Impact of Parasitic Drag on a Family of Optimal Lift Distributions

Austin J. Stewart

*Utah State University*

Follow this and additional works at: [https://digitalcommons.usu.edu/etd](https://digitalcommons.usu.edu/etd)

Part of the *Aerospace Engineering Commons*

**Recommended Citation**


[https://digitalcommons.usu.edu/etd/7740](https://digitalcommons.usu.edu/etd/7740)

This Thesis is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.
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

ABSTRACT ................................................................................................................................. iii

ACKNOWLEDGMENTS .............................................................................................................. v

LIST OF TABLES ........................................................................................................................ vii

LIST OF FIGURES ...................................................................................................................... viii

NOMENCLATURE ..................................................................................................................... ix

CHAPTER

I. INTRODUCTION .................................................................................................................... 1

II. PREDICTING DRAG ON AN ARBITRARY WING .............................................................. 8

III. RESULTS ............................................................................................................................ 14
    Geometric Twist or Washout .................................................................................................. 16
    Aerodynamic Twist or Camber ............................................................................................. 19
    Comparison of Drag Components ......................................................................................... 21

IV. CONCLUSIONS .................................................................................................................... 25

REFERENCES ............................................................................................................................ 27

APPENDICES ............................................................................................................................ 30
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$B_3$ values used in study with associated design constraints</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Effect of lift distribution, specifically the $B_3$ coefficient, on allowable $b$ with constant $W_s$</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Airfoil used for study and visual of twist types.</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>XFOIL data for lift, drag, and pitching moment coefficients as a function of $\alpha$ fitted with polynomial equations</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Comparison of lift distributions generated by MachUp and Eq. (20) for a $B_3 = 0.0$</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>Flow chart showing path of data and method</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Wing planforms of various values of $B_3$</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Lift distributions of various values of $B_3$</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>Reynold’s number changes with respect to the $B_3$ Fourier coefficient</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>Drag values for a variety of root twist and $B_3$ values when using washout</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>Minimum drag value as a function of $B_3$ using washout</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>Drag values for a variety of root camber and $B_3$ values when using camber</td>
<td>19</td>
</tr>
<tr>
<td>12</td>
<td>Minimum drag given a $B_3$ value using camber</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>Comparison of minimum drag values for different twist types</td>
<td>21</td>
</tr>
<tr>
<td>14</td>
<td>Section drag along semispan, comparing parasitic and induced drag for elliptic lift distribution, Prandtl’s 1933 lift distribution, the optimal lift distribution when only considering induced drag, and the optimal lift distribution when regarding total drag.</td>
<td>22</td>
</tr>
<tr>
<td>15</td>
<td>Lift distributions of $B_3$ values corresponding to optimal values, the elliptic, and Prandtl’s 1933</td>
<td>23</td>
</tr>
</tbody>
</table>
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

\( C_\delta \) = shape coefficient for the deflection-limited design Eq. (15)

\( C_\sigma \) = 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

\( \bar{L} \) = local wing section lift

\( \bar{M}_b \) = 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 \( \tilde{W}_s(z) \) and \( \tilde{M}_b(z) \) having units of length squared

\( t_{\text{max}} \) = maximum thickness of the local airfoil section

\( V_\infty \) = freestream airspeed

\( V_{\text{Stall}} \) = 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

\( \tilde{W}_n \) = net weight of the wing per unit span (i.e., total wing weight per unit span less \( \tilde{W}_s \))

\( \tilde{W}_s \) = weight of the wing structure per unit span required to support the wing bending moment distribution

\( z \) = spanwise coordinate relative to the midspan

\( \alpha \) = geometric twist of airfoil section

\( \gamma \) = specific weight of the beam material

\( \delta \) = flap deflection of airfoil section

\( \delta_{\text{max}} \) = maximum wing deflection

\( \theta \) = change of variables for spanwise coordinate, Eq. (1)

\( \kappa_W \) = weight distribution coefficient, Eq. (12)

\( \rho \) = air density

\( \sigma_{\text{max}} \) = 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]

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

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].

![Graph showing the relationship between $B_3$ and $b/b_{eff}$]  

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

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

\[
D_i = \frac{8 (W_b)^2}{3\pi \rho V_\infty^2} \tag{4}
\]

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

\[
W_n = W_r + \int_{z=\frac{b}{2}}^{\frac{b}{2}} \tilde{W}_n(z)dz \tag{5}
\]

\[
\tilde{W}_n = (W - W_r) \frac{\bar{L}(z)}{L} - \tilde{W}_s(z) \tag{6}
\]

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]

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

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

\[
W_r = \frac{n_g - 1}{n_m + n_g} W
\]  

(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]

\[
|\bar{M}_b(z)| = \kappa_W W_r \int_{z'=z}^{b/2} \frac{L(z')}{L} (z' - z) dz', \quad \text{for } z \geq 0
\]  

