IMPACT OF PARASITIC DRAG ON A FAMILY OF OPTIMAL LIFT DISTRIBUTIONS

by

Austin J. Stewart

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Aerospace Engineering

Approved:

Douglas F. Hunsaker, Ph.D. Major Professor	Thomas H. Fronk, Ph.D. Committee Member
Stephen A. Whitmore, Ph.D. Committee Member	Richard S. Inouye, Ph.D. Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2020

ABSTRACT

Impact of Parasitic Drag on a Family of Optimal Lift Distributions

by

Austin J. Stewart, Master of Science

Utah State University, 2020

Major Professor: Dr. Douglas F. Hunsaker
Department: Mechanical and Aerospace Engineering

Minimizing drag is a variational problem, and several minimum induced drag solutions have been found using different design constraints. The elliptic lift distribution is commonly used to minimize induced drag, but is only the optimal solution under one set of design constraints. Non-elliptic lift distributions are able to reduce induced drag, when compared to the elliptic lift distribution, by increasing the wingspan while maintaining a consistent wing–structure weight. However, these non-elliptic lift distributions are only optimal if the effects of viscous drag are neglected. In this study, numerical tools are used to estimate the total drag on rectangular wings that are twisted to give both elliptic and non-elliptic lift distributions. It is shown that the optimal lift distribution is described by B_n = 0 for all $n \neq 3$ and B_3 = -0.0901 or -0.103 depending on twist type. These optimal lift distributions reduce total drag by 1.01 or 1.23% respectively when compared to the elliptic lift distribution. These values are compared to lift distributions that minimize only induced drag, to understand the effects of using a non-elliptic lift distribution on the efficiency of an aircraft and the viability of using non-elliptic lift distributions on aircraft, specifically morphing-wing aircraft.

(78 pages)

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Douglas Hunsaker for this opportunity and the continued support and guidance though the process. I would also like to thank my committee members Dr. Stephen Whitmore and Dr. Thomas Fronk for the support and assistance they gave me.

I give special thanks to my family, friends, and colleagues, particularly Jeff Taylor, for their encouragement, moral support, and patience as I worked through many twists and turns this research presented. I am especially grateful for my wife for putting here schooling on hold so I could pursue this opportunity and the support she gave me as I was while here. I could not have done it without all of help and time you all gave me.

Austin J. Stewart

CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

A	= beam cross-sectional area
B_n	= Fourier coefficients in the lifting-line solution for the dimensionless section lift distribution, Eq. (1)
b	= wingspan
C_L	= wing lift coefficient
C_D	= wing drag coefficient
C_δ	= shape coefficient for the deflection-limited design Eq. (15)
C_σ	= shape coefficient for the stress-limited design Eq. (11)
$\tilde{C}$	= generic aerodynamic section coefficient
$\tilde{C}_L$	= airfoil section lift coefficient
c	= local wing chord length
D_i	= wing induced drag
E	= modulus of elasticity of the beam material
h	= height of the beam cross-section
I	= beam section moment of inertia
L	= total wing lift
$\tilde{L}$	= local wing section lift
$\widetilde{M}_b$	= local wing section bending moment
n_a	= load factor
n_g	= limiting load factor at the hard-landing design limit
n_m	= limiting load factor at the maneuvering-flight design limit
R_A	= aspect ratio
R_e	= Reynold’s number
S	= planform area

S_b	= proportionality constant between $\widetilde{W}_s(z)$ and $\widetilde{M}_b(z)$ having units of length squared
t_{max}	= maximum thickness of the local airfoil section
V_∞	= freestream airspeed
V_{Stall}	= stall speed of wing
W	= aircraft gross weight
W_n	= aircraft net weight (i.e., W-W_s)
W_r	= the portion of W_n carried at the wing root
W_s	= total weight of the wing structure required to support the wing bending moment
$\widetilde{W}_n$	= net weight of the wing per unit span (i.e., total wing weight per unit span less $\widetilde{W}_s$)
$\widetilde{W}_s$	= weight of the wing structure per unit span required to support the wing bending moment distribution
z	= spanwise coordinate relative to the midspan
α	= geometric twist of airfoil section
γ	= specific weight of the beam material
δ	= flap deflection of airfoil section
δ_{max}	= maximum wing deflection
θ	= change of variables for spanwise coordinate, Eq. (1)
κ_W	= weight distribution coefficient, Eq. (12)
ρ	= air density
σ_{max}	= maximum longitudinal stress

CHAPTER I

INTRODUCTION

For a wing with no sweep or dihedral immersed in uniform flow, Prandtl's lifting-line theory [1,2] relates the section lift distribution to the chord length and the aerodynamic angle of attack distributions. Additionally, for any wing with no sweep or dihedral immersed in uniform flow, Prandtl's lifting-line theory can be used to obtain a geometric and/or aerodynamic-twist distribution required to produce any desired section-lift distribution [3-8]. With Prandtl's lifting-line theory, an arbitrary spanwise-lift distribution is typically written in a Fourier sine series. Although the Fourier series can take many forms, the form we will use is [9]

$$\frac{b\tilde{L}}{L} = \frac{4}{\pi}\left[\sin(\theta) + \sum_{n=2}^{\infty} B_n \sin(n\theta)\right], \qquad \theta \equiv \cos^{-1}\left(-\frac{2z}{b}\right) \tag{1}$$

In addition, using classical lifting-line theory, there is also a solution to the induced drag caused by a wing under the same conditions. This solution can be written in terms of the B_n coefficients of Eq. (1). In steady level flight, when the total aircraft weight W is equal to the total aircraft lift L, the induced drag is written as [9]

$$D_i = \frac{2\left(\frac{W}{b}\right)^2}{\pi\rho V_\infty^2}\left(1 + \sum_{n=2}^{\infty} nB_n^2\right) \tag{2}$$

Equation 2 shows that, with a fixed W and b, the induced drag is minimized when all B_n = 0. The lift distribution produced by Eq. (1) with all B_n = 0 is known as the elliptic lift distribution, which was introduced by Prandtl [2]. However, Prandtl also made note that the drag that is produced by the elliptic lift distribution is not an absolute minimum and that fixing wingspan and weight might not be the best constraints to impose on the wing [10]. Some other lift distributions corresponding to different sets of non-zero B_n values allow wingspan to increase while maintaining the same wing-structure weight as that allowed by the elliptical lift distribution and therefore reduce drag. This increase in wingspan is based on the relationship between wingspan and wing-structure weight. If the lift distribution created by the B_n values produces lower wing section bending moments, then the wingspan can increase while keeping the wing-structure weight the same. To illustrate, when $B_n = 0$ for all $n \neq 3$ and B_3 is allowed to vary from -1/2 to 1/5, Fig. 1 shows the relationship between the resulting lift distribution and wingspan, for a given wing-structure weight. The wing-structure weight is a function of many variables which makes optimizing the wing-structure weight a variational problem. Prandtl and others have placed various constraints on these equations and produced different sets of lift distributions that minimize drag for different cases [11-16].

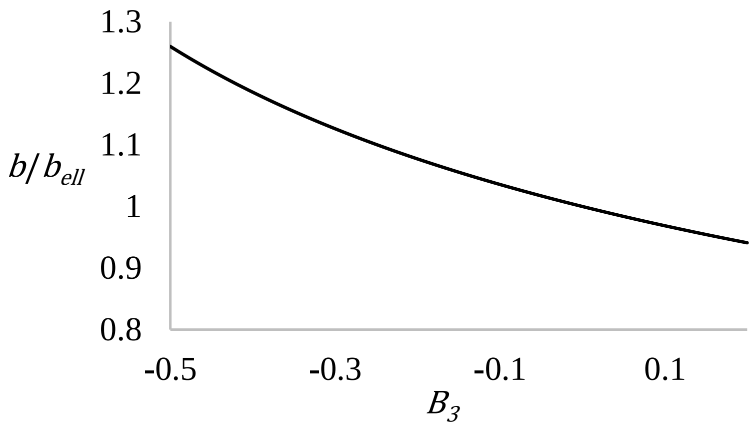


Fig. 1 Effect of lift distribution, specifically the B_3 coefficient, on allowable b with constant W_s.

In 1933, Prandtl solved the variational problem of minimizing induced drag with the constraints of fixed gross lift and fixed moment of inertia of gross lift on a rectangular wing [10]. This constrained problem leads to the dimensionless lift distribution

$$\frac{b\tilde{L}}{L} = \frac{4}{\pi}\left[\sin(\theta) - \frac{1}{3}\sin(3\theta)\right] \tag{3}$$

Comparing Eq. (3) to Eq. (1) we see that this lift distribution requires B_3 = -1/3 and B_n = 0 for all $n \neq 3$. Using these Fourier coefficients in Eq. (2) results in

$$D_i = \frac{8\left(\frac{W}{b}\right)^2}{3\pi\rho V_\infty^2} \tag{4}$$

Prandtl's 1933 lift distribution doesn't account for the moments produced by any weight in the wing, but does allow for a 22.5% increase in the span of the wing, and a 11.1% decrease in drag compared to the elliptic lift distribution [10]. Phillips, Hunsaker and Joo [9] relaxed some of the constraints used by Prandtl and included the effects of a weight distribution in the wing that fit the following form

$$W_n = W_r + \int_{z=-\frac{b}{2}}^{\frac{b}{2}} \widetilde{W}_n(z)dz \tag{5}$$

$$\widetilde{W}_n = (W - W_r)\frac{\tilde{L}(z)}{L} - \widetilde{W}_s(z) \tag{6}$$

Equations (5) and (6) do not completely specify the weight distribution but provide a relation between five design parameters. Using Eq. (5), W_n cannot be found until the other parameters have also been determined. Accounting for the lift and the weight carried in the wing, the bending moment takes the form [9]

$$\widetilde{M}_b(z) = \int_{z'=z}^{b/2} \left[\tilde{L}(z') - n_a\widetilde{W}_n(z') - n_a\widetilde{W}_s(z')\right](z'-z)dz' \quad \text{for} \quad z \geq 0 \tag{7}$$

The bending moment in the wing will determine the constraining limit at each section of the wing. The constraining load limit for stress- or deflection-limited designs is reached in maneuvering flight or during a hard landing. Using Eq. (6), the wing bending moment in Eq. (7) reduces significantly and can be integrated to give another constraint on the wing weight that will produce the optimal weight distribution. This additional constraint is the weight at the root of the wing, and is written as

$$W_r = \frac{n_g - 1}{n_m + n_g} W \tag{8}$$

This weight minimizes the bending moment produced at the constraining load limit. Using both Eqs. (6) and (8) yields a bending moment distribution for hard-landing that is exactly negative of the bending moment in maneuvering flight.

If W_r is larger than the value in Eq. (8), then maneuvering flight becomes the constraining condition; if W_r is smaller, then hard-landing becomes the constraining condition. Using Eq. (8), the bending moment in Eq. (7) reduces to [9]

$$\left|\widetilde{M}_b(z)\right| = \kappa_W W_r \int_{z'=z}^{\frac{b}{2}} \frac{\tilde{L}(z')}{L}(z' - z)dz', \quad \text{for } z \geq 0 \tag{9}$$

where

$$\kappa_W = \begin{cases} n_m, & W_r \geq \dfrac{n_g - 1}{n_m + n_g} W \\ (n_g - 1)\dfrac{W}{W_r} - n_g, & W_r < \dfrac{n_g - 1}{n_m + n_g} W \end{cases} \tag{10}$$

If the bending moment is supported by a vertically symmetric beam, for a wing with fixed maximum stress and spanwise-symmetric wing loading, the wing-structure weight can be expressed as [9]

$$W_s = 2\int_{z=0}^{b/2} \frac{\left|\widetilde{M}_b(z)\right|}{S_b(z)} dz\,; \quad S_b(z) = \frac{C_\sigma \left(\frac{t_{max}}{c}\right) c(z)\sigma_{max}}{\gamma}, \quad C_\sigma = \frac{2I\left(\frac{h}{t_{max}}\right)}{Ah^2} \tag{11}$$

Where C_σ is a structural property of the beam used by Phillips et. al. [9]. Values for some common beams are shown in reference [9]. If Eqs. (1), (9), and (11) are combined the wing-structure weight can be written as [9]

$$W_s = \frac{k_W W_r b^2}{32 S_b}(1 + B_3) \tag{12}$$

Equation (12) shows that although all B_n coefficients add to the induced drag, only B_3 influences the wing-structure weight for a rectangular wing with all positive lift and a spanwise-symmetric lift distribution.

Optimizing the wing-structure weight with respect to B_3 will allow an increase in wingspan and reduction of induced drag. Some examples of optimizing B_3 are given in references [9,13-16]. Phillips et. al. show several such optimized wing-structure weights with respect to B_3 [9]. With the constraints of fixed lift, fixed maximum stress, and fixed wing loading, the optimal W_s and B_3 are

$$W_s = \frac{\gamma\left(\frac{W}{S}\right)}{32 C_\sigma \left(\frac{t_{max}}{c}\right)\sigma_{max}} \frac{\kappa_W W_r b^3}{W}(1 + B_3) \tag{13}$$

$$B_3 = -\frac{3}{8} + \sqrt{\frac{9}{64} - \frac{1}{12}} \tag{14}$$

This results in a 4.98% increase in wingspan and a reduction of drag of 4.25% as compared to the elliptic lift distribution on a rectangular wing with the same wing-structure weight.

In the same paper, Phillips et. al. also introduced a similar derivation for a deflection-limited case with a fixed maximum deflection, fixed gross weight, fixed maximum gross weight, fixed lift distribution, and fixed wing loading [9]. This results in an optimal W_s and B_3 of

$$W_s = \frac{\gamma\left(\frac{W}{S}\right)^2}{32C_\delta E\left(\frac{t_{max}}{c}\right)^2\delta_{max}}\frac{\kappa_W W_r b^6}{W^2}(1+B_3); \quad C_\delta \equiv \frac{8I\left(\frac{h}{t_{max}}\right)^2}{Ah^2} \tag{15}$$

$$B_3 = -\frac{3}{7}+\sqrt{\frac{9}{49}-\frac{1}{21}} \tag{16}$$

which results in a 1.03% increase in wingspan and a 0.98% reduction in induced drag when compared to the elliptic lift distribution on a rectangular wing with all positive lift and a spanwise symmetric lift distribution and the same wing-structure weight.

