8-2022

Multidisciplinary Reference Solutions for Performance-Optimized Aircraft Wings with Tailored Aerodynamic Load Distributions

Jeffrey D. Taylor
Utah State University

MULTIDISCIPLINARY REFERENCE SOLUTIONS FOR PERFORMANCE-OPTIMIZED AIRCRAFT WINGS WITH TAILORED AERODYNAMIC LOAD DISTRIBUTIONS

by

Jeffrey D. Taylor

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

of

DOCTOR OF PHILOSOPHY

in

Mechanical Engineering

Approved:

_____	_____
Douglas F. Hunsaker, Ph.D.	Thomas H. Fronk, Ph.D.
Major Professor	Committee Member
_____	_____
Stephen A. Whitmore, Ph.D.	Matthew W. Harris, Ph.D.
Committee Member	Committee Member
_____	_____
Nhan T. Nguyen, Ph.D.	D. Richard Cutler, Ph.D.
Committee Member	Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2022

ABSTRACT

Multidisciplinary Reference Solutions for Performance-Optimized Aircraft Wings with Tailored Aerodynamic Load Distributions

by

Jeffrey D. Taylor

Utah State University, 2022

Major Professor: Douglas F. Hunsaker, Ph.D.
Department: Mechanical and Aerospace Engineering

The optimization of wings that employ static or active wing shaping to tailor the load distribution is a multidisciplinary task involving the coupling of various operational and wing-design parameters. Computational methods for multidisciplinary design and optimization are common, but obtaining relational understanding about the coupling between operational and design parameters from these methods is often very difficult. In this dissertation, analytic and low-order multidisciplinary methods are presented that capture the coupling between aerodynamics, structures, control, and the flight path trajectory in the design and optimization of wings with wing-shaping controls. These methods are used to obtain reference solutions that reveal important relational information and provide insights to inform future research in active wing shaping. Results are presented both for wings with static wing shaping and wings using active wing shaping that characterize the impact of wing shaping on aircraft efficiency and performance.

(513 pages)

PUBLIC ABSTRACT

Multidisciplinary Reference Solutions for Performance-Optimized Aircraft Wings with Tailored Aerodynamic Load Distributions

Jeffrey D. Taylor

Morphing wings, or wings that can change shape during flight, have the potential to substantially reduce the amount of fuel consumed by an aircraft over the course of its flight. However, the extent to which these wings can reduce fuel consumption depends on the design of the wing, including its aerodynamic efficiency and its structural layout, and how the aircraft flies, including its flight altitude and speed. Correctly predicting how these design and operational characteristics interact is critical to predicting how wing morphing may affect aircraft fuel consumption. Many computer prediction tools exist that include the effects of these interactions, but extracting the information needed to understand how the interactions work from most of these tools is very difficult. In this dissertation, some simplified models are presented that more directly reveal key information about the interplay between aerodynamics, structures, control, and the flight trajectory in the design of morphing wings. This information is used to characterize the impacts of wing morphing on aircraft efficiency.

ACKNOWLEDGMENTS

This dissertation would not have been possible without the support of many individuals. First, I thank Douglas Hunsker, my advisor, mentor, and friend, for his confidence and unfailing support throughout my graduate studies. He has gone above and beyond in supporting my professional, academic, and personal development. I thank Nhan Nguyen, whose support as my NASA fellowship advisor made this research possible. I also thank the many individuals who were willing to listen, offer advice, review my work, or support this work in other ways. These include Justice Schoenfeld, Christian Bolander, Zachary Montgomery, Jackson Reid, Warren Phillips, Matthew Harris, Steve Parris, Greg Reich, David Lazzara, and many others.

Most of all, I thank my wonderful wife, Nicole, who has encouraged me all along the way. Of all those whose efforts made this dissertation possible, she deserves the most credit. I thank her for her patience, confidence, and love through all the long days, late nights, and early mornings. I thank her for listening and caring as we shared the setbacks, frustrations, and successes along the way. Truly, this is as much her accomplishment as it is mine.

Jeffrey D. Taylor

CONTENTS

LIST OF TABLES

Page

MINIMUM INDUCED DRAG FOR TAPERED WINGS INCLUDING STRUCTURAL CONSTRAINTS

LOW-FIDELITY METHOD FOR RAPID AEROSTRUCTURAL OPTIMIZATION AND DESIGN-SPACE EXPLORATION OF PLANAR WINGS

COMPARISON OF THEORETICAL AND MULTI-FIDELITY OPTIMUM AEROSTRUCTURAL SOLUTIONS FOR WING DESIGN

EFFECTS OF WING MORPHING ON AIRCRAFT FUEL BURN ALONG FUEL-OPTIMAL TRAJECTORIES

Characterization of the Common Research Model Wing for Low-Fidelity Aerostructural Analysis

LIST OF FIGURES

Page

MINIMUM INDUCED DRAG FOR TAPERED WINGS INCLUDING STRUCTURAL CONSTRAINTS

LOW-FIDELITY METHOD FOR RAPID AEROSTRUCTURAL OPTIMIZATION AND DESIGN-SPACE EXPLORATION OF PLANAR WINGS

COMPARISON OF THEORETICAL AND MULTI-FIDELITY OPTIMUM AEROSTRUCTURAL SOLUTIONS FOR WING DESIGN

Simplified Trajectory Optimization Formulations using Optimal Control Theory and the Calculus of Variations

CHAPTER 1

INTRODUCTION

THE design and optimization of wings that use wing-shaping controls to tailor the load distribution requires a multidisciplinary approach that captures the coupling between various operational and design parameters. Computational methods for the multidisciplinary design and optimization (MDO) of aircraft wings can often accurately model this coupling for specific discrete design scenarios, but the field is lacking in theoretical reference solutions that advance our understanding of the general relationships between coupled operational and design parameters. Much of our relational understanding between wing design parameters and aerodynamic performance is based on solutions obtained from theory. Designers often rely on insights gained from aerodynamic theories in the conceptual and preliminary phases of aircraft design. In many cases, solutions based on theory have been shown to be in good agreement with experimental data and computational fluid dynamics [1-8], while providing substantially more mathematical and physical insight than most computational models.

This dissertation is focused on the development and use of analytical and low-order methods to obtain theoretical reference solutions that provide relational understanding between various coupled operational and design parameters in the optimization of wings that utilize wing shaping to tailor the load distribution. The majority of modern MDO methods link individual computational solvers to model the interaction between various coupled disciplines. Research in this area is trending toward higher-fidelity computational models that may require heavy computation but can capture intricate details and yield highly accurate results. In this dissertation, these methods will generally

be referred to as high-fidelity computational methods. Whereas high-fidelity computational MDO methods can be very valuable for complex design and analysis problems, their discrete nature often makes extracting general relational information from them very difficult. Moreover, because of the high computational cost of many high-fidelity computational MDO methods, it is often very expensive and time-consuming to use them for design-space exploration or optimization studies involving a large number of function calls. Therefore, when studying highly-coupled design spaces, it is sometimes useful to use analytic methods, which yield equations that show general relationships between operational and design parameters for a wide range of aircraft configurations, or efficient lower fidelity computational methods, which can be used to perform rapid design-space exploration and optimization. This dissertation presents several analytic and low-order MDO methods and shows how they can be used to obtain reference solutions that provide relational understanding to inform ongoing research in shape-adaptive aircraft wing structures.

The methods and solutions presented in this dissertation will also complement ongoing and future computational MDO research by providing insights for conceptual-level multidisciplinary design and innovation and by providing simple, theoretical validation cases for high-fidelity computational methods. One of the main challenges in MDO is to correctly link computational aerodynamic, structural, and other analysis tools to achieve accurate coupling. This process often requires substantial effort and introduces many opportunities for error. Therefore, it is valuable to have a simple, known solution that includes interdisciplinary coupling as a validation case to ensure that coupling between all individual computational components is properly implemented. Once a high-

fidelity computational solution is obtained, theoretical solutions can also serve as a point of reference for interpreting computational results.

Although wing design involves a large number of disciplines, this dissertation is primarily focused on aspects of the coupling between aerodynamics, structures, and control, since each of these areas is highly related to wing-shaping control and affects the wing load distribution and the wing performance.

1.1 Static Wing Shaping

Aircraft wings are typically designed with a fixed jig twist that is tailored to produce optimal aerodynamic characteristics at a desired design condition. When the wing operates away from this design condition, its performance is suboptimal. The elliptic lift distribution is traditionally considered to be the lift distribution that produces optimum aerodynamic efficiency for a wing in cruise. The elliptic lift distribution was first identified by Ludwig Prandtl in 1918 [9,10] from lifting-line theory as the lift distribution that minimizes induced drag for a wing with fixed weight and wingspan. Under these constraints, the elliptic lift distribution has since been studied extensively using analytic, computational, and experimental methods, and it has been shown to be optimal for many complex and unconventional wing designs in both high- and low-speed subsonic flight.

When aerostructural constraints are considered, however, the optimum lift distribution is typically non-elliptic. In 1933, Prandtl [11] showed that under aerostructural constraints, the optimum lift distribution for minimum induced drag depends on the coupling between the weight, wingspan, and lift distribution. The primary reasons for this can be conveniently illustrated using classical lifting-line theory [9,10]. Using the Fourier-series solution to classical lifting line theory, the lift distribution on an

unswept wing immersed in an incompressible, inviscid flow with freestream density ρ and freestream velocity V_∞ can be written as

$$\frac{b\widetilde{L}(\theta)}{L} = \frac{4}{\pi}\left[\sin(\theta) + \sum_{n=2}^{\infty} B_n \sin(n\theta)\right]; \qquad B_n \equiv \frac{A_n}{A_1}, \qquad \theta \equiv \cos^{-1}(-2z/b) \tag{1.1}$$

where L is the total lift, $\widetilde{L}(\theta)$ is the section lift, b is the wingspan, and B_n are normalized Fourier coefficients. The induced drag on a wing in steady-level flight can be written as

$$D_i = \frac{2(W/b)^2}{\pi\rho V_\infty^2}\left(1 + \sum_{n=2}^{\infty} nB_n^2\right) \tag{1.2}$$

where W is the weight.

When weight and wingspan are fixed, Eq. (1.2) is minimized with the elliptic lift distribution, which has $B_n = 0$ for all n. If the weight W and/or wingspan b are allowed to vary, Eq. (1.2) can be reduced by decreasing weight and/or increasing wingspan. However, this cannot be done arbitrarily because the weight of the wing-structure depends, in large part, on the bending moments. Any change in the lift distribution, wingspan, or weight distribution affects the wing bending moments and the corresponding wing-structure weight required to support the bending moments. When these structural effects are considered, the elliptic lift distribution does not provide an absolute minimum in induced drag. Instead, it is advantageous to shift lift inboard to alleviate the wing bending moments, which can serve to reduce the necessary wing-structure weight. Thus, truly minimizing induced drag involves a tradeoff between the lift distribution, wingspan, and wing-structure weight.

Because of this tradeoff, minimizing drag on a wing with a fixed wing-twist distribution or a fixed lift distribution involves a compromise between designs with high load-alleviation at the load limit and low drag during cruise. This often results in a non-elliptic lift distribution that is an aerostructural analogue to the aerodynamically-optimum elliptic lift distribution. The load alleviation provided by such a lift distribution can allow for a larger wingspan than would be structurally feasible with a fixed elliptic lift distribution with the same wing-structure weight. This increase in wingspan outweighs the drag penalty incurred by operating with a non-elliptic lift distribution, resulting in an overall reduction in induced drag over the flight envelope. The extent of this drag reduction depends on the design constraints.

In 1933, Prandtl [11] identified a bell-shaped lift distribution that minimizes induced drag on a rectangular wing with fixed gross weight. Prandtl's study [11] included constraints on the gross lift and the moment of inertia of gross lift, which is derived from the integrated bending moment. To obtain an analytic solution, Prandtl [11] assumed that the wing bending moments are solely due to the lift distribution, regardless of the weight of the wing. Prandtl also assumed that the wing bending moments $\widetilde{M}_b$ are related to the wing-structure weight W_s by a spanwise-invariant proportionality coefficient S_b, i.e.,

$$W_s = \int_0^{b/2} \frac{\widetilde{M}_b(y)}{S_b}\, dy \tag{1.3}$$

where b is the wingspan and y is the spanwise coordinate. This assumption corresponds to rectangular wings. Within the framework of these constraints and assumptions, Prandtl identified a bell-shaped lift distribution that allows a 22.5% larger wingspan and produces 11.1% less induced drag than the elliptic lift distribution with the same wing

weight [11]. Prandtl concluded that because the bell-shaped lift distribution corresponds more closely to the lift distribution of an untwisted tapered wing than the elliptic lift distribution, "[pointed-end] wings have an advantage over those with a nearly rectangular profile" when aerostructural effects are considered [11,12]. However, because Prandtl's assumptions and approximations correspond most closely with rectangular wings, his conclusions were not fully assessed.

The subsequent literature includes several additional theoretical solutions for minimizing drag using wingspan, lift distribution, and wing weight. Jones [13] sought to minimize induced drag under the constraints of fixed gross lift and root bending moment in cruise. Gopalarathnam and Norris [14], Verstraetan and Slingerland [15], and Ranjan [16] also sought to minimize total drag with root-bending-moment constraints. Pate and German [17] constrained the root bending moment at a given off-design lift coefficient, but did not allow the wingspan to change. DeYoung [18] replaced Jones' root-bending-moment constraint with a constraint on the bending moment at a prescribed spanwise location. Jones and Lasinski [19] later constrained the integrated bending moment. Klein and Viswanathan [20,21] considered both root and integrated bending moment [20] and included the effects of shear on the wing-structure weight [21]. Löbert [22] and McGeer [23] introduced a constraint based on the ratio of the bending-moment distribution and the wing-section thickness.

More recently, Phillips et al. [24,25] extended Prandtl's approach to account for the effects of the wing weight distribution, with the bending moments evaluated at critical high- and low-load limits. Revisiting Prandtl's 1933 [11] assumption that the wing-structure weight is proportional to the bending moments, Phillips et al. [24,25] used

simple beam theory to define the proportionality coefficient S_b in terms of the beam geometric and material properties, i.e., [24],

$$W_s = \int_0^{b/2} \frac{\left|\widetilde{M}_b(y)\right|}{S_b(y)}\,dy; \quad S_b(y) \equiv \frac{C_\sigma (t_{\max}/c)c(y)\sigma_{\max}}{\gamma}, \quad C_\sigma \equiv \frac{2I(h/t_{\max})}{Ah^2} \tag{1.4}$$

where $t_{\max}/c$ is the wing thickness-to-chord ratio, c is the wing chord, $\sigma_{\max}$ is the maximum allowable stress, γ is the specific weight of the wing-structure material, and I, A, and h are the second moment of inertia, area, and height of the wing structure, respectively. Note that Eq. (1.4) is analogous to Eq. (1.3) but includes a limit on the maximum allowable bending stress. Thus, Eq. (1.4) describes the wing-structure weight for stress-limited designs. Phillips et al. [24] also included deflection constraints by relating the maximum allowable deflection to the maximum allowable stress to give [24]

$$W_s = \int_0^{b/2} \frac{\left|\widetilde{M}_b(y)\right|}{S_b(y)}\,dy; \quad S_b(y) \equiv \frac{C_\delta E(t_{\max}/c)^2 c(y)^2 \delta_{\max}}{\gamma}, \quad C_\delta \equiv \frac{8I(h/t_{\max})^2}{Ah^2} \tag{1.5}$$

where E is the modulus of elasticity of the wing-structure material, and $\delta_{\max}$ is the maximum allowable deflection. Thus, Eq. (1.5) describes the wing-structure weight for deflection-limited designs.

Whereas Prandtl [11] assumed that the wing bending moments are a function of the lift distribution alone, Phillips et al. [24,25] assumed that the bending moments are related to the lift distribution and wing weight distribution according to the relation [24]

$$\widetilde{M}_b(z) = \int_{z'=z}^{b/2} [\widetilde{L}(z') - n_a \widetilde{W}_n(z') - n_a \widetilde{W}_s(z')](z'-z)dz', \quad \text{for } z \geq 0 \tag{1.6}$$

where $\widetilde{L}(z)$ is the section lift distribution, $\widetilde{W}_s(z)$ is the section wing-structure weight,

$\widetilde{W}_n(z)$ is the section weight of all non-structural elements carried in the wing, and n_a is the load factor. At all points, the wing structure must be designed to support the bending moments encountered during a high-load maneuver and during a negative-load maneuver, such as a hard landing. To obtain analytic results, Phillips et al. [24,25] assumed that the lift distribution is fixed for all flight phases and identified lift distributions that minimize induced drag under constraints of fixed gross weight [24], fixed net weight [25], fixed wing loading [24,25], and fixed stall speed [25].

The solutions from the studies described here show that by tailoring the lift distribution, the wingspan can be increased by between 1-33% over that allowed by the elliptic lift distribution with the same weight, giving a 1-16% reduction in induced drag, depending on the design constraints. However, each of these studies includes assumptions that may not be representative of all aircraft. For example, Refs. [11,20,21] include assumptions about the proportionality between the wing-structure weight and the wing bending moments that correspond to rectangular wings. References [11-22] include the assumption that the bending moments are caused by the lift alone, which limits their application to wings with negligible structural or payload weight. The formulations given by Phillips et al. [24,25] are arguably more general than those given in Refs. [11-22], but in order to obtain analytic solutions, Phillips et al. [24,25] limited their results to rectangular wings with a single ideal weight distribution.

Taylor [26] extended the method of Phillips et al. [24,25] and presented preliminary results of a semi-analytic method for the aerostructural optimization of wings with tapered planforms and a numerical method for wings with arbitrary planform and payload distribution [26]. Still, these methods, like the others discussed in this section assume that

the lift distribution is fixed throughout the course of the flight, which is only true for rigid wings operating at a fixed lift coefficient. For most aircraft, the lift distribution changes during flight, requiring active wing-shaping control to maintain optimum aerodynamic performance at all flight conditions.

1.2 Active Wing-Shaping Control

Active Wing-Shaping control is a general term denoting the use of control effectors or propulsion elements to actively change the wing geometry during flight to achieve desired aerodynamic characteristics. It is often colloquially referred to as "morphing". The use of active wing shaping is not new. As early as 1903, the Wright Brothers used wing shaping in the form of twist to control their flyer. Although the practice was generally discontinued in favor of fixed, discrete control surfaces, wing-shaping has reemerged in recent years as a potential improvement over traditional discrete control surfaces. Active wing-shaping concepts have been explored by organizations including NASA [27-29], the Air Force Research Laboratory (AFRL) [30], the Defense Advanced Research Projects Agency (DARPA) [31], Boeing [32], Airbus [33], the European Union [34]. The general idea of each of these concepts is to use active wing shaping to tailor the aerodynamics in flight to achieve a desired goal, such as noise reduction, flutter suppression, gust load alleviation, enhanced control, or performance improvement. In this dissertation, the methods and results are focused on the performance benefits of wing-shaping control in the context of load alleviation and drag and fuel-burn minimization.

One of the main benefits of wing-shaping control is the ability to tailor the wing geometric or aerodynamic twist to alter the lift distribution in-flight and achieve desired performance characteristics. This can be achieved through a variety of morphing

mechanisms, but the work presented in this dissertation is primarily relevant for morphing strategies that employ twist and camber morphing mechanisms. Some examples of this type of mechanism include the Mission Adaptive Compliant Wing designed by FlexSys [35], the University of Bristol's FishBAC morphing aerofoil [36], NASA's Variable Camber-Continuous Trailing-Edge Flap (VCCTEF) system [29], the Air Force Research Laboratory's Variable Camber-Compliant Wing (VCCW) [30], and many others [37-39].

1.2.1 Load Alleviation

Using active wing-shaping controls, the performance benefits of load alleviation are often more pronounced than for static wing designs. If the wing aerodynamic or geometric twist can change during flight, the designer is not limited to a single lift distribution or a fixed twist distribution. Instead, the twist can theoretically be tailored to produce the elliptic lift distribution at cruise and a load-alleviating lift distribution at the design limit. In some cases, this may result in even greater drag savings than those seen in Refs. [11-26].

For example, Hunsaker et al. [40] showed that when wing-shaping controls are used to actively tailor the lift distribution during flight, the drag can be reduced on the order of 5% over the single-point optimized solution. Similarly, Curiale and Zingg [41] showed that morphing can increase the max L/D on the order of 5% for a hypothetical transonic regional transport, and Burdette et al. [42] and Burdette and Martins [43] found that, using morphing, fuel burn can be reduced on the order of 5% over the non-morphing optimized design of the NASA Common Research Model (CRM) transonic transport configuration [44,45]. Burdette et al. [42] point out that morphing mechanisms improve

fuel burn using two mechanisms: maneuver load alleviation and reduced coupling between cruise and maneuver conditions. Thus, to fully take advantage of load alleviation, the use of wing-shaping controls should be considered from the beginning of the design process, instead of at the end to solely mitigate negative effects that arise later in the design process or in a retrofit scenario.

1.2.2 Aeroelastic Wing-Shaping Control

Active wing-shaping control can be leveraged to mitigate negative aeroelastic effects at off-design cruise conditions. This practice is known as aeroelastic wing-shaping control. The jig twist on modern aircraft wings is carefully designed to account for static aeroelastic deformations caused by the wing weight and aerodynamic load distributions at the design condition. However, for a wing with fixed jig twist, the aerodynamic load distribution is a function of the lift coefficient and therefore may change over the course of a flight as the lift coefficient changes. Moreover, for a commercial aircraft, which may begin cruise with as high as 80% fuel and end cruise with as low as 20% fuel, the wing weight distribution may change substantially during flight. The resulting changes to the static aeroelastic deformations on the wing may cause the aircraft to operate at off-design conditions for the majority of cruise.

Additionally, at off-design cruise conditions, the effects of aeroelasticity can further alter the lift distribution and result in even greater performance reductions. This is especially true for highly-flexible aircraft, which are becoming increasingly ubiquitous as advances in material science result in more-flexible aircraft wings made of light-weight, high-strength composites with reduced stiffness. Aeroelasticity affects the aircraft performance primarily through elastic twist induced by the spanloads. Under typical

cruise-loading scenarios, the aeroelastic twist tends to redistribute the lift inboard, bringing the lift distribution away from the optimum elliptical shape, and it tends to lower the overall lift coefficient, requiring the aircraft to operate at a higher angle of attack. Both of these phenomena have detrimental effects on the aircraft performance. However, distributed wing-shaping control effectors can be used dynamically throughout the course of a flight to effectively re-twist the wing to mitigate these aeroelastic effects and produce an optimal or near-optimal lift distribution at each individual flight condition.

This strategy is especially relevant when considering the benefits of retrofitting an existing wing with wing-shaping controls. The majority of the research done on the benefits of aeroelastic wing shaping has been along these lines. For example, Lebofsky et. al [46,47], Ippolito et al. [48], Nguyen et al. [49], Ting et al. [50] and Chaparro et al. [51] studied the potential benefits of using the VCCTEF system on the commercial-class NASA Generic Transport Model (GTM) aircraft. Under cruise trim conditions, Lebofsky et al. [46,47] showed that the VCCTEF can produce drag reductions on the order of 10-20% over the baseline GTM configuration at off-design conditions. Ting et al. [50] and Chaparro et al. [51] found that when transonic effects are considered, the VCCTEF can reduce total cruise drag by 6-8% over the baseline GTM configuration. Rodriguez et al. [52] showed that optimizing the VCCTEF by condition can reduce traditional cruise drag and overspeed cruise drag by about 5%. Nguyen et al. [49] showed that when flutter constraints and the effects of load alleviation are considered, the VCCTEF can reduce drag total cruise drag by up to 5.6% and reduce the root bending moment by around 25% over the baseline GTM model.

Results similar to those shown in refs. [46-52] have also been shown for other wing geometries, including the NASA CRM [53,54], the LANN wing [55], the Goland Wing [56], and others [57,58]. In general, these studies show that aeroelastic wing-shaping control can feasibly and substantially improve the performance of flexible aircraft under a variety of practical constraining conditions. Moreover, many of these studies show that benefits of aeroelastic wing-shaping control are more pronounced with increased wing flexibility.

1.2.3 Trajectory Optimization

To determine the effect of active wing shaping over the course of a flight, the effect of the wing shaping on the optimum flight trajectory must also be considered. Traditionally, fuel burn optimization has been done using multipoint methods. Multipoint optimization entails the simultaneous optimization of the wing design at several points in the flight, called a flight stencil. The flight stencil is chosen to represent the nominal flight conditions, design load conditions, and any other flight condition that the aircraft may encounter. Thus, the optimizer must balance between all of the flight conditions. Examples of multipoint optimization approaches for morphing aircraft are shown in Refs. [43,59-62] The multipoint approach depends on the flight points chosen, and it requires the user to select points from a pre-determined representative mission profile. However, the optimal trajectory for aircraft that employ wing-shaping control may differ somewhat from the optimal trajectory for a non-morphing aircraft.

In order to perform a fair comparison between aircraft with traditional control surfaces and an aircraft employing wing-shaping control, the potential effects of wing shaping on optimal trajectory must be considered. Although trajectory optimization

studies for morphing flight vehicles have become common in recent years, there are relatively few of these studies that consider the effects of camber morphing on the optimum trajectory of typical manned aircraft types. However, Nguyen et al. [63] recently presented preliminary results for the trajectory for a transonic truss-braced wing with active wing shaping controls employing camber-morphing mechanisms. Fasel et al. [64] preformed a concurrent design and trajectory optimization for an airborne wind energy system (energy kite) and found that camber morphing can increase power production by nearly 8% with relatively small changes in the flight trajectory. Jasa et al. [65] presented a coupled aerostructural and trajectory optimization for the CRM wing with a morphing trailing edge and showed modest fuel-burn reductions of less than 1% over the course of a long-range cruise. Rudnick-Cohen et al. [66,67] have also sought to develop methods for concurrent airframe and mission design of aircraft with camber-morphing wings.

The concurrent physical design and trajectory or mission design of an aircraft or other engineering system is sometimes referred to as co-design [68]. Modern methods for co-design are generally focused on linking a variety of independent high-fidelity or reduced-order models within a multidisciplinary optimization routine. Co-design methods for multidisciplinary aircraft design are currently under development by AFRL [69-74], NASA [75-79], and the University of Michigan [80], among others [81-82]. Such methods have been used for various multidisciplinary aircraft design applications. However, because these methods generally rely on linking black-box computational models, much of the relational understanding about the coupling between disciplines is largely lost. Moreover, ensuring that the models are properly linked requires some

reference solution or intuition regarding the coupling between each discipline.

In this dissertation, this need is addressed by presenting a series of multidisciplinary reference solutions that account for the effects of aerostructural load alleviation and the optimal trajectory in the design and optimization of aircraft wings that use static and active wing shaping to tailor the aerodynamic load distributions. The purpose of these reference solutions is twofold. One purpose is to present reference solutions that reveal to what extent wing shaping can be employed to reduce aircraft cruise fuel consumption. The second purpose is to advance fundamental understanding of multidisciplinary design and optimization by revealing relational information regarding the coupling between aerodynamics, wing structure, and the aircraft trajectory and to serve as simple multidisciplinary test cases that can be used to validate results from higher fidelity multidisciplinary design codes. The analytic and low-order methods used to obtain these solutions will be presented, and their utility for rapidly obtaining valuable relational design information will be demonstrated through design-space exploration and optimization. It should be noted that this dissertation is focused on the use of camber morphing for load alleviation, rather than aeroelastic wing shaping, to reduce weight, drag, and fuel burn over the course of a flight trajectory. Therefore, the solutions presented in this dissertation do not include the effects of aeroelasticity.

This dissertation is comprised of a series of four standalone papers, along with three appendices containing supplemental information, that present studies focused on the effects of aerostructural load alleviation from static and active wing shaping on the efficiency and optimum trajectory of aircraft. Chapters 2-4 build primarily on the early aerostructural work performed by Prandtl [11] and the more recent work of Phillips et

al. [24,25] to identify the effects of tailoring the lift distribution through static wing shaping on the coupling between aerodynamics and structures in the design of wings for minimum induced drag. The first paper, presented in Chapter 2, assesses Prandtl's 1933 conclusion that tapered wings have an advantage over wings with rectangular planforms by revisiting preliminary work of Taylor [26] on minimizing induced drag for tapered wings under structural constraints. In assessing Prandtl's conclusions [11,12], this paper also reveals important insights on how the wing planform affects aerostructural coupling and suggests that a triangular planform may achieve drag reductions of nearly 15% over a rectangular planform in the absence of stall.

In Chapter 3, the methods from Chapter 2 are generalized to wings with arbitrary planform and weight distribution to further identify how the planform, weight distribution, and other design variables and constraints may affect the optimum aerostructural design of wings with static wing shaping. The methods from this chapter are demonstrated through an optimization case study and design-space exploration on a high-endurance UAV, the results reveal important insights on the relative importance of aerodynamic, structural, and operational design parameters in designing a wing for minimum induced drag.

The paper presented in Chapter 4 addresses the extent to which theoretical aerostructural solutions based on static wing shaping, including many of those presented in Refs. [11-25], may apply to practical aircraft configurations and if they may serve as appropriate reference solutions for higher-fidelity computational studies. The results suggest that, when appropriate constraints are considered, theoretical aerostructural solutions agree well with solutions from high-fidelity computational studies for a variety

of aircraft configurations and flight conditions.

Chapter 5 presents a study focused on the use of active wing shaping to minimize fuel consumption over the optimal flight trajectory. The study combines a simple trajectory optimization framework with many of the aerostructural relationships presented in the previous chapters. The results from this study suggest that active wing shaping can be used to substantially reduce fuel burn over an optimized wing with static wing shaping, operating along its optimum trajectory. They also show that wing shaping can have a substantial effect on the optimum trajectory, with active wing shaping tending to favor lower-speed trajectories with higher lift coefficients and higher lift-to-drag ratios than the baseline non-morphing configurations.

REFERENCES

[1] 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)

[2] 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)

[3] Gallay, S., and Laurendeau, E., "Preliminary-Design Aerodynamic Model for Complex Configurations Using Lifting-Line Coupling Algorithm," *Journal of Aircraft*, Vol. 53, No. 4, 2016, pp. 1145–1159. (doi:10.2514/1.C033460)

[4] Phillips, W. F., and Hunsaker, D. F., "Lifting-Line Predictions for Induced Drag and Lift in Ground Effect," *Journal of Aircraft*, Vol. 50, No. 4, 2013, pp. 1226–1233. (doi:10.2514/1.C032152)

[5] Wickenheiser, A., and Garcia, E., "Aerodynamic Modeling of Morphing Wings Using an Extended Lifting-Line Analysis," *Journal of Aircraft*, Vol. 44, No. 1, 2007, pp. 10–16. (doi:10.2514/1.18323)

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

[7] Rasmussen, M. L., and Smith, D. E., "Lifting-Line Theory for Arbitrarily Shaped Wings," *Journal of Aircraft*, Vol. 36, No. 2, 1999, pp. 340–348. (doi:10.2514/2.2463)

[8] Bera, R. K., "Some remarks on the solution of the lifting line equation," *Journal of Aircraft*, Vol. 11, No. 10, 1974, pp. 647–648. (doi:10.2514/3.44397)

[9] Prandtl, L., "Tragflügel Theorie," *Nachricten von der Gesellschaft der Wissenschaften zu Göttingen*, Geschäeftliche Mitteilungen, Klasse, 1918, pp. 451–477.

[10] Prandtl, L., "Applications of Modern Hydrodynamics to Aeronautics," NACA TR-116, June 1921.

[11] Prandtl, L., "Über Tragflügel kleinsten induzierten Widerstandes," Zeitschrift für Flugtechnik und Motorluftschiffahrt, Vol. 24, No. 11, 1933, pp. 305–306.

[12] Hunsaker, D. F., "Ludwig Prandtl's 1933 Paper Concerning Wings for Minimum Induced Drag, Translation and Commentary," AIAA SciTech 2020 Forum, Orlando, Florida, 6-10 January 2020.

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

[14] Gopalarathnam, A., and Norris, R. K., "Ideal Lift Distributions and Flap Angles for Adaptive Wings," *Journal of Aircraft*, Vol. 46, No. 2, 2009, pp. 562–571. (doi:10.2514/1.38713)

[15] Verstraetan, J., G., and Slingerland, R., "Drag Characteristics for Optimally Span-Loaded Planar, Wingletted, and C Wings," *Journal of Aircraft*, Vol. 46, No. 3, May-June 2009, pp. 962-971. (doi:10.2515/1.39426)

[16] Ranjan, P., "Computational Analysis of Planar Wings Designed for Optimum Span-Load," *MS Thesis*, University of Illinois at Urbana-Champaign, 2016.

[17] Pate, D. J., and German, B. J., "Lift Distributions for Minimum Induced Drag with Generalized Bending Moment Constraints," *Journal of Aircraft*, Vol. 50, 2013, pp. 936–946. (doi:10.2514/1.C032074)

[18] DeYoung, J., "Minimization Theory of Induced Drag Subject to Constraint Conditions," NASA CR-3140, June 1979.

[19] Jones, R. T., and Lasinski, T. A., "Effect of Winglets on the Induced Drag of Ideal Wing Shapes," NASA TM-81230, Sept. 1980.

[20] Klein, A., and Viswanathan, S. P., "Minimum Induced Drag of Wings with Given Lift and Root-Bending Moment," *Zeitschrift fur Angewandte Mathematik und Physik*, Vol. 24, 1973, pp. 886–892.

[21] Klein, A., and Viswanathan, S. P., "Approximate Solution for Minimum Induced Drag of Wings with Given Structural Weight," *Journal of Aircraft*, Vol. 12, No. 2, 1975, pp. 124–126. (doi:10.2514/3.44425)

[22] Löbert, G., "Spanwise Lift Distribution for Forward- and Aft-Swept Wings in Comparison to the Optimum Distribution Form," *Journal of Aircraft*, Vol. 18, No. 6, 1981, pp. 496-498. (doi:10.2514/3.44717)

[23] McGeer, T., "Wing Design for Minimum Drag with Practical Constraints," *Journal of Aircraft*, Vol. 21, 1984, pp. 879–886. (doi:10.2514/3.45058)

[24] 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, pp. 431-441. (doi:10.2514/1.C035027)

[25] Phillips, W. F., Hunsaker, D. F., and Taylor, J. D., "Minimizing Induced Drag with Weight Distribution, Lift Distribution, Wingspan, and Wing-Structure Weight," AIAA 2019-3349, AIAA Aviation 2019 Forum, Dallas, Texas, 17-21 June 2019. (doi:10.2514/6.2019-3349)

[26] Taylor, J. D., "Methods for the Aerostructural Design and Optimization of Wings with Arbitrary Planform and Payload Distribution," M.S. Thesis, Utah State University, Logan, UT, 2017.

[27] Perry III, B., Cole, S. R., and Miller, G. D., "A Summary of the Active Flexible Wing Program," *Journal of Aircraft*, Vol. 32, No. 1, February 1992, pp. 10-15. (doi:10.2514/3.46677)

[28] Bonnema, K. L. and Lokos, W. A., "AFTI/F-111 Mission Adaptive Wing Flight Test Instrumentation Overview," *Instrumentation in the Aerospace Industry, Proceedings of the ISA Aerospace Instrumentation Symposium*, Vol. 35, pp. 809-840.

[29] Nguyen, N., Elastically Shaped Future Air Vehicle Concept, NASA Innovation Fund Award 2010 Report, October 2010.

[30] Joo, J., Marks, C. Zientarski, L., and Culler, A., "Variable Camber compliant Wing-Design," AIAA Paper 2015-1050, 23rd AIAA/AHS Adaptive Structures Conference, Kissimmee, FL, 5-9 January 2015.

[31] Weisshaar, T. A., "Morphing Aircraft Technology – New Shapes for Aircraft Design," RTO-MP-AVT-141, October 2006.

[32] Pendleton, E., Griffin, K.E., Kehoe, M. W., and Perry III, B. "A Flight Research Program for Active Aeroelastic Wing Technology, 37th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Salt Lake City, UT, 15-17 April 1996. (doi:10.2514/6.1996-1574)

[33] Wilson, T., Kirk, J., Hobday, J., and Castrichini, A., "Small Scale Flying Demonstration of Semi Aeroelastic Hinged Wing Tips," International Forum on Aeroelasticity and Structural Dynamics, Savannah GA, 9-13 June 2019.

[34] Pecora, R., "Morphing wing flaps for large civil aircraft: Evolution of a smart technology across the Clean Sky program," *Chinese Journal of Aeronautics*, Vol. 34, No. 7, July 2021, pp. 13-28. (doi:10.1016/j.cja.2020.08.004)

[35] Hetrick, J. Osborn, R., Kota, S. Flick, P., and Paul, D., AIAA paper 2007-1709, "Flight Testing of Mission Adaptive Compliant Wing," 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Honolulu, HI, 23-26 April 2007. (doi:10.2514/6.2007-1709)

[36] Woods B. K. S. and Friswell, M. I., "Preliminary Investigation of a Fishbone Active Camber Concept," *Proceedings of the ASME Conference on Smart Materials, Adaptive Structures, and Intelligent Systems*, 2012, pp. 555-563. (doi:10.1017/CBO9781107415324.004)

[37] Phillips, W. F., "New Twist on an Old Wing Theory," *Aerospace America*, January 2005, pp. 27-30.

[38] Vos, R., Gurdal, Z., and Abdalla, M., "Mechanism for Warp-Controlled Twist of a Morphing Wing," *Journal of Aircraft*, Vol. 47, No. 2, March-April 2010, pp. 450-457. (doi:10.2514/1.39328)

[39] Dale, A. S., Cooper, J. E., and Mosquera, A., "Adaptive Camber-Morphing Wing Using 0-v Honeycomb," AIAA paper 2013-1510, 54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Boston, MA, 8-11 April 2013. (doi:10.2514/6.2013-1510)

[40] Hunsaker, D. F., Phillips, W. F., and Joo, J. J., "Aerodynamic Shape Optimization of Morphing Wings at Multiple Flight Conditions," AIAA paper 2017-1420, 55th AIAA Aerospace Sciences Meeting, Grapevine, TX, 9-13 January 2017. (doi:10.2514/6.2017-1420)

[41] Curiale, N. J., and Zingg, D. W., "Morphing Wings: A Study Using Aerodynamic Shape Optimization," AIAA paper 2018-1910, 2018 AIAA/ASCE/AHS/ASC

Structures, Structural Dynamics, and Materials Conference, Kissimmee, FL, 8-12 January 2018. (doi:10.2516/6.2018-1910)

[42] Burdette, D.A., Kenway, G. K., Lyu, Z., and Martins, J. R. R. A., AIAA paper 2015-1129, "Aerostructural Design Optimization of an Adaptive Morphing Trailing Edge Wing," 56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Kissimmee, FL, 5-9 January 2015. (doi:10.2514/6.2015-1129)

[43] Burdette, D. A., and Martins, J. R. R. A, "Impact of Morphing Trailing Edges on Mission Performance for the Common Research Model," *Journal of Aircraft*, Vol. 56, No. 1, January 2019, pp. 369-384. (doi:10.2514/1.C034967)

[44] Vassberg, J. C., DeHaan, M. A., Rivers, S. M., and Wahls, R. A., "Development of a Common Research Model for Applied CFD Validation Studies," AIAA 2008-6919, 26th AIAA Applied Aerodynamics Conference, Honolulu, HI, 18-21 August 2008. (doi:10.2514/6.2008-6919)

[45] Brooks, T. R., Kenway, G. K. W., and Martins, J. R. R. A., "Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings," *AIAA Journal*, Vol. 56, No. 7, July 2018 pp. 2840- 2855. (doi:10.2514/1.J056603)

[46] Lebofsky, S., Ting, E., Nguyen, N. T., and Trinh, K. V., "Aeroelastic Modeling and Drag Optimization of Flexible Wing Aircraft with Variable Camber Continuous Trailing Edge Flap," AIAA paper 2014-2443, 32nd AIAA Applied Aerodynamics Conference, Atlanta, GA, 16-20 June 2014, (doi:10.2514/6.2014-2443)

[47] Lebofsky, S. Ting, E., and Nguyen, N. T., "Multidisciplinary Drag Optimization of Reduced Stiffness Flexible Wing Aircraft with Variable Camber Continuous Trailing Edge Flap," AIAA paper 2015-1408, 56th AIAA/ASCE/AHS/ASC Structures,

Structural Dynamics, and Materials Conference, Kissimmee, FL, 5-9 January 2015. (doi:10.2514/6.2015-1408)

[48] Ippolito, C., Nguyen, N. T., Totah, J., Trinh, K. V., and Ting, E., "Initial Assessment of a Variable-Camber continuous Trailing-Edge Flap System on a Rigid Wing for Drag Reduction in Subsonic Cruise," AIAA InfoTech@Aerospace Conference, Boston, MA, 19-22 August 2013.

[49] Nguyen, N. T., Ting, E., Chaparro, D., Drew, M. C., and Swei, S. S. "Multi-Objective Flight control for Drag Minimization and Load Alleviation of High-Aspect Ratio Flexible Wing Aircraft," AIAA paper 2017-1589, 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Grapevine, TX, 9-13 January 2017. (doi:10.2514/6.2017-1589)

[50] Ting, E., Chaparro, D., and Nguyen, N. T., "Aero-Structural Optimization of Variable Camber Continuous Trailing Edge Flap Configurations Using Transonic and Viscous Potential Flow Method," AIAA paper 2017-4420, 35th AIAA Applied Aerodynamics Conference, Denver, CO, 5-9 June 2017. (doi:10.2514/6.2017-4420)

[51] Chaparro, D., Fujiwara, G. E., Ting, E., and Nguyen, N. T., "Transonic and Viscous Potential Flow Method Applied to Flexible Wing Transport Aircraft," AIAA paper 2017-4221, 35th AIAA Applied Aerodynamics Conference, Denver, CO, 5-9 June 2017. (doi:10.2514/6.2017-4221)

[52] Rodriguez, D. L., Aftosmis, M. J., Nemec, M., and Anderson, G. R., "Optimization of Flexible Wings with Distributed Flaps at Off-Design Conditions," *Journal of Aircraft*, Vol. 53, No. 6, November 2016, pp. 1731-1745. (doi:10.2514/ 1.C033535)

[53] Fujiwara, G. E., Nguyen, N. T., Livne, E., and Bragg, M. B., "Aerostructural Design Optimization of a Flexible Wing Aircraft with Continuous Morphing Trailing Edge," AIAA paper 2018-3571, 2018 Multidisciplinary Analysis and Optimization Conference, Atlanta, GA, 25-29 June 2018. (doi:10.2514/6.2018-3571)

[54] Stanford, B. K., "Static and Dynamic Aeroelastic Tailoring with Variable Camber Control," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 11, November 2016, pp. 2522-2534. (doi:10.2514/1.G000413)

[55] Kier, T. M., Leitner, M., Suelozgen, O., and Pusch, M., "An Integrated Flexible Aircraft Model for Optimization of Lift Distributions," AIAA paper 2019-2039, AIAA SciTech 2019 Forum, San Diego, CA, 7-11 January 2019. (doi:10.2514/6.2019-2039)

[56] Kolonay, R. M., and Eastep, F. E., "Optimal Scheduling of Control Surfaces on Flexible Wings to Reduce Induced Drag," *Journal of Aircraft*, Vol. 43, No. 6, November 2006, pp. 1655-1661. (doi:10.2514/1.14604)

[57] Hammerton, J. R., Su, W., Zhu, G., and Swei, S. S., "Optimum distributed wing shaping and control loads for highly flexible aircraft," *Aerospace Science and Technology*, Vol. 79, August 2018, pp. 255-265. (doi:10.1016/j.ast.2018.05.045)

[58] Su. W., Swei, S. S., and Zhu, G. G., "Optimum Wing Shape of Highly Flexible Morphing Aircraft for Improved Flight Performance," *Journal of Aircraft*, Vol. 53, no. 5, September 2016, pp. 1305-1316. (doi:10.2514/1.C033490)

[59] Liem, R. P., Kenway, G. K. W., and Martins, J. R. R. A., "Multimission Aircraft Fuel-Burn Minimization via Multipoint Aerostructural Optimization," *AIAA Journal*, Vol. 53, No. 1, January 2015, pp. 104-122. (doi:10.2514/1.J052940)

[60] Adler, E. J., and Martins, J. R. R. A., "Aerostructural wing design optimization considering full mission analysis," AIAA paper 2022-0382, 2022 AIAA SciTech Forum, San Diego, CA and online, 3-7 January 2022. (doi:10.2514/6.2022-0382)

[61] Kenway, G. K. W. and Martins, J. R. R. A., "Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration," *Journal of Aircraft*, Vol. 51, No. 1, January 2014, pp. 144-160. (doi:10.2514/1.C032150)

[62] Molinari, G. quack, M., Dimitriev, V., Moriari, M., Jenny, P., and Ermanni, P., "Aero-Structural Optimization of Morphing Airfoils for Adaptive Wings," *Journal if Intelligent Material Systems and Structures*, Vol. 22, July 2011, pp. 1075-1089. (doi:10.1177/1045389X11414089)

[63] Nguyen, N. T., Xiong, J., and Sager, J., "Fuel-Optimal Trajectory Optimization of Mach 0.745 Transonic Truss-Braced Wing with Variable Camber Continuous Trailing Edge Flap," AIAA paper 2021-2575, AIAA Aviation 2021 Forum, Virtual, 2-6 August 2021. (doi:10.2514/6.2021-2575)

[64] Fasel, U., Tiso, P., Keidel, D., and Ermanni, E., "Concurrent Design and Flight Mission Optimization of Morphing Airborne Wind Energy Wings," *AIAA Journal*, Vol. 59, No. 4, April 2021, pp. 1254-1268. (doi:10.2514/1.J059621)

[65] Jasa, J. P., Hwang, J. T., and Martins, J. R. R. A., "Design and Trajectory Optimization of a Morphing Wing Aircraft," AIAA paper 2018-1382, 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Kissimmee, FL, 8-12 January 2018. (doi:10.2514/6.2018-1382)

[66] Rudnick-Cohen, E. S., Hodson, J. D., Reich, G. W., Pankonien, A. M., and Beran, P. S., "Design and Trajectory Planning Optimization of a Morphing Airfoil for 3-D

Flight Maneuvers," AIAA paper 2021-0890, AIAA SciTech 2021 Forum, Virtual, 11-15 & 19-21 January 2021. (doi:10.2514/6.2021-0890)

[67] Rudnick-Cohen, E. S., Hodson, J. D., Reich, G. W., Pankonien, A. M., and Beran, P. S., " Robust Optimal Design and Control of a Morphing Unmanned Aerial Vehicle (UAV) Airfoil for a Range of Flight Maneuvers," AIAA paper 2020-3112, AIAA Aviation 2020 Forum, Virtual, 15-19 June 2020. (doi:10.2514/6.2020-3112)

[68] Allison, J. T., and Herber, D. R., "Special section on multidisciplinary design optimization: multidisciplinary design optimization of dynamic engineering systems," *AIAA Journal*, Vol. 52, No.4, March 2014, pp.691–710. (doi:10.2514/1.J052182)

[69] Clive, P. D., Johnson, J. A., Moss, M. J., Zeh, J. M., Birkmire, B. M., and Hodson, D. D., "Advanced Framework for Simulation, Integration and Modeling (AFSIM)" International Conference on Scientific Computing (CSC), Las Vegas, NV, 27–30 July 2015.

[70] Reuter, R. A., Iden, S., Snyder, R. D., and Allison, D. L., AIAA paper 2016-0674, "An overview of the optimized integrated multidisciplinary systems program," 57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, San Diego, CA, 4–8 January 2016. (doi:10.2514/6.2016-0674)

[71] Alyanak, E. J., and Allison, D .L., "Multi-Parameter Performance Evaluation, the Next Step in Conceptual Design Concept Assessment," AIAA paper 2015-0648, 56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Kissimmee, FL, 5–9 January 2015. (10.2514/6.2015-0648)

[72] Allison, D. L. and Kolonay, R. M., "Expanded MDO for Effectiveness Based Design Technologies: EXPEDITE Program Introduction," AIAA paper 2018-3419, 2018 Multidisciplinary Analysis and Optimization Conference, Atlanta, GA, 25–29 June 2018. (doi:10.2415/6.2018-3419)

[73] Davies, C. C., "Lockheed Martin Overview of the AFRL EXPEDITE Program," AIAA paper 2020-1129, AIAA SciTech 2020 Forum, San Diego, CA, 6–10 January 2020. (doi:10.2514/6.2020-1129)

[74] Clark, D. L., Allison, D. L., Bae, H., and Forster, E. E., "Effectiveness-Based Design of an Aircraft Considering Mission Uncertainties," *Journal of Aircraft*, Vol. 56, No. 5, September 2019, pp.1–12. (doi:10.2514/1.C035402)

[75] Falck, R. D., Chin, J., Schnulo, S. L., Burt, J. M., and Gray, J. S., "Trajectory Optimization of Electric Aircraft Subject to Subsystem Thermal Constraints," AIAA paper 2017-4002, 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Denver, CO, 5–9 June 2017. (doi:10.2514/6.2017-4002)

[76] Hendricks, E. S., Falck, R. D., and Gray, J. S., "Simultaneous Propulsion System and Trajectory Optimization." AIAA paper 2017-4435, 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Denver, CO, 5–9 June 2017. (doi:10.2514/6.2017-4435)

[77] Schnulo, S. L., Chin, J., Falck, R. D. F., Gray, J. S., Papathakis, K. V., Clarke, S. C., Reid, N., and Borer, N.K., "Development of a Multi-Segment Mission Planning Tool for SCEPTOR X-57," AIAA paper 2018-3738, 2018 Multidisciplinary Analysis and Optimization Conference, Atlanta, GA, 25–29 June 2018. (doi:10.2514/6.2018-3738)

[78] Falck, R. D. and Gray, J. S., "Optimal control within the context of multidisciplinary design, analysis, and optimization," AIAA paper 2019-0976, AIAA SciTech 2019 Forum, San Diego, CA, 7–11 January 2019. (doi:10.2514/6.2019-0976)

[79] Hendricks, E. S., Falck, R. D., Gray, J. S., Aretskin-Hariton, E. D., Ingraham, D.J., Chapman, J. W., Schnulo, S. L., Chin, J. C., Jasa, J. P., and Bergeson, J. D., "Multidisciplinary Optimization of a Turboelectric Tiltwing Urban Air Mobility Aircraft," AIAA paper 2019-3551, AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Dallas, TX, USA, 17–21 June 2019. (doi:10.2514/6.2019-3551)

[80] Jasa, J. P., Brelje, B. J., Gray, J. S., Mader, C. A., and Martins, J. R. R. A., "Large-Scale Path-Dependent Optimization of Supersonic Aircraft," *Aerospace*, Vol. 7, No. 152, 2020, pp. 1-18. (doi:10.3390/aerospace7100152)

[81] Joshy, A. J., and Hwang, J. T., "A new architecture for large-scale system design optimization," AIAA paper 2020-3125, AIAA Aviation 2020 Forum, Virtual, 15-19 June 2020. (doi:10.2514/6.2020-3125)

[82] Hwang, J. T., Jain, A. V., and Ha, T. H., "Large-scale multidisciplinary design optimization – review and recommendations," AIAA paper 2019-3106, AIAA Aviation 2019 Forum, Dallas, TX, 17-21 June 2019. (doi:10.2514/6.2019-3106)

CHAPTER 2

Minimum Induced Drag for Tapered Wings Including Structural Constraints

Jeffrey D. Taylor* and Douglas F. Hunsaker†
Utah State University, Logan, Utah 84322-4130

For a wing in steady level flight, the lift distribution that minimizes induced drag depends on a tradeoff between wingspan and wing-structure weight. In 1933, Prandtl suggested that tapered wings have an advantage over rectangular wings due to this tradeoff. However, Prandtl's solutions were obtained using assumptions that correspond to rectangular wings. Therefore, his claim was not analytically proven by his 1933 publication. Here, an approach similar to Prandtl's is taken with more general approximations that apply to wings of arbitrary planform. This more general development is used to study Prandtl's claim about tapered wings. Closed-form solutions for the optimum wingspan and corresponding induced drag are presented for wings having elliptic and linearly-tapered planforms with constraints of fixed wing loading and maximum stress. It is shown that induced drag is minimized with a triangular planform, which gives a reduction in induced drag of up to 24.44% over the rectangular planform and up to 11.71% over the elliptic planform. Numerical solutions for the lift distributions that minimize induced drag for each planform are also presented. It is shown that the optimum lift distribution produces up to 5.94%

* PhD Candidate, Mechanical and Aerospace Engineering, 4130 Old Main Hill, AIAA Student Member
† Assistant Professor, Mechanical and Aerospace Engineering, 4130 Old Main Hill, AIAA Senior Member

This paper was presented at the AIAA SciTech 2020 Forum in Orlando, FL as:
Taylor, J. D. and Hunsaker, D. F., "Minimum Induced Drag for Tapered Wings Including Structural Constraints," AIAA 2020-2113, AIAA SciTech Forum, Orlando, FL, 6-10 January 2020. (doi:10.2514/6.2020-2113).
A revised version of this paper was published in *Journal of Aircraft* in 2020 as:
Taylor, J. D., and Hunsaker, D. F., "Minimum Induced Drag for Tapered Wings Including Structural Constraints," *Journal of Aircraft*, Vol. 57, No. 4, July-August 2020, pp. 782-786. (doi:10.2514/1.C035757)

less induced drag than the elliptic lift distribution when the triangular planform is used.

Nomenclature

A = beam cross-sectional area

a_m = fit coefficients in the polynomial approximation of C_n for linearly-tapered wings, Eq. (A1)

B_n = Fourier coefficients in the lifting-line solution for the dimensionless section-lift distribution, Eq. (1)

b = wingspan

C_n = weighting coefficients for B_n in the expressions for wing-structure weight, Eqs. (25) and (44)

C_σ = shape coefficient for the stress-limited design, Eq. (9)

c = local wing section chord length

$\bar{c}$ = wing mean geometric chord, defined as $\bar{c} \equiv S/b$

c_r = local wing section chord length at the wing root

c_t = local wing section chord length at the wing tip

D_i = wing induced drag

h = height of the beam cross-section

I = beam section moment of inertia

L = total wing lift

$\widetilde{L}$ = local wing section lift

$\widetilde{M}_b$ = local wing section bending moment

n_a = load factor, g

n_g = limiting load factor at the hard-landing design limit

n_m = limiting load factor at the maneuvering-flight design limit

R_T = wing taper ratio

R_{b_e} = wingspan ratio for the elliptic planform, Eq. (35)

R_{b_t} = wingspan ratio for the linearly-tapered planform, Eq. (55)

R_{D_e} = induced-drag ratio for the elliptic planform, Eq. (34)

R_{D_t} = induced-drag ratio for the linearly-tapered planform, Eq. (54)

S = wing planform area

S_b = proportionality coefficient between $\widetilde{W}_s(z)$ and $\widetilde{M}_b(z)$ having units of length squared, Eqs. (9) and (13)

$t_{\max}$ = maximum thickness of the local airfoil section

V_∞ = freestream airspeed

W = aircraft gross weight

W_n = aircraft net weight, defined as $W - W_s$

W_r = that portion of W_n carried at the wing root

W_s = total weight of the wing structure required to support the wing bending moment distribution

$\widetilde{W}_n$ = 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

γ = specific weight of the beam material

θ = change of variables for the spanwise coordinate, Eq. (1)

κ_W = weight distribution coefficient, Eq. (8)

ρ = air density

$\sigma_{\max}$ = maximum longitudinal stress

I. Introduction

PRANDTL'S classical lifting-line theory [1,2] relates the spanwise lift distribution to the spanwise chord-length and aerodynamic angle-of-attack distributions for an unswept wing immersed in an inviscid, incompressible, uniform flow. If any two of these distributions are known, Prandtl's classical lifting-line

equation can be used to find the third. For instance, below stall, any desired lift distribution can be produced on any given planform if the correct twist distribution is used. Given a planform shape, the lifting-line equation can be used to find the twist distribution needed to produce the desired lift distribution [3]. Therefore, in this work, we will treat planform and lift distribution as two independent parameters that are linked through the dependent parameter of wing twist. Lifting-line theory has long been used in the aerospace industry for aerodynamic analyses and optimization, and results based on this theory have been shown to be in good agreement with CFD [4-11]. However, lifting-line theory can also be used to gain insight into the aerodynamic and structural coupling involved in designing a wing for minimum induced drag.

From classical lifting-line theory, the spanwise lift distribution can be written in terms of a Fourier series. Although this series is generally written in an alternate form, here we shall use the dimensionless form [12]

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

where $B_1 = 1$. Equation (1) can be used to define any dimensionless lift distribution. However, in this paper, we will only consider spanwise-symmetric lift distributions, which have $B_n = 0$ for all even n. In steady level flight, the lift, L, is equal to the weight, W, and the induced drag can be written in terms of the Fourier coefficients, B_n, as [12]

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

Equation (2) shows that for a given flight condition, the induced drag is a function of the weight, the wingspan, and the Fourier coefficients that define the lift distribution. For any fixed weight and wingspan, Eq. (2) is minimized by using $B_n = 0$ for all $n \geq 2$. This yields the well-known elliptic lift distribution. Any other lift distribution having nonzero Fourier coefficients incurs a penalty in induced drag.

The elliptic lift distribution can be produced using an untwisted elliptically-tapered planform. Because of this, the elliptically-tapered planform is commonly considered the most efficient planform. The lift distribution produced by any other untwisted planform is non-elliptic and therefore produces more induced drag than the untwisted elliptically-tapered planform at a given span. However, it has been shown that linearly-tapered wings produce nearly elliptic lift distributions at taper ratios near $R_T = 0.4$ [13] without any twist. Because a wing with elliptic taper is much more difficult to manufacture than a linearly-tapered planform, linearly-tapered planforms with taper ratios near $R_T = 0.4$ are commonly favored over the elliptic planform to minimize induced drag for a fixed weight and wingspan.

If the weight and wingspan are allowed to vary, the elliptic lift distribution does not necessarily minimize induced drag. Equation (2) shows that the induced drag can be decreased by reducing weight and/or increasing wingspan. However, wingspan cannot be increased arbitrarily because as wingspan increases, the weight of the wing structure, which is proportional to the wing bending moments, also increases. Because of this, if weight and wingspan are not fixed, certain non-elliptic lift distributions that decrease bending moments across the span can allow a larger wingspan than that allowed by the elliptic lift distribution for the same wing-structure weight. Therefore, there exists some optimum wingspan, wing-structure weight, and lift distribution that minimizes induced drag on a wing in steady level flight.

Prandtl seems to be the first to have realized this and published a paper on the topic in 1933 [14]. In that publication, he showed that for a rectangular wing with fixed gross weight and moment of inertia of gross weight, the lift distribution that minimizes induced drag is a bell-shaped lift distribution having $B_3 = -1/3$ and $B_n = 0$ for all $n \neq 3$. Under Prandtl's design constraints [14], this bell-shaped lift distribution allows a 22.5% increase in wingspan and an 11.1% reduction in induced drag over the elliptic lift distribution. In order to obtain analytic results, Prandtl assumed that the wing bending moments are only a function of the lift distribution and that the wing-structure weight makes no contribution to the bending moments. He also assumed that at each section, the bending moment, $\widetilde{M}_b(z)$, is related to the wing-structure weight, $\widetilde{W}_s(z)$, by a spanwise-invariant proportionality coefficient, S_b [14], i.e.,

$$\widetilde{W}_s(z) = \frac{\widetilde{M}_b(z)}{S_b} \tag{3}$$

In his 1933 paper, Prandtl noted that the lift distribution given by $B_3 = -1/3$ corresponds more closely to that produced by what he called "spitzendigen Flügel" [14] (which can be translated as "pointed-end wings" or "tapered wings") than to the elliptic lift distribution. Within the framework of his solution [14], the elliptic lift distribution performs "noticeably worse" [14,15] than the lift distribution given by $B_3 = -1/3$. Therefore, near the end of his 1933 paper, Prandtl concluded that "tapered wings have an advantage over those with a nearly rectangular profile" [14,15]. Prandtl's statement is somewhat vague because any wing having a chord distribution that decreases with span could be considered a tapered wing. For example, elliptically-tapered wings and linearly-tapered wings are two common types of wings that could be considered tapered wings. However, because Prandtl assumed that the proportionality coefficient, S_b, is constant along the span and did not consider the effects of the chord distribution, his solution applies only to rectangular planforms. Therefore, his claim was not proven by his 1933 results [14]. In fact, it appears that no analytic proof of Prandtl's conclusion, with structural and planform effects included, has ever been shown.

