Propeller Design and Optimization Using a Robust Genetic Algorithm and a Computationally Efficient Solver

by

Grady Pastor

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
May 7, 2022

Abstract

A computationally efficient and reliable propeller design tool has been constructed using an advanced real coded Genetic Algorithm (GA) and a mid-fidelity potential flow solver. The GA constructs a population of propeller geometries using a series of Bernstein Polynomials (BP) which have a total of 63 coefficients. This population of propeller geometries is then tested using a reliable and efficient solver. The best four members from the population are then obtained by means of a tournament style selection followed by a round-robin style tournament to determine the true maximums. A following population is then built using the 63 characteristics from the four most optimal members. The process of build, test, select, and build is carried out for several demes or subpopulations which provide the initial population for the main generational loop. After all the deme and main generations have been executed, the GA will provide a propeller that matches the desired thrust input for the specified operating conditions and diameter while maintaining high propulsive efficiencies due to the nature of the fitness function. This thesis describes the technical details of the optimizer, solver, and associated tooling, validation cases for the solver, and sample optimization results.

Acknowledgments

I would first like to thank my professor Dr. Roy Hartfield for all his guidance during the development of my thesis and other graduate studies as well as his former student, Dr. Vivek Ahuja, who was an enormous source of information and assistance for my work. I would also like to thank my parents for their loving support, and finally, I would like to thank my girlfriend who always provided words of encouragement.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BP	Bernstein Polynomial
CCS	Component Cross Section
GA	Genetic Algorithm
MDAO	Multi-Disciplinary Analysis and Optimization
UAM	Urban Air Mobility
UAV	Unmanned Aerial Vehicle
UIUC	University of Illinois at Urbana-Champaign

Chapter 1: Introduction

The need for a high-speed, high-fidelity propeller design and optimization tool has been satisfied by using a vorticity based potential flow solver and a streamlined and efficient real coded genetic algorithm. With an increased interest towards electric powered flight, propellers are given more and more attention as they are the most practical applicants for propulsion in this genre of aircraft. The design of electric air vehicles requires extremely efficient propeller designs that can match the exact required thrust and power load that is applied to the motor for optimum operating conditions[1,2]. Commercial off the shelf propellers generally are not sufficiently well matched to electric aircraft performance requirements to achieve efficiencies required for vehicle viability. Thus, there is a need for a design tool which can provide a robust conceptual design based on flight conditions and thrust requirements. To drive a design toward a physically achievable solution, the tool must be able to model performance metrics with reliable fidelity and to employ population-based optimization methods the tool must be computationally efficient. By having a robust conceptual design process the propeller and vehicle designs emerging from the conceptual design process are far more mature, ultimately saving time and resources. There are computational fluid dynamic tools that are capable of predicting performance with high fidelity, but these tools are not sufficiently computationally efficient for conceptual design trade studies. Lower order methods such as blade element theory, lifting line, XFOIL and XROTOR can be used in various combinations to produce results for special cases; however, such tools are not suitable for angle of attack analysis, or for cases in which there is a two-way interaction between propellers or between propellers and other aerodynamic components. For this work, a modern surface vorticity solver known as FlightStream® was selected for the propeller performance analysis. FlightStream® has both steady and unsteady solution options which are capable of addressing the propeller performance analysis problems efficiently and with at least preliminary design level fidelity.

The concept of the Genetic Algorithm stems from a purely survival-based system, much like that of which is exhibited in nature[3]. A population of members is randomly created, and each member of the population is tested to determine its fitness. The fittest members pass their genes onto the next generation by crossing with other relatively fit members. These new members are then tested, and the fittest ones of the current population pass their genes to the next population and so on and so forth. The GA depends on mutations to expose good attributes of a member[4]. Without extensive mutating, positive genes and gene combinations will never become exposed and passed onto the next population. Thus, a weaker population is created. This same behavior can be examined in the wild. For instance, a species of animal that is present in an environment that requires its members to have two distinct survival characteristics: speed and proper camouflage. However, the current population only consists of members that possess the ability to attain high speeds. Over the course of hundreds of generations, a mutation occurs in one of the offspring which cause the pigment in the skin to change color to fit in with the surroundings more appropriately. This member would be considered extremely fit for its environment, and thus, it would reproduce more than the other less fit members. This mutation in the skin pigment would then become a dominating characteristic of the population. The GA operates on the same basis of survival. Unless there is a large amount of mutation across the generations, favorable characteristics will never be exposed[5].

The Genetic Algorithm is a tool used in design and optimization problems in which a maximum or minimum value is found depending on the fitness function. This GA takes advantage of the real coded genetic algorithm basis to find the best performing propeller for given flight conditions. The idea of the GA in application with aerospace design and optimization has been around since the 1980's, and it has been applied to a vast range of engineering problems[6] that vary from solid rocket motor design to guidance and control[7,8]. Many improvements have been made to GAs since their first appearance in the 50's and 60's[9] when John Holland from the University of Michigan made their discovery[10]. These improvements have shortened run times by developing more effective methods of parent selection to increasing the probability of a proper mutation or crossover. However, there is still a limit for the number of variables that must be predicted by the GA for a given computational time[11]. This GA extends that limit on two fronts by means of a better performing GA and an extremely computationally efficient flow solver that decreases run times.

Previous propeller performance tools have focused on 2-D, thin surface, geometries to efficiently analyze the general performance characteristics[12]. Here, 2-D refers to mean surface approximations which only use a thin surface for the propeller blade which captures twist, chord and sweep variations. The 3-D geometries refer to models which incorporate some thickness to the airfoil shape. While a 2-D model is an efficient method, it is not overly effective in the sense that any practical propeller geometry will be constructed and built based off a 3-dimensional propeller design which incorporates some cross-sectional airfoil shape. Therefore, to obtain the information about efficiencies and thrusts at various flight conditions for a 3-dimensional geometry is far more important. Moreover, while trends can be observed in both 2-dimensional geometries and 3-dimensional geometries, the 2D geometry is limited to chord, twist and sweep values. The 3D propeller blades can not only provide more realistic numbers regarding the propeller, but airfoil shapes can also be implemented to the design of the propeller. This allows for the conceptual design to further expose problems and information not seen by low fidelity design tools.

There are however challanges to the 3D geometry optimization. The most obvious of these draw backs is the increase of design parameters to model the added dimension from 2 to 3. This can often double the number of parameters, and thus, the convergence time for the GA portion can take much longer. The number of generations must be increased as well as the number of members in each generation. Furthermore, the solver used to analyze each of the geometries will become more computational expensive as the mesh becomes more complex with the added dimension. Therefore, an effort to streamline the basic methods of the GA to effectively handle the necessary number of parameters to model a propeller and the analysis tool used in the GA is required.

The main efforts for the work presented here:

- Develop an efficient GA capable of handling the vast amount of required design variables
- Develop a geometry definition that virtually has no limits on the design space other than user inputs
- Apply the GA to stand-alone propellers as well as coaxial propeller designs

Conceptual and preliminary design requires tools that possess high computational efficiency as well high fidelity to further extend the concepts and ideas in the design phase. Many tools exist that provide fast solutions, but they lack the fidelity needed to provide a sturdy preliminary design. Furthermore, there exist such tools that are the opposite in which they provide the necessary fidelity, but the computational times are not compatible with current large scale optimization techniques. This work uses FlightStream®, which provides the necessary fidelity to strength preliminary designs, but also offers computational times that can pair with optimization schemes.

Chapter 2: Basic Theory

The requirement for reliable preliminary design level fidelity and short computational times for the GA design approach drove the decision to use FlightStream®. This section provides explanation for the flow solvers that were not chosen to address this specific problem and why FlightStream® was implemented.

2.1 CFD Solvers

Three-dimensional Computational Fluid Dynamic (CFD) analysis tools are capable providing very high-fidelity solutions for propellers and solving an unsteady problem of propeller-wing interactions[13]. These tools are also capable of accurately modeling a given propeller over a wide range of different freestream velocities and rotational values without having to adjust the solvers methods or expand upon them; however, these tools are not capable of completing the tasks with high computational efficiency in most cases[14]. This informs a need for an accurate aerodynamic analysis tool with full three-dimensional flow solving capability, at least conceptual design level fidelity, robust and high-fidelity geometry input, unsteady solving capability and computational efficiency.

2.2 Lower Order Tools

Some of the existing lower order tools for obtaining performance data include QPROP or XROTOR; however, these aerodynamic tools generally offer only mean camber surface definitions for the geometry and steady flow solution options[15]. XROTOR is a tool designed specifically for free tip propellers, ducted rotors, and wind machines[16]. The tool can provide accurate predictions of axisymmetric and spatially non-uniform distributions of slipstream and the induced velocity. The geometry for XROTOR is modeled by radial distributions of the chord length, twist, and airfoil data. While this may produce an accurate model in reasonable time, it provides no information about the far field. A recent study was done on propeller-wing interactions. In the study, the aerodynamic tool used was XROTOR. The induced velocities at the location of the wing were not directly produced by XROTOR. Instead, a vortex theory-based procedure was constructed to model the propeller effects at the wing with the given XROTOR data. Furthermore, this aerodynamic tool also lacks the ability to model the propeller at any angle of attack in the freestream. Only freestream velocities parallel to the axis of rotation are modeled. XROTOR is incapable of modeling any "edgewise" flight conditions[17].

QPROP provides a relatively similar analysis for the propeller model which is implemented with a simpler geometry design. The propellers examined in this tool are modeled with a blade element/vortex method. The geometry file consists of distributed radial values and the corresponding twist and chord lengths. The model also requires extensive performance information about the airfoil such as $C_{L\alpha}$¸ C_{Lmin}, C_{Lmax}, etc[18]. Often XROTOR and QPROP can provide excellent results for cruise conditions or for specific design scenarios for which appropriate models have been adapted; however, they do not offer a full range of flight condition analysis capabilities, and they have only limited aero-propulsive integration analysis capabilities.

2.3 Potential Flow Solvers

There are two main types of potential flow solvers: vorticity based, and pressure based. Pressure based solvers use the determination of pressure fields along the provided geometry to determine the aerodynamic loads that act on the object. A combination of source and sink panels along with doublet panels on the trailing edge is used to provide a solution for pressure based potential flow solvers. Vorticity solvers depend on the circulation about a given geometry while taking advantage of the Kutta-Joukowski theorem and applying it to each panel on the surface. These potential flow solvers use vortex rings and doublet distributions over the surface of the geometries. Vorticity based solvers also allow for the use of non-manifold and non-conformal mesh surfaces unlike pressure-based solvers. In general, the vorticity based potential flow solvers are much more robust and allow for courser meshes which ultimately reduce the computational time[19]. The need for a robust solver with short computational times is required for the use of optimization with a GA. Early-stage solutions for the GA can be extremely rough and pose a plethora of problems for numerical flow solvers with extremely arbitrary designs. It is necessary to have tool that can provide a solution for these designs.

FlightStream® offers this suite of characteristics and was thus selected for this optimization study. FlightStream® is a high-fidelity tool with very short computational times because it is a vorticity based potential flow solver[20]. Unlike most three-dimensional flow solvers, FlightStream® uses surface vorticity sheets and vorticity-based loads which greatly expedites the solving time. The volumetric meshes seen in conventional solvers take far longer to provide results and offer minimal advantages in the accuracy of calculations[21].

2.4 Genetic Algorithms

A genetic algorithm (GA) is an optimization technique that mimics the behavior of the natural world in which organisms mutate and create offspring which share parents' genes. The fittest of these organisms will survive to produce more offspring who share their genes; the less fit of these organisms will die out. The GA operates by the same set of parameters: fitness-based selection and mutation/gene crossing. Proper mutations are rewarded, and mutations which do not show progress or hurt the population are discarded.

2.4.1 Overview

The genetic algorithm consists of many generations. Each of these generations has a population of *n* members. Each of which is a solution to the GA. Whether or not the member is a good solution depends on the fitness function of the GA. Each member has a set number of parameters or genes that describe it. These members are then assigned a fitness level based off a fitness function which describes that member's ability to do the task at hand. The members with highest fitness will move onto the gene pool where their parameters will be swapped and/or mutated with the other higher functioning members. The products of this gene pool will be the next generation and so on and so forth until the maximum number of generations has been met or the GA has converged on a solution.

There are two main operations to consider when constructing the real coded GA: mutate/cross parameters between members and determining fitness levels. There are many techniques for making alterations to members; however, they all follow the same basic principles of crossing over genes followed by mutations. It is extremely important for the mutations to have the capability to produce extremely wide variation in parameters as well as very fine adjustments. The fitness level for each of the members must be assigned. This fitness assignment is done by the fitness function which should incorporate every important aspect of what is trying to be optimized. Failure to include all necessary values can lead to untrustworthy fitness assignments, and thus, members are passed on who should have been terminated. This can lead to sub optimal solutions.

2.4.2 Binary vs Real Coded

The real coded GA and the binary GA are the two main types of genetic algorithms. The binary GA uses a string of binary numbers to represent each members attributes[22] whereas the real coded GA uses real numbers to model each characteristic[23]. In the case of the binary GA, the mutations occur to the string of binary numbers which are then converted to real numbers that are implemented in the model. The real coded GA simply applies the mutation to the real numbers themselves which describes the given member. While binary GAs have dominated since the existence of the Genetic Algorithm itself[24], the real coded GA proves to be the dominant of the two in many cases. There have been optimization studies done in which binary and real coded GAs run in parallel[25]; however, for the sake of simplicity, this section will focus on the use of one or the other. While the binary GA offers some advantages over the Real Coded GA, there are also many situations in which the roles are reversed which has ultimately led to the decision for using the real coded GA in this optimization study. One major issue with binary GAs is known as Hamming cliff which is when the values of adjacent numbers differ in each bit. This can lead to convergence issues as the GA approaches a global maximum. Furthermore, using real coded parameters to model each member allows for large optimization domains for each variable as well as gradual mutations over the generations to find solutions[26].

2.4.3 Cross Over Types

The crossovers in a GA provide the necessary bases for the variation between generations without straying too far from partially correct solutions. The GA will first make the new population by means of cross overs, and then mutate that population. Without the crossovers, the GA would lack any interaction between the best performing members; therefore, correct solutions would be left completely up to partially random guessing.

2.4.3.1 Laplace Cross Overs

The Laplace crossover method is used in many genetic algorithm applications[27] and is given by two equations each describing the children of the two parents. The following crossover is applied to each parameter individually.

$$y_1 = x_1 + \beta|x_1 - x_2| \tag{1}$$

$$y_2 = x_2 + \beta|x_1 - x_2| \tag{2}$$

x₁ and *x₂* are the parents (the fittest members from the previous population). *y₁* and *y₂* are the children of the fittest two members of the previous population. β is a random number that satisfies the Laplace distribution and is generated by the following equation.

$$f(x) = \begin{cases} a - b\log(u_i)\,, & r < 0.5 \\ a + b\log(u_i)\,, & r \geq 0.5 \end{cases} \tag{3}$$

a is the location parameter, and *b* is a scaling parameter. u_i and r_i are random numbers between 0 and 1. From the information provide in equations 1 and 2, a correct mutation is only likely to occur 50% of the time.

2.4.3.2 Point Cross Overs

Pont crossovers can be useful in the application of real coded GA's; however, they can be limiting without a proper amount of mutation. Point crossovers take the characteristics of two parents and directly apply those characteristics to the children without any alteration other than the

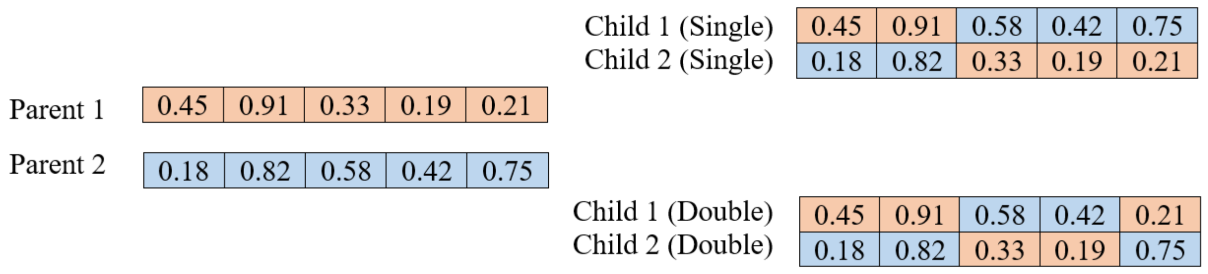


Figure 1: Point crossover example

blending of the two parents. An example of a single point and a double point crossover are shown in Figure 1. There can be as many cross overs as there are variables that describe a member minus one. In this case, the maximum cross over is a 4-point. In such a case, the children would have every other parameter swapped from each of the parents. Furthermore, the order in which the cross overs are represented can be changed (the location of the split point can move) to further increase crossover diversity. Nevertheless, the point cross over technique implies that the correct values already exist in the population. This is virtually never the case. Therefore, the point cross over mutation is very impractical unless combined with other mutation strategies

2.4.3.3 Linear Cross Overs

Linear crossovers provide a wider range of mutations than point cross overs, but the tendency of the mutation to move in the proper direction is not extremely effective. Linear cross overs take parent one and parent two and combine the attributes to form an infinite number of possible solutions. Provided parents x_1 and x_2 the following three solutions can be developed as example.

$$child_1 = 0.5x_1 + 0.5x_2 \tag{4}$$

$$child_2 = 1.5x_1 - 0.5x_2 \tag{5}$$

$$child_3 = -0.5x_1 + 1.5x_2 \tag{6}$$

Now it is seen that all possible solutions have the potential to converge on the correct value; however, the computational time to do so is quite extensive. A proper mutation is a one in three chance. The first of the solutions provides a child with a value that lies somewhere in the middle of the two parents. The second solution provides a child that exceeds the value of the first parent to some degree outside the bounds of parent one and two. The last solution provides some value to the offspring, that will exceed parent two beyond the bounds of parent one and parent two. The target value is unknown and can sit in the middle of the two parents or outside the bounds of the parents on either end. Therefore, a proper mutation is given one out of every three mutations. There are more solutions that can be developed by altering the coefficients of x_1 and x_2, but these will only increase or decrease the extent to which the above-mentioned crossovers will be behave and not the odds of properly mutating.

2.4.3.4 Blended Cross Overs

Blended cross overs offer the same range of mutation as the linear cross overs and provide a better method for convergence. These are very similar to the modified Laplace cross overs except the beta value is replaced by an alpha which is held constant throughout the time if the simulations. Using parents x_1 and x_2, two children can be formed as follows

$$child_1 = x_1 - \alpha(x_2 - x_1) \tag{7}$$

$$child_2 = x_2 + \alpha(x_2 - x_1) \tag{8}$$

With only two solutions provided, and a set value of alpha, the crossover variation is limited. While solutions will converge on a value due to the decreasing nature of the term in the parenthesis, the odds of making a proper mutation is not as likely as the modified Laplace crossover (discussed in section 6.3.1). Nevertheless, none of the crossover schemes provide a completely perfect cross. The pairing of the crossover types and minor advantages in each of them makes the difference in convergence times.

2.4.4 Mutation Types

Genetic Algorithms depend largely on the mutations of the generations. There are many mutations that provide a variety of solutions to the GA. A number of these mutation types were examined for this GA. The mutations themselves range in effectiveness and such effectiveness depends on the point at which the GA has progressed through its timeline. More drastic mutations prove to be very effective in early stages while the more moderate changes provide better solutions in later stages. These mutations are described in full in the following subsections.

2.4.4.1 Power Mutations

Power mutations account for a large majority of the mutations in each of the populations and provide some of the more radical mutations to the propellers. Power mutations are defined by the following equation[28].

$$x = \begin{cases} \bar{x} - s(\bar{x} - x^l), & t < r \\ \bar{x} + s(x^u - \bar{x}), & t \geq r \end{cases} \tag{9}$$

x^l and x^u are the lower and upper bounds for the parameter that is to be mutated. $\bar{x}$ is the parent or the most fit member of the previous population. *r* is a random number between 0 and 1. *t* is determining factor for the mutation and is presented.

$$t = \frac{\bar{x} - x^l}{x^u - \bar{x}} \tag{10}$$

s is included to determine the extent of the mutation and is defined as follows

$$s = s_1^p \tag{11}$$

s_1 is a random number between 0 and 1. *p* is the index of mutation. This parameter governs the amount of mutation for given parameters. As the index of mutation increases, the amount of mutation decreases and vice versa.

2.4.4.2 MPTM

The Makinen, Periaux, Toivanen Mutation (MPTM) can be described fully by Deep[29] in which a mutated child, $\hat{x}$, is produced by the parent *x*. The mutation process is described by equations 12 and 13.

$$\hat{x} = (1 - \hat{t}\,)x^l + \hat{t}x^u \tag{12}$$

$$\hat{t} = \begin{cases} t - t\left(\frac{t-r}{t}\right)^{b} & r < t \\ t & r = t \\ t + (1-t)\left(\frac{r-t}{1-t}\right)^{b} & r > r \end{cases} \tag{13}$$

t is described in the same manner as the power mutation. If one were to work out the problem, it would be shown that this mutation is the exact opposite of the original power mutation in the sense of the direction in which the mutations tend to move. Like the power mutation, the MPTM has a parameter which governs the strength of the mutation, *b*. As *b* is increased the mutations grow weaker and weaker. Furthermore, as the GA progresses through the generations it does not lose any strength in the mutations if the MPTM is used.

2.4.4.3 Non-Uniform Mutation

The Non-Uniform Mutation (NUM) provides an excellent technique for converging values near the final generations of the GA, yet it provides the necessary range of large mutations in the early stages. Examples of these have been used and explained in multiple applications[30,31]. The process by which the mutations occur is shown.

$$x^{t+1} = \begin{cases} x^{t} + \Delta(t, x^{u} - x^{t}), & r \leq 0.5 \\ x^{t} - \Delta(t, x^{t} - x^{l}), & r > 0.5 \end{cases} \tag{14}$$

t represents the generation that the GA is currently working on, and *r* is a random value between 0 and 1. The *Δ* operator is defined as follows.

$$\mathit{\Delta}(t, y) = y\left(1 - u^{\left(1-\frac{t}{T}\right)^{b}}\right) \tag{15}$$

u is a random value between 0 and 1, and *T* is the total number of generations in the GA. Furthermore, as the GA reaches the last of its generations, the mutations begin to decrease in strength do to the term in parentheses that is raised to the *b* power. Thus, the model will converge properly in theory.

2.4.5 Demes

Aside from the vast range of crossovers and mutations, Genetic Algorithms have a multitude of other approaches which not only assist in locating a maximum in less iterations but also in avoiding local maximums[32]. One method for avoiding local maximums is through the use of demes, or subpopulations, which operate for the sole purpose of providing the starting generation to the main generational loop[33]. These demes operate for a specified number of generations which can range anywhere from 1 to infinity depending on the available computational power. The demes then put forth their best members to create a population which starts the main set of generations. The demes can communicate throughout the generations of the main group or only once at the start. A visualization that uses 4 demes is provided in Figure 2. Here four demes

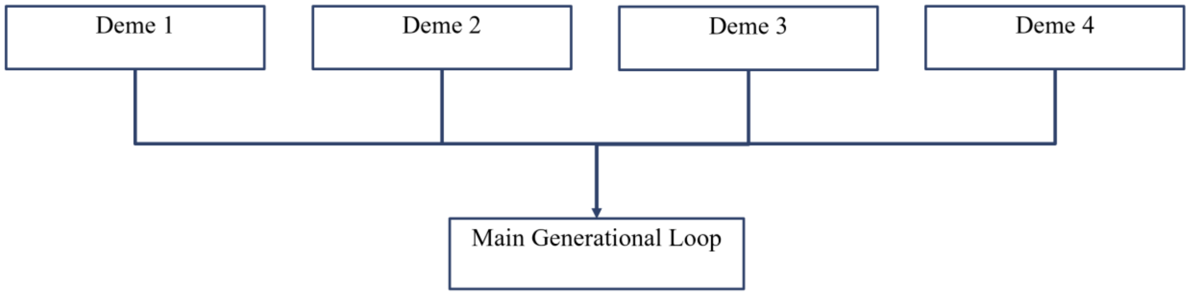


Figure 2: Deme Example

are used to run for a specified number of times and then provide information to the main generational loop. Demes in this application have been introduced in a number of GAs to improve the overall performance of the GA[34]. Because the demes operate entirely independently of each other, the best members should be similar with some differences. Obviously, if each of the demes is executed for many iterations, the most optimal member from each deme should be same, but in this application, the demes are only executed for ranges of 10 generations due to computational power availability. Thus, not enough time is provided for the demes or miniature generational loops to converge on a global maximum.

2.5 The Propeller

The propeller at its base is like that of any other propulsion device in that it simply takes some amount of a fluid and changes the momentum to impart a force on a vehicle by using radially distributed lifting surfaces or blades. The propeller for analytic solutions can be modeled as an actuator disk or even more rudimentary models as a control volume through which air flows in at a relatively low velocity and leaves at a high velocity. From an efficiency perspective, the one important characteristic of all propellers is the amount of energy it takes to cause this momentum change. The power to drive these essentially lifting surfaces is often gathered from a gas turbine or a piston driven engine. In recent years, strides have been made in the application of electrical power sources for propellers[35]. Thus, the argument is made for new more efficient propellers.

When new propeller driven aircraft are developed, a fuselage is developed by one company, the power sources for the propulsion systems are developed by another, and the propellers themselves are developed by a third company such as Master Airscrew, Aeronuat, Hartzell, etc. The propellers are designed to maximize the performance of the aircraft for the given operating conditions: RPM, freestream velocity, power source limits, etc.[36]. Much work has been done on analyzing and modeling electric propulsion systems to power these propellers because electric motors have increased power densities and high efficiencies[37,38]. The purpose of this thesis is to provide an expedited answer to the propeller design problem. Over the years many achievements have been made in the design and implementation for propellers. The rest of this subsection will provide a brief history to propeller theory.

The theory of modern propeller design begins with Rankine and Froude in the 1800's when they used momentum relation for analysis of marine propulsive devices. While the analysis was focused on a different propeller application than discussed here, the models are analogous to each other. The Wright brothers soon followed with their development of air-based propeller systems[39]. While the Wright brothers contributed very little to the theory of propeller design, the Wright brothers were the first to make truly revolutionary improvements to the modern propeller design. Their first propeller was developed in 1902 in which design they used a twist which obtained a constant angle of attack along the radius. The propeller produced roughly 12lbs of thrust at 1600RPM in a freestream of 25mph. The propeller was 28" in diameter and required 0.8hp to operate at this condition[40].

The Wright brothers developed propellers which were on the same level of efficiency as those present today (71%– 76%). The propellers used for their self-powered flight were arguably the hardest part to implement due to the lack of testing capabilities. The Wright brothers realized that the forward performance was vastly different from the static test which had been conducted; however, the Wright brothers only had access to simple static test environments with limited instrumentation. To properly model the performance of forward flight an actual flight test was required. This was both difficult and time consuming[41].

While the Wright Brothers' achievements were extraordinary for their time, the more remarkable of the achievements towards propeller theory come from Prandtl and his lifting line theory[42] and the continuously worked upon blade element theory[43]. The original application for lifting line theory is applied to wings on an aircraft, but seeing that propellers are just lifting surfaces themselves; it has been applied in a number of propeller applications[44]. Both are very popular for propeller analysis; however, the level of fidelity for both models is limiting as described in prior sections. Nevertheless, the background information is necessary for understanding the problem at hand.

2.6 Variable Pitch Propeller

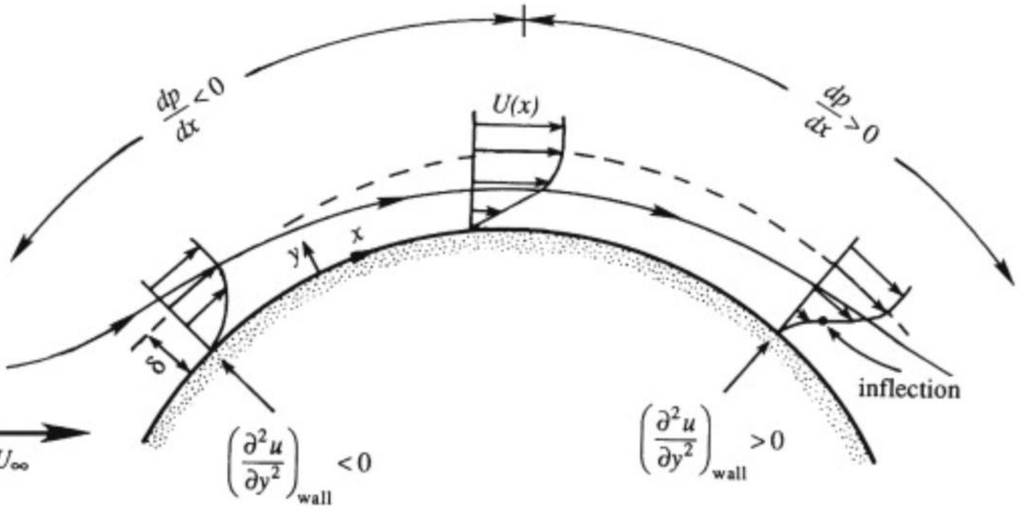
While this GA focuses on optimization for small UAV/UAM propellers with a constant pitch, it should be noted that the GA can run simulations for variable pitch rotors. This section is presented in order to provide the set up for the variable pitch problem. The variable pitch propeller operates by simply adjusting the pitch of the propeller blades to increase or decrease thrust at various flight conditions. In the variable pitch model, the GA optimizes a constant speed propeller. These propellers are typically used on larger aircraft due to the weight of the gears used to control the pitch of the propeller. There exist many types of mechanical devices for controlling the pitch of the propeller[45]; however, that is beyond the scope of this work. The variable pitch propeller is of great interest due to its ability to bring forth the full capabilities of larger, more powerful engines operating in adverse conditions such as high altitudes or increased free stream velocities[46]. In terms of the optimization problem, there are challenges to the variable pitch propeller. The main advantage for using a variable pitch propeller is its ability to easily adapt to a multitude of flight conditions whereas the fixed pitch propeller is typically designed for one flight condition (typically cruise)[47]. With the increased number of design conditions, the GA requires more inputs and linked with the increased inputs comes an increased requirement for the number of iterations. From the single point design of the fixed pitch propeller, the variable pitch problem must converge on thrust and efficiency for as many flight conditions as presented. When applied in this GA, the propeller is given 4 flight conditions, therefore, a total of 8 parameters to converge on. Moreover, at each one of these flight conditions comes an adjusted pitch angle. Thus, there is an increase in the number of design variables. Due to the complexities involved in the variable pitch optimization, the problem was merely explored in this study and should be a large focus for future efforts.

2.7 Coaxial Propellers

The coaxial propeller problem introduces a far greater challenge in modeling the behavior of the rotors when compared to the single standalone propeller model. The standalone propeller model is never a case which offers true fidelity because the stand-alone propeller model is never implemented in real world applications. A propeller will always experience effects from other objects in the flow whether that object be a wing placed in front of or behind the propeller or perhaps another propeller placed in the flow such as the coaxial configuration. Because the propeller application is subsonic in nature, objects placed behind the propeller will still have effects on the propeller though it may be acceptable to ignore them. The truly daunting challenge is to model the propeller with something placed in front of it such as the pusher-proper scenario or even more complex the coaxial problem. The coaxial propeller problem is one of the more challenging configurations to accurately capture because the entirety of the problem is unsteady. The forward propeller sees effects from the rear propeller because for the most part in these configurations the propellers are very close to each other. Furthermore, in most applications of the coaxial set up, the propellers are operating in a static environment, further permitting communication from the rear propeller to the forward propeller. Now, these complexities do not even begin to address the effects that the forward propeller will have on the rear propeller. The rear propeller is operating in an extremely unsteady environment from the wake of the forward propeller. As the forward propeller increases its rotational speeds, the rear propeller becomes far less effective as the incoming velocities will be increased. While the effects of the rear propeller on the front propeller are measurable, they are overall negligible[48]. Thus, it can be concurred that the effects of the forward propeller on the rear propeller are far greater than the effects of the rear rotor on the front rotor[49]. Therefore, in this case, to fully demonstrate the capabilities of the solver's computational efficiency and fidelity, the rear propeller is optimized.

2.8 Flow Separation

Flow separation is an important aspect of any aerodynamic analysis. As a fluid flows over a surface, it is either in the presence of an adverse or a favorable pressure gradient. Favorable pressure gradients are characterized by a decreasing pressure in the direction of flow and an increase in velocity which is why the boundary layer formed from this is referred to as an accelerating boundary layer. Adverse pressure gradients are characterized by increasing pressure in the direction of the flow and decrease in the velocity hence the boundary layer name, decelerating boundary layer[50]. Flow over surfaces that have an adverse pressure gradient have boundary layers that decelerate, thicken, and then form a point of inflection. Both pressure gradients can be shown on a single surface in a flow. Favorable pressure gradients are associated with converging surfaces while adverse pressure gradients are associated with diverging surfaces.



$\frac{dp}{dx} < 0$
$U(x)$
$\frac{dp}{dx} > 0$
y
x
δ
inflection
U_∞
$\left(\frac{\partial^2 u}{\partial y^2}\right)_{wall} < 0$
$\left(\frac{\partial^2 u}{\partial y^2}\right)_{wall} > 0$

Figure 3: Pressure Gradient Example

An example of the two is provided in Figure 3[51]. Here the leading edge of the sphere is a converging surface, and the trailing edge is a diverging surface. The favorable pressure gradient has no potential for separation; while, if the adverse pressure gradient is strong enough and persists long enough, the flow will eventually separate over the surface and reverse direction near the surface boundary[52]. This separated flow will then lead to increased amounts of drag on the surface and a loss of lift due[53]. Now, why is this important to the problem optimization problem?

For the FlightStream® solution a lower limit was set for flow separation that was allowed to occur on the top side of the propeller. Due the shape of a propeller blade and the high angles of attack, the flow across the surface of the airfoil sections can experience very strong adverse pressure gradients which lead to separation. While the numerical flow solver used in this analysis can detected where the separation has occurred, it cannot describe the aerodynamic effects that accompany the propeller blade under these conditions. Because airfoils experience increased amounts of drag, and losses in lift when the flow is separated, then a propeller will experience increased amounts of torque due to the increase in drag and a decreased amount of thrust. For this reason, a limit was set on the amount of separation. Propellers with more than a given percent of the upper surface separated were provided a fitness of -10.00. This fitness ensured that their characteristics namely, twist, would not be passed to the next generation. This hard coded value for fitness is justified because this is a thrust and efficiency optimized propeller solution, and if the separation across the top is that high, then the limited amounts of thrust and decreased efficiency of a separated propeller will lead to an unfit propeller.

2.9 Maximum Efficiency

There are realistic limits to the efficiency of propellers given the thrust produced, propeller radius, and freestream velocity. The propulsive efficiency for any propulsion device is a ratio of the thrust power and the amount of kinetic energy production. The thrust power is simply defined as the thrust produced multiplied by the freestream velocity. The kinetic energy production for a propeller is defined as one half the mass flow rate multiplied by the square of the change in velocity from freestream to the far field behind the propeller. This full relation is shown below

$$\eta_p = \frac{Tu}{\frac{1}{2}\dot{m}(u_e^2 - u_\infty^2)} \tag{16}$$

Given the equation, it is understood that there must be a limit to efficiency even in the most optimal of assumptions and operating environments. The thrust for a propeller is defined as follows

$$T = \frac{1}{2}\rho A(u_e^2 - u_\infty^2) \tag{17}$$

Plugging this into the efficiency equation

$$\eta_p = \frac{\left[\frac{1}{2}\rho A(u_e^2 - u_\infty^2)\right]u}{\frac{1}{2}\dot{m}(u_e^2 - u_\infty^2)} = \frac{\rho A u_\infty}{\dot{m}} \tag{18}$$

For a propeller, the mass flow rate is defined as the free stream density multiplied by the area over which the propeller accelerates the working fluid multiplied by the velocity of the fluid across the propeller disc. The velocity of the fluid across the disc is defined as an average of the free stream velocity and the velocity in the far field or

$$u_{disc} = \frac{(u_e + u_\infty)}{2} \tag{19}$$

Therefore,

$$\eta_p = \frac{\rho A u_\infty}{\rho A u_{disc}} = \frac{u_\infty}{\frac{(u_e + u_\infty)}{2}} = \frac{2u_\infty}{u_e + u_\infty} \tag{20}$$

While the exit velocity is unlikely know for propeller operating conditions in the design phase, the thrust is known. Therefore, to make appropriate assumptions to maximum operating efficiency for a propeller the thrust equation defined above is solved for the exit velocity which is given as follows

$$u_e = \sqrt{\frac{2T}{A\rho} + u_\infty^2} \tag{21}$$

Now, plugging this into the developed equation for estimating propulsive efficiency

$$\eta_p = \frac{2u_\infty}{u_\infty + \sqrt{\frac{2T}{A\rho} + u_\infty^2}} \tag{22}$$

With an estimation of the maximum efficiency for a propeller obtained, relations can be drawn regarding propeller performance as a function of its operational environment.

The original equation presented for efficiency suggests that the efficiency is only a function of the free stream velocity and the exit velocity which is true; however, for more accurate modeling of a propeller, the exit velocity variable was replaced by two other variables: area and thrust produced. While the density of the operating environment is important to the performance of the propeller, the actual sizing of the propeller and its efficiency are of higher importance and therefore, the density is held constant over the following examples.

From equation 16 for propulsive efficiency, it can be determined that the maximum efficiency of a propeller is experienced when the exit velocity matches the freestream velocity in forward flight. Yet, in this case, the propeller, or any thrust generating device for that matter, will not serve its purpose of producing any thrust by equation 17 presented for thrust. Therefore, the following trade study was done to demonstrate the effects of thrust, freestream velocity, and disc area on the propeller efficiency. Figure 4 provides efficiency and changes in velocity over different thrust production levels and free stream velocities. All propellers were given a radius of 10 cm and operate at sea level i.e., density is set to 1.225 kg/m^3. The right axis in the figure represents the change in velocity of the fluid from the free stream to the far field given a thrust, area, density, and freestream velocity. The left axis represents the efficiency of the propeller. As stated early and

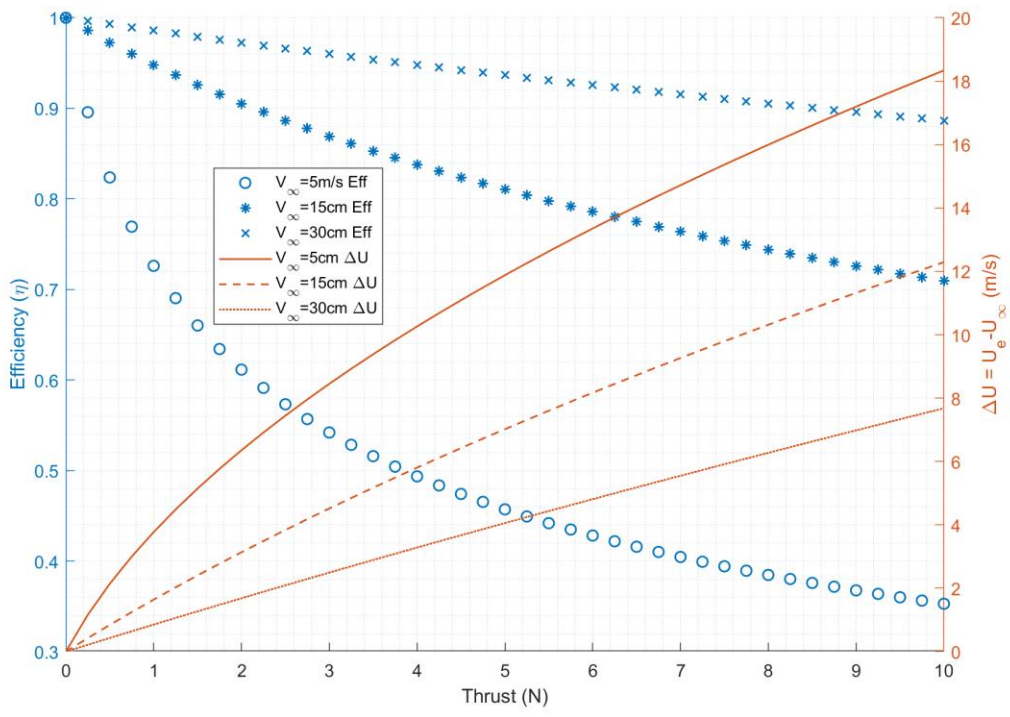


Figure 4: Efficiency and Velocity change vs Thrust

demonstrated here, the maximum efficiency occurs when the change in velocity is zero, and thus, thrust is zero. Note the solid lines which represent the performance of the propeller in the slowest of three freestream velocities plotted. This flight condition has the lowest efficiencies of the three because it also has the largest increase in freestream velocity at any point. At first glance, this may appear to be a mistake in the calculations; however, thrust is determined by the difference in the velocities squared. Therefore, as freestream velocity increases, the required differential in velocity to achieve the same thrust decreases. Take for example the operating condition in which 8 N of thrust is required. For the given predefined parameters, the propeller needs the velocity terms in the parenthesis to be equal to roughly 415 m^2/s^2. Therefore, the propeller operating in the 5m/s environment needs to produce exit velocities equal to ~21m/s which leads to a velocity differential of 16m/s. On the other hand, the propeller operating in the 30m/s environment only needs to produce exit velocities of ~36.25 m/s which is a 6.25m/s differential. Therefore, by the original equation for propulsive efficiency, the propeller operating in the higher velocity environment will produce greater maximum efficiency values.