In order to analyze multiple cases of B_3 and compare the resulting wing against a wing with an elliptic lift distribution and the same wing-structure weight, Eq. (12) for the non-elliptic lift distribution is set equal to Eq. (12) for the elliptic lift distribution ($B_3 = 0$). This new equation is rearranged to solve for the wingspan of the non-elliptic lift distribution

$$b_{non} = b_{ell} * \sqrt[3]{\frac{S_{non}}{S_{ell}(1+B_3)}} \tag{17}$$

Since we consider only cases of constant S, this term drops out of the equation and the resulting equation is only a function of b_{ell} and the B_3 coefficient of the non-elliptic lift distribution that is being analyzed.

All of the optimal wing-structure weights and Fourier coefficients discussed up to this point describe lift distributions that minimize induced drag with a given set of constraints. However, these lift distributions only minimize induced drag, not total drag. In these solutions, induced drag is only considered because it can be found analytically for a rectangular wing with all positive lift and a spanwise symmetric lift distribution. Total drag includes both induced and viscous drag terms and cannot be determined analytically. At lower speeds or high lift coefficients, induced drag is the dominant part of total drag. However, total drag is important to consider when trying to minimize drag over a flight envelope in order to achieve better efficiency, as there are points of flight where viscous drag is the dominant contributor for total drag.

One approach that has been taken to account for viscous effects is that of McGeer [17]. McGeer did account for some effects of parasitic drag analytically, but his work uses the parasitic drag as a constraint on the optimization of the wing. He constrains the parasitic drag to be equal to the parasitic drag that occurs on the elliptic wing during his optimization. He also focuses on using the sweep and chord distribution of the wing as well as airfoil thickness to chord ratio to achieve the different lift distributions. This study will use a numeric approach to find parasitic drag as well as constrain sweep, chord distribution, and airfoil thickness-to-chord ratio to be constant.

Morphing wing aircraft are beginning to be more viable as manufacturing technology improves. Modern morphing wing aircraft are capable of changing the lift distribution on the wing during flight, more precisely than a standard aircraft. Aircraft such as the U.S. Air Force Research Laboratory’s variable camber compliant wing (VCCW) [18-22] and the FlexSys Mission Adaptive Compliant Wing [23] are examples of aircraft that are able to change the lift distribution that the wing produces in flight. This morphing technology would allow the optimal lift distribution for each different part of a flight envelope to be implemented at every point of the flight and increase the efficiency of the aircraft. However, to understand which lift distribution is truly optimal the total drag must be analyzed, and not just induced drag.

CHAPTER II

PREDICTING DRAG ON AN ARBITRARY WING

Given values of B_n, a lift distribution can be described using Prandtl's lifting-line theory. In order to achieve this lift distribution, a rectangular wing must be twisted. In order to determine the aerodynamic properties of a twisted wing section, airfoil properties are needed for a variety of twisted airfoil shapes.

In this study the aerodynamic properties of airfoils are found using XFOIL, a 2D flow simulation tool [24]. XFOIL uses a two equation integral boundary layer method described by Drela and Giles [25] to determine viscous effects on an airfoil. In order to get the aerodynamic properties for the range of aerodynamic and geometric twist, a NACA 0015 airfoil shape is used as a base. This airfoil is analyzed at a variety of angles of attack to replicate geometric twist or washout. The same airfoil is also warped to simulate aerodynamic twist or camber. This is done by placing a parabolic flap on the airfoil, with the hinge point of the flap on the leading edge of the airfoil, and then deflecting that parabolic flap [26-27]. The airfoil with the parabolic flap is rotated until the chord line is horizontal to the flow and then resized to ensure that the flap deflection and rotation doesn't change the chord length of the airfoil. This process creates aerodynamic twist on the wing. Aerodynamic twist is commonly referred to as camber and percentage values are commonly used to describe the amount of twist produced by the camber. Our process is measured in degrees of flap deflection but will also be called camber. The base airfoil and resulting airfoils with 10° of washout or camber are shown in Fig. 2.

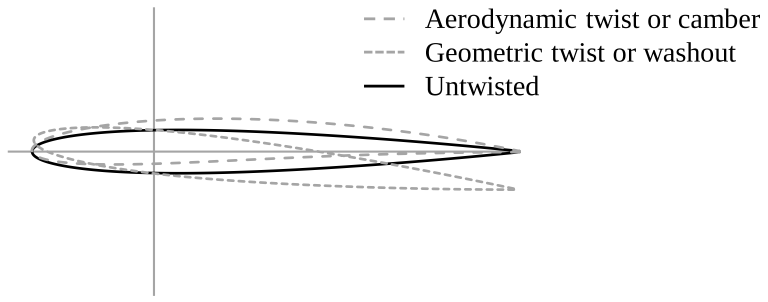


Fig. 2 Airfoil used for study and visual of twist types.

XFOIL gives lift, drag, and moment data for the airfoil shape at each specified washout and camber combination. The XFOIL data is taken at many points, but to have a continuous function for geometric and aerodynamic twist, a function must be fit to the results. This curve fit also helps relieve some of the problems that are common with XFOIL, like discontinuous or poorly behaved results. The drawbacks of the curve fit are that the curve fit equations will only be valid within the design space that was used in XFOIL. For this study, that design space is limited to ± 15° washout and ± 20° flap deflection to simulate camber. There is also some error associated with the curve fit. However, for each of the fits used in this study the error is small compared to the accuracy of XFOIL. The curve fits were obtained using a custom-built least squares best fit of the form [26]

$$F(\delta,\alpha) = f(\alpha)g(\delta) \tag{18}$$

where f and g are both polynomial functions of a single variable with polynomial orders N and M respectively. The polynomials in Eq. (18) can be used to give a more useful form of [26]

$$\tilde{C} = \sum_{m=0}^{M} \sum_{n=0}^{N} a_{nm} \delta^m \alpha^n \tag{19}$$

where a is the array of the polynomial coefficients, $\tilde{C}$ is one of the aerodynamic coefficients of interest (lift, drag, or moment) and δ and α are flap deflection and angle of attack respectively. A derivation of this least squares best fit routine is given in [26], Appendix B.

The tables in Appendices A-G show the polynomial fit coefficients for each aerodynamic coefficient at each Reynold's numbers of interest. The curve fits are done with the flap deflection and angle of attack in radians, so when using the tables and coefficients δ and α must both be radian values. The range of Reynold's numbers used to obtain XFOIL data is $500{,}000 \leq R_E \leq 1{,}100{,}000$. Each of the XFOIL results was compared to the resulting polynomial coefficient function using the coefficient of determination (R^2). These R^2 values were all ensured to be above 0.97 but where typically higher than 0.999. Figure 3 shows a series of polynomial fits using this method for a Reynold's number of 1,100,000 and a parabolic flap deflection of 3°.

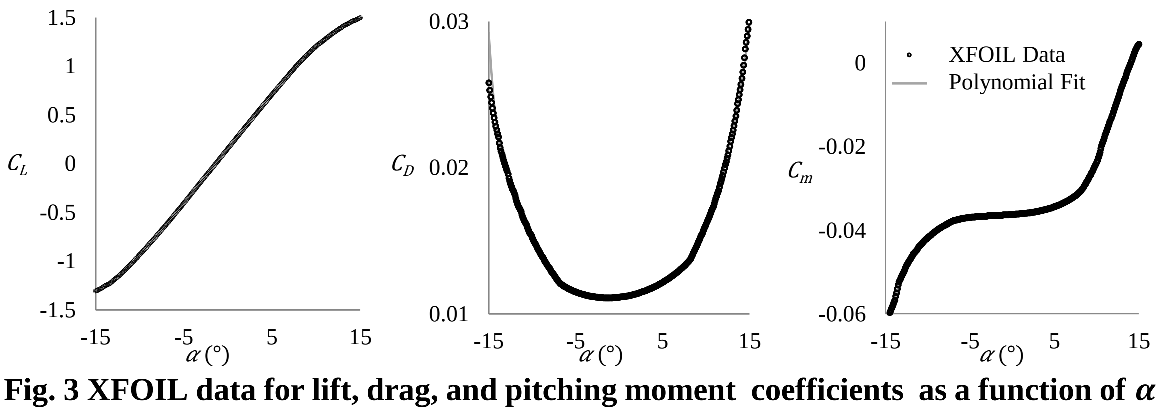


Fig. 3 XFOIL data for lift, drag, and pitching moment coefficients as a function of α fitted with polynomial equations.

The polynomial fit R^2 values for this case are all above 0.99 and match the XFOIL data well. These results are similar to results for other Reynold’s numbers and flap deflections.

To get the polynomial fit coefficients for a Reynold’s number that is not specified by the tables, linear interpolation was used between the given polynomial coefficients. The polynomial coefficients in each of the tables are well-behaved between each Reynold’s number, allowing analysis of airfoils at any Reynold’s number, δ, and α in the design space.

Using the airfoil properties for each wing section, the lift distribution given by the entire wing is determined using MachUp. MachUp is an in-house design tool that uses a numeric lifting-line algorithm developed by Phillips and Snyder [28] to solve for aerodynamic properties of an aircraft [29]. MachUp is an open source code available through github[3]. MachUp is given a wing with b_{non} and c obtained using Eq. (17) and a starting twist guess. This guess consists of a wing angle of attack and a set of twist values (either washout angles or camber values). The twist is specified at points clustered along the semispan according to the change of variables in Eq. (1). MachUp linearly interpolates between these control point twists, assigning a section washout and camber for each point along the wing. MachUp then outputs the section aerodynamic values along the entire wing using the airfoil values given by the polynomial coefficients defined in Eq. (19). The section lift coefficients generated by MachUp are compared to the analytic section lift coefficients. The analytic section lift is based on Eq. (1), but this equation is nondimensionalized in an unconventional way, so it is converted to a typical lift coefficient using

[3] https://github.com/usuaero/MachUp

$$\tilde{C}_L = \frac{b\tilde{L}}{L}\frac{W}{b}\frac{2}{\rho V_\infty^2 c} = \frac{\tilde{L}}{\frac{1}{2}\rho V_\infty^2 c} \tag{20}$$

The section lift given by MachUp is compared directly to Eq. (20).

In order to match the lift distribution given by Eq. (20), the RMS is calculated between the MachUp lift distribution and Eq. (20). The RMS value is then minimized using an in-house gradient based optimization tool called Optix. Optix utilizes the BFGS method to minimize the objective function [30-33]. Optix loops through the MachUp calculations varying the angle of attack and washout and/or camber values along the span of the wing, while keeping the root twist of the wing constant until the RMS is minimized. The final twist and angle of attack values are then run through MachUp once more to find the lift, drag, and moment generated by the wing that now has a lift distribution that matches the analytic lift distribution created with the given B_n values. Figure 4 shows the analytic lift distribution using Eq. (20) and the lift distribution that is achieved using the prescribed method with five control points along the semi-span, with one of those points being the root twist. Results are shown for the lift distributions obtained by varying only the camber and by varying only the washout.

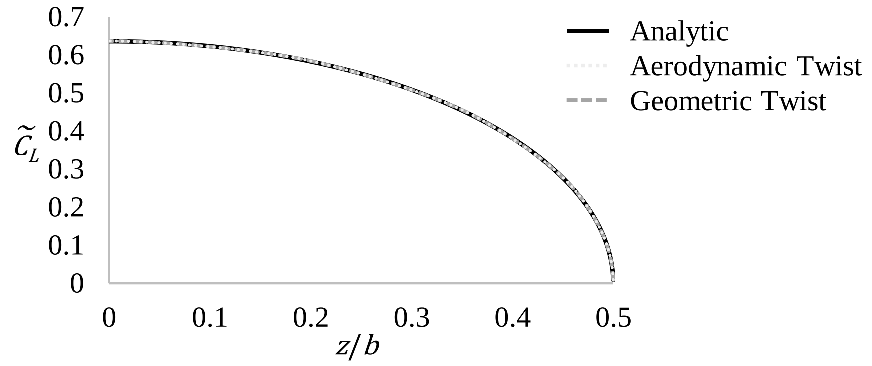


Fig. 4 Comparison of lift distributions generated by MachUp and Eq. (20) for B_3 =0.0.

In order to better understand the process and flow of data, a flow chart is shown in Figure 5. This shows the beginning steps where the user inputs design values for the baseline elliptic lift distribution wing. The information is used to create the geometry of a wing with a given non-elliptic lift distribution and given wing-structure weight. This wing is given to Optix and Optix passes this wing to MachUp which calculates the lift distribution. The lift distribution is compared to the analytic lift distribution and a RMS value is returned to Optix. Optix chooses a new twist profile based on the results from MachUp and iterates through this process until the RMS value is minimized. The minimized twist profile that is generated by Optix is then passed to a final version of MachUp that outputs the total drag value produced by the wing with the matching lift distribution.

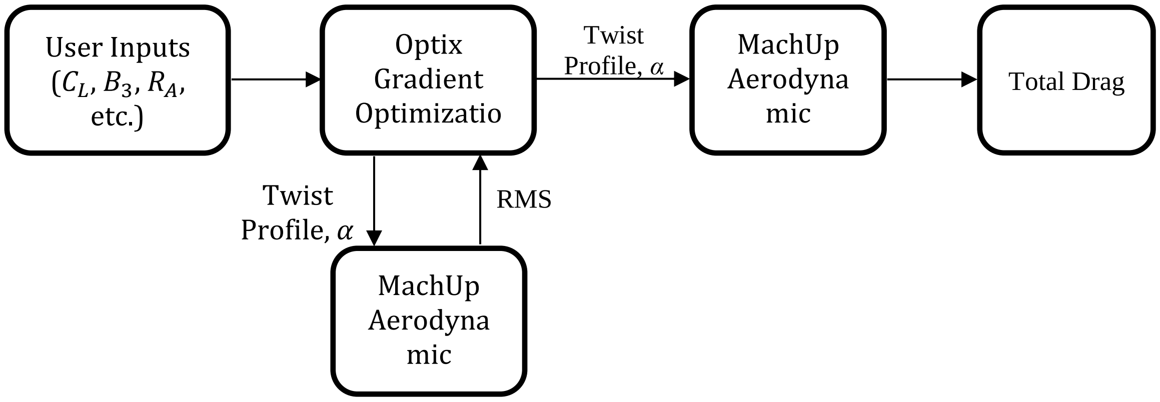


Fig. 5 Flow chart showing path of data and method.

