Optimum Propeller Design for Electric UAVs

by

David Lee Wall

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 4, 2012

Keywords: Propeller, Design, Optimization

Approved by

Gilbert Crouse, Associate Professor of Aerospace Engineering
Roy Hartfield, Professor of Aerospace Engineering
Brian Thurow, Associate Professor of Aerospace Engineering
George Flowers, Dean of the Graduate School, Professor of Mechanical Engineering

Abstract

A propeller behaves as a rotating wing producing lift in the direction of the axis of rotation. Many previous propeller optimization methods have been developed, but usually focus on piston or turboprop applications. This study discusses the more fundamental propeller theories and uses a hybrid blade element momentum theory to model the propellers. A brushless motor model is developed and coupled with the propeller theory in an optimizer. Two single point optimizations are made, one for a climb condition and the other for a cruise condition. A third optimization is presented with optimization at climb and cruise conditions. The optimizations are conducted with a hybrid pattern/search particle swarm optimizer. The airfoils for the propellers are optimized with the same optimizer and a simplex method. Multiple objective functions are evaluated for each of the conditions. One having non-dimensional values and another with dimensional values. Dimensional values prove to provide better results for all of the conditions. The optimized cruise propellers display smaller chords, higher pitches, and larger diameters while the optimized climb propellers have larger chords, lower pitches and smaller diameters. The multipoint optimization yields higher pitches with chords and diameters between the single point optimizations. All optimized propellers show improvement over comparable baseline propellers.

Acknowledgments

I would like to thank my parents, Larry and Patsy Wall, for their loving support over the years.

I would like to thank Dr. Gilbert Crouse for his guidance with this thesis and studies. I would also like to thank Dr. Brian Thurow and Dr. Roy Hartfield for their help throughout my stay at Auburn University.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

α	Angle of Attack
β	Blade/Helix Angle
ΔP	Change in Pressure
ϵ	Drag to Lift Ratio
η	Efficiency
Γ	Circulation
γ	Reaction Force Angle
λ_m	Flux Linkage of the Stator Winding Per Phase
ϕ	Flow Angle
ϕ_t	Flow Angle at Blade Tip
ρ	Density
σ	Solidity
τ	Electric Motor Torque
f	Parameter in Prandtl Momentum Tip Loss Equation
ξ	Nondimensional Radius
ζ	Displacement Velocity Ratio
A	Area or Disk Area $\left(\frac{\pi D^4}{4}\right)$

a	Axial Interference Factor
a'	Rotational Interference Factor
a_{air}	Speed of Sound
AR	Aspect Ratio
B	Number of Blades
b	Chord of a Blade Element, Axial Slipstream Factor
c	Chord
C_D	Drag Coefficient
C_L	Lift Coefficient
C_p	Power Coefficient
C_t	Thrust Coefficient
D	Diameter, Drag
F	Prandtl Momentum Tip Loss Factor
F_t	Thrust
I	Current
I_0	Idle Current
J	Advance Ratio
K_E	Back E.M.F. Constant
K_t	Motor Torque Constant
K_V	Motor Speed Constant $\frac{RPM}{V}$

L	Lift
m	Number of Phases
n	Rotational Speed
P	Power
p	Number of Poles in Electric Motor
P_0	Total Pressure
p_e	Effective Pitch
p_e	Geometric Pitch
q	Dynamic Pressure
R	Resultant Force in Blade Element
r	Radius
R_I	Internal Resistance
T_{em}	Electromagnetic Torque
u	Relative Velocity in Free Stream Direction
v'	Vortex Displacement Velocity
V_0	Initial/Free Stream Velocity
V_d	Flow Velocity at Disk
V_S	Velocity Downstream/Slipstream
V_{motor}	Voltage
V_{Rel}	Relative Velocity

x	Empirical Inflow Factor
z	Fitness of Objective Function

Chapter 1

Introduction

Propellers are one of the fundamental elements of propulsion and aircraft design, acting like a rotating wing to produce lift in the same direction as the axis of rotation. There are several different methods used to calculate the performance parameters of a propeller. These include momentum theories [1], blade element theories [2], hybrid blade element momentum theories [3][4], and lifting line theories [5]. Procedures have been developed to optimize propellers that do not require computers. With advances in computers, optimization is becoming more readily available and allows for more design variables to be optimized.

One optimization method was developed by Adkins and Liebeck in 1983 [6] and has several limitations, but is easily implemented. This method will only give the optimum blade angles and chords for a particular free stream velocity and will not solve for diameters or multiple design points. Fanjoy and Crossley performed a two dimensional optimization using a genetic algorithm [7]. Their method used a panel method for aerodynamic analysis on the propeller blades and included structural penalty functions to ensure a feasible propeller. This method showed some over prediction in airfoil data which sometimes led to bad results. Miller used a vortex lattice method for a three dimensional optimization of a propeller [8]. One panel was used in this method with no camber. Burger in 2007 [5] developed another method using lifting line theory and a genetic algorithm to optimize propellers for noise reduction over a range of operation.

Due to the fact that a propeller acts like a rotating wing its cross section is an airfoil. Consequently one aspect of optimizing a propeller is optimizing the airfoil shapes used along the blade span. Optimizing an an airfoil shape can be challenging if the shape is described using individual points. It can take 50-100 points to effectively describe an airfoil shape which is to many parameters to optimize efficiently. A common approach to describing the number of parameters used to describe an airfoil is to parameterize the shape. Addressing this obstacle Kulfan has developed a process that uses Bernstein polynomials to represent the points of an airfoil [9]. Her approach was adopted for this effort.

Small UAVs are becoming more popular. Advances in small brushless DC motors and lithium polymer battery technology have created useful drive systems for these UAVs. This leads to a desire to design propellers for UAV systems that are as efficient as possible when used with electric motors. The purpose of this study is to examine propeller optimization with a coupled electric motor. A method will be developed to optimize propellers given a brushless electric motor for single point or multiple point design conditions. A hybrid blade element momentum theory method will be used for propeller performance analysis. Validation of the propeller performance will be included. General trends in propellers at design conditions will be presented along with the results of optimized propellers.

Chapter 2

Airfoils

2.1 Airfoil Basics

Like a wing the cross section of a propeller blade is an airfoil. Airfoils produce a lifting force by creating a low pressure on the surface in the direction of lift and a higher pressure in the opposite direction of lift. An airfoil has several key geometric features. The leading and trailing edge mark the front and back of the airfoil as well as separate the upper surface from the lower surface. The chord, c, is a straight line drawn from the leading edge to the trailing edge. If the upper and lower surfaces are mirror images of each other the airfoil is said to be symmetric. The line consisting of points halfway between the upper and lower surface is known as the mean camber line [10]. The camber is defined as the maximum distance perpendicular to chord line and mean camber line. A visual interpretation of the nomenclature for an airfoil is shown in Figure 2.1.

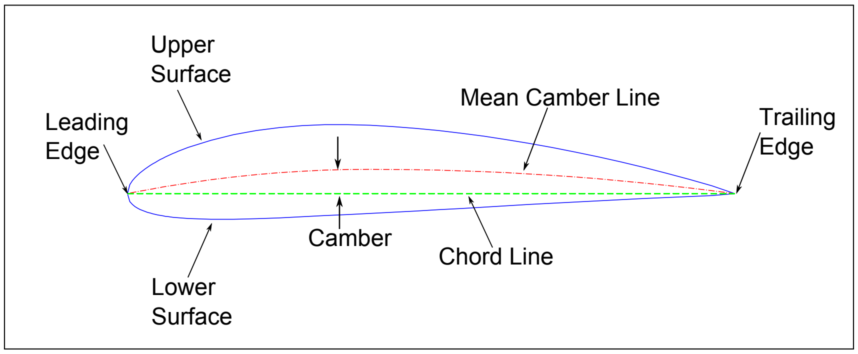

Figure 2.1: Example Airfoil with Nomenclature

2.2 Parameterization of Airfoils

Airfoil shapes are potentially hard to optimize if only the coordinates are known due to the high number of points that need to be used to define an airfoil accurately. To decrease the number of terms used to define an airfoil a process called parameterization is used for upper and lower surfaces.

2.2.1 Bezier Curves

Bezier curves are one of the many ways to represent an airfoil. A parametric Bezier curve of degree n is described in Equation 2.1.

$$B\left(t\right)=\sum_{i=0}^{n}B_i\frac{n!}{i!\left(n-i\right)!}t^i\left(1-t\right)^{n-i} \tag{2.1}$$

Venkataraman [11] used four cubic Bezier curves to describe an airfoil. His method spilt the airfoil into an upper and lower surface and used two curves to define each surface. Rogalsky [12] expanded on Venkataramans work by using four cubic Bezier curves to define a camber and thickness line. The curves can then be combined to form an airfoil. Using Equation 2.2 an example Bezier curve can be constructed and is shown in Figure 2.2.

$$B\left(t\right)=\left(1-t\right)^3B_0+3\left(1-t\right)^2tB_1+3\left(1-t\right)t^2B_2+t^3B_3 \tag{2.2}$$

$B_0,\ B_1,\ B_2,\ and\ B_3$ are coordinate location for each of the control points and t ranges from 0 to 1. Camber and thickness lines can then be made. The first of the two Beizer curves that make up the camber line is anchored at the origin at one end, and the other end is anchored at the location of maximum camber. The second line is anchored at the location of maximum camber and at one unit in the x-direction. The thickness line is constructed in the same manner with the location of maximum thickness between the inner anchored points. The other points used to construct the curves are placed in such a way to

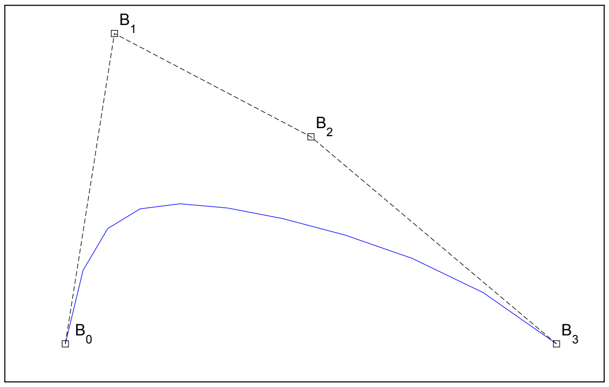

Figure 2.2: Example of Bezier Curve

provide an appropriate curve. These points along with the locations of the maximum camber and thickness are unknown variables that can be moved to create an airfoil. An example camber line, thickness line, and corresponding airfoil are shown in Figure 2.3 with the Beizer coefficients being displayed in the squares and the solid lines representing the upper and lower surfaces.

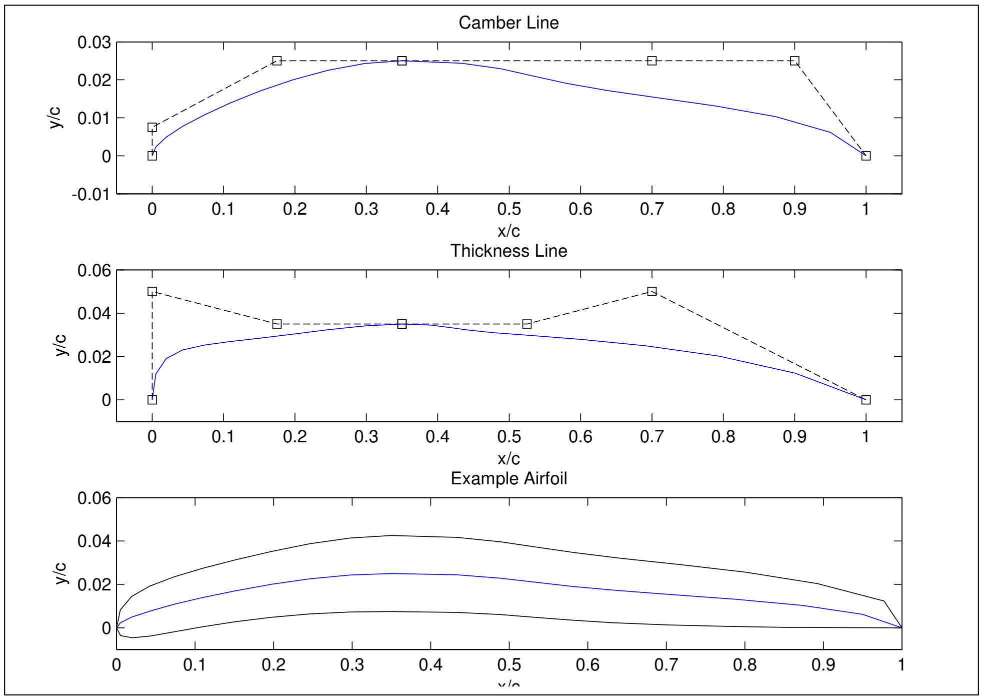

Figure 2.3: Example of Camber Line, Thickness Line and Corresponding Airfoil

2.2.2 Bernstein Polynomials Representation of the Unit Shape Function

Bernstein polynomials are a special case of Bezier curves. Bernstein polynomials only range from 0-1 on the x-axis. Kulfan discusses a method for parameterizing an airfoil using Bernstein polynomials in Reference [9], and her method will be discussed here. The airfoil shape is divided into an upper and lower surface. The following process will need to be repeated for the lower surface. First an overall shape function for the upper surface will be defined in Equation 2.3.

$$S\left(\psi\right) = \sum_{i=1}^{n} Au_i S_i\left(\psi\right) \tag{2.3}$$

Where $\psi = \frac{x}{c}$, $S_i\left(\psi\right)$ is a shape function, and Au_i are the unknown coefficients that define the contour. A unit shape function is then defined by the Bernstein polynomials in Equation 2.4.

$$S_{r,n}\left(x\right) = K_{r,n} x^r \left(1-x\right)^{n-r} \tag{2.4}$$

The Bernstein polynomial is of order n, $r = 0, 1, 2, ..., n$ and $K_{r,n}$ are binomial coefficients shown in Equation 2.5.

$$K_{r,n} \equiv \binom{n}{r} \equiv \frac{n!}{r!\left(n-r\right)!} \tag{2.5}$$

An example of Equation 2.4 evaluated for a 6^{th} order Bernstein polynomial is shown in Table 2.1. A class function can now be introduced in Equation 2.6. This class function defines the

i	$K_{r,n}$	$S_{r,n}\left(x\right)$
0	1	$\left(1-x\right)^6$
1	6	$6\left(1-x\right)^5 x$
2	15	$15\left(1-x\right)^4 x^2$
3	20	$20\left(1-x\right)^3 x^3$
4	15	$15\left(1-x\right)^2 x^4$
5	6	$6\left(1-x\right) x^5$
6	1	x^6

Table 2.1: 6^{th} Order Bernstein Polynomials Parameters

leading and trailing edges of the airfoil.

$$C_{N2}^{N1}(\psi) = \psi^{N1}(1-\psi)^{N2} \tag{2.6}$$

Table 2.2 provides example values of the $N1$ and $N2$ coefficients and the corresponding shape that they yield [13]. A trailing edge offset is defined by Equation 2.7 where z_{te} is the height

N1	N2	Description
0.5	1.0	Round-Nose and Pointed Aft End Airfoil
0.5	0.5	Elliptic or Ellipsoid Body of Revolution
1.0	1.0	Biconvex Airfoil
0.75	0.75	Sears-Haack Body
0.75	0.25	Low-Drag Projectile
1.0	0.001	Cone or Wedge
0.001	0.001	Rectangle, Circular Duct, or Circular Rod

Table 2.2: Class Function Coefficients and Corresponding Geometric Shapes

of the trailing edge from the x-axis.

$$\Delta\xi = \frac{z_{te}}{c} \tag{2.7}$$

An equation for the upper surface, ζ_{upper}, can defined in Equation 2.8 by multiplying Equations 2.4 and 2.6 and adding the offset in Equation 2.7, where $\zeta = \frac{z}{c}$.

$$\zeta_{upper} = C_{N2}^{N1}(\psi)\, S(\psi) + \psi\Delta\xi_{upper} \tag{2.8}$$

The results of an example of this process is shown in Figure 2.4 where the binomial coefficients can be found in Table 2.3. The coefficients for the class function are $N1 = 0.5$ and $N2 = 1.0$, and the leading edge radius is equal to 0.03.

i	0	1	2	3	4	5	6
A_{upper_i}	0.2	0.3	0.2	0.2	0.2	0.1	0.1
A_{lower_i}	-0.2	-0.2	0.2	- 0.2	0.2	- 0.05	-0.05

Table 2.3: Binomial Coefficients for Bernstein Polynomial Example

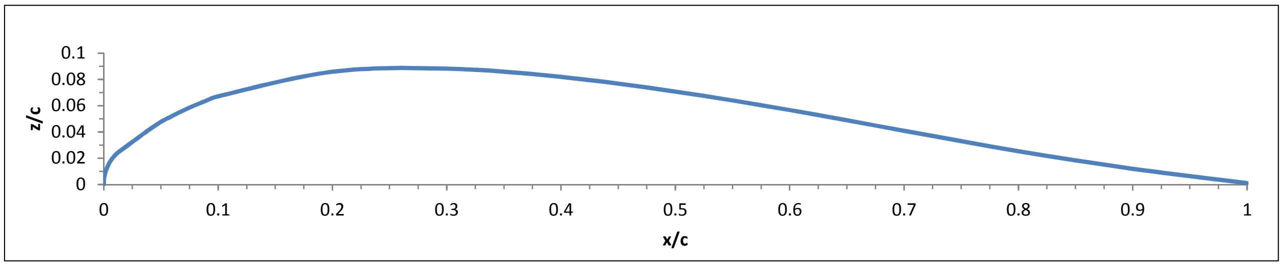

Figure 2.4: Example of Upper Surface Using Parameterization

If this process is repeated for the lower surface different binomial coefficients will need to be used to provide a different contour. The upper and lower surfaces can then be combined to form an airfoil. An example of a randomly generated airfoil is shown in Figure 2.5 using the binomial coefficients from Table 2.3. Kulfan provides results that indicate that this method

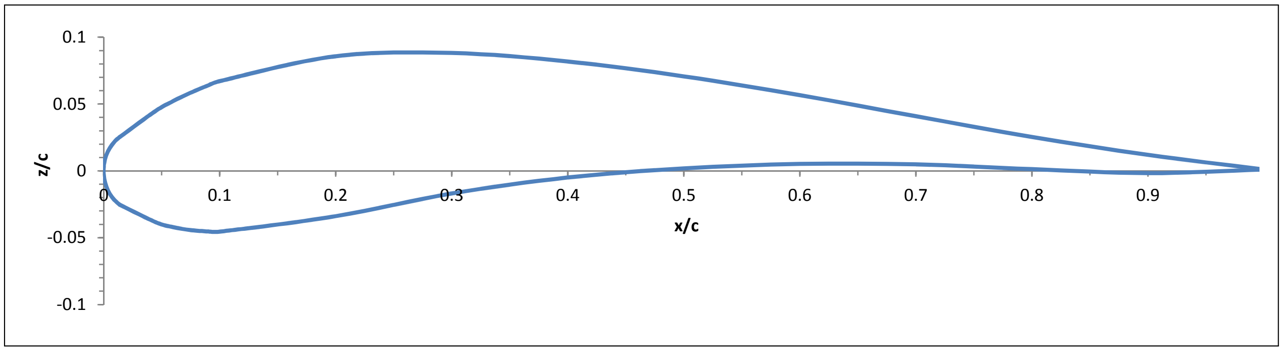

Figure 2.5: Example of a Random Airfoil Using Parameterization

is a suitable method for describing an airfoil when the minimum order for the Bernstein polynomials is higher than 5^{th} order [9].

2.3 XFOIL

There are many different established computer programs and methods for calculating two dimenisonal lift and drag coefficients of an airfoil. A program developed by Drela in 1986 called XFOIL will be briefly discussed here [14] [15]. XFOIL was originally designed for assisting with the development of airfoils for human powered aircraft and low Reynolds number aircraft. Appendix A contains sample inputs and output files for typical sessions where lift and drag are desired.

XFOIL can only provide results for two dimensional airfoils. Its capabilities include both inviscid and viscous solutions. Inviscid solutions are solved using a vortex sheet on the surface of the airfoil and a source sheet the surface of the airfoil and its wake. Once the unknown vortices are found a corresponding pressure distribution and lift coefficient for the airfoil can be found.

The viscous solution is much more complicated process. XFOIL's viscous solution is based on the transonic ISES code with a few improvements. The ISES code solves for the boundary layer and finds separation bubbles using the inviscid solution to solve for the potential flow field. XFOIL includes the Karman-Tsien compressibility correction which is reliable up to sonic conditions and provides reliable pressure distributions, lift, and drag coefficients at low Reynolds numbers [15].

Chapter 3

Propellers

3.1 Basics of Propellers

A propeller is a device used for creating thrust in a fluid through rotational means. Figure 3.1 is velocity diagram for a cross section of a propeller blade. This illustrates that both the free stream and rotation velocities that are seen by the propeller.

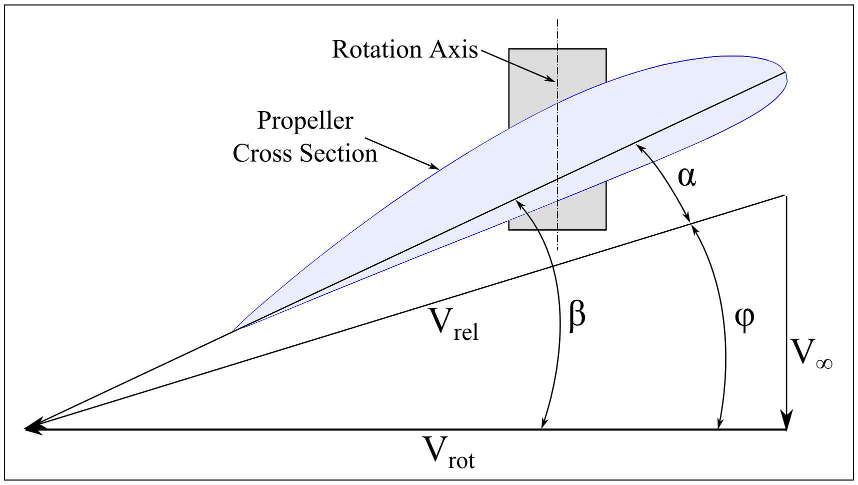

Figure 3.1: Blade Cross Section Velocity Disgram

3.1.1 Geometry

Propellers are very similar to wings. The lifting surface on a propeller is called a blade, and a propeller can have any number of blades. Most propellers have two to four blades. Any given point along a blade the cross section has all the same characteristics as an airfoil: leading and trailing edges, mean camber line, chord line, thickness, etc. Where the blades connect is called the hub which is either directly attached to an engine or to a transmission. The root is the area between the hub and the blade, and the tip is end of the blade opposite the hub. The blade angle, β, is the resultant angle between the free stream and rotational velocity components and is shown in the velocity diagram in Figure 3.1. The effective pitch, p_e, is the distance a propeller advances in one rotation. While the geometric pitch, g_e, is the theoretical distance an element of a propeller blade would travel in one rotation and may not be constant along the length of blade [16] [17]. Several of these geometric paratmeters can be seen in Figures 3.2 and 3.3.

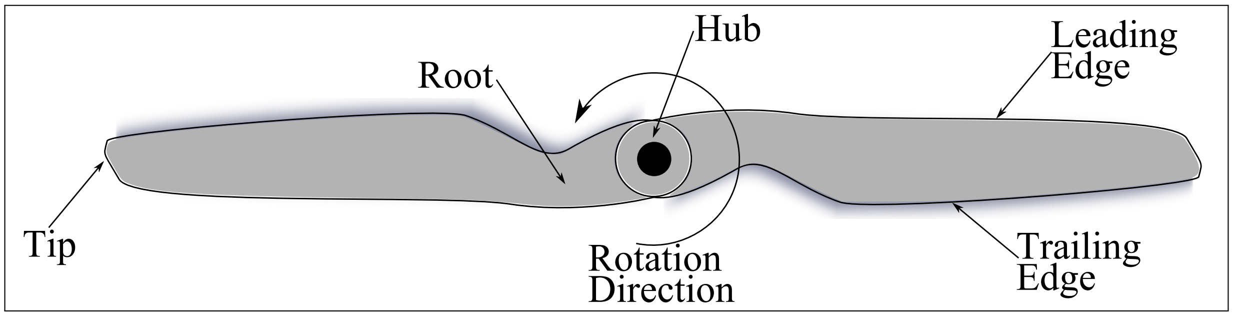

Figure 3.2: Propeller Geometry

3.1.2 Other Parameters

There are many other parameters that are useful in describing propellers. The advance ratio, J, is the ratio between the distance the propeller moves forward through one rotation and the blade diameter.

$$J = \frac{V}{nD} \tag{3.1}$$

Where n is in rotations per second. The aspect ratio, AR, is the tip radius divided by the maximum blade width. A spinner is a conical or parabolic shaped fairing that is mounted over the center of the center of the propeller where it is connected to the hub. The blade face is the lower surface of the propeller airfoil and is also known as the thrust or driving face. The blade back is the upper surface of the propeller airfoil. Several of these parameters are shown in 3.3. The rake or tilt of a propeller is the mean angle between a line drawn through the center of area of each section of a blade and a plane perpendicular to the rotation axis. Some of these parameters are shown in Figure 3.3.

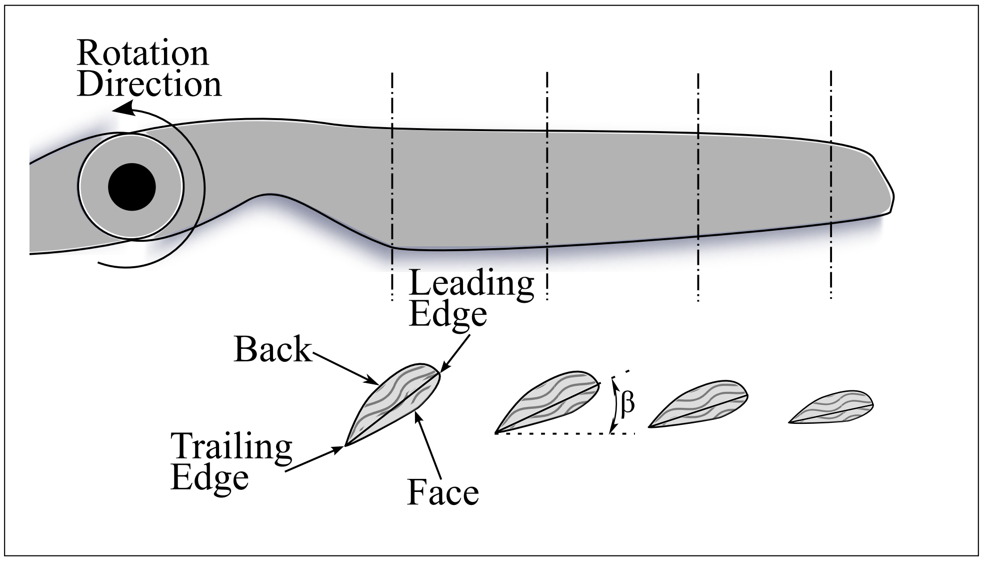

Figure 3.3: Propeller Blade Cross Sections

3.1.3 Types

Propellers are either tractor or pusher propellers. A tractor propeller is placed in a configuration where the engine is downstream of the propeller and pulls the aircraft. While a pusher propeller is placed where the engine is upstream of propeller and pushes the aircraft. Propellers can also be classed as either fixed or variable pitched propellers. A fixed pitch propeller's blades are rigidly connected to the hub. A variable pitch propeller's blades can be adjusted either on the ground or during flight to allow the propeller to operate at maximum performance throughout its operation range.

3.2 Propeller Theories

There are several methods for solving for propeller performance factors. The following discusses a few fundamental methods which are computationally friendly and provide accurate results. Before these methods are explained, nondimensional expressions for thrust, power, and efficiency will be given. These expressions are similar to the lift and drag coefficients used to characterize airfoils and show how the performance of a propeller changes with scale or rotation speed. The thrust coefficient, C_t, power coefficient, C_p, and the efficiency,

η, are shown in Equations 3.2 - 3.4.

$$C_t = \frac{F_t}{\rho n^2 D^4} \tag{3.2}$$

$$C_p = \frac{Q}{\rho n^3 D^5} \tag{3.3}$$

$$\eta = \frac{C_t J}{C_p} \tag{3.4}$$

where n is the rotation speed, D is the Diameter of the propeller, ρ is the density of air, and J is the advance ratio.

3.2.1 Momentum Theory

Momentum theory is most the fundamental of all of the propeller theories. The following explanation for the momentum theory was taken from Nelson in Reference [16]. This theory assumes the propeller is a disk that creates a uniform thrust through a pressure differential between the front and back of the propeller. The theory does not take in account compressibility or viscous effects. Figure 3.4 is a reproduction from Nelson's work in Reference [16] and shows continuous stream lines that form a stream tube. The pressure and velocity

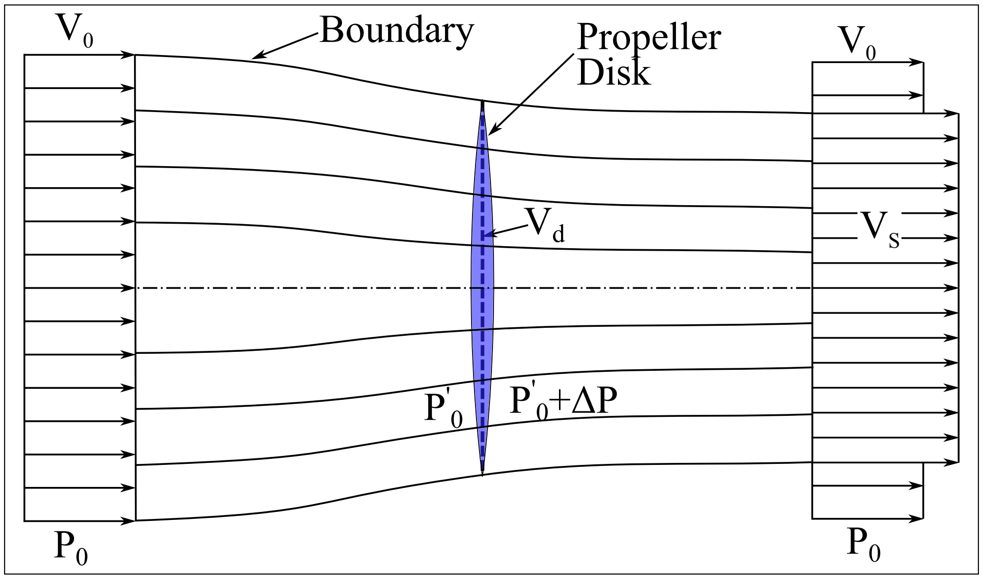

Figure 3.4: Momentum Theory Stream Tube

before and after the disk are shown in Figure 3.5. The far upstream pressure, P_0, is shown to change by ΔP at the propeller disk then return to P_0 far downstream. It should also be noted that the pressure drops by $P_0 - P_0'$ at the beginning of the disk then quickly rises by ΔP before asymptotically returning to P_0.The velocity is shown to start at V_0 upstream and slowly rise to a final value of V_s. With this information thrust from a propeller can be

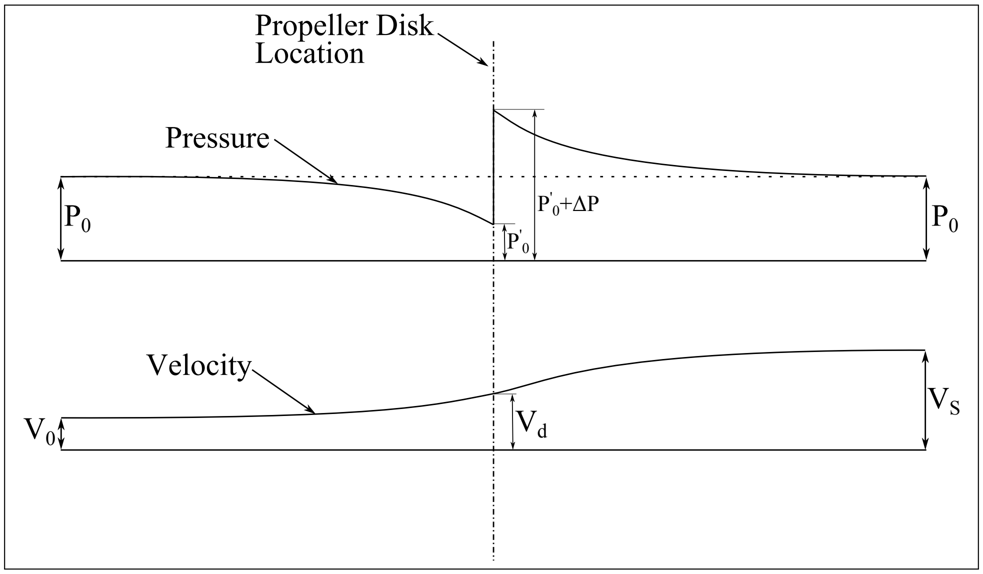

Figure 3.5: Momentum Theory Pressure and Velocity through Propeller Disk

calculated using classic momentum theory.

$$F_t = A\rho V_d \left(V_S - V_0\right) \tag{3.5}$$

where $A\rho V_d$ is the mass per unit time through the disk and $(V_s - V_0)$ is the velocity increase from far upstream to far downstream. The pressure change across the propeller disk, velocities upstream and downstream, and the area of the propeller can be used to calculate the thrust using Bernoulli's equation.

$$\begin{aligned} F_t &= A\Delta P \\ \Delta P &= \left[P_0 + \frac{1}{2}\rho V_s^2\right] - \left[P_0 + \frac{1}{2}\rho V_0^2\right] \\ \Delta P &= \frac{1}{2}\rho\left(V_s^2 - V_0^2\right) \end{aligned}$$

$$F_t = \frac{A\rho}{2}\left(V_S^2 - V_0^2\right) \tag{3.6}$$

Combining Equations 3.5 and 3.6 the velocity at the disk, V_d, can be found.

$$\begin{aligned} A\rho V_d\left(V_S - V_0\right) &= \frac{A\rho}{2}\left(V_S^2 - V_0^2\right) \\ V_d &= \frac{\frac{A\rho}{2}\left(V_S^2 - V_0^2\right)}{A\rho\left(V_S - V_0\right)} \end{aligned}$$

$$V_d = \frac{V_s + V_0}{2} \tag{3.7}$$

It can then be seen that half of the downstream velocity, V_S, is added before the propeller disk.

Efficiency is defined as the work output divided by the input work. Kinetic energy can be used to describe the input work, and thrust times velocity defines the work output. The following process shows the efficiency in terms of the free stream velocity and the downstream velocity.

$$\begin{aligned} \eta &= \frac{TV_0}{K.E.} \\ &= \frac{TV_0}{\left(\frac{1}{2}A\rho V_d\right)\left(V_S^2 - V_S^2\right)} \end{aligned}$$

plug in V_d from Equation 3.7, and F_T from Equation 3.6;

$$\begin{aligned}\eta &= \frac{\frac{A\rho}{2}\left(V_S^2 - V_0^2\right)V_0}{\left(\frac{1}{2}A\rho\frac{V_s+V_0}{2}\right)\left(V_S^2 - V_0^2\right)} \\ &= \frac{2A\rho V_0\left(V_S^2 - V_0^2\right)}{\left(A\rho\left(V_s + V_0\right)\right)\left(V_S^2 - V_0^2\right)} \\ &= \frac{2V_0}{V_S + V_0} \\ &= \frac{2}{1 + \frac{V_S}{V_0}} \end{aligned} \tag{3.8}$$

$$\frac{F_t}{2\rho V_0^2} = \frac{1-\eta}{\eta^2} \tag{3.9}$$

Equations 3.8 and 3.6 can be combined to get a theoretical efficiency in terms of density, free stream velocity, disk area, and thrust.

Equation 3.9 can be used to observe how thrust, free stream velocity, and disk area affect the efficiency of a propeller. Figure 3.6 shows efficiency verses the thrust coefficient, C_t. Where C_t is a dimensionless thrust found in Equation 3.10.

$$C_t = \frac{F_t}{\frac{1}{2}\rho V_0^2 A} \tag{3.10}$$

This shows that increasing thrust a propeller produces, slowing the free stream velocity, or decreasing the propeller disk area decreases the efficiency. The most efficient propeller would then produce no thrust, have a high free stream velocity, and be infinitely large.