Various analytic or low-order studies on minimizing induced drag using lifting-line theory and similar methods have been published since 1933. Whereas many of these studies approach the problem from a purely aerodynamic point of view [16-24], others follow an approach similar to that taken by Prandtl in 1933 [25-38]. Many of the early analytical studies in this second group consider the wing structure independent of the wing geometry. For instance, Jones [32] used the root bending moment as the primary structural constraint, without considering any physical wing structure. DeYoung [33] replaced the root-bending-moment constraint with a constraint on the bending moment at an arbitrary spanwise location. Later, Jones and Lasinski [34] and Klein and Viswanathan [35,36] incorporated constraints on the integrated bending moment, relating the bending moments to the wing-structure weight using the relationship shown in Eq. (3). However, like Prandtl [14], Jones and Lasinski [34] and Klein and Viswanathan [35,36] did not include the effects of the chord distribution and treated the proportionality

coefficient as a fixed parameter, independent of the wing geometry. Löbert [37], on the other hand, introduced a structural constraint based on the ratio of the section bending moment and the wing-section thickness. Because the wing-section thickness is often related to the chord distribution, Löbert's [37] constraint indirectly accounts for the effects of the chord distribution. However, instead of investigating how the chord distribution affects the induced drag, Löbert [37] investigated the effect of sweep on the induced drag for a given chord distribution.

More recently, Phillips et al. [12,38] revisited Prandtl's 1933 analysis [14] and relaxed many of his main assumptions, including the assumption that the proportionality coefficient is spanwise invariant and independent of the wing geometry. Instead, Phillips et al. [12,38] related the proportionality coefficient to the local wing dimensions, wing-structure shape, and the wing-structure material. Thus, the development given by Phillips et al. [12] includes the effects of the wing-structure and the chord distribution. However, like Prandtl [14], Phillips et al. [12,38] limited their results to rectangular wings. In this paper, the work of Phillips et al. [12,38] will be extended analytically to identify expressions for the induced drag of non-rectangular wings, including the effects of the chord distribution on the wing structure. The new expressions will then be used to evaluate Prandtl's claim that tapered wings have an advantage over rectangular wings [14,15]. As will be shown, the results in this paper demonstrate that Prandtl's claim is indeed correct for elliptically-tapered and linearly-tapered planforms when the effects of planform shape are considered. Because it provides a foundation for the work presented in this paper, a brief review of the work of Phillips et al. [12,38] is given in the following section.

II. Analytical Foundation

Whereas Prandtl assumed that the wing bending moments are produced by the lift distribution alone, Phillips et al. [12] assumed that at a given load factor, n_a, the bending moments are caused by the lift distribution, the wing-structure weight distribution, $\widetilde{W}_s(z)$, and the distribution of the net weight of all non-structural components carried in the wing, $\widetilde{W}_n(z)$ [12], i.e.,

$$\widetilde{M}_b(z) = \int_{z'=z}^{b/2} \left[\widetilde{L}(z') - n_a \widetilde{W}_n(z') - n_a \widetilde{W}_s(z')\right](z'-z)dz', \quad \text{for } z \geq 0 \tag{4}$$

For wing-structure design, the bending moment must be evaluated at the design load factor for maneuvering flight, n_m, and the design load factor for a hard landing, n_g. In general, the integral in Eq. (4) must be evaluated numerically. However, it can be evaluated analytically if the weight distribution introduced by Phillips et al. [12] is used, i.e.,

$$\widetilde{W}_n(z) = (W - W_r)\frac{\widetilde{L}(z)}{L} - \widetilde{W}_s(z) \tag{5}$$

where W_r is the net weight of all non-structural components carried at the wing root. Equation (5) minimizes the bending moments from Eq. (4) when the weight carried at the root satisfies the condition [12]

$$W_r = \frac{n_g - 1}{n_m + n_g} W \tag{6}$$

Using Eq. (5) in Eq. (4), Phillips et al. [12] found that the bending-moment distribution reduces to a form that is proportional to that used by Prandtl in 1933 [14], i.e.,

$$\left|\widetilde{M}_b(z)\right| = \kappa_W W_r \int_{z'=z}^{b/2} \frac{\widetilde{L}(z')}{L}(z'-z)dz', \quad \text{for } z \geq 0 \tag{7}$$

where

$$\kappa_W = \begin{cases} n_m & W_r \geq \dfrac{n_g - 1}{n_m + n_g} W \\ (n_g - 1)\dfrac{W}{W_r} - n_g & W_r \leq \dfrac{n_g - 1}{n_m + n_g} W \end{cases} \tag{8}$$

Like Prandtl, Phillips et al. [12] also assumed that the bending moments are related to the wing-structure weight by a proportionality coefficient. However, unlike Prandtl, they did not assume that the

proportionality coefficient is spanwise invariant for all planforms. Instead, Phillips et al. [12] defined the proportionality coefficient in terms of the local chord, $c(z)$, and the beam properties, i.e.,

$$W_s = 2\int_{z=0}^{b/2} \frac{\left|\widetilde{M}_b(z)\right|}{S_b(z)}dz; \quad S_b(z) \equiv \frac{C_\sigma (t_{\max}/c)c(z)\sigma_{\max}}{\gamma}, \quad C_\sigma \equiv \frac{2I(h/t_{\max})}{Ah^2} \tag{9}$$

Although Eq. (9) can be used for any planform shape, in order to obtain analytic results, Phillips et al. [12,38] considered only the rectangular planform, for which S_b is constant. Using Eqs. (4) and (5) in Eq. (9), and assuming a rectangular planform with an all-positive, spanwise-symmetric lift distribution, the integral in Eq. (9) can be evaluated to give the total wing-structure weight

$$W_s = \frac{\kappa_W W_r b^2}{32 S_b}(1+B_3) \tag{10}$$

Using the relation given in Eq. (6) and the definition for S_b from Eq. (9), Eq. (10) can be solved for the wingspan to give [38]

$$b = \sqrt[3]{\frac{16C_\sigma (t_{\max}/c)\,\sigma_{\max} W_n}{(1+B_3)\gamma(W/S)} \frac{n_m + n_g}{n_m(n_g - 1)}} \tag{11}$$

Using Eqs. (10) and (11) in Eq. (2), along with the relation given in Eq. (6), gives the associated induced drag [38]

$$D_i = \frac{9}{2\pi\rho V_\infty^2}\left[\frac{(1+B_3)\gamma(W/S)W_n^2}{16C_\sigma (t_{\max}/c)\sigma_{\max}} \frac{n_m(n_g-1)}{n_m+n_g}\right]^{2/3}\left(1+\sum_{n=2}^{\infty} nB_n^2\right) \tag{12}$$

If the wing loading, W/S, is fixed, Phillips et al. [38] show that this induced drag is minimized with a lift distribution having $B_3 = -3/8 + \sqrt{9/64 - 1/12}$ and $B_n = 0$ for $n = 2$ and all $n > 3$. This lift distribution, along with Prandtl's 1933 lift distribution and other optimum lift distributions found by Phillips et al. [12,38] under different constraints, differ only in the value of B_3. All of the optimum lift distributions

found by Phillips et al. [12,38] have values for B_3 that fall somewhere between that of Prandtl's 1933 lift distribution ($B_3 = -1/3$) and that of the elliptic lift distribution ($B_3 = 0$). However, due to the constraints and assumptions imposed by Prandtl [14] and Phillips et al. [12,38], including that the proportionality coefficient, S_b, the chord, c, and the spar height, h, are not a function of z, their optimum solutions apply only to wings with a rectangular planform. In this paper, we will relax some of these assumptions and consider wings with non-rectangular planforms.

For non-rectangular wings, including tapered wings, the proportionality coefficient, S_b, as defined in Eq. (9), is a function of spanwise location. Therefore, for non-rectangular planforms, it is often convenient to rewrite the proportionality coefficient in terms of the mean geometric chord, $\bar{c} = S/b$, such that it remains spanwise invariant, i.e.,

$$S_b \equiv \frac{C_\sigma (t_{\max}/c)\bar{c}\,\sigma_{\max}}{\gamma} \tag{13}$$

This new definition for the proportionality coefficient will be used for the remainder of this paper. If Eq. (13) is used to define the proportionality coefficient, Eq. (9) must also be rewritten to give a new expression for the wing-structure weight that includes the effects of the chord distribution

$$W_s = 2\int_{z=0}^{b/2} \frac{\left|\widetilde{M}_b(z)\right|}{S_b\, c(z)/\bar{c}}\, dz \tag{14}$$

Note that, like Eq. (9), Eqs. (13)-(14) are valid for any arbitrary planform, provided that the planform is expressed using a chord distribution that can be integrated in z. However, in light of Prandtl's observations about tapered wings, in this paper, we will consider wings with elliptically-tapered and linearly-tapered planforms. For these planforms, integrating Eq. (14) gives closed-form expressions for the wing-structure weight that can be used to predict the induced drag and identify the lift distributions that minimize induced drag.

III. Elliptically-Tapered Planforms

For a wing with fixed wingspan and lift distribution, the elliptically-tapered planform (which will be referred to in this paper as the elliptic planform) is commonly accepted as the most efficient planform because it produces an elliptic lift distribution with no aerodynamic or geometric twist when immersed in a uniform flow. For a wing with an elliptic planform, the normalized chord distribution can be written as

$$\frac{c(z)}{\bar{c}} = \frac{4}{\pi}\sqrt{1-(2z/b)^2} \tag{15}$$

where $\bar{c}$ is the mean geometric chord, and is given by

$$\bar{c} = \frac{2}{b}\int_{z=0}^{b/2}\sqrt{1-(2z/b)^2}\,dz = \frac{\pi}{4}c_r \tag{16}$$

Note that the dimensionless chord distribution given in Eq. (15) depends only on the wingspan. Equation (15) can be used in Eq. (14) to give the wing-structure weight required to support the bending moments on a wing with an elliptic planform and any fixed all-positive spanwise-symmetric lift distribution

$$W_s = \frac{\pi}{2S_b}\int_{z=0}^{b/2}\frac{\left|\widetilde{M}_b(z)\right|}{\sqrt{1-(2z/b)^2}}\,dz \tag{17}$$

where the proportionality coefficient is defined in Eq. (13). If the weight distribution from Eq. (5) is used, the moment distribution from Eq. (7) can be used in Eq. (17) to give

$$W_s = \frac{\pi\kappa_W W_r}{2S_b}\int_{z=0}^{b/2}\int_{z'=z}^{b/2}\frac{\widetilde{L}(z')}{L\sqrt{1-(2z/b)^2}}(z'-z)\,dz'\,dz \tag{18}$$

Equation (18) is written in terms of the spanwise coordinate z. However, in order to evaluate the integral in Eq. (18), the change of variables from Eq. (1) can be used to rewrite the integrand in a more convenient

form. Using the change of variables and the lift distribution from Eq. (1), along with the trigonometric identity $\sin(2\theta) = 2\cos(\theta)\sin(\theta)$, Eq. (18) becomes

$$W_s = \frac{\kappa_W W_r b^2}{8S_b} \sum_{n=1}^{\infty} B_n \int_{\theta=\pi/2}^{\pi} \int_{\theta'=\theta}^{\pi} \sin(n\theta') \left[\frac{\sin(2\theta)}{\sin\theta} \sin\theta' - \sin(2\theta') \right] d\theta' \, d\theta \tag{19}$$

In this form, the inner integral in Eq. (19) can be evaluated analytically using the relations

$$\int_{\theta'=\theta}^{\pi} \sin(n\theta')\sin(\theta')d\theta' = \begin{cases} \dfrac{\pi}{2} - \dfrac{\theta}{2} + \dfrac{\sin(2\theta)}{4}, & \text{for } n = 1 \\ \dfrac{\sin[(n+1)\theta]}{2(n+1)} - \dfrac{\sin[(n-1)\theta]}{2(n-1)}, & \text{for } n \neq 1 \end{cases} \tag{20}$$

and

$$\int_{\theta'=\theta}^{\pi} \sin(n\theta')\sin(2\theta')d\theta' = \begin{cases} \dfrac{\pi}{2} - \dfrac{\theta}{2} + \dfrac{\sin(4\theta)}{8}, & \text{for } n = 2 \\ \dfrac{\sin[(n+2)\theta]}{2(n+2)} - \dfrac{\sin[(n-2)\theta]}{2(n-2)}, & \text{for } n \neq 2 \end{cases} \tag{21}$$

which, when used in Eq. (19), give

$$\begin{aligned} W_s = \frac{\kappa_W W_r b^2}{8S_b} \Bigg\{ & \int_{\theta=\pi/2}^{\pi} \frac{[2\pi - 2\theta + \sin(2\theta)]\sin(2\theta)}{4\sin\theta} d\theta - \int_{\theta=\pi/2}^{\pi} \frac{\sin(3\theta) - 3\sin\theta}{6} d\theta \\ & + B_3 \int_{\theta=\pi/2}^{\pi} \frac{[\sin(4\theta) - 2\sin(2\theta)]\sin(2\theta)}{8\sin\theta} d\theta - B_3 \int_{\theta=\pi/2}^{\pi} \frac{\sin(5\theta) - 5\sin\theta}{10} d\theta \\ & + \sum_{n=5}^{\infty} B_n \int_{\theta=\pi/2}^{\pi} \frac{(n-1)\sin[(n+1)\theta] - (n+1)\sin[(n-1)\theta]\sin(2\theta)}{2(n^2-1)\sin\theta} d\theta \\ & - \sum_{n=5}^{\infty} B_n \int_{\theta=\pi/2}^{\pi} \frac{(n-2)\sin[(n+2)\theta] - (n+2)\sin[(n-2)\theta]}{2(n^2-4)} d\theta \Bigg\} \end{aligned} \tag{22}$$

Each of the integrals in Eq. (22) can be evaluated analytically. Carrying out the integration gives a closed-form expression for the wing-structure weight required to support the bending moments on a wing with an elliptic planform, the weight distribution given by Eq. (5), and any all-positive spanwise-symmetric lift distribution

$$W_s = \frac{\kappa_W W_r b^2}{8 S_b}\left\{\frac{16}{9} - \frac{\pi}{2} + \sum_{n=3}^{\infty}\left[\frac{16}{n(n^2-4)^2}\right] B_n\right\} \tag{23}$$

For convenience, Eq. (23) can be rewritten in the form

$$W_s = \frac{\kappa_W W_r b^2}{8 S_b}\left\{C_1 + \sum_{n=3}^{\infty} C_n B_n\right\} \tag{24}$$

where

$$C_n \equiv \begin{cases} \dfrac{16}{9} - \dfrac{\pi}{2}, & \text{for } n = 1 \\ \dfrac{16}{n(n^2-4)^2}, & \text{for } n \neq 1 \end{cases} \tag{25}$$

Note that whereas the wing-structure weight given in Eq. (10) for a rectangular wing is a function of only the third Fourier coefficient, B_3, Eq. (24) shows that for the elliptic planform with an all-positive spanwise-symmetric lift distribution, the wing-structure weight is dependent on all odd Fourier coefficients. However, it is also important to note from Eq. (25) that as n increases, the coefficient, C_n, for each n scales roughly as $1/n^5$, meaning that as n increases, the influence of the Fourier coefficient on the wing-structure weight decreases.

Because the wing loading, W/S, is often fixed by airspeed requirements, it is also sometimes convenient to rewrite Eq. (24) in terms of the wing loading. This can be done by using the relation $\bar{c} = S/b$ and Eq. (13) in Eq. (24) to give

$$W_s = \frac{\gamma(W/S)}{8 C_\sigma (t_{\max}/c)\sigma_{\max}} \frac{\kappa_W W_r b^3}{W}\left(C_1 + \sum_{n=3}^{\infty} C_n B_n\right) \tag{26}$$

Equation (26) can be used to find the weight of the wing structure required for a wing having an elliptic planform with fixed wing loading and fixed gross weight. The corresponding wingspan can be found by rearranging Eq. (26) to give

$$b = \sqrt[3]{\frac{8C_\sigma (t_{\max}/c)\sigma_{\max}}{\gamma(W/S)\left(C_1 + \sum_{n=3}^{\infty} C_n B_n\right)} \frac{WW_s}{\kappa_W W_r}} \tag{27}$$

Using Eq. (27) in Eq. (2), the induced drag can be written as

$$D_i = \frac{2}{\pi\rho V_\infty^2}\left[\frac{\gamma(W/S)\left(C_1 + \sum_{n=3}^{\infty} C_n B_n\right)}{8C_\sigma (t_{\max}/c)\sigma_{\max}} \frac{\kappa_W W_r W^2}{W_s}\right]^{2/3} \left(1 + \sum_{n=3}^{\infty} n B_n^2\right) \tag{28}$$

The gross weight, W , is the sum of the wing-structure weight, W_s, and the net weight, W_n. Equation (28) requires that the gross weight and the wing-structure weight be known. However, in many cases, it is more useful to fix the net weight and allow the gross weight to vary. Applying the relationship given in Eq. (6) to minimize the bending moments, the gross weight can be eliminated from Eq. (26), and the wing-structure weight can be rewritten as

$$W_s = \frac{\gamma(W/S)b^3}{8C_\sigma (t_{\max}/c)\sigma_{\max}} \frac{n_m(n_g - 1)}{n_m + n_g}\left(C_1 + \sum_{n=3}^{\infty} C_n B_n\right) \tag{29}$$

Using the relation $W = W_n + W_s$ and Eq. (29) in Eq. (2), the induced drag can be written as

$$D_i = \frac{2}{\pi\rho V_\infty^2}\left[\frac{W_n}{b} + \frac{\gamma(W/S)\left(C_1 + \sum_{n=3}^{\infty} C_n B_n\right)}{8C_\sigma (t_{\max}/c)\sigma_{\max}} \frac{n_m(n_g - 1)}{n_m + n_g} b^2\right]^2 \left(1 + \sum_{n=3}^{\infty} n B_n^2\right) \tag{30}$$

The wingspan that minimizes the induced drag can be found by differentiating Eq. (30) with respect to the wingspan and setting the result equal to zero. This gives

$$b = \sqrt[3]{\frac{4W_n C_\sigma (t_{\max}/c)\sigma_{\max}}{\gamma(W/S)\left(C_1 + \sum_{n=3}^{\infty} C_n B_n\right)} \frac{(n_m + n_g)}{n_m (n_g - 1)}} \tag{31}$$

Using Eq. (31) in Eq. (29) gives the wing-structure weight that minimizes the induced drag

$$W_s = \frac{1}{2} W_n \tag{32}$$

Note that Eq. (32) matches the result found by Phillips et al. in [38] for the wing-structure weight that minimizes induced drag for a stress-limited rectangular wing with fixed wing loading. Thus, although Eq. (14) shows that the wing-structure weight is, in general, a function of the chord distribution, the optimum total wing-structure weight, as a percent of the net weight, is the same for wings with elliptic planforms and rectangular planforms. Using Eqs. (31) and (32) in Eq. (30) gives the minimum induced drag

$$D_i = \frac{9}{2\pi\rho V_\infty^2} \left[\frac{\left(C_1 + \sum_{n=3}^{\infty} C_n B_n\right)\gamma(W/S) W_n^2}{4 C_\sigma (t_{\max}/c)\sigma_{\max}} \frac{n_m (n_g - 1)}{n_m + n_g} \right]^{2/3} \left(1 + \sum_{n=3}^{\infty} n B_n^2\right) \tag{33}$$

Because Eq. (33) is in the same form as Eq. (12), the minimum induced drag produced by a wing with an elliptic planform and a given lift distribution can be easily compared to the minimum induced drag produced by a rectangular wing with the same lift distribution by defining an induced-drag ratio

$$R_{D_e} \equiv \frac{(D_i)_{\text{elliptic}}}{(D_i)_{\text{rectangular}}} = \left[\frac{4}{(1 + B_3)} \left(C_1 + \sum_{n=3}^{\infty} C_n B_n \right) \right]^{2/3} \tag{34}$$

Similarly, comparing Eqs. (31) and (11) gives a wingspan ratio

$$R_{b_e} \equiv \frac{b_{\text{elliptic}}}{b_{\text{rectangular}}} = \sqrt[3]{\frac{1+B_3}{4\left(C_1 + \sum_{n=3}^{\infty} C_n B_n\right)}} \tag{35}$$

Because Eq. (34) is a function of an infinite number of Fourier coefficients, in general, minimizing Eq. (34) requires the use of numerical methods. However, Eq. (25) shows that the influence of each Fourier coefficient decreases as n increases. This means that Eqs. (34) and (35) are most heavily influenced by the coefficient B_3. Therefore, it is useful to consider the case where $B_n = 0$ for all $n > 3$. Figure 11 shows how Eqs. (34) and (35) vary with B_3 for this special case. Note that results are only shown for $-1/3 \le B_3 \le 0$. This is because for $B_3 < -1/3$, the section lift becomes negative near the wingtips. As seen in Eq. (18), the wing-structure weight is proportional to the integral of the lift distribution. Negative lift near the wingtips would result in zero integrated lift and zero wing-structure weight at some inboard location, which is not physically valid. Therefore, in this paper, we assume that the lift distribution is all positive.

Recall that each value of B_3 in Fig. 11 corresponds to a different lift distribution. For any given planform, any lift distribution can be obtained using wing twist. The elliptic planform is commonly considered the most efficient planform because it produces the elliptic lift distribution with no twist. However, Fig. 11 shows that when structural effects are included, the elliptic planform produces less induced drag than the rectangular planform not only for the elliptic lift distribution, but for all of the lift distributions shown in Fig. 11. The induced-drag ratio is minimized, and the wingspan ratio is maximized, using $B_3 = -1/3$, which corresponds to Prandtl's 1933 lift distribution. This supports Prandtl's claim that "tapered wings have an advantage over those with a nearly rectangular profile" [14,15],

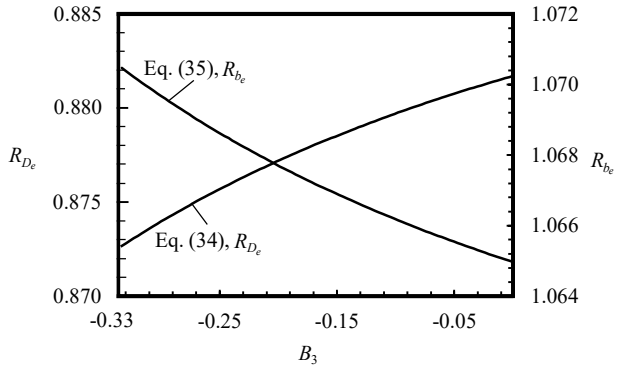


Fig. 1 Ratio of the induced drag and corresponding wingspan produced by an elliptic planform to those produced by a rectangular planform, plotted as a function of B_3.

especially when Prandtl's 1933 lift distribution is used. Using this lift distribution, Eqs. (34) and (35) show that the elliptic planform produces up to 12.73% less induced drag than the rectangular planform with a wingspan increase of 7.05%. However, although the induced-drag ratio in Eq. (34) is minimized using Prandtl's 1933 lift distribution, this lift distribution is not the same as the lift distribution that minimizes the induced drag given in Eq. (33). As will be shown later, the optimum lift distribution that gives an absolute minimum in induced drag for the elliptic planform has a B_3 value that falls somewhere between that of the elliptic lift distribution and that of Prandtl's 1933 lift distribution.

IV. Linearly-Tapered Planforms

The process that was shown in the previous section for elliptic planforms can be repeated for wings with linearly-tapered planforms. Linearly-tapered wings are commonly used on modern aircraft as a tradeoff between efficiency and ease of manufacture. Consider a linearly-tapered wing with tip chord c_t and root chord c_r. The wing taper ratio is defined as $R_T \equiv c_t / c_r$, and the normalized chord distribution can be written in terms of the taper ratio as

$$\frac{c(z)}{\bar{c}} = \frac{2\left[1-(1-R_T)\left|2z/b\right|\right]}{1+R_T} \tag{36}$$

where $\bar{c}$ is simply the average of the root chord and the tip chord, and can be expressed in terms of the root chord and the taper ratio as

$$\bar{c} = \frac{c_r}{2}(1 + R_T) \tag{37}$$

Using Eq. (36) in Eq. (14), along with the definition of S_b given in Eq. (13), the wing-structure weight required to support the bending moments on a wing with a linearly-tapered planform becomes

$$W_s = \frac{(1 + R_T)}{S_b} \int_{z=0}^{b/2} \frac{\left|\widetilde{M}_b(z)\right|}{1 - (1 - R_T)\left|2z/b\right|} dz \tag{38}$$

If the weight distribution given by Eq. (5) is used, the bending-moment distribution given by Eq. (7) can be used in Eq. (38) to give

$$W_s = \frac{\kappa_W W_r (1 + R_T)}{S_b} \int_{z=\pi/2}^{b/2} \int_{z'=z}^{b/2} \frac{\widetilde{L}(z')}{L\left[1 - (1 - R_T)\left|2z/b\right|\right]} (z' - z) dz' dz \tag{39}$$

For convenience, Eq. (39) can be rewritten in a form similar to Eq. (19) using the lift distribution and change of variables from Eq. (1). This gives

$$W_s = \frac{\kappa_W W_r b^2 (1 + R_T)}{4\pi S_b} \sum_{n=0}^{\infty} B_n \int_{\theta=\pi/2}^{\pi} \int_{\theta'=\theta}^{\pi} \frac{\sin(n\theta')\left[\sin(2\theta)\sin\theta' - \sin(2\theta')\sin\theta\right]}{1 + (1 - R_T)\cos\theta} d\theta' d\theta \tag{40}$$

Using the integral identities given in Eqs. (20) and (21), the inner integral from Eq. (40) can be evaluated analytically. For any spanwise-symmetric lift distribution, the result of these evaluations can be written as

$$W_s = \frac{\kappa_W W_r b^2 (1+R_T)}{4\pi S_b}\left\{ \int_{\theta=\pi/2}^{\pi} \frac{\left[2\pi - 2\theta + \sin(2\theta)\right]\sin(2\theta)}{4\left[1+(1-R_T)\cos\theta\right]} d\theta - \int_{\theta=\pi/2}^{\pi} \frac{\left[\sin(3\theta) - 3\sin\theta\right]\sin\theta}{6\left[1+(1-R_T)\cos\theta\right]} d\theta \right.$$

$$+ B_3 \int_{\theta=\pi/2}^{\pi} \frac{\left[\sin(4\theta) - 2\sin(2\theta)\right]\sin(2\theta)}{8\left[1+(1-R_T)\cos\theta\right]} d\theta - B_3 \int_{\theta=\pi/2}^{\pi} \frac{\left[\sin(5\theta) - 5\sin\theta\right]\sin\theta}{10\left[1+(1-R_T)\cos\theta\right]} d\theta \tag{41}$$

$$+ B_n \int_{\theta=\pi/2}^{\pi} \frac{(n-1)\sin[(n+1)\theta] - (n+1)\sin[(n-1)\theta]}{2(n^2-1)\left[1+(1-R_T)\cos\theta\right]} \sin(2\theta) d\theta$$

$$\left. - B_n \int_{\theta=\pi/2}^{\pi} \frac{(n-2)\sin[(n+2)\theta] - (n+2)\sin[(n-2)\theta]}{2(n^2-4)\left[1+(1-R_T)\cos\theta\right]} \sin\theta d\theta \right\}$$

In general, the complexity of the integrals in Eq. (41) prohibits any simple analytical evaluation. However, for the specific case of a rectangular wing ($R_T = 1$), the integrals can be evaluated analytically, and Eq. (41) reduces to

$$W_s = \frac{\kappa_W W_r b^2}{2\pi S_b}\left(\frac{\pi}{16} + \frac{\pi}{16} B_3\right) \tag{42}$$

which is equivalent to Eq. (10). Note that for a rectangular wing, $\bar{c} = c(z)$, and the definition of S_b given in Eq. (13) is the same as that given in Eq. (9).

Here, again, it is convenient to rewrite Eq. (41) in terms of the coefficients C_n

$$W_s = \frac{\kappa_W W_r b^2 (1+R_T)}{4\pi S_b}\left(C_1 + \sum_{n=3}^{\infty} C_n B_n\right) \tag{43}$$

where

$$C_n = \begin{cases} \displaystyle\int_{\theta=\pi/2}^{\pi} \frac{\left[2\pi - 2\theta + \sin(2\theta)\right]\sin(2\theta)}{4\left[1+(1-R_T)\cos\theta\right]}d\theta - \int_{\theta=\pi/2}^{\pi} \frac{\left[\sin(3\theta) - 3\sin\theta\right]\sin\theta}{6\left[1+(1-R_T)\cos\theta\right]}d\theta, & \text{for } n=1 \\ \displaystyle\int_{\theta=\pi/2}^{\pi} \frac{(n-1)\sin[(n+1)\theta] - (n+1)\sin[(n-1)\theta]}{2(n^2-1)\left[1+(1-R_T)\cos\theta\right]}\sin(2\theta)d\theta & \\ \quad \displaystyle - \int_{\theta=\pi/2}^{\pi} \frac{(n-2)\sin[(n+2)\theta] - (n+2)\sin[(n-2)\theta]}{2(n^2-4)\left[1+(1-R_T)\cos\theta\right]}\sin\theta d\theta & \text{for } n \neq 1 \end{cases}, \tag{44}$$

Note that whereas the coefficients, C_n, for the elliptic planform can be found analytically, the coefficients given in Eq. (44) must be found numerically. This can be done using any high-order integration scheme. Because numerical integration techniques require that the integrand be evaluated at the limits of integration, special care must be taken in the case of a wing with a triangular planform ($R_T = 0$). In this case, each of the integrands in Eq. (44) is indeterminate when evaluated at $\theta = \pi$. Thus, L'hospital's rule can be used to evaluate the limit of each integrand as θ approaches π. Applying L'hospital's rule twice to each of the integrands in Eq. (44) gives

$$\lim_{\theta\to\pi} C_n = 0 \tag{45}$$

Using Simpson's rule and a step size of $d\theta = 1.5\times10^{-4}$, solutions to Eq. (44) were obtained for taper ratios in the range $0 \le R_T \le 1$. The resulting values of C_n for all odd n in the range $1 \le n \le 29$ are given in the appendix, along with closed-form expressions that can be used to approximate C_n as a function of taper ratio in the range $0.2 \le R_T \le 1.5$.

If the wing loading is fixed, Eq. (43) can be rewritten in terms of the gross weight using Eq. (13) and the relation $\bar{c} = S/b$ to give

$$W_s = \frac{\gamma(W/S)(1+R_T)}{4\pi C_\sigma (t_{\max}/c)\sigma_{\max}} \frac{\kappa_W W_r b^3}{W}\left(C_1 + \sum_{n=3}^{\infty} C_n B_n\right) \tag{46}$$

Rearranging Eq. (46) gives the wingspan allowed by a given lift distribution and wing-structure weight on a wing with a linearly-tapered planform with fixed weight and wing loading

$$b = \sqrt[3]{\frac{4\pi C_\sigma (t_{\max}/c)\sigma_{\max}}{\gamma(1+R_T)(W/S)\left(1+\sum_{n=3}^{\infty} C_n B_n\right)} \frac{WW_s}{\kappa_W W_r}} \tag{47}$$

Using Eq. (47) in Eq. (2) gives the induced drag

$$D_i = \frac{2}{\pi\rho V_\infty^2}\left[\frac{(1+R_T)(W/S)\left(C_1+\sum_{n=3}^{\infty} C_n B_n\right)\gamma}{4\pi C_\sigma (t_{\max}/c)\sigma_{\max}} \frac{\kappa_W W_r W^2}{W_s}\right]^{2/3}\left(1+\sum_{n=3}^{\infty} nB_n^2\right) \tag{48}$$

If the gross weight is not known, the weight constraint from Eq. (6) can be used in Eq. (46) to give

$$W_s = \frac{\gamma(W/S)(1+R_T)b^3}{4\pi C_\sigma (t_{\max}/c)\sigma_{\max}} \frac{n_m(n_g-1)}{n_m+n_g}\left(C_1+\sum_{n=3}^{\infty} C_n B_n\right) \tag{49}$$

Using the relation $W = W_s + W_n$ and Eq. (49) in Eq. (2) gives the induced drag

$$D_i = \frac{2}{\pi\rho V_\infty^2}\left[\frac{W_n}{b} + \frac{\gamma(W/S)(1+R_T)\left(C_1+\sum_{n=3}^{\infty} C_n B_n\right)}{4\pi C_\sigma (t_{\max}/c)\sigma_{\max}} \frac{n_m(n_g-1)}{n_m+n_g} b^2\right]^2 \left(1+\sum_{n=3}^{\infty} nB_n^2\right) \tag{50}$$

The wingspan that minimizes Eq. (50) is

$$b = \sqrt[3]{\frac{2\pi W_n C_\sigma (t_{\max}/c)\sigma_{\max}}{(1+R_T)\gamma(W/S)\left(C_1+\sum_{n=3}^{\infty} C_n B_n\right)} \frac{(n_m+n_g)}{n_m(n_g-1)}} \tag{51}$$

When used in Eq. (49), this wingspan gives the wing-structure weight that minimizes induced drag for a linearly-tapered wing with fixed wing loading

$$W_s = \frac{1}{2} W_n \tag{52}$$

which is the same result as that shown in Eq. (32). Thus we see that for the planforms considered in this paper, the optimum wing-structure weight is always $W_n/2$, independent of the planform shape. Using Eq. (51) in Eq. (50) gives the minimum induced drag

$$D_i = \frac{9}{2\pi\rho V_\infty^2}\left[\frac{(1+R_T)\left(C_1+\sum_{n=3}^{\infty}C_nB_n\right)\gamma(W/S)W_n^2}{2\pi C_\sigma(t_{\max}/c)\sigma_{\max}}\frac{n_m(n_g-1)}{n_m+n_g}\right]^{2/3}\left(1+\sum_{n=3}^{\infty}nB_n^2\right) \tag{53}$$

Note that for a rectangular wing, Eq. (53) reduces to the same result given in Eq. (12) for the stress-limited design of a rectangular wing with fixed wing loading and fixed net weight. However, to compare the minimum induced drag produced by linearly-tapered planforms having $R_T \neq 1$ to that produced by the rectangular planform, it is convenient to define an induced-drag ratio for linearly-tapered wings, i.e.,

$$R_{D_t} \equiv \frac{(D_i)_{\text{tapered}}}{(D_i)_{\text{rectangular}}} = \left[\frac{8(1+R_T)}{\pi(1+B_3)}\left(C_1+\sum_{n=3}^{\infty}C_nB_n\right)\right]^{2/3} \tag{54}$$

and a wingspan ratio for linearly-tapered wings, i.e.,

$$R_{b_t} \equiv \frac{b_{\text{tapered}}}{b_{\text{rectangular}}} = \left[\frac{\pi(1+B_3)}{8(1+R_T)\left(C_1+\sum_{n=3}^{\infty}C_nB_n\right)}\right]^{1/3} \tag{55}$$

Note that when the values for C_n and R_T for a rectangular wing are used in Eqs. (54) and (55), the induced-drag ratio and the wingspan ratio reduce to $R_{b_t} = R_{D_t} = 1$. Fully minimizing Eq. (54) requires the use of numerical methods. However, the solutions to Eq. (44) show that, like Eq. (34), Eq. (54) is most heavily influenced by B_3. Therefore, we again consider the case where $B_n = 0$ for all $n > 3$. For this special case, the variation in Eq. (54) with B_3 is shown in Fig. 2 for several linearly-tapered planforms having $0 \le R_T \le 1$. The results from Eq. (34) are also included for reference. The variation in Eq. (55) with B_3 is shown in Fig. 3 for the same range of taper ratios, along with results from Eq. (35). Again, we assume that the lift distribution is all positive. Therefore, results are only shown for $-1/3 \le B_3 \le 0$.

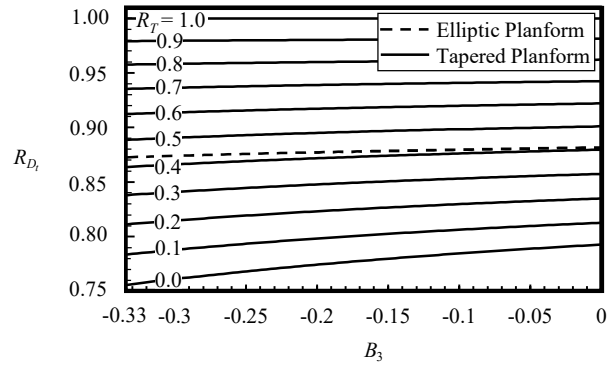


Fig. 2 Ratio of the induced drag produced by the elliptic planform and linearly-tapered planforms with $0 \leq R_T \leq 1$ to that produced by a rectangular planform, plotted as a function of B_3.

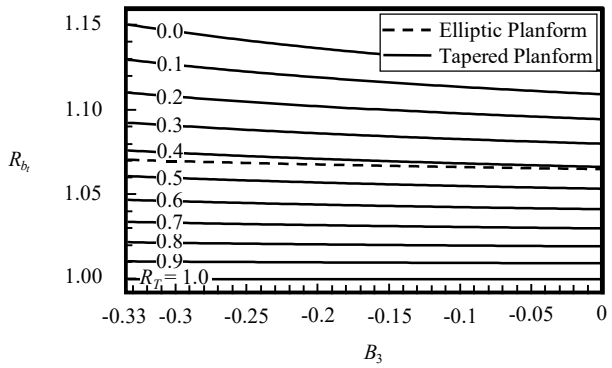


Fig. 3 Ratio of the allowable wingspan for the elliptic planform and linearly-tapered planforms with $0 \leq R_T \leq 1$ to the allowable wingspan for rectangular planform, plotted as a function of B_3.

As was true for the elliptic planform, the lift distribution that minimizes the induced-drag ratio and maximizes the wingspan ratio for all of the linearly-tapered planforms shown in Figs. 2 and 3 is Prandtl's 1933 lift distribution ($B_3 = -1/3$). However, it is interesting to note that as taper ratio decreases, the degree to which B_3 influences the induced-drag ratio and wingspan ratio increases. Therefore, for this case, the induced-drag ratio is minimized, and the wingspan ratio is maximized, using a triangular wing with $R_T = 0$. When compared to the elliptic planform, this planform produces up to 11.71% less induced drag. Equations (54) and (55) show that when compared to the rectangular planform, the triangular planform can reduce induced drag by up to 24.44%, with a wingspan increase of 15.04%. Thus, Prandtl's argument that

"tapered wings have an advantage over those with a nearly rectangular profile" [14,15] holds when the effects of taper on wing-structure weight are taken into account for all of the lift distributions considered here, with the greatest advantage at $B_3 = -1/3$. However, the reader is reminded that although the lift distribution having $B_3 = -1/3$ minimizes Eq. (54), it may not be the lift distribution that gives an absolute minimum in induced drag for the linearly-tapered planform. This lift distribution is discussed further in the following section.

V. Optimum Lift Distributions

Equations (28) and (48) give the induced drag for a wing with fixed gross weight and wing-structure weight, and Eqs. (33) and (53) give the induced drag for a wing with fixed net weight. However, it is important to note that Eqs. (33) and (53) were obtained under the assumption that the wing-structure weight satisfies Eqs. (32) and (52). Thus, because gross weight is the sum of the wing-structure weight and the net weight, minimizing induced drag under the constraint of fixed net weight is, in effect, the same as minimizing induced drag for a wing with fixed gross weight and the wing-structure weight set at one-half the net weight. Therefore, in this section, we will only consider the constraint of fixed net weight.

Under this constraint, Eqs. (33) and (53) give the minimum induced drag for wings with elliptic and linearly-tapered planforms, respectively, given a known lift distribution and fixed wing loading. However, neither equation produces an absolute minimum in induced drag unless the optimum lift distribution is also used. Because the wing-structure weight and induced drag are both functions of all the Fourier coefficients that define the lift distribution, in general, the optimum lift distribution is also a function of all the Fourier coefficients. Therefore, to find the lift distribution that minimizes induced drag, the Fourier series must be truncated at a finite value of n, and a numerical optimization framework must be employed. For example, Fig. 4 shows the lift distributions that minimize Eqs. (33) and (53) for the elliptic planform and for linearly-tapered planforms with $0 \le R_T \le 1$. Each lift distribution was found using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) [40-43] method with the Fourier coefficients, B_n, for all odd n in the range $1 \le n \le 29$. The elliptic lift distribution, Prandtl's 1933 lift distribution [14], and the lift distribution produced by an

untwisted linearly-tapered wing with aspect ratio 8 and $R_T = 0$ are also included for reference in Fig. 4. The values of B_n that correspond to the optimum lift distributions for wings having the elliptic planform and linearly-tapered planforms with $0 \le R_T \le 1$ are given in the appendix. Note that because we only wish to consider spanwise-symmetric lift distributions, each even Fourier coefficient is identically zero.

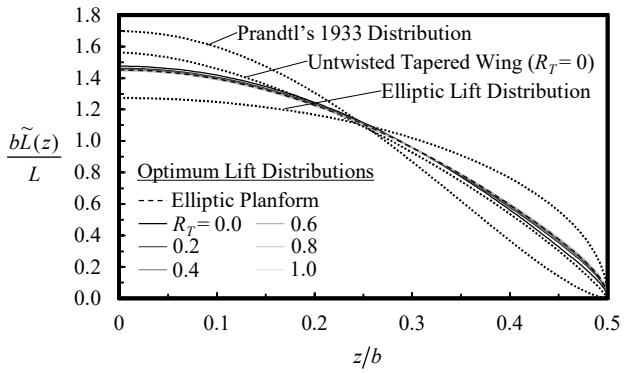


Fig. 4 Solutions for the lift distributions that minimize induced drag for an elliptic planform and linearly-tapered planforms with $0 \le R_T \le 1$.

Figure 4 shows that for each of the planforms considered here, including the elliptic planform, the optimum lift distribution takes neither a bell shape nor an elliptic shape. Instead, each of the optimum lift distributions takes a shape that is somewhat similar to the lift distribution produced by an untwisted linearly-tapered wing with $R_T = 0$. This agrees with Prandtl's observation that the lift distribution that minimizes induced drag is not elliptic but corresponds more closely to that produced by tapered wings [14,15]. It is also interesting to note that for the planforms considered here, the optimum lift distribution is only a weak function of planform shape.

The optimum lift distributions shown in Fig. 4 were obtained by truncating the Fourier series in Eq. (1) at $n = 29$. Because the lift distributions that truly minimize Eqs. (33) and (53) are a function of an infinite number of Fourier coefficients, the results shown in Fig. 4 are only an approximation. However, as seen in Eqs. (25) and (44), as n increases, the magnitude of the coefficients, C_n, for wings with elliptic and linearly-tapered planforms decrease. This means that the relative influence of each Fourier coefficient on the wing-structure weight, wingspan, and induced drag also decreases as n increases. An example of this is given in

Fig. 55, which shows the percent change in induced drag caused by including Fourier coefficients up to $n = 29$ in the definition for the optimum lift distribution. The percent change in induced drag shown in Fig. 55 at point *n* is the percent change between the induced drag obtained using *n* Fourier coefficients and the induced drag obtained by including up to $n - 2$ Fourier coefficients. For example, for the data point at $n = 7$, the percent change in induced drag is the percent change between the induced drag obtained by including coefficients up to $n = 7$ in the optimum lift distribution and the induced drag obtained by including coefficients up to $n = 5$ in the optimum lift distribution.

Note that as *n* increases, the effect of the corresponding Fourier coefficient on the induced drag decreases, as expected. With as few as two Fourier coefficients ($n = 5$), the percent change in induced drag drops below 0.1% for all the planforms shown. This suggests that lift distributions at or near the optimum lift distribution are dominated by B_3 and can be described using B_3 alone with little loss in accuracy. Figure 6 shows the value of B_3 in the optimum lift distribution for the elliptic planform and linearly-tapered planforms with $0 \le R_T \le 1$. Note that all of the B_3 values shown in Fig. 6 fall between that of the elliptic lift distribution ($B_3 = 0$) and that of Prandtl's 1933 lift distribution ($B_3 = -1/3$).

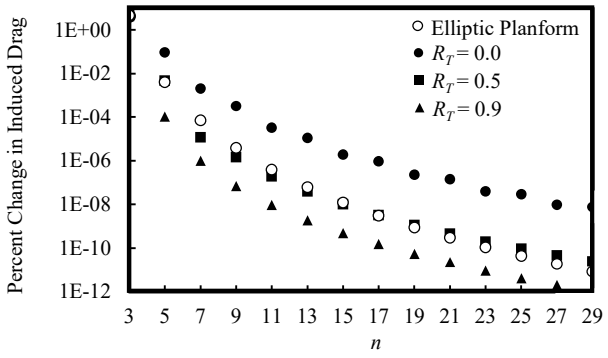


Fig. 5 Percent change in minimum induced drag resulting from including up to *n* Fourier coefficients in the solution for the optimum lift distribution.

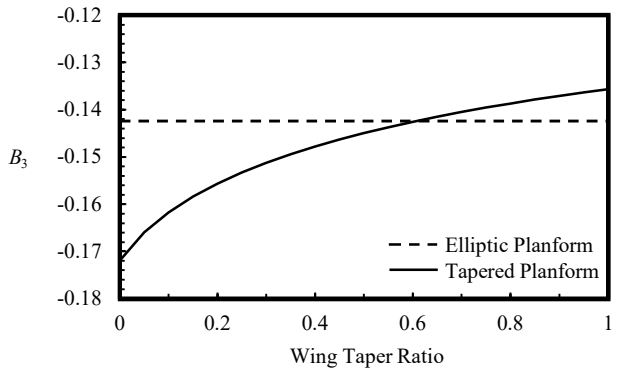


Fig. 6 B_3 values in the Fourier series defining the optimum lift distributions for the elliptic planform and linearly-tapered planforms with $0 \le R_T \le 1$.

Although the optimum lift distributions shown in Fig. 4 are all very similar in shape, Fig. 6 shows that the value of B_3 for each lift distribution exhibits a well-defined trend. As taper ratio decreases, minimum induced drag is obtained with lower values of B_3. In general, as the value of B_3 decreases, the lift distribution is shifted more toward the root, which means that at low taper ratios, slightly more lift is shifted inboard than at taper ratios near $R_T = 1$.

For an elliptic planform, the optimum wingspan and minimum possible induced drag are obtained using Eqs. (31) and (33) with the optimum lift distribution for the elliptic planform. For linearly-tapered planforms, the optimum wingspan and the minimum possible induced drag are obtained using Eqs. (51) and (53) with the optimum lift distribution for the linearly-tapered planform. When these values are compared to the wingspan and induced drag obtained using the elliptic lift distribution, the percent change in wingspan and induced drag depends on the planform shape. Figures 7 and 8 show the percent change in induced drag and wingspan, respectively, obtained using the optimum lift distribution compared to those obtained using a fixed elliptic lift distribution for the elliptic planform and linearly-tapered planforms with $0 \le R_T \le 1$.

The trends shown in Figs. 7 and 8 reveal that the effect of using the optimum lift distribution instead of the elliptic lift distribution on the minimum induced drag and corresponding wingspan is greater for wings with low taper ratios than for wings with nearly rectangular planforms. The most significant changes in induced drag and corresponding wingspan occur at a taper ratio of $R_T = 0$. At this taper ratio, the optimum

lift distribution allows a 7.63% increase in wingspan and a 5.94% reduction in induced drag over the elliptic lift distribution. Thus, it is shown that Prandtl's argument for tapered wings [14,15] holds not only when the lift distribution is fixed, but also when the lift distribution is optimized for each planform.

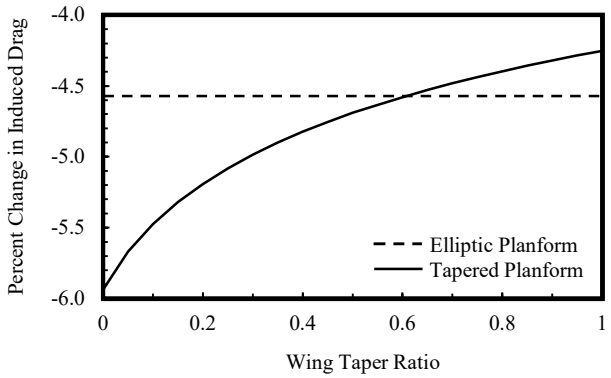


Fig. 7 Percent change in minimum induced drag produced by the optimum lift distribution compared to the elliptic lift distribution for the elliptic planform and linearly-tapered planforms with $0 \leq R_T \leq 1$.

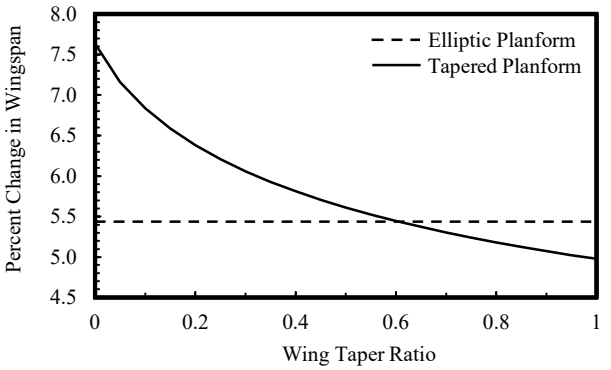


Fig. 8 Percent change in wingspan allowed by the optimum lift distribution compared to the elliptic lift distribution for the elliptic planform and linearly-tapered planforms with $0 \leq R_T \leq 1$.

For all of the results shown here, it is important to remember that the aspect ratio is not fixed. Because wing loading and gross weight are constant, the wing area must also be constant. At lower taper ratios, Eqs. (31) and (51) predict higher wingspans and, therefore, higher aspect ratios than at high taper ratios. Several examples of high-aspect-ratio, low-taper-ratio wings can be found in nature, particularly on high-endurance birds, such as the wandering albatross. Therefore, it should not be surprising that the results shown here

predict that minimum induced drag is obtained using a high-aspect-ratio, low-taper-ratio wing when structural constraints are considered.

VI. Example Results

As an example of minimizing induced drag for an elliptic or linearly-tapered planform, consider an aircraft with net weight fixed at $W_n = 7000$ lbf, wing loading fixed at $W/S = 30$, and the additional parameters $C_\sigma = 0.165$, $n_m = n_g = 3.75$, $t_{\max}/c = 0.12$, $\sigma_{\max} = 15\times10^3$ psi, $\gamma = 0.10$ lbf/in^3, $V_\infty = 200$ ft/s, and $\rho = 0.0023769$ slug/ft^3. The weight distributions given in Eqs. (5) and (6) are used to minimize the critical wing bending moments.

Minimum induced drag is obtained by using the optimum lift distributions from Fig. 4 in Eq. (33) for the elliptic planform and Eq. (53) for the linearly-tapered planform. Minimum induced-drag solutions for each of these cases are shown in black in Fig. 9. The wingspans that correspond to each of these cases are found by using the optimum lift distributions from Fig. 4 in Eqs. (31) and (51), and are shown in black in Fig. 10. For reference, the induced drag and corresponding allowable wingspan for the elliptic planform and linearly-tapered planform with a fixed elliptic lift distribution are also included in gray in Figs. 9 and 10.

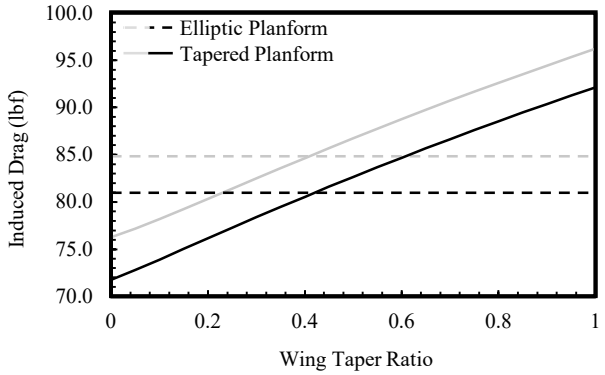


Fig. 9 Example minimum-induced-drag solutions for the elliptic planform and linearly-tapered planforms with $0 \le R_T \le 1$. Black: Optimum Lift Distribution, Gray: Elliptic Lift Distribution.

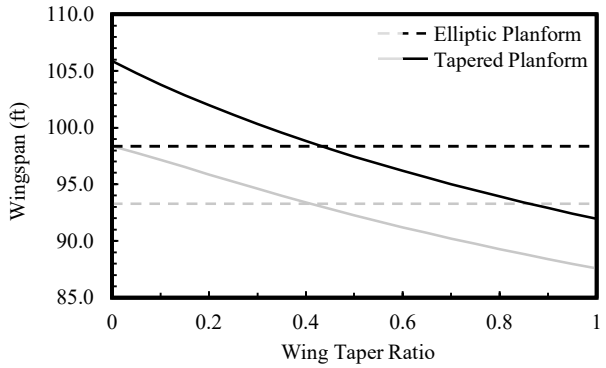


Fig. 10 Example wingspan solutions that correspond to the minimum-induced-drag solutions for the elliptic planform and linearly-tapered planforms with $0 \le R_T \le 1$. Black: Optimum Lift Distribution, Gray: Elliptic Lift Distribution.

Note that the elliptic planform gives results that closely match those obtained using a linearly-tapered planform with taper ratio just over $R_T = 0.4$. This agrees with traditional intuition based on classical lifting-line theory [44,45]. However, it is important to note that whereas the elliptic planform is commonly considered the optimum planform shape, the elliptic planform does not minimize induced drag for this case. Instead, induced drag is minimized for both the fixed elliptic lift distribution and the optimum lift distribution at a taper ratio of $R_T = 0$, which corresponds to a triangular wing. This agrees with the results shown in Figs. 2 and 7. Using the optimum lift distribution, the triangular planform allows a wingspan increase of 15.16% and an induced-drag reduction of 22.13% when compared to the rectangular planform. In contrast, using the optimum lift distribution, the elliptic planform gives a maximum reduction in induced drag of only 12.12% over the rectangular planform. Thus, for this example, the minimum induced drag is obtained with a linearly-tapered planform having a taper ratio of $R_T = 0$, with $D_i = 71.74617$ lbf at $B_3 = -0.17193$ and $b = 105.88820$ ft. The optimum wing-structure weight for this solution is $W_s = 3500$ lbf. Induced-drag contours around the minimum-induced-drag solution are shown in Fig. 1111 as a function of B_3 and wingspan.

Although the results presented in this section predict minimum induced drag at a taper ratio of $R_T = 0$, it is important to remember that wings with low taper ratios have low Reynolds numbers near the wingtips, which often cause the wing to stall first in these regions. For most aircraft, this can create serious handling problems, especially during stall recovery. For this reason, wings with low taper ratios are seldom used in

practice on aircraft. Nevertheless, the results presented here provide important insight into the aerodynamic and structural coupling in the stress-limited design of wings with elliptic and linearly-tapered planforms for minimum induced drag.

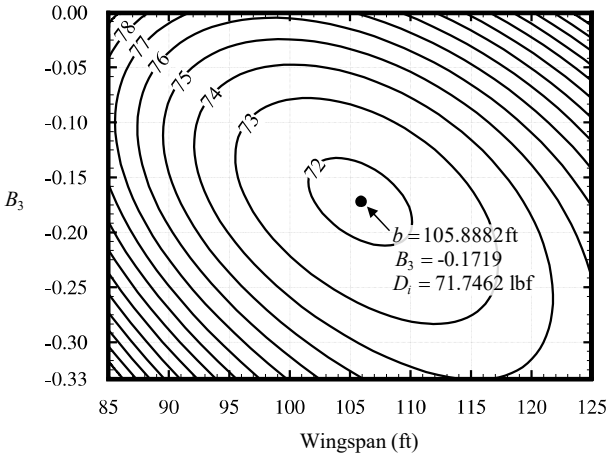


Fig. 11 Induced-drag contours around the minimum-induced drag solution for fixed net weight.

VII. Conclusions

From classical lifting-line theory, the induced drag on a wing in steady level flight is given by Eq. (2) and is a function of wingspan, wing weight, and lift distribution. For a given lift distribution, Eq. (2) is minimized by maximizing wingspan and/or minimizing wing weight. Increasing wingspan increases the required wing-structure weight, but certain non-elliptic lift distributions can alleviate bending moments near the wingtips, allowing an increase in wingspan with no increase in the wing-structure weight. However, any non-elliptic lift distribution incurs a penalty in induced drag. Thus, there exists an optimum wingspan and lift distribution that minimizes induced drag and depends on the tradeoff between wingspan, wing weight, and associated lift distribution.

In a 1933 paper, Prandtl noted that these optimum lift distributions correspond more closely to those produced by untwisted tapered wings than the elliptic lift distribution, giving tapered wings an advantage over nearly rectangular wings. However, Prandtl's mathematical development for finding the optimum wingspan and lift distribution that minimize induced drag was limited to rectangular wings. Here, we have

relaxed many of the assumptions used in Prandtl's development so that it applies to non-rectangular wings. Closed-form solutions for the optimum wingspan and corresponding induced drag are given in Eqs. (27) and (28), respectively, for the elliptic planform with fixed gross weight, Eqs. (31) and (33) for the elliptic planform with fixed net weight, Eqs. (47) and (48) for linearly-tapered planforms with fixed gross weight, and Eqs. (51) and (53) for linearly-tapered planforms with fixed net weight. If the net weight is fixed and gross weight is allowed to vary, there also exists an optimum wing-structure weight that is one-half the net weight for both elliptic and linearly-tapered planforms. Thus, if the optimum wing-structure weight is used, the constraint of fixed gross weight and the constraint of fixed net weight are equivalent.

Figure 2 shows that for a linearly-tapered planform with a fixed lift distribution having $-1/3 \le B_3 \le 0$ and $B_n = 0$ for all $n \ne 3$, the induced drag is always minimized using a triangular planform having $R_T = 0$. When compared to the rectangular planform, the triangular planform allows a wingspan increase of up to 15.04% and a reduction in induced drag of up to 24.44%, depending on the lift distribution. When compared to the elliptic planform, the triangular planform gives a reduction in induced drag of up to 11.71%. Results similar to these are typical for any fixed lift distribution; however, an absolute minimum in induced drag is not obtained unless the optimum lift distribution is also used.

In general, the optimum lift distribution that minimizes induced drag for a wing with an elliptic or linearly-tapered planform is an infinite Fourier series with coefficients B_n. In order to predict the optimum lift distribution, the infinite series must be truncated at some finite value of n and the values of B_n that minimize induced drag must be obtained numerically. For the planforms considered in this paper, the optimum values for B_n are given in Table A3. These values depend on the planform shape and the design constraints. However, Fig. 4 shows that for the planforms and design constraints considered here, the general shape of the optimum lift distribution varies only slightly with planform shape. Figures 7 and 8 show that, when compared to the elliptic lift distribution, the optimum lift distribution can allow a wingspan increase of up to 7.63% and an induced-drag reduction of up to 5.94% at $R_T = 0$. Thus, it has been shown that Prandtl's conclusions about tapered wings maintain their validity when the effects of planform on the wing-structure weight are taken into account.

It has also been shown that although the optimum lift distributions that minimize induced drag for wings with elliptic and linearly-tapered planforms depend on an infinite number of Fourier coefficients, for the planforms and design constraints considered here, a good approximation for the optimum lift distribution can be made by including only B_3 in the series defining the lift distribution. Figure 6 shows that the value of B_3 in the optimum lift distribution decreases as the taper ratio decreases.

The results presented in this paper provide valuable insight into the aerodynamic and structural coupling involved in the stress-limited design of wings with elliptic and linearly-tapered planforms for minimum induced drag. Although the results favor wings with low taper ratios, it should always be remembered that planforms with small chord values near the wingtips are seldom practical because they tend to stall at the wingtips and exhibit poor handling qualities, especially during stall recovery. Nevertheless, the results shown here may shed light on why many high-endurance birds have low-taper-ratio wings. It is also important to remember that we have only considered wings with the weight distribution given by Eqs. (5) and (6) and elliptic or linearly-tapered planforms with $0 \le R_T \le 1$. If any other weight distribution is used, the wing-structure weight, optimum wingspan, and minimum induced drag may need to be found using numerical methods. However, if the weight distribution given by Eqs. (5) and (6) is used, the methods presented in this paper can be repeated for any planform with a chord distribution that can be integrated in z.