(9)

where

\[
\kappa_W = \begin{cases} 
  n_m, & W_r \geq \frac{n_g - 1}{n_m + n_g} W \\
  \left(n_g - 1\right) \frac{W_r}{W} - n_g, & W_r < \frac{n_g - 1}{n_m + n_g} W
\end{cases}
\]  

(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]

\[
W_S = 2 \int_{z=0}^{b/2} \frac{|\bar{M}_b(z)|}{S_b(z)} dz ; \quad S_b(z) = C_\sigma \left( \frac{t_{\text{max}}}{c} \right) c(z) \sigma_{\text{max}}, \quad C_\sigma = \frac{2I(h \frac{h}{I_{\text{max}}})}{A h^2}
\]  

(11)

Where \(C_\sigma\) 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]

\[ W_s = \frac{k_W W_r b^2}{32 S_b} (1 + B_3) \] (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

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

\[ B_3 = -\frac{3}{8} + \frac{9}{64} - \frac{1}{12} \] (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
\[ W_s = \frac{\gamma \left( \frac{W}{S} \right)^2 \kappa_w W_r b^6}{32 C_\delta E \left( \frac{t_{\text{max}}}{c} \right)^2 \delta_{\text{max}} W^2} (1 + B_3); \quad C_\delta \equiv \frac{8I \left( \frac{h}{t_{\text{max}}} \right)^2}{Ah^2} \] (15)

\[ B_3 = -\frac{3}{7} + \sqrt{\frac{9}{49} - \frac{1}{21}} \] (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

\[ b_{\text{non}} = b_{\text{ell}} \times \frac{3 \sqrt{S_{\text{non}}}}{S_{\text{ell}}(1 + B_3)} \] (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 \( b_{\text{ell}} \) 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.
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]

\[ F(\delta, \alpha) = f(\alpha)g(\delta) \]  

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]
\[ \tilde{C} = \sum_{m=0}^{M} \sum_{n=0}^{N} a_{nm} \delta^m \alpha^n \]  

(19)

where \( \alpha \) is the array of the polynomial coefficients, \( \tilde{C} \) is one of the aerodynamic coefficients of interest (lift, drag, or moment) and \( \delta \) and \( \alpha \) 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 \( \delta \) and \( \alpha \) 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°.

![Figure 3: XFOIL data for lift, drag, and pitching moment coefficients as a function of \( \alpha \) fitted with polynomial equations.](image-url)

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, \( \delta \), and \( \alpha \) 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
\[
\tilde{C}_L = \frac{b \tilde{L} W}{L} \frac{2}{b \rho V_{\infty}^2 c} = \frac{\tilde{L}}{2 \rho V_{\infty}^2 c}
\]  

(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 \( B_3 =0.0 \).](image-url)
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.

<table>
<thead>
<tr>
<th>$B_3$</th>
<th>Design Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.333</td>
<td>Fixed $W_n$, $\sigma_{max}$, and $V_{stall}$</td>
</tr>
<tr>
<td>-0.177</td>
<td>Fixed $W_n$, $\delta_{max}$, and $V_{stall}$</td>
</tr>
<tr>
<td>-0.136</td>
<td>Fixed $W_n$, $\sigma_{max}$, and $W/S$</td>
</tr>
<tr>
<td>-0.060</td>
<td>Fixed $W_n$, $\delta_{max}$, and $W/S$</td>
</tr>
<tr>
<td>0.000</td>
<td>Fixed $b$ and $W$</td>
</tr>
<tr>
<td>0.050</td>
<td>Only for Study</td>
</tr>
<tr>
<td>0.100</td>
<td>Only for Study</td>
</tr>
</tbody>
</table>

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).
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. 6 Wing planforms 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.

**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.

![Figure 9: Drag values for a variety of root twist and $B_3$ values when using washout.](image)

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.

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 $-4^\circ$ to $16^\circ$ 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.](image)

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 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.
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.
REFERENCES


