Numerically-Based Ducted Propeller Design

Using Vortex Lattice Lifting Line Theory

by

John M. Stubblefield

B.S., (1993) United States Naval Academy
M.S., (1998) Naval Postgraduate School

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

Master of Science in Naval Architecture and Marine Engineering

and

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2008

Signature of
Author_____
Department of Mechanical Engineering
May 9, 2008

Certified
by_____
Patrick J. Keenan
Professor of Naval Architecture
Thesis Supervisor

Certified
by_____
Richard W. Kimball
Thesis Supervisor

Accepted
by_____
Lallit Anand
Professor of Mechanical Engineering
Chairman, Departmental Committee on Graduate Students

Numerically-Based Ducted Propeller Design Using Vortex Lattice Lifting Line Theory

by
John M. Stubblefield

Submitted to the Department of Mechanical Engineering on May 9, 2008
in Partial Fulfillment of the Requirements for the Degrees of
Master of Science in Naval Architecture and Marine Engineering
and
Master of Science in Mechanical Engineering

Abstract

This thesis used vortex lattice lifting line theory to model an axisymmetrical-ducted propeller with no gap between the duct and the propeller. The theory required to model the duct and its interaction with the propeller were discussed and implemented in *Open-source Propeller Design and Analysis Program* (OpenProp). Two routines for determining the optimum circulation distribution were considered, and a method based on calculus of variations was selected. The results of this model were compared with the MIT Propeller Lifting Line Program (PLL) output for the purpose of validation.

Ducted propellers are prevalent in modern marine propulsion systems, and the application of this technology continues to expand. The theory associated with ducted propellers applies to a wide-range of devices which include azimuth thrusters, pumpjets, and tidal turbines. Regardless of the application, engineers need tools such as OpenProp to design these devices for their expected operating conditions. OpenProp is an open source MATLAB®-based suite of propeller numerical design tools. Previously, the program only designed open propellers. The code developed in this thesis extended OpenProp's capability to be able to design a propeller within an axisymmetrical duct.

Thesis Supervisor: Patrick J. Keenan
Title: Professor of Naval Architecture

Thesis Supervisor: Richard W. Kimball

Acknowledgements

The author thanks the following individuals for all of their support and assistance with this thesis:

Professor Rich Kimball for all his wisdom and guidance during not only the thesis process but also during the two classes he taught. His classes and teaching method were among the best experienced at MIT.

CAPT Patrick Keenan for the leadership and direction he provided both during the thesis process and throughout the entire course 2N program. His course was also one of the best the author experienced at MIT.

Brenden Epps for his assistance with the MATLAB® coding involved with modeling ducted propellers.

And most of all, my family for all of their support and understanding during my time at MIT. While Christine and Jake experienced the entire adventure, Hank arrived just in time to see the thesis completed.

Table of Contents

Acknowledgements 3

List of Figures 6

1. Introduction 8

2. Overview of the Propeller Design code: OpenProp 11

3. Theoretical Foundation 16

3.1 Ducted Propeller Theory 16

3.2 Vortex Ring Theory and Algorithm 22

3.3 Circumferential Mean Velocity 25

3.4 Circulation Optimization 32

4. Implementation and Validation 37

5. Conclusions and Recommendations 47

5.1 Conclusions 47

5.2 Recommendations for further work 48

References 49

Appendix A. Duct Theory MATLAB® Code 51

A.1 ductVort.m 51

A.2 ductThrust.m 53

A.3 vRing.m 55

A.4 vpfDuct.m 57

A.5 CMV.m 58

A.6 ductPlot.m 60

Appendix B. Mathematical Functions MATLAB® Code 63

B.1 Q2half.m 63

B.2 Q2Mhalf.m 63

B.3 Heuman.m 64
Appendix C. Variational Optimization Routine MATLAB® Code 65
C.1 Coney.m 65
C.2 Align_wake.m 71
Appendix D. Test Case Setup Data 72

List of Figures

Figure 1-1: Ducted propeller and associated vortex system representation from Coney (1)..... 9

Figure 2-1: OpenProp's parametric analysis input GUI 12
Figure 2-2: Efficiency diagrams produced by OpenProp's parametric analysis 12
Figure 2-3: OpenProp's single propeller design input GUI..... 13
Figure 2-4: OpenProp's graphical out of key propeller parameters 14
Figure 2-5: Blade and propeller representations from OpenProp..... 14

Figure 3-1: Diagram of the ducted propeller vortex system from Coney (1) 17
Figure 3-2: Optimum circulation distributions for ducted and open propellers 19
Figure 3-3: Optimization circulation distribution for a zero gap ducted propeller..... 20
Figure 3-4: *CMV.m* Validation Representative Circulation..... 28
Figure 3-5: Heuman's Lambda Function..... 31
Figure 3-6: OpenProp τ=0.80 test case using the Lerbs-based optimization routine 33

Figure 4-1: OpenProp v2 single propeller design GUI with duct parameters 37
Figure 4-2: OpenProp v2 required duct parameters..... 37
Figure 4-3: Sample rendering of ducted propeller produced in the test cases 39
Figure 4-4: OpenProp v2 algorithm propeller results using PLL circulation 40
Figure 4-5: OpenProp v2 algorithm duct results using PLL circulation..... 41
Figure 4-6: Efficiency versus thrust ratio (τ) comparison between OpenProp v2 and PLL..... 42
Figure 4-7: OpenProp v2 graphical output for test case with $\tau = 0.8$..... 43
Figure 4-8: OpenProp v2 comparison with PLL for $\tau = 0.8$ 43
Figure 4-9: OpenProp v2 comparison with PLL for $\tau = 0.8$ (duct ring velocities) 44
Figure 4-10: OpenProp v2 comparison with PLL for $\tau = 1.0$ 45
Figure 4-11: OpenProp v2 comparison with PLL for $\tau = 1.2$ 46

Figure D-1: Test case parameters 72
Figure D-2: OpenProp v2 input GUI for test cases..... 72
Figure D-3: PLL current settings for inviscid and viscid test cases 73

Figure D-4: PLL overall input file for test cases 73
Figure D-5: Sample PLL output summary for test case run 74

1. Introduction

Ducted propellers are widely used in marine propulsion systems for a variety of reasons. As shown by Kort in 1934, ducted propellers can achieve higher efficiency particularly in slow, heavily loaded applications such as tugboats and ocean platforms. While this efficiency gain is lost at higher speeds due to the viscous drag from the duct, there are many more reasons that a designer might choose a ducted propeller or derivative such as a pump jet or water jet instead of a traditional open propeller. Several of these reasons are listed below.

- Greater power density. A ducted propeller can produce more thrust than an open propeller of the same size. If ship geometry limits the size of the propulsor, a ducted propeller might be the best option.
- Physical protection. A duct provides protection for the propeller blades.
- Cavitation reduction. As described below, a decelerating duct can be used to increase the static pressure at the propeller which reduces or eliminates cavitation.
- Simplicity. Because ducted propulsors often incorporate directional control (vectored thrust via a trainable duct, steerable nozzle, ect), rudders can be eliminated.
- Expanded operational environment. Because the duct provides protection and can reduce or eliminate other appendages such as a rudder, ducted propellers can improve shallow water operation. This is especially true for a water jet since only the inlet must be submerged for proper operation.

Another promising use of ducted propeller technology is the tidal turbine market. With the world looking increasingly towards renewable energy sources, harnessing the power of the ocean is of great interest and importance. Development of tidal turbines directly leverages the research associated with ducted propellers. The geometries of both problems are essentially the same, and the design process of each involve an optimization involving thrust and torque. For ducted propulsors, the goal is provide a certain amount of thrust while minimizing the required torque a ship's engines must provide through a shaft or electric motor. The optimization is essentially reversed for a tidal turbine that extracts power from the ocean via a turbine generator. In this case, the designer desires to maximize torque and minimize thrust. While this thesis develops

the design tool for a traditional ducted propeller, the concepts and most of the code are directly applicable to tidal turbines.

Ducted propellers and associated derivatives (electric drive, pods, azimuth thrusters, water jets, pumpjets, etc) will continue to play an important role in ship propulsion and will have an expanding role in renewable energy efforts as described above. For this reason, engineers need tools to design the devices for specific operating conditions.

Following the approach presented by Coney in (1), this thesis developed the MATLAB® algorithms necessary to model a duct and its interactions with a propeller. These algorithms were then integrated into the existing OpenProp propeller design program. The model assumed that there was no gap between the duct and propeller. The duct was represented by an image system of vorticity and a system of ring vorticies at the radius of the duct cylinder (Figure 1-1).

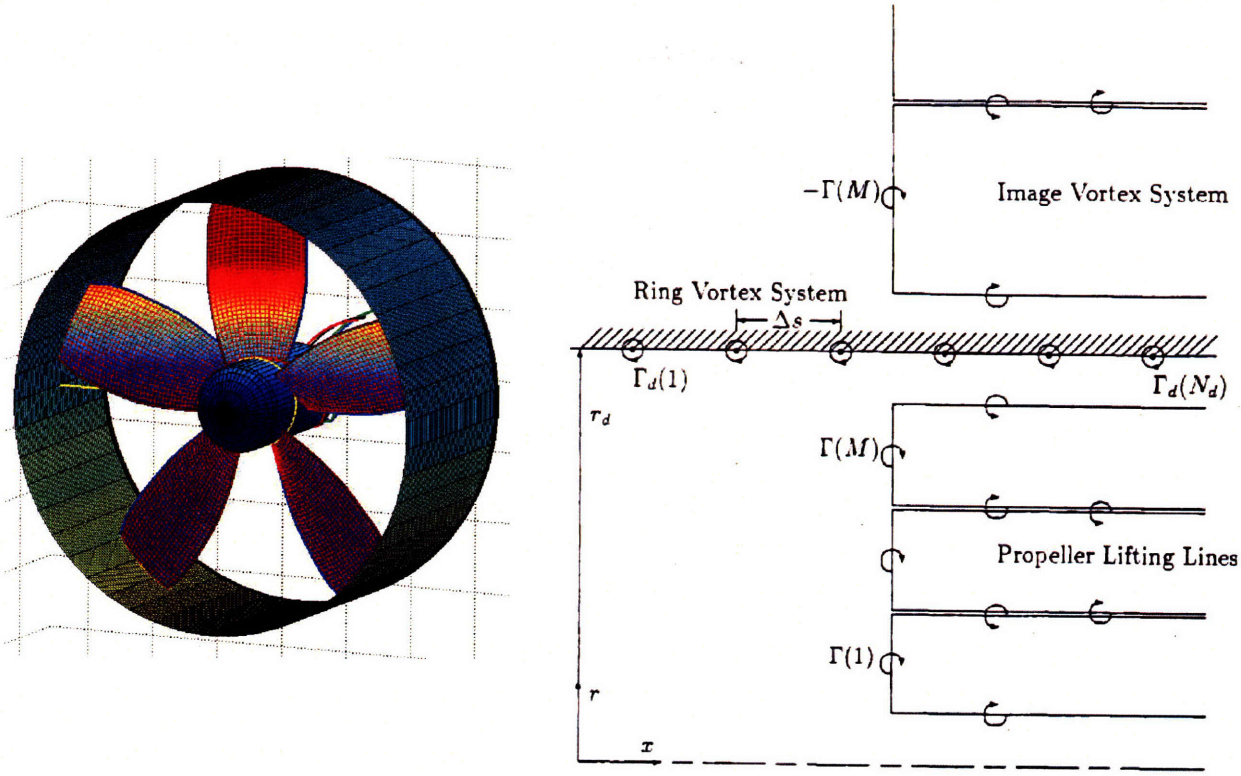

Figure 1-1: Ducted propeller and associated vortex system representation from Coney (1)

The image system modeled the nonaxisymmetric effect of the duct while the ring vorticies provided an estimate of the resulting duct force. The influence functions calculated for the radial lifting line control points included the effects from the duct image system, and the inflow was modified by the effect of the duct ring vorticies. A variational optimization routine was employed to determine the optimum circulation distribution for the lifting line. The model accounted for viscous drag but duct thickness was neglected. To the greatest extent possible, this thesis used the notation presented in (1).

2. Overview of the Propeller Design code: OpenProp

Open-source Propeller Design and Analysis Program (OpenProp)[1] is an open source MATLAB®-based suite of propeller numerical design tools. This program is an enhanced version of the *MIT Propeller Vortex Lattice Lifting Line Program* (PVL) developed by Professor Justin Kerwin at MIT in 2001. OpenProp v1.0, originally titled MPVL, was written in 2007 by Hsin-Lung Chung and Kate D'Epagnier and is described in detail in (2) and (3). Two of its main improvements versus PVL are its intuitive graphical user interfaces (GUIs) and greatly improved data visualization which includes graphic output and three-dimensional renderings.

OpenProp was designed to perform two primary tasks: parametric analysis and single propeller design. Both tasks begin with a desired operating condition defined primarily by the required thrust, ship speed, and inflow profile. The parametric analysis produces efficiency diagrams for all possible combinations of number of blades, propeller speed, and propeller diameter for ranges and increments entered by the user. The efficiency diagrams are then used to determine the optimum propeller parameters for the desired operating conditions given any constraints (e.g. propeller speed or diameter) specified by the user. Figure 2-1 shows the input GUI for the parametric analysis routine, and Figure 2-2 shows the efficiency diagrams produced by that routine.

[1] Throughout this thesis, OpenProp refers to the design program in general. OpenProp v1.0 (version 1.0) refers to the original version of OpenProp which was developed for open propellers only, and OpenProp v2.0 refers to the version associated with this thesis which includes the capability to model ducted propellers.

	Min	Max	Increment
Number of Blades	3	6	1
Propeller Speed (RPM)	50	200	50
Propeller Diameter (m)	2	5	0.5

☑ Hub Image Flag (Check for YES)

112024	Required Thrust (N)
6.675	Ship Velocity (m/s)
0.6	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
1031	Water Density (kg/m^3)

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

Figure 2-1: OpenProp's parametric analysis input GUI

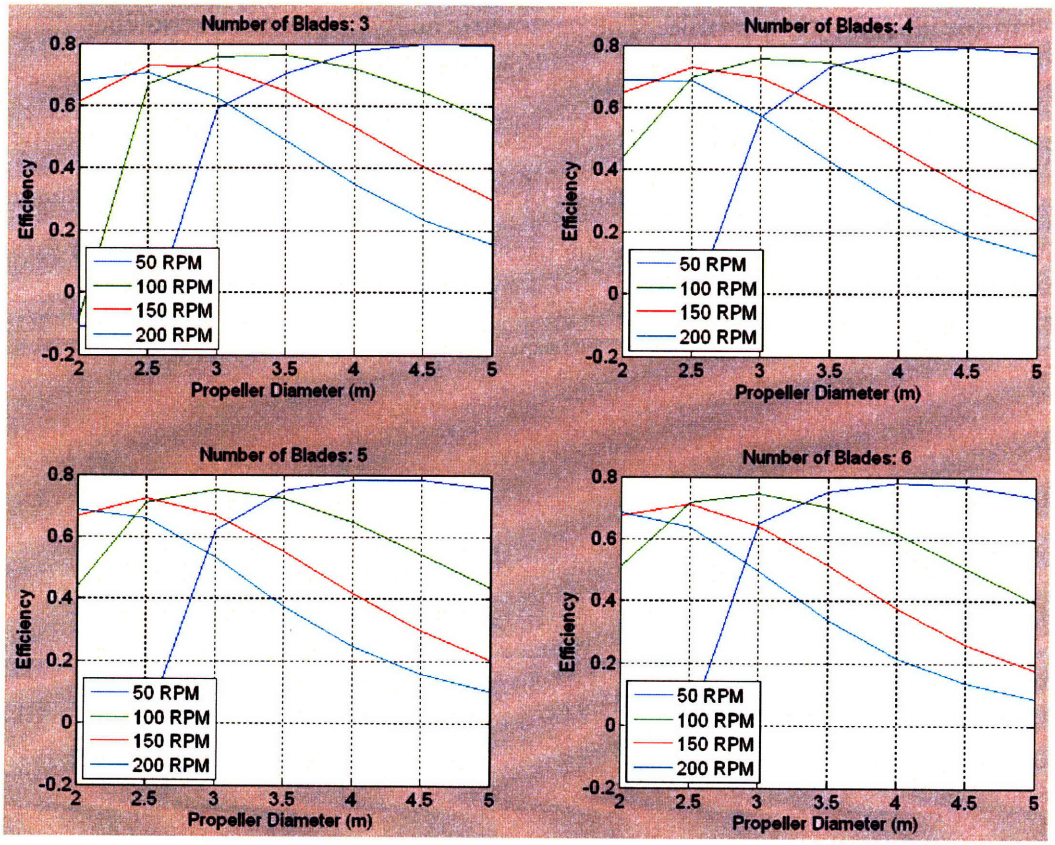

Figure 2-2: Efficiency diagrams produced by OpenProp's parametric analysis

The single propeller design routine produces a complete propeller design for the desired operating condition and defined propeller parameters (number of blades, propeller speed, propeller diameter, hub diameter, etc). Figure 2-3 shows the input GUI for the single propeller design routine. OpenProp's graphical out of key propeller parametersFigure 2-4 shows the graphical output of key propeller parameters, and Figure 2-5 shows blade profiles and complete three-dimensional representation of the propeller.