3.2.2 Simple Blade Element Theory

Blade element theory is the next fundamental propeller theory. This theory separates each blade of a propeller into elements and calculates the lift and torque generated by each of the element. The thrust and torque are then summed up to find the total thrust and torque. The following process is a summarized explanation of Dommasch, Nelson, and Weick's work in References [2], [16], and [17]. An example blade element is shown in Figure 3.7 which a

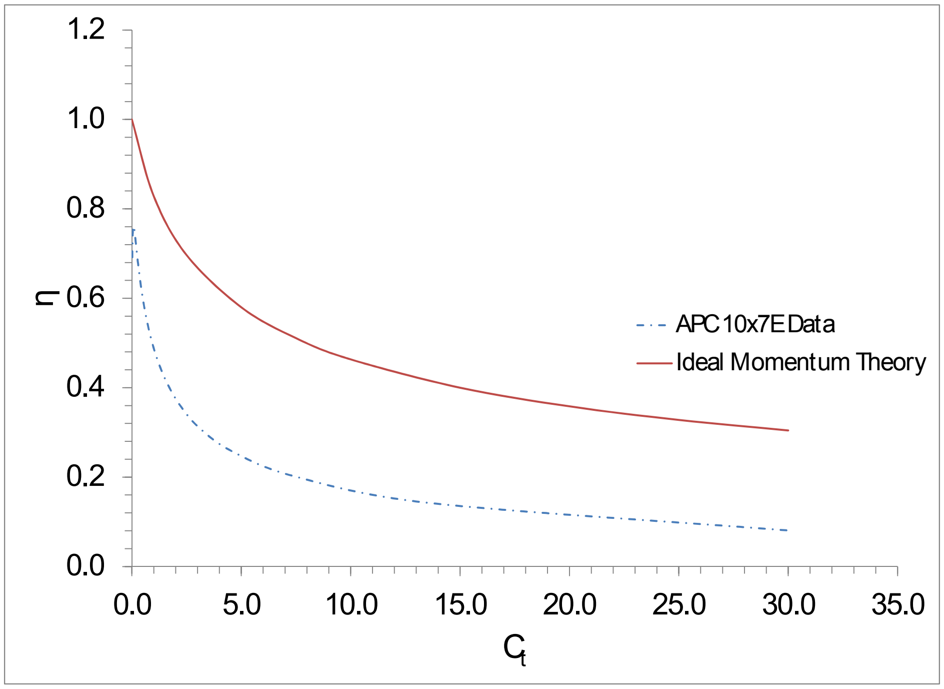

Figure 3.6: Ideal Momentum Theory Efficiency and Actual Propeller Efficiency Verse Thrust Coefficient

reproduction from Dommasch's work in Reference [2]. First consider an individual element. The lift and drag on the element can be calculated using a differential form of the classic lift and drag calculations.

$$dL = qC_L b dr \tag{3.11}$$

$$dD = qC_D b dr \tag{3.12}$$

Where the dynamic pressure is defined in Equation 3.13.

$$q = \frac{1}{2}\rho V_{Rel}^2 \tag{3.13}$$

V_{Rel} is the relative flow velocity which is shown in Figure 3.8 which is a slightly modified version of Figure 3.1 discussed in Section 3.1.

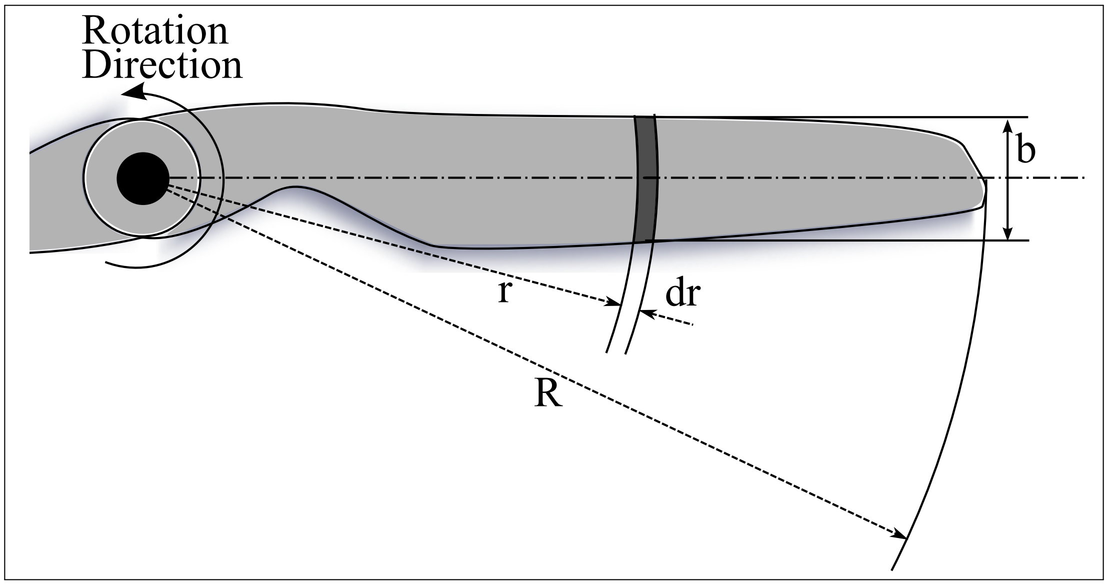

Figure 3.7: Example of a Blade Element

Using the reaction force diagram in Figure 3.8 the differential thrust can be calculated by adding the lift and drag components.

$$dF_t = dL\,cos\left(\phi\right) - dD\,sin\left(\phi\right) \tag{3.14}$$

Substitute Equations 3.11, 3.12, and 3.13 into Equation 3.14.

$$dF_t = \frac{1}{2}V_{Rel}^2 b dr\left(C_L cos\left(\phi\right) - C_D sin\left(\phi\right)\right) \tag{3.15}$$

A new angle γ, which describes the reaction force of the lift and drag components can now be used to simplify the differential thrust equation and is defined in Equation 3.16.

$$tan\left(\gamma\right) = \frac{D}{L} = \frac{C_D}{C_L} \tag{3.16}$$

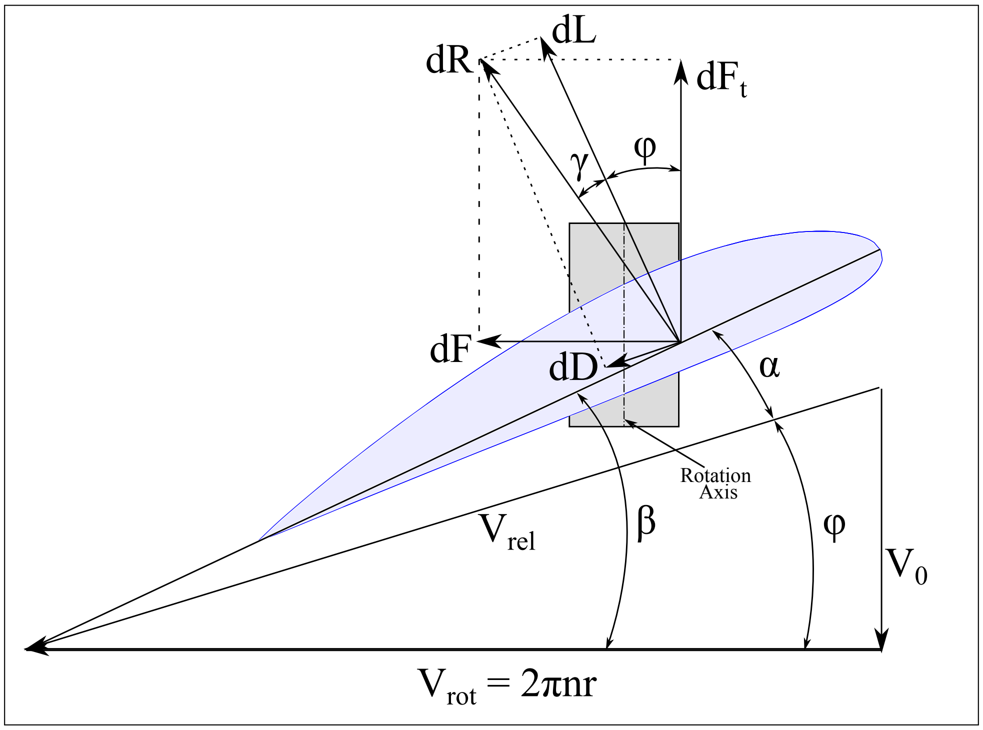

Figure 3.8: Velocity Vector Diagram with Reactions on a Blade Element

The following process simplifies the differential thrust equation into its final form where $V_{Rel} = \frac{V_0}{sin(\phi)}$.

$$\begin{aligned}
dF_t &= \frac{1}{2}\rho V_{Rel}^2 bdr\left(C_L cos\left(\phi\right) - C_D sin\left(\phi\right)\right)\\
&= \frac{1}{2}\rho V_{Rel}^2 bdr\left(\frac{C_L}{C_L}C_L cos\left(\phi\right) - \frac{C_L}{C_L}C_D sin\left(\phi\right)\right)\\
&= \frac{1}{2}\rho V_{Rel}^2 bdrC_L\left(cos\left(\phi\right) - tan\left(\gamma\right)sin\left(\phi\right)\right)\\
&= \frac{1}{2}\rho V_{Rel}^2 bdrC_L\left(cos\left(\phi\right) - \frac{sin\left(\gamma\right)}{cos\left(\gamma\right)}sin\left(\phi\right)\right)\\
&= \frac{1}{2}\rho V_{Rel}^2 bdrC_L\left(\frac{cos\left(\gamma\right)cos\left(\phi\right) - sin\left(\gamma\right)sin\left(\phi\right)}{cos\left(\gamma\right)}\right)\\
&= \frac{1}{2}\rho V_{Rel}^2 bdrC_L\left(\frac{cos\left(\gamma+\phi\right)}{cos\left(\gamma\right)}\right)\\
&= \frac{1}{2}\rho V^2 bdrC_L\left(\frac{cos\left(\gamma+\phi\right)}{sin^2\left(\phi\right)cos\left(\gamma\right)}\right) \qquad (3.17)
\end{aligned}$$

The torque can be found in the same manner using dF found in Figure 3.7.

$$\begin{aligned} dQ &= r\ dF \\ &= \frac{1}{2}\rho V^2 br\, dr C_L \left(\frac{sin\left(\gamma+\phi\right)}{cos\left(\gamma\right) sin^2\left(\phi\right)}\right) \end{aligned} \tag{3.18}$$

Equations 3.17 and 3.18 can then be integrated across the radius of the blade to find the total thrust and torque generated by one blade. That value is multiplied by the number of blades, B, to find the total thrust and total torque for the propeller.

$$F_t = \int_0^r \frac{1}{2} V^2 b dr B C_L \left(\frac{cos\left(\gamma+\phi\right)}{sin^2\left(\phi\right) cos\left(\gamma\right)}\right) \tag{3.19}$$

$$Q = \int_0^r \frac{1}{2}\rho V^2 br\, dr B C_L \left(\frac{sin\left(\gamma+\phi\right)}{cos\left(\gamma\right) sin^2\left(\phi\right)}\right) \tag{3.20}$$

The efficiency can be found with the same process discussed in the momentum theory.

$$\eta = \frac{F_t V_0}{2\pi n Q} \tag{3.21}$$

Nelson in Reference [16] says that if the efficiency is taken at a three quarter radius element that it would characterize the efficiency of the entire blade. He then describes a process to show the efficiency only it terms of the effective pitch angle, ϕ and the reaction angle, γ.

$$\begin{aligned} \eta &= \frac{dTV}{dQ 2\pi n} \\ &= \frac{dR cos\left(\gamma+\phi\right) V}{dR sin\left(\gamma+\phi\right) 2\pi n r} \\ &= \frac{tan\left(\phi\right)}{tan\left(\gamma+\phi\right)} \end{aligned} \tag{3.22}$$

Figure 3.9 is a reproduction of Nelson's work. It plots multiple curves which each correspond to different lift to drag ratio blade elements. It shows that a realistic maximum efficiency a propeller can have is approximately 93%, but using an average lift to drag ratio would be

80%. This figure also shows theoretical maximum effective pitch angle which is shown by

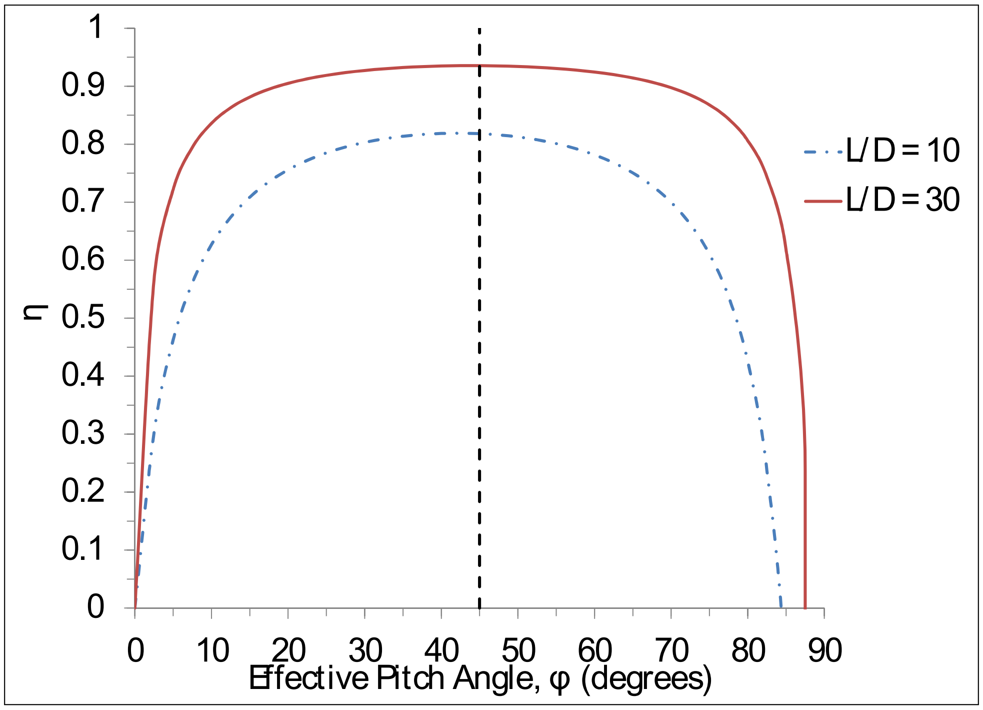

Figure 3.9: Simple Blade Element Helix Angle Efficiency

the dashed line. If it is assumed that the propeller tip, the fastest location on a propeller, cannot have a relative velocity exceeding the speed of sound, a_{air}, then at the three quarters location it cannot exceed three quarter the speed of sound. An effective pitch angle of 45° is shown to provide the highest efficiency.

$$tan\,(\phi) = \frac{V_0}{V_{Rot}} = \frac{V_0}{\frac{3}{4}a_{air}} \tag{3.23}$$

If the speed of sound is approximately 1000 feet per second then using Equation 3.23 an efficiency verse free stream velocity curve can be drawn in Figure 3.10.. It should be noted that the free stream velocity corresponds to an effective pitch angle that provides the maximum efficiency. The simple blade element theory agrees with the momentum theory that efficiency of a propeller is increased with a higher free stream velocity. The simple blade element theory adds that the blade angle also needs to increase for added efficiency.

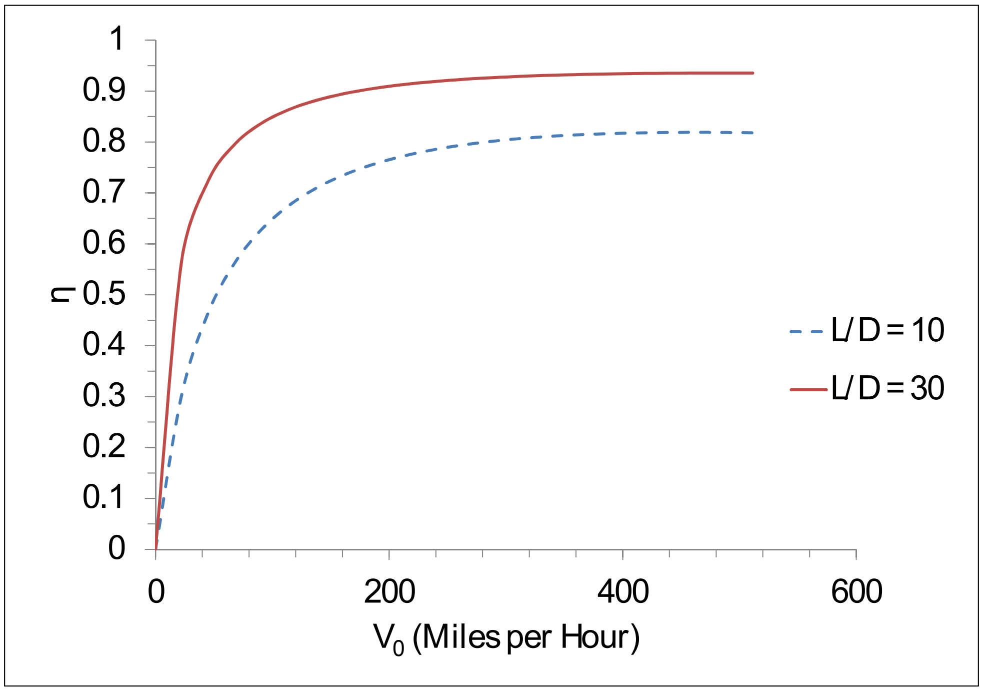

Figure 3.10: Simple Blade Element Free Stream Velocity Efficiency

3.2.3 Hybrid Momentum Blade Element Theory

There are many different hybrid momentum blade element theories. Two similar methods will be discussed here. These methods introduce factors to account for radial flow, blade interference, and tip losses.

Axial Slipstream Factor Method

First will be a method from Weick in Reference [17]. Earlier in the momentum theory section it was shown that half of the velocity increase is in front of the propeller in the slipstream. This leads to a new term known as inflow. Inflow is added to the free stream velocity to increase the overall velocity of the flow in the axial direction. Figure 3.11 is a reproduction from Weick's work showing the additional velocity included in the axial direction. It is shown that the velocity in the free stream direction is given by Equation 3.24 and a new angle of attack needs to be found for the blade element using Equations 3.25 and

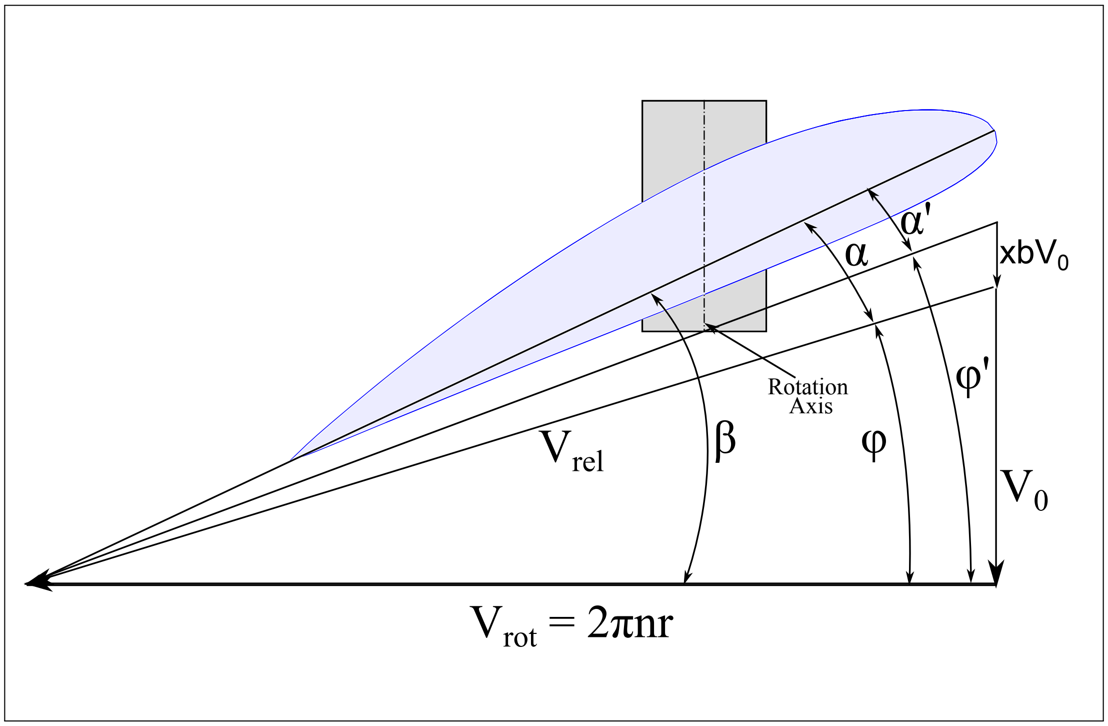

Figure 3.11: Weick's Inflow Method Velocity Vectors

3.26.

$$u = V_0 + xbV_0 \tag{3.24}$$

$$tan\left(\phi'\right) = \left(1 + xb\right) tan\left(\phi\right) \tag{3.25}$$

$$\alpha' = \beta - \phi' \tag{3.26}$$

The inflow for this model is an average inflow distributed over the whole blade. More robust methods calculate the inflow per blade element. According to Weick's method inflow can be determined by Equation 3.27. Where b in this case is not the chord but is the axial slipstream factor and x is an empirical factor that ranges from one third to two thirds.

$$F_t = A'\rho V^2 b\left(1 + xb\right) \tag{3.27}$$

A' is the effective disk area and is given to be between 0.7 and 0.8. The differential thrust can then be found using the same process derived in the in the previous section. Where b in

Equations 3.28 and 3.29 is the chord for the blade element.

$$dF_t = \frac{1}{2}\rho u^2 b dr C_L \left(\frac{cos\,(\gamma + \phi')}{sin^2\,(\phi')\, cos\,(\gamma)} \right) \tag{3.28}$$

$$dQ = \frac{1}{2}\rho u^2 br\, dr C_L \left(\frac{sin\,(\gamma + \phi')}{cos\,(\gamma)\, sin^2\,(\phi')} \right) \tag{3.29}$$

This method must be iterated to find the value of the axial slipstream factor, b. This is executed by establishing an initial guess for the flow angle ϕ. Equation 3.28 is evaluated, and the thrust found is substituted in Equation 3.27. The axial slipstream factor, b, is then solved and now ϕ' can be found. This ϕ' is plugged into Equation 3.28 and a new thrust is solved. The new thrust and the old thrust are compared. If they are not within an acceptable tolerance, the new thrust is plugged in Equation 3.27 and a new axial slipstream factor is found. This process is repeated until an acceptable tolerance is achieved. One problem with this method is error introduced by the empirical factor x. This factor varies from propeller to propeller leading to inconsistent results [17].

Inflow with Axial and Rotational Interference Factors

The second method that will be discussed is an optimum design paper developed by Adkins and Liebeck in 1983 and was reproduced with more detail in 1994 [3]. This is the same procedure discussed by Glauert in Reference [1] with updated nomenclature and a more detailed explanation. This method also uses an axial interference factor, a, and a rotational interference factor, a'. Using momentum theory the axial interference flow factor, a, is the increase in flow velocity in front of the propeller, and the rotational interference flow factor, a', is the decrease in the relative rotational flow velocity. Figure 3.12 shows these new modifications to velocity vector diagram for a blade element. Using momentum theory and Figure 3.12 it can easily be shown that the thrust per unit radius is given by Equation

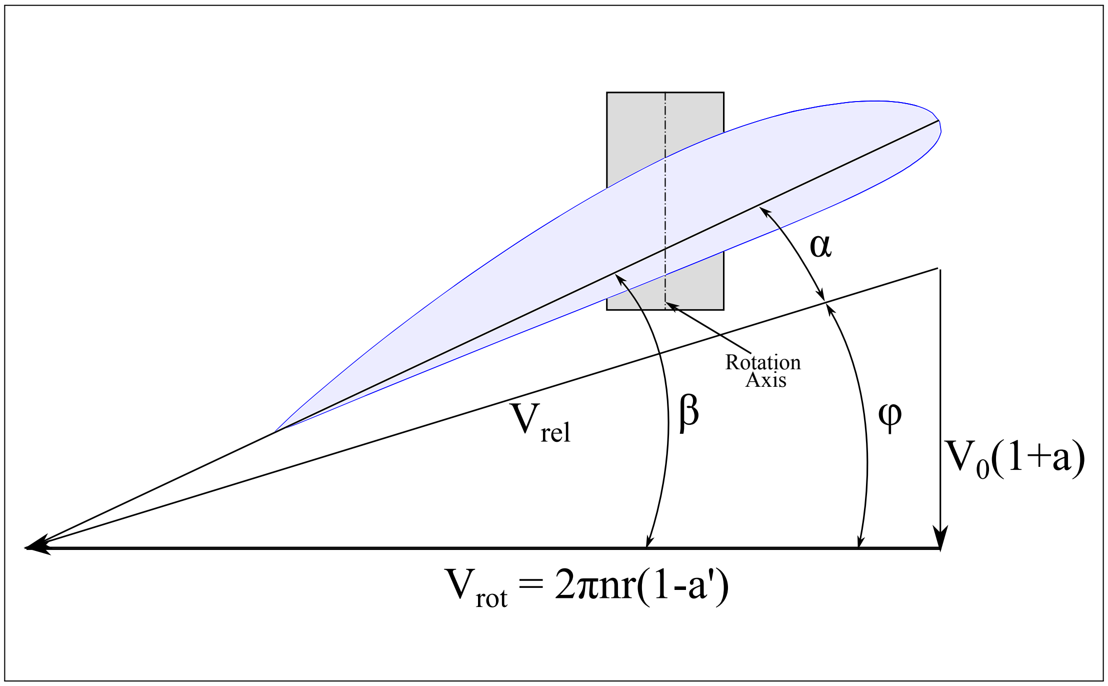

Figure 3.12: Axial and Rotational Interference Factor Blade Element Velocity Vectors

3.30, and the torque per unit radius is given by Equation 3.31.

$$dF_t = 4\pi\rho rV^2a\,(1+a)\,F\,dr \tag{3.30}$$

$$\frac{dQ}{r} = 8\pi^2\rho nr^2Va'\,(1+a)\,F\,dr \tag{3.31}$$

With F being the Prandtl momentum tip loss factor. Adkins does not discuss this factor in any detail in his paper, but it is discussed in more detail by Glauert in Reference [4]. It was originally developed by Prandtl and describes the losses due to induced velocities at the tip of the blade. This loss factor would be equal to 1.0 across the radius if the propeller is shrouded or in a duct. If it is not it will start at 1.0 and at a given location will begin to decay to 0.0. The location for the start of the decay is due to geometry. Prandtl's original method used the inflow angle based on the blade tip which Glauert later revised to using the local inflow angle. Adkins returns to Prandtl's original method by using the inflow angle at the tip [3] [4]. The Prandtl momentum tip loss factor is expressed in Equation 3.32.

$$F = \frac{2}{\pi} cos^{-1}\left(e^{-f}\right) \tag{3.32}$$

where,

$$f = \frac{B}{2}\frac{(1-\xi)}{sin\left(\phi_t\right)} \tag{3.33}$$

and the flow angle at the tip, ϕ_t is defined by

$$tan\left(\phi_t\right) = \xi tan\left(\phi\right) \tag{3.34}$$

Adkins includes the addition of circulation into this method. Which a simplified version of the circulation is defined by Equation 3.35. The circulation equation introduces ζ which is the displacement velocity ratio, $\frac{v'}{V_0}$. The vortex displacement velocity, v', is the axial velocity of the vortex filament in the vortex sheet in the wake of the propeller.

$$\Gamma = \frac{2\pi r V_0 \zeta F}{B} cos\left(\phi\right) sin\left(\phi\right) \tag{3.35}$$

This circulation is used to described the lift per unit radius and is given by Equation 3.36. The thrust per unit radius and torque per unit radius can be found using Equation 3.36 and Figure 3.8. With ϵ being the drag to lift ratio.

$$\frac{dL}{dr} = B\rho V_{rel}\Gamma \tag{3.36}$$

$$dF_t = \left(\frac{dL}{dr} cos\left(\phi\right)\left(1 - \epsilon\, tan\left(\phi\right)\right)\right) dr \tag{3.37}$$

$$\frac{dQ}{r} = \left(\frac{dL}{dr} sin\left(\phi\right)\left(1 + \frac{\epsilon}{tan\left(\phi\right)}\right)\right) dr \tag{3.38}$$

C_y, C_x, C_l, and C_d replacing dF_t, dF, dL, and dD respectively in Figure 3.8.

$$C_y = C_l cos\left(\phi\right) - C_d sin\left(\phi\right) = C_l \left(cos\left(\phi\right) + \epsilon\, sin\left(\phi\right)\right) \tag{3.39}$$

$$C_x = C_l sin\left(\phi\right) + C_d cos\left(\phi\right) = C_l \left(sin\left(\phi\right) + \epsilon\, cos\left(\phi\right)\right) \tag{3.40}$$

The axial and rotational interference factors are expressed in Equations 3.41 and 3.42.

$$a = \frac{\sigma K}{F - \sigma K} \tag{3.41}$$

$$a' = \frac{\sigma K'}{F + \sigma K'} \tag{3.42}$$

where,

$$K = \frac{C_y}{4sin^2\left(\phi\right)} \tag{3.43}$$

$$K' = \frac{C_x}{4cos\left(\phi\right) sin\left(\phi\right)} \tag{3.44}$$

The solidity, σ, is defined by Equation 3.45.

$$\sigma = \frac{Bb}{2\pi r} \tag{3.45}$$

Figure 3.12 shows these new modifications to velocity vector diagram for a blade element. It is easily shown that the flow angle is Equation 3.46.

$$tan\left(\phi\right) = \frac{V}{2\pi n r}\frac{\left(1 + a\right)}{\left(1 - a'\right)} \tag{3.46}$$

The procedure for solving for the thrust, torque, and efficiency for a propeller goes as follows. An initial guess for ϕ is found by setting a and a' equal to zero in Equation 3.46. This ϕ then allows for the angle of attack, α, for the blade element to be found which also yields the lift and drag coefficients. Equations 3.41 and 3.42 can then be solved for the interference factors.

The interference factors are plugged back into Equation 3.46 to solve for a new flow angle, ϕ. The new and old values of ϕ are compared, and if they are not within a set tolerance of each other they average of the two values is used to repeat the process. These iterations continue until the solution has converged within the set tolerance. It should also be noted that Adkins suggests using a clipping method by Viterna and Janetzke [18] in which a and a' are limited to a maximum value of 0.7. Once the convergence is met the thrust coefficient and power coefficient for the propeller can be found and is shown in Equations 3.47 and 3.48. These equations can be integrated to solve for the overall thrust and power coefficients. The coefficients can be substituted into Equations 3.2-3.4 for final thrust, power, and efficiency values.

$$\frac{dC_t}{d\xi} = \frac{\pi^3}{4}\sigma\xi^3 F^2 \frac{C_y}{\left[(F + \sigma K')\, cos\,(\phi)\right]^2} \tag{3.47}$$

$$\frac{dC_p}{d\xi} = \frac{dC_t}{d\xi}\pi\xi\frac{C_x}{C_y} \tag{3.48}$$

Detailed derivations for Equations 3.47 and 3.48 are shown in Appendix B. This derivation presented corrects an error in the differential thrust equation in Adkins [3].

Chapter 4
Electric Motors

Electric motors are being used more frequently in small UAVs due to the high energy density lithium polymer batteries that are becoming more available and the overall decrease in sound production of the electric propulsion system compared to an internal combustion system. Two types of electric motors will be briefly discussed here. First a brushed DC motor will be introduced followed by a brushless DC motor. More details for DC motors can found in References [19], [20], [21], and [22].

4.1 Brushed DC Electric Motors

A brushed DC electric motor as the name implies uses a DC voltage source for power. The motor consists of a rotor, stator, field system, armature, brushes, and a commutator which are several of main components. The rotor is the rotating part of the motor. The stator is the stationary part of the motor. The field system provides the magnetic flux used to create the torque on the motor. The armature carries the current that interacts with the field flux to create torque. For most brushed DC motors the rotor and the armature are one in the same since the rotor will have windings that are used to move the current from the brushes and commutator to the rotor. The brushes connect the armature to the power supply and a motor requires a minimum of one pair of brushes. The commutator distributes the current properly to the armature coils [19]. With these components the motor can be modeled as a resistor or armature resistance and the back e.m.f. A simple equivalent circuit of a DC motor is shown if Figure 4.1. Where the back e.m.f. is the back electromotive force, and K_E is the back e.m.f. constant.

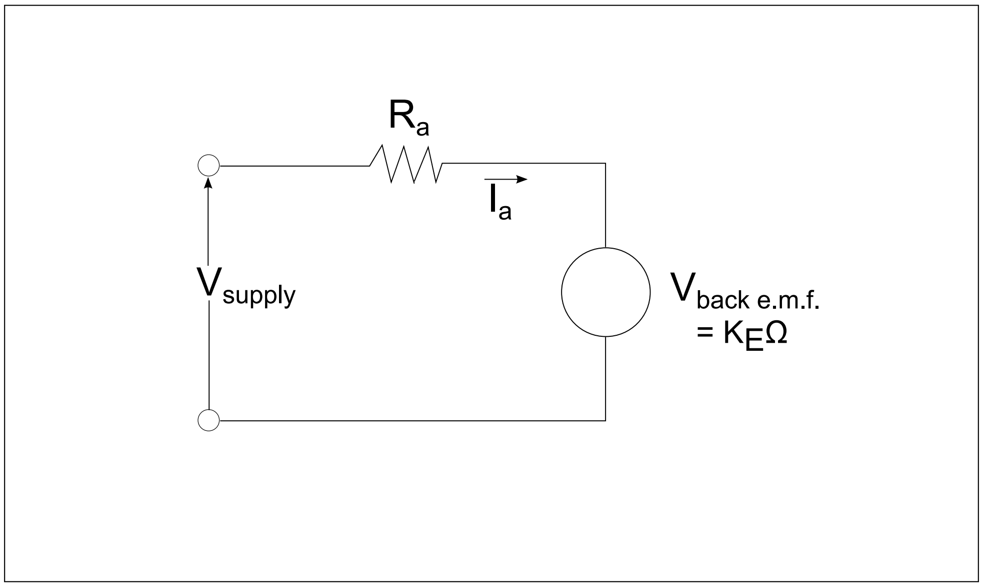

Figure 4.1: A Simple Equivalent Circuit of a DC Motor

For the scope of this work it is now assumed that when all of the previous components are combined the motor is a torque generator. The motor generates torque according to Fleming's left hand rule. A detailed explanation of how a DC motor produces torque can be found in Reference [19]. The torque of a motor can then be found using the torque constant, K_t, and the current draw, I, on the motor.

$$\tau = K_t I \tag{4.1}$$

If the constant units are in the SI system K_E is equal to K_t and if the imperial system is used $1.352\,K_t = K_E$. Another useful term to describe a DC electric is the K_V value. This has units of $\frac{RPM}{V}$. This motor speed constant is used to find the motors speed with no load given a particular input voltage. Equation 4.2 shows the relation between the torque and speed constants where K_t has units of $\frac{oz-in}{A}$ [21].

$$K_t = \frac{1000}{K_V} 1.352 \tag{4.2}$$

A torque speed curve can be produced using Equations 4.1 and 4.2. The maximum/stall torque is found from the stall current in Equation 4.3 and Equation 4.1 with R_I being the internal resistance of the motor. The maximum RPM is found using the motor voltage and the K_v value. A straight line is connected between the maximum RPM and the stall torque to form the torque speed curve. Figure 4.2 shows the torque speed curve has a negative slope, and it is shown that as the voltage is increased the curves remain parallel with a new maximum/stall torque and peak rotational speed.

$$I_{stall} = \frac{V_{motor}}{R_I} \tag{4.3}$$

The power required for motor is then found using Equation 4.4. A power curve can be

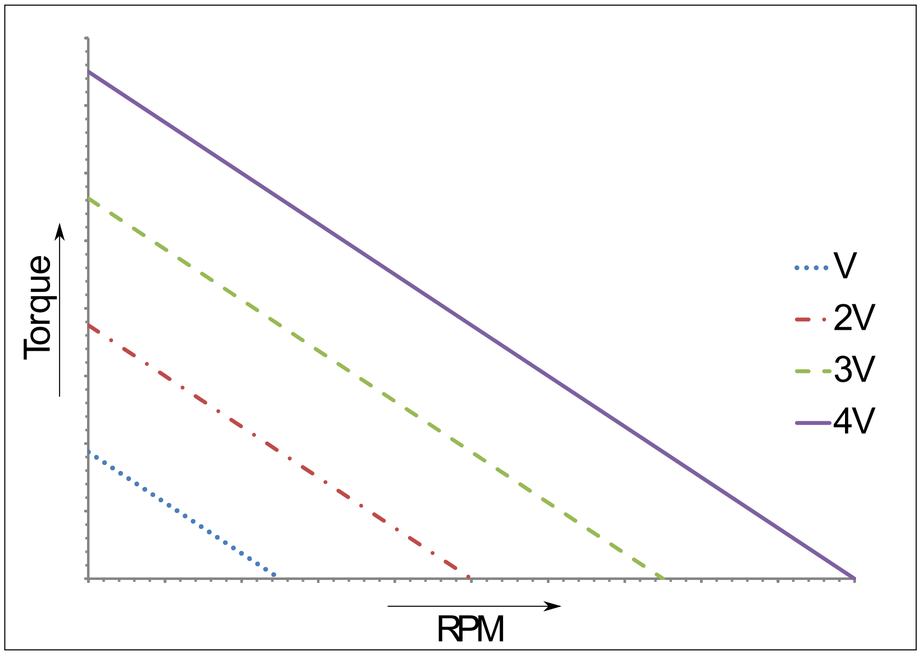

Figure 4.2: Torque Speed Curve Example for a DC Motor

added to the torque speed curve. Figure 4.3 shows an example torque speed power curve for an arbitrary DC motor with $K_v = 1000$, $K_t = 1.5\left(\frac{oz-in}{A}\right)$, $R_I = 0.2\,\Omega$, and $V = 11.1\,Volts$.

$$P = \tau\frac{2\pi n}{60} = V_{motor}I \tag{4.4}$$

Figure 4.3 is a useful tool to describe a motor. It shows that the maximum power con-

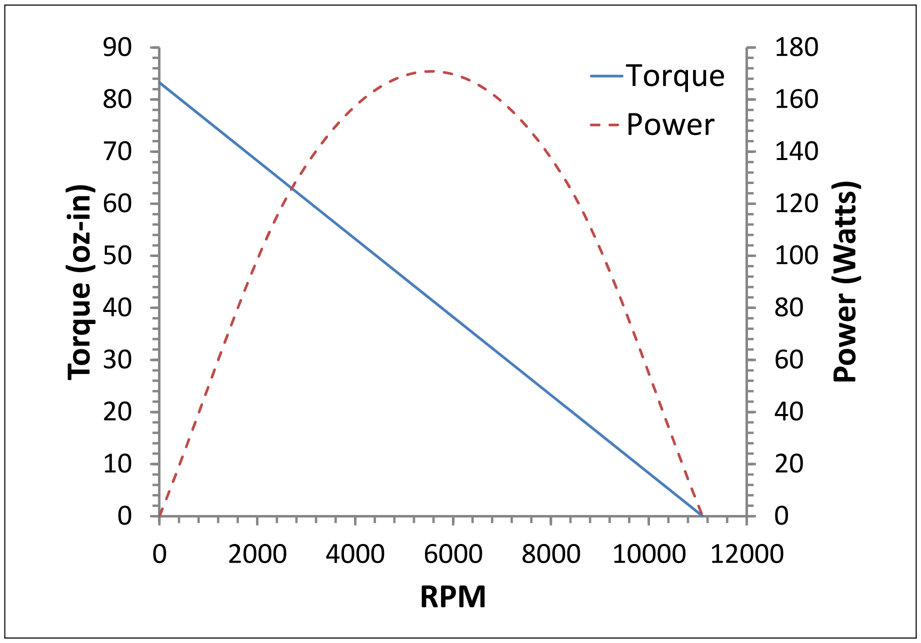

Figure 4.3: Torque Speed Power Curve Example for a DC Motor

sumption is at half the maximum RPM and conversely that little power is used near the maximum and minimum RPM ranges. Figure 4.4 compares the electric motor from Figure 4.3 with a Cox 0.09 2-stroke model aircraft engine running 30% nitro [23]. Even these motor are not operating at the same speeds they produce approximately the same power. This plot shows the main difference between a internal combustion engine and an electric motor, an electric motor torque starts at a peak and goes to zero with an increase in speed while an internal combustion engine starts at zero reaches a peak then decreases back to zero. An electric motor What Figure 4.3 does not show is the actual limits of a motor. The wasted energy or losses in the motor are due to heat and friction. In most electric motors friction is minimal and can usually be ignored, but the heat produced by the current flowing through the motor cannot. The heat that is produced can melt the coils on the armature if too much current is allowed to flow through the motor without sufficient cooling. This leads to motors having thermal limits which are identified with continuous and maximum/burst current ratings or continuous and maximum/burst power ratings. A continuous rating is

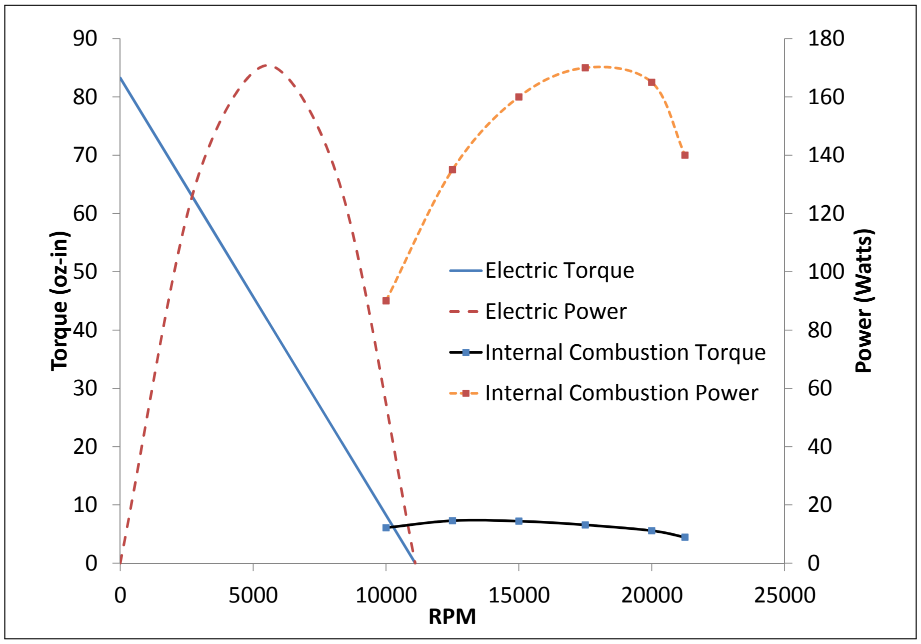

Figure 4.4: Electric Motor and Internal Combustion Engine Comparison

the maximum allowable current or power the motor can experience to run indefinitely. The maximum/burst ratings are the absolute limits on the motor for a certain time span. This allowable time for maximum conditions is set by the manufacturer, and this time ranges from 15-60 seconds.