CHAPTER III

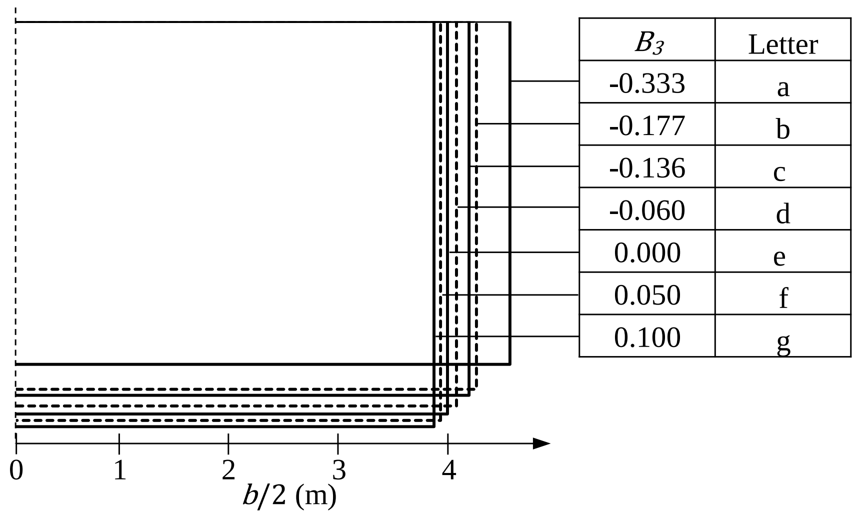
RESULTS

The optimized lift distributions given in Eqs. (14) and (16), as well as the elliptic lift distribution and a few additional lift distributions defined by B_3 are presented in Table 1 with the associated design constraints.

Table 1 B_3 values used in study with associated design constraints.

B_3	Design Constraints
-0.333	Fixed W_n, σ_{max}, and V_{stall}
-0.177	Fixed W_n, δ_{max}, and V_{stall}
-0.136	Fixed W_n, σ_{max}, and W/S
-0.060	Fixed W_n, δ_{max}, and W/S
0.000	Fixed b and W
0.050	Only for Study
0.100	Only for Study

These B_3 values are the values that were tested to show the trends in the drag. For this study the wing with the elliptic lift distribution that will be used for comparison and wing-structure weight has an R_A = 8 and b = 8 and is flying at standard sea level with R_E = 1,000,000 and C_L = 0.5. Figure 6 shows the wing planforms for each of the chosen B_3 values. The change in aspect ratio will change the R_E for each B_3. The minimal changes in the span between each B_3 value is what allow for changes in induced drag according to Eq. (2).



B_3	Letter
-0.333	a
-0.177	b
-0.136	c
-0.060	d
0.000	e
0.050	f
0.100	g

Fig. 6 Wing planforms of various values of B_3.

The lettering corresponding to each value of B_3 shown in Fig. 5 will be consistent throughout the remainder of the document. These planforms were all generated in MachUp and then twisted to achieve the desired lift distribution. The lift distributions created using the planforms shown in Fig. 6 and the B_3 values given in Table 1 are shown in Fig. 7. The higher the value of B_3, the more lift is carried near the wing tips. This creates a larger bending moment along the span and, in turn, shortens the wing, as shown in Fig. 6. The longer wingspan creates more parasitic drag, due to decrease in Reynold’s number, but less induced drag. At some point there is a minimum location of total drag. This minimum B_3 value is very useful to know for improved efficiency of aircraft.

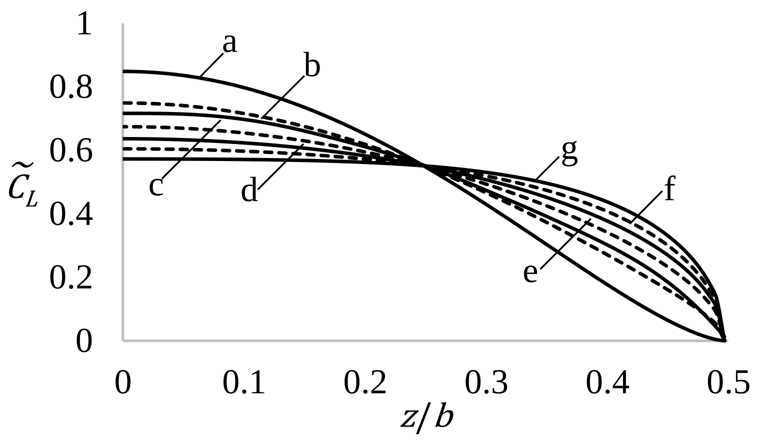


Fig. 7 Lift distributions of various values of B_3.

As the wing planform changes due to the ability of the lift distribution to carry the load more or less toward the root, the Reynold's number also changes due to the change in chord. Figure. 8 shows the Reynold's number as a function of B_3. This change in Reynold's number is part of the reason that the parasitic drag varies between different cases.

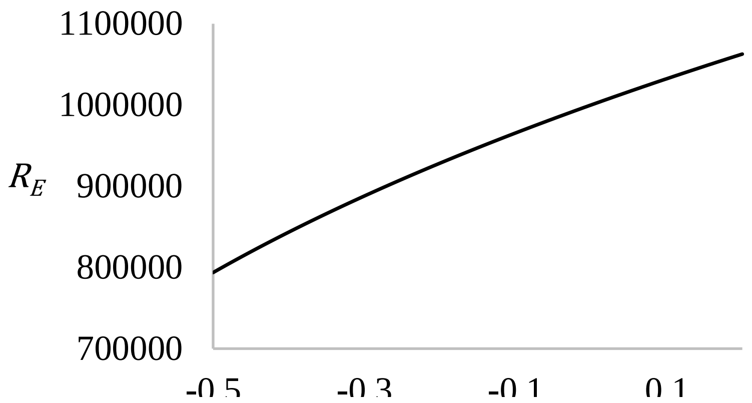


Fig. 8 Reynold's number changes with respect to the B_3 Fourier coefficient.

Geometric Twist or Washout

When only using geometric twist or washout to replicate a lift distribution on a rectangular wing with no sweep or dihedral, the airfoil shape is fixed. Here we use a

NACA 0015. In order to achieve a desired lift distribution, the twist profile will always be the same regardless of what root twist is used. This is because the overall angle of attack of the wing is one of the parameters that Optix is allowed to vary. To match the desired lift distribution, the root of the wing must produce a certain amount of lift. To get this lift, the root airfoil will have to be positioned at a certain angle of attack relative to the incoming flow. The root airfoil achieves this angle of attack through a combination of the root geometric twist and the wing angle of attack. The same is true for each location along the span of the wing. This means that the drag values for a given value of B_3, with camber held constant, will be independent of the root geometric twist. Figure 9 shows the total drag values as a function of root twist for several B_3 values.

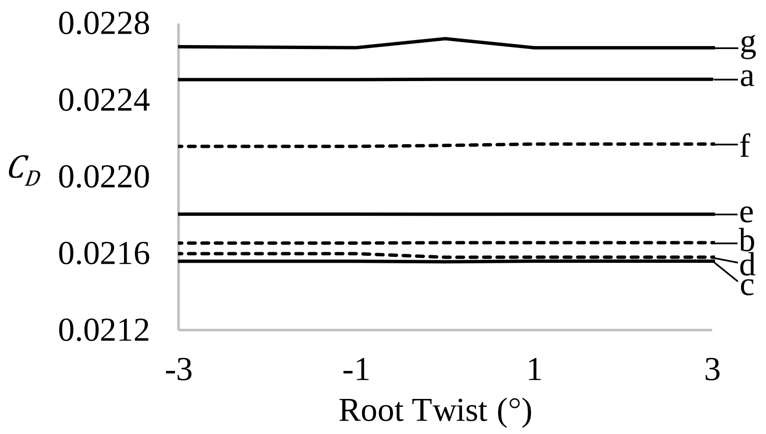


Fig. 9 Drag values for a variety of root twist and B_3 values when using washout.

The changing root twist values have no impact on the drag the wing experiences as shown by the nearly horizontal C_D values. The minimal deviations from horizontal are within the bound of precision that the process can reproduce. This validates that the code is working as expected and reveals important aspects of the design space. Figure 9 shows that there is a minimum C_D value somewhere between B_3 = -0.06 and B_3 = -0.136. To find this minimum C_D value a fourth order polynomial was fit to the drag as a function of

B_3. Figure 10 shows the resulting polynomial fit in grey with the data overlaid on it as grey circles.

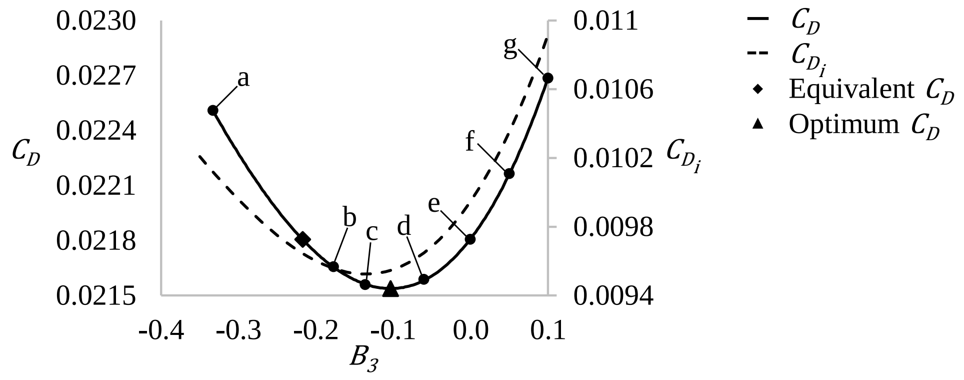


Fig. 10 Minimum drag value as a function of B_3 using washout.

This polynomial fit predicts a minimum C_D value at B_3 = -0.103. This minimum B_3 values is shown in Fig. 10 as a triangle. Using this B_3 value in the process described in Chapter II results in a 1.232% reduction of drag as compared to the elliptic lift distribution on a rectangular wing with no sweep or dihedral and the same wing-structure weight. Figure 10 also shows that not all of the lift distributions that were described in Table 1 produce less drag than the elliptical lift distribution. There is a region where the reduction in induced drag is greater than the increase in parasitic drag due to increased span. For all 0.0 > $B_3 \geq$ -0.2170 there is a reduction of C_D as compared to the elliptic lift distribution. However, outside this range the reduction in induced drag is outweighed by the increase in parasitic drag. This equivalent B_3 value is shown in Fig. 10 as a diamond. The optimal lift distribution when only induced drag is considered is close to the optimal valued shown in Fig. 10. This means that doing the analytic optimizations neglecting parasitic drag does result in near optimal results. To further investigate the region of reduced total drag, camber is considered.

Aerodynamic Twist or Camber

When using aerodynamic twist or camber and angle of attack of the wing to produce the desired lift distribution, the total drag produced by the wing varies depending on root camber. This means that if washout is held constant along the wing, there will be at least one minimum total drag value for each value of B_3. For each value of B_3, the simulation was run with a unique range of root camber. The values of root camber for each B_3 value were chosen to be around the minimum total drag for that B_3 value. They range from camber values related to -4° to 16° of flap deflection. Figure 11 shows total drag for the different B_3 values as a function of root camber. There is a minimum for each of the different B_3 values, and this minimum occurs at lower root camber as B_3 goes up.

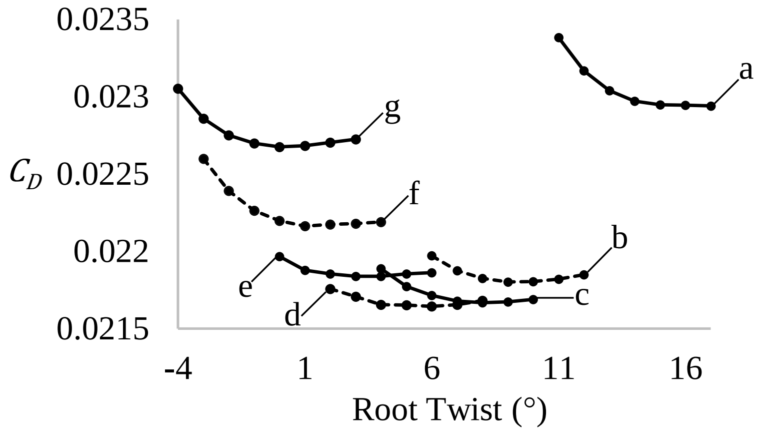


Fig. 11 Drag values for a variety of root camber and B_3 values when using camber.

Just like in the geometric twist case, there is a minimum C_D between B_3 = -0.06 and B_3 = -0.136. A fourth order polynomial was fit to each of the C_D curves and the minimum expected drag value was found for each B_3 value. The minimum drag value for each B_3 was used to create an additional fourth order polynomial fit of C_D as a function of B_3. The resulting polynomial fit and the minimum drag points are shown in Fig. 12 as the grey line and circles, respectively.

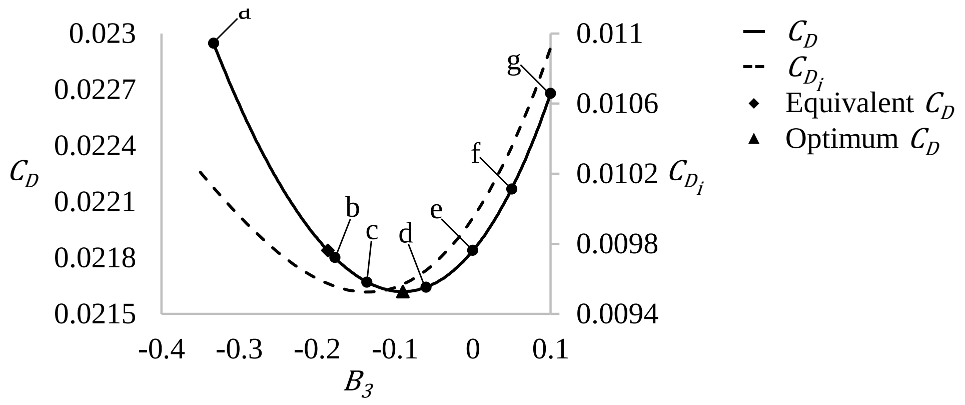


Fig. 12 Minimum drag given a B_3 value using camber.

