4.94
0.591  7.07  13.95  15.85      0.0150       0.530        5.015        4.63
0.633  7.55  13.30  14.96      0.0141       0.467        4.392        4.36
0.675  8.03  12.67  14.15      0.0133       0.419        3.880        4.12
0.717  8.51  12.07  13.42      0.0125       0.378        3.453        3.90
0.759  8.99  11.50  12.77      0.0117       0.345        3.093        3.71
0.801  9.48  10.98  12.22      0.0108       0.318        2.785        3.53
0.843  9.96  10.50  11.78      0.0099       0.301        2.521        3.36
0.885  10.45 10.06  11.44      0.0088       0.293        2.291        3.21
0.927  10.94 9.67   11.22      0.0073       0.280        2.090        3.07
0.969  11.43 9.34   11.12      0.0050       0.304        1.913        2.94

r/R         [ ], radial position of control points / propeller radius.
V*          [m/s], total inflow velocity.
beta        [deg], undisturbed flow angle.
betai       [deg], hydrodynamic Pitch angle.
Gamma       [m^2/s], vortex sheet strength.
CL          [ ], section lift coefficient.
Sigma       [ ], cavitation number.
d_alpha     [deg], inflow variation bucket width.
```

Appendix I – Modified Geometry m File

```
% ==========================================================================
% =================================== Determine Propeller Geometry Function
%
% This function determines the geometry of the propeller.  It outputs
% the geometry as a 2D image, 3D image, and Rhino CAD file.
%
% Modified by: Richard S Duelley to output tab delineated text files for
% use in CAD drawings, this modification broke the 3-D image plot.
%
% Reference:
%   J.S. Carlton, "Marine Propellers & Propulsion", ch. 3, 1994.
%
%   Abbott, I. H., and Von Doenhoff, A. E.; Theory of Wing Sections.
%   Dover, 1959.
%
% --------------------------------------------------------------------------
% Input Variables:
%
%   filename              file name prefix for all output files
%   Date_string           time and date to print on reports
%   Make2Dplot_flag       flag for whether to make 2D geometry plot
%   Make3Dplot_flag       flag for whether to make 3D geometry plot
%   Make_Rhino_flag       flag for whetehr to make a Rhino output file
%   Meanline              flag for choice of meanline  form
%   Thickness             flag for choice of thickness form
%
%   XR          [ ],    input radii / propeller radius
%   f0oc0       [ ],    input camber    / chord at each radius
%   t0oc0       [ ],    input thickness / chord at each radius
%   skew0       [deg],  input skew              at each radius
%   rake0       [ ],    input rake / diameter   at each radius
%
%   RC          [ ],    control point radii / propeller radius
%   CL          [ ],    section lift coefficients
%   Beta_c      [deg],  Beta  at the control points
%   BetaI_c     [deg],  BetaI at the control points
%   alphaI      [deg],  ideal angle of attack
%
%   D           [m],    propeller diameter
%   Z           [ ],    number of blades
%   N           [RPM],  propeller speed
%   Dhub        [m],    hub diameter
%   Rhub        [m],    hub radius
%
%   CoD         [ ],    chord / diameter at each control point radius
%   R           [m],    propeller radius
%   Mp          [ ],    number of radial 2D cross-sections
%   Np          [ ],    number of points in each 2D section
%   Js          [ ],    advance coefficient based on ship speed
%
% Output Variables:
%
% The function has graphical and file outputs, in addition to the geometry
% data structure.
%
```

```
% -------------------------------------------------------------------------

function [geometry] = Geometry(pt)
%%
% -------------------------------------------------------- Unpack variables
Date_string      = pt.date;

filename         = pt.input.filename;
Make2Dplot_flag = pt.input.Make2Dplot_flag;
Make3Dplot_flag = pt.input.Make3Dplot_flag;
Make_Rhino_flag = pt.input.Make_Rhino_flag;
Hub_flag         = pt.input.Hub_flag;        % 0 == no hub, 1 == hub
Duct_flag        = pt.input.Duct_flag;       % 0 == no duct, 1 == duct
Chord_flag       = pt.input.Chord_flag;      % 0 == do not optimize chord
lengths, 1 == optimize chord lengths

Meanline         = pt.input.Meanline;
Thickness        = pt.input.Thickness;

XR               = pt.input.XR;
t0oc0            = pt.input.t0oc0;
XCoD             = pt.input.XCoD;
skew0            = pt.input.skew0;   % [deg]
rake0            = pt.input.rake0;

Z                = pt.input.Z;
Js               = pt.input.Js;
Vs               = pt.input.Vs;      % [m/s]
R                = pt.input.R;       % [m]
Rhub             = pt.input.Rhub;    % [m] hub radius

Mp               = pt.input.Mp;
Np               = pt.input.Np;

RC               = pt.design.RC;
RV               = pt.design.RV;
CL               = pt.design.CL;
Beta_c           = atand(pt.design.TANBC);  % [deg]
BetaI_c          = pt.design.BetaIC*180/pi; % [deg]
CoD              = pt.design.CoD;
t0oc             = pt.design.t0oc;
TANBIV           = pt.design.TANBIV;


D       = 2*R;     % [m]
Dhub    = 2*Rhub; % [m]
Rhub_oR = Rhub/R;
N       = 60*Vs/(Js*D); % [RPM]


% -------------------------------------------------------------------------
% % ---------------- Interpolate input geometry at selected radial sections
% RG = [0.9*Rhub_oR,RV(2:end-1),1];
% Interpolate input geometry at sections with cosine spacing along the span
RG = 0.9*Rhub_oR + (1-0.9*Rhub_oR)*(sin((0:Mp)*pi/(2*Mp)));  % [0.9*Rhub_oR :
1]

CL      = interp1(RC,CL     ,RG,'pchip','extrap');
t0oc    = interp1(RC,t0oc   ,RG,'pchip','extrap');
BetaI_c = interp1(RC,BetaI_c,RG,'pchip','extrap');

skew = pchip(XR,skew0,RG);       % [deg], angular translation along mid-chord
helix
rake = pchip(XR,rake0,RG)*D;     % [m],   translation along propeller axis
(3D X-axis)


if Chord_flag == 1
    CoD  =  interp1(RC,CoD,RG,'pchip','extrap');
else
    CoD  = pchip(XR,XCoD,RG);
end

c = CoD.*D;                          % section chord at the RG sections [m]
r = RG.*R;                           % radius of        the RG sections [m]

% -------------------------------------------------------------------------

%
% ---------------------------------------- Lay out the 2D coordinate system
%
% xN   [ ], x/c coordinate in 2D NACA foil tables
%                At the Leading  Edge: xN = 0, x1 =  c/2, x0 = 0
%                At the Trailing Edge: xN = 1, x1 = -c/2, x0 = 1
% x0   [ ], x/c distance along mid-chord line to interpolate NACA foil table
data.
% x1   [m], x   distance along mid-chord line to evaluate elliptical or
parabolic formulae.
%                By definition, x1 == c/2 - c*x0.
%
% x2D  [m], x   position in 2D space on upper (x2D_u) and lower (x2D_l) foil
surfaces
% y2D  [m], y   position in 2D space on upper (x2D_u) and lower (x2D_l) foil
surfaces
% x2Dr [m], x   position in 2D space after rotation for pitch angle
% y2Dr [m], y   position in 2D space after rotation for pitch angle
%

xN = [0 .5 .75 1.25 2.5 5 7.5 10 15 20 25 30 35 40 45 50 ...
     55 60 65 70 75 80 85 90 95 100]./100;

x0 = zeros(1,Np);
x1 = zeros(Mp+1,Np);
% % Even spacing along the chord
% % for i = 1:Mp                       % for each radial section along the span
% for i = 1:Mp+1                     % for each radial section along the span
%     for j = 1:Np                   % for each point          along the
chord
```