4.2 Brushless DC Electric Motors

Brushless DC electric motor have become more popular over the last several years for small remote controlled aircraft due to there more efficient nature and little to no need for maintenance. Its increase in efficiency over a brushed DC motor comes from the lack of mechanical brushes in the motor. This reduces the friction inside the motor, and removes parts that need to be serviced. For the simplicity friction will be ignored in this motor model. The only advantage of a brushless motor to a brushed motor is the brushless motor has fewer parts to wear out.

A brushless DC motor may appear to be an AC motor, but it is not. Besides the type of current being provided to the motor, what separates an AC from DC motor is the brushless

DC motor uses sensors to detect the rotor position to control the pulses to the motor [19]. Simply put these pulses create magnetic fields which cause the motor to rotate. Usually the type of sensor used is a Hall Effect sensor, but most hobby grade brushless DC motors which are used with most small electric UAVs have no physical sensors in the motor. The speed controllers used to power the motor read the back e.m.f. from the motor and determine the motors position from that.

A brushless DC motor can be modeled in the same manner as brushed DC motors approximately yielding the same type of torque speed power curves. The rotational speed in radians per second of the brushless DC motor under load can be found in Equation 4.5 [22].

$$\omega_r = \frac{V_{motor}}{p\frac{\lambda_m}{2}} - \frac{R_I}{m\left(p\frac{\lambda_m}{2}\right)^2}T_{em} \tag{4.5}$$

Where λ_m is the flux linkage of the stator winding, p is the number of poles in the motor, m is the number of phases, and T_{em} is the electromagnetic torque defined by Equation 4.6. A poles is the set of windings in a motor and is an even number. The number of phases is the number of conductors connected to the motor that supply voltage to the motor. All of the motors discussed here will be three phases motor. This leads to the voltage waveforms on each of the phases of the motor will be offset by 60°.

$$T_{em} = \frac{mp}{2}\lambda_m I \tag{4.6}$$

The efficiency of the brushless DC motor is found by Equations 4.7-4.9. I_0 is the idle current of the motor, this is the current the motor will draw with no load.

$$\eta = \frac{P_{out}}{P_{in}} \tag{4.7}$$

$$P_{out} = \left(V_{motor} - IR_I\right)\left(I - I_0\right) \tag{4.8}$$

$$P_{in} = V_{motor}I \tag{4.9}$$

Chapter 5
Optimization Methods

Optimization methods are used to minimize or maximize a problem with multiple unknowns. These methods greatly decrease the number of function calls over a brute force method in which every possible combination of unknowns is evaluated. They can also be used to find local or global minimums. An objective function is evaluated by the optimization method to determine if the problem is optimized. The optimizers that will be used for this work will be treated as black boxes, and no math will be presented in this section. This chapter will provide a brief overview of a particle swarm method, a pattern search method, and a simplex method.

5.1 Particle Swarm

Particle swarm optimization was first suggested by Kennedy and Eberhart [24] as a stochastic methodology based on crowding behavior and collective intelligence. Similar to genetic algorithms in practice, the particle swarm technique relies on communication and interaction among its members of a population to collectively move throughout a design space. Like any stochastic based optimization routine, swarming has the ability to escape the local optima of a problem in search of a better solution. The prime attractor to particle swarm optimization is its simplicity in implementation compared to other stochastic techniques. The particles move through the design space to find the optimum location with a varying velocity. The particles move in the directions in which the best particles are performing and are swayed by their own best position and the absolute best location of all the particles.

Particle swarm methods can move from local optimum if other particles find an improved solution unlike gradient methods which will get stuck on local optimum points. The objective function used in the particle swarm does not have to differentiable or smooth. This allows for more problems to be solved. The biggest problem with particle swarms is a particle uses its previous position and velocity to solve where it should go. This is sometimes a problem if a particle is located at the optimum point. This leads to particle swarms are good at finding area of the design space that should yield the optimum global point [24].

5.2 Pattern Search

The pattern search method was originally developed by Hooke and Jeeves and is a direct search technique [25]. This method works by monitoring the changes each of the design parameters have on the objective function. The pattern search starts with an initial design case, and then performs an investigative move on one of the design variables holding the others constant. It then evaluates the objective function and if it is better than a solution it stores it. This process is repeated for the rest of the design variables. A pattern move is then performed which is a changing all of the variables and evaluating the objective function. If this new evaluated objective function is better the design variables are stored, and if it is worse the process is repeated with a decreased initial move by the design variables [25].

The pattern search method is a good technique for finding local optimum locations. Its solution is highly dependent on its initial location. If placed in the correct location in the design space it can provide rapid results, but more design variables used and worst starting location can yield a long computation time [26].

5.3 Hybrid Pattern Search/Particle Swarm Method

The hybrid pattern search and particle swarm method that will be presented was developed by Jenkins and Hartfield [26]. It combines the local optimization features of the pattern search and the global optimization features of the particle swarm methods. This method first starts by initializing a population with acceptable values of the design parameters that fulfill the objective function. The pattern search method is then run through user a defined number of generations. The particle swarm method is then run went the results of the pattern search. This process is repeated for a defined number of generations to provide adequate convergence and to ensure a global optimum is found [26].

5.4 Simplex Method

The simplex method that will be presented is a direct search method and does not use gradients. This method can be found in MATLAB 2010a as the $fminsearch()$ function [27]. It uses a method developed by Lagarias et al. in Reference [28]. This function uses only function evaluations like the pattern search and particle swarm methods and does not require derivatives to be solved. This method uses a simplex which is a geometric shape of same number of unknowns in the problem with the number of unknowns plus one points describing the shape. For example if the problem has three unknowns it would be a three dimensional problem with four points describing the simplex which would resemble a pyramid. When the method is executed a new point near the simplex is evaluated and compared to the values of the points of the simplex. If the new point is "better" than one of the simplex points, the "bad" point is replaced. This process is repeated until the diameter of the simplex is within a user defined tolerance [27] [29].

The simplex method can be used with discontinuous objective functions. It is not guaranteed to provide a global optimum solution for discontinuous function and will usually provide a local optimum minimum [27].

Chapter 6

Implementation

The optimization of a propeller is split into two major sections. First an airfoil is optimized for a maximum lift to drag ratio. Second a propeller is optimized using the optimized airfoil. The process was performed in this matter to decrease the run time of the finished optimization. MATLAB 2010a was used to execute the optimizations.

6.1 Airfoil Optimization

A set of airfoils for the optimized propellers is found first. This set of airfoils consists of optimized airfoils for a range of angle of attacks. The process is shown in a flow chart in Figures 6.1 and 6.2.

A series of functions, which are found in Appendix C, were constructed using the process discussed in Section 2.2.2 to describe an airfoil. A 6^{th} order Bernstein polynomial was used for the upper and lower surfaces to provide a good representation of an airfoil surface [9]. The first function will be called *AirfoilMaker().* The fourteen unknowns or coefficients as well as a desired angle of attack for the airfoil is sent to this function. It starts by devising a name for the airfoil using the coefficients which will be used as the file name for the airfoil data. The function then calls another function, *ParametricAirfoil()* to produce x and y coordinates of the airfoil. It also checks so see if the upper and lower surfaces cross, and if they do it returns that the airfoil is a bad airfoil. *AirfoilMaker()* then calls *ClCdFinder()* and passes it the x and y values for the airfoil and the airfoil name.

ClCdFinder() checks a folder containing already made airfoils and searches for it. If it is not found a file corresponding to the airfoil an instruction file for XFOIL is written. The instruction file looks similar to the example in Appendix A. The function then runs

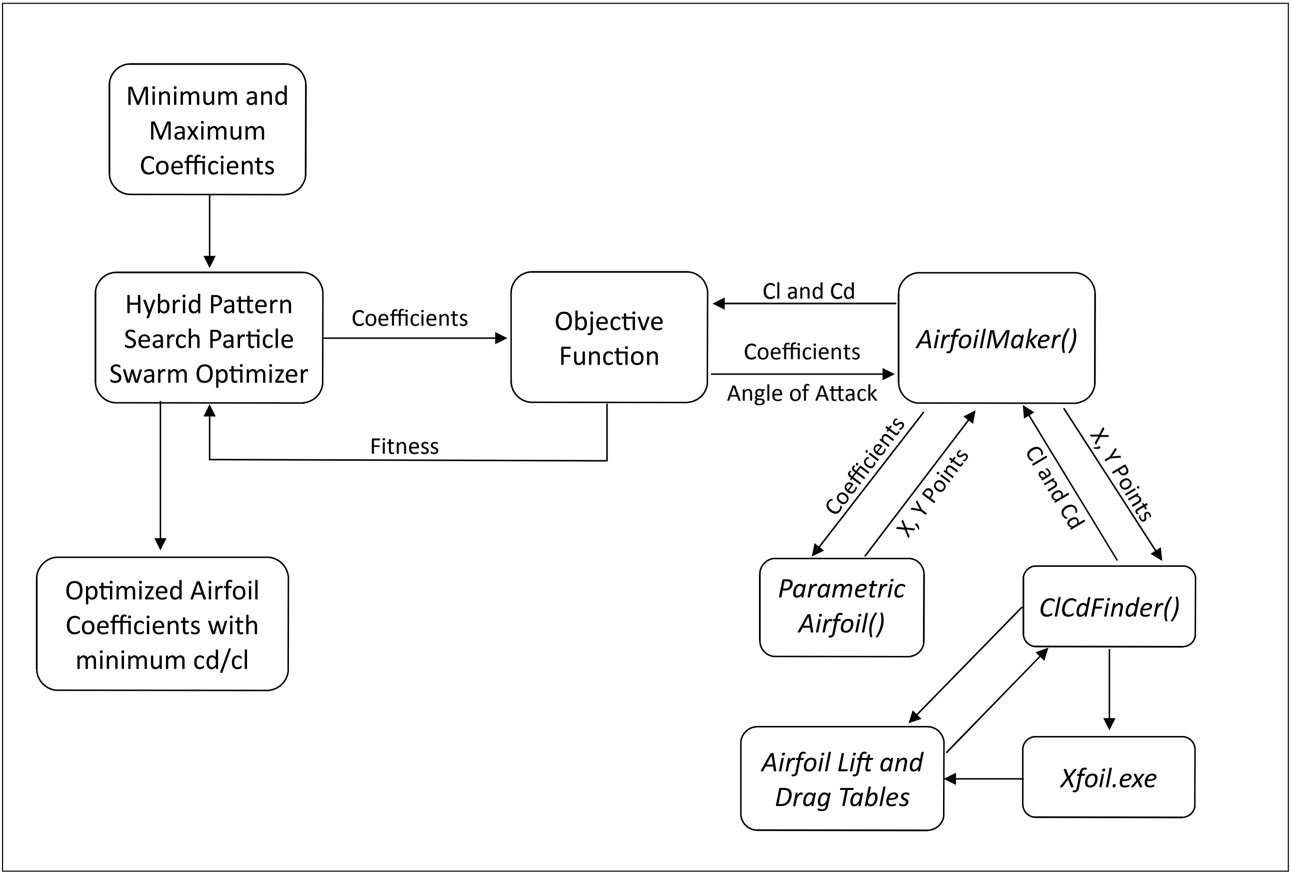

Figure 6.1: Flowchart for Airfoil Optimization Process

a batch file that runs XFOIL with the points and instruction files. A timeout was added to the batch program in the event that XFOIL could not finish running due to improper convergence. The function then checks to confirm that XFOIL produced an acceptable table of lift and drag coefficients. The table is not acceptable if any of the values for the lift or drag coefficients are NaN (not a number) or if XFOIL failed to produce enough data points to describe the lift and drag curve slopes. If it does fail the process a new instruction file is written increasing the number of panels on the airfoil. This is done until the airfoil successfully passes or reaches a maximum panel size in which case the airfoil is considered to be a bad airfoil.

After either creating the new airfoil table or locating a previously made file the function reads in the data from the file. It interpolates for the lift and drag coefficients at the desired angle of attack and returns the values to *AirfoilMaker()*.

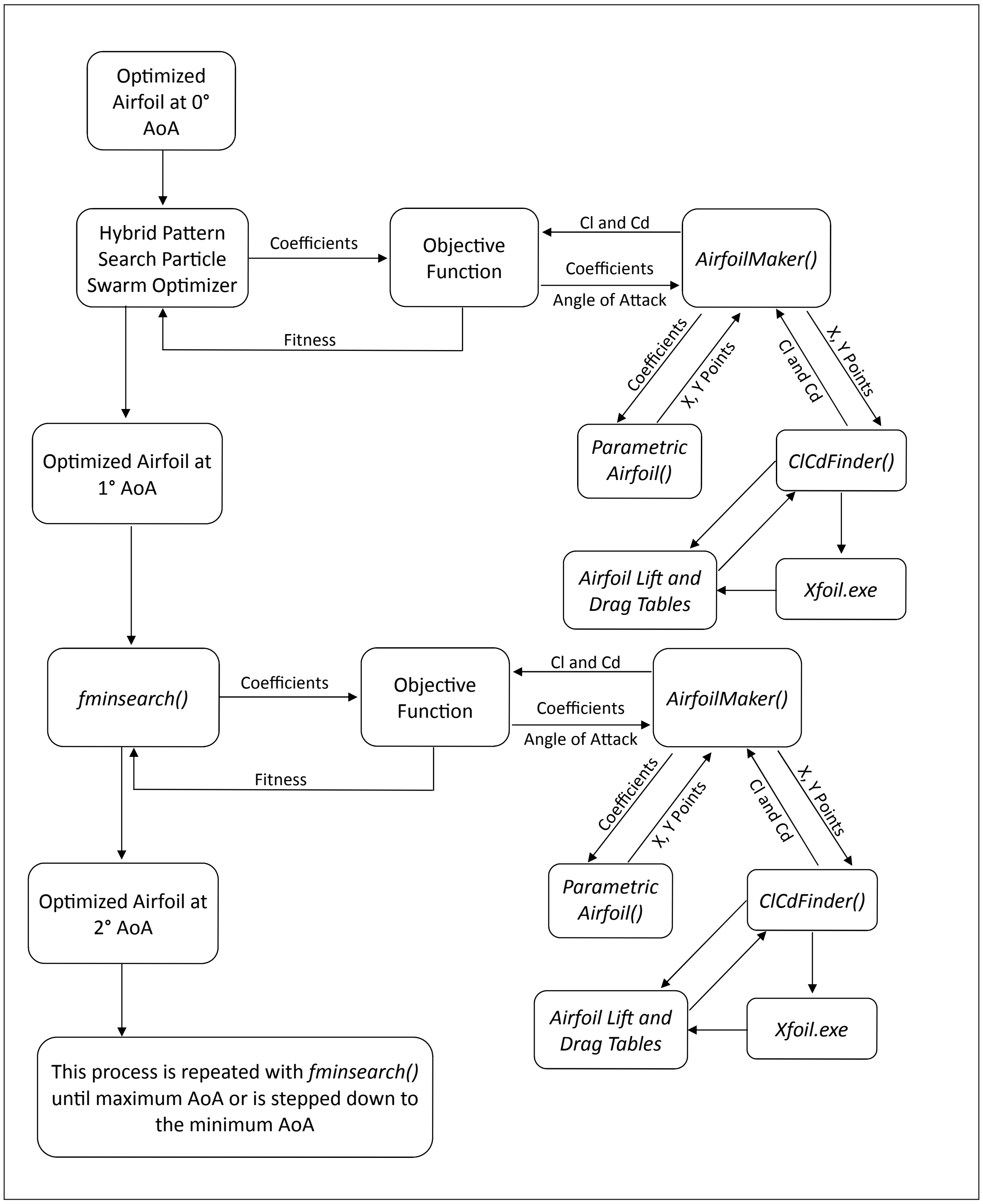

Figure 6.2: Flowchart for Propeller Optimization Process using *fminsearch()*

An objective function, *AirfoilCostFunction*, was developed for the optimizers to call. It called the *AirfoilMaker()* function with the fourteen coefficients and the angle of attack to be optimized. The objective function returns the fitness of airfoil to the optimizer. Since all of the optimization method discussed are minimization methods the optimum airfoil would be found by minimizing the drag to lift ratio. The fitness equation is shown if Equation 6.1 where z is the fitness.

$$z = \frac{cd}{cl} \tag{6.1}$$

If coefficients used in *AirfoilMaker()* return lift or drag coefficients less than zero the fitness for that set of coefficients was set to 10^8. This reiterates to the optimizer that those coefficients produce corrupt results.

The optimized airfoils were found be first using the hybrid pattern search particle swarm method discuss in Chapter 5. An airfoil was optimized at a zero degree angle of attack using this method. Then airfoils at a range of angles of attack were optimized using MATLABs simplex method *fminsearch()*. It was executed in this matter due to increased run time of the hybrid optimizer verse *fminsearch()*. The time results from each of the optimization runs will be discussed later in the Chapter 7. The zero degree angle of attack airfoil was used as the initial starting location. As the optimization sweep advanced the previous airfoil was used as the initial starting location for next airfoil. A list of optimized airfoils was saved to be used by the propeller optimizer.

6.2 Propeller Optimization

The propeller optimizer code, like the airfoil optimizer, is made up of a series of functions coded in MATLAB 2010a. Flowcharts for the propeller optimization methods, cruise and climb are shown in Figures 6.3 and 6.4. All of the functions used for this optimizer can be found in Appendix D. A brushless DC electric motor model is coupled into the optimizer for the most efficient system.

The brushless DC motor model function was named *BrushlessDCMotor()*. The function has two different sets of input and output cases depending on what is being optimized. The K_v value, internal resistance, number of poles, number of phases, the idle current, and which case is being used are inputs that are used for both cases. The first condition has inputs of

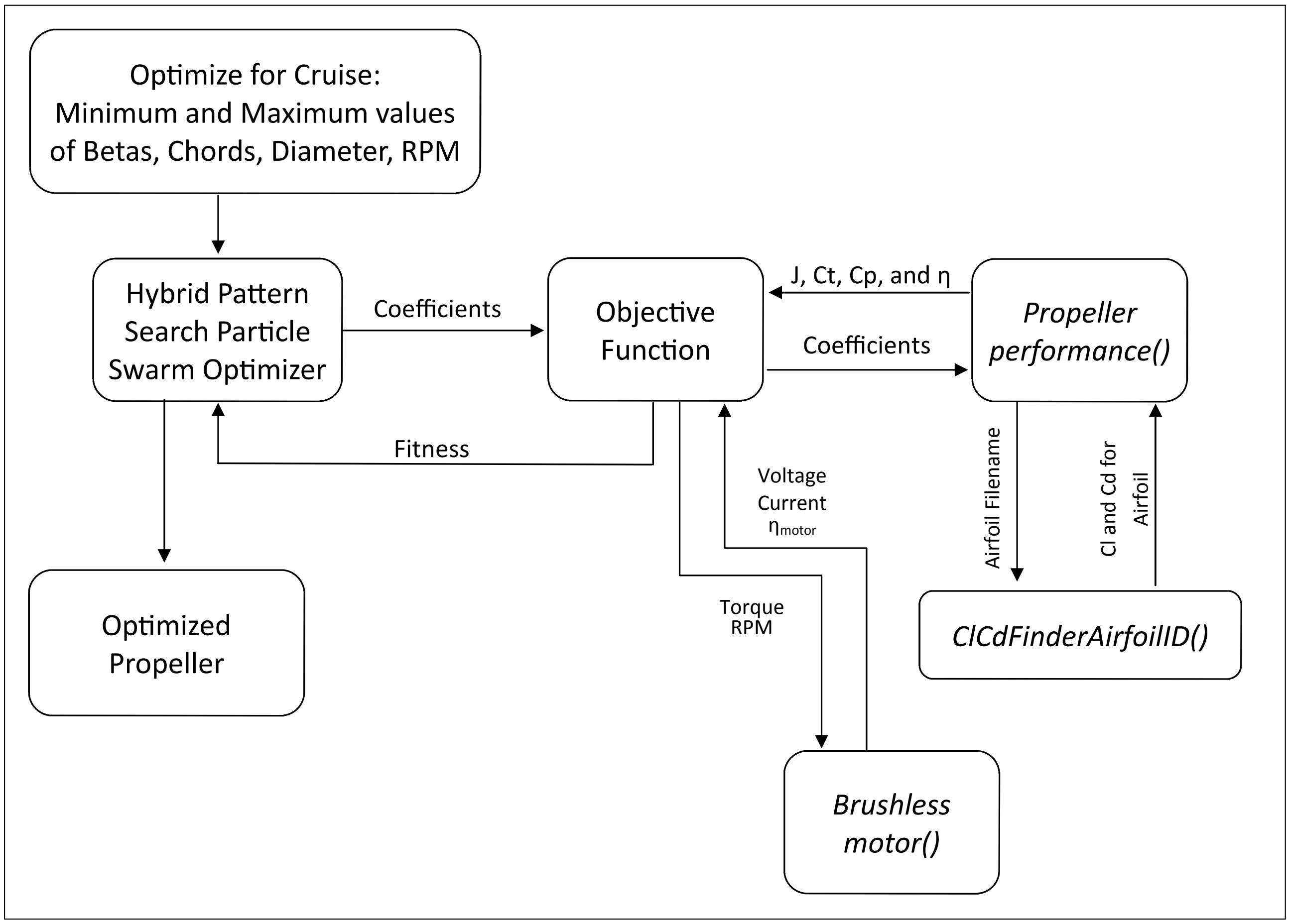

Figure 6.3: Flowchart for Propeller Optimization Process using Hybrid Optimizer for Cruise Condition

RPM and torque and outputs of voltage, current, and motor efficiency. The current is found by dividing the input torque with the torque constant, K_t. The voltage is found using a brute force iterative method evaluating stepping the voltage in Equation 4.5 until the rotational speed matches the input rotational speed. The efficiency is found the same way as the first case using Equations 4.7 - 4.9 and dividing the output power with the input power.

The second condition has inputs of voltage and current and outputs of torque, RPM, and motor efficiency. The rotational speed of the motor is found using Equation 4.5. The torque is found by using the K_v to find the K_t and then multiplied by the input current. The efficiency is found using the process expressed in Equations 4.7 - 4.9 by dividing the output power with the input power.

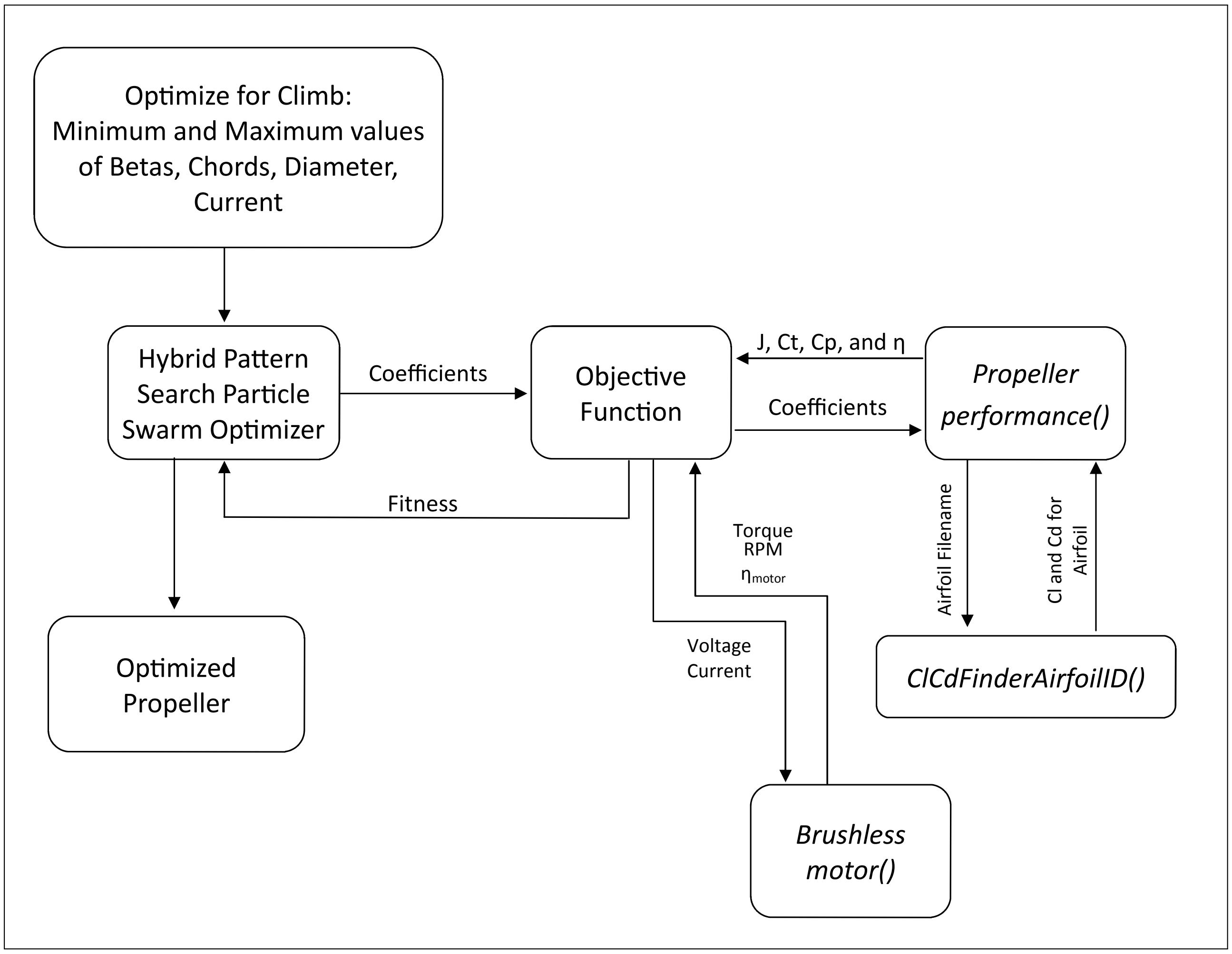

Figure 6.4: Flowchart for Propeller Optimization Process using Hybrid Optimizer for Climb Condition

The function that calculates the performance parameters of a propeller was named *PropellerPerformance()*. The functions inputs are the blade angles, β, the chord width of the elements, the radial position of the elements, the rotational speed, the number of blades, and the free stream velocity. It uses the method "Inflow with Axial and Rotational Interference Factors" discussed in more detail in Section 3.2.3. The function uses the inputs to iterate for the inflow factors, radial factors and the flow angles for each element. The iteration stops when a maximum iteration limit is reached or a tolerance is met. The thrust and power coefficients are found for each element and integrated using the trapezoid method to find the overall coefficients. The efficiency is then found using Equation 3.4. The function returns the advance ratio of the propeller, the thrust coefficient, the power coefficient, and the efficiency of the propeller.

The lift and drag coefficients for each element is found from the function *ClCdFinderAirfoilID()*. An “Airfoil ID” and the angle of attack is sent to this function. The angle of attack desired from *PropellerPerformance()* is rounded to the nearest integer value and that is used to identify the which airfoil to use. If the angle of attack is less than or greater than the minimum or maximum optimized airfoil the closest airfoil to that value is used. If the angle of attack is greater than 20° the largest optimized airfoil is used, but the lift and drag coefficients are estimated using flat plate theory [30]. Flat plate theory says $C_l = 2sin\left(\alpha\right)cos\left(\alpha\right)$ and $C_d = 2sin\left(\alpha\right)^2$.

A third condition, optimizing for climb-cruise, is very similar to the first condition and is a multi-point optimization. The flowchart for this method is shown in Figure 6.5. This method has inputs of inputs of RPM and torque of a cruise condition and runs the *PropellerPerformance()* function and the *BrushlessMotor()* function. If a “good” propeller is found the process is repeated with the climb condition.

The optimizer that was chosen was the hybrid pattern search particle swarm method. This method was chosen over the other methods due to its ability to potentially find a global optimum better than the other methods. The objective function includes conditional statements to confirm the optimizer has found a viable solution. These include statements to confirm the current draw from the motor is not too high, the propeller does not produce negative thrust, efficiencies are not greater than 100 percent, and several others. The actual objective function used for this optimizer will be presented in the Chapter 7.

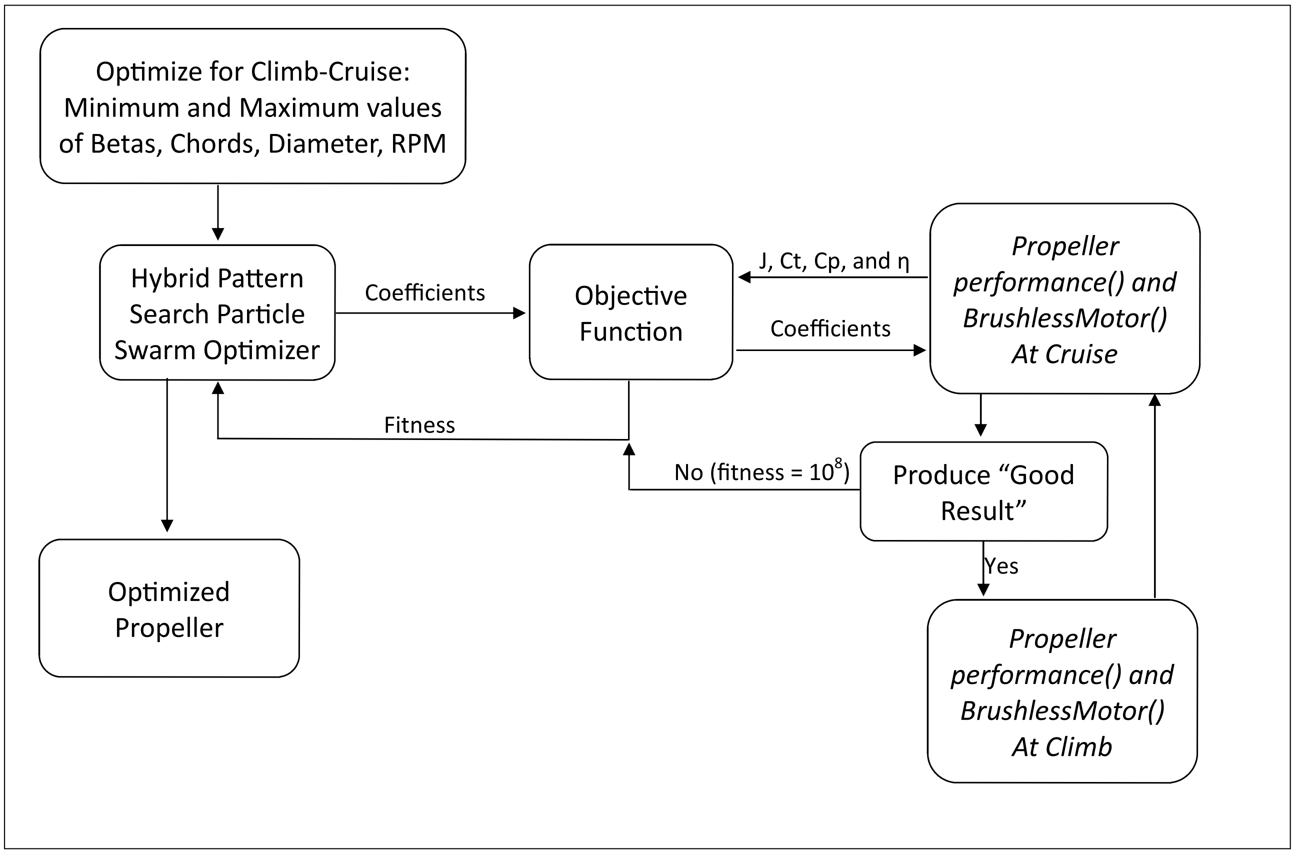
Optimize for Climb-Cruise: Minimum and Maximum values of Betas, Chords, Diameter, RPM
Hybrid Pattern Search Particle Swarm Optimizer
Coefficients
Objective Function
J, Ct, Cp, and η
Coefficients
Propeller performance() and BrushlessMotor() At Cruise
Fitness
No (fitness = 10^8)
Produce "Good Result"
Yes
Optimized Propeller
Propeller performance() and BrushlessMotor() At Climb

Figure 6.5: Flowchart for Propeller Optimization Process using Hybrid Optimizer for Climb-Cruise Condition

Chapter 7

Results

The propeller optimization code was performed on three particular flight conditions. For each of the conditions a propeller is designed using the same brushless DC motor. The motor is modeled after an Eflite Park 450 Outrunner. The parameters for the motor are listed in Table 7.1 [31]. The propeller is designed to be placed on a small aircraft with the

Table 7.1: DC Brushless Motor Parameters

Parameter:	Value:
Internal Resistance, R_I	0.2 Ohms
$K_v \left(\frac{RPM}{Volt}\right)$	890
Idle Current, I_0	0.70 Amps
Continuous Current	14 Amps
Maximum Burst Current	18 Amps for 15s
Voltage Range	7.2-12 Volts
Weight	2.5 oz

requirements listed in Table 7.2. Based on the requirements in Table 7.2 the three optimized

Table 7.2: Flight Condition Design Parameters

Design Parameter:	Value:
Stall Speed	15 MPH
Cruise Speed	50 MPH
Drag at Cruise	7.5 oz

propeller optimizations will be for a cruise, a climb, and a multi-point climb cruise case. Two different objective functions where used for each of the optimizations to see the effect an objective function using coefficients would perform verse dimensional values. The number of elements used was 10. This was decided on to give a good representation of the propeller blade and to keep the number of unknowns a small as possible. The element distribution was an even distribution starting at 12% of the radius. The number of blades was fixed to

2 blades for all of the propellers. In the optimizer the blade angles, β, could only decrease from as they traveled outward to the tip, and the chord was allowed to expand until the three quarter radius location then could only decrease to zero at the tip. The diameter was allowed to vary from 8 to 12 inches.