Appendix

As seen in Eq. (44), each of the coefficients, C_n, for a linearly-tapered wing is a nonlinear function of taper ratio and must be evaluated numerically. The result of evaluating Eq. (44) for taper ratios in the range $0 \le R_T \le 1$ is shown in Table A1. The composite Simpson's rule was used for all numerical integration. For convenience, each of the coefficients, C_n, was also fit, as a function of taper ratio, to a sixth-order polynomial of the form

$$C_n = \sum_{m=0}^{6} a_m R_T^m \tag{A1}$$

where a_m are the fit coefficients given in Table A2 for all odd $n \leq 29$ and taper ratios in the range $0.2 \leq R_T \leq 1.5$. Outside this range of taper ratios, the value of C_n as a function of taper ratio cannot be accurately approximated using a polynomial of reasonably low order.

With the coefficients from Table A1, Eqs. (33) and (53) can be minimized using a numerical optimization framework with wingspan and the coefficients, B_n, as the design variables. Using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) [40-43] method with the Fourier coefficients, B_n, for all odd n in the range $3 \leq n \leq 29$ gives the optimum lift distributions shown in Fig. 4 and the optimum values for the Fourier coefficients, B_n, shown in Table A3.

Table A1 C_n coefficients for all odd $n \leq 29$ for wings with linearly-tapered planforms having $0 \leq R_T \leq 1$.

	$R_T = 0.0$	$R_T = 0.1$	$R_T = 0.2$	$R_T = 0.3$	$R_T = 0.4$	$R_T = 0.5$	$R_T = 0.6$	$R_T = 0.7$	$R_T = 0.8$	$R_T = 0.9$	$R_T = 1.0$
C_1	2.7716×10^{-1}	2.6155×10^{-1}	2.4966×10^{-1}	2.3983×10^{-1}	2.3139×10^{-1}	2.2398×10^{-1}	2.1737×10^{-1}	2.1140×10^{-1}	2.0596×10^{-1}	2.0097×10^{-1}	1.9635×10^{-1}
C_3	3.1562×10^{-1}	2.8932×10^{-1}	2.7072×10^{-1}	2.5600×10^{-1}	2.4378×10^{-1}	2.3332×10^{-1}	2.2420×10^{-1}	2.1612×10^{-1}	2.0887×10^{-1}	2.0232×10^{-1}	1.9635×10^{-1}
C_5	4.3190×10^{-2}	2.8797×10^{-2}	2.0825×10^{-2}	1.5437×10^{-2}	1.1491×10^{-2}	8.4614×10^{-3}	6.0592×10^{-3}	4.1092×10^{-3}	2.4975×10^{-3}	1.1462×10^{-3}	0
C_7	7.6085×10^{-3}	2.1946×10^{-3}	5.1974×10^{-4}	-1.5985×10^{-4}	-4.2757×10^{-4}	-4.9724×10^{-4}	-4.6465×10^{-4}	-3.7786×10^{-4}	-2.6275×10^{-4}	-1.3397×10^{-4}	0
C_9	3.4235×10^{-3}	9.5812×10^{-4}	4.8609×10^{-4}	3.2466×10^{-4}	2.4850×10^{-4}	1.9910×10^{-4}	1.5792×10^{-4}	1.1872×10^{-4}	7.9520×10^{-5}	3.9910×10^{-5}	0
C_{11}	1.1890×10^{-3}	5.1200×10^{-6}	-9.7040×10^{-5}	-1.0446×10^{-4}	-9.4190×10^{-5}	-7.9840×10^{-5}	-6.4390×10^{-5}	-4.8540×10^{-5}	-3.2470×10^{-5}	-1.6270×10^{-5}	0
C_{13}	7.5613×10^{-4}	1.0620×10^{-4}	6.6420×10^{-5}	5.5000×10^{-5}	4.6900×10^{-5}	3.9190×10^{-5}	3.1470×10^{-5}	2.3670×10^{-5}	1.5820×10^{-5}	7.9200×10^{-6}	0
C_{15}	3.3871×10^{-4}	-2.7780×10^{-5}	-3.3150×10^{-5}	-2.9780×10^{-5}	-2.5690×10^{-5}	-2.1480×10^{-5}	-1.7230×10^{-5}	-1.2950×10^{-5}	-8.6500×10^{-6}	-4.3300×10^{-6}	0
C_{17}	2.5479×10^{-4}	2.5700×10^{-5}	2.0420×10^{-5}	1.7800×10^{-5}	1.5300×10^{-5}	1.2780×10^{-5}	1.0240×10^{-5}	7.7000×10^{-6}	5.1400×10^{-6}	2.5700×10^{-6}	0
C_{19}	1.3071×10^{-4}	-1.3500×10^{-5}	-1.2810×10^{-5}	-1.1270×10^{-5}	-9.6800×10^{-6}	-8.0800×10^{-6}	-6.4800×10^{-6}	-4.8600×10^{-6}	-3.2500×10^{-6}	-1.6200×10^{-6}	0
C_{21}	1.0865×10^{-4}	9.8600×10^{-6}	8.5400×10^{-6}	7.4800×10^{-6}	6.4300×10^{-6}	5.3600×10^{-6}	4.3000×10^{-6}	3.2300×10^{-6}	2.1500×10^{-6}	1.0800×10^{-6}	0
C_{23}	6.0660×10^{-5}	-6.5300×10^{-6}	-5.8900×10^{-6}	-5.1700×10^{-6}	-4.4400×10^{-6}	-3.7000×10^{-6}	-2.9600×10^{-6}	-2.2300×10^{-6}	-1.4800×10^{-6}	-7.4000×10^{-7}	0
C_{25}	5.3880×10^{-5}	4.7500×10^{-6}	4.2000×10^{-6}	3.6800×10^{-6}	3.1600×10^{-6}	2.6400×10^{-6}	2.1100×10^{-6}	1.5900×10^{-6}	1.0600×10^{-6}	5.3000×10^{-7}	0
C_{27}	3.1880×10^{-5}	-3.4500×10^{-6}	-3.0800×10^{-6}	-2.7000×10^{-6}	-2.3100×10^{-6}	-1.9300×10^{-6}	-1.5500×10^{-6}	-1.1600×10^{-6}	-7.7000×10^{-7}	-3.9000×10^{-7}	0
C_{29}	2.9690×10^{-5}	2.5900×10^{-6}	2.3100×10^{-6}	2.0200×10^{-6}	1.7300×10^{-6}	1.4500×10^{-6}	1.1600×10^{-6}	8.7000×10^{-7}	5.8000×10^{-7}	2.9000×10^{-7}	0

Table A2 Fit coefficients in the polynomial approximation for C_n as a function of taper ratio for wings with linearly-tapered planforms having $0.2 \le R_T \le 1.5$.

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	RMS
C_1	2.7264×10^{-1}	-1.3235×10^{-1}	8.8200×10^{-2}	-4.0101×10^{-2}	7.9669×10^{-3}	0	0	5.3456×10^{-3}
C_3	3.0641×10^{-1}	-2.1072×10^{-1}	1.6649×10^{-1}	-8.3750×10^{-2}	1.7923×10^{-2}	0	0	7.9136×10^{-3}
C_5	3.8614×10^{-2}	-1.2292×10^{-1}	2.1171×10^{-1}	-2.4417×10^{-1}	1.7341×10^{-1}	-6.7698×10^{-2}	1.1056×10^{-2}	2.3554×10^{-3}
C_7	4.1527×10^{-3}	-3.0318×10^{-2}	7.8593×10^{-2}	-1.0626×10^{-1}	8.1457×10^{-2}	-3.3185×10^{-2}	5.5653×10^{-3}	1.7653×10^{-3}
C_9	1.3151×10^{-3}	-7.0966×10^{-3}	1.9698×10^{-2}	-2.9850×10^{-2}	2.4629×10^{-2}	-1.0516×10^{-2}	1.8203×10^{-3}	1.0419×10^{-3}
C_{11}	1.5970×10^{-5}	-1.1787×10^{-3}	4.0982×10^{-3}	-6.4574×10^{-3}	5.5203×10^{-3}	-2.4288×10^{-3}	4.3041×10^{-4}	5.8846×10^{-4}
C_{13}	1.0667×10^{-4}	-3.1870×10^{-4}	7.8674×10^{-4}	-1.2858×10^{-3}	1.1205×10^{-3}	-4.9856×10^{-4}	8.9100×10^{-5}	2.8810×10^{-4}
C_{15}	-4.1110×10^{-5}	3.6170×10^{-5}	7.4800×10^{-6}	-2.5400×10^{-6}	0	0	0	2.0678×10^{-4}
C_{17}	2.5570×10^{-5}	-2.5570×10^{-5}	0	0	0	0	0	1.6191×10^{-4}
C_{19}	-1.6170×10^{-5}	1.6170×10^{-5}	0	0	0	0	0	1.2791×10^{-4}
C_{21}	1.0730×10^{-5}	-1.0730×10^{-5}	0	0	0	0	0	9.0310×10^{-5}
C_{23}	-7.4000×10^{-6}	7.4000×10^{-6}	0	0	0	0	0	7.2520×10^{-5}
C_{25}	5.2700×10^{-6}	-5.2700×10^{-6}	0	0	0	0	0	5.5860×10^{-5}
C_{27}	-3.8600×10^{-6}	3.8600×10^{-6}	0	0	0	0	0	4.4150×10^{-5}
C_{29}	2.8900×10^{-6}	-2.8900×10^{-6}	0	0	0	0	0	3.5500×10^{-5}

Table A3 Optimum B_n coefficients for the elliptic planform and linearly-tapered planforms having $0 \leq R_T \leq 1$.

	Elliptic	$R_T = 0.0$	$R_T = 0.1$	$R_T = 0.2$	$R_T = 0.3$	$R_T = 0.4$	$R_T = 0.5$	$R_T = 0.6$	$R_T = 0.7$	$R_T = 0.8$	$R_T = 0.9$	$R_T = 1.0$
B_3	-1.4241×10^{-1}	-1.7193×10^{-1}	-1.6170×10^{-1}	-1.5562×10^{-1}	-1.5122×10^{-1}	-1.4777×10^{-1}	-1.4495×10^{-1}	-1.4256×10^{-1}	-1.4051×10^{-1}	-1.3870×10^{-1}	-1.3709×10^{-1}	-1.3564×10^{-1}
B_5	-2.9064×10^{-3}	-1.4116×10^{-2}	-9.6570×10^{-3}	-7.1827×10^{-3}	-5.4712×10^{-3}	-4.1795×10^{-3}	-3.1540×10^{-3}	-2.3118×10^{-3}	-1.6029×10^{-3}	-9.9503×10^{-4}	-4.6599×10^{-4}	9.0000×10^{-12}
B_7	-3.2293×10^{-4}	-1.7762×10^{-3}	-5.2567×10^{-4}	-1.2804×10^{-4}	4.0467×10^{-5}	1.1108×10^{-4}	1.3239×10^{-4}	1.2663×10^{-4}	1.0528×10^{-4}	7.4774×10^{-5}	3.8903×10^{-5}	2.0000×10^{-12}
B_9	-6.6720×10^{-5}	-6.2162×10^{-4}	-1.7850×10^{-4}	-9.3141×10^{-5}	-6.3924×10^{-5}	-5.0212×10^{-5}	-4.1229×10^{-5}	-3.3472×10^{-5}	-2.5726×10^{-5}	-1.7602×10^{-5}	-9.0136×10^{-6}	-1.0000×10^{-11}
B_{11}	-1.9347×10^{-5}	-1.7664×10^{-4}	-7.8088×10^{-7}	1.5213×10^{-5}	1.6829×10^{-5}	1.5571×10^{-5}	1.3527×10^{-5}	1.1167×10^{-5}	8.6058×10^{-6}	5.8801×10^{-6}	3.0064×10^{-6}	-1.8900×10^{-10}
B_{13}	-6.9638×10^{-6}	-9.5050×10^{-5}	-1.3697×10^{-5}	-8.8098×10^{-6}	-7.4968×10^{-6}	-6.5608×10^{-6}	-5.6190×10^{-6}	-4.6176×10^{-6}	-3.5519×10^{-6}	-2.4234×10^{-6}	-1.2382×10^{-6}	9.1600×10^{-10}
B_{15}	-2.9162×10^{-6}	-3.6901×10^{-5}	3.1050×10^{-6}	3.8109×10^{-6}	3.5170×10^{-6}	3.1145×10^{-6}	2.6692×10^{-6}	2.1918×10^{-6}	1.6852×10^{-6}	1.1479×10^{-6}	5.8668×10^{-7}	-1.1860×10^{-9}
B_{17}	-1.3650×10^{-6}	-2.4492×10^{-5}	-2.5354×10^{-6}	-2.0725×10^{-6}	-1.8559×10^{-6}	-1.6362×10^{-6}	-1.4009×10^{-6}	-1.1497×10^{-6}	-8.8294×10^{-7}	-6.0131×10^{-7}	-3.0728×10^{-7}	4.9300×10^{-10}
B_{19}	-6.9478×10^{-7}	-1.1242×10^{-5}	1.1916×10^{-6}	1.1628×10^{-6}	1.0507×10^{-6}	9.2567×10^{-7}	7.9261×10^{-7}	6.5023×10^{-7}	4.9931×10^{-7}	3.4024×10^{-7}	1.7371×10^{-7}	5.6000×10^{-11}
B_{21}	-3.7990×10^{-7}	-8.4548×10^{-6}	-7.8731×10^{-7}	-7.0147×10^{-7}	-6.3170×10^{-7}	-5.5646×10^{-7}	-4.7621×10^{-7}	-3.9035×10^{-7}	-2.9985×10^{-7}	-2.0409×10^{-7}	-1.0423×10^{-7}	-2.1700×10^{-10}
B_{23}	-2.1930×10^{-7}	-4.3099×10^{-6}	4.7602×10^{-7}	4.4285×10^{-7}	3.9864×10^{-7}	3.5092×10^{-7}	2.9990×10^{-7}	2.4586×10^{-7}	1.8871×10^{-7}	1.2888×10^{-7}	6.5063×10^{-8}	-5.8000×10^{-11}
B_{25}	-1.3294×10^{-7}	-3.5226×10^{-6}	-3.1846×10^{-7}	-2.9050×10^{-7}	-2.6115×10^{-7}	-2.2987×10^{-7}	-1.9673×10^{-7}	-1.6023×10^{-7}	-1.2354×10^{-7}	-8.4664×10^{-8}	-4.3076×10^{-8}	3.7500×10^{-10}
B_{27}	-8.4463×10^{-8}	-1.9291×10^{-6}	2.1416×10^{-7}	1.9705×10^{-7}	1.7719×10^{-7}	1.5597×10^{-7}	1.3237×10^{-7}	1.0902×10^{-7}	8.3873×10^{-8}	5.3819×10^{-8}	2.9154×10^{-8}	-1.9700×10^{-10}
B_{29}	-5.3073×10^{-8}	-1.6725×10^{-6}	-1.5014×10^{-7}	-1.3694×10^{-7}	-1.2425×10^{-7}	-1.0813×10^{-7}	-9.3527×10^{-8}	-7.5954×10^{-8}	-5.4803×10^{-8}	-3.9496×10^{-8}	-1.9609×10^{-8}	5.1200×10^{-10}

Acknowledgements

This material is partially based upon work supported by the National Aeronautics and Space Administration under Grant No. 80NSSC18K1696 issued through the Aeronautics Research Mission Directorate through the 2018 NASA Fellowship Activity, with Nhan Nguyen as the NASA Technical Advisor.

References

[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., and Hunsaker, D. F., "Designing Wing Twist or Planform Distributions for Specified Lift Distributions," *Journal of Aircraft*, Vol. 56, No. 2, 2019, pp. 847–849. (doi:10.2514/1.C035206)

[4] 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)

[5] 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)

[6] Gallay, S., and Laurendeau, E., "Preliminary-Design Aerodynamic Model for Complex Configurations Using Lifting-Line Coupling Algorithm," *Journal of Aircraft*, Vol. 53, No. 4, 2016, pp. 1145–1159. (doi:10.2514/1.C033460)

[7] Phillips, W. F., and Hunsaker, D. F., "Lifting-Line Predictions for Induced Drag and Lift in Ground Effect," *Journal of Aircraft*, Vol. 50, No. 4, 2013, pp. 1226–1233. (doi:10.2514/1.C032152)

[8] Wickenheiser, A., and Garcia, E., "Aerodynamic Modeling of Morphing Wings Using an Extended Lifting-Line Analysis," *Journal of Aircraft*, Vol. 44, No. 1, 2007, pp. 10–16. (doi:10.2514/1.18323)

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

[10] Rasmussen, M. L., and Smith, D. E., "Lifting-Line Theory for Arbitrarily Shaped Wings," *Journal of Aircraft*, Vol. 36, No. 2, 1999, pp. 340–348. (doi:10.2514/2.2463)

[11] Bera, R. K., "Some remarks on the solution of the lifting line equation," *Journal of Aircraft*, Vol. 11, No. 10, 1974, pp. 647–648. (doi:10.2514/3.44397)

[12] 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 pp. 431-441. (doi: 10.2514/1.C035027)

[13] McCormick, B. W., *Aerodynamics, Aeronautics, and Flight Mechanics*, 1st ed., Wiley, New York, 1979

[14] Prandtl, L., "Über Tragflügel kleinsten induzierten Widerstandes," Zeitschrift für Flugtechnik und Motorluftschiffahrt, Vol. 24, No. 11, 1933, pp. 305–306.

[15] Hunsaker, D. F., "Ludwig Prandtl's 1933 Paper Concerning wings for Minimum Induced Drag, Translation and Commentary," AIAA SciTech 2020 Forum, Orlando, Florida, 6-10 January 2020.

[16] Lundry, J. L., "Minimum Swept-Wing Induced Drag with Constraints on Lift and Pitching Moment," *Journal of Aircraft*, Vol. 4, 1967, pp. 73–74. (doi:10.2514/3.43797)

[17] Lissaman, P. B. S., and Lundry, J. L., "A Numerical Solution for the Minimum Induced Drag of Nonplanar Wings," *Journal of Aircraft*, Vol. 5, 1968, pp. 17–21. (doi:10.2514/3.43901)

[18] Ashenberg, J., and Weihsradius, D., "Minimum Induced Drag of Wings with Curved Planform," *Journal of Aircraft*, Vol. 21, 1984, pp. 89–91. (doi:10.2514/3.56733)

[19] Rokhsaz, K., "Effect of Viscous Drag on Optimum Spanwise Lift Distribution," *Journal of Aircraft*, Vol. 30, 1993, pp. 152–154. (doi:10.2514/3.46328)

[20] Demasi, L., "Induced Drag Minimization: A Variational Approach Using the Acceleration Potential," *Journal of Aircraft*, Vol. 43, 2006, pp. 669–680. (doi:10.2514/1.15982)

[21] Demasi, L., "Erratum on Induced Drag Minimization: A Variational Approach Using the Acceleration Potential," *Journal of Aircraft*, Vol. 43, 2006, p. 1247. (doi:10.2514/1.26648)

[22] Demasi, L., "Investigation on the Conditions of Minimum Induced Drag of Closed Wing Systems and C-Wings," *Journal of Aircraft*, Vol. 44, 2007, pp. 81–99. (doi:10.2514/1.21884)

[23] Demasi, L., Monegato, G., and Cavallaro, R., "Minimum Induced Drag Theorems for Multiwing Systems," *AIAA Journal*, Vol. 55, 2017, pp. 3266–3287. (doi:10.2514/1.J055652)

[24] Demasi, L., Dipace, A., Monegato, G., and Cavallaro, R., "Invariant Formulation for the Minimum Induced Drag Conditions of Nonplanar Wing Systems," *AIAA Journal*, Vol. 52, 2014, pp. 2223–2240. (doi:10.2514/1.J052837)

[25] Grossman, B., Gurdal, Z., Strauch, G. J., Eppard, W. M., and Haftka, R. T., "Integrated Aerodynamic/Structural Design of a Sailplane Wing," *Journal of Aircraft*, Vol. 25, No. 9, 1988, pp. 855–860. (doi: 10.2514/3.45670)

[26] Craig, A. P., and McLean, D. J., "Spanload Optimization for Strength Designed Lifting Surfaces," AIAA 88-2512, 6th Applied Aerodynamics Conference, Williamsburg, Virginia, 5–8 June 1988.

[27] Iglesias, S., and Mason, W. H., "Optimum Spanloads Incorporating Wing Structural Weight," AIAA 2001-5234, 1st Aircraft, Technology Integration, and Operations Forum, Los Angeles, California, 16–18 October 2001.

[28] Wroblewski, G. E., and Ansell, P. J., "Prediction and Experimental Evaluation of Planar Wing Spanloads for Minimum Drag," *Journal of Aircraft*, Vol. 54, 2017, pp. 1664–1674. (doi:10.2514/1.C034156)

[29] Gopalarathnam, A., and Norris, R. K., "Ideal Lift Distributions and Flap Angles for Adaptive Wings," *Journal of Aircraft*, Vol. 46, No. 2, 2009, pp. 562–571. (doi: 10.2514/1.38713)

[30] McGeer, T., "Wing Design for Minimum Drag with Practical Constraints," *Journal of Aircraft*, Vol. 21, 1984, pp. 879–886. (doi:10.2514/3.45058)

[31] Pate, D. J., and German, B. J., "Lift Distributions for Minimum Induced Drag with Generalized Bending Moment Constraints," *Journal of Aircraft*, Vol. 50, 2013, pp. 936–946. (doi:10.2514/1.C032074)

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

[33] DeYoung, J., "Minimization Theory of Induced Drag Subject to Constraint Conditions," NASA CR-3140, June 1979.

[34] Jones, R. T., and Lasinski, T. A., "Effect of Winglets on the Induced Drag of Ideal Wing Shapes," NASA TM-81230, September 1980.

[35] Klein, A., and Viswanathan, S. P., "Minimum Induced Drag of Wings with Given Lift and Root-Bending Moment," *Zeitschrift fur Angewandte Mathematik und Physik*, Vol. 24, 1973, pp. 886–892.

[36] Klein, A., and Viswanathan, S. P., "Approximate Solution for Minimum Induced Drag of Wings with Given Structural Weight," *Journal of Aircraft*, Vol. 12, No. 2, 1975, pp. 124–126. (doi:10.2514/3.44425)

[37] Löbert, G., "Spanwise Lift Distribution of Forward- and Aft-Swept Wings in Comparison to the Optimum Distribution Form," *Journal of Aircraft*, Vol. 18, No. 4, 1981, pp. 496-498. (doi: 10.2514/3.44717)

[38] Phillips, W. F., Hunsaker, D. F., and Taylor, J. D., "Minimizing Induced Drag with Weight Distribution, Lift Distribution, Wingspan, and Wing-Structure Weight," AIAA 2019-3349, AIAA Aviation 2019 Forum, Dallas, Texas, 17-21 June 2019. (doi: 10.2514/6.2019-3349)

[40] 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)

[41] 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)

[42] 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)

[43] 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)

[44] Glauert, H., *The Elements of Aerofoil and Airscrew Theory,* Cambridge University Press, London, 1926.

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

CHAPTER 3

Low-Fidelity Method for Rapid Aerostructural Optimization and Design-Space Exploration of Planar Wings

Jeffrey D. Taylor and Douglas F. Hunsaker
Utah State University
Logan, Utah
84322-4130
USA

Abstract

During early phases of wing design, analytic and low-fidelity methods are often used to identify promising design concepts. In many cases, solutions obtained using these methods provide intuition about the design space that is not easily obtained using higher-fidelity methods. This is especially true for aerostructural design. However, many analytic and low-fidelity aerostructural solutions are limited in application to wings with specific planforms and weight distributions. Here, a numerical method for minimizing induced drag with structural constraints is presented that uses approximations that apply to unswept planar wings with arbitrary planforms and weight distributions. The method is applied to the NASA Ikhana airframe to show how it can be used for rapid aerostructural optimization and design-space exploration. The design space around the optimum solution is visualized, and the sensitivity of the optimum solution to changes in weight distribution, structural properties, wing loading, and taper ratio is shown. The optimum lift distribution and wing-structure weight for the Ikhana airframe are shown to be in good agreement with analytic solutions. Whereas most modern high-fidelity solvers obtain solutions in a matter of hours, all of the solutions shown here can be obtained in a matter of seconds.

This paper was published in *The Aeronautical Journal* in 2021 as:
Taylor, J. D., and Hunsaker, D. F., "Low-Fidelity Method for Rapid Aerostructural Optimization and Design-Space Exploration of Planar Wings," *The Aeronautical Journal*, Vol. 125, No. 1289, July 2021, pp. 1209-1230. (doi:10.1017/aer.2021.14)

Keywords: Lifting-Line Theory; Multidisciplinary Design Optimization; Aerostructural Optimization; Induced-Drag Minimization

Nomenclature

A	beam cross-sectional area
A_n	Fourier coefficients in the lifting-line solution for the section-lift distribution, Equation (1)
b	wingspan
B_n	Fourier coefficients in the lifting-line solution for the dimensionless section-lift distribution, Equation (1)
c	local wing section chord length
C_δ	shape coefficient for the deflection-limited design, Equation (15)
C_σ	shape coefficient for the stress-limited design, Equation (5)
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
K	scaling coefficient in the equation for the fuel distribution, Equation (21)
L	total wing lift
$\widetilde{L}$	local wing section lift
$\widetilde{M}_b$	local wing section bending moment
n_a	load factor, g
n_g	limiting load factor at the hard-landing design limit
n_m	limiting load factor at the maneuvering-flight design limit
R_A	wing aspect ratio
R_T	wing taper ratio
S	wing planform area

S_b proportionality coefficient between $\tilde{W}_s(z)$ and $\tilde{M}_b(z)$ having units of length squared, Equations (5) and (15)

$t_{\max}$ maximum thickness of the local airfoil section

V_∞ freestream airspeed

w width of the beam cross-section

$w_{\max}$ maximum allowable width of the beam cross-section

W aircraft gross weight

W_f gross weight of fuel

W_n aircraft net weight, defined as $W - W_s$

W_r that portion of W_n carried at the wing root

W_s total weight of the wing structure required to support the wing bending moment distribution

$\tilde{W}_n$ 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

γ specific weight of the beam material

δ local wing deflection

$\delta_{\max}$ maximum wing deflection

θ change of variables for the spanwise coordinate, Equation (1)

ρ air density

$\sigma_{\max}$ maximum longitudinal stress

1.0 Introduction

When designing a wing for minimum drag, low-fidelity tools are useful for rapid design-space exploration and for gaining important insight into how the design variables, parameters, and constraints influence the optimum solution. Designers often rely on rules-of-thumb based on these insights during the

conceptual and preliminary design phases. In many cases, low-fidelity solutions have been shown to be in good agreement with experimental data and computational fluid dynamics[1-8], while providing significantly more mathematical and physical insight than higher-fidelity models. For example, the well-known elliptic lift distribution, which minimizes induced drag on an unswept planar wing with fixed weight and wingspan, was first identified from analytic solutions based on lifting-line theory[9,10] by Prandtl[9] and later by Munk[11]. The elliptic lift distribution remains a common benchmark in many mid- and high-fidelity computational studies[12-19]. However, the elliptic lift distribution does not minimize drag under all conditions[20-29]. In particular, when structural effects are considered, drag is typically minimized using a non-elliptic lift distribution that depends on the design constraints[12,13,16-18,30-49]. Low-fidelity and analytic aerostructural methods are valuable for identifying these non-elliptic lift distributions and for understanding how structural considerations affect the minimum-drag solution.

There are many mid- and high-fidelity computational studies for minimizing drag under structural constraints that include solutions with non-elliptic lift distributions[12,13,16-18,30-38]. However, there are relatively few studies that approach this multidisciplinary problem from an analytic or low-fidelity point of view[39-49]. Prandtl seems to be the first do so, minimizing induced drag with fixed lift and moment of inertia of gross lift[39]. Jones later[40] sought to minimize induced drag under the constraints of fixed gross lift and root bending moment in cruise. Pate and German[41] constrained the root bending moment at a given off-design lift coefficient. DeYoung[42] replaced Jones' root-bending-moment constraint with a constraint on the bending moment at a prescribed spanwise location. Jones and Lasinski[43] constrained the integrated bending moment. Klein and Viswanathan[44,45] considered both root and integrated bending moment[44] and included the effects of shear on the wing-structure weight[45]. Löbert[46] introduced a constraint based on the ratio of the bending-moment distribution and the wing-section thickness. More recently, Phillips et al.[47,48] and Taylor and Hunsaker[49] minimized induced drag under constraints of fixed gross weight[47,49], fixed net weight[48,49], fixed wing loading[47-49], and fixed stall speed[48], including the effects of the planform shape on the wing-structure weight and the effects of the wing weight distribution on the bending moments.

Each of the studies in Refs. (39-49) includes assumptions that may not be representative of all aircraft. For example, Refs. (39,44,45) include assumptions about the proportionality between the wing-structure weight and the bending moments that correspond to rectangular wings. References (39-46) include the assumption that the bending moments are caused by the lift alone, which limits their application to wings with negligible structural or payload weight. The formulations given by Phillips et al.[47,48] and Taylor and Hunsaker[49] are arguably more general than those given in Refs. (39-46). Still, in order to obtain analytic solutions, Phillips et al.[47,48] and Taylor and Hunsaker[49] limited their results to specific wing planforms with a single ideal weight distribution.

The purpose of this paper is to present a low-fidelity numerical method that extends the work of Phillips et al.[47,48] and Taylor and Hunsaker[49] to more practical aircraft configurations with arbitrary planforms and weight distributions. We will apply the method to a high-endurance unmanned aircraft configuration to demonstrate how it can be used for rapid conceptual design and for gaining intuition about the aerostructural design space. The present work builds on the approach taken by Prandtl[39] and Phillips et al.[47,48]. Therefore, we will first briefly review the work of these authors.

2.0 Analytical Foundation

Using Prandtl's classical lifting-line theory[9,10], the dimensionless spanwise section-lift distribution on a finite wing with no dihedral or sweep immersed in a uniform flow can be written as[47]

$$\frac{b\widetilde{L}(\theta)}{L} = \frac{4}{\pi}\left[\sin(\theta) + \sum_{n=2}^{\infty} B_n \sin(n\theta)\right]; \qquad B_n \equiv \frac{A_n}{A_1}, \qquad \theta \equiv \cos^{-1}(-2z/b) \tag{1}$$

where B_n are normalized Fourier coefficients. Below stall, any lift distribution can be produced by a twisted wing of any planform if the correct twist distribution is used[50]. Therefore, in this paper, the lift distribution and the planform are treated as independent parameters, related through the wing twist, which is assumed to be correctly designed to achieve the desired lift distribution. In steady-level flight, the drag induced by such a wing can be written as

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

where W is the wing weight, and b is the wingspan. Because this study focuses on minimizing induced drag, we will neglect the effects of viscous drag.

Equation (2) reveals that induced drag depends on the weight, wingspan, and lift distribution. For a fixed ratio of weight to wingspan, Equation (2) is minimized with a lift distribution having $B_n = 0$ for all $n > 1$, which gives the well-known elliptic lift distribution. If weight and wingspan are allowed to vary, the induced drag can be reduced by increasing wingspan or decreasing wing weight. However, as wingspan increases, the weight of the wing structure required to support the bending moments also increases, which increases the total weight. Certain lift distributions that shift lift inboard can alleviate bending moments near the wingtips, allowing a higher wingspan with no increase in wing-structure weight. Therefore, to fully minimize Equation (2) for a given flight condition, the weight, wingspan, and lift distribution must all be considered.

In 1933, Prandtl[39] identified a bell-shaped lift distribution having $B_2 = 0$, $B_3 = -1/3$, and $B_n = 0$ for $n > 3$ that minimizes induced drag for rectangular wings under constraints of fixed gross weight and moment of inertia of gross weight. Prandtl assumed that the wing-structure weight distribution $\widetilde{W}_s(z)$ is related to the bending-moment distribution $\widetilde{M}_b(z)$ by a spanwise-invariant proportionality coefficient S_b, i.e.,

$$\widetilde{W}_s(z) = \frac{\widetilde{M}_b(z)}{S_b} \tag{3}$$

This assumption is best matched by a rectangular wing with a constant thickness-to-chord ratio[39]. Prandtl also assumed that the bending-moment distribution is a function of the lift distribution alone. Under the constraints of these assumptions, Prandtl's 1933 lift distribution allows an increase in wingspan of 22.5% and a reduction in induced drag of 11.1% when compared to that of the elliptic lift distribution with the

same wing-structure weight. However, Prandtl acknowledged that his formulation of the problem may not be the most appropriate for practical wing designs[39].

Phillips et al.[47,48] reformulated the problem with more practical assumptions and constraints. They pointed out that at each spanwise location, the wing bending moments are a function of the lift distribution, the net-weight distribution $\widetilde{W}_n(z)$ of all non-structural components carried in the wing, and the wing-structure weight distribution $\widetilde{W}_s(z)$ according to the relation[47]

$$\widetilde{M}_b(z) = \int_{z'=z}^{b/2} [\widetilde{L}(z') - n_a \widetilde{W}_n(z') - n_a \widetilde{W}_s(z')](z'-z)dz', \quad \text{for } z \geq 0 \tag{4}$$

where n_a is the load factor. The wing structure must be designed to support the bending moments during a high-load maneuver with a positive load limit n_m and during a hard landing with a negative load limit n_g. Assuming that all of the wing bending moments are supported by a single, vertically-symmetric beam in pure bending with maximum allowable stress $\sigma_{\max}$, the weight of the wing structure required to support the bending moments can be written[47]

$$W_s = 2\int_0^{b/2} \frac{\left|\widetilde{M}_b(z)\right|}{S_b(z)}dz; \quad S_b(z) = \frac{C_\sigma \left[t_{\max}(z)/c(z)\right] c(z)\,\sigma_{\max}}{\gamma}, \quad C_\sigma = \frac{2I(h/t_{\max})}{Ah^2} \tag{5}$$

where $c(z)$ is the section chord-length distribution, γ is the specific weight of the beam material, $t_{\max}/c$ is the maximum-thickness-to-chord ratio of the local airfoil section, and C_σ is a beam shape factor. A list of shape factors for common beam cross sections is given in Ref. (47). For deflection-limited designs, Equation (5) can be rewritten as[47]

$$W_s = 2\int_0^{b/2} \frac{\left|\widetilde{M}_b(z)\right|}{S_b(z)}dz; \quad S_b(z) = \frac{C_\delta E\left[t_{\max}(z)/c(z)\right]^2\ \delta_{\max}}{\gamma (W/S)^2}\frac{W^2}{b^4}, \quad C_\delta = \frac{8I(h/t_{\max})^2}{Ah^2} \tag{6}$$

where C_δ is the beam shape factor for the deflection-limited design and $\delta_{\max}$ is the maximum allowable vertical wingtip deflection. Although vertical deflection limits are seldom explicitly enforced in practice,

excessive vertical wingtip deflection can result in serious adverse effects, including wingtip strike at landing and dynamic instabilities during flight. Therefore, we will include both stress and vertical deflection limits in this paper. Nevertheless, the deflection limits in this paper are for structural sizing only. The static aeroelastic effects of structural bending and torsion are not explicitly considered. Instead, we assume that these effects can be corrected using wing twist.

The total weight of the wing is the sum of the wing-structure weight and the net weight of all non-structural components, i.e.,

$$W = W_s + W_n \tag{7}$$

The net weight W_n is found from the relation

$$W_n = W_r + \int_{z=-b/2}^{b/2} \widetilde{W}_n(z)dz \tag{8}$$

where W_r is the portion of the net weight carried at the wing root. The bending moments are minimized when the net weight is distributed according to the weight constraints given by[47]

$$\widetilde{W}_n(z) = (W - W_r)\frac{\widetilde{L}(z)}{L} - \widetilde{W}_s(z) \tag{9}$$

$$W_r = \frac{n_g - 1}{n_m + n_g}W \tag{10}$$

For a rectangular wing having the weight distribution from Equation (9), Equations (5) and (6) can be evaluated analytically. Assuming that the wing loading is fixed and a single lift distribution is used at all flight phases, Phillips et al.[48] showed that induced drag is minimized with a lift distribution having $B_2 = 0$, $B_3 = -3/8 + \sqrt{9/64 - 1/12}$, with $B_n = 0$ for $n > 3$ for the stress-limited design and $B_2 = 0$, $B_3 = -3/7 + \sqrt{9/49 - 1/21}$, with $B_n = 0$ for $n > 3$ for the deflection-limited design.

In this paper, we extend the work of Phillips et al.[47,48] and present a method for minimizing induced drag for wings with non-rectangular planforms and weight distributions other than Equation (9). It should

be remembered that the present method maintains the assumptions associated with lifting-line theory, including a planar wing with zero sweep and moderate to high aspect ratio. For other wing configurations, modifications to this method may be needed.

3.0 Wing-Structure Weight and Induced Drag

For the stress-limited design of a wing with a non-rectangular planform and a weight distribution other than Equation (9), the integrals in Equations (4) and (5) must often be evaluated numerically. Moreover, for any given flight condition, Equations (4) and (5) show that the wing bending moments and wing-structure weight distribution are coupled. Therefore, for a wing with any weight distribution other than Equation (9), a numerical iterative method is required to compute the wing-structure weight. The induced drag can be then found by using Equation (7) in Equation (2). An implementation of one such iterative process is given by Taylor et al.[51] for the stress-limited design.

For deflection-limited designs, the vertical spar deflection can be found using the relation[47]

$$\frac{d^2\delta}{dz^2} = \frac{2\sigma_{\max}}{Eh(z)} \tag{11}$$

where E is the modulus of elasticity of the beam material. For any spanwise-symmetric load distribution, the boundary conditions on Equation (11) are

$$\delta(0) = 0, \quad \left.\frac{d\delta}{dz}\right|_{z=0} = 0 \tag{12}$$

Integrating Equation (11) subject to Equation (12), the deflection at any spanwise location z_0 becomes

$$\delta(z_0) = \frac{2\sigma_{\max}}{E} \int_0^{z_0} \int_0^{z} \frac{1}{h(z')} dz' dz \tag{13}$$

If both maneuvering and hard-landing design limits are considered, maximum deflection always occurs at the wingtips. Using Equation (13), the deflection at the wingtip is

$$\delta_{\max} = \frac{2\sigma_{\max}}{E} \int_0^{b/2} \int_0^z \frac{1}{h(z')} dz' dz \tag{14}$$

Because airfoil thickness is typically a fraction of the chord length, the beam-height distribution $h(z)$ is typically related to the chord distribution. If the beam-height or chord distribution is an arbitrary function of spanwise location,

Equation (14) must be evaluated using numerical methods.

Using Equation (14) to replace $\sigma_{\max}$ in Equation (5), the wing-structure weight required to support the bending moments for the deflection-limited design can be written

$$W_s = 2 \int_0^{b/2} \frac{\left|\tilde{M}_b(z)\right|}{S_b(z)} dz; \quad S_b(z) = \frac{C_\delta E \left[t_{\max}(z)/c(z)\right] c(z)\, \delta_{\max}}{8\gamma \int_0^{b/2} \int_0^z \left[t_{\max}(z')/c(z')\right]^{-1} c(z')^{-1} dz' dz}, \quad C_\delta = \frac{8I(h/t_{\max})^2}{Ah^2} \tag{15}$$

Like Equation (5), Equation (15) is coupled with the bending-moment distribution. Thus, an iterative solver is needed to compute the wing-structure weight for the deflection-limited design.

If Equation (5) predicts a wing-structure weight that is greater than that predicted by Equation (15), the design is stress limited; if Equation (15) gives a value greater than Equation (5), the design is deflection limited. Because the limiting constraint depends on the design parameters, both stress and deflection limits must be considered at each spanwise location. However, recall that in this study, the aerodynamic effects of structural bending and twist are not included.

4.0 Numerical Methodology

Here, we present a method to iteratively compute the wing-structure weight and minimize induced drag. This method is similar to that given by Taylor et al.[51], but here we will include the deflection-limited design and several additional constraints that were not considered in Ref. (51).

4.1 Solving for Wing-Structure Weight

A fixed-point iteration scheme is used to compute the wing-structure weight and bending-moment distribution. An initial guess for the wing-structure weight is used in Equation (4) to calculate the section bending-moment distribution for both the maneuvering and hard-landing limits. At each section, the limit that produces a higher-magnitude section bending moment is the design limit. The limiting section bending moment is used in Equations (5) and (15) to predict the section wing-structure weight for the stress- and deflection-limited designs. At each section, the limiting wing-structure weight is then passed back as the guess for the next iteration. The process is repeated until the wing-structure weight converges within some specified tolerance. For the purposes of this study, an initial guess of $\widetilde{W}_s(z) = 0$ provides good results. The process is summarized as follows:

1. Input b, $\widetilde{L}(z)/L$, W_r, $\widetilde{W}_n(z)$, $c(z)$, $t_{\max}(z)/c(z)$, γ, E, $\sigma_{\max}$, $\delta_{\max}$, n_m, n_g, C_σ, and C_δ.
2. Calculate the total weight using Equation (7). For the initial guess, use $\widetilde{W}_s(z) = 0$, $W_s = 0$.
3. Calculate the total net weight using Equation (8).
4. Calculate the maneuvering and hard-landing bending-moment distributions using Equation (4).
5. Using the higher-magnitude section bending moment from step 4 in Equations (5) and (15), calculate the wing-structure weight distribution for the stress-and deflection-limited designs.
6. Calculate the total wing-structure weight by integrating either Equation (5) or (15).
7. Repeat steps 2 through 6 until the wing-structure weight has converged to within a specified tolerance.

Once the wing-structure weight is known, the induced drag is calculated using Equation (2). A schematic of the process is shown in Fig. 1. Note that after the first iteration, step 3 is only required if the net weight is a function of the wing-structure weight, as it is in Equation (9). In this paper, this special case will be used only for benchmarking the wing-structure weight solver against analytic solutions.

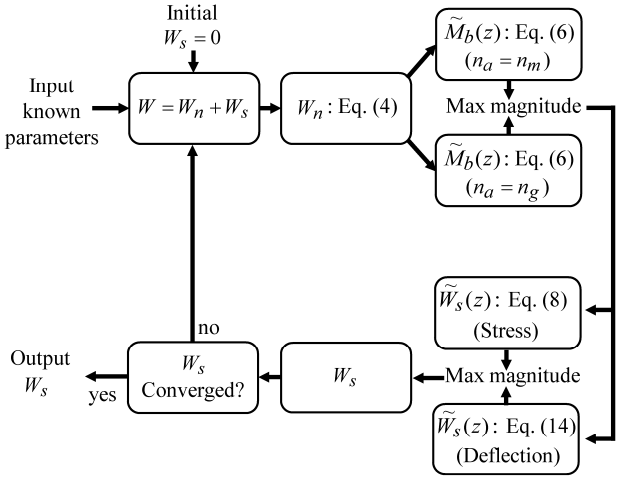


Figure 1. Schematic of the iterative wing-structure weight solver.

In general, any high-order integration scheme can be used to evaluate the integrals in Equations (4), (5), (8), and (15). In this study, the composite Simpson's rule is used. The wing is discretized using the cosine clustering scheme given in Equation (1), with even spacing in θ. The resulting grid is shown in Fig. 2. Using Simpson's rule, the wing-structure weight is evaluated as[51]

$$W_s = -b(\theta_m - \theta_0)\left(\frac{\widetilde{W}_{s,0}\sin\theta_0 + 4\sum_{i=\text{odd}}^{\text{m-1}}\widetilde{W}_{s,i}\sin\theta_i + 2\sum_{i=\text{even}}^{\text{m-2}}\widetilde{W}_{s,i}\sin\theta_i + \widetilde{W}_{s,m}\sin\theta_m}{3m}\right) \tag{16}$$

where m is the number of nodes, and $\widetilde{W}_{s,i}$ is evaluated from Equation (5) or Equation (15), i.e.,

$$\widetilde{W}_{s,i} = \frac{\left|\widetilde{M}_{b,i}\right|}{S_{b,i}}; \quad S_{b,i} = \frac{C_\sigma\left[t_{\max,i}/c_i\right]c_i\,\sigma_{\max}}{\gamma} \tag{17}$$

$$\widetilde{W}_{s,i} = \frac{\left|\widetilde{M}_{b,i}\right|}{S_{b,i}} \quad S_{b,i} = \frac{C_\delta E\left[t_{\max,i}/c_i\right]c_i\,\delta_{\max}}{4b\gamma\int_0^\pi\int_0^z\left[t_{\max}(\theta')/c(\theta')\right]^{-1}c(\theta')^{-1}\sin\theta'\sin\theta d\theta' d\theta} \tag{18}$$

The integral in the denominator of Equation (18) is also evaluated using Simpson's rule, and $\widetilde{M}_{b,i}$ is found from Equation (4), i.e.[51],

$$\widetilde{M}_{b,i} = -\frac{b}{2}(\theta_m - \theta_i)\left(\frac{f_i + 4\sum_{j=i+\text{odd}}^{m-1} f_j + 2\sum_{j=i+\text{even}}^{m-1} f_j + f_m}{3(m-i)}\right); \quad f_j = \frac{b}{2}\sin\theta_j\left(\widetilde{L}_j - n_a\widetilde{W}_{n,j} - n_a\widetilde{W}_{s,j}\right)\left(\cos\theta_j - \cos\theta_i\right) \tag{19}$$

Note that Simpson's rule requires even grid spacing. Therefore, Equations (16)-(19) are written in terms of θ.

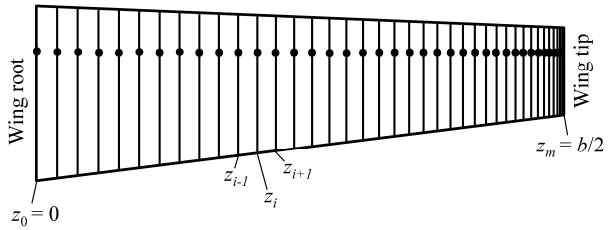


Figure 2. Discretization of a tapered semispan with 40 nodes and cosine clustering near the wing tip.

Figure 3 shows the results of a grid-resolution study for the iterative wing-structure weight solver using a wing with the parameters $R_T = 0.5$, $b = 66.0$ ft, $S = 267.3$ ft^2, $t/c = 0.1875$, $C_\sigma = 0.165$, $C_\delta = 0.653$, $\sigma_{\max} = 25\times10^3$ psi, $\delta_{\max} = 3.5$ ft, $E = 10.0\times10^6$ psi, $\gamma = 0.10$ lbf/in^3, $W_r = 4500$ lbf, $W_n = 7500$ lbf, $n_m = n_g = 3.75$, and the weight distribution given by Equation (9). Results were compared using grids with node counts ranging between 10 and 1280, and Richardson Extrapolation[52] was used to project a fully-grid-resolved value from the results obtained with 160, 320, and 640 nodes. Above 40 nodes, the method shows second-order convergence, meaning that as the grid size is halved, the solution error is approximately reduced to one-fourth the previous value. The extrapolated value differs from the analytic solution[49] by only 0.001%. With as few as 160 nodes, the predicted wing-structure weight falls within 0.003 % of the extrapolated value. Therefore, 160 nodes will be used for all subsequent results. With 160 nodes, the total predicted wing-structure weight matches the analytic solution to within 0.004%, and Fig. 4 shows that the predicted wing-structure weight distribution is in good agreement with the analytic solution.

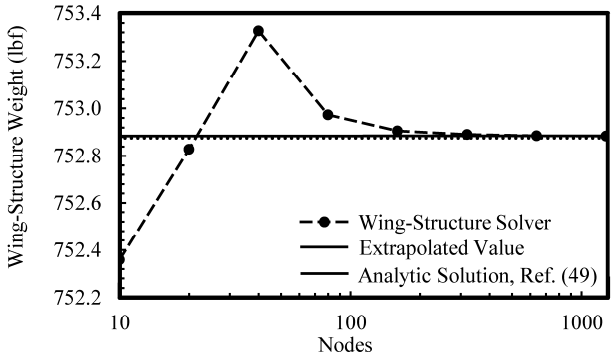


Figure 3. Grid-resolution results for the iterative wing-structure weight solver.

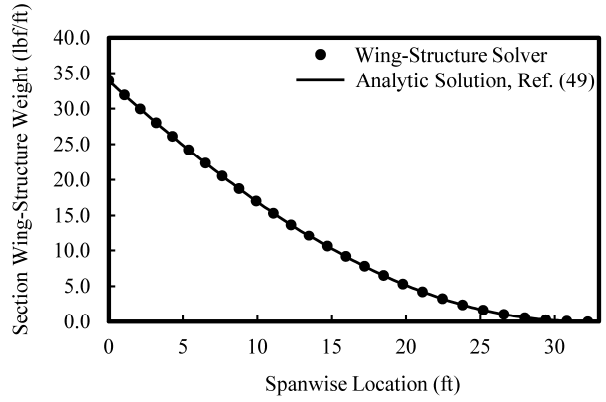


Figure 4. Comparison of the wing-structure weight predicted by the numerical wing-structure weight solver and the analytic solution from Ref. (49).

4.2 Minimizing Induced Drag in an Optimization Framework

The induced drag from the wing-structure solver can be used as an objective function in an optimization framework similar to that shown in Fig. 5. Any of the parameters from Equations (2), (5), (7), or (15) could be used as design variables. However, in this study, we will use only the lift distribution (B_n) and wingspan (b). Note that in the previous sections, the lift distribution is assumed to be spanwise symmetric ($B_n = 0$ for all even n). Therefore, for the remainder of this study, we will assume that the even Fourier coefficients are identically zero.

The optimization process is summarized as follows: an initial guess is made for the design variables b and B_n; the wing-structure weight and induced drag are computed using the methodology explained in the previous section; the design variables are updated using an optimization method of the user's choosing,

subject to relevant constraints; and the updated design variables are fed back to the wing-structure weight solver. The process is repeated until the induced drag converges within some specified tolerance. Because the relationship between induced drag and the design variables is well behaved, any gradient-based method with appropriate constraints should be adequate for updating the design variables b and B_n. The method used in this study for updating b and B_n is discussed in the following section.

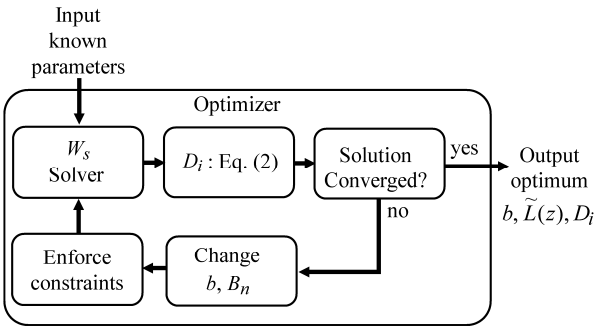


Figure 5. Example optimization framework for minimizing induced drag using wingspan and lift distribution.

The choice of design constraints can have a significant impact on the minimum-induced-drag solution[47,48]. In this study, we will consider only a few example constraints proposed by Phillips et al.[47,48], including an all-positive spanwise lift distribution, fixed net weight, and fixed wing loading. We will also constrain spar height h and width w, as explained in Ref. (53), to ensure that the spar fits within the local airfoil section. The optimization problem can be summarized as follows:

$$\begin{aligned}
\text{minimize :} \quad & D_i \\
\text{with respect to :} \quad & b,\ B_n \ (n \text{ odd}) \\
\text{subject to :} \quad & L - W = 0 \\
& \sigma_{\max} - \sigma(z) \geq 0 \\
& \delta_{\max} - \delta(b/2) \geq 0 \\
& \widetilde{L}(z) \geq 0 \\
& 1 - h(z)/t_{\max}(z) \geq 0 \\
& w_{\max}/c - w(z)/c(z) \geq 0 \\
& W_{n_0} - W_n = 0 \\
& \widetilde{W}_{n_0}(z) - \widetilde{W}_n(z) = 0 \\
& (W/S)_0 - W/S = 0
\end{aligned} \tag{20}$$

where the subscript 0 indicates that the parameter value is prescribed. The first three constraints in Equation (20) are enforced implicitly in Equations (2), (5), and (15). The remaining constraints can be enforced as explained in Ref. (53).

5.0 Results

As an example of minimizing induced drag for a wing with a non-rectangular planform and a net-weight distribution other than Equation (9), consider the NASA Ikhana airframe[54-57]. Ikhana has a linearly-tapered wing with a wingspan $b = 66$ ft, an aspect ratio $R_A = 16.296$, and a taper ratio $R_T = 0.421$. A generic instrumentation pod weighing 500 lbf[57] is sometimes mounted at a hard point outboard of the wing root. Assuming that all of the fuel is distributed in fuel bladders that extend to 83.1% semispan[53], the net-weight distribution can be approximated as

$$\tilde{W}_n(z) = Kc(z)^2 \tag{21}$$

where K is a scaling constant that depends on the length of the fuel bladder and the weight of the fuel carried in the wing. Using Equation (21) in Equation (8) gives a relationship that can be solved to find K for a fuel bladder that extends to 83.1% semispan with a given fuel weight W_f, i.e.,

$$W_f = 2K \int_0^{0.831b/2} c(z)^2 \, dz \tag{22}$$

For this study, we will consider two example Ikhana configurations in steady level flight at sea level with a cruise velocity of 287 ft/s[55]. The first configuration has 3000 lbf of fuel distributed according to Equation (21) in fuel bladders spanning 83.1% semispan with no instrumentation pod. This gives a scaling constant $K = 2.8212$. The second example configuration includes a generic instrument pod mounted on each wing at hard points located at 25% semispan that each cover 1 ft spanwise. To maintain the same fixed net weight as the no-pod configuration, the fuel weight is reduced to 2000 lbf, which gives a scaling constant $K = 1.8808$. The resulting net-weight distribution is shown in Fig. 6. All other parameters for both

configurations are given in Table 1. Note that the values for C_σ and C_δ correspond to a beam with a rectangular cross section, and the values for σ_{max}, E, and γ were selected to be conservative. The maneuvering and hard-landing load limits represent a typical load limit of 2.5 g with a safety factor of 1.5. The maximum deflection is just over 10% of the semispan, which is reasonable for a high-aspect-ratio wing. However, it will be shown that results are sensitive to changes in this parameter.

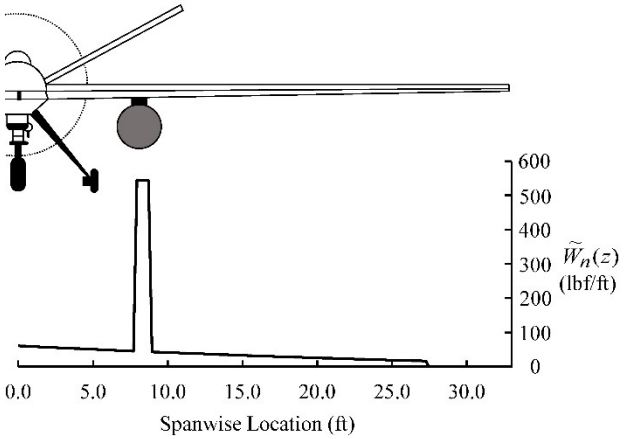


Figure 6. Example net-weight distribution for the Ikhana wing carrying 2000 lbf of fuel and a generic instrumentation pod.

Table 1. Example Specifications for the Ikhana airframe

S (ft^2)	267.3
b (ft)	66
R_T	0.421
t/c	0.1875
C_σ	0.165
C_δ	0.653
σ_{max} (psi)	15.0×10^3
δ_{max} (ft)	3.5
E (psi)	10.0×10^6
γ (lbf/in^3)	0.10
n_m	3.75
n_g	3.75
W_r (lbf)	4500
W_n (lbf)	7500
W_f (No Pod) (lbf)	3000
W_f (With Pod) (lbf)	2000
Pod Weight (lbf)	500 (×2)
ρ (slug/ft^3)	0.0023769
V_∞ (ft/s)	287.0

Wings with taper ratios near $R_T = 0.4$ produce a nearly elliptic lift distribution with no aerodynamic or geometric twist[58,59]. Therefore, we will use the elliptic lift distribution for the baseline design. The solver described in Section 4.1 predicts a wing-structure weight of 1008.4 lbf and induced drag of 54.040 lbf for the baseline no-pod configuration. The total weight is 8508.4 lbf, and the wing loading is 31.831. For the baseline pod configuration, the solver predicts a wing-structure weight of 1080.5 lbf, giving a total weight of 8580.5 lbf and a wing loading of 32.101. The induced drag is 54.959 lbf. A summary of the results for the baseline design is included in Table 2.

5.1 Minimizing Induced Drag

The lift distribution, wingspan, and wing-structure weight that minimize induced drag were found using the framework from Fig. 5, in conjunction with the SciPy* implementation of the Sequential Least-Squares Programming (SLSQP) method[60]. Using SLSQP, the nonlinear constrained optimization problem is cast as an approximate linear least squares problem around the initial design variables $\boldsymbol{x}$. This problem is solved to give an update for the design variables $\Delta\boldsymbol{x}$. The original problem is then recast as a linear least squares problem around the updated point $\boldsymbol{x} + \Delta\boldsymbol{x}$, and the process is repeated until $\Delta\boldsymbol{x}$ falls below a specified tolerance. Gradients for the objectives and constraints are calculated using finite differencing. For additional details, see Ref. (60).

The wing loading is fixed at 31.831 for the no-pod Ikhana configuration and at 32.101 for the pod configuration. The net weight for both configurations is fixed at $W_n = 7500$ lbf. A spar-width constraint of $w/c \le 0.1$ is also imposed. The wingspan b and the Fourier coefficients B_n that define the lift distribution are the design variables. For the results shown here, the Fourier series is truncated at $n = 29$.

The optimum lift distribution for each configuration is shown in Fig. 7, along with five reference lift distributions labeled *a*, *b*, *c*, *d*, and *e*. Curve *a* is the elliptic lift distribution. Curve *b* is Prandtl's 1933 lift distribution[39]. Curves *c* and *d* are the optimum lift distributions found by Phillips et al.[48] for the stress- and deflection-limited designs, respectively, of a rectangular wing with fixed wing loading and the weight

* docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html

distribution given by Equations (9) and (10). Curve *e* is the optimum lift distribution found by Taylor and Hunsaker[49] for the stress-limited design of a tapered wing with fixed wing loading, the weight distribution given by Equations (9) and (10), and a taper ratio of $R_T = 0.4$. Additional optimization results are summarized in Table 2. Note that in this study, we have fixed the taper ratio to $R_T = 0.421$ for all configurations. Therefore, the optimum solutions shown in Table 2 have a different root and tip chord than the baseline configuration.

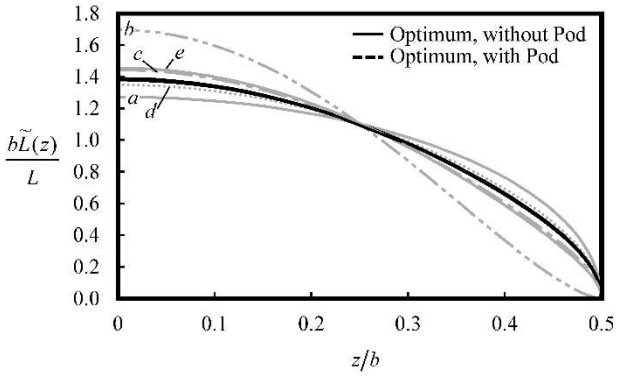


Figure 7. Solutions for the lift distributions that minimize induced drag for the example no-pod and pod configurations of the NASA Ikhana airframe.