```
%          x0(1,j)      =                (j-1)/(Np-1);  % [0   :    1]
%          x1(i,j)      = c(i)/2 - c(i)*(j-1)/(Np-1);  % [c/2 : -c/2]
%      end
% end

% Cosine spacing along the chord
for i = 1:Mp+1                           % for each radial section along the span
    for j = 1:Np                         % for each point          along the chord
        x1(i,j)      = c(i)/2 - 0.5*c(i)*(1-cos(pi*(j-1)/(Np-1)));  % [c/2 :
-c/2]
    end
end
x0 = 0.5-x1(1,:)/c(1);

% ------------------ Find meanline and thickness profiles (at x1 positions)
%
% foc    = camber / chord ratio (NACA data at xN positions)
% dfdxN  = slope of camber line (NACA data at xN positions)
% fscale = scale to set max camber    ratio to f0oc for each section
% tscale = scale to set max thickness ratio to t0oc for each section
% f      = camber               at x1 positions
% dfdx   = slope of camber line at x1 positions
% t      = thickness            at x1 positions

t    = zeros(Mp+1,Np);
f    = zeros(Mp+1,Np);
dfdx = zeros(Mp+1,Np);


if Meanline==0 | strcmp(Meanline,'NACA a=0.8 (modified)')  % --------------
Use NACA a=0.8 (modified) meanline
    Meanline = 'NACA a=0.8 (modified)';

    foc = [0 0.281 0.396 0.603 1.055 1.803 2.432 2.981 3.903 4.651 5.257 ...
             5.742 6.120 6.394 6.571 6.651 6.631 6.508 6.274 5.913 5.401 ...
             4.673 3.607 2.452 1.226 0  ]./100;

    dfdxN = [0 0.47539  0.44004  0.39531  0.33404  0.27149  0.23378  0.20618
0.16546 ...
               0.13452  0.10873  0.08595  0.06498  0.04507  0.02559  0.00607
...
              -0.01404 -0.03537 -0.05887 -0.08610 -0.12058 -0.18034 -0.23430
...
              -0.24521 -0.24521 -0.24521];

    CLI        = 1.00;     % NACA data ideal lift coefficient
    alphaItilde    = 1.40;     % [deg]
    fscale     = CL / CLI;
    f0octilde = max(foc);                % f0/c of NACA data with CLI == 1
    f0oc      = f0octilde * CL / CLI;    % f0/c, scaled for CL at RG


    dfdxLE = 0.47539*fscale;      % slope at leading edge
    % for i = 1:Mp                         % for each radial section along the
span
    for i = 1:Mp+1                         % for each radial section along the
span
        for j = 1:Np
            f(i,:)    = pchip(xN,foc  .*fscale(i).*c(i),x0);
            dfdx(i,:) = pchip(xN,dfdxN.*fscale(i)      ,x0);
        end
    end

    % alphaItilde = 1.40

elseif Meanline==1 | strcmp(Meanline,'NACA a=0.8')          % --------------
----------- Use NACA a=0.8 meanline
    Meanline = 'NACA a=0.8';

    foc = [0 .287 .404 .616 1.077 1.841 2.483 3.043 3.985 4.748 ...
           5.367 5.863 6.248 6.528 6.709 6.790 6.770 6.644 6.405  ...
           6.037 5.514 4.771 3.683 2.435 1.163 0]./100;

    dfdxN = [0 .48535 .44925 .40359 .34104 .27718 .23868 .21050 ...
             .16892 .13734 .11101 .08775 .06634 .04601 .02613 ...
             .00620 -.01433 -.03611 -.06010 -.08790 -.12311   ...
             -.18412 -.23921 -.25583 -.24904 -.20385];

    CLI        = 1.00;     % NACA data ideal lift coefficient
    alphaItilde    = 1.54;     % [deg]
    fscale    = CL / CLI;
    f0octilde = max(foc);                 % f0/c of NACA data with CLI == 1
    f0oc      = f0octilde * CL / CLI;    % f0/c, scaled for CL at RG

    dfdxLE = 0.48535*fscale;      % slope at leading edge

    % for i = 1:Mp                         % for each radial section along the
span
    for i = 1:Mp+1                         % for each radial section along the
span
        for j = 1:Np
            f(i,:)    = pchip(xN,foc  .*fscale(i).*c(i),x0);
            dfdx(i,:) = pchip(xN,dfdxN.*fscale(i)      ,x0);
        end
    end

    % alphaItilde = 1.54

elseif Meanline==2 | strcmp(Meanline,'parabolic')       % -------------------
------- Use parabolic meanline
    Meanline = 'parabolic';

    % For parabolic meanline: alphaItilde == 0, CLI == 4*pi*f0oc
    % However, set CLI and f0octilde such that f0oc == f0octilde * CL / CLI
    alphaItilde = 0;
    CLI         = 1;
    f0octilde   = 1  / (4*pi);
    f0oc        = CL / (4*pi); % == f0octilde * CL / CLI;
    % for i = 1:Mp                         % for each radial section along the
span
    for i = 1:Mp+1                         % for each radial section along the
span
        for j = 1:Np
            f(i,j)     =    f0oc(i)*c(i)*(1-(2*x1(i,j)/c(i))^2);
            dfdx(i,j) = -8*f0oc(i)*x1(i,j)/c(i);
        end
    end
end

if Thickness==1 | strcmp(Thickness,'NACA 65A010')          % ----------------
-- Use NACA 65A010 thickness form
    Thickness = 'NACA 65A010';

    toc_65 = [0 .765 .928 1.183 1.623 2.182 2.65 3.04 3.658 4.127 ...
              4.483 4.742 4.912 4.995 4.983 4.863 4.632 4.304     ...
              3.899 3.432 2.912 2.352 1.771 1.188 .604 .021]./100;

    tscale = t0oc / max(toc_65);


    rLE    = 0.00639*c.*tscale; % leading edge radius

    % for i = 1:Mp                         % for each radial section along the
span
    for i = 1:Mp+1                         % for each radial section along the
span
        for j = 1:Np
            t(i,:) = pchip(xN,toc_65.*tscale(i).*c(i),x0);
        end
    end

elseif Thickness==2 | strcmp(Thickness,'elliptical')     % ------------------
- Use elliptical thickness form
    Thickness = 'elliptical';