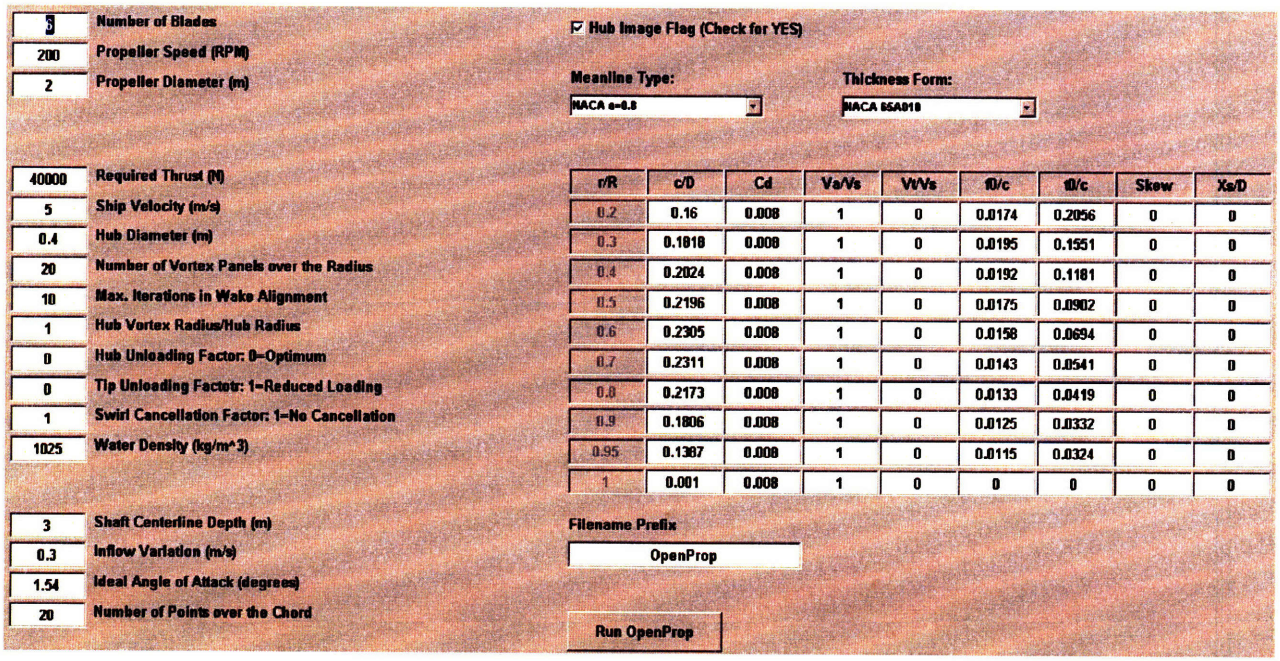

Figure 2-3: OpenProp's single propeller design input GUI

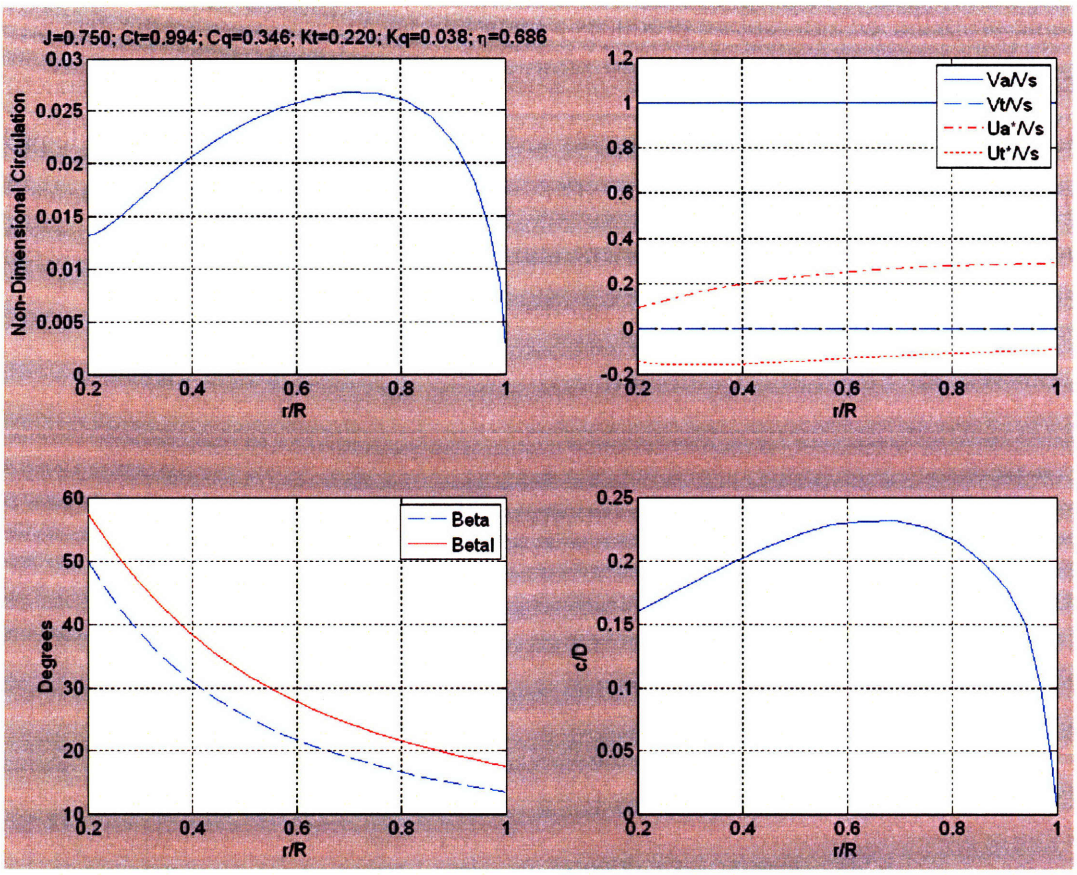

Figure 2-4: OpenProp's graphical out of key propeller parameters

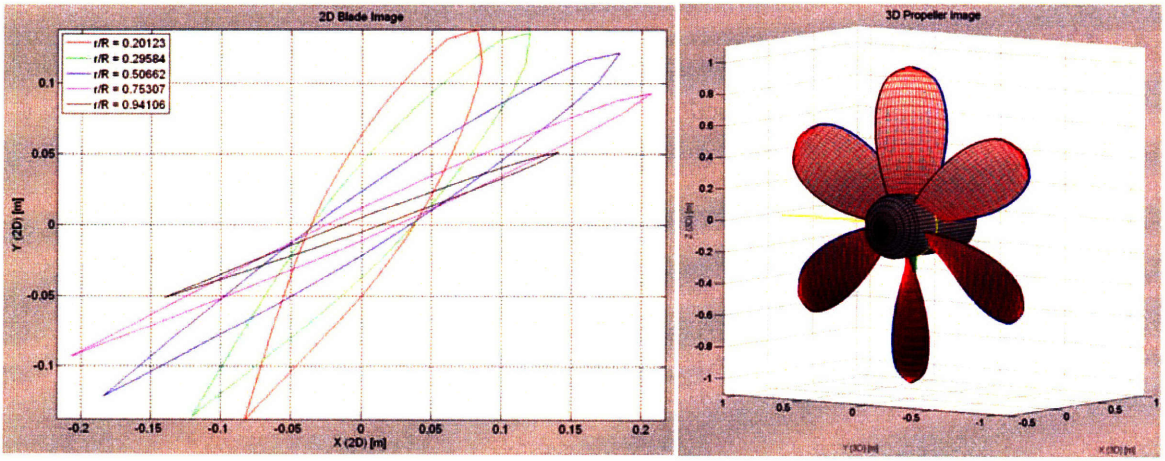

Figure 2-5: Blade and propeller representations from OpenProp

OpenProp was developed to serve as an open source code for propeller design. While it is currently a tool that is only used in the initial design phase, it is a base program that can be continually expanded to perform detailed design and analysis of sophisticated marine propulsors and turbines. Extending OpenProp to include a duct was the main focus of this thesis.

3. Theoretical Foundation

3.1 Ducted Propeller Theory

Ducted propellers are generally divided into two types: accelerating and decelerating nozzles. The accelerating duct or "Kort" nozzle has been widely used since Kort showed in 1934 that this type of duct produces a positive thrust and can increase efficiency in heavily loaded applications such as tugboats. It was also shown that the optimum diameter for a ducted propeller is smaller than that for an open propeller. Because of this, accelerating ducts are sometimes used when increased thrust is needed from a propeller whose size is constrained by the ship's characteristics or operating conditions.

A decelerating duct increases the static pressure at the propeller and is used to reduce cavitation. A reduction in cavitation lowers the noise generated by a propeller and reduces erosion of the blades.

From momentum theory, the ideal efficiency, η_I, of a ducted propeller is given in Equation 3-1 where τ is the thrust ratio (Equation 3-2), C_T is the thrust coefficient (Equation 3-3), ρ is the fluid density, V_S is the ship speed, and D is the propeller diameter (4).

$$\eta_I = \frac{2}{1+\sqrt{1+\tau C_T}} \qquad \textbf{3-1}$$

$$\tau = \frac{T_P}{T} = \frac{Propeller\ Thrust}{Total\ Thrust} \qquad \textbf{3-2}$$

$$C_T = \frac{T}{\frac{1}{2}\rho V_S^2 \frac{\pi}{4} D^2} \qquad \textbf{3-3}$$

As the thrust ratio is lowered, the duct produces more of the required total thrust and the ideal efficiency increases. However, when finite blade effects are considered a penalty is paid for the increased axial velocity at the propeller plane and efficiency decreases after reaching a maximum at approximately $\tau = 0.9$ (1). Additionally, since the thrust ratio is multiplied by the thrust coefficient, a large thrust coefficient is required in order to realize a significant efficiency gain. Hence, ducted propellers are commonly used in heavily loaded situations such as tugboats.

As introduced above, this thesis modeled the duct as an infinite cylinder by adding an image vortex system to the vortex lattice representing the lifting line and adding a system of ring vortices to account for duct forces and the axisymmetric mean inflow modification by the duct. Figure 3-1 shows the ducted propeller vortex system used by Coney and implemented in OpenProp. The propeller coordinate system[2] is shown in the lower left-hand corner of Figure 3-1.

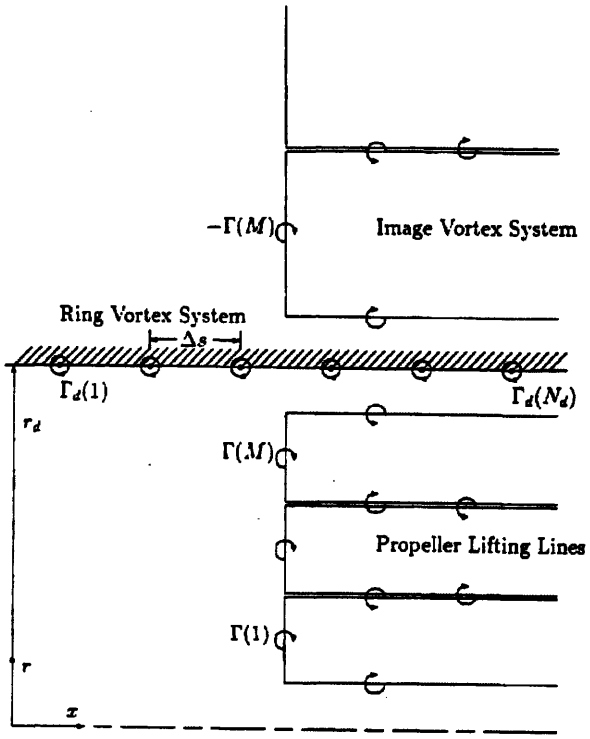

Figure 3-1: Diagram of the ducted propeller vortex system from Coney (1)

[2] The propeller coordinate system used in OpenProp and this thesis has positive x in the downstream direction.

The image system method was essentially the same as the method used originally in OpenProp to model the hub. The underlying assumption here was that the image system will approximately satisfy the condition of zero radial velocity at the cylinder. The image system used a constant pitch angle based on the tip trailing vortex on the lifting line.

The radii of the image vortices and their associated $\tan(\beta_i)$ were determined using Equations 3-4 and 3-5 where r_i is the radius of the duct image shed vortex trailer, r_d is the radius of the duct cylinder, r is the radius of the helical trailing vortex shed by the lifting line, β_i is the hydrodynamic pitch angle, and the subscript "tip" refers to the M^{th} or last shed vortex trailer on the lifting line.

$$r_i = \frac{r_d^2}{r} \tag{3-4}$$

$$tan(\beta_i)_{r_i} = tan\,(\beta_i)_{tip} * \frac{r_{tip}}{r_i} \tag{3-5}$$

The radial and tangential influence functions from the duct image were added to the radial lifting line influence functions as follows:

$$[\bar{u}_a^*(n,m)]_{total} = \bar{u}_a^*(n,m) + [\bar{u}_a^*(n,m)]_{duct\ image} \tag{3-6}$$

$$[\bar{u}_t^*(n,m)]_{total} = \bar{u}_t^*(n,m) + [\bar{u}_t^*(n,m)]_{duct\ image} \tag{3-7}$$

These influence functions were used in the variational optimization routine described below to determine the optimum circulation distribution for ducted propellers.

As the gap between the duct and propeller tip is decreased, the optimum circulation distribution becomes more tip loaded. At the limiting case of zero tip gap, the circulation reaches its maximum value at the tip. Figure 3-2 shows the OpenProp v2 and PLL results for the optimum

circulation distribution for neutrally loaded ($\tau = 1.0$) ducted propellers ($C_T = 1.2,\ J_S = 0.6$) represented by a duct image system. The OpenProp v2 results were essentially identical to those obtained from PLL.

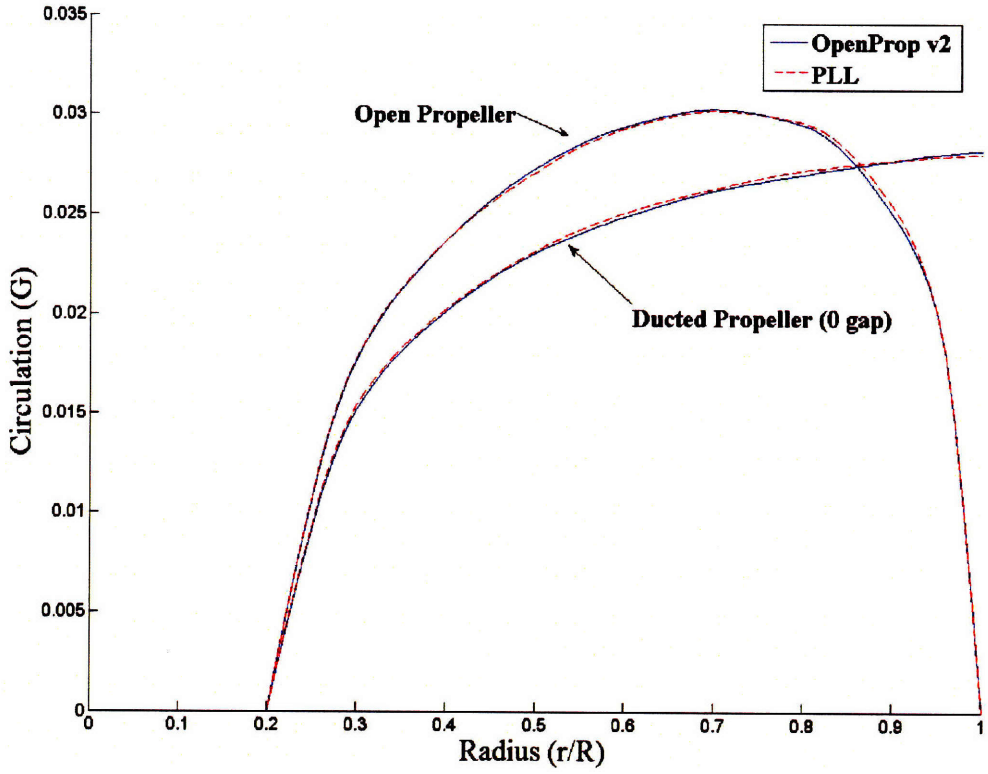

Figure 3-2: Optimum circulation distributions for ducted and open propellers

In (1), Coney compared the duct image method with a more sophisticated panel method representation of the duct and determined that the optimum circulation distribution agreed very well. Since a panel method would be too computationally intensive for an early stage design tool such as OpenProp, the results were extremely fortuitous. Coney's results (1) are shown in Figure 3-3.

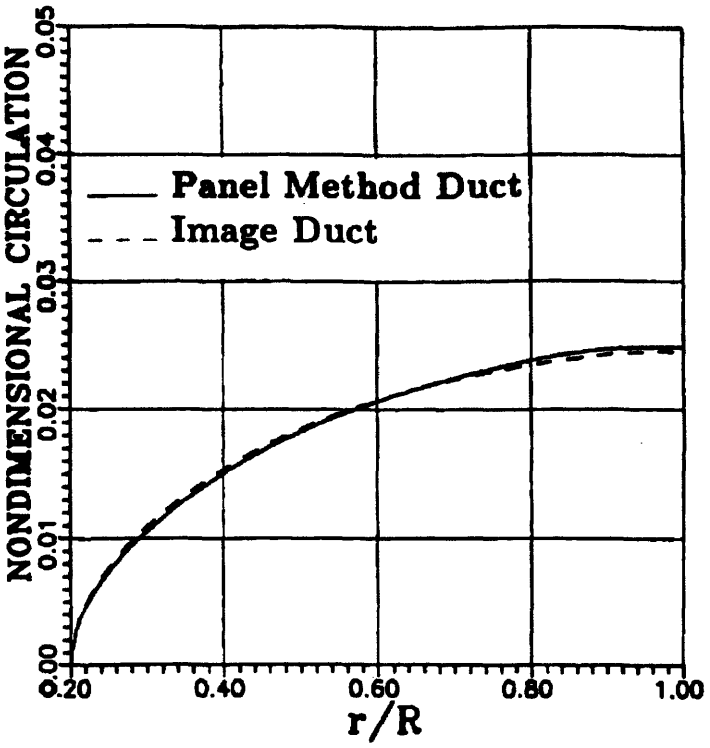

Figure 3-3: Optimization circulation distribution for a zero gap ducted propeller as determined from image and panel representations of the duct from Coney (1).

The ring vortex system was used to estimate the duct force and the mean modification of the inflow at the propeller lifting line. The length of the system represented the duct chord length, c_d, which was chosen to be equal to the propeller radius, R. The ring vortices of the system were spaced (Δs) evenly at approximately the same constant interval used to discretize the lifting line. Furthermore, the system was positioned such that the lifting line was located at the duct mid-chord and between ring vortices. The strengths of ring vorticies were calculated to represent a NACA a=0.8 meanline over the length of the duct.

The axial component of the velocity that the ring vortex system induced on the lifting line is shown in Equation 3-8 where N_d is the number of ring vortices used to represent the duct. Γ_d is the strength of the n^{th} ring vortex. $\bar{u}^*_{a_{pd}}(i,n)$ is the velocity induced by the n^{th} vortex ring of unit strength on the i^{th} propeller lifting line control point. The algorithm used to calculate the induced velocities from the vortex rings is discussed in the next section.

$$u_{a_{pd}}(i) = \sum_{n=1}^{N_d} \bar{u}^*_{a_{pd}}(i,n)\Gamma_{\mathrm{d}}(\mathrm{n})$$

3-8

The axial force on each vortex ring was calculating by using the Kutta-Joukowski law. As shown in Equation 3-9, the total thrust, T_d, on the duct was calculated by summing the axial forces on the vortex rings and adding a viscous drag by using a two-dimensional airfoil sectional drag coefficient, C_{D_d}.

$$T_d = -2\rho\pi r_d \sum_{n=1}^{N_d} \left\{[V_r(r_d) + u_r(n,r_d)]\Gamma_{\mathrm{d}}(\mathrm{n}) + [V_a(r_d) + u_a(n,r_d)]\mathrm{c_d}C_{D_d}\right\}\Delta s$$

3-9

Here, $V_r(r_d)$ and $V_a(r_d)$ are the radial and axial inflow velocities, respectively, at the duct radius, r_d. $u_r(n,r_d)$ and $u_a(n,r_d)$ are the radial and axial components of the circumferential mean velocity induced on the n^{th} ring vortex by the propeller. Calculation of the circumferential mean velocities is discussed below.

The theory above was implemented in OpenProp v2 using two new MATLAB® functions, *ductVort.m* and *ductThrust.m* (Appendix A). *ductVort.m* was used to determine the vortex ring system that represented the duct and to calculate the induced velocity on the lifting line from that system. *ductThrust.m* was used to calculate the total duct thrust coefficient derived from equation 3-9 that was required in the optimization routine discussed below. It also scaled the duct vortex ring system circulation so that the duct provides the thrust specified by the thrust ratio.