Table 2. Example optimization results for the NASA Ikhana airframe

	Without Pod		With Pod	
	Baseline	Optimum	Baseline	Optimum
b (ft)	66	78.083	66	77.084
S (ft^2)	267.3	298.10	267.3	296.35
R_A	16.296	20.453	16.296	20.050
W_s (lbf)	1008.4	1988.6	1080.5	2013.1
D_i (lbf)	54.040	49.213	54.959	50.588
$(w/c)_{\max}$	0.037602	0.072507	0.039047	0.070664
B_3	0	−0.091066	0	−0.084530
B_5	0	1.6121×10^{-3}	0	1.2429×10^{-3}
B_7	0	2.9248×10^{-4}	0	2.6259×10^{-4}
B_9	0	-5.1777×10^{-6}	0	-3.5980×10^{-5}
B_{11}	0	1.2718×10^{-5}	0	1.1619×10^{-5}
B_{13}	0	-5.1777×10^{-6}	0	-4.7294×10^{-6}
B_{15}	0	2.3058×10^{-6}	0	2.1291×10^{-6}
B_{17}	0	-1.3044×10^{-6}	0	-1.1761×10^{-6}
B_{19}	0	6.1712×10^{-7}	0	5.7982×10^{-7}
B_{21}	0	-4.8380×10^{-7}	0	-4.2720×10^{-7}
B_{23}	0	1.8249×10^{-7}	0	1.6479×10^{-7}
B_{25}	0	-2.3663×10^{-7}	0	-2.1818×10^{-7}
B_{27}	0	3.9513×10^{-8}	0	3.3079×10^{-8}
B_{29}	0	-1.4703×10^{-7}	0	-1.3633×10^{-7}

Figure 7 shows that the lift distributions that minimize induced drag for the no-pod and pod configurations are nearly identical, and both lift distributions are noticeably non-elliptic. Table 2 shows that the magnitude of the Fourier coefficients decreases rapidly as n increases. The same trend is shown in Refs. (49) and (51). Both lift distributions are primarily dominated by B_3, with $B_3 = -0.091066$ for the no-pod configuration and $B_3 = -0.084530$ for the pod configuration. These values fall near the theoretical optimum $B_3 = -0.059716$ for the deflection-limited design of a rectangular wing with fixed wing loading[47]. Indeed, both optimum Ikahana designs are deflection-limited.

The reader is reminded that in order to obtain any of the lift distributions in Fig. 7, the wing must be twisted. For an unswept wing with any given planform shape, the twist distribution required to produce a desired lift distribution, specified by B_n, can be computed using the method shown by Phillips and Hunsaker[50]. However, in this study we assume that the wing is correctly twisted to produce the desired lift distribution.

From Table 2, we see that for the no-pod configuration, using the optimum lift distribution allows an increase in wingspan of 18.31%, an increase in wing-structure weight of 97.21%, and results in a reduction in induced drag of 8.93% over the baseline no-pod configuration. For the pod configuration, the optimum lift distribution allows an increase in wingspan of 16.79%, an increase in wing-structure weight of 86.32%, and a reduction in induced drag of 7.95% over the baseline pod configuration.

The wing-structure weight distributions for the baseline Ikhana designs and the optimum designs are shown in Fig. 8, along with their corresponding planforms. Although Ikhana has a non-rectangular planform and a weight distribution other than Equation (9), the optimum wing-structure weight for each configuration is just over 26% of the net weight. This agrees relatively well with the theoretical optimum wing-structure weight of $W_s = W_n/4$[48] for the deflection-limited design of a rectangular wing with the weight distribution given by Equation (9).

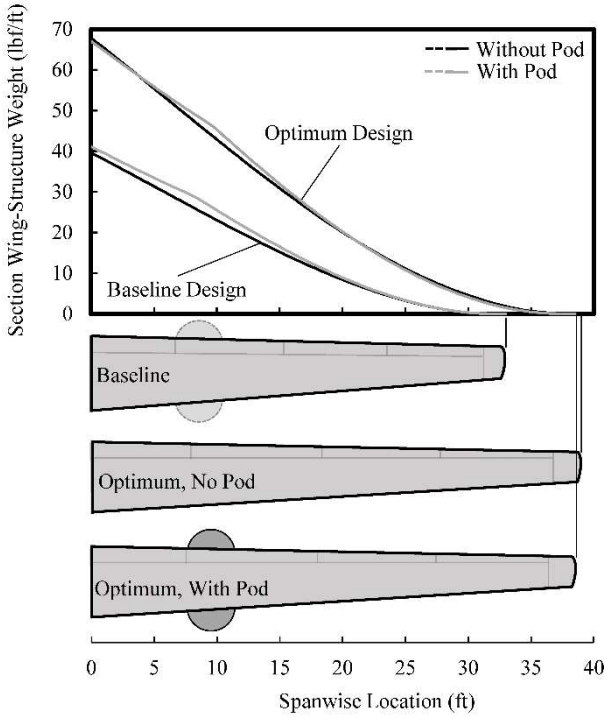


Figure 8. Wing-structure weight distributions and corresponding planforms for the baseline design and optimum design of the example no-pod configuration and pod configuration of the NASA Ikhana airframe.

Induced-drag contours around the optimum design for each example Ikhana configuration are shown in Fig. 9 as a function of the design variables b and B_3. In reality, the lift distribution is a function of n Fourier coefficients, and the design space is more than n-dimensonal. However, because the optimum lift distribution for each Ikhana configuration is dominated by B_3, we approximate the lift distribution using B_3 alone. Note that the induced-drag contours are not smooth at low wingspans, since the wing design transitions from stress-limited to deflection-limited at a low wingspan for each Ikhana configuration.

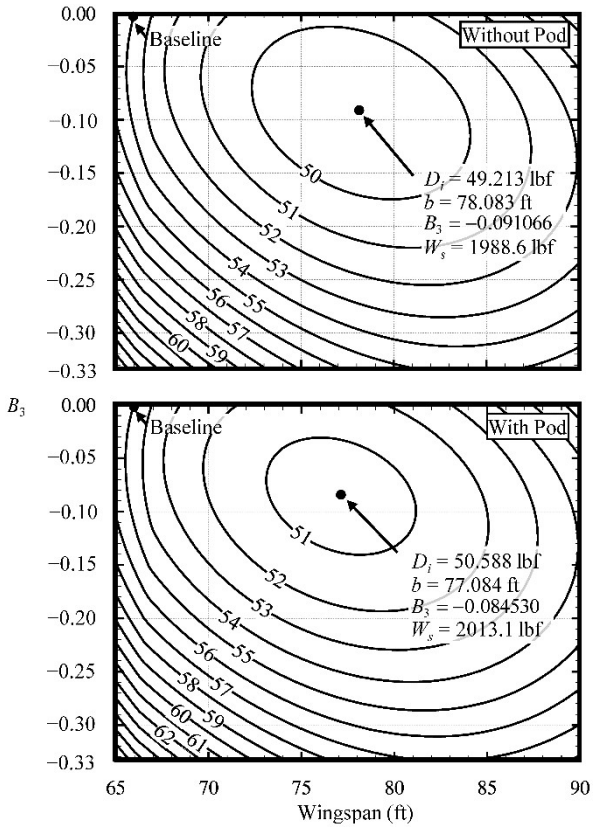


Figure 9. Induced-drag contours for the example no-pod configuration and pod configuration of the NASA Ikhana airframe.

Figure 9 gives some insight into the relative influence of the wingspan, weight, and lift distribution on the induced drag at different points in the design space. For example, for both Ikhana configurations, the induced drag is much more sensitive to changes in wingspan than it is to changes in lift distribution around the baseline design. Since the wing-structure weight typically increases as B_3 and b increase, it is more

advantageous to increase the wingspan and the weight than to decrease the weight by changing the lift distribution near the baseline design. On the other hand, there are regions in the design space where reducing the weight by changing the lift distribution gives a greater reduction in induced drag than changing the wingspan.

The characteristics of the design space depend on the wing configuration, and a figure like Fig. 9 can require more than 100,000 function evaluations. However, using the methods presented in this paper, Fig. 9 was produced in seconds. Understanding of the design space during early design phases can facilitate rapid conceptual optimization and reveal important aspects of the design that cannot be easily seen using high-fidelity methods alone.

5.2 Sensitivity of Optimum Solution to Design Parameters

To illustrate the sensitivity of the optimum solutions in this paper to changes in design parameters, Fig. 10 shows the percent change in the minimum induced drag, optimum wingspan, optimum B_3, and optimum wing-structure weight as a function of the percent change in pod location, average S_b, and the parameters W_r, W/S, and R_T for the pod configuration of the NASA Ikhana airframe. The percent change in pod location is measured in percent semispan.

The plots for W_r and pod location in Fig. 10 show that the optimum lift distribution, characterized by B_3, is most sensitive to the weight distribution. As the pod is shifted away from the wing root and the root weight decreases, the value for B_3 also decreases. This corresponds to a less-elliptic lift distribution, which results in an increase in the wingspan and lower induced drag. This supports the result found by Phillips et al.[47] that the optimum root weight is given by Equation (10), which predicts that the theoretical optimum root weight for Ikhana is close to $W_r \approx 3500$ lbf. The lift distribution is not sensitive to changes in average S_b or W/S, and B_3 only changes by about ± 1% with ± 10% changes in R_T, which agrees with the observation made by Taylor and Hunsaker[49] that the optimum lift distribution is relatively insensitive to the taper ratio.

Figure 10 also shows that the wing-structure weight does not change with changes in average S_b and W/S, and it changes by less than ± 0.65% with ± 10% changes in pod location, W_r, and R_T. This supports

the analytic solutions found by Phillips et al.[48] and Taylor and Hunsaker[49] that the optimum wing-structure weight is independent of all other design parameters.

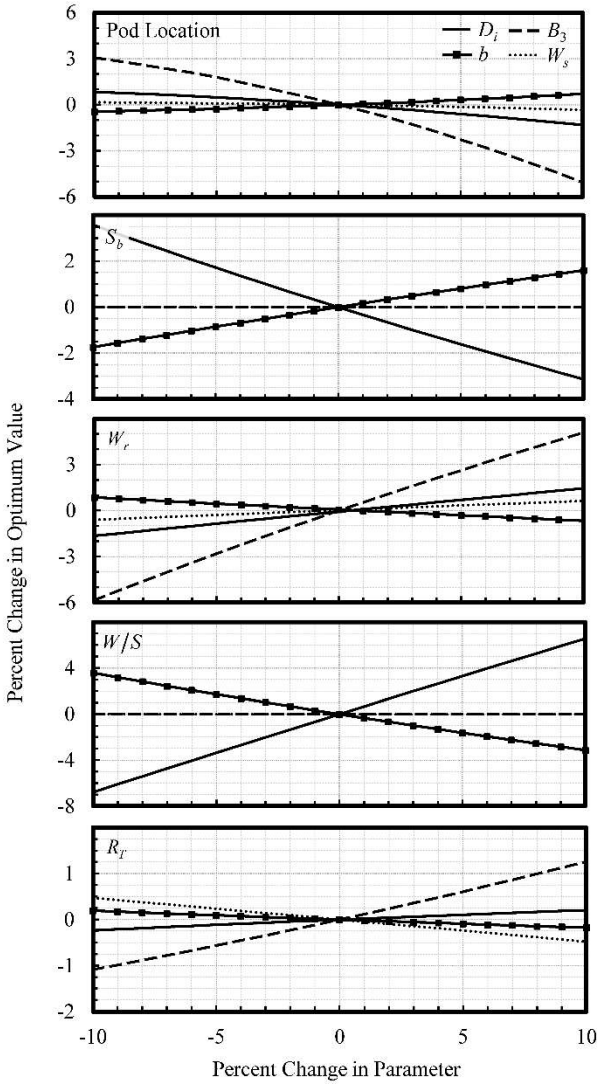


Figure 10. Percent change in minimum induced drag and optimum wingspan, B_3, and wing-structure weight with change in pod location and the parameters S_b, W_r, W/S, and R_T for the example Ikhana pod configuration.

Only the optimum wingspan and corresponding induced drag are affected by changes in average S_b and W/S. For S_b, this is not surprising, since increasing S_b means that less weight is required to support the bending moments. This allows for larger increases in wingspan with smaller corresponding increases in wing-structure weight. For the range of S_b shown, the optimum design is deflection-limited, which means

that S_b is inversely proportional to γ and directly proportional to C_δ, E, and $\delta_{\max}$, as shown in Equation (15). Therefore, the sensitivities shown in Fig. 10 for S_b are also characteristic of the sensitivities for C_δ, E, $\delta_{\max}$, and the quantity $1/\gamma$.

The results in this section show how the methods presented in this paper can be used for design-space exploration. Because the methods are fast, they can be used to rapidly visualize the coupled aerostructural design space and obtain solution sensitivities to various design parameters. It should be remembered that the results shown here are only valid for the two example configurations of the NASA Ikhana airframe given in Table 1. Nevertheless, the methods presented in this paper can be used for any unswept planar wing with arbitrary planform and weight distribution to rapidly iterate on possible design concepts.

6.0 Conclusions

Low-fidelity methods are valuable for rapid aerostructural optimization during the conceptual and preliminary design phases. However, most modern aerostructural methods use mid- and high-fidelity solvers, which are better suited for later design phases. The majority of analytic and low-fidelity aerostructural optimization methods are limited in application to wings with specific planforms and weight distributions. Here, a low-fidelity numerical method has been presented that includes more general approximations corresponding to arbitrary planforms and weight distributions. The method uses an iterative solver to determine the wing-structure weight and induced drag for a given lift distribution and wingspan. The solver is used within an optimization framework for rapid design-space exploration and optimization.

Section 5.0 shows an example application of the method presented in this paper to two configurations of the NASA Ikhana airframe. A summary of the optimization results, including the optimum wingspans, wing-structure weights, and lift distributions are given in Table 2. The optimum lift distributions for both Ikhana configurations are shown in Fig. 7. It has been shown that the optimum lift distributions for the Ikhana configurations are very similar to the analytic optimum lift distribution for a rectangular wing with the ideal weight distribution given in Equation (9). The optimum wing-structure weight for each Ikhana configuration is also in good agreement with theoretical solutions.

A visualization of the design space for each Ikhana configuration is shown in Fig. 9. The relative influence of the wingspan, lift distribution, and wing-structure weight depend on the location of the design in the design space. Figure 10 shows the sensitivities of the design values around the optimum solution to changes in pod location, proportionality coefficient, root weight, wing loading, and taper ratio for the pod configuration of the Ikhana airframe. The optimum wingspan is most sensitive to the proportionality coefficient and wing loading, and the optimum lift distribution is most sensitive to the weight distribution. The optimum wing-structure weight is nearly independent of all other parameters. For the Ikhana configurations considered here, the optimum design allows a wingspan increase of up to 18.31%, an increase in wing-structure weight of up to 97.21%, and a reduction in induced drag of up to 8.93% over the baseline Ikhana configuration. All results were obtained in a matter of seconds.

It should be remembered that the methods presented here were derived using the assumptions associated with lifting-line theory, including wing planarity, zero sweep, and moderate to high aspect ratio. For other wing designs, modifications to these methods may be needed. However, the methods presented here are useful for many practical aircraft configurations. In early design phases, these methods can be used for rapid conceptual optimization and visualization of the design space. These results can provide important insight into the effects of the wing aerodynamic and structural properties and the wing weight distribution on the minimum-induced-drag design.

Acknowledgements

This material is partially based upon work supported by NASA under Grant 80NSSC18K1696 issued by the Aeronautics Research Mission Directorate through the 2018 NASA Fellowship Activity with Nhan Nguyen as the NASA Technical Advisor.

References

1. PHILLIPS, W.F. Lifting-Line Analysis for Twisted Wings and Washout-Optimized Wings, *Journal of Aircraft*, 2004, **41**, (1), pp. 128–136. (doi:10.2514/1.262)

2. PHILLIPS, W.F., FUGAL, S.R. and SPALL, R.E. Minimizing Induced Drag with Wing Twist, Computational-Fluid-Dynamics Validation, *Journal of Aircraft*, 2006, **43**, (2), pp. 437–444. (doi:10.2514/1.15089)

3. GALLAY, S. and LAURENDEAU, E. Preliminary-Design Aerodynamic Model for Complex Configurations Using Lifting-Line Coupling Algorithm, *Journal of Aircraft*, 2016, **53**, (4), pp. 1145–1159. (doi:10.2514/1.C033460)

4. PHILLIPS, W.F. and HUNSAKER, D.F. Lifting-Line Predictions for Induced Drag and Lift in Ground Effect, *Journal of Aircraft*, 2013, **50**, (4), pp. 1226–1233. (doi:10.2514/1.C032152)

5. WICKENHEISER, A. and GARCIA, E. Aerodynamic Modeling of Morphing Wings Using an Extended Lifting-Line Analysis, *Journal of Aircraft*, 2007, **44**, (1), pp. 10–16. (doi:10.2514/1.18323)

6. PHILLIPS, W.F. and SNYDER, D.O. Modern Adaptation of Prandtl's Classic Lifting-Line Theory, *Journal of Aircraft*, 2000, **37**, (4), pp. 662–670. (doi:10.2514/2.2649)

7. RASMUSSEN, M.L. and SMITH, D.E. Lifting-Line Theory for Arbitrarily Shaped Wings, *Journal of Aircraft*, 1999, **36**, (2), pp. 340–348. (doi:10.2514/2.2463)

8. BERA, R.K. Some remarks on the solution of the lifting line equation, *Journal of Aircraft*, 1974, **11**, (10), pp. 647–648. (doi:10.2514/3.44397)

9. PRANDTL, L. Tragflügel Theorie, *Nachricten von der Gesellschaft der Wissenschaften zu Göttingen*, Geschäeftliche Mitteilungen, Klasse, 1918, pp. 451–477.

10. PRANDTL, L. Applications of Modern Hydrodynamics to Aeronautics, NACA TR-116, June 1921.

11. MUNK, M.M. The Minimum Induced Drag of Aerofoils, NACA TR-12, 1923.

12. BURDETTE, D.A. and MARTINS, J.R.R.A. Impact of Morphing Trailing Edges on Mission Performance for the Common Research Model, *Journal of Aircraft*, 2019, **56**, (1), pp. 369-384. (doi:10.2514/1.C034967)

13. KENWAY, G.K.W. and MARTINS, J.R.R.A. Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration, *Journal of Aircraft*, 2014, **51**, (1), pp. 144-160. (doi:10.2514/1.C032150)

14. LIEM, R.P., MARTINS, J.R.R.A. and KENWAY, G.K.W. Expected drag minimization for aerodynamic design optimization based on aircraft operational data, *Aerospace Science and Technology*, 2017, **63**, pp. 344-362. (doi:10.1016/j.ast.2017.01.006)

15. MADER, C.A. and MARTINS, J.R.R.A Stability-Constrained Aerodynamic Shape Optimization of Flying Wings, *Journal of Aircraft*, 2013, **50**, (5), pp. 1431-1449. (doi:10.2514/1.C031956)

16. ZHANG, Z.J., KHOSRAVI, S. and ZINGG, D.W. High-fidelity aerostructural optimization with integrated geometry parameterization and mesh movement, *Structural and Multidisciplinary Optimization*, 2017, **55**, pp. 1217-1235. (doi:10.1007/s00158-016-1562-7)

17. JAMES, K.A., KENNEDY, G.J. and MARTINS, J.R.R.A. Concurrent aerostructural topology optimization of a wing box, *Computers and Structures*, 2014, **134**, pp. 1-17. (doi:10.1016/j.compstruc.2013.12.007)

18. JANSEN, P.W., PEREZ, R.E. and MARTINS, J.R.R.A. Aerostructural Optimization of Nonplanar Lifting Surfaces, *Journal of Aircraft*, 2010, **47**, (5), pp. 1490–1503. (doi: 10.2514/1.44727)

19. TING, E., CHAPARRO, D., NGUYEN, N. and FUJIWARA, G.E.C. Optimization of Variable-Camber Continuous Trailing-Edge Flap Configuration for Drag Reduction, *Journal of Aircraft*, 2018, **55**, (6), pp. 2217-2239. (doi:10.2514/1.C034810)

20. LUNDRY, J.L. Minimum Swept-Wing Induced Drag with Constraints on Lift and Pitching Moment, *Journal of Aircraft*, 1967, **4**, pp. 73–74. (doi:10.2514/3.43797)

21. LISSAMAN, P.B.S. and LUNDRY, J.L. A Numerical Solution for the Minimum Induced Drag of Nonplanar Wings, *Journal of Aircraft*, 1968, **5**, pp. 17–21. (doi:10.2514/3.43901)

22. ASHENBERG, J. and WEIHSRADIUS, D. Minimum Induced Drag of Wings with Curved Planform, *Journal of Aircraft*, 1984, **21**, pp. 89–91. (doi:10.2514/3.56733)

23. ROKHSAZ, K. Effect of Viscous Drag on Optimum Spanwise Lift Distribution, *Journal of Aircraft*, 1993, **30**, pp. 152–154. (doi:10.2514/3.46328)

24. DEMASI, L. Induced Drag Minimization: A Variational Approach Using the Acceleration Potential, *Journal of Aircraft*, 2006, **43**, pp. 669–680. (doi:10.2514/1.15982)

25. DEMASI, L. Erratum on Induced Drag Minimization: A Variational Approach Using the Acceleration Potential, *Journal of Aircraft*, 2006, **43**, p. 1247. (doi:10.2514/1.26648)

26. DEMASI, L. Investigation on the Conditions of Minimum Induced Drag of Closed Wing Systems and C-Wings, *Journal of Aircraft*, 2007, **44**, pp. 81–99. (doi:10.2514/1.21884)

27. DEMASI, L., DIPACE, A., MONEGATO, G. and CAVALLARO, R. Invariant Formulation for the Minimum Induced Drag Conditions of Nonplanar Wing Systems, *AIAA Journal*, 2014, **52**, pp. 2223–2240. (doi:10.2514/1.J052837)

28. DEMASI, L., MONEGATO, G. and CAVALLARO, R. Minimum Induced Drag Theorems for Multiwing Systems, *AIAA Journal*, 2017, **55**, pp. 3266–3287. (doi:10.2514/1.J055652)

29. GRAY, W.L. and SCHENK, K.M. A Method for Calculating the Subsonic Steady-State Loading on an Airplane with a Wing of Arbitrary Planform and Stiffness, NACA TR-3030, December, 1953.

30. CRAIG, A.P. and MCLEAN, D.J. Spanload Optimization for Strength Designed Lifting Surfaces, AIAA 88-2512, 6th Applied Aerodynamics Conference, Williamsburg, Virginia, 5–8 June 1988.

31. HAFTKA, R.T. Optimization of Flexible Wing Structures Subject to Strength and Induced Drag Constraints, *AIAA Journal*, 1977, **14**, (8), pp. 1101–1106. (doi: 10.2514/3.7400)

32. GROSSMAN, B., GURDAL, Z., STRAUCH, G.J., EPPARD, W.M. and HAFTKA, R.T. Integrated Aerodynamic/Structural Design of a Sailplane Wing, *Journal of Aircraft*, 1988, **25**, (9), pp. 855–860. (doi: 10.2514/3.45670)

33. WAKAYAMA, S. and KROO, I.M. Subsonic Wing Planform Design Using Multidisciplinary Optimization, *Journal of Aircraft*, 1995, **32**, (4), pp. 746–753. (doi: 10.2514/3.46786)

34. CALDERON, D.E., COOPER, J.E., LOWENBERG, M. and NEILD, S.A. On the Effect of Including Geometric Nonlinearity in the Sizing of a Wing, AIAA 2018-1680, 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Kissimmee, Florida, 8–12 January, 2018.

35. IGLESIAS, S. and MASON, W.H. Optimum Spanloads Incorporating Wing Structural Weight, AIAA 2001-5234, 1st Aircraft, Technology Integration, and Operations Forum, Los Angeles, California, 16–18 October 2001.

36. WROBLEWSKI, G.E. and ANSELL, P.J. Prediction and Experimental Evaluation of Planar Wing Spanloads for Minimum Drag, *Journal of Aircraft*, 2017, **54**, pp. 1664–1674. (doi:10.2514/1.C034156)

37. MCGEER, T. Wing Design for Minimum Drag with Practical Constraints, *Journal of Aircraft*, 1984, **21**, pp. 879–886. (doi:10.2514/3.45058)

38. GOPALARATHNAM, A. and NORRIS, R.K. Ideal Lift Distributions and Flap Angles for Adaptive Wings, *Journal of Aircraft*, 2009, **46**, (2), pp. 562–571. (doi: 10.2514/1.38713)

39. PRANDTL, L. Über Tragflügel kleinsten induzierten Widerstandes, *Zeitschrift für Flugtechnik und Motorluftschiffahrt*, 1933, **24**, (11), pp. 305–306.

40. JONES, R.T. The Spanwise Distribution of Lift for Minimum Induced Drag of Wings Having a Given Lift and a Given Bending Moment, NACA TR-2249, December 1950.

41. PATE, D.J. and GERMAN, B.J. Lift Distributions for Minimum Induced Drag with Generalized Bending Moment Constraints, *Journal of Aircraft*, 2013, **50**, pp. 936–946. (doi:10.2514/1.C032074)

42. DEYOUNG, J. Minimization Theory of Induced Drag Subject to Constraint Conditions, NASA CR-3140, June 1979.

43. JONES, R.T. and LASINSKI, T.A. Effect of Winglets on the Induced Drag of Ideal Wing Shapes, NASA TM-81230, Sept. 1980.

44. KLEIN, A. and VISWANATHAN, S.P. Minimum Induced Drag of Wings with Given Lift and Root-Bending Moment, *Zeitschrift fur Angewandte Mathematik und Physik*, 1973, **24**, pp. 886–892.

45. KLEIN, A. and VISWANATHAN, S.P. Approximate Solution for Minimum Induced Drag of Wings with Given Structural Weight, *Journal of Aircraft*, 1975, **12**, (2), pp. 124–126. (doi:10.2514/3.44425)

46. LÖBERT, G. Spanwise Lift Distribution for Forward- and Aft-Swept Wings in Comparison to the Optimum Distribution Form, *Journal of Aircraft*, 1981, **18**, (6), pp. 496-498. (doi:10.2514/3.44717)

47. PHILLIPS, W.F., HUNSAKER, D.F. and JOO, J.J. Minimizing Induced Drag with Lift Distribution and Wingspan, *Journal of Aircraft*, 2019, **56**, (2), pp. 431-441. (doi:10.2514/1.C035027)

48. PHILLIPS, W.F., HUNSAKER, D.F. and TAYLOR, J.D. Minimizing Induced Drag with Weight Distribution, Lift Distribution, Wingspan, and Wing-Structure Weight, AIAA 2019-3349, AIAA Aviation 2019 Forum, Dallas, Texas, 17-21 June 2019. (doi: 10.2514/6.2019-3349)

49. TAYLOR, J.D. and HUNSAKER, D.F. Minimum Induced Drag for Tapered Wings Including Structural Constraints, *Journal of Aircraft*, 2020, **57**, (4), pp. 782-786. (doi:10.2514/1.C035757)

50. PHILLIPS, W F. and HUNSAKER, D.F. Designing Wing Twist or Planform Distributions for Specified Lift Distributions, *Journal of Aircraft*, 2019, **56**, (2), pp. 847-849. (doi:10.2514/1.C035206)

51. TAYLOR, J.D., HUNSAKER, D.F. and JOO, J.J. Numerical Algorithm for Wing-Structure Design, AIAA 2018-1050, 2018 AIAA Aerospace Sciences Meeting, Kissimmee, Florida, 8-12 January 2018. (doi: 10.2514/6.2018-1050)

52. RICHARDSON, L.F. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam, 1910, *Philosophical Transactions of the Royal Society of London*, Series A, **210**, pp. 307-357.

53. TAYLOR, J.D. and HUNSAKER D.F. Numerical Method for Rapid Aerostructural Design and Optimization, AIAA 2020-3175, AIAA Aviation 2020 Virtual Forum, 15-19 June 2020. (doi: 10.2514/6.2020-3175)

54. MERLIN, P. W., 'Don't Fear the Reaper,' Ikhana: Unmanned Aircraft System Western States Fire Missions, *Monographs in Aerospace History*, 2009, **44**, National Aeronautics and Space Administration, NASA SP-2009-4544, pp. 1-18.

55. COBLEIGH, B. R., Ikhana: A NASA UAS Supporting Long Duration Earth Science Missions, NASA TM-2007-214614, March 2007.

56. KO, W.L., RICHARDS, W.L. and FLEISCHER, V.T. Applications of Ko Displacement Theory to the Deformed

Shape Predictions of the Doubly-Tapered Ikhana Wing, NASA TP-2009-214652, October 2009.

57. Ikhana: Unmanned Science and Research Aircraft System, NASA FS-097-DFRC, 2014.

58. GLAUERT, H. *The Elements of Aerofoil and Airscrew Theory*, Cambridge University Press, London, 1926.

59. PHILLIPS, W.F. Incompressible Flow over Finite Wings, *Mechanics of Flight*, 2nd ed., Wiley, Hoboken, NJ, 2010, pp. 46-94.

60. KRAFT, D. A software package for sequential quadratic programming, DLR German Aerospace Center – Institute for Flight Mechanics, DFVLR-FB 88-28, Koln, Germany, 1988.

CHAPTER 4

Comparison of Theoretical and Multi-Fidelity Optimum Aerostructural Solutions for Wing Design

Jeffrey D. Taylor* and Douglas F. Hunsaker†
Utah State University, Logan, Utah 84322-4130

As contemporary aerostructural research for aircraft design trends toward high-fidelity computational methods, aerostructural solutions based on theory are often neglected or forgotten. In fact, in many modern aerostructural wing optimization studies, the elliptic lift distribution is used as a reference in place of theoretical aerostructural solutions with more appropriate constraints. In this paper, we review several theoretical aerostructural solutions that could be used as reference cases for wing design studies, and we compare them to high-fidelity solutions with similar constraints. Solutions are presented for studies with 1) constraints related to the wing integrated bending moment, 2) constraints related to the wing root bending moment, and 3) structural constraints combined with operational constraints related to either wing stall or wing loading. It is shown that, under appropriate design constraints, theoretical solutions for the optimum lift distribution may capture aerostructural coupling sufficiently to serve as appropriate reference cases for higher fidelity solvers. A comparison of theoretical and high-fidelity solutions for the optimum wingspan and corresponding drag reveals important insights into the effects of certain aerodynamic and structural parameters

* PhD Candidate, Mechanical and Aerospace Engineering, 4130 Old Main Hill, AIAA Student Member
† Assistant Professor, Mechanical and Aerospace Engineering, 4130 Old Main Hill, AIAA Senior Member

This paper was published in *Journal of Aircraft* in 2022 as:
Taylor, J. D., and Hunsaker, D. F., "Comparison of Theoretical and Multi-Fidelity Optimum Aerostructural Solutions for Wing Design," *Journal of Aircraft*, Vol. 59, No. 1, January 2022, pp. 103-116
(doi:10.2514/1.C036374)

and constraints on the aerodynamic and structural coupling involved in aerostructural wing design and optimization.

Nomenclature

A	beam cross-sectional area
B_n	Fourier coefficients in the lifting-line solution for the dimensionless section-lift distribution
b	wingspan
C_D	global drag coefficient
C_L	global lift coefficient
$\widetilde{C}_{L_{\max}}$	maximum lift coefficient of the local airfoil section
C_δ	shape coefficient for the deflection-limited design, Eq. (5)
C_σ	shape coefficient for the stress-limited design, Eq. (4)
c	local wing section chord length
D	total drag
D_i	induced drag
D_{ref}	reference drag
E	modulus of elasticity of the beam material
h	height of the beam cross-section
I	beam section moment of inertia
J	aerostructural cost function based on a linear combination of drag and weight
L	total lift
$\widetilde{L}$	local wing section lift
$\widetilde{M}_b$	local wing section bending moment
n_a	load factor, g
n_g	limiting load factor at the hard-landing design limit
n_m	limiting load factor at the maneuvering-flight design limit

R_T	wing taper ratio
$s_{g,TO}$	takeoff ground roll
$s_{g,L}$	landing ground roll
S_b	proportionality coefficient between $\widetilde{W}_s(y)$ and $\widetilde{M}_b(y)$ having units of length squared
S_W	wing planform area
T	thrust
t_f	landing brake-engagement reaction time
t_r	takeoff rotation time
$t_{\max}$	maximum thickness of the local airfoil section
V_{stall}	stall speed
V_∞	freestream airspeed
W	aircraft gross weight
W_n	aircraft net weight, defined as $W - W_s$
W_s	total weight of the wing structure required to support the wing bending-moment distribution
W_{ref}	reference weight
$\widetilde{W}_n$	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
y	spanwise coordinate relative to the midspan
β	relative weighting coefficient in the linear combination of drag and weight
γ	specific weight of the beam material
$\delta_{\max}$	maximum wing deflection
μ_r	coefficient of rolling friction between the aircraft landing gear and the ground
ρ	air density
$\sigma_{\max}$	maximum longitudinal stress

I. Introduction

While modern computational tools have enhanced our understanding of finite-wing design, much of our relational understanding between wing design parameters and aerodynamic performance is based on solutions obtained from analytic theories. Designers often rely on insights gained from these theories in the conceptual and preliminary phases of aircraft design. In many cases, solutions based on theory have been shown to be in good agreement with experimental data and computational fluid dynamics [1-8], while providing significantly more mathematical and physical insight than higher fidelity models. In some cases, the applicability of a theoretical solution extends far beyond the assumptions and approximations associated with the original theory. For example, the well-known elliptic lift distribution, which minimizes induced drag on an unswept planar wing with fixed weight and wingspan, was first identified in 1918 by Prandtl [9,10] and later by Munk [11] from analytic solutions based on lifting-line theory [9,10], and it is often used today as a reference solution in many multi- and high-fidelity aerodynamic studies. Since 1918, the elliptic lift distribution has appeared repeatedly in analytic, computational, and experimental studies, and it has been shown to be optimal for many complex and unconventional wing designs in both high- and low-speed subsonic flight. Still, the elliptic lift distribution is only optimal under a limited set of aerodynamic design constraints [12-21].

When aerostructural constraints are considered, the elliptic lift distribution is not always optimal. From classical lifting-line theory, the induced drag D_i on a wing in steady-level flight with freestream density ρ and freestream velocity V_∞ can be written as

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

where W is the weight, b is the wingspan, and B_n are Fourier coefficients that define the lift distribution. When weight and wingspan are fixed, Eq. (1) is minimized with the elliptic lift distribution, which has $B_n = 0$ for all n. If the weight and wingspan are allowed to vary, Eq. (1) can be reduced by decreasing the weight and/or increasing the wingspan. However, this cannot be done arbitrarily because the wingspan, lift distribution, and weight are all coupled through the bending moments. Certain non-elliptic lift distributions

can alleviate bending moments, allowing a larger wingspan with little or no increase in wing weight. Thus, the solution found by minimizing Eq. (1) with variable weight and/or wingspan often includes a non-elliptic lift distribution that is the aerostructural analogue of the aerodynamically-optimum elliptic lift distribution [22-32].

Although the induced drag is not generally the main focus in modern aerostructural optimization, most theoretical aerostructural studies primarily focus on minimizing induced drag with a variety of simple structural and operational constraints, and results from these studies can provide significant insight into the aerodynamic and structural coupling involved in aerostructural wing design and optimization [33-43]. In 1933, Prandtl identified a bell-shaped lift distribution that minimizes induced drag on a rectangular wing with fixed gross weight and moment of inertia of gross weight [33]. Independently, Jones [34] sought to minimize induced drag under the constraints of fixed gross lift and root bending moment in cruise. Pate and German [35] constrained the root bending moment at a given off-design lift coefficient but did not allow the wingspan to change. DeYoung [36] used a constraint on the bending moment at a prescribed spanwise location. Following Prandtl's lead [33], Jones and Lasinski [37] sought to minimize induced drag on non-planar wings with constrained integrated bending moment. Klein and Viswanathan [38,39] considered both root and integrated bending moment [38] and included the effects of shear on the wing-structure weight [39]. Extending Prandtl's [33] and Jones and Lasinski's [37] structural constraints, Löbert [40] introduced a constraint based on the ratio of the bending-moment distribution and the wing-section thickness. More recently, Phillips et al. [41,42] and Taylor and Hunsaker [43] extended Prandtl's approach [33] to account for the effects of the planform shape and the wing weight distribution and identified lift distributions that minimize induced drag under constraints of fixed gross weight [41], fixed net weight [42,43], fixed wing loading [41-43], and fixed stall speed [42].

In modern aerostructural literature, these theoretical solutions are seldom revisited. Like most theoretical studies, each of the studies in Refs. [33-43] includes assumptions and approximations that are not fully representative of all aircraft wings, particularly those with unconventional designs. Most modern research in aerostructural design and optimization focuses on high-fidelity computational methods that can

handle complex geometries and design conditions. In the modern aerostructural literature, the elliptic lift distribution is often included as a reference [22-25,44,46-48] in place of theoretical aerostructural solutions with more applicable constraints that include the effects aerodynamic and structural coupling.

The main value of theoretical aerostructural solutions is not that they can or should replace or reduce the use of high-fidelity solutions for aircraft design, but that they can enhance high-level aerostructural insight to inform conceptual design and provide simple validation cases for high-fidelity computational methods. A firm understanding of theoretical solutions can serve as a point of reference for evaluating conceptual designs with aerostructural constraints and for interpreting results from high-fidelity aerostructural solvers. Also, in high-fidelity aerostructural optimization, one of the main challenges is to correctly link computational aerodynamic and structural analysis tools to achieve accurate aerostructural coupling. This process often requires significant effort and introduces many opportunities for error. Therefore, it is valuable to have a simple, known aerostructural solution that includes aerodynamic and structural coupling as a validation case to ensure that coupling between aerodynamic and structural computational components is properly implemented in the development of high-fidelity aerostructural optimization codes. In both cases, the greatest value is obtained when the primary constraints that affect aerodynamic and structural coupling in the high-fidelity study of interest are included or approximated within the theoretical reference solution.

To that end, in this paper, we address two questions: 1) How well do theoretical aerostructural solutions apply to typical real-world aircraft configurations? and 2) Can theoretical solutions serve as appropriate aerostructural reference cases for higher fidelity studies? We address these questions by reviewing solutions from several theoretical aerostructural studies [33,34,36-43] and comparing their solutions for the optimum lift distribution, wingspan, and drag to results from several multi- and high-fidelity computational studies on various practical aircraft configurations with comparable constraints [44-46,49-60]. Results are also compared to the elliptic lift distribution. As will be shown, the answers to the above questions largely depend on the design constraints and assumptions associated with theoretical solutions. However, for certain sets of design constraints, the optimum lift distributions predicted by the theoretical methods

considered here agree well with high-fidelity solutions for various practical wing configurations and flight conditions. It follows that, in certain cases, the optimum lift distributions predicted by theoretical methods can serve as appropriate aerostructural reference solutions for higher-fidelity methods.

Due to differences is design objectives, variables, and flight conditions, fully consistent comparisons between theoretical and high-fidelity aerostructural solutions, especially those for the optimum wingspan and corresponding drag, are often difficult, and sometimes impossible, to make. Because of the complexities associated with aerostructural optimization, analytic solutions often require simplifying assumptions that sometimes neglect important considerations including the effects of viscosity, compressibility, aeroelasticity, composite structures, buckling, and fatigue, among others. Moreover, because of the high computational costs associated with high-fidelity aerostructural optimization, most high-fidelity aerostructural studies report only a few solutions or a single solution, from which relational information is very difficult to obtain. Nevertheless, in this paper, we have sought to highlight important relational considerations by comparing analytic and high-fidelity aerostructural solutions. These comparisons give important insights into how certain parameters and constraints are likely to affect aerostructural wing design.

In some respects, this paper can be thought of as a survey of the subset of theoretical literature concerning aerostructural optimization for minimum drag. However, in this paper, no attempt is made to present a comprehensive review of the complete body of comparable high-fidelity aerostructural literature. Instead, we have selected only a few available solutions from several multi- and high-fidelity studies [44-46,49-60] with constraints that are most comparable to those used in the theoretical studies discussed in the following section.

II. Analytic Aerostructural Solutions

In this section, we will briefly review the key assumptions, constraints, and solutions from several foundational aerostructural studies. In this section and the following sections, solutions are grouped into three major categories: 1) those that include constraints involving the integrated bending

moment [33,37,39,44,45], 2) those with constraints involving the root bending moment [34,36,38,49-52], and 3) those that combine constraints on wing stress and deflection with operational constraints related to either wing stall [42,46,53,59] or wing loading [40-43,54-58,60]. A summary of the key objectives, design variables, and constraints for each of the analytic studies considered here is given in in the appendix.

A. Analytic Solutions with Integrated Bending Moment Constraints

In 1933, Ludwig Prandtl published one of the first known studies [33] involving minimizing drag under structural constraints. In this publication [33], Prandtl presented a method for identifying the optimum lift distribution and wingspan that minimize induced drag, including the effects of the wing weight. Prandtl's study included constraints on the gross lift and the moment of inertia of gross lift, which is derived from the integrated bending moment. To obtain an analytic solution, Prandtl assumed that the wing bending moments are solely due to the lift distribution, regardless of the weight of the wing. Prandtl also assumed that the wing bending moments $\widetilde{M}_b$ are related to the wing-structure weight W_s by a spanwise-invariant proportionality coefficient S_b, i.e.,

$$W_s = \int_0^{b/2} \frac{\widetilde{M}_b(y)}{S_b} dy \tag{2}$$

where b is the wingspan and y is the spanwise coordinate. This assumption best corresponds to rectangular wings. Within the framework of these constraints and assumptions, Prandtl identified a bell-shaped lift distribution that allows a 22.5% larger wingspan and produces 11.1% less induced drag than the elliptic lift distribution with the same wing weight [33].

Prandtl's solution was revisited in 1975 by Klein and Viswanathan [39] and in 1980 by Jones and Lasinski [37]. Klein and Viswanathan noted that the wing-structure weight is not only dependent on the bending-moment distribution, but it also depends on the distribution of shear force in the wing. Thus, in addition to constraints of fixed gross lift and integrated bending moment, Klein and Viswanathan [39] imposed a constraint on the integrated shear force. Their solution results in a 16% larger wingspan and about 7% less induced drag than the elliptic lift distribution for the same wing-structure weight, or about 6% smaller wingspan and 4% more induced drag than Prandtl's solution [33]. Jones and Lasinksi [37]

extended Prandtl's methodology to non-planar wings and considered the effects of winglets. Both Klein and Viswanathan and Jones and Lasinski assumed an arbitrary constant value for S_b in Eq. (2), making their solutions most representative of rectangular wings.

B. Analytic Solutions with Root Bending Moment Constraints

In 1950, R.T. Jones [34] sought to identify the lift distribution that minimizes induced drag from a family of lift distributions that produce a given root bending moment and gross lift. Assuming that the lift distribution is all-positive, Jones found that there exists a triangular-shaped lift distribution that can allow up to a 33% increase in wingspan and a reduction in induced drag of over 15% when compared to the elliptic lift distribution. However, Jones noted that nearly the same induced-drag reduction can be achieved with a 15% increase in wingspan, which, in many cases, is more practical. Thus, Jones [34] reported his "optimum" solution as having a 15% larger wingspan and producing 15% less induced drag than the elliptic lift distribution with the same root bending moment.

In the 1970's, Klein and Viswanathan [38] and DeYoung [36] obtained similar results to those found by Jones in 1950. Klein and Viswanathan [38] identified an optimum lift distribution that corresponds to a 33.3% increase in wingspan and a 15.6% reduction in induced drag over the elliptic lift distribution by modifying Prandtl's 1933 method [33] to include a constraint on the root bending moment, rather than the integrated bending moment. DeYoung [36] obtained the same result from a more general method with a constraint on the bending moment at any given location on the wing. Like the theoretical studies in Refs. [33,37,39], the studies of Jones [34], Klein and Viswanathan [38], and DeYoung [36] include the assumption that the bending moments are only due to the lift distribution. Moreover, by using the root bending moment as a surrogate for wing weight and constraining the root bending moment to a fixed value, each author implicitly assumes that the wing weight is constant.

The analytic solutions given in Refs. [34,36,38] each include a considerably larger wingspan than that resulting from the elliptic lift distribution with the same root bending moment. Because no area constraints are included in any of these studies, the wing areas corresponding to the optimum solutions are also large.

Since more wing area typically results in higher viscous drag, viscous effects can significantly reduce the practical optimality of these solutions.

C. Analytic Solutions Combining Stress and Deflection Limits with Operational Constraints

In each of the analytic studies described thus far, the wingspan is allowed to vary without any constraint on the wing area. However, as pointed out by Iglesias and Mason [31], if no wing-area constraint is imposed, changing the wingspan changes the wing area, which results in a comparison between wings with fundamentally different operational performance characteristics. Since aircraft are typically designed to meet at least one specified performance parameter, it is unhelpful to compare any "optimized" wing to a baseline configuration if the "optimum" wing does not have similar operational performance characteristics as the baseline wing. In order to ensure a fair comparison, Phillips et al. [41,42] suggested that the wing design be constrained so that either the wing loading, which affects several key airspeed requirements, or the stall speed, which is critical for takeoff and landing performance, be fixed.

Phillips et al. [41,42] extended Prandtl's 1933 study by relaxing many of his main assumptions. For example, whereas Prandtl [33] assumed that the wing bending moments are a function of the lift distribution alone, Phillips et al. [41,42] assumed that the bending moments are related to the lift distribution and wing weight distribution according to the relation [41]

$$\widetilde{M}_b(y) = \int_{z'=z}^{b/2} [\widetilde{L}(y') - n_a \widetilde{W}_n(y') - n_a \widetilde{W}_s(y')](y'-y)dy', \quad \text{for } y \geq 0 \tag{3}$$

where n_a is the load factor at the design limit, $\widetilde{W}_n(y)$ is the weight of all non-structural components in the wing, and $\widetilde{L}(y)$ is the section lift distribution. At all points, the wing structure must be designed to support the bending moments encountered during a high-load maneuver and during a negative-load maneuver, such as a hard landing. To obtain analytic results, Phillips et al. [41,42] assumed that the lift distribution is fixed for all flight phases.

Revisiting Prandtl's 1933 assumption that the wing-structure weight is proportional to the bending moments, Phillips et al. [41,42] used simple beam theory to define the proportionality coefficient S_b in terms of the beam geometric and material properties, i.e., [41],

$$W_s = \int_0^{b/2} \frac{\left|\widetilde{M}_b(y)\right|}{S_b(y)} dy; \quad S_b(y) \equiv \frac{C_\sigma (t_{\max}/c) c(y) \sigma_{\max}}{\gamma}, \quad C_\sigma \equiv \frac{2I(h/t_{\max})}{Ah^2} \tag{4}$$

where $t_{\max}/c$ is the wing thickness-to-chord ratio, c is the wing chord, $\sigma_{\max}$ is the maximum allowable stress, γ is the specific weight of the wing-structure material, and I, A, and h are the second moment of inertia, area, and height of the wing structure, respectively. Note that Eq. (4) is analogous to Eq. (2) but includes a limit on the maximum allowable bending stress within the definition of S_b. Thus, Eq. (4) describes the wing-structure weight for the stress-limited design.

Phillips et al. [41] also included deflection constraints by relating the maximum allowable deflection to the maximum allowable stress to give [41]

$$W_s = \int_0^{b/2} \frac{\left|\widetilde{M}_b(y)\right|}{S_b(y)} dy; \quad S_b(y) \equiv \frac{C_\delta E (t_{\max}/c)^2 c(y)^2 \delta_{\max}}{\gamma}, \quad C_\delta \equiv \frac{8I(h/t_{\max})^2}{Ah^2} \tag{5}$$

where E is the modulus of elasticity of the wing-structure material, and $\delta_{\max}$ is the maximum allowable deflection. Thus, Eq. (5) describes the wing-structure weight for the deflection-limited design.

1. Stall-Related Constraints

For most aircraft, the takeoff and landing performance are heavily influenced by the stall speed V_{stall}. For example, FAR regulations dictate that the takeoff speed must be at least 10% higher than the stall speed and that reference landing speed must be 30% higher than the stall speed. Because of this, the stall speed can be constrained to ensure that any optimal wing design maintains similar takeoff and landing performance to the baseline design. Phillips et al. [41,42] defined the stall speed as the speed at which stall begins at any section of the wing. This happens when the local lift coefficient exceeds the maximum lift coefficient $\widetilde{C}_{L_{\max}}$ of the airfoil section. For a rectangular wing with chord c and lift distribution $\widetilde{L}(y)$, this occurs when

$$\frac{\widetilde{L}(y)_{\max}}{\frac{1}{2}\rho V_{\text{stall}}^2 c} = \widetilde{C}_{L_{\max}} \tag{6}$$

Equation (6) shows that for a given lift distribution and freestream density, the stall speed and maximum lift coefficient are related. If $\widetilde{C}_{L_{\max}}$ is fixed, then the chord must change to ensure that the local lift coefficient does not exceed $\widetilde{C}_{L_{\max}}$, which alters the wing area.

Under the constraint of fixed stall speed, Phillips et al. [42] found that the optimum lift distribution for the stress-limited design is the same as that found by Prandtl in 1933 [33], but corresponds to a 25.99% increase in wingspan and a 16.01% reduction in induced drag over the elliptic lift distribution. For the deflection-limited design, Phillips et al. [42] identified an alternate lift distribution that corresponds to a 9.07% larger wingspan and 8.03% less induced drag than those corresponding to the elliptic lift distribution.

2. Wing-Loading Constraints

As shown by Phillips [63], The wing loading W/S_W affects several aircraft performance metrics, including takeoff and landing and several additional key performance airspeeds. Fixing the wing area ensures that any optimum wing design has similar performance to the baseline wing design in these areas. In order to maintain fixed wing loading with no constraint on the wing weight, the wing area must be constrained such that as the weight changes, the wing area changes to maintain the wing loading. This is the approach taken by Phillips et al. [41,42] and Taylor and Hunsaker [43].

For the stress-limited design of a wing with fixed wing loading, Phillips et al. [41,42] found that the optimum lift distribution corresponds to a 4.98% increase in wingspan and a 4.25% reduction in induced drag over the elliptic lift distribution. For the deflection-limited design, the optimum solution allows a wingspan increase of 1.03% and a drag reduction of 0.98%. Taylor and Hunsaker [43] extended the methodology of Phillips et al. [41,42] to tapered wings, and found that depending on the taper ratio, the optimum solution for the stress-limited design may result in a wingspan increase of up to 7.63% and a drag reduction of up to 5.94%.

III. Comparison of Solutions with Integrated Bending Moment Constraints

In this section, we compare the solutions from the analytic studies of Prandtl [33], Klein and Viswanathan [39], and Jones and Lasinksi [37] to solutions from two recent high-fidelity aerostructural optimization studies by Zhang [44] and Hoogervorst and Elham [45]. Zhang [44] sought to minimize a combination of drag and weight by optimizing the wingspan and wing twist for an aircraft wing

configuration similar to that of a Boeing 737-900, subject to constraints on the maneuver stress and buckling stress. Hoogervorst and Elham [45] sought to minimize fuel weight with respect to the wingspan and wing twist at three spanwise locations for a wing based on the Airbus A320, subject to stress and fatigue constraints. A summary of the design objectives, key design variables, and key constraints for each study is included in Table A1 of the appendix.

Note that, like Prandtl [33] and Klein and Viswanathan [39], Zhang [44] did not include any constraints on the wing area. However, Hoogervorst and Elham [45] included a constraint on the maximum wing loading – the ratio of weight to wing area – that could, under certain conditions, place some lower limit on the wing area. Still, since wing loading decreases as the wing area increases, a limit on the maximum wing loading places no functional upper limit on the wing area. In fact, in the study by Hoogervorst and Elham [45], the optimal solution has a wing loading that is about 10% less than the baseline solution, suggesting that the maximum wing loading constraint is inactive. By comparison, the wing loading in the study of Klein and Viswanathan [39] reduces by about 13%, and in Prandtl's 1933 study [33], the wing loading reduces by 18%.

Although the design objectives, variables, and constraints from these two studies are much more comprehensive than those used by Prandtl [33], Klein and Viswanathan [39], and Jones and Lasinksi [37], in many respects, they are comparable. For instance, Zhang [44] sought to minimize a weighted combination of induced drag D_i and weight W of the form

$$J = \beta \frac{D_i}{D_{\text{ref}}} + (1 - \beta) \frac{W}{W_{\text{ref}}} \tag{7}$$

where D_{ref} and W_{ref} are reference drag and weight values, respectively, and β is a weighting value. However, the results considered here place considerably more emphasis on minimizing induced drag than weight. Hoogervorst and Elham [45] sought to minimize fuel weight, which is closely related to drag through the fuel burn. Instead of using the lift distribution as a design variable, both Zhang [44] and Hoogervorst and Elham [45] used the wingspan and wing twist as design variables. Nevertheless, as

evident from lifting-line theory and as shown by Phillips and Hunsaker [62], for a wing with a given planform, the lift distribution is a direct function of the wing twist distribution.

Whereas Prandtl [33], Klein and Viswanathan [39], and Jones and Lasinksi [37] imposed constraints on the integrated bending moment, Zhang [44] used constraints on the wing stress, and Hoogervorst and Elham [45] used constraints on the wing stress, buckling, and fatigue. In modern aerostructural literature, there are few, if any, studies that use constraints on the integrated bending moment and/or integrated shear force alone. However, Phillips et al. [41] have shown that the maximum allowable stress of the wing structure can be related to the bending moments by defining the proportionality coefficient S_b in Eq. (2) in terms of the properties of the wing structure. When viewed from this perspective, the wing stress is implicit in Eq. (2), and constraints on wing stress can be thought of as analogous to constraints on the integrated bending moment. Buckling and fatigue are not included in any of the studies in Refs. [33,37,39].

The optimum cruise lift distributions identified by Prandtl [33], Klein and Viswanathan [39], Jones and Lasinksi [37], Zhang [44] (with $\beta = 0.75$), and Hoogervorst and Elham [45] are shown in Fig. 1. The elliptic lift distribution is also included for reference. From Fig. 1, we see that Zhang's solution [44] matches Prandtl's solution very well [33]. The solution of Hoogervorst and Elham [45] deviates from each of the analytic solutions shown. However, it should be remembered that Hoogervorst and Elham only allowed the wing to twist at the root, the tip, and one other intermediate location [45], which results in a low-resolution approximation of the optimum lift distribution. A significantly different result may be obtained with more wing-twist design variables. Still, both high-fidelity studies appear to agree more closely with all of the theoretical aerostructural solutions shown here than with the elliptic lift distribution.

The relative agreement between Prandtl's solution [33] and Zhang's solution [44] is most remarkable because Zhang's [44] solution is for a tapered wing configuration similar to that of a Boeing 737-900, whereas Prandtl's solution is for a generic rectangular, planar wing. Since the solution for Zhang shown here primarily minimizes induced drag, the main differences between Zhang's solution and Prandtl's solution is in the wing geometry, most notably taper, and structural variables and constraints, including

structural layout and buckling constraints. Figure 1 suggests that these differences do not result in significantly different solutions for the optimum lift distribution.

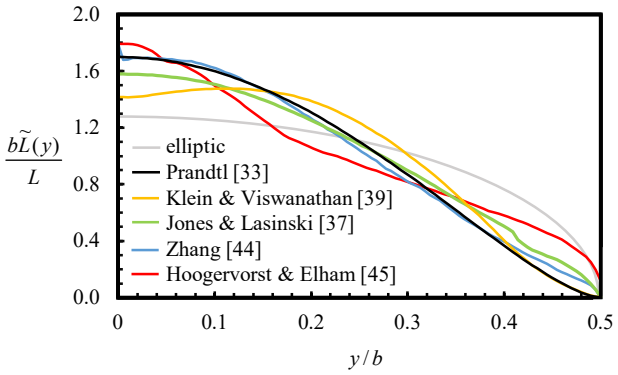


Fig. 1 Normalized optimum lift distributions from solutions with constraints related to the integrated bending moment.

Figure 2 compares the drag and wingspan from the solutions of Prandtl [33], Klein and Viswanathan [39], and Zhang [44]. For each solution, the drag and the wingspan are presented as ratios of the drag and wingspan resulting from the elliptic lift distribution on the respective study's "baseline" wing configuration. These ratios will hereafter be referred to as the "drag ratio" and the "wingspan ratio", respectively. Results from Jones and Lasinski [37] and Hoogervorst and Elham [45] were not available. Note that both Prandtl [33] and Klein and Viswanathan [39] provide solutions for the drag ratio as a function of the wingspan ratio, whereas Zhang provides results for a single, optimum configuration. The optimum solutions for Prandtl [33] and Klein and Viswanathan [39] are marked with black circles.

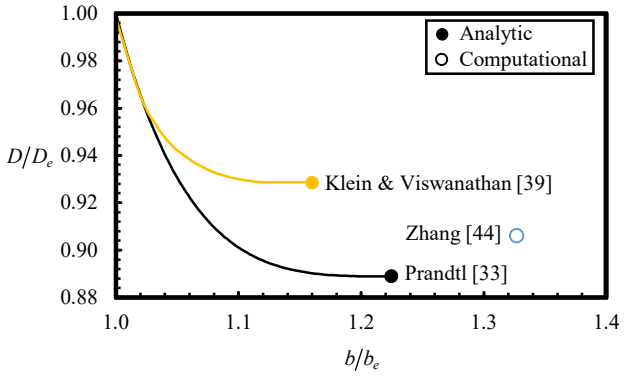


Fig. 2 Drag ratio with respect to wingspan ratio from solutions with constraints related to the integrated bending moment.

From Fig. 2, we see that whereas the drag ratio from Zhang's solution [44] falls between the drag ratios from the solutions of Prandtl [33] and Klein and Viswanathan [39], the wingspan ratio for Zhang's solution is between 10-11% higher than Prandtl's theoretical solution [33]. As is common in computational studies, the data presented do not provide definitive relational information. Hence, we are largely unable to quantitatively assess all of the reasons for the differences between theoretical and high-fidelity solutions. However, a qualitative assessment of a few likely reasons can help to build some insight into how certain constraints can affect aerostructural optimization.

For example, the 10-11% difference in wingspans between Zhang's [44] and Prandtl's solutions [33] is consistent with solutions given by Taylor and Hunsaker [43] that include the effects of wing taper on the optimum solution. In fact, whereas Prandtl [33] and Klein and Viswanathan [39] limited their solutions to rectangular wings, Zhang used a wing configuration with a taper ratio of nearly 0.16 [44]. Using the method given by Taylor and Hunsaker [43], the optimum wingspan for a tapered wing with a taper ratio of 0.16 and Prandtl's lift distribution [33] is about 11.5% higher than the optimum wingspan for a rectangular wing with the same lift distribution, which is in excellent agreement with the wingspan difference of 10-11% shown in Fig. 2.

It is also possible that Zhang's [44] solution takes advantage of passive aeroelastic load alleviation, by which maneuver loads induce aeroelastic deflections, which result in a lift distribution that alleviates bending moments at the maneuver condition. This allows the wing to be designed with a higher wingspan

than would be allowed for a wing with no passive aeroelastic load alleviation. Moreover, in the absence of constraints on the wing area, increasing the wingspan increases the aspect ratio, which tends to increase the wing flexibility and induce even more aeroelastic load alleviation. The result is a larger wingspan than that of a corresponding rigid wing. In fact, Zhang's solution [44] includes a maneuver lift distribution (not shown in Fig. 2) that features high load near the wing root and negative load near the wing tips, which results in lower bending moments at the maneuver condition than those resulting from the cruise lift distribution.

It is worth noting that the solutions of Prandtl [33] and Klein and Viswanathan [39] are both limited by the constraint that the lift distribution is fixed for all flight conditions and positive at all spanwise locations. The optimum solution for both studies lies at the limit of this second assumption, where the slope of the lift distribution at the wingtip is zero. For analytic solutions employing the methods of Prandtl [33] and Klein and Viswanathan [39], a solution having a wingspan ratio higher than the optimum shown in Fig. 2 requires negative lift at the wingtips. Under the constraints of the assumption described in Eq. (2), this would result in zero bending moment and, therefore, zero weight at some spanwise location, which is not physically valid.

IV. Comparison of Solutions with Root Bending Moment Constraints

In this section, we compare analytic solutions from Jones [34], Klein and Viswanathan [38], and DeYoung [36] to a few multi-fidelity computational studies with constraints on the root bending moment. In 2009, Verstraetan and Slingerland [50] performed a computational study to minimize drag on both planar and nonplanar wings with fixed lift and root bending moment, including viscous effects. For a planar wing, the solution of Verstraetan and Slingerland [50] allows a wingspan increase of 22% and a drag reduction of 8% over the elliptic lift distribution. Later, Ranjan [49] and Wroblewski and Ansell [51] obtained similar results using a similar computational method. However, experimental data from Wroblewski and Ansell [51] deviates slightly from the predicted optimum solution.