7.1 Airfoil Optimization

The set of airfoils used for the propeller were found first. The first airfoil found using the hybrid pattern search particle swarm optimizer was implemented at a zero angle of attack. The drag to lift ratio was minimized. It took approximately 22 hours to complete on an Intel i7 930 with 6 gigabytes of ram running 64-bit Windows 7. Xfoil was called and airfoils were made 25,000 times before the optimizer was finished running using a population size of 15 for 30 generations with 2 pattern searches per generation. This computer was used for the remainder of the computations. Figure 7.1 shows the optimized for a zero angle of attack airfoil. At an angle of attack of 0° it has a $C_d = 0.01305$, a $C_l = 0.8404$. This gives a drag to lift ratio of 0.0155.

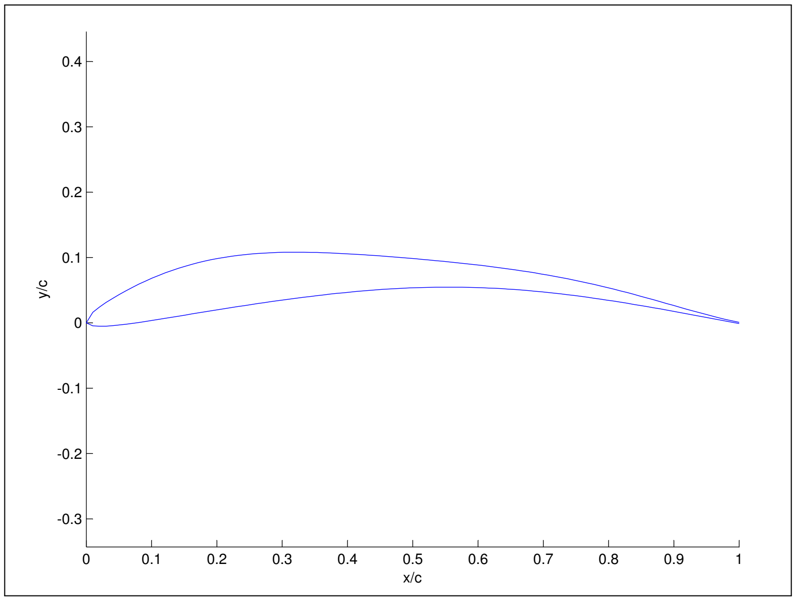

Figure 7.1: Optimized Airfoil at 0° Angle of Attack

The rest of the airfoils, $-5°$ to $15,°$ were found using *fminsearch()* which is a simplex function in MATLAB. Each of these airfoils took 5-10 minutes to complete. Tables E.1, E.2, and E.3 in Appendix E contain the coefficients for all of the airfoils, the drag to lift ratios, and the lift coefficients at the design point. Figure 7.2 shows a comparison of the different optimized airfoils. The axis in the plot are not equal to show the differences between the airfoils. The $-5°$ airfoil is not show due to the very slight change between the $0°$ airfoil. It is shown that as the angle of attack increases the upper surface changes, but the lower surface slightly changes.

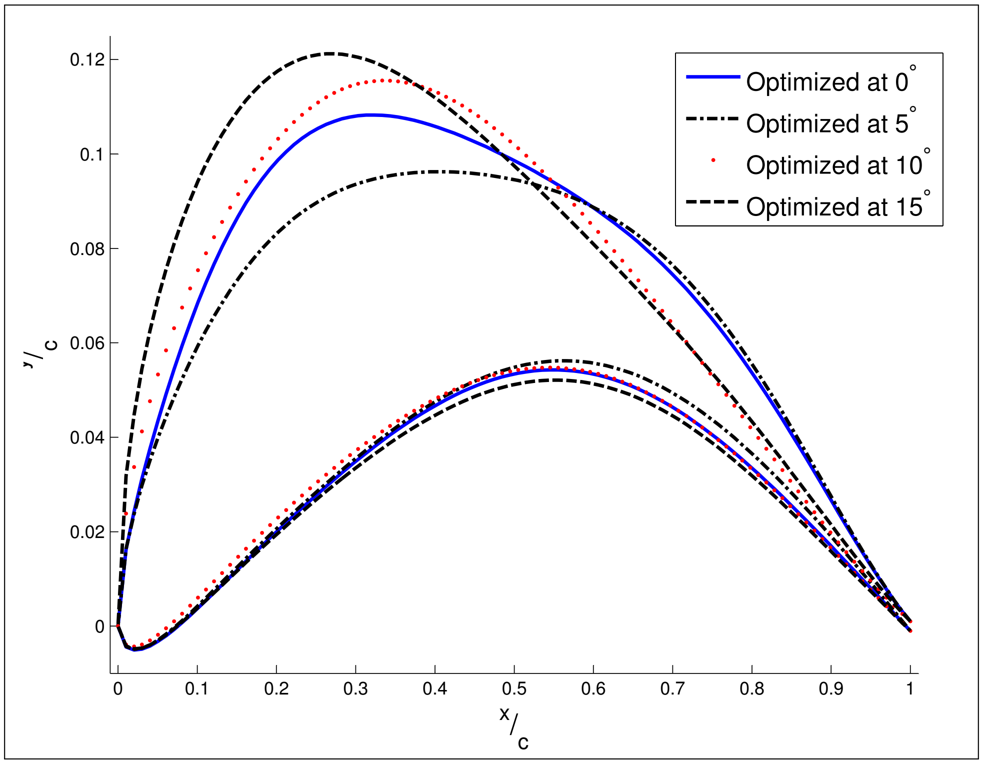

Figure 7.2: Optimized Airfoil Comparison

7.2 Validation of Propeller Code

To confirm the propeller code was providing accurate data three commercial propeller's were executed and discussed in this section. The three propellers chosen will be the propellers used as a baseline comparison for each of the conditions, cruise, climb, and climb cruise. The propeller for cruise condition comparison was an APC 11x10E, the climb condition baseline propeller was an APC 8x6E, and the propeller for the climb cruise condition was an APC 10X7E. The reasonings for these baseline propellers will be discussed later in this chapter. The geometry was provided from the manufacturer [32] and wind tunnel data for each of these propellers was found in the UIUC Propeller Database [33].

7.2.1 Validation Results for Baseline Cruise Propeller

The cruise propeller chosen as the baseline propeller was an APC 11x10E. The following three figures are a comparison with thrust, power, and efficiency verse free stream velocity. The propeller code and the wind tunnel data was run at a rotational speed of 8000 rpm. The thrust comparison is shown in Figure 7.3. The thrust calculated using the propeller code is shown to slightly over predict, but does follow the same trend as the data. The increase in thrust over the wind tunnel data can be attributed to incorrect lift coefficients from XFOIL or other losses that were not accounted for in the propeller code.

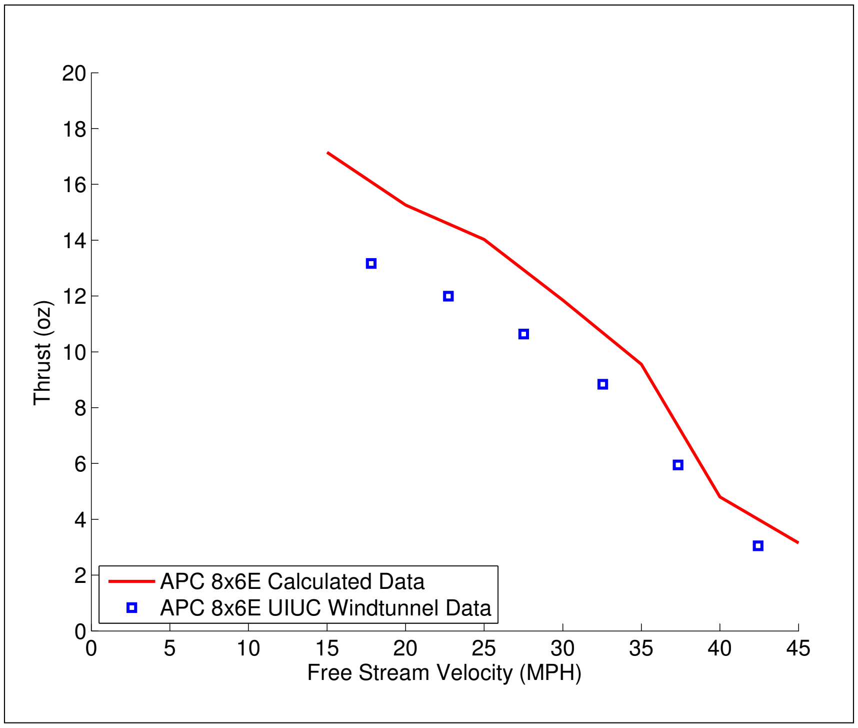

Figure 7.3: Thrust Validation for APC 11x10E Cruise Propeller at 8000 rpm

The propeller power comparison is shown in Figure 7.4. The propeller code under predicts the power required, but follows the same trend as the propeller code. This under prediction shows that XFOIL's drag estimation is small. The efficiency plot is shown in Figure 7.5. The calculated efficiency is much higher than the wind tunnel data. This is due to the under estimation in power required and the over estimation of thrust produced.

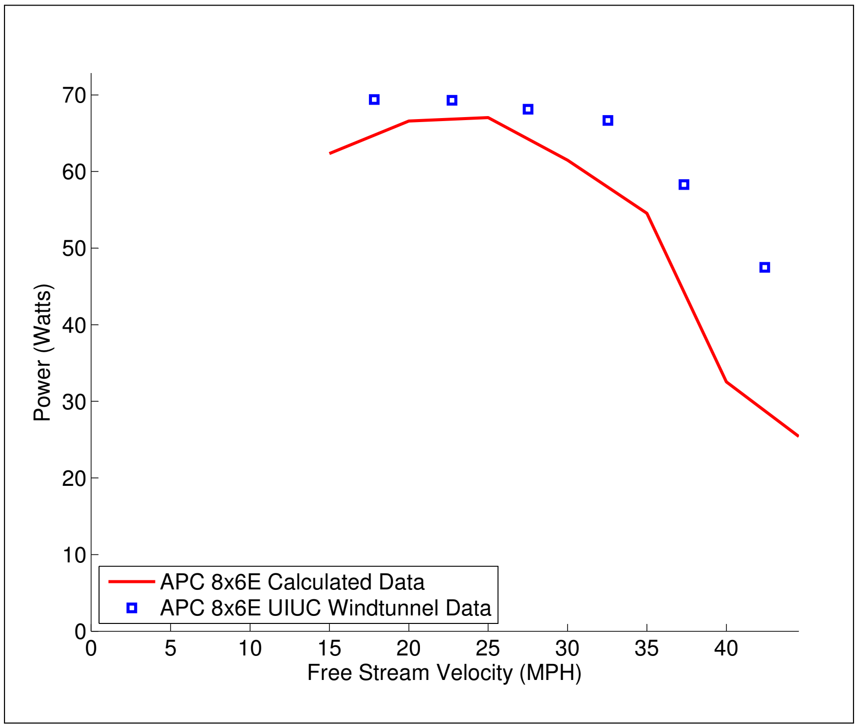

Figure 7.4: Power Validation for APC 11x10E Cruise Propeller at 8000 rpm

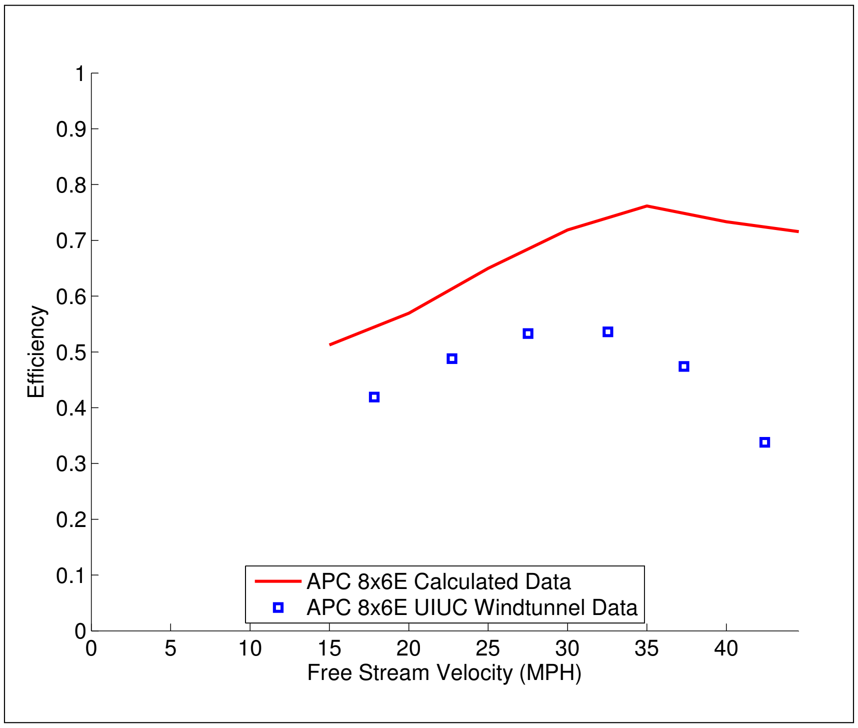

Figure 7.5: Efficiency Validation for APC 11x10E Cruise Propeller at 8000 rpm

7.2.2 Validation Results for Baseline Climb Propeller

The climb propeller chosen as the baseline propeller was an APC 8x6E. The following three figures are a comparison with thrust, power, and efficiency verse free stream velocity. The propeller code and the wind tunnel data was run at a rotational speed of 8000 rpm. The thrust comparison is shown in Figure 7.6. The thrust calculated using the propeller code is shown to slightly over predict, but does follow the same trend as the data. The increase in thrust over the wind tunnel data can be attributed to incorrect lift coefficients from XFOIL or other losses that were not accounted for in the propeller code.

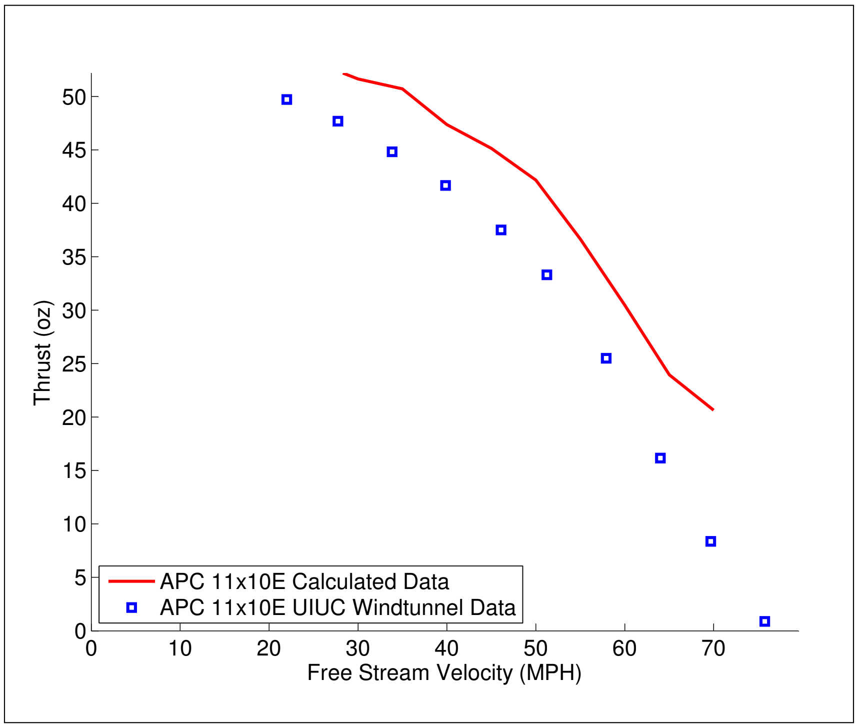

Figure 7.6: Thrust Validation for APC 8x6E Climb Propeller at 8000 rpm

The propeller power comparison is shown in Figure 7.7. The propeller code calculates power required approximately the same for free stream velocities above 30 miles per hour. The propeller code does under predict power required below a free stream velocity of 30 miles per hour. This under prediction shows that XFOIL's drag estimation could be small. The efficiency plot is shown in Figure 7.5. The calculated efficiency is higher than the wind tunnel data. This is due to the over estimation of thrust produced.

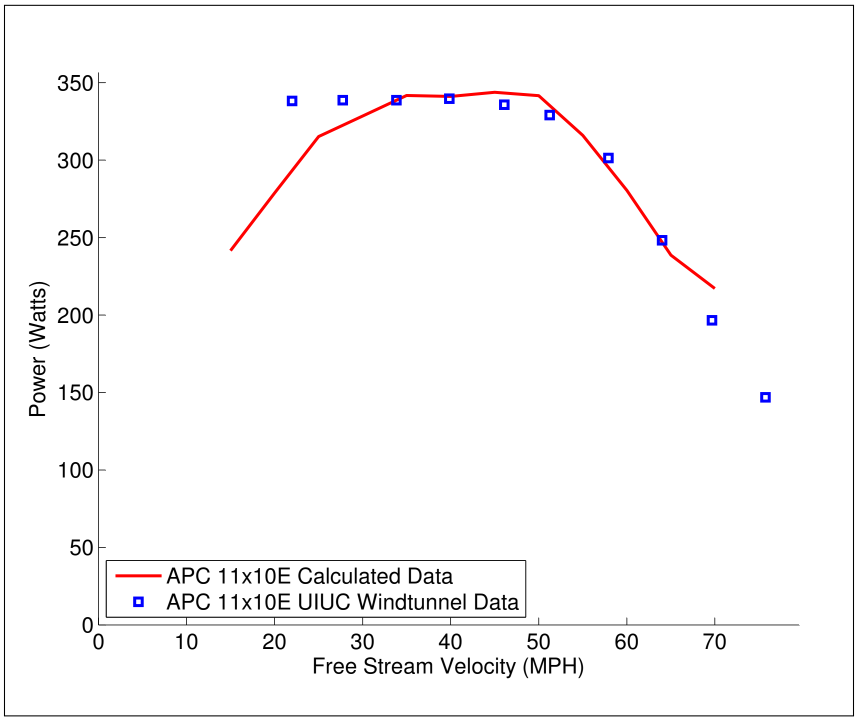

Figure 7.7: Power Validation for APC 8x6E Climb Propeller at 8000 rpm

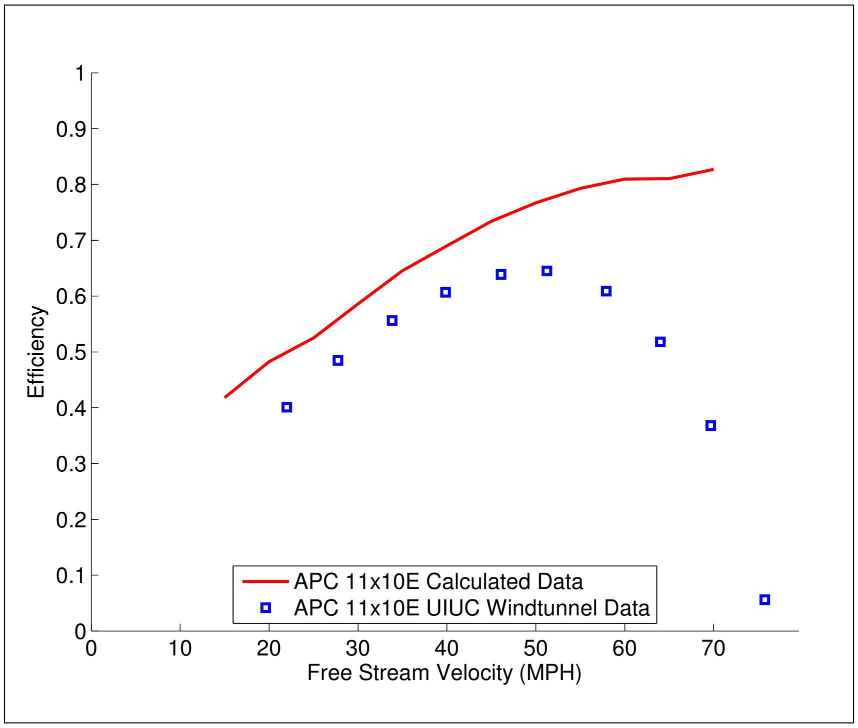

Figure 7.8: Efficiency Validation for APC 8x6E Climb Propeller at 8000 rpm

7.2.3 Validation Results for Baseline Climb Cruise Propeller

The climb cruise propeller chosen as the baseline propeller was an APC 10x7E. The following three figures are a comparison with thrust, power, and efficiency verse free stream velocity. The propeller code and the wind tunnel data was run at a rotational speed of 8000 rpm. The thrust comparison is shown in Figure 7.3. The thrust calculated using the propeller code is shown to slightly over predict, but does follow the same trend as the data. The increase in thrust over the wind tunnel data can be attributed to incorrect lift coefficients from XFOIL or other losses that were not accounted for in the propeller code.

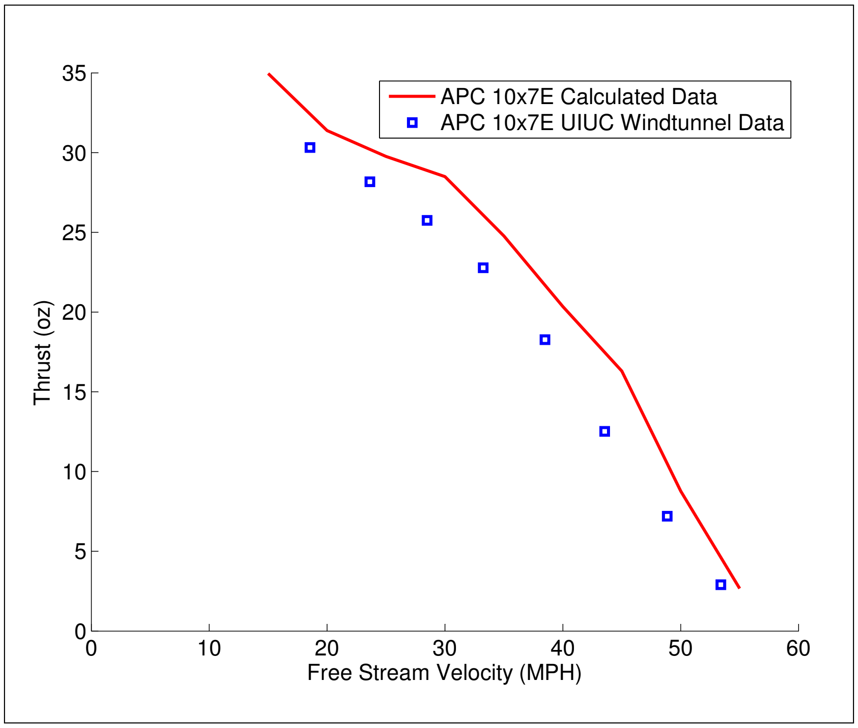

Figure 7.9: Thrust Validation for APC 10x7E Climb Cruise Propeller at 8000 rpm

The propeller power comparison is shown in Figure 7.4. Similar to the APC 8X6E propeller, the propeller code calculates power required approximately the same for free stream velocities above 30 miles per hour. The propeller code does under predict power required below a free stream velocity of 30 miles per hour. This under prediction shows that XFOIL's drag estimation is small. The efficiency plot is shown in Figure 7.5. The calculated efficiency is much higher than the wind tunnel data. This is due to the under estimation in power required and the over estimation of thrust produced.

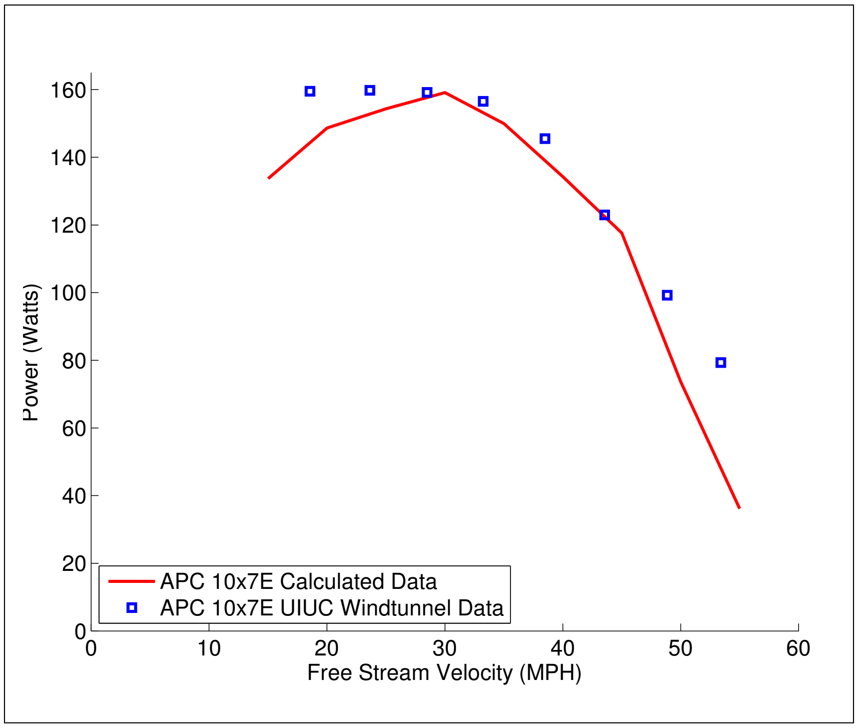

Figure 7.10: Power Validation for APC 10x7E Climb Cruise Propeller at 8000 rpm

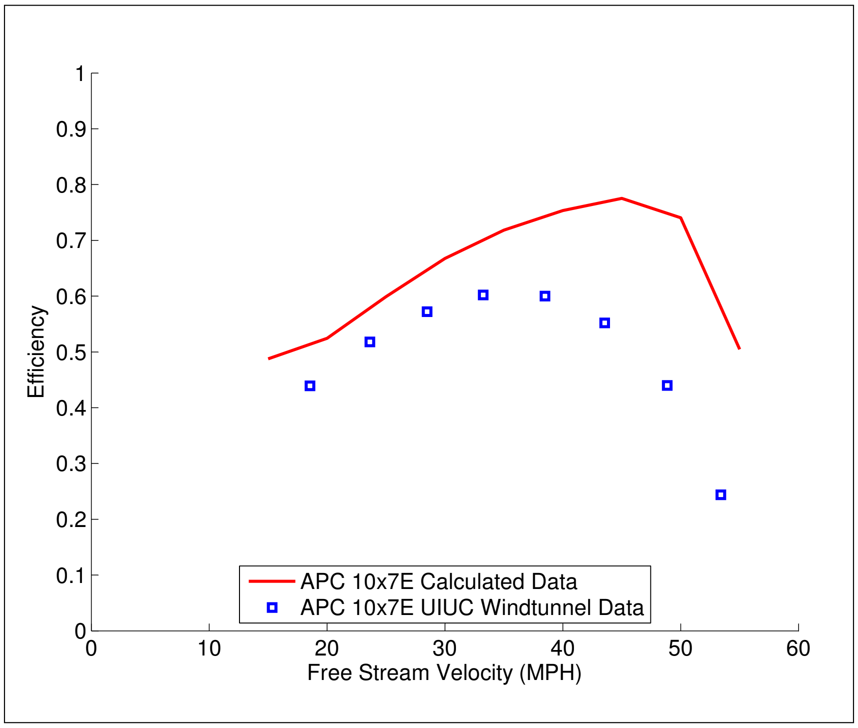

Figure 7.11: Efficiency Validation for APC 10x7E Climb Cruise Propeller at 8000 rpm

7.3 Propeller Optimized for Cruise

The first optimization that was tested was the cruise condition. From Table 7.2 the propeller needs to produce 7.5 ounces of thrust at a free stream velocity of 50 miles per hour. If the thrust is fixed for the propeller to be as efficient as possible power for the propeller must be minimized. This leads to the three different objective functions used and are shown in Table 7.3 where z is the fitness. Case 2a and 2b were chosen to show the effect on which power is minimized. The range of values used by the optimizer are shown

Table 7.3: Objective Functions used for Cruise Condition

Case Number	Objective Function:
1	$z = C_p$
2a	z=Propeller Power (Watts)
2b	z=System Power (Watts)

in Table 7.4. The values of β could only decrease to the tip, and the chord was allowed to expand to approximately the three quarters radius location then decrease to zero at the tip. The optimizer was run for 100 generations with 5 pattern searches per generation and

Table 7.4: Optimizer Limits for Cruise Condition

Parameter	Minimum	Maximum
β_{start} (°)	35	65
$chord_{start}(in)$	0.5	2.0
$Diameter(in)$	8	12
RPM	3000	5500
$Currrent(Amps)$	0	14
$Voltage(Volts)$	0	11.1

a population size of 15 members. Each case had an execution time of 5 hours. Figures 7.12, 7.13, and 7.14 show the evaluated objective function for each member of each generation. The members that were "bad" propellers are not plotted. As discussed in the previous chapter these "bad" propellers do not meet the requirements and the objective function assigns them a value of 10^8. Each member is represented by a "x." A dashed line connects the maximum values of each generation as well as the minimum values. A solid line is used to show the best fitness found as of that generation.

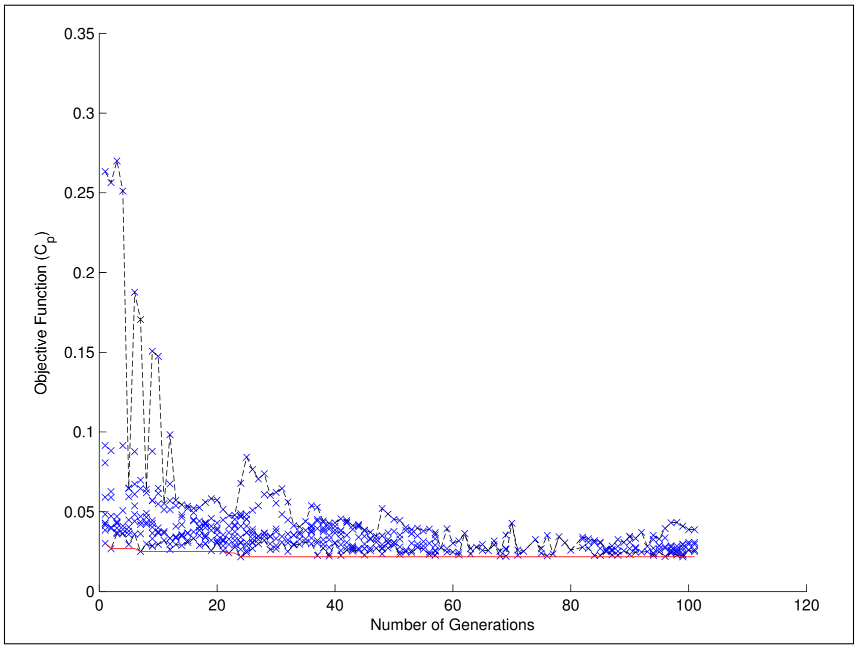

Figure 7.12: Fitness verse Number of Generations Case 1 (Objective Function = C_p)

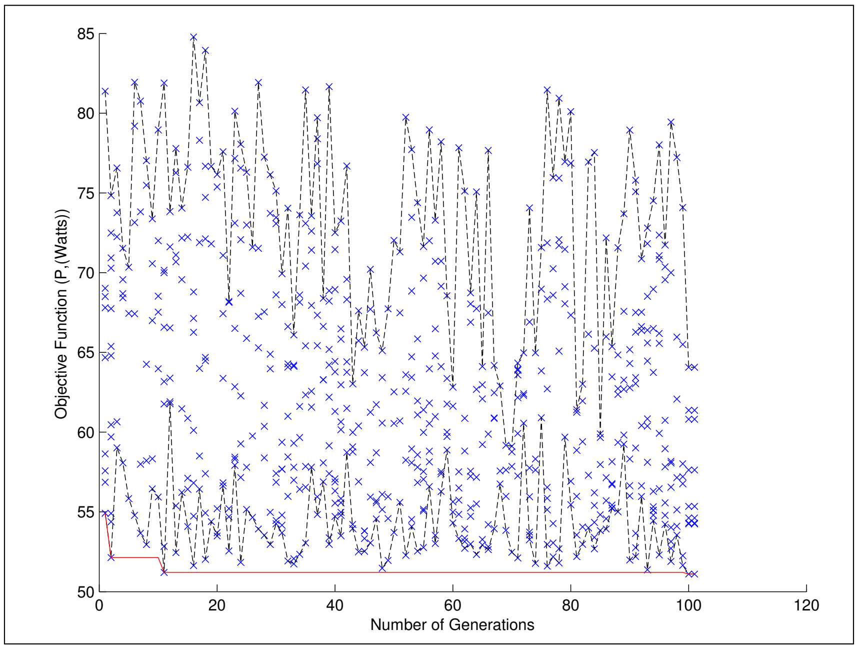

Figure 7.13: Fitness verse Number of Generations Case 2a (Objective Function = $PropellerPower$)

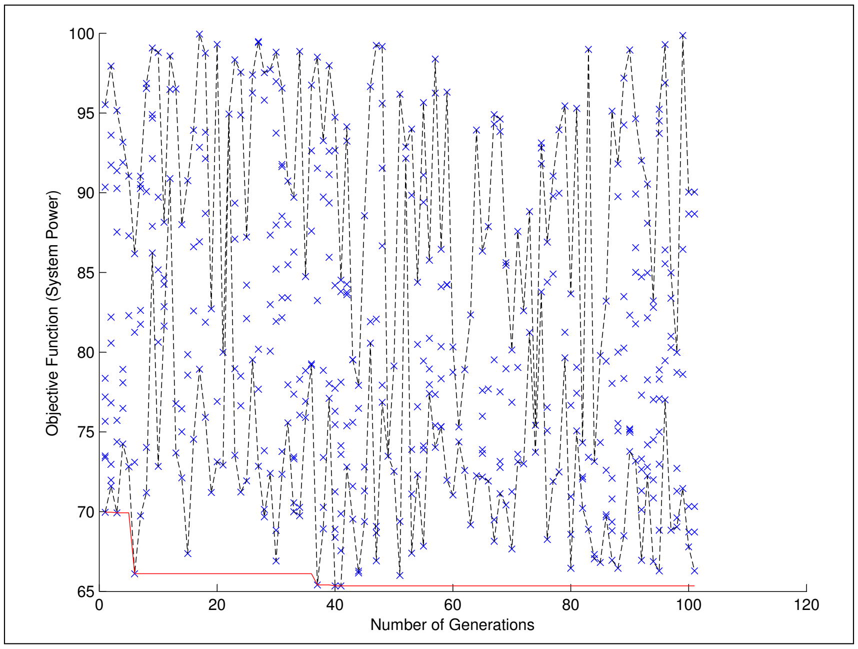

Figure 7.14: Fitness verse Number of Generations Case 2b (Objective Function = $SystemPower$)

Each case was run for 100 generations to show that the optimizer had converged on a solution. For Case 1 the optimizer found the "best" solution at generation 24 with an objection function value of 0.0218. Case 2a the “best” solution was found on generation 11 with an objection function value of 51.19 Watts. Case 2b the “best” solution was found on generation 40 with an objection function value of 65.35 Watts. It is also shown that Case 2a and 2b finds a wider range of values compared to Case 1 were the values are more localized. The performance the three propellers are shown in Tables 7.5 and 7.6. Detailed properties (i.e. blade angles, chord sizes, pitch, and element position) for the two cases are shown in Tables F.1, F.2, F.3 in Appendix F.