The minimum of this polynomial fit occurs at B_3 = -0.0901. Using this value in the process described in Chapter II results in a 1.013% reduction in drag, when compared to the elliptic lift distribution on a rectangular wing with no sweep or dihedral with the same wing-structure weight. This minimum C_D point is marked in Fig. 12 with a triangle. The range of B_3 values that produce less drag than the elliptic lift distribution is $0.0 > B_3 \geq$ -0.1865. The lower bound is indicated in Fig. 12 with a diamond. Once again the total optimum is near the induced drag only optimum, which validates using analytic approaches to solve for optimal solutions.

Figure 13 compares the minimum drag values from Fig. 10 and 12. The grey line comes from Fig. 12 and represents the minimum drag values when using camber and the black line comes from Fig. 10 represents the minimum drag values when using washout.

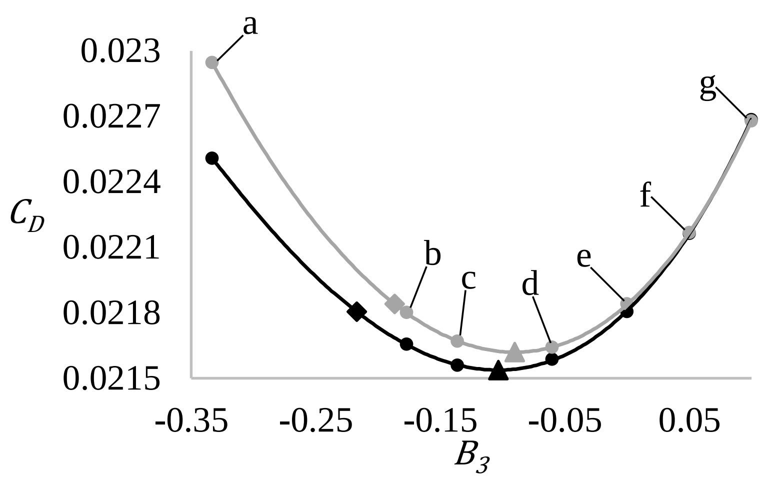


Fig. 13 Comparison of minimum drag values for different twist types.

Figure 13 shows that the range of B_3 values that results in a reduction of drag is larger when using washout than when using camber to achieve the lift distributions. Additionally, the lift distribution described by the optimal value of B_3 for washout case gives a greater reduction of drag than the optimal lift distribution obtained using camber. In fact, the C_D values obtained using washout were smaller for all values of B_3 than those obtained using camber. This indicates that using washout to match lift distributions will provide a greater range of drag reducing options when compared to the elliptic lift distribution and will have less drag regardless of which lift distribution is used.

Comparison of Drag Components

The optimized lift distributions given in Table 1 provide solutions for the minimum induced drag along a wing. This is mainly due to the ability to increase the span as shown in Fig. 1 and Fig. 6. However, these optimized lift distributions were found without taking parasitic drag into account. Figure 14 shows the comparison of the section drag broken down into the drag components for the elliptic lift distribution with B_3 = 0.0, Prandtl's 1933 lift distribution described in Eq. (14) with B_3 = -1/3, the optimal lift distribution when only considering induced drag described by Eq. (14) with B_3 = -0.136, and the optimal lift distribution found when using camber with B_3 = -0.0901.

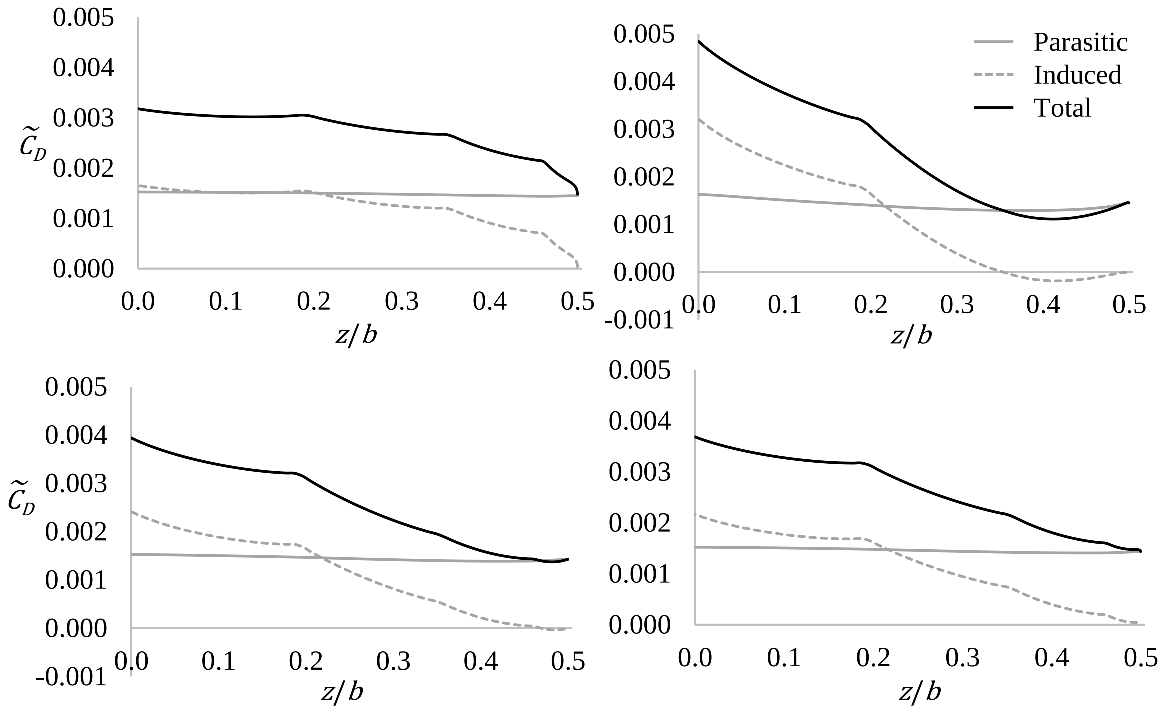


Fig. 14 Section drag along semispan, comparing parasitic and induced drag for elliptic lift distribution (upper left), Prandtl's 1933 lift distribution (upper right), the optimal lift distribution when only considering induced drag (lower left), and the optimal lift distribution when regarding total drag (lower right).

The section parasitic drag for all cases is nearly constant along the entire span, which is expected on a rectangular wing. The section parasitic drag for each section is similar between all cases, but the additional span that results from using the non-elliptic lift distributions means that the summation of the section drag results in an overall increase in parasitic drag. The parasitic drag makes up more than half of the total drag for all cases, with 54.47% of total drag for the elliptic lift distribution and 56.03% of the total drag for Prandtl's 1933 lift distribution. Therefore, the minor reductions in induced drag that Prandtl's lift distribution achieves are outweighed by the increases in parasitic drag caused by increasing the span and twisting the wing to create this lift distribution.

However, for both the optimal induced drag case and the optimal total drag case the increase in parasitic drag is made up for in the reduction of induced drag. When only using camber to obtain the optimal lift distribution given in Fig. 12, the tradeoff between parasitic and induced drag results in a 1.365% increase in parasitic drag but a 4.010% decrease in induced drag when compared to the elliptic lift distribution. This is why this lift distribution produces 1.013% less total drag than the elliptic lift distribution on a wing with the same wing-structure weight. To compare the optimal lift distribution when only considering induced drag to the optimal lift distribution for total drag, Fig. 15 shows all four lift distributions from Fig. 14 as well as the optimal lift distribution when using washout shown in Fig. 10.

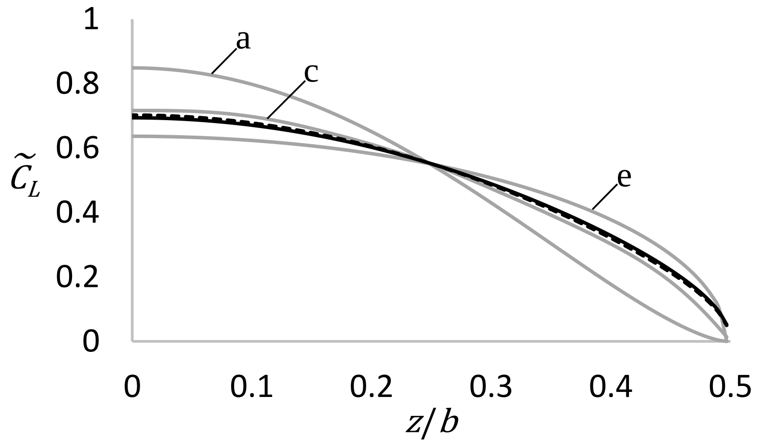


Fig. 15 Lift distributions of B_3 values corresponding to optimal values, the elliptic, and Prandtl's 1933.

The dashed black line in Fig. 15 represents the lift distribution described by B_3 = -0.103 and the solid black line represents the lift distribution described by B_3 = -0.0901. The two optimal values regarding total drag are almost identical along the entire span. The optimal with respect to only induced drag is close to the optimal regarding total drag but carries slightly more weight toward the center of the wing. Prandtl’s 1933 lift distribution carries significantly more weight toward the center and the elliptic cares weight more evenly across the span.

CHAPTER IV

CONCLUSIONS

When only considering induced drag on a rectangular wing in uniform flow with no sweep or dihedral, the elliptic lift distribution is not necessarily ideal. Using Prandtl's lifting-line theory, a lift distribution can be described by Eq. (1) and the induced drag from the same lift distribution can be described by Eq. (2). Minimizing the induced drag in Eq. (2) is a variational problem that can be solved several ways. The elliptic lift distribution is one solution to the problem. If different design constraints are used to solve the problem, non-elliptic lift distributions are the solution. These other non-elliptic optimal lift distributions reduce induced drag, when compared to the elliptic lift distribution, by moving the bending moment inboard on the wing. Moving the bending moment allows for larger wingspans, while maintaining the same wing-structure weight. Equation (12) shows that the only Fourier coefficient that influences the wing-structure weight of rectangular wings with the non-structural weight distribution given by Eqs. (5), (6) and (8) is B_3. Several optimized values of B_3 are shown in Eqs. (14) and (16) as well as in Table 1. These optimal lift distributions were found without taking parasitic drag into account. As such, they do not minimize total drag experienced by a wing. In this paper, a numerical approach using a numeric lifting-line tool called MachUp and a gradient based optimizer called Optix is used to generate lift distributions that matching analytic lift distribution created using Eq. (20) with the values of B_3 shown in Table 1.

The total drag of a rectangular wing having the lift distributions described in Table 1, is found by twisting a wing using either washout or camber and using a numeric lifting-line tool called MachUp on the resulting wing. The total drag values obtained using this method are presented in Chapter III. The results indicate that when considering total drag on a rectangular wing with no sweep or dihedral immersed in uniform flow, the minimum drag is not obtained using the elliptic lift distribution or any of the optimized lift distributions when considering only induced drag. Instead the optimal value is around B_3 = -0.1 and depends on the way the wing is being twisted to produce the lift distribution described by this B_3 value. This lift distribution balances the parasitic and induced drag components along a rectangular wing optimally to minimize drag and maximize efficiency. This lift distribution is close to the optimal lift distribution when only considering induced drag, but does distribute slightly more weight along the span. Some modern morphing wing aircraft have rectangular wings. Therefore, the optimal lift distribution shown in this paper can be used to reduce the total drag experienced by the wing by 1.01%-1.23% depending on the twist type. Using only geometric twist will result in lower total drag, regardless of which lift distribution is used and result in the greatest benefit if the optimal lift distribution of B_3 = -0.103 is used. Some of the lift distributions described in Table 1 also result in less drag and could also be used to reduce drag, while meeting additional design requirements like load or deflection alleviation.

REFERNECES

[1] Prandtl, L., “Tragflügel Theorie,” Nachricten von der Gesellschaft der Wissenschaften zu Güttingen, Ges-chäeftliche Mitteilungen, Klasse, 1918, pp. 451-477

[2] Prandtl, L., “Applications of Modern Hydrodynamics to Aeronautics,” NACA TR-116, June 1921

[3] Phillips, W. F., “Lifting-Line Analysis for Twisted Wings and Washout-Optimized Wings,” Journal of Aircraft, Vol. 41, No. 1, 2004, pp. 128-136. (doi:10.2514/1.262)

[4] Phillips, W.F., Alley, N. R., and Goodrich, W. D., “Lifting-Line Analysis of Roll Control and Variable Twist,” Journal of Aircraft, Vol. 41, No. 5, 2004, pp. 1169-1176. (doi:10.2514/1.3846)

[5] Phillips, W. F., “New Twist on an Old Wing Theory,” Aerospace America, January, 2005, pp. 27-30.

[6] Phillips, W. F., Fugal, S. R., and Spall, R. E., “Minimizing Induced Drag with Wing Twist, Computational-Fluid-Dynamics Validation,” Journal of Aircraft, Vol. 43, No. 2, 2006, pp.437-444. (doi:10.2514/1.15089)

[7] Phillips, W. F., and Alley, N. R., “Predicting Maximum Lift Coefficients for Twisted Wings Using Lifting Line Theory,” Journal of Aircraft, Vol. 44, No. 3, 2007, pp. 898-910. (doi:10.2514/1.25640)

[8] Phillips, W. F., “Incompressible Flow over Finite Wings,” Mechanics of Flight, 2nd ed., Wiley, Hoboken, NJ, 2010, pp. 46-94

[9] Phillips, W. F., Hunsaker, D. F., and Joo, J. J., “Minimizing Induced Drag with Lift Distribution and Wingspan,” Journal of Aircraft, Vol. 56, No. 2, 2019. (doi:10.2514/1.C035027)

[10] Prandlt, L., “Über Tragflügel kleinsten induzierten Wilderstandes,” Zeitschrift für Flugtechnik und Motorluftschiffahrt, Vol. 24, No. 11, 1933, pp. 305-306.

[11] Jones, R. T., “The Spanwise Distribution of Lift for Minimum Induced Drag of Wings Having a Given Lift and Given Bending Moment,” NACA TR-2249, Dec. 1950.