    % for i = 1:Mp                         % for each radial section along the
span
    for i = 1:Mp+1                         % for each radial section along the
span
        for j = 1:Np
            t(i,j) = t0oc(i)*c(i)*real(sqrt(1-(2*x1(i,j)/c(i))^2));
        end
    end

    rLE = 0; % leading edge radius

elseif Thickness==3 | strcmp(Thickness,'parabolic')     % -------------------
- Use parabolic thickness form
    Thickness = 'parabolic';
    % for i = 1:Mp                          % for each radial section along the
span
    for i = 1:Mp+1                          % for each radial section along the
span
        for j = 1:Np
            t(i,j) = t0oc(i)*c(i)*(1-(2*x1(i,j)/c(i))^2);
        end
    end

    rLE = 0; % leading edge radius




elseif Thickness==4 | strcmp(Thickness,'NACA 65A010 (modified)')     % ------
-------------- Use modified NACA 65A010 thickness form
    Thickness = 'NACA 65A010 (modified)';

    xx65mod = [0    0.005000000000000   0.007500000000000   0.012500000000000
...
                0.025000000000000   0.050000000000000   0.075000000000000
0.100000000000000 ...
                0.150000000000000   0.200000000000000   0.250000000000000
0.300000000000000 ...
                0.350000000000000   0.400000000000000   0.471204188481675
0.523560209424084 ...
                0.575916230366492   0.628272251308901   0.680628272251309
0.732984293193717 ...
                0.785340314136126   0.837696335078534   0.890052356020942
0.942408376963351 ...
                0.968586387434555   0.981675392670157   0.989528795811518
0.994764397905759 ...
                0.997382198952880   1.000000000000000];
    tt65mod = [0    0.007650000000000   0.009280000000000   0.011830000000000
...
                0.016230000000000   0.021820000000000   0.026500000000000
0.030400000000000 ...
                0.036580000000000   0.041270000000000   0.044830000000000
0.047420000000000 ...
                0.049120000000000   0.049950000000000   0.049830000000000
0.048630000000000 ...
                0.046320000000000   0.043040000000000   0.038990000000000
0.034320000000000 ...
                0.029120000000000   0.023520000000000   0.017710000000000
0.011880000000000 ...
                0.008960000000000   0.007499530848329   0.006623639691517
0.006040000000000 ...
                0.004049015364794   0.000210000000000];

    tscale = t0oc / max(tt65mod);


    rLE    = 0.00639*c.*tscale; % leading edge radius
    % for i = 1:Mp                       % for each radial section along the
span
    for i = 1:Mp+1                       % for each radial section along the
span
        for j = 1:Np
            t(i,:) = pchip(xx65mod,tt65mod.*tscale(i).*c(i),x0);
        end
    end

end

%
% ---------------------------------------------------------------------------
% ------------------------------------- Find 2D unroatated section profiles
% x2D  [m], x   position in 2D space on upper (x2D_u) and lower (x2D_l) foil
surfaces
% y2D  [m], y   position in 2D space on upper (x2D_u) and lower (x2D_l) foil
surfaces
x2D_u = zeros(Mp+1,Np);     x2D_l = zeros(Mp+1,Np);
y2D_u = zeros(Mp+1,Np);     y2D_l = zeros(Mp+1,Np);

for i = 1:Mp+1                            % for each section along the span
    for j = 1:Np                          % for each point   along the chord
        x2D_u(i,j) = x1(i,j) + (t(i,j)/2)*sin(atan(dfdx(i,j))); % 2D upper
surface x
        x2D_l(i,j) = x1(i,j) - (t(i,j)/2)*sin(atan(dfdx(i,j))); % 2D lower
surface x
        y2D_u(i,j) =  f(i,j) + (t(i,j)/2)*cos(atan(dfdx(i,j))); % 2D upper
surface y
        y2D_l(i,j) =  f(i,j) - (t(i,j)/2)*cos(atan(dfdx(i,j))); % 2D lower
surface y
    end
end



% % -------------------------------------- Compute leading edge radius points
% phiLEC = atan(dfdxLE);
% NLE    = 3; % must be odd to capture leading edge point
% phiLEs = 3*pi/8;
% phiLE  = phiLEs:(pi-2*phiLEs)/(NLE-1):pi-phiLEs;
% xLEC   = x1(:,1)' - rLE.*cos(phiLEC);
% yLEC   =            rLE.*sin(phiLEC);
%
% xLE = zeros(Mp+1,NLE);
% yLE = zeros(Mp+1,NLE);
%
% for i = 1:Mp+1                            % for each section along the span
%     xLE(i,:) = xLEC(i) + rLE(i)*sin(phiLE+phiLEC(i));
%     yLE(i,:) = yLEC(i) - rLE(i)*cos(phiLE+phiLEC(i));
% end


% ----------------------------------------- Put all the numbers in one list
% % Nose -> suctioin side -> tail -> pressure side -> nose
% x2D(:,   1:Np   ) = x2D_u(:,1:Np);     % The first Np values are the upper
surface (suction side),
% x2D(:,1+Np:Np+Np) = x2D_l(:,Np:-1:1);  % and the second Np values are the
lower surface (pressure side).
% y2D(:,   1:Np   ) = y2D_u(:,1:Np);
% y2D(:,1+Np:Np+Np) = y2D_l(:,Np:-1:1);

% % j = 1          == tail
% % j = 1:Np       == suction side
% % j = Np         == nose
% % j = Np + 1     == nose
% % j = Np+ 1:2*Np == pressure side
% % j = 2*Np       == tail
% % Tail -> suctioin side -> nose, nose -> pressure side -> tail
x2D(:,   1:Np   ) = x2D_u(:,Np:-1:1);   % The first Np values are the upper
surface (suction side),
x2D(:,Np+1:Np+Np) = x2D_l(:,1:Np);      % and the second Np values are the
lower surface (pressure side).
y2D(:,   1:Np   ) = y2D_u(:,Np:-1:1);
y2D(:,Np+1:Np+Np) = y2D_l(:,1:Np);



% % % Arrange points as follows:
% % %     Tail -> suctioin side -> leading edge (with radius and nose) ->
pressure side -> tail
% % % j = 1                               == [1         point ] tail
% % % j = 1              : Np-1           == [Np-1      points] suction side
(tail to point aft of leading edge radius)
% % % j = Np-1+1         : Np-1+(NLE-1)/2 == [(NLE-1)/2 points] suction side
along leading edge radius
% % % j = Np+(NLE-1)/2                    == [1         point ] nose
% % % j = Np+(NLE-1)/2+1 :    Np-1+NLE    == [(NLE-1)/2 points] pressure side
along leading edge radius
% % % j = Np-1+NLE+1     : 2*(Np-1)+NLE   == [Np-1      points] pressure side
(point aft of leading edge radius to tail)
% % % j = 2*(Np-1)+NLE                    == [1         point ] tail
% % %
% % % j =            1   : Np+(NLE-1)/2   == suction  side (tail to nose)
% % % j = Np+(NLE-1)/2   : 2*(Np-1)+NLE   == pressure side (nose to tail)
% %
% % x2D(:,1              :    Np-1     ) = x2D_u(:,Np:-1:2); % [Np-1 points]
suction  side (tail to point aft of leading edge radius)
% % x2D(:,Np-1+1         :    Np-1 +NLE) =   xLE(:,NLE:-1:1);     % [NLE
points] leading edge radius
% % x2D(:,Np-1+NLE+1     : 2*(Np-1)+NLE) = x2D_l(:,2:Np);    % [Np-1 points]
pressure side (point aft of leading edge radius to tail)
% %
% % y2D(:,1              :    Np-1     ) = y2D_u(:,Np:-1:2); % [Np-1 points]
suction  side (tail to point aft of leading edge radius)
% % y2D(:,Np-1+1         :    Np-1 +NLE) =   yLE(:,NLE:-1:1);     % [NLE
points] leading edge radius
% % y2D(:,Np-1+NLE+1     : 2*(Np-1)+NLE) = y2D_l(:,2:Np);    % [Np-1 points]
pressure side (point aft of leading edge radius to tail)
```