## APPENDIX A

**POLYNOMIAL FIT COEFFICIENT TABLES FOR NACA 0015 PARABOLIC FLAP**

**AIRFOIL AT $R_e = 500,000$**

<table>
<thead>
<tr>
<th>$\tilde{C}_L$</th>
<th>$\tilde{C}_D$</th>
<th>$\tilde{C}_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\eta$</td>
<td>$m$</td>
<td>$n$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>8.12171E-11</td>
<td>6.338068E+00</td>
<td>1.456780E+00</td>
</tr>
<tr>
<td>9.0869E+01</td>
<td>4.994696E-02</td>
<td>-2.509928E-02</td>
</tr>
<tr>
<td>2.8044E+02</td>
<td>-1.171512E-02</td>
<td>6.317115E-03</td>
</tr>
<tr>
<td>8.12171E-11</td>
<td>6.338068E+00</td>
<td>1.456780E+00</td>
</tr>
<tr>
<td>9.0869E+01</td>
<td>4.994696E-02</td>
<td>-2.509928E-02</td>
</tr>
<tr>
<td>2.8044E+02</td>
<td>-1.171512E-02</td>
<td>6.317115E-03</td>
</tr>
<tr>
<td>3.654763E-11</td>
<td>-7.196154E-06</td>
<td>-1.028566E-03</td>
</tr>
</tbody>
</table>
### Table B1 polynomial fit coefficient for $C_{\tilde{L}}$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$C_{\tilde{L}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6.368163E+00</td>
</tr>
<tr>
<td>1</td>
<td>5.044926E-02</td>
</tr>
<tr>
<td>2</td>
<td>-3.521221E-03</td>
</tr>
<tr>
<td>3</td>
<td>-3.423913E-05</td>
</tr>
<tr>
<td>4</td>
<td>-1.318734E-06</td>
</tr>
<tr>
<td>5</td>
<td>-1.719534E-08</td>
</tr>
<tr>
<td>6</td>
<td>1.087184E-08</td>
</tr>
<tr>
<td>7</td>
<td>6.337234E-11</td>
</tr>
</tbody>
</table>

### Table B2 polynomial fit coefficient for $C_{\tilde{D}}$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$C_{\tilde{D}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.253277E-02</td>
</tr>
<tr>
<td>1</td>
<td>1.128441E-03</td>
</tr>
<tr>
<td>2</td>
<td>1.084579E-05</td>
</tr>
<tr>
<td>3</td>
<td>6.006794E-06</td>
</tr>
<tr>
<td>4</td>
<td>-4.884299E-09</td>
</tr>
<tr>
<td>5</td>
<td>-4.390498E-08</td>
</tr>
<tr>
<td>6</td>
<td>-4.831181E-11</td>
</tr>
<tr>
<td>7</td>
<td>2.105292E-10</td>
</tr>
<tr>
<td>8</td>
<td>3.177439E-13</td>
</tr>
<tr>
<td>9</td>
<td>-2.913599E-13</td>
</tr>
<tr>
<td>10</td>
<td>-4.225152E-16</td>
</tr>
</tbody>
</table>

### Table B3 polynomial fit coefficient for $C_m$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$C_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.419684E+00</td>
</tr>
<tr>
<td>1</td>
<td>3.0532E+00</td>
</tr>
<tr>
<td>2</td>
<td>-1.80635E+01</td>
</tr>
<tr>
<td>3</td>
<td>4.998E+01</td>
</tr>
<tr>
<td>4</td>
<td>-1.251E+02</td>
</tr>
<tr>
<td>5</td>
<td>6.76E+02</td>
</tr>
<tr>
<td>6</td>
<td>-2.90E+03</td>
</tr>
<tr>
<td>7</td>
<td>1.007E+04</td>
</tr>
<tr>
<td>8</td>
<td>-3.01E+04</td>
</tr>
<tr>
<td>9</td>
<td>1.027E+05</td>
</tr>
<tr>
<td>10</td>
<td>-7.78E+05</td>
</tr>
</tbody>
</table>

---

**APPENDIX B**

**POLYNOMIAL FIT COEFFICIENT TABLES FOR NACA 0015 PARABOLIC FLAP**

**AIRFOIL AT $Re = 600,000$**

**Table B1 polynomial fit coefficient for $C_{\tilde{L}}$**

**Table B2 polynomial fit coefficient for $C_{\tilde{D}}$**

**Table B3 polynomial fit coefficient for $C_m$**
## Table C1 polynomial fit coefficient for $C_l$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$C_l$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.132540E+03</td>
</tr>
<tr>
<td>1</td>
<td>1.518668E-01</td>
</tr>
<tr>
<td>2</td>
<td>2.799424E-08</td>
</tr>
<tr>
<td>3</td>
<td>6.404323E+00</td>
</tr>
<tr>
<td>4</td>
<td>-1.121690E+00</td>
</tr>
<tr>
<td>5</td>
<td>4.984544E-07</td>
</tr>
<tr>
<td>6</td>
<td>-2.761745E-05</td>
</tr>
<tr>
<td>7</td>
<td>1.025802E-01</td>
</tr>
<tr>
<td>8</td>
<td>-8.163357E-11</td>
</tr>
<tr>
<td>9</td>
<td>5.250561E-06</td>
</tr>
<tr>
<td>10</td>
<td>-3.393444E-08</td>
</tr>
</tbody>
</table>

## Table C2 polynomial fit coefficient for $C_m$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$C_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.828297E+14</td>
</tr>
<tr>
<td>1</td>
<td>7.329000E+09</td>
</tr>
<tr>
<td>2</td>
<td>2.139016E+04</td>
</tr>
<tr>
<td>3</td>
<td>6.622541E-04</td>
</tr>
<tr>
<td>4</td>
<td>-2.231224E-01</td>
</tr>
<tr>
<td>5</td>
<td>9.854141E-07</td>
</tr>
<tr>
<td>6</td>
<td>-2.243100E-06</td>
</tr>
<tr>
<td>7</td>
<td>7.503641E-09</td>
</tr>
<tr>
<td>8</td>
<td>-7.869211E-07</td>
</tr>
<tr>
<td>9</td>
<td>-2.137066E-07</td>
</tr>
<tr>
<td>10</td>
<td>1.675438E-05</td>
</tr>
</tbody>
</table>