3.2 Vortex Ring Theory and Algorithm

OpenProp uses vortex rings to model the duct's axisymmetric mean modification of the inflow to the propeller and to calculate the thrust provided by the duct. A vortex ring is a vortex filament that forms a circle of radius r'. Applying the Biot-Savart law, Kuchemann and Weber (5) derived the influence from a vortex ring in terms of complete elliptic integrals, *K(k)* and *E(k)*. *K(k)* denotes the complete elliptic integral of the first kind, and *E(k)* denotes the complete elliptic integral of the second kind. The argument, *k*, is known as the *modulus*.

As derived, the vortex ring is in the *Y-Z* plane and is located at $x = 0$. Due to the inherent symmetry of a ring, all points contained on a circle in the *Y-Z* have the same induced velocity. Therefore, Kuchemann and Weber derived equations for the axial and radial components of a vortex ring with radius $r'$and strength Γ for a point located on a circle of radius *R* in a plane *X* units from and parallel to the *Y-Z* plane. Additionally, the center of the circle is on the *x-axis*. The axial induced velocity is given in Equation 3-10 with the arguments and variables shown in Equations 3-11 and 3-12. The radial induced velocity is given in Equation 3-13.

$$v_{yx}(x,r) = \frac{\Gamma}{2\pi r'} \frac{1}{\sqrt{x^2 + (r+1)^2}} \left\{ K(k) - \left[1 + \frac{2(r-1)}{x^2 + (r-1)^2} \right] E(k) \right\}$$

3-10

$$k^2 = \frac{4r}{x^2 + (r+1)^2}$$

3-11

$$x = \frac{X}{r'} \quad , \quad r = \frac{R}{r'}$$

3-12

$$v_{yr}(x,r) = \frac{\Gamma}{2\pi r'} \frac{-x}{r\sqrt{x^2 + (r+1)^2}} \left\{ K(k) - \left[1 + \frac{2r}{x^2 + (r-1)^2} \right] E(k) \right\}$$

3-13

For *OpenProp*, these equations were implemented in a function named *vRing.m* (Appendix A). *vRing.m* was successfully validated using Tables 1 and 2 contained in (6). *vpfDuct.m* (Appendix A) is a velocity prediction function for a duct modeled only with vortex rings (i.e. no thickness). A future improvement will include source rings to model duct thickness.

MATLAB® contains a defined function, *ELLIPKE*, for calculating the elliptic integrals of the first and second kind. However, the argument for this function is not the modulus, *k*. Instead, *ELLIPKE* uses *m* which is known as the *parameter*. The modulus, m, is related to the parameter, k, as shown in Equation 3-14.

$$k^2 = m$$

3-14

The induced velocity can be calculated at any location except a point on the vortex ring (i.e. the velocity a vortex ring induces on itself). Future versions of OpenProp may require the duct algorithm to calculate the velocity a vortex ring induces on itself. Two options were explored to overcome this singularity problem.

First, the average of the induced velocity of selected locations on a sphere surrounding a point on the vortex ring was calculated. This was done for consecutively smaller spheres. The result did not converge, but rather grew without bound as the sphere size was reduced.

The second attempt to overcome the singularity problem consisted of discretizing the vortex ring into vortex filaments. The influence of each vortex filament on the desired point was summed. The vortex filament that contained the desired point was not included in this summation as it was assumed that the point was on this filament, and it is well know that there is no influence on any

point collinear with a vortex filament. This method also failed to converge. As the discretation increased, so did the resulting induced velocity.

Table 1 gives the result for the self-induced axial velocity of a vortex ring of unit radius and strength.

Discretation	Axial Induced Velocity
10^1	0.2256
10^2	0.4072
10^3	0.5904
10^4	0.7737
10^5	0.9569
10^6	1.1401
10^7	1.3234
10^8	Not Enough Memory

Table 1: Vortex Ring Self-Induced Axial Velocity

The failure to overcome this singularity did not affect the duct algorithm used in OpenProp v2.0 because the self-influence of a ring vortex[3] is not required. If it is required in a future version, the error can be mitigated by increasing the discretation of the duct (i.e. the number of vortex rings used to model the duct is increased).

[3] Real vortices do not have this singularity problem because of their viscous core dissipation.

3.3 Circumferential Mean Velocity

When a propulsor includes more than one component, it becomes necessary to calculate the velocities that one component induces on another. For rotating components such as the propeller, the time-averaged induced velocities are used and are equal to the circumferential mean velocities calculated in the rotating reference frame of the component. Formulas for calculating the tangential, axial, and radial induced velocities induced from a horseshoe vortex are presented below. The formulas are derived from Coney (1) and Hough and Ordway (7), and the notation most closely matches Coney (1).

From Kelvin's theorem, Equation 3-15 gives the tangential circumferential mean velocity, $\bar{u}_t^*$, induced on a control point at radius $r_c(m)$ of another component from a horseshoe vortex of strength Γ with lattice points at radii $r_v(p-1)$ and $r_v(p)$. Equation 3-16 defines the parameter S which directly relates to whether or not the control point is in the slipstream. x_f is the axial distance from the horseshoe vortex lattice point to the control point with positive x_f being in the downstream (i.e. positive x-axis) direction.

$$\bar{u}_t^*(m,n) = \begin{cases} 0, & S > 0, \quad -\infty \le x_f \le \infty \\ 0, & S \le 0, \quad x_f < 0 \\ \dfrac{-Z\Gamma}{2\pi r_c(m)}, & S < 0, \quad x_f > 0 \end{cases}$$

3-15

$$S = (r_v(p-1) - r_c(m))(r_v(p) - r_c(m))$$

3-16

That is, the tangential velocity vanishes everywhere outside of the slipstream of the horseshoe vortex and is proportional to $\Gamma(p)/r_c(m)$ inside the slipstream. The tangential circumferential mean velocity induced by both the bound and trailing vorticity can be found from the above equations.

The bound vortices on a radial lifting line only induce tangential circumferential mean velocities. The axial and radial circumferential mean velocities induced from the trailing vortices must now be calculated. Hough and Ordway (7) used Fourier analysis to derive formulas for the induced velocities in terms of the Heuman Lambda function and Legendre functions of the second kind and half integer order. As Coney noted in (1), these can be thought of as the velocities induced by a propeller with an infinite number of blades, and since the circumferential mean velocities are the average of the sum of local induced velocities along a circle, Equations 3-17 and 3-18 can be applied to calculate the axial and radial circumferential mean velocities. The constant C_1 is defined in Equation 3-19 where $Q_{\pm\frac{1}{2}}(q)$ are the Lengendre functions of the second kind and half integer order and $\Lambda_0(\varphi,\kappa)$ is the Heuman's Lambda function with the amplitude, φ, and modulus, κ, as the arguments. The arguments for the Legendre and Heuman Lambda functions are given in Equations 3-20, 3-21, and 3-22.

$$\bar{u}_a(m,p) = \Gamma\frac{Z}{\pi r_v(p)\tan(\beta_v(\mathrm{p}))}C_1$$

3-17

$$\bar{u}_r(m,p) = \Gamma\frac{Z}{\pi\sqrt{r_c(m)r_v(p)tan(\beta_v(p))}}Q_{\frac{1}{2}}$$

3-18

$$C_1 = \begin{cases} \pi + \dfrac{x_f}{2\sqrt{r_c(m)r_v(p)}}Q_{\frac{1}{2}}(q) + \dfrac{\pi}{2}\Lambda_0(\varphi,\kappa), & r_c(m) \le r_v(p) \\ \dfrac{x_f}{2\sqrt{r_c(m)r_v(p)}}Q_{-\frac{1}{2}}(q) - \dfrac{\pi}{2}\Lambda_0(\varphi,\kappa), & r_c(m) > r_v(p) \end{cases}$$

3-19

$$q = 1 + \frac{x_f^2 + \left(r_c(m) - r_v(p)\right)^2}{2r_c(m)r_v(p)}$$

3-20

$$\varphi = \sin^{-1}\left[\frac{x_f}{\sqrt{x_f^2 + \left(r_c(m) - r_v(p)\right)^2}}\right]$$

3-21

$$\kappa = \sqrt{\frac{4r_c(m)r_v(p)}{x_f^2 + \left(r_c(m) + r_v(p)\right)^2}}$$

3-22

As previously stated, the bound vorticity of each horseshoe vortex only induces a tangential velocity. Therefore, equations 3-17 and 3-18 can be applied as shown in Equations 3-23 and 3-24 to calculate the axial and radial velocities induced from a horseshoe vortex that is used in representing a radial lifting line.

$$\bar{u}_a^*(m,p) = \bar{u}_a(m,p) - \bar{u}_a(m,p+1)$$

3-23

$$\bar{u}_r^*(m,p) = \bar{u}_r(m,p) - \bar{u}_r(m,p+1)$$

3-24

For *OpenProp*, the above algorithms were implemented in the *CMV.m* function (Appendix A). This function was validated with Tables 1 and 2 in (7). The test case used the representative blade circulation distribution shown in Figure 3-4 and assumed that the helical path of the trailing vortex system was determined solely by the incoming free stream and propeller rotation (i.e. the advance angle, β, was used instead of the hydrodynamic advance angle, β_i).

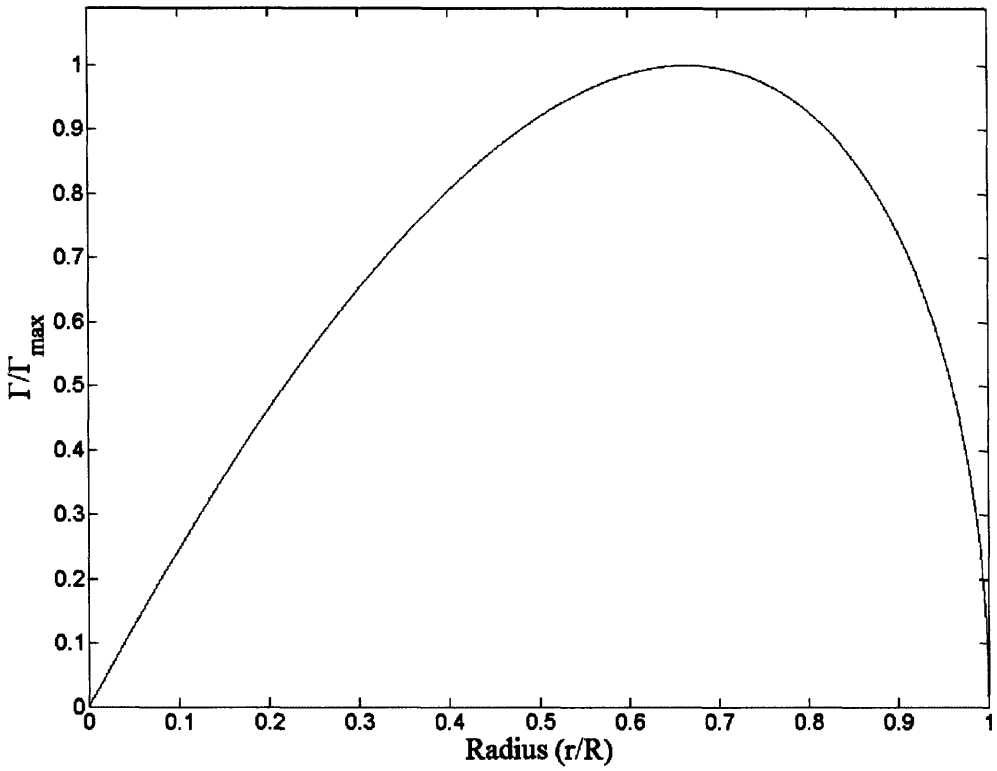

Figure 3-4: *CMV.m* Validation Representative Circulation

CMV.m required two Legendre functions and the Heuman Lambda function. As these functions were not available as class functions in MATLAB®, they were written and validated for this thesis. They are described below.

Q2half.m (Appendix B) computes the Legendre function of the second kind and positive half order of the argument q in accordance with (8) as shown in Equation 3-25.

$$Q_{\frac{1}{2}}(q) = q\sqrt{\frac{2}{q+1}}\,K\left(\sqrt{\frac{2}{q+1}}\right) - [2(q+1)]^{\frac{1}{2}}\,E\left(\sqrt{\frac{2}{q+1}}\right) \qquad \textbf{3-25}$$

K(k) denotes the complete elliptic integral of the first kind, and *E(k)* denotes the complete elliptic integral of the second kind with the *modulus*, *k*, as the argument. This was implemented in

MATLAB® using *ELLIPKE* with the *parameter, m,* as the argument which is shown in Equation 3-26.

$$m = k^2 = \frac{2}{q+1}$$

3-26

This function was validated with Table XIII in (9). For example:

$$Q_{\frac{1}{2}}(1.5) = Q2half(1.5) = 0.39175$$
$$Q_{\frac{1}{2}}(2.7) = Q2half(2.7) = 0.134035$$
$$Q_{\frac{1}{2}}(8.4) = Q2half(8.4) = 0.0229646$$

Q2Mhalf.m (Appendix B) computes the Legendre function of the second kind and minus half order of the argument q in accordance with (8) as shown in Equation 3-27.

$$Q_{-\frac{1}{2}}(q) = \sqrt{\frac{2}{q+1}}\, K\left(\sqrt{\frac{2}{q+1}}\right)$$

3-27

As before, *K(k)* denotes the complete elliptic integral of the first kind with the *modulus, k,* as the argument. This was implemented in MATLAB® using *ELLIPKE* with the *parameter, m,* as the argument as described above for *Q2half.m.*

This function was validated with Table XIII in (9). For example:

$$Q_{-\frac{1}{2}}(1.5) = Q2half(1.5) = 2.01891$$
$$Q_{-\frac{1}{2}}(2.7) = Q2half(2.7) = 1.38958$$
$$Q_{-\frac{1}{2}}(8.4) = Q2half(8.4) = 0.768523$$

Heuman.m (Appendix B) computes Heuman's Lambda function of the arguments φ (*amplitude*) and α (*modular angle*) in accordance with (8) as shown in Equation 3-28.

$$\Lambda_0(\varphi\backslash\alpha) = \frac{2}{\pi}\left\{K(\alpha)E(\varphi\backslash\tfrac{\pi}{2} - \alpha) - [K(\alpha) - E(\alpha)]F(\varphi\backslash\tfrac{\pi}{2} - \alpha)\right\}$$

3-28

$$\alpha = \sin^{-1}(k)$$

3-29

K(α) denotes the complete elliptic integral of the first kind with the modular angle, α, as the argument. $F(\varphi\backslash\frac{\pi}{2} - \alpha)$ denotes the incomplete elliptic integral of the first kind with the amplitude, φ, and complementary modular angle, $\frac{\pi}{2} - \alpha$, as the arguments. $E(\varphi\backslash\frac{\pi}{2} - \alpha)$ denotes the incomplete elliptic integral of the second kind. with the amplitude, φ, and complementary modular angle, $\frac{\pi}{2} - \alpha$, as the arguments.

K(α) was implemented in MATLAB® using *ELLIPKE* with the *parameter*, *m*, as the argument as shown in Equation 3-30.

$$m = k^2 = \sin(\alpha)^2$$

3-30

$F(\varphi\backslash\frac{\pi}{2} - \alpha)$ was implemented in MATLAB® using the imbedded Maple function *EllipticF* with the sine of the *amplitude*, sin(φ), and the *parameter*, *k*, as the arguments. Since the complementary modular angle is used for the incomplete elliptic integral in this case, the *parameter* is defined as shown in Equation 3-31.

$$k = sin\left(\frac{\pi}{2} - \alpha\right)$$

3-31

$E(\varphi \backslash \frac{\pi}{2} - \alpha)$ was implemented similarly in MATLAB® using the imbedded Maple function *EllipticE*.

Heuman.m was validated with (8).

Figure 3-5 was generated using *Heuman.m* and agrees with Figure 17.10 in (8).

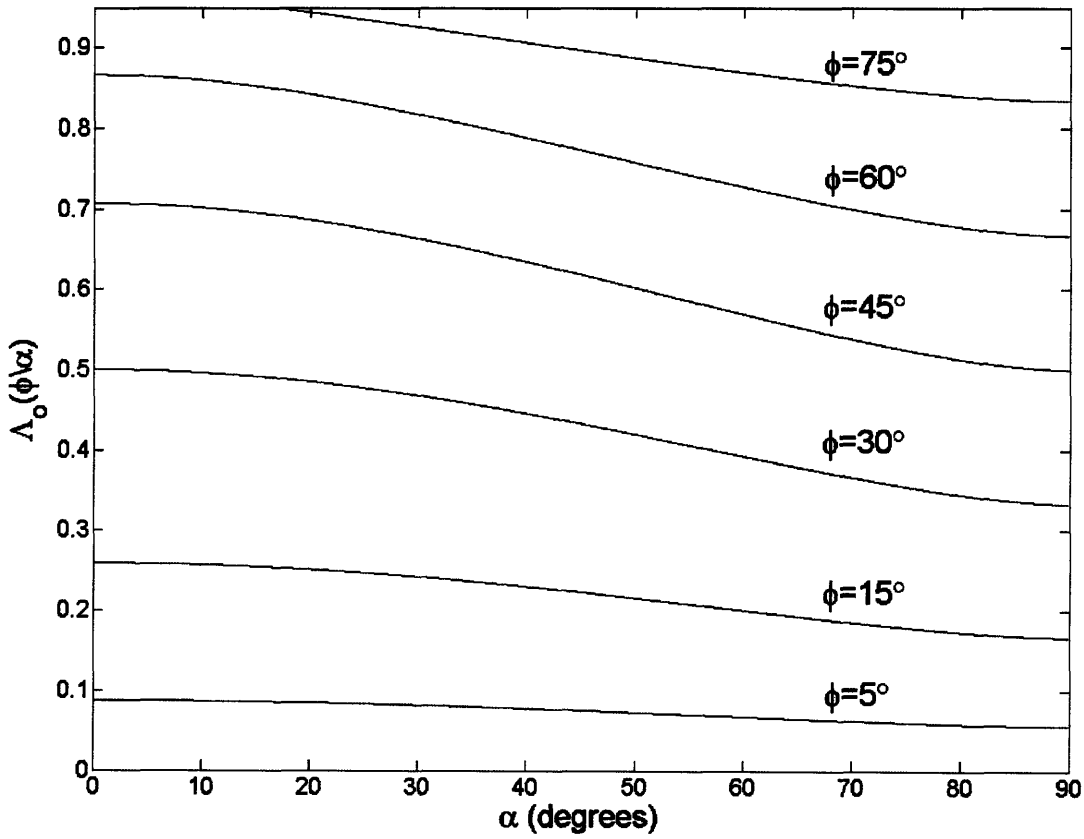

Figure 3-5: Heuman's Lambda Function

The sample calculations[4] given below agree with Table 17.8 in (8).

$$\Lambda_0(5^o \backslash 10^o) = Heuman(5^o \backslash 10^o) = 0.086495$$
$$\Lambda_0(45^o \backslash 60^o) = Heuman(45^o \backslash 60^o) = 0.569122$$
$$\Lambda_0(75^o \backslash 40^o) = Heuman(75^o \backslash 40^o) = 0.906056$$

[4] When executing *Heuman.m* in MATLAB, φ and α are entered in radians vice degrees.