```
% %--------------------------------------- plot unrotated blade
%  Fig2_S = figure('units','normalized','position',[0.31 .06 .4
.3],'name',...
%          'Blade Image','numbertitle','off');
%     style=['r' 'g' 'b' 'm' 'k'];
%     str_prefix = {'r/R = '};
%     flag=1;
%     for i = 1:ceil(Mp/5):Mp     % for five radial sections from root to tip
%         plot(x2D(i,:)*39.37,y2D(i,:)*39.37,style(flag));
%         str_legend(flag)=strcat(str_prefix,num2str(RC(i)));
%         hold on;
%         flag = flag+1;
%     end
%     legend(str_legend,'location','northwest');
%     axis equal;     grid on;
%     title('2D Blade Image');  xlabel('X (2D) [m]');  ylabel('Y (2D) [m]');
% %---------------------------------------


% ----------------------------------------------- Find pitch angle and pitch
theta    = BetaI_c + alphaItilde.*CL/CLI; % Nose-tail pitch angle, [deg]
PoD      = tand(theta).*pi.*RG;           % Pitch / propeller diameter, [ ]
theta_Z  = 0:360/Z:360;                   % angle between blades [deg]




% ---------------------------------------- Find 2D roatated section profiles
% x2Dr [m], x position in 2D space after rotation for pitch angle
% y2Dr [m], y position in 2D space after rotation for pitch angle
% x2Dr = zeros(Mp+1,2*(Np-1)+NLE);
% y2Dr = zeros(Mp+1,2*(Np-1)+NLE);
x2Dr = zeros(Mp+1,2*Np);
y2Dr = zeros(Mp+1,2*Np);
% for i = 1:Mp        % for each section along the span
for i = 1:Mp+1        % for each section along the span
    x2Dr(i,:) = x2D(i,:)*cosd(theta(i)) - y2D(i,:)*sind(theta(i)); % rotated
2D upper and lower surface x
    y2Dr(i,:) = x2D(i,:)*sind(theta(i)) + y2D(i,:)*cosd(theta(i)); % rotated
2D upper and lower surface y
end

% ---------------------------- Invoke skew and rake, and find 3D coordinates
% X3D [m], X position in 3D space (corresponds to y position in 2D space)
% Y2D [m], Y position in 3D space
% Z3D [m], Z position in 3D space
% X3D = zeros(Mp+1,2*(Np-1)+NLE,Z);
% Y3D = zeros(Mp+1,2*(Np-1)+NLE,Z);
% Z3D = zeros(Mp+1,2*(Np-1)+NLE,Z);
X3D = zeros(Mp+1,2*Np,Z);
Y3D = zeros(Mp+1,2*Np,Z);
Z3D = zeros(Mp+1,2*Np,Z);
```

```
%%%%%%%%%%%%ADDED 39.37007874 TO CONVERT FROM METERS TO INCHES FOR Nx 3D
%%%%%%%%%%%%MODEL



% for i = 1:Mp          % for each section along the span
for i = 1:Mp+1          % for each section along the span
%     for j = 1:2*(Np-1)+NLE    % for each point   along the upper and lower
surfaces
    for j = 1:2*Np     % for each point   along the upper and lower surfaces
        for k = 1:Z    % for each blade
            X3D(i,j,k) = (- rake(i) - r(i)*(pi*skew(i)/180)*tand(theta(i)) +
y2Dr(i,j))*39.37007874;

            Y3D(i,j,k) = (r(i)*sind(skew(i) - (180/pi)*x2Dr(i,j)/r(i) -
theta_Z(k)))*39.37007874;
            Z3D(i,j,k) = (r(i)*cosd(skew(i) - (180/pi)*x2Dr(i,j)/r(i) -
theta_Z(k)))*39.37007874;
        end
    end
end











% ==========================================================================
% ============================= Pack up geometry data at the control points
CL      = pt.design.CL;
BetaI_c = pt.design.BetaIC*180/pi;          % [deg]
CoD     = pt.design.CoD;                    % [ ],   c/D
t0oc    = pt.design.t0oc;                   % [ ],   t0/c
skew    = pchip(XR,skew0,RC);               % [deg],
rake    = pchip(XR,rake0,RC)*D;             % [m],
f0oc    = f0octilde   * CL/CLI;             % [ ],
alphaI  = alphaItilde * CL/CLI;             % [deg], ideal angle of attack
alpha   = alphaItilde * CL/CLI;             % [deg], blade angle of attack
theta   = BetaI_c + alphaI;                 % [deg], Nose-tail pitch angle
PoD     = tand(BetaI_c + alphaI).*pi.*RC;   % [ ],   pitch / D
```

```
geometry.Meanline  = Meanline;
geometry.Thickness = Thickness;

geometry.Z         = Z;
geometry.D         = D;                              % [m]
geometry.Dhub      = Dhub;                           % [m]
geometry.N         = N;                              % [RPM]

geometry.RC        = RC;                             % r/R
geometry.CoD       = pt.design.CoD;
geometry.t0oc      = t0oc;
geometry.skew      = skew;% [deg] angular translation along mid-chord helix
geometry.rake      = rake;% [m] translation along propeller axis, 3D X-axis
geometry.f0oc      = f0oc;
geometry.alphaI    = alphaI;
geometry.alpha     = alpha;
geometry.theta     = theta;
geometry.PoD       = PoD;
% =========================================================================
% =========================================================================
% ========================================== Create plots and text outputs
%%
% ----------------------------------------- Create 2D Propeller Blade Image
if Make2Dplot_flag
    Fig2_S = figure('units','normalized','position',[0.31 .06 .4
.3],'name',...
        'Blade Image','numbertitle','off');
    style=['r' 'g' 'b' 'm' 'k'];
    str_prefix = {'r/R = '};
    flag=1;
    for i = 1:ceil(Mp/5):Mp     % for five radial sections from root to tip
        plot(x2Dr(i,:),y2Dr(i,:),style(flag));
        str_legend(flag)=strcat(str_prefix,num2str(RC(i)));
        hold on;
        flag = flag+1;
    end
    legend(str_legend,'location','northwest');
    axis equal;     grid on;
    title('2D Blade Image');  xlabel('X (2D) [m]');  ylabel('Y (2D) [m]');

%     filename_2D = strcat(filename,'_2D_Blade_Image');
%     saveas(gcf,filename_2D,'jpg')
end