There are very few high-fidelity studies that include constraints on the root bending moment. However, in 2014, Lyu and Martins [52] performed one such study within in a series of high-fidelity optimization case studies aimed at minimizing the drag coefficient at cruise on a swept-wing blended-wing-body aircraft. Lyu and Martins [52] added the root-bending-moment constraint only as a limiter within an aerodynamic optimization framework. The result was a marginal increase in the wingspan and a small reduction in drag. A summary of the design objectives and key design variables and constraints for this and each of the other studies considered in this section is given in Table A2 in the appendix.

Figure 3 shows the normalized lift distributions from the solutions of Jones [34], Klein and Viswanathan [38], DeYoung [36], Verstraetan and Slingerland [50], and Lyu and Martins [52]. The optimum lift distributions predicted by Ranjan [49] and Wroblewski and Ansell [51] for minimum inviscid drag and total drag are indistinguishable from those given by Klein and Viswanathan [38] and Verstraetan and Slingerland [50], respectively, and are therefore not shown. From Fig. 3, we see that the lift distribution from the high-fidelity solution of Lyu and Martins [52] is most similar to the lift distribution given by Jones [34].

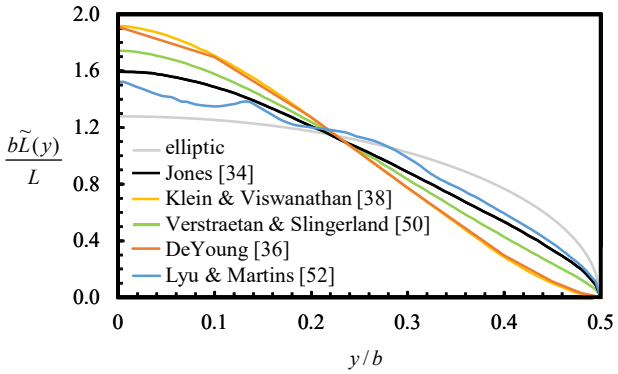


Fig. 3 Normalized optimum lift distributions from solutions with constraints related to the root bending moment.

Figure 4 shows the drag and wingspan ratios for each of the solutions discussed in this section. The results in Fig. 4 include two groups: analytic solutions that consider only induced drag [34, 36, 38], which are shown in black, and computational solutions that include viscous effects [49-51], which are shown in

gray. The experimental results from Wroblewski and Ansell [51] are for wing designs based on the optimum solutions of Ranjan [49] and Klein and Viswanathan [38]. Notice that the high-fidelity solution given by Lyu and Martins [52] falls very near unity for both the wingspan ratio and drag ratio. This is somewhat surprising, since Fig. 3 shows that the optimum lift distribution from this solution is similar to that given by Jones [34]. Nevertheless, Lyu and Martins [52] note that any additional increase in wingspan is limited by the root bending moment and by the increase in viscous drag due to additional wing surface area.

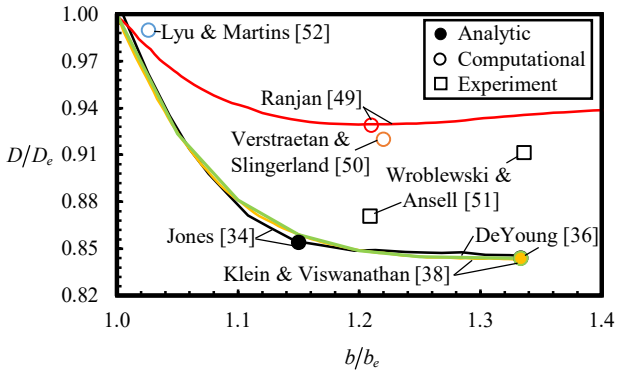


Fig. 4 Drag ratio with respect to wingspan ratio from solutions with constraints related to the root bending moment.

The difference between the results of Ranjan [49] and Verstraetan and Slingerland [50] and those of Jones [34], Klein and Viswanathan [38], and DeYoung [36] highlight the importance of viscous drag on the optimum solution. In low-speed cruise, viscous drag makes up about half of the total drag. Because of this, any reduction in induced drag shown in Fig. 4 corresponds to a much smaller reduction in total drag. In fact, if viscous drag remains relatively constant, we should expect that the induced drag reductions of around 15% reported by Jones [34], Klein and Viswanathan [38], and DeYoung [36] translate to only about 7 or 8% total drag reduction, which agrees very well with the computational results of Ranjan [49] and Verstraetan and Slingerland [50]. Moreover, as noted by Jones [34] viscous drag effectively limits the optimum wingspan, since large increases in wingspan correspond to more wetted area and higher viscous drag, which offsets the induced-drag benefits obtained by increasing the wingspan. Thus, the results of

Ranjan [49] and Verstraetan and Slingerland [50] have lower wingspans than the inviscid results of Klein and Viswanathan [38] and DeYoung [36]. When viscous effects are considered, these two viscous solutions [49,50] and Jones' "optimum" solution, which produces nearly minimum induced drag with a much smaller wingspan and wing area than the true minimum-induced-drag solution, are likely to have less total drag than those given by Klein and Viswanathan [38] and DeYoung [36].

Like the solutions of Prandtl [33] and Klein and Viswanathan [39] shown in the previous section, the solutions of Jones [34], Klein and Viswanathan [38], and DeYoung [36] are limited to all-positive lift distributions. In each case, the optimum lift distribution again lies at the limit of this assumption, where the lift distribution has zero slope at the wingtip. However, the result of Ranjan [49] shows that when viscous effects are considered, the optimum wingspan falls well below the maximum allowed under the all-positive lift-distribution constraint.

V. Comparison of Solutions having Stress and Deflection Limits with Operational Constraints

Here, we compare solutions from studies having both structural constraints and operational constraints. The following section is grouped into solutions with operational constraints related to the stall speed and operational constraints related to the wing loading.

A. Stall-Related Constraints

The lift distribution from the solution of Phillips et al. [42] that minimizes induced drag for a stress-limited wing with fixed stall speed is shown in Fig. 5, alongside the optimum lift distribution from a high-fidelity study by van den Kieboom and Elham [53] aimed at minimizing fuel burn for a Fokker 100 class regional jet aircraft wing in low-speed flight with fixed maximum takeoff weight (MTOW). In van den Kieboom's and Elham's study, the lift distribution is controlled by a small number of discrete high-lift flaps. The flap deflection, flap shape, wingspan, and wing shape are all included as design variables, and the wing is subject to constraints on wing stress and takeoff and landing distance.

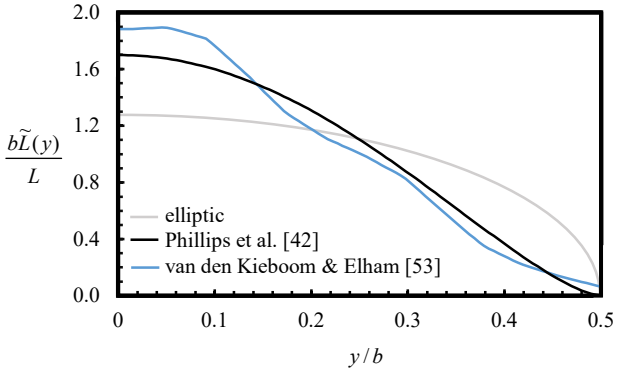


Fig. 5 Normalized optimum lift distributions from solutions with constraints related to the wing stress and the stall speed.

Although van den Kieboom and Elham [53] did not explicitly enforce constraints on the stall speed, Phillips [63] has shown that the no-wind takeoff distance $s_{\mathrm{g,TO}}$ for an aircraft with a takeoff speed of $1.1V_{\mathrm{stall}}$ can be approximated as [63]

$$s_{g,TO} \cong \frac{1.21W^2}{\rho g S_w C_{L_{\max}} (T - D - F_r)_{0.77V_{\mathrm{stall}}}} + 1.1 t_r V_{\mathrm{stall}} \tag{8}$$

where S_w is the wing area, t_r is the rotation time, and the thrust $T = T(V)$, the drag $D = D(V)$, and the rolling friction $F_r = F_r(V)$ are evaluated at 77% of the stall speed. The landing distance $s_{g,L}$ for an aircraft with a landing speed of $1.3V_{\mathrm{stall}}$ can be approximated as [63]

$$s_{g,L} = \frac{W/S_w}{\rho g (C_D - \mu_r C_L)} \ln\left[1 + \frac{1.69\rho V_{\mathrm{stall}}^2}{2W/S_w}\left(\frac{C_D}{\mu_r} - C_L\right)\right] + 1.30 t_f V_{\mathrm{stall}} \tag{9}$$

where t_f is the brake-engagement reaction time, μ_r is the coefficient of rolling friction, and C_L and C_D are constant. Equations (8) and (9) show that for a wing with fixed MTOW, S_w, and $\widetilde{C}_{L_{\max}}$ on a surface with known μ_r, constraints on takeoff and landing distance are a function of the stall speed. Thus, van den Kieboom and Elham's [53] constraints on the takeoff and landing distance are closely related to the stall speed. A summary of the key design variables and constraints for the studies of Phillips et al. [42] and van den Kieboom and Elham [53] are given in Table A3 in the appendix.

Figure 5 shows that the lift distributions of Phillips et al. [42] and van den Kieboom and Elham [53] are in general agreement but exhibit some differences. As was the case with Hoogervorst and Elham [45], these differences are likely due to the relatively low number of flaps used by van den Kieboom and Elham [53] to

control the lift distribution, which results in a low-resolution approximation of the optimum lift distribution. Figure 6 shows that the wingspan ratio and corresponding drag ratio from van den Kieboom and Elham [53] agree relatively well with the trend predicted by Phillips et al. [42], but the drag ratio is higher, and the wingspan ratio is lower, than the optimum solution from Phillips et al. [42]. While there are likely many reasons for this, it should be noted that van den Kieboom and Elham [53] included both buckling and fatigue constraints in their solution, which in some cases may limit the allowable wingspan. Although van den Kieboom and Elham [53] do not report if these constraints are active in their study, for many wing structures, buckling and fatigue are the critical failure modes that drive the structural sizing.

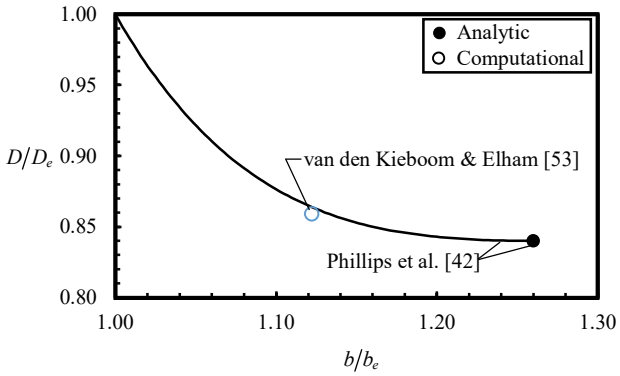


Fig. 6 Drag ratio with respect to wingspan ratio from solutions with constraints related to the wing stress and the stall speed.

The optimum lift distribution found by Phillips et al. [42] for the deflection-limited design of a wing with fixed stall speed is shown in Fig. 7, alongside four additional lift distributions from high-fidelity solutions found by Jansen et al. [46] for a flexible tapered wing with wingtip devices and three lift distributions from a high-fidelity study by Mader et al. [59] for the flexible D8 wing in transonic flight with cruise Mach numbers of 0.72, 0.78, and 0.82. In their study, Jansen et al. [46] sought to minimize induced and total drag on wings with winglets and raked wingtips using design variables including the wingspan, jig twist, sweep angle, and dihedral distribution, with constraints on lift, maneuver stress, and wing stall. The results labeled *a, b, c,* and *d* in Fig. 7 correspond to solutions for minimum total drag with raked wingtips (*a*), minimum total drag with winglets (*b*), minimum induced drag with winglets (*c*), and

minimum induced drag with raked wingtips (d). Mader et al. [59] sought to minimize fuel burn using wingspan, twist, airfoil shape, sweep angle, and other design variables, subject to constraints on the wing stress, pitching moment, and flow separation.

Although static wing deflection constraints, like those used by Phillips et al. [41,42], are seldom, if ever, enforced explicitly in practice, wing deflection can have significant aerodynamic effects, especially for highly flexible wings. While flexible wings often benefit from some passive aeroelastic maneuver load alleviation, excessive wing deflection can negatively impact cruise performance. Conceptually, there is some limit on flexibility at which negative effects during cruise outweigh passive load alleviation during a maneuver. This limit can be thought of as a "soft" deflection limit. Because wings with high aspect ratios often have greater flexibility, this "soft" limit on wing deflection can also serve as a limit on the aspect ratio. Thus, although Phillips et al. [41,42] did not account for the aerodynamic effects of static wing deflections, their "hard" deflection limit acts as a surrogate for the natural aerostructural efficiency limit associated with high-aspect-ratio designs. Nevertheless, it should be remembered that the deflection limit imposed by Phillips et al. [41,42] is an imperfect approximation of natural aeroelastic limits. Therefore, a comparison between deflection-limited solutions from Phillips et al. [41,42] and high-fidelity studies with flexible wings that include possible "soft" limits on wing deflection is also imperfect.

It is important to note that in place of a constraint on the stall speed, Jansen et al. [46] placed a constraint on the maximum section lift coefficient $\widetilde{C}_{L_{\max}}$, and Mader et al. [59] included a constraint on flow separation to preclude stall due to buffet. Although these constraints differ from the fixed-stall-speed constraint used by Phillips et al. [42], the stall speed and $\widetilde{C}_{L_{\max}}$ are related through Eq. (6), and since stall is a result of flow separation, the stall speed and $\widetilde{C}_{L_{\max}}$ can be thought of as surrogate indicators of flow separation. Thus, the constraints on flow separation are also closely related to the stall speed and $\widetilde{C}_{L_{\max}}$.

Figure 7 shows that the optimum lift distribution of Phillips et al. [42] falls well within the range of solutions given by Jansen et al. [46] and shows good agreement with the results given by Mader et al. [59], with the closest agreement at $M = 0.82$. The reason for this may be that at $M = 0.82$, the flow-separation constraint is most active. In fact, the results from Mader et al. [59] show little to no flow separation at

$M = 0.72$ and $M = 0.78$, but indicate small regions of flow separation at $M = 0.82$, which suggests that at this Mach number, the optimum design may be approaching the constraining flow-separation limit.

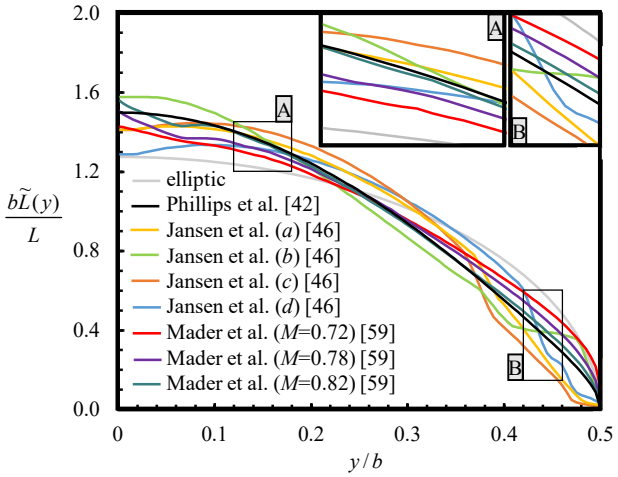


Fig. 7 Normalized optimum lift distributions from solutions with constraints related to the wing deflection and the stall speed.

The drag ratio and wingspan ratio for the solutions of Phillips et al. [42] and Jansen et al. [46] are shown in Fig. 8. Results from Mader et al. [59] were not available. Although the wingspan ratios given by Jansen et al. [46] are generally higher than that given by Phillips et al. [42], we see that the solutions of Jansen et al. [46] follow the general trend of the solution of Phillips et al. [42] reasonably well. Note that because Jansen et al. [46] did not include the vertical portion of the winglet in the wingspan measurement, the solutions for wings with raked wingtips (*a* and *d*) have significantly higher wingspan ratios than the solutions for wings with winglets (*b* and *c*).

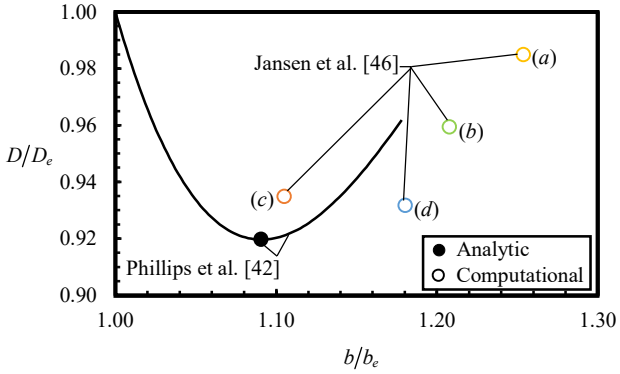


Fig. 8 Drag ratio with respect to wingspan ratio from solutions with constraints related to the wing deflection and the stall speed.

B. Wing-Loading Constraints

Here we compare the solutions of Phillips et al. [41,42] and Taylor and Hunsaker [43] for planar wings with fixed wing loading to several high-fidelity studies with similar constraints. In the studies of Stewart and Hunsaker [60], Löbert [40], McGeer [54], Piperni et al. [55], and Liem et al. [56], the wing loading is fixed through a combination of constraints on the wing area and the weight. However, we will also consider results from studies by Kenway et al. [58] and Ning and Kroo [57], in which the wing area is fixed with no constraint on the weight. In general, this allows for changes in the wing loading. However, in Ref. [58], the wing loading changes by only 1.7%. In Ref. [57], the weight is not given, but we assume that changes in wing loading are similarly small.

1. Stress-Limited Design

Key results from several studies with constraints related to the wing stress and wing loading are shown in Figs. 9 and 10. Figure 9 shows the optimum cruise lift distribution from each study, and Fig. 10 shows the optimum wingspan and drag ratios. Results shown in Fig. 9 include the theoretical studies of Phillips et al. [41,42], Taylor and Hunsaker [43] (with $R_T = 0$), and Löbert [40]; the multi-fidelity results of Stewart and Hunsaker [60] (with geometric and aerodynamic twist) and McGeer [54]; and the high-fidelity studies of Piperni et al. [55], Ning and Kroo [57], and Liem et al. [56]. A summary of the optimization objectives, key design variables, and key design constraints for each of these studies is given in Table A5 in the

appendix. Note that because Piperni et al. [55] and Ning and Kroo [57] do not give data for the drag and wingspan ratios, their solutions are not included in Fig. 10.

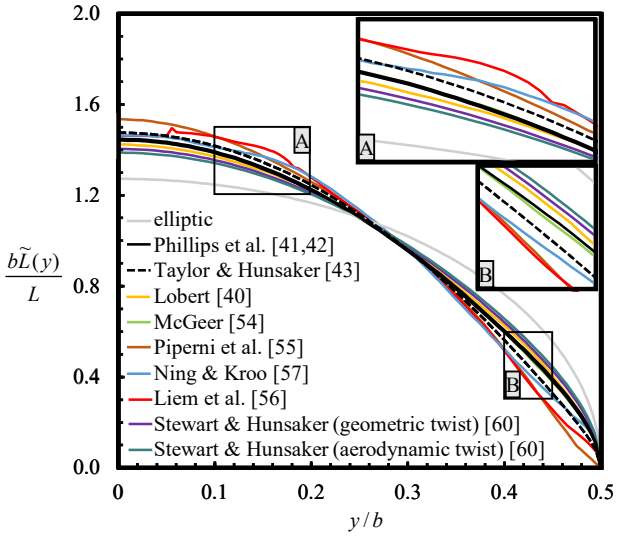


Fig. 9 Normalized optimum lift distributions from solutions with constraints related to the wing stress and the wing loading.

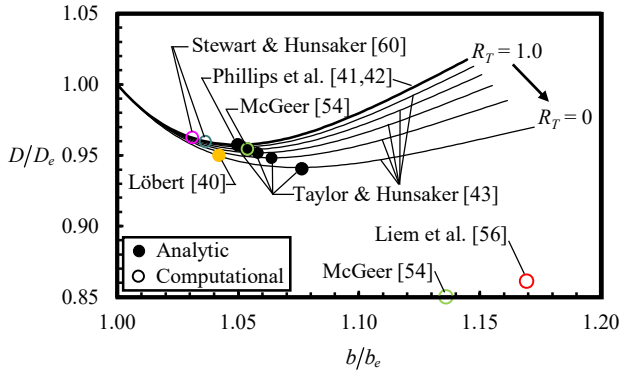


Fig. 10 Drag ratio with respect to wingspan ratio from solutions with constraints related to the wing stress and the wing loading.

The most striking observation from Fig. 9 is the high level of agreement between all of the cruise lift distributions shown, despite significant differences in the design objectives, variables, and assumptions used in each study. Phillips et. al [41,42] used only the wingspan and lift distribution as design variables to minimize induced drag on a planar, unswept rectangular wing. Taylor and Hunsaker [43] extended the

work of Phillips et al. [41,42] to tapered wings. Stewart and Hunsaker [60] considered the effects of parasitic drag on the results given by Phillips et al. [41,42] when the lift distribution is achieved using either geometric or aerodynamic twist alone. The approaches of Löbert [40] and McGeer [54] are similar to those of Phillips et al. [41,42] and Taylor and Hunsaker [43], but both considered swept wings, and McGeer [54] allowed the airfoil thickness to change, while imposing constraints on the parasitic drag. In the high-fidelity study by Ning and Kroo [57], the cruise and maneuver twist distributions are treated as separate design variables and are optimized to minimize total drag on a trapezoidal wing typical of a commercial transport. Piperni et al. [55] sought to minimize the cash operating cost on the wing of a large transonic business jet, including the effects of wing flexibility. Liem et al. [56] sought to minimize the fuel burn on the Common Research Model wing including aeroelastic and transonic effects. The agreement of the results in Fig. 9 suggests that the optimum cruise lift distribution for a wing with stress and wing-loading constraints is relatively consistent over a wide range of aircraft configurations and flight conditions and is well approximated by the theoretical solutions of Phillips et al. [41,42], Taylor and Hunsaker [43], and Löbert [40].

Figure 10 shows the optimum drag ratios and wingspan ratios for solutions given by Phillips et al. [41,42], Taylor and Hunsaker [43], Stewart and Hunsaker [60], Löbert [40], McGeer [54] and Liem et al. [56]. Note that the wingspan ratios from the solutions of Stewart and Hunsaker [60] are slightly smaller, and the drag ratios are slightly higher, than the optimum solutions given by Phillips et al. [41,42], Taylor and Hunsaker [43], and Löbert [40]. Although Stewart and Hunsaker [60] used very nearly the same constraints as Phillips et al. [41,42] and Taylor and Hunsaker [43], the solutions of Stewart and Hunsaker [60] are for minimum total drag, whereas the results of Phillips et al. [41,42], Taylor and Hunsaker [43], and Löbert [40] are for minimum induced drag. Since in each of these cases, the wing-structure weight is constant, the wing area must also be constant, and changes in parasitic drag are primarily a result of changes in geometric or aerodynamic twist. Figure 9 shows that the lift distributions from Phillips et al. [41,42] and Taylor and Hunsaker [43], which minimize induced drag, feature higher lift at the wing root and lower lift at outboard portions of the wing than the elliptic lift distribution. However,

on a rectangular wing, these lift distributions require more twist, and therefore, produce more parasitic drag than lift distributions that are more nearly elliptic. Therefore, when minimizing total drag on subsonic wings with fixed wing area, there is a tradeoff between induced and parasitic drag resulting from the relationship between the lift distribution, wing twist, and drag. The result of this tradeoff is an optimum lift distribution that is slightly more elliptic, and requires less twist, than the optimum lift distribution for minimum induced drag, as shown in Fig. 9. If the wing-structure weight is fixed, then the corresponding wingspan ratio is slightly lower, and the drag ratio is slightly higher, than the minimum-induced drag solution. This is reflected in the differences between the solutions of Stewart and Hunsaker [60] and those of Phillips et al. [41,42], Taylor and Hunsaker [43], and Löbert [40] in Fig. 10.

It is also important to note that two solutions from McGeer [54] are included in Fig. 10. Both are solutions for the design of a light, low-speed wing. The only difference between these two solutions is that the solution at the bottom of Fig. 10 includes the airfoil thickness as a design variable, while the solution near those of Phillips et al. [41,42], Taylor and Hunsaker [43], and Löbert [40] only includes the wingspan and lift distribution as design variables. Within the constraints of McGeer's study [54], the allowable height of the wing structure inside the airfoil section increases as the airfoil thickness increases, which reduces the amount of structure needed to support a given distribution of wing bending moments. Since McGeer's solution is for a low-speed wing, the airfoil thickness is not constrained by transonic effects, which tend to favor thin airfoils that reduce transonic shock. Thus, when the thickness is included as a design variable, as is the case with McGeer's solution [54], we expect the solution to favor a thick airfoil that allows for a more efficient wing-structure design and results in a higher wingspan and lower drag than solutions with prescribed thickness, such as those in Refs. [40,41,42,54].

In the case of Liem et al. [56], the relatively high wingspan ratio and low drag ratio shown in Fig. 10 are likely due to several effects, including passive aeroelastic load alleviation, as described in Section III, wave-drag reduction, and the use of composite structures. Whereas all other solutions in Fig. 10 are for low-speed flight, the solution given by Liem et al. [56] is for flight in the transonic regime, where wave drag constitutes a significant portion of total drag. Wave drag can be reduced by changing wing sweep and

by tailoring the airfoil cross sections to delay shock. In fact, in their study, Liem et al. [56] include the sweep angle and the airfoil shapes as design variables and show that the wave drag is the largest contributor to the drag reduction achieved by their optimum solution. Additionally, Liem et al. [56] used composite materials for the wing structure, which have been shown to allow larger wingspans, and result in greater drag reductions, than isotropic materials such as aluminum [61], which are generally used for structural modeling in theoretical and analytic studies.

2. Deflection-Limited Design

Phillips et al. [41,42] also presented a solution for the deflection-limited design of a wing with fixed wing loading. Here, we compare this solution to the minimum-drag solution presented by McGeer [54] for a light, high-speed elastic wing with fixed wing-structure weight and fixed wing area and the minimum-fuel-burn solution presented by Kenway et al. [58] for the flexible undeflected Common Research Model wing in transonic flight. A summary of the optimization setup for each of these studies is given in Table A6 in the appendix. Recall that the deflection limit imposed by Phillips et al. [41,42] can be thought of as an approximate surrogate for the natural aeroelastic deflection limits encountered by flexible wings, as described in Section V.A. The optimum lift distribution from each solution is shown in Fig. 11, and the corresponding drag ratios and wingspan ratios are shown in Fig. 12.

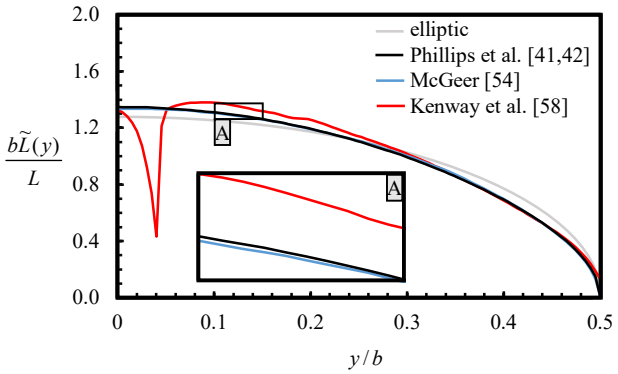


Fig. 11 Normalized optimum lift distributions from solutions with constraints related to the wing deflection and the wing loading.

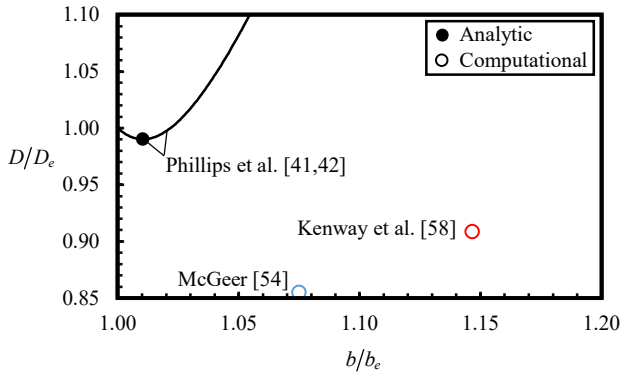


Fig. 12 Drag ratio with respect to wingspan ratio for solutions with constraints related to the wing deflection and the wing loading.

From Fig. 11, we see that, as was the case for solutions with stress and wing-loading constraints, the optimum lift distribution from solutions with deflection and wing-loading constraints show remarkable consistency, especially considering the range of configurations and flight conditions represented by these three studies. However, as expected, the wingspan and drag ratios of McGeer [54] and Kenway et al. [58] shown in Fig. 12 are significantly different from those given by Phillips et al. [41,42]. Again, in the case of McGeer [54], this is likely due to thickness effects, and in the case of Kenway et al. [58], this may be influenced by passive aeroelastic load alleviation and wave drag reduction achieved by changing the sweep angle and airfoil shapes.

VI. Conclusions

As aerostructural research trends more toward computational methods, theoretical aerostructural solutions are often neglected and are sometimes forgotten. However, as evidenced by the aerodynamically-optimum elliptic lift distribution, solutions based on theory can sometimes have value well beyond the assumptions of the original theory. Because of this, in this paper, we have sought to address two foundational questions regarding theoretical solutions: 1) How well do theoretical aerostructural solutions apply to practical aircraft configurations? and 2) Can these solutions be used as appropriate reference solutions for higher fidelity methods? Analysis of these two questions was made by comparing results from theoretical aerostructural solutions to several multi- and high-fidelity aerostructural solutions. It has been

shown that, depending on the design constraints, certain theoretical solutions agree reasonably well with results from high-fidelity studies for a wide variety of wing configurations and can serve as appropriate reference solutions for higher fidelity results. Each of the theoretical solutions shown in this paper captures important aerostructural trends that are useful for gaining insights into the aerodynamic and structural coupling involved in aerostructural design and optimization, and each solution could be used as a simple validation case to help address the challenge of achieving proper coupling between computational aerodynamic and structural analysis components within a high-fidelity aerostructural optimization code.

For appropriate comparison, the results in this paper are divided into categories based on the design constraints. Section III compares solutions from studies with constraints related to the integrated bending moment, with no other constraint on wing area. Section IV compares solutions from studies with constraints on the root bending moment. Section V compares solutions combining constraints related to wing stress or wing deflection with constraints on either wing stall or the wing loading. The results are summarized in Figs. 1-12.

The optimum lift distributions for each of these categories are shown in Figs. 1, 3, 5, 7, 9, and 11. The best agreement in lift distributions are those for the stress- and deflection-limited design of a wing with fixed wing loading, as shown in Figs. 9 and 11. Because of variations in the design objectives, variables and assumptions, a true consistent comparison of drag and wingspan values often cannot be obtained. Still, a qualitative comparison of these results highlights the importance of understanding and accounting for the limiting approximations of theoretical solutions when using them as a reference for conceptual design or interpreting results from higher-fidelity solvers.

It should be remembered that the studies considered in this paper do not represent an exhaustive review of the aerostructural literature. Instead, this paper only focuses on the specific subset of theoretical aerostructural studies concerned with minimizing drag with respect to the wingspan and the lift distribution. The multi- and high-fidelity studies shown here were selected based on their design constraints to provide as appropriate a comparison as possible to the theoretical results. It should also be remembered that, in the absence of definitive relational information from computational studies, the results and discussion in this

paper do not provide a quantitative analysis of all of the physical mechanisms that contribute to the solutions shown here. Certain key considerations, including the effects of viscosity, aeroelasticity, composite materials, buckling, and fatigue, are only qualitatively assessed. Nevertheless, the comparisons made here do provide important insights into the aerodynamic and structural coupling involved in aerostructural wing design and optimization and the ways in which theoretical aerostructural solutions can be used to inform and validate higher fidelity aerostructural research.

Appendix

The following tables give a summary of the key design objectives, design variables, and design constraints for each of the aerostructural studies discussed in this paper. Table A1 includes studies with constraints related to the wing integrated bending moment. Table A2 shows studies with constraints related to the wing root bending moment. Tables A3 and A4 are for studies combining constraints on the wing stress and deflection with constraints related to wing stall. Tables A5 and A6 are for studies combining constraints on the wing stress and deflection with constraints related to the wing loading. For each table, the primary constraints or assumptions that relate to these categories are typeset in bold, along with key design variables related to the wingspan and lift distribution. It should be remembered that these tables are intended for high-level reference and comparison only. In many cases, the design variables and constraints shown here do not represent an exhaustive list of all design variables and constraints considered in the respective study.

Table A1 Optimization summary and key constraints for studies including constraints related to the integrated bending moment.

	study type	*configuration*	*objective*	*key design variables*	*key constraints*
Prandtl [33]	analytic	planar, unswept rectangular wing	minimum induced drag	**wingspan** **lift distribution**	fixed gross weight **fixed moment of inertia of weight** fixed chord fixed *t*/*c*
Klein & Viswanathan [39]	analytic	planar, unswept wing	minimum induced drag	**wingspan** **lift distribution**	fixed wing-structure weight fixed lift **max integrated bending moment** max integrated shear force
Jones & Lasinski [37]	analytic	unswept wing with winglets	minimum induced drag	**wingspan** **lift distribution**	**max integrated bending moment** fixed weight
Zhang [44]	high fidelity	Boeing 737 (similar)	minimum linear combination, induced drag & weight	**wingspan** **twist distribution** airfoil shape angle of attack structure thickness	**max maneuver stress** fixed net weight fixed Mach number fixed altitude
Hoogervorst & Elham [45]	high fidelity	Airbus A320 (similar)	minimum fuel weight	**wingspan** **break, tip twist** airfoil shape angle of attack root chord taper ratio sweep angle structure thicknesses takeoff weight	steady level lift coefficient **max stress, 2.5 g pull up** **max stress, -1.0 g push over** max fatigue stress, 1.3 g gust **max stress, 1.0 g roll** min aileron effectiveness fixed maximum wing loading

Table A2 Optimization summary and key constraints for studies including constraints related to the root bending moment.

	study type	*configuration*	*objective*	*key design variables*	*key constraints*
Jones [34]	analytic	planar, unswept wing	minimum induced drag	**wingspan** **lift distribution**	fixed lift **fixed root bending moment**
Klein & Viswanathan [38]	analytic	planar, unswept wing	minimize induced drag	**wingspan** **lift distribution**	fixed lift **fixed root bending moment**
DeYoung [36]	analytic	planar, unswept wing	minimum induced drag	**wingspan** **lift distribution**	fixed lift **fixed bending moment, given location**
Verstraetan & Slingerland [50]	multi fidelity	unswept wing with winglets	minimum drag	**wingspan** **lift distribution** winglet height winglet length	fixed lift **fixed root bending moment** fixed wing area
Ranjan [49]	multi fidelity	planar, unswept wing	minimum drag	**wingspan** **twist distribution** chord	fixed lift **fixed root bending moment, maneuver** fixed wing area fixed Reynolds number fixed airfoil shape fixed taper ratio
Wroblewski & Ansell [51]	experimental	planar, unswept wing	minimum drag	**wingspan** **twist distribution**	fixed lift **fixed root bending moment, maneuver** fixed wing area fixed Reynolds number fixed airfoil shape fixed taper ratio
Lyu & Martins [52]	high fidelity	Blended Wing-Body	minimum drag	**wingspan** **twist distribution** airfoil shape angle of attack chord sweep angle structure thickness	steady level lift coefficient **max root bending moment** fixed taper ratio min internal volume fixed static margin fixed center of gravity trim

Table A3 Optimization summary and key constraints for studies including constraints related to the wing stress and stall.

	study type	*configuration*	*objective*	*key design variables*	*key constraints*
Phillips et al. [42]	analytic	planar, unswept rectangular wing	minimum induced drag	**wingspan** **lift distribution** wing weight	**max maneuver/hard-landing stress** **fixed stall speed** **fixed max lift coefficient**
van den Kieboom & Elham [53]	high fidelity	Fokker 100-class (similar)	minimum fuel weight	**wingspan** **flap deflection** airfoil shape chord distribution flap planform structure thickness	steady level lift coefficient **max stress, 2.5 g pull up** **max stress, -1.0 g push over** max fatigue stress, 1.3 g gust max buckling stress max stress, 1.0 g roll min aileron effectiveness **max takeoff distance** **min takeoff distance** fixed max takeoff weight

Table A4 Optimization summary and key constraints for studies including constraints related to the wing deflection and stall.

	study type	*configuration*	*objective*	*key design variables*	*key constraints*
Phillips et al. [42]	analytic	planar, unswept rectangular wing	minimum induced drag	**wingspan** **lift distribution** wing weight	**max maneuver/hard-landing deflection** **fixed stall speed** **fixed max lift coefficient**
Jansen et al. [46]	high fidelity	tapered, **elastic** wing with wingtip devices	maximum range	**wingspan** **jig twist** angle of attack root chord sweep angle dihedral distribution taper ratio structure thickness	steady level lift **max section lift coefficient (stall)** max stress, 2.5g maneuver
Mader et al. [59]	high fidelity	MIT D8 "double bubble" (**elastic**)	minimum fuel burn	**wingspan** **twist distribution** airfoil shapes angle of attack chord distribution tail rotation cruise altitude structure thickness structure design	steady level lift zero pitching moment (trim) max wingspan (gate constraint) **separation constraint (buffet)** min wing thickness min wing volume max yield stress, 2.5g/1g gust max buckling stress, 2.5g/-1g/1g gust structural thickness adjacency

Table A5 Optimization summary and key constraints for studies including constraints related to the wing stress and the wing loading.

	study type	*configuration*	*objective*	*key design variables*	*key constraints*
Phillips et al. [41,42]	analytic	planar, unswept rectangular wing	minimum induced drag	**wingspan** **lift distribution** wing weight [42]	**max maneuver/hard-landing stress** **fixed wing loading** fixed gross weight [41]
Taylor & Hunsaker [43]	analytic	planar, unswept tapered wing	minimum induced drag	**wingspan** **lift distribution** wing weight	**max maneuver/hard-landing stress** **fixed wing loading**
Löbert [40]	analytic	planar, unswept wing	minimum induced drag	**wingspan** **lift distribution**	**fixed wing area** fixed airfoil thickness **fixed gross weight** **max integrated moment/thickness ratio**
Stewart & Hunsaker [60]	multi fidelity	planar, unswept rectangular wing	minimum drag	**wingspan** **twist distribution**	**max maneuver/hard-landing stress** max maneuver/hard-landing deflection **fixed wing area** **fixed wing-structure weight**
McGeer [54]	multi fidelity	planar, unswept wing (light, low-speed)	minimum drag	**wingspan** **lift distribution** airfoil thickness	max section lift coefficient (stall) fixed parasitic drag coefficient **fixed wing-structure weight** **fixed wing area**
Ning & Kroo [57]	high fidelity	swept, planar trapezoidal wing	minimum drag	**wingspan** **cruise twist** **maneuver twist** chord distribution	min cruise/maneuver lift max section lift coefficient (inactive) **fixed wetted area** **max maneuver stress**
Piperni et al. [55]	high fidelity	large business jet	minimum cash operating cost	**aspect ratio** **lift distribution** airfoil shapes break chords sweep angle structure thickness	relative inboard/outboard sweep max/min strain fixed flight condition **max critical maneuver load** **fixed wing area** **fixed max takeoff weight**
Liem et al. [56]	high fidelity	Common Research Model	minimum fuel burn	**wingspan** **twist distribution** airfoil shapes angle of attack chord distribution sweep angle tail rotation	**min wing area** min wing-box volume fixed mean aerodynamic chord fixed center of gravity **fixed cruise/maneuver lift** **max maneuver stress** max gust stress

Table A6 Optimization summary and key constraints for studies including constraints related to the wing deflection and the wing loading.

	study type	*configuration*	*objective*	*key design variables*	*key constraints*
Phillips et al. [41,42]	analytic	planar, unswept rectangular wing	minimum induced drag	**wingspan**	**max maneuver/hard-landing deflection**
				lift distribution	**fixed wing loading**
				wing weight [42]	fixed gross weight [41]
McGeer [54]	multi fidelity	swept, **elastic** wing (light, high-speed)	minimum drag	**wingspan**	max integrated moment/thickness ratio
				lift distribution	max section lift coefficient (stall)
				airfoil thickness	fixed crest-critical Mach number
					fixed wing area
					fixed wing-structure weight
Kenway et al. [58]	high fidelity	undeflected Common Research Model (**elastic**)	minimum fuel burn	**wingspan**	steady level lift coefficient
				twist distribution	max buffet lift coefficient
				airfoil shapes	structural thickness adjacency
				angle of attack	**fixed wing area**
				chord length	min fuel volume
				sweep angle	max yield stress, 2.5 g maneuver
				altitude	max yield stress, -1.0 g push over
				structure dimensions	max buckling stress
				structure location	

Acknowledgements

This material is partially based upon work supported by NASA under Grant No. 80NSSC18K1696 issued by the Aeronautics Research Mission Directorate through the 2018 NASA Fellowship Activity with Nhan Nguyen as the NASA Technical Advisor.

References

[1] 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)

[2] 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)

[3] Gallay, S., and Laurendeau, E., "Preliminary-Design Aerodynamic Model for Complex Configurations Using Lifting-Line Coupling Algorithm," *Journal of Aircraft*, Vol. 53, No. 4, 2016, pp. 1145–1159. (doi:10.2514/1.C033460)

[4] Phillips, W. F., and Hunsaker, D. F., "Lifting-Line Predictions for Induced Drag and Lift in Ground Effect," *Journal of Aircraft*, Vol. 50, No. 4, 2013, pp. 1226–1233. (doi:10.2514/1.C032152)

[5] Wickenheiser, A., and Garcia, E., "Aerodynamic Modeling of Morphing Wings Using an Extended Lifting-Line Analysis," *Journal of Aircraft*, Vol. 44, No. 1, 2007, pp. 10–16. (doi:10.2514/1.18323)

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

[7] Rasmussen, M. L., and Smith, D. E., "Lifting-Line Theory for Arbitrarily Shaped Wings," *Journal of Aircraft*, Vol. 36, No. 2, 1999, pp. 340–348. (doi:10.2514/2.2463)

[8] Bera, R. K., "Some remarks on the solution of the lifting line equation," *Journal of Aircraft*, Vol. 11, No. 10, 1974, pp. 647–648. (doi:10.2514/3.44397)

[9] Prandtl, L., "Tragflügel Theorie," *Nachricten von der Gesellschaft der Wissenschaften zu Göttingen*, Geschäeftliche Mitteilungen, Klasse, 1918, pp. 451–477.

[10] Prandtl, L., "Applications of Modern Hydrodynamics to Aeronautics," NACA TR-116, June 1921.

[11] Munk, M. M., "The Minimum Induced Drag of Aerofoils," NACA TR-12, 1923.

[12] Lundry, J. L., "Minimum Swept-Wing Induced Drag with Constraints on Lift and Pitching Moment," *Journal of Aircraft*, Vol. 4, 1967, pp. 73–74. (doi:10.2514/3.43797)

[13] Lissaman, P. B. S., and Lundry, J. L., "A Numerical Solution for the Minimum Induced Drag of Nonplanar Wings," *Journal of Aircraft*, Vol. 5, 1968, pp. 17–21. (doi:10.2514/3.43901)

[14] Ashenberg, J., and Weihsradius, D., "Minimum Induced Drag of Wings with Curved Planform," *Journal of Aircraft*, Vol. 21, 1984, pp. 89–91. (doi:10.2514/3.56733)

[15] Rokhsaz, K., "Effect of Viscous Drag on Optimum Spanwise Lift Distribution," *Journal of Aircraft*, Vol. 30, 1993, pp. 152–154. (doi:10.2514/3.46328)

[16] Demasi, L., "Induced Drag Minimization: A Variational Approach Using the Acceleration Potential," *Journal of Aircraft*, Vol. 43, 2006, pp. 669–680. (doi:10.2514/1.15982)

[17] Demasi, L., "Erratum on Induced Drag Minimization: A Variational Approach Using the Acceleration Potential," *Journal of Aircraft*, Vol. 43, 2006, p. 1247. (doi:10.2514/1.26648)

[18] Demasi, L., "Investigation on the Conditions of Minimum Induced Drag of Closed Wing Systems and C-Wings," *Journal of Aircraft*, Vol. 44, 2007, pp. 81–99. (doi:10.2514/1.21884)

[19] Demasi, L., Dipace, A., Monegato, G., and Cavallaro, R., "Invariant Formulation for the Minimum Induced Drag Conditions of Nonplanar Wing Systems," *AIAA Journal*, Vol. 52, 2014, pp. 2223–2240. (doi:10.2514/1.J052837)

[20] Demasi, L., Monegato, G., and Cavallaro, R., "Minimum Induced Drag Theorems for Multiwing Systems," *AIAA Journal*, Vol. 55, 2017, pp. 3266–3287. (doi:10.2514/1.J055652)

[21] Gray, W. L., and Schenk, K. M., "A Method for Calculating the Subsonic Steady-State Loading on an Airplane with a Wing of Arbitrary Planform and Stiffness," NACA TR-3030, December, 1953.

[22] Burdette, D. A., and Martins, J. R. R. A., "Impact of Morphing Trailing Edges on Mission Performance for the Common Research Model," *Journal of Aircraft*, Vol. 56, No. 1, 2019, pp. 369-384. (doi:10.2514/1.C034967)

[23] Kenway, G. K. W., and Martins, J. R. R. A., "Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration," *Journal of Aircraft*, Vol. 51, No. 1, 2014, pp. 144-160. (doi:10.2514/1.C032150)

[24] James, K. A., Kennedy, G. J., and Martins, J. R. R. A., "Concurrent aerostructural topology optimization of a wing box," *Computers and Structures*, Vol. 134, 2014, pp. 1-17. (doi:10.1016/j.compstruc.2013.12.007)

[25] Mader, C. A., and Martins, J. R. R. A, "Stability-Constrained Aerodynamic Shape Optimization of Flying Wings," *Journal of Aircraft*, Vol. 50, No. 5, 2013, pp. 1431-1449. (doi:10.2514/1.C031956)

[26] Craig, A. P., and McLean, D. J., "Spanload Optimization for Strength Designed Lifting Surfaces," AIAA 88-2512, 6th Applied Aerodynamics Conference, Williamsburg, Virginia, 5–8 June 1988.

[27] Haftka, R. T., "Optimization of Flexible Wing Structures Subject to Strength and Induced Drag Constraints," *AIAA Journal*, Vol. 14, No. 8, 1977, pp. 1101–1106. (doi: 10.2514/3.7400)

[28] Grossman, B., Gurdal, Z., Strauch, G. J., Eppard, W. M., and Haftka, R. T., "Integrated Aerodynamic/Structural Design of a Sailplane Wing," *Journal of Aircraft*, Vol. 25, No. 9, 1988, pp. 855–860. (doi: 10.2514/3.45670)

[29] Wakayama, S., and Kroo, I. M., "Subsonic Wing Planform Design Using Multidisciplinary Optimization," *Journal of Aircraft*, Vol. 32, No. 4, 1995, pp. 746–753. (doi: 10.2514/3.46786)

[30] Calderon, D. E., Cooper, J. E., Lowenberg, M., and Neild, S. A., "On the Effect of Including Geometric Nonlinearity in the Sizing of a Wing," AIAA 2018-1680, 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Kissimmee, Florida, 8–12 January, 2018.

[31] Iglesias, S., and Mason, W. H., "Optimum Spanloads Incorporating Wing Structural Weight," AIAA 2001-5234, 1st Aircraft, Technology Integration, and Operations Forum, Los Angeles, California, 16–18 October 2001.

[32] Gopalarathnam, A., and Norris, R. K., "Ideal Lift Distributions and Flap Angles for Adaptive Wings," *Journal of Aircraft*, Vol. 46, No. 2, 2009, pp. 562–571. (doi: 10.2514/1.38713)

[33] Prandtl, L., "Über Tragflügel kleinsten induzierten Widerstandes," Zeitschrift für Flugtechnik und Motorluftschiffahrt, Vol. 24, No. 11, 1933, pp. 305–306.

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

[35] Pate, D. J., and German, B. J., "Lift Distributions for Minimum Induced Drag with Generalized Bending Moment Constraints," *Journal of Aircraft*, Vol. 50, 2013, pp. 936–946. (doi:10.2514/1.C032074)

[36] DeYoung, J., "Minimization Theory of Induced Drag Subject to Constraint Conditions," NASA CR-3140, June 1979.

[37] Jones, R. T., and Lasinski, T. A., "Effect of Winglets on the Induced Drag of Ideal Wing Shapes," NASA TM-81230, Sept. 1980.

[38] Klein, A., and Viswanathan, S. P., "Minimum Induced Drag of Wings with Given Lift and Root-Bending Moment," *Zeitschrift fur Angewandte Mathematik und Physik*, Vol. 24, 1973, pp. 886–892.

[39] Klein, A., and Viswanathan, S. P., "Approximate Solution for Minimum Induced Drag of Wings with Given Structural Weight," *Journal of Aircraft*, Vol. 12, No. 2, 1975, pp. 124–126. (doi:10.2514/3.44425)

[40] Löbert, G., "Spanwise Lift Distribution for Forward- and Aft-Swept Wings in Comparison to the Optimum Distribution Form," *Journal of Aircraft*, Vol. 18, No. 6, 1981, pp. 496-498. (doi:10.2514/3.44717)

[41] 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, pp. 431-441. (doi:10.2514/1.C035027)

[42] Phillips, W. F., Hunsaker, D. F., and Taylor, J. D., "Minimizing Induced Drag with Weight Distribution, Lift Distribution, Wingspan, and Wing-Structure Weight," AIAA 2019-3349, AIAA Aviation 2019 Forum, Dallas, Texas, 17-21 June 2019. (doi: 10.2514/6.2019-3349)

[43] Taylor, J. D., and Hunsaker, D. F., "Minimum Induced Drag for Tapered Wings Including Structural Constraints," *Journal of Aircraft*, Article in Advance. (doi:10.2514/1.C035757)

[44] Zhang, Z. J., "Exploratory High-Fidelity Aerostructural Optimization Using an Efficient Monolithic Solution Method," *PhD Thesis*, University of Toronto, 2017.

[45] Hoogervorst, J. E. K. and Elham, A., "Wing aerostructural optimization using the Individual Discipline Feasible Architecture," *Aerospace Science and Technology*, Vol. 65, 2017, pp. 90-99. (doi:10.1016/j.ast.2017.02.012)

[46] Jansen, P. W., Perez, R. E., and Martins, J. R. R. A., "Aerostructural Optimization of Nonplanar Lifting Surfaces," *Journal of Aircraft*, Vol. 47, No. 5, 2010, pp. 1490–1503. (doi: 10.2514/1.44727)

[47] Liem, R. P., Martins, J. R. R. A., and Kenway, G. K. W., "Expected drag minimization for aerodynamic design optimization based on aircraft operational data," *Aerospace Science and Technology*, Vol. 63, 2017, pp. 344-362. (doi:10.1016/j.ast.2017.01.006)

[48] Ting, E., Chaparro, D., Nguyen, N., and Fujiwara, G. E. C., "Optimization of Variable-Camber Continuous Trailing-Edge Flap Configuration for Drag Reduction," *Journal of Aircraft*, Vol. 55, No. 6, 2018, pp. 2217-2239. (doi:10.2514/1.C034810)

[49] Ranjan, P., "Computational Analysis of Planar Wings Designed for Optimum Span-Load," *MS Thesis*, University of Illinois at Urbana-Champaign, 2016.

[50] Verstraetan, J., G., and Slingerland, R., "Drag Characteristics for Optimally Span-Loaded Planar, Wingletted, and C Wings," *Journal of Aircraft*, Vol. 46, No. 3, May-June 2009, pp. 962-971. (doi:10.2515/1.39426)

[51] Wroblewski, G. E., and Ansell, P. J., "Prediction and Experimental Evaluation of Planar Wing Spanloads for Minimum Drag," *Journal of Aircraft*, Vol. 54, 2017, pp. 1664–1674. (doi:10.2514/1.C034156)

[52] Lyu, Z., and Martins, J. R. R. A., "Aerodynamic Design Optimization Studies of a Blended-Wing-Body Aircraft," *Journal of Aircraft*, Vol. 51, No. 5, September-October 2014, pp. 1604-1617. (doi:10.2514/1.C032491)

[53] van den Kieboom, K. T. H. and Elham, A., "Concurrent wing and high-lift system aerostructural optimization," *Structural and Multidisciplinary Optimization*, Vol. 57, No. 3, 2018, pp. 947-963. (doi:10.1007/s00158-017-1787-0)

[54] McGeer, T., "Wing Design for Minimum Drag with Practical Constraints," *Journal of Aircraft*, Vol. 21, 1984, pp. 879–886. (doi:10.2514/3.45058)

[55] Piperni, P., Abdo, M., Kafyeke, F., and Isikveren, A. T., "Preliminary Aerostructural Optimization of a Large Business Jet," *Journal of Aircraft*, Vol. 44, No. 5, September-October 2007, pp. 1422-1438. (doi:10.2514/1.26989)

[56] Liem, R. P., Kenway, G. K. W., and Martins, J. R. R. A., "Multimission Aircraft Fuel-Burn Minimization via Multipoint Aerostructural Optimization," *AIAA Journal*, Vol. 53, No. 1, January 2015, pp. 104-122. (doi:10.2514/1.J052940)

[57] Ning, S. A. and Kroo, I., "Multidisciplinary Considerations in the Design of Wings and Wing Tip Devices," *Journal of Aircraft*, Vol. 47, No. 2, March-April 2010, pp. 534-543. (doi:10.2514/1.41833)

[58] Kenway, G. K. W., Martins, J. R. R. A., and Kennedy, G. J., "Aerostructural optimization of the Common Research Model configuration," AIAA 2014-3274, 15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Atlanta, GA, 16-20 June 2014. (doi:10.2514/6.2014-3274)

[59] Mader, C. A., Kenway, G. K. W., Martins, J. R. R. A., Uranga, A., "Aerostructural Optimization of the D8 Wing with Varying Cruise Mach Numbers," AIAA 2017-4436, 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Denver, CO, 5-9 June 2017. (doi:10.2514/6.2017-4436)

[60] Stewart, A. J. and Hunsaker, D. F., "Minimization of Induced and Parasitic Drag on Variable-Camber Morphing Wings," AIAA 2020-0277, AIAA Scitech 2020 Forum, Orlando, FL, 6-10 January 2020. (doi:10.2514/6.2020-0277)

[61] Kennedy, G. J., Kenway, G. W., and Martins, J. R. R. A., " High Aspect Ratio Wing Design: Optimal Aerostructural Tradeoffs for the Next Generation of Materials," AIAA 2014-0596, AIAA SciTEch 2014 Forum, National Harbor, MD, 13-17 January 2014. (doi:10.2514/6.2014-0596)

[62] Phillips, W. F. and Hunsaker, D. F., "Designing Wing Twist or Planform Distributions for Specified Lift Distributions," *Journal of Aircraft*, Vol. 56, No. 2, March-April 2019, pp. 847-849. (doi:10.2514/1.C035206)

[63] Phillips, W. F., "Aircraft Performance," *Mechanics of Flight*, 2nd ed., Wiley, Hoboken, NJ, 2010, pp. 259-376.

CHAPTER 5

Effects of Wing Morphing on Aircraft Fuel Burn Along Fuel-Optimal Trajectories

Jeffrey D. Taylor* and Douglas F. Hunsaker†
Utah State University, Logan, Utah 84322-4130

Active wing shaping, or morphing, of an aircraft wing has the potential to substantially improve aircraft efficiency. In recent years, several studies have sought to quantify the efficiency improvements possible through active wing shaping, but relatively few have considered how it may affect the optimum flight-path trajectory. In this paper, we seek to characterize the fuel savings from active wing shaping over an approximate optimum flight trajectory. To accomplish this, we present a simple direct trajectory optimization framework that can be used to perform a large number of trajectory optimizations to rapidly explore the design space of aircraft employing active wing shaping controls and identify how wing shaping may affect the total aircraft fuel consumption. Example solutions are presented for the approximate optimal flight-path trajectory and fuel consumption of the NASA Ikhana high-endurance UAV configuration and the NASA Common Research Model configuration. Results indicate that the use of active wing-shaping controls for load alleviation can result in up to around 8% fuel savings over an optimized baseline design operating along the optimized trajectory. It is also shown that active wing shaping tends to favor optimal trajectories with lower velocity, higher lift coefficient, and higher lift-to-drag ratio, than the baseline design.

* PhD Candidate, Mechanical and Aerospace Engineering, 4130 Old Main Hill, AIAA Student Member
† Associate Professor, Mechanical and Aerospace Engineering, 4130 Old Main Hill, AIAA Senior Member

This paper was submitted to the AIAA SciTech 2023 Forum in Orlando, FL.

Nomenclature

A = wing-structure cross-sectional area

a_1 = fit coefficient in the power model for a high-bypass ratio turbofan engine as a function of velocity

a_2 = fit coefficient in the power model for a high-bypass ratio turbofan engine as a function of velocity

B_n = Fourier coefficients in the lifting-line solution for the dimensionless section-lift distribution, Eq. (14)

b = wingspan

C_D = drag coefficient

C_{D_0} = fit coefficient in the parabolic approximation of the drag coefficient as a function of the lift coefficient

C_{D_1} = fit coefficient in the parabolic approximation of the drag coefficient as a function of the lift coefficient

C_{D_2} = fit coefficient in the parabolic approximation of the drag coefficient as a function of the lift coefficient

$C_{D_{2,p}}$ = parasitic drag component of the fit coefficient in the parabolic approximation of the drag coefficient as a function of the lift coefficient

C_{D_i} = induced drag coefficient

C_L = lift coefficient

C_{M_1} = fit coefficient in the exponential approximation of the drag coefficient with respect to Mach number

C_{M_2} = fit coefficient in the exponential approximation of the drag coefficient with respect to Mach number

C_n = weighting coefficients for B_n in the expression for wing-structure weight of tapered wings

C_{TSFC} = static thrust-specific fuel consumption coefficient in the fuel consumption model for a turbofan engine

C_{δ} = shape coefficient for deflection-limited sizing of the wing structure, Eq. (A2)

C_{σ} = shape coefficient for stress-limited sizing of the wing structure, Eq. (33)

c = thrust-specific fuel consumption

c_W = local wing section chord length

D = total drag

E = modulus of elasticity of the wing-structure material

e = Oswald efficiency factor

e_s = span efficiency factor

h = altitude

h_s = height of the beam cross-section

I = beam section moment of inertia

L = total lift

$\widetilde{L}$ = local wing section lift

M = Mach number

$\widetilde{M}_b$ = local wing section bending moment

m = exponential fit coefficient for the ratio of temperature at altitude h to temperature at sea level in the power model for a high-bypass ratio turbofan engine

N_1 = throttle parameter in the power model for a high-bypass ratio turbofan engine

n_e = number of engines

n_m = limiting load factor at the maneuvering-flight design limit

P_A = engine power available

P_R = power required to maintain steady level flight

p_{ij} = fit coefficients in the multidimensional polynomial available power model for a turboprop engine, as a function of altitude and airspeed

q = exponential fit coefficient for the Mach number in the fuel consumption model for a turbofan engine

q_{ij} = fit coefficients in the multidimensional polynomial power-specific fuel consumption model for a turboprop engine, as a function of altitude and airspeed

q_p = power-specific fuel consumption

R_A = wing aspect ratio

R_T = wing taper ratio

S_W = wing area

T = engine thrust

T_0 = reference static engine thrust at sea-level

T_R = thrust required to maintain steady level flight

t = cruise time

$t_{\max}$ = maximum thickness of the local airfoil wing section

V = freestream velocity

V_c = aircraft climb rate

W = aircraft gross weight

W_{end} = aircraft weight at the end of cruise

W_f = aircraft fuel weight

W_n = aircraft net weight, defined as $W - W_s$

W_r = that portion of W_n carried at the wing root

W_s = wing-structure weight

$\widetilde{W}_n$ = 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

x = downrange distance variable

z = spanwise wing coordinate relative to the midspan

δ_{max} = maximum wing deflection

γ = specific weight of the wing-structure material

σ_{max} = maximum longitudinal stress

θ = change of variables for the spanwise coordinate, Eq. (14)

ρ = air density

τ_0 = reference air temperature at sea-level

τ_h = air temperature at altitude h

I. Introduction

The ability to morph, or actively change the shape of, an aircraft's wings may have the potential to substantially reduce aircraft fuel consumption. The level of fuel savings possible through wing morphing depends on a variety of operational and design parameters. Recent years have seen increased interest in the development of aircraft morphing mechanisms [1-8], including several camber-morphing mechanisms [9-13], that allow designers to actively tailor the shape of a wing to achieve desired aerodynamic characteristics across a variety of flight conditions. In conjunction with these efforts, several studies have sought to determine how wing shaping using morphing mechanisms may affect aircraft performance over a representative flight-path trajectory. However, relatively few of these studies have considered how wing shaping may also affect the *optimal* flight-path trajectory. In this paper, we present a series of reference solutions that illustrate some of the ways in which static and active wing shaping may affect the efficiency of an aircraft over its optimal flight-path trajectory.