3.4 Circulation Optimization

The goal of OpenProp's optimization routine is to calculate the radial distribution of circulation on the lifting line that minimizes torque for a given thrust. Also specified are the propeller diameter, number of blades, advance coefficient (J_S), and inflow velocity profile. OpenProp v1.0 uses the Lerbs criterion where $\tan(\beta_i)$ is obtained from $\tan(\beta)$ in terms of an unknown multiplicative factor (10). The optimization routine initially estimates the hydrodynamic pitch angle (β_i) based on the undisturbed flow angle (β) and the efficiency of the actuator disk. The system of equations represented by Equation 3-32 is then solved to obtain the optimum circulation:

$$\sum_{m=1}^{M} \left[\bar{u}_a(n,m) - \bar{u}_t(n,m)\tan(\beta(n))\right]\Gamma_m = V_a(n)\left(\frac{\tan(\beta_i(n))}{\tan(\beta(n))} - 1\right) \qquad n = 1, \ldots M$$

3-32

Using the circulation distribution, the vortex induced velocities on each panel of the lifting line are then solved. This allows the forces to be calculated, and the resulting thrust is compared with the desired thrust. The hydrodynamic pitch angle is then iteratively adjusted, and the process is repeated until the desired thrust is achieved (2).

This thesis integrated the duct model into the Lerbs-based optimization routine as follows:

- The influence functions included the effects of the duct image horseshoes.
- The inflow velocities were modified to include the effect of the duct vortex rings.
- The duct circulation was an entered value. The circumferential mean velocity induced by the propeller on the duct was added to the inflow, and the thrust produced by the duct was calculated using the Kutta-Joukowski law.
- The duct thrust was subtracted from the desired thrust.

Using PLL, several test cases were run for validation. In each case, the results indicated that the existing Lerbs-based optimization routine in OpenProp did not converge to the same result as PLL which uses a calculus of variations optimization routine. The results for the $\tau = 0.8$ case

for a 5-bladed, 10 foot diameter propeller with C_T=1.2 and J_S=0.60 is presented below. Viscous forces were ignored.

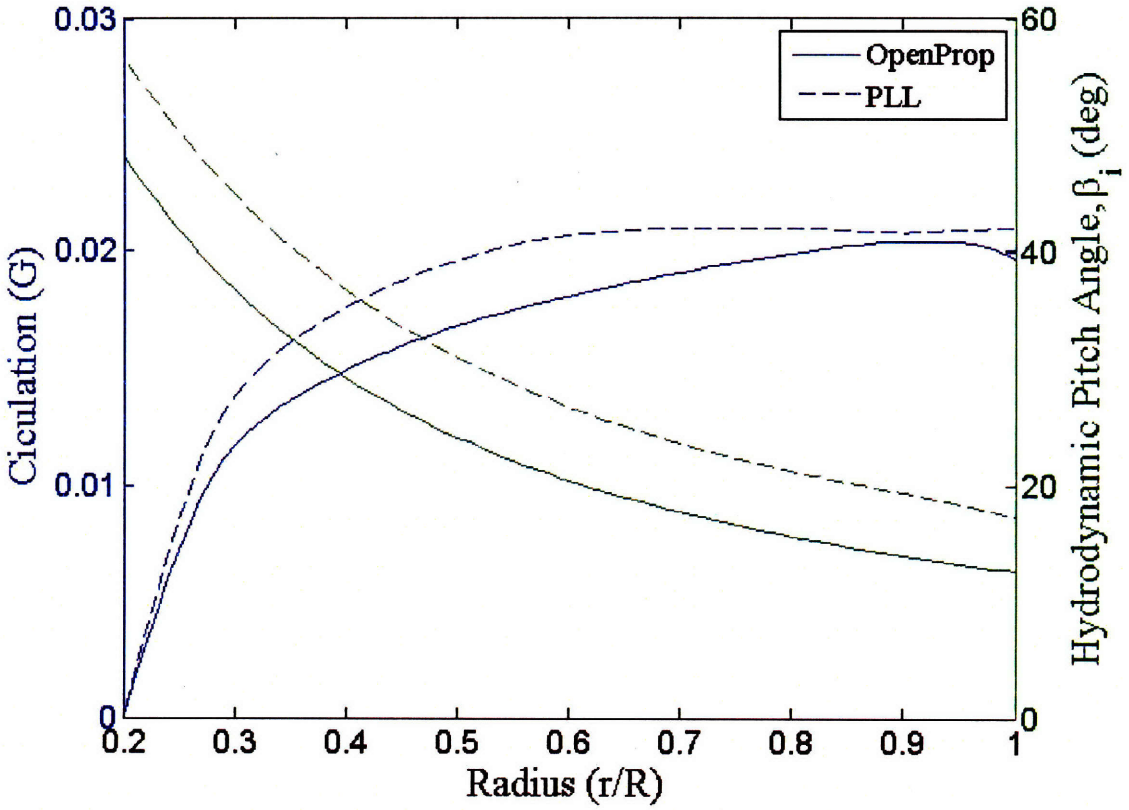

Figure 3-6: OpenProp τ=0.80 test case using the Lerbs-based optimization routine

The results showed that the Lerbs-based optimization routine in its current form was not extendable to a ducted propeller. Upon further consideration, this was logical given that the routine relied on the initial undisturbed flow angle, β, at the propeller plane but did include the effect of the duct in this initial determination of β. It may be possible to integrate a β iteration loop into the Lerbs-based routine, but that option was not pursued for this thesis.

Following the unsuccessful attempt with the Lerbs-based optimization routine, the variational optimization routine presented in (1) was integrated into OpenProp. This option was chosen for two reasons: 1) it is the proven routine used in PLL and 2) it is a more general optimizer that can be used for many different propulsor configurations. In this routine the wake geometry is frozen while the optimum circulation distribution is calculated such that torque,

$$Q = \rho Z \sum_{m=1}^{M} [V_a(m) + u_a^*(m)] r(m) \Gamma(m) \Delta r,$$

3-33

is minimized, subject to the constraint that the thrust,

$$T = \rho Z \sum_{m=1}^{M} [V_t(m) + \omega r(m) + u_t^*(m)] \Gamma(m) \Delta r,$$

3-34

has a prescribed value, T_r.

The auxiliary function $H = Q + \lambda(T - T_r)$ is formed, and its partial derivatives with respect to the unknown variables ($\Gamma(i)'s$ and λ) are set to zero as shown in equations 3-35 and 3-36 below. The Lagrange multiplier, λ, is an additional unknown variable and must be solved for along with the discrete circulation strengths, the $\Gamma(i)'s$.

$$\frac{\partial H}{\partial \Gamma(i)} = 0 \qquad for \qquad i = 1..M$$

3-35

$$\frac{\partial H}{\partial \lambda} = 0$$

3-36

Equations 3-35 and 3-36 form a nonlinear system of $M + 1$ equations with M unknown values of circulation and an unknown Lagrange multiplier. By assuming that the Lagrange multiplier is known where it forms quadratic terms with the circulation and that the tangential induced velocity, $u_t^*(m)$, is known, the solution to the nonlinear system of equations can be found by iteratively solving the following linear system of equations:

$$\frac{\partial H}{\partial \Gamma(i)} = 0 = V_a(i)r(i)\Delta r$$
$$+ \sum_{m=1}^{M} [\Gamma(m)\bar{u}_a^*(i,m)r(m)\Delta r + \Gamma(m)\bar{u}_a^*(m,i)r(i)\Delta r]$$
$$+\lambda[V_t(i) + \omega r(i)]\Delta r$$
$$+\grave{\lambda} \sum_{m=1}^{M} [\Gamma(m)\bar{u}_t^*(i,m)\Delta r + \Gamma(m)\bar{u}_t^*(m,i)\Delta r]$$
$$for \quad i = 1 \ldots M$$

3-37

$$T_r = \rho Z \sum_{m=1}^{M} [V_t(m) + \omega r(m) + u_t^*(m)]\Gamma(m)\Delta r,$$

3-38

For each iteration, the frozen Lagrange multiplier, $\grave{\lambda}$, in equation 3-37 and the tangential induced velocity in equation 3-38 take on values from the previous values. Furthermore, Coney determined that initially setting the induced velocities equal to zero and the Lagrange multiplier equal to -1 are suitable initial estimates of these quantities.

After the optimum circulation distribution is found, the wake is aligned with the velocities induced by that circulation distribution. The aligned wake is now frozen and the above process is repeated in order to determine a new optimum circulation distribution. By using this double iterative approach, velocities and forces consistent with moderately loaded lifting line theory can be obtained.

The above procedure was adapted to handle a ducted propeller. Additionally, an estimate of the effects of viscous drag was added. The following modifications were made:

- The effects of the duct image horseshoes were added to the influence functions in equation 3-37 as shown in equations 3-6 and 3-7.
- The effects of the duct ring vortices were included in calculating the induced velocities on the lifting line.

- The NACA a=0.8 meanline circulation distribution for the duct ring vortex system was scaled so that the duct produces the thrust specified by the thrust ratio.
- The duct thrust, T_d, calculated as shown in equation 3-9, was subtracted from the required thrust in equation 3-38. Additionally, an estimate of the propeller's viscous drag, $T_{viscous}$, was added to the required thrust. The modified required thrust is given in Equation 3-39.

$$T_r = \rho Z \sum_{m=1}^{M} [V_t(m) + \omega r(m) + u_t^*(m)]\Gamma(m)\Delta r + T_{viscous} - T_d \qquad \textbf{3-39}$$

The propeller's viscous drag was estimated using the two-dimensional airfoil sectional drag coefficients, $C_{D_v}(m)$, and the section chord lengths, $c(m)$ as shown in Equation 3-40.

$$T_{viscous} = \frac{1}{2}\rho Z \sum_{m=1}^{M} V^*(m)[V_a(m) + u_a^*(m)]c(m)C_{D_v}(m)\Delta r \qquad \textbf{3-40}$$

In OpenProp v2, the variational optimization routine described above was integrated as a MATLAB® function named *Coney.m*[5] (Appendix C). Within *Coney.m*, the *ductVort.m* and *ductThrust.m* functions discussed above were used to perform specific duct-related calculations.

[5] Brenden Epps wrote the initial version of *Coney.m*. The version of *Coney.m* implemented in OpenProp v2 was a modified and expanded version of his code.

4. Implementation and Validation

In order to enable OpenProp to design a ducted propeller, the variational optimization routine (*Coney.m*) and associated MATLAB® functions (*ductVort.m, ductThrust.m, vRing.m, CMV.m, Q2half.m, Q2Mhalf.m,* and *Heuman.m*) discussed above were integrated into the OpenProp code. The graphical user interfaces (GUIs) were updated to include the required parameters for a ducted propeller as shown in Figure 4-1.

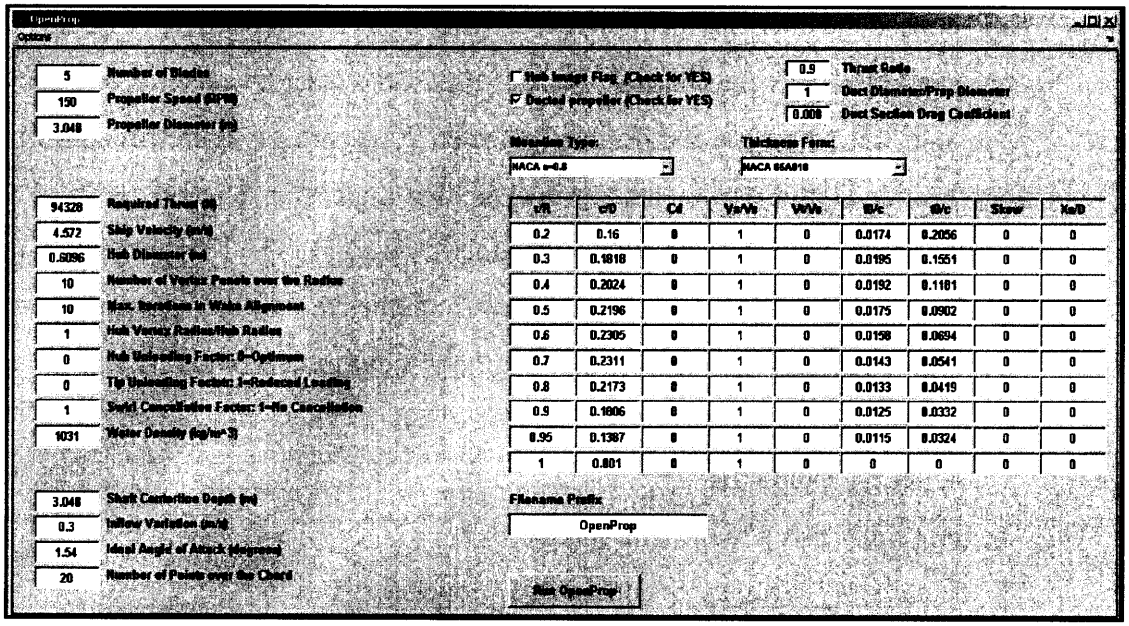

Figure 4-1: OpenProp v2 single propeller design GUI with duct parameters

The required duct parameters are located in the upper-right corner and are shown enlarged in Figure 4-2.

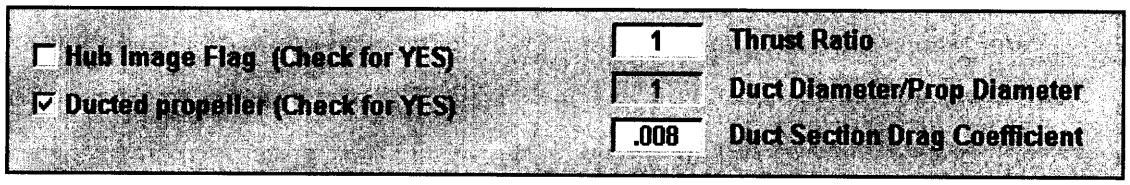

Figure 4-2: OpenProp v2 required duct parameters

If the user desires to design a propeller operating within a duct, the Ducted Propeller check box is selected and three parameters are required:

- Thrust Ratio: as given in equation 3-2, the thrust ratio specifies the portion of the desired thrust that the propeller will produce. Typical values of thrust ratio are 0.7 to 1.3. Values less than one result in an accelerating duct that produces a positive thrust while values greater than one result in a decelerating duct that produces a negative thrust which must be overcome by the propeller.
- Duct Diameter/Prop Diameter: determines the size of the duct in terms of the propeller diameter. This value must be equal to or greater than one as a duct must at least be as large as the propeller which it surrounds. This parameter determines the gap between the duct and the propeller tip which has a major influence on the optimum circulation distribution. For this thesis, the Duct Diameter/Prop Diameter parameter was set to one and disabled since only the zero gap case was considered.
- Duct Section Drag Coefficient: specifies the two-dimensional airfoil sectional drag coefficient which is used to estimate the duct's viscous drag. This parameter must be equal to or greater than zero. A value of zero implies the inviscid case where viscous effects are neglected. A typical value for the viscous case is 0.008.

OpenProp v2 assumes the following:

- The duct surrounds a radial lifting line.
- The duct chord length, c_d, is equal to the propeller radius, R.
- The duct is positioned such that the mid-chord is located at the lifting line.
- The circulation distribution of the duct vortex ring system represents a NACA a=0.8 meanline.

In order to demonstrate the capability of OpenProp v2 and provide validation via a PLL comparison, several test cases were completed using a 5-bladed ducted propeller with optimum circulation distribution operating at $J_S = 0.60$ and $C_T = 1.20$. The duct had zero thickness, and there was zero gap between the duct and the propeller. For viscid runs, a sectional drag coefficient of 0.008 was used for both the duct and the propeller. Appendix D contains all of the

run settings for both OpenProp v2 and PLL[6]. Figure 4-3 shows a sample rendering of the ducted propellers designed in the test cases.

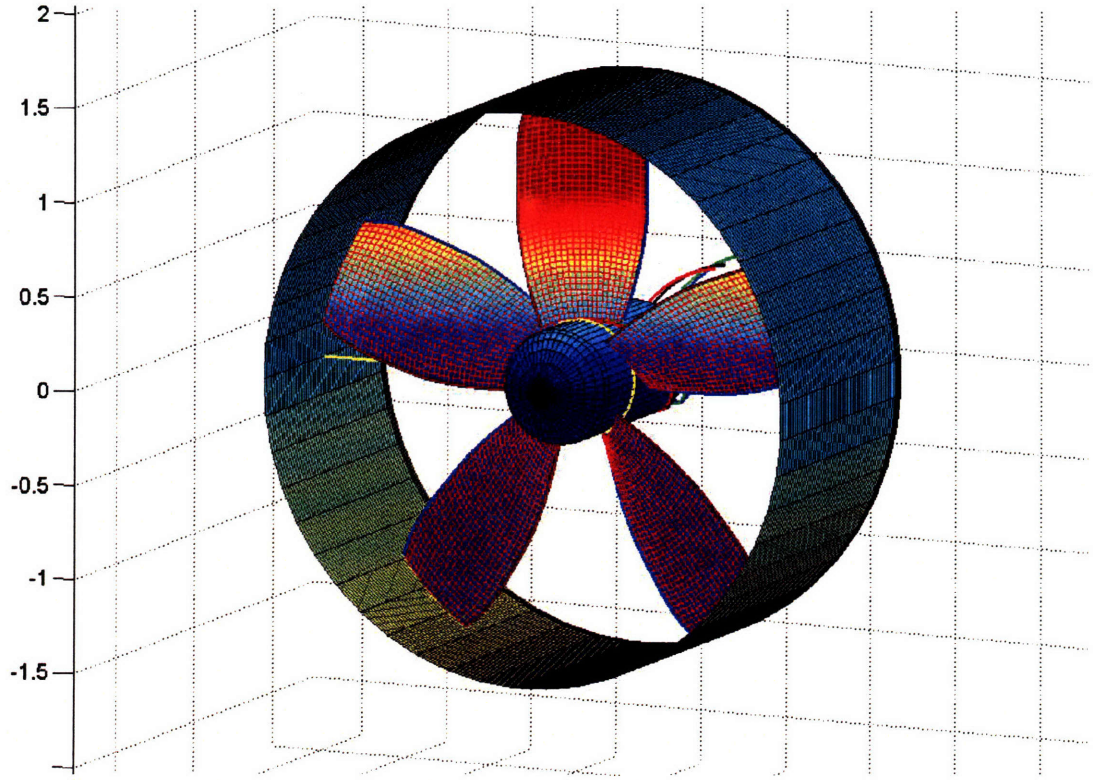

Figure 4-3: Sample rendering of ducted propeller produced in the test cases

First, to ensure the various functions and routines in the optimization routine were working correctly, the circulation from a $\tau = 0.8$ PLL run was fed into OpenProp v2. Figure 4-4 and Figure 4-5 show given the same circulation distribution, OpenProp v2 calculated the same induced velocities as PLL. This included the total induced velocities on the propeller control points (UA^*/V_s *and* UT^*/V_s), the axial velocities induced by the duct rings on the propeller control points, and the velocities on the duct rings by the propeller lifting line.