%%
% ----------------------------------------------- Create 3D Propeller Image
if Make3Dplot_flag
%%
    Fig3_S = figure('units','normalized','position',[.61 .06 .4 .3],...
                    'name','Propeller Image','numbertitle','off');
    hold on;

    % ------------------------------------------ Plot the propeller surface
    for k = 1:Z
        surf(X3D(:,:,1),Y3D(:,:,k),Z3D(:,:,k));
    end

    colormap gray;
    shading interp;
    grid on;
    if Duct_flag == 0
        axis([-R/2 R -1.1*R 1.1*R -1.1*R 1.1*R]);
    else
        axis([-R R -1.5*R 1.5*R -1.5*R 1.5*R]);       %modified for duct
    end
    axis equal;
    xlabel('X (3D) [m]','FontSize',12);
    ylabel('Y (3D) [m]','FontSize',12);
    zlabel('Z (3D) [m]','FontSize',12);
    title('3D Propeller Image','FontSize',16);

    % -------------------------------------------------------- Plot the hub
    Lhub = Dhub;

    tick = 90:-15:0;
    [yh0,zh0,xh0] = cylinder(Rhub*sind(tick),50);
    xh0 = -Lhub/4*xh0 - Rhub;
    surf(xh0,yh0,zh0);

    [yh1,zh1,xh1] = cylinder(Rhub,50); % xh1 = [0,1]
    xh1 = Lhub*xh1 - Rhub;            % xh1 = [-Rhub,c(1)-Rhub]
    surf(xh1,yh1,zh1);



    % ----------------- Plot the suction side (green) & pressure side (red)
    for i = 1:Mp+1          % for each section along the span
        for k = 1:Z       % for each blade
            plot3(X3D(i,1:Np,1),     Y3D(i,1:Np,k),     Z3D(i,1:Np,k),
'g','Linewidth',1); % suction surface

plot3(X3D(i,Np+1:2*Np,1),Y3D(i,Np+1:2*Np,k),Z3D(i,Np+1:2*Np,k),'r','Linewidth
',1); % pressure surface
        end
    end

    for j = 1:Np          % for each point along the chord
        for k = 1:Z       % for each blade
            plot3(X3D(:,j,1),   Y3D(:,j,k),   Z3D(:,j,k),
'g','Linewidth',1); % suction surface

plot3(X3D(:,j+Np,1),Y3D(:,j+Np,k),Z3D(:,j+Np,k),'r','Linewidth',1); %
pressure surface
        end
    end

    % ---------------------------------------------------- Plot the tip black
    i = Mp+1; % tip section
    for k = 1:Z
        for j = 1:Np-2
                plot3([X3D(i,1+j,k), X3D(i,2*Np-j,k)],...
                      [Y3D(i,1+j,k), Y3D(i,2*Np-j,k)],...
                      [Z3D(i,1+j,k), Z3D(i,2*Np-j,k)],'k','Linewidth',1); %
tip surface
        end
    end

    % --------------------------------- Plot the leading and trailing edges
    for k = 1:Z           % for each blade
        plot3(X3D(:,1,1), Y3D(:,1,k), Z3D(:,1,k), 'b','Linewidth',2); %
leading edge
        plot3(X3D(:,Np,1),Y3D(:,Np,k),Z3D(:,Np,k),'k','Linewidth',2); %
trailing edge
    end

    % ------------------------------------------ Plot the coordinate system
    % Axes
    plot3([0 R],[0 0],[0 0],'y','LineWidth',2),
    plot3([0 0],[0 R],[0 0],'r','LineWidth',2),
    plot3([0 0],[0 0],[0 R],'b','LineWidth',2),

    % Circle at the X = 0 location on the hub
    phi = 0:0.01:2*pi;
    Xhc =   zeros(size(phi));
    Yhc = - Rhub * sin(phi);
    Zhc =   Rhub * cos(phi);
    plot3(Xhc,Yhc,Zhc,'y','LineWidth',2),

    % Propeller reference line (i.e. the directrix)
    for k = 1:Z
        PRL(:,k) = [1,                0,                 0; ...
                    0, cosd(theta_Z(k)), -sind(theta_Z(k)); ...
                    0, sind(theta_Z(k)),  cosd(theta_Z(k))]*[0; 0; R];

        plot3([0, PRL(1,k)],[0, PRL(2,k)],[0, PRL(3,k)],'y--','LineWidth',1)
    end

    % ---------------------------------------------- Plot propeller helices
    % Advance coefficient helix 0, black
    phi = 0:0.01:pi/4;
    thetaH = atan((Js/pi)*(R/Rhub));
    Xh0 =   Rhub * phi * tan(thetaH);
    Yh0 = - Rhub * sin(phi);
    Zh0 =   Rhub * cos(phi);

    % Beta angle helix 1, red
    phi = 0:0.01:pi/4;
    thetaH = Beta_c(1)*pi/180;
    Xh1 =   Rhub * phi * tan(thetaH);
    Yh1 = - Rhub * sin(phi);
    Zh1 =   Rhub * cos(phi);
    % BetaI angle helix 2, green
    phi = 0:0.01:pi/4;
    thetaH = BetaI_c(1)*pi/180;
    Xh2 =   Rhub * phi * tan(thetaH);
    Yh2 = - Rhub * sin(phi);
    Zh2 =   Rhub * cos(phi);

    % Pitch angle helix 3, blue
    phi = 0:0.01:pi/4;
    thetaH = theta(1)*pi/180;
    Xh3 =   Rhub * phi * tan(thetaH);
    Yh3 = - Rhub * sin(phi);
    Zh3 =   Rhub * cos(phi);

    plot3(Xh0,Yh0,Zh0,'k','LineWidth',2),
    plot3(Xh1,Yh1,Zh1,'r','LineWidth',2),
    plot3(Xh2,Yh2,Zh2,'g','LineWidth',2),
    plot3(Xh3,Yh3,Zh3,'b','LineWidth',2),

% % ---------------------------------------------- Plot the trailing vortices
%     % Beta angle helix at each votex point (each trailing vortex)
%     BetaI_v = atand(TANBIV);
%
%     for m = 1:Mp+1
%         phi = 0:0.01:2*pi;
%         thetaH = BetaI_v(m)*pi/180;
%         Xh4 = - RV(m)*R * phi * tan(thetaH);
%         Yh4 =   RV(m)*R * sin(phi);
%         Zh4 =   RV(m)*R * cos(phi);
%
%         plot3(Xh4,Yh4,Zh4,'g','LineWidth',2),
%     end
% %
% %     % Beta angle image helix for each spanwise section
% %     for m = 1:Mp+1
% %         RVW   = Rhub_oR^2/RV(m);
% %         TANBW = TANBIV(1)*RV(1)/RVW;
% %         phi = 0:0.01:2*pi;
% %         thetaH = atand(TANBW)*pi/180;
% %         Xh4 = - RVW*R * phi * tan(thetaH);
% %         Yh4 =   RVW*R * sin(phi);
% %         Zh4 =   RVW*R * cos(phi);
% %
% %         plot3(Xh4,Yh4,Zh4,'--r','LineWidth',2),
% %     end
% %
% % --------------------------------------------- Plot the horseshoe vortices
%     % Beta angle helix at each votex point (each trailing vortex)
%     BetaI_v = atand(TANBIV);
%     dR = 0.005*R;
%
%     for m = 1:Mp
%         phi = 0:0.01:2*pi;
%         thetaH = BetaI_v(m)*pi/180;
%         Xh4 = - (RV(m)+dR)*R * phi * tan(thetaH);
%         Yh4 =   (RV(m)+dR)*R * sin(phi);
%         Zh4 =   (RV(m)+dR)*R * cos(phi);
%
%         plot3(Xh4,Yh4,Zh4,'g','LineWidth',2),
%
%         thetaH = BetaI_v(m+1)*pi/180;
%         Xh4 = - (RV(m+1)-dR)*R * phi * tan(thetaH);
%         Yh4 =   (RV(m+1)-dR)*R * sin(phi);
%         Zh4 =   (RV(m+1)-dR)*R * cos(phi);
%
%         plot3(Xh4,Yh4,Zh4,'g','LineWidth',2),
%     end