The same function for maximum propeller efficiency is then used to observe the behavior of efficiency on propeller areas and freestream velocities again. In cases, the thrust is fixed to a constant 5 N, and the propellers have radii of 5, 15, and 30 cm. The smallest of these propellers is obviously the most inefficient, as it must accelerate the fluid through an area that is 36 times

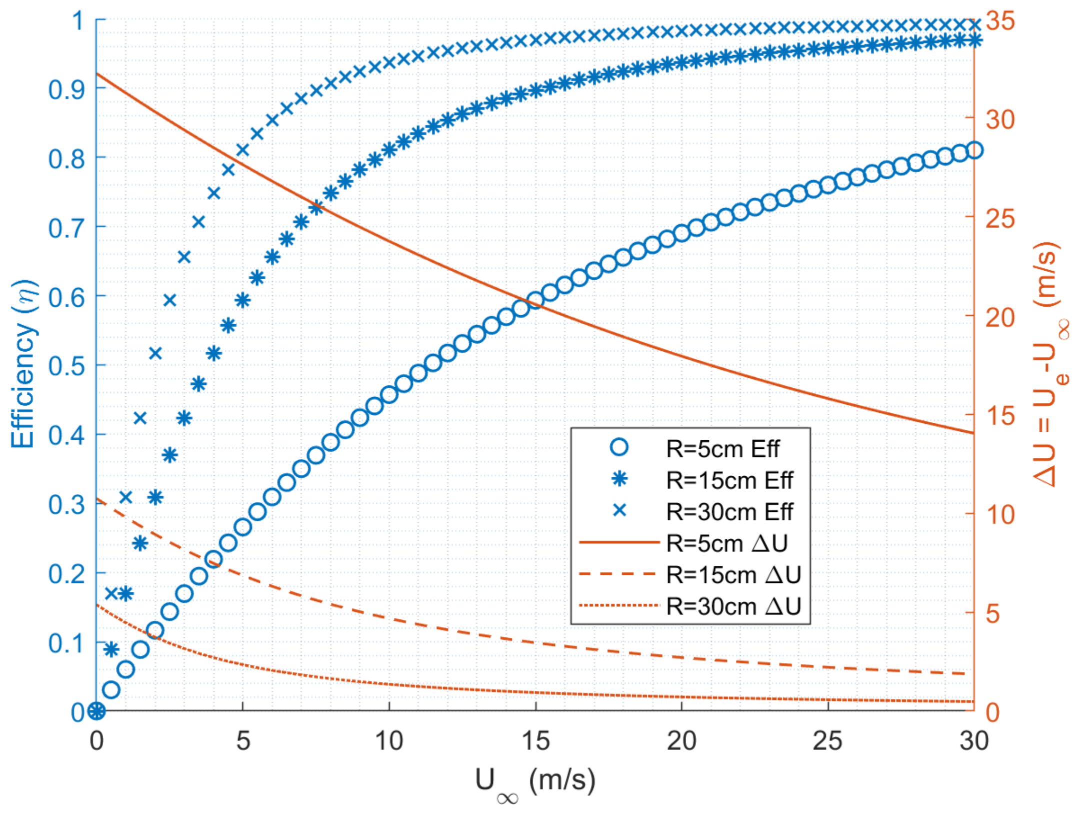


Figure 5: Efficiency and Velocity Change vs Freestream Velocity

smaller than the area of the largest propeller. To compensate for this reduced area and mass flow rate, the propeller must provide a greater acceleration to the fluid in order to match the thrust that is required. The effect of this large fluid acceleration is seen in the efficiency line for the smallest propeller as it persists to be the lowest of the three propellers. One important factor to note here is the continuous increase in efficiency. This is because the thrust is constant for all cases which is not necessarily the case in a nominal operating scenario; however, for the sake of maximum propeller efficiency analysis this will be accepted. As the thrust is held constant the exit velocity remains greater than the free stream velocity for all cases, and therefore, as the freestream velocity extends to infinity, the exit velocity will effectively match the freestream velocity while still producing thrust which is indeed false. Nevertheless, the results for absolute maximum efficiency are presented in this section to provide a basis for these comparisons in the results. While the propeller optimizer, converges on propellers with optimal efficiency, it should never produce propellers that have efficiencies that match the ones presented here.

Chapter 3: The Propeller Geometry

The process by which the geometry is conceived and the program or means by which it is built are crucial aspects to the development of the objective function, necessary to any trade study or optimization including a GA approach. There are many programs for building a CAD model of the geometry and there are even more options in the process of developing the geometry. The process by which the geometry is constructed should be accurate and replicable. Furthermore, the design space should not be limited by the design tool used to create the geometry.

3.1 The Geometric Build

There are many processes and types of geometric builds for constructing a propeller. One important consideration is the difference between “Machine Geometry” and “CFD Geometry”. Often, the level of geometric detail included in a drawing to build a part is too expansive for practical flow solutions. Outer mold lines for fluid solutions are routinely simplified to correctly capture the flow surfaces but not detailed features such as fasteners, seam lines, and other features required for machining parts. For surface vorticity solvers, there is the additional consideration of whether a simple thin airfoil shape mean camber line will be used or whether an airfoil with some considerable thickness will be implemented. If the thin sheet propeller representation is implemented, then a simple mean camber model can be constructed. If the application of the propeller is to have a thick airfoil, then the consideration of how to construct the propeller becomes more complicated. For the objective function used in this work, two processes by which the propeller geometry is to be built were considered: an OpenVSP model or a Component Cross Section file. Both representations considered are capable of representing a geometrically correct outer mold line.

3.1.1 OpenVSP

OpenVSP is an open-source tool developed by NASA for conceptual design and study of aircraft and aircraft components. VSP is easy to use and provides fast models. The program allows the user to create a three-dimensional model and then export that model as a surface mesh[54]. An example of the meshing provided by OpenVSP is shown Figure 6. A general propeller geometry is already built into the program. Therefore, only certain aspects of the propeller such as the twist, chord, sweep, thickness, and airfoil sections needed to be altered. OpenVSP is also fully scriptable which is a main requirement for the GA's geometry building component[55]; however, it can provide rough geometries with meshing that have the potential to cause problems for the FlightStream® solver upon importing the .stl file. Furthermore, the application of the script to run OpenVSP in the GA provided little freedom in the geometry design. Predefined variable names were vast, as the scripting language uses angel script. Moreover, the scripting files, CompGeom files, and geometry files themselves occupy an extensive amount of memory. Lastly, by having to access a secondary application in the process of constructing the geometry, computational times would be compromised and one of the main advantages of this GA would be lost. Therefore, while OpenVSP remains an efficient program for building propellers, the application of it in a genetic algorithm is not cost-effective. The OpenVSP tool was used, however, for the validation cases' geometry.

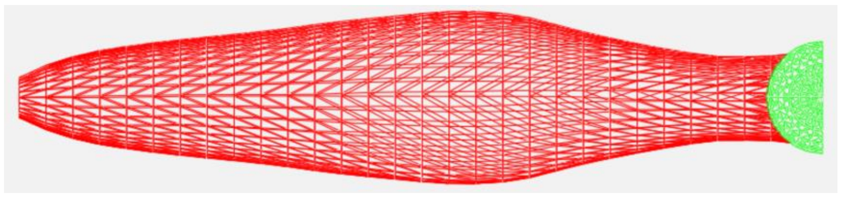


Figure 6: Example OpenVSP Mesh

3.1.2 Component Cross Section

The Component Cross Section (CCS) geometry provides the best means of importing a geometry into the solver. The file is specific to the numerical solver used in this Optimization exercise, FlightStream®. The file import works exactly as the name describes it. A number of cross sections are imported into solver and smooth curved surfaces are lofted between each of the cross-sections in order to provide a surface. The CCS geometry also provides compatible meshing

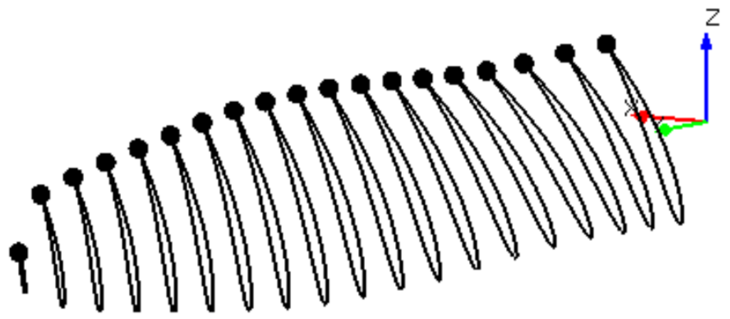
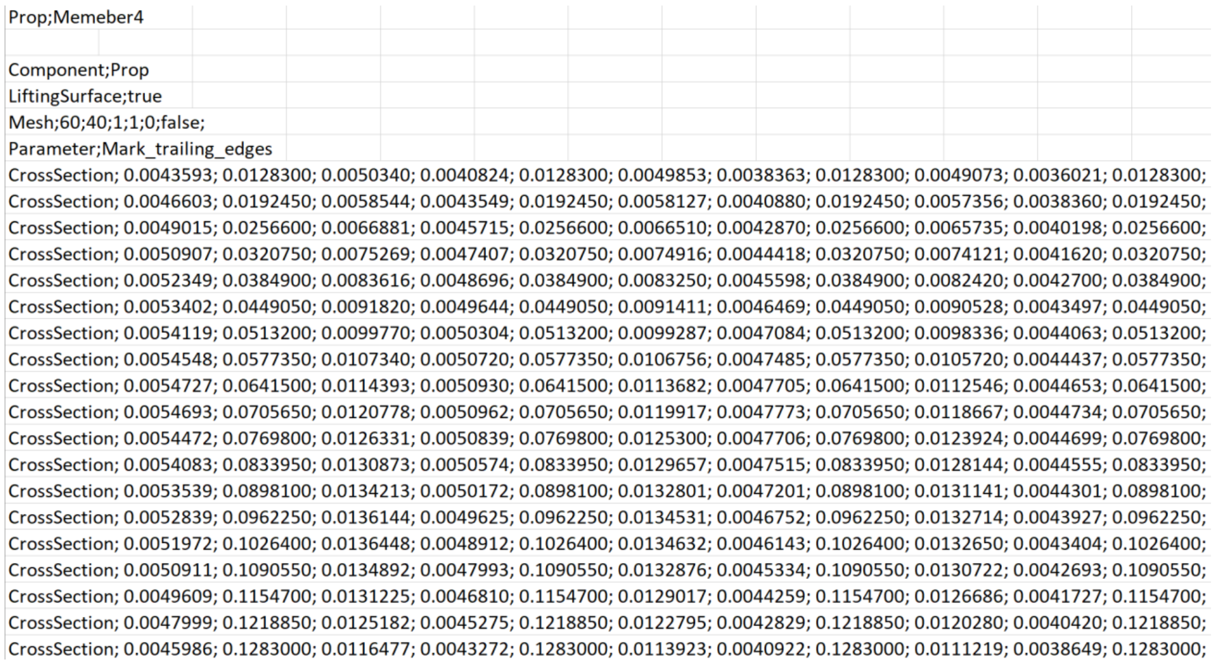


Figure 7: Example Cross Section Import

because it is meshed in the solver. Furthermore, the import file permits the user to define the number of spanwise and chord wise meshing points. Trailing edge detection is another key attribute to the CCS import in which the first point of each of the cross sections is selected to be the trailing edge. An example of the component cross-sections in FlightStream® can be seen in Figure 7. The CCS file is a csv file which is created in a subroutine of the GA and stored in a folder with all the other geometry files. The CCS file consist of all the specified x, y, and z points of the cross sections as well as some additional information regarding the geometry and the nature of the import. An example of the CCS file shown in Figure 8. The first line provides a name of the file import, and the next line provides a name to the component once it is in the solver. The following line informs the solver that this is a lifting surface. The 4th line in the file defines the mesh for the propeller. Here a 60 chordwise by 40 spanwise mesh grid is placed over the propeller. The growth type is set to uniform, and the growth rate is set to 1. The periodicity and the refinement for the object mesh are both turned off. Lastly, in the setup, the trailing edges are marked. The following lines then contain the x, y, and z coordinates in that order for each cross section. The cross sections start nearest to the hub and work their way out. As Figure 8 displays, it is very easy to build these geometry files and store them with little computational cost. These geometry files can be built for every propeller in a population, and then, the entire population can be tested.

3.2 Geometry Definition

The defining characteristics for the propeller geometries come in two forms. The first and limiting geometry definition comes from a pre-built airfoil. The second is provided by a Bernstein Polynomial. Both geometry types rely on the use of polynomials for twist, chord, and sweep values along the span of the propeller, and they rely on the CCS import function for getting the geometry into the solver. The only differences lay in the airfoil cross sectional shapes and the types of polynomials used in developing the blade shapes.



```
Prop;Memeber4

Component;Prop
LiftingSurface;true
Mesh;60;40;1;1;0;false;
Parameter;Mark_trailing_edges
CrossSection; 0.0043593; 0.0128300; 0.0050340; 0.0040824; 0.0128300; 0.0049853; 0.0038363; 0.0128300; 0.0049073; 0.0036021; 0.0128300;
CrossSection; 0.0046603; 0.0192450; 0.0058544; 0.0043549; 0.0192450; 0.0058127; 0.0040880; 0.0192450; 0.0057356; 0.0038360; 0.0192450;
CrossSection; 0.0049015; 0.0256600; 0.0066881; 0.0045715; 0.0256600; 0.0066510; 0.0042870; 0.0256600; 0.0065735; 0.0040198; 0.0256600;
CrossSection; 0.0050907; 0.0320750; 0.0075269; 0.0047407; 0.0320750; 0.0074916; 0.0044418; 0.0320750; 0.0074121; 0.0041620; 0.0320750;
CrossSection; 0.0052349; 0.0384900; 0.0083616; 0.0048696; 0.0384900; 0.0083250; 0.0045598; 0.0384900; 0.0082420; 0.0042700; 0.0384900;
CrossSection; 0.0053402; 0.0449050; 0.0091820; 0.0049644; 0.0449050; 0.0091411; 0.0046469; 0.0449050; 0.0090528; 0.0043497; 0.0449050;
CrossSection; 0.0054119; 0.0513200; 0.0099770; 0.0050304; 0.0513200; 0.0099287; 0.0047084; 0.0513200; 0.0098336; 0.0044063; 0.0513200;
CrossSection; 0.0054548; 0.0577350; 0.0107340; 0.0050720; 0.0577350; 0.0106756; 0.0047485; 0.0577350; 0.0105720; 0.0044437; 0.0577350;
CrossSection; 0.0054727; 0.0641500; 0.0114393; 0.0050930; 0.0641500; 0.0113682; 0.0047705; 0.0641500; 0.0112546; 0.0044653; 0.0641500;
CrossSection; 0.0054693; 0.0705650; 0.0120778; 0.0050962; 0.0705650; 0.0119917; 0.0047773; 0.0705650; 0.0118667; 0.0044734; 0.0705650;
CrossSection; 0.0054472; 0.0769800; 0.0126331; 0.0050839; 0.0769800; 0.0125300; 0.0047706; 0.0769800; 0.0123924; 0.0044699; 0.0769800;
CrossSection; 0.0054083; 0.0833950; 0.0130873; 0.0050574; 0.0833950; 0.0129657; 0.0047515; 0.0833950; 0.0128144; 0.0044555; 0.0833950;
CrossSection; 0.0053539; 0.0898100; 0.0134213; 0.0050172; 0.0898100; 0.0132801; 0.0047201; 0.0898100; 0.0131141; 0.0044301; 0.0898100;
CrossSection; 0.0052839; 0.0962250; 0.0136144; 0.0049625; 0.0962250; 0.0134531; 0.0046752; 0.0962250; 0.0132714; 0.0043927; 0.0962250;
CrossSection; 0.0051972; 0.1026400; 0.0136448; 0.0048912; 0.1026400; 0.0134632; 0.0046143; 0.1026400; 0.0132650; 0.0043404; 0.1026400;
CrossSection; 0.0050911; 0.1090550; 0.0134892; 0.0047993; 0.1090550; 0.0132876; 0.0045334; 0.1090550; 0.0130722; 0.0042693; 0.1090550;
CrossSection; 0.0049609; 0.1154700; 0.0131225; 0.0046810; 0.1154700; 0.0129017; 0.0044259; 0.1154700; 0.0126686; 0.0041727; 0.1154700;
CrossSection; 0.0047999; 0.1218850; 0.0125182; 0.0045275; 0.1218850; 0.0122795; 0.0042829; 0.1218850; 0.0120280; 0.0040420; 0.1218850;
CrossSection; 0.0045986; 0.1283000; 0.0116477; 0.0043272; 0.1283000; 0.0113923; 0.0040922; 0.1283000; 0.0111219; 0.0038649; 0.1283000;
```

Figure 8: Example CCS File

3.2.1 Airfoil Import

The airfoil import is much easier to implement, but the extent to which the geometries must expand in design space is limited. The airfoil import takes a predefined airfoil shape and uses it as the airfoil shape along the propeller. The idea behind this propeller modeling is that the GA would rely on a bank of airfoils to implement on each propeller at given cross sections, and a numerical value in the string of genes for each propeller would determine which airfoil will be used for that propeller at that given cross section. This modeling technique was only used in the early stages of this work and only went as far as having two different airfoils in a propeller. That is not to say the airfoils were set in stone when applied to the propeller. Once the airfoil had been imported for a geometry, it was then scaled by a chord function which was a 6th order polynomial describing the chord along the span of the propeller. Furthermore, the airfoil was also adjusted by some thickness to change the airfoil shape. This thickness as well a pitch and sweep adjustment were also 6th order polynomials. While this method is simple and easily applied, it requires a lot of information about existing airfoil shapes and all the data points that describe them.

The airfoil import method for propeller design is effective in achieving a simple propeller design as it does not require as many parameters to describe the propeller shape and the airfoil cross sections are already built. These two advantages are, however, weaknesses and lead to its elimination from the final version of this work. The design space for the airfoil import method is limited because it uses existing airfoils and only allows for changes in airfoil shape regarding the thickness. Due to its tight geometry restrictions, the following method was chosen as the geometry construction method.

3.2.2 Bernstein Polynomials

The Bernstein Polynomials allow for a vast range of propeller designs and seldom limit the GA in terms of the propeller's geometry. This freedom can be observed in the early generations of the GA as the designs are completely random in nature. In fact, in rare cases, the lack of constraint can be a problem as solutions do not converge due to the abstractness of the propeller design. Nevertheless, this issue is resolved in further generations given the circumstance that it even occurs. This is due to several reasons, but most obvious is the number of cases that are executed. One of the cases will eventually converge, and thus, light will be shed on the propeller's true performance.

The propeller GA takes advantage of Bernstein Polynomials (BP) to construct the definitions of airfoil shape, pitch, chord, and sweep. The equations which describe theses parameters are expanded from Kulfman[56] which first develops the ideas; furthermore, the equations are presented in Burger[57], but here they are expanded and used on a 3-dimensional geometry. The BP used to construct the airfoil can be divided into two separate parts: the shape function and the shaping terms. The shape function is used to model the leading and trailing edge of the airfoil. This is provided below.

$$C_{N2}^{N1}(\psi) = (\psi)^{N1}[1-\psi]^{N2} \tag{23}$$

C is the shape function, and ψ is the percent of the chord length. The first term on the right-hand side describes the radius of the leading edge. Values for the *N1* term range from 0.5 to 1. A value of 0.5 will lead to a rounded leading edge, and a value of 1 will lead to a sharper leading edge. The same relationship exists for the second term which describes the trailing edge shape. An example of the effects of these values can be seen in Figure 9. All shaping terms in the in the airfoils shown

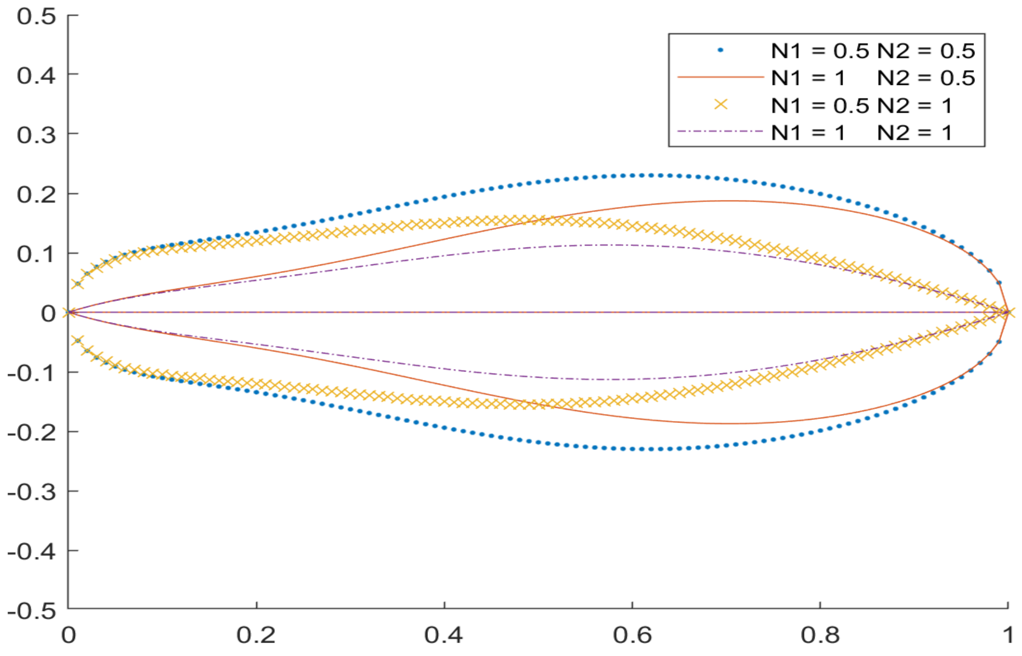


Figure 9: Example of Shaping Function Effects

in Figure 9 are equal, thus demonstrating the effects of the shape function. The shaping terms themselves use a 3rd order BP. Putting the shape function with the shape terms, the airfoil surface at a given point along the propeller blade can be described.

$$Z_{up}(x,y) = \left(\frac{x}{c}\right)^{N1}\left(1-\frac{x}{c}\right)^{N2}\left[Au_1(y)\left(\frac{x}{c}\right)^4 + 4Au_2(y)\left(1-\frac{x}{c}\right)\left(\frac{x}{c}\right)^3 + 6Au_3(y)\left(1-\frac{x}{c}\right)^2\left(\frac{x}{c}\right)^2 + 4Au_4(y)\left(1-\frac{x}{c}\right)^3\left(\frac{x}{c}\right)^2 + Au_5(y)\left(1-\frac{x}{c}\right)^4\right] \quad (24)$$

$$Z_{low}(x,y) = -\left(\frac{x}{c}\right)^{N1}\left(1-\frac{x}{c}\right)^{N2}\left[Al_1(y)\left(\frac{x}{c}\right)^4 + 4Al_2(y)\left(1-\frac{x}{c}\right)\left(\frac{x}{c}\right)^3 + 6Al_3(y)\left(1-\frac{x}{c}\right)^2\left(\frac{x}{c}\right)^2 + 4Al_4(y)\left(1-\frac{x}{c}\right)^3\left(\frac{x}{c}\right)^2 + Al_5(y)\left(1-\frac{x}{c}\right)^4\right] \quad (25)$$

The first two terms in both the upper and lower equations are the shape functions which describe the leading and trailing edge, and the terms in the brackets are the shaping terms that describe the shape of the airfoil between the leading and trailing edge. The *Au* and *Al* terms are function of y (the span) and provide the necessary information to model the change in the airfoil shape in the y direction or spanwise direction. These BP equations can be represented and visualized using Pascal's triangle and increasing the order of the BP shown in Figure 10. By increasing the order of the BP, the variation in airfoil shape from leading to trailing edge can be greatly increased;

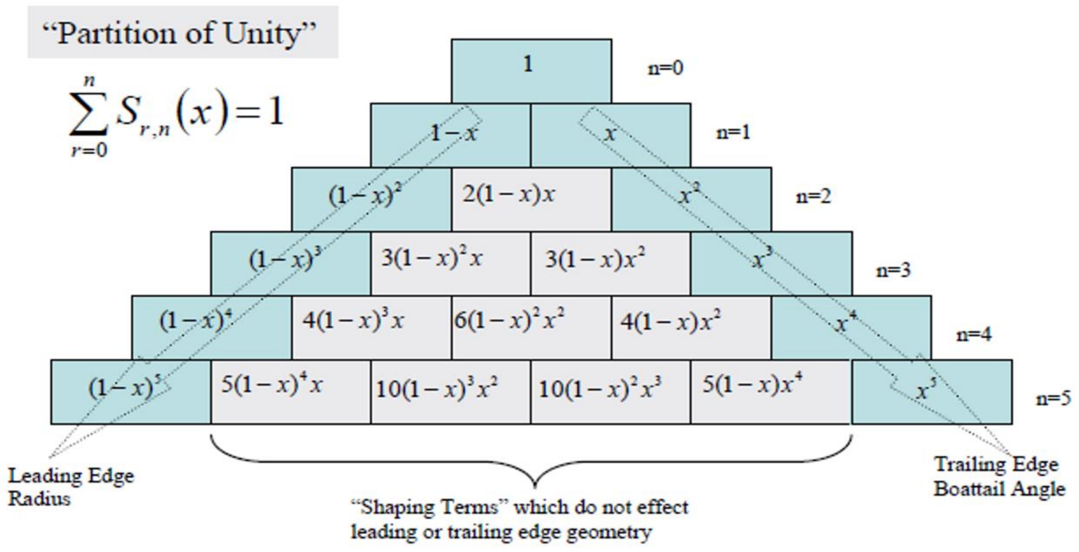


Figure 10: Pascal's Triangle

however, it comes at a cost. The number of variables required for each BP is equal to the order of the polynomial plus one. Furthermore, this only describes the airfoil shape at a given cross section along the propeller blade. The function to describe the upper airfoil surface shaping terms is dependent upon 20 variables. This equation is show below as a 3rd order BP

$$Au_i(y) = au_i\left(1-\frac{y}{r}\right)^3 + 3au_{i+5}\left(1-\frac{y}{r}\right)^2\left(\frac{y}{r}\right) + 3au_{i+10}\left(1-\frac{y}{r}\right)^2\left(\frac{y}{r}\right) + au_{i+15}\left(\frac{y}{r}\right)^3 \quad (26)$$

Y is the distance measured outward from the hub of the propeller and *r* is the radius of the propeller. The same equation is used for the lower surface shaping values

$$Al_i(\mathrm{y}) = au_i\left(1-\frac{y}{r}\right)^3 + 3al_{i+5}\left(1-\frac{y}{r}\right)^2\left(\frac{y}{r}\right) + 3al_{i+10}\left(1-\frac{y}{r}\right)^2\left(\frac{y}{r}\right) + al_{i+15}\left(\frac{y}{r}\right)^3 \quad (27)$$

Each one of *au* terms is used describe the A_{upper} which describe the change in the propeller airfoil in the y direction (hub to tip direction). Therefore, there are four terms used to describe each of the five *au* values resulting in 20 parameters for the top airfoil shaping terms and 20 for the bottom. There also exist a similar set of equations for the shape functions. These use a 1st order BP to provide the N1 and N2 values for the top and bottom which is shown below.

$$Nu_i(y) = nu_i\left(\frac{y}{r}\right) + nu_{i+2}\left(1-\frac{y}{r}\right) \quad (28)$$

$$Nl_i(y) = nl_i\left(\frac{y}{r}\right) + nl_{i+2}\left(1-\frac{y}{r}\right) \quad (29)$$

Now, there are a total of 48 parameters which describe the airfoil shape along the direction of span and chord lengths. There are also another 15 parameters which describe the chord, twist, and sweep along the span of the propeller. These are modeled using 3rd order BP and can be seen below.

$$C(y) = \mathrm{c}_5\left[c_1\left(1-\frac{y}{r}\right)^3 + 3c_2\left(1-\frac{y}{r}\right)^2\left(\frac{y}{r}\right) + 3c_3\left(1-\frac{y}{r}\right)^2\left(\frac{y}{r}\right) + c_4\left(\frac{y}{r}\right)^3\right] \quad (30)$$

$$\mathrm{B}(y) = \beta_5\left[\beta_1\left(1-\frac{y}{r}\right)^3 + 3\beta_2\left(1-\frac{y}{r}\right)^2\left(\frac{y}{r}\right) + 3\beta_3\left(1-\frac{y}{r}\right)^2\left(\frac{y}{r}\right) + \beta_4\left(\frac{y}{r}\right)^3\right] \quad (31)$$

$$\Psi(y) = \psi_5\left[\psi_1\left(1-\frac{y}{r}\right)^3 + 3\psi_2\left(1-\frac{y}{r}\right)^2\left(\frac{y}{r}\right) + 3\psi_3\left(1-\frac{y}{r}\right)^2\left(\frac{y}{r}\right) + \psi_4\left(\frac{y}{r}\right)^3\right] \quad (32)$$

There are a total of 63 parameters which govern the shape of the propeller blade: 24 for the upper airfoil surface, 24 for the lower airfoil surface, 5 for the chord function, 5 for the twist function and 5 for the sweep function. Now comes the question as how to implement the geometric data on the actual propeller itself.

The airfoil implementation has been expanded to require one more step that of which leads to a vast extension in the design space of the propellers themselves. Here, the BP have not only been extended to include a 2-D shape with upper and lower surface curves, but they also include the ability for airfoils to possess large amounts of camber. This is achieved by not applying the lower surface equation directly, but instead, subtracting it from the upper airfoil curve. In doing so, the possibility for an inflection point has been added to the bottom shape design leading to

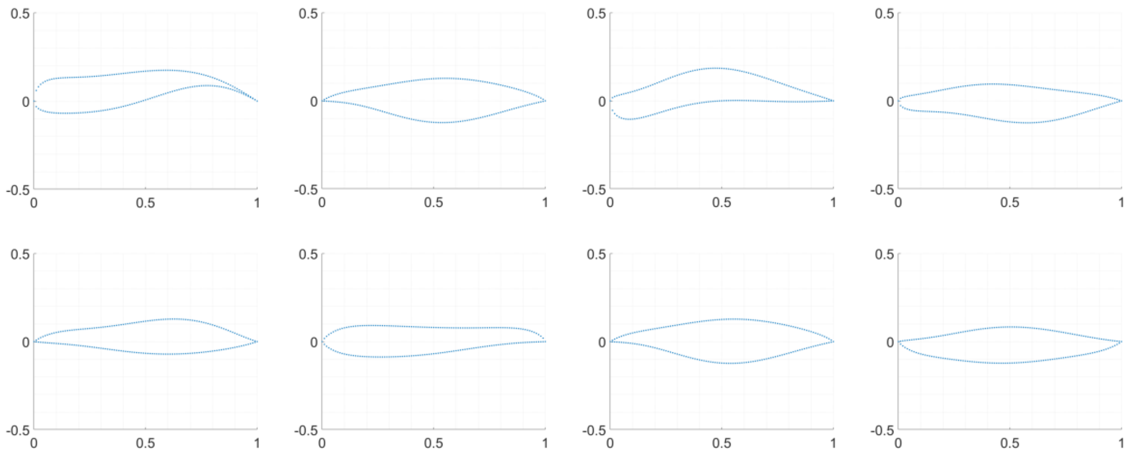


Figure 11: Example Airfoil Variation

camber in the airfoil. This adjustment does not sacrifice any of the design space either. Figure 11 provides examples of direct application of the lower surface equation and then the subtracted modification of the lower surface curve. All the airfoils on the top of Figure 11 are generated from the modified method of airfoil generation, and the bottom four images are generated form the direct application of the lower surface equation. Note the amount of camber in the top airfoils, as well as the absence of it. This demonstrates that by applying the modified method for lower surface calculations, a large design space is achieved. Furthermore, cambered airfoils are ubiquitous on efficiency propellers and most subsonic lifting surfaces for that matter.

Lastly, it should be noted that majority of the propellers presented in Figure 11 are inefficient in design and do not take on the shape of a typical airfoil. A CST method[58] was considered to develop the airfoil; however, it was not used because one favorable aspect of the GA is the vast design space. By using the BP with no geometric input constraints, the design space for the propellers is much larger than CST method and its applied inputs. Furthermore, the solver should be able to determine 'good' airfoil designs from 'bad' ones due to the level of fidelity. Therefore, the poor performing airfoils should be filtered out from the solver results.

3.2.3 Geometry Implementation

To model the geometry, a loop is implemented to apply each of the BP over the span of the propeller. The radial values range from 10% of the radius all the way to the tip of the propeller. This vector of radial values contains the coordinates for each point on the propeller. Within the loop, a vector of the chord lengths along the span is created first. This vector is then used individually throughout the rest of the loop to model the other points at their given locations along the span. Multiple loops are then implemented to fill a matrix for the upper and lower points for the airfoil using the BP which describes the airfoil shape itself as well as the BP which describes the shape functions. The loops range from 0% of the chord to 100% at the given y location. The chord values are represented by the x coordinate direction. The z coordinates are the actual coordinates that are altered by the BP for the airfoil function. After this has been completed a geometry has been built which contains multiple airfoil cross sections along the span of the propeller. These cross sections can be characterized by their airfoil shape and chord lengths. To

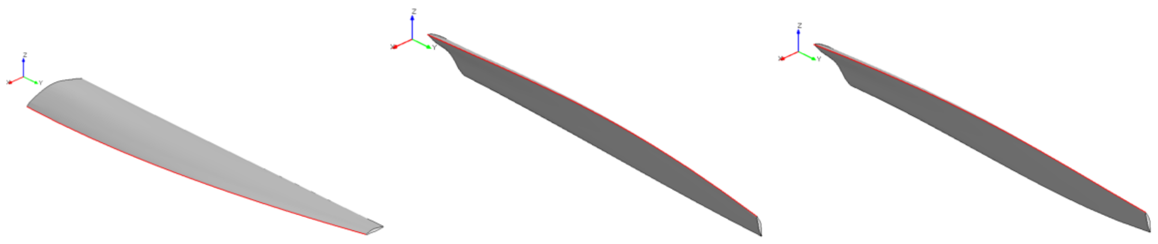


Figure 12: Isometric View of Geometry Build

obtain a functioning propeller, the simple wing that has been created is twisted by the BP for pitch. This is done by rotating given cross sections about the y axis as follows

$$X_{rotated} = X\,cos(\psi) - Zsi\,n(\psi) \tag{33}$$

$$Z_{rotated} = X\sin(\psi) + Z\cos(\psi) \tag{34}$$

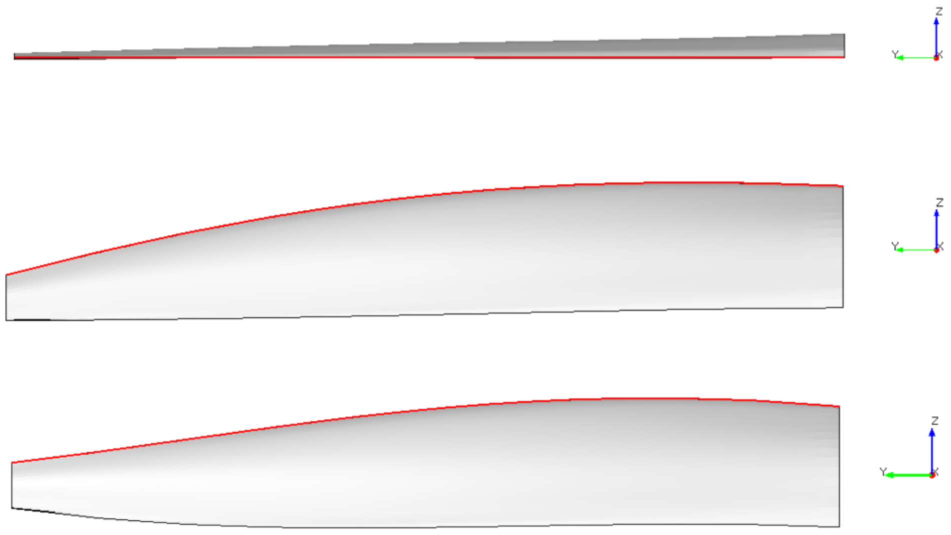


Figure 13: Front View of Geometry Build

This has now twisted all the cross sections by some angle to provide appropriate pitch angles at given cross section to generate lift. Lastly, the sweep polynomial is applied to the propeller in the same manner. Except, this rotation is done about the x axis. An example of the process is provided in Figure 12 and Figure 13. The figures work left to right and top to bottom respectively. The first image present in the figure is the meshed propeller blade with no twist or sweep applied to it and only chord and airfoil functions to describe its shape. The second image has been rotated to allow the propeller to have some pitch. Lastly, the final propeller is presented which is described by the polynomials for airfoil shape, chord, twist, and sweep.

Chapter 4: The Solver

FlightStream® is highly applicable to modeling entire aircraft as well as propellers in steady and unsteady flow regimes. The unsteady model is precisely the tool required for fast and accurate propeller modeling from standalone models to coaxial rotor simulations, and unlike other solvers, FlightStream® provides a solution in minutes or even seconds in some cases[59]. Furthermore, the solver provides the ability to analyze propellers operating with their axial at some angle of attack to the freestream. Lastly, the wide range of geometry types that are compatible with FlightStream® make it extremely easy to create and import a geometry for testing. FlightStream® has been used to demonstrate longitudinal and lateral characteristics of full-scale airplanes, vertical takeoff aircraft, as well as optimizing engine placement along a wing[60,61,62].

4.1 Script and Setup

A fully scriptable solver is crucial to the propeller GA, and the scripting ability that FlightStream® provides is a major driving factor for the use of this flow solver in this GA. The scripts for FlightStream® allow for any operation that can also be conducted manually[63]. The script in this application involves importing the geometry and setting up the simulation for the given inputs to the GA. This script is quite extensive due to the large availability of options for running the solver. The text file starts by importing a single propeller blade from a CCS import as a csv file. The trailing edges are already selected in the csv file by inserting a simple function. The solver then changes the altitude to the provided altitude in the GA text file input. The solver is then changed from the default steady setting to either the steady rotary or unsteady case depending on which propeller optimization is being executed. The rotational speeds for the propeller are then set, and the number of blades is implemented. The solver is then initialized using a wake termination location that gives the propeller the opportunity to make at least 1 full rotation. Once the solver is initialized, the freestream velocity is set as well as the reference velocity. The solver is then executed. After the solver converges, the necessary files are exported for that propeller. The solver is cleared, and the next propeller is loaded uploaded for testing.

FlightStream® solver can be executed in four different settings: steady, steady (rotary), steady (viscous) and unsteady. This subsection will discuss the steady(rotary) and unsteady settings as they are the only settings specifically applicable to the propeller case. While this GA uses the steady rotary solver and the unsteady solver, the unsteady solver provides a far more expansive regime of flight conditions and possibilities. FlightStream® can also be used in vorticity or pressure-based load setting. The propeller GA at hand uses the vorticity-based loads; however, the difference in them should be understood for the purpose of this thesis.

4.2 Steady (Rotary)

The steady (rotary) solver in FlightStream® provides a solution to the propeller problem in seconds which is why it is used for the propeller GA in the stand-alone model. This solver can operate under mirrored conditions. In other words, the geometries can be mirrored across planes or periodically about an axis without increasing the number mesh faces. The one original geometry is geometry that the solver calculates loads and moments on and then simply applies these to the mirrored surfaces. The minimized number of mesh faces allows the solver to converge on a solution in extremely short computational times, and with 128 members per generation, computational speed becomes a large factor in determining the performance aspect of the GA. The steady (rotary) solution requires only an RPM input to the freestream, and the rest of the governing parameters follow the same as other solver settings. In general, propellers converge in less iterations than the maximum setting of 1000, further resulting in short computation times.

4.3 Unsteady

The unsteady solver in FlightStream® provides a wider range of options for running simulations. While the steady rotary solver is virtually only single propeller type simulations, the unsteady solver provides solutions for everything from modeling ground effect to coaxial propeller interactions[64]. The unsteady solution does require a longer run time. The unsteady solver must be run for a large number of time steps to arrive at a converged solution. This solver will run for a user defined number of time steps at defined time increments. Each of these time step iterations will have to converge on a solution or run until the maximum number of iterations is reached. The unsteady solver is used in this GA because of its ability to capture complex flow interactions; however, run times in this application are much greater which impose some limitations to the GA. The unsteady solver also can capture propellers in edgewise flight, and therefore, the objective function could be modified to address optimization for propellers that operate in edgewise flight.

4.4 Vorticity and Pressure Solver

FlightStream® solver can operate in a pressure or a vorticity-based mode to gather forces and moments operating on a body. In the case of the propeller in the steady rotary solver, the vorticity mode was used to gather both forces and moments. The vorticity mode provides a purely inviscid solution by using a method of integrated circulation. This is discussed in greatly detail in Solver Theory section of this work. The pressure mode uses the vorticity mode to calculate pressure fields along the surface of the objects. Furthermore, the pressure mode can apply a boundary layer module which allows for the solver to account for viscous flows. The documentation for the FlightStream® solver can be found in several papers and in the FlightStream User's Guide[65,66].

4.5 Solver Theory

The original development for the solver comes from the work done my Ahuja which is outlined in his dissertation[67]. The solver is a product of Research in Flight®, and much progress has been made toward in enhancing its abilities since the solvers beginning. The solver makes use of the Fast Multipole Method (FMM) which combines the far field source/sink doublets into a single body for computing the effect on the near field. The FMM is used to expedite the solutions for calculating far field effects on near field objects[68]. This method has been applied to numerous topics in aerodynamics[69,70] and is used in FlightStream to combine far-field vorticity into localized point doublets. These point doublets contain the vorticity strength for the entirety of the far-field. Using the far-field, near-field thresholds are established, a spatial octree is constructed around the geometry as shown in Figure 14.