## Table C3 polynomial fit coefficient for $C_n$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$C_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.132540E+03</td>
</tr>
<tr>
<td>1</td>
<td>1.518668E-01</td>
</tr>
<tr>
<td>2</td>
<td>2.799424E-08</td>
</tr>
<tr>
<td>3</td>
<td>6.404323E+00</td>
</tr>
<tr>
<td>4</td>
<td>-1.121690E+00</td>
</tr>
<tr>
<td>5</td>
<td>4.984544E-07</td>
</tr>
<tr>
<td>6</td>
<td>-2.761745E-05</td>
</tr>
<tr>
<td>7</td>
<td>1.025802E-01</td>
</tr>
<tr>
<td>8</td>
<td>-8.163357E-11</td>
</tr>
<tr>
<td>9</td>
<td>5.250561E-06</td>
</tr>
<tr>
<td>10</td>
<td>-3.393444E-08</td>
</tr>
</tbody>
</table>

### Table S polynomial fit coefficient for $S_{0015}$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$S_{0015}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.132540E+03</td>
</tr>
<tr>
<td>1</td>
<td>1.518668E-01</td>
</tr>
<tr>
<td>2</td>
<td>2.799424E-08</td>
</tr>
<tr>
<td>3</td>
<td>6.404323E+00</td>
</tr>
<tr>
<td>4</td>
<td>-1.121690E+00</td>
</tr>
<tr>
<td>5</td>
<td>4.984544E-07</td>
</tr>
<tr>
<td>6</td>
<td>-2.761745E-05</td>
</tr>
<tr>
<td>7</td>
<td>1.025802E-01</td>
</tr>
<tr>
<td>8</td>
<td>-8.163357E-11</td>
</tr>
<tr>
<td>9</td>
<td>5.250561E-06</td>
</tr>
<tr>
<td>10</td>
<td>-3.393444E-08</td>
</tr>
</tbody>
</table>
## APPENDIX D

### POLYNOMIAL FIT COEFFICIENTS FOR NAÇA 0015 PARABOLIC FLAP

#### Table D1 polynomial fit coefficient for $C_L$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$C_L$ coefficient</th>
<th>$C_L$ coefficient</th>
<th>$C_L$ coefficient</th>
<th>$C_L$ coefficient</th>
<th>$C_L$ coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$2.042465\times10^{-2}$</td>
<td>$-1.173327\times10^{0}$</td>
<td>$1.102441\times10^{2}$</td>
<td>$-1.000337\times10^{3}$</td>
<td>$0$</td>
</tr>
<tr>
<td>1</td>
<td>$-1.191649\times10^{-2}$</td>
<td>$-2.041545\times10^{-2}$</td>
<td>$3.188106\times10^{0}$</td>
<td>$-3.582896\times10^{1}$</td>
<td>$0$</td>
</tr>
<tr>
<td>2</td>
<td>$-1.033760\times10^{-4}$</td>
<td>$1.025705\times10^{0}$</td>
<td>$-3.284561\times10^{0}$</td>
<td>$2.700688\times10^{1}$</td>
<td>$0$</td>
</tr>
<tr>
<td>3</td>
<td>$6.663793\times10^{-6}$</td>
<td>$1.398280\times10^{-3}$</td>
<td>$-6.331199\times10^{-2}$</td>
<td>$6.457723\times10^{-1}$</td>
<td>$0$</td>
</tr>
<tr>
<td>4</td>
<td>$1.006808\times10^{-5}$</td>
<td>$-1.199010\times10^{-3}$</td>
<td>$3.299182\times10^{-2}$</td>
<td>$-2.617871\times10^{-1}$</td>
<td>$0$</td>
</tr>
<tr>
<td>5</td>
<td>$2.348069\times10^{-8}$</td>
<td>$-9.242349\times10^{-6}$</td>
<td>$3.621814\times10^{-4}$</td>
<td>$-3.514869\times10^{-3}$</td>
<td>$0$</td>
</tr>
<tr>
<td>6</td>
<td>$-4.054338\times10^{-8}$</td>
<td>$4.133184\times10^{-6}$</td>
<td>$-1.068401\times10^{-4}$</td>
<td>$8.016431\times10^{-4}$</td>
<td>$0$</td>
</tr>
<tr>
<td>7</td>
<td>$-8.865049\times10^{-11}$</td>
<td>$2.389084\times10^{-8}$</td>
<td>$-8.952933\times10^{-7}$</td>
<td>$8.648075\times10^{-6}$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