Table 7.5: Propeller Performance Parameters for Cruise Case 1

Case 1 (Objective Function = C_p)				
J	0.8000		Pitch at 3/4 radius	9.83
Ct	0.0244		Diameter (in)	12.0
Cp	0.0218		$\beta_{3/4}$ (°)	19.23
η_{prop}	89.54%		RPM	5500
			Motor Power (Watts)	66.20
Motor Voltage (Volts)	7.78		Torque (oz-in)	12.87
Motor Current (Amps)	8.51		Thrust (oz)	7.55
η_{motor}	71.70%		η_{system}	62.20%

Table 7.6: Propeller Performance Parameters for Cruise Case 2a and 2b

Case 2a (Objective Function = $PropellerPower$)				
J	0.8085		Pitch at 3/4 radius	8.67
Ct	0.0248		Diameter (in)	12.0
Cp	0.0220		$\beta_{3/4}$ (°)	17.16
η_{prop}	91.14%		RPM	5442
			Motor Power (Watts)	64.62
Motor Voltage (Volts)	7.69		Torque (oz-in)	12.72
Motor Current (Amps)	8.40		Thrust (oz)	7.51
η_{motor}	71.63%		η_{system}	65.29%
Case 2b (Objective Function = $SystemPower$)				
J	0.8359		Pitch at 3/4 radius	10.20
Ct	0.0283		Diameter (in)	11.6
Cp	0.0261		$\beta_{3/4}$ (°)	20.50
η_{prop}	90.64%		RPM	5433
			Motor Power (Watts)	65.35
Motor Voltage (Volts)	7.70		Torque (oz-in)	12.84
Motor Current (Amps)	8.49		Thrust (oz)	7.52
η_{motor}	71.53%		η_{system}	64.83%

Table 7.5 shows that both propellers produced slightly higher than the desired thrust of 7.5 ounces. This is due to a tolerance set in the objective function that allowed the optimizer to produce pick propellers that produced more thrust in the early generations and then bring the thrust down as close as possible to the desired thrust. The propeller from Case 1 has an overall system efficiency of 62.20%, Case 2a has a slightly better system efficiency of 65.29%, and Case 2b has a system efficiency of 64.83%.. For this flight condition using the propeller power proved to provide better results. The system power should provide better results but

does not. This could be due the selection process from the optimizer. The optimizer was executed a total of three times to try to find a solution for the system power that exceeded the propeller power case with no success.

For these Cases a baseline propeller for comparison was chosen to be an APC 11x10E. This propeller is a commercially available propeller that is 11 inches in diameter with a constant pitch of 10 and is designed for electric motors.This was chosen due to its similar diameter and pitch compared to the optimized propellers. Figures 7.15 and 7.16 show the chord and blade angle distribution across the radius of the blade. The chord in the baseline propeller expands greatly until mid-blade span then decreases to the tip while Case 1 tapers to the tip. Case 2a's chord expands before it tapers to the tip, and Case 2b maintain until half the blade span then tapers to the tip. Both cases show the blade angle to taper slowly to the tip. All of optimized propellers are shown to have a smaller chord than the baseline propeller. The baselines root angle is approximately $55°$ and tapers to $\approx 18°$ while Case 1's root blade angle is $57.75°$ and tapers to $1.5°$ at the tip. Case 2a's root angle is smaller at $40.95°$ and tapers to $8.7°$, and Case 2b's root angle is larger at $54°$ and tapers to $4.80°$.

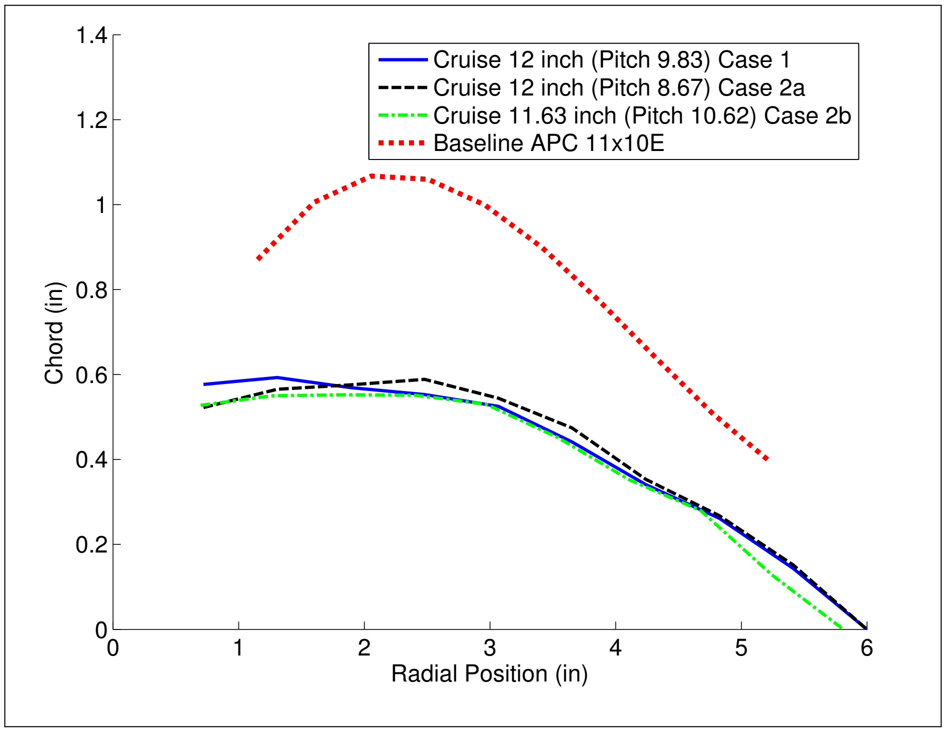

Figure 7.15: Chord Distribution for Cruise Condition Propellers

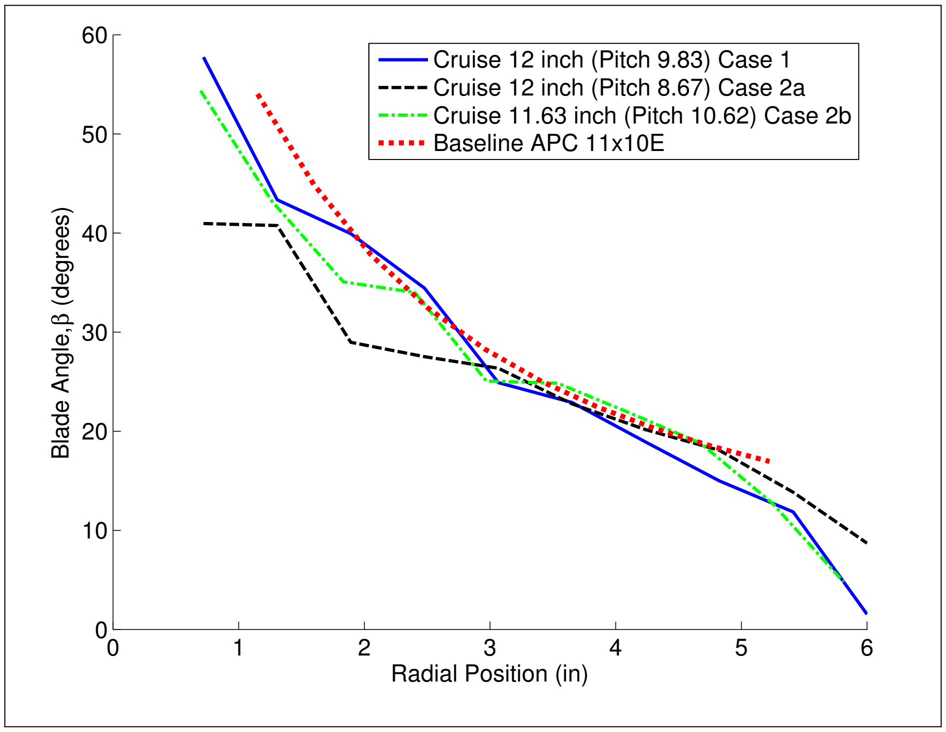

Figure 7.16: Blade Angle Distribution for Cruise Condition Propellers

Figure 7.17 shows the possible blade chord dimensions and optimized blade chord dimensions for Case 1 only. This plot is for a fixed diameter of 12 inches. If the optimizer would have chosen a different diameter the plot would be different. The solid line is the contour that was picked, and the dashed and dotted lines are the maximum and minimum possible chord contours.

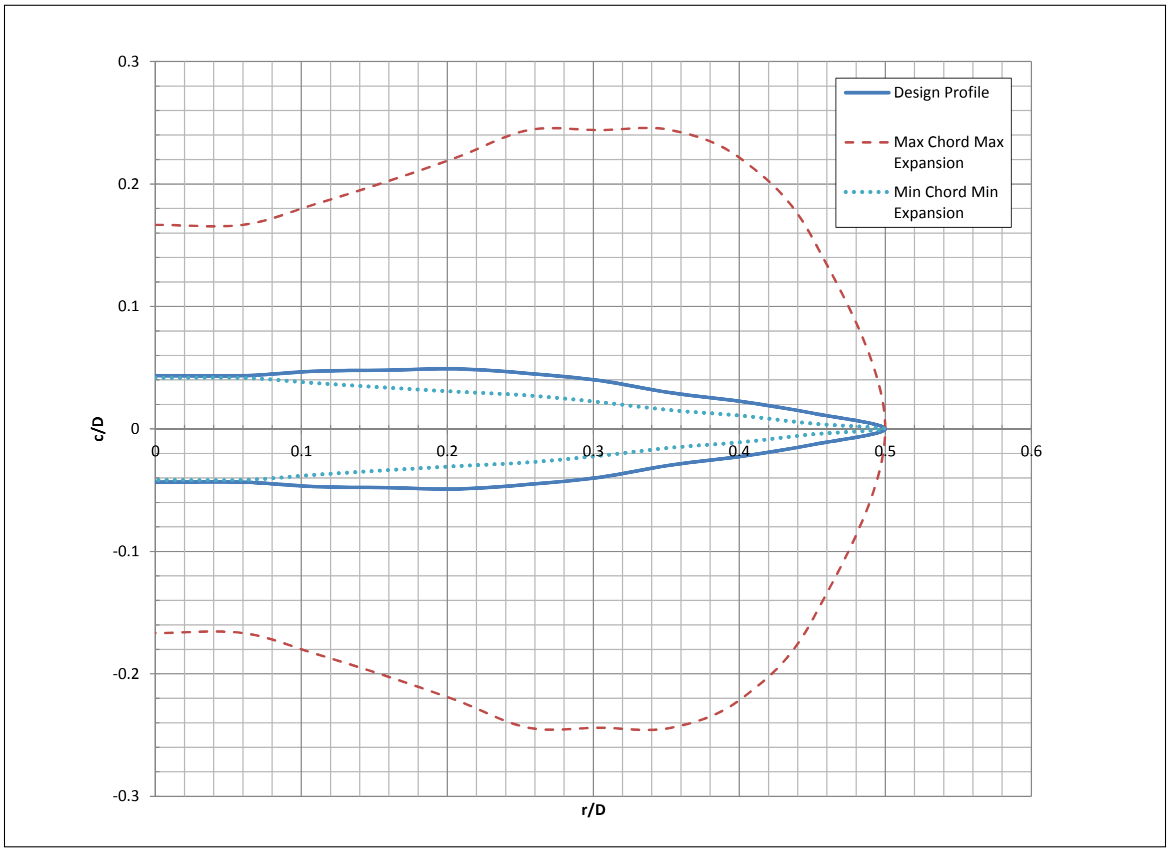

Figure 7.17: Comparison of Possible Blade Chord Profiles for Case 1

Different number of elements used to find the performance parameters was used. Originally 10 elements were used to describe each blade, but 100 was also used to see if there was any improvement when lift distribution over the blades was integrated. The 100 elements were found using interpolation between the original 10 optimized elements. For Case 1 $C_t = 0.0256$ and $C_p = 0.0225$ with 10 elements and with 100 elements the values changed to $C_t = 0.0253$ and $C_p = 0.0222$. Figure 7.18 shows the lift distribution on the blade. The increased elements smooths the distribution, but it does not make a significant difference in the integrated values.

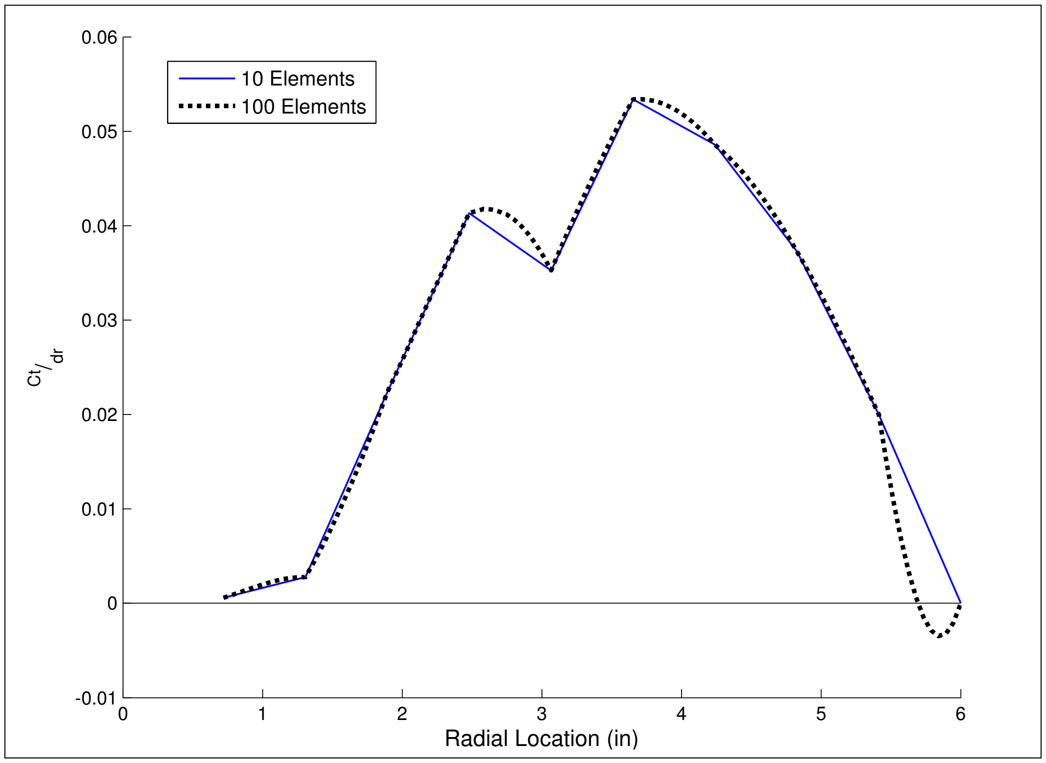

Figure 7.18: Lift Distribution on Blade Comparing the Number of Elements used to find Performance Parameters for Case 1

The performance parameters for each of the propellers were evaluated from a free stream velocity of 10 mph to whenever the propeller fails to produce any thrust. The APC 11x10E baseline propeller was included for comparison. The data for the baseline propeller was calculated using the same propeller code and XFOIL used to calculate the performance parameters of the other propellers. The thrust was calculated assuming an input voltage of 11.1 volts for the same motor used in optimizer, Brushless Park 450. This simulates a full throttle input throughout the free stream velocity range. The rotation speed, rpm, was found through an iteration process between the electric motor model and the propeller performance codes. Figure 7.19 shows the thrust verse free stream with the rpm at 11.1 volts, Figure 7.20 shows the power for the propeller verse free stream and current draw at 11.1 volts, and Figure 7.21 shows the efficiency of the propeller verse free stream. It should be noted that the power plotted in Figure 7.20 is the power required for the propeller. The motor will require more power due to losses in the motor. The optimized locations are shown on the Figures with indicators.

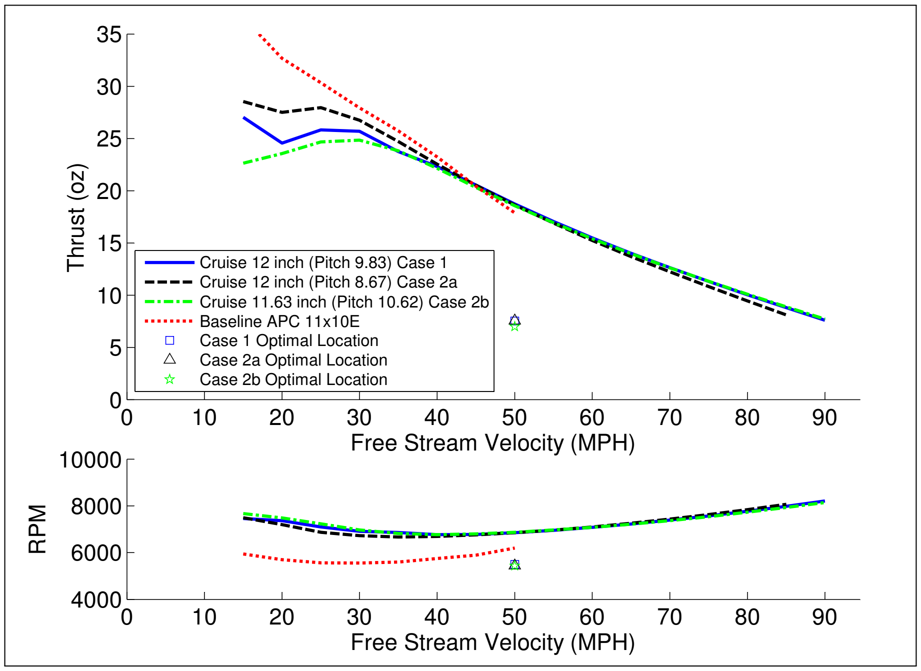

Figure 7.19: Comparison of Case 1, Case 2a, and Case 2b Propellers Against Baseline Propeller Thrust Over a Range of Free Stream Velocities

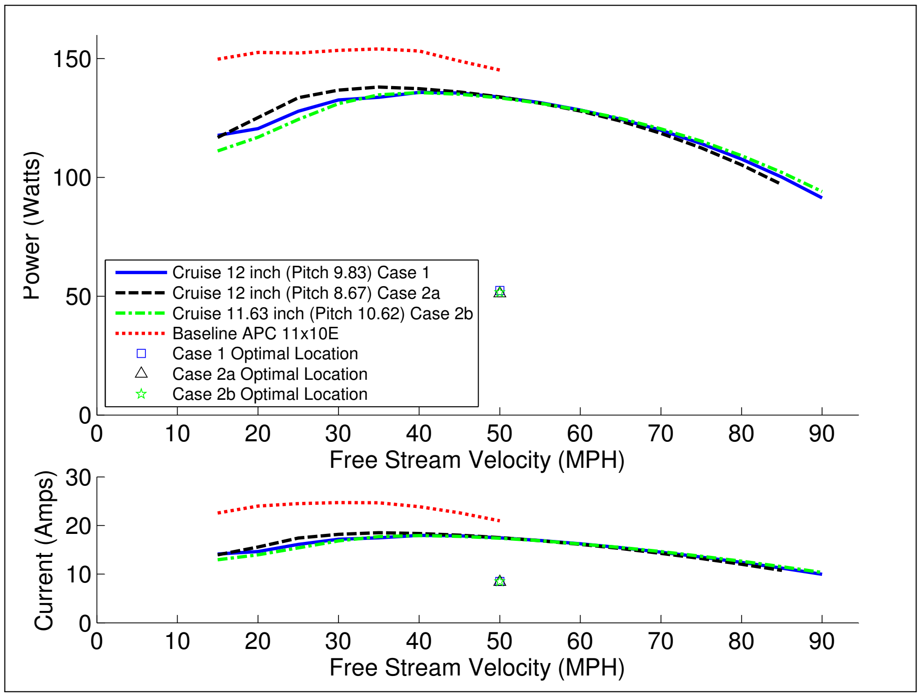

Figure 7.20: Comparison of Case 1, Case 2a, and Case 2b Propellers Against Baseline Propeller Power Over a Range of Free Stream Velocities

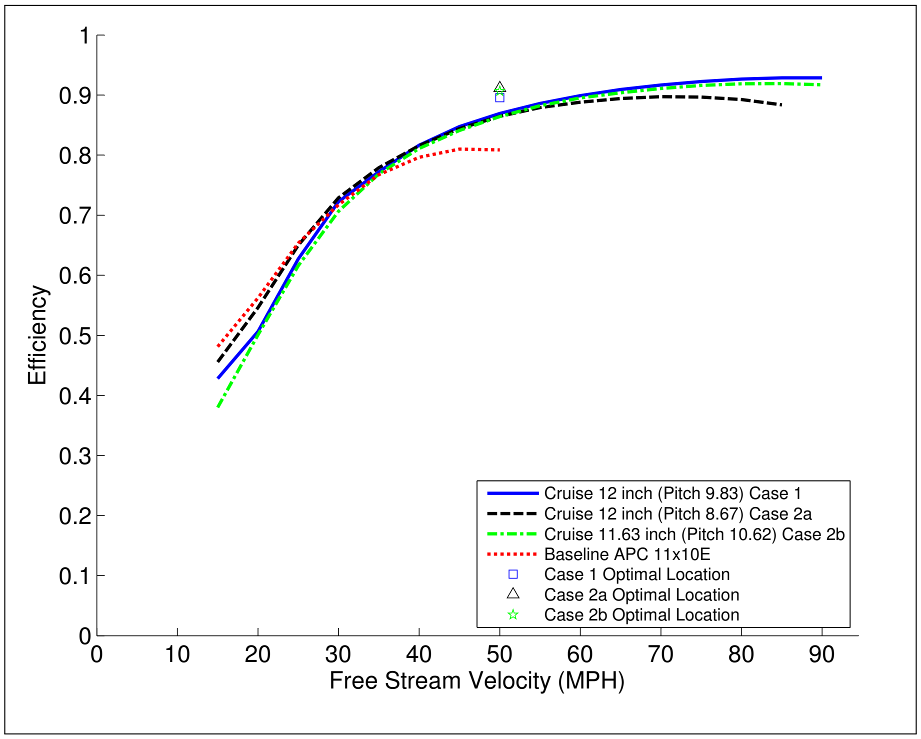

Figure 7.21: Comparison of Case 1, Case 2a, and Case 2b Propellers Against Baseline Propeller Efficiency Over a Range of Free Stream Velocities

All of the optimized propeller are shown to require less power than the baseline propeller with approximately equal thrust. This leads to the baseline propeller be less efficient than the optimized locations. The optimized locations are also shown to have a higher efficiency at the design speed then the propeller at full throttle at the design speed. This is due to the increase in power required and more losses in the motor and propeller. All of the propellers have about the same blade angle distribution, but different chord distributions. Since the blades have about the same blade angles then only difference is the airfoils. The optimized airfoils are shown to out perform the baseline propellers airfoils. Case 2a requires less power and closely matches the desired thrust at the optimized locations it was chosen as the "best" for cruise.

7.4 Propeller Optimized for Climb

The second optimization that was tested was the climb condition. From Table 7.2 the propeller needs to produce as much thrust possible at a free stream velocity of 15 miles per hour given the maximum current draw the motor can perform at maximum voltage which was set to 11.1 volts to simulate a 3-cell lithium polymer battery power supply for the motor. If the power is nearly constant for the propeller then to be as efficient as possible thrust needs to be maximized. The optimizer is designed to minimize a problem. This leads to inverse of the thrust is used as the objective function. The two different objective functions, one non-dimensional and the other with dimensional values, and are shown in Table 7.7 where z is the fitness. The range of values used by the optimizer are shown in Table 7.8.

Table 7.7: Objective Functions used for Climb Condition

Case Number	Objective Function:
3	$z = \frac{1}{C_t}$
4	$z = \frac{1}{F_t(ounces)}$

The values of β could only decrease to the tip, and the chord was allowed to expand to approximately the three quarters radius location then decrease to zero at the tip. For both

Table 7.8: Optimizer Limits for Climb Condition

Parameter	Minimum	Maximum
β_{start} $(^{\circ})$	20	65
$chord_{start}(in)$	0.5	2.0
$Diameter(in)$	8	12
$Motor\, Current(Amps)$	1	14

of the objective functions used the optimizer was run for 100 generations with 5 pattern searches per generation and a population size of 30 members. The number of members was increased from the cruise condition due to the inadequate number of members that would find a viable solution in each generation. Shown in Figures 7.22 and 7.23 a large number the members did not provide a "good" solution. These Figures show the evaluated objective function for each member of each generation. The members that were "bad" propellers are not plotted. As discussed in the previous chapter these "bad" propellers do not meet the requirements and the objective function assigns them a value of 10^8. Each member is represented by a "x." A dashed line connects the maximum values of each generation as well as the minimum values. A solid line is used to show the best fitness found as of that generation.

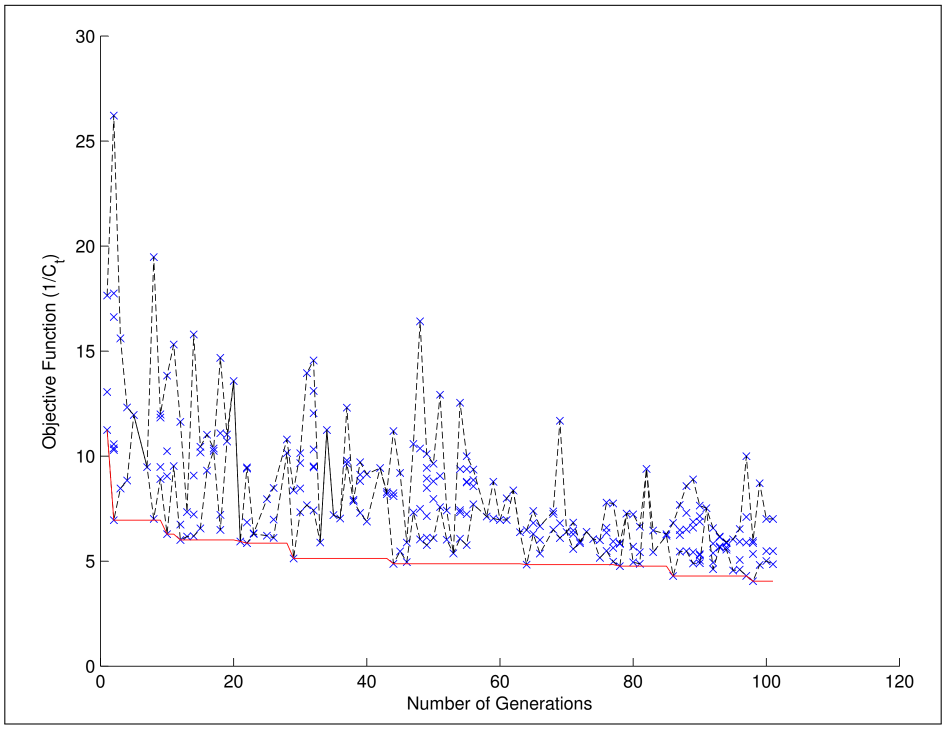

Figure 7.22: Fitness verse Number of Generations Case 3 (Objective Function = $\frac{1}{C_t}$)

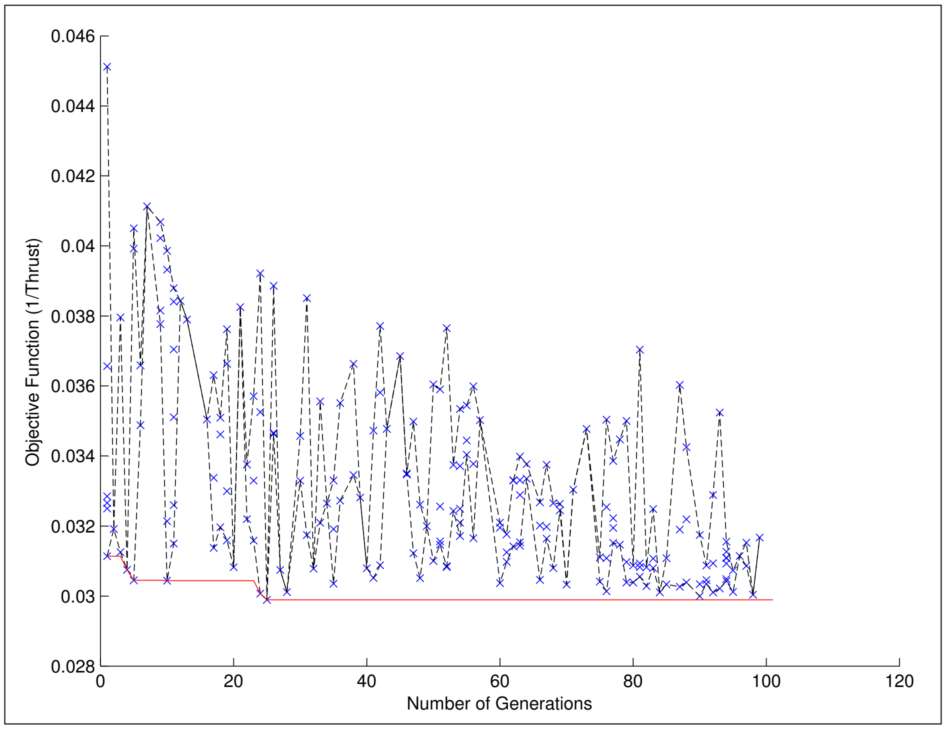

Figure 7.23: Fitness verse Number of Generations Case 4 (Objective Function = $\frac{1}{F_t}$)

Each case was run for 100 generations to show that the optimizer had converged on a solution. For Case 3 the optimizer found the “best” solution at generation 99 with an objection function value of 4.826 and took 15 hours to complete the 100 generations. For Case 4 the “best” solution was found on generation 25 with an objection function value of 0.030 and took 11 hours to complete. Again it is shown that Case 4 finds a wider range of values compared to Case 3 were the values are more localized as the number of generations increase. The performance the two propellers are shown in Table 7.9. Detailed properties (i.e. blade angles, chord sizes, pitch, and element position) for the two cases are shown in Tables G.1 and G.2 in Appendix G. Table 7.9 shows that both propellers produced slightly

Table 7.9: Propeller Performance Parameters for Climb Case 3 and Case 4

Case 3 (Objective Function = $\frac{1}{C_t}$)				
J	0.2452		Pitch at 3/4 radius	5.25
Ct	0.2072		Diameter (in)	8.12
Cp	0.0982		$\beta_{3/4}$ (°)	15.38
η_{prop}	51.74%		RPM	7956
			Motor Power (Watts)	126.19
Motor Voltage (Volts)	11.1		Torque (oz-in)	17.21
Motor Current (Amps)	11.37		Thrust (oz)	28.10
η_{motor}	74.62%		η_{system}	38.61%
Case 4 (Objective Function = $\frac{1}{F_t}$)				
J	0.2067		Pitch at 3/4 radius	3.29
Ct	0.1054		Diameter (in)	9.73
Cp	0.0415		$\beta_{3/4}$ (°)	8.26
η_{prop}	52.51%		RPM	7876
			Motor Power (Watts)	131.10
Motor Voltage (Volts)	11.1		Torque (oz-in)	17.60
Motor Current (Amps)	11.81		Thrust (oz)	28.87
η_{motor}	74.05%		η_{system}	38.88%

lower than the maximum allowable current of 18 Amps. This is due to a tolerance set in the objective function that allowed the optimizer to produce pick propellers that used less than the maximum current in the early generations and then bring the current up to drive the thrust as high as possible. The propeller from Case 3 has an overall system efficiency of

38.61% and produces 28.10 ounces of thrust at a free stream velocity of 15 miles per hour. Case 4 has a slightly better system efficiency of 38.88% and produces 28.87 ounces of thrust. Case 4 with proved to produce a more efficient propeller with more thrust requiring less power than Case 3. An APC 8x6E propeller was chosen for baseline comparison due to its very similar nature to the optimized climb propellers. The baseline propeller has a diameter of 8 inches with a constant pitch of 6. Figures 7.24 and 7.25 show the chord and blade angle distribution across the radius of the blade. The chord in Case 3 expands before it tapers to the tip while Case 4 remains approximately constant for half of the radius. Both cases show the blade angle to follow an exponential decay to tip. Case 3's root blade angle is 58.05° and is 4.89° at the tip. Case 4's root angle is smaller at 42.67° and tapers to 1.46°. It is shown that chords are much larger and the blade angles are approximately when compared to the baseline propeller.

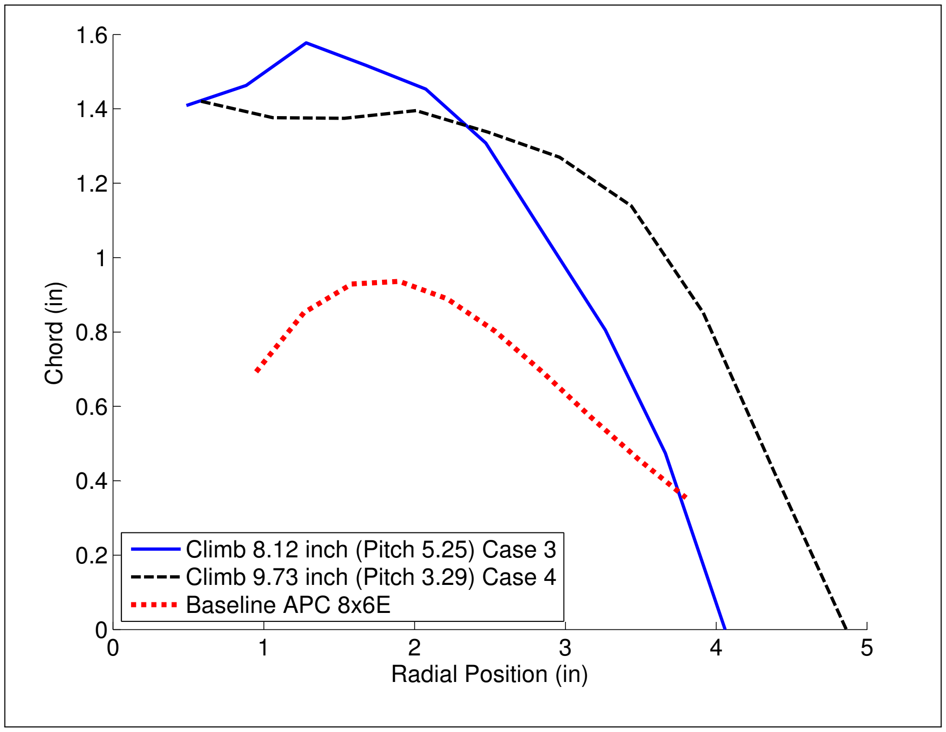

Figure 7.24: Chord Distribution for Climb Condition Propellers

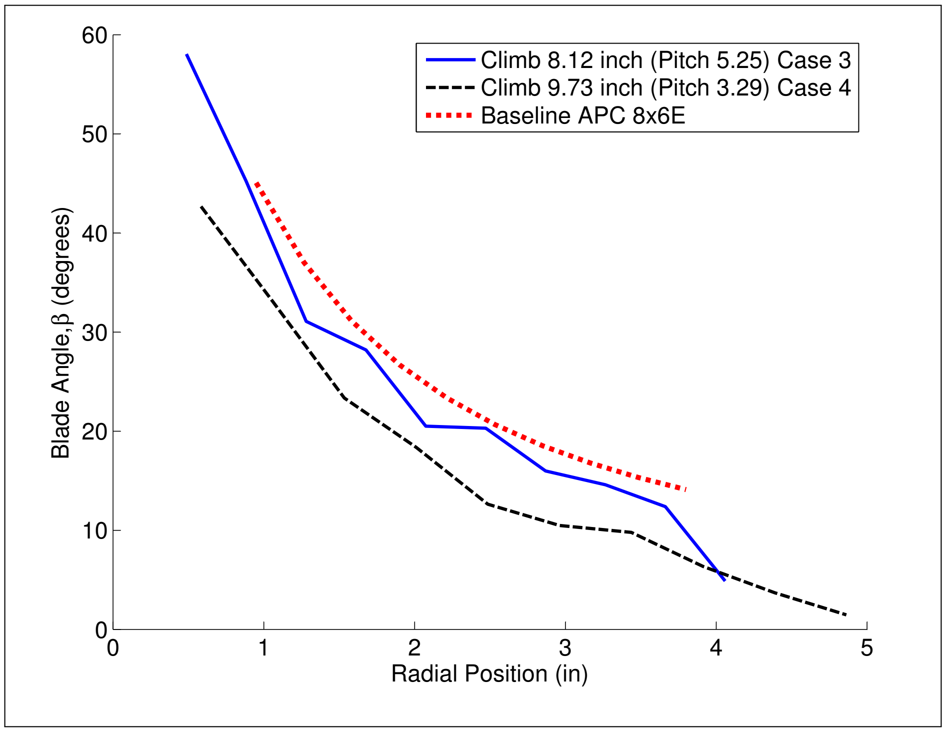

Figure 7.25: Blade Angle Distribution for Climb Condition Propellers

To see if any of the propeller blade is stalled at the optimized climb location Figure 7.26 was created. This plot shows that the relative angle of attack on each element for Case 3 which reaches a maximum around 8°. This is not close to the stall range which is important since an average Reynolds number was used for these airfoils, and the Reynolds number dictates stall location. Case 4 is not shown due to its similar blade angles and flight conditions that produces a similar result.