[6] All runs for both PLL and OpenProp v2 used 10 vortex panels. Neither PLL or OpenProp v2 would run successfully with greater than 20 vortex panels. PLL would crash for an unknown reason, and OpenProp v2 experienced an error in the Heuman.m function.

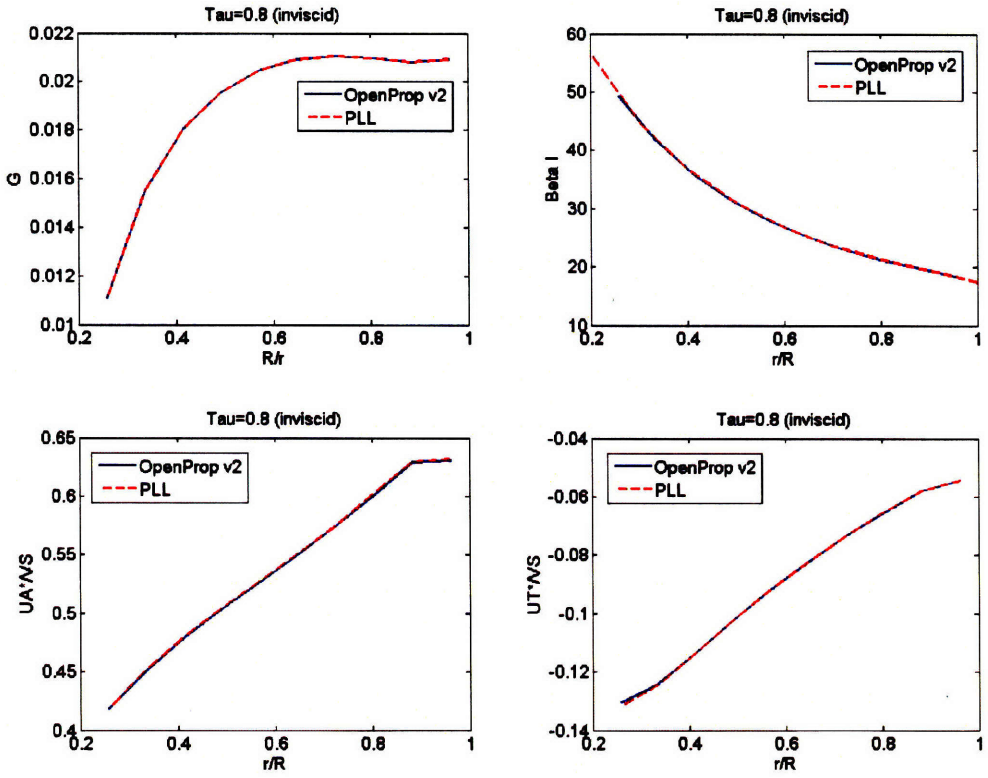

Figure 4-4: OpenProp v2 algorithm propeller results using PLL circulation

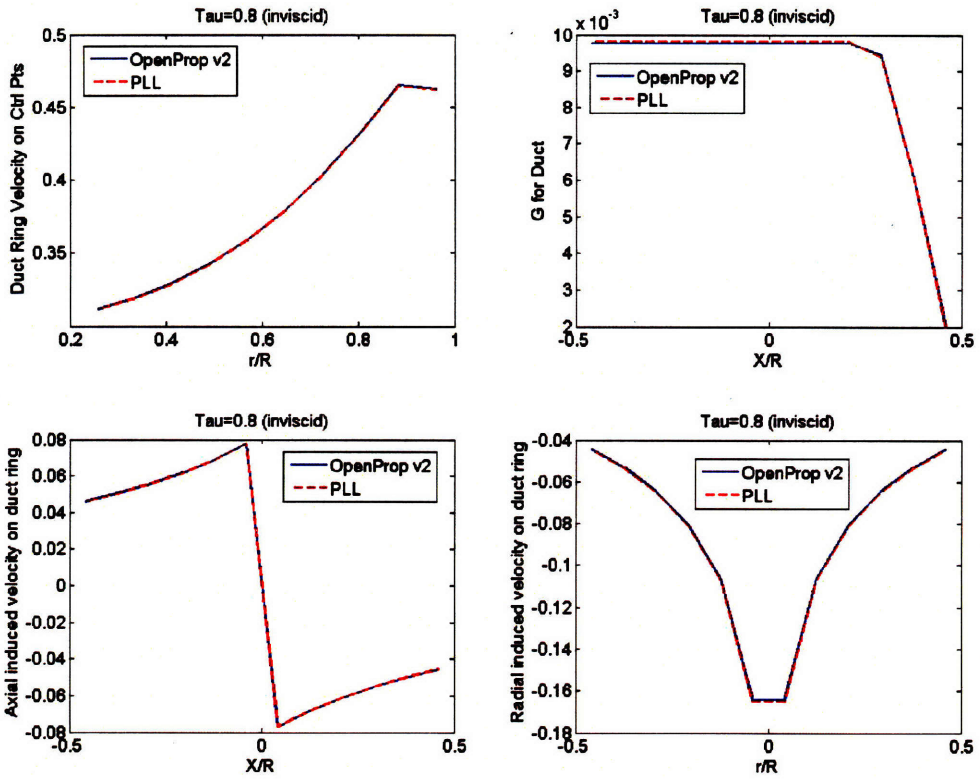

Figure 4-5: OpenProp v2 algorithm duct results using PLL circulation

Figure 4-6 gives an efficiency versus thrust ratio comparison for PLL and OpenProp v2. The ideal efficiency as calculated by equation 3-1 using actuator disk theory is also shown. For this figure, PLL and OpenProp v2 each ran independently (i.e. OpenProp v2 calculated its own optimum circulation distribution instead of using PLL's as was the case in Figure 4-4 and Figure 4-5).

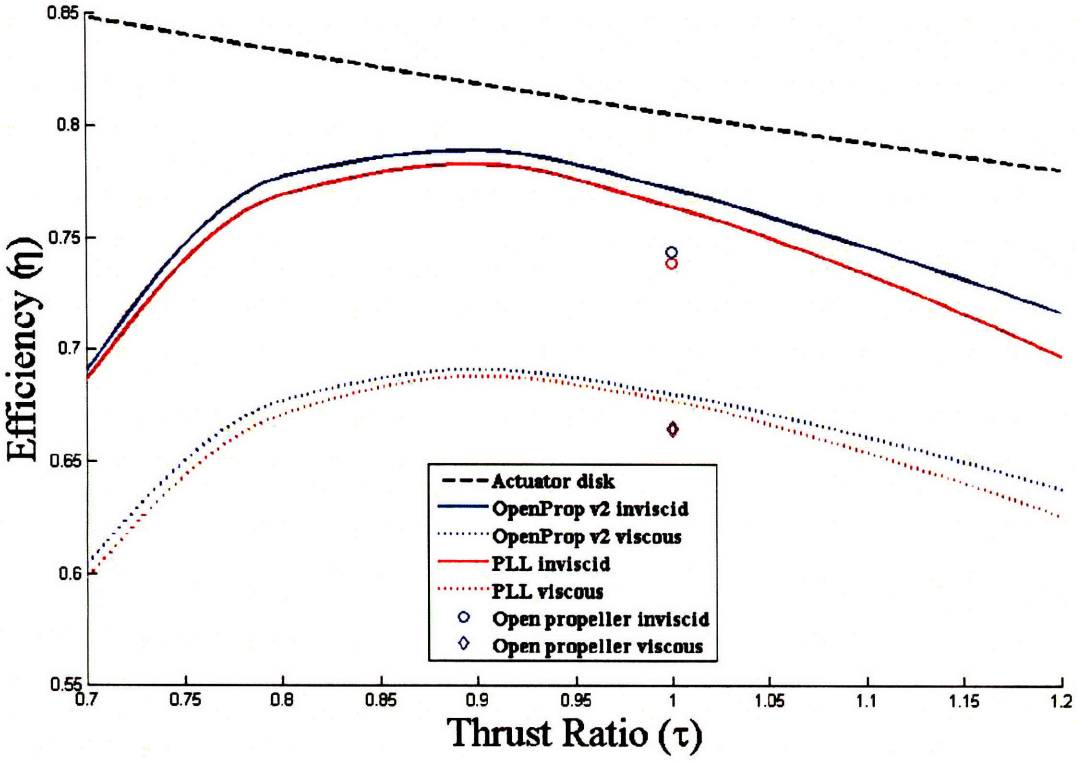

Figure 4-6: Efficiency versus thrust ratio (τ) comparison between OpenProp v2 and PLL

The OpenProp v2 results did not match PLL exactly, but they are close (within 1% for $\tau \leq 1.0$) and follow the same trend as PLL. As with PLL, the maximum efficiency for the ducted propeller occurred at a thrust ratio of approximately $\tau = 0.9$. The difference between OpenProp v2 and PLL increased above $\tau = 1.0$ and was approximately 2% at $\tau = 1.2$.[7] The reason for the difference is explained by the optimum circulation distribution calculated by OpenProp v2.

The following group of figures show comparisons between OpenProp v2 and PLL for thrust ratios of 0.8, 1.0, and 1.2. Viscosity is neglected. For $\tau = 0.8$, three figures are shown: Figure 4-7 gives the standard OpenProp graphical output and Figure 4-8 and Figure 4-9 show the comparison between OpenProp v2 and PLL. For $\tau = 1.0$ and $\tau = 1.2$, only the quad-chart with the circulation comparison is shown.

[7] OpenProp v2 did not converge for thrust ratios greater than 1.2.

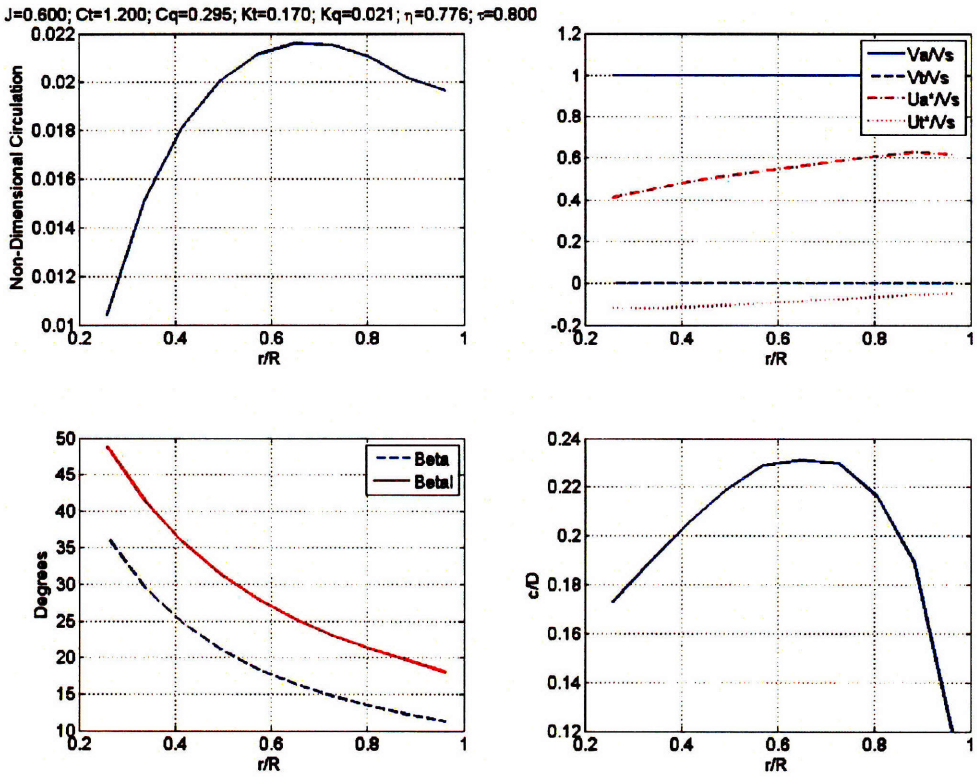

Figure 4-7: OpenProp v2 graphical output for test case with $\tau = 0.8$

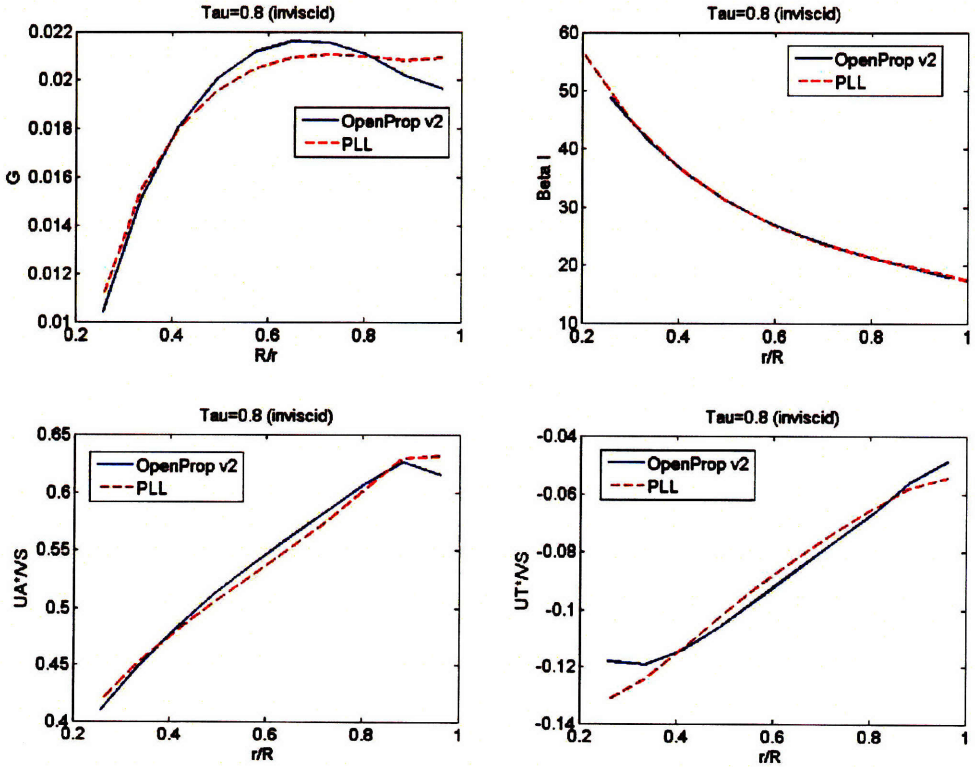

Figure 4-8: OpenProp v2 comparison with PLL for $\tau = 0.8$

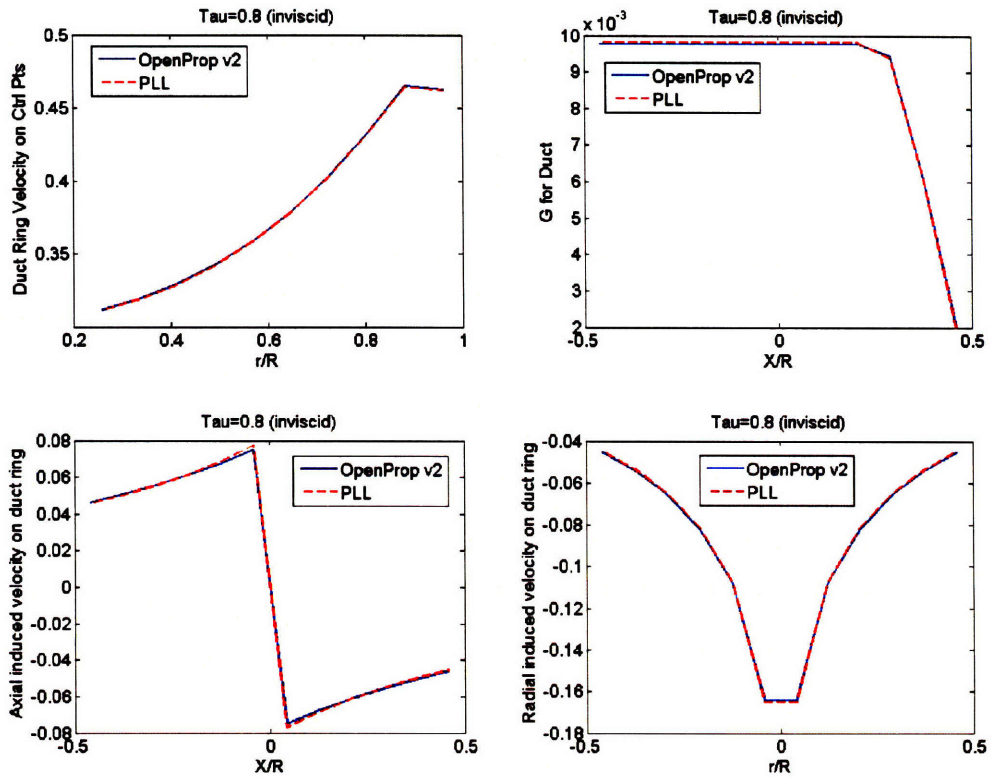

Figure 4-9: OpenProp v2 comparison with PLL for $\tau = 0.8$ (duct ring velocities)

Figure 4-8 shows that the optimum circulation distribution obtained from OpenProp v2 is slightly different than PLL's solution for the accelerating duct case. The main difference was that with OpenProp v2 the circulation reached a maximum value at approximately r/R = 0.7 and decreased slightly at the tip.

Figure 4-10 shows the comparison between OpenProp v2 and PLL for $\tau = 1.0$ (neutral duct). The results match very well.

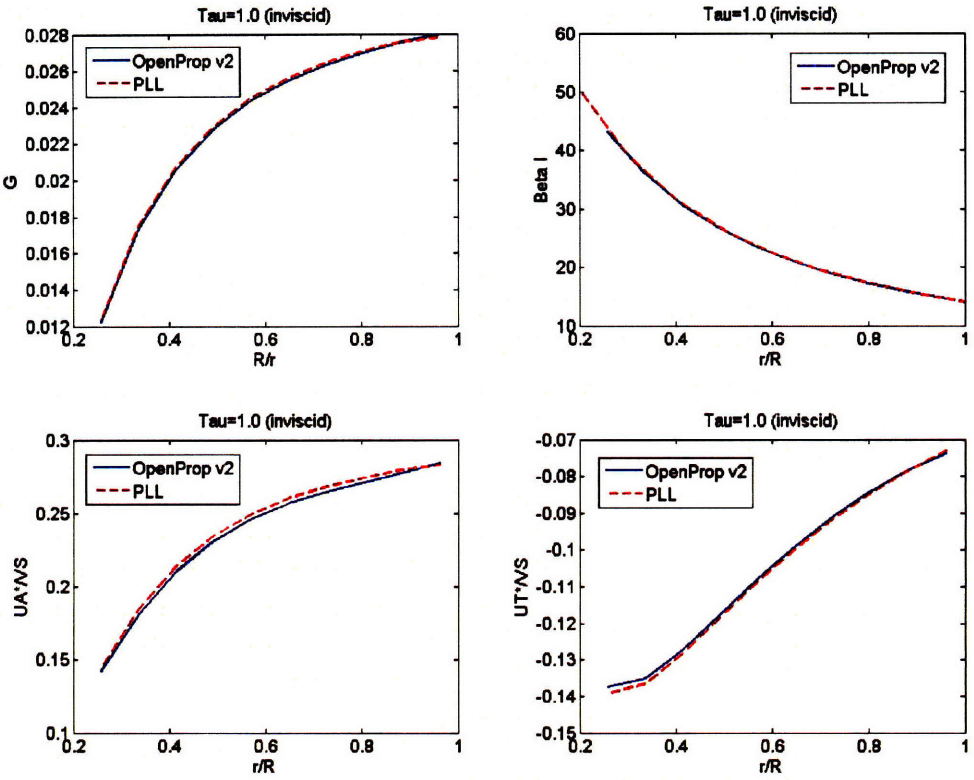

Figure 4-10: OpenProp v2 comparison with PLL for $\tau = 1.0$

Figure 4-11 shows the comparison between OpenProp v2 and PLL for the decelerating duct case of $\tau = 1.2$. As with the accelerating duct case, the optimum circulation distribution obtained from OpenProp v2 is slightly different than PLL's solution. For the decelerating duct, OpenProp v2's optimum circulation distribution has an inflection point at approximately r/R = 0.7 and the tip circulation is slightly higher than the PLL solution.