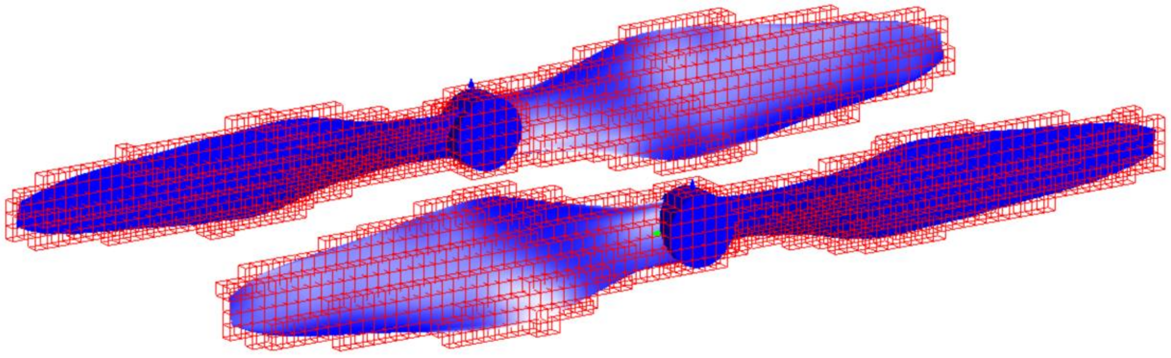


Figure 14: Octree Example

The solver has a laminar and a turbulent method for boundary layer analysis. The boundary analysis calculations apply a 2-D method to the streamlines on the surface of the geometry of interest. The laminar model uses a standard two-parameter Thwaites momentum integral equation as shown below and described in full by Ahuja[71].

$$U\frac{d}{d\eta}\left[\frac{\theta^2}{\nu}\right] = 0.45 - 6\frac{\theta^2}{\nu}\frac{dU}{d\eta} \tag{35}$$

The turbulent model assumes the fluid outside the boundary layer to be isentropic, subsonic, and compressible. The momentum thickness and compressibility factor are given as follows

$$\frac{d\theta_i}{dX} = -\frac{\theta_i}{M_e}\frac{dM_e}{dX}(2 + H_{TR}) + \frac{dx}{dX}\frac{c_f}{2}\left(\frac{T_e}{T_0}\right)^3 \tag{36}$$

$$\frac{dH_i}{dx} = -\frac{(H_i - 0.7)^{3.715}}{4.17}\left[\frac{F}{\theta_i}\frac{dX}{dx} - \frac{H_{\delta-\delta^*}}{M_e}\frac{dM_e}{dx} - \frac{H_{\delta-\delta^*}}{\theta_i}\frac{d\theta_i}{dx}\right] \tag{37}$$

The development for this turbulent model is obtained from Standen[72]. Here x and X are the longitudinal coordinate parallel to the wall boundary and its transform. M_e is the Mach number at the outer edge of the boundary layer. H_{tr} is the transformed shape factor and $H_{\delta-\delta^*}$ is the shape factor associated with the entrainment rate. The non-dimensional mass entrainment rate is given by F. The presented equations are then integrated using a higher order Runge-Kutta[73].

4.6 Run Times

When using a program for optimization or any purpose for that matter it is crucial to take advantage of the system or program in every way possible. This GA is executed on a machine that has 28 cores. While the run times for FlightStream® are extremely rapid relative to other solvers with the same fidelity and even faster when running the steady case, in the process of constructing this GA, there was still need for even faster computational times. FlightStream® allows the user to specify the number of cores that each simulation will occupy for a given run. For original single simulations, this value was set to 6. For the simultaneous run model, the number of cores per run was set to 3. The runs were split into 7 groups each of which was a different flight condition that all propellers were run through. Each group had its very own script that was executed simultaneously.

Furthermore, the run times were largely impacted by the number of iterations given to a single propeller simulation as well as the convergence threshold. In the beginning phases of the GA, FlightStream$^{®}$ solver was set to run for 500 iterations with a convergence threshold of 1E-5 where a forced execution of all the iterations was not implemented. In other words, the simulation would run until either the threshold was met, or the number of iterations was exceeded. The first of these two options is the ideal case for a GA to ensure that true values are provided, and thus, the GA can eliminate unfit members of the population. However, with such a low number of iterations and an absurdly large design space, exotic propellers seen in the first few random generations would lead to nonconverging values in which they ran the full length of the set number of iterations. These leads to two major problems in the design. The first and the most obvious is the fact that a non-converging solution is being recorded and taken into the fitness function calculation. Therefore, a propeller whose thrust that should have been negative is now being reported as positive, or perhaps an unrealistic efficiency is calculated from FlightStream$^{®}$ provided values for torque and thrust. Now, a propeller that should have been eliminated is producing even more like itself in the population. This ultimately leads to a non-converged solution for the GA itself which is far worse than that of a local maximum which will be discussed later. Furthermore, with the unrealistic propeller running for the full number of iterations, the total solver run times increase by extreme amounts. This was also observed in a separate project for FlightStream$^{®}$ involving the unsteady solver for a quadcopter. Furthermore, because the propeller is creating offspring like itself through the GA mechanism, run times are not only increased for the faulty propeller but also for the offspring. Therefore, with the lower number of iterations, the GA produces non-converged solutions and in fact takes longer to run. To solve this issue, a maximum number of iterations was set to 1000. This proved to be the most optimum setting for the GA regarding run times and convergence. While initial populations may take longer to run, every propeller has true solutions provided, and on an entire scale of the GA, the run times are less because as more solvable propellers are produced, FlightStream® has an easier time predicting values. Thus, run times are shorter. The steady solver can converge on an accurate solution in roughly 30 seconds depending on the geometry shape. Abstract geometries with sharp leading edges or odd twist or chord distributions may take longer. Therefore, with 128 members per generation and 140 total generations, the total computational time for the stand-alone propeller model is about 6.2 days.

The unsteady solver requires more computational time as well as attention in the solver settings. The coaxial simulations will take longer to run simply because the interactions between the two propellers is far more complicated to capture. The unsteady simulations run for a number of user defined time steps in user defined increments. The solver must converge on each time step. For propeller models the unsteady solver should allow for propellers to advance 5 to 7 degrees per time step. For the coaxial model, the front propeller should complete one full rotation once its wake has come into contact with the rear rotor to ensure that the wake is full established, and the solver is not converging on a poor solution. That being said, the unsteady solver in the coaxial set up requires roughly 26 minutes to complete one full simulation. With 128 members per generation this would take 55.4 hours or 2.3 days to complete one generation. Therefore, to increase the efficiency of the GA, simulations were executed simultaneously to reduce the number of effective simulations to 38. Therefore, one generation of the coaxial propellers takes 16.5 hours or 0.686 days.

4.7 Meshing

The mesh used for the propellers built in the csv file is a crucial parameter to the solvers ability to converge. The mesh used had to be large enough such that run times in the solver remained relatively low and small enough to allow for the geometry to be smooth to allow for the solver to converge accurately. The mesh implemented on the propeller designs uses 60 points in the chord wise direction and 40 in the radial direction, resulting in roughly 4700 faces for one

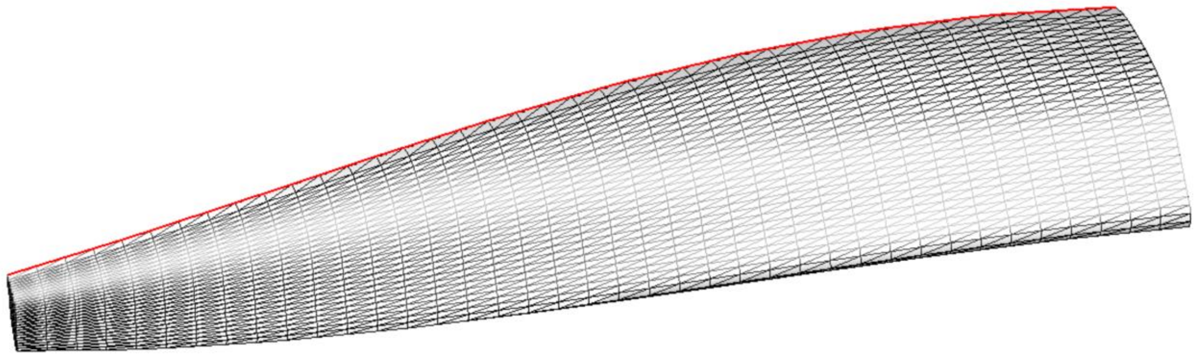


Figure 16: 60x40 Mesh Used in Steady Rotary Analysis

blade. In the steady rotary solver, this will be the total number of mesh faces despite the number of blades, because the steady rotary solver can set up periodic symmetry when initializing. An example of the mesh is shown in Figure 16. The unsteady, coaxial simulations use more mesh faces than the steady solver, but the mesh refinement is far rougher. The forward propeller is a thin surface model which uses a total of 2000 mesh faces for both propellers. Unlike the steady solver,

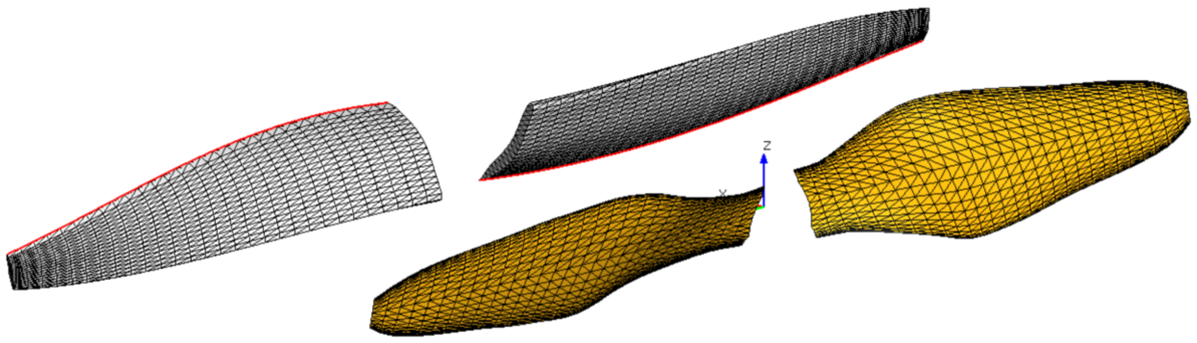


Figure 15: Coaxial Mesh Configuration

the unsteady cannot operate using a periodic symmetry; therefore, every mesh face must be accounted for. The GA produced propeller uses roughly 4600 mesh faces for both blades. An image of the coaxial set up is provided in Figure 15.

Chapter 5: Validation for the Solver

Several sources of propeller performance and geometry data provided ample amounts of validation for the solver. In this chapter, propellers from the UIUC propeller database and the NACA Technical report 640 are examined in FlightStream® and compared to the provided wind tunnel data for the standalone propeller model. The coaxial propellers are validated using a master airscrew propeller that was testing in a wind tunnel.

5.1 UIUC Propellers

The UIUC propeller database was chosen for this validation largely because of the data that was provided. The database provides an appropriate amount of wind tunnel propeller performance data, as well as enough data regarding the propeller geometry. The UIUC propeller database was initially released in 2011 with performance data for 78 propellers. Now the database consists of three volumes with the performance data for nearly 140 propellers. The data consists of a large variety of propellers from different manufactures. The propellers are relatively small and used mainly on small UAVs and model aircraft. The data regarding their performance consists of thrust and power coefficients as well as efficiencies over a range of advance ratios at given values for RPM[74]. Calculations for these thrust coefficients and advance ratios were provided in detail in the release notes. These calculations were reversed to obtain the necessary data needed to run the validation cases. The data for the propellers comes from experiments conducted in a low-speed wind tunnel. The propellers were mounted in the wind tunnel and attached to a small electric motor. Thrust correction factors were implemented to obtain the true thrust produced by the propeller at the given conditions[75].

The data for this validation case comes from Volume 1 of the database and is comprised of an Electric Only 11x7 and a G/F 10x8 propeller from Master Airscrew. The first number in the propeller name refers to the diameter in inches, and the second number refers to the pitch given in inches/rev. The data for these propellers consists of advance ratios ranging from 0.1 to 0.9 and

r/R	c/R	beta
0.15	0.154	32.48
0.20	0.155	37.51
0.25	0.161	38.30
0.30	0.165	35.96
0.35	0.167	32.45
0.40	0.170	28.98
0.45	0.173	26.06
0.50	0.175	23.68
0.55	0.175	21.62
0.60	0.172	19.84
0.65	0.167	18.22
0.70	0.158	16.86
0.75	0.147	15.73
0.80	0.136	14.82
0.85	0.125	14.11
0.90	0.114	13.40
0.95	0.095	13.30
1.00	0.077	13.23

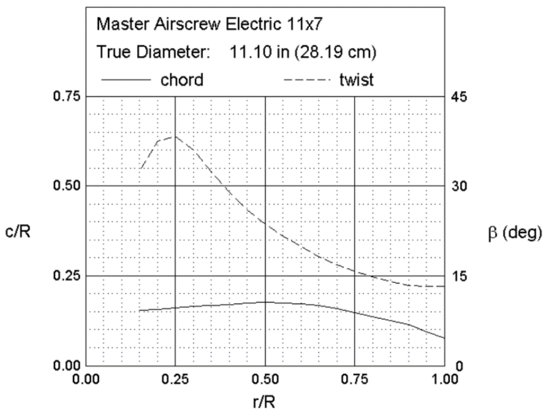


Figure 17: Example UIUC Propeller Database Geometry Data

RPM values ranging from 3005 to 6803 depending on the propeller. The geometry data given in the database consists of a radial value and the corresponding beta and chord value. An example of one of these geometry data sets is provided Figure 17[76]. Notice the absence of any airfoil data or thickness measurements. To properly validate the numerical solver, an accurate geometry model was required. Therefore, some information regarding the thickness and airfoil shape of the propeller at various sections was necessary for modeling the propeller geometries. A 3-D scanner was used to obtain the information regarding the airfoils shape. The scanner used in this data collection was the Go!SCAN20 shown on the left in Figure 18. This scanner has a volumetric accuracy of 0.3 mm/m and point accuracy of 0.1mm. Furthermore, the scanner requires that three points be completely known to collect more points[77]. This data collection set up is exhibited in the right of Figure 18. Here the propeller being scanned is surrounded by textured and colored objects. These objects have three textured points and three colored points that assist the scanner in locating

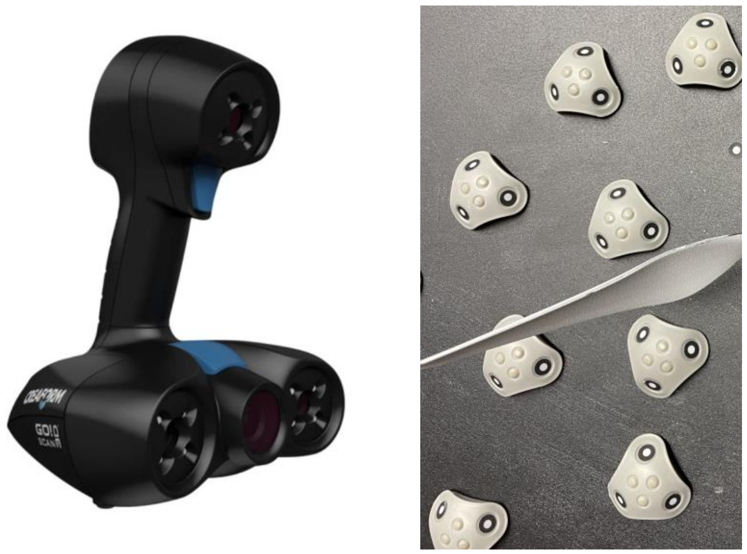


Figure 18: Scanner and Scanner Setup

known points to capture more data.   A raw image from the scanner is provided in the top of Figure 19. The white film that is on the propeller is a powder spray that was used to add some texture and color to the dark surface of the propeller. Originally, the scanner would not collect clean points from the propeller as the dark surface of the propeller did not permit any light to reflect back to

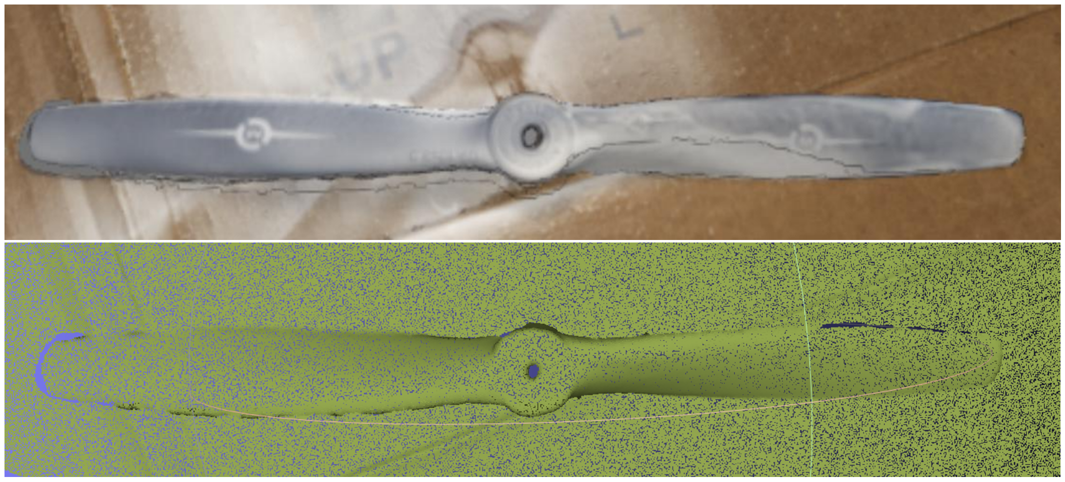
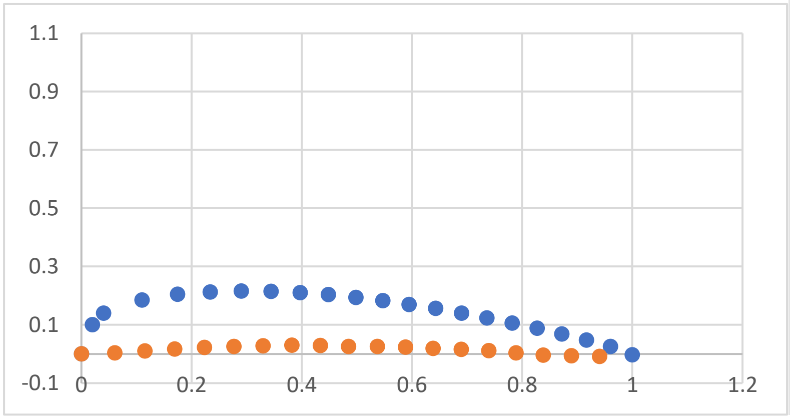
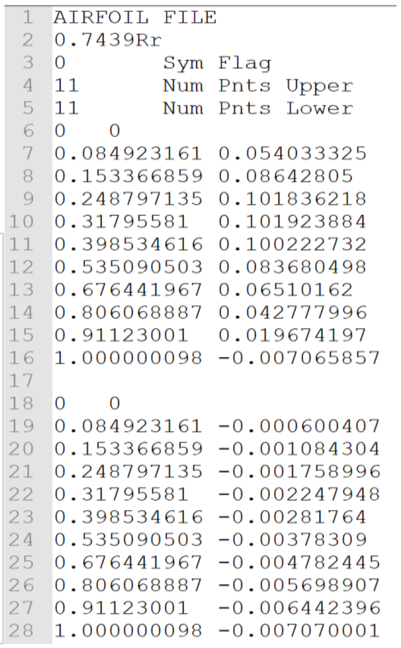


Figure 19: Raw Scan and Enhanced Point Cloud

the scanner. The lower image in Figure 19 is a higher density point cloud from the original scan. More points were added to the scan using a Poison-disk sampling tool in MeshLab, an open-source mesh processing application[78]. The increased number of points serves the purpose of removing the background from the propeller. As the images demonstrate, the background was blended with the propeller itself; therefore, by adding an increased number of points, the edges of the propeller were more clearly defined. Once the point cloud was cleaned, the data was loaded into MATLAB where cross sections of points were collected to obtain the airfoil. These cross sections were placed into an OpenVSP specific file format called an airfoil file. The airfoil files were then loaded into the propeller at various cross sections. An example of an airfoil and the airfoil file are provided in Figure 20. The points gathered from the top scan are shown in blue, and the points from the bottom





```
AIRFOIL FILE
0.7439Rr
0           Sym Flag
11          Num Pnts Upper
11          Num Pnts Lower
0     0
0.084923161 0.054033325
0.153366859 0.08642805
0.248797135 0.101836218
0.31795581  0.101923884
0.398534616 0.100222732
0.535090503 0.083680498
0.676441967 0.06510162
0.806068887 0.042777996
0.91123001  0.019674197
1.000000098 -0.007065857

0     0
0.084923161 -0.000600407
0.153366859 -0.001084304
0.248797135 -0.001758996
0.31795581  -0.002247948
0.398534616 -0.00281764
0.535090503 -0.00378309
0.676441967 -0.004782445
0.806068887 -0.005698907
0.91123001  -0.006442396
1.000000098 -0.007070001
```

Figure 20: Example Airfoil and Airfoil File

scan are shown in orange. While there is only a small amount of camber to the airfoil, it is important that it was captured for the validation case. The full process of geometry construction is explained in detail in Appendix I.

The results from the validation consist of thrust coefficients, power coefficients and efficiencies which are calculated as follows.

$$C_T = \frac{T}{\rho n^2 D^4} \tag{38}$$

$$C_p = \frac{P}{\rho n^3 D^5} = 2\pi C_Q \rightarrow C_Q = \frac{Q}{\rho n^2 D^5} \tag{39}$$

$$\eta = \frac{C_T J}{C_P} \tag{40}$$

The UIUC propellers had data collected over the same advance ratios using different RPM values by adjusting the freestream velocity. Due to viscous effects, there is a large scatter in the data. As the RPM increased with an adjustment in the velocity to maintain a similar advance ratio the propellers saw an increase in thrust and torque as well as efficiency because the thrust increased by a greater amount. The viscous model in solver is capable of modeling viscous effects only for higher Reynolds numbers on the order of a couple 100,000 whereas the UIUC propellers have a Reynolds number of roughly 60,000. This leads to constant thrust and torque values produced by the solver for a given advance ratio despite changes in the velocity and RPM.

Both propellers were modeled using viscous coupling and without viscous coupling. The non-viscous coupling propellers are labeled with "(NV)" while the models with viscous coupling are

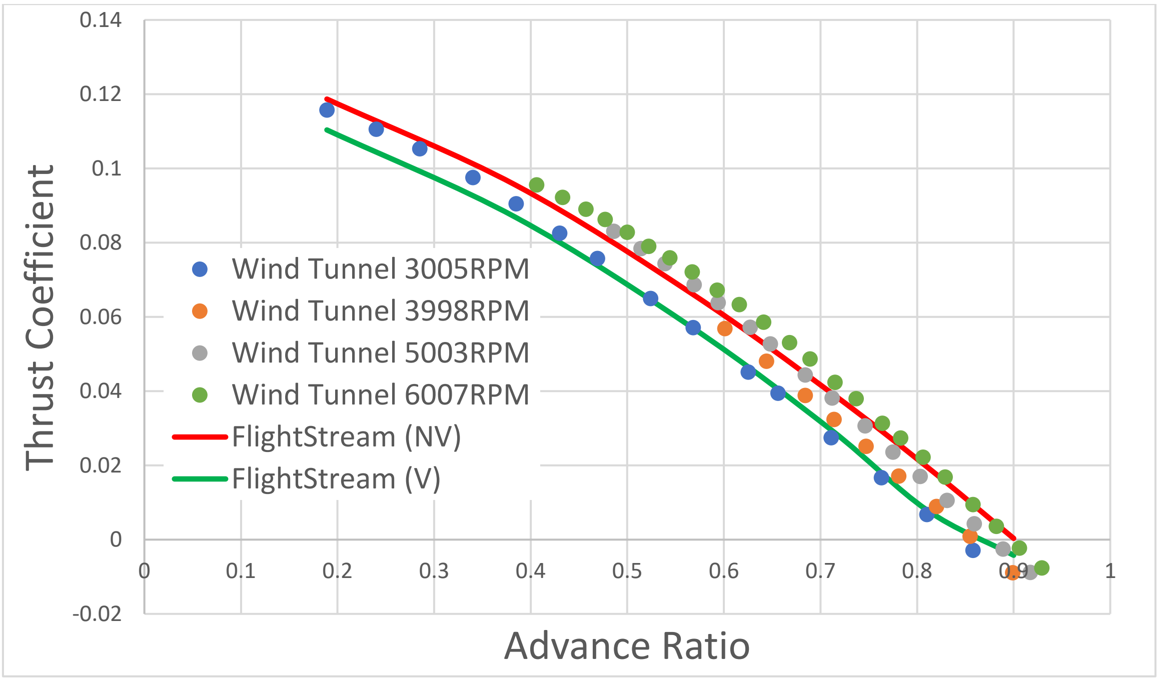


Figure 21: GF 10x8 Thrust Coefficient

labeled with “(V)”. Figure 21 provides the thrust coefficient results for the GF 10x8 propeller. As shown in the plot, FlightStream® captures the behavior of the propeller extremely well with both the viscous coupling model and model without viscous coupling. The model that does not have the viscous coupling setting on intuitively predicts higher values for the thrust than the viscous model does, but from the UIUC data the model that does not have the viscous coupling accounted for is the more accurate of the two. Nevertheless, both models are placed within the scatter or very near the scatter of the wind tunnel data for the thrust coefficient.

The power coefficient is presented next in Figure 22. The performance from both models is grouped tighter together for the power coefficient than the thrust coefficient which will lead to discrepancies in the efficiency plots. Furthermore, both models exhibit more linear behaviors than the actual wind tunnel data suggests. Causes for this could be explained by the complicated geometry build process and possible errors in scanned point clouds. Despite these irregularities, the data for power coefficient is matched well by the numerical solver. At the lower advance ratios

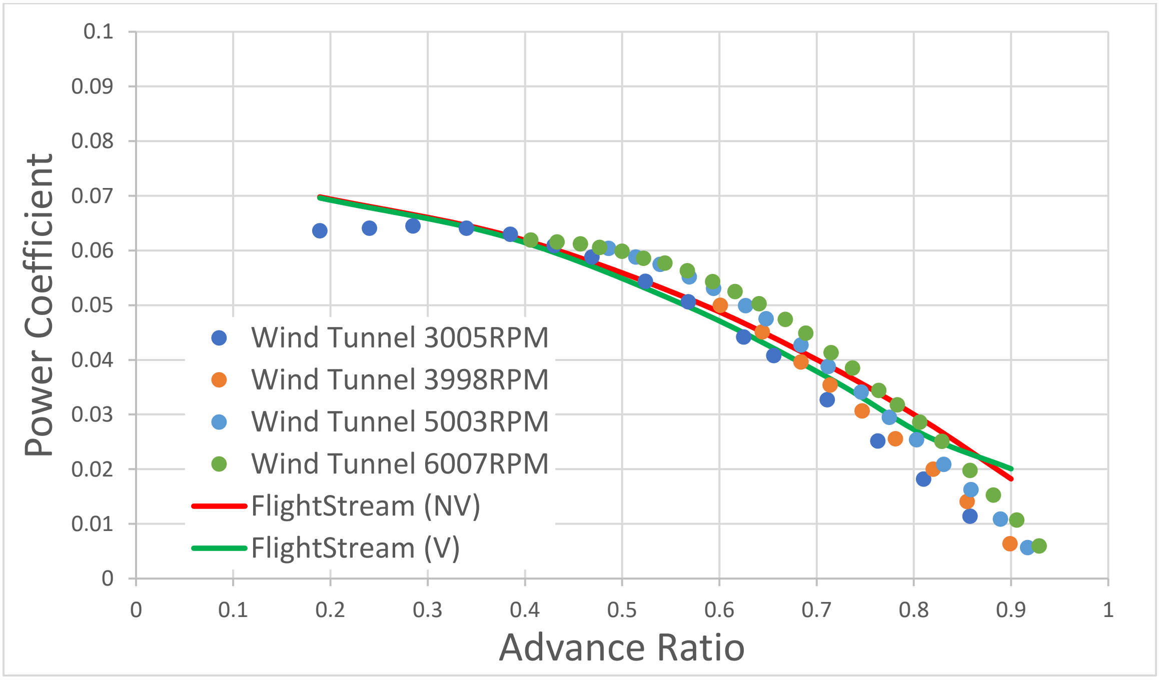


Figure 22: GF 10x8 Power Coefficient

of 0.25 and less and the higher advance ratios just before windmilling occurs, there are over predictions of about 7%. At any other advance ratio between those values, the predictions by the solver are within the scatter of the wind tunnel data.

Lastly, the plot for the efficiency of the GF 10x8 propeller is presented. Due the to the differences in the thrust prediction between the two models and then the similarities in the power coefficients, the difference in efficiency plots between the viscous and non-viscous model are quite large, but the same difference in performance data is exhibited in the wind tunnel data. The scatter

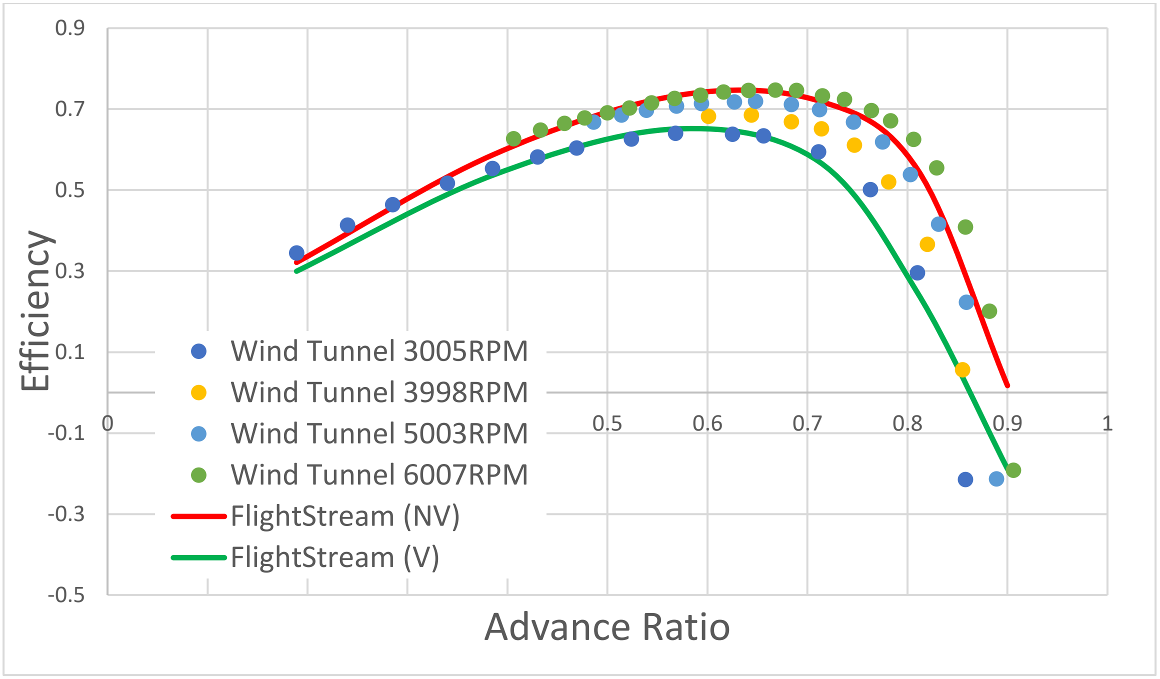


Figure 23: GF 10x8 Efficiency

for thrust and power from the wind tunnel leads to even greater scatter in the efficiency plots. Both models fit within the provided range of efficiency at the given advance ratios. The viscous model provides a lower prediction for efficiency than does the non-viscous coupling model which is sound. The viscous coupling model even under predicts efficiency compared to the wind tunnel data. This, again, could be due to errors in the geometry or perhaps a minor over prediction in the torque on the model in the solver itself. However, it is appropriate to conclude that the solver is validated for the GF 10x8 propeller. Plots for thrust and power coefficient place results within the upper and lower ranges of RPM for the wind tunnel data, and most importantly the efficiency predicted by the solver is accurate.

The Electric 11x7 propeller is presented with similar characteristics to that of the previous propeller. Figure 24 provides the thrust coefficient for the E11x7 propeller from the numerical solver. Here, the viscous coupling model is very similar to the non-viscous coupling model as the

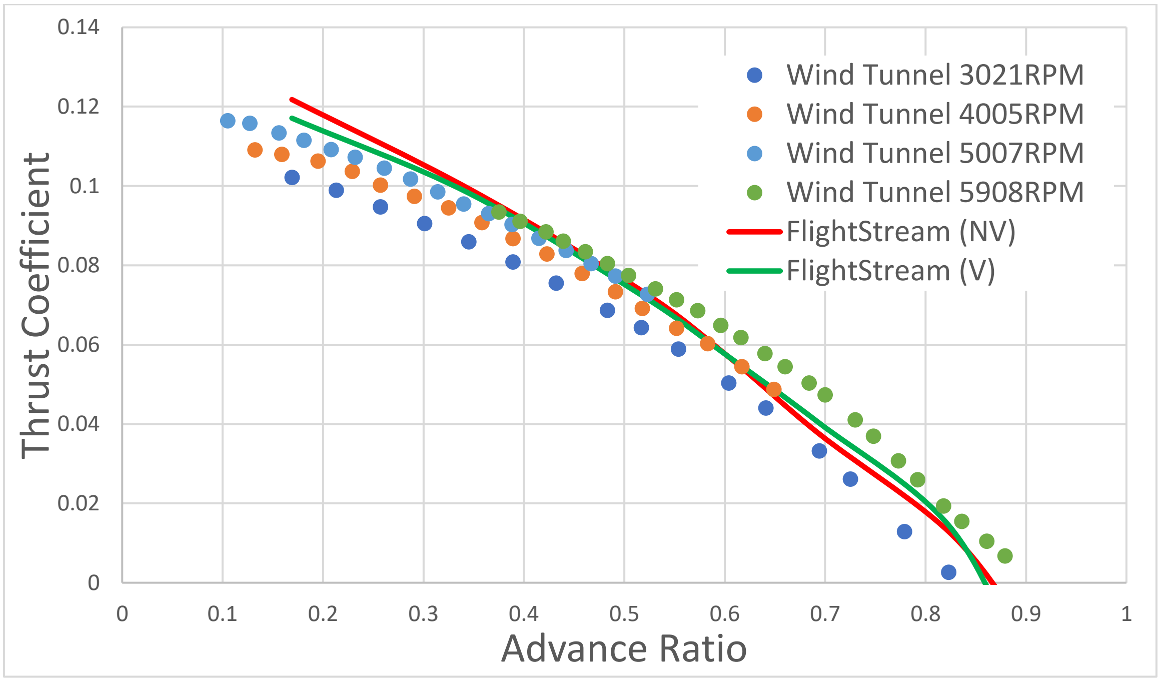


Figure 24: E11x7 Thrust Coefficient

plots for both are placed almost directly on each other. While the curve predicted by the numerical solver tends to be more linear than the provided data, both models stay within the variance of the wind tunnel data other than a slight over prediction in the lower advance ratios.

The power coefficient predictions for the E11x7 propeller are provided in Figure 25 which are far more accurate than that of the GF10x8. The numerical solver provides a clean non-linear curve for the power predictions. Although the spread for the wind tunnel data is quite large for the given advance ratios, FlightStream® produced results are consistently placed in the center of the scattered data. One oddity noticed form the plots, however, is the predictions for the viscous coupling model and the non-viscous coupling model. It should be assumed that the viscous coupling model would produce higher predictions in the torque on the propeller blades and ultimately the power required, but here the opposite is shown. One cause for this is a non-

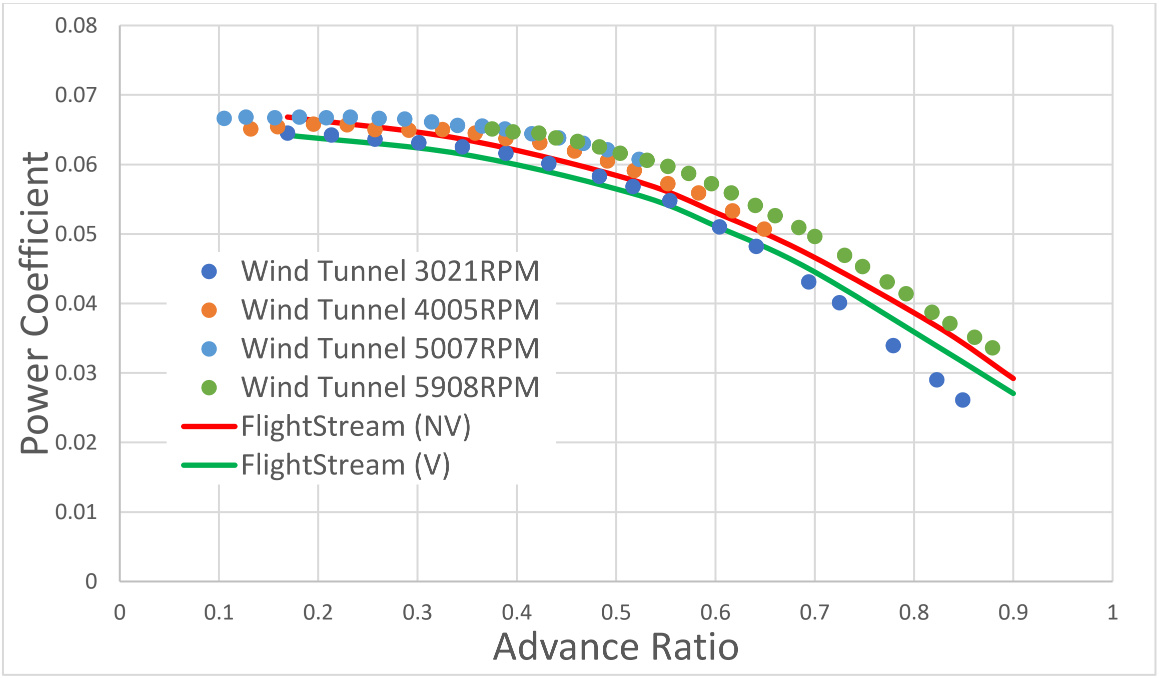


Figure 25: E11x7 Power Coefficient

converging solution in the solver or perhaps a faulty geometry from the scan. Still, the fact of the matter remains that the predictions for both models are accurate and are placed in the range of the provided data.

Lastly, the efficiency plots for the E11x7 are provided in Figure 26. Both models are the propeller remain in the range of the UIUC data. Because the results for the thrust coefficient in Figure 24 where so similar the main defining characteristics between the viscous coupling model and the non-viscous coupling model are rooted the power coefficient. Therefore, the plot follows almost the exact same behavior as the power coefficient in that the viscous model produces greater efficiencies where it would be assumed that the opposite would happen. Again, the issues may remain in the solver not converging in the provided number of iterations or the geometry of the propeller itself. In reality, this is a miniscule error because the results for both are still very accurate.

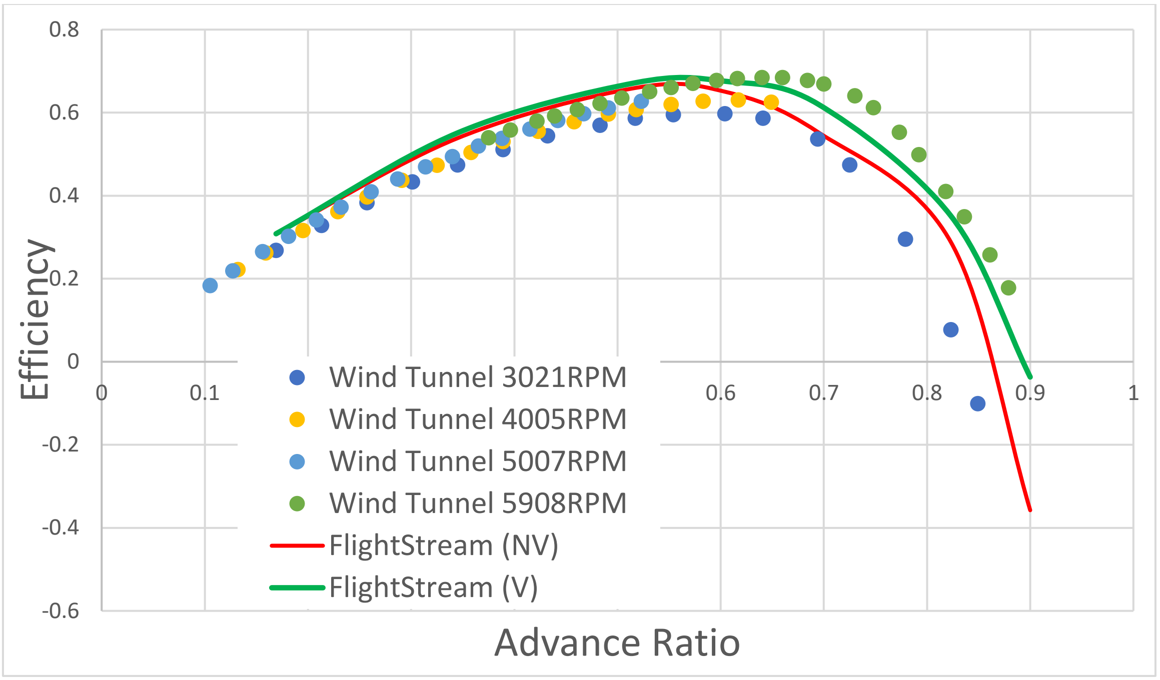


Figure 26: E11x7 Efficiency

Data from the University of Illinois propeller data base has been collected and implemented into an OpenVSP model which in turn provided the meshed geometries for the solver. The numerical solver, FlightStream®, was then validated over advance ratios for the thrust coefficient, power coefficient, and efficiency. All results produced by the solver follow the necessary trends and are placed directly in the scatter of the wind tunnel data or within a reasonable error.