The ability to actively change the shape of a wing, particularly through twist or camber morphing, can be leveraged to improve aerodynamic efficiency and provide aerostructural load alleviation. The lift distribution on a wing is related to the wing planform shape, the wing geometric twist distribution, and the airfoil cross sections across the wing (aerodynamic twist distribution). Active shaping of wing geometric or aerodynamic twist therefore produces changes in the aerodynamic lift distribution. The elliptic lift distribution, which was first identified by Prandtl in 1918 [14,15] as the lift distribution that minimizes

induced drag, is considered to be the optimum lift distribution for cruise. Since 1918, the elliptic lift distribution has been shown to be optimal for a wide range of aircraft and flight conditions. Therefore, aircraft typically operate with nearly elliptic lift distributions during cruise. However, for non-morphing aircraft wings, as flight conditions change over the course of a typical trajectory, the aerodynamic load distribution also may change, and the aircraft may experience suboptimal performance.

Active wing-shaping control can be used to mitigate these negative effects by adjusting the aerodynamic load distribution to maintain desired performance at a variety of design conditions. This especially true for flexible wings, where aeroelastic effects can further degrade performance at off-design conditions. For example, Lebofsky et al. [16,17] showed that drag may be reduced by 10-20% under trim cruise conditions when the wing of the Generic Transport Model (GTM) is actively shaped to mitigate negative aeroelastic effects using a morphing mechanism known as the Variable Camber-Continuous Trailing-Edge Flap (VCCTEF). In transonic flight, Ting et al. [18] and Chaparro et al. [19] showed that drag on the GTM can be reduced by 5-8% at off-design conditions using the VCCTEF. Similar results have been shown for various other aircraft configurations and morphing mechanisms [20-28].

When wing-sizing constraints are considered, static wing design involves a tradeoff between efficiency during cruise and structural requirements in high-load maneuver conditions. The wing structure is generally sized based on limiting load conditions, including a high-load maneuver. Several theoretical aerostructural studies [29-43], beginning with Prandtl in 1933 [29], show that, under structural constraints, optimizing a wing for minimum induced drag involves tradeoffs between the wingspan, the lift distribution, and the wing weight distribution. These tradeoffs often result in an optimum wing design that takes advantage of load alleviation provided by a non-elliptic lift distribution to reduce weight or extend the wingspan without adding weight. Taylor and Hunsaker [44] provide a thorough review of theoretical aerosturctural literature for minimizing induced drag, as well as a sampling of more recent computational studies aimed at optimizing aircraft efficiency through aerostructural wing design. These studies highlight how tailoring the aerodynamic lift distribution to alleviate bending moments may have a substantial impact on aircraft efficiency.

For wings with active wing shaping, the lift distribution may be changed dynamically to provide both load alleviation at high-load maneuvers and high efficiency during cruise. Recently, Hunsaker et al. [45] estimated that this load alleviation could result in around 10% reduction in cruise drag on a long-endurance UAV. Nguyen et al. [21] showed that active wing shaping using the VCCTEF could reduce drag on the GTM by 6% and reduce the root bending moment by around 25%, including both flutter constraints and the effects of load alleviation. Burdette et al. [46,47] showed that for a morphing retrofit on the CRM, fuel burn may be reduced by between about 0.3% and 1% over the optimized non-morphing configuration. This is in good agreement with results shown by Lyu and Martins [48] for a variety of flight ranges. Fujiwara estimated a 4.7% reduction in fuel burn using morphing on the CRM under similar conditions [23].

The majority of published literature on the impacts of wing morphing or active wing shaping on aircraft efficiency uses multipoint analysis or optimization at a series of predetermined flight conditions or a given fixed flight trajectory. For example, in Refs. [29-43], the wing lift distribution is assumed either to be fixed for all flight conditions, including the critical load maneuver, or to change only due to passive aeroelastic or aerodynamic effects. The studies in Refs. [16-23] and [45-48] use multipoint optimization at only a handful of points in the flight trajectory. Relatively few studies consider how morphing may also affect the optimal flight-path trajectory. Nguyen et al. [49] presented trajectory optimization results for a transonic truss-braced wing employing the VCCTEF to minimize fuel consumption. Fasel et al. [50] sought to simultaneously optimize the design and trajectory of an energy kite with camber morphing and found that morphing may result in up to 8% increase in power production. Jasa et al. [51] performed simultaneous aerostructural weight reduction and trajectory optimization for the CRM and showed fuel burn reductions of under 1% over the static cruise-optimized design. Rudnick-Cohen et al. [52,53] have also performed simultaneous design and trajectory optimizations for wings with camber morphing for a variety of performance objectives.

In this paper, we present a series of solutions highlighting the effects of active wing shaping on aircraft performance, considering both aerostructural load alleviation and the impact of morphing on the optimum

flight trajectory. Note that in this paper, we will not consider the effects of aeroelasticity in our analysis. We anticipate that these solutions will add to the relatively sparse literature including the effects of wing morphing on the optimal flight trajectory and serve as reference solutions to inform future design and research efforts. Many of the methods used in the publications referenced here rely on linking black-box computational models from which relational information is very difficult to obtain. Such relational information is highly valuable in revealing how certain design and operational parameters may affect overall flight performance. The discrete nature of computational models means that obtaining relational information requires a large number of individual computational runs to reveal trends. Due to the computational expense of many of these methods, obtaining a sufficient number of results to reveal relational trends is infeasible. By using low- and multi-fidelity methods, we can quickly obtain a wide range of solutions with relatively low computational cost and reveal important trends and insights to support ongoing research and development in the design and optimization of wings with active wing-shaping controls. Therefore, in this paper, we use low- and multi-fidelity analysis and optimization methods, which are described in Sections II and III. In Section IV, we give a description of the case studies considered in this paper, and in Section V, we present a discussion of the insights that can be gained from the results of those case studies.

In the following sections, we will consider cases involving both static and active wing shaping. In this paper, static wing shaping refers to the use of aerodynamic or geometric twist distribution to achieve a single, fixed lift distribution for all flight conditions along the flight trajectory. In this way, static wing shaping is meant to approximate a rigid non-morphing wing operating with only small variations in the lift coefficient. Cases involving static wing shaping are included in this paper primarily for reference purposes. Active wing shaping refers to the active use of wing morphing mechanisms to dynamically tailor the aerodynamic lift distribution over the flight trajectory.

II. Fuel Consumption for Quasi-Steady Level Flight

In order to determine how wing shaping may affect the efficiency of an aircraft over the flight-path trajectory, we use the fuel consumption as our principal efficiency metric. For most long-haul trajectories, we can assume that changes in altitude and velocity are small relative to changes in downrange position. Therefore, we will assume that the aircraft is in quasi-steady level flight. Consider an aircraft in quasi-steady level flight with weight W and thrust T. If we assume that the thrust is oriented in the direction of flight, then level flight requires that the lift L be equal to the weight, i.e.,

$$L = W \tag{1}$$

and steady flight requires that the drag D be equal to the thrust, i.e.,

$$D = T \tag{2}$$

The lift and drag can be rewritten in terms of the lift coefficient C_L and the drag coefficient C_D as

$$L = \frac{1}{2}\rho V^2 S_W C_L \tag{3}$$

$$D = \frac{1}{2}\rho V^2 S_W C_D \tag{4}$$

where ρ is the freestream density, V is the freestream velocity, and S_W is the reference area of the aircraft wing. Rearranging Eq. (3) and using the relation given in Eq. (1) gives an expression for the lift coefficient in quasi-steady level flight, i.e.,

$$C_L = \frac{W}{\frac{1}{2}\rho V^2 S_W} \tag{5}$$

which is also sometimes referred to as the weight coefficient. For low-speed subsonic flows, where the freestream can be assumed to be nearly incompressible, the drag coefficient is well approximated as a parabolic function of the lift coefficient, i.e.,

$$C_D = C_{D_0} + C_{D_1} C_L + C_{D_2} C_L^2 \tag{6}$$

where C_{D_0}, C_{D_1}, and C_{D_2} are constant fit coefficients that depend on the aircraft configuration.

In high-speed subsonic flight, compressibility effects can alter the lift and drag coefficients. Near a Mach 1, the formation of shockwaves in the flow causes a substantial increase in total drag. This is

sometimes known as drag divergence. For most aircraft, the drag divergence can be approximated below M=1 by modifying Eq. (6) as

$$C_D(C_L, M) = \left(C_{D_0} + C_{D_1} C_L + C_{D_2} C_L^2\right)\left(1 + C_{M_1} M^{C_{M_2}}\right) \tag{7}$$

where C_{M_1} and C_{M_2} are fit coefficients in the expression of the drag coefficient as a function of Mach number. Note that Eq. (7) does not approximate the drag around and above $M = 1$. Nevertheless, for the purposes of this study, it does provide a reasonable approximation of the drag divergence for high-speed subsonic flight below $M = 1$.

A. Fuel Consumption

The amount of fuel used by an aircraft power plant can be related to the thrust through a parameter known as the thrust-specific fuel consumption as

$$\dot{W} = -cT \tag{8}$$

where $\dot{W}$ is the time rate of change of weight of the aircraft due to the fuel burn and c is the thrust-specific fuel consumption. Equation (8) can also be rewritten in terms of the range variable as

$$W' = -\frac{cT}{V} \tag{9}$$

where the notation $\bullet'$ represents a derivative with respect to the downrange variable x. Using Eq. (8), the total fuel consumption W_f of an aircraft over a specified time interval $t_0 \le t \le t_1$ can be written as

$$W_f = \int_{t_0}^{t_1} cT dt \tag{10}$$

Equation (9) can be used to give the total fuel consumption over a specified distance $x_0 \le x \le x_1$, i.e.,

$$W_f = \int_{x_0}^{x_1} \frac{cT}{V} dx \tag{11}$$

The fuel consumption can also be written in terms of engine power rather than engine thrust using the power-specific fuel consumption, q_p, as

$$\dot{W} = -q_p TV \tag{12}$$

where the product TV represents the power produced by the aircraft engine. In terms of the range variable x, Eq. (12) can be rewritten as

$$W' = -q_p T \tag{13}$$

The thrust-specific fuel consumption and the power-specific fuel consumption are dependent on the powerplant and are typically functions of the flight velocity and the air properties, which change with altitude.

B. Effects of Active Wing Shaping

In this paper, we model the effects of morphing primarily through the term C_{D_2} in Eqs. (6) and (7). This term includes effects from both parasitic drag and induced drag. For a wing that is optimized for efficiency at a given cruise condition, the addition of morphing is expected to have very little effect on the parasitic drag. However, changing the aerodynamic lift distribution through morphing may have a substantial impact on the induced drag.

Classical lifting-line theory describes the relationship between the aerodynamic lift distribution and the induced drag. From classical lifting-line theory, the normalized lift distribution can be expressed in terms of a Fourier sine series as [42]

$$\frac{b\widetilde{L}(\theta)}{L} = \frac{4}{\pi}\left[\sum_{n=1}^{\infty} B_n \sin(n\theta)\right]; \qquad B_n = \frac{A_n}{A_1} \tag{14}$$

where B_n are the normalized Fourier coefficients. These coefficients are determined based on the planform distribution and aerodynamic and geometric twist distributions. Therefore, in this study, we will assume that the aerodynamic effects of wing shaping through the aerodynamic and/or geometric twist can be modeled using the Fourier coefficients B_n.

The induced drag coefficient from classical lifting-line theory is written as

$$C_{D_i} = \frac{C_L^2}{\pi e_s R_A} \tag{15}$$

where e_s is the span efficiency factor, given by

$$e_s = \frac{1}{1 + \sum_{n=2}^{N} nB_n^2} \tag{16}$$

and R_A is the aspect ratio, given by

$$R_A = \frac{b^2}{S_W} \tag{17}$$

Using the induced drag coefficient from Eq. (15), the term C_{D_2} can be rewritten as

$$C_{D_2} = C_{D_{2,p}} + \frac{1}{\pi e_s R_A} \tag{18}$$

where $C_{D_{2,p}}$ is the component of the C_{D_2} term that comes from the parasitic drag. It is important to note that the span efficiency factor e_s, which includes only induced-drag effects, is different from the Oswald efficiency factor e, which includes both parasitic and induced drag effects. Equation (18) can alternatively be written as

$$C_{D_2} = \frac{1}{\pi e R_A} \tag{19}$$

where the effects of parasitic drag are included in the Oswald efficiency factor.

For aircraft without wing shaping, the wing is typically designed to achieve desired aerodynamic characteristics, such as minimum drag, at a design cruise lift coefficient. As the aircraft operates away from the design lift coefficient, the lift distribution changes and the aircraft no longer has minimum drag. However, for wings with active wing shaping, the wing shape can be tailored to achieve minimum drag at all lift coefficients, resulting in a drag polar with lower curvature, and lower drag at all off-design lift coefficients than for a wing with no wing shaping. This tends to increase the maximum lift-to-drag ratio and the lift coefficient at which it is achieved for a wing with wing shaping.

III. Trajectory Optimization Framework

In order to provide a consistent assessment of the effects of wing morphing on aircraft fuel consumption, we compare the fuel usage of non-morphing designs to that of morphing designs along each design's respective optimum trajectory. Many methods for aircraft trajectory optimization can either be

described as direct or indirect methods. Using a direct method, the aircraft trajectory is discretized directly and trajectory characteristics are obtained, typically using numerical optimization methods. Using an indirect method, the conditions for optimality are derived from optimal control theory in the form of a system of differential equations that can be solved to obtain the optimum trajectory. In this study, we employed a direct trajectory optimization method, which will be described in this section, to approximate the minimum-fuel optimal trajectory.

A. Minimizing Fuel Burn with Altitude and Velocity

We seek to identify the trajectory that minimizes fuel burn for an aircraft in quasi-steady level flight, using the altitude and velocity as the control variables. In general, this requires that we minimize Eq. (10) or (11). However, for a wing in steady-level flight, the calculus of variations shows that minimizing the functional in Eq. (10) or (11) is equivalent to minimizing the fuel consumption rate shown in Eq. (8) or (9), respectively. From the calculus of variations, minimizing a functional with respect to any function requires, by theorem, that the Euler Lagrange equation be satisfied, i.e.,

$$L_y - \frac{d}{dx} L_{y'} = 0 \tag{20}$$

where L is the Lagrangian of the functional, y is the design variable of interest, and the subscript denotes a partial derivative. For the case where we wish to find both the altitude h and velocity V that minimize Eq. (11), the Euler-Lagrange equation from Eq. (20) becomes

$$\frac{\partial}{\partial h}\left(\frac{cT}{V}\right) - \frac{d}{dx}\frac{\partial}{\partial h'}\left(\frac{cT}{V}\right) = 0 \tag{21}$$

and

$$\frac{\partial}{\partial V}\left(\frac{cT}{V}\right) - \frac{d}{dx}\frac{\partial}{\partial V}\left(\frac{cT}{V}\right) = 0 \tag{22}$$

For quasi-steady level flight, we assume that the time rate of change of altitude h' and velocity V' are negligible, meaning that the fuel consumption rate, cT/V is not dependent on h' or V'. Equations (21) and (22) then reduce to

$$\frac{\partial}{\partial h}\left(\frac{cT}{V}\right)=0 \tag{23}$$

and

$$\frac{\partial}{\partial V}\left(\frac{cT}{V}\right)=0 \tag{24}$$

which can be solved for h and V to find the altitude and velocity that minimize fuel consumption. Note that this is equivalent to minimizing the fuel consumption rate from Eq. (9). This suggests that we can obtain a good approximation of the overall fuel consumption by minimizing the fuel consumption rate at each point along the flight trajectory.

Therefore, in this study, the altitude and velocity are chosen at each trajectory point to minimize the fuel burn at that point. For a typical airframe/powerplant combination, there is a tradeoff between the thrust- or power-specific fuel consumption, the flight velocity, the air density (which depends on the flight altitude), and the engine thrust (which for quasi-steady level flight is equal to the drag). Based on this tradeoff, there is often an altitude and velocity that minimize fuel burn for a given fixed aircraft configuration. This optimum altitude and velocity depend on various aircraft design parameters, including the weight, lift coefficient, and drag coefficient, which may change over the course of a flight, particularly for wings with active wing-shaping controls.

To determine the optimum altitude and velocity for each point in the cruise trajectory, the trajectory is discretized into sections with N evenly spaced control points. The objective is to minimize fuel consumption at each control point. The fuel consumption is evaluated using Eq. (9) or Eq. (13), where the thrust is equal to the drag, as shown in Eq. (2). The drag is found by combining Eqs. (4) and (7), with the lift coefficient specified by Eq. (5). In order to minimize the fuel consumption, the altitude and velocity are chosen such that Eq. (9) or Eq. (13) is minimized at each control point. This can be accomplished using a variety of existing numerical optimization methods. In this paper, we will utilize the SciPy* implementation of the Sequential Least-Squares Programming (SLSQP) method [54].

* docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html

It is important to note that the lift coefficient from Eq. (5) depends on the current aircraft weight. Therefore, the fuel consumption at point *i* in the flight trajectory depends on the fuel weight that was lost due to fuel consumption at point *i*-1, which means that the optimization must be performed sequentially. If the aircraft initial weight is known, then the optimization should begin at the start cruise point $i = 0$ and proceed until the final end-cruise weight is obtained. If the aircraft end-cruise weight is specified, then the optimization can begin at the end-cruise point $i = N$ and proceed in reverse order until the initial weight is obtained. The total fuel burned over the course of the trajectory can then be found as

$$W_f = W_{i=0} - W_{i=N} \tag{25}$$

B. Aerodynamic Ceiling

For an aircraft in powered flight, there is a maximum altitude at which the aircraft can fly which depends on the powerplant performance and the aerodynamic characteristics of the aircraft. For an aircraft in steady level flight with velocity V, the power required, P_R, for an aircraft to maintain steady level flight can be expressed as

$$P_R = T_R V \tag{26}$$

where T_R is the thrust required to maintain steady-level flight and is equal to the drag, according to Eq. (2). The power available to the aircraft, P_A, depends on the powerplant and generally decreases with density as the altitude increases. When the power required exceeds the power available, the aircraft sinks; when the power available exceeds the power required, the aircraft climbs, according to the relation

$$V_c = \frac{P_A - P_R}{W} \tag{27}$$

where V_c is the aircraft rate of climb. Because the power available generally decreases with altitude, the climb rate also tends to decrease with altitude. When the climb rate is zero, the aircraft is said to have reached its absolute ceiling; when the climb rate reaches 100 ft/min, the aircraft is said to have reached its service ceiling. In this study, we constrain the optimization such that the aircraft remains below the service ceiling. All atmospheric properties are evaluated using the 1976 U.S. standard atmosphere model [55].

C. Optimization Summary

In this paper, we will consider trajectories where the altitude varies along the length of the trajectory, trajectories where the altitude is constant along the length of the trajectory and is chosen to minimize fuel consumption, and trajectories that have a given altitude that remains fixed along the length of the trajectory. For trajectories with varying altitude $h(x)$ along their length, both the altitude and velocity are optimized at every control point to minimize the fuel consumption rate. For each point, the optimization can be summarized as

$$\begin{aligned} \text{minimize:} \quad & W'(x_i) \\ \text{with respect to:} \quad & h(x_i),\ V(x_i) \\ \text{subject to:} \quad & V_{c,i} \geq 100 \text{ ft/min} \\ & h_i > 0 \\ & V_i > 0 \end{aligned}$$

The climb rate is evaluated at each optimizer iteration to ensure that the aircraft remains below the service ceiling. When the optimum altitude, velocity, and corresponding fuel consumption for point x_i are returned by the optimizer, the aircraft weight for the subsequent point (x_{i+1} if the initial aircraft weight is specified, x_{i-1} if the end-cruise weight is specified) is updated according to

$$\begin{cases} W(x_{i+1}) = W(x_i) - W'(x_i)(x_{i+1} - x_i), & \text{for } W(x_0) \text{ known} \\ W(x_{i-1}) = W(x_i) + W'(x_i)(x_i - x_{i-1}), & \text{for } W(x_N) \text{ known} \end{cases} \tag{28}$$

The optimization is then performed at the subsequent point. A schematic of the optimization procedure is shown in Fig. 1.

Aircraft trajectories are often constrained such that the altitude is constant over cruise. Therefore, in addition to operation over an optimum variable-altitude trajectory, we estimate the effects of wing shaping over a constant-altitude trajectory, where the altitude is optimized to minimize fuel consumption. We will also consider a fixed-altitude trajectory, where the altitude is fixed at some prescribed value. Throughout the remaining sections, we refer to these trajectory types as variable altitude, constant altitude, and fixed altitude, respectively.

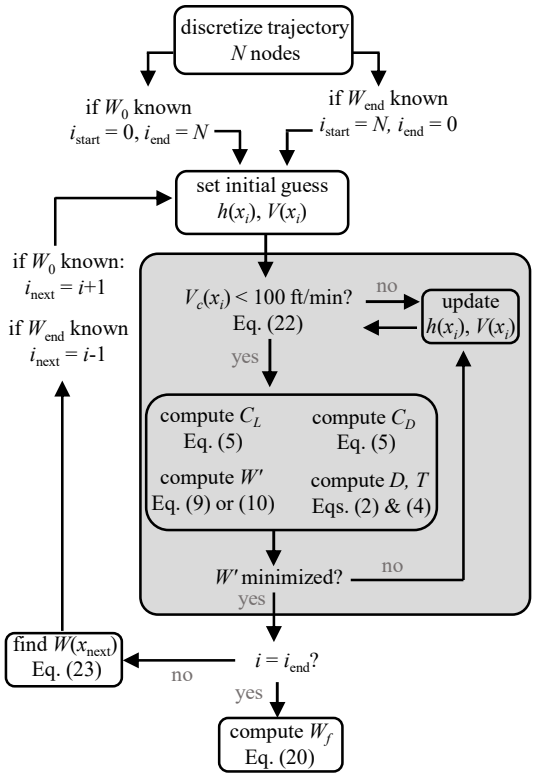


Fig. 1 Schematic of the trajectory optimization procedure for trajectories with varying altitude.

For trajectories with constant altitude along their length, the optimization procedure requires nested optimization. In the inner loop, the velocity at each point x_i is chosen to minimize the instantaneous fuel consumption rate for a given altitude. The inner-loop optimization is summarized as

$$\begin{aligned} \text{minimize:} \quad & W'(x_i) \\ \text{with respect to:} \quad & V(x_i) \\ \text{subject to:} \quad & V_{c,i} \geq 100 \text{ ft/min} \\ & V_i > 0 \end{aligned}$$

Again, the optimization is performed sequentially, and at each point, the weight is updated according to Eq. (28). In the outer loop, the altitude is chosen to minimize the total trajectory fuel burn. For the outer loop, the optimization is summarized as follows:

$$\begin{aligned} \text{minimize:} \quad & W_f \\ \text{with respect to:} \quad & h \\ \text{subject to:} \quad & h > 0 \end{aligned}$$

A schematic of the nested setup for the constant-altitude optimization is shown in Fig. 2.

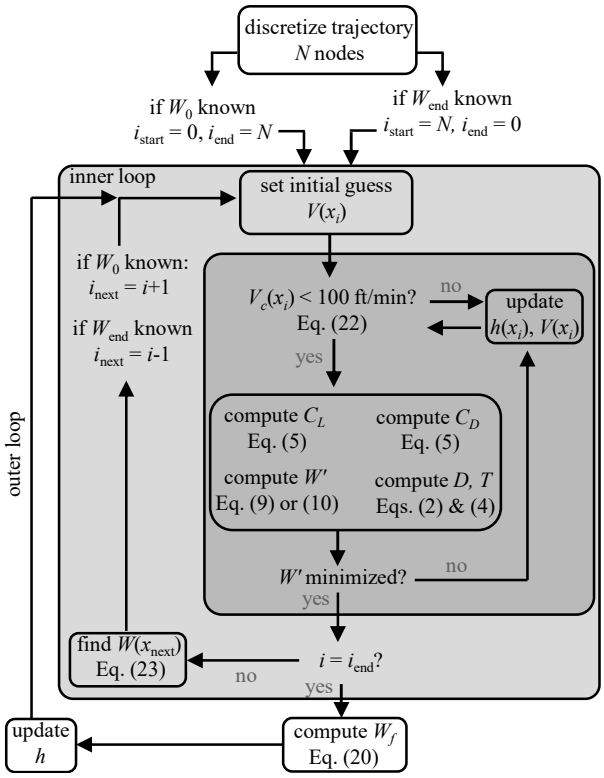


Fig. 2 Schematic of the trajectory optimization procedure for trajectories with constant altitude.

IV. Case Studies

In order to assess the impact of active wing shaping using the methodology described above, we performed trajectory optimization case studies on the Ikhana long-endurance UAV airframe and an augmented variant of the NASA Common Research Model (CRM) aircraft. These two aircraft were selected based on the accessibility of publicly available data and because they represent distinct aircraft configurations and flight regimes.

A. NASA Ikhana

The NASA Ikhana airframe is a modified Predator-B airframe used by NASA for scientific and wildfire monitoring missions. Ikhana has an unswept, tapered wing with negligible dihedral. A characterization of the Ikhana geometry from publicly available data is given by Taylor and Hunsaker [56].

For the purposes of this study, we approximate Ikhana's wing area to be S_W = 265.6 ft^2 with a maximum weight of W = 8500 lbf. The end-cruise weight is approximated at W_{end} = 5950 lbf and includes the airframe empty weight and some reserve fuel weight, estimated at 15% of the total fuel capacity. For Ikhana, the drag parameters C_{D_0}, C_{D_1}, and C_{D_2} are assumed to be C_{D_0} = 0.023, C_{D_1} = 0.0, and C_{D_2} = 0.0364. The parasitic drag parameter $C_{D_{2,p}}$ is estimated to be $C_{D_{2,p}}$ = 0.017. Additional example aerodynamic and weight parameters for Ikhana are given in Table 1.

Table 1: Example aerodynamic and weight design parameters for the NASA Ikhana aircraft.

net weight, lbf	7,500
end-cruise weight, lbf	5,950
root weight, lbf	4,500
wing area, ft^2	265.63
design cruise range, mi	3500
max wing loading, lbf/ft^2	31.87
C_{D_0}	0.023
C_{D_1}	0.0
C_{D_2}	0.0364
$C_{D_{2,p}}$	0.017
C_{M_1}	3.0
C_{M_2}	30

Ikhana is equipped with a Honeywell TPE331-10 turboprop engine [57]. The performance characteristics of the TPE331-10 can be approximated from charts given by Honeywell [58]. Based on data provided in these charts, the power-specific fuel consumption and available power are each approximated as functions of altitude and velocity using a multidimensional parabolic fit of the form

$$q_p(h,V) = \sum_{i=0}^{2}\sum_{j=0}^{2} q_{ij} h^i V^j \tag{29}$$

$$P_A(h,V) = \sum_{i=0}^{2}\sum_{j=0}^{2} p_{ij} h^i V^j \tag{30}$$

where q_{ij} and p_{ij} are, respectively, the fit coefficients for the power-specific fuel consumption and power available. Example values for q_{ij} and p_{ij} are given in Table 2 for the Honeywell TPE331-10 engine.

Table 2: Example fit coefficients for the approximate parabolic engine performance model of the Honeywell TPE331-10 turboprop engine.

p_{00}	1.0048×10^{3}	q_{00}	5.5686×10^{-1}
p_{01}	3.6841×10^{-2}	q_{01}	-2.7803×10^{-5}
p_{02}	1.6006×10^{-3}	q_{02}	-3.3804×10^{-7}
p_{10}	-2.1717×10^{-2}	q_{10}	-2.6096×10^{-6}
p_{11}	2.3175×10^{-8}	q_{11}	-4.7139×10^{-9}
p_{12}	-1.9694×10^{-8}	q_{12}	2.0069×10^{-11}
p_{20}	8.4470×10^{-8}	q_{20}	7.0364×10^{-11}
p_{21}	-4.3564×10^{-11}	q_{21}	1.6919×10^{-13}
p_{22}	2.5221×10^{-14}	q_{22}	-7.6756×10^{-16}

B. NASA Common Research Model (CRM)

The CRM is a benchmark geometry typical of a swept-wing wide-body transonic transport aircraft that was originally intended for validation of Computational Fluid Dynamics (CFD) studies. The CRM geometry was first described by Vassberg [59], and CAD models of the CRM are provided by NASA.[†] The original CRM geometry includes only the outer mold line of the aircraft, but several variants exist that augment the original CRM with models that can be used to obtain additional information, including the weight distribution, structural layout, and fuel distribution. In this study, we will use the augmented CRM variant given by Taylor and Hunsaker [60], which includes basic structural properties, a fuel distribution and burn scheduling model, and a weight distribution model.

The augmented CRM given by Taylor and Hunsaker [60] includes data derived or inferred from publicly available information on the Boeing 777-200ER, which is very similar to the CRM. Using the data provided by Taylor and Hunsaker [60], we approximate the maximum takeoff weight of the CRM as W = 628,342 lbf, with a maximum fuel load of W_f = 302,270 lbf [61]. The end-cruise weight (with 15% of total fuel capacity) is estimated as W_{end} = 370,664 lbf. The wing reference area is approximated as S_W = 4130 ft^2 [59]. The example drag polar for the CRM used in this study was obtained from trimmed drag data from case 1b of the fourth AIAA drag-prediction workshop [62] and data from a general implementation of the numerical lifting-line method of Phillips and Snyder [63], as presented by Goates and Hunsaker [64]. Data from this method was obtained using transonic airfoil data from a transonic small-

[†] https://commonresearchmodel.larc.nasa.gov/

disturbance theory/integral boundary layer code [65], as detailed in [60]. The average C_{D_0}, C_{D_1}, and C_{D_2} values obtained by fitting these data using the parabolic approximation given in Eq. (6) are given by $C_{D_0} = 0.0194$, $C_{D_1} = -0.0159$, and $C_{D_2} = 0.0666$. From additional data in Ref. [62], the term $C_{D_{2,p}}$ for the parasitic drag is estimated to be $C_{D_{2,p}} = 0.0316$. Based on data given by Vassberg [59], the drag divergence begins just above $M = 0.85$, and can be modeled below $M = 1$ using values for the coefficients C_{M_1} and C_{M_2} from Eq. (6) of $C_{M_1} = 3$ and $C_{M_2} = 30$. A summary of the example aerodynamic and weight parameters for the CRM are given in Table 3.

Table 3: Example aerodynamic and weight design parameters for the CRM aircraft.

net weight, lbf	628,342
end-cruise weight, lbf	370,664
root weight, lbf	233,343
wing area, ft^2	4130
design cruise range, mi	7725
max wing loading, lbf/ft^2	152.14
C_{D_0}	0.0194
C_{D_1}	-0.0159
C_{D_2}	0.0666
$C_{D_{2,p}}$	0.0316
C_{M_1}	3.0
C_{M_2}	30

In this study, we assume that the CRM operates with a high-bypass ratio turbofan engine similar to the GE-90. The thrust-specific fuel-consumption is approximated using a relation given by Eshelby [66]:

$$c(h, M) \approx C_{\text{TSFC}} \left(\frac{\tau_h}{\tau_0} \right)^{1/2} M^q \tag{31}$$

where C_{TSFC} is a constant coefficient that depends on the engine, τ_h is the atmospheric temperature at altitude h, τ_0 is the atmospheric temperature at sea level, and q is an exponent that depends on the engine. Eshelby [66] notes that q is around 0.6 for a high-bypass ratio turbofan engine. The coefficient C_{TSFC} can be found using Eq. (31) with a reference value for the GE-90 high-bypass ratio turbofan engine of $c = 0.0563$ kg/N·h [67], or $c = 4.77 \times 10^{-6}$ slug/lbf·s, during Mach 0.84 cruise at an altitude of 35,000 ft.

Using these values in Eq. (31), and assuming that $q = 0.6$ for the high-bypass ratio GE-90 power plant gives a constant $C_{\text{TSFC}} = 6.0706 \times 10^{-6}$ slug/lbf·s.

To predict the available power, we use the engine model given by Daidzic [68], i.e.,

$$P_A = n_e N_1 T_0 \left(\frac{\tau_h}{\tau_0} \right)^m \left(V + a_1 V^2 + a_2 V^3 \right) \tag{32}$$

where n_e is the number of engines, N_1 is a throttle parameter, T_0 is the static thrust at sea level, and a_1 and a_2 are parameters that depend on the engine. For the CRM, $n_e = 2$, and for the GE-90, the static thrust at sea level is $T_0 = 93{,}000$ lbf. The exponent m is set to $m = 0.7$ when $h < 36{,}131$ ft and $m = 1.0$ for $h > 36{,}131$ ft [68]. The remaining parameters in Eq. (32) can be estimated as $N_1 = 0.9$, $a_1 = -9.50 \times 10^{-4}$, and $a_2 = 5.00 \times 10^{-7}$ [68].

C. Static and Active Wing Shaping

To assess the effects of static and active wing shaping, we compute the fuel burn along the approximate optimum trajectory for a range of configurations employing static or active wing shaping for aerodynamic efficiency and aerostructural load alleviation. Wing shaping is modeled aerodynamically through variations in the Fourier coefficient B_3 from Eq. (14), which impacts the span efficiency factor, Oswald efficiency factor, and the term C_{D_2} from Eqs. (6) and (7). The effects of aerostructural load alleviation are approximated using wing-structure weight relationships given by Taylor and Hunsaker [56,69] for wings with fixed wing loading. For both Ikhana and the CRM, we consider three cases: 1) load alleviation through static wing shaping is leveraged to decrease aircraft weight, 2) load alleviation from static wing shaping is leveraged to increase the wingspan and/or aspect ratio, and 3) load alleviation from active wing shaping is leveraged to increase the wingspan and/or aspect ratio.

For Ikhana, aerostructural predictions for the wing-structure weight and wingspan are obtained using the closed-form relationships for the stress-limited design of unswept tapered wings given in Ref. [69]. Using these relationships, the maximum wingspan structurally allowed for a wing with given weight and lift distribution is approximated by

$$b = \sqrt[3]{\frac{4\pi C_\sigma (t_{\max}/c_W)\sigma_{\max}}{\gamma(1+R_T)(W/S_W)\left(C_1 + \sum_{n=3}^{\infty} C_n B_n\right)} \frac{(W_n + W_s)W_s}{n_m W_r}}; \qquad C_\sigma \equiv \frac{2I(h_s/t_{\max})}{A h_s^{\,2}} \tag{33}$$

where R_T is the taper ratio, n_m is the maneuvering load limit in g's, W_r is the weight of the aircraft at the wing root, $t_{\max}/c_W$ is the maximum thickness-to-chord ratio of the wing airfoil section, $\sigma_{\max}$ is the maximum allowable stress, and γ is the specific weight of the wing-structure material. The parameters I, A, and h_s represent the moment of inertia, cross-sectional area, and height of the wing structure, respectively. The terms C_1 and C_n are coefficients that depend on the taper ratio. In this study, we will approximate the lift distribution using B_3 alone, as suggested by Taylor and Hunsaker [69], meaning that we will only use the C_1 and C_3 coefficients in our aeorstructural predictions. For Ikhana, we use the coefficients that correspond to a taper ratio of $R_T = 0.4$, which gives $C_1 = 2.3139\times10^{-1}$ and $C_3 = 2.4378\times10^{-1}$. The remaining relevant structural parameters for Ikhana are given in Table 4.

Table 4: Example structural parameters for Ikhana.

taper ratio	0.4
thickness-to-chord-ratio	0.1365
maneuver load limit, g	3.75
max allowable stress, psf	3.6×10^6
specific weight, slug/(ft^{2}s^2)	172.8
wing-structure weight, lbf	1,008
C_σ	0.165
C_1	0.0666
C_3	0.0316

For the CRM, the aerostructural predictions are performed using the numerical wing-structure prediction algorithm presented by Taylor and Hunsaker in ref. [56], combined with numerical optimization. In each case, optimization is carried out using the SciPy implementation of the SLSQP algorithm [54]. Again, we approximate the lift distribution assuming that $B_n = 0$ for all $n > 3$. The structural parameters for the CRM are approximated based on the low-fidelity CRM characterization given by Taylor and Hunsaker [60], as described in Appendix A. Note that in this paper, predictions for the wing-structure weight of the CRM include the weight of ribs located at each of the 48 wing-sections shown in Table A2 in Appendix A.

For case 1, where load alleviation through static wing shaping is leveraged to decrease the wing-structure weight, we hold the wingspan constant while varying the wing-structure weight. For each wing-structure weight, we determine the lift distribution that provides sufficient load alleviation to meet the given structural-weight requirement. For Ikhana, this is done by rearranging Eq. (33). Assuming that $B_n = 0$ for all $n > 3$, solving for B_3 gives

$$B_3 = \frac{4\pi C_\sigma (t_{\max}/c_W)\sigma_{\max}}{\gamma(1+R_T)(W/S_W)b^3 C_3}\frac{(W_n + W_s)W_s}{n_m W_r} - \frac{C_1}{C_3} \tag{34}$$

For the CRM, we use numerical optimization to predict the value of B_3 that minimizes the difference between the wing-structure weight predicted by the algorithm of Taylor and Hunsaker [56] and the desired wing-structure weight. In this way, we simulate tailoring of the lift distribution through wing shaping to reduce the structural weight. Note that as the lift distribution changes to alleviate more loads, the span efficiency factor given in Eq. (16) decreases, resulting in a tradeoff between lift distributions that alleviate loads at the high-load limiting condition and lift distributions that provide aerodynamic efficiency in cruise. We constrain the wing loading *W/S* to a fixed value, which means that as the weight changes, the wing area changes. Because the parasitic drag is proportional to the wetted area of the wing, as the wing area changes, we also scale the terms C_{D_0}, C_{D_1}, and $C_{D_{2,p}}$ by the new wing area.

For case 2, where load alleviation through static wing shaping is leveraged to increase the wingspan, we vary the lift distribution through B_3, and for each value of B_3, we compute the wingspan that results in the same wing-structure weight as the baseline design. For Ikhana, this is done using Eq. (33). For the CRM, we use numerical optimization to determine the wingspan that minimizes the difference between the wing-structure weight given by the algorithm of Taylor and Hunsaker [56] and the fixed wing-structure weight from the CRM baseline design for each given value of B_3. Again, for this case, as the lift distribution changes, we expect to see a tradeoff between load alleviation and aerodynamic efficiency, which results in changes in the span efficiency factor as B_3 changes.

For case 3, where active wing shaping is used to increase the wingspan, the same process is repeated as in case 2, but the span efficiency factor is fixed at 1. In this way, we simulate the use of active wing

shaping to tailor the lift distribution during maneuvers to alleviate loads (with $B_3 \neq 0$), and the use of an aerodynamically optimum configuration with $e_s = 1$ for high efficiency during cruise. For this case, the B_3 values in Table 5 represent different levels of morphing capability, with $B_3 = 0$ being no morphing capability, $B_3 = -1/3$ being full morphing capability, and all other values being some intermediate level of morphing capability. Note that for this study, we only consider B_3 values that result in all-positive spanwise lift distributions. Therefore, we limit our study to values of $B_3 > -1/3$, because when $B_3 < -1/3$, the lift distribution is no longer all-positive. A summary of the three case studies to be considered in this paper is given in Table 5.

Table 5: Summary of trajectory optimization cases for Ikhana and the CRM

Case 1: static wing shaping to reduce wing-structure weight

Ikhana						CRM					
W_s, lbf	B_3	R_A	e_s	C_{D_2}	e	W_s, lbf	B_3	R_A	e_s	C_{D_2}	e
675	-0.3384	17.0908	0.7443	0.0414	0.4503	32000	-0.2954	9.3761	0.7926	0.0728	0.4660
700	-0.3139	17.0387	0.7719	0.0406	0.4603	34000	-0.2709	9.3451	0.8196	0.0717	0.4752
725	-0.2892	16.9869	0.7994	0.0399	0.4699	36000	-0.2465	9.3143	0.8458	0.0706	0.4839
750	-0.2644	16.9354	0.8267	0.0392	0.4792	38000	-0.2223	9.2838	0.8708	0.0697	0.4920
775	-0.2394	16.8843	0.8533	0.0386	0.4881	40000	-0.1983	9.2534	0.8945	0.0689	0.4995
800	-0.2143	16.8334	0.8789	0.0381	0.4963	42000	-0.1745	9.2232	0.9163	0.0682	0.5062
825	-0.1890	16.7829	0.9032	0.0376	0.5040	44000	-0.1507	9.1932	0.9362	0.0676	0.5122
850	-0.1636	16.7326	0.9256	0.0372	0.5109	46000	-0.1272	9.1635	0.9537	0.0671	0.5174
875	-0.1381	16.6827	0.9459	0.0369	0.5170	48000	-0.1038	9.1339	0.9687	0.0668	0.5218
900	-0.1125	16.6330	0.9634	0.0366	0.5222	50000	-0.0805	9.1045	0.9809	0.0666	0.5253
925	-0.0867	16.5837	0.9780	0.0365	0.5264	52000	-0.0574	9.0752	0.9902	0.0664	0.5280
950	-0.0607	16.5346	0.9891	0.0363	0.5296	54000	-0.0345	9.0462	0.9964	0.0664	0.5298
975	-0.0346	16.4858	0.9964	0.0363	0.5317	56000	-0.0117	9.0174	0.9996	0.0665	0.5306
1008	0.0000	16.4219	1.0000	0.0364	0.5328	57028	0.0000	9.0026	1.0000	0.0666	0.5308

Case 2: static wing shaping to increase wingspan

Ikhana						CRM					
B_3	e_s	b, ft	R_A	C_{D_2}	e	B_3	e_s	b, ft	R_A	C_{D_2}	e
-0.3333	0.7500	76.4773	21.9116	0.0364	0.3994	-0.3333	0.7500	215.1715	11.2104	0.0691	0.4108
-0.3000	0.7874	75.1453	21.1550	0.0361	0.4167	-0.3000	0.7874	212.0891	10.8915	0.0684	0.4274
-0.2750	0.8151	74.2047	20.6287	0.0359	0.4294	-0.2750	0.8151	209.9479	10.6727	0.0679	0.4396
-0.2500	0.8421	73.3095	20.1340	0.0358	0.4419	-0.2500	0.8421	207.9368	10.4692	0.0674	0.4513
-0.2250	0.8681	72.4560	19.6679	0.0356	0.4541	-0.2250	0.8681	206.0433	10.2794	0.0669	0.4627
-0.2000	0.8929	71.6409	19.2279	0.0355	0.4658	-0.2000	0.8929	204.2563	10.1018	0.0666	0.4735
-0.1750	0.9159	70.8613	18.8117	0.0355	0.4770	-0.1750	0.9159	202.5659	9.9353	0.0662	0.4837
-0.1500	0.9368	70.1146	18.4173	0.0354	0.4875	-0.1500	0.9368	200.9635	9.7788	0.0660	0.4931
-0.1250	0.9552	69.3983	18.0429	0.0355	0.4974	-0.1250	0.9552	199.4414	9.6312	0.0659	0.5018
-0.1000	0.9709	68.7105	17.6870	0.0355	0.5064	-0.1000	0.9709	197.9930	9.4918	0.0658	0.5096
-0.0750	0.9834	68.0492	17.3482	0.0357	0.5146	-0.0750	0.9834	196.6120	9.3599	0.0658	0.5165
-0.0500	0.9926	67.4126	17.0251	0.0358	0.5217	-0.0500	0.9926	195.2932	9.2347	0.0660	0.5224
-0.0250	0.9981	66.7991	16.7167	0.0361	0.5278	-0.0250	0.9981	194.0317	9.1158	0.0662	0.5271
0.0000	1.0000	66.2074	16.4219	0.0364	0.5328	0.0000	1.0000	192.8234	9.0026	0.0666	0.5308

Case 3: active wing shaping (morphing) to increase wingspan

Ikhana						CRM					
B_3	e_s	b, ft	R_A	C_{D_2}	e	B_3	e_s	b, ft	R_A	C_{D_2}	e
-0.3333	1.0000	76.4773	21.9116	0.0315	0.4608	-0.3333	1.0000	215.1715	11.2104	0.0597	0.4760
-0.3000	1.0000	75.1453	21.1550	0.0320	0.4695	-0.3000	1.0000	212.0891	10.8915	0.0605	0.4832
-0.2750	1.0000	74.2047	20.6287	0.0324	0.4758	-0.2750	1.0000	209.9479	10.6727	0.0611	0.4883
-0.2500	1.0000	73.3095	20.1340	0.0328	0.4819	-0.2500	1.0000	207.9368	10.4692	0.0617	0.4931
-0.2250	1.0000	72.4560	19.6679	0.0332	0.4877	-0.2250	1.0000	206.0433	10.2794	0.0622	0.4976
-0.2000	1.0000	71.6409	19.2279	0.0336	0.4934	-0.2000	1.0000	204.2563	10.1018	0.0628	0.5020
-0.1750	1.0000	70.8613	18.8117	0.0339	0.4988	-0.1750	1.0000	202.5659	9.9353	0.0633	0.5061
-0.1500	1.0000	70.1146	18.4173	0.0343	0.5041	-0.1500	1.0000	200.9635	9.7788	0.0638	0.5101
-0.1250	1.0000	69.3983	18.0429	0.0346	0.5093	-0.1250	1.0000	199.4414	9.6312	0.0643	0.5139
-0.1000	1.0000	68.7105	17.6870	0.0350	0.5142	-0.1000	1.0000	197.9930	9.4918	0.0648	0.5176
-0.0750	1.0000	68.0492	17.3482	0.0353	0.5191	-0.0750	1.0000	196.6120	9.3599	0.0653	0.5211
-0.0500	1.0000	67.4126	17.0251	0.0357	0.5238	-0.0500	1.0000	195.2932	9.2347	0.0657	0.5244
-0.0250	1.0000	66.7991	16.7167	0.0360	0.5283	-0.0250	1.0000	194.0317	9.1158	0.0662	0.5276
0.0000	1.0000	66.2074	16.4219	0.0364	0.5328	0.0000	1.0000	192.8234	9.0026	0.0666	0.5308

V. Results

For each of the cases described in the previous section, we approximate the optimum trajectory and the overall fuel burn along that trajectory. These values are compared to identify which static and active wing-shaping configurations for both the Ikhana airframe and the CRM provide a minimum in overall fuel consumption. The results of this study are presented in this section.

Recall that in this paper, the impact of wing morphing on aircraft fuel burn is primarily modeled through the span efficiency factor, which changes as wing shaping is used to tailor the lift distribution through B_3. This, in turn, affects the Oswald efficiency factor and the term C_{D_2}. In order to assess the extent to which variations in these parameters affect the overall aircraft fuel consumption over the optimal trajectory, we approximate the optimum trajectory and the corresponding fuel consumption for various values of the Oswald efficiency factor e and various wing aspect ratios for both the Ikhana and the CRM. The results are shown in Figs. 3 and 4, respectively, for Oswald efficiency factors ranging between $0.4 \leq e \leq 1.0$ and aspect ratios between $6 \leq R_A \leq 20$. It is important to remember that each of the points in the curves shown in Figs. 3 and 4 represents a full trajectory optimization. Similar results to those shown in Figs. 3 and 4 can be obtained by optimizing the trajectory such that the altitude is constant along the flight path, as described in Section III.C.

Figure 5 shows the altitude and velocity profiles from the optimum trajectory for four points in Ikhana design space, as represented in Fig. 3, for both variable-altitude and constant-altitude cruise. Analogous results are shown for the CRM in Fig. 6, for four points in the CRM design space, as represented in Fig. 4. Note that at $R_A = 16$, the optimum altitude for Ikhana tends to decrease with increasing Oswald efficiency factor. For the CRM, at $R_A = 10$, the optimum altitude tends to increase as the Oswald efficiency factor increases. For both aircraft, the optimum velocity tends to either remain constant or decrease as the Oswald efficiency factor increases.

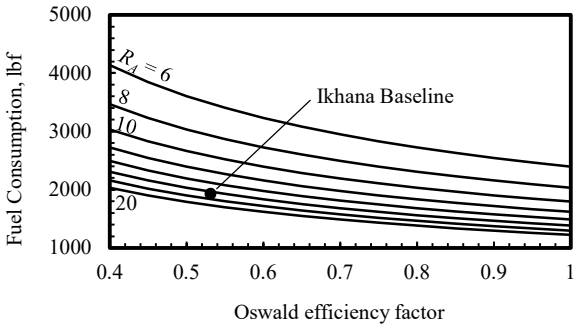


Fig. 3 Variation in the overall fuel consumption over the optimized trajectory with changes in aspect ratio and Oswald efficiency factor for Ikhana.

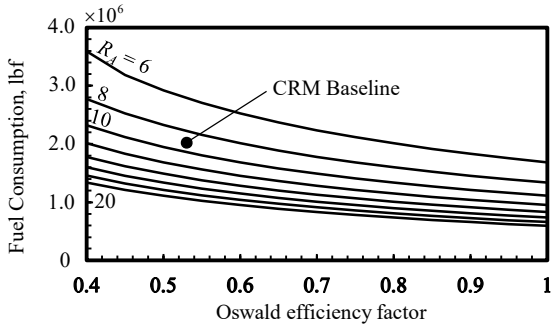


Fig. 4 Variation in the overall fuel consumption over the optimized trajectory with changes in aspect ratio and Oswald efficiency factor for the CRM.

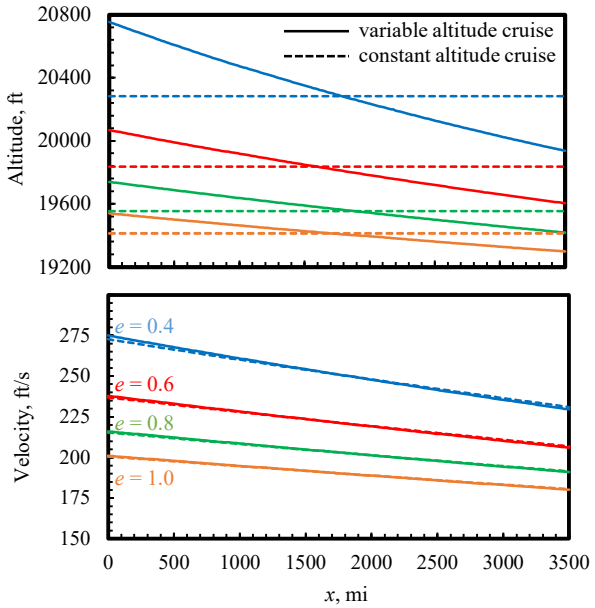


Fig. 5 Example optimized altitude and velocity profiles for variations of Ikhana with aspect ratio $R_A = 16$ and various Oswald efficiency factors.

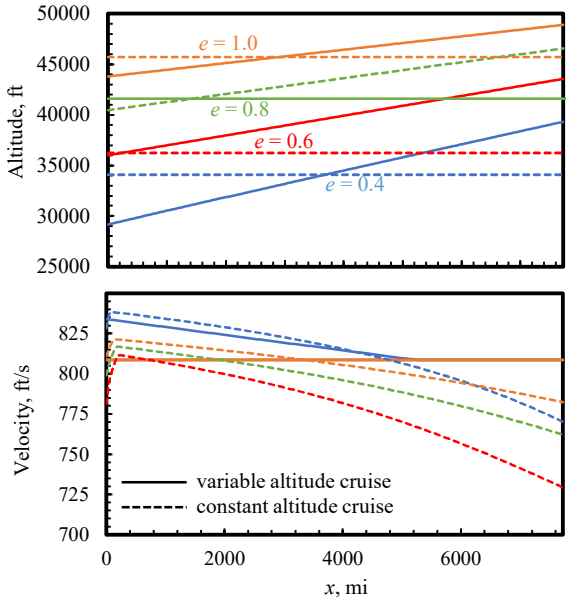


Fig. 6 Example optimized altitude and velocity profiles for variations of the CRM with aspect ratio R_A = 10 and various Oswald efficiency factors.

Because C_{D_2} is a function of both the aspect ratio and the Oswald efficiency factor, as shown in Eq. (19), the results from Figs. 3 and 4 can be simplified in terms of C_{D_2}. Figures 7 and 8 show the fuel consumption for Ikhana and the CRM, respectively, as a function of C_{D_2}. The vertical lines in Figs. 7 and 8 represent the C_{D_2} values for the baseline design and the optimum configurations for each of the cases listed in Table 5, which are described later in this section. Notice that there is a nearly linear relationship between the coefficient C_{D_2} and the overall fuel consumption over the optimal trajectory for both Ikhana and the CRM. Variation in average characteristics of the optimum trajectories corresponding to the fuel-consumption values in Figs. 7 and 8 are given as a function of C_{D_2} in Appendix B for Ikhana and the CRM in both variable-altitude and fixed-altitude cruise.

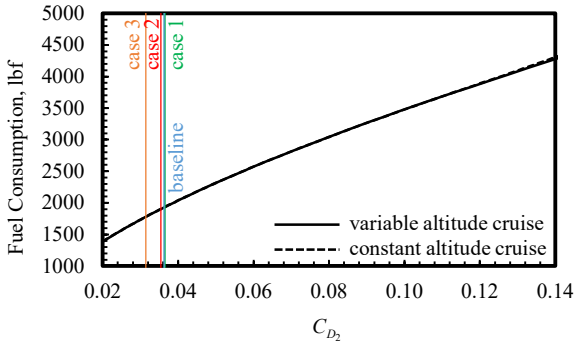


Fig. 7 Variation in overall fuel consumption over the optimized trajectory for Ikhana with respect to C_{D_2}.

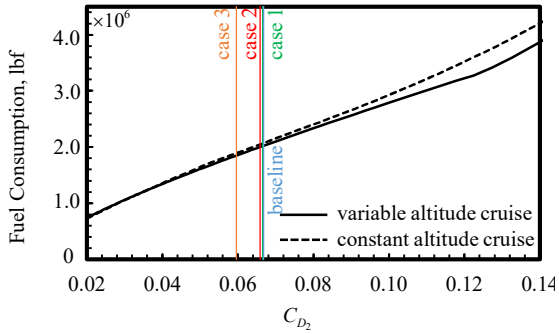


Fig. 8 Variation in overall fuel consumption over the optimized trajectory for the CRM with respect to C_{D_2}.

It is important to note that the results in Figs. 3-8 and in Appendix B represent very large variations in C_{D_2}, which fall well outside the capabilities of typical morphing mechanisms, which are more appropriately represented by the range of C_{D_2} values shown for the baseline configuration and the optimum configurations for each wing-shaping case listed in Table 5. In the following subsections, we compare results for the applications of wing shaping represented by these case studies. Results are shown for the optimum variable-altitude trajectory, the constant-altitude trajectory, where the altitude has been optimized to minimize fuel consumption, and the fixed-altitude trajectory, where the altitude is prescribed to be h = 20,000 ft for Ikhana and h = 35,000 ft for the CRM.

A. Case 1: Use of Static Wing Shaping to Reduce Wing Weight

For case 1, we will consider the use of load alleviation achieved through static wing shaping to reduce the wing weight for a fixed wingspan. Using the methods described in Section III, we can compute the optimum trajectory and the resulting total fuel consumption for each of the aircraft configurations listed for case 1 in Table 5. The total fuel consumption over the optimum trajectory for Ikhana and the CRM, respectively, is shown as a function of the wing-structure weight in Figs. 9 and 10. Note that for each aircraft, there is a wing-structure weight at which the fuel consumption is minimized.

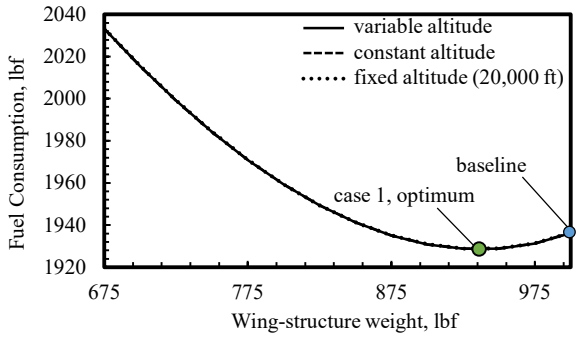


Fig. 9 Summary of the fuel consumption for each of the Ikhana configurations listed for case 1 in Table 5, operating with variable altitude, constant altitude, and with the altitude fixed at 20,000 ft.

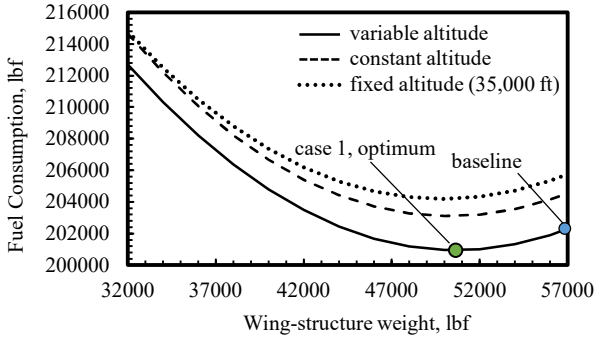


Fig. 10 Summary of the fuel consumption for each of the CRM configurations listed for case 1 in Table 5, operating with variable altitude, constant altitude, and with the altitude fixed at 35,000 ft.

For Ikhana, minimum fuel consumption occurs at W_s = 936.2 lbf, which is 7.12% less than the wing-structure weight of the baseline design. The total fuel burn for the configuration that corresponds to the optimum wing-structure weight is nearly W_f = 1929 lbf, which is 0.48% lower than the fuel consumption

for the baseline design. For the optimum configuration, load alleviation is achieved with a lift distribution characterized by B_3 = -0.0751, which, through static wing shaping, is assumed to be constant over the course of cruise and at the high-load structural design limit. A schematic of the optimum wing configuration and lift distribution for case 1 is shown in Fig. 11, alongside the baseline design.

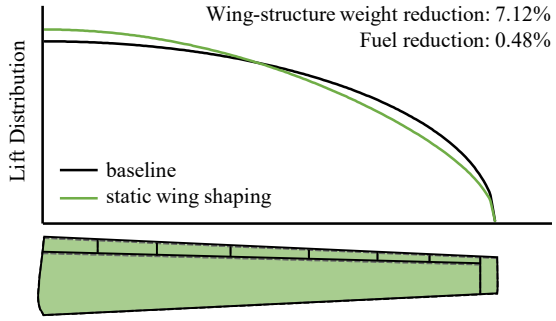


Fig. 11 Comparison of the Ikhana baseline wing and the optimum Ikhana wing configuration for case 1.

For the CRM, fuel consumption is minimized using a wing configuration having a wing-structure weight of W_s = 50,636 lbf, which is about 11.21% lower than the wing-structure weight for the baseline design. This reduction in wing-structure weight is a result of load alleviation provided by operating with a lift distribution characterized by B_3 = -0.0988. The result is that the optimum configuration has a total fuel consumption of W_f = 200,940 lbf, or about 0.97% less than the baseline design. A schematic of the optimized wing and its corresponding lift distribution is shown in Fig. 12.

It is important to note that for the three trajectory types considered here (variable altitude, constant altitude, or fixed altitude), the fuel burn reductions between the baseline design and the optimum configuration for case 1 vary by only around 0.1% for the CRM and less than 0.001% for Ikhana. However, for the CRM, Fig. 10 shows that the minimum fuel consumption for the optimum variable-altitude trajectory is just over 1% lower than the optimum constant-altitude trajectory and nearly 1.6% lower than the fixed-altitude trajectory.