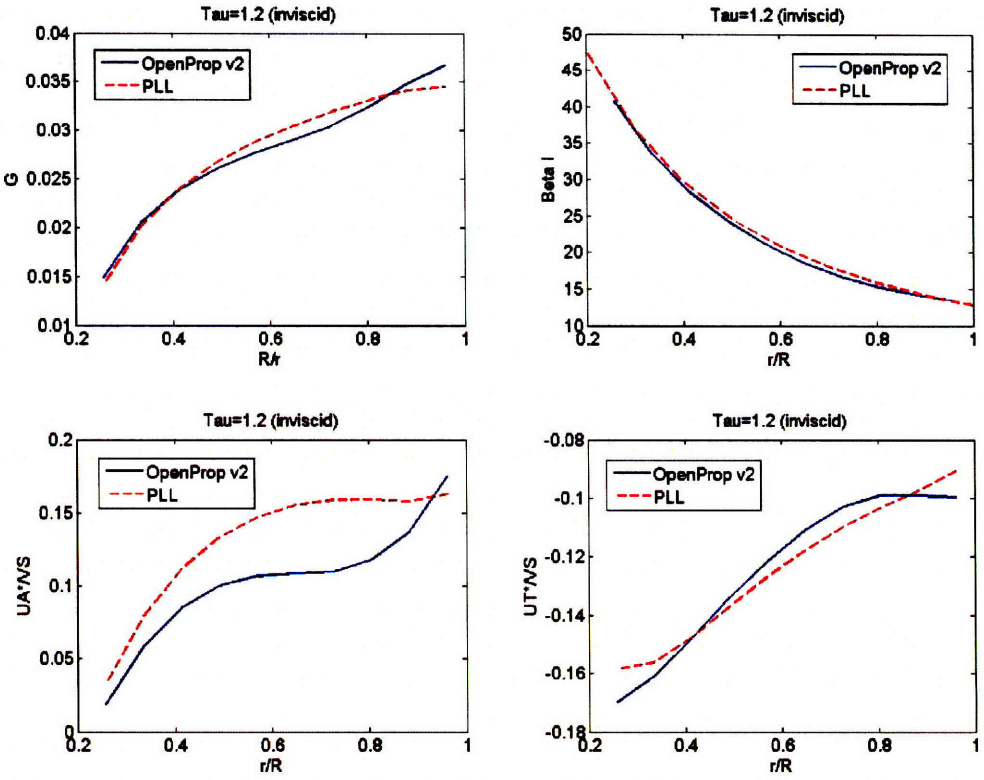

Figure 4-11: OpenProp v2 comparison with PLL for $\tau = 1.2$

5. Conclusions and Recommendations

5.1 Conclusions

This thesis successfully extended OpenProp's capability such that it can now design a propeller operating inside a duct with no gap for thrust ratios between 0.7 and 1.2. The results confirmed that maximum efficiency is obtained with a thrust ratio of approximately 0.9.

The variational optimization routine used in OpenProp v2 was validated with output from PLL. All induced velocity calculations matched PLL, and the optimum circulation distribution matched PLL for the neutral duct case. However, the optimum circulation distribution obtained from OpenProp v2 for both the accelerating and decelerating duct cases varied slightly but distinctively from PLL. With PLL, the circulation distribution always reached a maximum at the tip. This was not the case with OpenProp v2. Only for the neutral duct case was this true for OpenProp v2. For the accelerating duct, the circulation peaked at approximately $r/R = 0.7$ and the tip circulation was lower than PLL's tip circulation. For the decelerating duct, the circulation had an inflection point at approximately $r/R = 0.7$ and the tip circulation was higher than PLL's.

The specific reason for the differing optimum circulation distributions was not discovered. However, the author did verify that the variational optimization routine implemented in OpenProp v2 was a faithful representation of the routine presented by Coney in (1). It was assumed that PLL used this variational optimization routine as well, but it is possible that PLL added additional constraints that were not discussed in (1).

5.2 Recommendations for further work

This thesis only examined a duct with no thickness. Source rings could be integrated into the algorithm to model duct thickness.

A tip gap model could be added to represent the flow between the propeller and duct when a gap exists. This would allow the design of ducted propellers with gaps greater than zero.

This thesis did not analyze the flow around the duct nor did it attempt to define the true duct orientation. By analyzing the flow around the duct three important objectives could be obtained. First, the designer could ensure that flow separation does not occur on the duct. This is critical because if separation occurs, drag will increase dramatically. Second, understanding the flow characteristics in the gap is essential to analyzing the performance of the entire system under various loading conditions. Third, the flow streamlines would outline the shape of the duct and reveal the duct angle of attack (i.e. orientation). Coupling OpenProp with a computational fluid dynamics code would be the ultimate goal to properly analyze the flow.

References

1. **Coney, W. B.** *A Method For The Design Of A Class Of Optimum Marine Propulsors, PhD Thesis.* s.l. : Massachusetts Institute of Technology, Department of Ocean Engineering, 1989.

2. **Chung, H.** *An Enhanced Propeller Design Program Based on Propeller Vortex Lattice Lifting Line Theory, Master's Thesis.* s.l. : Massachusetts Institute of Technology, Department of Mechanical Engineering, 2007.

3. **D'Epagnier, K.P.** A Computational Tool for the Rapid Design and Prototyping of Propellers, Master's Thesis. s.l. : Massachusetts Institue of Technology, Department of Mechanical Engineering, 2007.

4. **Lewis.** *Principals of Naval Architecture, Volume II: Resistance, Propulsion, and Vibration.* Jersey City : Society of Naval Architects and Marine Engineers, 1988.

5. **Weber, D. Kuchemann and J.** *Aerodynamics of Propulsion.* New York : McGraw-Hill, 1953.

6. **Kuchemann, D. and Weber, J.** *Aerodynamics of Propulsion.* New York : McGraw-Hill, 1953.

7. **Hough, G.R. and Ordway, D.E.** The Generalized Actuator Disk. *Therm Advanced Research. Developments in Theoretical and Applied Mechanics.* Oxford : Pergamon Press, 1964. Vol. 2.

8. **Abramowitz, M. and Stegun, I. A.** *Handbook of Mathematical Functions.* Washington DC : National Bureau of Standards, Applied Mathematics, 1972.

9. **National Bureau of Standards.** *Tables of Associated Legendre Functions.* New York : Columbia Univeristy Press, 1945.

10. **Kerwin, Justin E.** Hydrofoils and Propellers Lecture Notes. s.l. : Massachusetts Institute of Technology, 2001.

11. **Lamb, H.** *Hydrodynamics 6th Edition.* New York : Dover Publications, 1945.

12. **Katz, J. and Plotkin, A.** *Low-Speed Aerodynamics 2nd Edition.* Cambridge, UK : Cambridge University Press, 2001.

13. **Hsin, C.** Efficient Computational Methods For Multi-Component Lifting Line Calculations, Masters Thesis. s.l. : Massachusetts Institute of Technology, Department of Ocean Engineering, 1987.

14. **Byrd, P.F. and Friedman, M.D.** *Handbook of Elliptic Integrals for Engineers and Physicists.* Berlin : Springer-Verlag, 1954.

15. **Caja, A.S.** On the Optimum Propeller Loading with Inclusion of Duct and Hub, Master's Thesis. s.l. : Massachusetts Institute of Technology, Department of Ocean Engineering, 1988.

16. **Sluyter, M.M.** A Computational Program and Extended Tabulation of Legendre Functions of Second Kind and Half Order. s.l. : Therm Advance Research, 1960.

17. **Hughes, M.J.** A Comparison of Experiment and Analysis for a Ducted Propeller, Master's Thesis. s.l. : Massachusetts Institute of Technology, Department of Ocean Engineering, 1990.

Appendix A. Duct Theory MATLAB® Code

A.1 ductVort.m

```
%Discrete representation of vorticity (circulation) on vortex rings
%that represent duct as a NACA a=0.8 meanline.

%Calculates duct vortex ring influence (UADUCT)
%on prop lifting line ctrl pts

%Variables:
% %   R [m]:            propeller radius
% %   rDuct [m]:        duct radius (formerly vrRad), vortex ring radius at
% %                     xDuct
% %   Mp:               # of control points (N=M, M: # of vortex rings)
% %   RC:               radius of ctrl pts on lifting line

%Note: duct chord = R

%Returns:
%         vRingLoc [m]:     location of each vortex ring (x,y,z vector),
%                           (formerly (vortex)
%         dVort [m^2/s]:    circulation distribution of each vortex ring
%         UADUCT [m/s]:     duct vortex ring axial influence on prop
%                           lifting line control pts

%close all;clear all;clc;

function [vRingLoc,dVort,UADUCT] = ductVort(R,rDuct,Mp,RC)

%setup vRing spacing
M=Mp+2;
if rem(M,2)~=0       %ensures Mp is even
    M=M+1;
end
dS=1/M;              %spacing between vRings (non-dim with R)
hdS=0.5*dS;          %half of dS

%computes the circulation on vortex rings which each represent the
%vorticity on a piece of NACA a=0.8 mean line located at position XvRing
%(L.E.=0.0, T.E.=1.0) and is of length dS.

XvRing=zeros(1,M);dVort=XvRing;

for n=1:M
    XvRing(n)=(n-1)*dS+hdS;

    X2 = XvRing(n) + hdS;
    X1 = XvRing(n) - hdS;
    if X2 <= 0.8
        dVort(n) = dS/0.9;
    elseif X1 >= 0.8
        Y1 = 1.0 - (X1 - 0.8)/0.2;
```

```
        Y2 = 1.0 - (X2 - 0.8)/0.2;
        dVort(n) = dS*0.5*(Y1 + Y2)/0.9;
    else
        Y2 = 1.0 - (X2 - 0.8)/0.2;
        FRONT = 0.8 - X1;
        BACK = 0.5*(1.0 + Y2)*(X2 - 0.8);
        dVort(n) = (FRONT + BACK)/0.9;
    end
end

LED=-(M/2)*dS;          %location of leading vRing on duct
XvRing=XvRing+LED;
vRingLoc=zeros(3,M);
vRingLoc(1,:)=XvRing;
vRingLoc(2,:)=rDuct/R;

% % plot(XvRing,dVort,'*',XvRing,dVort1,'+')

%Calc duct vortex ring influence (UADUCT) on prop lifting line ctrl pts
    %Note: No tangential influence
    %Note: Radial influence does not create a force on radial lifting line

UADUCT=zeros(Mp,1);             %axial influence
URDUCT=UADUCT;UTDUCT=UADUCT;    %radial and tantential influence
for n=1:Mp                      %cycle thru all ctrl pts on lifting line
    P=[0;RC(n);0];              %3D coord for ctrl pt
    for m=1:M                   %cycle thru all vortex rings on duct
        UD = vRing(vRingLoc(1,m),vRingLoc(2,m),P,dVort(m));
        UADUCT(n)=UADUCT(n)+UD(1);
        URDUCT(n)=URDUCT(n)+UD(2);
        UTDUCT(n)=UTDUCT(n)+UD(3);
    end
end

UADUCT=UADUCT*2*pi;         %2*pi needed for non-dimensional circulation (G)
URDUCT=URDUCT*2*pi;
UTDUCT=URDUCT*2*pi;
```

A.2 ductThrust.m

```
%Calculates total duct thrust coefficient and associated parameters

%This version handles one propeller.  Modifications required if additional
    %rotors or stators desired.

%Variables:
% %   vRingLoc:          location of duct vortex rings
% %   dVort:             circulation distribution on duct rings
% %   dCirc:             strength of duct vorticity
% %   rDuct:             radius of duct
% %   CDd:               coefficient of drag for the duct
% %   U [m/s]:           x-dir inflow velocity magnitude
% %   rho [kg/m3]:       density of fluid
% %   RV:                radius of trailing helical vortices on lifting line
% %   G:                 non-dim circulation for panels on lifting line
% %   TanBI:             tangent of betaI for RV points
% %   Z:                 # of blades
% %   R:                 radius of propeller

%Returns: CTD:           total duct thrust coeff (viscous drag included)
%         dCirc          new dCirc scaled to provide desired duct thrust
%         UAdVS          induced axl velocity on duct ring from lifting line
%         URdVS          induced rad velocity on duct ring from lifting line

function [CTD,dCirc,UAdVS,URdVS] = ductThrust(vRingLoc,dVort,dCirc,...
                                    rDuct,CDd,U,rho,RV,G,TanBI,Z,R,CTDDES)

M=length(dVort);              %# of duct vortex rings
k=[0 0 1];                    %unit vector in Z direction
Uvec=[U;0;0];                 %free stream velocity vector

%Velocity vector at each vortex (Vvortex)
Vvortex=zeros(3,M);
VvortexInduced=Vvortex;
Lvortex=Vvortex;

for m=1:M                     %cycles thru all vortex rings
    VvortexInduced(:,m)=VvortexInduced(:,m) + CMV1(vRingLoc(1,m),...
                        vRingLoc(2,m),RV,G,TanBI,Z);   %radial lifting line
    Vvortex(:,m)       =VvortexInduced(:,m) + Uvec;
    Lvortex(:,m)       =rho*cross(Vvortex(:,m),k*dCirc*dVort(m));
                                  %lift on a vortex ring
end

URdVS=VvortexInduced(2,:);      %induced rad vel on duct ring from prop

%Note: CMV1.m values for axial and tangential induced velocities on the duct
%rings don't match PLL results. Tangential velocities are not needed, so that
%discrepancy is not resolved.  Axial velocities only match for -x locations.
%However, from PLL, +x are a negative mirror of -x values so I have adjusted
%accordingly so that axial tangential velocities can be used to calculate the
%duct viscous drag.
```

```
%Adjust axial CMV so that results matches PLL
m1 = M/2;                    %last -x vortex
for m=1:M/2
    VvortexInduced(1,m1+m)=-VvortexInduced(1,m1-m+1);
end

Vvortex(1,:)=VvortexInduced(1,:) + Uvec(1);
UAdVS=VvortexInduced(1,:);      %induced axl velocity on duct ring from prop


%Duct thrust (CONEY P. 77, EQN 3.28.)
Lift=sum(Lvortex,2);
dThrust=-2*pi*rDuct*Lift(1);     %thrust (-x direction) for duct [N]:')
                                 %positive implies thrust to the ship

%Viscous drag for duct (Drag = 0.5*rho*V^2*Chord*CDd * 2*pi*rDuct)
delS=abs(vRingLoc(1,1)-vRingLoc(1,2));          %vortex spacing
                                                %linear spacing assumed
dDrag = 0;
for m=1:M
    dDrag = dDrag + Vvortex(1,m)^2;
end
dDrag=0.5*rho*dDrag*delS*CDd*2*pi*rDuct;        %R = duct chord length
CTDdrag   = 4*dDrag  /(R*2*pi)/(rho*R);         %normalized duct "drag" CT
CTDthrust = 4*dThrust         /(rho*R);         %normalized duct "thrust" CT

dThrustTot=dThrust-dDrag/(R*2*pi);              %total thrust for the duct
                                                %(R*2*pi) required to make
                                                %dDrag dimensions match
                                                %dThrust

CTD = 4*dThrustTot / (rho*R);                   %CT for duct

%scale duct circulation so that duct provides required thrust
if dCirc~=0
    dCirc = dCirc/CTDthrust*(CTDDES+CTDdrag);
end
```

A.3 vRing.m

```
%Returns velocity vector induced by vortex ring of input strength
%gamma at a point (p) in space given the x-axis location (vrX) and
%radius (vrRad) of the vortex ring:

%Axis of the vortex ring is in the direction of the x-axis.

%Variables
    %gamma:     vortex ring strength
    %vrX:       x-axis location of vortex ring
    %vrRad:     radius of vortex ring
    %p          (Px,Py,Pz) point at which velocity is induced

%Returns: Vp:   velocity at point P

%Ref: Kuchemann and Weber, Aerodynamics of Propulsion p 305.

function [Vp] = vRing(vrX,vrRad,p,gamma)

if vrRad == 0               %stops function if vrRad = 0
    Vp=[0; 0; 0];
    return
end

if vrRad < 0                %stops function if vrRad < 0
    Vp=[NaN; NaN; NaN];
    return
end

Px=p(1);Py=p(2);Pz=p(3);

if Pz==0
    if Py<0
        thetaP=-pi/2;               %cylindrical coord angle for P
    else
        thetaP=pi/2;
    end
else
    if Pz>0                         %logic for atan ambiguity
        thetaP=atan(Py/Pz);
    else
        thetaP=atan(Py/Pz)+pi;
    end
end
Prad=sqrt(Pz^2 + Py^2);             %cylindrical coord radius for P

if vrX==p(1) & vrRad==Prad          %stops function if P on vortex ring
    Vp=[NaN; NaN; NaN];
    return
end

x=(Px-vrX)/vrRad;                   %x/r' from Kuchemann
r=Prad/vrRad;                       %r/r' from Kuchemann
```

```
%Elliptic integral method (Kuchemann p. 305)
%uses parameter k where k^2 = m for elliptic integrals

k=sqrt(4*r/(x^2+(r+1)^2));
[K,E]=ellipke(k^2);

Vx=gamma/(2*pi*vrRad)/sqrt(x^2+(r+1)^2)*(K-(1+2*(r-1)/(x^2+(r-1)^2))*E);

if r==0
    Vr=0;
else
    Vr=gamma/(2*pi*vrRad)*(-x)/r/sqrt(x^2+(r+1)^2)*(K-(1+2*r/...
      (x^2+(r-1)^2))*E);
end

Vy=Vr*sin(thetaP);
Vz=Vr*cos(thetaP);

Vp=[Vx; Vy; Vz];
```