A simple analysis was also conducted on comparisons between the same E11x7 propeller with and without a spinner or hub in the center of the propeller. This analysis was carried out to validate the methods for the optimization derived geometries because the propellers in the GA do not have spinners. The thrust coefficient, power coefficient, and efficiency for both propellers are shown in Figure 27, Figure 28, and Figure 29. The torque loads on the blade remain relatively constant between the two case which leads to power coefficient being almost identical. The main discrepancy in the results arises in the thrust coefficient predictions which are understandable. The spinner provides only structural support to the propeller as it is what holds the blades themselves. It provides no thrust and therefore can only increase drag on the propeller, effectively reducing the

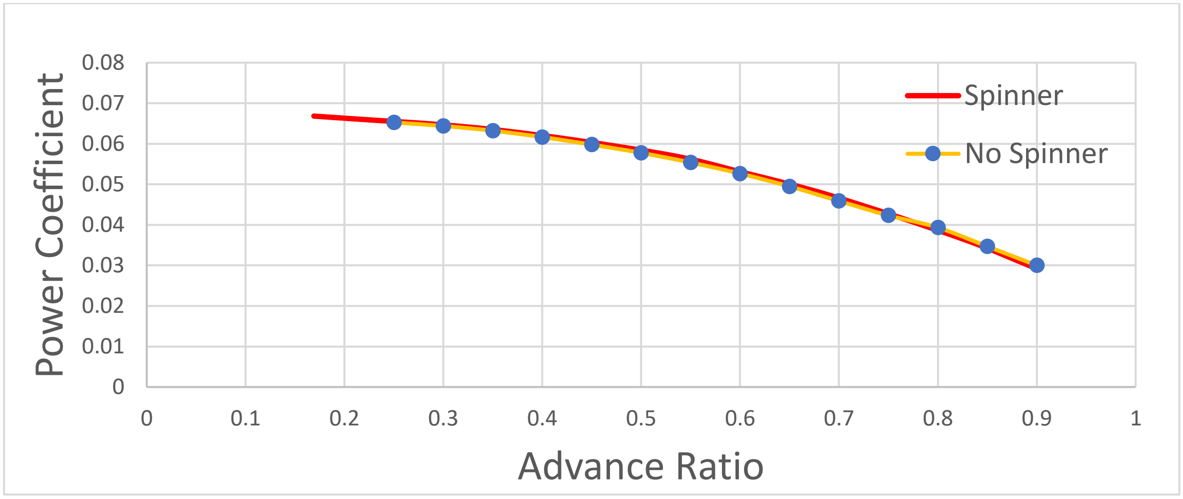


Figure 27: Spinner vs No Spinner Power Coefficient

thrust. Thus, it is expected for the thrust to be slightly over predicted for the no-spinner model. Due to the increased thrust and similar power requirements, the no-spinner model has an increase

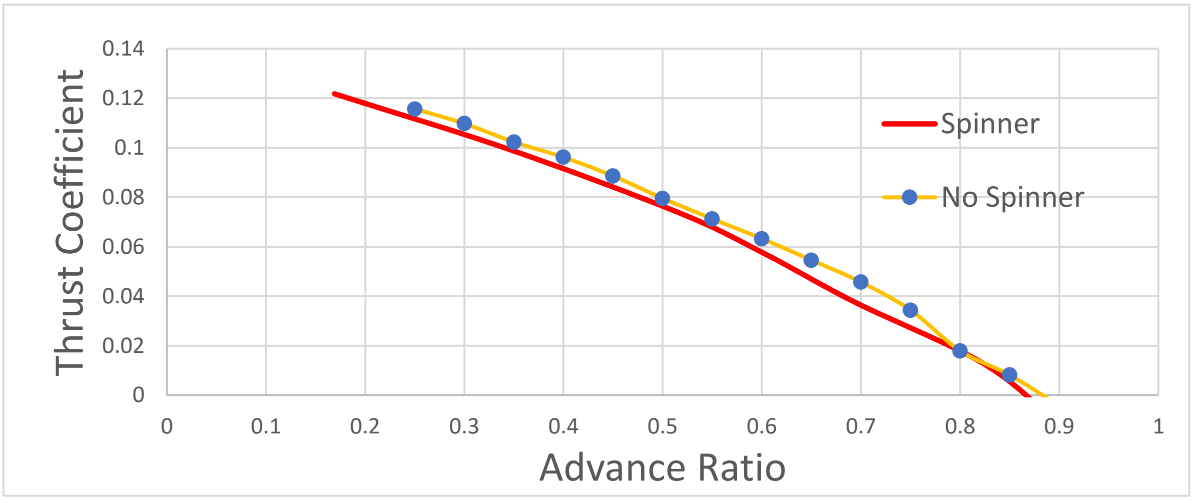


Figure 28: Spinner vs No Spinner Thrust Coefficient

in the efficiency from the model with the spinner. While there are discrepancies between the spinner and no spinner analysis, the trends between the two models are identical in nature. Therefore, it is reasonable to use a geometry model in the GA that does not have a spinner. This

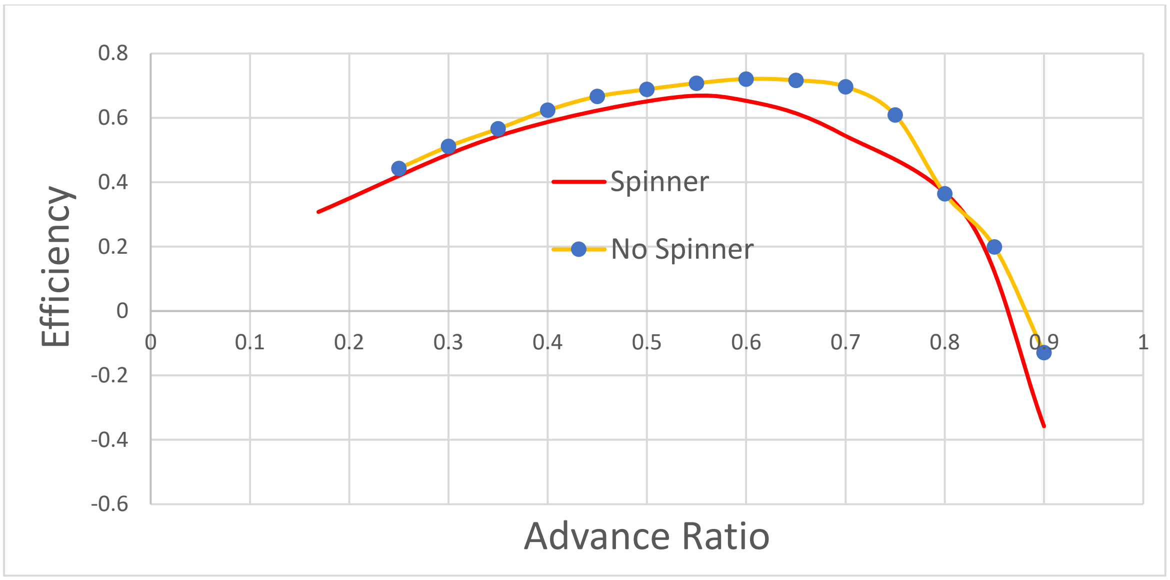


Figure 29: Spinner vs No Spinner Efficiency

saves in computational time because there are less mesh faces to account for. Furthermore, it allows room for more error in the geometry designs and modeling in FlightStream®. As these models are loaded into the numerical solver, a mesh is created across the cross sections that are loaded in. If a spinner is to be meshed with the propeller as well, it is likely that more errors will arise in the models leading to non-converging solutions which provide the GA with no useful information and take longer to run in the solver. The conclusion is that producing models with no spinner is far more practical for blade optimization purposes. Lastly, the the optimization process described in this work is a preliminary design tool and therefore results are not required to be finalized but rather to provide further insight to the development process.

5.2 NACA Technical Report

The NACA TR-640 provides extensive wind tunnel performance data as well as the necessary information for reconstructing the propellers. The propellers provided in this report are 10 feet in diameter. A total of three groups of propellers are tested. The first group has Clark Y airfoil sections and consist of 2, 3 and 4 bladed propellers. The second group is the same as the first except an R.A.F. 6 airfoil composes the cross section. The last group consists of a single two bladed propeller with an R.A.F. 6 airfoil, but the chord is 50% greater than the two bladed propeller in group 2. The geometry data provided is shown in Figure 30. Because the propellers are very similar, only one group is studied in this validation. The R.A.F. 6 propellers are studied for the 2, 3, and 4 bladed designs. These three propellers are studied with the pitch at 15 degrees at 75% of the radius as well as one model where the pitch is increased to 25 degrees at 75% of the radius providing a total of 4 propellers in this validation. One notable aspect of the geometry is the thickness of the propeller blade. The chord, airfoil shape, and thickness of the blade were all provided which would lead the propeller geometry to be over constrained. The airfoil shape is

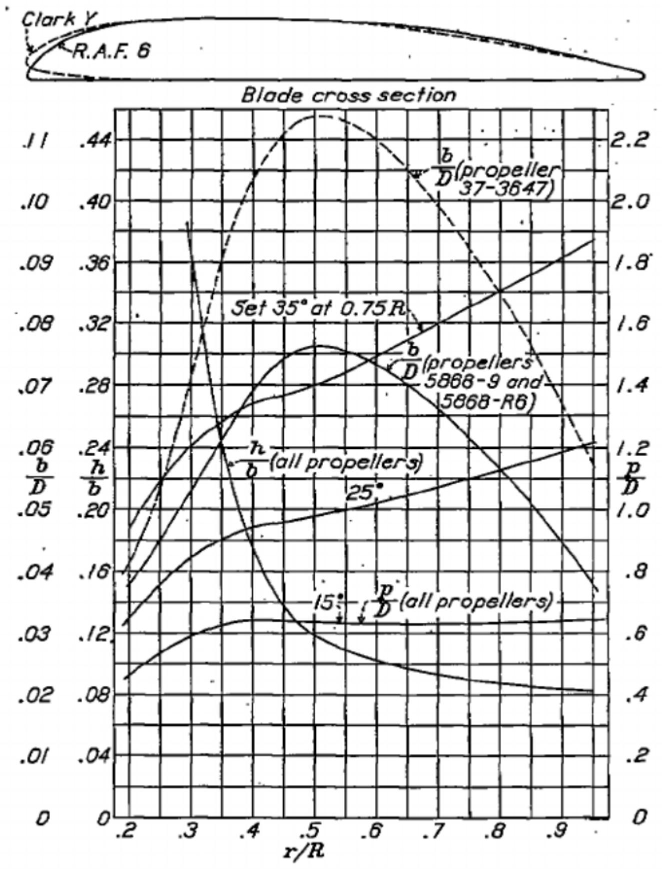


Figure 30: NACA TR-640 Example Data

normalized by the chord, and therefore, when a chord and airfoil are given, the full geometry is provided. Appling anymore constraints regarding the cross section would lead to changing the airfoil itself. Thus, two models were created for the three-bladed model: one that uses the airfoil and chord only, and on that uses the airfoil, chord, and thickness. The geometry model which only uses the chord and airfoil shape is named “Airfoil” in the plots, and the model which uses an adjusted thickness is named “Adjusted”. Furthermore, each of these models was tested with and without the viscous coupling enabled. The viscous coupling models are indicated by “(V)”, and the models which do not use viscous coupling are indicated by “(NV)” after the geometry identification name.

The three bladed propeller with 15 degrees of pitch at 75% of the radius thrust coefficient is provided in Figure 31. The viscous coupling and non-viscous coupling models provide practically the same results as both plots lie directly on top of each other. The model which uses the adjusted thickness is thicker than the airfoil modeled thickness which leads to the over

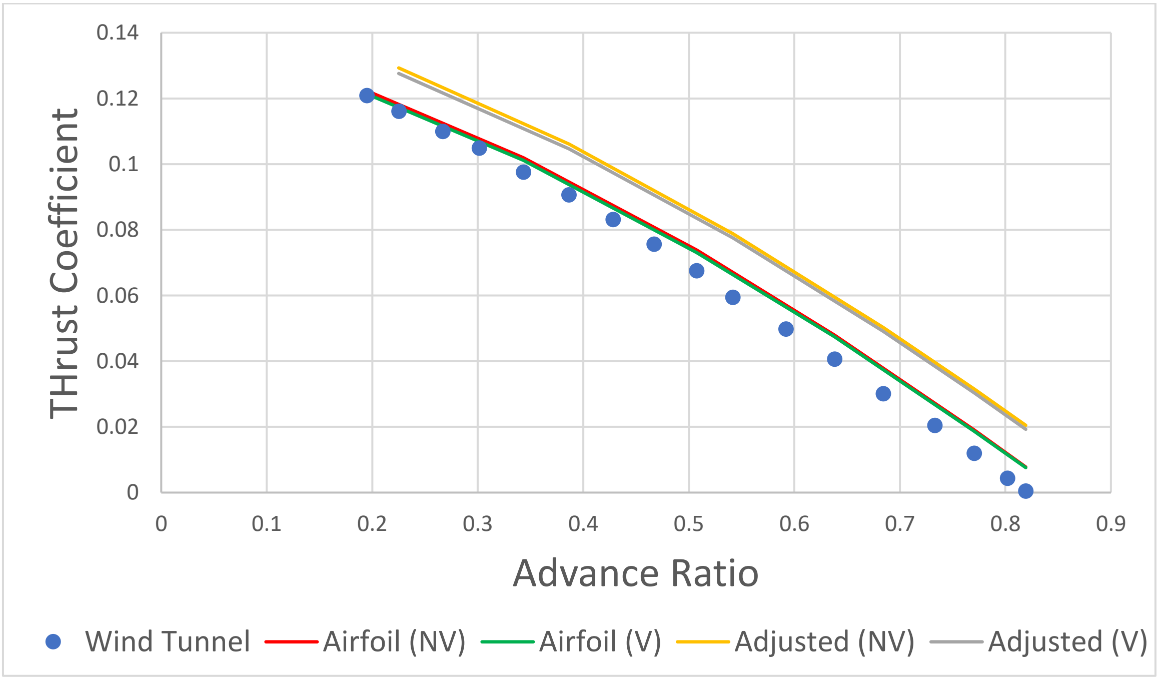


Figure 31: 3-Bladed 15° Pitch at 75% Radius Thrust Coefficient

predictions in the thrust coefficient. The models which use the airfoil captured thickness provide much more accurate data with respect to the NACA 640 report. The same trend is seen in the results for the power coefficient in Figure 32. Here the data is only provided until the thrust becomes negative at an advance ratio of ~0.81. This is because the GA only searches for propellers

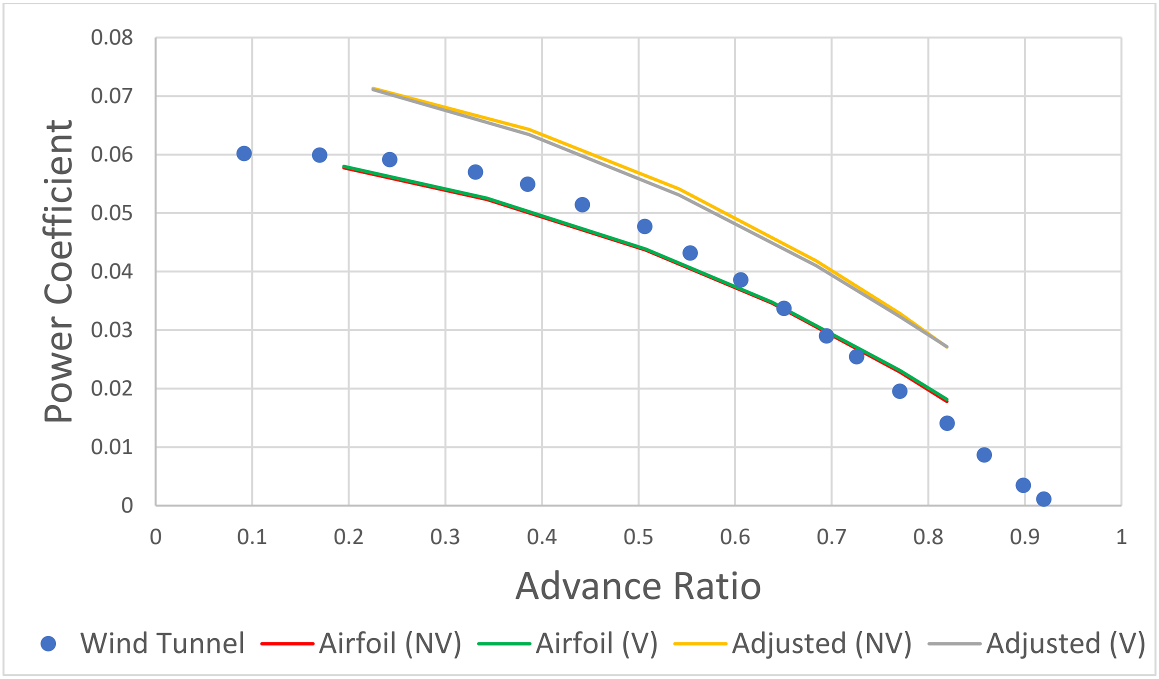


Figure 32: 3-Bladed 15° Pitch at 75% Radius Power Coefficient

that are producing a positive thrust. Thus, there is no need to validate the flight condition. Both the airfoil thickness model and the adjusted thickness model provide the same trends as the wind tunnel data; however, the airfoil thickness model provides much more accurate results with some under predictions at lower advance ratios. The adjusted thickness model has an increased thickness radially outward along the propeller. This leads to the over prediction in the torque calculations which is reflected in the power coefficient. Although both models produce different results for thrust and power coefficients, the efficiency for both is similar due to the consistent over prediction by the adjusted thickness model. Figure 33 provides the efficiency for both geometry models. It is shown that the airfoil modeled thickness provides a better representation of the efficiency curve as

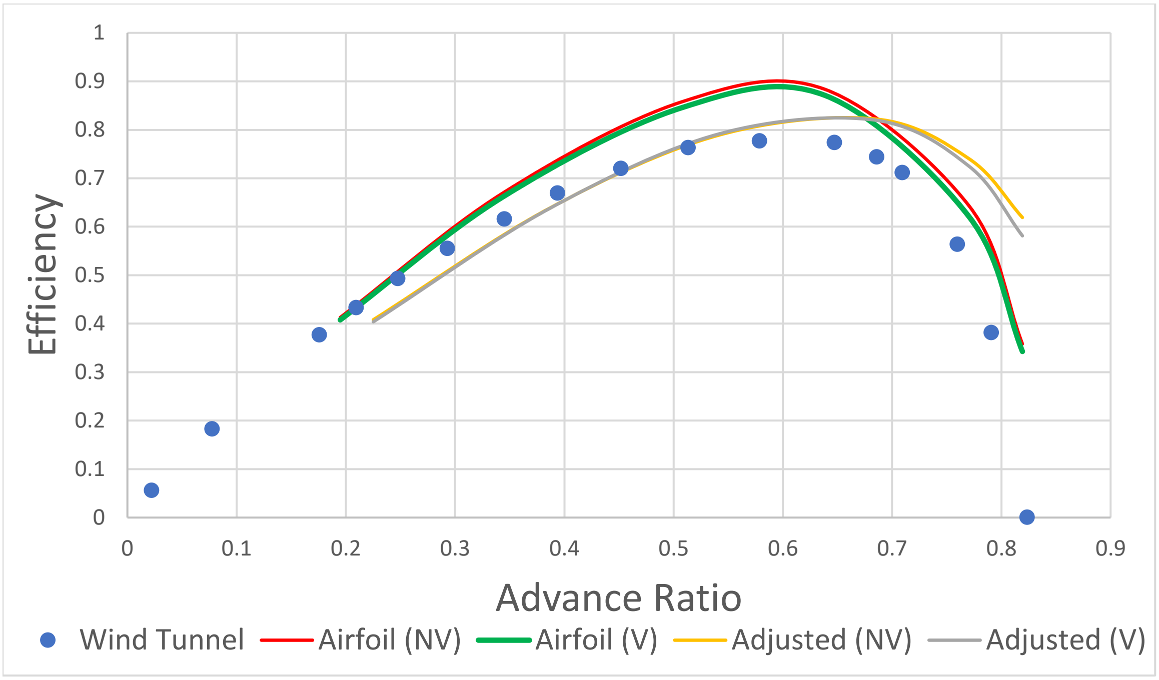


Figure 33: 3-Bladed 15° Pitch at 75% Radius Efficiency

it increases and then experiences a sharp decline at an advance ratio of 0.6. The adjusted thickness model over predicts the location of this maximum followed by a drop off; in this model, the over increasing efficiency extends to an advance ratio of 0.7. Ultimately, the airfoil modeled thickness provides better results regarding the wind tunnel data produced from the NACA TR-640. Therefore, for the remainder of this validation, the airfoil captured thickness models were used.

The two-bladed propeller with 15-degree pitch is presented next. The results from this validation are similar to the three-bladed model largely because it is the same blade. Figure 34 provides the results for thrust coefficient from the 2 blade propeller simulations. For both the viscous coupling model and the non-viscous coupling model, the thrust coefficient predicted by the numerical solver is identical to that of the wind tunnel. Other than slight over predictions at higher advance ratios, the numerical solver data is an exact match. The power coefficient data,

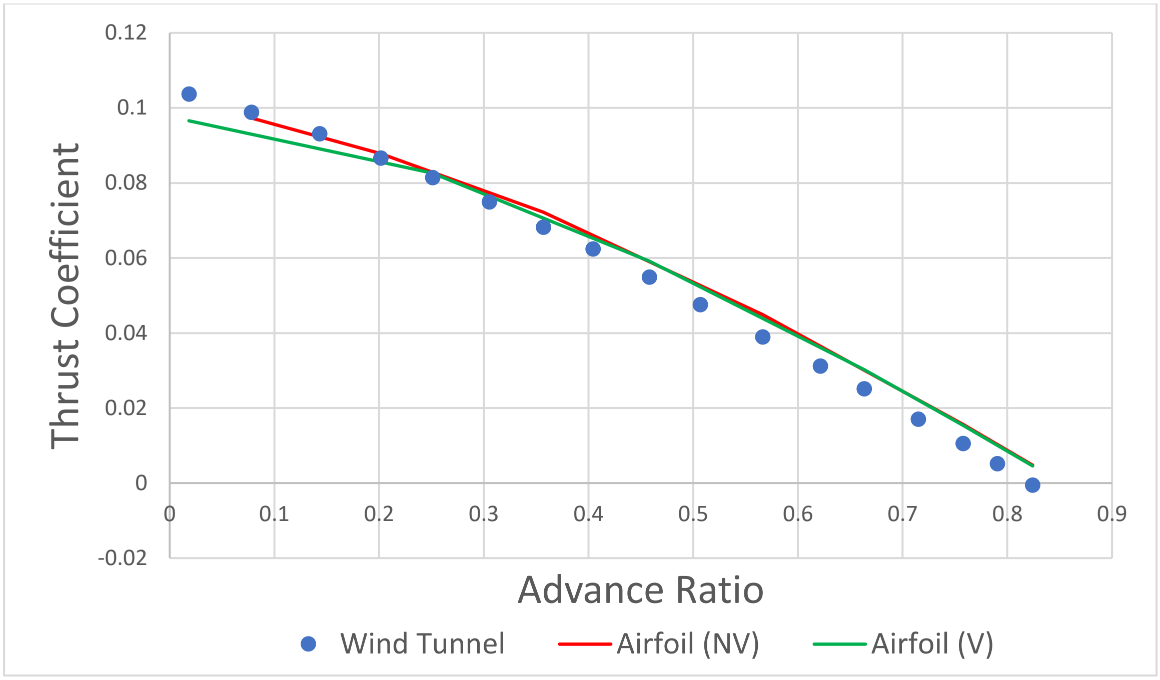


Figure 34: 2-Bladed 15° Pitch at 75% Radius Thrust Coefficient

however, does not provide nearly as accurate results as the thrust coefficient modeling, but the power coefficient performance is still valid. Figure 35 provides the power coefficient data for this

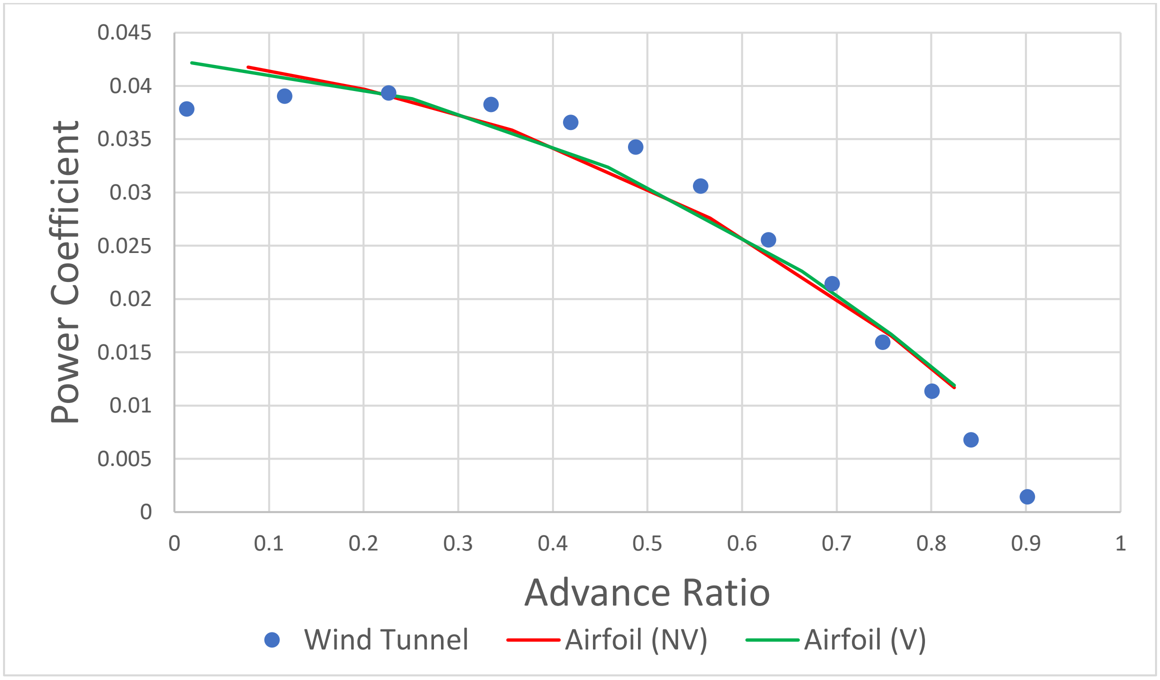


Figure 35: 2-Bladed 15° Pitch at 75% Radius Power Coefficient

propeller and demonstrates a more linear curve than is shown by the wind tunnel data. This linearity is exhibited in both the viscous and non-viscous coupling models. Although the data does not experience an exact match, the general location of the data points is very similar to what was

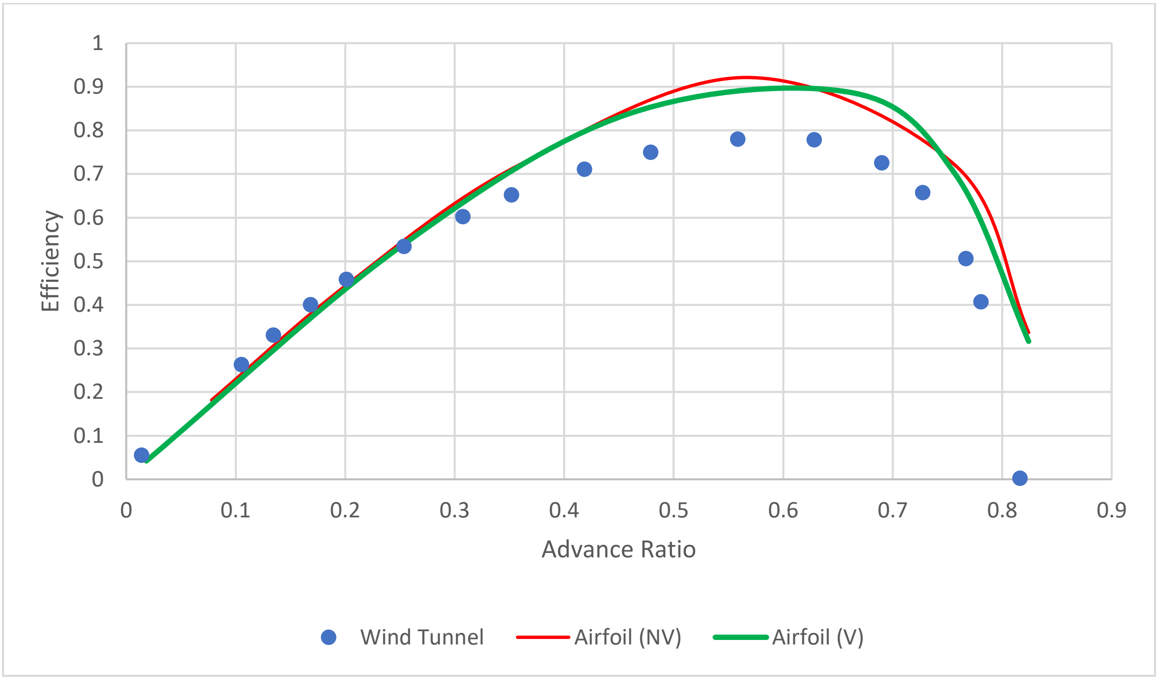


Figure 36: 2-Bladed 15° Pitch at 75% Radius Efficiency

expected form the wind tunnel. The largest error in prediction arises between the advance ratios of 0.3 and 0.6, and this under prediction is shown in the efficiency plots as the efficiency for the

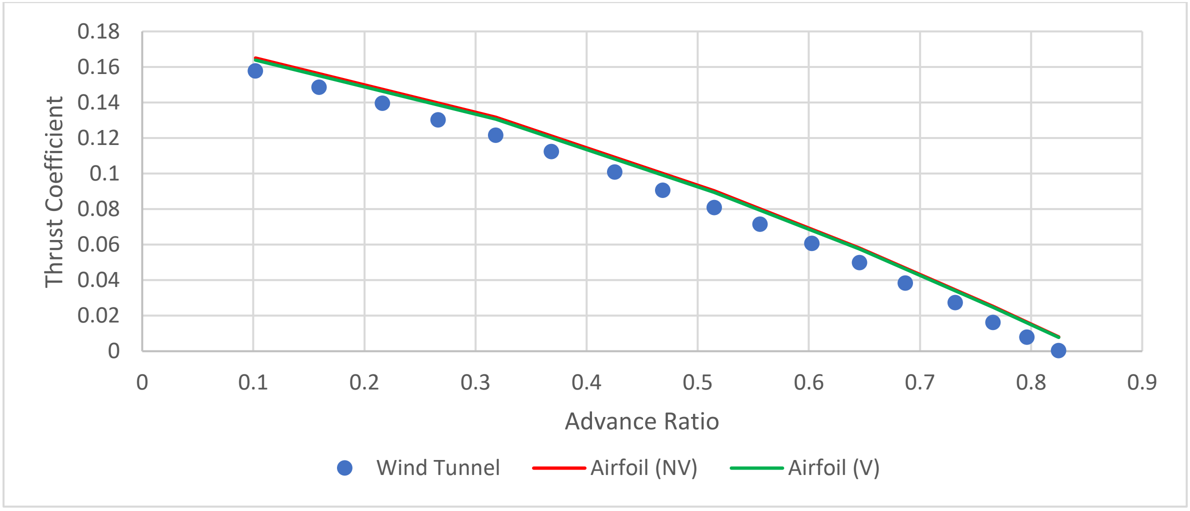


Figure 37: 4-Bladed 15° Pitch at 75% Radius Thrust Coefficient

propeller is over predicted in that range. The plot of propeller efficiency for the 2-bladed propeller with 15-degree pitch at 75% the radius is provided in Figure 36. At the lower advance ratios, the efficiency predictions are identical to the wind tunnel performance; however, as the advance ratios increase the numerical solver over predicts the efficiency of the propeller. Despite this over prediction, the trend of the efficiency vs advance ratio is captured. Furthermore, the plots from the solver remain near to that of the wind tunnel. One important take away from this modeling is the lack of difference in the viscous coupling and non-viscous coupling models. One reason for this is the operating condition for these propellers which will be discussed in the following section.

The four-bladed propeller with the same pitch as the first two propellers is provided next. As seen before the thrust coefficient in Figure 37 has slight over predictions but remains very accurate in terms of the overall trends provided by the solver and the values of the plot itself. The

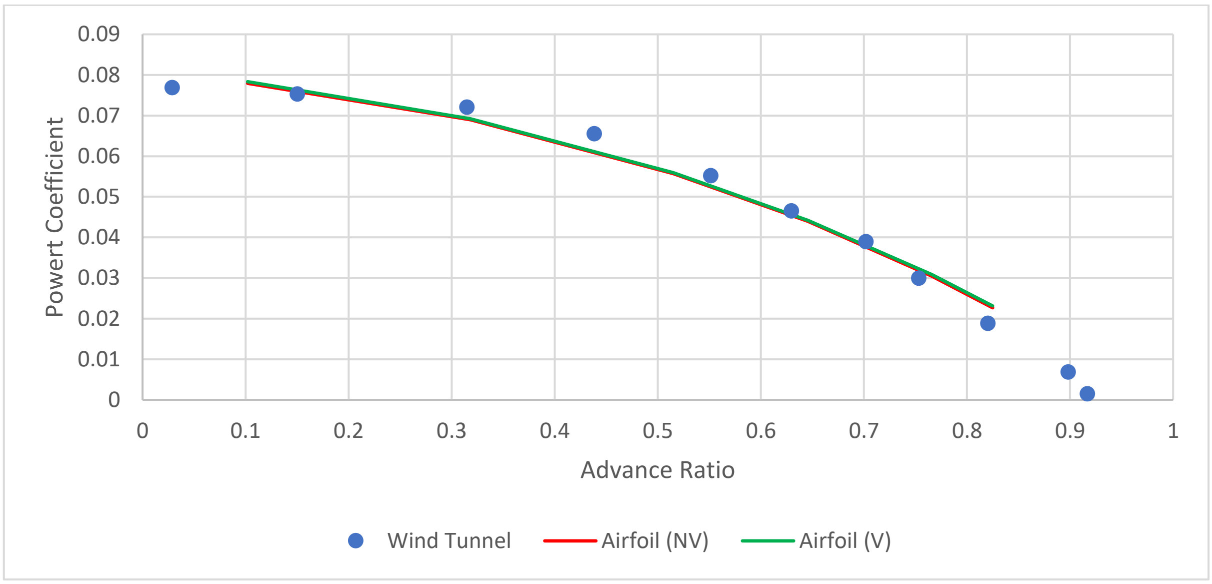


Figure 38: 4-Bladed 15° Pitch at 75% Radius Power Coefficient

same behavior as the two and three bladed propeller is also seen for the power coefficient given in Figure 38. Accurate values are provided at the lower and higher advance ratios but from advance ratios of 0.3 to 0.6 the power is under predicted. Furthermore, the efficiency plot for the four-bladed propeller, provided in Figure 39, is the same as the efficiency plots for the 2 and 3 bladed propellers where the solver provides the appropriate trend to the plot and accurate values for lower

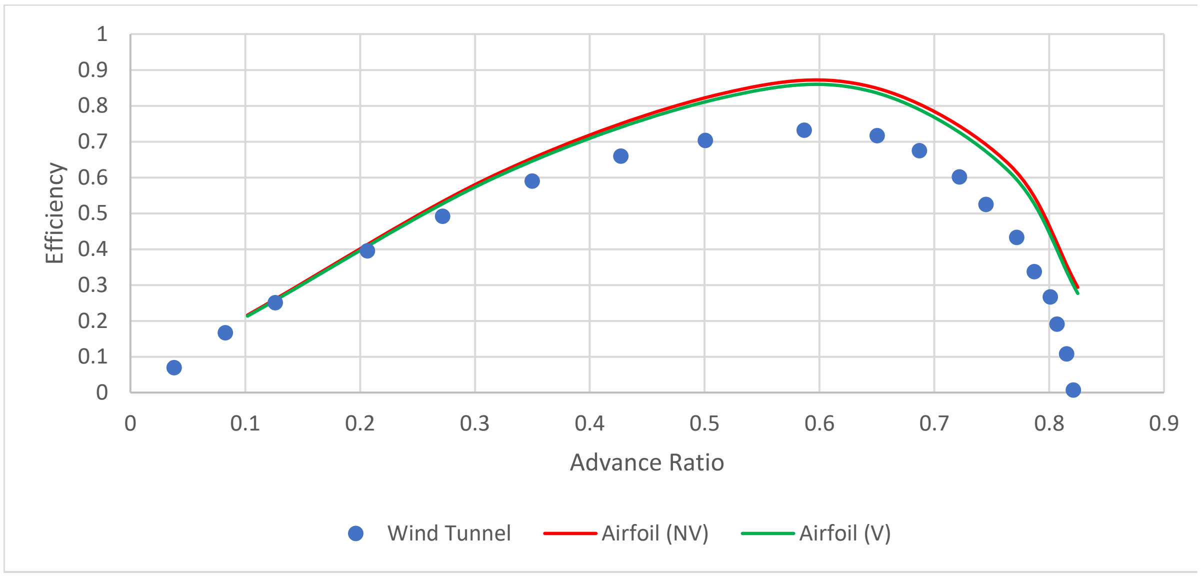


Figure 39: 4-Bladed 15° Pitch at 75% Radius Efficiency

advance ratios and an over prediction for the higher advance ratios. Therefore, it is concluded that the solver is validated for the NACA TR-640 propellers with various numbers of blades.

Lastly, the propeller with a different pitch is provided for the validation to ensure that the solver offers a full range of propeller modeling capabilities. Here the three-bladed propeller is twisted by 10 degrees to offer 25 degrees of pitch at 75% of the radius. The thrust coefficient for this model is provided in Figure 40. Again, the same trend is shown for the propeller where the thrust coefficient is slightly over predicted in the range of advance ratios and remains accurate and within a reasonable range of the provided data. The main difference between this propeller and the others previously validated is the power calculations which are given in Figure 41. While accurate results are provided at the higher advance ratios, the solver over predicts torque values at the lower advance ratios. The cause for this is rooted in the steady rotary solver itself. At lower advance ratios, the steady rotary solver experiences difficulties in converging. For these flight conditions the unsteady solver must be used, but for this case, only the steady rotary solver needs to be validated at the advance ratios provided to the GA which are well above these values which have

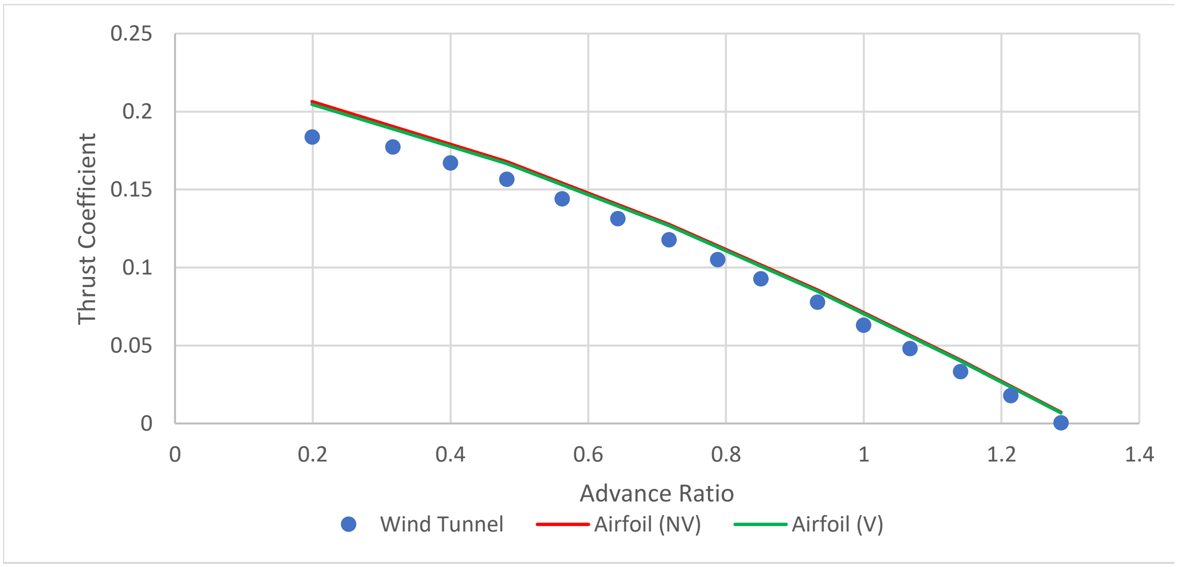


Figure 40: 3-Bladed 25° Pitch at 75% Radius Thrust Coefficient

error. Although, there are large errors at lower advance ratios for power predictions, the efficiency of the propeller is well matched by the solver in Figure 42. While there are some over predictions

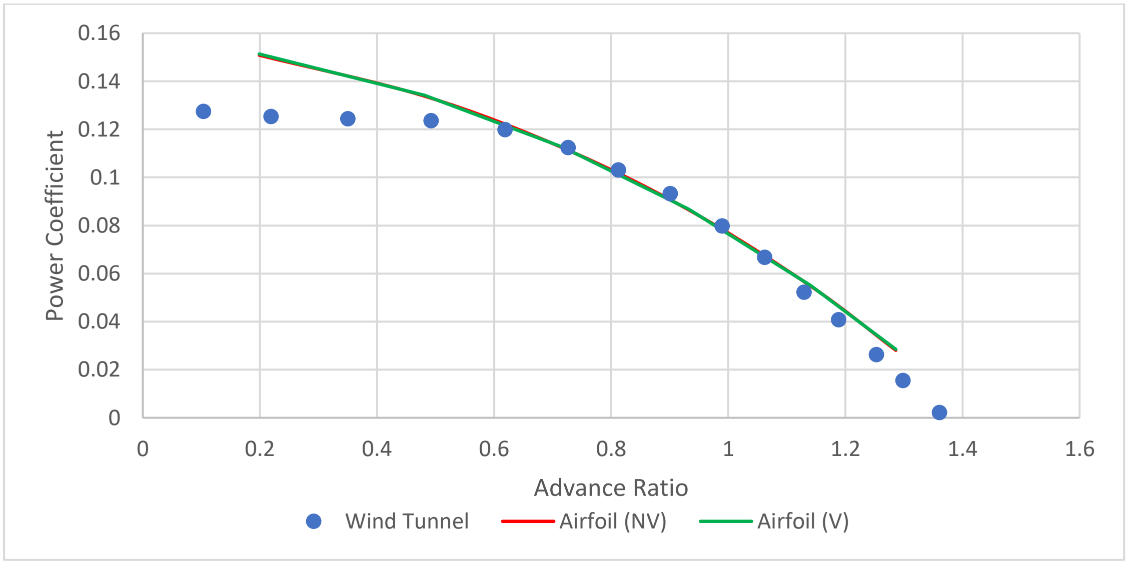


Figure 41: 3-Bladed 25° Pitch at 75% Radius Power Coefficient

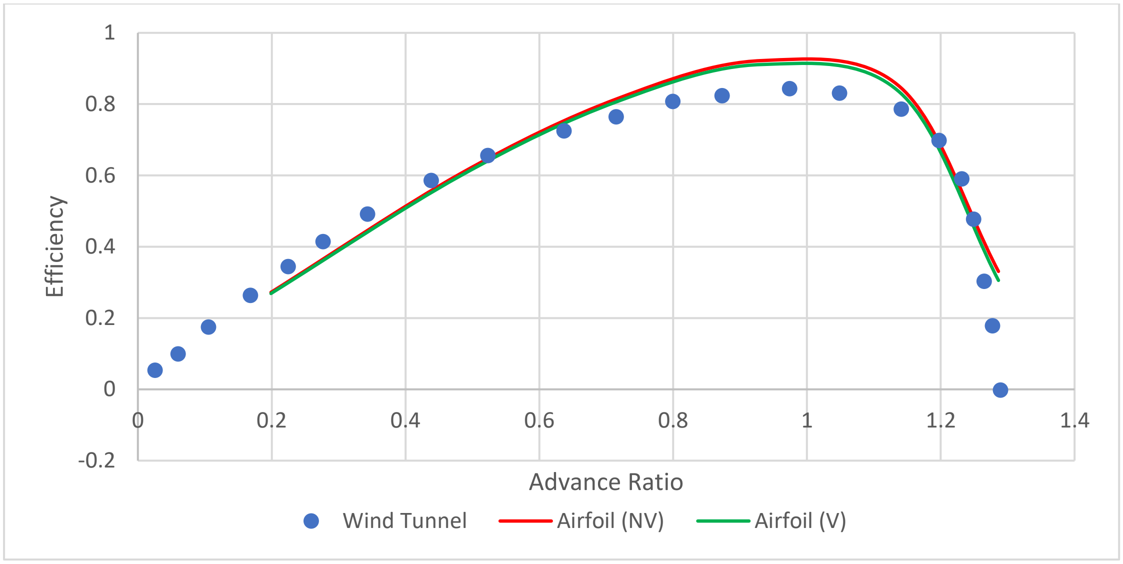


Figure 42: 3-Bladed 25° Pitch at 75% Radius Efficiency

for maximum efficiency, the solver predicts the drop off for efficiency and provides a correctly modeled trend for the curve with respect to the provided wind tunnel data.