%     % Beta angle image helix for each spanwise section
%     for m = 1:Mp+1
%         RVW    = Rhub_oR^2/RV(m);
%         TANBW = TANBIV(1)*RV(1)/RVW;
%         phi = 0:0.01:2*pi;
%         thetaH = atand(TANBW)*pi/180;
%         Xh4 =   RVW*R * phi * tan(thetaH);
%         Yh4 = - RVW*R * sin(phi);
%         Zh4 =   RVW*R * cos(phi);
%
%         plot3(Xh4,Yh4,Zh4,'--r','LineWidth',2),
%     end

% ---------------------------------------------- Plot duct
    if Duct_flag == 1
        Duct_Ang=0;
        %ductPlot(vrRad,c,fo,to,alpha,ductRef)
        %shading interp
        colormap(jet)
        ductPlot(0.5*D,0.5*D,0,0,Duct_Ang*pi/180,0.5)
% %         ductPlot(0.5*D,0.5*D,-.04,.13,10*pi/180,0.5)
        %axis equal
    end

    view(-50,30)
    set(gca,'XTickLabel',{''},'YTickLabel',{''},'ZTickLabel',{''})
    set(gca,'TickLength',[0 0])
    xlabel(''), ylabel(''), zlabel(''), title('')
    grid off, axis off
%%
%     % ----------------------------------------------------- Save the image
%     view(2)
%     saveas(gcf,[filename,'_3D_Propeller_Image','1'],'jpg')
%
%     view(3)
%     saveas(gcf,[filename,'_3D_Propeller_Image','2'],'jpg')

end                                               % (END IF Make3Dplot_flag)


%%
% % SolidWorks_v14 and prior
% Make_SWrks_flag = 1;
% % --------------------------------------------------- Make SolidWorks files
% if Make_SWrks_flag
%     % Make SolidWorks Curve_n.txt files, with coordinates for a single
blade
%
%     % -------------------------------- Blade geometry:
%     % X3D(i,j,k) [m], X position in 3D space
%     % Y2D(i,j,k) [m], Y position in 3D space
%     % Z3D(i,j,k) [m], Z position in 3D space
%     %
%     % i = 1:Mp+1      % for each section along the span
%     % j = 1:2*Np      % for each point   along the upper and lower surfaces
%     % k = 1:Z         % for each blade
%
%     filename_SolidWorks = strcat(filename,'_SolidWorks.txt');
%     fid = fopen(filename_SolidWorks,'w');
%
%     % Prop Parameters at beginning of file
%     fprintf(fid,'%g, ' ,Np);
%     fprintf(fid,'%g, ' ,Mp);
%     fprintf(fid,'%g,\n',Z);
% %     fprintf(fid,'%g, ' ,Z);
% %     fprintf(fid,'%g,\n',NLE);
%
%     % Output curves defining each 2D section along the span
%     % for each section along the span
%     for i = 1:Mp+1
%         fprintf(fid,strcat('SectionCurve',num2str(i),',\n'));
%
%         % for each point along the suction and pressure surfaces
%         % (trailing edge -> leading edge -> trailing edge, close the curve)
% %         for j = [1:Np,Np+2:2*Np,1] % (2*Np   points) does not double
print the leading edge but does double print the trailing edge to close the
curves
% %         for j = [1:Np,Np+2:2*Np-1,1] % (2*Np-1 points) does not double
print the leading edge but does double print the trailing edge to close the
curves
%         for j = [1:Np,Np+2:2*Np] % (2*Np-1 points) does not double print
the leading edge
% %         for j = 1:2*(Np-1)+NLE % each curve contains (2*(Np-1)+NLE)
points
%             fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
%         end
%     end
%
%
%     % Make guide curves
%     n = 0;
%     % for 7 points along the chord
%     for j = [1 floor(Np/3) floor(2*Np/3) Np floor(4*Np/3) floor(5*Np/3)
2*Np];
% %     for j = [1 floor(1*(2*(Np-1)+NLE)/6) floor(2*(2*(Np-1)+NLE)/6) ...
% %                floor(3*(2*(Np-1)+NLE)/6) floor(4*(2*(Np-1)+NLE)/6) ...
% %                floor(5*(2*(Np-1)+NLE)/6) floor(6*(2*(Np-1)+NLE)/4)];
%         n = n + 1;
%
%
%         fprintf(fid,strcat('GuideCurve',num2str(n),',\n'));
% %         for i = 1:Mp  % for each section along the span except the last
one
%         for i = 1:Mp+1  % for each section along the span
%
% %             if i == Mp+1 && j == 2*Np
% %                 fprintf(fid,'%f,%f,%f',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
% %                 continue
% %             end
%
%
% %         plot3(X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1),'.b','markersize',20)
% %         pause,
%
%             fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
%         end
%
%     end
%
%
%
%     % Output duplicate trailing edge guide curves:
%     % Guide curve 1:
%     fprintf(fid,'TEGuideCurve1,\n');
%         j = 1;
%     for i = 1:Mp+1  % for each section along the span
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
%     end
%
%     % Guide curve 7:
%     fprintf(fid,'TEGuideCurve7,\n');
%         j = 2*Np;
%     for i = 1:Mp+1  % for each section along the span
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
%     end
%
%
%     % Output duplicate tip section profile:
%     i = Mp+1;
%     fprintf(fid,strcat('TipSectionCurve',num2str(i),',\n'));
%     % for each point along the suction and pressure surfaces
%     % (trailing edge -> leading edge -> trailing edge)
%     for j = [1:Np,Np+2:2*Np] % (2*Np-1 points) does not double print the
leading edge
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
%     end
%
%     % Output tip curves
%     for j = 1:Np-2
% %     for j = 1:(Np-1+(NLE-1)/2-1)
%         fprintf(fid,strcat('TipCurve',num2str(j),',\n'));
%         i=Mp+1;
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,   1+j,1),Y3D(i,   1+j,1),Z3D(i,
1+j,1));
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,2*Np-j,1),Y3D(i,2*Np-
j,1),Z3D(i,2*Np-j,1));
% %         fprintf(fid,'%f,%f,%f,\n',X3D(i,             1+j,1),Y3D(i,
1+j,1),Z3D(i,             1+j,1));
```