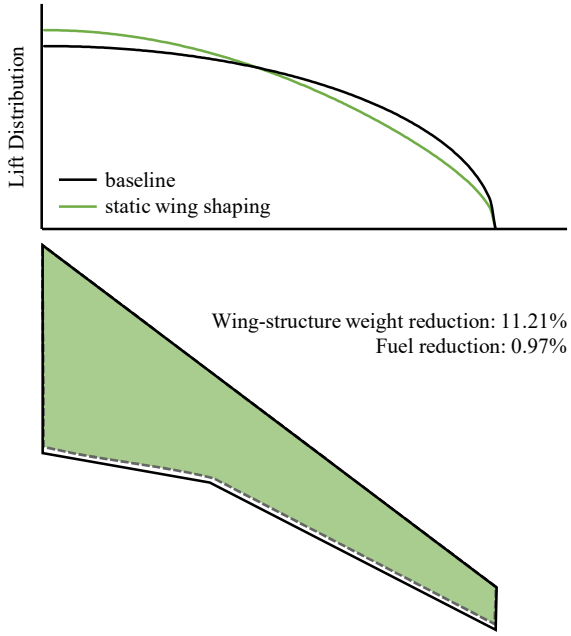


Fig. 12 Comparison of the CRM baseline wing and the optimum CRM wing configuration for case 1.

B. Case 2: Use of Static Wing Shaping to Increase Wingspan

For case 2, we consider the use of load alleviation through static wing shaping to increase the wingspan, while holding wing-structure weight constant. The total fuel burn for each of the case 2 configurations from Table 5 is shown in Fig. 13 for Ikhana and Fig. 14 for the CRM, as a function of the lift distribution, as characterized by the Fourier coefficient B_3. Recall that each of the values of B_3 shown in Figs. 13 and 14 corresponds to a wing configuration that produces a distinct lift distribution, which through load alleviation results in a different wingspan and aspect ratio for each configuration. Again, we see that for each aircraft, there is a value of B_3 that gives a minimum in fuel consumption. For each aircraft, this optimum B_3 is consistent over the three trajectory types considered here.

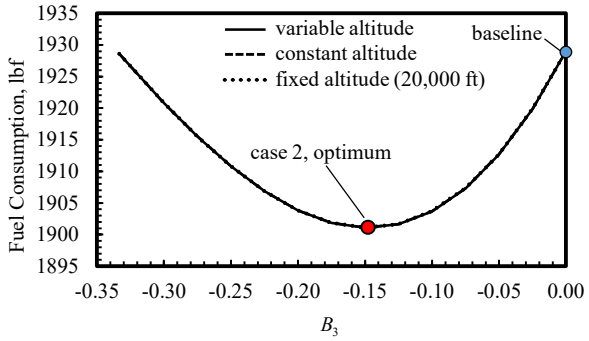


Fig. 13 Summary of the fuel consumption for each of the Ikhana configurations listed for case 2 in Table 5, operating with variable altitude, constant altitude, and with the altitude fixed at 20,000 ft.

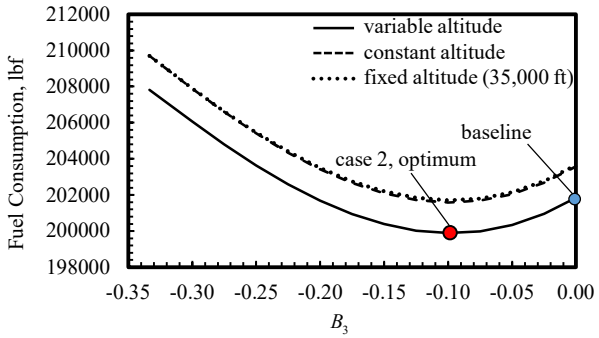


Fig. 14 Summary of the fuel consumption for each of the CRM configurations listed for case 2 in Table 5, operating with variable altitude, constant altitude, and with the altitude fixed at 35,000 ft.

For Ikhana, fuel consumption is minimized with a wing configuration having B_3 = -0.1476. This corresponds to a wingspan of b = 70.04 ft, which is 5.8% larger than that of the baseline Ikhana configuration. The result is a total fuel consumption of W_f = 1901 lbf, or 1.45% less than the baseline design. For the CRM, the optimum wing configuration operates with a lift distribution characterized by B_3 = -0.0985, which corresponds to a wingspan of nearly b = 198 ft. This is around 2.6% larger than the baseline CRM wingspan, and results a fuel consumption of W_f = 199,890 lbf, which represents a small reduction of about 0.97% over the baseline design. A schematic of the optimum Ikhana and CRM wing configurations and their corresponding lift distributions are shown in Figs. 15 and 16, respectively.

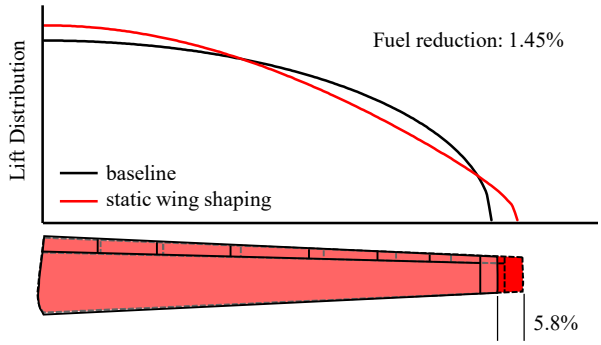


Fig. 15 Comparison of the Ikhana baseline wing and the optimum Ikhana wing configuration for case 2.

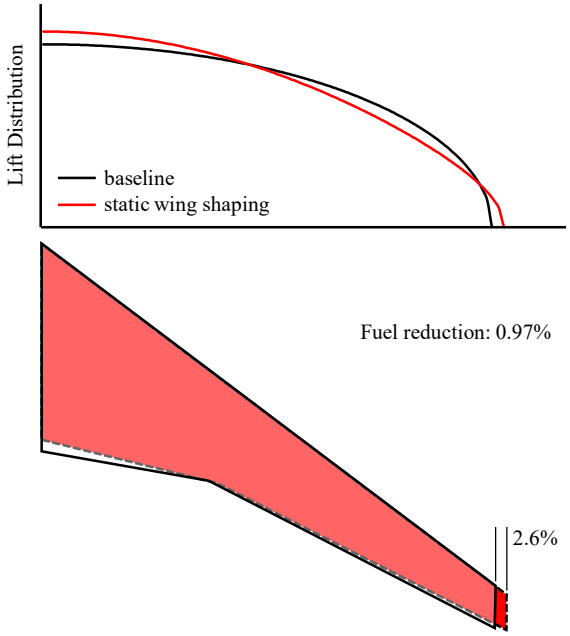


Fig. 16 Comparison of the CRM baseline wing and the optimum CRM wing configuration for case 2.

As was true for case 1, the variation in the fuel burn reductions between the baseline and optimum configuration for each of the trajectory types considered here is very small. For this case, the optimum variable altitude trajectory for the CRM results in about 0.8% less minimum fuel consumption than the constant-altitude and fixed-altitude trajectories. The difference in minimum fuel consumption between the trajectory types for Ikhana is negligible.

C. Case 3: Use of Active Wing Shaping to Increase Wingspan

Here, we consider case 3, which represents the use of active wing shaping to alleviate loads at during high-load maneuvers and to operate with high-efficiency during cruise. Recall that the configurations shown in Table 5 for this case represent different degrees of morphing capability, as characterized by B_3. For example, a configuration with B_3 = -0.2 represents a configuration with the ability to morph the wing during a high-load maneuver to achieve the lift distribution characterized by B_3 = -0.2 and operate with the elliptic lift distribution (B_3 = 0) during cruise. Therefore, the configuration with B_3 = 0 represents the baseline with no morphing ability and the configuration with B_3 = -1/3 represents maximum morphing capability. The overall fuel consumption over the optimum trajectory for each of the case 3 configurations is shown in Fig. 17 for Ikhana and Fig. 18 for the CRM.

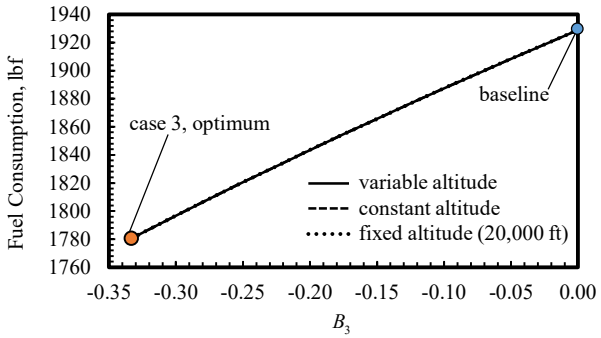


Fig. 17 Summary of the fuel consumption for each of the Ikhana configurations listed for case 3 in Table 5, operating with variable altitude, constant altitude, and with the altitude fixed at 20,000 ft.

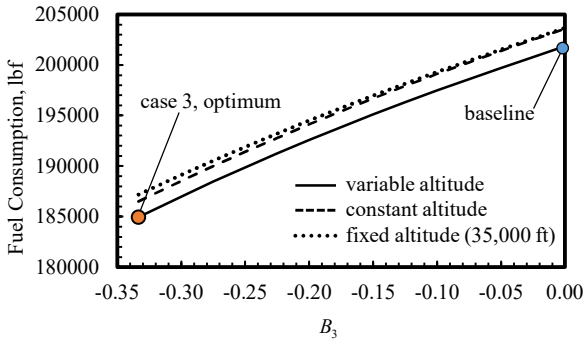


Fig. 18 Summary of the fuel consumption for each of the CRM configurations listed for case 3 in Table 5, operating with variable altitude, constant altitude, and with the altitude fixed at 35,000 ft.

As expected, minimum fuel consumption for both Ikhana and the CRM is achieved using the configuration with B_3 = -1/3, or full morphing capability. For Ikhana, this configuration has a wingspan of b = 76.48 ft and has total fuel consumption over the optimum trajectory of around W_f = 1780 lbf. These values correspond to a wingspan increase of 15.5% and a fuel burn reduction of about 7.70% over the baseline configuration. A schematic of this wing and its corresponding cruise and maneuver lift distributions is shown in Fig. 19. For the CRM, the optimum configuration has a wingspan of just over b = 215 ft, which is around 11.5% larger than the baseline configuration. The total fuel consumption for this configuration is about W_f = 184,941 lbf, or 8.38% less than the baseline configuration. A schematic of this configuration is shown in Fig. 20. Again, for both Ikhana and the CRM, the fuel savings between the baseline and optimized design is consistent across trajectory types, and the difference in the minimum fuel consumption between the trajectory types is negligible for Ikhana and around 0.8% for the CRM.

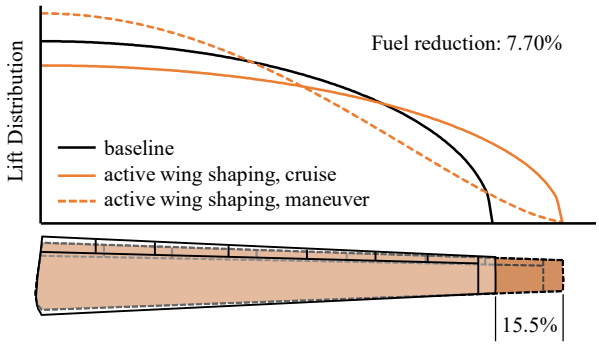


Fig. 19 Comparison of the Ikhana baseline wing and the optimum Ikhana wing configuration for case 3.

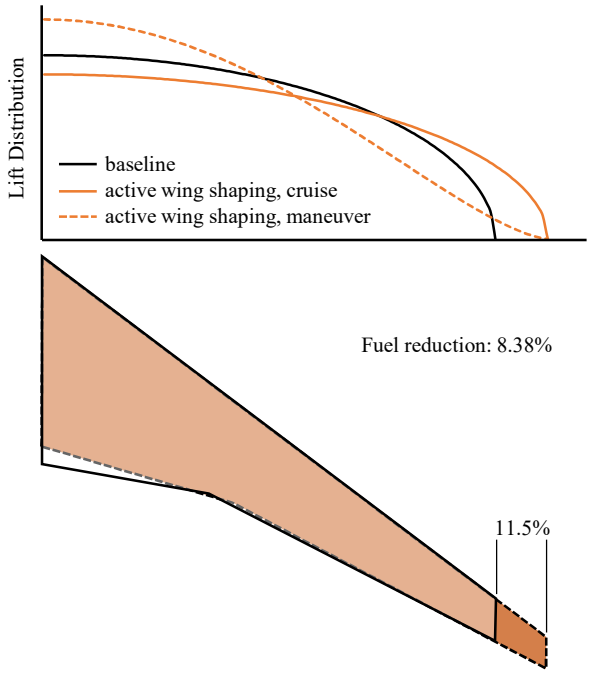


Fig. 20 Comparison of the CRM baseline wing and the optimum CRM wing configuration for case 3.

A summary of the optimization results for all three cases is given in Table 6. Figures 21 and 22 show the altitude and velocity profiles for the optimum trajectory for each of the optimum solutions given in Table 6. Results for Ikhana are shown in Fig. 21, and results for the CRM are shown in Fig. 22. Additional characteristics of these optimum trajectories are shown in Appendix B.

Table 6 Summary of results for the baseline configuration and the optimum configurations from Cases 1-3 for Ikhana and the CRM.

Baseline configuration

	Ikhana				CRM		
	Variable h	constant h	fixed h		Variable h	constant h	fixed h
W_s, lbf	1008	1008	1008	W_s, lbf	57,028	57,028	57,028
R_A	16.50	16.50	16.50	R_A	9.00	9.00	9.00
B_3	0.0	0.0	0.0	B_3	0.0	0.0	0.0
W_f, lbf	1929	1929	1929	W_f, lbf	201,850	203,586	203,657

Case 1: static wing shaping to reduce wing-structure weight

	Ikhana				CRM		
	Variable h	constant h	fixed h		Variable h	constant h	fixed h
W_s, lbf	936.2	936.2	936.2	W_s, lbf	50,636	50,291	49,907
B_3	-0.0750	-0.0751	-0.0751	B_3	-0.0988	-0.0889	-0.0818
W_f, lbf	1928	1928	1928	W_f, lbf	200,941	203,099	201,700
% ΔW_f	-0.48	-0.48	-0.48	% ΔW_f	-0.67	-0.71	-0.77

Case 2: static wing shaping to increase wingspan

	Ikhana				CRM		
	Variable h	constant h	fixed h		Variable h	constant h	fixed h
R_A	18.38	18.38	18.38	R_A	9.48	9.48	9.48
B_3	-0.1476	-0.1476	-0.1476	B_3	-0.0949	-0.0949	-0.0949
W_f, lbf	1901	1901	1901	W_f, lbf	199,890	201,582	201,670
% ΔW_f	-1.45	-1.45	-1.45	% ΔW_f	-0.97	-0.98	-0.96

Case 3: active wing shaping to increase wingspan

	Ikhana				CRM		
	Variable h	constant h	fixed h		Variable h	constant h	fixed h
R_A	21.91	21.91	21.91	R_A	11.21	11.21	11.21
B_3	-0.3333	-0.3333	-0.3333	B_3	-0.3333	-0.3333	-0.3333
W_f, lbf	1780	1780	1780	W_f, lbf	184,941	186,500	187,173
% ΔW_f	-7.70	-7.70	-7.70	% ΔW_f	-8.38	-8.39	-8.09

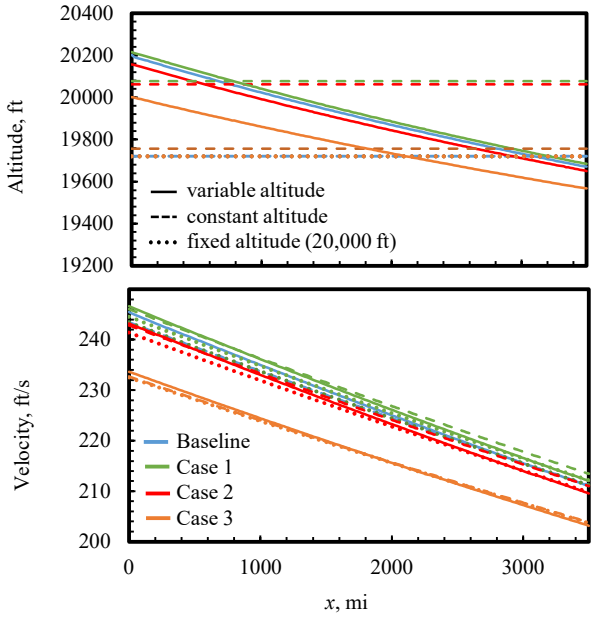


Fig. 21 Altitude and velocity profiles for the optimum trajectories for the baseline and optimum Ikhana configurations.

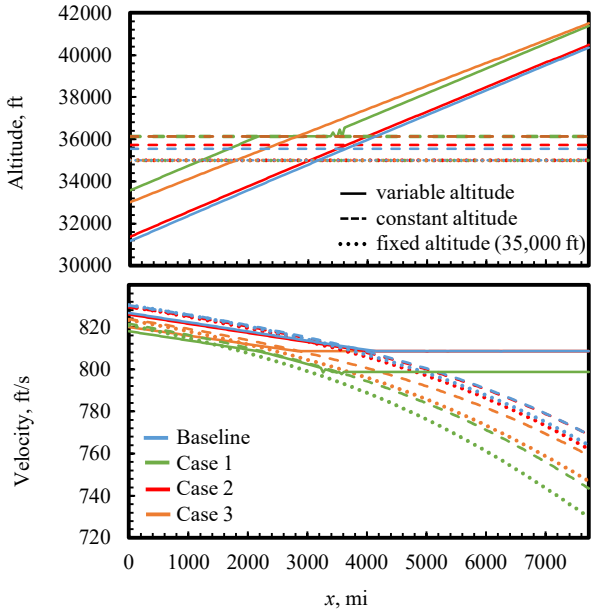


Fig. 22 Altitude and velocity profiles for the optimum trajectories for the baseline and optimum CRM configurations.

Notice that although the effects of wing shaping do not vary substantially over the three trajectory types considered here, wing shaping does impact the optimum trajectory profile. For Ikhana, static and active

wing shaping tend to result in a lower altitude optimal trajectory than the baseline design. However, for the CRM, wing shaping tends to result in a higher altitude optimal trajectory than the baseline design. For both Ikhana and the CRM, the velocity profiles for the configurations with wing shaping are lower than for the baseline design. Note that for case 1, the altitude profile includes a small region where the altitude is fixed at h = 36,131 ft. This is due to a discontinuity in the calculation of the service ceiling. Recall that in Eq. (32), the exponent m switches from 0.7 at altitudes under 36,131 ft to 1.0 at altitudes over 36,131 ft. This creates a discontinuous break in the power available, which, in turn, creates a discontinuity in the aircraft climb rate. For case 1, the aerodynamic ceiling constraint is active near 36,131 ft. As the altitude profile passes through this altitude, the climb rate abruptly switches from a value just above the service-ceiling limit of 100 ft/s to a value that violates the limit. This leads the optimizer to revert to an altitude of h = 36,131 ft until the other trajectory characteristics change such that the service-ceiling is sufficiently above 36,131 ft to overcome the discontinuity. This same effect causes the altitude profiles for the constant-altitude cruise to cluster around h = 36,131 ft. The effects of this implementation artifact on the optimum trajectory estimation can be seen in several of Figs. B1-B16 in Appendix B.

Figures B10-B16 in Appendix B provide additional insights about the effects of wing shaping on optimum trajectory. In general, morphing tends to result in a slower flight profile with lower thrust and a higher lift coefficient than the baseline design. When altitude can vary, the optimum trajectory tends to have a nearly constant Mach number, lift coefficient, drag coefficient, lift-to-drag ratio, and specific fuel consumption. When the altitude is constant or fixed, the trajectory parameters seem to vary such that the lift-to-drag ratio remains relatively constant. It is important to note that none of these parameters were assumed to be constant *a priori*.

Figure B13 shows that wing shaping tends to result in a higher lift-to-drag ratio than the baseline design over the optimum trajectory. This is because each wing-shaping case has a C_{D_2} value that is less than the baseline design, which increases the maximum lift-to-drag ratio and the lift coefficient at which it occurs. An example of this effect is illustrated in Fig. 23, which shows the drag polar and lift-to-drag ratio for the case 3 optimum configuration of the CRM with active wing shaping, compared to the drag polar and lift-to-

drag ratio for the baseline CRM. The result is that the optimum configuration for each wing-shaping case tends to operate at a higher lift coefficient than the baseline design to achieve its respective maximum *L*/*D*.

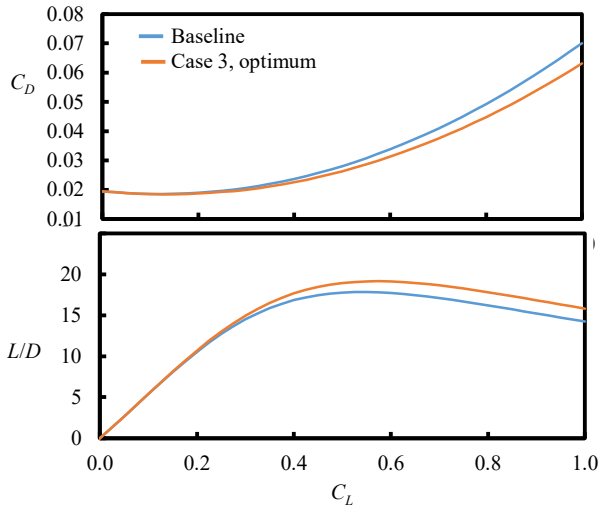


Fig. 23 Drag polar and lift-to-drag ratio for the CRM baseline design and the optimum CRM configuration for case 3 with active wing shaping.

Because each configuration tends to operate at or near its maximum lift-to-drag ratio, it is not surprising that for our analysis, the optimum trajectory features a nearly constant lift-to-drag ratio. The lift-to-drag ratio is wholly dependent on the aerodynamic characteristics of the aircraft. Therefore, if we consider a morphing retrofit in which a non-morphing configuration is optimized for maximum *L*/*D* at the same design cruise lift coefficient as maximum *L*/*D* for the morphing configuration, we expect that the morphing retrofit will result in very little, if any improvement in the maximum *L*/*D* unless the morphing retrofit is coupled with a reduction in wing weight or modification of the wing design. In other words, unless aerostructural effects, including load alleviation or aeroelasticity, are leveraged, we expect that retrofitting optimized non-morphing wings with morphing mechanisms will have little effect on the overall aircraft fuel burn over the optimal trajectory.

It is important to remember that the results in this paper are not specific to any morphing mechanism for wing shaping. Instead, in this study, we have focused on the desired aerodynamic load distribution, which we have assumed can be achieved through wing shaping using any distributed flap or morphing system.

The actuator scheduling required to achieve these load distributions depends on the mechanism used for wing shaping.

VI. Conclusions

The degree to which wing morphing, or active wing shaping, can improve aircraft performance depends on various design and operational parameters related to the aircraft configuration and its flight-path trajectory. Most of the existing literature on the effects of wing shaping on aircraft efficiency assesses the effects of wing morphing over a fixed flight trajectory or over a small number of points intended to represent critical design and off-design conditions. However, studies considering the effect that morphing may have on the optimal flight-path trajectory are relatively few. In this paper, we have presented a series of solutions that reveal important insights into the effects of active wing shaping on aircraft efficiency, represented by the aircraft fuel consumption, over the fuel-optimal flight-path trajectory.

The solutions presented in this paper were obtained using low- and multi-fidelity computational methods, as described in Sections II and III. The effects of wing shaping were modeled through changes in the lift distribution, which was characterized using the Fourier coefficient B_3 from Eq. (14), and the span efficiency factor, as described in Eq. (16). The fuel-optimal trajectory and the associated total fuel consumption were obtained using a direct optimization method described in Section III, in which the altitude and velocity are selected to minimize fuel consumption at each individual point in the discretized trajectory. At each location, the altitude is constrained such that it remains below the aircraft service ceiling, as described in Section III.B. Schematics of the optimization procedure are given in Figs. 1 and 2.

Using the methods presented in Sections II and III, optimum trajectories along with their corresponding fuel consumption were found for a range of variations of the NASA Ikhana high-endurance UAV and the NASA CRM configuration, which are described in Section IV. Figures 3-8 show how the total fuel consumption and additional characteristics of the optimum trajectory vary with changes in the Oswald efficiency factor for both Ikhana and the CRM. A series of static and active wing-shaping case-studies was selected for each aircraft to represent three practical wing-shaping applications: 1) the use of static wing

shaping to reduce wing weight, 2) the use of static wing shaping to increase wingspan, and 3) the use of active wing shaping to increase wingspan. A summary of the study cases is given in Table 5.

Static wing-shaping results for case 1 (wing shaping to decrease wing-structure weight) and 2 (wing shaping to increase wingspan) are summarized in Figs. 9-16. Figures 9 and 10 show the variation in total fuel consumption for each of the configurations in case 1. Figures 13 and 14 show the variation in total fuel consumption for each of the configurations in case 2. For each case, it has been shown that there is a solution with the range of configurations shown that minimizes fuel consumption. For case 1, the optimum Ikhana configuration results in less than 0.5% reduction in fuel consumption over the baseline design. The optimum CRM configuration results in about 0.7% fuel reduction. For case 2, the optimum configurations result in fuel reductions of about 1.5% for Ikhana and just under 1% for the CRM. A schematic of the optimum wing planforms for cases 1 and 2 are shown in Figs. 11,12,15, and 16. These results suggest that for the cases considered here, leveraging load alleviation through wing shaping to increase wingspan can result in greater reductions in fuel consumption than using load alleviation to reduce wing weight.

Active wing-shaping results for case 3 (active wing shaping to increase wingspan) are shown in Figs. 17-20 for Ikhana and the CRM. The results in these figures suggest that utilizing active wing shaping can result in reductions of up to 7.7% for Ikhana and around 8.3% for the CRM. These reductions are achieved by leveraging maneuver load alleviation at the high-load structural design limit to increase the wingspan by up to 15.5%. Here, the greatest fuel burn reductions are achieved using the maximum morphing capability possible (B_3 = -1/3). Schematics of the optimum configurations for Ikhana and the CRM are shown in Figs. 19 and 20, respectively. A summary of the optimization results for all three cases is shown in Table 6.

The altitude and velocity profiles for the optimum trajectories corresponding to the optimum configurations from cases 1-3 are shown in Figs. 21 and 22, with additional trajectory characteristics shown in Figs. B10-B16 in Appendix B. An examination of these figures reveals that the optimum trajectory tends to have nearly constant lift-to-drag ratio over its length. When altitude is allowed to vary over the flight-path trajectory, the lift coefficient, drag coefficient, Mach number, and specific fuel consumption are also

nearly constant. It has been shown that the application of wing shaping tends to result in trajectories that have lower velocity, higher lift coefficients, and higher lift-to-drag ratios than the baseline design. These differences result in fuel burn reductions of about 1% for the CRM. Fuel savings from wing shaping relative to the baseline design are consistent between trajectories with variable altitude along their length, trajectories with constant altitude that has been optimized to minimize fuel consumption, and trajectories with fixed altitude.

Taken together, the results in this paper suggest that static wing shaping or active wing shaping achieved through morphing mechanisms can have a substantial effect on aircraft efficiency and the fuel-optimal trajectory. It is important to remember that the results shown in this paper are intended as reference solutions to inform ongoing research on morphing mechanisms for wing shaping and to provide insight for conceptual design phases. Therefore, some effects that may be important in later design phases have not been considered, including aeroelasticity and 3-D transonic effects. When designing a wing with any specific morphing mechanism, additional methods may be required to assess the full effects of the mechanism on the aircraft performance. Nevertheless, the results presented in this paper provide valuable insight into the ways in which wing shaping can affect the overall performance and optimal flight-path trajectory of aircraft.

Appendix A: Example Structural Properties of the CRM

The example structural model for the CRM used in this paper is derived from the wing-box model provided by the University of Michigan for the undeflected Common Research Model (uCRM), an aerostructural variant of the CRM [70]. The uCRM wingbox geometry is characterized in detail by Taylor and Hunsaker in Ref. [60]. In order to approximate the wing-structure weight using the numerical method from Ref. [56], some parameters not presented by Taylor and Hunsaker [60] are required. The additional approximations that were used to obtain these properties are detailed in this appendix.

Using the method presented by Taylor and Hunsaker in [56], the wing-structure weight can be written for wings with stress-limited designs as

$$\widetilde{W}_s(z) = 2\int_0^{b/2} \frac{\left|\widetilde{M}_b(z)\right|}{S_b(z)}dz; \quad S_b(z) = \frac{C_\sigma\left[t_{\max}(z)/c_W(z)\right]c_W(z)\,\sigma_{\max}}{\gamma}, \quad C_\sigma = \frac{2I(h_s/t_{\max})}{Ah_s^2} \tag{A1}$$

or, for deflection-limited designs, as

$$\widetilde{W}_s(z) = 2\int_0^{b/2} \frac{\left|\widetilde{M}_b(z)\right|}{S_b(z)}dz; \quad S_b(z) = \frac{C_\delta E\left[t_{\max}(z)/c_W(z)\right]^2\,\delta_{\max}}{\gamma(W/S)^2}\frac{W^2}{b^4}, \quad C_\delta = \frac{8I(h_s/t_{\max})^2}{Ah_s^2} \tag{A2}$$

where the bending-moments $\widetilde{M}_b(z)$ resulting from the lift distribution $\widetilde{L}(z)$, wing-structure weight distribution $\widetilde{W}_s(z)$, and the distribution $\widetilde{W}_n(z)$ of all non-structural components in the wing are given by

$$\widetilde{M}_b(z) = \int_{z'=z}^{b/2}[\widetilde{L}(z') - n_a\widetilde{W}_n(z') - n_a\widetilde{W}_s(z')](z'-z)dz', \quad \text{for } z \geq 0 \tag{A3}$$

For the CRM, the chord c(z), thickness-to-chord ratio $t_{max}(z)/c_W$(z), spar height ratio $h_s(z)/t_{max}$(z), and shape factors $C_\sigma(z)$ and $C_\delta(z)$ all change along the span of the wing. Therefore, an estimation of all of these values is needed as a function of the spanwise location. The wing chord and thickness-to-chord ratio distributions are given by Vassberg [59], and are shown in Table A1. The remaining distributions can be obtained using a simplified geometric approximation for the wing box, as shown in Fig. A1.

Table A1: Wing chord and thickness-to-chord ratio for the CRM.

section	$2z/b$	chord, ft	t_{max}/c_W
1	0.00	44.6818	0.1542
2	0.10	39.0425	0.1380
3	0.15	36.2230	0.1280
4	0.20	33.4028	0.1198
5	0.25	30.5830	0.1137
6	0.30	27.7632	0.1092
7	0.35	24.9430	0.1060
8	0.37	23.8151	0.1052
9	0.40	23.1073	0.1038
10	0.45	21.9276	0.1019
11	0.50	20.7479	0.1000
12	0.55	19.5682	0.0988
13	0.60	18.3881	0.0978
14	0.65	17.2084	0.0970
15	0.70	16.0287	0.0962
16	0.75	14.8490	0.0958
17	0.80	13.6689	0.0955
18	0.85	12.4892	0.0953
19	0.90	11.3095	0.0952
20	0.95	10.1298	0.0951
21	1.00	8.9501	0.0950

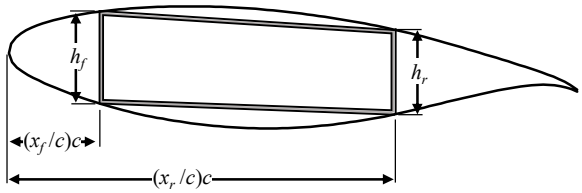


Fig. A1 Schematic of the geometric approximation of the CRM wingbox model for wing-structure weight prediction.

For the wing-box model shown in Fig. A1, the leading-edge and trailing-edge spar locations, as well as the spar thicknesses are given by Taylor and Hunsaker [60], and are summarized here in Table A1. The upper and lower edges of the wing box are assumed to have the same thickness as the upper and lower wing skins of the CRM, which are also given by Taylor and Hunsaker [60]. Using these data, the wing-box cross-sectional area, moment of inertia, and height distribution can all be found. The shape factors $C_\sigma(z)$ and $C_\delta(z)$ are computed using the definitions in Eqs. (A1) and (A2). The wing and structural parameter distributions for the CRM wingbox are summarized in Table A2. The remaining material properties γ, $\sigma_{\max}$, $\delta_{\max}$, and E, which are shown in Table A3, are assumed to be constant and are representative of 7000-series aluminum alloy.

The net weight distribution of the CRM is detailed in Ref. [60], including the weight of the engines and fuel distributed in fuel tanks that extend to around 76% semisipan. The net weight distribution for various fuel loadings for the baseline CRM is given in Fig. A2. The wing-structure weight distribution predicted by the algorithm presented by Taylor and Hunsaker in Ref. [56] is shown in Fig. A3.

Table A2: Example structural parameters for the CRM.

section	$2z/b$	h_{LE}, ft	h_{TE}, ft	I, ft^4	A, ft^2	C_σ	C_δ
0	0.0000	6.6849	4.2560	19.3153	2.8814	1.7540	2.0371
1	0.0264	6.0919	4.2471	17.7264	2.9278	1.7540	1.8564
2	0.0527	5.4160	4.2193	15.8264	2.9873	1.7551	1.6504
3	0.0791	4.6477	4.1712	13.6963	3.1081	1.7553	1.4163
4	0.1055	3.7458	4.0931	3.2198	0.8680	1.7539	1.1415
5	0.1139	3.5429	4.0815	2.7338	0.7514	1.7519	1.0797
6	0.1212	3.3449	4.0713	4.4484	1.2528	1.7501	1.0193
7	0.1295	3.1513	4.0598	5.0550	1.7436	1.7481	0.9603
8	0.1416	2.9696	4.0430	5.7388	2.0324	1.7452	0.9049
9	0.1639	2.8024	3.6818	4.4934	1.8431	1.7411	0.8540
10	0.1860	2.6489	3.3853	3.5730	1.6911	1.7330	0.8072
11	0.2079	2.5079	3.1414	2.9308	1.5858	1.7245	0.7642
12	0.2298	2.3790	2.9257	2.5787	1.5856	1.7154	0.7250
13	0.2516	2.2620	2.7303	2.2798	1.5867	1.7062	0.6893
14	0.2735	2.1552	2.5520	2.0230	1.5885	1.6962	0.6568
15	0.2953	2.0586	2.3859	1.8149	1.6147	1.6847	0.6273
16	0.3172	1.9720	2.2276	1.6331	1.6608	1.6736	0.6009
17	0.3392	1.8956	2.0707	1.4094	1.6044	1.6621	0.5777
18	0.3615	1.8503	1.9088	1.0885	1.3762	1.6492	0.5639
19	0.3839	1.8093	1.7401	0.9897	1.4027	1.6364	0.5514
20	0.4059	1.7715	1.6089	0.8340	1.3051	1.6222	0.5398
21	0.4271	1.7351	1.5624	0.7478	1.2271	1.6096	0.5287
22	0.4482	1.6979	1.5209	0.6700	1.1538	1.6025	0.5174
23	0.4693	1.6618	1.4812	0.5984	1.0815	1.5927	0.5064
24	0.4904	1.6298	1.4438	0.5321	1.0059	1.5855	0.4967
25	0.5116	1.6013	1.4058	0.4715	0.9318	1.5779	0.4880
26	0.5327	1.5741	1.3687	0.4165	0.8598	1.5706	0.4797
27	0.5538	1.5473	1.3340	0.3674	0.7932	1.5633	0.4715
28	0.5749	1.5207	1.3007	0.3218	0.7239	1.5559	0.4634
29	0.5961	1.4942	1.2666	0.2803	0.6594	1.5486	0.4553
30	0.6172	1.4674	1.2322	0.2428	0.5981	1.5367	0.4472
31	0.6383	1.4390	1.1989	0.2113	0.5459	1.5284	0.4385
32	0.6593	1.4082	1.1665	0.1834	0.4983	1.5198	0.4291
33	0.6804	1.3764	1.1355	0.1589	0.4548	1.5108	0.4194
34	0.7014	1.3448	1.1060	0.1370	0.4130	1.5015	0.4098
35	0.7224	1.3132	1.0775	0.1174	0.3734	1.4921	0.4002
36	0.7435	1.2811	1.0499	0.0996	0.3346	1.4761	0.3904
37	0.7645	1.2486	1.0203	0.0837	0.2986	1.4658	0.3805
38	0.7855	1.2152	0.9860	0.0703	0.2678	1.4548	0.3703
39	0.8066	1.1810	0.9496	0.0586	0.2402	1.4427	0.3599
40	0.8277	1.1458	0.9143	0.0484	0.2138	1.4299	0.3492
41	0.8487	1.1097	0.8803	0.0393	0.1883	1.4141	0.3382
42	0.8697	1.0727	0.8477	0.0318	0.1657	1.3989	0.3269
43	0.8907	1.0351	0.8162	0.0254	0.1445	1.3701	0.3154
44	0.9117	0.9970	0.7830	0.0205	0.1280	1.3518	0.3038
45	0.9328	0.9589	0.7470	0.0163	0.1125	1.3322	0.2922
46	0.9538	0.9214	0.7111	0.0132	0.1014	1.3113	0.2808
47	0.9748	0.8829	0.6767	0.0107	0.0913	1.2890	0.2691
48	1.0000	0.8402	0.6203	0.0107	0.0913	1.2495	0.2560

Table A3: Material properties for the low-fidelity CRM wing-structure weight estimation.

Density, slug/ft^3	5.39
Specific Weight, slug/(ft^{2}s^2)	173.6
Modulus of Elasticity, psf	15.26×10^8
Yield Strength, psf	8.77×10^6
Poisson Ratio	0.33
Shear Modulus, psf	5.74×10^8

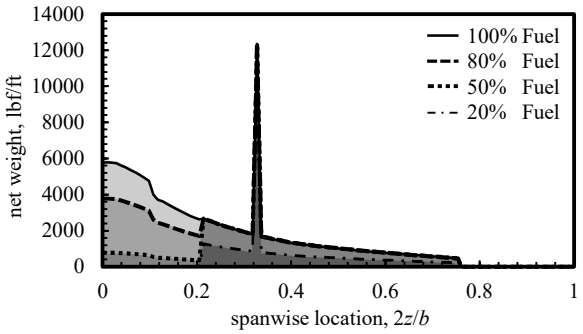


Fig. A2 Example net-weight distributions for the CRM.

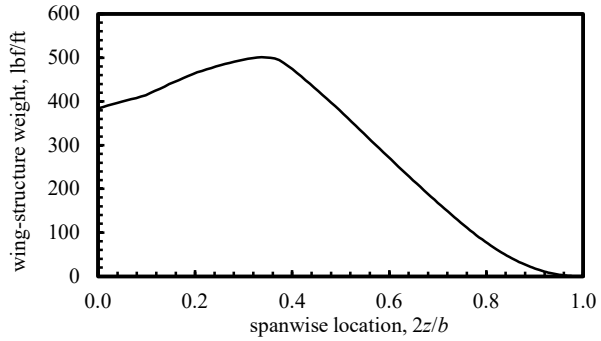
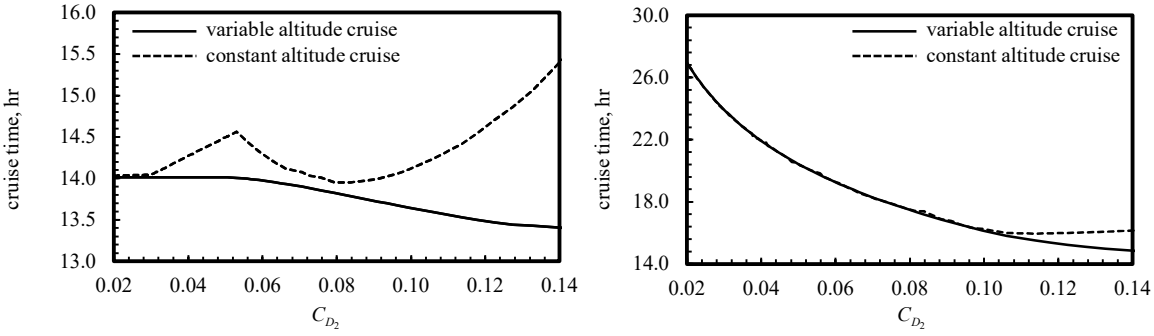


Fig. A3 Approximate wing-structure weight distribution for the CRM wing.

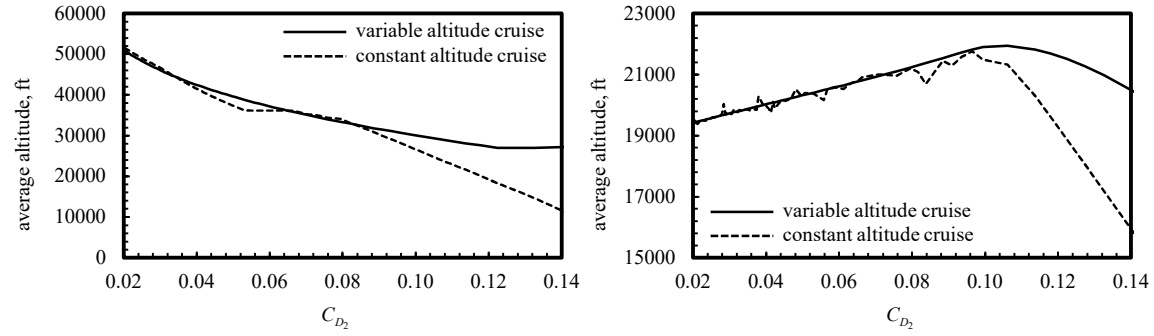
Appendix B: Trajectory Characteristics

The figures in this appendix show characteristics of the optimum trajectories for the solutions presented in section V. Figures B1-B9 show the variations in average optimum trajectory parameters for Ikhana and the CRM with respect to changes in C_{D_2}. Note that the jogs seen in the results for the CRM in Figs. B1-B6 and Fig. B9 are due to the discontinuity in the implementation of the power available model for the CRM, as described in Section V.C.



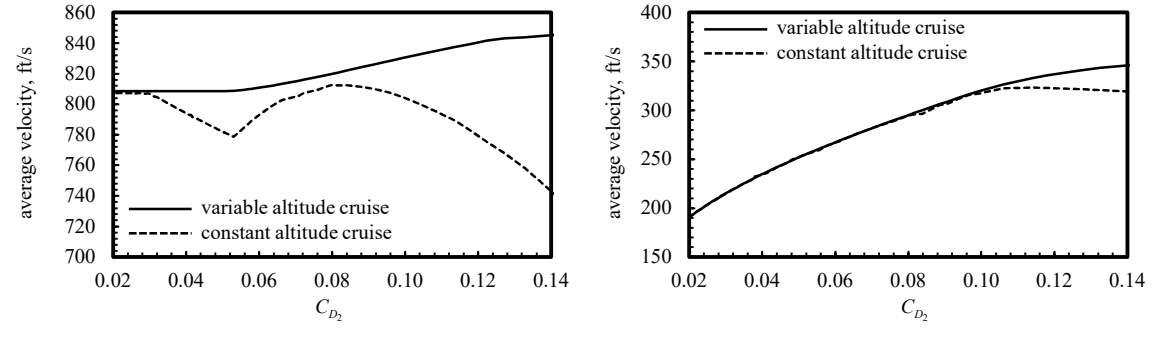
(a) (b)

Fig. B1: Variation in average cruise time over the optimized trajectory with respect to C_{D_2} for a) the CRM and b) Ikhana.



(a) (b)

Fig. B2: Variation in average altitude over the optimized trajectory with respect to C_{D_2} for a) the CRM and b) Ikhana.



(a) (b)

Fig. B3: Variation in average velocity over the optimized trajectory with respect to C_{D_2} for a) the CRM and b) Ikhana.

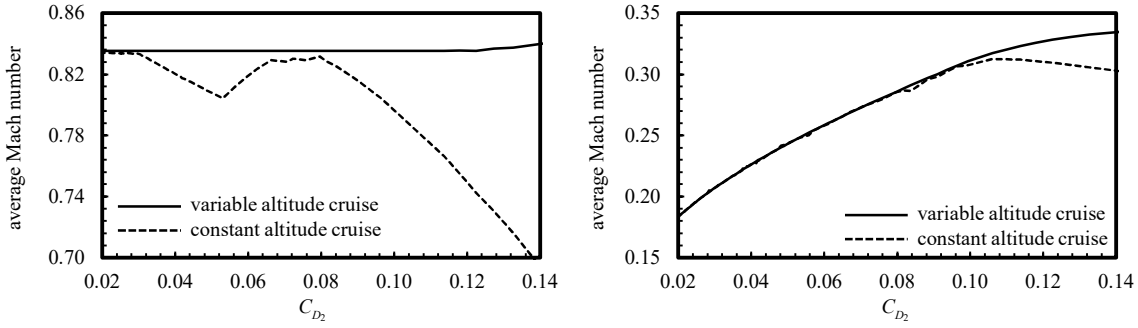


Fig. B4: Variation in average Mach number over the optimized trajectory with respect to C_{D_2} for a) the CRM and b) Ikhana.

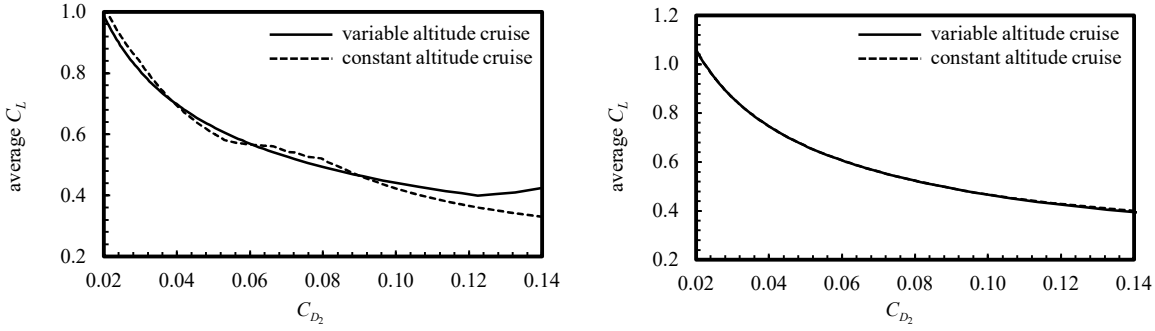


Fig. B5: Variation in average lift coefficient over the optimized trajectory with respect to C_{D_2} for a) the CRM and b) Ikhana.

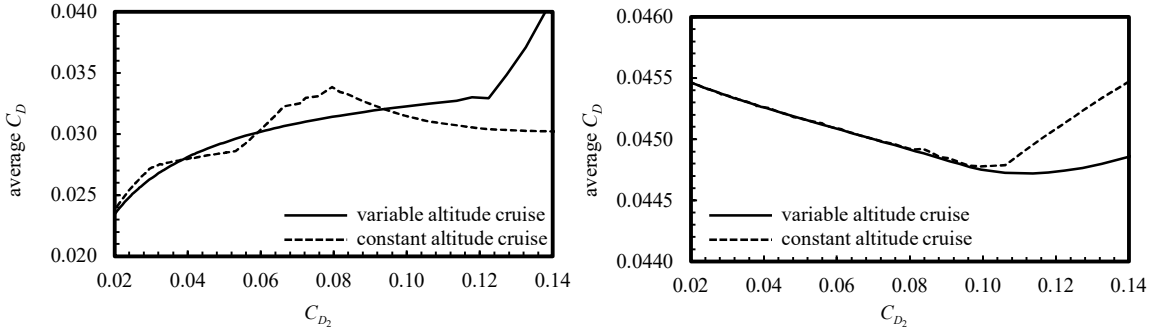


Fig. B6: Variation in average drag coefficient over the optimized trajectory with respect to C_{D_2} for a) the CRM and b) Ikhana.

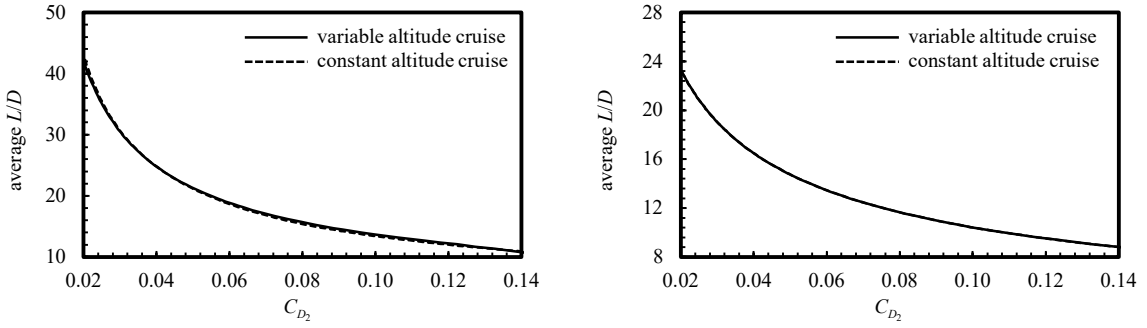


Fig. B7: Variation in average lift-to-drag ratio over the optimized trajectory with respect to C_{D_2} for a) the CRM and b) Ikhana.

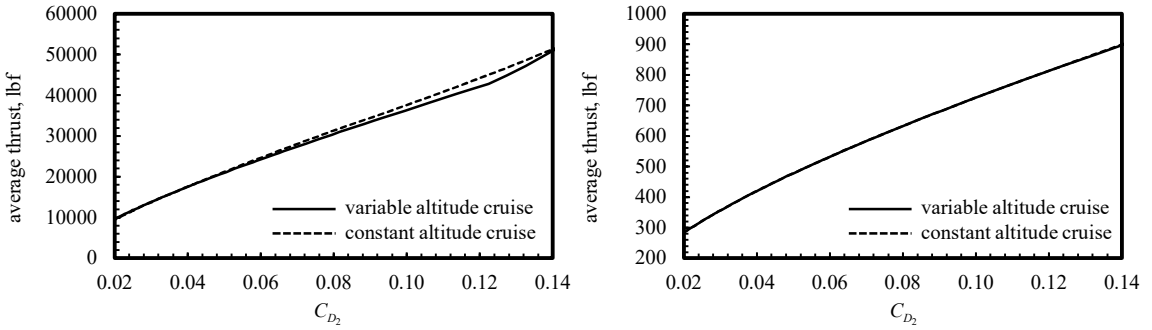


Fig. B8: Variation in average thrust over the optimized trajectory with respect to C_{D_2} for a) the CRM and b) Ikhana.

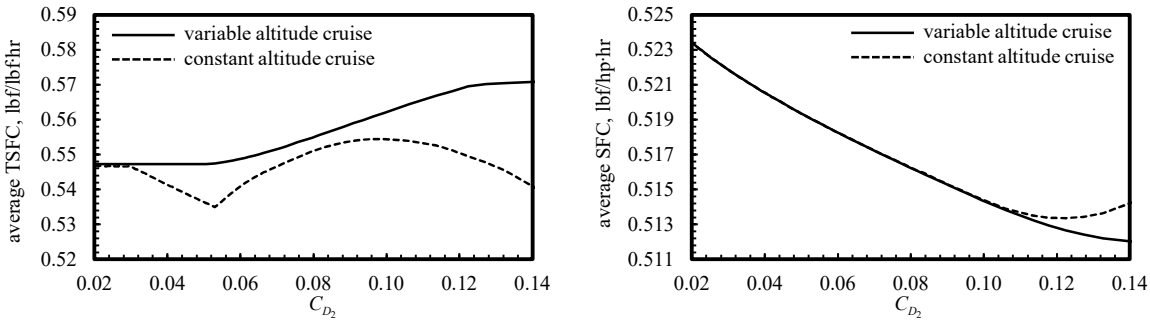


Fig. B9: Variation in average thrust-specific fuel consumption (a) or power-specific fuel consumption (b) over the optimized trajectory with respect to C_{D_2} for a) the CRM and b) Ikhana.

Characteristics of the optimum trajectories for each of the wing-shaping configurations given in Table 5 is shown in Figs. B10-B16 for Ikhana and the CRM, alongside the respective baseline design. Results are shown for optimum trajectories with variable altitude cruise, constant altitude cruise, and fixed altitude cruise. Again, note that the jogs shown for the CRM case 1 in Figs. 10-12 and Fig. 15 are due to a discontinuity in the implementation of the power available model for the CRM, as described in Section V.C.

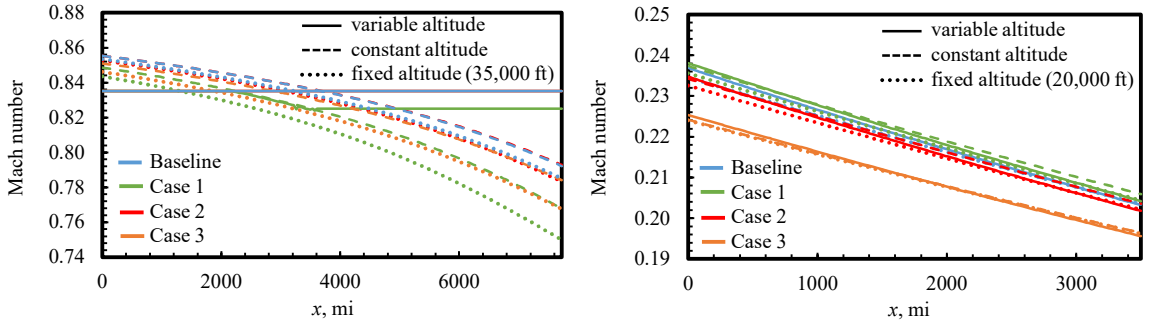


Fig. B10: Variation in Mach number over the optimum trajectories for each of the wing-shaping configurations from Table 5, along with the optimum trajectory for the baseline configuration for a) the CRM and b) Ikhana.

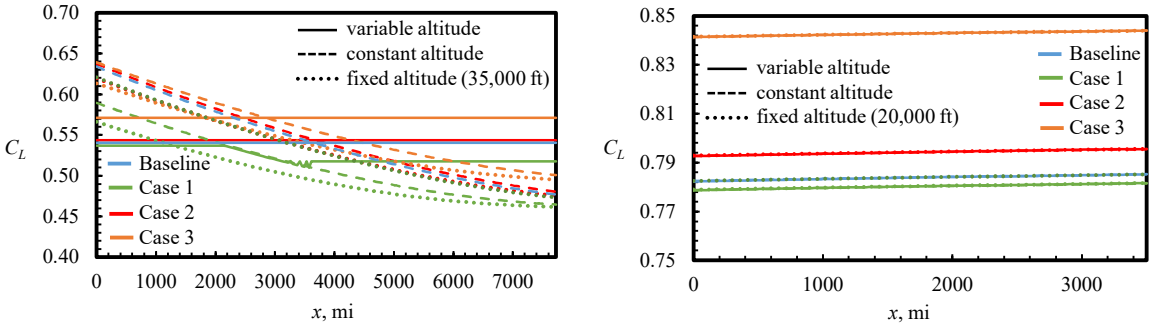
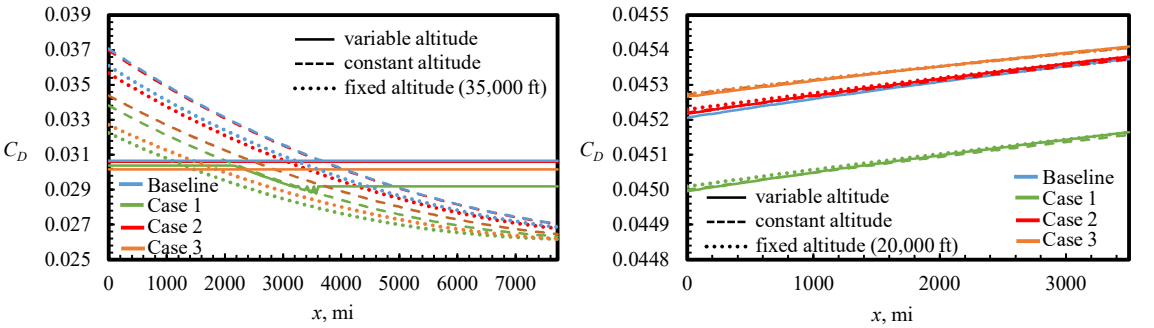
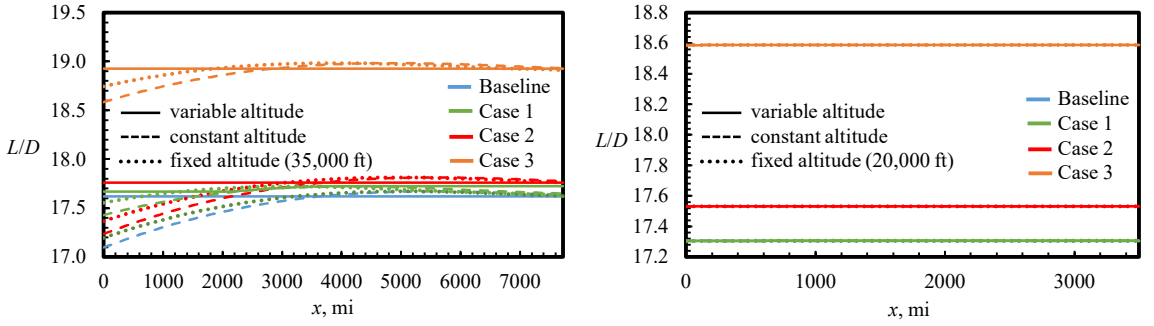


Fig. B11: Variation in lift coefficient over the optimum trajectories for each of the wing-shaping configurations from Table 5, along with the optimum trajectory for the baseline configuration for a) the CRM and b) Ikhana.



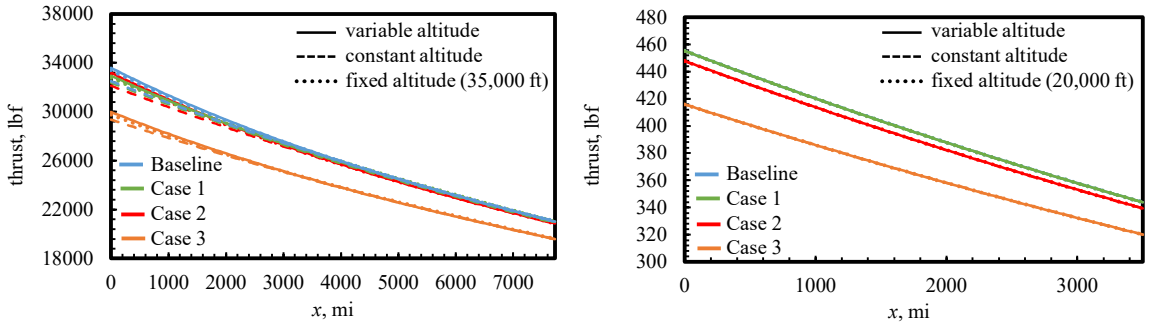
(a) (b)

Fig. B12: Variation in drag coefficient over the optimum trajectories for each of the wing-shaping configurations from Table 5, along with the optimum trajectory for the baseline configuration for a) the CRM and b) Ikhana.



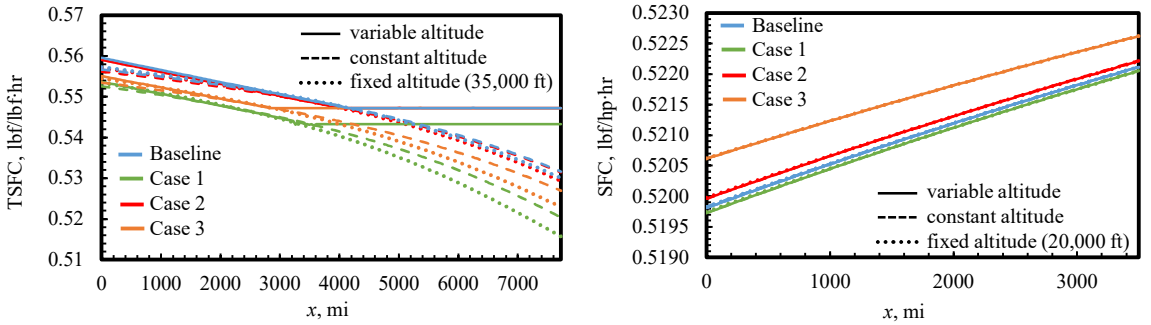
(a) (b)

Fig. B13: Variation in lift-to-drag ratio over the optimum trajectories for each of the wing-shaping configurations from Table 5, along with the optimum trajectory for the baseline configuration for a) the CRM and b) Ikhana.