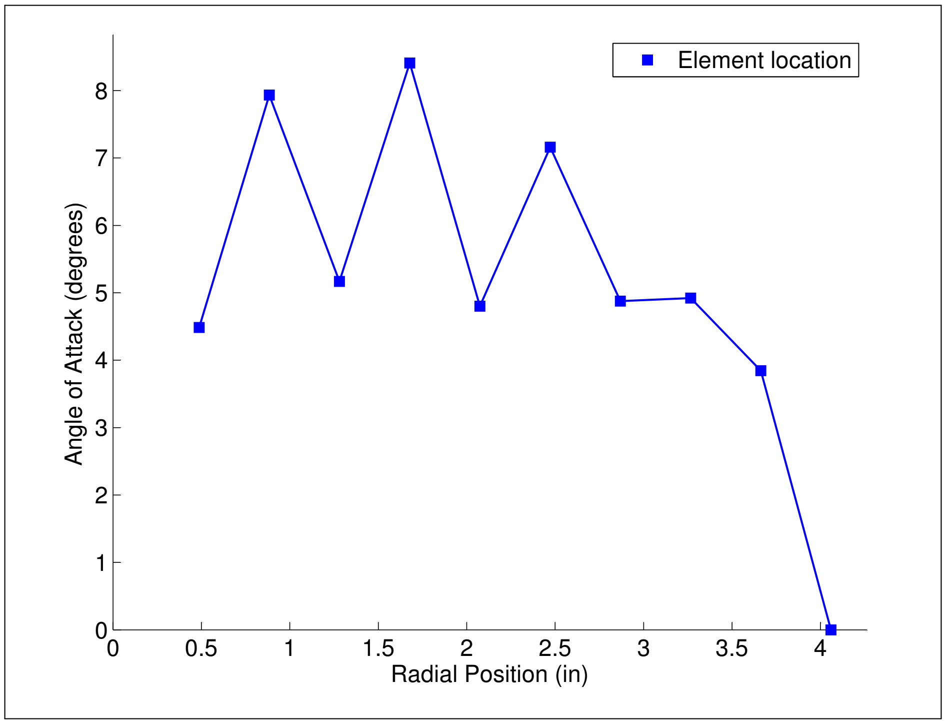

Figure 7.26: Case 3 Angle of Attack Distribution

The performance parameters for each of the propellers were evaluated from a free stream velocity of 10 mph to whenever the propeller fails to produce any thrust. A baseline propeller was included for comparison.This propeller was chosen to be an APC 8x6E due to it similar geometry to the optimized propellers in Case 3 and 4. The data for the baseline propeller was calculated using the same propeller code and XFOIL used to calculate the performance parameters of the other propellers. The thrust was calculated assuming an input voltage of 11.1 volts for the same motor used in optimizer, Brushless Park 450. This simulates a full throttle input throughout the free stream velocity range. The rotation speed, rpm, was found through an iteration process between the electric motor model and the propeller performance codes. Figure 7.27 shows the thrust verse free stream with the rpm at 11.1 volts, Figure 7.28 shows the power for the propeller verse free stream and current draw at 11.1 volts, and Figure 7.29 shows the efficiency of the propeller verse free stream. It should be noted that the power plotted in Figure 7.28 is the power required for the propeller. The motor will require more power due to losses in the motor. The optimized locations are shown on the Figures with indicators.

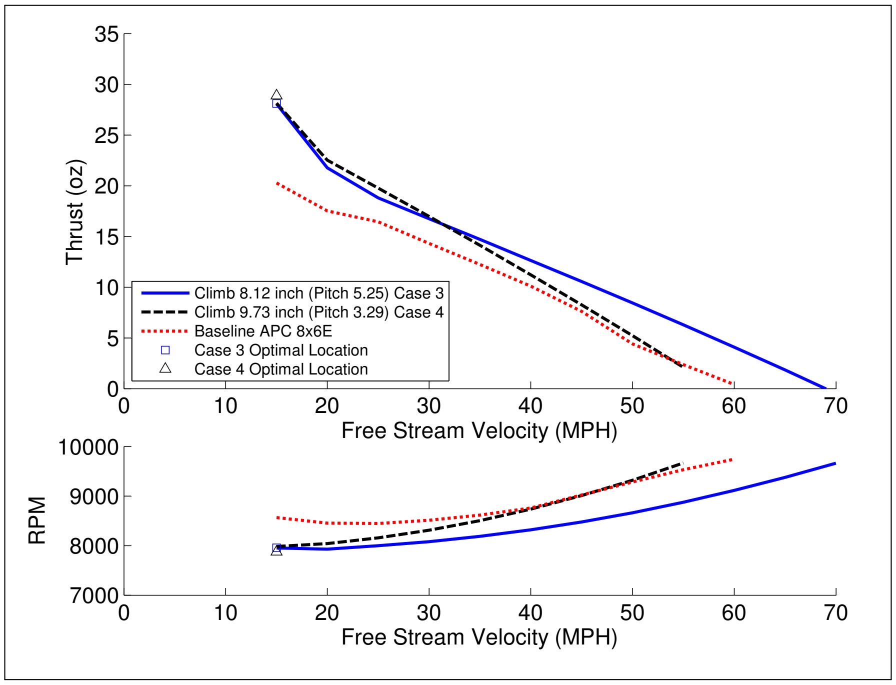

Figure 7.27: Comparison of Case 3 and Case 4 Propellers Thrust over a Range of Free Stream Velocities

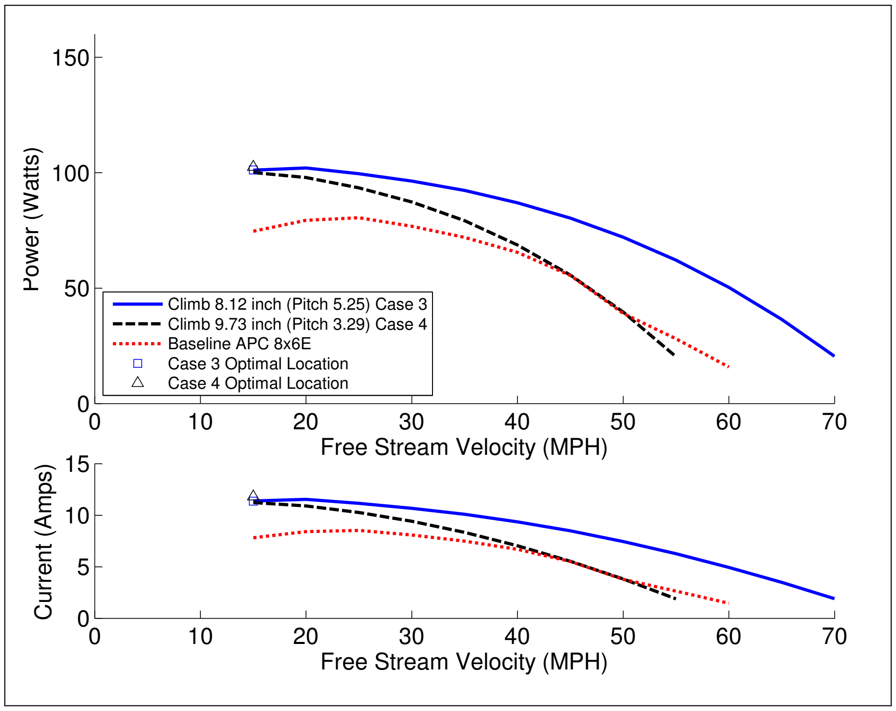

Figure 7.28: Comparison of Case 3 and Case 4 Propellers Power over a Range of Free Stream Velocities

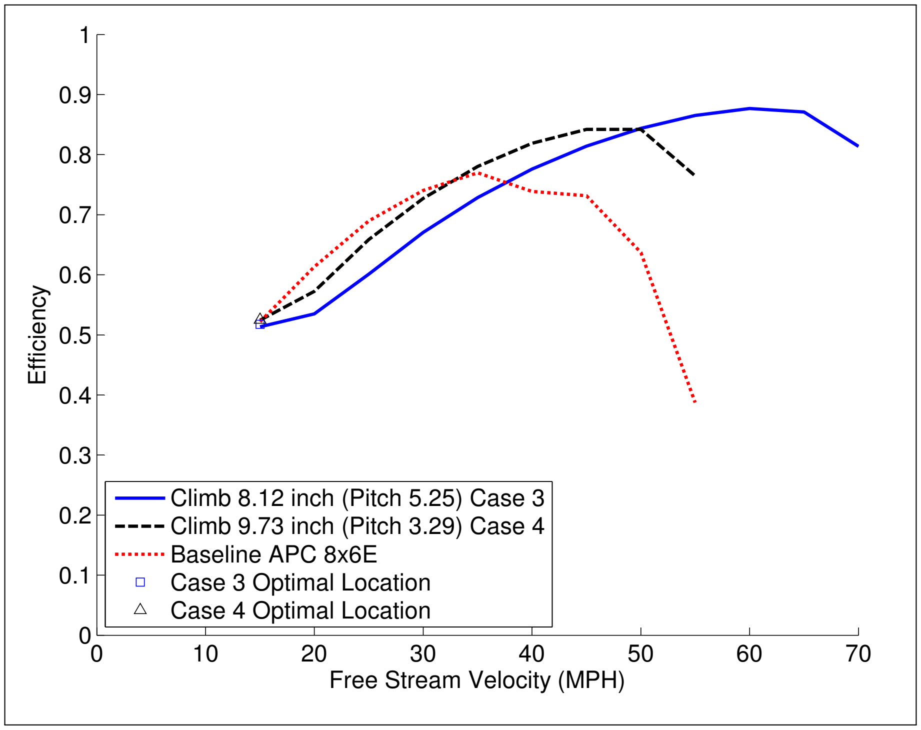

Figure 7.29: Comparison of Case 3 and Case 4 Propellers Efficiency over a Range of Free Stream Velocities

The optimized propeller from Case 4 is shown to have a less thrust, require less power, and have a higher efficiency throughout the free stream velocities. Both propellers are shown to have much larger chords than the baseline while the blade angles follow the same trend. The baseline propeller requires more power compared to the optimized propellers. The Case 3 propeller will produce thrust in a wider range in free stream velocities at full throttle. Case 3 and Case 4 were marginally different, but Case 4 met the design requirements better than Case 3.

7.5 Propeller Optimized for Climb-Cruise

The third optimization that was tested was the multipoint climb cruise condition. From Table 7.2 the propeller needs to produce as much thrust possible at a free stream velocity of 15 miles per hour. The propeller also needs to produce 7.5 ounces of thrust at a free stream velocity of 50 miles per hour. Similar to the previous optimizations two different objective functions were used, but the it was changed slightly for the climb cases. An attempt was made combining the methods from the cruise and climb optimizer, but the optimizer could not make an initial population in a reasonable period of time. A minimum desired thrust of 24 ounces was then set for the climb condition. The first is non-dimensional and is maximizing the efficiency at the two different design conditions. The other tries to minimize the power at cruise and the thrust during climb. The two different objective functions are shown in Table 7.10 where z is the fitness. The range of values used by the optimizer are shown in

Table 7.10: Objective Functions used for Climb Cruise Condition

Case Number	Objective Function:
5	$z = 1 - \eta_{cruise}\eta_{climb}$
6	$z = \frac{Power}{Thrust}$

Table 7.11. The values of β could only decrease to the tip, and the chord was allowed to expand to approximately the three quarters radius location then decrease to zero at the tip. For both of the objective functions used the optimizer was run for 75 generations with 5

Table 7.11: Optimizer Limits for Climb-Cruise Condition

Parameter	Minimum	Maximum
β_{start} ($^\circ$)	45	65
$chord_{start}(in)$	0.5	2.0
$Diameter(in)$	8	12
RPM_{cruise}	3000	5500
RPM_{climb}	5000	8500
$Currrent_{cruise}\ (Amps)$	0	14
$Currrent_{climb}\ (Amps)$	0	18
$Voltage\ (Volts)$	0	11.1

pattern searches per generation and a population size of 15 members. As shown in Figures 7.30 and 7.31 a large number the members did not provide a "good" solution. These Figures show the evaluated objective function for each member of each generation. The members that were "bad" propellers are not plotted. As discussed in the previous chapter these "bad" propellers do not meet the requirements and the objective function assigns them a value of 10^8. Each member is represented by a "x." A dashed line connects the maximum values of each generation as well as the minimum values. A solid line is used to show the best fitness found as of that generation.

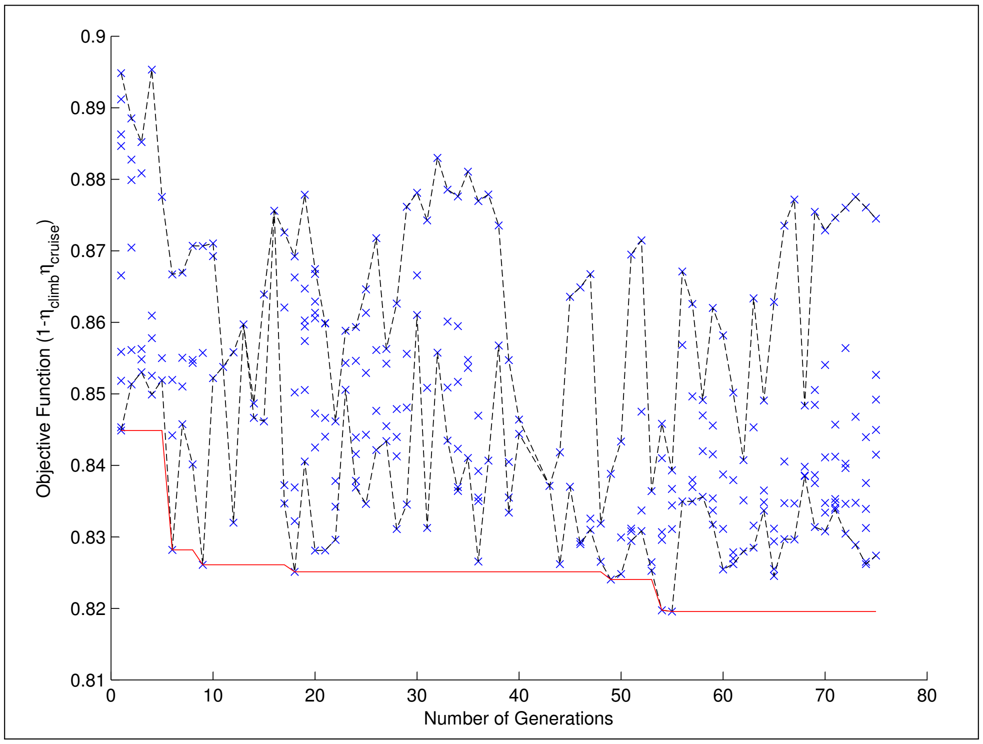

Figure 7.30: Fitness verse Number of Generations Case 5 (Objective Function = $1 - \eta_{prop}\eta_{motor}$)

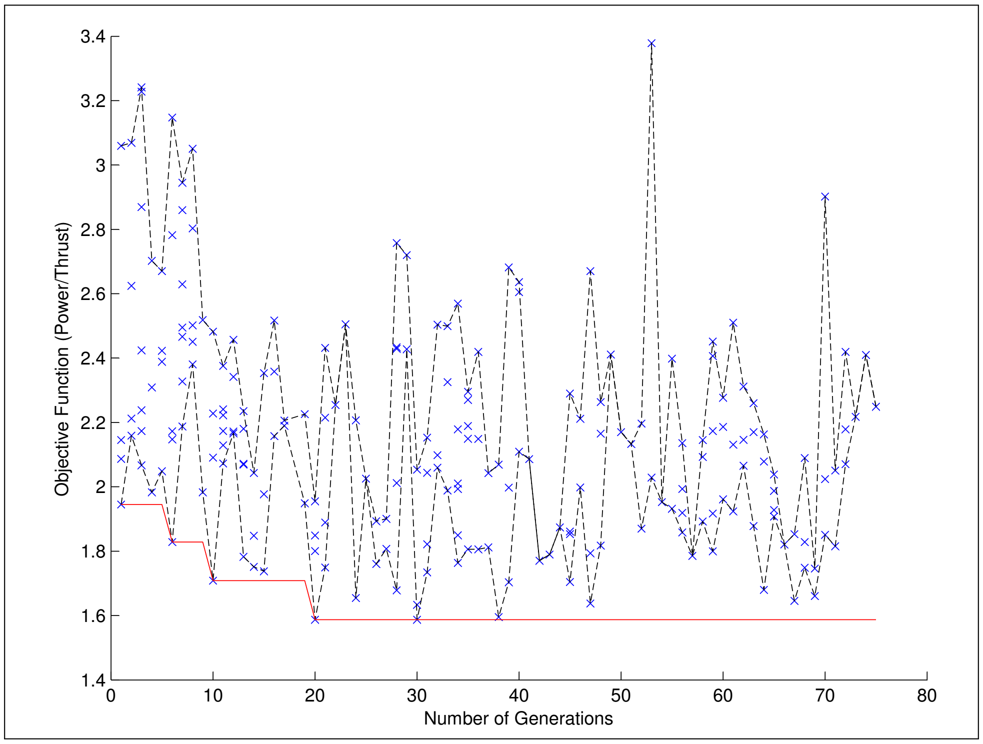

Figure 7.31: Fitness verse Number of Generations Case 6 (Objective Function = $\frac{Power_{Cruise}}{Thrust_{Climb}}$)

Each case was run for 75 generation to show that the optimizer had converged on a solution. For Case 5 the optimizer found the "best" solution at generation 56 with an objection function value of 0.8196 and took 5 hours to complete the 75 generations. For Case 6 the "best" solution was found on generation 30 with an objection function value of 1.587 and took 5.5 hours to complete. Again it is shown that Case 6 finds a wider range of values compared to Case 5 were the values are more localized as the number of generations increase. The performance the two propellers are shown in Tables 7.12 and 7.13. Detailed properties (i.e. blade angles, chord sizes, pitch, and element position) for the two cases are shown in Tables H.1 and H.2 in Appendix H.

Table 7.12: Propeller Performance Parameters for Climb Cruise Case 5

Case 5 (Objective Function = $1 - \eta_{cruise}\eta_{climb}$)				
Cruise				
$\beta_{3/4}$ (°)	19.61		Pitch at 3/4 radius	8.92
J	0.9009		Diameter (in)	10.6559
Ct	0.0457		RPM	5500
Cp	0.0470		Motor Power (Watts)	82.20
η_{prop}	87.60%		Torque (oz-in)	15.33
			Thrust (oz)	8.79
Motor Voltage (Volts)	8.11			
Motor Current (Amps)	10.14			
η_{motor}	69.82%		η_{system}	61.16%
Climb				
J	0.2684			
Ct	0.1413		RPM	5538
Cp	0.0739		Motor Power (Watts)	151.45
η_{prop}	51.32%		Torque (oz-in)	24.43
			Thrust (oz)	27.54
Motor Voltage (Volts)	9.37			
Motor Current (Amps)	16.16			
η_{motor}	62.66%		η_{system}	32.16%

Tables 7.12 and 7.13 shows that both propellers produced higher than the minimum thrust at climb conditions. The Case 5 propeller where the objective function was based on the efficiencies at the design points produced a thrust of 24.43 ounces with a system efficiency

Table 7.13: Propeller Performance Parameters for Climb Cruise Case 6

Case 6 (Objective Function = $\frac{Power}{Thrust}$)				
Cruise				
$\beta_{3/4}$ (°)	19.78		Pitch at 3/4 radius	8.80
J	0.9345		Diameter (in)	10.4566
Ct	0.0441		RPM	5404
Cp	0.0469		Motor Power (Watts)	68.78
η_{prop}	87.87%		Torque (oz-in)	13.43
			Thrust (oz)	7.59
Motor Voltage (Volts)	7.74			
Motor Current (Amps)	8.89			
η_{motor}	70.94%		η_{system}	62.34%
Climb				
J	0.2230			
Ct	0.1244		RPM	6792
Cp	0.0578		Motor Power (Watts)	190.41
η_{prop}	48.00%		Torque (oz-in)	26.15
			Thrust (oz)	33.82
Motor Voltage (Volts)	11.00			
Motor Current (Amps)	17.31			
η_{motor}	65.74%		η_{system}	31.55%

of 32.16%, and Case 6 produced a greater thrust at 33.82 ounces with a system efficiency of 31.55%. At cruise Case 5 produced 8.79 ounces of thrust with a system efficiency of 61.16%,, and Case 6 produced 7.59 ounces of thrust with a system efficiency of 62.34%. Case 5 and Case 6 peak in efficiency at 88%. Case 6 is shown to produce as much thrust as possible and achieve closer to a desired cruise speed compared to Case 5. A baseline propeller was added to the following series of Figures for comparison. The baseline propeller was chosen to be an APC 10x7E which has a diameter of 10 inches and a constant pitch of 7. Figures 7.32 and 7.33 show the chord and blade angle distribution across the radius of the blade. The chord in both cases expands before it tapers to the tip. Both cases show the blade angle to follow an exponential decay to approximately 15° at the tip. Case 5's root blade angle is 53.74° and is 16.42° at the tip. Case 6's root angle is smaller at 46.86° and tapers to 13.79°.

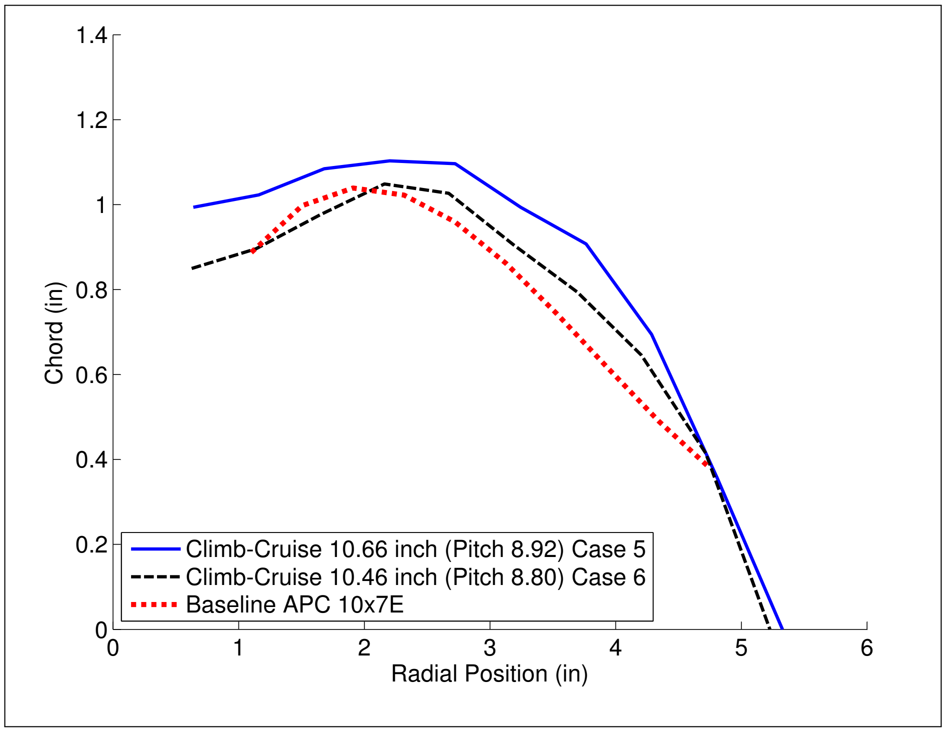

Figure 7.32: Chord Distribution for Climb Cruise Condition Propellers

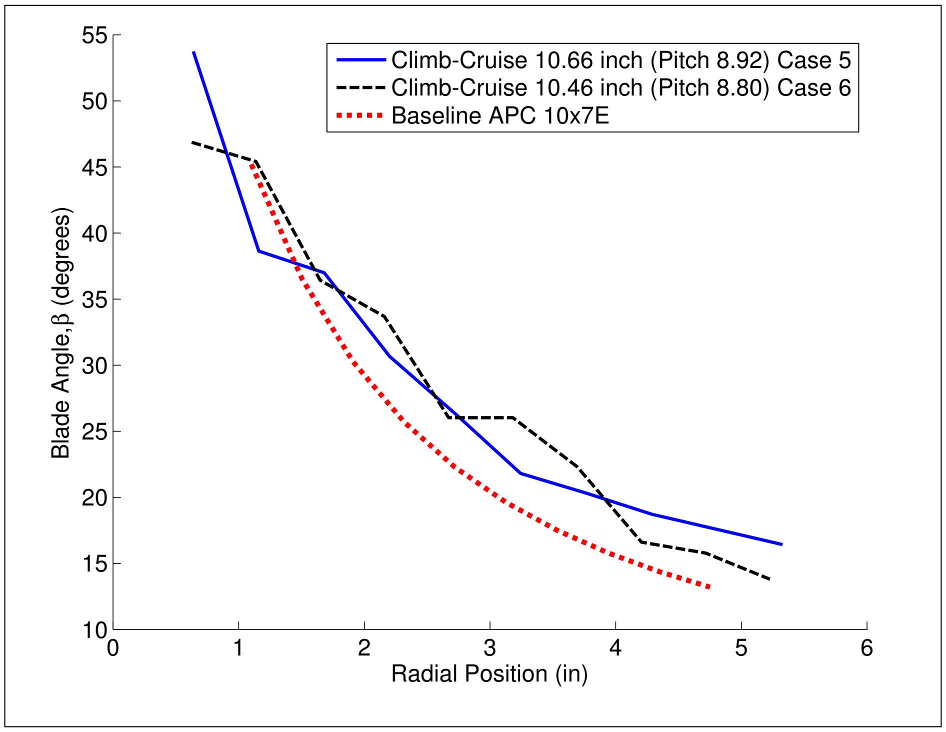

Figure 7.33: Blade Angle Distribution for Climb Cruise Condition Propellers

The performance parameters for each of the propellers were evaluated from a free stream velocity of 15 mph to whenever the propeller fails to produce any thrust. A baseline propeller was included for comparison. For these cases an APC 10x7 thin electric propeller was chosen because it is similar to Case 5 and Case 6 propellers. This is a 10 inch propeller with a constant blade pitch of 7 inches. The data for the baseline propeller was obtained from the UIUC Propeller Database [33]. Figure 7.34 shows the thrust verse free stream, Figure 7.35 shows the power for the propeller verse free stream, and Figure 7.36 shows the efficiency of the propeller verse free stream. It should be noted that the power plotted in Figure 7.35 is the power required for the propeller. The motor will require more power due to losses in the motor. The Figures are plotted at a fixed rotational speed for the propeller which is the optimized rotational speed.

The performance parameters for each of the propellers were evaluated from a free stream velocity of 15 mph to whenever the propeller fails to produce any thrust. A baseline propeller was included for comparison.This propeller was chosen to be an APC 10x7E due to it similar geometry to the optimized propellers in Case 5 and 6. The data for the baseline propeller was calculated using the same propeller code and XFOIL used to calculate the performance parameters of the other propellers. The thrust was calculated assuming an input voltage of 11.1 volts for the same motor used in optimizer, Brushless Park 450. This simulates a full throttle input throughout the free stream velocity range. The rotation speed, rpm, was found through an iteration process between the electric motor model and the propeller performance codes. Figure 7.34 shows the thrust verse free stream with the rpm at 11.1 volts, Figure 7.35 shows the power for the propeller verse free stream and current draw at 11.1 volts, and Figure 7.36 shows the efficiency of the propeller verse free stream. It should be noted that the power plotted in Figure 7.35 is the power required for the propeller. The motor will require more power due to losses in the motor. The optimized locations are shown on the Figures with indicators.

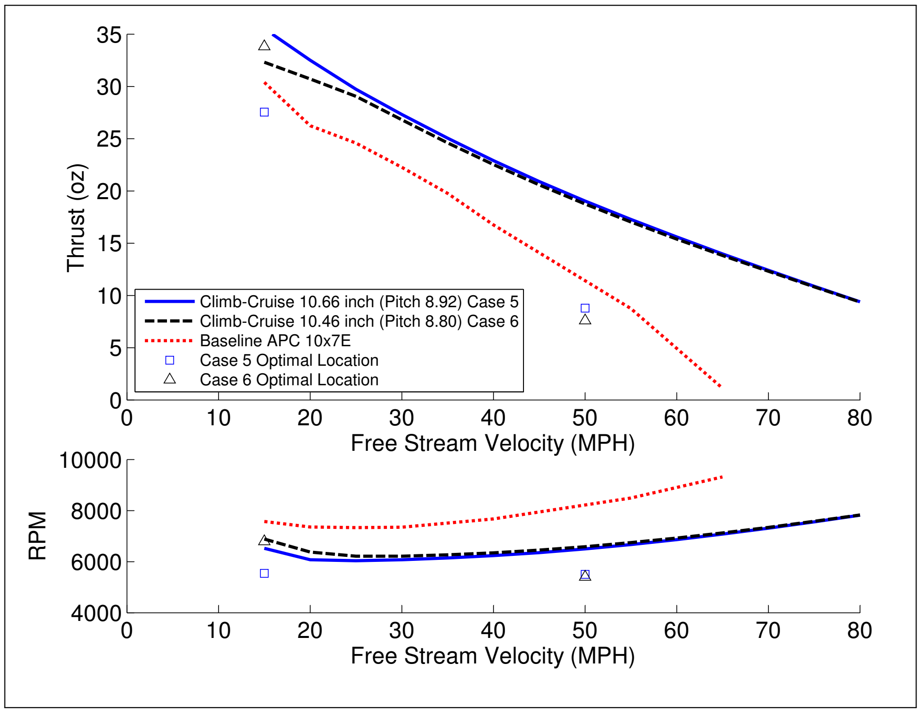

Figure 7.34: Comparison of Case 5 and Case 6 Propellers Thrust over a Range of Free Stream Velocities

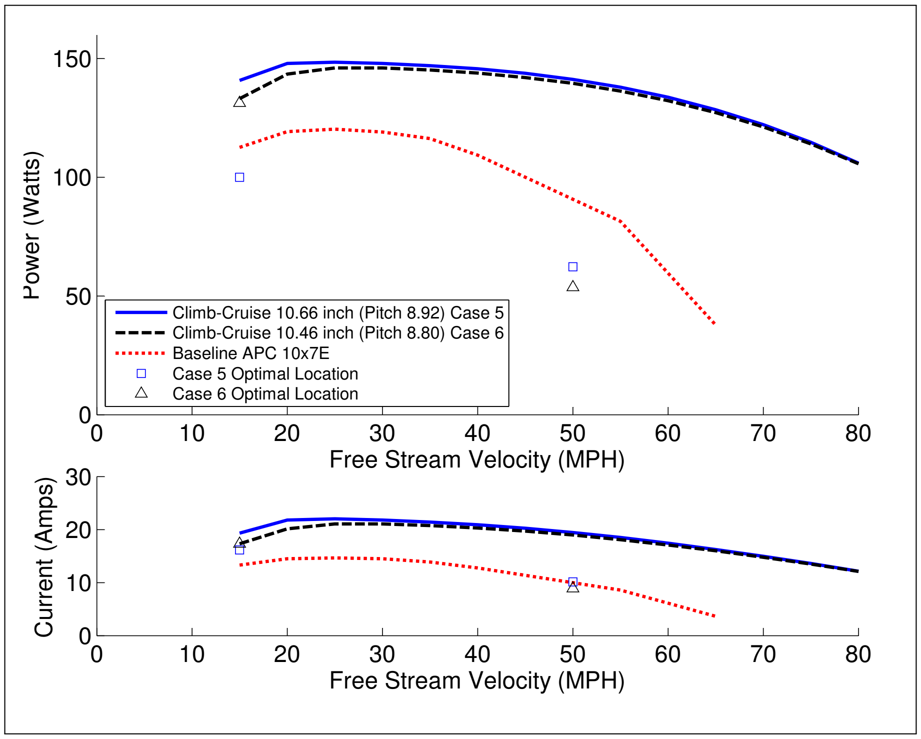

Figure 7.35: Comparison of Case 5 and Case 6 Propellers Power over a Range of Free Stream Velocities

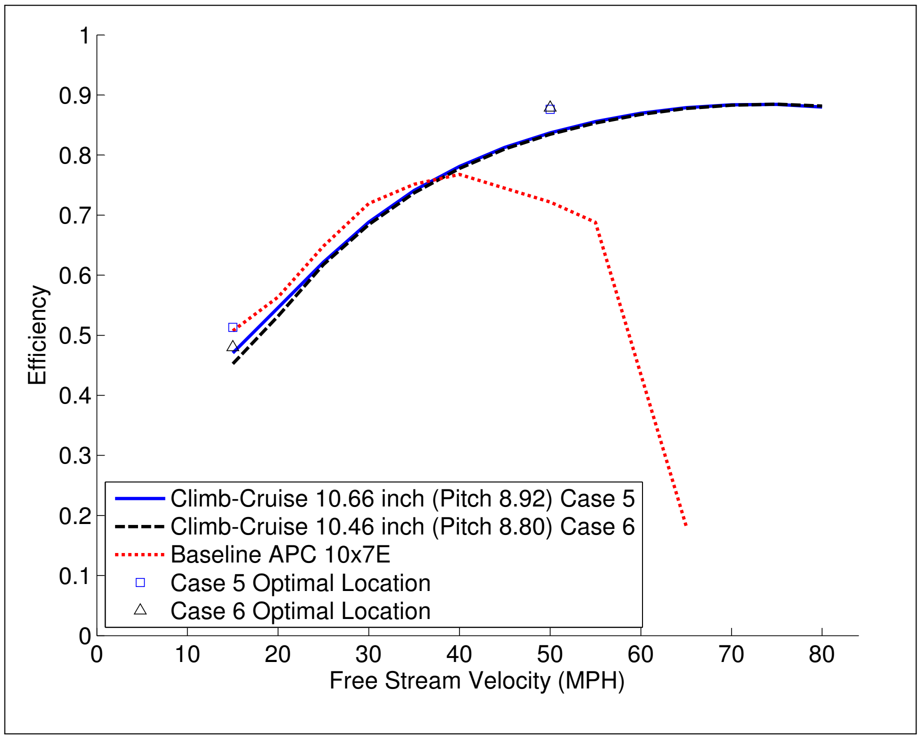

Figure 7.36: Comparison of Case 5 and Case 6 Propellers Efficiency over a Range of Free Stream Velocities

The optimized propellers from Case 5 and Case 6 is shown to have approximately the same performance parameters at the full throttle setting, but the Case 6 propeller out performs Case 5 at the design condition for climb. Both optimized propellers produce more thrust and require less power when compared to the baseline propeller. The geometry Case 5 and Case 6 are only slightly different from the baseline. The chord distributions follow the same trends with approximately the same values, and the blade angles follow the same trends and approximate values as well. The optimized propellers are also shown to operate at a much higher free stream velocity compared to the the baseline propeller. Case 5 does produce higher thrust and require more power compared to the baseline and Case 6. Case 6 does produce higher thrust at the climb design point and more closely matches the thrust desired for cruise.

7.6 Cruise, Climb, and Climb-Cruise Comparisons

The “best” propeller from each of the optimized conditions was compared. The “best” cases were found to be Case 2a, Case 4, and Case 6. These propellers were shown to out perform the other cases. The blade angles are compared in Figure 7.37. All of the propellers follow the same trend with the Climb-Cruise and Cruise propellers have similar values and the Climb propeller having smaller values.

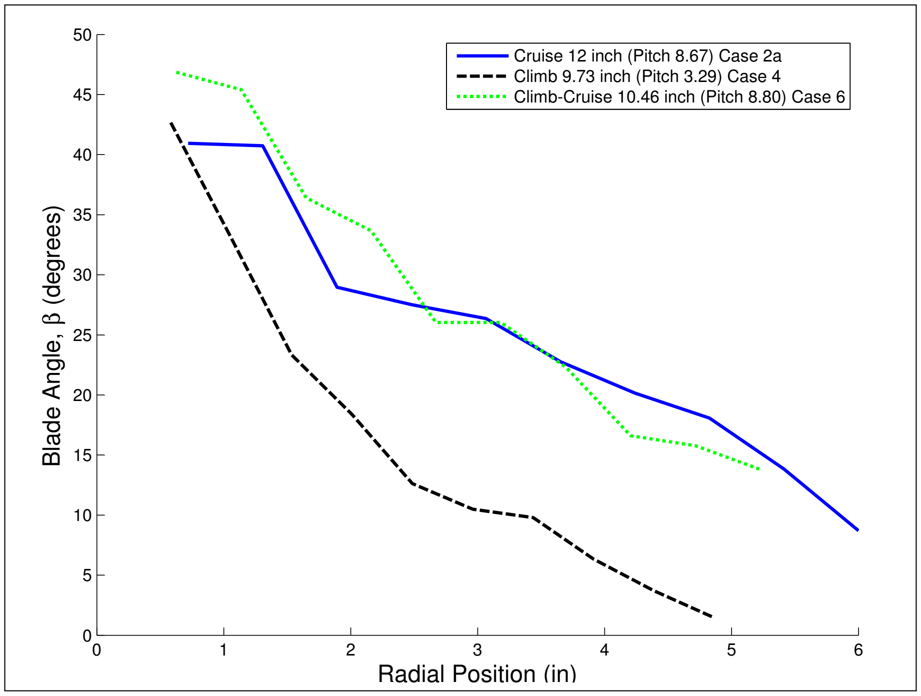

Figure 7.37: Comparison of Blade Angles, β, for the “Best” Propellers (Case 2a, Case 4, and Case 6)

Figure 7.38 shows the chord distribution for each of the “best” propellers. The Cruise propeller is shown to be the largest diameter propeller with the smallest overall chord. The climb propeller has the largest chord and the smallest diameter, and the Climb-Cruise propeller has a diameter and chord that is in between the other conditions.