The propellers from the NACA TR-640 report have been used to validate the numerical solver that is implemented in the Genetic Algorithm. The solver has been shown to capture the effects of adding blades to a propeller, adjusting the pitch and other general characteristics. The propellers used in this validation used an R.A.F. 6 propeller with 2, 3, and 4 blades. A fourth model was also added to the group of propellers which had 3 blades but had a pitch which was increased by 10-degrees. The solver demonstrated appropriate modeling and accuracy for each propeller.

5.3 Flow Separation Modeling

One notable characteristic between the previous two cases for propeller validation is the effects of including or not including the viscous coupling models for the solver validation. In section 5.1, the propellers from the UIUC Propeller Database experienced large differences between the cases when the flow viscous coupling was enabled and when it was not. This is due to the amounts of flow separation that occur on the top of the blade. This analysis of flow separation and difference in viscous coupling models is indeed important to the performance modeling for the GA. Because the viscous coupling setting takes longer for the solutions to converge, it is not used in the modeling for the GA; however, the flow separation is still accounted for by a 7% rule. As demonstrated by the previous validation case which uses the NACA TR-640 propellers, some solutions do not need to have the viscous coupling effects enabled as the solution

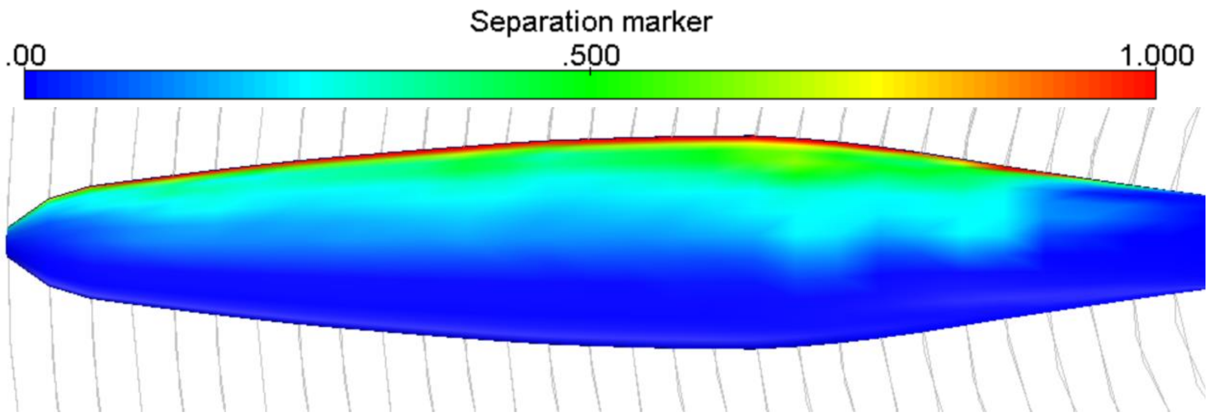


Figure 43: 3-Bladed NACA Propeller Flow Separation at an Advance Ratio of 0.5

converges on the same results. These cases consist of propellers with very low amounts of flow separation. Figure 43 provides the separation marker across the three-bladed NACA TR-640 propeller with the pitch at 15 degrees at 75% of the radius. A separation marker of 1 indicates fully separated flow, and a separation marker of 0 indicates fully attached flow. Notice that there is almost no separation across the propeller. Therefore, whether the viscous coupling is turned on or off, it will not matter, only slight adjustments in thrust and torque will occur between the two models. This is the reason for the similarity in the models from the NACA report. When the separated flow becomes the dominating flow regime over the propeller, the viscous coupling effects are very important to capture. Figure 44 provides the separation for the GF 10x8 propeller.

Notice the large increase in the amount of separation. This separation is what causes the drastic difference in the results between the viscous coupling and non-viscous coupling cases.

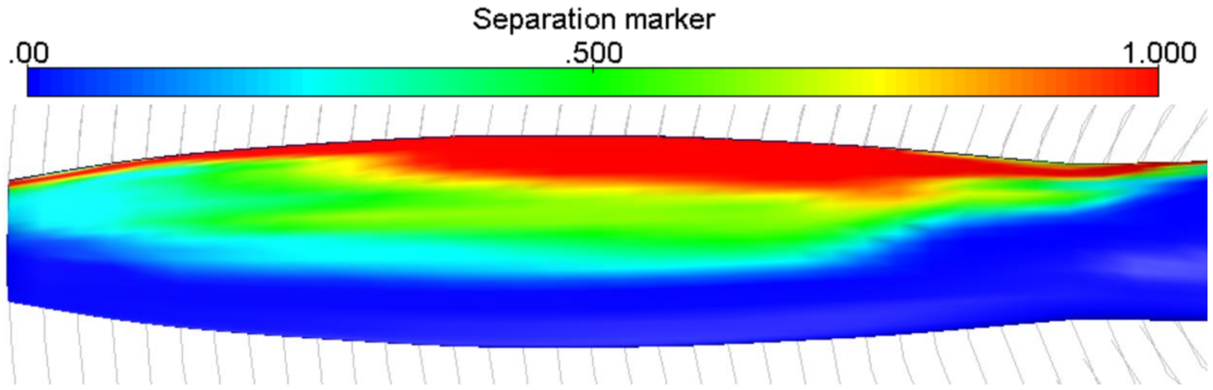


Figure 44: GF 10x8 Flow Separation at an Advance Ratio of 0.5

Now that the reason for the difference in the cases is identified, a shortcut can be implemented in the optimizer to provide shorter run times. If there is no difference between the viscous coupling models and the non-viscous coupling models for conditions in which the flow is mostly attached, then only propellers with fully attached flow should be considered, and the viscous coupling mode should be turned off to provide faster run times. Furthermore, this analysis is supported that this is an optimization tool. Therefore, it should only consider optimal propellers for the solutions, and propellers with more separation are less optimal than propellers with less separation. Thus, a marker is set in the GA which indicates propellers with overloaded amounts of separation. When a propeller with more than 5% of the flow has separation markers of 90% or greater, the propeller is given a fitness of -10. This eliminates the propeller from being a possible candidate for the best propellers which pass their characteristics on to the next generation. This provides faster run times, and it only allows the GA to consider more optimal propellers.

5.4 Coaxial Propeller Validation

The coaxial propellers were validated in the solver for thrust and torque loads. The geometry for the propeller comes from a scan of an MR9x4.5 propeller. The scan was conducted in the same manner as the UIUC propellers; however, in the case of this geometry construction,

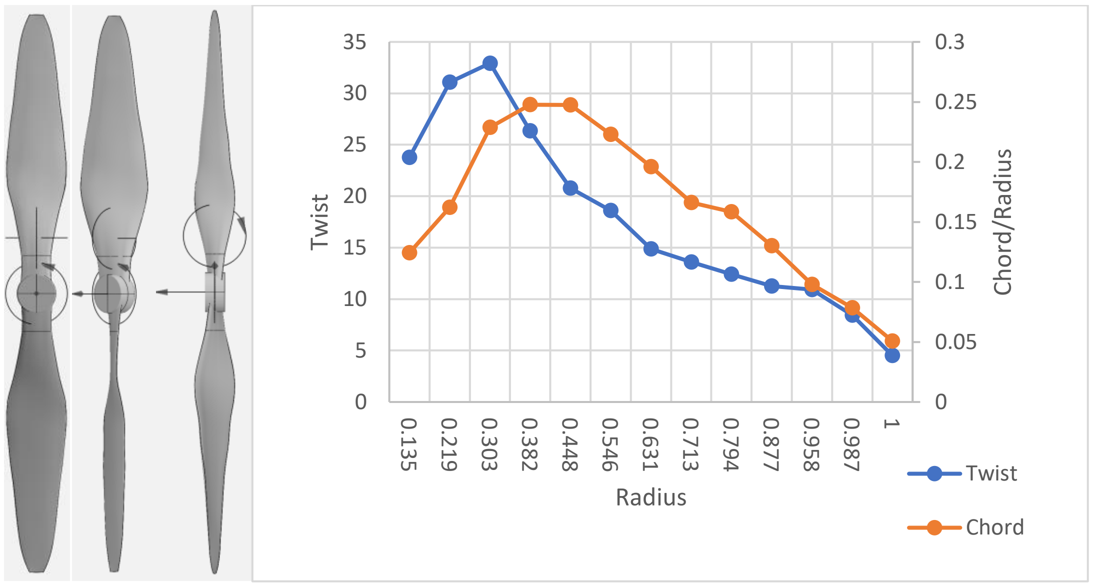


Figure 45: MR9x4.5 Geometry

the entire geometry was gathered via scanning whereas the UIUC scans only collected airfoil shape. An example of the geometry that was constructed in the scan is provided in Figure 45, and

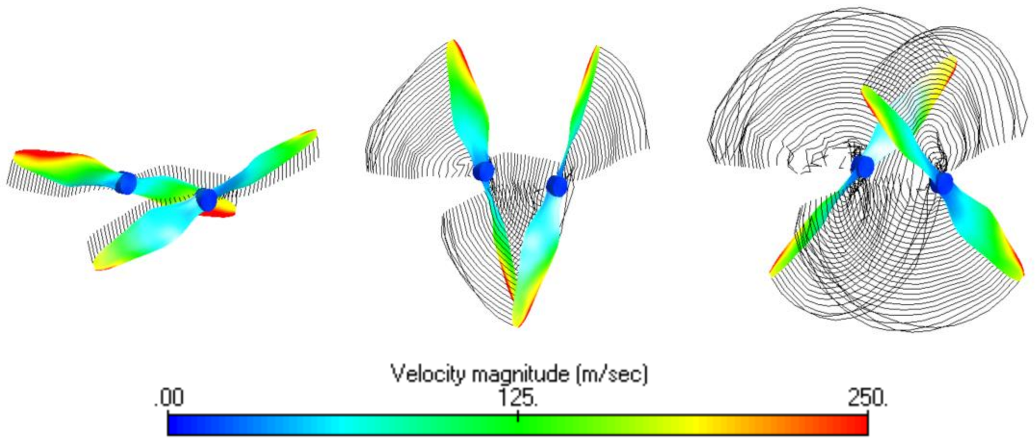


Figure 46: Coaxial Propellers at Various Points in the Validation

an image of the set up in FlightStream® is provided in Figure 46. Here the propellers are shown after a number of time iterations with the velocity magnitude color map shown across the propellers. The time stepping increment was set to 0.0001 seconds, and the solver was set to run

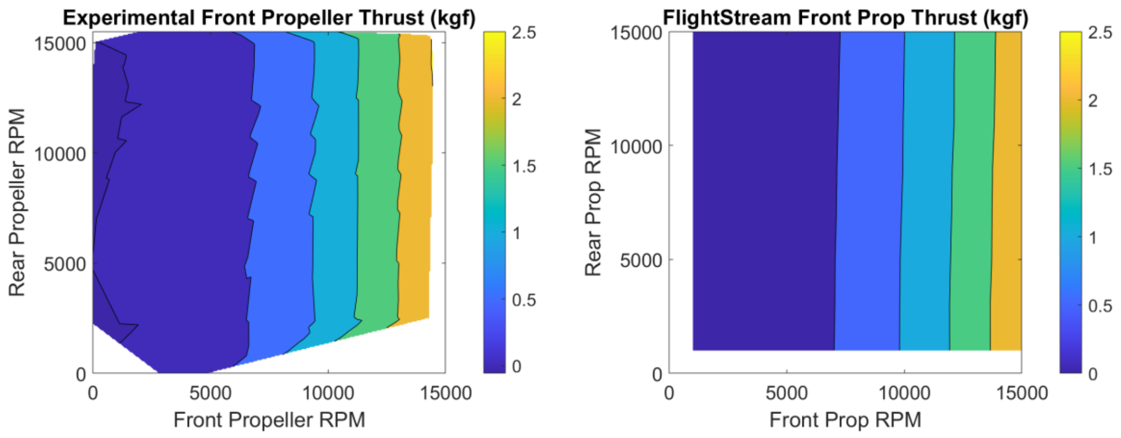


Figure 47: Front Propeller Thrust Validation

such that the front propeller completed one full rotation after its wake encountered the rear propeller to ensure that all the effects were captured. The propellers were tested over RPM values ranging from 1000 to 15000 for both the front and rear rotor. The results for the thrust by both rotors are provided in Figure 47 and Figure 48. The front propeller thrust is predicted accurately

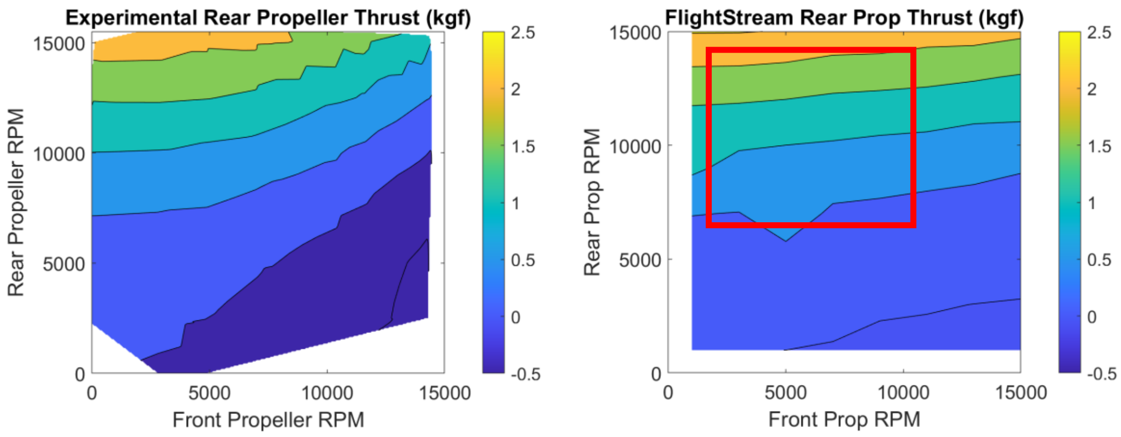


Figure 48: Rear Propeller Thrust Validation

by the solver for all RPM values of both the rear and forward rotor. As predicted, the rear rotor has little impact on the front rotor due to the nature of the setup described in Chapter 2. The rear propeller on the other hand, has a large dependence on the operating conditions of the forward rotor as shown by the experimental data in left graphic of Figure 48. The numerical solver results are accurate compared to the experimental data; however, the FlightStream® results fail to capture the full effect of the forward propeller, and thus, the solver is more accurately validated for the

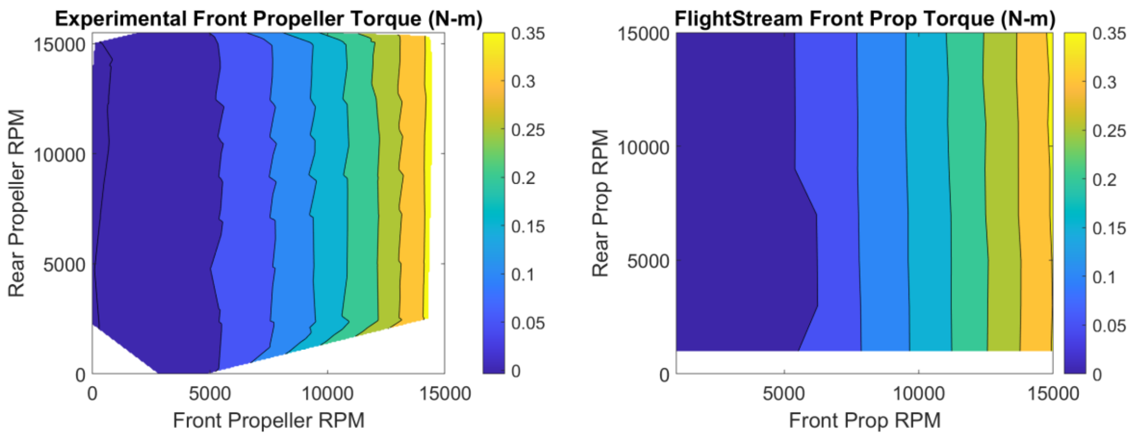


Figure 49: Front Propeller Torque Validation

cases which fall in the red square. These cases consist of lower RPM values for the front propeller and higher RPM values for the rear propeller. This same accuracy is seen for the torque measurements as well which are shown in Figure 49 and Figure 50. The front propeller's

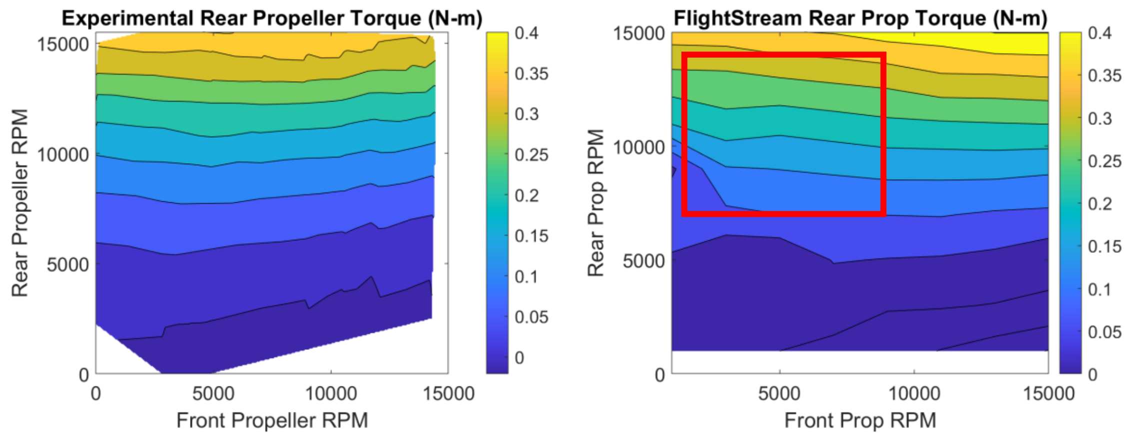


Figure 50: Rear Propeller Torque Validation

performance is modeled well by the solver and is congruent with what is expected from the experimental results. Furthermore, it is shown once again that the rear rotor has little if any impact on the performance of the forward rotor for the solver produced plots and the experimental data. The rear propeller torque values are once again modeled with accuracy but fail to capture the full effect of the front rotor. The range for accuracy of this validation is also shown by the red square and consists of the front rotor operating at a lower RPM and the rear rotor operating at a higher RPM. Though majority of the data is validated using the numerical solver, the GA will optimize a rear propeller that fits in the range of the red boxes. Because the solvers viscous models are aircraft centric, it is accurate only for higher Reynolds number simulations. The small rotors in this validation have Reynolds numbers on the order of 60,000 to 80,000 where the solver model is designed for Reynolds numbers on the order of few hundred thousand. This leads an error in the data at the lower RPM/lower Reynolds number values.

Chapter 6: The Genetic Algorithm Model

The propeller Genetic Algorithm is written in FORTRAN and requires a text file for input and a terminal command to access FlightStream® which provides the performance modeling for the propellers. This section provides a map for the GA, the variables considered, mutation techniques and the parent selection process for each generation.

6.1 Model Blueprints

The full GA model is presented Figure 51. This model includes the logic and thought processes that the GA uses to go through the entire execution including the demes and the main generational loop. The GA starts with a simple text file input which specifies the number of generations for each deme, the number of generations for the main generational loop, the operating

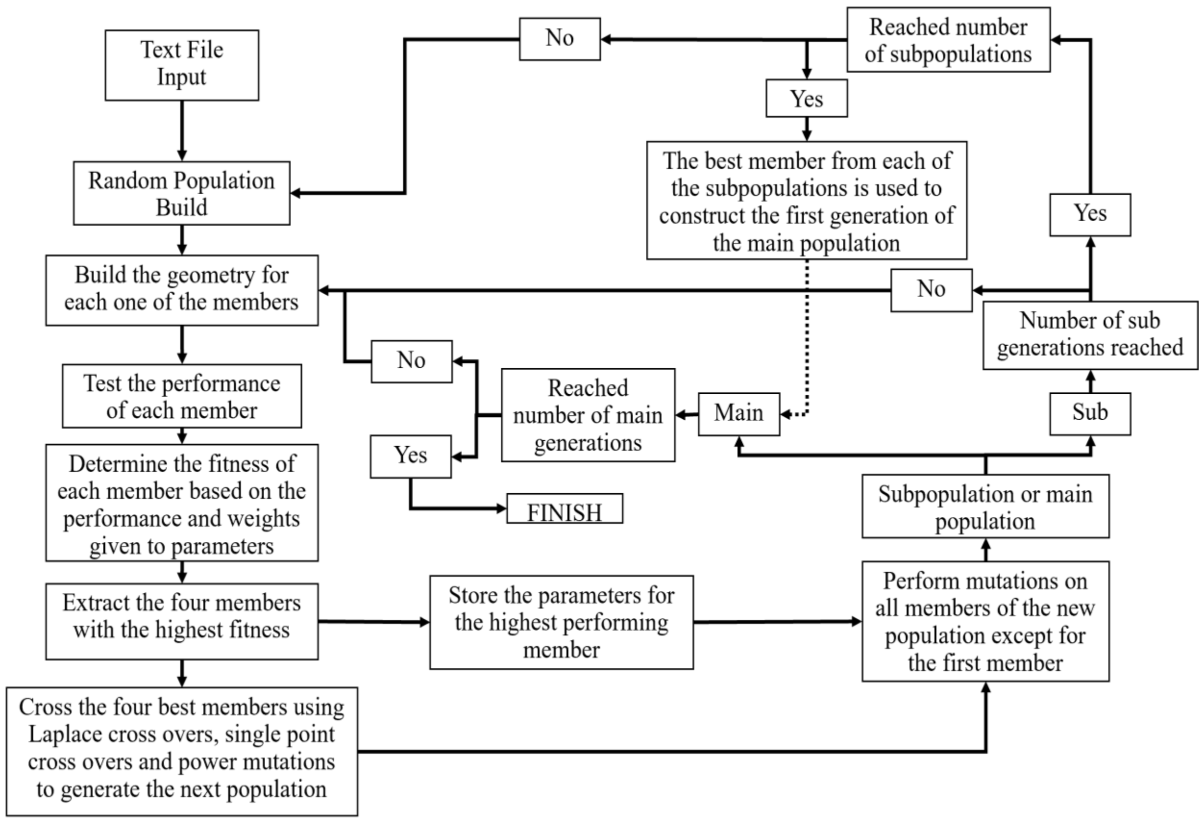

Text File Input
Random Population Build
Build the geometry for each one of the members
Test the performance of each member
Determine the fitness of each member based on the performance and weights given to parameters
Extract the four members with the highest fitness
Cross the four best members using Laplace cross overs, single point cross overs and power mutations to generate the next population
Store the parameters for the highest performing member
Perform mutations on all members of the new population except for the first member
Subpopulation or main population
Sub
Main
Number of sub generations reached
Yes
No
Reached number of subpopulations
Yes
No
The best member from each of the subpopulations is used to construct the first generation of the main population
Reached number of main generations
No
Yes
FINISH

Figure 51: GA Blueprints

conditions for the propeller, the minimum and maximum value for each of the parameters, etc. The GA then builds a completely random set of members as the starting population for deme number 1. These members are then tested and have the fitness determined. If the multiple flight condition setting is being executed, all the flight conditions are tested at the same time. Once each propeller has been evaluated, the fitness is determined for each member. Then, the four best members are placed in a mating pool where the next population is created. Finally, the new population is mutated and tested again. This is carried out for the user specified number of deme generations. The process of creating a new population, testing, building a new population off the four fittest, testing, and so on is carried out for each deme. Once each deme has been completed, the demes put forth their best member to create the starting population of the main generational loop. The process of testing, building, mutating, and testing is then carried out for the number of main generational loops.

The GA is built using five subroutines which are called in order throughout the main script. These subroutines are "Prop_Build", "FlightStream®_Script", "File_Read", "ParentSelector", and

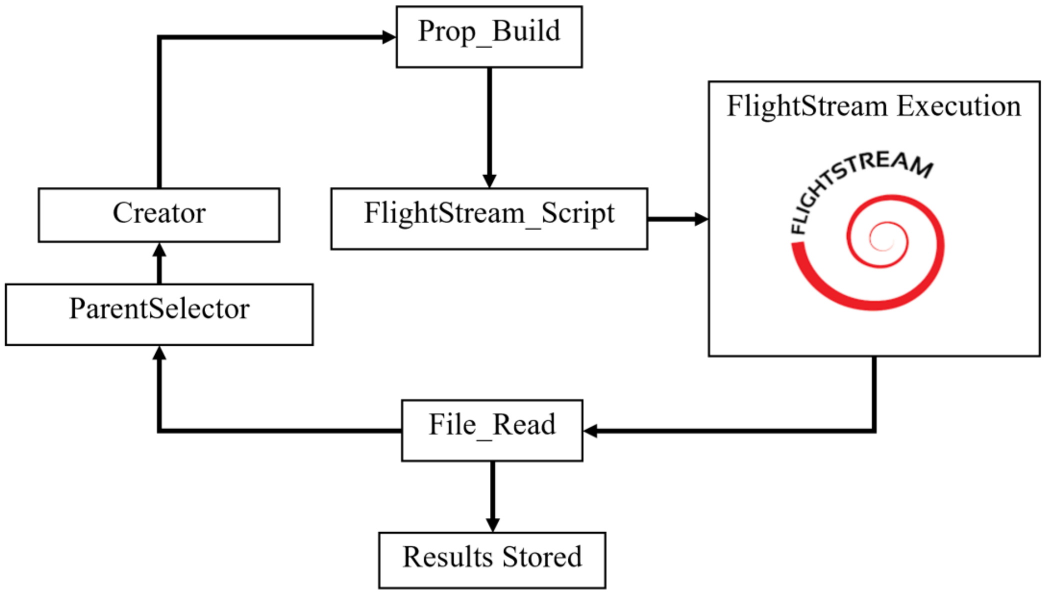


Figure 52: Subroutine Map

“Creator”. The order in which they are called is provided in Figure 52. This figure describes the process for a single iteration in the generations. “Prop_Build” is the first subroutine called in the generation. This subroutine uses the Bernstein Polynomials discussed in section 3.2 to build the CCS file for each propeller geometry. “FlightStream®_Script” builds a FlightStream® script which runs each member of a population in batch. After the script has been built a command line is used in the code to access FlightStream® and run the script. As each propeller is tested, and the performance is saved in a text file from FlightStream® which is then read back into the GA using the subroutine “File_Read”. “File_Read” reads in the performance data for each propeller which consists of the thrust, torque and flow separation data for the propeller. Using the torque and the thrust the efficiency of the propeller is calculated by the following equations.

$$C_q = \frac{Q}{\rho n^2 d^5} \tag{41}$$

$$C_p = 2\pi C_q \tag{42}$$

$$P = C_p \rho n^3 d^5 \tag{43}$$

$$\eta = \frac{T V_\infty}{P} \tag{44}$$

Q and P are the torque acting on the propeller and the power required respectively, and T is the thrust produced. C_q and C_p are the torque coefficient and power coefficient. n is the rotational speed of the propeller measured in rotations per second. After the efficiency and thrust have been obtained, the percent of separated flow across the surface of the propeller is determined. If more than 7% of the propeller has flow which is more than 90% separated a fitness of -10 is assigned. After reading and calculating the necessary data, “ParentSelector” then determines the fitness of each member and performs the parent selection process which is discussed at the end of this chapter. “Creator” takes the parents from “ParentSelector” and performs crossovers to create the new generation. This generation is then mutated, and the characteristics of each member are provided back to “Prop_Build” to create the geometries for the new generation.

6.2 Variables

The basic propeller depends on three main sets of variables each of which have several defining parameters. The propeller’s geometry, performance, and the flight conditions under which the propeller operates define the GA and promote the GA to converge on a solution. In this case, the GA makes use of the propeller’s geometry and mutates it to obtain the performance of the propeller under the specified operating conditions. The geometric set of variables consist of the radius, airfoil shape function, twist angle (beta) function, chord length function, and the sweep function of the propeller. The latter four of these variables depend on several other variables to provide a curve along the operate directions of the propeller. The performance set of variables consists of thrust, torque (power required), and efficiency. In the case of this model, two of the three performance variables must be set to achieve any useful results. If only the thrust is chosen to maximize, then an extremely inefficient propeller will be selected. If the power required is chosen to be minimized, then the GA will converge on a propeller that produces very little thrust. The one confusing parameter to hold is the efficiency. If only the efficiency is chosen, it is possible that the GA will favor a decrease in power over an increase in thrust. Therefore, the two most optimal performance variables to set for general use are the thrust and the efficiency. Lastly, the flight conditions must be reasonably set with the given thrust, efficiency, and radius specifications. This set consists of the free stream velocity and the RPM setting. Failure to set appropriate values will lead to faulty convergences. An example would be a very low thrust setting for a high RPM and radius value which will lead to a very skinny propeller which is trying to match the specified thrust.

In reality, models should be based off the power and thrust only. Therefore, the propellers will be maximizing thrust while absorbing a given power or perhaps minimizing power while producing a given thrust requirement. In this set up, propellers will be designed accurately for very specific operating conditions because some information has now been provided to the model which considers the performance of the power source. Nevertheless, the modeling approach used here which incorporates the thrust and the efficiency provides suitable results for the given study.

6.3 Mutations and Crossovers

The GA uses a variation of a modified Laplace crossover, selective point crossovers, and random point crossovers to make the next population which is then mutated by a variation of power mutations. The modifications and variations used in the GA are explained and examined below.

6.3.1 Laplace Cross Overs Modification

The Laplace cross over is one of the techniques used in this GA to generate members; however, this GA uses a modified version to provide better results. From the two equations described in the Laplace crossover section, six additional equations were added and observed over the six possible scenarios of parent 1, parent 2, and the target value. All of which were adjusted with respect to each other.

$$y_1 = x_1 + \beta\,(x_1 - x_2) \tag{45}$$

$$y_2 = x_1 + \beta\,|x_1 - x_2| \tag{46}$$

$$y_3 = x_1 - \beta\,(x_1 - x_2) \tag{47}$$

$$y_4 = x_1 - \beta\,|x_1 - x_2| \qquad (48)$$

$$y_5 = x_2 + \beta\,(x_1 - x_2) \qquad (49)$$

$$y_6 = x_2 + \beta\,|x_1 - x_2| \qquad (50)$$

$$y_7 = x_2 - \beta\,(x_1 - x_2) \qquad (51)$$

$$y_8 = x_2 - \beta\,|x_1 - x_2| \qquad (52)$$

The eight presented equations were compared in the six possible combinations of parent 1, parent 2 and the target value that the GA is trying to converge on. All six scenarios can be seen in Figure 53. It is now clear that certain equations make the appropriate mutation for certain scenarios. Table

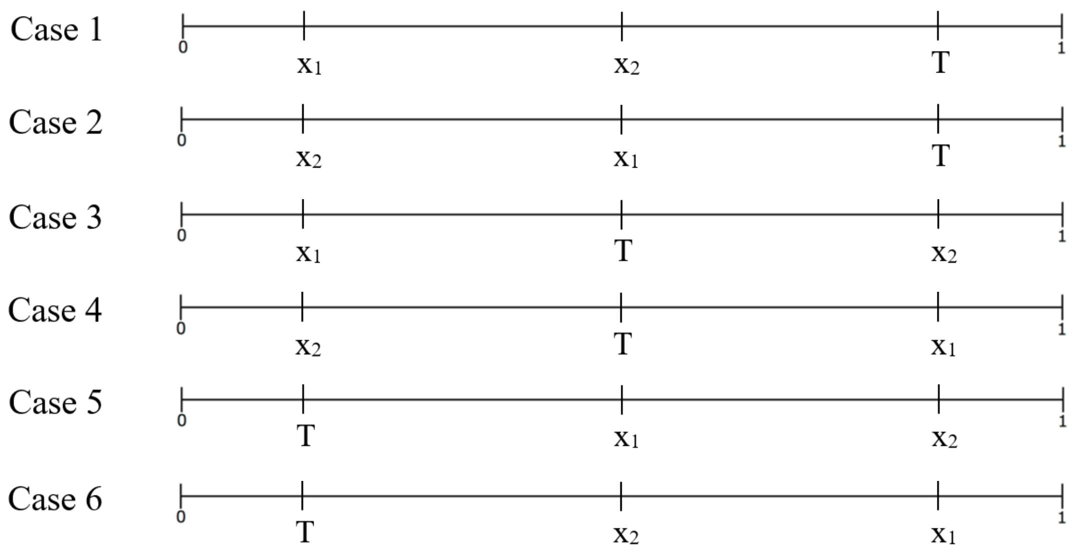


Figure 53: Possible Values of Parents and Target

1 is provided to shed light on which equations prove useful in their crossovers and which do not. The green coloration indicates a proper mutation; the uncolored cells indicated an incorrect mutation.

Table 1: Mutation Direction for Modified Equations

	Eqn 1	Eqn 2	Eqn 3	Eqn 4	Eqn 5	Eqn 6	Eqn 7	Eqn 8	x_1 & x_2 relation
Case 1		🟩	🟩			🟩	🟩		$x_1<x_2$
Case 2	🟩	🟩			🟩	🟩			$x_1>x_2$
Case 3		🟩	🟩		🟩			🟩	$x_1<x_2$
Case 4			🟩	🟩	🟩	🟩			$x_1>x_2$
Case 5	🟩			🟩	🟩			🟩	$x_1<x_2$
Case 6			🟩	🟩			🟩	🟩	$x_1>x_2$

It is now seen that child 3 and child 5 are the most useful of the 8 equations. These two equations will provide a 66.66% chance of a proper crossover. However, by restricting the criteria for what equation to use based off the values of the parents with respect to each other, more equations can be used to provide a 66.66% chance of a proper mutation. If just the cases where parent 2 is greater than parent 1 (cases 1, 3, and 5) are observed, children 2, 3, 5, and 8 will provide a 66.66% chance of success, and if only cases where parent 1 is greater than parent 2 (cases 2, 4, and 6) are observed, children 3, 4, 5, and 6 will provide a 66.66% chance of a proper mutation. By extending the number of equations to use, the variation in the child produced increases and mutations are not consistent. This allows for the GA to run through a wider range of results.

6.3.2 Selective Point and Random Point Crossovers

The selective and random point crossovers are used to create a child that is very similar to the two parents that were used in the synthesis. Both crossovers operate by simply exchanging parameters between two parents. The selective point crossovers exchange certain parameters between the two parents. In the case of this GA, these parameters consist of sets of polynomial coefficients that describe the characteristics of a parameter. The parameters referred to here can be the airfoil shape, chord, twist, and sweep. The polynomials which describe these are provided in chapter 3 section 2. For the selective point crossover, only entire sets of parameters are selected to be exchanged between parents i.e. the selected point for the crossover is only at the end of a group of parameters. For example, the first 48 parameters that describe a propeller geometry are used to define the shape. The next 3 intervals of 5 parameters are used to describe the twist, chord, and sweep. Therefore, crossovers only occur at 48, 53, and 58. This ensures that the entire set of genes which describe a propeller defining parameter are transferred over. Thus, the child will have exact characteristics of the two parents.

The random point crossover operates based on two parents combining at a random point to make a new child. Therefore, it is very likely that half of the defining genes for a parameter are derived from parent 1 and the other come from parent 2. With four parents, there are a total of 12 combinations of parents (1-2, 2-1, 1-3, 3-1, 4-1, 1-4, etc.). Therefore, two random numbers are selected for each crossover. The first random number is between 1 and 12, and it describes which parents will be combine and the order in which they will be blended. The second is between 1 and 63, and it describes the gene at which they will split. For instance, if the first random number is 3 and the second number is 34, then parents 1 and 3 will be split at the gene 34. The first part of the child will come from the first 34 genes of parent 1 and the last 29 genes will come from parent 3.

6.3.3 Power Mutation

The power mutation is mutation type chosen in this GA. The power mutation offers a wide range of mutation strength with the mutation being raised to a user defined value. The way by

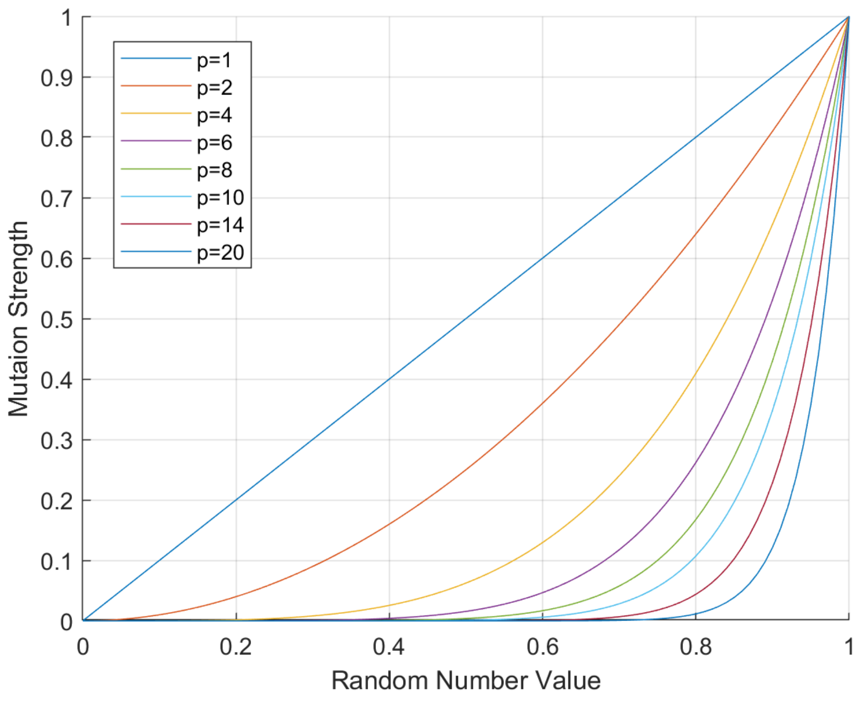


Figure 54: Mutation Strength vs Random Number Selection Over a Range of p Values

which the power mutation operates is described in section 2.4.3.6. The level of mutation is almost completely dependent on the value of p. A representation of the mutation variation can be described by a power function with varying levels of power. This is shown in Figure 54. The y-axis represents the amount of mutation for given values of *p* over a range of possible random values for *s*. The distribution of random values is evenly distributed between 0 and 1. At *p*=1, the mutation strength is linear with respect to the random value; therefore, one can expect to see a mutation strength greater than 0.50 50% of the time. This is an extreme amount of mutation and will lead to over and under shoots on the target value. At the opposite end of the spectrum, a *p* value of 20 will lead to less than a 0.10 mutation strength roughly 90% of the time. This is far too low and will not lead to sufficient mutation amounts in the early stages of the GA.

The correctness of the power mutation is dependent on a random value *r* and the value of the parent which is implemented upon *t*. The degree of correctness of the mutation will be ignored here i.e., the mutation either moves closer or further from the target; the degree to which it moves further or closer will be ignored. The following table presents evenly distributed possible values of a parent and the corresponding value for *t*. The table also presents the probability of the power mutation being correct.

Table 2: Odds of Mutation for Original Power Mutation

Parent Value	*t*	Odds if target is greater than parent	Odds if target is less than parent
0.1	0.111	11.1%	88.9%
0.2	0.250	25.0%	75.0%
0.3	0.428	42.8%	57.2%
0.4	0.666	66.6%	33.3%
0.5	1	100%	0%
0.6	>1	100%	0%
0.7	>1	100%	0%
0.8	>1	100%	0%
0.9	>1	100%	0%

The columns “Odds if target is greater than parent” and “Odds if target is less than parent” represent the odds of a correct mutation if the target value is above the parent or below the parent, respectively. Take the first row for example: given that the range of values for a parent are between 0 and 1, if the parent value is 0.1, then by equation 10 the value of *t* is 0.111. Now, *r* is a random number between 0 and 1 which means that there is a 11.1% chance that *r* is less than *t*, and there is an 88.9% chance it is greater than *t*. The target value is the most optimum value that the GA is looking for and is unknown. If the target value is greater than the parent value, then the mutation needs to increase, so by equation 9 *r* needs to be less than .111 which is an 11.1% chance. If the target is less than the parent, then we need a decrease in the mutation, so by equation 9, *r* needs to be greater than 0.111 which is an 88.8% chance. Because the target is unknown, it can be said that there is a 90% chance the target is above the parent, and there is a 10% chance the target is below the parent. By this observation, the equality of *t* and *r* should be flipped in equation 9 because it would be better odds to have an 88.8% chance of being correct 90% of the time than having a 11.1% chance of being correct 90% of the time.

Now, observe the behavior of a parent value being equal to 0.5. Therefore, there is a 50% chance the target is below and a 50% chance the target is above the parent. This also results of *t* value being equal to 1. This value for *t* results in a zero probability of the *r* value ever being greater than *t*. Thus, the mutation will only increase which means if the target is above the parent than it is guaranteed to approach it; however, if the target is below the parent, it will never be reached. This results in a true 50/50 chance of the target being approached which provides no weight to the argument of whether the inequalities in the power mutation should be flipped.

Lastly, observe the last condition in which the parent value is equal to 0.9. This results in a value for *t* which is greater than one leaving it impossible for *r* to be less than *t*. Therefore, there is a 0% chance that the mutation will decrease. There is a 90% that the target value is less than the parent, and a 10% chance that the target is above the parent. That means the GA will only be making a proper mutation 10% of the time. If the inequalities are flipped in the power mutation definition in equation 9, then we are left with a 90% of being right. Therefore, the direction of the inequalities should be flipped in order to promote better mutations in the Genetic Algorithm. Table 3 provides the odds of correct mutation for various parent values.