```
% %         fprintf(fid,'%f,%f,%f,\n',X3D(i,(2*(Np-1)+NLE)-j,1),Y3D(i,(2*(Np-
1)+NLE)-j,1),Z3D(i,(2*(Np-1)+NLE)-j,1));
%     end
%
%
%     % Output duplicate root section profile:
%     i = 1;
%     fprintf(fid,strcat('RootSectionCurve',num2str(i),',\n'));
%     % for each point along the suction and pressure surfaces
%     % (trailing edge -> leading edge -> trailing edge)
%     for j = [1:Np,Np+2:2*Np] % (2*Np-1 points) does not double print the
leading edge
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
%     end
%
%
%     % Output root curves
%     for j = 1:Np-2
% %     for j = 1:(Np-1+(NLE-1)/2-1)
%         fprintf(fid,strcat('RootCurve',num2str(j),',\n'));
%         i=1;
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,   1+j,1),Y3D(i,   1+j,1),Z3D(i,
1+j,1));
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,2*Np-j,1),Y3D(i,2*Np-
j,1),Z3D(i,2*Np-j,1));
% %         fprintf(fid,'%f,%f,%f,\n',X3D(i,             1+j,1),Y3D(i,
1+j,1),Z3D(i,             1+j,1));
% %         fprintf(fid,'%f,%f,%f,\n',X3D(i,(2*(Np-1)+NLE)-j,1),Y3D(i,(2*(Np-
1)+NLE)-j,1),Z3D(i,(2*(Np-1)+NLE)-j,1));
%     end
%
%
%     % Output trailing edge curves for each 2D section along the span
%     for i = 1:Mp+1
%         fprintf(fid,strcat('TECurve',num2str(i),',\n'));
%         j=1;
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
%         j=2*Np;
% %         j = 2*(Np-1)+NLE;
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
%     end
%
%     fclose(fid);
% end                                                   % (END IF Make_SWrks_flag)
%%
% SolidWorks_v18

Make_SWrks_flag = 1;
% --------------------------------------------------- Make SolidWorks files
if Make_SWrks_flag
    % Make SolidWorks.txt files, with coordinates for a single blade

    % --------------------------------- Blade geometry:
    % X3D(i,j,k) [m], X position in 3D space
    % Y2D(i,j,k) [m], Y position in 3D space
    % Z3D(i,j,k) [m], Z position in 3D space
    %
    % i = 1:Mp+1      % for each section along the span
    % j = 1:2*Np      % for each point   along the upper and lower surfaces
    % k = 1:Z         % for each blade

    filename_SolidWorks = strcat(filename,'_SolidWorks.txt');
    fid = fopen(filename_SolidWorks,'w');

    % Prop Parameters at beginning of file
    fprintf(fid,'%g, ' ,Np);
    fprintf(fid,'%g, ' ,Mp);
    fprintf(fid,'%g,\n',Z);

    % Output curves defining each 2D section along the span
    % for each section along the span
    for i = 1:Mp+1
        fprintf(fid,strcat('SectionCurve',num2str(i),',\n'));

        % for each point along the suction and pressure surfaces
        % (trailing edge -> leading edge -> trailing edge, close the curve)
        for j = [1:Np,Np+2:2*Np-1,1] % (2*Np-1 points) does not double print
the leading edge
            fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
        end
    end


    % Make guide curves
    n = 0;
    % for 7 points along the chord
    for j = [1 floor(Np/3) floor(2*Np/3) Np floor(4*Np/3) floor(5*Np/3) 2*Np-
1];
        n = n + 1;

        fprintf(fid,strcat('GuideCurve',num2str(n),',\n'));
        for i = 1:Mp+1  % for each section along the span
            fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
        end

    end

%     % Output duplicate trailing edge guide curves:
%     % Guide curve 1:
%     fprintf(fid,'TEGuideCurve1,\n');
%         j = 1;
%     for i = 1:Mp+1  % for each section along the span
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
%     end
%
%     % Guide curve 7:
%     fprintf(fid,'TEGuideCurve7,\n');
%         j = 2*Np;
%     for i = 1:Mp+1  % for each section along the span
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
%     end
```

```
    % Output duplicate tip section profile:
    i = Mp+1;
    fprintf(fid,strcat('TipSectionCurve',num2str(i),',\n'));
    for j = [1:Np,Np+2:2*Np-1,1] % (2*Np-1 points) does not double print the
leading edge
        fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
    end

    % Output tip curves
    for j = 1:Np-2
        fprintf(fid,strcat('TipCurve',num2str(j),',\n'));
        i=Mp+1;
        fprintf(fid,'%f,%f,%f,\n',X3D(i,   1+j,1),Y3D(i,   1+j,1),Z3D(i,
1+j,1));
        fprintf(fid,'%f,%f,%f,\n',X3D(i,2*Np-j,1),Y3D(i,2*Np-j,1),Z3D(i,2*Np-
j,1));
    end


    % Output duplicate root section profile:
    i = 1;
    fprintf(fid,strcat('RootSectionCurve',num2str(i),',\n'));
    % for each point along the suction and pressure surfaces
    % (trailing edge -> leading edge -> trailing edge)
    for j = [1:Np,Np+2:2*Np-1,1] % (2*Np-1 points) does not double print the
leading edge
        fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
    end


    % Output root curves
    for j = 1:Np-2
        fprintf(fid,strcat('RootCurve',num2str(j),',\n'));
        i=1;
        fprintf(fid,'%f,%f,%f,\n',X3D(i,   1+j,1),Y3D(i,   1+j,1),Z3D(i,
1+j,1));
        fprintf(fid,'%f,%f,%f,\n',X3D(i,2*Np-j,1),Y3D(i,2*Np-j,1),Z3D(i,2*Np-
j,1));
    end


%     % Output trailing edge curves for each 2D section along the span
%     for i = 1:Mp+1
%         fprintf(fid,strcat('TECurve',num2str(i),',\n'));
%         j=1;
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
%         j=2*Np;
%         fprintf(fid,'%f,%f,%f,\n',X3D(i,j,1),Y3D(i,j,1),Z3D(i,j,1));
%     end

    fclose(fid);
end                                                   % (END IF Make_SWrks_flag)
%%
```

```
% ---------------------------------------------------------- Make Rhino files
% Modified: 9/25/09 by Jordan Stanway and Brenden Epps
%
% This code makes a script that you can run in Rhino.  Here are the steps:
%   1) At the Command: prompt, type "ReadCommandFile"
%           -- Locate the script file, e.g. OpenProp_RhinoProp.txt
%   2) When the Document Properties window opens, set:
%           -- Model Units: meters
%           -- Absolute tolerance: 0.00001
%           -- Relative tolerance: 0.1
%           -- Angle    tolerance: 0.1
%   3) Watch as Rhino reads in all the points, makes each section, fills
%      the tip section, lofts the remaining sections, joins the surfaces,
%      makes Z blades from the key blade, and makes the hub
%   4) If your propeller does not loft or join automatically, then try
%      increasing or decreasing the tolerance values.
%
if Make_Rhino_flag
    % Make _RhinoBlade.txt, with coordinates for a single blade and
    % commands to make Z blades