[12] Klein, A., and Viswanathan, S. P., “Minimum Induced Drag of Wings with Given Lift and Root-Bending Moment,” Zeitschrift für Angewandte Mathematik und Physik, Vol. 24, No. 6, 1973, pp. 8889-892. (doi:10.1007/bf01590797)

[13] Klein, A., and Viswanathan, S. P., “Approximate Solution for Minimum Induced Drag of Wings with Given Structural Weight,” Journal of Aircraft, Vol. 12, Np. 2, 1975, pp. 124-126. (doi:10.2514/3.44425)

[14] Klein, A., and Viswanathan, S. P., “Errata: Approximate Solution for Minimum Induced Drag of Wings with Given Structural Weight,” Journal of Aircraft, Vol. 12, No. 9, 1975, p. 756. (doi:10.2514/3.59866)

[15] Bowers, A. H., Murillo, O. J., Jensen, R., Eslinger, B., and Gelzer, C., “On Wings of the Minimum Induced Drag: Spanload Implications for Aircraft and Birds,” NASA TP-2016-219072, March 2016.

[16] Phillips, W. F., Hunsaker, D. F., and Taylor, J. D., “Minimizing Induced Drag with Weight Distribution, Lift Distribution, Wingspan, and Wing-Structure Weight,” to be presented at AIAA Aviation, 2019.”

[17] McGeer, T., “Wing Design for Minimum Drag with Practical Constraints,” Journal of Aircraft, Vol. 21, No. 11, Nov. 1984. (doi:10.2514/3.45058)

[18] Joo, J. J., Marks, C. R., Zientarski, L., and Culler, A. J., “Variable Camber Compliant Wing-Design,” 23rd AIAA/AHS Adaptive Structures Conference, AIAA Paper 2015-1050, Jan. 2015.

[19] Marks, C. R., Zientarski, L., Culler, A. J., Hagen, B., Smyers, B. M., and Joo, J. J., “Variable Camber Compliant Wing-Wind Tunnel Testing,” 23rd AIAA/AHS Adaptive Structures Conference, AIAA Paper 2015-1051, Jan. 2015.

[20] Miller, S. C., Rumpfkeil, M. P., and Joo, J. J., “Fluid-Structure Interaction of a Variable Camber Compliant Wing,” 53rd AIAA Aerospace Sciences Meeting, AIAA Paper 2015-1235, Jan. 2015.

[21] Joo, J. J., Marks, C. R., and Zienstarski, L., “Active Wing Shape Reconfiguration Using a Variable Camber Compliant Wing System,” 20th International Conference on Composite Materials, Copenhagen, Program Number is 4121-4, July 2015.

[22] Marks, C. R., Zientarski, L., and Joo, J. J., “Investigation into the Effect of Shape Deviation on Variable Camber Compliant Wing Performance,” 24th AIAA/AHS Adaptive Structures Conference, AIAA Paper 2016-1313, Jan. 2016

[23] Hetrick, J., Osborn, R., Kota, S., Flick, P., and Paul, D., “Flight Testing of Mission Adaptive Compliant Wing,” 48th AIAA/ASME/ASCE/AHS/ASC structures, Structural Dynamics, and Materials Conference, AIAA Paper 2007-1709, April 2007.

[24] Drela, M., “XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils,” Conference on Low Reynolds Number Aerodynamics, University of Notre Dame, June 1989.

[25] Drela, M. Giles, M.B., "Viscous-Inviscid Analysis of Transonic and Low Reynolds Number Airfoils," AIAA Journal, Vol. 25, No. 10, Oct. 1987. (doi:10.2514/3.9789)

[26] Hunsaker, D.F., Reid, J.T., Moorthamers, B., Joo, J.J., "Geometry and Aerodynamic Performance of Parabolic Trailing-Edge Flaps," AIAA Aerospace Sciences Meeting, AIAA Paper 2018-1278, Jan. 2018. (doi:10.2514/2018-1278)

[27] Ullah, A. H., Fabijanic, C., Estevadeordal, J., Montgomery, Z. S., Hunsaker, D. F., Staiger, J. M., Joo, J. J., "Experimental and Numerical Evaluation of the Performance of Parabolic Flaps", Submitted to AIAA SciTech, 2020.

[28] Phillips, W.F., and Snyder D.O., "Modern Adaptation of Prandtl's Classic Lifting-Line Theory," Journal of Aircraft, Vol. 37, No.4, July 2000, pp. 662-670. (doi:10.2514/2.2649)

[29] Hodson, J., Hunsaker, D.F., Spall, R., "Wing Optimization using Dual Number Automatic Differentiation in MachUp," 55th AIAA Aerospace Science Meeting, AIAA Paper 2017-0033, Jan. 2017. (doi:10.2514/6.2017-0033)

[30] Broyden, C., "The convergence of a Class of Double-Rank Minimization Algorithms," Journal of the Institute of Mathematics and its Applications, Vol. 6, 1970 pp. 76-90. (doi: 10.1093/imamat/6.1.76)

[31] Fletcher, R., "A New Approach to Variable Metric Algorithms," Computer Journal, Vol. 13, No. 3, 1970, pp. 317-322. (doi:10.1093/comjnl/13.3.317)

[32] Goldfarb, D., "A Family of Variable Metric Updates Derived by Variational Means," Mathematics of Computation, Vol. 24, No. 109, 1970, pp. 23-26. (doi: 10.1090/S0025-5718-1970-0258249-6)

[33] Shanno, D., "Conditioning of Quasi-Newton Methods for Function Minimization," Mathematics of Computation, Vol. 24, No. 111, 1970, pp. 647-656. (doi: 10.1090/S0025-5718-1970-0274029-X)

APPENDIX A

POLYNOMIAL FIT COEFFICIENT TABLES FOR NACA 0015 PARABOLIC FLAP AIRFOIL AT R_e = 500,000

Table A3 polynomial fit coefficient for $\tilde{C}_m$

$\tilde{C}_m$						n			
		0	1	2	3	4	5	6	7
	0		3.196431E-02		-1.355588E+00		1.435450E+02		-1.464256E+03
	1	-1.171512E-02		6.317115E-03		2.096448E+00		-2.622941E+01	
	2		2.360793E-04		8.292556E-02		-3.175935E+00		2.821833E+01
	3	7.685359E-06		1.291372E-03		-6.425903E-02		7.152403E-01	
m	4		7.196154E-06		-1.028566E-03		3.120157E-02		-2.601204E-01
	5	2.044302E-08		-1.017882E-05		4.484386E-04		-4.931360E-03	
	6		-3.212812E-08		3.557204E-06		-9.692598E-05		7.304992E-04
	7	-9.085691E-11		3.005182E-08		-1.318831E-06		1.500498E-05	
	8		3.654763E-11		-3.621692E-09		8.992984E-08		-6.070004E-07
	9	1.012184E-13		-3.096893E-11		1.381046E-09		-1.610412E-08	

Table A2 polynomial fit coefficient for $\tilde{C}_D$

$\tilde{C}_D$							n					
		0	1	2	3	4	5	6	7	8	9	10
	0	1.304920E-02		1.447117E-01		-6.022799E+00		4.520021E+02		-8.105087E+03		5.282078E+04
	1		1.165324E-03		1.627026E-01		-1.626660E+01		4.087994E+02		-2.872064E+03	
	2	1.129945E-05		-9.372366E-04		1.900107E-01		-8.737800E+00		1.375665E+02		-6.998793E+02
	3		5.929536E-06		-2.904311E-03		2.213837E-01		-4.374725E+00		2.228111E+01	
	4	-6.578966E-09		1.902573E-06		-6.756294E-04		6.023519E-02		-1.330263E+00		1.014376E+01
m	5		-4.354626E-08		2.491752E-05		-1.555126E-03		2.510988E-02		-7.833420E-02	
	6	-3.423724E-11		4.218985E-08		3.457737E-06		-5.254818E-04		1.399077E-02		-1.201622E-01
	7		2.227255E-10		-9.261191E-08		5.228952E-06		-7.393083E-05		1.089590E-04	
	8	2.866875E-13		-1.486462E-10		-1.997760E-08		2.372106E-06		-6.290219E-05		5.386104E-04
	9		-3.223657E-13		1.138751E-10		-6.149132E-09		8.243884E-08		-7.632055E-08	
	10	-4.024940E-16		1.314642E-13		3.302405E-11		-3.319222E-09		8.621510E-08		-7.258410E-07

Table A1 polynomial fit coefficient for $\tilde{C}_L$

$\tilde{C}_L$						n			
		0	1	2	3	4	5	6	7
	0		6.338068E+00		1.456780E+00		-6.037411E+02		3.837259E+03
	1	4.994696E-02		-2.509928E-02		-1.319903E+01		1.469109E+02	
	2		-4.293500E-03		-1.303236E-01		6.878347E+00		-6.647025E+01
m	3	-3.437830E-05		-4.746167E-03		2.387172E-01		-2.523148E+00	
	4		6.026864E-06		7.385005E-05		-6.354210E-03		7.779223E-02
	5	-2.436665E-08		2.681350E-05		-1.272850E-03		1.388682E-02	
	6		-4.101547E-09		2.631810E-07		-4.626359E-06		-7.099461E-05
	7	8.121270E-11		-3.811993E-08		1.742040E-06		-1.834524E-05	

APPENDIX B

POLYNOMIAL FIT COEFFICIENT TABLES FOR NACA 0015 PARABOLIC FLAP AIRFOIL AT R_e = 600,000

Table B3 polynomial fit coefficient for $\tilde{C}_m$

$\tilde{C}_m$		n							
		0	1	2	3	4	5	6	7
m	0		2.804801E-02		-1.401790E+00		1.338933E+02		-1.306005E+03
	1	-1.180000E-02		-3.302174E-03		2.497991E+00		-2.971693E+01	
	2		1.393389E-04		8.512583E-02		-2.995890E+00		2.530880E+01
	3	7.155779E-06		1.395868E-03		-6.715662E-02		7.253644E-01	
	4		7.922415E-06		-1.026587E-03		2.909385E-02		-2.304811E-01
	5	2.363172E-08		-1.062682E-05		4.564869E-04		-4.857539E-03	
	6		-3.475792E-08		3.640899E-06		-9.437468E-05		6.850502E-04
	7	-9.680208E-11		3.033959E-08		-1.291638E-06		1.410865E-05	
	8		4.123527E-11		-4.041492E-09		1.000039E-07		-6.875065E-07
	9	1.026707E-13		-2.969652E-11		1.273962E-09		-1.421969E-08	

Table B2 polynomial fit coefficient for $\tilde{C}_D$

$\tilde{C}_D$		n										
		0	1	2	3	4	5	6	7	8	9	10
m	0	1.253277E-02		1.370723E-01		-6.290537E+00		4.820403E+02		-9.390870E+03		6.445554E+04
	1		1.128441E-03		1.632206E-01		-1.700202E+01		4.669316E+02		-3.654180E+03	
	2	1.084579E-05		-8.874386E-04		1.984075E-01		-1.048491E+01		1.943780E+02		-1.137555E+03
	3		6.006794E-06		-3.070188E-03		2.626106E-01		-6.631441E+00		4.997780E+01	
	4	-4.884299E-09		7.152607E-08		-5.299744E-04		5.626136E-02		-1.244809E+00		7.961740E+00
	5		-4.390498E-08		2.512621E-05		-1.817610E-03		4.241949E-02		-3.029896E-01	
	6	-4.831181E-11		6.185795E-08		-6.226695E-07		-2.469705E-04		6.795574E-03		-5.029761E-02
	7		2.105292E-10		-8.784067E-08		5.762808E-06		-1.256259E-04		8.333474E-04	
	8	3.177439E-13		-2.237578E-10		-4.267464E-10		9.602655E-07		-2.639931E-05		2.065702E-04
	9		-2.913599E-13		1.020401E-10		-6.367615E-09		1.337580E-07		-8.489945E-07	
	10	-4.225152E-16		2.244425E-13		5.723256E-12		-1.312314E-09		3.454393E-08		-2.712647E-07

Table B1 polynomial fit coefficient for $\tilde{C}_L$

$\tilde{C}_L$		n							
		0	1	2	3	4	5	6	7
m	0		6.368163E+00		9.837050E-01		-5.303267E+02		3.146583E+03
	1	5.044926E-02		3.712344E-03		-1.436612E+01		1.579584E+02	
	2		-3.521221E-03		-1.988070E-01		8.476455E+00		-7.812034E+01
	3	-3.423913E-05		-4.527918E-03		2.265988E-01		-2.414716E+00	
	4		-1.318734E-06		9.350243E-04		-3.187698E-02		2.871496E-01
	5	-1.719534E-08		2.286150E-05		-1.076829E-03		1.189620E-02	
	6		1.087184E-08		-1.598132E-06		5.008524E-05		-4.735149E-04
	7	6.337234E-11		-3.035935E-08		1.393271E-06		-1.535777E-05	

APPENDIX C

POLYNOMIAL FIT COEFFICIENT TABLES FOR NACA 0015 PARABOLIC FLAP AIRFOIL AT R_e = 700,000

Table C1 polynomial fit coefficient for $\tilde{C}_L$

$\tilde{C}_L$		n							
		0	1	2	3	4	5	6	7
m	0		6.404323E+00		-1.121690E+00		-4.209701E+02		2.247604E+03
	1	5.076415E-02		5.544886E-02		-1.635684E+01		1.733216E+02	
	2		-3.699224E-03		-1.303196E-01		5.540820E+00		-5.016412E+01
	3	-3.153899E-05		-5.184210E-03		2.517433E-01		-2.620897E+00	
	4		3.751849E-07		4.680462E-04		-1.261953E-02		1.025802E-01
	5	-2.888636E-08		2.552341E-05		-1.170519E-03		1.250731E-02	
	6		7.503641E-09		-7.869211E-07		1.675438E-05		-1.431747E-04
	7	7.869258E-11		-3.393444E-08		1.518668E-06		-1.615339E-05	

Table C2 polynomial fit coefficient for $\tilde{C}_D$

$\tilde{C}_D$		n										
		0	1	2	3	4	5	6	7	8	9	10
m	0	1.212236E-02		1.325932E-01		-6.915381E+00		5.382133E+02		-1.126433E+04		8.055758E+04
	1		1.114167E-03		1.560569E-01		-1.675669E+01		4.819793E+02		-3.962113E+03	
	2	1.055013E-05		-9.821686E-04		2.419135E-01		-1.499215E+01		3.369944E+02		-2.395029E+03
	3		5.952911E-06		-3.074256E-03		2.777645E-01		-7.730326E+00		6.535406E+01	
	4	-3.698469E-09		2.999745E-07		-9.274863E-04		1.003724E-01		-2.732998E+00		2.109498E+01
	5		-4.550699E-08		2.502244E-05		-1.948282E-03		5.240177E-02		-4.475040E-01	
	6	-6.184139E-11		6.676188E-08		-4.984544E-07		-3.370180E-04		1.122469E-02		-9.020302E-02
	7		2.114824E-10		-8.559867E-08		6.103343E-06		-1.583055E-04		1.337069E-03	
	8	3.537952E-13		-2.495088E-10		4.489743E-09		7.943884E-07		-2.761745E-05		2.192434E-04
	9		-2.857634E-13		9.753113E-11		-6.637325E-09		1.689214E-07		-1.423060E-06	
	10	-4.516963E-16		2.567106E-13		-2.718670E-12		-8.597949E-10		2.799424E-08		-2.137066E-07