A.4 vpfDuct.m

```
%Velocity Prediction Function for a Duct
%Returns velocity (Vp) at pt P due to a set of vortex rings,
    %source rings (for thickness), and free stream

%Variables:
% %   P:                point at which velocity is desired (column vector)
% %   vortex:           matrix of vortex and source locations
% %   gamma:            matrix of vortex ring strengths
% %   S:                matrix of source ring strengths
% %   Uvec:             free stream velocity vector

%Returns: Vp:           velocity at point P

function [Vp] = vpfDuct(P,vortex,gamma,S,Uvec)

Vp = [0; 0; 0];
for m=1:size(vortex,2)              %cycle thru all vorticies
    Rvp=P-vortex(:,m);              %vector from vortex to P
    if norm(Rvp)<10e-5              %skips vortex if P is on vortex
    else
        Jp=Rvp / (2*pi*norm(Rvp)^2);
        Vp=Vp + vRing(vortex(1,m),vortex(2,m),P,gamma(m));
        %future improvement: add source ring influence
    end
end
Vp=Vp+Uvec;
```

A.5 CMV.m

```
%Returns circumferential mean tangential,axial, and radial induced
%velocities of a propeller at any desired axial location

%Uses Coney's version of Hough and Ordway's Formulas

%uHough and vHough are included as a validation case.  They are only
    %valid for no hub and trailing vortex system whose path is determined
    %soley by the incoming free stream with translation U and rotation
    %(omega). (Hough p.319)

%Variables
    %xC:    axl dist btwn prop (i.e. vortex plane) and control point plane
    %       positive if ctrl pt downstream of propeller
    %rC:    radius to calculate CMV
    %M:     # of panels on lifing line
    %gamma: circulation for panels on lifting line
    %rtv:   vector of radius of trailing vorticies on prop lifting line
    %Z:     number of blades
    %TanBI: vector of tangent of advance angles of trailing vortices

%Returns: axlCMV: axial CMV at xC,rC
     %    radCMV: radial CMV at xC, rC
     %    tanCMV: tangential CMV at xC, rC

function [Vp] = CMV(xC,rC,rtv,gamma,TanBI,Z)

if abs(xC)<10e-10 & rC==rtv(end)      %logic to skip routine
                                      %if ctrl pt on lifting line
    Vp = [0; 0; 0]
    return
end

M=length(gamma);

%tangential velocity induced from a horseshoe vortex
%(bound and trailing vorticity)
tanCMV=0;

% % %Coney's implementation.  tanCMV = 0 if rtv=rC or if xC=0
% % for i=1:M
% %     S=(rtv(i)-rC)*(rtv(i+1)-rC);
% %     if S<0 & xC>0
% %         tanCMV=tanCMV-Z*gamma(i)/(2*pi*rC);
% %     end
% % end

%axial and radial velocity induced from trailing vorticies
    %sum effect from every trailing vortex. except for ends, each rtv
    %represents two trailing vortex with different circulation.
    %for each horseshoe, lower trailer gamma is same sign as bound gamma
    %and upper tailer is opposite sign. (coney p. 171).
```

```
axlCMV=0;radCMV=0;

for i=1:M+1
    q=1+(xC^2+(rC-rtv(i))^2)/(2*rC*rtv(i));
    s=asin(xC/sqrt(xC^2+(rC-rtv(i))^2));
                                    %amplitude wrt elliptical integrals
    t=sqrt(4*rC*rtv(i)/(xC^2+(rC+rtv(i))^2));
                                    %t=k (modulus wrt elliptical integrals)

    if rC>rtv(i)          %agrees with Coney
                          %Hough has rc>=rtv(i)
        c1=   xC/(2*sqrt(rC*rtv(i)))*Q2Mhalf(q)-pi/2*Heuman(s,asin(t));
    else
        c1=pi+xC/(2*sqrt(rC*rtv(i)))*Q2Mhalf(q)+pi/2*Heuman(s,asin(t));
    end

    % c2 is not needed if tanCMV calculated using Kelvin's theorem
% %     if rC<rtv(i)
% %         c2=   xC/(2*sqrt(rC*rtv(i)))*Q2Mhalf(q)-pi/2*Heuman(s,asin(t));
% %     else
% %         c2=pi+xC/(2*sqrt(rC*rtv(i)))*Q2Mhalf(q)+pi/2*Heuman(s,asin(t));
% %     end

    if i~=1 && i~=M+1            %logic for interior trailer vortices
        axlCMV=axlCMV+Z*c1/(pi*rtv(i)                *TanBI(i))*...
                    (gamma(i)-gamma(i-1));
        radCMV=radCMV+Z*Q2half(q)/(pi*sqrt(rC*rtv(i))*TanBI(i))*...
                    (gamma(i)-gamma(i-1));
%         tanCMV=tanCMV+Z*c2/(pi*rtv(i)                *...
                    %(gamma(i)-gamma(i-1)));

    else if i==1                %logic for first and last trailer vortices
            axlCMV=axlCMV+gamma(i)*Z*c1/(pi*rtv(i)*...
                        TanBI(i));
            radCMV=radCMV+gamma(i)*Z*Q2half(q)/(pi*sqrt(rC*rtv(i))*...
                        TanBI(i));
%             tanCMV=tanCMV+gamma(i)*Z*c2/(pi*rtv(i));

        else axlCMV=axlCMV-gamma(i-1)*Z*c1/(pi*rtv(i)*...
                        TanBI(i));
            radCMV=radCMV-gamma(i-1)*Z*Q2half(q)/(pi*sqrt(rC*rtv(i))*...
                        TanBI(i));
%             tanCMV=tanCMV-gamma(i-1)*Z*c2/(pi*rtv(i));

        end
    end
end

axlCMV=-axlCMV/2;           %adjust to match PLL output for UA/VS
                            %(only matches negative x values)
radCMV=radCMV/2;            %adjust to match PLL output for UR/VS

Vp = [axlCMV; radCMV; tanCMV];
```

A.6 ductPlot.m

```
%Plots duct

%Variables:
% %   c [m]:            chordlength
% %   alpha [radians]:  angle of attack
% %   vrRad [m]:        vortex ring radius (duct radius to meanline)
% %   ductRef:          chordwise reference position on duct
% %                     fixed at 0.5 but could be passed as a variable
% %   xDuct [m]:        global propeller x-coord of ductRef
% %   fo:               max camber (% of chordlength)
% %   to:               max thickness (% of chordlength)

% % Notes: X-axis positive in streamwise direction (i.e. downstream).

function[] = ductPlot(vrRad,c,fo,to,alpha,ductRef)

%Read in meanline f(x) and thickness t(x) distribution data
        %Read foil data (x, f/fo, t/to) from text file
        %Foil_data.txt contains parabolic meanline (f/fo)
        %and elliptical thickness(t/to) data

[x_over_c,f_over_fo,t_over_to]=textread('foil_data.txt','%f%f%f',...
                                        'headerlines',3);
%x_over_c range is -c/2 to c/2 (this is converted to x=0 to x=c below)

f=fo*c*f_over_fo;          %camber distribution
t=to*c*t_over_to;          %thickness distribution

x=x_over_c*c + c/2;        %dimensionalizes x with a range of 0 to c
                           %range of 0 to c is needed for cosine spacing

fpp=spline(x,f);           %spline camber data
% % fPpp=fnder(fpp);         %splines slope of fpp
tpp=spline(x,t);           %spline thickness data

%alt method not using FNDER (Spline Toolbox)
xl=length(x);
theta=zeros(xl,1);
for m=1:xl
    if m==xl
        theta(m)=theta(m-1);
    else
        theta(m)=atan((ppval(fpp,x(m+1))-ppval(fpp,x(m)))/(x(m+1)-x(m)));
    end
end

%Generate 2-D flat cross-section
% % theta   = atan(ppval(fPpp,x));
x_upper = x - t/2.*sin(theta);
y_upper = f + t/2.*cos(theta);
x_lower = x + t/2.*sin(theta);
y_lower = f - t/2.*cos(theta);
```

```
% % % %Plot 2-D section with 0 degrees angle of attack
% % % plot(x_lower,y_lower)
% % % hold on
% % % plot(x_upper,y_upper)
% % % xlabel('X-axis');ylabel('Y-axis');
% % % title('Duct section with 0 degrees angle of attack');
% % % axis equal
% % % figure

%Reposition section
var1=ductRef*c;                 %offset (x-dir), ductRef at x=0
var2=ppval(fpp,ductRef*c);      %offset (y-dir) for camber, ductRef at y=0
var3=0.5*ppval(tpp,ductRef*c);  %offset (y-dir) for thick, ductRef at y=0
                                %ductRef is point where blade and duct meet
x_upper = x_upper - var1;
y_upper = y_upper - var2 + var3;
x_lower = x_lower - var1;
y_lower = y_lower - var2 + var3;

%Rotate for angle of attack and place section at correct radius
x_upper_rot = x_upper*cos(-alpha) - y_upper*sin(-alpha);
y_upper_rot = x_upper*sin(-alpha) + y_upper*cos(-alpha) + vrRad;
x_lower_rot = x_lower*cos(-alpha) - y_lower*sin(-alpha);
y_lower_rot = x_lower*sin(-alpha) + y_lower*cos(-alpha) + vrRad;

% % % %Plot duct section rotated
% % % plot(x_lower_rot,y_lower_rot)
% % % hold on
% % % plot(x_upper_rot,y_upper_rot)
% % % xlabel('X-axis');ylabel('Y-axis');
% % % title(['Duct section (repositioned) with ',num2str(alpha*180/pi),...
% % %                                          ' degrees angle of attack']);
% % % axis equal
% % % figure

%Build all sections (upper and lower surfaces) for complete 3-D duct
z=zeros(length(x),1);
[thetaU,phiU,RU]=cart2sph(y_upper_rot,x_upper_rot,z);
[thetaL,phiL,RL]=cart2sph(y_lower_rot,x_lower_rot,z);

nds=50;                         %# of duct sections for plotting
for n=1:nds
%     phi=0+pi:1.9*pi/(nds-1):2.1*pi+pi;    %360 deg coverage for duct
    phi=0:2*pi/(nds-1):2.1*pi;    %360 deg coverage for duct
    [x_u_3D(n,:),y_u_3D(n,:),z_u_3D(n,:)]=sph2cart(phi(n),thetaU,RU);
    [x_l_3D(n,:),y_l_3D(n,:),z_l_3D(n,:)]=sph2cart(phi(n),thetaL,RL);

% % %     %Plot duct sections individually
% % %     plot3(z_u_3D(n,:),x_u_3D(n,:),y_u_3D(n,:))
% % %     hold on
% % %     plot3(z_l_3D(n,:),x_l_3D(n,:),y_l_3D(n,:))
end
% % % xlabel('X-axis');ylabel('Y-axis');zlabel('Z-axis')
% % % title(['Duct with ',num2str(alpha*180/pi),' degrees angle of attack'])
```

```
% % % axis equal
% % % figure

%Plot duct as 3-D surface
surfl(z_u_3D,x_u_3D,y_u_3D)
hold on
surfl(z_l_3D,x_l_3D,y_l_3D)
% % % xlabel('X-axis');ylabel('Y-axis');zlabel('Z-axis')
% % % title(['Duct with ',num2str(alpha*180/pi),' degrees angle of attack'])
% % % axis equal

%shading interp
%colormap(copper)
```

Appendix B. Mathematical Functions MATLAB® Code

B.1 Q2half.m

```
%Q2half: Legendre fuction of the second kind and positive half order
        %Ref: Handbook of Math Functions, Abramowitz and Stegun, 1972
        %section 8.13.7, p.337
        %uses modulus k for elliptic integrals (m=k^2)
        %ellipke uses parameter m.

function [Q2] = Q2half(q)

k=sqrt(2/(q+1));
[K,E]=ellipke(k^2);
Q2=q*k*K-sqrt(2*(q+1))*E;

%Validated with the National Bureau of Standards Tables of
%Associated Legendre Functions
%(Columbia University Press, New York, 1945), p.266.
%From the tables: Q2half(1.5)=.393175, Q2half(2.7)=.134035,
%Q2half(6)=.0382887, Q2half(8.4)=.0229646, Q2half(10)=.0176449
```

B.2 Q2Mhalf.m

```
%Q2Mhalf: Legendre fuction of the second kind and minus half order
        %Ref: Handbook of Math Functions, Abramowitz and Stegun, 1972
        %section 8.13.3, p.337
        %uses modulus k for elliptic integrals (m=k^2)
        %ellipke uses parameter m.

function [Q2M] = Q2Mhalf(q)

k=sqrt(2/(q+1));
[K,E]=ellipke(k^2);
Q2M=k*K;

%checked with ref p.340 example
%Validated with the National Bureau of Standards Tables of
%Associated Legendre Functions
%(Columbia University Press, New York, 1945), p.264.
%%From the tables: Q2Mhalf(1.5)=2.01891, Q2Mhalf(2.7)=1.38958,
%Q2Mhalf(6)=0.911696, Q2Mhalf(8.4)=0.768523, Q2Mhalf(10)=0.703806
```

B.3 Heuman.m

```
%Heuman: Heuman's Lambda fuction
%Ref: Handbook of Math Functions, Abramowitz and Stegun, 1972
        %section 17.4.39, p.595
%Ref: Handbook of Elliptic Integrals for Engineers and Physicists, Byrd and
        %Friedman, 1954, p37.

%phi:       amplitude (radians),        (CMV sends 's' as phi)
%alpha:     modular angle (radians),    (CMV sends 't' alpha)

function [H] = Heuman(phi,alpha)

[K,E]=ellipke(sin(alpha)^2);
F=mfun('EllipticF',sin(phi),sin(pi/2-alpha));
                                %Incomplete elliptic integral, 1st kind
EE=mfun('EllipticE',sin(phi),sin(pi/2-alpha));
                                %Incomplete elliptic Integral, 2nd kind

H=2/pi*(K*EE-(K-E)*F);
```