#### Table D2 polynomial fit coefficient for $C_D$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$C_D$ coefficient</th>
<th>$C_D$ coefficient</th>
<th>$C_D$ coefficient</th>
<th>$C_D$ coefficient</th>
<th>$C_D$ coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$6.429100\times10^{0}$</td>
<td>$-2.067211\times10^{0}$</td>
<td>$-3.615552\times10^{2}$</td>
<td>$1.802343\times10^{3}$</td>
<td>$0$</td>
</tr>
<tr>
<td>1</td>
<td>$5.110754\times10^{-2}$</td>
<td>$6.867178\times10^{-2}$</td>
<td>$-1.659522\times10^{1}$</td>
<td>$1.720596\times10^{2}$</td>
<td>$0$</td>
</tr>
<tr>
<td>2</td>
<td>$-3.427826\times10^{-3}$</td>
<td>$-1.329033\times10^{-1}$</td>
<td>$5.091650\times10^{0}$</td>
<td>$-4.403098\times10^{1}$</td>
<td>$0$</td>
</tr>
<tr>
<td>3</td>
<td>$-3.114671\times10^{-5}$</td>
<td>$-5.059913\times10^{-3}$</td>
<td>$2.433353\times10^{-1}$</td>
<td>$-2.536968\times10^{0}$</td>
<td>$0$</td>
</tr>
<tr>
<td>4</td>
<td>$-2.349070\times10^{-6}$</td>
<td>$6.943055\times10^{-4}$</td>
<td>$-1.780278\times10^{-2}$</td>
<td>$1.406877\times10^{-1}$</td>
<td>$0$</td>
</tr>
<tr>
<td>5</td>
<td>$-2.624205\times10^{-8}$</td>
<td>$2.337709\times10^{-5}$</td>
<td>$-1.056317\times10^{-3}$</td>
<td>$1.129155\times10^{-2}$</td>
<td>$0$</td>
</tr>
<tr>
<td>6</td>
<td>$1.326887\times10^{-8}$</td>
<td>$-1.418686\times10^{-6}$</td>
<td>$3.524776\times10^{-5}$</td>
<td>$-3.021687\times10^{-4}$</td>
<td>$0$</td>
</tr>
<tr>
<td>7</td>
<td>$6.941618\times10^{-11}$</td>
<td>$-2.911089\times10^{-8}$</td>
<td>$1.270101\times10^{-6}$</td>
<td>$-1.344652\times10^{-5}$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

#### Table D3 polynomial fit coefficient for $C_m$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$C_m$ coefficient</th>
<th>$C_m$ coefficient</th>
<th>$C_m$ coefficient</th>
<th>$C_m$ coefficient</th>
<th>$C_m$ coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$1.178579\times10^{-2}$</td>
<td>$1.281148\times10^{0}$</td>
<td>$-7.259545\times10^{0}$</td>
<td>$5.736868\times10^{2}$</td>
<td>$-1.250251\times10^{4}$</td>
</tr>
<tr>
<td>1</td>
<td>$1.131753\times10^{-3}$</td>
<td>$1.358528\times10^{-1}$</td>
<td>$-1.501205\times10^{1}$</td>
<td>$4.378456\times10^{2}$</td>
<td>$-3.610753\times10^{3}$</td>
</tr>
<tr>
<td>2</td>
<td>$1.034643\times10^{-5}$</td>
<td>$-1.156612\times10^{-3}$</td>
<td>$2.938629\times10^{-1}$</td>
<td>$-1.960194\times10^{1}$</td>
<td>$4.748448\times10^{2}$</td>
</tr>
<tr>
<td>3</td>
<td>$4.930500\times10^{-6}$</td>
<td>$-2.580213\times10^{-3}$</td>
<td>$2.332350\times10^{-1}$</td>
<td>$-6.469623\times10^{0}$</td>
<td>$5.398787\times10^{1}$</td>
</tr>
<tr>
<td>4</td>
<td>$-4.641892\times10^{-9}$</td>
<td>$3.567171\times10^{-6}$</td>
<td>$-1.806742\times10^{-3}$</td>
<td>$1.705203\times10^{-1}$</td>
<td>$-4.789121\times10^{0}$</td>
</tr>
<tr>
<td>5</td>
<td>$-3.768289\times10^{-8}$</td>
<td>$2.024193\times10^{-5}$</td>
<td>$-1.511347\times10^{-3}$</td>
<td>$3.945232\times10^{-2}$</td>
<td>$-3.235768\times10^{-1}$</td>
</tr>
<tr>
<td>6</td>
<td>$-5.320576\times10^{-11}$</td>
<td>$4.229378\times10^{-8}$</td>
<td>$4.880667\times10^{-6}$</td>
<td>$-7.445426\times10^{-4}$</td>
<td>$2.324667\times10^{-2}$</td>
</tr>
<tr>
<td>7</td>
<td>$1.791606\times10^{-10}$</td>
<td>$-6.692652\times10^{-8}$</td>
<td>$4.437531\times10^{-6}$</td>
<td>$-1.089284\times10^{-4}$</td>
<td>$8.578894\times10^{-4}$</td>
</tr>
<tr>
<td>8</td>
<td>$3.142871\times10^{-13}$</td>
<td>$-1.758744\times10^{-10}$</td>
<td>$-9.151523\times10^{-9}$</td>
<td>$1.790551\times10^{-6}$</td>
<td>$-5.757638\times10^{-5}$</td>
</tr>
<tr>
<td>9</td>
<td>$-2.408628\times10^{-13}$</td>
<td>$7.370114\times10^{-11}$</td>
<td>$-4.561650\times10^{-9}$</td>
<td>$1.079232\times10^{-7}$</td>
<td>$-8.290140\times10^{-7}$</td>
</tr>
<tr>
<td>10</td>
<td>$-3.986105\times10^{-16}$</td>
<td>$1.812268\times10^{-13}$</td>
<td>$9.293685\times10^{-12}$</td>
<td>$-1.710235\times10^{-9}$</td>
<td>$5.435685\times10^{-8}$</td>
</tr>
</tbody>
</table>
Table E1 polynomial fit coefficient for $C_m$