Table C3 polynomial fit coefficient for $\tilde{C}_m$

$\tilde{C}_m$		n							
		0	1	2	3	4	5	6	7
m	0		2.434817E-02		-1.360632E+00		1.242933E+02		-1.176252E+03
	1	-1.186846E-02		-1.054204E-02		2.745899E+00		-3.106397E+01	
	2		-1.241064E-05		9.946096E-02		-3.403331E+00		2.969914E+01
	3	7.116168E-06		1.295431E-03		-5.835232E-02		5.765109E-01	
	4		9.373722E-06		-1.189674E-03		3.496399E-02		-3.013558E-01
	5	2.081749E-08		-8.585734E-06		3.200816E-04		-2.795720E-03	
	6		-3.880912E-08		4.162292E-06		-1.161925E-04		9.775940E-04
	7	-8.163357E-11		2.163308E-08		-7.294658E-07		5.683757E-06	
	8		4.457799E-11		-4.521946E-09		1.232500E-07		-1.031935E-06
	9	8.283297E-14		-1.900076E-11		5.843964E-10		-3.822163E-09	

APPENDIX D

POLYNOMIAL FIT COEFFICIENT TABLES FOR NACA 0015 PARABOLIC FLAP AIRFOIL AT R_e = 800,000

Table D3 polynomial fit coefficient for $\tilde{C}_m$

$\tilde{C}_m$		n							
		0	1	2	3	4	5	6	7
	0		2.042465E-02		-1.173327E+00		1.102441E+02		-1.000337E+03
	1	-1.191649E-02		-2.041545E-02		3.188106E+00		-3.582896E+01	
	2		-1.033760E-04		1.025705E-01		-3.284561E+00		2.700688E+01
	3	6.663793E-06		1.398280E-03		-6.331199E-02		6.457723E-01	
m	4		1.006808E-05		-1.199010E-03		3.299182E-02		-2.617871E-01
	5	2.348069E-08		-9.242349E-06		3.621814E-04		-3.514869E-03	
	6		-4.054338E-08		4.133184E-06		-1.068401E-04		8.016431E-04
	7	-8.865049E-11		2.389084E-08		-8.952933E-07		8.648075E-06	
	8		4.632245E-11		-4.481988E-09		1.119317E-07		-8.101078E-07
	9	8.976108E-14		-2.172515E-11		7.968248E-10		-7.687394E-09	

Table D2 polynomial fit coefficient for $\tilde{C}_D$

$\tilde{C}_D$		n										
		0	1	2	3	4	5	6	7	8	9	10
	0	1.178579E-02		1.281148E-01		-7.259545E+00		5.736868E+02		-1.250251E+04		9.144015E+04
	1		1.131753E-03		1.358528E-01		-1.501205E+01		4.378456E+02		-3.610753E+03	
	2	1.034643E-05		-1.156612E-03		2.938629E-01		-1.960194E+01		4.748448E+02		-3.631685E+03
	3		4.930500E-06		-2.580213E-03		2.332350E-01		-6.469623E+00		5.398787E+01	
	4	-4.641892E-09		3.567171E-06		-1.806742E-03		1.705203E-01		-4.789121E+00		3.979114E+01
m	5		-3.768289E-08		2.024193E-05		-1.511347E-03		3.945232E-02		-3.235768E-01	
	6	-5.320576E-11		4.229378E-08		4.880667E-06		-7.445426E-04		2.324667E-02		-2.025247E-01
	7		1.791606E-10		-6.692652E-08		4.437531E-06		-1.089284E-04		8.578894E-04	
	8	3.142871E-13		-1.758744E-10		-9.151523E-09		1.790551E-06		-5.757638E-05		5.090683E-04
	9		-2.408628E-13		7.370114E-11		-4.561650E-09		1.079232E-07		-8.290140E-07	
	10	-3.986105E-16		1.812268E-13		9.293685E-12		-1.710235E-09		5.435685E-08		-4.797627E-07

Table D1 polynomial fit coefficient for $\tilde{C}_L$

$\tilde{C}_L$		n							
		0	1	2	3	4	5	6	7
	0		6.429100E+00		-2.067211E+00		-3.615552E+02		1.802343E+03
	1	5.110754E-02		6.867178E-02		-1.659522E+01		1.720596E+02	
	2		-3.427826E-03		-1.329033E-01		5.091650E+00		-4.403098E+01
m	3	-3.114671E-05		-5.059913E-03		2.433353E-01		-2.536968E+00	
	4		-2.349070E-06		6.943055E-04		-1.780278E-02		1.406877E-01
	5	-2.624205E-08		2.337709E-05		-1.056317E-03		1.129155E-02	
	6		1.326887E-08		-1.418686E-06		3.524776E-05		-3.021687E-04
	7	6.941618E-11		-2.911089E-08		1.270101E-06		-1.344652E-05	

APPENDIX E

POLYNOMIAL FIT COEFFICIENT TABLES FOR NACA 0015 PARABOLIC FLAP AIRFOIL AT R_e = 900,000

Table E3 polynomial fit coefficient for $\tilde{C}_m$

$\tilde{C}_m$		n							
		0	1	2	3	4	5	6	7
	0		1.644177E-02		-9.357242E-01		9.664647E+01		-8.482256E+02
	1	-1.196049E-02		-2.695144E-02		3.414648E+00		-3.737964E+01	
	2		-1.161817E-04		9.586123E-02		-2.901080E+00		2.271759E+01
	3	6.306412E-06		1.453300E-03		-6.427400E-02		6.403107E-01	
m	4		9.617156E-06		-1.061864E-03		2.699999E-02		-1.986380E-01
	5	2.583702E-08		-9.646050E-06		3.728057E-04		-3.537880E-03	
	6		-3.703527E-08		3.424557E-06		-7.887389E-05		5.209121E-04
	7	-9.607803E-11		2.559952E-08		-9.548152E-07		9.006684E-06	
	8		4.059128E-11		-3.459452E-09		7.341280E-08		-4.338684E-07
	9	9.870152E-14		-2.416586E-11		8.911635E-10		-8.379944E-09	

Table E2 polynomial fit coefficient for $\tilde{C}_D$

$\tilde{C}_D$		n										
		0	1	2	3	4	5	6	7	8	9	10
	0	1.150341E-02		1.216638E-01		-6.996759E+00		5.656041E+02		-1.252675E+04		9.195068E+04
	1		1.174967E-03		1.098201E-01		-1.283018E+01		3.828373E+02		-3.219251E+03	
	2	1.008418E-05		-1.137625E-03		2.977079E-01		-2.034413E+01		4.999227E+02		-3.844865E+03
	3		3.367317E-06		-1.936357E-03		1.757515E-01		-4.886109E+00		4.184013E+01	
	4	-4.256372E-09		3.865592E-06		-1.907847E-03		1.760129E-01		-4.884161E+00		3.995891E+01
m	5		-2.658673E-08		1.487446E-05		-1.019809E-03		2.546949E-02		-2.125266E-01	
	6	-5.204005E-11		3.567024E-08		5.621795E-06		-7.513192E-04		2.257369E-02		-1.900910E-01
	7		1.423157E-10		-4.916637E-08		2.812486E-06		-6.247261E-05		4.876666E-04	
	8	2.944596E-13		-1.473898E-10		-1.155058E-08		1.770908E-06		-5.373599E-05		4.526662E-04
	9		-1.982558E-13		5.414557E-11		-2.769167E-09		5.651494E-08		-4.194506E-07	
	10	-3.646106E-16		1.456343E-13		1.219953E-11		-1.686164E-09		4.974217E-08		-4.130626E-07

Table E1 polynomial fit coefficient for $\tilde{C}_L$

$\tilde{C}_L$		n							
		0	1	2	3	4	5	6	7
	0		6.452283E+00		-3.147055E+00		-3.053916E+02		1.400745E+03
	1	5.137539E-02		8.546614E-02		-1.686333E+01		1.692290E+02	
	2		-3.468304E-03		-9.210325E-02		3.202004E+00		-2.476299E+01
m	3	-2.994401E-05		-5.195322E-03		2.446019E-01		-2.508227E+00	
	4		-2.268429E-06		5.103989E-04		-8.763237E-03		4.493770E-02
	5	-2.959184E-08		2.329824E-05		-1.028198E-03		1.075684E-02	
	6		1.298044E-08		-1.135185E-06		2.110407E-05		-1.460951E-04
	7	7.215808E-11		-2.857415E-08		1.212474E-06		-1.249872E-05	

APPENDIX F

POLYNOMIAL FIT COEFFICIENT TABLES FOR NACA 0015 PARABOLIC FLAP AIRFOIL AT R_e = 1,000,000

Table F1 polynomial fit coefficient for $\tilde{C}_L$

$\tilde{C}_L$		n							
		0	1	2	3	4	5	6	7
m	0		6.472310E+00		-4.121284E+00		-2.546114E+02		1.002048E+03
	1	5.160916E-02		1.006542E-01		-1.722202E+01		1.704526E+02	
	2		-3.473464E-03		-5.771197E-02		1.478606E+00		-5.732392E+00
	3	-2.867136E-05		-5.408677E-03		2.530505E-01		-2.618305E+00	
	4		-2.653874E-06		3.986004E-04		-1.674315E-03		-4.302100E-02
	5	-3.388669E-08		2.387993E-05		-1.052313E-03		1.115116E-02	
	6		1.403519E-08		-1.055265E-06		1.315884E-05		-3.148856E-05
	7	7.632730E-11		-2.901608E-08		1.233640E-06		-1.296834E-05	

Table F2 polynomial fit coefficient for $\tilde{C}_D$

$\tilde{C}_D$		n										
		0	1	2	3	4	5	6	7	8	9	10
m	0	1.126201E-02		1.138299E-01		-6.385950E+00		5.382860E+02		-1.213318E+04		8.996688E+04
	1		1.231231E-03		7.989080E-02		-1.028219E+01		3.144609E+02		-2.677388E+03	
	2	9.774487E-06		-9.909637E-04		2.763171E-01		-1.944314E+01		4.866536E+02		-3.808152E+03
	3		1.500085E-06		-1.168063E-03		1.053065E-01		-2.806711E+00		2.376727E+01	
	4	-3.074378E-09		2.503390E-06		-1.670107E-03		1.600932E-01		-4.510483E+00		3.769013E+01
	5		-1.271115E-08		8.461106E-06		-4.150351E-04		6.898425E-03		-4.370229E-02	
	6	-5.334943E-11		3.680185E-08		4.800349E-06		-6.633820E-04		2.005430E-02		-1.733500E-01
	7		9.653905E-11		-2.800174E-08		8.064769E-07		6.478765E-08		-9.211073E-05	
	8	2.762882E-13		-1.325411E-10		-1.121856E-08		1.597128E-06		-4.765141E-05		4.116528E-04
	9		-1.456020E-13		3.046958E-11		-5.209145E-10		-1.411884E-08		2.427891E-07	
	10	-3.284525E-16		1.172543E-13		1.350913E-11		-1.581769E-09		4.483416E-08		-3.805478E-07

Table F3 polynomial fit coefficient for $\tilde{C}_m$

$\tilde{C}_m$		n							
		0	1	2	3	4	5	6	7
m	0		1.251096E-02		-6.533281E-01		8.210532E+01		-6.877919E+02
	1	-1.199424E-02		-3.459881E-02		3.728542E+00		-4.049638E+01	
	2		-7.220719E-05		8.062985E-02		-2.189687E+00		1.474666E+01
	3	5.617266E-06		1.652578E-03		-7.390343E-02		7.585464E-01	
	4		8.672131E-06		-8.484818E-04		1.784132E-02		-9.702383E-02
	5	3.199156E-08		-1.167450E-05		4.798038E-04		-4.923215E-03	
	6		-3.264222E-08		2.565528E-06		-4.332134E-05		1.300434E-04
	7	-1.169934E-10		3.306505E-08		-1.359581E-06		1.430802E-05	
	8		3.515397E-11		-2.451736E-09		3.224361E-08		1.936415E-08
	9	1.223133E-13		-3.292568E-11		1.372297E-09		-1.472511E-08	

APPENDIX G

POLYNOMIAL FIT COEFFICIENT TABLES FOR NACA 0015 PARABOLIC FLAP AIRFOIL AT R_e = 1,100,000

Table G3 polynomial fit coefficient for $\tilde{C}_m$

$\tilde{C}_m$		n							
		0	1	2	3	4	5	6	7
m	0		9.028071E-03		-4.167995E-01		7.057869E+01		-5.673680E+02
	1	-1.202338E-02		-4.112445E-02		3.972410E+00		-4.262150E+01	
	2		-4.213308E-05		6.937922E-02		-1.688399E+00		9.513325E+00
	3	4.997309E-06		1.816146E-03		-8.100096E-02		8.357417E-01	
	4		8.220185E-06		-7.370752E-04		1.334300E-02		-5.211615E-02
	5	3.669069E-08		-1.303496E-05		5.429770E-04		-5.638641E-03	
	6		-3.118715E-08		2.273525E-06		-3.280896E-05		3.806299E-05
	7	-1.300261E-10		3.701569E-08		-1.542476E-06		1.632361E-05	
	8		3.417796E-11		-2.287491E-09		2.797351E-08		3.382808E-08
	9	1.341386E-13		-3.649984E-11		1.531599E-09		-1.636993E-08	