Table 3: Odds of Proper Mutation for Flipped Inequalities

Parent Value	*t*	Odds if target is greater than parent	Odds if target is less than parent
0.1	0.111	88.8%	11.1%
0.2	0.250	75.0%	25.0%
0.3	0.428	57.2%	42.8%
0.4	0.666	33.3%	66.6%
0.5	1	0%	100%
0.6	>1	0%	100%
0.7	>1	0%	100%
0.8	>1	0%	100%
0.9	>1	0%	100%

A graphical representation of the odds is presented in Figure 55. Clearly the plots of both cases will simply flip from case to case; however, this plot provides an image of the area for which the odds of a 100% correctness mutation are present when the signs are flipped. It also stresses the

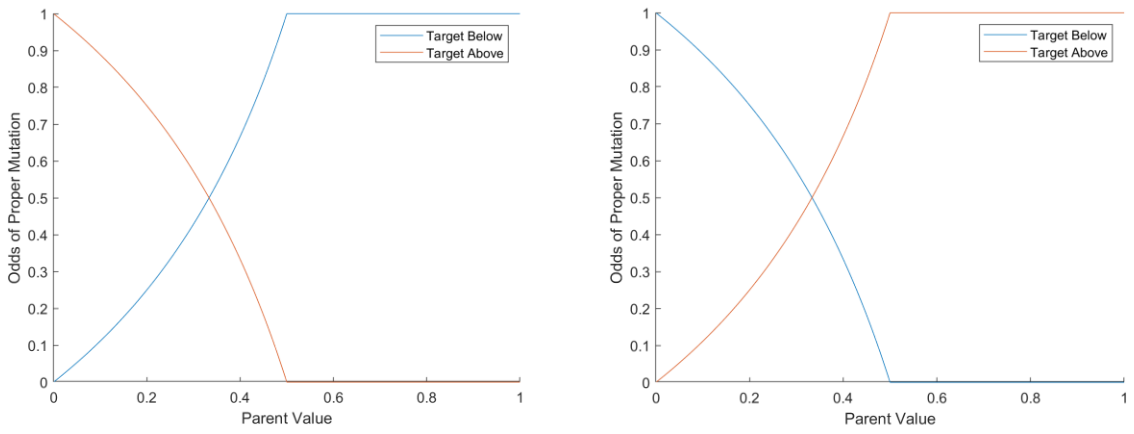


Figure 55: Odds of a Proper Mutation Given a Parent Value and the Location of the Target

point of the original power mutation in that if a value is already small and needs to decrease more it should rely on the original power mutation to do so. A key take away is that the modified power mutation will operate more effectively on a wider population which is present in the early stages of the GA. The original power mutation will be more effective in later stages for fine tuning parameters.

6.3.4 Mutation bounds

Using crossovers and power mutations as the operations for creating and mutating members, a check point has been implemented for both schemes to ensure that values outside the range of the specified parameters are not created. The Laplace crossover uses one of the eight mentioned equations to ensure upper and lower bounds are not crossed. The upper bound equation is provided from the 8th of the developed equations and is shown below

$$y_8 = x_2 - \beta\ |x_1 - x_2| \tag{53}$$

The lower bound check uses the 2nd of the 8 equations

$$y_2 = x_1 + \beta\ |x_1 - x_2| \tag{54}$$

It is seen that if the upper bound of the parameter at hand is crossed, then the value is decreased, and if the lower bound is crossed with the original mutation, then the value of the child's parameter is adjusted to increase. This ensures that all values and mutations stay within the user specified range.

The power mutation uses one of the same equations presented in the original mutation to readjust values back into the limits. The conditional set for the values of *t* and *r* is replaced with a minimum and maximum range value. If the upper limit is surpassed, then the following is used

$$x = \bar{x} - s(\bar{x} - x^l) \tag{55}$$

Furthermore, if the lower limit is met, then the following is used to mutate the parameter

$$x = \bar{x} + s(x^u - \bar{x}) \tag{56}$$

By using these equations and stated conditionals for minimum and maximum values, no mutation can produce a value for a parameter that is outside the range of the specified values.

6.4 Parent Selection

The parent selection for this GA can vary more than most due to the different optimization points that the user describes. In general, the fittest four members of each population are selected to be placed in a mating where they will pass their characteristics to the offspring. The fitness of each member is determined by the following

$$Fit = \frac{\sum\left(T_c w_t + \eta\, w_\eta\right)_i}{\sum\left(w_t + w_\eta\right)_i} \tag{57}$$

Here the T_c values are the thrust correctness of each flight condition and the w_t values are the weight given to each of those thrust values.

$$T = 1 - \frac{\left|T_{req} - T_{act}\right|}{T_{req}} \tag{58}$$

T_{req} is the required thrust provided to the GA, and T_{act} is the actual thrust found from the solver. When the required thrust is equal to the actual thrust the second term goes to zero and the fitness for that particular requirement is 1. The further the actual thrust strays from the required thrust leads to further decreases in the fitness. The same weighted method is applied to the efficiency where η is the efficiency at a given flight condition and the w_η is the weight of that flight condition efficiency. The number flight conditions can range from 1 to infinity; however, convergence issues arise when the number of flight conditions increases. Models have only been executed up to four flight conditions.

The actual parent selection itself is described by Figure 56. All the members are first tested against each other in an unseeded tournament. In the first round, the first half of the members are tested against their counterpart in the second half i.e., member 1 faces member 65, member 2 faces member 66 as shown. This is not a complete method because the tournament is unseeded, and

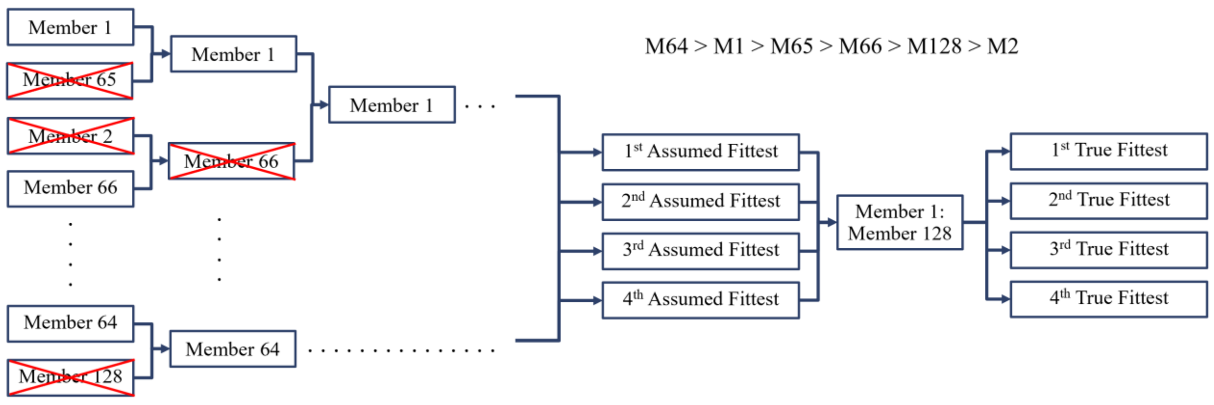


Figure 56: Example Parent Selection

therefore, a secondary test must be conducted. From the tournament, four assumed fittest members will be selected. These four members are then individually tested against the entire population, and if any of the assumed fittest members have a lesser fitness, then that member is replaced. In Figure 56, the errors of the unseeded tournament are exposed in the elimination of member 65. Member 65 is the third best member and should be in the mating pool, but it is eliminated in the first round of the tournament; however, with the implantation of the round-robin style tournament, member 65 will end up replacing the assumed 3rd best member from the population.

6.5 Generational Variance

Each population consists of 128 members. The four fittest members of each population are selected to be in the mating pool which will create the next population. To ensure that progress is not lost, the overall fittest member is chosen to be a part of the next generation unchanged. This is known as elitism. Once in the mating pool, modified Laplace, selective point, and random point crossovers are performed on the four fittest members. These cross overs provide the foundation for the next set of members which are mutated.

The first crossovers seen in the population generation are the Laplace crossovers. These Laplace crossovers produce 36 of the members in the new population. These crossovers are done evenly over combinations of parents (parents 1 and 2, parents 1 and 3, etc.). Each combination of parents crosses 6 times. The next crossover done to create the new population is the selective point. There are 48 of these crossovers which occur between every parent combination including the order at every characteristic change in the vector of parameters. Lastly, the random point crossovers produce the remainder of the 128 members. Table 4 provides a summary of how each generation is constructed. Because a large portion of the new generation comes from non-mutating

Table 4: Summary of Population Generation

Member 1	Members 2-37	Members 38-85	Members 86-128
Fittest from Previous Population	Modified Laplace Crossover	Selective Point Crossover	Random Point Crossover

parameters, a large number of mutations need to occur. The selective point and random point crossovers do create some variance in the populations, but they assume that the optimal values for each parameter already exist in the population, only not in the correct order. Therefore, the power mutation was implemented to provide some variance. The power of mutation varies depending on the accuracy of the fittest members. This is discussed in the following section.

It is important to note that all the random values used in this genetic algorithm come from the prebuilt function in FORTRAN for random number generation. This random number generator provides real numbers between 0 and 1 when it is called. This function is called multiple times

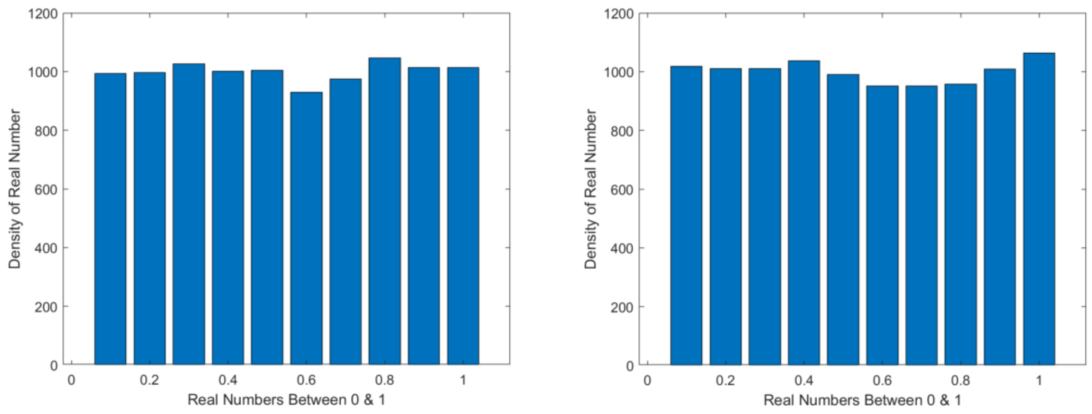


Figure 57: Random Number Variation

throughout the use of the GA, and to ensure its randomness is truly uniform. 2 test cases were executed. The results from these cases are provided in Figure 57. Here the random number generator was called 10,000 times. The bar graphs were then created which provide the amount of times a random number was between 0 and 0.1, 0.1 and 0.2, etc. As the plots show, the random number is indeed a from a uniform random number generator.

6.6 GA Check Points

Lastly, the different check points implemented in the GA are discussed. These check points provide many advantages to the code over a variety of aspects which ultimately assist the GA and the solver in finding a global maximum. There a 6 main check points in the GA: 2 fitness-mutation checks, a flow separation check, an achievable efficiency, and 2 geometry check points.

6.6.1 Mutation Check Points

The 2 fitness-mutation check points tell the GA how the next population will be created depending on the maximum fitness in the current population. Because many of the check points are constructed around eliminating poor performing members (these members are given a fitness of -10 to ensure the genes are not passed to the next population), in early stages, it is possible for every member to have a fitness of -10. In such a case, the GA will not know which member is most optimal and therefore, it will simply pick the last four members in the population to build the next generation. The odds of these members being good members is very unlikely, so the next population will simply be filled with poor performing members. This process will continue until the maximum number of iterations is reached. To combat this, the GA has a check. If the maximum fitness of the entire population is -10, then the GA builds a completely new and random population. With 128 members per population, it is unlikely that the GA will produce more than 2 random populations where every member is a poor performer. The second fitness-mutation check point deals with the mutation strength of the power mutation. If the maximum fitness in the previous population is not higher than a set value, the power of mutation is decreased, leading to an increase in the mutation strength. This gives the population a large amount of variance as it searches in the early stages. Once the GA starts producing members within a desired fitness, the mutation strength is decreased to allow for proper amounts of creep in finding the solution.

6.6.2 Performance Check Points

The next check point mutation deals with the flow separation setting discussed in chapter 2. If more than a certain number of faces on the top side of the propeller experience flow separation, then the propeller is given a fitness of -10. This is for two reasons: propellers with more flow separation are not good performing propellers and the solver does not model flow separation impacts to their full extent. However, due to the randomness of the first generation, a lot of propellers experience massive amounts of flow separation, and none fall below the lower limit of faces which can have flow separation. Flow separation should be limited as much as possible, but if it is limited too much, none of the propellers will have a fitness other than -10. Therefore, there are two searches placed in the GA. The first looks for propellers with less than 7% of the surfaces separated. If none of the propellers fall in this category, then the GA makes use of its second search which finds propellers with less than 12% of the faces having separated flow. The GA continues this process throughout every generation; however, once the GA finds one propeller with less than 12%, it will build more resembling these characteristics, and eventually one will be constructed which has less than 7% of the faces separated. At this point 7% becomes the new criteria.

A realistic efficiency check point has also been placed in the fitness calculation for the GA. While the numerical solver has mid-level fidelity capabilities, there are some cases in which the solver converges on non-realistic solutions or more likely, the solver does not converge in the given number of iterations. In this case, it is unlikely, but possible, that the solver will provide results for a non-realistic propeller that matches the thrust very well and has an extremely low torque on the blades, resulting in efficiencies greater than the possible limit. To address these

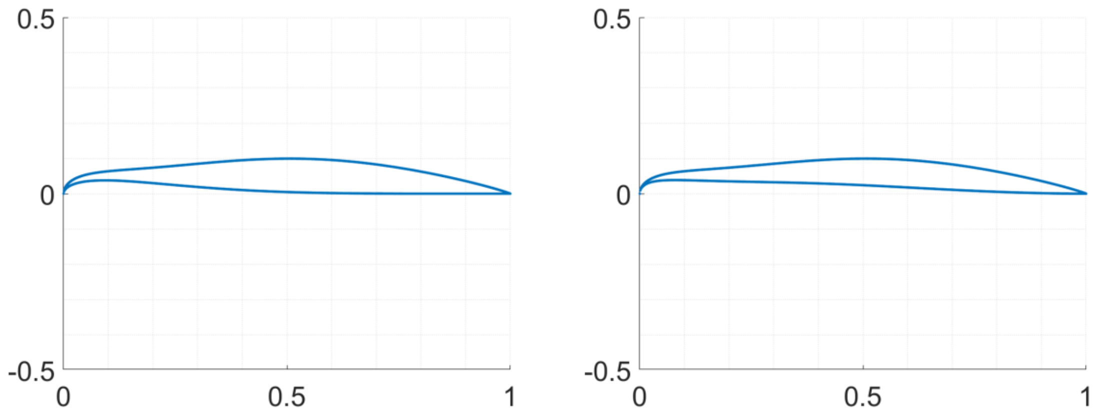


Figure 58: Unrestricted Potential Airfoil Shape

propellers, a loop is placed in the fitness calculation which searches for propellers that efficiencies above the set limit. If a propeller is found to have an efficiency greater than this limit, it is given a fitness of -10.

6.6.3 Geometry Check Points

Lastly, there are two geometry check points which are implemented to assist both the numerical solver and the GA. The first geometry check point that was implemented and the GA sees in constructing the geometries is the airfoil leading edge limit. Unchecked, the abstractness of the BP allows for the airfoils to take on shapes as shown in Figure 58. These two airfoil shapes appear to be identical, yet they were created by two different means. The airfoil on the left was created by having a lower surface *au* value that was much less than the upper surface. Because the lower surface is subtracted from the upper surface, when the lower surface subtraction is small, the leading edge can take shapes as shown. The airfoil on the right was created by setting the lower leading-edge shape to a value greater than the upper leading-edge term. Again, because the lower surface is subtracted, under these conditions the airfoil leading edge will take this shape. This is not only a bad design, but the solver has difficulty in converging on propellers or any lifting surface for that matter with these oddly shaped cross sections. To limit these shapes from being constructed, conditionals were set in place to make the lower surface term equal to the upper surface term if it is less the upper surface value. Note: this is only done for the first term located nearest the leading, therefore, propeller airfoils are still allowed to have extensive amounts of camber. The same sort of conditional is also applied to the leading-edge shaping terms.

The chord shapes are also limited in order to help the GA limit the extent to what it can create. Propellers with high efficiency, typically have chord lengths that taper towards the tip of the propeller blade. Simply to help the GA, the last two stations that model the chord at the tip and near the tip were limited. The chord along the span of the propeller is described by five coefficients for a BP. The first coefficient describes the tip chord, and the fourth coefficient describes the chord at the hub. The second and third coefficients fill the chord descriptions between the tip in the hub. The last coefficient describes the overall length of the chord. To ensure that the chord had some amount of preferred taper, a conditional was set such that the first coefficient must be less than the second, and the second has must be less than the third. This means that the chord can potential increase up until the mid-span of the propeller, but after that point, the chord will be limited to a constant or decreasing shape.

Chapter 7: Results

7.1 Stand Alone Propeller Model

The Genetic Algorithm was first set to optimize a single propeller operating in the freestream. This model used the steady rotary solver which offers the periodic symmetry. Using the periodic symmetry can cut run times down to half of what they would be with the full mesh; therefore, the GA ultimately converges faster. In this set up the propeller was optimized for thrust matching while keeping efficiencies very high for a single operating point. This model was used several times on multiple thrust settings, and a strong argument is made for the overall convergence of the GA.

Each propeller was tested in a free stream of 20 m/s and operated at 6000RPM. All propellers were set to 25.56cm (10inches) which corresponds to an advance ration of 0.71. Typical propellers at these conditions and sizes produce thrust values from 1 to 3 newtons[79]. The thrust input for the GA had a much higher range than the expected thrust values just to observe if its members would indeed try to optimize on the grossly over predicted inputs. Lastly, each propeller was provided a weight of 3 for the efficiency and 7 for the thrust, so it should be understood that the GA will favor thrust accuracy over an efficiency jump in the mutation process.

7.1.1 Single Point GA Analysis

A single case is observed here to demonstrate the effectiveness of the GA. All thrust cases had the same behavior across the generations which is the reasoning for only examining one of the thrust settings in detail over the demes and main generations. This GA analysis focuses on the 3N requirement and outlines the GA from the first generation of the first deme to the 100th and final generation of the main loop.

First, the thrust, efficiency, and fitness of four demes are analyzed over the course of 10 generations in Figure 59. The thrust and the efficiency of the most optimal member of each

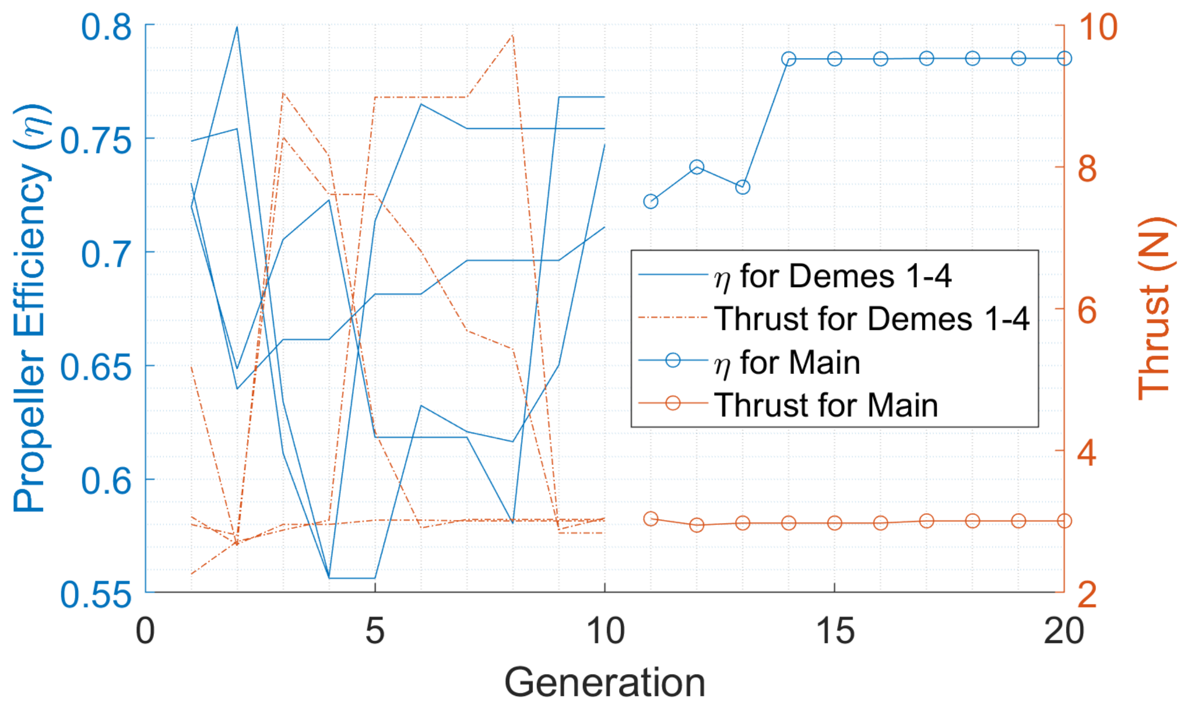


Figure 59: Thrust and Efficiency over 10 Deme Generations and 10 Main Generations

generation of the demes are shown by the lines on the left-hand side of the plot which range from 1 to 10. The lines with circle on the right-hand side represent the main generational loop. All the demes eventually produce propellers with greater than 70% efficiencies and thrust values that are near the requirement. Through this process, the demes are seen to be erratic in behavior as they bounce and mutate across the board, and in some cases decreases in thrust and efficiency are seen to be almost at random. But this behavior is rather easily explained by 1 one of two things: the favoritism shown towards accurate thrust propellers and one of the check points in the GA. The favoritism towards propellers with accurate thrust develops from the different weights given to the efficiency and thrust. Because thrust has a higher weight, the GA will choose propellers with accurate thrust while disregarding the efficiency of the propeller. This can therefore explain some of the absurd decreases in the efficiency. Second, recall that excessive amounts of flow separation across the top of the propeller are indicative of a poor performing propeller which is subject to large amounts of torque and decreases in thrust, thus large decreases in efficiency. To combat propellers of this nature the GA has a check point that of which assigns fitness levels of -10 to propellers with greater than 7% of the faces experiencing flow separation. One issue with this method, however, is the fact that it is very likely all the propellers will have greater than 7% flow separation. Therefore, if no propeller with less than 7% separation is created, then the GA looks for propellers with less than 12% separation. Once a propeller with the required separation is found then it will be given a high fitness and used to populate the next generation. Even though the GA is using a propeller with 12% separation, a propeller with less than 7% separation will inevitable be created over a few generations. Because the GA searches for the 7% criteria first, once it finds one with less than 7% separation, this propeller will then be the new dominating form even though

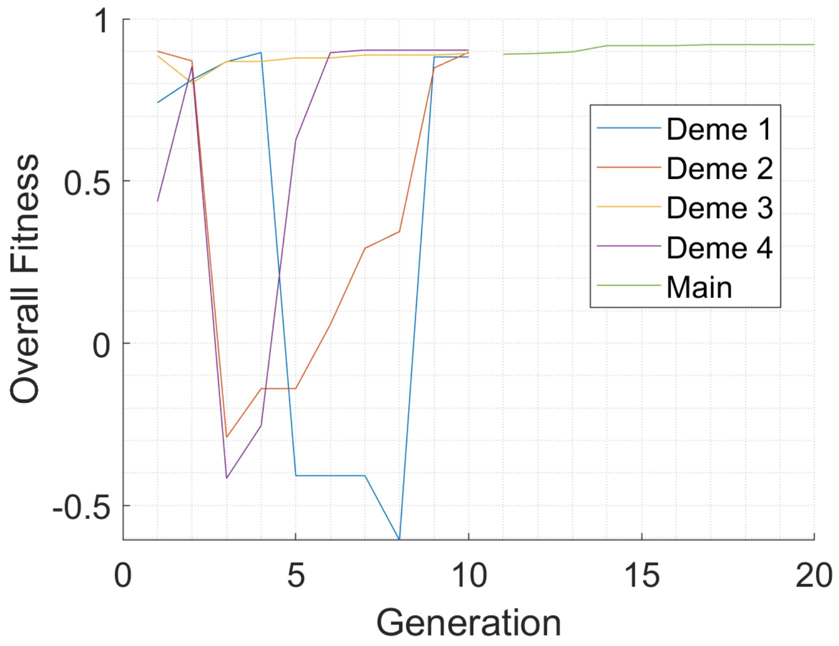


Figure 60: Fitness over 10 Deme Generations and 10 Main Generations

its fitness may be lower than the original propeller with 12% of the faces separated (this decrease in fitness is common over the transition from 12% to 7% separation). Figure 60 demonstrates the decrease the in fitness from the transition of 12% separation to 7% separation. In this figure, large decreases in the fitness occur for each one of the demes. On average, the transition takes place after the 2 or 3 generation. It is rare to see an initial population that already has a propeller which has less than 7% separation as well for a deme to operate until the 8th generation before the separation occurs. Despite the decreases in fitness, all demes reach fitness levels of 0.88 or higher by the time the top performing members are passed to the main generational loop. Lastly a tabulated form of this data is provided Table 5. The yellow column shows the location of the transition point.

Table 5: Deme Behavior over 10 Generations

Generation	1	2	3	4	5	6	7	8	9	10
Fitness	0.436	0.854	-0.416	-0.254	0.626	0.895	0.904	0.904	0.904	0.904
Efficiency	0.720	0.799	0.634	0.557	0.714	0.765	0.754	0.754	0.754	0.754
Thrust	5.18	2.65	9.05	8.15	4.27	2.91	3.03	3.03	3.03	3.03

The fittest propeller geometries across the generations are provided in Figure 61. These sample geometries are taken from deme 2 and notice in Figure 60 that deme 2 experiences a massive decrease in fitness at the 3rd generation. This is due to the flow separation transition. The propeller now has less than 7% of the upper surface separated, but it is inefficient, and the thrust is over predicted. The most notable change in the geometries is the change from generation 9 to 10. This large jump in propeller shape is permitted to occur because in the early stages of the GA, the mutation strength is extremely high. The higher mutation strength allows the GA to cover an extensive amount of design space with no loss in fine tuning the geometries because the GA is in the early stages of the generations. All the demes have the same behavior other than some of the propellers that are produced in the final generation because each deme starts with a completely random population and only operates over 10 generations which is not enough time to have a fully developed propeller.

The main generational loop for the 3-newton propeller is examined next to demonstrate the how the GA mutates towards its most optimum value. The first population in the main loop was

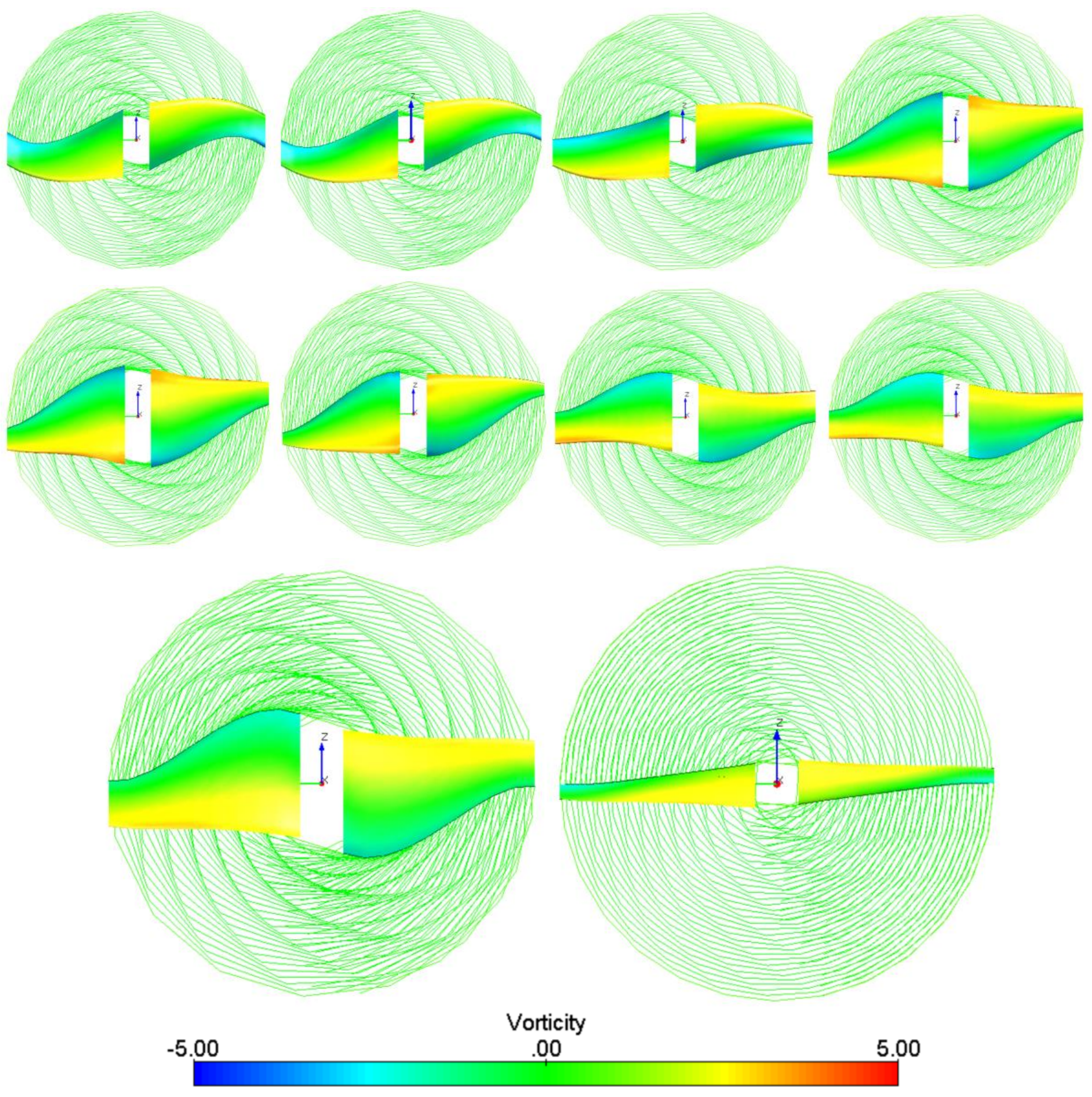


Figure 61: Example Deme Mutations over 10 Generations

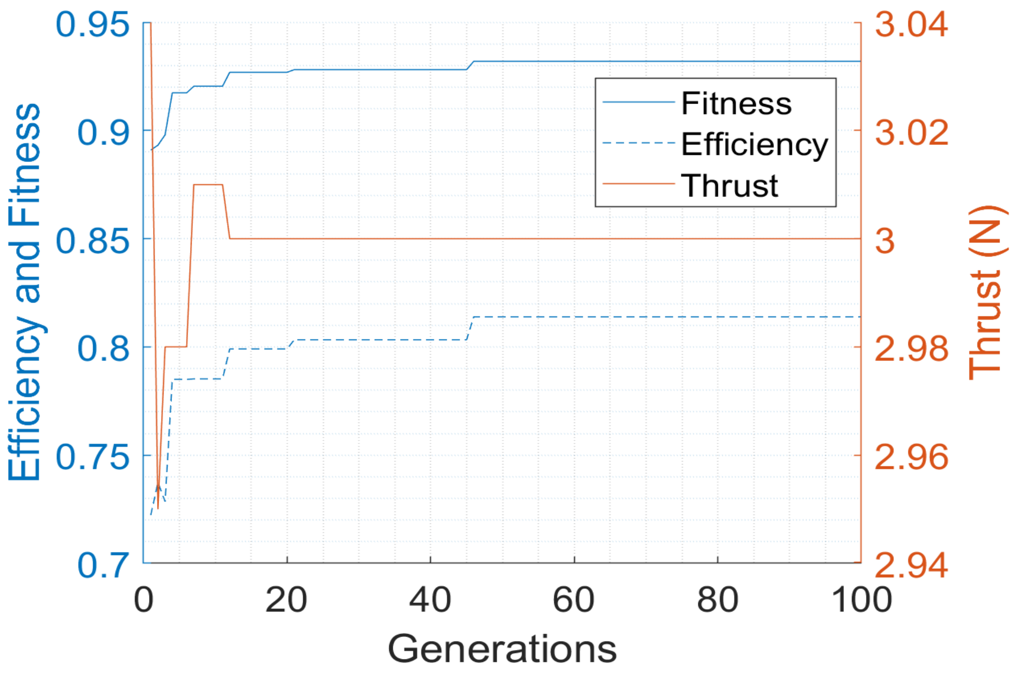


Figure 62: Main Generational Behavior for 3-Newton Propeller

constructed off the best propeller from each of the demes, and therefore, the propellers are for the most part good performing propellers. The fitness, efficiency and thrust of the best member from each generation across the total number of generations is provided in Figure 62. The propellers

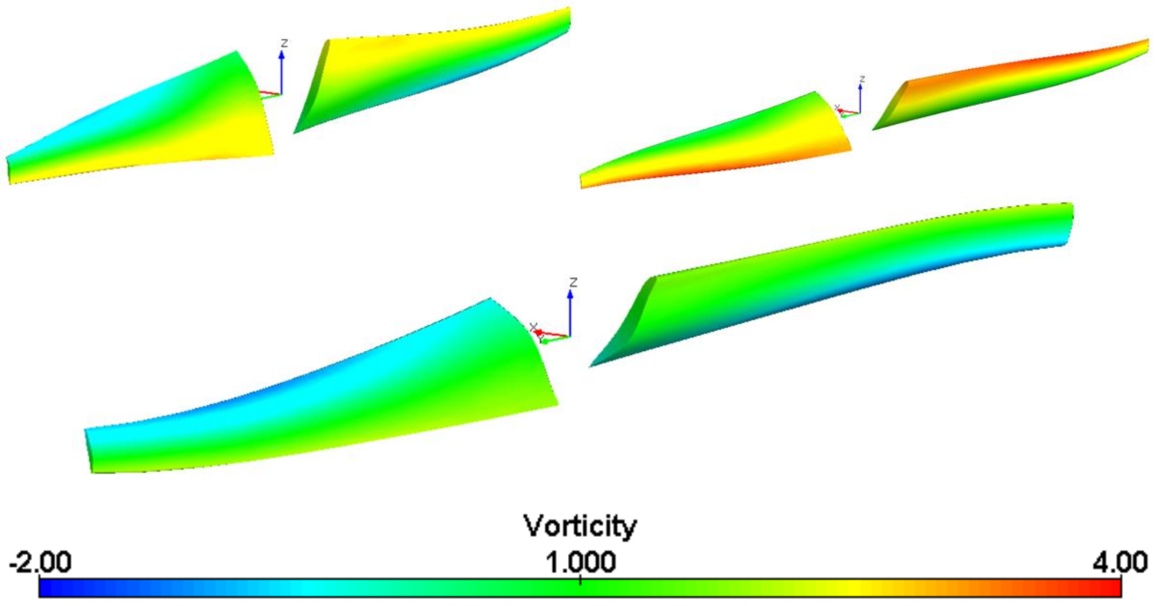


Figure 63: Fittest Propellers over 3N Thrust Requirement Main Generation

start out with fitness levels of about 0.89 and thrust values within 2% of the required value. Because weight of thrust is far greater than the efficiency, propellers often converge on thrust and then move towards increased efficiency designs which is present here. The demes have served the purpose of producing propellers that have thrust values near the required value, and now, the main loop will fine tune the propeller geometries to obtain a high efficiency propeller. The GA continues to make large mutations from generation to generation until about the 10th generation and has a final mutation at generation 46 where the efficiency is increased by 1%. The final propeller has a thrust of 3.00N and an efficiency of 81.4% and an overall fitness of 0.931. Lastly, the most optimum propellers from generations 1, 25, 50, 75, and 100 are provided in Figure 63. Only three propellers are shown here because the GA stopped obtaining better members after the 46th generation. Therefore, the maximum propeller at generation 50, 75 and 100 are identical. The propeller in the top left is from generation 1, and the propeller in the top right is from generation

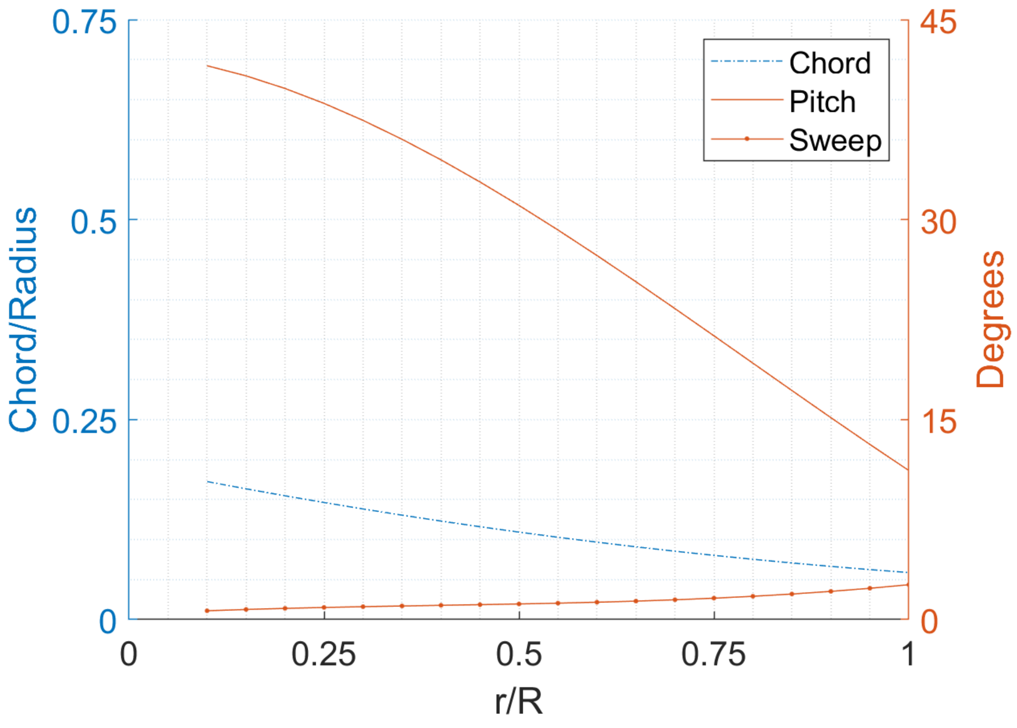


Figure 64: Geometric Parameters for 3N Propeller

25. The geometric parameters for the most optimal propeller of the 3N thrust requirement are provided in Figure 64. The propeller has characteristics that are indicative of an efficient propeller for each geometric parameter. Towards the tip of the propeller the blade has decreases in the pitch and the chord to prevent the blades from experiencing increased torque loads which will decrease efficiency. Furthermore, the propeller has some sweep at the tip of the propeller which is another characteristic of a typical efficient propeller.

7.1.2 Single Design Point (2, 3, 5, 7, and 10 Newton Requirements)

The GA was set for various thrust requirements over a range from 2 to 10 N to ensure that the GA and the solver would respond appropriately to the thrust that is demanded. The geometries and corresponding performance from each of the propellers is presented and demonstrates reasonable convergence given the thrust requirement for the constant diameter and operating conditions that all propellers were subject to.

Table 6: Thrust, Efficiency, and Fitness for Thrust Requirements

Design Thrust	2N	3N	5N	7N	10N
Thrust (N)	2.00	3.00	5.00	6.95	10.12
Efficiency	81.24%	81.39%	78.06%	76.53%	71.55%
Fitness	0.93148	0.93196	0.91987	0.90967	0.88921

Table 6 provides the most optimum from each of the thrust requirements. It is obvious from the understand of section 2.8 that as the thrust is increases for a constant diameter propeller operating in constant conditions, the efficiency of the propeller will decrease. This is also demonstrated in the table as well. Furthermore, because the efficiencies are dropping off, the fitness levels of the propellers will also be limited as the thrust acquired fitness and the efficiency determined fitness battle each other. A representation of this tug-of-war between efficiency and thrust is provided Figure 65. The theoretical maximum efficiency for a propeller given thrust, free stream, and

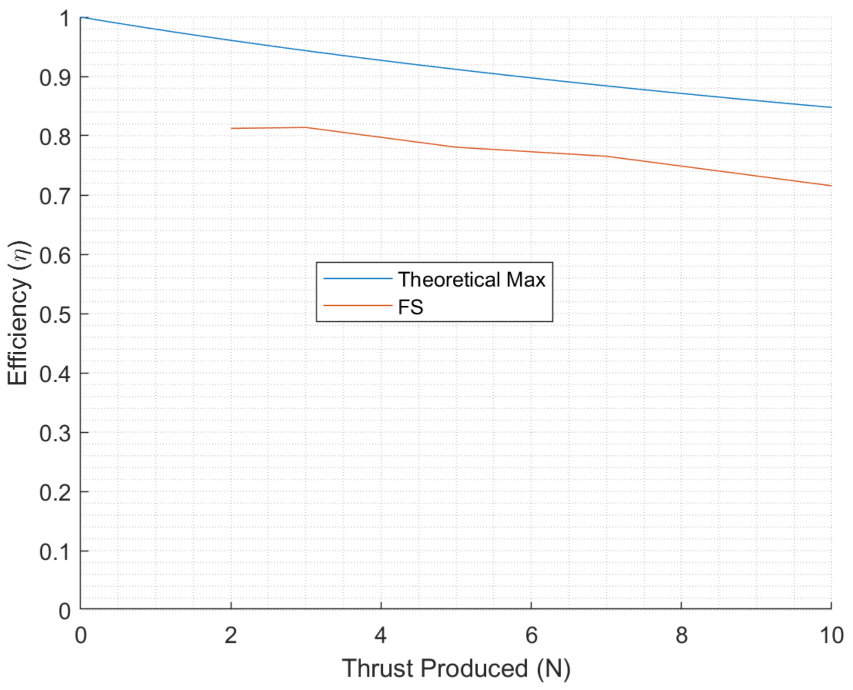


Figure 65: Efficiency vs Thrust Production for Constant Operating Conditions

density, is derived in section 2.9. The propellers produced by the GA follow the trend of this theoretical maximum efficiency over the range of given thrust values with constant under prediction of roughly 14%.