<table>
<thead>
<tr>
<th>$u$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.115034</td>
<td>0.121664</td>
<td>-0.699676</td>
<td>5.656041</td>
<td>-1.252675</td>
<td>9.195068</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.001175</td>
<td>0.010982</td>
<td>-0.128302</td>
<td>3.828373</td>
<td>-0.321925</td>
<td>2.499922</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.000101</td>
<td>-0.011376</td>
<td>0.029771</td>
<td>-0.203441</td>
<td>4.999227</td>
<td>-3.844865</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.000034</td>
<td>-0.001936</td>
<td>0.001758</td>
<td>-0.048861</td>
<td>0.418401</td>
<td>-0.250823</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.000004</td>
<td>0.000510</td>
<td>-0.000877</td>
<td>0.004494</td>
<td>-0.044938</td>
<td>0.004395</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.000003</td>
<td>-0.000233</td>
<td>0.000025</td>
<td>-0.000255</td>
<td>0.000254</td>
<td>-0.000254</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.000001</td>
<td>0.000002</td>
<td>-0.000001</td>
<td>0.000000</td>
<td>-0.000000</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table E2 polynomial fit coefficient for $C_L$

<table>
<thead>
<tr>
<th>$u$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.115034</td>
<td>0.121664</td>
<td>-0.699676</td>
<td>5.656041</td>
<td>-1.252675</td>
<td>9.195068</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.001175</td>
<td>0.010982</td>
<td>-0.128302</td>
<td>3.828373</td>
<td>-0.321925</td>
<td>2.499922</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.000101</td>
<td>-0.011376</td>
<td>0.029771</td>
<td>-0.203441</td>
<td>4.999227</td>
<td>-3.844865</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.000034</td>
<td>-0.001936</td>
<td>0.001758</td>
<td>-0.048861</td>
<td>0.418401</td>
<td>-0.250823</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.000004</td>
<td>0.000510</td>
<td>-0.000877</td>
<td>0.004494</td>
<td>-0.044938</td>
<td>0.004395</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.000003</td>
<td>-0.000233</td>
<td>0.000025</td>
<td>-0.000255</td>
<td>0.000254</td>
<td>-0.000254</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.000001</td>
<td>0.000002</td>
<td>-0.000001</td>
<td>0.000000</td>
<td>-0.000000</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table E3 polynomial fit coefficient for $C_D$

<table>
<thead>
<tr>
<th>$u$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.115034</td>
<td>0.121664</td>
<td>-0.699676</td>
<td>5.656041</td>
<td>-1.252675</td>
<td>9.195068</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.001175</td>
<td>0.010982</td>
<td>-0.128302</td>
<td>3.828373</td>
<td>-0.321925</td>
<td>2.499922</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.000101</td>
<td>-0.011376</td>
<td>0.029771</td>
<td>-0.203441</td>
<td>4.999227</td>
<td>-3.844865</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.000034</td>
<td>-0.001936</td>
<td>0.001758</td>
<td>-0.048861</td>
<td>0.418401</td>
<td>-0.250823</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.000004</td>
<td>0.000510</td>
<td>-0.000877</td>
<td>0.004494</td>
<td>-0.044938</td>
<td>0.004395</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.000003</td>
<td>-0.000233</td>
<td>0.000025</td>
<td>-0.000255</td>
<td>0.000254</td>
<td>-0.000254</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.000001</td>
<td>0.000002</td>
<td>-0.000001</td>
<td>0.000000</td>
<td>-0.000000</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

APPENDIX E

POLYNOMIAL FIT COEFFICIENT TABLES FOR NACA 0015 PARABOLIC FLAP

AIRFOIL AT $R_e = 900,000$
### APPENDIX F

**POLYNOMIAL FIT COEFFICIENT TABLES FOR NACA 0015 PARABOLIC FLAP**

**AIRFOIL AT \( Re = 1,000,000 \)**

**Table F1 polynomial fit coefficient for \( \tilde{C}_L \)**

<table>
<thead>
<tr>
<th>( m )</th>
<th>( n )</th>
<th>( \tilde{C}_L )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1.1250E+01</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-2.2171E-01</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1.7244E+01</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>-6.3860E+00</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>5.3831E+02</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>-1.2133E+04</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>8.9967E+04</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>-1.1370E+07</td>
</tr>
</tbody>
</table>