Table G2 polynomial fit coefficient for $\tilde{C}_D$

$\tilde{C}_D$		n										
		0	1	2	3	4	5	6	7	8	9	10
m	0	1.105130E-02		1.062088E-01		-5.703164E+00		5.070025E+02		-1.165265E+04		8.751458E+04
	1		1.274124E-03		5.532465E-02		-8.268602E+00		2.643118E+02		-2.330155E+03	
	2	9.454678E-06		-8.241372E-04		2.482180E-01		-1.803758E+01		4.604823E+02		-3.665128E+03
	3		-1.587689E-08		-5.147949E-04		4.821709E-02		-1.259041E+00		1.197747E+01	
	4	-1.247799E-09		6.933233E-07		-1.293295E-03		1.344672E-01		-3.892426E+00		3.329946E+01
	5		-7.721216E-10		2.746858E-06		8.536535E-05		-6.808435E-03		6.254006E-02	
	6	-6.135391E-11		4.462748E-08		2.322670E-06		-4.641882E-04		1.476281E-02		-1.325431E-01
	7		5.444505E-11		-8.340151E-09		-8.814394E-07		4.547628E-05		-4.371758E-04	
	8	2.869673E-13		-1.509355E-10		-3.220560E-09		9.114461E-07		-2.883143E-05		2.611391E-04
	9		-9.464770E-14		7.695550E-12		1.397120E-09		-6.454665E-08		6.148174E-07	
	10	-3.315838E-16		1.372268E-13		3.901364E-12		-7.475246E-10		2.175250E-08		-1.926794E-07

Table G1 polynomial fit coefficient for $\tilde{C}_L$

$\tilde{C}_L$		n							
		0	1	2	3	4	5	6	7
m	0		6.486454E+00		-4.387091E+00		-2.351970E+02		9.324441E+02
	1	5.186308E-02		9.288474E-02		-1.631773E+01		1.551968E+02	
	2		-3.296270E-03		-5.796230E-02		1.234340E+00		-3.595684E+00
	3	-2.885558E-05		-5.021286E-03		2.272644E-01		-2.262472E+00	
	4		-4.419147E-06		5.471211E-04		-5.767531E-03		-1.483949E-03
	5	-2.972230E-08		2.096663E-05		-8.749127E-04		8.771251E-03	
	6		1.755576E-08		-1.442715E-06		2.553992E-05		-1.559246E-04
	7	6.717569E-11		-2.391145E-08		9.320457E-07		-8.955091E-06	

APPENDIX H

PYTHON SCRIPT: RECEIVE USER INPUTS AND FORMAT OUTPUT

```
import Austin_opt

#import optix

from numpy import array, zeros


"""

Author:        Austin Stewart

Date:          4 March 2019

Input:         Takes no input

Output:        Minimum drag value for a wing

Example usage:

               import Austin_outer_loop as AOL

               Drag=AOL.Austin_outer_loop()

"""


def Austin_outer_loop():

    # Have user give inputs

    rho=1.225 #float(input('What is the density of the air? \nDensity in kg/m^3 (Standard
sea level= 1.229)\n'))

    mu=1.789*10**-5 #float(input('What is the dynamic viscosity of the air? \nDynamic
viscosity in kg/m*s (Standard sea level= 1.73*10**-5)\n'))
```

```
    Re=999999.999999#float(input('What Reynolds number is your elliptic lift distribution wing at? \nValue between 0.6e6 and 1.0e6\n'))
    CL=0.5#float(input('What is the lift coefficient of your wing? \nCL is dimensionless (0.5 for testing)\n'))
    #c=float(input('What is the chord length of the elliptic wing that is being compared? \nLength in m\n'))
    Ra=8.0#float(input('What is the aspect ratio for the elliptic lift distribution wing that the wing is being compared to? \nAspect ratio is dimensionless (8 for testing)\n'))
    [B3,lift_case]=get_B3_value()
    [variable_case,washout_val,camber_val,root_twist]=get_variable_case()
    x_length=get_control_point_length()
    c_ell=1.0
    v=Re*mu/(rho*c_ell)
    b_ell=Ra*c_ell
    weight=CL*0.5*rho*v**2*b_ell*c_ell
    b_opt=b_ell*(1/(1+B3))**(1.0/3)
    c_opt=(b_ell*c_ell)/b_opt
    Re_opt=v*rho*c_opt/mu
    # Set inputs into form used in optix

args=[root_twist,B3,rho,v,weight,camber_val,washout_val,variable_case,b_opt,c_opt,Re_opt]
    # Determine size of x based on way lift distribution is being matched
    if variable_case == 'Aero' or variable_case == 'Geo':
        x=zeros((x_length,1))
        # Set the values in x if known to converge to value faster
#
==============================================================
==============
#         x = array([[4.595],
#                    [5.2470],
#                    [-1.0235],
#                    [-6.5705],
#                    [-10.1554]])
#
==============================================================
==============
    # Allow both Camber and Washout
    if variable_case == 'Both':
        x=zeros((2*x_length,1))
    Optimized_Drag = Austin_opt.min_CD_optix(x,args)
    # Use Optix to find the minimum drag case for the rectangular wing
#
==============================================================
==============
#     min_drag = optix.minimize(Austin_opt.min_CD_optix,x,args,
#                termination_tol=1e-12,
#                grad_tol=1e-12,
#                verbose=False,
#                max_processes=1,
#                dx=0.001,
#                max_iterations=1000
#                )
#
==========================================================================
==============
    print('Lift Case:',lift_case,'\n')
    if lift_case == 'Other':
        print('B3 Value:',B3,'\n')
    print('Variable Case:',variable_case,'\n')
    print('Reynolds number chosen B3:',Re_opt,'\n')
    print('Optimized Aspect ratio:', b_opt/c_opt,'\n')
    return Optimized_Drag


def get_variable_case():
    variable_case=input('Do you want to vary aerodynamic or geometric twist or both?
\nAcceptable values Aero or Geo or Both\n')
    if variable_case == 'Geo':
        washout_val=0.0
        camber_val=float(input('What is constant aerodynamic twist value for the wing?
\nAerodynamic twist value in percent (-20 to 20)\n'))
        root_twist=float(input('What is the geometric twist at the root of your wing?
\nGeometric twist value in deg (-15 to 15)\n\n'))
    elif variable_case == 'Aero':
        camber_val=0.0
        washout_val=float(input('What is constant geometric twist value for the wing?
\nGeometric twist value in deg (-15 to 15)\n'))
        root_twist=float(input('What is the aerodynamic twist at the root of your wing?
\nAerodynamic twist value in percent (-20 to 20)\n\n'))
    elif variable_case == 'Both':
        camber_val=0.0
        washout_val=0.0
    else:
        print('Error: please provide appropriate answer.')
        [variable_case,washout_val,camber_val,root_twist]=get_variable_case()
    return(variable_case,washout_val,camber_val,root_twist)
def get_control_point_length():
    x_length=int(input('How many control points do you want to have on the wing
including root point? (integer between 1 and 20)\n'))
    if not 1<= x_length <= 20:
        print('Please choose appropriate value.')
        [x_length]=get_control_point_length()
    return x_length


def get_B3_value():
    lift_case=input('What distribution case are you trying to match?\nElliptic = 1 \nFixed
lift dist, fixed net weight, fixed max stress, fixed stall speed or Prandtls 1933 = 2 \nFixed
lift dist, fixed gross weight, fixed max stress, fixed wing loading = 3 \nFixed lift dist,
fixed gross weight, fixed max deflection, fixed wing loading = 4 \nFixed lift dist, fixed
net weight, fixed max deflection, fixed stall speed= 5\nUser Specified B3 (enter
Other)\n')
    if lift_case == 'Other':
        B3=float(input('What is the B3 value you would like to use?\nTypical range is -1/3
to 0\n'))
    elif int(float(lift_case)) in [1,2,3,4,5]:
        lift_case=int(float(lift_case))
        # Determine B3 value to use based on lift distribution case
        # Lift Distribution Case {Elliptic}
        if lift_case==1:
            B3=0.0
        # Lift Distribution Case {fixed lift dist, fixed net weight, fixed max stress, fixed stall
speed} {Prandtl's 1933}
        elif lift_case==2:
            B3=-1.0/3
        # Lift Distribution Case {fixed lift dist, fixed gross weight, fixed max stress, fixed wing loading}
        elif lift_case==3:
            B3=-0.13564322
        # Lift Distribution Case {fixed lift dist, fixed gross weight, fixed max deflection, fixed wing loading}
        elif lift_case==4:
            B3=-0.05971587
        # Lift Distribution Case {fixed lift dist, fixed net weight, fixed max deflection, fixed stall speed}
        elif lift_case==5:
            B3=-0.17714856
    else:
        print('\nError: Please provide appropriate answer.\n')
        [B3,lift_case]=get_B3_value()
    return(B3,lift_case)
```

APPENDIX I

PYTHON SCRIPT: WRAPPER TO OPTIX

```
import optix
import Austin_mach as AM
import os
import json
import numpy as np
import Reynolds_Interpolation as RI
import math
"""
Author: Austin Stewart
Date:   4 March 2019
Input: -vars_in: Type(list), Size(varies 5 or 10), variable that will
        be used to match the given lift distribution case
       -const_in: Type(list), Size(7), values that will be constant
        while matching lift distribution,
        const_in[0] = density in kg/m^3
        const_in[1] = velocity of wing in m/s
        const_in[2] = dynamic viscosity in kg/m*s
        const_in[3] = weight of wing in N
        const_in[4] = lift distribution case that is being matched see lines 32-45 for
explanation
Output: Drag for a wing that matches a lift distribution
```

```
"""


def min_CD_optix(var_in,const_in):
    Opt_case=AM.Match_CL()
    Opt_case.set_vars(const_in)
    x0=var_in
    # Use optix on match CL
    Opt = optix.minimize(Opt_case.Run_MachUp,x0,
                                    termination_tol=1e-8,
                                    grad_tol=1e-5,
                                    verbose=True,
                                    max_processes=8,
                                    dx=0.1,
                                    max_iterations=1000,
                                    alpha_mult=2.0)
    print('\nThe twist profile that matches the analytic lift distribution.')
    print('alpha:',Opt.x[0,0],'(deg)')
    print('Root twist:',Opt_case.root_twist)
    size_x=len(x0)
    for i in range(1,size_x-1):
        thet=math.pi/2*(1+i/(size_x-1))
        z=-math.cos(thet)
```

```
#
==================================================================
==============
#        z=i/size_x # for even spacing along the semi-span
#
==================================================================
==============
        print('Twist at ',z,' span:',Opt.x[i,0])
    print('Twist at full span:',Opt.x[size_x-1,0],'\n')
    print('\nThe RMS value for the results.')
    print(Opt.f)
    print('\n\n')
    # Create Reynolds specific airfoil values
    RI.Reynolds_Interpolation(const_in[10])
    # Use MachUp to determine CD from the matching CL

    if Opt_case.Twist_type=='Geo':
        wash=np.zeros(size_x)
        camb=np.zeros(size_x)
        wash[0]=Opt_case.root_twist
        camb[0]=Opt_case.Camber_value
        for i in range(1,size_x):
            wash[i]=float(Opt.x[i,0])
            camb[i]=Opt_case.Camber_value
    if Opt_case.Twist_type == 'Aero':
        wash=np.zeros(size_x)
        camb=np.zeros(size_x)
        camb[0]=Opt_case.root_twist
        wash[0]=Opt_case.Washout_value
        for i in range(1,size_x):
            camb[i]=float(Opt.x[i,0])
            wash[i]=Opt_case.Washout_value
    if Opt_case.Twist_type == 'Both':
        wash=np.zeros(size_x/2)
        camb=np.zeros(size_x/2)
        camb[0]=Opt.x[size_x/2]
        wash[0]=Opt.x[0]
        for i in range(1,size_x):
            wash[i]=Opt.x[i]
            camb[i]=Opt.x[i+size_x/2]


    # Generate washout input file
    CambWash_length=len(wash)
    twist_vars = {'r1':  {'c1': 0.00, 'c2': wash[0]}}
    for i in range(1,CambWash_length):
        thet=math.pi/2*(1+i/(CambWash_length-1))
```

```
        z=-math.cos(thet)
#
=========================================================================
==============
#        z=i/size_x # for even spacing along the semi-span
#
=========================================================================
==============
        twist_vars.update({'r'+str(i+1):  {'c1': z, 'c2': wash[i]}})
    with open('Final_washout.json', 'w') as data_file:
        json.dump(twist_vars, data_file, sort_keys=True, indent=4)
    # Generate airfoil ratio input file
    af_ratio_vars = {'r1':  {'c1': 0.00, 'c2': camb[0]}}
    for i in range(1,CambWash_length):
        thet=math.pi/2*(1+i/(CambWash_length-1))
        z=-math.cos(thet)
#
=========================================================================
==============
#        z=i/size_x # for even spacing along the semi-span
#
=========================================================================
==============
    af_ratio_vars.update({'r'+str(i+1):  {'c1': z, 'c2': camb[i]}})
with open('Final_af_ratio.json', 'w') as data_file:
    json.dump(af_ratio_vars, data_file, sort_keys=True, indent=4)
machup_input = json.load(open('Final_input.json'))
# change angle of attack to achieve desired lift (scales lift distribution)
machup_input['condition']['alpha'] = Opt.x[0,0]
machup_input['reference']['area'] = Opt_case.S_opt
machup_input['reference']['lateral_length'] = Opt_case.b_opt
machup_input['reference']['longitudinal_length'] = Opt_case.c_opt
machup_input['wings']['Main']['root_chord'] = Opt_case.c_opt
machup_input['wings']['Main']['tip_chord'] = Opt_case.c_opt
machup_input['wings']['Main']['span'] = Opt_case.b_opt/2
with open('Final_input.json', 'w') as machup_file:
    json.dump(machup_input, machup_file, sort_keys=True, indent=4)
# Execute MachUp
os.system('./MachUp.out Final_input.json > Final_values.txt')
#####################################################################
# Extract data from distributions output file
CD_dist_temp=[]
CL_dist_temp=[]
y_coord_temp=[]
sec_alpha_temp=[]
with open('Final_output.txt') as Machup_data:
    for line in Machup_data.readlines()[1:201]:
        line = line.strip()
        Name, controlx, controly, controlz, ch, twist, sweep, dihed, area, sec_alph, \
            CL_list, CD_list, Cm_dist_Machup, CL_ref, sec_alph_L0 = line.split()
        CD_dist_temp.append(float(CD_list))
        CL_dist_temp.append(float(CL_list))
        y_coord_temp.append(float(controly))
        sec_alpha_temp.append((Opt.x[0,0]-float(sec_alph))*np.pi/180)
# Print the lift distributions out to see if they match
z_size=int(len(CL_dist_temp))
# Print the MachUp lift distribution out
# Determine what the z step size is
CL_dist_Machup=np.zeros([z_size])
CD_par=np.zeros([z_size])
y_cord=np.zeros([z_size])
sec_alpha_coord=np.zeros([z_size])
#move values so MachUp follows same - b/2 to b/2
for i in range (0,int(z_size/2)):
    CL_dist_Machup[i]=CL_dist_temp[int(z_size/2)-i-1]
    CD_par[i]=CD_dist_temp[int(z_size/2)-i-1]
    y_cord[i]=y_coord_temp[int(z_size/2)-i-1]
    sec_alpha_coord[i]=sec_alpha_temp[int(z_size/2)-i-1]
for i in range (int(z_size/2),z_size):
    CL_dist_Machup[i]=CL_dist_temp[i]
    CD_par[i]=CD_dist_temp[i]
    y_cord[i]=y_coord_temp[i]
    sec_alpha_coord[i]=sec_alpha_temp[i]
# Integrate the CD distribution to find total drag
CD_p=np.zeros([z_size])
CD_i=np.zeros([z_size])
CD_t=np.zeros([z_size])
for i in range (0,z_size):
    CD_p[i]= CD_par[i]*np.cos(sec_alpha_coord[i])
    CD_i[i]= CL_dist_Machup[i]*np.sin(sec_alpha_coord[i])
    CD_t[i]= CD_p[i]+CD_i[i]
CD_Opt=0
for i in range (1,z_size):
    CD_Opt+=((CD_t[i-1]+CD_t[i])/2*np.abs(y_cord[i]-y_cord[i-1]))/Opt_case.b_opt
print('\nMachUp Lift Distribution\n')
for i in range (0,z_size):
    print(CL_dist_Machup[i])
print('\nAnalytic Lift Distribution\n')
#CL_dist_diff=np.zeros([z_size])
for i in range (0,z_size):
    z=y_cord[i]
    theta=math.acos(-z*2.0/Opt_case.b_opt)
```

```
        # Calculate analytic CL needed to match

CL_dist_analytic=Opt_case.weight/Opt_case.b_opt/(1/2.0*Opt_case.rho*Opt_case.v**2
*Opt_case.c_opt)* (4.0/math.pi*(math.sin(theta) \
                                +Opt_case.B_3*math.sin(3.0*theta)))
        print(CL_dist_analytic)
    print('\nZ/b Distribution\n')
    for i in range (0,z_size):
        z=y_cord[i]
        print(z/Opt_case.b_opt)
        #CL_dist_diff[i]=(CL_dist_analytic-CL_dist_Machup[i])**2
    return CD_Opt
```