Appendix C. Variational Optimization Routine MATLAB® Code

C.1 Coney.m

```
% ========================================================================
% ========================================================== Coney Function
%
% The Coney function determines the  "optimum" circulation distribution
% that satisfies the input operating conditions, using a variational
% optimization algorithm, as described on Coney, page 25.  The Coney
% function returns performance specs, such as thrust coefficient and
% efficiency, as well as the circulation distribution, ect.
%
% Reference: Coney, William, "A Method for the Design of a Class of Optimum
%            Marine Propulsors", Ph.D. thesis, MIT, 1989.
%
% Includes Coney and Align_wak functions
% Authors: Brenden Epps (variational optimization and wake alignment)
%          Mitch Stubblefield (duct integration)
% ------------------------------------------------------------------------

function [CT,CQ,CP,KT,KQ,VMIV,EFFY,RC,G,VAC,VTC,UASTAR,UTSTAR,TANBC,...
            TANBIC,CoD,CD,TAU,Xring,dVort,UADUCT,dCirc,UAdVS,URdVS]...
         = Coney(Rhub,R,Z,Mp,ITER,Rhv,HUF,TUF,SCF,Js,CTDES,Hub_Flag,...
            Duct_Flag,TAU,rDuct_oR,CDd,XR,XCoD,XCD,XVA,XVT,rho,Vs);
clc

%------------------- Initialize variables needed in functions
rDuct=rDuct_oR*R;       %duct radius
UADUCT=zeros(1,Mp);     %Induction of duct on lifting line ctrl pts
CTD=0;                  %CT for duct
dVort=zeros(1,Mp+2);    %circulation distribution of each vortex ring
Xring=zeros(1,Mp+2);UAdVS=Xring;URdVS=Xring;

%------------- Compute the Volumetric Mean Inflow Velocity, eqn 163, p.138
Rhub_oR = Rhub/R;               % [ ], hub radius       / propeller radius
RoR     = 1;                    % [ ], propeller radius / propeller radius

VMIV = 2*trapz(XR,XR.*XVA)/(RoR^2-Rhub_oR^2);     % [ ], VMIV/ship velocity

% ------------------------ Compute evenly-spaced vortex & control pt. radii
RV=zeros(1,Mp+1);RC=zeros(1,Mp);                    % initialize RC and RV
if Duct_Flag==0 & Hub_Flag==0               % no duct image or hub image
    DRR = (RoR-Rhub_oR)/(Mp+.5);            % panel size
    RV(Mp+1)=RoR-.25*DRR;                   % 25% tip inset
    RV(1)=Rhub_oR+.25*DRR;                  % 25% hub inset
elseif Duct_Flag==1 & Hub_Flag==0           % duct image but no hub image
    DRR = (RoR-Rhub_oR)/(Mp+.25);           % panel size
    RV(Mp+1)=RoR;                           % no tip inset
    RV(1)=Rhub_oR+.25*DRR;                  % 25% hub inset
elseif Duct_Flag==1 & Hub_Flag==1           % duct image and hub image
    DRR = (RoR-Rhub_oR)/(Mp);               % panel size
    RV(Mp+1)=RoR;                           % no tip inset
    RV(1)=Rhub_oR;                          % no hub inset
```

```
elseif Duct_Flag==0 & Hub_Flag==1          % no duct image but hub image
    DRR = (RoR-Rhub_oR)/(Mp+.25);          % panel size
    RV(Mp+1)=RoR-.25*DRR;                  % 25% inset for tip
    RV(1)=Rhub_oR;                         % no hub inset
end

RC(1)=RV(1)+.5*DRR;                        % ctrl pt at mid-panel
for m=2:Mp
    RV(m)=RV(m-1)+DRR;
    RC(m)=RC(m-1)+DRR;
end

DR = diff(RV);              % difference in vortex radii / propeller radius

% ------------ Interpolate Va, Vt, Cd, and c/D at vortices & control points
VAV = pchip(XR,XVA,RV);     % axial      inflow vel. / ship vel. at vort pts
VTV = pchip(XR,XVT,RV);     % tangential inflow vel. / ship vel. at vort pts
VAC = pchip(XR,XVA,RC);     % axial      inflow vel. / ship vel. at ctrl pts
VTC = pchip(XR,XVT,RC);     % tangential inflow vel. / ship vel. at ctrl pts
CD  = pchip(XR,XCD,RC);     % section drag coefficient          at ctrl pts
CoD = pchip(XR,XCoD,RC);    % section chord / propeller diameter at ctrl pts

TANBC  = VAC./(pi.*RC./Js + VTC);              % tan(Beta) at control pts.

%Allocate CTDES between propeller and duct
CTPDES     = CTDES*TAU;               %CT desired for the propeller
CTDDES     = CTPDES/TAU-CTPDES;       %CT desired for the duct
VD         = 0;                       %viscous drag

%Initial guess for dCirc (circulation on duct)
dCirc = 0.5*(1-TAU);
if TAU==1 & CDd~=0      %provides a small duct circulation to offset drag
    dCirc = .001;
end

% ------ Compute vortex ring influence functions from duct
if Duct_Flag == 1
    [vRingLoc,dVort,UADUCT] = ductVort(R,rDuct,Mp,RC);
    Xring=vRingLoc(1,:);
end

% ====================== Determine optimum circulation distribution function
%
% See William Coney Ph.D. thesis "A Method For the Design of a Class of
% Optimum Marine Propulsors", MIT, 1989, page 25 for a discussion of
% this variational optimization algorithm.

% ---- Initialize induced velocities & Lagrange multiplier (Coney p.27)
UASTAR(1:Mp) =  0;              % UASTAR / ship speed
UTSTAR(1:Mp) =  0;              % UTSTAR / ship speed
LM_last      = -1;              % last value of the Lagrange Multiplier, LM
 G_last      =  0;              % last value of the circulations, G

A = zeros(Mp+1);                % A matrix for linear system of equations
B = zeros(Mp+1,1);              % B matrix for linear system of equations
```

```
%Estimate the axial induced velocity with actuator disc approx
for m=1:Mp
    UASTAR(m)=0.5*(sqrt(1+CTPDES)-1) + dCirc*UADUCT(m);
end

%Initial TANBIC and TANBIV estimates
[TANBIC,TANBIV] = find_tan_BetaI(VAC,VTC,UASTAR,UTSTAR,RC,RV,Js);

% --------- Iterate to solve simultaneous equations for G, LM, & BetaI. Fix
%           BetaI, and solve simultaneous equations for G and LM.
B_iter = 1;                     % iteration in the BetaI loop
B_res  = 1;                     % residual BetaI between interations
TANBIC_last  = TANBIC;          % the last value of TANBIC


while B_iter < ITER & B_res > 1e-5                          % (WHILE LOOP B1)

    % ------------- Compute the vortex Horseshoe Influence Functions, p.179
    [UAHIF,UTHIF] = Horseshoe(Mp,Z,TANBIV,RC,RV,SCF,Hub_Flag,Rhub_oR,...
                              Duct_Flag,rDuct_oR);

    % -------- Iterate to solve simultaneous equations for G and LM for the
    %          current values of BetaI. (Coney eqns. 2.32 & 2.33, p.27)
     G_iter = 1;                % iteration in the G loop
     G_res  = 1;                % residual G  between interations
    LM_res  = 1;                % residual LM between interations

    while G_iter < ITER & (G_res > 1e-5 | LM_res > 1e-5)  % (WHILE LOOP G1)

        % ----------------------- Solve simultaneous equations for G and LM
        for i = 1:Mp                              % for each vortex  panel, i
            for m = 1:Mp                          % for each vortex  panel, m
                A(i,m) = UAHIF(i,m)*RC(m)*DR(m)   + ...                  % A
                         UAHIF(m,i)*RC(i)*DR(i)   + ...
                         LM_last*UTHIF(i,m)*DR(m) + ...
                         LM_last*UTHIF(m,i)*DR(i);
            end

            A(i,Mp+1) = (VTC(i) + pi*RC(i)/Js)            *DR(i);        % C
            A(Mp+1,i) = (VTC(i) + pi*RC(i)/Js + UTSTAR(i))*DR(i);        % D

            B(i)  = -(VAC(i)+dCirc*UADUCT(i))*RC(i)*DR(i);               % B
        end

        B(Mp+1) = (CTPDES+VD)/(4*Z);                                     % D

        GLM = linsolve(A,B);        % Solve Mp+1 by Mp+1 system of equations

        G   = GLM(1:Mp);            % G is the first Mp entries
        LM  = GLM(Mp+1);            % LM is the last entry

        % ----- Compute induced velocities at control points eqn 254, p.179
```

```
        [UASTAR,UTSTAR] = Induced_Velocity(Mp,G,UAHIF,UTHIF,UADUCT,dCirc);

        % ------ Calculate duct thrust (including propeller influence)
        %        and scale duct circulation (dCirc)
        if Duct_Flag == 1
            [CTD,dCirc,UAdVS,URdVS] = ductThrust(vRingLoc,dVort,dCirc,...
                        rDuct,CDd,XVA(end),rho,RV,G',TANBIV,Z,R,CTDDES);
        end

        % ---------------------------------- Prepare for the next iteration
        G_iter = G_iter + 1             % iteration in the G loop
        G_res  = abs(G  -  G_last);     % residual G  between interations
        LM_res  = abs(LM - LM_last);    % residual LM between interations
        G_last = G;                     % the last value of G
        LM_last = LM;                   % the last value of LM

        if G_iter > ITER
            warning('on'),
            warning('WARNING: While loop G1 did NOT converge.'),
            warning('off'),
        end
    end                                                 % (END WHILE LOOP G1)

    % ------------------ Align the wake to the new circulation distribution
    [UAHIF,UTHIF,UASTAR,UTSTAR,TANBIC,TANBIV] = ...
         Align_wake(TANBIC,TANBIV,ITER,Mp,Z,RC,RV,SCF,Hub_Flag,Rhub_oR,...
                    G,VAC,VTC,Js,Duct_Flag,rDuct_oR,UADUCT,dCirc);

    % ------------------------------------- Prepare for the next iteration
    B_iter = B_iter + 1                 % iteration in the BetaI loop
    B_res  = abs(TANBIC - TANBIC_last); % residual BetaI between interations
    TANBIC_last = TANBIC;               % the last value of TANBIC

    if B_iter > ITER
        warning('on'),
        warning('WARNING: While loop B1 did NOT converge.'),
        warning('off'),
    end

    [CT,CQ,CP,KT,KQ,EFFY,TAU,VD] = Forces(CD,RV,VAC,TANBC,UASTAR,UTSTAR,...
                                CoD,G,Mp,RC,Hub_Flag,Rhv,Z,Js,VMIV,CTDDES);

end                                                     % (END WHILE LOOP B1)

% --------------------- Compute thrust & torque coefficients and efficiency
[CT,CQ,CP,KT,KQ,EFFY,TAU,VD] = Forces(CD,RV,VAC,TANBC,UASTAR,UTSTAR,...
                                CoD,G,Mp,RC,Hub_Flag,Rhv,Z,Js,VMIV,CTDDES);

% % ------- If required, unload the hub and tip, then rescale the circulation
% %         distribution to get the desired value of the thrust coefficient.
% if Hub_Flag & (HUF > 0 | TUF > 0)                       % (IF STATEMENT U1)
%
%     % ---------------------- Unload hub and tip as specified by HUF and TUF
%     RU = (RC - Rhub_oR)./(RoR - Rhub_oR);
%
```

```
%     nH = 4;
%     nT = 3;
%
%     GH = HUF*G(1) *sqrt(1-RU.^2).*(1-RU.^2).^(2*nH-2);
%     GT = TUF*G(Mp)*sqrt(1-RU.^2).*(  RU.^2).^(2*nT-2);
%
%     G  = G - GH' - GT';
%
%     % ============= END determine optimum circulation distribution function
%
%     % ------------------ Align the wake to the new circulation distribution
%     [UAHIF,UTHIF,UASTAR,UTSTAR,TANBIC,TANBIV] = ...
%
Align_wake(TANBIC,TANBIV,ITER,Mp,Z,RC,RV,SCF,Hub_Flag,Rhub_oR,G,VAC,VTC,Js,..
.
%                       Duct_Flag,rDuct_oR,UADUCT,dCirc)
%
%     % ------- Iterate to scale G to get desired value of thrust coefficient
%     CT_iter   = 1;                          % iteration in the CT loop
%     CT_res    = 1;                          % residual CT
%     CT_last2  = 0;                          % the CT prior to the last CT
%     CT_last1  = 0;                          % the last value of CT
%     GMF_last2 = 0;                          % the GMF prior to the last GMF
%     GMF_last1 = 0;                          % the last value of GMF
%
%     while CT_iter < ITER & CT_res > 1e-5                % (WHILE LOOP CT1)
%
%         if CT_iter == 1
%             GMF = 1;
%
%         elseif CT_iter == 2
%             GMF = 1+(CTPDES-CT)/(5*CTPDES);
%
%         elseif CT_iter > 2
%             GMF = GMF_last1 + (GMF_last1-GMF_last2)*(CTPDES-CT_last1)/...
%                               ( CT_last1- CT_last2);
%         end
%
%         G = GMF.*G;                   % GMF = G Multiplication Factor
%
%         % ---- Compute induced velocities at control points. eqn 254, p.179
%         [UASTAR,UTSTAR] = Induced_Velocity(Mp,G,UAHIF,UTHIF,UADUCT,dCirc);
%
%         % ------------- Compute thrust & torque coefficients and efficiency
%         [CT,CQ,CP,KT,KQ,EFFY,TAU,VD] = ...
%
Forces(CD,RV,VAC,TANBC,UASTAR,UTSTAR,CoD,G,Mp,RC,Hub_Flag,Rhv,Z,Js,VMIV,CTD);
%
%         % ---------------------------------- Prepare for the next iteration
%         CT_iter   = CT_iter + 1;           % iteration in the CT loop
%         CT_res    = abs(CT - CTPDES);       % residual CT
%         CT_last2  = CT_last1;               % the CT prior to the last CT
%         CT_last1  = CT;                     % the last value of CT
%         GMF_last2 = GMF_last1;              % the GMF prior to the last GMF
%         GMF_last1 = GMF;                    % the last value of GMF
%
%     end                                                 % (END WHILE LOOP CT1)
```

```
%
% end                                                    % (END IF STATEMENT U1)

% ========================================================= END Coney Function
% ============================================================================


% ============================================================================
% ========================================================= Align_wake Function
%
% This function aligns the wake to the given circulation distribution by
% iteratively computing:
%   UAHIF,UTHIF   = the horseshoe influence functions
%   UASTAR,UTSTAR = the induced velocities
%   TANBIC,TANBIV = the velocity angles
%
% ----------------------------------------------------------------------------
```

C.2 Align_wake.m

```
function [UAHIF,UTHIF,UASTAR,UTSTAR,TANBIC,TANBIV] = ...
        Align_wake(TANBIC,TANBIV,ITER,Mp,Z,RC,RV,SCF,Hub_Flag,Rhub_oR,...
                   G,VAC,VTC,Js,Duct_Flag,rDuct_oR,UADUCT,dCirc)

    % ----------- Iterate to ALIGN WAKE to the new circulation distribution
    W_iter = 1;                     % iteration in the wake alignment loop
    W_res  = 1;                     % residual BetaI between interations
    TANBIW_last = TANBIC;           % the last value of TANBIC

    while W_iter < ITER & W_res > 1e-5                    % (WHILE LOOP WA1)

        % --------- Compute the vortex Horseshoe Influence Functions, p.179
        [UAHIF,UTHIF]   = Horseshoe(Mp,Z,TANBIV,RC,RV,SCF,Hub_Flag,...
                                    Rhub_oR,Duct_Flag,rDuct_oR);
        % ---- Compute induced velocities at control points. eqn 254, p.179
        [UASTAR,UTSTAR] = Induced_Velocity(Mp,G,UAHIF,UTHIF,UADUCT,dCirc);

        % --------------- Compute tan(BetaI) for the new induced velocities
        [TANBIC,TANBIV] = find_tan_BetaI(VAC,VTC,UASTAR,UTSTAR,RC,RV,Js);

        % ---------------------------------- Prepare for the next iteration
        W_iter = W_iter + 1                 % iteration in the BetaI loop
        W_res  = abs(TANBIC - TANBIW_last); % residual BetaI
        TANBIW_last = TANBIC;               % the last value of TANBIC

        if W_iter > ITER
            warning('on'),
            warning('WARNING: While loop WA1 did NOT converge.'),
            warning('off'),
        end
    end                                                   % (END WHILE LOOP WA1)

% ================================================ END Align_wake Function
% ========================================================================
```

Appendix D. Test Case Setup Data

	PLL	OpenProp v2
Advance coefficient, J_S	0.60	
Thrust Coefficient, C_T	1.20	
Blade number	5	
Speed (propeller)	150 RPM	
Drag Coefficient	0.008	
Vortex panels (M_P)	10	
Diameter (prop)	10 ft	3.048 m
Diameter (duct)	10 ft	3.048 m
Speed (ship)	15 ft/s	4.572 m/s
Thrust	21206 lbf	94328 N

Figure D-1: Test case parameters

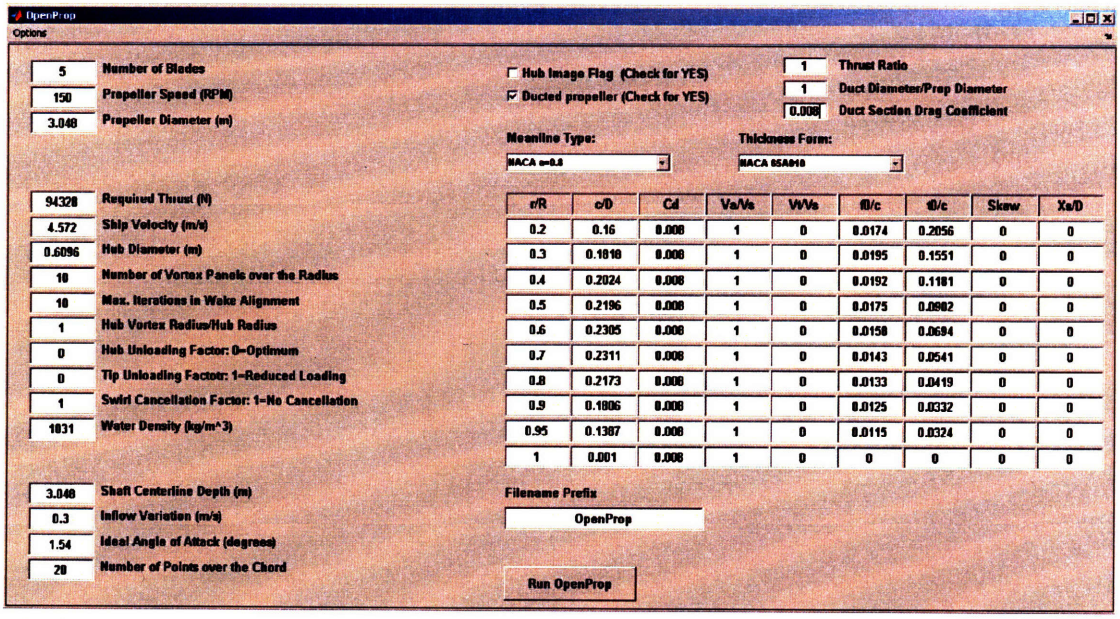

Figure D-2: OpenProp v2 input GUI for test cases

PLL CURRENT SETTINGS (inviscid runs)

1.....circulation optimum enabled....T	2.....chord optimization enabled.....F
3.....wake alignment disabled..... F	4.....a = 0.8 meanline for duct.....T
5.....number of panels..... 10	6.....hub vortex radius..... 1.00
7.....tip thickness/diameter... 0.0040	8.....Lagrange multiplier..... -1.00
9.....max. lift coefficient..... 0.6000	10.....max. thickness/chord..... 0.20
11.....minimum root chord..... 0.1600	12.....enable computed drag coeff.....T
13.....drag coeff. multiplier..... 0.0000	14.....duct tip gap factor..... 1.00

PLL CURRENT SETTINGS (viscid runs)

1.....circulation optimum enabled....T	2.....chord optimization enabled.....F
3.....wake alignment disabled..... F	4.....a = 0.8 meanline for duct.....T
5.....number of panels..... 10	6.....hub vortex radius..... 1.00
7.....tip thickness/diameter... 0.0040	8.....Lagrange multiplier..... -1.00
9.....max. lift coefficient..... 0.6000	10.....max. thickness/chord..... 0.20
11.....minimum root chord..... 0.1600	12.....enable computed drag coeff.....F
13.....drag coeff. multiplier..... 0.0080	14.....duct tip gap factor..... 1.00

Figure D-3: PLL current settings for inviscid and viscid test cases

```
   PROPELLER LIFTING LINE RUN: eta_case                    00/00/94
          OVERALL INPUT FILE
15.0000 ............. Ship speed (ft/sec)
 2.0000 ............. Fluid density
10.0000 ............. Shaft centerline depth (ft)
 1                    Number of components
 N                    No image hub to be used
 Y                    Image duct to be used
 0.5000 ............. (Duct chord length)/(Component #1 diameter)
 0.0000 ............. Drag coefficient for the duct
 0.0000 ............. (Duct thickness)/(Component #1 diameter)
10.0000 ............. Duct diameter (ft)
 0.0000 ............. Axial location of duct mid-chord (ft)
 5                    Number of blades on component 1
10.0000 ............. Diameter of component 1 (ft)
a.bld                 File containing blade inputs for comp. 1
10.0000 ............. Diameter of wake for component 1
a.wak                 File containing wake inputs for component 1
```

Figure D-4: PLL overall input file for test cases

```
PROPELLER LIFTING LINE RUN: eta_case                    00/00/94      00:0
                ************************************************
Js=0.6, Ct=1.2, Inviscid
                ************************************************
                        PLL VERSION 4.2 SUMMARY OUTPUT

 SHIP SPEED (ft/sec):          15.00    FLUID DENSITY:              2.0000
 SHAFT DEPTH (ft):             10.00    # PROP. COMPONENTS:             1
 TOTAL THRUST (lbs):        21203.93    HORSEPOWER:               756.539
 EFFICIENCY:                   0.764    VOL.MN.VEL.(1-W):           1.000
 IMAGE HUB USED:                   F    IMAGE DUCT USED:                T
 WAKE ALIGN. DISABLED:             F    CIRCULATION OPTIMIZED:          T
 TIP GAP FACTOR:              1.0000    CHORDLENGTHS OPTIMIZED:         F
 DUCT DIAMETER (ft):          10.000    DUCT AXIAL LOC. (ft):       0.000
                ************************************************

                                DUCT OUTPUTS

 DUCT AREA RATIO:              2.000    DUCT VOLUME (cu ft):         0.00
 DUCT THRUST (lbs):             0.64    DUCT DIAMETER (ft):       10.0000
 TAU (Tp/T):                  1.0000    DUCT KT:                   0.0000
 CHD/PROP DIAM:               0.5000    DRAG COEFFICIENT:          0.0000
 THICKNESS/P DIAM:            0.0000    DUCT CIRC.:                0.0000
 CT:                          0.0000    CL:                        0.0001
 AXIAL INFLOW VEL:            1.0000    RADIAL INFLOW VEL:         0.0000
 VISCOUS DRAG (lbs):            0.00    INV. THRUST (lbs):            0.64
                ************************************************
                        SUMMARY OUTPUT FOR COMPONENT NUMBER 1

 AXIAL LOCATION(ft):            0.00    DIAMETER (ft):               10.00
 NUMBER OF BLADES:                 5    HUB DIAMETER (ft):            2.00
 NUMBER OF PANELS:                10    REVS. PER MINUTE:           150.00
 BLADE FILE:       a.bld              WAKE DIAMETER (ft):          10.00
 WAKE FILE:        a.wak              THRUST (lbs):             21203.29
 TORQUE (ft-lbs):           26489.53    HORSEPOWER:                756.539
 J SHIP:                      0.600    VOLUMETRIC J:                0.600
 BLADE VOLUME (ft**3):       23.1175   MOM INERTIA (ft**5):         157.17
 EXPANDED AREA RATIO:          0.508

 CT:  1.1999   CQ:  0.1499   CP:  1.5697   KT:  0.1696   KQ:  0.0212
```

Figure D-5: Sample PLL output summary for test case run