The thrust for these propellers was calculated in the same manner as the previous result sections. The motor model was given 11.1 volts and a rpm was found through iteration to match propeller torque and the motor torque. This simulates a theoretical full throttle

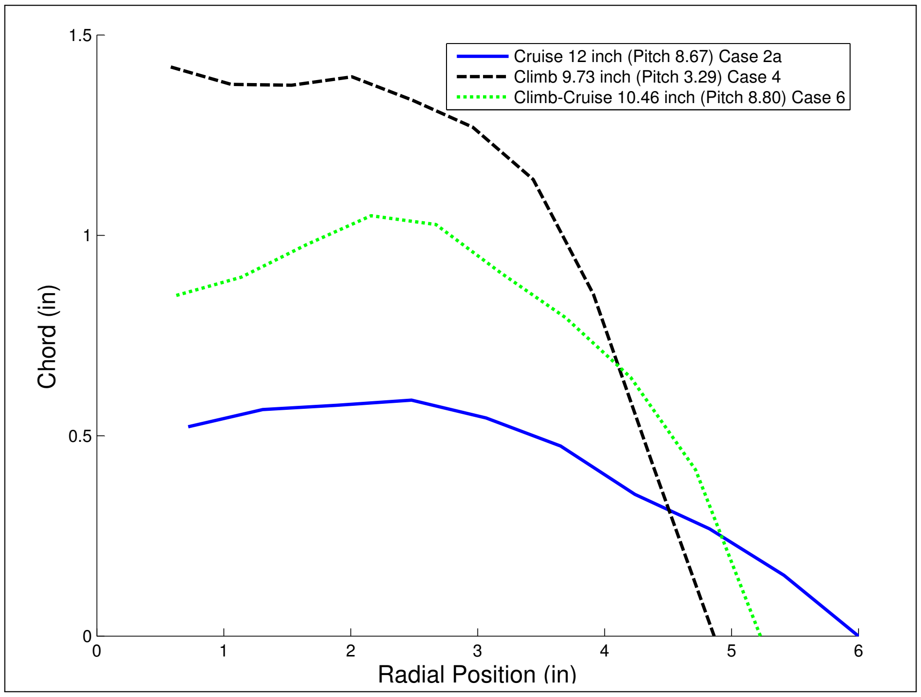

Figure 7.38: Comparison of Chords, β, for the “Best” Propellers (Case 2a, Case 4, and Case 6)

scenario. The thrust comparison for the “best” propellers can be found in Figure 7.39. The rpm of the propeller/motor is shown in a subplot on the same Figure. The Cruise and Climb-Cruise propellers are shown to produce more thrust than the climb propeller. They are also able to produce thrust longer. This is due to their higher blade angles that allow the relative angle of attack of the propeller to produce positive lift coefficients for each of the elements. The Cruise and Climb-Cruise propellers also have a much lower rotation speed at this full throttle setting due to there increased diameter.

The power requirement for the propellers is shown in Figure 7.40. Again the Cruise and Climb-Cruise propellers are shown to be similar. The climb propeller requires less power since it was optimized for this design condition. The current draw is also shown and follows the same trends as the power curves.

Once more the Cruise and Climb-Cruise propellers are shown to be very similar in the efficiency plot in Figure 7.41, but the Cruise propeller is slightly better. These propellers peak in efficiency at 90% around 70 miles per hour for the Cruise propeller and 80 miles

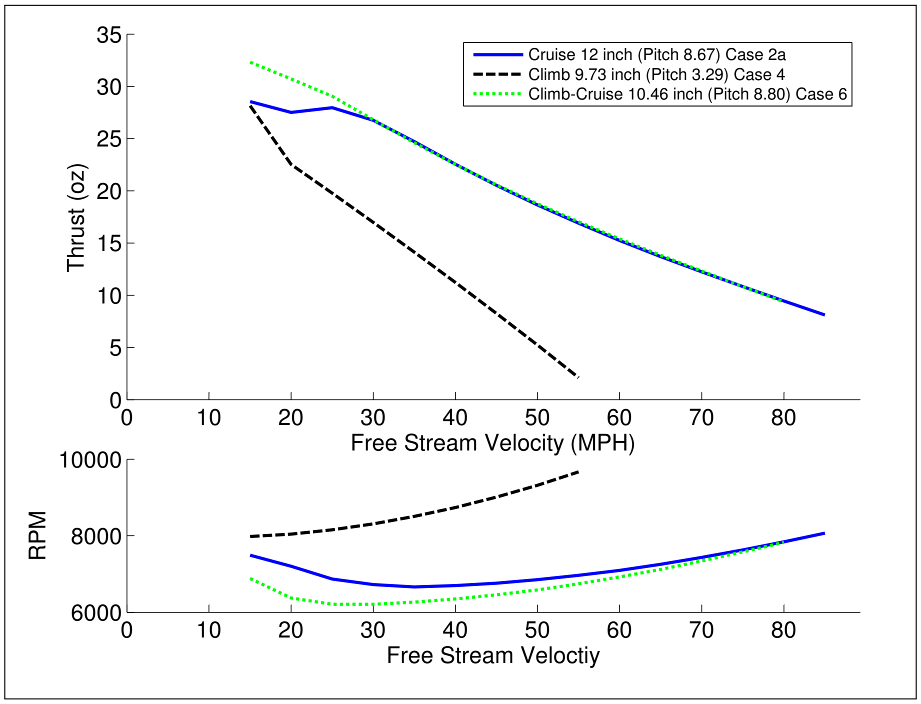

Figure 7.39: Comparison of Thrust, β, for the "Best" Propellers (Case 2a, Case 4, and Case 6)

per hour for the Climb-Cruise propeller. The Climb propeller matches the Cruise propeller's efficiency until 45 miles per hour at ≈ 81 then rapidly decreases.

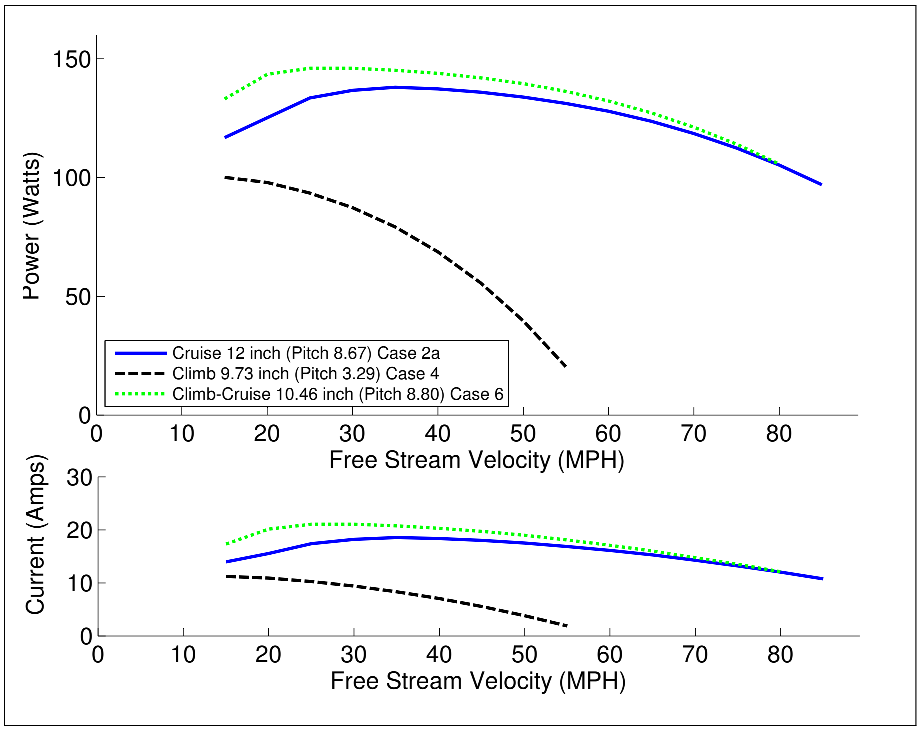

Figure 7.40: Comparison of Power, β, for the "Best" Propellers (Case 2a, Case 4, and Case 6)

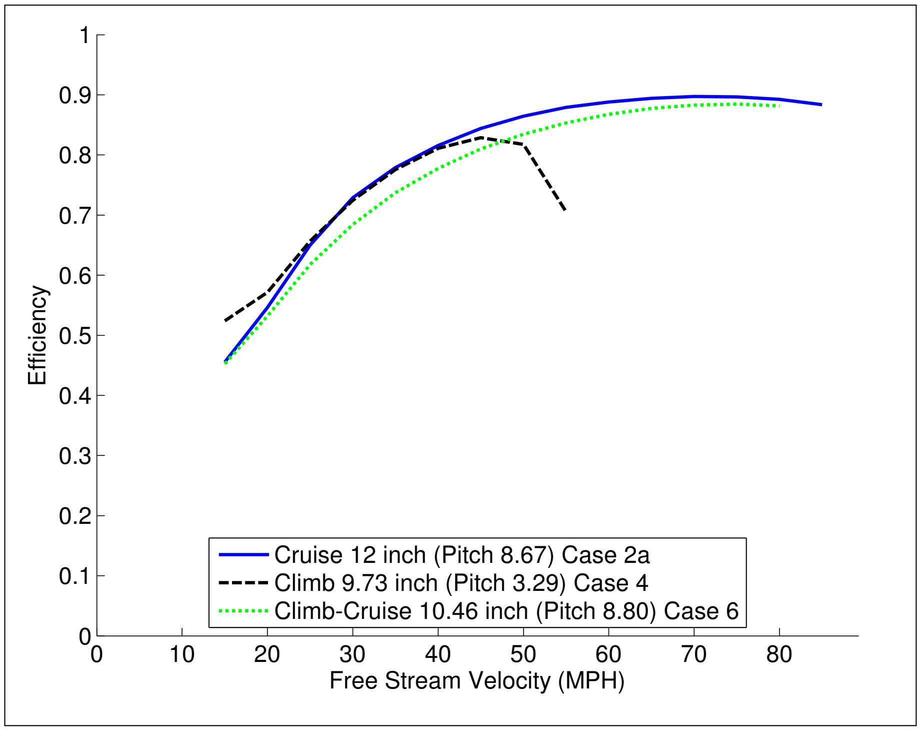

Figure 7.41: Comparison of Efficiency, β, for the "Best" Propellers (Case 2a, Case 4, and Case 6)

Chapter 8

Conclusions

A method for optimizing an electrically driven propeller for single and multiple conditions with a hybrid pattern search particle swarm optimizer was performed. 6th order Bernstein polynomials were used to parameterize airfoils that were used with Xfoil to calculate lift and drag coefficients. The hybrid optimizer was used to maximize the lift to drag ratio at zero angle of attack for an airfoil. A simplex optimizer was then used to find airfoils over a range of angle of attack from $-5^\circ - 15^\circ$. A table was generated with the optimum airfoils and was used for the optimized propellers. The propeller analysis was evaluated using a momentum blade element method with axial and rotational inflow factors.

The pattern search particle swarm optimizer produced global minimums in the large design spaces of the propellers. This optimizer required a large amount of time compared to the simplex method that was used for the airfoil optimizer beyond 0° airfoil. However the simplex was also shown to be highly dependent on initial location. The objective functions used by the optimizers strongly influence the design. The objective functions using dimensional values proved to produce better results. Several restrictions were also required on the chord and blade angle distributions which if left unrestrained would produce impractical designs.

For the Cruise condition the power was minimized. It was shown that an objective function using dimensional values verse non-dimensional values was better. The dimensional value case using the propeller power required had a 3% overall system efficiency improvement over the non-dimensional case. The system power optimization was shown to require slightly more power than the propeller power optimization. The optimized propellers were all shown to have improvements over a commercially available baseline propeller. These optimized propellers were shown to have smaller chords and approximately equal blades when compared to the baseline propeller.

For the Climb condition the thrust was maximized. The dimensional value objective function proved marginally better than the non-dimensional. The results were very similar with each case producing 28 ounces of thrust and having an overall system efficiency of 39%. Even though the Case 3 propeller could produce more thrust over a wider range of free stream velocities, Case 4 is the better propeller due to its higher efficiency at the design point. The non-dimensional objective function case did have a run time approximately 40% longer than the dimensional case. When compared to a baseline propeller both cases showed much improvement in power required. The optimized propellers had larger chords than the baseline propeller, and the blade angles were slightly smaller.

For the Climb-Cruise condition two different methods were used. The first was non-dimensional objective function maximizing the efficiencies at climb and cruise conditions. The second objection function minimized power at cruise over thrust at climb. Both methods proved to have advantages and disadvantages. The optimizer in the non-dimensional case drove thrust at climb to the minimum desire thrust, but had a higher efficiency across the range of operation compared to the dimensional objective function. The dimensional case showed to have a substantial more amount of thrust at climb, but this caused the efficiency to suffer. The optimized propellers were compared to a baseline propeller with marginally different chord and blade angle distributions. These results produced two propellers that when compared to a baseline propeller require less power, produce more thrust, and have higher efficiencies.

All of the optimized propellers proved to be more efficient than the baseline propellers. The optimized airfoils allowed the propellers to have increased performance and a larger range of operation. Cruise propellers were shown to have larger blade angles, smaller chords, and larger diameters while Climb propellers had smaller blade angles, larger chords, and smaller diameters. Climb-Cruise propellers exhibited blade angles similar to cruise propellers and diameters and chords in between the Climb and Cruise conditions.

A method for optimizing an electrical motor driven propeller has been presented and shown to be successful for single and multiple design points. The inclusion of a model for the electric motor assists the optimizer in finding the most efficient rotational speed for the propeller and eliminates the iterative process of manually matching an electric motor and a propeller. Compared to baseline propeller performance, the method produced more efficient propeller designs for single point and multipoint objective functions. As with any method, additional improvements could be made. A better method for calculating induced airflow through the rotor disk could be implemented and would provide more accurate results. Integrating an airfoil optimizer in with the propeller optimizer might provide a significant performance increase but would also dramatically increase run time. Improvements in the accuracy at off design conditions where the relative angle of attack on the propeller blades could be in a stalled region would make the analysis more robust. Other optimization techniques such as a genetic algorithm could be evaluated to minimize computational times and possibly improve results. An electric motor optimizer could be developed to investigate if any improvements over commercially available motors are possible. In addition to these method improvements, thoughts for future work include actual prototyping a propeller design optimized by this method and conducting performance tests to further verify and validate the method. Also providing an internal combustion motor model to the method could be considered. This addition would allow comparison between propellers optimized for electric and internal combustion motors and provide insight into the fundamental differences between propellers tuned to the two types of powerplants.

Bibliography

[1] Glauert, H., *The Elements of Aerofoil and Airscrew Theory*, Cambridge, 1947.

[2] Dommasch, D. O., *Element of Propeller and Helicopter Aerodynamics*, Sir Isaac Pitman & Sons, LTD, 1953.

[3] Adkins, C. N. and Liebeck, R. H., "Design of Optimum Propellers," *Journal of Propulsion and Power*, Vol. 10, No. 5, Sept.-Oct 1994.

[4] Glauert, H., *Airplane Propellers*, Springer Verlag, 1935, Aerodynamic Theory: Volume IV Chapter XI.

[5] Burger, C., *Propeller Performance Analysis and Multidisciplinary Optimization using a Genetic Algorithm*, Ph.D. thesis, Auburn University, 2007.

[6] Adkins, C. N. and Liebeck, R. H., "Design of Optimum Propellers," AIAA Paper 83-0190, AIAA 21^{st} Aerospace Sciences Meeting, 1983.

[7] Fanjoy, D. W. and Crossley, W. A., "Aerodynamic Shape Design for Rotor Airfoils via Genetic Algorithm," American Helicopter Society 53^{rd} Annual Forum, 1998.

[8] Miller, C. J., *Optimally Designed Propellers Constrained by Noise*, Ph.D. thesis, Purdue University, 1984.

[9] Kulfan, B. M., "Universal Parametric Geometry Representation Method," *Journal of Aircraft*, Vol. 45, No. 1, 2008, DOI: 10.2514/1.29958.

[10] Anderson, J. D., *Fundamental of Aerodynamics*, McGraw-Hill, 2007.

[11] Venkataraman, P., "A New Procedure for Airfoil Definition," AIAA Paper 95-1875-CP, 13^{th} Applied Aerodynamics Conference, 1995.

[12] Rogalsky, T., Kocabiyik, S., and Derksen, R. W., "Differential Evolution in Aerodynamic Optimization," *Industrial Engineering*, Vol. 46, No. 4, 2000, pp. 183–190.

[13] Kulfan, B. M. and Bussoletti, J. E., "Fundamental Parametric Geometry Representations for Aircraft Component Shapes," AIAA paper 2006-6948, 2006.

[14] *XFOIL User Guide 6.96*, MIT Aero & Astro, 2001.

[15] Drela, M., "XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils," *Low Reynolds Number Aerodynamics*, Vol. 54, Springer-Verlag, 1989.

[16] Nelson, W. C., *Airplane Propeller Principles*, John Wiley and Sons, 1944.

[17] Weick, F. E., *Aircraft Propeller Design*, McGraw-Hill, 1930.

[18] Viterna, A. and Janetzke, D., "Theoretical and Experimental Power from Large Horizontal-Axis Wind Turbines," *Proceedings from the Large Horizontal-Axis Wind Turbine Conference*, July 1981, DOE/NASA-LeRC.

[19] Kenjo, T. and Nagamori, S., *Permanent-Magnet and Brushless DC Motors*, Oxford University Press, 1985.

[20] Sokira, T. J. and Jaffe, W., *Brushless DC Motors Electronic Commutation and Controls*, TAB BOOKS, Inc., 1990.

[21] *Pittman Servo Motor Application Notes*, Pittman.

[22] Zhu, J. G. and Watterson, P., "Electromechanical Systems - Chapter 12. Brushless DC Motors," Lecture Notes.

[23] Hepperle, M., 2008, http://www.mh-aerotools.de/airfoils/cox_performance.htm.

[24] Eberhart, R. and Kennedy, J., "A New Optimizer using Particle Swarm Theory," *Proceedings Sixth Symposium on Micro Machine and Human Science*, IEEE Service Center, 1995.

[25] Hooke, R. and Jeeves, T., *Direct Search Solution of Numerical and Statistical Problems*, 1961.

[26] Jenkins, R. and Hartfield, R., "Hybrid Particle Swarm-Pattern Search Optimizer for Aerospace Propulsion Applications," AIAA Paper 2010-7078, 2010.

[27] *MATLAB User Guide R2010a*, The MathWorks Inc., 2010.

[28] J. C. Lagarias, J. A. Reeds, M. H. W. and Wright, P. E., "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions," *SIAM Journal of Optimization*, Vol. 9, No. 1, 1998, pp. 112–147.

[29] William H. Press, Saul A. Teukolsky, W. T. V. and Flannery, B. P., *Numerical Recipes in Fortarn 77*, Cambridge University Press, 1992.

[30] Tangler, J. and Kocureki, J. D., "Wind Turbine Post-Stall Airfoil Performance Characteristics Guidelines for Blade-Element Momentum Methods," National Renewable Energy Laboratory, Report NREL/CP-500-36900, October 2004.

[31] *Eflite Park 450 Brushless Outrunner Instructions*, Horizon Hobby, Inc., 2009.

[32] APCProp@AOL.com, 2012, Personal Correspondance with APC Propellers.

[33] Ananda, G. et al., "UIUC Propeller Database," 2008, http://www.ae.illinois.edu/m-selig/props/propDB.html.

Appendices

Appendix A

Xfoil Inputs and Outputs

A.1 Example XFOIL Command Inputs

The following is a list of commands that would be entered for a standard XFOIL session. More inputs can be found in the users manual [14].

plop	(plot menu)
G	(disable plotting)
	(blank return)
load	(load menu)
Point_File.dat	(file containing x-y points)
Airfoil_Name	(name of airfoil)
ppar	(paneling parameters menu)
N	(change number of panels)
140	(increase number of panels to 140)
	(blank return)
	(blank return)
oper	(direct operating points menu)
visc	(toggle to viscous mode)
150000	(enter a Reynolds number)
iter	(change the maximum iteration limit)
500	(new iteration limit)
pacc	(toggle to auto point accumulation to active polar menu)
Airfoil_data.dat	(file for polar to be saved to)
Airfoil_data_dump.dat	(file for information to be dumped to)
aseq -5 20 1	(run an angle of attack sweep in 1° degree increments from -5° to 20°
	(blank return)
quit	(exit XFOIL)

A.2 Example XFOIL Point File

The following is a example input for an airfoil which would be placed in a .dat or .txt file. The first column is the x-coordinates and the second column is the y-coordinates. The coordinates are sepearated by a space not a tab.

```
1.0000000 0.0015000
0.8998800 0.0331400
0.7998101 0.0537800
0.6997400 0.0714200
0.5997000 0.0840600
0.4996700 0.0917000
0.3996600 0.0953400
0.2996600 0.0957800
0.1996700 0.0906200
0.0997300 0.0742600
0.0498100 0.0540800
0.0248700 0.0359900
0.0000000 0.0000000
0.0250200 -.0050100
0.0500200 -.0049200
0.1000200 -.0047400
0.2000200 -.0043800
0.3000100 -.0040200
0.4000100 -.0036600
0.5000100 -.0033000
0.6000100 -.0029400
0.7000100 -.0025800
0.8000100 -.0022200
0.9000100 -.0018600
1.0000000 -.0015000
```

A.3 Example XFOIL Output File

The following is an example output file from XFOIL.

```
       XFOIL         Version 6.96

 Calculated polar for: Airfoil_name

 1 1 Reynolds number fixed          Mach number fixed

 xtrf =   1.000 (top)        1.000 (bottom)
 Mach =   0.000     Re =     0.150 e 6     Ncrit =   9.000

   alpha    CL        CD       CDp       CM     Top_Xtr  Bot_Xtr
  ------- -------- --------- --------- -------- -------- --------
  -5.000  -0.2971   0.06867   0.06517  -0.0435   0.9774   0.0423
  -4.000  -0.1719   0.05244   0.04819  -0.0617   0.9602   0.0537
  -3.000  -0.0243   0.04023   0.03482  -0.0746   0.9400   0.0797
  -1.000   0.3144   0.02246   0.01434  -0.0905   0.8947   0.0633
   0.000   0.4805   0.01534   0.00969  -0.0980   0.8748   1.0000
   1.000   0.6333   0.01355   0.00722  -0.1036   0.8420   1.0000
   2.000   0.7748   0.01237   0.00566  -0.1071   0.7816   1.0000
   3.000   0.8683   0.01245   0.00503  -0.1012   0.6140   1.0000
   4.000   0.9332   0.01526   0.00624  -0.0917   0.4074   1.0000
   5.000   1.0193   0.01757   0.00800  -0.0876   0.3374   1.0000
   6.000   1.1143   0.01973   0.01000  -0.0854   0.2994   1.0000
   7.000   1.2096   0.02195   0.01239  -0.0834   0.2708   1.0000
   8.000   1.3077   0.02469   0.01528  -0.0823   0.2469   1.0000
   9.000   1.3956   0.02764   0.01878  -0.0794   0.2243   1.0000
  10.000   1.4389   0.02869   0.02017  -0.0687   0.1916   1.0000
  11.000   1.4503   0.02956   0.02164  -0.0538   0.1456   1.0000
  12.000   1.4317   0.03907   0.03050  -0.0408   0.0300   1.0000
  13.000   1.4107   0.05045   0.04293  -0.0330   0.0256   1.0000
  14.000   1.3559   0.06895   0.06228  -0.0327   0.0235   1.0000
  15.000   1.2904   0.09461   0.08865  -0.0415   0.0229   1.0000
  16.000   1.2540   0.11515   0.10958  -0.0479   0.0219   1.0000
  17.000   1.2702   0.12359   0.11825  -0.0443   0.0204   1.0000
  18.000   1.2434   0.14474   0.14006  -0.0535   0.0200   1.0000
  19.000   1.2073   0.17196   0.16791  -0.0713   0.0204   1.0000
  20.000   0.7811   0.17806   0.17484  -0.0597   0.0285   1.0000
```

Appendix B

Derivation of Adkins-Liebeck Differential Coefficients

This appendix is for the derivation of Equations 3.47 and 3.48 which comes from Adkins and Liebeck in Reference [3]. This is to clear any questions of their derivation process and to correct an error in $\frac{dC_t}{d\xi}$ equation in Reference [3] even though the error is not found in the original paper [6].

First the momentum propeller theory and Figure 3.8 is used to find the thrust per unit radius.

$$\begin{aligned}\frac{dF_t}{dr} &= (mass\,per\,unit\,non\,dimensional\,radius)\,(velocity\,increase\,axial\,direction)\\ &= (2\pi r\rho V_0\,(1+a))\,(2V_0aF)\\ &= 4\pi r\rho V_0^2\,(1+a)\,aF \end{aligned} \tag{B.1}$$

The torque per unit radius is found using the same process.

$$\begin{aligned}\frac{dQ}{r\,dr} &= (mass\,per\,unit\,non\,dimensional\,radius)\,(velocity\,increase\,rotation\,direction)\\ &= (2\pi r\rho V_0\,(1+a))\,(4\pi nra'F)\\ &= 8\pi^2r^2\rho V_0n\,(1+a)\,a'F \end{aligned} \tag{B.2}$$

If C_y, C_x, C_l, and C_d are substituted for dF_t, dF, dL, and dD respectively in Figure 3.8. C_y and C_x and then defined by the following equations.

$$C_y = C_l cos\,(\phi) - C_d sin\,(\phi) = C_l\,(cos\,(\phi) + \epsilon\,sin\,(\phi)) \tag{B.3}$$

$$C_x = C_l sin\,(\phi) + C_d cos\,(\phi) = C_l\,(sin\,(\phi) + \epsilon\,cos\,(\phi)) \tag{B.4}$$

The blade element theory can then be used to define the thrust and torque per unit radius where b is the chord of the element.

$$\frac{dF_t}{dr} = \frac{1}{2}\rho V_{rel}^2 BbC_y \tag{B.5}$$

$$\frac{dQ}{r\,dr} = \frac{1}{2}\rho V_{rel}^2 BbC_x \tag{B.6}$$

With Equations B.1 and B.5 set equal to each other the axial interference factor, a, can be found. Using $V_{rel} = \frac{V_0(1+a)}{sin(\phi)}$ from Figure 3.8.

$$\begin{aligned}
\frac{dF_t}{dr} &= \frac{1}{2}\rho V_{rel}^2 BbC_y = 4\pi r\rho V_0^2 \left(1+a\right) aF \\
\frac{1}{2}\rho\left[\frac{V_0\left(1+a\right)}{sin\left(\phi\right)}\right]^2 BbC_y &= 4\pi r\rho V_0^2\left(1+a\right)aF \\
\frac{1}{2}\frac{V_0^2\left(1+a\right)^2}{sin^2\left(\phi\right)}BbC_y &= 4\pi rV_0^2\left(1+a\right)aF \\
a &= \frac{1}{F}\frac{Bb}{2\pi r}\frac{C_y}{4\,sin^2\left(\phi\right)}\left(1+a\right) \\
a &= \left[\frac{1}{F}\frac{Bb}{2\pi r}\frac{C_y}{4\,sin^2\left(\phi\right)}\right] + \left[\frac{1}{F}\frac{Bb}{2\pi r}\frac{C_y}{4\,sin^2\left(\phi\right)}\right]a \\
a\left[1-\frac{1}{F}\frac{Bb}{2\pi r}\frac{C_y}{4\,sin^2\left(\phi\right)}\right] &= \left[\frac{1}{F}\frac{Bb}{2\pi r}\frac{C_y}{4\,sin^2\left(\phi\right)}\right] \\
a &= \frac{\left[\frac{1}{F}\frac{Bb}{2\pi r}\frac{C_y}{4\,sin^2(\phi)}\right]}{\left[1-\frac{1}{F}\frac{Bb}{2\pi r}\frac{C_y}{4\,sin^2(\phi)}\right]}
\end{aligned}$$

where the solidity is $\sigma = \frac{Bb}{2\pi r}$ and a constant $K = \frac{C_y}{4\,sin^2(\phi)}$ a final simplified equation can be found.

$$a = \frac{\sigma K}{(F-\sigma K)} \tag{B.7}$$

The same process is repeated with Equations B.2 and B.6 to find the rotational interference factor, a', but before this is found a geometric relationship from Figure 3.8 for ϕ needs to be expressed.

$$tan\left(\phi\right) = \frac{V_0(1+a)}{2\pi nr(1-a')} \tag{B.8}$$

a' is then found as follows.

$$\begin{aligned}
\frac{dQ}{r\,dr} &= \frac{1}{2}\rho V_{rel}^2 BbC_x = 8\pi^2 r^2\rho V_0 n\left(1+a\right)a'F \\
\frac{1}{2}\rho\left[\frac{V_0(1+a)}{sin(\phi)}\right]^2 BbC_x &= 8\pi^2r^2\rho V_0 n\left(1+a\right)a'F \\
\frac{1}{2}\frac{V_0^2(1+a)^2}{sin^2(\phi)}BbC_x &= 8\pi^2r^2V_0n\left(1+a\right)a'F \\
\frac{Bb}{2\pi r}\frac{V_0(1+a)}{sin^2(\phi)}C_x &= 8\pi rna'F \\
\frac{Bb}{2\pi r}\frac{2\pi nr(1-a')tan(\phi)}{sin^2(\phi)}C_x &= 8\pi rna'F
\end{aligned}$$

$$\frac{1}{F}\frac{Bb}{2\pi r}\frac{C_x}{4cos(\phi)sin(\phi)}(1-a')=a'$$

$$a'=\left[\frac{1}{F}\frac{Bb}{2\pi r}\frac{C_x}{4cos(\phi)sin(\phi)}\right]-\left[\frac{1}{F}\frac{Bb}{2\pi r}\frac{C_x}{4cos(\phi)sin(\phi)}\right]a'$$

$$a'+\left[\frac{1}{F}\frac{Bb}{2\pi r}\frac{C_x}{4cos(\phi)sin(\phi)}\right]a'=\left[\frac{1}{F}\frac{Bb}{2\pi r}\frac{C_x}{4cos(\phi)sin(\phi)}\right]$$

$$a'=\frac{\left[\frac{1}{F}\frac{Bb}{2\pi r}\frac{C_x}{4cos(\phi)sin(\phi)}\right]}{1+\left[\frac{1}{F}\frac{Bb}{2\pi r}\frac{C_x}{4cos(\phi)sin(\phi)}\right]}$$

where $K'=\frac{C_x}{4cos(\phi)sin(\phi)}$,

$$a'=\frac{\sigma K'}{F+\sigma K'} \tag{B.9}$$

The differential forms of thrust coefficient with respect to ξ can now be found using Equation B.10, the thrust coefficient equation and Equation B.1, the thrust equation.

$$C_t=\frac{F_t}{\rho n^2D^4} \tag{B.10}$$

$$\frac{dC_t}{d\xi}=\frac{\frac{dF_t}{d\xi}}{\rho n^2D^4}$$

$$\frac{dC_t}{d\xi}=\frac{4\pi r\rho V_0^2\left(1+a\right)aFR}{\rho n^2D^4}$$

$$\frac{dC_t}{d\xi}=\frac{4\pi rV_0^2\left(1+\frac{\sigma K}{(F-\sigma K)}\right)\frac{\sigma K}{(F-\sigma K)}FR}{n^2D^4}$$

$$\frac{dC_t}{d\xi}=\frac{4\pi rRV_0^2F}{16n^2R^4}\left[\frac{\sigma K}{(F-\sigma K)}+\frac{\sigma^2K^2}{(F-\sigma K)^2}\right]$$

$$\frac{dC_t}{d\xi}=\frac{\pi\xi V_0^2F}{4n^2R^2}\left[\frac{(\sigma K)(F-\sigma K)+\sigma^2K^2}{(F-\sigma K)^2}\right]$$

$$\frac{dC_t}{d\xi}=\frac{\pi\xi V_0^2F}{4n^2R^2}\left[\frac{\sigma KF}{(F-\sigma K)^2}\right]$$

$$\frac{dC_t}{d\xi}=\frac{\pi\xi V_0^2F^2}{4n^2R^2}\left[\frac{\sigma K}{(F-\sigma K)^2}\right]$$

$$\frac{dC_t}{d\xi}=\frac{\pi\xi V_0^2F^2}{4n^2R^2}\left[\frac{\sigma}{(F-\sigma K)^2}\right]\frac{C_y}{4sin^2(\phi)}$$

Equation B.8 must be modified now before the derivation can continue.

$$\begin{aligned}
tan\left(\phi\right) &= \frac{V_0(1+a)}{2\pi nr(1-a')} \\
tan\left(\phi\right) &= \frac{V_0}{2\pi nr}\frac{\left(1+\frac{\sigma K}{(F-\sigma K)}\right)}{\left(1-\frac{\sigma K'}{F+\sigma K'}\right)} \\
tan\left(\phi\right) &= \frac{V_0}{2\pi nr}\frac{\frac{F-\sigma K+\sigma K}{F-\sigma K}}{\frac{F+\sigma K'-\sigma K'}{F+\sigma K'}} \\
tan\left(\phi\right) &= \frac{V_0}{2\pi nr}\frac{F\left(F+\sigma K'\right)}{\left(F-\sigma K\right)F} \\
\frac{sin\left(\phi\right)}{cos\left(\phi\right)} &= \frac{V_0}{2\pi nr}\frac{\left(F+\sigma K'\right)}{\left(F-\sigma K\right)} \\
\left(F-\sigma K\right)sin\left(\phi\right) &= \frac{V_0}{2\pi nr}\left(F+\sigma K'\right)cos\left(\phi\right) \qquad \text{(B.11)}
\end{aligned}$$

Equation B.11 can now be substituted into the derivation.

$$\begin{aligned}
\frac{dC_t}{d\xi} &= \frac{\pi}{16}\frac{\sigma C_y V_0^2 \xi F^3}{n^2R^2}\frac{1}{\left(F-\sigma K\right)^2 sin^2\left(\phi\right)} \\
\frac{dC_t}{d\xi} &= \frac{\pi}{16}\frac{\sigma C_y V_0^2 \xi F^3}{n^2R^2}\frac{1}{\frac{V_0^2}{4\pi^2n^2r^2}\left(F+\sigma K'\right)^2 cos^2\left(\phi\right)} \\
\frac{dC_t}{d\xi} &= \frac{\pi^3}{4}\sigma\xi^3F^2\frac{C_y}{\left[\left(F+\sigma K'\right)cos\left(\phi\right)\right]^2} \qquad \text{(B.12)}
\end{aligned}$$

The differential form of torque with respect to ξ follows the same procedure as the previous with using B.13 and B.2.

$$C_p = \frac{P}{\rho n^3 D^5} \qquad \text{(B.13)}$$

$$\begin{aligned}
\frac{dC_p}{d\xi} = \frac{\frac{dP}{d\xi}}{\rho n^3D^5} = \frac{\frac{dQ}{d\xi}2\pi n}{\rho n^3D^5} &= \frac{\left(2\pi n\right)\left(8\pi^2r^3\rho V_0 n\left(1+a\right)a'F\right)R}{\rho n^3D^5} \\
\frac{dC_p}{d\xi} &= \frac{16n^2\pi^3r^3V_0\left(1+a\right)a'FR}{n^3D^5} \\
\frac{dC_p}{d\xi} &= \frac{16\pi^3r^3V_0\left(1+a\right)a'FR}{32nR^5} \\
\frac{dC_p}{d\xi} &= \frac{\pi^3\xi^3V_0Fa'\left(1+a\right)}{2nR} \\
\frac{dC_p}{d\xi} &= \frac{\pi^3\xi^3V_0F}{2nR}\left[\frac{\sigma K'}{F+\sigma K'}\left(1+\frac{\sigma K}{(F-\sigma K)}\right)\right]
\end{aligned}$$

$$
\begin{aligned}
\frac{dC_p}{d\xi} &= \frac{\pi^3\xi^3V_0F}{2nR}\left[\frac{\sigma K'}{F+\sigma K'}+\frac{\sigma K'}{(F+\sigma K')}\frac{\sigma K}{(F-\sigma K)}\right]\\
\frac{dC_p}{d\xi} &= \frac{\pi^3\xi^3V_0F}{2nR}\left[\frac{(\sigma K')(F-\sigma K)+\sigma^2K'K}{(F+\sigma K')(F-\sigma K)}\right]\\
\frac{dC_p}{d\xi} &= \frac{\pi^3\xi^3V_0F}{2nR}\left[\frac{\sigma FK'}{(F+\sigma K')(F-\sigma K)}\right]\\
\frac{dC_p}{d\xi} &= \frac{\pi^3\xi^3V_0F}{2nR}\frac{\sigma F}{(F+\sigma K')(F-\sigma K)}\frac{C_x}{4cos(\phi)sin(\phi)}
\end{aligned}
$$