APPENDIX J

PYTHON SCRIPT: WRAPPER TO MACHUP

```
import os

import json

import shutil

import math

from numpy import zeros

import Reynolds_Interpolation as RI

import uuid

"""

Author: Austin Stewart

Date:   4 March 2019

Input:

Output:

"""


class Match_CL():


    def __init__(self):

        self.alpha=0.0

        self.root_twist=0.0

        self.B_3=0.0

        self.rho=0.0
        self.v=0.0

        self.weight=0.0

        self.Camber_value=0.0

        self.Washout_value=0.0

        self.Twist_type="String"

        self.b_opt=0.0

        self.c_opt=0.0

        self.RE=0.0

        self.work_dir="/home/austin/Documents/Integration_py_mach"

        self.orig_dir=self.work_dir + '/' + 'Original_case'

        self.S_opt=0.0


    def set_vars(self,args):

        self.root_twist=args[0]

        self.B_3=args[1]

        self.rho=args[2]

        self.v=args[3]

        self.weight=args[4]

        self.Camber_value=args[5]

        self.Washout_value=args[6]

        self.Twist_type=args[7]

        self.b_opt=args[8]

        self.c_opt=args[9]
        self.RE=args[10]
        self.S_opt=self.b_opt*self.c_opt


    def Run_MachUp(self,x):
        alpha = x[0,0]
        size_x=len(x)
#
==============================================================
==============
        case_uuid=str(uuid.uuid4())
        # Copy original files into case directory
        shutil.copytree(self.orig_dir, case_uuid)
        # Make the temporary directory current
        os.chdir(case_uuid)
#
==============================================================
==============
        # Calculate Reynolds
        machup_input = json.load(open('input.json'))
        # Create Reynolds specific airfoil values
        RI.Reynolds_Interpolation(self.RE)
        if self.Twist_type=='Geo':
            wash=zeros(size_x)
    camb=zeros(size_x)
    wash[0]=self.root_twist
    camb[0]=self.Camber_value
    for i in range(1,size_x):
        wash[i]=float(x[i,0])
        camb[i]=self.Camber_value
if self.Twist_type == 'Aero':
    wash=zeros(size_x)
    camb=zeros(size_x)
    camb[0]=self.root_twist
    wash[0]=self.Washout_value
    for i in range(1,size_x):
        camb[i]=float(x[i,0])
        wash[i]=self.Washout_value
if self.Twist_type == 'Both':
    wash=zeros(size_x/2)
    camb=zeros(size_x/2)
    camb[0]=x[size_x/2]
    wash[0]=x[0]
    for i in range(1,size_x):
        wash[i]=x[i]
        camb[i]=x[i+size_x/2]
# Generate washout input file
```

```
    CambWash_length=len(wash)
    twist_vars = {'r1':  {'c1': 0.00, 'c2': wash[0]}}
    for i in range(1,CambWash_length):
        thet=math.pi/2*(1+i/(CambWash_length-1))
        z=-math.cos(thet)
#
========================================================================================
==================
#           z=i/size_x # for even spacing along the semi-span
#
========================================================================================
==================
        twist_vars.update({'r'+str(i+1):  {'c1': z, 'c2': wash[i]}})
    with open('washout.json', 'w') as data_file:
        json.dump(twist_vars, data_file, sort_keys=True, indent=4)
    # Generate airfoil ratio input file
    af_ratio_vars = {'r1':  {'c1': 0.00, 'c2': camb[0]}}
    for i in range(1,CambWash_length):
        thet=math.pi/2*(1+i/(CambWash_length-1))
        z=-math.cos(thet)
#
========================================================================================
==================
#           z=i/size_x # for even spacing along the semi-span
#
==============================================================
==============
        af_ratio_vars.update({'r'+str(i+1):  {'c1': z, 'c2': camb[i]}})
    with open('af_ratio.json', 'w') as data_file:
        json.dump(af_ratio_vars, data_file, sort_keys=True, indent=4)
    # change angle of attack to achieve desired lift (scales lift distribution)
    machup_input['condition']['alpha'] = alpha
    machup_input['reference']['area'] = self.S_opt
    machup_input['reference']['lateral_length'] = self.b_opt
    machup_input['reference']['longitudinal_length'] = self.c_opt
    machup_input['wings']['Main']['root_chord'] = self.c_opt
    machup_input['wings']['Main']['tip_chord'] = self.c_opt
    machup_input['wings']['Main']['span'] = self.b_opt/2
    with open('input.json', 'w') as machup_file:
        json.dump(machup_input, machup_file, sort_keys=True, indent=4)
    # Execute MachUp
    os.system('./MachUp.out input.json > out.txt')
#
==============================================================
==============
    # Extract data from distributions output file
```

```
    CL_dist_temp=[]

    y_coord_temp=[]

    with open('myfile.txt') as Machup_data:

        for line in Machup_data.readlines()[1:201]:

            line = line.strip()

            Name, controlx, controly, controlz, ch, twist, sweep, dihed, area, sec_alph, \

                c11, CD_p_dist_Machup, Cm_dist_Machup, CL_ref, sec_alph_L0 =
line.split()

            CL_dist_temp.append(float(c11))

            y_coord_temp.append(float(controly))

    # Determine what the z step size is

    z_size=int(len(CL_dist_temp))

    #z_step=float(self.b_opt/(z_size-1))

    CL_dist_Machup=zeros([z_size])

    y_cord=zeros([z_size])

    #move values so MachUp follows same - b/2 to b/2

    for i in range (0,int(z_size/2)):

        CL_dist_Machup[i]=CL_dist_temp[int(z_size/2)-i-1]

        y_cord[i]=y_coord_temp[int(z_size/2)-i-1]

    for i in range (int(z_size/2),z_size):

        CL_dist_Machup[i]=CL_dist_temp[i]

        y_cord[i]=y_coord_temp[i]

#
```

```
==============================================================
==============
#        print('\nMachUp Lift Distribution\n')
#        for i in range (0,z_size):
#            print(CL_dist_Machup[i])
#
==============================================================
==============
      CL_dist_diff=zeros([z_size])
      for i in range (0,z_size):
          z=y_cord[i]
          theta=math.acos(-z*2.0/self.b_opt)
          # Calculate analytic CL needed to match
          CL_dist_analytic=self.weight/self.b_opt/(1/2.0*self.rho*self.v**2*self.c_opt)*
(4.0/math.pi*(math.sin(theta) \
                            +self.B_3*math.sin(3.0*theta)))
          # Compare CL data to analytic CL
          CL_dist_diff[i]=(CL_dist_analytic-CL_dist_Machup[i])**2
      #Calculate RMS
      CL_RMS=math.sqrt(1.0/z_size*sum(CL_dist_diff))
#
==============================================================
==============
      os.chdir(self.work_dir)

      shutil.rmtree(case_uuid)

      return CL_RMS
```

APPENDIX K

PYTHON SCRIPT: FORMATS POLYNOMIAL FIT COEFFICEINTS INTO READABLE FILE

```
import json
import Linear_Interpolation as LI
"""
Author: Austin Stewart
Date:   31 January 2019
Input:  Exact Reynold's Number
Output: Set of curve fit coefficients that to allow CL, CD and Cm to be
        determined given camber and angle of attack on a wing section
"""
def Reynolds_Interpolation(Re):
    """Inputs"""
    data_file='Airfoil_data.json' # this data file has the polynomial fit found using XFOIL
    Reynolds_low=5e5
    Reynolds_high=1.1e6
    """Determine sections of json to use"""
    Re_round=Re//100000
    if Re > Reynolds_high or Re < Reynolds_low:
        print("Reynolds Number outside of range.\nReynolds must be between %d and %d.
\n"%(Reynolds_low,Reynolds_high))
        return
    with open(data_file) as f:
        Curve_fit_data=json.load(f)
    Re_file_low=Re_round*100000//1
    Re_file_high=Re_file_low+100000//1
    # Create python dictionary
    Re_specific={'Re_specific' : {
        'properties' :  {
                'type' :  'polynomial',
                'is_function' : 1,
                'CL' : {},
                'CD' : {},
                'Cm' : {},
                'CL_max' :  "",
                'Comments' : "All angles in radians and slopes in 1/radians"}}}
    #CL
    variable='CL'
    alpha_range=7
    camber_range=7

Re_specific=odd_airfoil_value(Re_specific,Re,variable,alpha_range,camber_range,Re_fi
le_low,Re_file_high,Curve_fit_data)

    #CD
    variable='CD'
    alpha_range=10
    camber_range=7

Re_specific=even_airfoil_value(Re_specific,Re,variable,alpha_range,camber_range,Re_f
ile_low,Re_file_high,Curve_fit_data)

    #Cm
    variable='Cm'
    alpha_range=7
    camber_range=9

Re_specific=odd_airfoil_value(Re_specific,Re,variable,alpha_range,camber_range,Re_fi
le_low,Re_file_high,Curve_fit_data)

    # Turn dictionary into json
    with open('Re_specific.json','w') as outfile:
        json.dump(Re_specific,outfile, indent=4)
    return

def
odd_airfoil_value(Re_specific,Re,variable,alpha_range,camber_range,Re_file_low,Re_fil
e_high,Curve_fit_data):
    for i in range (0,alpha_range):
        c_alpha='C'+str(i)
        Re_specific['Re_specific']['properties'][variable][c_alpha]={
                }
        j=0
        while j <= camber_range:
            if i % 2 ==0:
                j=j+1
                c_camber='C'+str(j)

value=LI.Linear_Interpolation(Re,Re_file_low,Re_file_high,Curve_fit_data[str(int(Re_fi
le_low))][variable][c_alpha][c_camber],Curve_fit_data[str(int(Re_file_high))][variable][
c_alpha][c_camber])
                Re_specific['Re_specific']['properties'][variable][c_alpha][c_camber] = value
            else:
                c_camber='C'+str(j)

value=LI.Linear_Interpolation(Re,Re_file_low,Re_file_high,Curve_fit_data[str(int(Re_fi
le_low))][variable][c_alpha][c_camber],Curve_fit_data[str(int(Re_file_high))][variable][
c_alpha][c_camber])
                Re_specific['Re_specific']['properties'][variable][c_alpha][c_camber] = value
                j=j+1
            j=j+1
    return Re_specific


def
even_airfoil_value(Re_specific,Re,variable,alpha_range,camber_range,Re_file_low,Re_f
ile_high,Curve_fit_data):
    for i in range (0,alpha_range):
        c_alpha='C'+str(i)
        Re_specific['Re_specific']['properties'][variable][c_alpha]={
                }
        j=0
        while j <= camber_range:
            if i % 2 ==0:
                c_camber='C'+str(j)

value=LI.Linear_Interpolation(Re,Re_file_low,Re_file_high,Curve_fit_data[str(int(Re_fi
le_low))][variable][c_alpha][c_camber],Curve_fit_data[str(int(Re_file_high))][variable][
c_alpha][c_camber])
                Re_specific['Re_specific']['properties'][variable][c_alpha][c_camber] = value
                j=j+1
            else:
                j=j+1
                c_camber='C'+str(j)
```

```
value=LI.Linear_Interpolation(Re,Re_file_low,Re_file_high,Curve_fit_data[str(int(Re_fi
le_low))][variable][c_alpha][c_camber],Curve_fit_data[str(int(Re_file_high))][variable][
c_alpha][c_camber])
            Re_specific['Re_specific']['properties'][variable][c_alpha][c_camber] = value
         j=j+1
   return Re_specific
```

APPENDIX L

PYTHON SCRIPT: PERFORM LINEAR INTERPOLATION

```
def Linear_Interpolation(Re,Re_low,Re_high,term_low,term_high):
    y=term_low+(Re-Re_low)*(term_high-term_low)/(Re_high-Re_low)
    return y
```