The final propeller designs are presented in the following four images. The designs for each of the propellers at the given thrust values are quite intuitive. As the thrust is increased, the chord and pitch along the propeller are increased to an extent. All propellers exhibit appropriate sweep as well to maintain efficiency. As the propellers increased in size, the sweep was also increased but remained similar between the cases and appears to simply have been scaled from case to case.

The geometries for each optimized propeller are provided Figure 66 where the 2 Newton propeller is in the top left and the 10 Newton propeller is on the bottom. The 5, 7, and 10 newton propellers are simply scaled versions of each other. Because the GA limited the propellers to a

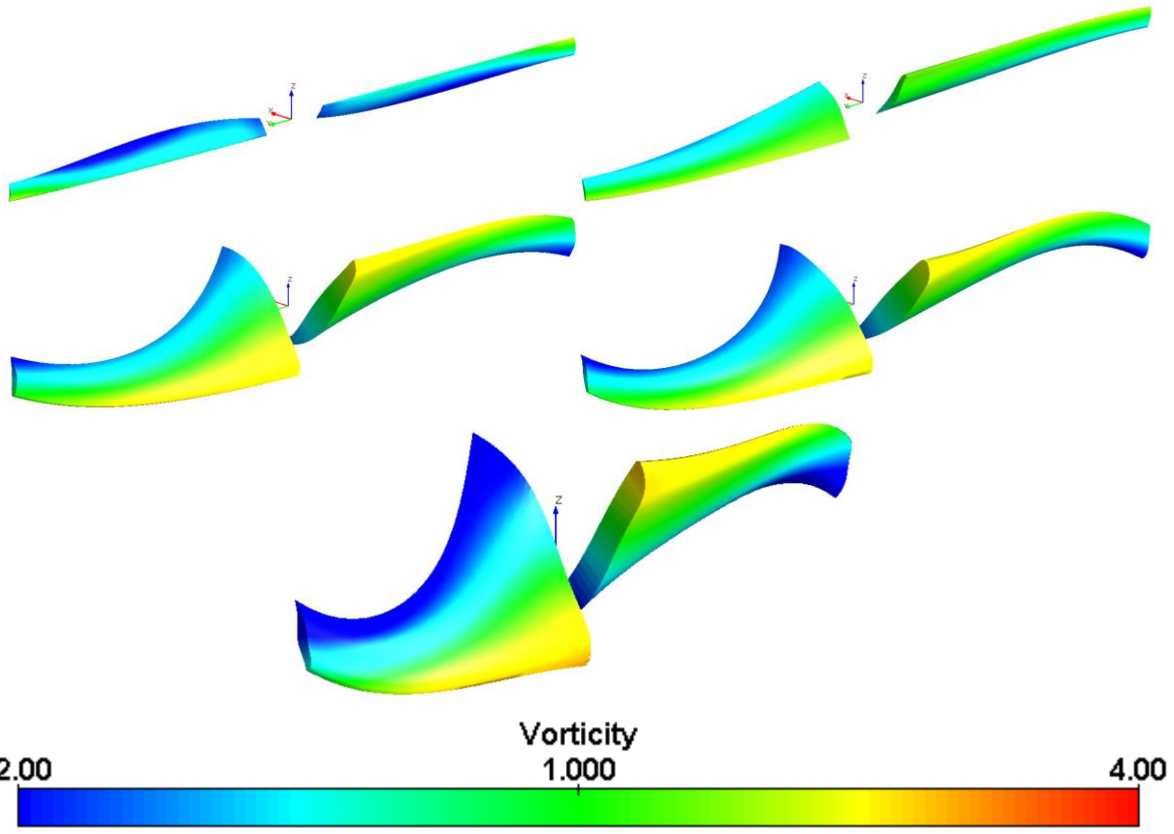


Figure 66: Optimum Propellers from Thrust Requirements of 2, 3, 5, 7, 10 N

certain amount of separation, the propellers could only increase the pitch angle so much to increase the thrust production. The GA was permitted to increase the pitch further out towards the tip and still have meet the requirement for minimum separation but doing so would lead to large drag forces on the tip which in turn would increase the torque and decrease efficiency. Therefore, the GA used the chord to match the thrust requirements once the pitch angles had been put to the maximum. Note that each of the propellers from the 5, 7 and 10 requirements increased the chord nearest to the hub. This was done to keep the propellers more efficiency by reducing the drag on the tip of the propeller and ultimately reducing the torque. There then persist an issue in the optimization where the GA can increase the chord at the hub by some large amount to account for the thrust or increase the pitch of the propeller at the tip by a miniscule amount. Both methods will achieve the necessary thrust but sacrifice some loss in efficiency. This GA found the following forms for twist and chord to be the most effective, presented in Figure 67. The chord values nearest

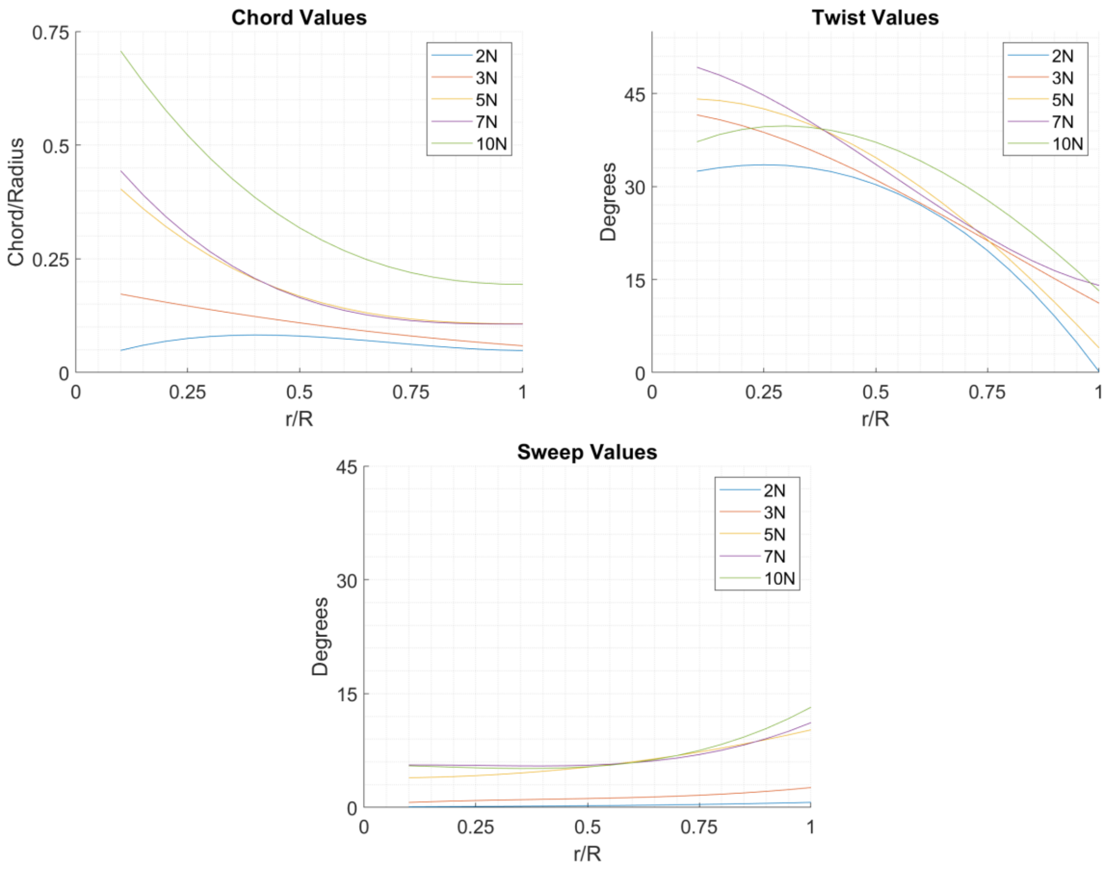


Figure 67: Geometric Plots for Optimum Propellers

towards the hub increase exponentially with increases in thrust requirement whereas the tip values have a more linear increase in length with the increase in thrust requirement. All the curves for twist distribution along the span of the propeller are very similar and seem to reach a maximum value which is caused by the flow separation limit. One remarkable similarity in cases is between the 5 Newton requirement and the 7 Newton requirement. These two propellers are almost scaled versions of each other in which the greater of the two has increases in the chord, twist and sweep distribution. The sweep for each of the propellers demonstrates a common behavior, relatively constant values until about 75% of the radius at which point the propellers sweep begins to increase. This increase in sweep is amplified as the propellers grow larger.

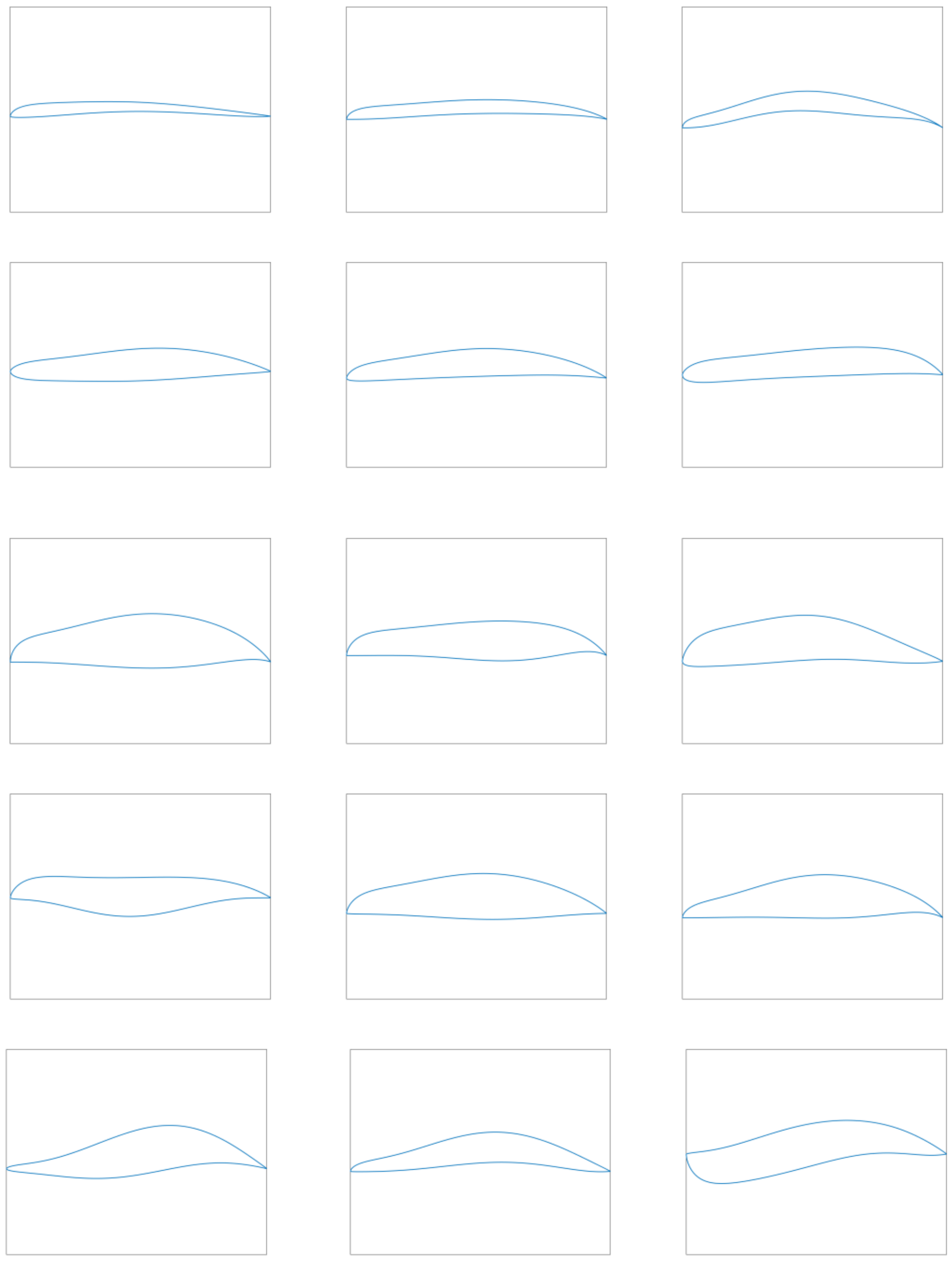


Figure 68: Airfoil Sections of Optimal Blades

Lastly, the airfoil shapes are examined between each one of the cases. Figure 68 provides the airfoil shape at 10%, 50% and 100% of the radius for each of the propellers. The first row provides airfoil shapes for the 2 Newton model, the second row provides results for the 3 Newton model and so on all the way to the 10 Newton model. The most notable trend that is drawn from this is the increased thickness in the airfoil shape as the thrust is increased. This same behavior was seen in the validation cases. Original geometry models over predicted thrust in the validation cases because the airfoil cross sections were too thick. This issue was eventually resolved, but it brought forth an important point that a thicker airfoil brings with it an increase in the thrust or lift produced. The same trend is seen here in the results for the GA. Second, notice the abstractness of the airfoil shapes as the thrust is increased. Until the thrust requirement reaches approximately 7 Newtons, the airfoils are reasonably shaped. This could be due to an issue in the GA where it is trying to converge on these unreasonable thrust demands from such small diameter propellers. Therefore, the GA applied more attention to the chord and twist values while paying little attention to the airfoil shapes.

7.2 Coaxial Model

The number of generations used in the coaxial models was truncated due to the extensive run times; however, appropriate convergence is still seen in each model due to the nature of the first generation. The GA for the coaxial model operated in the same manner as the single standalone application excepts for two distinct differences. The most notable of these is the absence of demes and the decreased number of generations. The run times for the coaxial simulation were too large to achieve full scale optimization from random generations to maximum members; therefore, the demes were neglected, and the number of generations was set to 10. In order to account for all the losses in generational strength, the coaxial GA used data from the stand-alone models. The first generation of the coaxial GA used the most optimal member from the 3-newton requirement GA execution. The first member of the first population was the 3-newton propeller exactly. The rest of the 127 members were created by using weakened power mutations on the same propeller parameters. By using an already optimized propeller only slight adjustments were required to optimize in a different but similar operating condition. These slight adjustments are expected to occur to a sufficient degree with in the provided 10 generations. The GA is then expected to finish all 10 generations in 6.86 days.

7.2.1 Setup

The optimized coaxial simulation was based on the validation case which is seen provided in Coaxial Propeller Validation section. Here the propellers were placed 7.5cm apart. The free stream velocity was set to 0m/s and the propeller rotational speeds were set with in the described limits of the validation. The forward propeller had a rotational speed of 6000RPM, and the rear optimized propeller had a rotational speed of 10,000 RPM. Both propellers had a diameter of 22.86 cm. An

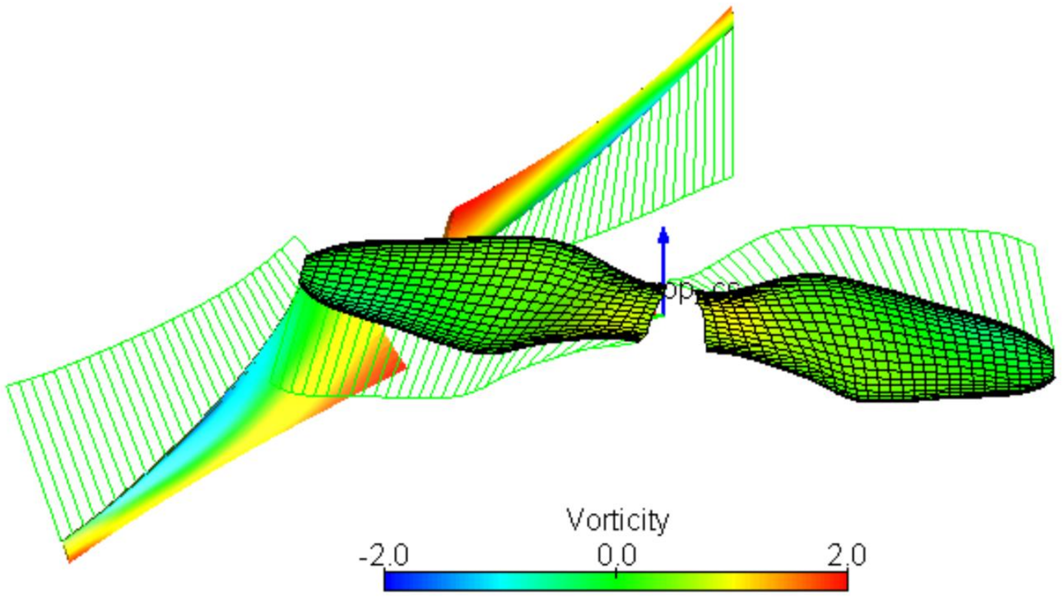


Figure 69: Example Coaxial Set up

example of the set up in the solver is provided. The solver was allowed to run for 80 iterations. The time step for each iteration was set to 0.0002 seconds. This time step and number of iterations allowed for the wake of the front propeller to come into full contact with the rear propeller and converge.

The fitness function for the coaxial simulation is slightly different than that of the single propeller model. Instead of having an efficiency and thrust based model, a torque and a thrust based model were used to determine fitness. This was done because the efficiency becomes zero in static environments such as the hover conditions. The overall fitness calculation remained the same with the efficiency being replaced with torque, and instead of maximizing the efficiency, the torque was minimized. A weight of 4 was given to the torque values and a weight of 7 and 6 were provided to the thrust values.

7.2.2 Coaxial Results

The coaxial GA was executed for two different thrust requirements: 10N and 16N. The 16N model had a thrust weight of 7 while the 10N model had a weight of 6. To provide some perspective, the

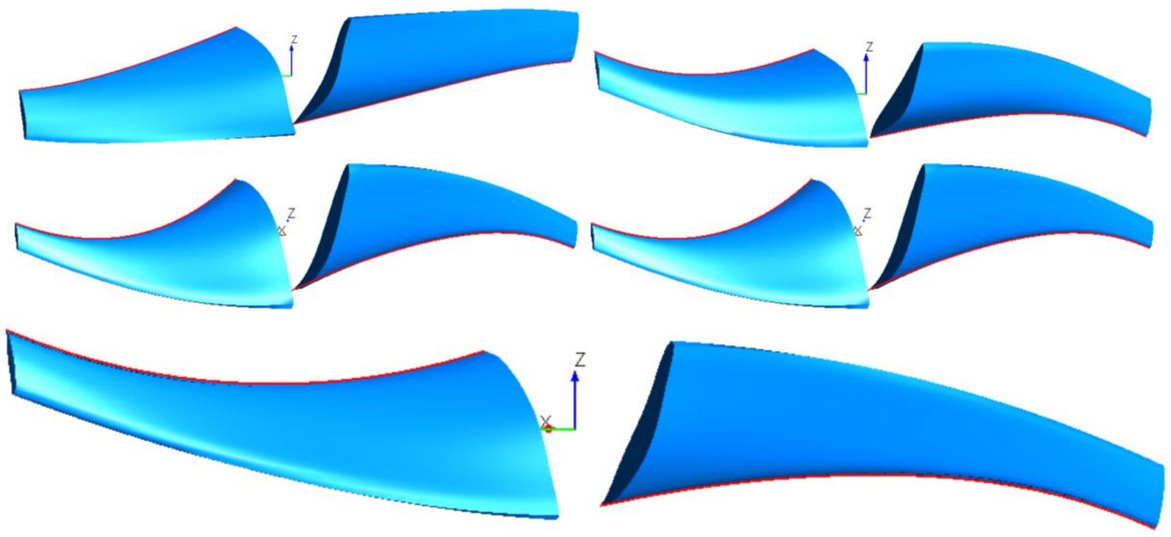


Figure 70: 16N Thrust Requirement Propellers at Generations 1, 3, 5, 7, and 10 (from left to right and top to bottom)

validated propeller at these exact operating conditions produced ~8.8N with a torque of 0.170 N-m. The 16N results are provided first in Figure 70. The maximum propeller from each of the generations is provided. All these propellers have 15.49N to 15.98N and torque values from 0.4083N-m to 0.4524N-m. One notable characteristic that the GA favors is the sweep of the propeller. As the GA moves through the generations, the propellers have more and more sweep. This swept back blade is typical for efficient propellers.

There are cases of non-convergence in the solver where the same propeller will produce different values of thrust and torque which are provided in Figure 71. Due to the inherently unsteady flow that is introduced by the front propeller, the rear propeller tends to oscillate in its

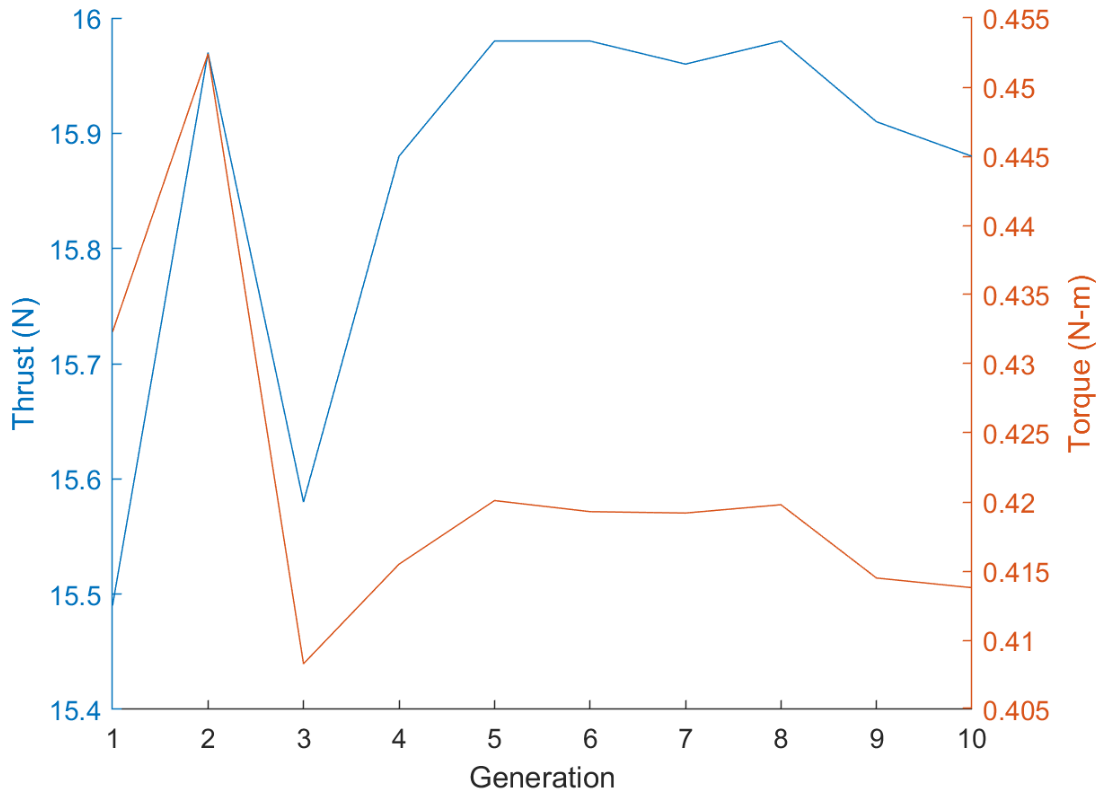


Figure 71: Thrust and Torque vs Generation

values for torque and thrust. This can lead to errors in the solver where the maximum performing propeller performs less than it had originally which then allows a less dominant propeller to pass its genes to the next generation. Nevertheless, the values for the plot deviate very little. The final propeller from the 10 generations had a thrust of 15.88N and a torque of 0.4138N-m.

The 10N model produced results similar to the 16N requirement only scaled down so that the model would meet the required thrust. The 10N model started with in the same manner as the 16N model by mutating the stand-alone 3N model. The results from the 10 generations are provided below in Figure 72. The first noticeable difference between the two is the amount of

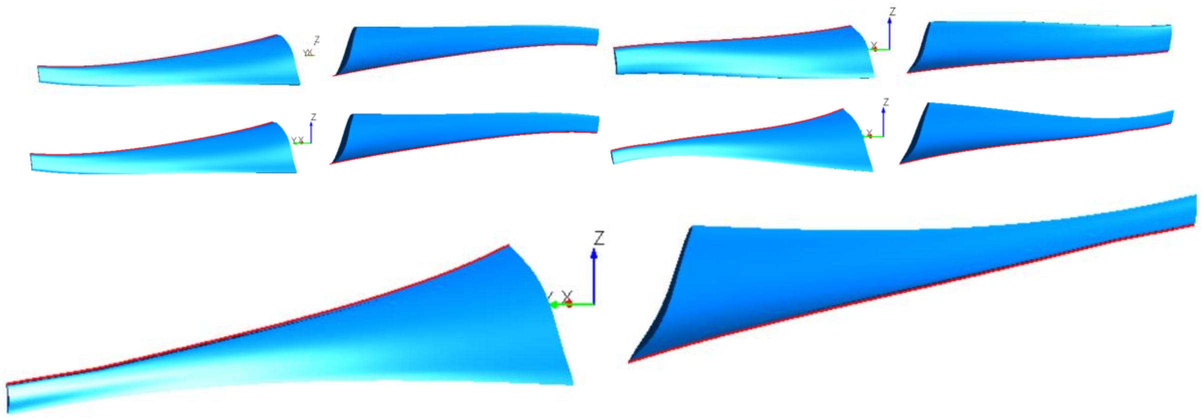


Figure 72: 10N Thrust Requirement Propellers at Generations 1, 3, 5, 7, and 10 (from left to right and top to bottom)

sweep that the propellers took on. Ideally, the GA would have found a swept back blade to me be the most optimial; however, the GA focused on decreasing the chord at the tip of the propeller. The chord lengths are also decreased from the 16N model which is an obvious geometry difference given the reduction in the thrust requirement. While the airfoils that the two models converged are similar, the 10N model found that by decreasing the thickness of the airfoil it could obtain a propeller which produced less torque. The propeller, however, is limited in the model to a given thickness so, that top and bottom airfoil curves are not over lapping.

The geometric parameters for both propellers are plotted against the radial values in Figure 73. All of the results for the 10N model are given in red while the 16N results are provided in blue.

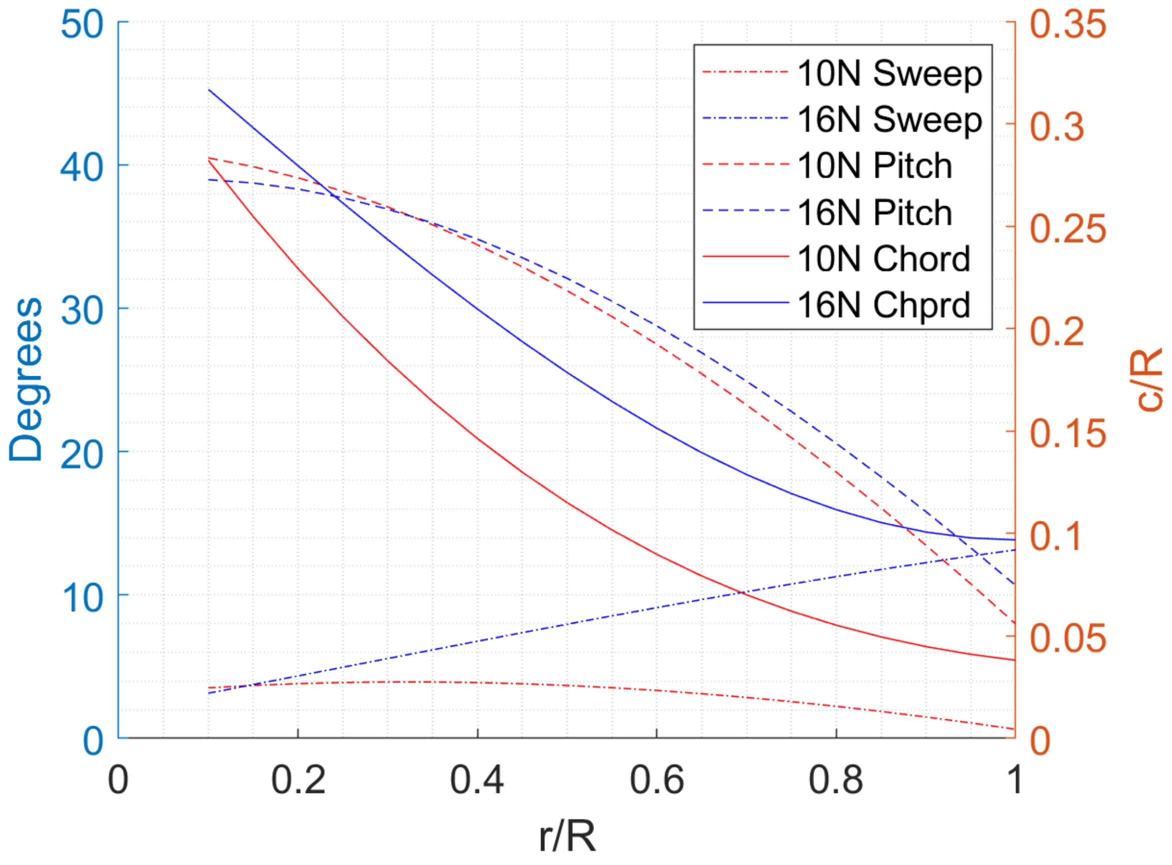


Figure 73: Geometry Values for 10N and 16N Thrust Requirement

Both propeller models demonstrate the same relationship with the chord values. Infact, the 16N model chord values are almost a linear increase from the 10N models. The pitch for both models is identical which implies that the difference in thrust values was accounted for in the changes in the chord only. The sweep for the 16N model is more reasonable with respect to what is generally seen for efficient propellers while the 10N model takes on a more unique form. The 3-N stand-alone model which was used to produce the first geometry did not take on these sweep values; therefore, the attribute was picked up in the generations. Perhaps a non-converging result was provided by the solver leaving the GA with faulty data. Nevertheless, both geometric characteristics follow typical values and show improvements in the performance.

Chapter 8: Conclusions and Remarks

Using an advanced real coded genetic algorithm which takes advantage of demes and mutation directions and a computational efficient solver, a robust real coded genetic algorithm has been developed for propeller blade shape designs in both steady and unsteady flight conditions. The solver was first validated using performance data from the UIUC propeller database, NACA Technical Report 640, and wind tunnel data for a coaxial rotor configuration. The results from the GA prove to be reasonable and within an expected range provided the performance data from wind tunnel experiments and existing knowledge of propellers. The steady propeller optimization designs were permitted to run over 140 total generations with 40 of those generations comprising the demes. The coaxial models only ran over 10 generations; however, designs were not constructed from a totally random population like the stand-alone models. The unsteady, coaxial initial population was developed from one of the stand-alone model convergences. Thus, an already optimized propeller design was used in the initial generations, and only small adjustments were required to optimize the propeller for a different flight condition. Furthermore, with 128 members per generation it is reasonable to say that 10 generations would provide enough time for convergence.

There are a several areas of work that still need to be delt with to maximize the efficiency of the GA and solver itself. The first of these is the computational power and efficiency that was used in these executions. While the coaxial simulation simultaneously ran members of the population the steady, stand-alone models were run one at a time. Now, this is not to say that only one model was run at once. By running each of the members one at a time, multiple GA thrust requirements were run on the same computer which ultimately reduced the total run time; however, for even faster results, the stand-alone model should be altered such that it operates in the same manner as the coaxial where it runs multiple sets of observations at a time. From a simple calculation, it is estimated that a single propeller model with 128 members per generation over 140 generations can be completed in less than 2 days whereas the current model takes just over 6 days to complete. The second improvement which can be in regard to computational efficiency is the amount of computational power itself. The current machine that the GA is running on has 28 cores; therefore, 7 runs were executed simultaneously providing 4 cores to each. By increasing the number of cores available, a full-scale coaxial optimization is plausible.

It is mentioned in this work that the optimization does have built in which it focuses on variable pitch propellers. While the results are not provided in the work due to a larger focus on the unsteady problem, the variable pitch propeller model is very useful and not just in the sense of propeller itself. The variable pitch model focuses on four different flight conditions, therefore, even if the propeller was not variable pitch, the multiple point optimization would be very useful for different applications. A future study of this work should consist of the variable pitch model applied to aircraft or perhaps focus on a constant pitch propeller with differing flight conditions.

The last suggestion for the future of this work and propeller optimization in general is the most interesting and arguably the most important. During the propeller validation process, it was discovered to be quite arduous in taking a scanned propeller and turning it into a suitable geometry for testing, but it was shown to be possible. Furthermore, during the development of the coaxial simulation, the idea of using an already optimized propeller to build the initial generation gave a spark to this process. The argument made here is that propellers in general need to be optimized for the given advances in electric rotor craft for UAM and UAV applications. Many efficient and well performing propellers exist; however, none perfectly fit the desired flight conditions therefore, there exists the need for the optimization scheme. It would be very efficient and effective to have an algorithm which matches a Bernstein polynomial to a given point cloud collected from a scan. There are 63 parameters to predict in this case just as the problem described in this paper, so the GA here can handle the number of parameters. The point cloud would offer the most optimal configuration of points which the GA is working towards constructing using the BP. The evaluation times would be short compared to the solver run times, and the fitness function would be based off the accuracy of the geometry curves from the BP to the point cloud. In summary, the user would simply input the same file as described in appendix II as well as a point cloud or perhaps even a matrix which contains the points from the point cloud. This scanned propeller would then be optimized given a flight condition. It is suggested that the most efficient way to match the point cloud geometry would be to generate geometry curves (chord, twist, sweep) as a function of the radius and match these with the BP. The airfoil development should only be done at certain section. Optimization of the entire propeller and all the point cloud points would prove to be difficult. After all, it will be sent through an optimization process, so the geometry only needs to be similar not 100% correct.

References

[1] McDonald, Robert A., “Modeling Electric Motor Driven Propellers for Conceptual Aircraft Design,” AIAA SciTech 2015-1676, January 2015.

[2] McDonald, Robert A., “Electric Propulsion Modeling for Conceptual Aircraft Design,” AIAA SciTech 2014-0536, January 2014

[3] Holland, J., “Genetic Algorithms,” *Scientific American*, Vol. 267, No. 1, July 1992, pp. 66-73.

[4] D. Kalyanmoy, “An Introduction to Genetic Algorithms,” *Sadhana*, Vol. 24, No. 4 & 5, August 1999.

[5] Anderson, M., “User’s Manual for Improve Version 3.0,” Sverdrup Technology Inc./TEAS Group, January 2008

[6] Miettinen, K., Makela, M., and Toivanen, J., “Numerical Comparison of Some Penalty-Based Constraint Handling Techniques in Genetic Algorithms,” *Journal of Global Optimization*, Vol. 27, April 2003, pp. 427-446.

[7] Anderson, M., “Genetic Algorithms in Aerospace Design: Substantial Progress, Tremendous Potential,” Sverdrup Technology Inc./TEAS Group, May 2002.

[8] Anderson, M., Burkhalter, J., and Jenkins, R., “Design of a Ground-Launched Ballistic Missile Interceptor Using a Genetic Algorithm,” Sverdrup Technology Inc., TEAS Group and Auburn University

[9] Holland, J., “Genetic Algorithms,” *Scientific American*, Vol. 267, No. 1, July 1992, pp. 66-73.

[10] Goldberg, D., “Genetic Algorithms for Search, Optimization, and Machine Learning,” , Reading, MA: Adision-Wesley, 1989.

[11] Adekanmbi, O., and Green, P., “Conceptual Comparison of Population Based Metaheuristics for Engineering Problems,” *The Scientific World Journal*, October 2014.

[12] Burger, C., “Propeller Performance Analysis and Multidisciplinary Optimization Using a Genetic Algorithm,” Ph. D Dissertation, Auburn University, Auburn, AL, December 2007

[13] Aref, P., Ghoreyshi, M., Jirasek, A., Satchell, M., and Bergeron, K., “Computational Study of Propeller-Wing Aerodynamic Interaction,” *Aerospace*, Vol. 5, No. 79, 2018.

[14] Doyle, J., Hartfield, R., and Roy., C., “Aerodynamic Optimization for Freight Trucks Using a Genetic Algorithm,” AIAA 2008-323, Reno, NV, January 2008.

[15] Burger, C., “Propeller Performance Analysis and Multidisciplinary Optimization Using a Genetic Algorithm,” Ph. D Dissertation, Auburn University, Auburn, AL, December 2007

[16] Drela, M. “XROTOR User Guide”, 2003.

[17] Alba, C., Elham, A., German, B. J., and Veldhuis, L., “A Surrogate-Based Multi-Disciplinary Design Optimization Framework Modeling Wing-Propeller Interaction,” *Aerospace Science and Technology*, Vol. 78, Elsevier Masson SAS, 2018, pp. 721-733

[18] Drela, M. “QPROP User Guide”, 2007

[19] Ahuja, V., “Aerodynamic Loads Over Arbitrary Bodies by Method of Integrated Circulation,” Ph.D. Dissertation, Aerospace Engineering Dept., Auburn Univ., Auburn, AL, 2013

[20] Olson, E. D., and Albertson, C. W., “Aircraft High-Lift Aerodynamic Analysis Using a Surface Vorticity Solver”, AIAA 2016-0779, January 2016

[21] Vivek Ahuja and R. J. Hartfield. "Aerodynamic Loads over Arbitrary Bodies by Method of Integrated Circulation", Journal of Aircraft, Vol. 53, No. 6 (2016), pp. 1719-1730

[22] Forrest, S., “Genetic Algorithms,” *ACM Computing Surveys*, Vol. 28, No. 1, March 1996.

[23] Goldberg, D., “Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking,” University of illinios at Urbana-Champaign, Department of General Engineering.

[24] Koza, J., “Genetic Programming,” The MIT Press, Cambridge, MA, 1992.

[25] R. Arora, R. Tulshyan and K. Deb, "Parallelization of binary and real-coded genetic algorithms on GPU using CUDA," *IEEE Congress on Evolutionary Computation*, 2010, pp. 1-8

[26] Roopesh, K., Umesha, P., and Kalappa, M.S., “Software Based on Heuristic Technique for Optimization of Transmission Line Towers,” *Journal of Structural Engineering*, Vol. 33, No. 2, June 2006, pp. 1-10.

[27] Deep, Kusum, Singh, Krishna, Kansal, and Mohan, C., “A Real Coded Genetic Algorithm for Solving Integer and Mixed Integer Optimization Problems,” *Applied Mathematics and Computation*, Vol. 212, Issue 2, 2009, pp. 505-518

[28] Deep, Kusum, Singh, Krishna, Kansal, and Mohan, C., “A Real Coded Genetic Algorithm for Solving Integer and Mixed Integer Optimization Problems,” *Applied Mathematics and Computation*, Vol. 212, Issue 2, 2009, pp. 505-518

[29] Deep, K., and Thakur, M., “A New Mutation Operator for Real Coded Genetic Algorithms,” *Applied Mathematics and Computation*, 193 (2007) 211-230, 2007

[30] Deep, K., and Thakur, M., "A New Mutation Operator for Real Coded Genetic Algorithms," *Applied Mathematics and Computation*, 193 (2007) 211-230, 2007

[31] Fikret Tokan and Filiz Gunes, "The Multi-Objective Optimization of Non-Uniform Linear Phased Arrays Using the Genetic Algorithm," Progress In Electromagnetics Research B, Vol. 17, 135-151, 2009.

[32] Miettinen, K., Makela, M., and Toivanen, J., "Numerical Comparison of Some Penalty-Based Constraint Handling Techniques in Genetic Algorithms," *Journal of Global Optimization*, Vol. 27, April 2003, pp. 427-446.

[33] Britt, W. "A Meta-Parallel Evolutionary System for Solving Optimization Problems," Master's Thesis, Computer Science and Software Engineering Dept., Auburn University., Auburn AL, 2007.

[34] Dozier, G. "Distributed Steady-State Neuro-Evolutionary Path Planning in Non-Stationary Environments Using Adaptive Replacement," Auburn University, Auburn, AL, Department of Computer Science and Software Engineering

[35] Xiang, S., Liu, Y., Tong, G., Zhao, W., Tong, S., and Li, Y., "An Improved Propeller Design Method for the Electric Aircraft," *Aerospace Science and Technology*, Vol. 78, July 2018, pp. 488-493.

[36] Carroll, T. J., "Wright Brothers' Invention of the 1903 and Genesis of Modern Propeller Theory," *Journal of Aircraft*, Vol. 42, No. 1, January 2005.

[37] McDonald, R. A., "Modeling Electric Motor Driven Propellers for Conceptual Aircraft Design," AIAA SciTech 2015-1676, January 2015.

[38] McDonald, R. A., "Electric Propulsion Modeling for Conceptual Aircraft Design," AIAA SciTech 2014-0536, January 2014

[39] Wald, Q. R., "The Aerodynamics of Propellers," *Progress in Aerospace Sciences* 42 (2006) 85-128.

[40] Ash, R. L., Miley, S. J., Landman, D., and Hyde, K. W., "Evolution of Wright Flyer Propellers between 1903 and 1912," AIAA 2001-0309 January 2001.

[41] Miley, S. J., Ash, R. L., Hyde, K. W., Landman, D., and Sparks, A. K., "Propeller Performance Tests of Wright Brothers' "Bent-End" Propellers," *Journal of Aircraft*, Vol. 39, No. 2, March 2002.

[42] Philips, W. F., and Snyder, D. O., "Modern Adaptation of Prandtl's Classic Lifting-Line Theory," *Journal of Aircraft*, Vol. 37, No. 4, July 2000.

[43] Mahmuddin, F., "Rotor Blade Performance Analysis with Blade Elements Momentum Theory," *Energy Procedia* 105 (2017) 1123-1129.

[44] Epps, B. P., and Kimball, R. W., "Unified Rotor Lifting Line Theory," *Journal of Ship Researc*, Vol. 57, No. 4, December 2013.

[45] Bacon, D., "Variable Pitch Propellers," NACA TM 2, September 1920.

[46] Heinzen, S., Hall, C., and Gopalarathnam, A., "Development and Testing of a Passive Variable-Pitch Propeller," *Journal of Aircraft*, Vol. 25, No. 3, May 2015.