Using the relationship found in Equation B.11,

$$
\begin{aligned}
\frac{dC_p}{d\xi} &= \frac{\pi^3\xi^3V_0F}{2nR}\frac{\sigma F}{4cos(\phi)(F+\sigma K')}\frac{C_x}{\frac{V_0}{2\pi nr}(F+\sigma K')cos(\phi)}\\
\frac{dC_p}{d\xi} &= \frac{\pi^3}{4}\xi^3F^2\sigma\frac{C_y}{[(F+\sigma K')cos(\phi)]^2}\pi\xi\frac{C_x}{C_y}\\
\frac{dC_p}{d\xi} &= \frac{dC_t}{d\xi}\pi\xi\frac{C_x}{C_y} \qquad \text{(B.14)}
\end{aligned}
$$

Appendix C

Airfoil Optimization Codes

C.1 *AirfoilMaker()*

```
function [cl cd angles cls cds ] = AirfoilMaker(alpha,Atop,Abottom)

Order = 6;
Trailz = 0.001;
numPoints = 50;
N1 = 0.5;
N2 = 1.0;

%Figure Out Airfoil file name (AirfoilID)
temp='';
for n=1:length(Atop)
    if round(Atop(n)*100) < 10 && round(Atop(n)*100) >= 0.0
        temp = strcat(temp,'0',num2str(round(Atop(n)*100)));
    else
        temp = strcat(temp,num2str(round(Atop(n)*100)));
    end
end
for n=1:length(Abottom)
    if round(Abottom(n)*100) < 10 && round(Abottom(n)*100) >= 0.0
        temp = strcat(temp,'0',num2str(round(Abottom(n)*100)));
    elseif round(Abottom(n)*100) < 0.0...
            && round(Abottom(n)*100) > -10.0
        temp = strcat(temp,'_0',num2str(-1*round(Abottom(n)*100)));
    elseif round(Abottom(n)*100) <=-10
        temp = strcat(temp,'_',num2str(-1*round(Abottom(n)*100)));
    else
        temp = strcat(temp,num2str(round(Abottom(n)*100)));
    end
end
AirfoilID = temp;

%Get the points for the Airfoil
[x,y,isGood] =...
    ParametricAirfoil(Atop,Abottom,Order,Trailz,numPoints,N1,N2);
```

```
%Get the lift and drag coeff
if isGood == 1
    [cl,cd,angles,cls,cds] = ClCdFinder(x,y,alpha,AirfoilID);
else
    cl = -10;
    cd = 100;
    angles = linspace(0,15,15);
    cls = -1*linspace(15,20,15);
    cds = 100*linspace(15,20,15);
end
```

C.2 *ParametricAirfoil()*

```
function [x,y,isGood] = ...
    ParametricAirfoil(Atop,Abottom,Order,Trailz,numPoints,N1,N2)

PolyOrder = 6;
xoc = linspace(0,1,numPoints);

%find the K_i values
for n=0:Order
    Ki(n+1) = factorial(Order)/(factorial(n)*factorial(Order-n));
end

%find class and shape function values. Finished with the z/c values
for n=1:length(xoc)
    C(n) = xoc(n)^N1*(1-xoc(n))^N2;
    if xoc(n) == 0.0
        Stop(n) = 0.0;
        Sbottom(n) = 0.0;
    elseif xoc(n) == 1.0
        Stop(n) = 0.0;
        Sbottom(n) = 0.0;
    else
        Stop(n) = 0.0;
        Sbottom(n) = 0.0;
        for m=0:Order
            Stop(n) = Stop(n) +...
                Atop(m+1)*Ki(m+1)*xoc(n)^m*(1-xoc(n))^(Order-m);
            Sbottom(n) = Sbottom(n) +...
                Abottom(m+1)*Ki(m+1)*xoc(n)^m*(1-xoc(n))^(Order-m);
        end
    end
    zocTop(n)=C(n)*Stop(n)+xoc(n)*Trailz;
    zocBottom(n)=C(n)*Sbottom(n)-xoc(n)*Trailz;
end

%Check to see if the bottom curve
%intersects or passes the upper curve
check = 0;
for n=1:length(zocTop)
    if zocTop(n) - zocBottom(n) < 0.0
        check = check + 1;
    end
end

if check == 0
    isGood = 1;
else
    isGood = 0;
end
%End of intersection check

%combine upper and lower surface
%Atop == Abottom
for n=1:length(xoc)
    x(n) = xoc(length(xoc)-n+1);
    y(n) = zocTop(length(xoc)-n+1);
    if n>1
        x(n+length(xoc)-1) = xoc(n);
        y(n+length(xoc)-1) = -zocTop(n);
    end
end

end
```

C.3 *ClCdFinder()*

```
function [cl,cd,angles,cls,cds] = ClCdFinder(x,y,alpha,AirfoilID)
%This function takes a set of points for an airfoil and
%runs xfoil.exe with the points and a set alpha. It then
%returns the lift and drag coefficients

warning off all
%User Defined Settings
```

```
DirectoryName = 'OptimumAirfoils';
N = 120;    %number of segments for xfoil
AoAstart = 0;
AoAend = 15;
AoAstep = 0.5;
isGood = 1;
%Check for Main Directory
CheckMDir = isdir(DirectoryName);
if CheckMDir == 0
    mkdir(DirectoryName);
    DoesFileExist = 0;
else %Check for Airfoil
    fidTemp = fopen(strcat(DirectoryName,'\',AirfoilID,'.dat'),'r');
    if fidTemp ~= -1
        fclose(fidTemp);
        DoesFileExist = 1;
    else
        DoesFileExist = 0;
    end
end

%-----------------------------------------------
%if the AirfoilID file does not exist it will be created
if DoesFileExist == 0
    stop = 1;
    while stop == 1
        delete('points.dat')
        delete('dump.dat')
        delete('BatchInstr.inp')
        %Create Points File
        fid = fopen('points.dat','w');
        for m=1:length(x)
            fprintf(fid,'%8.4f %8.4f\r\n',x(m),y(m));
        end
        fclose(fid);
        %Create Instruction File for Batch File
        fid = fopen('BatchInstr.inp','w');
        fprintf(fid,'plop\r\nG \r\n \r\n');
        fprintf(fid,'load\r\n');
        fprintf(fid,'points.dat\r\n%s\r\n',AirfoilID);
        fprintf(fid,'ppar\r\nN\r\n%i\r\n \r\n \r\n',N);
        fprintf(fid,'oper\r\n');
        fprintf(fid,'visc\r\n150000\r\n');
        fprintf(fid,'iter\r\n400\r\n');
        fprintf(fid,'pacc\r\n%s\\%s.dat\r\ndump.dat\r\n',...
            DirectoryName, AirfoilID);
        fprintf(fid,'aseq %4.2f %4.2f %4.2f\r\n',...
            AoAstart,AoAend,AoAstep);
        fprintf(fid,'\r\nquit\r\n');
        fclose(fid);
        %Run Batch File and delete some unwanted folders
        [sh,wow]=dos('xfoiltablefilecreate.bat');
        clear sh
        clear wow
        %Check for NaN
        filenameNaN = strcat(DirectoryName,'\',AirfoilID,'.dat');
        fidNaN = fopen(filenameNaN,'r');
        if fidNaN == -1
            Nplus = 1;
        else
            currentLine = 1;
            c=1;
            Nplus = 0;
            while ~feof(fidNaN)
                temp = fgetl(fidNaN);
                Angles = 0.0;
                if currentLine >= 13
                    airfoilData = str2num(temp);
                    Angles = airfoilData(1);
                    Cl = airfoilData(2);
                    Cd = airfoilData(3);
                    if Nplus == 0
                        if isnan(Cl) || isnan(Cd)
                            N=N+10;
                            Nplus = 1;
                        end
                    end
                end
                currentLine = currentLine+1;
            end
            if Nplus == 0
                if Angles <= 0.6667*AoAend
                    N=N+10;
                    Nplus = 1;
                end
            end
            fclose(fidNaN);
            if Nplus ~= 1 || N==170;
                if N>165 && Nplus==1
                    isGood = 0;
                    end
                    stop = 0;
                else
                    delete(filenameNaN)
                end
            end
        end
        delete('points.dat')
        delete('dump.dat')
        delete('BatchInstr.inp')
end

%----------------------------------------------------
%Open file and get the lift and drag curves
if isGood == 1
    fid = fopen(strcat(DirectoryName,'\',AirfoilID,'.dat'),'r');
    currentLine = 1;
    n=1;
    while ~feof(fid)
        temp = fgetl(fid);
        if currentLine >= 13
            airfoilData = str2num(temp);
            angles(n) = airfoilData(1);
            cls(n) = airfoilData(2);
            cds(n) = airfoilData(3);
            n=n+1;
        end
        currentLine=currentLine+1;
    end
else
    fid = fopen(strcat(DirectoryName,'\',AirfoilID,'.dat'),'w');
    for n=1:40
        fprintf(fid,'%i -10000 10000\r\n',n);
    end
    fclose(fid);
    cl = -10;
    cd = 100;
    angles = linspace(0,15,15);
    cls = -1*linspace(15,20,15);
    cds = 100*linspace(15,20,15);
end
%Find cl and cd at desired alpha using interpolation
if isGood == 1
    AoA = alpha;
    %Begin Table Lookup
```

```
stopLookup = 0;
nn=1;
while stopLookup == 0
    if AoA < 0
        stopZero = 0;
        m=1;
        while stopZero == 0
            if angles(m) >= 0
                ZeroLoc = m;
                stopZero = 1;
            end
            m=m+1;
        end
        shift = 10;
        cl = cls(ZeroLoc+shift) - ...
            (angles(ZeroLoc+shift)-AoA)*(cls(ZeroLoc+shift)-...
            cls(ZeroLoc))/(angles(ZeroLoc+shift)-...
            angles(ZeroLoc));
        cd = cds(ZeroLoc+shift) -...
            (angles(ZeroLoc+shift)-AoA)*(cds(ZeroLoc+shift)-...
            cds(ZeroLoc))/(angles(ZeroLoc+shift)-...
            angles(ZeroLoc));
        stopLookup = 1;
    elseif AoA == angles(nn)
        cl = cls(nn);
        cd = cds(nn);
        stopLookup = 1;
    elseif AoA < angles(nn+1) && AoA > angles(nn)
        cl = cls(nn+1) - (angles(nn+1)-AoA)*...
            (cls(nn+1)-cls(nn))/(angles(nn+1)-angles(nn));
        cd = cds(nn+1) - (angles(nn+1)-AoA)*...
            (cds(nn+1)-cds(nn))/(angles(nn+1)-angles(nn));
        stopLookup = 1;
    elseif AoA > angles(length(angles))
        cl = cls(length(angles)) - ...
            (angles(length(angles))-AoA)*...
            (cls(length(angles))-cls(length(angles)-1))...
            /(angles(length(angles))-...
            angles(length(angles)-1));
        cd = cds(length(angles)) - ...
            (angles(length(angles))-AoA)*...
            (cds(length(angles))-cds(length(angles)-1))/...
            (angles(length(angles))-...
            angles(length(angles)-1));
        stopLookup = 1;
        elseif AoA < angles(1)
            cl = cls(2) - (angles(2)-AoA)*...
                (cls(2)-cls(1))/(angles(2)-angles(1));
            cd = cds(2) - (angles(2)-AoA)*...
                (cds(2)-cds(1))/(angles(2)-angles(1));
            stopLookup = 1;
        end
        nn=nn+1;
    end
    %End Table Lookup
end

fclose('all');
```

Appendix D

Propeller Optimization Codes

D.1 *BrushlessMotor()*

```
function [Output] = ...
    BrushlessMotorV2(Input,Kv,Internal_Resistance,...
    numPoles,numPhases,IdleCurrent,WhichCase)

%Cases
%WhichCase = 1: INPUTS:[Voltage, Current] OUTPUTS:[Torque, RPM, Eta]
%WhichCase = 2: INPUTS:[RPM, Torque] OUTPUTS:[Voltage, Current, Eta]

%Constants
Kt = 1000/Kv*1.345;
VoltageStep = 0.01;
RPMtol = 0.1;               %+,- rpm tolerance range

%Solve for Unknowns
if WhichCase == 1       %Voltage and Current are inputs
    Voltage = Input(1);
    Current = Input(2);
    %find RPM
    Max_RPM = Voltage*Kv;
    lambda = 2/(numPoles*Max_RPM)*...
        (Voltage-Internal_Resistance*IdleCurrent);
    Tem = (numPhases*numPoles)/2*lambda*Current;
    RPM = (Voltage/(numPoles*lambda/2)-...
        Internal_Resistance/(numPhases*(numPoles*lambda/2)^2)*Tem);

    %find Torque
    Torque = Kt*Current;

    %Find Efficiency
    Power_In = Voltage*Current;
    Power_Out =(Voltage-Current*...
        Internal_Resistance)*(Current-IdleCurrent);
    Eta = Power_Out/Power_In;
    %Set Outputs
    Output(1) = Torque;
    Output(2) = RPM;
    Output(3) = Eta;

elseif WhichCase == 2   %RPM and Torque are inputs
    RPM = Input(1);
    Torque = Input(2);
    %Find the Current for the Motor given the Torque
    Current = Torque/Kt;

    %Find the Voltage required for the given Torque and RPM
    Voltage = 1.0;
    stop = 0;
    while stop == 0
        Max_RPM = Voltage*Kv;
        lambda = 2/(numPoles*Max_RPM)*...
            (Voltage-Internal_Resistance*IdleCurrent);
        Tem = (numPhases*numPoles)/2*lambda*Current;
        RPMVolt = (Voltage/(numPoles*lambda/2)-...
            Internal_Resistance/(numPhases*...
            (numPoles*lambda/2)^2)*Tem);
        if abs(RPM - RPMVolt) < RPMtol || RPMVolt > RPM
            stop = 1;
        else
            Voltage = Voltage+VoltageStep;
        end
    end
    %Solve for Efficiency
    Power_In = Voltage*Current;
    Power_Out =(Voltage-Current*Internal_Resistance)...
        *(Current-IdleCurrent);
    Eta = Power_Out/Power_In;

    %Set Outputs
    Output(1) = Voltage;
    Output(2) = Current;
    Output(3) = Eta;

end
```

D.2 ***PropellerPerformance()***

```
function [Ct,Cp,Eta,J] = PropellerPerformance(Beta,...
    Diameter,Chord,Position,Props)

%This is 'The' function for calcuating performance
%parameters for propellers. It uses the Adkins/Glauert
%method to find thrust/power/speed curves

load AirfoilIDlist.mat

RPM = Props(1);
FreeStreamMPH = Props(2);
numBlades = Props(3);
Position = Position*Diameter/2;
MaxIter = 150;

%----------Units-----------
% Beta = Radians
% RPM = Revolutions per Minute
% Diameter = inches
% Chord = inches
% Position = inches
% FreeStreampMPH = Miles per Hour
% numBlades = non-dim
% rho = Slugs per cubic foot
% OutputData = non-dim


%constants
damp = 0.5;     %this is the damping coefficient for convergence
J = (FreeStreamMPH*5280/3600)/((RPM/60)*(Diameter/12));
numSections = length(Position);

%------------------------------------------------------------------
%--------------------------------MAIN------------------------------
%------------------------------------------------------------------

%-----Initialize Arrays----------
alpha = zeros(numSections);
cl = zeros(numSections);
cd = zeros(numSections);
a = zeros(numSections);
a_prime = zeros(numSections);
phi_new = zeros(numSections,1);
cdcl = zeros(numSections);
Cy = zeros(numSections);
```

```
K = zeros(numSections);
Cx = zeros(numSections);
Kprime = zeros(numSections);
sigma = zeros(numSections);
xi = zeros(numSections);
phi_t = zeros(numSections);
f = zeros(numSections);
F = zeros(numSections);
Ct_prime = zeros(numSections);
Cp_prime = zeros(numSections);
W = zeros(numSections);

%------Solve for initial inflow angles---------
for n=1:numSections
    initial_phi(n,1) = atan((FreeStreamMPH*5280/3600)/...
        (2*pi*Position(n)/12*RPM/60));
    sigma(n) = (numBlades*Chord(n)/12)/(2*pi*Position(n)/12);
    xi(n) = Position(n)/(Diameter/2);
end

%-----Run the Interation Part of the Code---------
%Adkins/Glauert Method
phi = initial_phi;
stop = 0;
numIt = 1;
angles = linspace(-5,15,21);
% angles = linspace(0,15,16);
while stop == 0;
    %find alpha/cd/cl/Cy/Cx/everything
    for n=1:numSections
        alpha(n) = real(Beta(n) - phi(n));
        if n == numSections
            alpha(n) = 0.0;
            cl(n) = cl(n-1);
            cd(n) = cd(n-1);
        else
            if alpha(n) >= 19*pi/180    %Flat plate Theory
                cl(n) = 2*sin(alpha(n))*cos(alpha(n));
                cd(n) = 2*(sin(alpha(n)))^2;
            else
                AoA = round(alpha(n)*180/pi);
                if AoA <= -5
                    idNum = 1;
                elseif AoA >= 15
                    idNum = 21;
            else
                stopAoA = 0;
                c=1;
                while stopAoA == 0
                    if AoA<= angles(c+1) && AoA>angles(c)
                        idNum = c;
                        stopAoA=1;
                    end
                    c=c+1;
                end
            end
            [cl(n) cd(n)] = ClCdFinderAirfoilID(...
                num2str(AirfoilIDlist(idNum,:)),alpha(n)...
                *180/pi);
        end
    end
    cdcl(n) = cd(n)/cl(n);
    Cy(n) = cl(n)*(cos(phi(n))-cdcl(n)*sin(phi(n)));
    K(n) = Cy(n)/(4*sin(phi(n))*sin(phi(n)));
    Cx(n) = cl(n)*(sin(phi(n))+cdcl(n)*cos(phi(n)));
    Kprime(n) = Cx(n)/(4*cos(phi(n))*sin(phi(n)));
    phi_t(n) = atan(xi(n)*tan(phi(n)));
    f(n) = (numBlades/2)*(1-xi(n))/sin(phi_t(n));
    F(n) = (2/pi)*acos(exp(-1*f(n)));
    a(n) = (sigma(n)*K(n))/(F(n)-sigma(n)*K(n));
    if isnan(a(n))
        a(n) = 0.0;
    elseif a(n) > 0.7
        a(n) = 0.7;
    elseif a(n) < -1.0
        a(n) = 0.7;
    end
    a_prime(n) = (sigma(n)*Kprime(n))/...
        (F(n)+sigma(n)*Kprime(n));
    if isnan(a_prime(n))
        a_prime(n) = 0.0;
    elseif a_prime(n)>0.7
        a_prime(n) = 0.7;
    end
    phi_new(n,1) = atan2(((FreeStreamMPH*5280/3600)*...
        (1+a(n)))/(RPM/60*2*pi*Position(n)/...
        12*(1-a_prime(n))),1);
    if isnan(phi_new(n))
        phi_new(n,1) = 0.0;
    elseif n == numSections
```

```
            phi_new(n,1) = phi_new(n-1,1);
        end
    end
    if (abs(phi_new - phi) <= 10^-4)
        stop = 1;
    else
        phi = phi*damp +(1-damp)*phi_new;
    end
    if (numIt >= MaxIter)
        stop = 1;
    end
    numIt = numIt+1;
end
%-----------Finished with Interating--------------

%Find Ct,Cp,Eta
for n=1:numSections
   W(n) = (FreeStreamMPH*5280/3600)*(1+a(n))/sin(phi(n));
    if isnan(W(n))
       W(n) = W(n-1);
    end
    Ct_prime(n) = (pi^3/4)*sigma(n)*Cy(n)*xi(n)^3*...
        F(n)^2/((F(n)+sigma(n)*Kprime(n))*cos(phi(n)))^2;
    if isnan(Ct_prime(n))
        Ct_prime(n) = 0.0;
    end
    Cp_prime(n) = Ct_prime(n)*pi*xi(n)*Cx(n)/Cy(n);
    if isnan(Cp_prime(n))
        Cp_prime(n) = 0.0;
    end
end

Ct = 0;
Cp = 0;
%sum up Thrust and Torque
for n=1:numSections
    if n==1
        Ct=Ct+Ct_prime(n)/2*xi(n);
        Cp=Cp+Cp_prime(n)/2*xi(n);
    else
        Ct = Ct + (Ct_prime(n)+Ct_prime(n-1))/2*(xi(n)-xi(n-1));
        Cp = Cp + (Cp_prime(n)+Cp_prime(n-1))/2*(xi(n)-xi(n-1));
    end
end
```

```
Eta = Ct*J/Cp;
```

D.3 *ClCdFinderAirfoilID()*

```
function [cl,cd] = ClCdFinderAirfoilID(AirfoilID ,alpha)

DirectoryName = 'OptimumAirfoils';

fid = fopen(strcat(DirectoryName,'\',AirfoilID ,'.dat'),'r');
currentLine = 1;
n=1;
while ~feof(fid)
    temp = fgetl(fid);
    if currentLine >= 13
        airfoilData = str2num(temp);
        angles(n) = airfoilData(1);
        cls(n) = airfoilData(2);
        cds(n) = airfoilData(3);
        n=n+1;
    end
    currentLine=currentLine+1;
end

fclose(fid);

AoA = alpha;
%Begin Table Lookup
stopLookup = 0;
nn=1;
while stopLookup == 0
    if AoA < 0
        stopZero = 0;
        m=1;
        while stopZero == 0
            if angles(m) >= 0
                ZeroLoc = m;
                stopZero = 1;
            end
            m=m+1;
        end
        shift = 10;
        cl = cls(ZeroLoc+shift) - (angles(ZeroLoc+shift)-AoA)...
            *(cls(ZeroLoc+shift)-cls(ZeroLoc))/...
            (angles(ZeroLoc+shift)-angles(ZeroLoc));
        cd = cds(ZeroLoc+shift) - (angles(ZeroLoc+shift)-AoA)...
            *(cds(ZeroLoc+shift)-cds(ZeroLoc))/...
            (angles(ZeroLoc+shift)-angles(ZeroLoc));
        stopLookup = 1;
    elseif AoA == angles(nn)
        cl = cls(nn);
        cd = cds(nn);
        stopLookup = 1;
    elseif AoA < angles(nn+1) && AoA > angles(nn)
        cl = cls(nn+1) - (angles(nn+1)-AoA)*...
            (cls(nn+1)-cls(nn))/(angles(nn+1)-angles(nn));
        cd = cds(nn+1) - (angles(nn+1)-AoA)*...
            (cds(nn+1)-cds(nn))/(angles(nn+1)-angles(nn));
        stopLookup = 1;
    elseif AoA > angles(length(angles))
        cl = cls(length(angles)) - (angles(length(angles))-...
            AoA)*(cls(length(angles))-cls(length(angles)-1))/...
            (angles(length(angles))-angles(length(angles)-1));
        cd = cds(length(angles)) - (angles(length(angles))-...
            AoA)*(cds(length(angles))-cds(length(angles)-1))/...
            (angles(length(angles))-angles(length(angles)-1));
        stopLookup = 1;
    elseif AoA < angles(1)
        cl = cls(2) - (angles(2)-AoA)*(cls(2)-cls(1))/...
            (angles(2)-angles(1));
        cd = cds(2) - (angles(2)-AoA)*(cds(2)-cds(1))/...
            (angles(2)-angles(1));
        stopLookup = 1;
    end
    nn=nn+1;
end
%End Table Lookup
```

Appendix E

Optimized Airfoil Data

Table E.1: Coefficients for Bernstein Polynomial for of Upper Surface of Optimized Airfoils

Angle (°)	A_{upper_0}	A_{upper_1}	A_{upper_2}	A_{upper_3}	A_{upper_4}	A_{upper_5}	A_{upper_6}
-5	0.1507	0.3571	0.2964	0.1991	0.3068	0.3724	0.1995
-4	0.1507	0.3571	0.2964	0.1991	0.3068	0.3724	0.1995
-3	0.1507	0.3571	0.2964	0.1991	0.3068	0.3724	0.1995
-2	0.1507	0.3571	0.2964	0.1991	0.3068	0.3724	0.1995
-1	0.1507	0.3571	0.2964	0.1991	0.3068	0.3724	0.1995
0	0.1507	0.3571	0.2964	0.1991	0.3068	0.3724	0.1995
1	0.1484	0.3410	0.3030	0.2072	0.3134	0.3827	0.2060
2	0.1532	0.3209	0.2909	0.2036	0.3152	0.3787	0.1995
3	0.1505	0.3043	0.2892	0.2008	0.3207	0.3847	0.1994
4	0.1517	0.2783	0.2942	0.2085	0.3084	0.3876	0.1992
5	0.1579	0.2671	0.2583	0.1967	0.3297	0.3845	0.2032
6	0.1558	0.2443	0.2630	0.2046	0.3419	0.3804	0.2000
7	0.1790	0.2279	0.2994	0.2307	0.3354	0.2938	0.1671
8	0.1999	0.2495	0.3054	0.2365	0.3235	0.2938	0.1550
9	0.2234	0.2509	0.3607	0.2605	0.2886	0.2383	0.1525
10	0.2373	0.2707	0.3762	0.2573	0.2763	0.2354	0.1551
11	0.2564	0.2855	0.3963	0.2463	0.2657	0.2461	0.1572
12	0.2780	0.3070	0.4063	0.2324	0.2557	0.2499	0.1613
13	0.3016	0.3325	0.3953	0.2201	0.2497	0.2645	0.1683
14	0.3181	0.3450	0.3522	0.2162	0.2496	0.2739	0.1719
15	0.3181	0.3450	0.3522	0.2162	0.2496	0.2739	0.1719

Table E.2: Coefficients for Bernstein Polynomial for of Lower Surface of Optimized Airfoils

Angle (°)	A_{lower_0}	A_{lower_1}	A_{lower_2}	A_{lower_3}	A_{lower_4}	A_{lower_5}	A_{lower_6}
-5	-0.0524	0.0904	0.0602	0.2000	0.1988	0.2100	0.1865
-4	-0.0524	0.0904	0.0602	0.2000	0.1988	0.2100	0.1865
-3	-0.0524	0.0904	0.0602	0.2000	0.1988	0.2100	0.1865
-2	-0.0524	0.0904	0.0602	0.2000	0.1988	0.2100	0.1865
-1	-0.0524	0.0904	0.0602	0.2000	0.1988	0.2000	0.1865
0	-0.0524	0.0904	0.0602	0.2000	0.1988	0.2000	0.1776
1	-0.0509	0.0907	0.0599	0.1918	0.1945	0.2038	0.1829
2	-0.0515	0.0910	0.0599	0.2032	0.2011	0.2040	0.1852
3	-0.0531	0.0954	0.0606	0.2006	0.2030	0.2016	0.1781
4	-0.0518	0.0915	0.0598	0.2049	0.2019	0.2066	0.1927
5	-0.0520	0.0948	0.0609	0.1953	0.2099	0.2209	0.1873
6	-0.0522	0.0960	0.0607	0.1983	0.2078	0.2275	0.1911
7	-0.0487	0.1015	0.0650	0.2017	0.2073	0.2285	0.1777
8	-0.0479	0.1022	0.0671	0.1943	0.2065	0.2177	0.1715
9	-0.0485	0.1033	0.0656	0.1978	0.2010	0.2085	0.1744
10	-0.0481	0.1042	0.0655	0.1966	0.1999	0.2046	0.1655
11	-0.0482	0.0921	0.0655	0.2058	0.1961	0.1996	0.1624
12	-0.0490	0.0916	0.0647	0.1889	0.1943	0.2012	0.1578
13	-0.0501	0.0876	0.0611	0.1796	0.1943	0.1935	0.1579
14	-0.0505	0.0887	0.0615	0.1782	0.1994	0.1836	0.1644
15	-0.0505	0.0887	0.0615	0.1782	0.2069	0.1859	0.1644

Table E.3: Lift Coefficients, and Drag to Lift Ratios for Optimized Airfoils

Angles ($^\circ$)	C_l	$\frac{C_l}{C_d}$
-5	0.3255	54.2500
-4	0.4299	58.0946
-3	0.5343	60.7159
-2	0.6386	62.6078
-1	0.7430	63.2821
0	0.8404	64.3985
1	0.9425	77.1324
2	1.0599	85.3908
3	1.1620	88.6870
4	1.2625	91.0250
5	1.3636	94.1172
6	1.4540	93.2293
7	1.4929	88.6897
8	1.4575	81.8492
9	1.6282	79.0274
10	1.6905	73.6789
11	1.7652	67.2100
12	1.8412	59.9403
13	1.9087	50.8843
14	1.9264	41.1357
15	1.9274	31.0871

Appendix F

Optimized Cruise Propeller

Table F.1: Propeller Properties for Cruise Case 1 (Objective Function = C_p)

Element:	Position (in)	Chord (in)	β (°)	Pitch
1	0.720	0.522	40.946	3.93
2	1.307	0.565	40.737	7.07
3	1.893	0.505	28.958	6.58
4	2.480	0.589	27.518	8.12
5	3.067	0.544	26.356	9.55
6	3.653	0.474	22.756	9.63
7	4.240	0.353	20.156	9.78
8	4.827	0.268	18.071	9.90
9	5.413	0.152	13.832	8.37
10	6.000	0.000	8.704	5.77
			Pitch at 3/4 radius	9.83
J	0.8000		Diameter (in)	12.0
Ct	0.0244		RPM	5500
Cp	0.0218		Motor Power (Watts)	66.20
η_{prop}	89.54%		Torque (oz-in)	12.87
			Thrust (oz)	7.55
Motor Voltage (Volts)	7.78			
Motor Current (Amps)	8.51			
η_{motor}	71.70%		η_{system}	62.20%

Table F.2: Propeller Properties for Cruise Case 2a (Objective Function = $PropellerPower$)

Element:	Position (in)	Chord (in)	β (°)	Pitch
1	0.720	0.577	57.746	7.17
2	1.307	0.593	43.346	7.75
3	1.893	0.597	39.937	9.96
4	2.480	0.553	34.422	10.68
5	3.067	0.525	24.889	8.94
6	3.653	0.441	22.895	9.69
7	4.240	0.341	18.898	9.12
8	4.827	0.262	14.973	8.11
9	5.413	0.143	11.845	7.13
10	6.000	0.000	1.526	1.00
			Pitch at 3/4 radius	8.67
J	0.8085		Diameter (in)	12.0
Ct	0.0248		RPM	5442
Cp	0.0220		Motor Power (Watts)	64.62
η_{prop}	91.14%		Torque (oz-in)	12.72
			Thrust (oz)	7.51
Motor Voltage (Volts)	7.69			
Motor Current (Amps)	8.4036			
η_{motor}	71.63%		η_{system}	65.29%

Table F.3: Propeller Properties for Cruise Case 2a (Objective Function = $PropellerPower$)

Element:	Position (in)	Chord (in)	β (°)	Pitch
1	0.698	0.527	54.353	6.11
2	1.266	0.550	43.204	7.47
3	1.834	0.612	35.056	8.09
4	2.971	0.551	33.987	10.18
5	3.539	0.530	25.063	8.73
6	4.108	0.451	24.838	10.29
7	4.108	0.353	21.876	10.36
8	4.676	0.280	18.775	9.99
9	5.245	0.128	12.768	7.47
10	5.813	0.000	4.803	3.07
			Pitch at 3/4 radius	10.20
J	0.8359		Diameter (in)	11.6
Ct	0.0283		RPM	5433
Cp	0.0261		Motor Power (Watts)	65.35
η_{prop}	90.64%		Torque (oz-in)	12.84
			Thrust (oz)	7.52
Motor Voltage (Volts)	7.70			
Motor Current (Amps)	8.49			
η_{motor}	71.53%		η_{system}	64.83%

Appendix G

Optimized Climb Propeller

Table G.1: Propeller Properties for Climb Case 3 (Objective Function = $\frac{1}{C_t}$)

Element:	Position (in)	Chord (in)	β (°)	Pitch
1	0.487	1.409	58.049	4.908
2	0.884	1.463	45.207	5.595
3	1.281	1.578	31.079	4.852
4	1.678	1.517	28.190	5.651
5	2.075	1.534	20.501	4.875
6	2.472	1.308	20.296	5.744
7	2.869	1.056	15.991	5.166
8	3.266	0.805	14.601	5.345
9	3.663	0.474	12.378	5.051
10	4.060	0.000	4.885	2.180
$\beta_{3/4}$ (°)	15.38		Pitch at 3/4 radius	5.25
J	0.2452		Diameter (in)	8.12
Ct	0.2072		RPM	7956
Cp	0.0982		Motor Power (Watts)	190.71
η_{prop}	51.74%		Torque (oz-in)	17.21
			Thrust (oz)	28.10
Motor Voltage (Volts)	11.1			
Motor Current (Amps)	17.18			
η_{motor}	74.62%		η_{system}	38.61%

Table G.2: Propeller Properties for Climb Case 4 (Objective Function = $\frac{1}{F_t}$)

Element:	Position (in)	Chord (in)	β (°)	Pitch
1	0.584	1.420	42.671	3.381
2	1.059	1.376	33.071	4.334
3	1.535	1.374	23.345	4.162
4	2.010	1.395	18.384	4.198
5	2.486	1.337	12.619	3.497
6	2.962	1.270	10.490	3.446
7	3.437	1.139	9.783	3.724
8	3.913	0.853	6.348	2.735
9	4.388	0.420	3.700	1.783
10	4.864	0.000	1.465	0.781
$\beta_{3/4}$ (°)	8.26		Pitch at 3/4 radius	3.29
J	0.2067		Diameter (in)	9.73
Ct	0.1054		RPM	7876
Cp	0.0.0415		Motor Power (Watts)	131.10
η_{prop}	52.51%		Torque (oz-in)	17.60
			Thrust (oz)	28.87
Motor Voltage (Volts)	11.1			
Motor Current (Amps)	11.81			
η_{motor}	74.05%		η_{system}	38.88%

Appendix H

Optimized Climb-Cruise Propeller

Table H.1: Propeller Properties for Climb Cruise Case 5 (Objective Function = $1 - \eta_{prop}\eta_{motor}$)

Element:	Position (in)	Chord (in)	β (°)	Pitch
1	0.639	0.994	53.738	5.476
2	1.160	1.023	38.633	5.827
3	1.681	1.084	36.985	7.956
4	2.202	1.103	30.661	8.203
5	2.723	1.096	26.319	8.464
6	3.244	0.994	21.798	8.152
7	3.765	0.908	20.314	8.758
8	4.286	0.694	18.726	9.129
9	4.807	0.357	17.575	9.567
10	5.328	0.000	16.416	9.863
Cruise Case 5				
$\beta_{3/4}$ (°)	19.61		Pitch at 3/4 radius	8.92
J	0.9009		Diameter (in)	10.6559
Ct	0.0457		RPM	5500
Cp	0.0470		Motor Power (Watts)	82.20
η_{prop}	87.60%		Torque (oz-in)	15.33
			Thrust (oz)	8.79
Motor Voltage (Volts)	8.11			
Motor Current (Amps)	10.14			
η_{motor}	69.82%		η_{system}	61.16%
Climb Case 5				
J	0.2684			
Ct	0.1413		RPM	5538
Cp	0.0739		Motor Power (Watts)	151.45
η_{prop}	51.32%		Torque (oz-in)	24.43
			Thrust (oz)	27.54
Motor Voltage (Volts)	9.37			
Motor Current (Amps)	16.16			
η_{motor}	62.66%		η_{system}	32.16%

Table H.2: Propeller Properties for Climb Cruise Case 6 (Objective Function = $\frac{Power_{Cruise}}{Thrust_{Climb}}$)

Element:	Position (in)	Chord (in)	β (°)	Pitch
1	0.627	0.850	46.855	4.206
2	1.139	0.896	45.400	7.255
3	1.650	0.976	36.412	7.646
4	2.161	1.049	33.671	9.046
5	2.672	1.027	26.021	8.197
6	3.183	0.906	26.021	9.765
7	3.695	0.794	22.309	9.525
8	4.206	0.645	16.611	7.884
9	4.717	0.415	15.770	8.370
10	5.228	0.000	13.792	8.064
Cruise Case 6				
$\beta_{3/4}$ (°)	19.78		Pitch at 3/4 radius	8.80
J	0.9345		Diameter (in)	10.4566
Ct	0.0441		RPM	5404
Cp	0.0469		Motor Power (Watts)	68.78
η_{prop}	87.87%		Torque (oz-in)	13.43
			Thrust (oz)	7.59
Motor Voltage (Volts)	7.74			
Motor Current (Amps)	8.89			
η_{motor}	70.94%		η_{system}	62.34%
Climb Case 6				
J	0.2230			
Ct	0.1244		RPM	6792
Cp	0.0578		Motor Power (Watts)	190.41
η_{prop}	48.00%		Torque (oz-in)	26.15
			Thrust (oz)	33.82
Motor Voltage (Volts)	11.00			
Motor Current (Amps)	17.31			
η_{motor}	65.74%		η_{system}	31.55%