**Table F2 polynomial fit coefficient for \( \tilde{C}_D \)**

<table>
<thead>
<tr>
<th>( m )</th>
<th>( n )</th>
<th>( \tilde{C}_D )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1.1250E+01</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-2.2171E-01</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1.7244E+01</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>-6.3860E+00</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>5.3831E+02</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>-1.2133E+04</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>8.9967E+04</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>-1.1370E+07</td>
</tr>
</tbody>
</table>

**Table F3 polynomial fit coefficient for \( \tilde{C}_m \)**

<table>
<thead>
<tr>
<th>( m )</th>
<th>( n )</th>
<th>( \tilde{C}_m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1.1250E+01</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-2.2171E-01</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1.7244E+01</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>-6.3860E+00</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>5.3831E+02</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>-1.2133E+04</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>8.9967E+04</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>-1.1370E+07</td>
</tr>
</tbody>
</table>

---

\( C \) Ń\( m \) Ń

\( n \) 0 1 2 3 4 5 6 7 8 9 10

\( C \tilde{L} \) Ń\( n \) 0 1 2 3 4 5 6 7

\( C \tilde{D} \) Ń\( n \) 0 1 2 3 4 5 6 7 8 9 10

\( C \tilde{m} \) Ń\( n \) 0 1 2 3 4 5 6 7 8 9 10
<table>
<thead>
<tr>
<th>$m$</th>
<th>$c_l$</th>
<th>$c_m$</th>
<th>$c_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9.028071E-03</td>
<td>-4.167995E-01</td>
<td>7.057869E+01</td>
</tr>
<tr>
<td>1</td>
<td>-1.202338E-02</td>
<td>-4.112445E-02</td>
<td>3.972410E+00</td>
</tr>
<tr>
<td>2</td>
<td>-4.213308E-05</td>
<td>6.937922E-02</td>
<td>-1.688399E+00</td>
</tr>
<tr>
<td>3</td>
<td>4.997309E-06</td>
<td>1.816146E-03</td>
<td>-8.100096E-02</td>
</tr>
<tr>
<td>4</td>
<td>8.220185E-06</td>
<td>-7.370752E-04</td>
<td>1.334300E-02</td>
</tr>
<tr>
<td>5</td>
<td>3.669069E-08</td>
<td>-1.303496E-05</td>
<td>5.429770E-04</td>
</tr>
<tr>
<td>6</td>
<td>-3.118715E-08</td>
<td>2.273525E-06</td>
<td>-3.280896E-05</td>
</tr>
<tr>
<td>7</td>
<td>-1.300261E-10</td>
<td>3.701569E-08</td>
<td>-1.542476E-06</td>
</tr>
<tr>
<td>8</td>
<td>3.417796E-11</td>
<td>-2.287491E-09</td>
<td>2.797351E-08</td>
</tr>
<tr>
<td>9</td>
<td>1.341386E-13</td>
<td>-3.649984E-11</td>
<td>1.531599E-09</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$m$</th>
<th>$c_l$</th>
<th>$c_m$</th>
<th>$c_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6.486454E+00</td>
<td>-4.387091E+00</td>
<td>-2.351970E+02</td>
</tr>
<tr>
<td>1</td>
<td>5.186308E-02</td>
<td>9.288474E-02</td>
<td>-1.631773E+01</td>
</tr>
<tr>
<td>2</td>
<td>-3.296270E-03</td>
<td>-5.796230E-02</td>
<td>1.234340E+00</td>
</tr>
<tr>
<td>3</td>
<td>-2.885558E-05</td>
<td>-5.021286E-03</td>
<td>2.272644E-01</td>
</tr>
<tr>
<td>4</td>
<td>-4.419147E-06</td>
<td>5.471211E-04</td>
<td>-5.767531E-03</td>
</tr>
<tr>
<td>5</td>
<td>-2.972230E-08</td>
<td>2.096663E-05</td>
<td>-8.749127E-04</td>
</tr>
<tr>
<td>6</td>
<td>1.755576E-08</td>
<td>-1.442715E-06</td>
<td>2.553992E-05</td>
</tr>
<tr>
<td>7</td>
<td>6.717569E-11</td>
<td>-2.391145E-08</td>
<td>9.320457E-07</td>
</tr>
<tr>
<td>8</td>
<td>1.105130E-02</td>
<td>1.062088E-01</td>
<td>-5.703164E+00</td>
</tr>
<tr>
<td>9</td>
<td>1.274124E-03</td>
<td>5.532465E-02</td>
<td>-8.268602E+00</td>
</tr>
<tr>
<td>10</td>
<td>9.454678E-06</td>
<td>-8.241372E-04</td>
<td>2.482180E-01</td>
</tr>
<tr>
<td>11</td>
<td>-1.587689E-08</td>
<td>-5.147949E-04</td>
<td>4.821709E-02</td>
</tr>
<tr>
<td>12</td>
<td>-1.247799E-09</td>
<td>6.933233E-07</td>
<td>-1.293295E-03</td>
</tr>
<tr>
<td>13</td>
<td>-7.721216E-10</td>
<td>2.746858E-06</td>
<td>8.536535E-05</td>
</tr>
<tr>
<td>14</td>
<td>-6.135391E-11</td>
<td>4.462748E-08</td>
<td>2.322670E-06</td>
</tr>
<tr>
<td>15</td>
<td>-5.444505E-11</td>
<td>-8.340151E-09</td>
<td>-8.814394E-07</td>
</tr>
<tr>
<td>16</td>
<td>-2.869673E-13</td>
<td>-1.509355E-10</td>
<td>-3.220560E-09</td>
</tr>
<tr>
<td>17</td>
<td>-9.464770E-14</td>
<td>7.695550E-12</td>
<td>-8.955091E-06</td>
</tr>
</tbody>
</table>
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? 
Density in kg/m^3 (Standard sea level= 1.229)'))