    %%%%%%fprintf(fid,'!_SetActiveViewport Perspective\n');

    % In order for the surface to loft correctly, you probably will
    % need to manually set the "absolute precision" of Rhino to be
    % "10^-5 units" and manually change the "model units" to meters.
    % Note: the coordinates output from OpenProp are in meters.
    % This command should pause the script fro
    %%%%%%fprintf(fid,'_DocumentPropertiesPage Units \n');

    % Define Rhino curve type (choose one)
    % curve_cmd   = 'Curve \n';
    %%%%%%curve_cmd   = 'InterpCrv \n';

    % Compute where the blade tip should be
    tip_x = -rake(end) - R*pchip(XR,skew0,1)*(pi/180);
    tip_y =         R*sind(pchip(XR,skew0,1));
    tip_z =         R*cosd(pchip(XR,skew0,1));
    tip = [tip_x, tip_y, tip_z];
   % Initialize file
  filename_Rhino = strcat('0','_RhinoProp.txt');
    fid = fopen(filename_Rhino,'w');

    for i = 1:Mp+1 % For each section along the span

        % For each point along the upper and lower surfaces:
        for j = [1:Np,Np+2:2*Np] % (2*Np-1 points) does not double print the
leading edge
            fprintf(fid,'%.9f\t%.9f\t%.9f\n',X3D(i,j),Y3D(i,j),Z3D(i,j));  %
print to file with 9 decimal places
        end

        % If the first and last points in the section are identical, then
        % do nothing, else close the curve by adding another point the
        % same as the first one.
        if strcmp(sprintf('%.9f\t%.9f\t%.9f\n',X3D(i,1)  ,Y3D(i,1)
,Z3D(i,1)),...

sprintf('%.9f\t%.9f\t%.9f\n',X3D(i,end),Y3D(i,end),Z3D(i,end)))
            % disp(sprintf('%i start and end are identical, not adding point
to close', i));
        else
            fprintf(fid,'%.9f\t%.9f\t%.9f\n',X3D(i,1),Y3D(i,1),Z3D(i,1));
        end
        % Close the file:
    fclose(fid);
    file = sprintf('%.0f.txt',i);
    filename_Rhino = strcat(file);
    fid = fopen(filename_Rhino,'w');
    end

    fclose(fid);
end
% Extrude the tip section curve to the "tip" point
    %fprintf(fid,'SelNone\n');
    %fprintf(fid,'SelLast\n');
    %fprintf(fid,'ExtrudeCrv Mode=ToPoint \n');
    %fprintf(fid,'%.9f\t%.9f,%\t9f\n',tip(1),tip(2),tip(3));
    %fprintf(fid,'enter\n');

    % Loft the other sections to the tip section
    %fprintf(fid,'SelNone\n');
    %fprintf(fid,'SelClosedCrv\n');
    %fprintf(fid,'-Loft Type=Tight Simplify=None \n');
    %fprintf(fid,'enter\n');
    %fprintf(fid,'enter\n');
    %fprintf(fid,'enter\n');
    %fprintf(fid,'SelNone\n');
    %fprintf(fid,'SelSrf\n');
    %fprintf(fid,'Join\n');
    %fprintf(fid,'SelNone\n');
    %fprintf(fid,'Zoom All Extents\n');
    %fprintf(fid,'enter\n');


    % ---------- Commands to make Z blades:
    %fprintf(fid,'SelPolysrf\n');
    %fprintf(fid,'Rotate3D\n');
    %fprintf(fid,'0,0,0\n');
    %fprintf(fid,'1,0,0\n');
    %fprintf(fid,'Copy=Yes\n');
    % copy blades
    %for k=2:Z
    %    fprintf(fid,'%f\n',(k-1)*(360/Z));
    %end
    %fprintf(fid,'enter\n');


    % ----------- Commands to make hub
    %fprintf(fid,'Circle Vertical 0,0,0 \n');
    %fprintf(fid,'%f \n',Rhub);
    %fprintf(fid,'0,0,%f\n',Rhub);
    %fprintf(fid,'0,%f,0\n',Rhub);
    % to choose direction of the circle
        % ** this doesn't seem to work all the time... :-(
    %fprintf(fid,'SelNone\n');
    %fprintf(fid,'SelLast\n');
    %fprintf(fid,'ExtrudeCrv BothSides=Yes Cap=Yes DeleteInput=Yes \n');
    %Lhub = 2*R;
    %fprintf(fid,'%f \n',Lhub);
    %fprintf(fid,'Zoom All Extents\n');
    %fprintf(fid,'enter\n');




                                            % (END IF Make_Rhino_flag)
%%
% ----------------------------------------------- Make OpenProp_Geometry.txt
filename_geometry = strcat(filename,'_Geometry.txt');
fid = fopen(filename_geometry,'w');

fprintf(fid,'\t\t\t\t\t %s \n\n',filename_geometry);
fprintf(fid,'\t\t\t\t\t Propeller Geometry Table\n\n');
fprintf(fid,'Date and time: %s\n\n',Date_string);

fprintf(fid,'Propeller Diameter \t = %.4f m\n',    D);
fprintf(fid,'Number of Blades \t = %.0f\n',        Z);
fprintf(fid,'Propeller Speed \t = %.0f RPM\n',     N);
fprintf(fid,'Propeller Hub Diameter \t = %.4f m\n',Dhub);

fprintf(fid,['Meanline  Type: ',Meanline,'\n']);
fprintf(fid,['Thickness Type: ',Thickness,'\n']);

% if Meanline==1
%     fprintf(fid,['Meanline Type:  NACA a=0.8\n');
% elseif Meanline==2
%     fprintf(fid,'Meanline Type:  Parabolic\n');
% end
%
% if Thickness==1
%     fprintf(fid,'Thickness Type: NACA 65A010\n\n');
% elseif Thickness==2
%     fprintf(fid,'Thickness Type: Elliptical\n\n');
% elseif Thickness==3
%     fprintf(fid,'Thickness Type: Parabolic\n\n');
% end

fprintf(fid,' \n');
fprintf(fid,' \n');

fprintf(fid,' r/R\t P/D\t Skew\t Xs/D\t  c/D\t  f0/c\t  t0/c\n');
for i = 1:Mp
    fprintf(fid, '%.4f\t %.4f\t %.4f\t %.4f\t %.4f\t %.4f\t %.4f\n'...
    ,RC(i),PoD(i),skew(i),rake(i)/D,CoD(i),f0oc(i),t0oc(i));
```

```
end

fprintf(fid,' \n');
fprintf(fid,'\nr/R \t [ ], radial position of control points / propeller
radius.\n');
fprintf(fid,'P/D \t [ ], section pitch / diameter.\n');
fprintf(fid,'c/D \t [ ], section chord-length / diameter.\n');
fprintf(fid,'fo/C \t [ ], section camber / section chord-length.\n');
fprintf(fid,'to/C \t [ ], section thickness / section chord-length.\n');

fclose(fid);

% =============================== END Determine Propeller Geometry Function
% =========================================================================
```