[47] Pistolesi, E., "Variable Pitch Propeller," NACA TM 216, July 1923.

[48] Holzsager, J. E., "The Effects of Coaxial Propellers for the Propulsion of Multirotor Systems," Master's Thesis, Aerospace Engineering Dept, Rutgers University, New Brunswick, NJ, 2017.

[49] DuWalt, F. A., "Wakes of Lifting Propellers (Rotors) in Ground Effect," ONR Contract No. 3691(00) Final Report, November 2017.

[50] Drela, M., "Boundary Layer Analysis" *Flight Vehicle Dynamics*, The MIT Press, Cambridge, MA., 2014

[51] Kundu, P., Cohen, I., and Dowling, D., "Boundary Layers and Related Topics," *Fluid Mechanics*, 6th ed. Elsevier Inc., New York, NY, 2012

[52] Kundu, P., Cohen, I., and Dowling, D., "Boundary Layers and Related Topics," *Fluid Mechanics*, 6th ed. Elsevier Inc., New York, NY, 2012

[53] Anderson, J., "Inviscid, Compressible Flow," *Fundamentals of Aerodynamics*, 6th ed. McGraw Hill, New York, NY, 2017.

[54]Olson, E. D., "Three-Dimensional Modeling of Aircraft High-Lift Components with Vehicle Sketch Pad." AIAA 2016-1274, January 2016.

[55] McDonald, R. A., "Advanced Modeling in OpenVSP," AIAA 2016-3282, June2016.

[56] Kulfman, B., and Bussoletti, J., "Fundamental Parametric Geometry Representations for Aircraft Component Shapes," AIAA-2006-6948, Portsmouth, VA, September 2006.

[57] Burger, C., "Propeller Performance Analysis and Multidisciplinary Optimization Using a Genetic Algorithm," Ph. D Dissertation, Auburn University, Auburn, AL, December 2007

[58] Lane, K. and Marshall, D., "Inverse Airfoil Design Utilizing CST Parameterization," AIAA 2010-1228, January 2010

[59] Ahuja, V., Hartfield, R., “Predicting the Aero Loads Behind a Propeller in the Presence of a Wing Using FlightStream®,” AIAA 2015-2734, presented at the AIAA Aviation 2015 Conference, Dallas, June 2015.

[60] Ahuja, Vivek, Hartfield, Roy, and Burkhalter, John, “Optimizing Engine Placement on an Aircraft Wing using Biomimetic optimization and FlightStream®”, AIAA *2017-0235,* Proceedings of AIAA SciTech 2017, Dallas, TX, January 2017

[61] Johnson, S., Hartfield, R., van Dommelen, D, and Ahuja, V., “Investigation of the Static Longitudinal and Lateral Characteristics of a Full-Scale Light Single-Engine Airplane using a Surface Vorticity Solver and CFD” AIAA 2018-1259, Presented at the AIAA SciTech Conference, Orlando, FL, January 2018

[62] Sandoz, B., Ahuja, V., and Hartfield, R., “Longitudinal Aerodynamic Characteristics of a V/STOL Tilt-wing Four-Propeller Transport Model using a Surface Vorticity Flow Solver”, AIAA 2018-2070, Presented at the AIAA SciTech Conference, Orlando, FL, January 2018.

[63] Ahuja, V., “FlightStream® User Guide,” 2020.

[64] Pastor, G., Hartfield, R., Ahuja, V., and McClearen, J., “Numerical and Experimental Testing of a Coaxial Propeller for UAM Applications,” Aviation 2022, June 2022. (Submitted for publication)

[65] Ahuja, V. and Hartfield, R, “Reduced-Order Aerodynamics with the Method of Integrated Circulation,” AIAA SciTech 2022, January 2022.

[66] Ahuja, V., and Hartfield, R., “Predicting the Aerodynamic Loads behind a Propeller in the presence of a wing using FlightStream,” AIAA 2015-2734, Dallas, Tx, June 2015.

[67] Ahuja, V., “Aerodynamic Loads Over Arbitrary Bodies by Method of Integrated Circulation,” Ph.D. Dissertation, Aerospace Engineering Dept., Auburn Univ., Auburn, AL, 2013

[68] Darve, E., “The Fast Multipole Method: Numerical Implementation,” *Journal of Computational Physics,* Vol. 160, No. 1, May 2000, pp. 195-240.

[69] Kebbie-Anthony, A., Gumerov, N., Preidikman, S., Balachandran, B., and Azarm, S., “Fast Multipole Method for Nonlinear, Unsteady Aerodynamic Simulations,” Sci Tech 2018, Kissimmee, Florida, January 2018.

[70] Wolf, W. and Lele, S., “Aeroacoustics Integrals Accelerated by Fast Multipole Method,” *AIAA Journal*, Vol. 49, No. 7, July 2011.

[71] Ahuja, V. and Hartfield, R, “Reduced-Order Aerodynamics with the Method of Integrated Circulation,” AIAA SciTech 2022, January 2022.

[72] Standen, N., “Calculation of Integral Parameters of a Compressible Turbulent Boundary Layer Using a Concept of Mass Entrainment,” Master’s Thesis, Department of Mechanical Engineering, McGill University, Montreal, 1964.

[73] Ahuja, V., and Hartfield, R. “Novel Viscous Surface Vorticity Method for Fast Aerodynamic Analysis of Road Vehicles”, Aerovehicles (4), August 2021

[74] Gavin, A., “Release Notes”, *UIUC Propeller Database* [online database], URL: https://m-selig.ae.illinois.edu/props/volume-1/releaseNotes.html [retrieved 08 February 2021].

[75] Brandt, J. B., and Selig, M. S., “Propeller Performance Data at Low Reynolds Numbers.” AIAA 2011-1255, January 20011

[76] Gavin, A., “Release Notes”, *UIUC Propeller Database* [online database], URL: https://mselig.ae.illinois.edu/props/volume-1/propDB-volume-1.html [retrieved 08 February 2021].

[77] Polo, M.E., Cuartero, A. and Felicísimo, Á.M., 2019. “Study of uncertainty and repeatability in structured-light 3D scanners”. arXiv preprint arXiv:1910.13199.

[78] Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., and Ranzuglia, G., “MeshLab: an Open-Source Mesh Processing Tool,” Eurographics Italian Chapter Conference, 2008

[79] Gavin, A., “Release Notes”, *UIUC Propeller Database* [online database], URL: https://mselig.ae.illinois.edu/props/volume-1/propDB-volume-1.html [retrieved 08 February 2021].

Appendix I: Geometry Transformation

The complete process for developing a propeller's cross sections from a scan are detailed in this section. The point cloud is first loaded into MeshLab as a .ply file, shown in top of Figure 74. While the scans in the validation case consist of a top scan and a bottom scan of the propeller, it is suggested that the data be gathered in one scan for better accuracy. If this cannot be completed

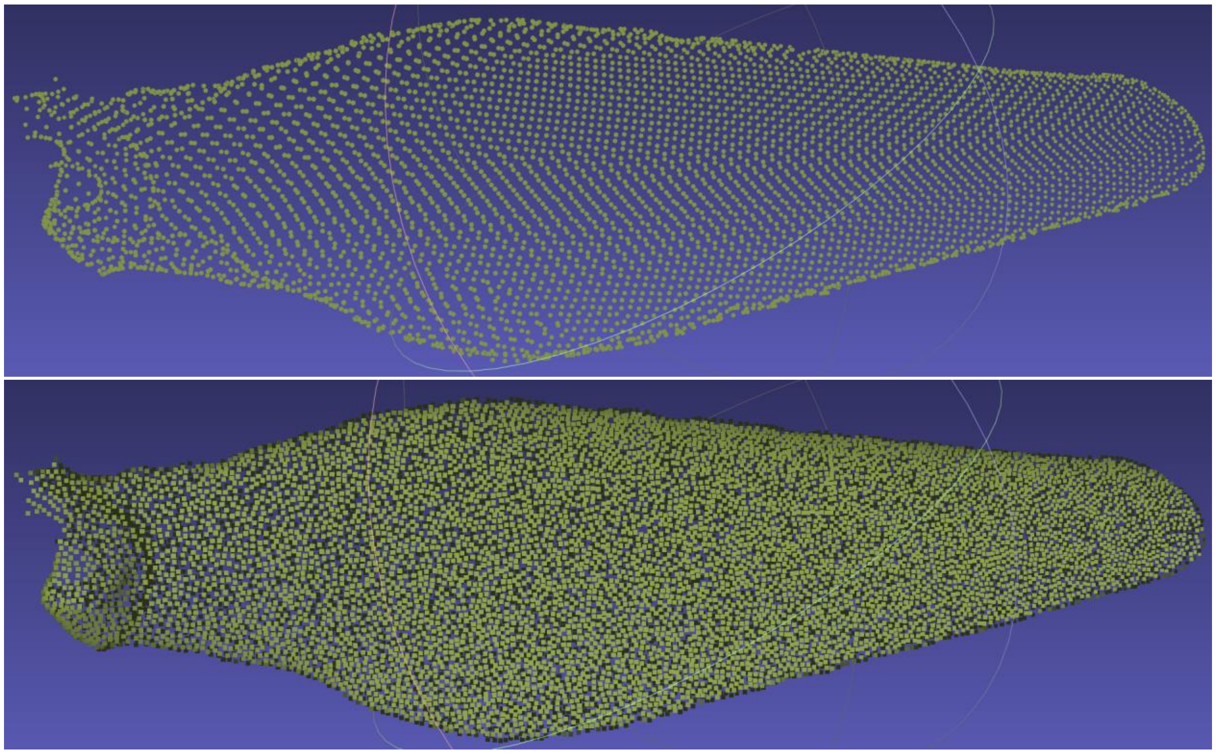
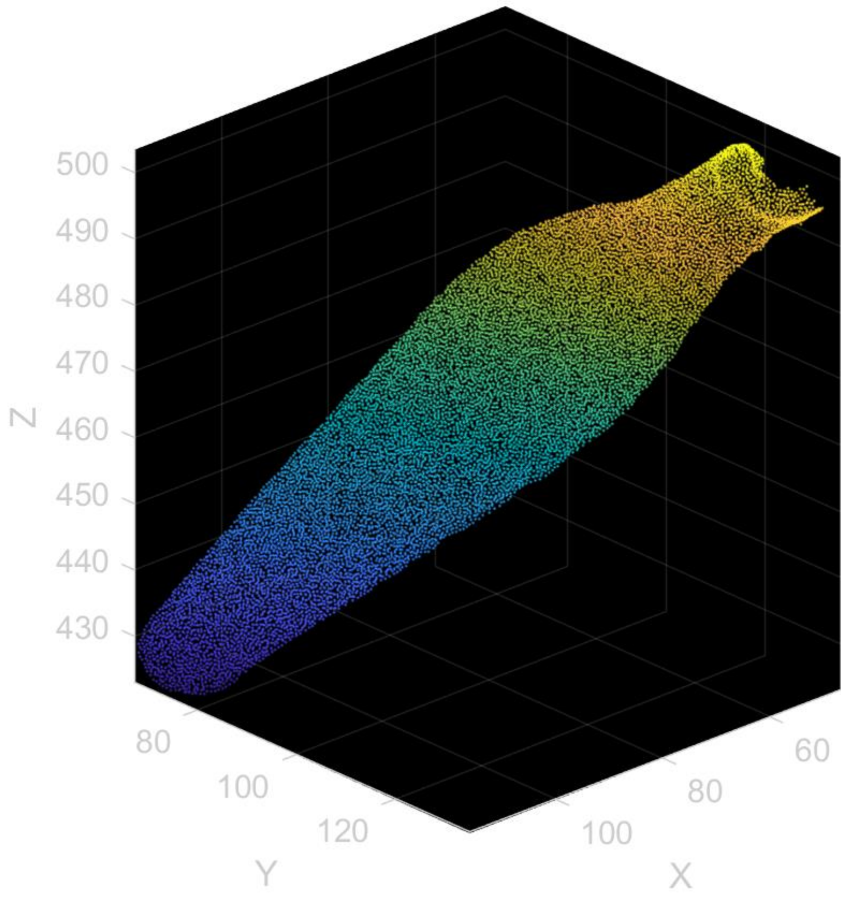


Figure 74: Point Clouds from Propeller Scan

in one scan, the process follows the exact same steps as described in this section with the exception of adding a step in which the top and bottom scans are put together. Once the original scan is loaded, the number of points is increased using the Poison disk-sampling tool in MeshLab is used to increase the number of points. This adjustment can be observed in bottom of **Error! Reference source not found.**. With the number of points increased to an appropriate amount, the .ply file is loaded into MATLAB using the built-in function. Once the file is ply is loaded into MATLAB, it is transformed and rotated to be oriented along an axis to allow for easier data collection. An



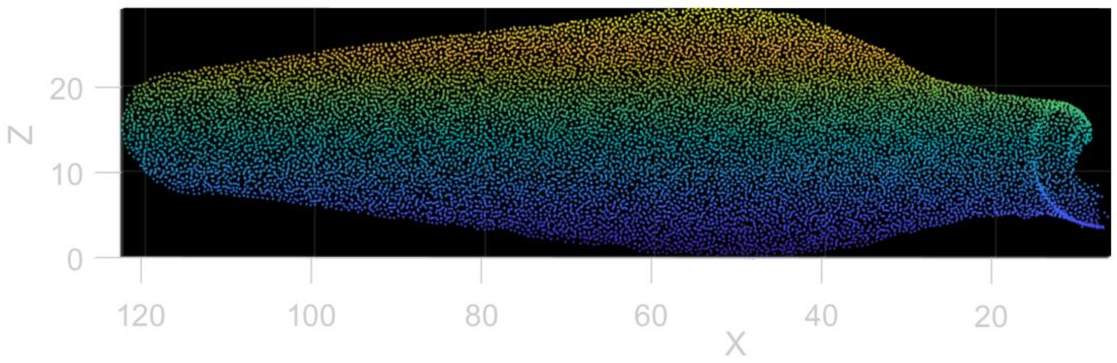
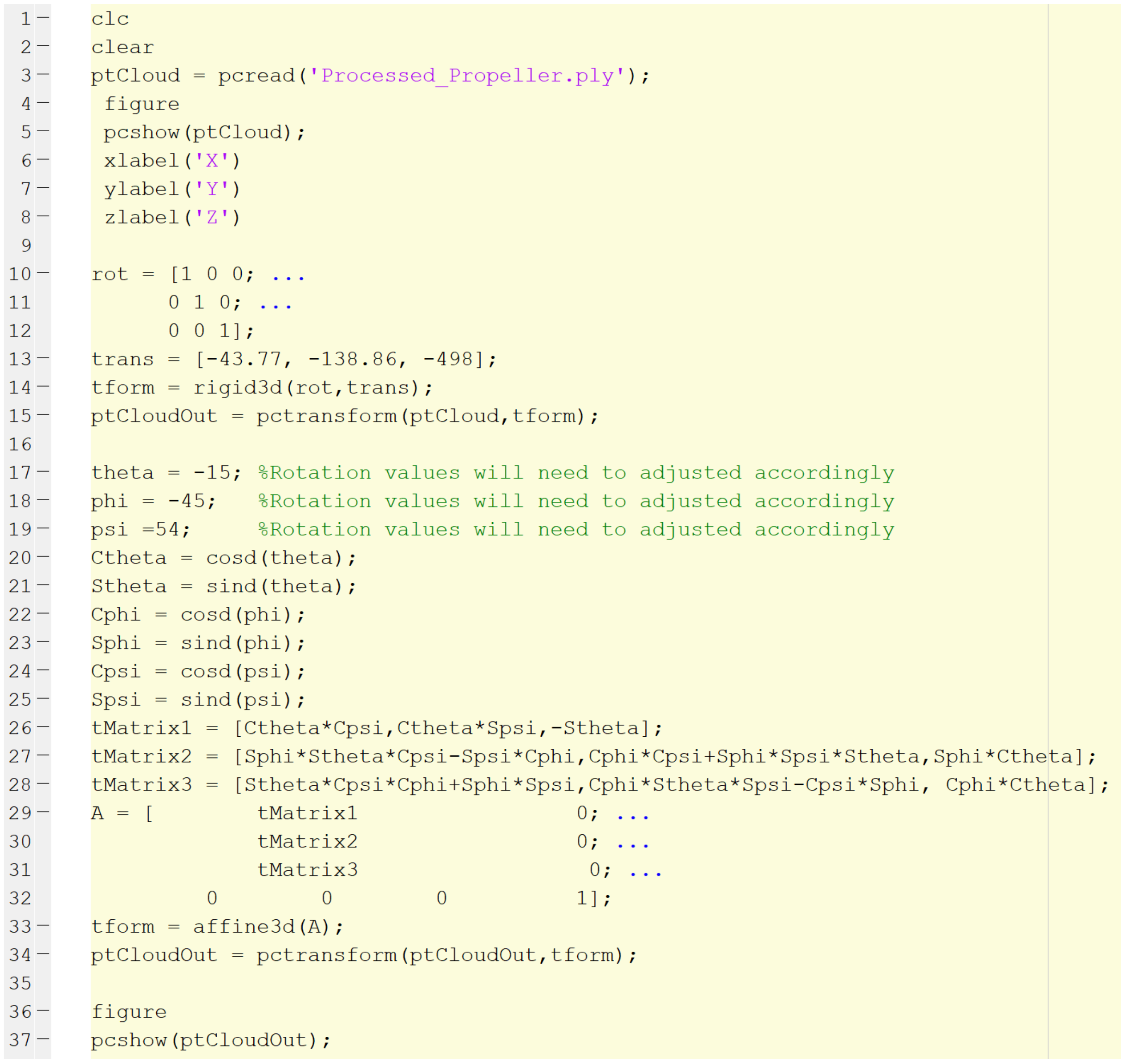


Figure 75: MATLAB Point Cloud Rotation

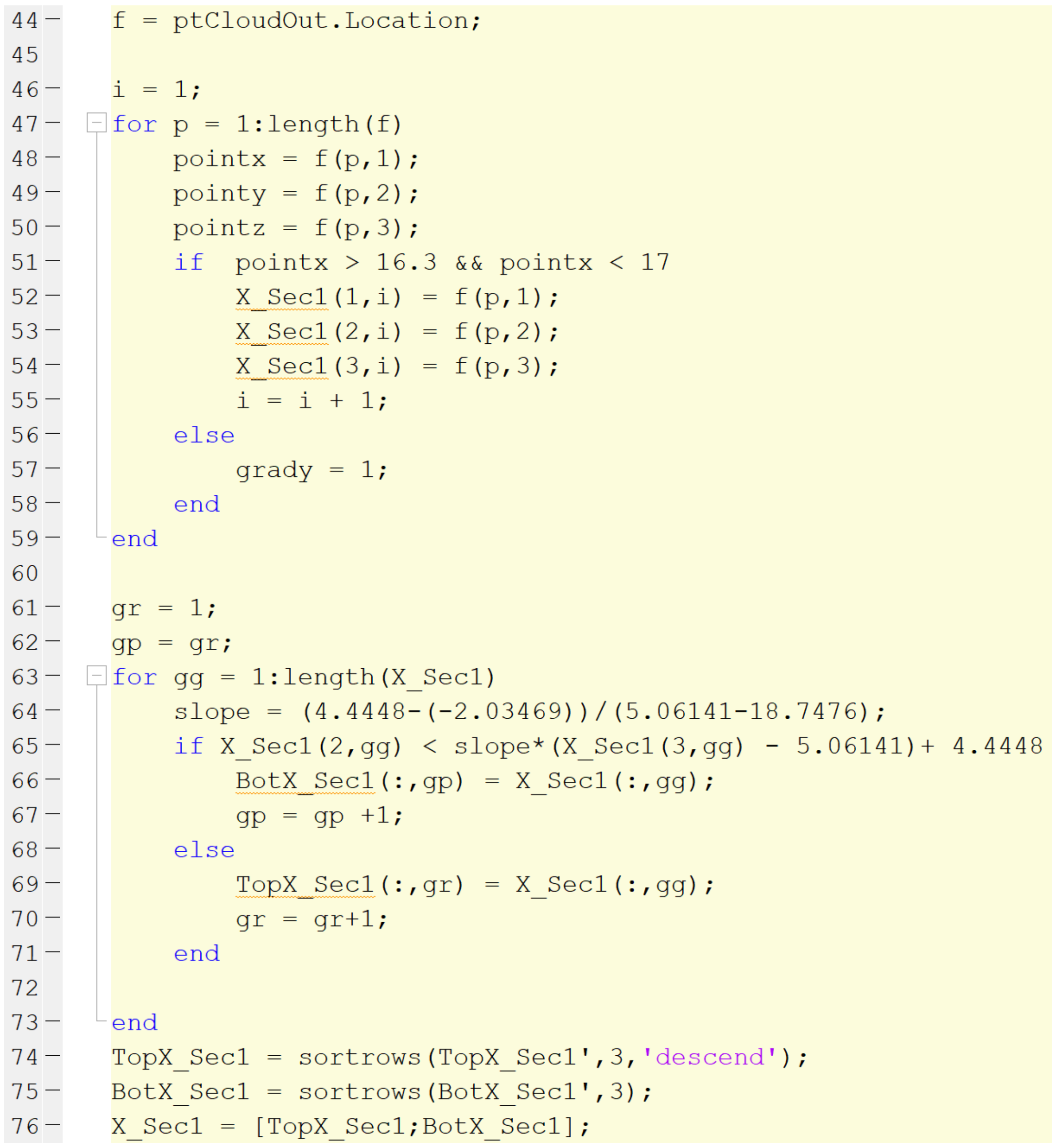
example of this rotation is provided in Figure 75. The rotation is done by using the built in MATLAB “affine3d” this is the function that rotates the point cloud using a specified matrix. The full code is provided in Figure 76. Lines 3-8 read and display the point cloud as it is directly provided. Lines 10-15 conduct the translation of the point cloud so that the hub of the propeller is at the origin. Lines 17-28 provide the rotation angles and the rotation matrix used. Lines 29-33 create the matrix used in the “affine3d” function as well as create the rotation variable, “tform”.



```
 1    clc
 2    clear
 3    ptCloud = pcread('Processed_Propeller.ply');
 4     figure
 5     pcshow(ptCloud);
 6     xlabel('X')
 7     ylabel('Y')
 8     zlabel('Z')
 9
10    rot = [1 0 0; ...
11          0 1 0; ...
12          0 0 1];
13    trans = [-43.77, -138.86, -498];
14    tform = rigid3d(rot,trans);
15    ptCloudOut = pctransform(ptCloud,tform);
16
17    theta = -15; %Rotation values will need to adjusted accordingly
18    phi = -45;   %Rotation values will need to adjusted accordingly
19    psi =54;     %Rotation values will need to adjusted accordingly
20    Ctheta = cosd(theta);
21    Stheta = sind(theta);
22    Cphi = cosd(phi);
23    Sphi = sind(phi);
24    Cpsi = cosd(psi);
25    Spsi = sind(psi);
26    tMatrix1 = [Ctheta*Cpsi,Ctheta*Spsi,-Stheta];
27    tMatrix2 = [Sphi*Stheta*Cpsi-Spsi*Cphi,Cphi*Cpsi+Sphi*Spsi*Stheta,Sphi*Ctheta];
28    tMatrix3 = [Stheta*Cpsi*Cphi+Sphi*Spsi,Cphi*Stheta*Spsi-Cpsi*Sphi, Cphi*Ctheta];
29    A = [        tMatrix1                 0; ...
30                 tMatrix2                 0; ...
31                 tMatrix3                  0; ...
32             0        0        0          1];
33    tform = affine3d(A);
34    ptCloudOut = pctransform(ptCloudOut,tform);
35
36    figure
37    pcshow(ptCloudOut);
```

Figure 76: Code for Point Cloud Translation and Transformation

Line 34 executes the transformation, and lines 36-37 display the transformed the point cloud. Once transformed such that the point cloud is oriented on an axis with the hub positioned at the origin, the cross-sectional data can be collected using sets of conditionals. The information containing each of the data points is stored in a structure of whatever the point clouds name is under the field "Location" i.e., the data for each point is found in variable "*PointCloudName*.Location". The information in the Location matrix is stored as x, y, and z coordinates. Therefore, all of the points



```
f = ptCloudOut.Location;

i = 1;
for p = 1:length(f)
    pointx = f(p,1);
    pointy = f(p,2);
    pointz = f(p,3);
    if  pointx > 16.3 && pointx < 17
        X_Sec1(1,i) = f(p,1);
        X_Sec1(2,i) = f(p,2);
        X_Sec1(3,i) = f(p,3);
        i = i + 1;
    else
        grady = 1;
    end
end

gr = 1;
gp = gr;
for gg = 1:length(X_Sec1)
    slope = (4.4448-(-2.03469))/(5.06141-18.7476);
    if X_Sec1(2,gg) < slope*(X_Sec1(3,gg) - 5.06141)+ 4.4448
        BotX_Sec1(:,gp) = X_Sec1(:,gg);
        gp = gp +1;
    else
        TopX_Sec1(:,gr) = X_Sec1(:,gg);
        gr = gr+1;
    end

end
TopX_Sec1 = sortrows(TopX_Sec1',3,'descend');
BotX_Sec1 = sortrows(BotX_Sec1',3);
X_Sec1 = [TopX_Sec1;BotX_Sec1];
```

Figure 77: Cross Section Collection

in the matrix are analyzed through a loop. If the value of a point lies within the given constraints for a cross section it is stored to build the geometry of that cross section. An example of the first cross section collection loop is provided in Figure 77. The first line defines a variable, “f”, that contains the data for all points in the point cloud. The first “for” loop runs through each point in the point cloud. If the x-value location (radial value) is in between the given range for the cross section, the x, y, and z coordinates for that point are stored in a matrix called “X_Sec1” which contains all the points for the first cross section. The second “for” loop divides the cross section into upper and lower sections. The airfoil file used in OpenVSP has to have the bottom and top curve separated; therefore, the division is made here by drawing a line through the airfoil and separating points along the top and bottom. This procedure can be seen in Figure 78. If points are

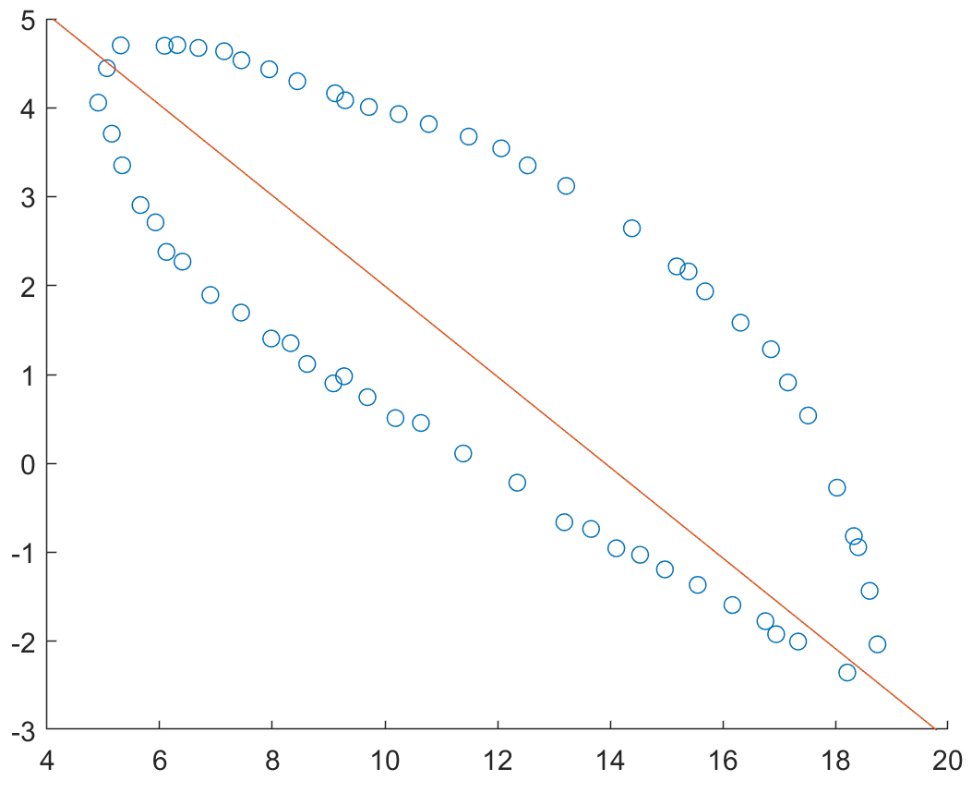


Figure 78: Airfoil Top and Bottom Division

above the red line, they are placed in the upper airfoil curve, and if they are below the red line, they are placed in the lower airfoil curve. The entirety of this code shown in Figure 77 and is done for each cross section. Once all the cross sections have been obtained, they are translated such that the leading edge of each airfoil is at its own respective origin and the trailing edge is on the x axis. To do this, all the points that make a cross section or airfoil are adjusted by the leading-edge point.

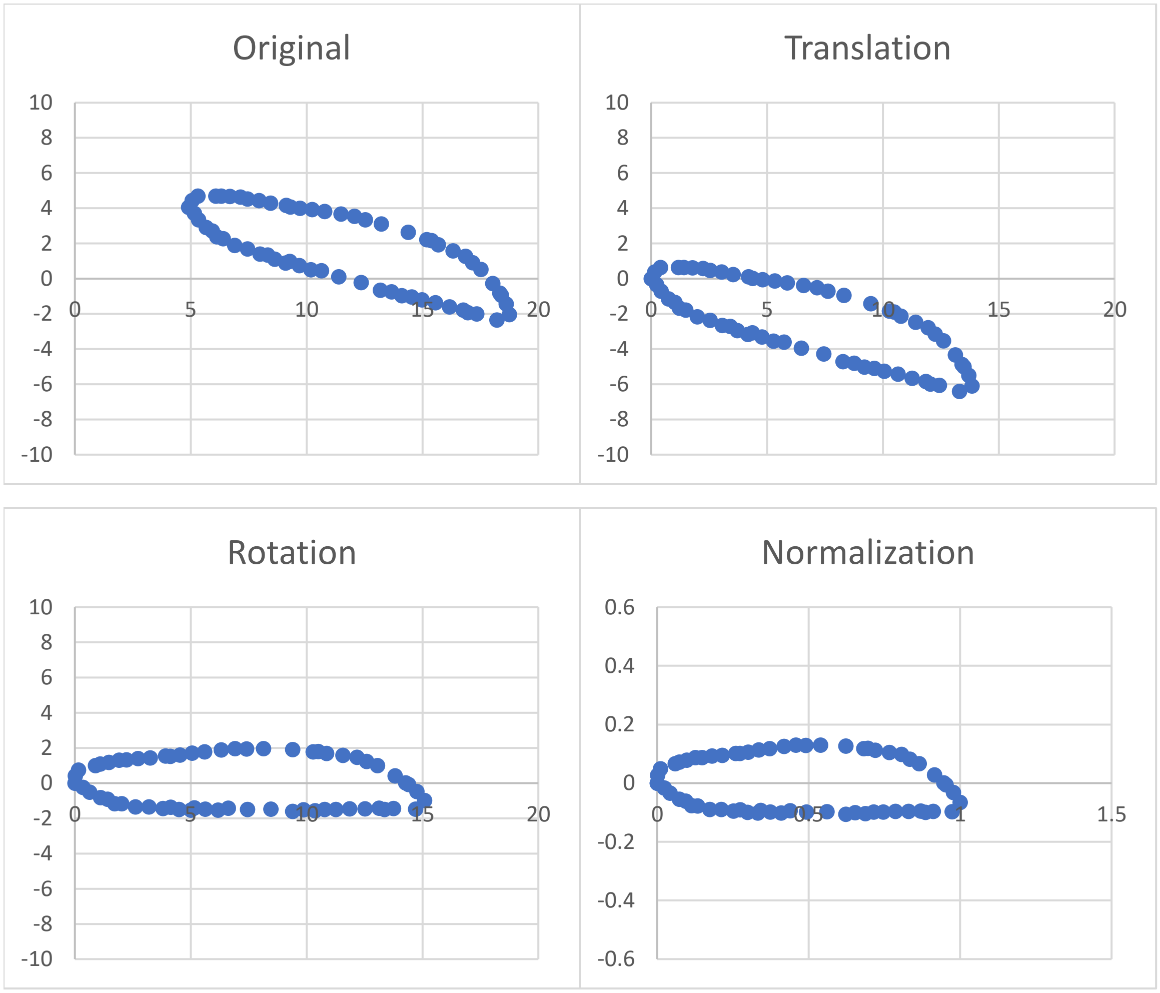


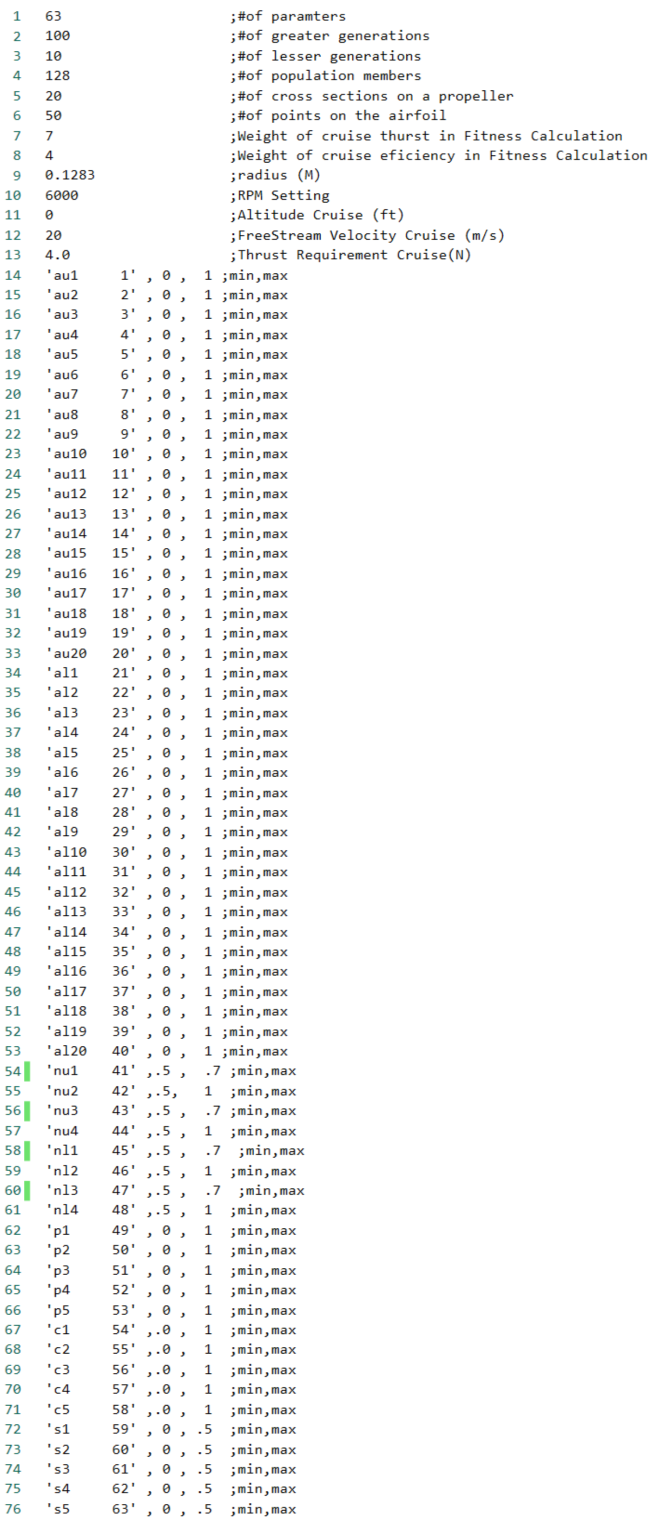
Figure 79: Airfoil Transformation

For example, if the leading the edge is at (-453, 205), then all points will have 453 added to the x value, and the y values will have 205 subtracted from them. The rotation is down by simply applying the following two equations for all the points for given axes.

$$X_{rot} = X_{og}\cos(\beta) - Y_{og}\sin(\beta)$$
$$Y_{rot} = Y_{og}\cos(\beta) - X_{og}\sin(\beta)$$

β is the angle by which the airfoil is rotated to become aligned with the x axis. The *og* terms represent the original x and y coordinates, and the *rot* terms represent the rotated coordinates of the airfoil. Lastly all the points of the airfoil are normalized by the chord length. The full process is provided in Figure 79. The order of operations moves from left to right then top to bottom. The final normalized airfoil is then used to create the OpenVSP airfoil file for the propeller. An example airfoil file is provided in the UIUC validation section in Figure 20.

Appendix II: Input File Overview



```
1   63                    ;#of paramters
2   100                   ;#of greater generations
3   10                    ;#of lesser generations
4   128                   ;#of population members
5   20                    ;#of cross sections on a propeller
6   50                    ;#of points on the airfoil
7   7                     ;Weight of cruise thurst in Fitness Calculation
8   4                     ;Weight of cruise eficiency in Fitness Calculation
9   0.1283                ;radius (M)
10  6000                  ;RPM Setting
11  0                     ;Altitude Cruise (ft)
12  20                    ;FreeStream Velocity Cruise (m/s)
13  4.0                   ;Thrust Requirement Cruise(N)
14  'au1     1' , 0 ,  1 ;min,max
15  'au2     2' , 0 ,  1 ;min,max
16  'au3     3' , 0 ,  1 ;min,max
17  'au4     4' , 0 ,  1 ;min,max
18  'au5     5' , 0 ,  1 ;min,max
19  'au6     6' , 0 ,  1 ;min,max
20  'au7     7' , 0 ,  1 ;min,max
21  'au8     8' , 0 ,  1 ;min,max
22  'au9     9' , 0 ,  1 ;min,max
23  'au10   10' , 0 ,  1 ;min,max
24  'au11   11' , 0 ,  1 ;min,max
25  'au12   12' , 0 ,  1 ;min,max
26  'au13   13' , 0 ,  1 ;min,max
27  'au14   14' , 0 ,  1 ;min,max
28  'au15   15' , 0 ,  1 ;min,max
29  'au16   16' , 0 ,  1 ;min,max
30  'au17   17' , 0 ,  1 ;min,max
31  'au18   18' , 0 ,  1 ;min,max
32  'au19   19' , 0 ,  1 ;min,max
33  'au20   20' , 0 ,  1 ;min,max
34  'al1    21' , 0 ,  1 ;min,max
35  'al2    22' , 0 ,  1 ;min,max
36  'al3    23' , 0 ,  1 ;min,max
37  'al4    24' , 0 ,  1 ;min,max
38  'al5    25' , 0 ,  1 ;min,max
39  'al6    26' , 0 ,  1 ;min,max
40  'al7    27' , 0 ,  1 ;min,max
41  'al8    28' , 0 ,  1 ;min,max
42  'al9    29' , 0 ,  1 ;min,max
43  'al10   30' , 0 ,  1 ;min,max
44  'al11   31' , 0 ,  1 ;min,max
45  'al12   32' , 0 ,  1 ;min,max
46  'al13   33' , 0 ,  1 ;min,max
47  'al14   34' , 0 ,  1 ;min,max
48  'al15   35' , 0 ,  1 ;min,max
49  'al16   36' , 0 ,  1 ;min,max
50  'al17   37' , 0 ,  1 ;min,max
51  'al18   38' , 0 ,  1 ;min,max
52  'al19   39' , 0 ,  1 ;min,max
53  'al20   40' , 0 ,  1 ;min,max
54  'nu1    41' ,.5 ,  .7 ;min,max
55  'nu2    42' ,.5,   1  ;min,max
56  'nu3    43' ,.5 ,  .7 ;min,max
57  'nu4    44' ,.5 ,  1  ;min,max
58  'nl1    45' ,.5 ,  .7  ;min,max
59  'nl2    46' ,.5 ,  1  ;min,max
60  'nl3    47' ,.5 ,  .7  ;min,max
61  'nl4    48' ,.5 ,  1  ;min,max
62  'p1     49' , 0 ,  1  ;min,max
63  'p2     50' , 0 ,  1  ;min,max
64  'p3     51' , 0 ,  1  ;min,max
65  'p4     52' , 0 ,  1  ;min,max
66  'p5     53' , 0 ,  1  ;min,max
67  'c1     54' ,.0 ,  1  ;min,max
68  'c2     55' ,.0 ,  1  ;min,max
69  'c3     56' ,.0 ,  1  ;min,max
70  'c4     57' ,.0 ,  1  ;min,max
71  'c5     58' ,.0 ,  1  ;min,max
72  's1     59' , 0 , .5  ;min,max
73  's2     60' , 0 , .5  ;min,max
74  's3     61' , 0 , .5  ;min,max
75  's4     62' , 0 , .5  ;min,max
76  's5     63' , 0 , .5  ;min,max
```

Figure 80: Text File Input

The text file input for the single point optimization is provided in Figure 80. The first 4 lines describe the behavior and set up of the GA itself. The number of greater generations is the number of generations in the main generational loop, and the number of lesser generations is the number of generations in each deme. The number of demes is constructed by calling the subroutine by the specified number of times. It has been done in other projects where the number of demes can be set as an input, but here it is set to a concrete value of 4. Lines 5 and 6 describe the governing limits for the construction of each of the geometries. Line 5 gives the number of cross sections in the radial direction while line 6 gives the number of points that describe the airfoil. Here it is set to 50 so there are 25 points to describe the upper surface and 25 to describe the lower.

Lines 7 and 8 provides weights for the fitness calculation. Line 9 gives the diameter for the propeller. Lines 10, 11, and 12 provide the solver set up and flight conditions for the propeller. Line 13 gives the thrust which the propeller is to be optimized for at the provided flight conditions. The rest of the inputs describe the minimum and maximum values for each of the coefficients that describe the behavior of the BP. Notice that all of the values are set from 0 to 1 except for the leading-edge terms of the airfoil and the sweep terms. The leading-edge terms are limited so that the leading edge cannot be extremely sharp. For subsonic flow regimes it is expected for the most optimal airfoil to have a rounded leading edge. Furthermore, the solver has difficulties on converging on propellers with these very sharp leading edges with thin surfaces. The sweep is limited to 0.5 due to initial results seen from the GA. None of the most optimal propellers had sweep coefficients greater than 0.5; therefore, it was limited in order to help the GA converge. It should be noted that even with this limitation, propellers are still capable of having sweep angles as high as 28.6 degrees.