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

...
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,'

if lift_case == 'Other':
    print('B3 Value:',B3,'
print('Variable Case:',variable_case,'
print('Reynolds number chosen B3:',Re_opt,'
print('Optimized Aspect ratio:', b_opt/c_opt,'
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?

Aerodynamic twist value in percent (-20 to 20)\n'))

root_twist=float(input('What is the geometric twist at the root of your wing?

Geometric twist value in deg (-15 to 15)\n'))

elif variable_case == 'Aero':
    camber_val=0.0
    washout_val=float(input('What is constant geometric twist value for the wing?

Geometric twist value in deg (-15 to 15)\n'))

    root_twist=float(input('What is the aerodynamic twist at the root of your wing?

Aerodynamic twist value in percent (-20 to 20)\n'))

elif variable_case == 'Both':
    camber_val=0.0
    washout_val=0.0

else:
    print('Error: please provide appropriate answer.\n')
    [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.\n')
        [x_length]=get_control_point_length()
def get_B3_value():
    lift_case=input('What distribution case are you trying to match?
Elliptic = 1
Fixed lift dist, fixed net weight, fixed max stress, fixed stall speed or Prandtl's 1933 = 2
Fixed lift dist, fixed gross weight, fixed max stress, fixed wing loading = 3
Fixed lift dist, fixed gross weight, fixed max deflection, fixed wing loading = 4
Fixed lift dist, fixed net weight, fixed max deflection, fixed stall speed = 5
User Specified B3 (enter Other)\n')
    if lift_case == 'Other':
        B3=float(input('What is the B3 value you would like to use?
Typical 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}

eelif lift_case==3:
    B3=-0.13564322

# Lift Distribution Case {fixed lift dist, fixed gross weight, fixed max deflection, fixed wing loading}

eelif lift_case==4:
    B3=-0.05971587

# Lift Distribution Case {fixed lift dist, fixed net weight, fixed max deflection, fixed stall speed}

eelif lift_case==5:
    B3=-0.17714856
else:
    print('Error: Please provide appropriate answer.\n')

[B3,lift_case]=get_B3_value()
return(B3,lift_case)
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(\n    \n    \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],'

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)
#
# ==============================================================
#             # 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)
#
# ==============================================================
#             # 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:
    # Read data from output file

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('MachUp Lift Distribution')

for i in range (0,z_size):
    print(CL_dist_Machup[i])

print('Analytic Lift Distribution')

#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

\[
CL_{\text{dist\_analytic}} = \frac{\text{Opt\_case.weight}}{\text{Opt\_case.b\_opt}} \times \frac{1}{2.0} \times \frac{\text{Opt\_case.rho}}{\text{Opt\_case.v}^2} \times \frac{\text{Opt\_case.c\_opt}}{(4.0/\pi) \times (\sin(\theta) + \text{Opt\_case.B\_3} \times \sin(3.0 \times \theta))}
\]

print(CL_dist_analytic)

print('nZ/b Distributionn')

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
import os
import json
import shutil
import math
from numpy import zeros
import Reynolds_Interpolation as RI
import uuid

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)

#

# 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_Mach[up[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
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_file_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_file_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_file_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_file_high,Curve_fit_data):
for i in range (0, alpha_range):
    c_alpha='C'+str(i)
    Re_specific['Re_specific'][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_file_low))][variable][c_alpha][c_camber ],Curve_fit_data[str(int(Re_file_high))][variable][c_alpha][c_camber])
            Re_specific['Re_specific'][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_file_low))][variable][c_alpha][c_camber ],Curve_fit_data[str(int(Re_file_high))][variable][c_alpha][c_camber])
            Re_specific['Re_specific'][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_file_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_file_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_file_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