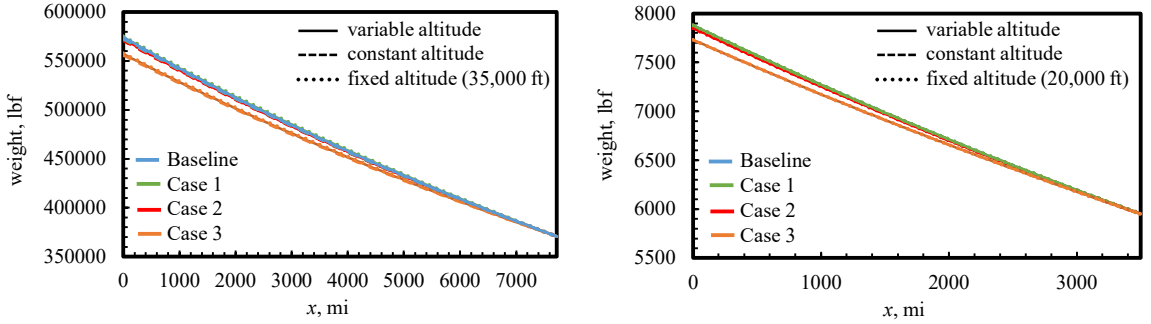
(a) (b)

Fig. B14: Variation in thrust over the optimum trajectories for each of the wing-shaping configurations from Table 5, along with the optimum trajectory for the baseline configuration for a) the CRM and b) Ikhana.



(a) (b)

Fig. B15: Variation in thrust-specific fuel consumption (a) or power-specific fuel consumption (b) over the optimum trajectories for each of the wing-shaping configurations from Table 5, along with the optimum trajectory for the baseline configuration for a) the CRM and b) Ikhana.



(a) (b)

Fig. B16: Variation in aircraft weight over the optimum trajectories for each of the wing-shaping configurations from Table 5, along with the optimum trajectory for the baseline configuration for a) the CRM and b) Ikhana.

Acknowledgements

This material is partially based upon work supported by NASA under Grant No. 80NSSC18K1696 issued by the Aeronautics Research Mission Directorate through the 2018 NASA Fellowship Activity with Nhan Nguyen as the NASA Technical Advisor.

References

[1] Perry III, B., Cole, S. R., and Miller, G. D., "A Summary of the Active Flexible Wing Program," *Journal of Aircraft*, Vol. 32, No. 1, February 1992, pp. 10-15. (doi:10.2514/3.46677)

[2] Bonnema, K. L. and Lokos, W. A., "AFTI/F-111 Mission Adaptive Wing Flight Test Instrumentation Overview," *Instrumentation in the Aerospace Industry, Proceedings of the ISA Aerospace Instrumentation Symposium*, Vol. 35, pp. 809-840.

[3] Nguyen, N., Elastically Shaped Future Air Vehicle Concept, NASA Innovation Fund Award 2010 Report, October 2010.

[4] Joo, J., Marks, C. Zientarski, L., and Culler, A., "Variable Camber compliant Wing-Design," AIAA Paper 2015-1050, 23rd AIAA/AHS Adaptive Structures Conference, Kissimmee, FL, 5-9 January 2015.

[5] Weisshaar, T. A., "Morphing Aircraft Technology – New Shapes for Aircraft Design," RTO-MP-AVT-141, October 2006.

[6] Pendleton, E., Griffin, K.E., Kehoe, M. W., and Perry III, B. "A Flight Research Program for Active Aeroelastic Wing Technology, 37th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Salt Lake City, UT, 15-17 April 1996. (doi:10.2514/6.1996-1574)

[7] Wilson, T., Kirk, J., Hobday, J., and Castrichini, A., "Small Scale Flying Demonstration of Semi Aeroelastic Hinged Wing Tips," International Forum on Aeroelasticity and Structural Dynamics, Savannah GA, 9-13 June 2019.

[8] Pecora, R., "Morphing wing flaps for large civil aircraft: Evolution of a smart technology across the Clean Sky program," *Chinese Journal of Aeronautics*, Vol. 34, No. 7, July 2021, pp. 13-28. (doi:10.1016/j.cja.2020.08.004)

[9] Hetrick, J. Osborn, R., Kota, S. Flick, P., and Paul, D., AIAA paper 2007-1709, "Flight Testing of Mission Adaptive Compliant Wing," 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Honolulu, HI, 23-26 April 2007. (doi:10.2514/6.2007-1709)

[10] Woods B. K. S. and Friswell, M. I., "Preliminary Investigation of a Fishbone Active Camber Concept," *Proceedings of the ASME Conference on Smart Materials, Adaptive Structures, and Intelligent Systems*, 2012, pp. 555-563. (doi:10.1017/CBO9781107415324.004)

[11] Phillips, W. F., "New Twist on an Old Wing Theory," *Aerospace America*, January 2005, pp. 27-30.

[12] Vos, R., Gurdal, Z., and Abdalla, M., "Mechanism for Warp-Controlled Twist of a Morphing Wing," *Journal of Aircraft*, Vol. 47, No. 2, March-April 2010, pp. 450-457. (doi:10.2514/1.39328)

[13] Dale, A. S., Cooper, J. E., and Mosquera, A., "Adaptive Camber-Morphing Wing Using 0-v Honeycomb," AIAA paper 2013-1510, 54th

[14] Prandtl, L., "Tragflügel Theorie," *Nachricten von der Gesellschaft der Wissenschaften zu Göttingen*, Geschäeftliche Mitteilungen, Klasse, 1918, pp. 451–477.

[15] Prandtl, L., "Applications of Modern Hydrodynamics to Aeronautics," NACA TR-116, June 1921.

[16] Lebofsky, S., Ting, E., Nguyen, N. T., and Trinh, K. V., "Aeroelastic Modeling and Drag Optimization of Flexible Wing Aircraft with Variable Camber Continuous Trailing Edge Flap," AIAA paper 2014-2443, 32nd AIAA Applied Aerodynamics Conference, Atlanta, GA, 16-20 June 2014, (doi:10.2514/6.2014-2443)

[17] Lebofsky, S. Ting, E., and Nguyen, N. T., "Multidisciplinary Drag Optimization of Reduced Stiffness Flexible Wing Aircraft with Variable Camber Continuous Trailing Edge Flap," AIAA paper 2015-1408, 56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Kissimmee, FL, 5-9 January 2015. (doi:10.2514/6.2015-1408)

[18] Ting, E., Chaparro, D., and Nguyen, N. T., "Aero-Structural Optimization of Variable Camber Continuous Trailing Edge Flap Configurations Using Transonic and Viscous Potential Flow Method," AIAA paper 2017-4420, 35th AIAA Applied Aerodynamics Conference, Denver, CO, 5-9 June 2017. (doi:10.2514/6.2017-4420)

[19] Chaparro, D., Fujiwara, G. E., Ting, E., and Nguyen, N. T., "Transonic and Viscous Potential Flow Method Applied to Flexible Wing Transport Aircraft," AIAA paper 2017-4221, 35th AIAA Applied Aerodynamics Conference, Denver, CO, 5-9 June 2017. (doi:10.2514/6.2017-4221)

[20] Ippolito, C., Nguyen, N. T., Totah, J., Trinh, K. V., and Ting, E., "Initial Assessment of a Variable-Camber continuous Trailing-Edge Flap System on a Rigid Wing for Drag Reduction in Subsonic Cruise," AIAA InfoTech@Aerospace Conference, Boston, MA, 19-22 August 2013.

[21] Nguyen, N. T., Ting, E., Chaparro, D., Drew, M. C., and Swei, S. S. "Multi-Objective Flight control for Drag Minimization and Load Alleviation of High-Aspect Ratio Flexible Wing Aircraft," AIAA paper 2017-1589, 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Grapevine, TX, 9-13 January 2017. (doi:10.2514/6.2017-1589)

[22] Rodriguez, D. L., Aftosmis, M. J., Nemec, M., and Anderson, G. R., "Optimization of Flexible Wings with Distributed Flaps at Off-Design Conditions," *Journal of Aircraft*, Vol. 53, No. 6, November 2016, pp. 1731-1745. (doi:10.2514/ 1.C033535)

[23] Fujiwara, G. E., Nguyen, N. T., Livne, E., and Bragg, M. B., "Aerostructural Design Optimization of a Flexible Wing Aircraft with Continuous Morphing Trailing Edge," AIAA paper 2018-3571, 2018 Multidisciplinary Analysis and Optimization Conference, Atlanta, GA, 25-29 June 2018. (doi:10.2514/6.2018-3571)

[24] Stanford, B. K., "Static and Dynamic Aeroelastic Tailoring with Variable Camber Control," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 11, November 2016, pp. 2522-2534. (doi:10.2514/1.G000413)

[25] Kier, T. M., Leitner, M., Suelozgen, O., and Pusch, M., "An Integrated Flexible Aircraft Model for Optimization of Lift Distributions," AIAA paper 2019-2039, AIAA SciTech 2019 Forum, San Diego, CA, 7-11 January 2019. (doi:10.2514/6.2019-2039)

[26] Kolonay, R. M., and Eastep, F. E., "Optimal Scheduling of Control Surfaces on Flexible Wings to Reduce Induced Drag," *Journal of Aircraft*, Vol. 43, No. 6, November 2006, pp. 1655-1661. (doi:10.2514/1.14604)

[27] Hammerton, J. R., Su, W., Zhu, G., and Swei, S. S., "Optimum distributed wing shaping and control loads for highly flexible aircraft," *Aerospace Science and Technology*, Vol. 79, August 2018, pp. 255-265. (doi:10.1016/j.ast.2018.05.045)

[28] Su. W., Swei, S. S., and Zhu, G. G., "Optimum Wing Shape of Highly Flexible Morphing Aircraft for Improved Flight Performance," *Journal of Aircraft*, Vol. 53, no. 5, September 2016, pp. 1305-1316. (doi:10.2514/1.C033490)

[29] Prandtl, L., "Über Tragflügel kleinsten induzierten Widerstandes," Zeitschrift für Flugtechnik und Motorluftschiffahrt, Vol. 24, No. 11, 1933, pp. 305–306.

[30] Hunsaker, D. F., "Ludwig Prandtl's 1933 Paper Concerning Wings for Minimum Induced Drag, Translation and Commentary," AIAA SciTech 2020 Forum, Orlando, Florida, 6-10 January 2020.

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

[32] Gopalarathnam, A., and Norris, R. K., "Ideal Lift Distributions and Flap Angles for Adaptive Wings," *Journal of Aircraft*, Vol. 46, No. 2, 2009, pp. 562–571. (doi:10.2514/1.38713)

[33] Verstraetan, J., G., and Slingerland, R., "Drag Characteristics for Optimally Span-Loaded Planar, Wingletted, and C Wings," *Journal of Aircraft*, Vol. 46, No. 3, May-June 2009, pp. 962-971. (doi:10.2515/1.39426)

[34] Ranjan, P., "Computational Analysis of Planar Wings Designed for Optimum Span-Load," *MS Thesis*, University of Illinois at Urbana-Champaign, 2016.

[35] Pate, D. J., and German, B. J., "Lift Distributions for Minimum Induced Drag with Generalized Bending Moment Constraints," *Journal of Aircraft*, Vol. 50, 2013, pp. 936–946. (doi:10.2514/1.C032074)

[36] DeYoung, J., "Minimization Theory of Induced Drag Subject to Constraint Conditions," NASA CR-3140, June 1979.

[37] Jones, R. T., and Lasinski, T. A., "Effect of Winglets on the Induced Drag of Ideal Wing Shapes," NASA TM-81230, Sept. 1980.

[38] Klein, A., and Viswanathan, S. P., "Minimum Induced Drag of Wings with Given Lift and Root-Bending Moment," *Zeitschrift fur Angewandte Mathematik und Physik*, Vol. 24, 1973, pp. 886–892.

[39] Klein, A., and Viswanathan, S. P., "Approximate Solution for Minimum Induced Drag of Wings with Given Structural Weight," *Journal of Aircraft*, Vol. 12, No. 2, 1975, pp. 124–126. (doi:10.2514/3.44425)

[40] Löbert, G., "Spanwise Lift Distribution for Forward- and Aft-Swept Wings in Comparison to the Optimum Distribution Form," *Journal of Aircraft*, Vol. 18, No. 6, 1981, pp. 496-498. (doi:10.2514/3.44717)

[41] McGeer, T., "Wing Design for Minimum Drag with Practical Constraints," *Journal of Aircraft*, Vol. 21, 1984, pp. 879–886. (doi:10.2514/3.45058)

[42] 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, pp. 431-441. (doi:10.2514/1.C035027)

[43] Phillips, W. F., Hunsaker, D. F., and Taylor, J. D., "Minimising Induced Drag with Weight Distribution, Lift Distribution, Wingspan, and Wing-Structure Weight," *Aeronautical Journal*, Vol. 124, no. 1278, August 2020, pp. 1208-1235. (doi:10.1017/aer.2020.24).

[44] Taylor, J. D. and Hunsaker D. F., "Comparison of Theoretical and Multifidelity Optimum Aerostructural Solutions for Wing Design," *Journal of Aircraft*, Vol. 59, No. 1, January 2022, pp. 103-116. (doi:10.2514/1.C036374)

[45] Hunsaker, D. F., Phillips, W. F., and Joo, J. J., "Aerodynamic Shape Optimization of Morphing Wings at Multiple Flight Conditions," AIAA 2017-1420, 55th AIAA Aerospace Sciences Meeting, Grapevine, TX, 9-13 January 2017. (doi:10.2514/6.2017-1420)

[46] Burdette, D. A., Kenway, G. K. W., Lyu, Z., and Martins, J. R. R. A., "Aerostructural Design Optimization of a an Adaptive Morphing Trailing Edge Wing," AIAA 2015-1129, 56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Kissimmee, FL, 5-9 January 2015. (doi:10.2514/6.2015-1129)

[47] Burdette, D. A. and Martins, J. R. R. A., "Impact of Morphing Trailing Edges on Mission Performance for the Common Research Model," *Journal of Aircraft*, Vol. 56, No. 1, January 2019, pp. 369-384. (doi:10.2514/1.C034967)

[48] Lyu, Z. and Martins, J. R. R. A., "Aerodynamic Shape Optimization of an Adaptive Morphing Trailing-Edge Wing," *Journal of Aircraft*, Vol. 52, No. 6, November-December 2015, pp.1951- 1970. (doi:10.2514/1.C033116)

[49] Nguyen, N. T., Xiong, J., and Sager, J., "Fuel-Optimal Trajectory Optimization of Mach 0.745 Transonic Truss-Braced Wing with Variable Camber Continuous Trailing Edge Flap," AIAA paper 2021-2575, AIAA Aviation 2021 Forum, Virtual, 2-6 August 2021. (doi:10.2514/6.2021-2575)

[50] Fasel, U., Tiso, P., Keidel, D., and Ermanni, E., "Concurrent Design and Flight Mission Optimization of Morphing Airborne Wind Energy Wings," *AIAA Journal*, Vol. 59, No. 4, April 2021, pp. 1254-1268. (doi:10.2514/1.J059621)

[51] Jasa, J. P., Hwang, J. T., and Martins, J. R. R. A., "Design and Trajectory Optimization of a Morphing Wing Aircraft," AIAA paper 2018-1382, 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Kissimmee, FL, 8-12 January 2018. (doi:10.2514/6.2018-1382)

[52] Rudnick-Cohen, E. S., Hodson, J. D., Reich, G. W., Pankonien, A. M., and Beran, P. S., "Design and Trajectory Planning Optimization of a Morphing Airfoil for 3-D Flight Maneuvers," AIAA paper 2021-0890, AIAA SciTech 2021 Forum, Virtual, 11-15 & 19-21 January 2021. (doi:10.2514/6.2021-0890)

[53] Rudnick-Cohen, E. S., Hodson, J. D., Reich, G. W., Pankonien, A. M., and Beran, P. S., " Robust Optimal Design and Control of a Morphing Unmanned Aerial Vehicle (UAV) Airfoil for a Range of Flight Maneuvers," AIAA paper 2020-3112, AIAA Aviation 2020 Forum, Virtual, 15-19 June 2020. (doi:10.2514/6.2020-3112)

[54] Kraft, D. "A software package for sequential quadratic programming," DLR German Aerospace Center – Institute for Flight Mechanics, DFVLR-FB 88-28, Koln, Germany, 1988.

[55] *U.S. Standard Atmosphere, 1976*, National Oceanic and Atmospheric Administration, National Aeronautics and Space Administration, and United States Air Force, Washington, D. C., 1976.

[56] Taylor, J. D. and Hunsaker, D. F., "Low-fidelity method for rapid aerostructural optimization and design-space exploration of planar wings," *Aeronautical Journal*, Vol. 125, No. 1289, July 2021, pp. 1209-1230. (doi:10.1017/aer.2021.14)

[57] Merlin, P. W., "'Don't Fear the Reaper,' Ikhana: Unmanned Aircraft System Western States Fire Missions," *Monographs in Aerospace History*, Vol 44, National Aeronautics and Space Administration, NASA SP-2009-4544, 2009, pp. 1-18.

[58] Honeywell Aerospace, "TPE331-10 Turboprop Engine," N61-1491-000-000, August 2016.

[59] Vassberg, J. C., DeHaan, M. A., Rivers, S. M., and Wahls, R. A., "Development of a Common Research Model for Applied CFD Validation Studies," AIAA 2008-6919, 26th AIAA Applied Aerodynamics Conference, Honolulu, HI, 18-21 August 2008. (doi:10.2514/6.2008-6919)

[60] Taylor, J. D. and Hunsaker, D. F., "Characterization of the Common Research Model Wing for Low-Fidelity Aerostructural Analysis," AIAA 2021-1591, AIAA Scitech 2021 Forum, Virtual, 11-15 & 19-21 January 2021. (doi:10.2514/6.2021-1591)

[61] "777-200/300 Airplane Characteristics for Airport Planning", D6-58329, Boeing Commercial Airplanes, July, 1998.

[62] Vassberg, J. C., Tinoco, E. N., Mani, M., Rider, B., Zickuhr, T., Levy, D. W., Brodersen, O. P., Eisfeld, B., Crippa, S., Wahls, R. A., Morrison, J. H., Mavriplis, D. J., and Murayama, M., "Summary of the Fourth AIAA

Computational Fluid Dynamics Drag Prediction Workshop," *Journal of Aircraft*, Vol., 51, No. 4, July-August 2014. (doi:10.2514/1.C032418)

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

[64] Goates, C. D. and Hunsaker, D. F., "Practical Implementation of a General Numerical Lifting-Line Method," AIAA 2021-0118, AIAA Scitech 2021 Forum, Virtual, 11-15 & 19-21 January 2021. (doi:10.2514/6.2021-0118).

[65] Fujiwara, G. E. C., Chaparro, D., and Nguyen, N., "An Integral Boundary Layer Direct Method Applied to 2D Transonic Small-Disturbance Equations," AIAA 2016-3568, 34th AIAA Applied Aerodynamics Conference, Washington, D. C., 13-17 June 2016. (doi:10.2514/6.2016-3568)

[66] Eshelby, M. E., "Cruising Performance," *Aircraft Performance: Theory and Practice*, AIAA Education Series, American Institute of Aeronautics and Astronautics, Reston, VA, 2000, p. 66.

[67] de Servi, C., Azzini, L., Pini, M., and Colonna, P., "Exploratory assessment of a combined-cycle engine concept for aircraft propulsion," 1st Global Power and Propulsion Forum, Zurich, Switzerland,16-18 January 2017.

[68] Daidzic, N. E., "Estimation of Performance Airspeeds for High-Bypass Turbofans Equipped Transport-Category Airplanes," *Journal of Aviation Technology and Engineering*, Vol. 5, No. 2, 2016, pp. 27-50.

[69] Taylor, J. D., and Hunsaker, D. F., "Minimum induced Drag for Tapered Wings Including Structural Constraints," *Journal of Aircraft*, Vol. 57, No. 4, July 2020, pp. 782-786. (doi:10.2514/1.C035757)

[70] Brooks, T. R., Kenway, G. K. W., and Martins, J. R. R. A., "Benchmark Aerostrucural Models for the Study of Transonic Aircraft Wings," *AIAA Journal*, Vol. 56, No. 7, July 2018, pp. 2840-2855. (doi:10.2514/1.J056603)

CHAPTER 6

CONCLUSION

The reference solutions in this dissertation suggest that tailoring the lift distribution on an aircraft wing through static or active wing shaping can be used to leverage tradeoffs between the lift distribution, the wingspan, and the wing weight to achieve substantial efficiency benefits over the course of a flight-path trajectory. Predicting these efficiency benefits requires a multidisciplinary approach that considers, among other things, the coupling between aerodynamics, structures, flight mechanics, and control. There are a variety of existing multidisciplinary design and optimization tools that have been shown to provide accurate predictions for specific design scenarios, but most rely on linking computationally expensive black-box tools from which relational information about the coupling between design and operational parameters is difficult to obtain. This dissertation presents an alternative approach using analytic and low-fidelity methods to obtain relational information about the effects of load alleviation through wing shaping on the efficiency and optimum flight-path trajectory of aircraft wings through closed form mathematical relationships and rapid design-space exploration and optimization. The solutions presented in this dissertation are meant to serve as reference solutions in efforts to predict the efficiency benefits of wing shaping and to support and inform ongoing research on adaptive wing morphing for aircraft performance.

The papers presented in this dissertation primarily focus on the effects of load alleviation in designing a wing with wing shaping for minimum induced drag or minimum fuel burn over the optimal flight-path trajectory. In Chapter 2, closed-form solutions are presented for the lift distribution and wing-structure weight that minimize

induced drag on tapered wings with structural constraints and static wing-shaping to produce a fixed lift distribution at all flight phases. These solutions extend earlier analytic work on rectangular wings and show that using a tapered planform rather than a rectangular planform can allow up to a 15% larger wingspan for the same wing-structure weight, reducing induced drag by up to 25%. Minimum induced drag is obtained below stall with a triangular wing having a taper ratio of $R_T = 0$. When the lift distribution and wing-structure weight are optimized, the optimum design has a lift distribution that is nearly fully characterized by the Fourier coefficient B_3 from Eq. (1.1) in the introduction, with all other Fourier coefficients having very little influence. The theoretical optimum wing-structure weight matches that of the rectangular wing at one-half the net weight of all other wing components for the stress-limited design, independent of all other design parameters.

The results from Chapter 2 provide valuable insights into the effects of the planform on the aerodynamic and structural coupling involved in designing a wing with static wing shaping for minimum induced drag. However, they include some assumptions that are not necessarily representative of many practical aircraft configurations. In particular, the solutions in Chapter 2 are all obtained assuming that the weight is distributed in the wing according to an ideal weight distribution. The solutions are also limited to wings with elliptic or linear taper. Therefore, in Chapter 3, the methodology from Chapter 2 is generalized using low-order numerical methods to accommodate wings with arbitrary planform and payload distribution.

In Chapter 3, these methods are used to perform optimization and a design-space exploration on the NASA Ikhana high-endurance UAV, including a sensitivity study to

estimate the relative influence of various aerodynamic, structural, and operational constraints on the induced drag for an aircraft configuration more representative of a typical aircraft design. This study shows that the relative influence of design parameters, including the wingspan and wing-structure weight, depend on where the design falls in the design space. For example, for the baseline Ikhana design, the induced drag is much more sensitive to the wingspan than the wing-structure weight. Therefore, a design that can alleviate loads to allow a larger wingspan can have greater efficiency benefits than a design that reduces the wing-structure weight through load alleviation. The results also show that for Ikhana, the optimum lift distribution from static wing shaping is very similar to the analytic and closed-form solutions for rectangular wings and tapered wings and is nearly independent of the degree of taper in the wing. Moreover, the optimum wing-structure weight is nearly independent of all other design variables and is very near the optimum analytical wing-structure weight for a rectangular wing of one-half the weight of all other components for the stress-limited design and one-fourth the weight of all other components for the deflection-limited design. This confirms the analytic results for rectangular wings and the closed-form results for tapered wings shown in Chapter 2.

The results in Chapter 3 suggest that analytic aerostructural solutions for static wing shaping may be representative for a variety of aircraft configurations and flight conditions. In order to assess this, and by extension, to assess whether these solutions may serve as aerostructural reference solutions in the development of higher-fidelity computational models, in Chapter 4, a series of theoretical aerostructural solutions are compared to results from multi- and high-fidelity static-wing-shaping studies with similar constraints. The results in Chapter 4 suggest that depending on the design constraints, the

theoretical aerostructural solutions for the optimum lift distribution show good agreement with high-fidelity results. The greatest agreement is seen between theoretical and high-fidelity solutions for the stress-limited design of wings with operational constraints related to the wing loading. This is perhaps not very surprising, since these constraints were chosen specifically to represent typical aircraft operation. However, the results showed a surprising level of agreement across aircraft configurations and flight conditions, suggesting that for this case, the operational constraints may play a relatively important role in determining optimal wing shaping configurations that minimize drag. The high level of agreement in these results also suggests that using the analytic solutions presented in this dissertation and in other studies can provide a good reference for determining optimal wing shaping configurations under structural constraints. However, it is important to note that the results in Chapter 4 also suggest that several effects that are not considered in the analytic studies, including transonic effects, viscosity, and aeroelasticity, are important in predicting the extent to which using the optimal lift distribution can reduce drag or fuel consumption.

In Chapter 5, a simple method is presented for predicting the optimum flight-path trajectory and the corresponding fuel consumption for an aircraft in quasi-steady level cruise. This method is used to predict how both static and active wing shaping may affect the optimum trajectory. The effects of wing shaping are modeled through changes in the B_3 alone, as suggested by the results in Chapter 2. The aerostructural relationships from Chapters 2 and 3 are used to model the effects of load alleviation from both static and active wing shaping. Case studies are presented for the NASA Ikhana and the NASA Common Research Model Aircraft. The results in Chapter 5 show that using active wing

shaping can reduce fuel consumption by up to 8% over the baseline design between around 6-7% over the optimum static wing-shaping configuration. Greater fuel savings are achieved by using load alleviation from wing shaping to increase the wingspan than by using load alleviation from wing shaping to reduce the wing-structure weight. The greatest fuel savings are achieved by using active morphing to dynamically change the lift distribution to alleviate loads at the high-load structural design condition and to operate with high efficiency during cruise. The results in Chapter 5 also suggest that wing shaping can have a substantial effect on the optimum flight-path trajectory, generally favoring a lower velocity profile with a high lift coefficient and high lift-to-drag ratio. When the altitude is allowed to vary along the trajectory, the optimum flight profile has a constant lift-to-drag ratio and a constant lit coefficient. Therefore, if a non-morphing wing is optimized to operate with maximum L/D at the design cruise lift coefficient, the non-morphing wing would be able to operate at the design condition for the vast majority of cruise. For such a wing, it is expected that morphing retrofit without any load alleviation would provide minimal efficiency benefits, since the morphing mechanism would be unable to significantly improve L/D over the non-morphing configuration at the design lift coefficient, and both would operate with the same configuration at the same condition for nearly the entirety of the cruise.

Taken as a whole, the results in this dissertation reveal some important insights related to static and active wing shaping. For example, Chapters 2, 3, and 5 all suggest that for many typical aircraft designs, the design space is such that load allevation from active or static wing shaping of wing shaping is more effectively leveraged to increase the wingspan of a wing, rather than reduce the weight of the wing. While this may seem

intuitive, aerostructural literature often tends to focus on the load alleviation benefits of wing shaping in terms of weight reduction alone, rather than in terms of increasing the wingspan. As was the case in Chapter 5, many of these studies show similarly small benefits from using load alleviation from wing shaping to reduce the weight. In fact, Results for the optimum wing-structure weight in Chapters 2 and 3 suggest that drag may be minimized with a wing having a lift distribution and wingspan that results in greater wing-structure weight than the baseline design. This optimum wing-structure weight has been shown to be nearly independent of all other design variables. In addition, as discussed in Chapters 2 and 3, the optimum lift distribution for the wing configurations shown here are primarily characterized by only the Fourier coefficient B_3 from Eq. (1.1). As this coefficient represents a low-frequency symmetric harmonic in the lift distribution, it plays a substantial role in determining the general shape of the lift distribution. This suggests that substantial benefits from wing morphing may be obtained using relatively low-frequency and low-resolution changes in the lift distribution, reducing the need to produce morphing mechanisms with the capability to produce high-frequency variations in the lift distribution.

APPENDICES

APPENDIX A

Characterization of the Common Research Model Wing for Low-Fidelity Aerostructural Analysis

Jeffrey D. Taylor* and Douglas F. Hunsaker†
Utah State University, Logan, Utah 84322-4130

A characterization of the Common Research Model (CRM) wing for low-fidelity aerostructural optimization is presented. The geometric and structural properties are based on the CAD geometries and finite-element models for the CRM wing and the undeflected Common Research Model Wing (uCRM). Three approximations are presented for the elastic axis from previously-published studies on wing boxes similar to the uCRM, and approximations of the flexural and torsional rigidity are presented from a previously-published study using the uCRM wing. The characterization presented in this paper is intended to be used within low-fidelity aerostructural analysis tools to facilitate rapid design optimization and exploratory studies using the CRM wing.

Nomenclature

$a_{i,b}$ = fit coefficients in the exponential fit for flexural rigidity

$a_{i,cg}$ = fit coefficients in the polynomial fit for section center of gravity

$a_{i,ea}$ = fit coefficients in the polynomial fit for section center of gravity

$a_{i,t}$ = fit coefficients in the exponential fit for torsional rigidity

$a_{\mathrm{L},ijk}$ = fit coefficients in the multidimensional fit for section lift coefficient

$a_{\mathrm{m},ijk}$ = fit coefficients in the multidimensional fit for section moment coefficient

* PhD Candidate, Mechanical and Aerospace Engineering, 4130 Old Main Hill, AIAA Student Member
† Assistant Professor, Mechanical and Aerospace Engineering, 4130 Old Main Hill, AIAA Senior Member

This paper was presented at the AIAA SciTech 2021 Virtual Forum as:
Taylor, J. D., and Hunsaker, D. F., "Characterization of the Common Research Model Wing for Low-Fidelity Aerostructural Analysis," AIAA 2021-1591, AIAA SciTech 2021 Virtual Forum, 11-15 & 19-21 January 2021. (doi:10.2514/6.2021-1591).

$a_{D,ijk}$ = fit coefficients in the multidimensional fit for section drag coefficient

b = wingspan

C_δ = shape coefficient for the deflection-limited design

C_σ = shape coefficient for the stress-limited design

C_b = fit coefficient in the exponential fit for flexural rigidity

C_D = drag coefficient

C_L = lift coefficient

C_m = moment coefficient

C_t = fit coefficient in the exponential fit for torsional rigidity

c = local wing section chord length

c_{ref} = wing reference chord

c_t = local wing section chord length at the wing tip

D_i = wing induced drag

E = modulus of elasticity of the wing-structure material

G = shear modulus of the wing-structure material

h = spar height of the wing-structure cross-section

I = beam section moment of inertia

J = torsion constant of the wing-structure cross section

M = freestream Mach number

R_A = wing aspect ratio

R_T = wing taper ratio

S = wing planform area

S_{exp} = exposed wing area

S_{ref} = wing reference area

t = panel thickness of the wing-structure

$x_{c/4}$ = x location of the wing-section quarter chord

$z_{c/4}$ = z location of the wing-section quarter chord

ρ = air density

ξ = normalized spanwise coordinate

I. Introduction

THE common research model (CRM)[1] is an open-source aircraft geometry that was developed in 2007-2008 [1] through a partnership between NASA, Boeing, and other industry and government groups for the validation and assessment of computational-fluid-dynamics (CFD) tools [1,2]. The CRM geometry is representative of a typical wide-body transonic transport aircraft. NASA has compiled extensive experimental data for the CRM from at least four wind-tunnel tests [3-5], and several CRM variants have been developed for further study, including a high-lift variant (CRM-HL) [6], a natural laminar flow variant (CRM-NLF) [7-9], and additional variants created by the Office National d'Etudes et de Recherches Aerospaciales (ONERA) [10,11] in France, the Japan Aerospace Exploration Agency (JAXA) [12] in Japan, and the National Research Council (NRC) [13] in Canada.

Aerostructural CRM variants that include a representative wing box have also been presented by Kilmmek [14], Kennedy et al. [15] (QCRM) and Brooks et al. [16] (uCRM-9). Because the CRM was originally developed for aerodynamic validation, the CRM wind-tunnel model wing was designed to match the 1-*g* cruise geometry. However, as pointed out by Keye et al. [17], the wind-tunnel model experiences significant aeroelastic deflection at the cruise condition, which can cause discrepancies between rigid-wing computational results and wind-tunnel data. The aerostructural models presented by Klimmek [14] and Brooks et al. [16] were created to address this concern, and to facilitate analysis of the CRM at multiple flight conditions, including off-design conditions.

In fulfillment of its original purpose, the CRM and its variants have been used in hundreds of high-fidelity CFD studies throughout government, industry, and academia. For example, the CRM was the subject for AIAA CFD drag prediction workshops IV-VI [18-23]. The CRM-HL configuration has been

[1] https://commonresearchmodel.larc.nasa.gov/

used in AIAA high-lift prediction workshops III and IV and is currently the subject of the AIAA stability and control prediction workshop. The uCRM has been used in several aerostructural and multidisciplinary design optimization studies [16,24]. In addition to benchmarking, the CRM and its variants have also been used as a baseline configuration in studies regarding aircraft icing [25,26], flutter [27], and morphing-wing technologies [28].

Although the CRM was originally intended for validation of high-fidelity CFD tools, it can also provide an excellent benchmark case for the validation of low- to mid-fidelity aerodynamic and aerostructural tools. Low- and mid-fidelity methods also require less computation time than higher-fidelity methods, which makes them ideal for exploratory and proof-of-concept studies. In many cases, these low- and mid-fidelity methods have been shown to be in good agreement with grid-resolved CFD [29-36]. However, to date, there have been very few low- to mid-fidelity studies that use the CRM geometry. This may be, in part, because the publicly-available CRM geometry presents some challenges for many low-fidelity tools. The most apparent challenge is that the official CRM geometry is presented only in initial graphics exchange specification (IGES) and CAD format. The uCRM wing and wing-box geometries are also available in CAD format[2]. In each case, only the outer mold line of the aircraft is given. This is convenient for CFD meshing, but it is not useful for many low-fidelity tools.

It appears that Vassberg et al. [1] give the most detailed description of the full-scale CRM model in their inaugural CRM publication. In this publication, Vassberg et al. [1] present data for the wing leading- and trailing-edge coordinates, twist, chord, thickness-to-chord ratio, max camber, and camber slope at 21 spanwise sections. However, neither the airfoil geometries nor the airfoil performance data is given. This creates a challenge for low-fidelity tools that require 2-D airfoil data for aerodynamic analysis, such as tools based on lifting-line theory [34,37-38]. Moreover, the locus of aerodynamic centers, the quarter-chord-sweep distribution, and the dihedral distribution must be inferred or extracted from the CAD geometry. The same is true for the uCRM geometry. The process of extracting the geometric details from the CAD models and other resources often requires significant time and effort.

[2] http://mdolab.engin.umich.edu/ucrm

The purpose of this paper is to present a detailed overview of the CRM and uCRM-9 wing geometries for use in low-fidelity aerodynamic and aerostructural analyses. In the following sections, we describe the geometry of the CRM and uCRM wings and the uCRM wing box, extracted from publicly-available CAD models, and we present an example weight breakdown for the uCRM-9 model for use in aerostructural analyses.

II. Wing Geometry

The coordinate system used in this paper is shown in Fig. 1. The axes are aligned with the conventional body-fixed axes, with the origin at the quarter-chord location of the root airfoil section, as projected to the fuselage centerline. The x-axis is aligned with the horizontal and points out the nose of the aircraft, the y-axis is aligned with the horizontal and points out the right wing, and the z-axis is aligned vertically and points straight down out the bottom of the aircraft, as shown.

The data in this section were extracted from the CAD models for the CRM and uCRM wings using SolidWorks. The CRM wing has a wingspan of $b = 58.76$ m and an aspect ratio of $R_A = 9$. The total wing area is $S = 412.7$ m^2, the reference area is $S_{\text{ref}} = 383.74$ m^2, and the exposed wing area is $S_{\text{exp}} = 337.05$ m^2. The wing is double tapered with a break at 37% semispan and a taper ratio of $R_T = 0.533$ inboard of the break and $R_T = 0.376$ outboard of the break. The reference chord is $c_{\text{ref}} = 7.01$ m. The CRM wing is designed for cruise at $M = 0.85$ at an altitude of 37000 ft (11275 m) and a lift coefficient of $C_L = 0.5$. For standard atmospheric conditions with no temperature offset, this gives a Reynolds number near $\text{Re} = 4.3 \times 10^7$. Wing and flight reference values are summarized in Table 1.

The uCRM wing has the same wingspan and planform shape as the CRM but is designed to represent the undeflected, 0g loading case for the CRM. The uCRM wing also includes a wing box that was designed through a reverse-engineering process and produces the original CRM shape when loaded at cruise. A top-down view of the CRM/uCRM planform is shown in Fig. 2.

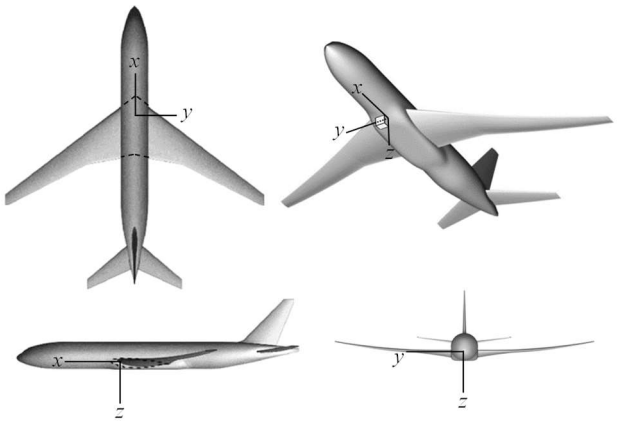


Fig. 1 Coordinate system for the CRM and uCRM wing.

Table 1 Wing and flight reference values for the CRM/uCRM.

wingspan, m	58.76
aspect ratio	9.00
total wing area, m^2	412.70
reference area, m^2	383.74
exposed wing area, m^2	337.05
reference chord, m	7.01
altitude, m	11275.19
Mach number	0.85
lift coefficient	0.50
Reynolds number	4.33×10^7

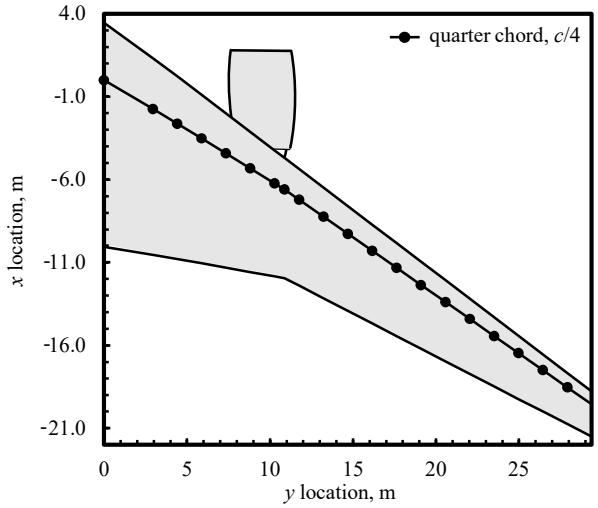


Fig. 2 Planform view of the CRM/uCRM wing geometry.

A. Chord distribution

In Ref. [1], Vassberg et al. give the wing chord distribution, and other wing geometry parameters, at 21 spanwise locations beginning at the wing root and ending at the wing tip. For consistency, the data in this section are shown at the same 21 spanwise locations. The chord was verified from the uCRM CAD geometry by slicing the wing at each spanwise location of interest on a plane parallel to the *x*-axis and perpendicular to the projection of a spline fit through the locus of section quarter-chord points in the *y*-*z* plane. Accounting for the wing twist, the chord was measured from the local airfoil cross-section leading edge to trailing edge. The resulting chord distribution matched the data given by Vassberg et al. [1] for the CRM. The chord distribution is shown in Fig. 3, and values for the local chord at the 21 locations given by Vassberg et al. [1] are given in Table A1 in the appendix.

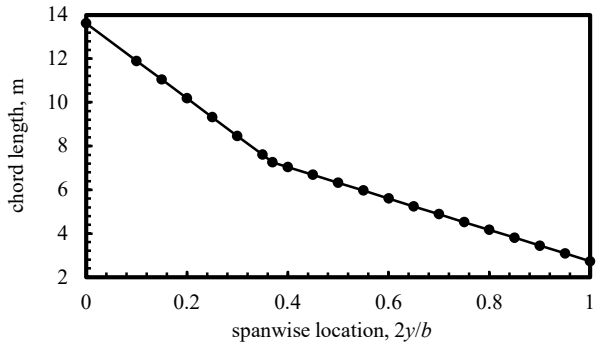


Fig. 3 Chord distribution for the CRM/uCRM wing.

B. Quarter-Chord Sweep

The spanwise variation in quarter-chord sweep for the uCRM and CRM wings is shown in Fig. 4. The sweep angle was extracted from the CAD model by measuring the angle in the *x*-*y* plane between the *y*-axis and a line tangent to the projection in the *x*-*y* plane of the locus of section quarter-chord points at each of the 21 spanwise locations of interest. The results in Fig. 4 show that outboard of the break ($2y/b = 0.37$), the sweep angle is fairly constant at around 35 degrees. Note that the sweep distribution for the CRM at cruise and uCRM at 0*g* have slight differences to account for the effects of bending about the z-axis. However,

these differences are small. Values for the sweep distributions of the CRM and uCRM are given in Table A1 in the appendix.

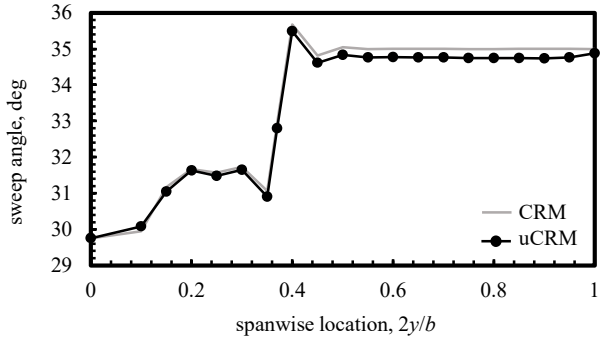


Fig. 4 Spanwise variation in quarter-chord sweep angle for the CRM and uCRM wings.

C. Quarter-Chord dihedral

The dihedral angle was obtained in a manner similar to that used to obtain the quarter-chord sweep angle. However, the dihedral angle at each spanwise section was measured between the *y*-axis and line in the *y*-*z* plane tangent to the projection of the locus of section quarter-chord points in the same plane. The resulting dihedral distributions for the CRM and uCRM wings are shown in Fig. 5. Here, we see that the uCRM dihedral distribution at 0g differs significantly from the CRM dihedral distribution at cruise due to the aeroelastic effects of bending about the *x*-axis. In fact, comparing the deflected 1g CRM geometry to the uCRM at 0g, bending in cruise results in a wingtip deflection of about 2.56 m, or 8.7% semispan. Values for the dihedral distributions for the CRM and uCRM are given in Table A1 in the appendix.

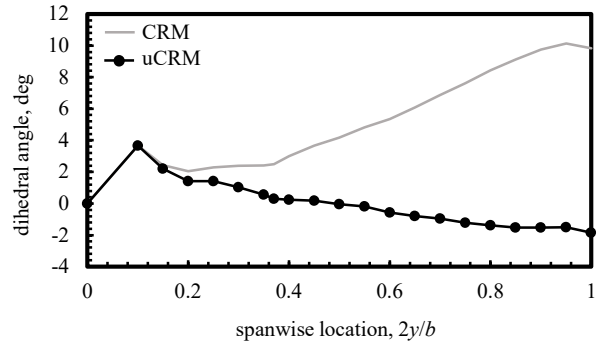


Fig. 5 Spanwise variation in quarter-chord dihedral angle for the CRM and uCRM wings.

D. Wing twist

Figure 6 shows the wing-twist distribution for the CRM and uCRM wings. The wing twist was obtained by measuring the angle between the *x*-axis and the chord of the local airfoil section, which was obtained as described in Section II.A. Figure 5 shows that the twist distribution for the uCRM varies significantly from the CRM twist distribution to account for the effects of aeroelastic twist. The twist distributions for the CRM and uCRM wings are given in Table A1 in the appendix.

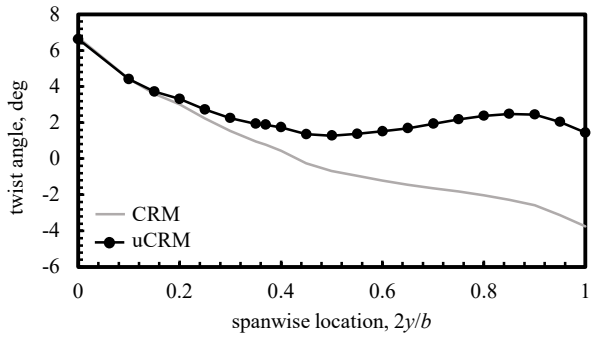


Fig. 6 Wing twist distribution for the CRM and uCRM wings.

E. Section Airfoil properties

Some low-fidelity aerodynamic tools require section aerodynamic properties. In order to obtain these properties, the airfoil section geometric profiles must be known. The airfoil section profiles were obtained from the CRM and uCRM CAD geometries by extracting the intersection curves between the wing surface and the plane parallel to the *x*-axis and perpendicular to the projection in the *y*-*z* plane of a spline fit through the locus of section quarter-chord points. By extracting the airfoil profiles in this manner, the rigid-body rotation of the wing due to bending about the *x*-axis is preserved, so that the uCRM airfoils and CRM airfoils are consistent. The airfoil stacks for the CRM and uCRM wings are shown in Fig. 7. For better visualization of the airfoils, Fig. 8 shows a schematic of the CRM/uCRM airfoils with zero twist, aligned at

the quarter chord location. The coordinates of the airfoil surfaces are available from the Utah State University library repository.[3]

With the airfoil geometric profiles known, the section properties can be obtained using any airfoil analysis tool. In this paper, transonic data for lift coefficient, moment coefficient, and drag coefficient were obtained using the method given by Fujiwara et al. [39], which couples the transonic small-disturbance theory code TSFOIL with an integral boundary-layer method. Data were obtained for a series of angles of attack, Reynolds numbers, and Mach numbers.

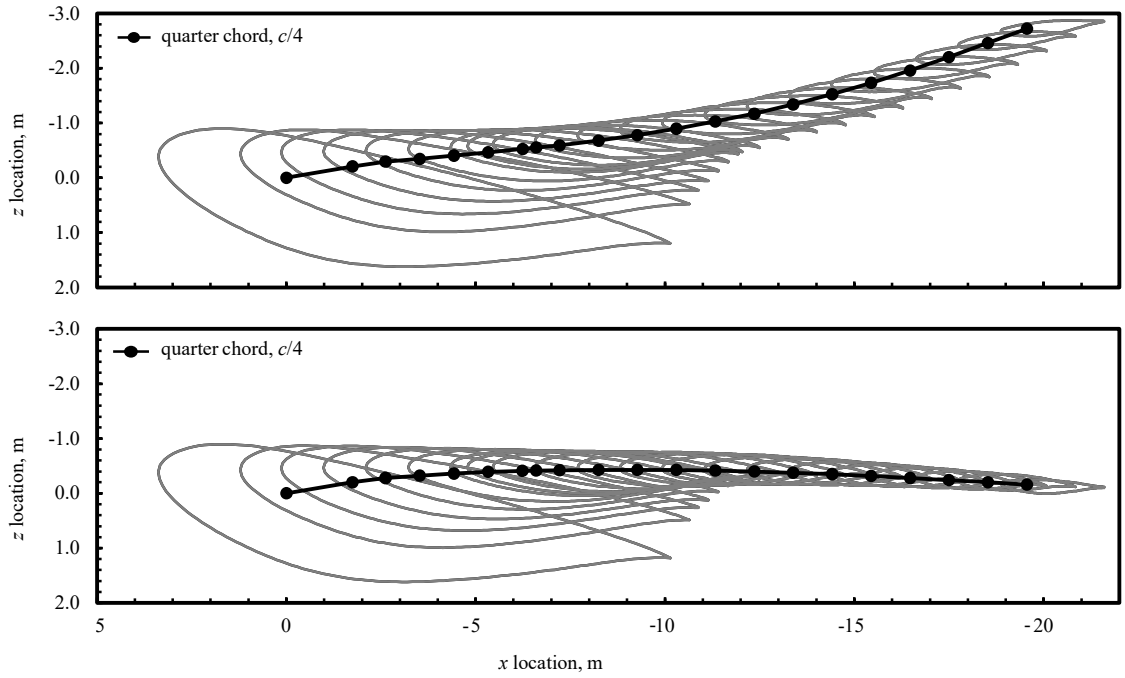


Fig. 7 Airfoil stacks for the CRM wing (top) and the uCRM wing (bottom).

[3] https://digitalcommons.usu.edu/all_datasets/125 (doi: 10.26078/8nv8-yj03)

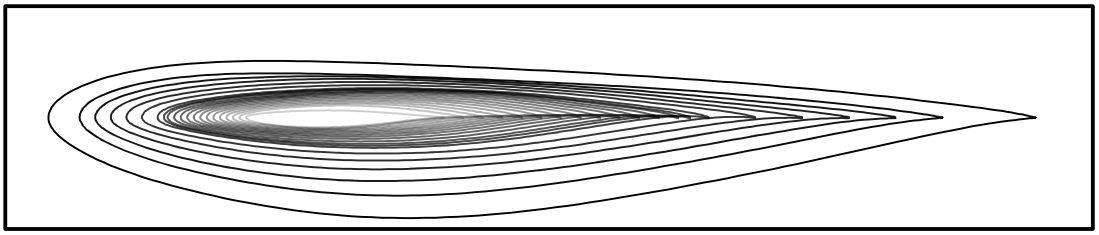


Fig. 8 Schematic of the CRM/uCRM airfoils with zero twist, aligned at the quarter-chord location.

Full airfoil data is available to the reader through the Utah State University library repository.[4] The method shown by Ullah et al. [40] was used to obtain a series of multi-dimensional curve fits to data for airfoil lift coefficient, drag coefficient, and moment coefficient as a function of angle of attack, Reynolds number, and Mach number. For simplicity, in this paper, we use multidimensional linear fits for the lift coefficient and moment coefficient and multidimensional parabolic fits for the drag coefficient, i.e.,

$$\widetilde{C}_L = \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} a_{L,ijk}\,\alpha^i\,\mathrm{Re}^j\,M^k \tag{1}$$

$$\widetilde{C}_m = \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} a_{m,ijk}\,\alpha^i\,\mathrm{Re}^j\,M^k \tag{2}$$

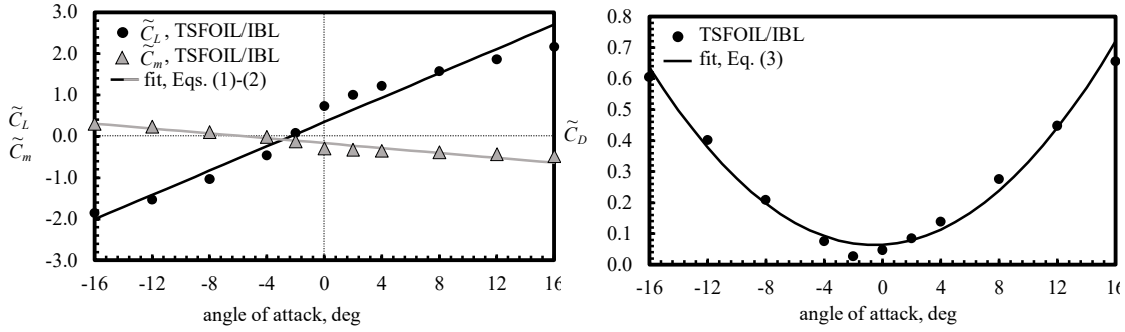
$$\widetilde{C}_D = \sum_{i=0}^{2}\sum_{j=0}^{2}\sum_{k=0}^{2} a_{D,ijk}\,\alpha^i\,\mathrm{Re}^j\,M^k \tag{3}$$

where $a_{L,ijk}$, $a_{m,ijk}$, are $a_{D,ijk}$ are arrays of fit coefficients, which are given in Tables A2-A6 in the appendix for all of the CRM/uCRM airfoils. For reference, the data and fits for the lift coefficient, moment coefficient, and drag coefficient, as a function of angle of attack, of the break airfoil ($2y/b = 0.37$) at a Reynolds number of $\mathrm{Re} = 3.22\times10^7$ and a Mach number of $M = 0.84$ are shown in Fig. 9.

To give a more intuitive visualization of the spanwise variation in airfoil properties for the CRM and uCRM wings, the spanwise change in the lift slope $\widetilde{C}_{L,\alpha}$ and the coefficient $\widetilde{C}_{L_0}$; the moment parameters $\widetilde{C}_{m,\alpha}$ and $\widetilde{C}_{m_0}$; and the drag parameters $\widetilde{C}_{D_0}$, $\widetilde{C}_{D_L}$, and $\widetilde{C}_{D_{L^2}}$, are shown in Fig. 10. Note that the parameters $\widetilde{C}_{L,\alpha}$ and $\widetilde{C}_{L_0}$ come from the linear approximation for lift as a function of angle of attack, the parameters $\widetilde{C}_{m,\alpha}$ and $\widetilde{C}_{m_0}$ come from the linear approximation for the moment coefficient as a function of angle of attack, whereas coefficients $\widetilde{C}_{D_0}$, $\widetilde{C}_{D_L}$, and $\widetilde{C}_{D_{L^2}}$ come from the parabolic approximation for the drag

[4] https://digitalcommons.usu.edu/all_datasets/125 (doi: 10.26078/8nv8-yj03)

coefficient as a function of α. Although we have chosen to use linear and low-order fits for the airfoil data in this paper, the methods shown in this subsection can be generalized to obtain higher-order polynomial fits for any of the airfoil data.



(a) (b)

Fig. 9 Airfoil data and polynomial fits for (a) the lift and moment coefficients and (b) the drag coefficient for the break airfoil located at $2y/b = 0.37$ with a Reynolds number of 3.22×10^7 and a Mach number of 0.84

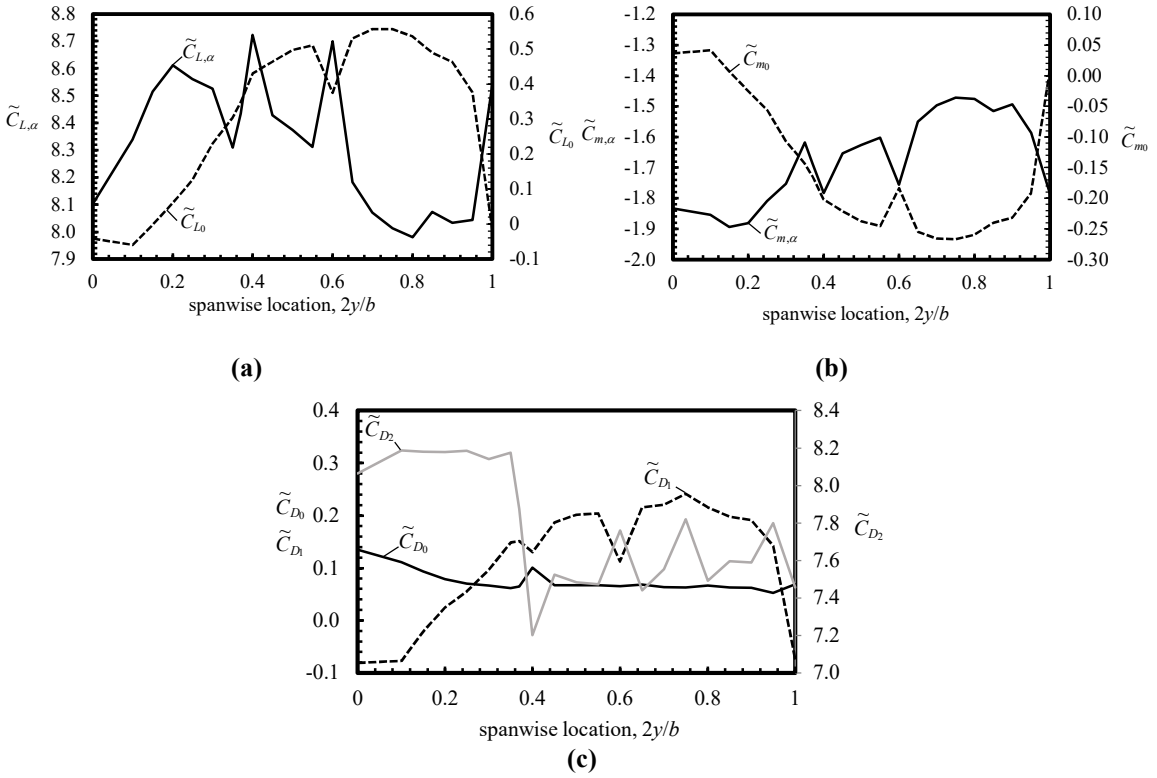


Fig. 10 Section airfoil properties as a function of spanwise location; (a) parameters for the linear approximation of the lift coefficient with respect to angle of attack, (b) parameters for the linear approximation of the moment coefficient with respect to angle of attack, and (c) parameters for the parabolic approximation of the drag coefficient with respect to angle of attack.

III. Wing Box Geometry

The uCRM-9 wing box was designed based on cutaway drawings for the Boeing 777-200ER wing structure and tailored to conform to the CRM 1-g outer mold line [16]. The jig twist for the uCRM geometry was then obtained using an inverse-engineering process, as described by Brooks et al. [16]. The wing box includes upper and lower skins, a front and rear spar, and 49 ribs, placed chordwise along the wing box running length. The data in this section is reported at each of these rib locations. Figure 11 shows a planform view of the wing box and its location within the uCRM wing. A description of the outer dimensions of the wing box, approximations for the center of gravity and the elastic axis, and approximations for the wing flexural and torsional rigidity are given in the following subsections. Note that

because the ribs are oriented perpendicular to the wing running length, the *y* data for the front spar, rear spar, center of gravity, and elastic axis vary slightly in the swept portion of the wingbox. The flexural and torsional rigidity are reported at the *y* coordinates of the elastic axis. Values for key wing box geometric parameters are shown in Tables A7 and A8 in the appendix.

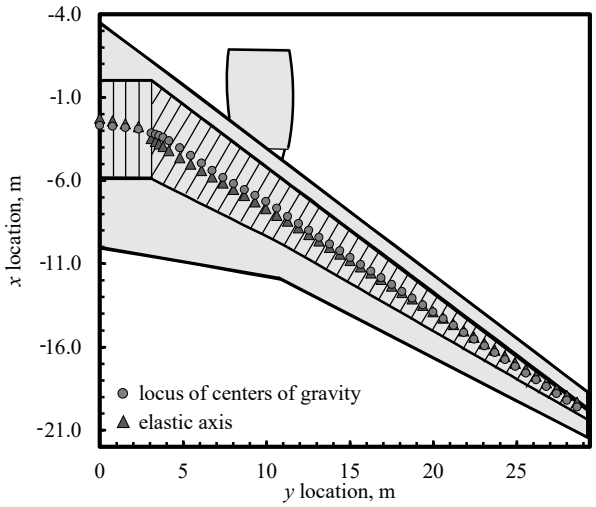


Fig. 11 Planform view of the uCRM-9 wing box as extracted from the CAD geometry.

A. Wing Box Dimensions

The uCRM-9 wingbox includes a leading-edge spar and trailing-edge spar connected by upper and lower panels that conform to the upper and lower wing skins. The wing-box is straight from the root to the fuselage body, which lies at about 10% of the semispan and swept outboard of the fuselage body. Outboard of the fuselage, the leading-edge spar is nearly straight, with a minor kink at the wing break. The trailing-edge spar also has a minor kink at the break. The normalized chordwise location of the leading and trailing-edge spars are shown in Fig. 12 as a function of span.

Because the wing box conforms to the airfoil geometry, the leading- and trailing-edge spars have different heights. Figure 13 shows the normalized spar height for each spar as a function of span. The thicknesses of the wingbox components was obtained from the wing box finite-element file for the

uCRM-9, given by the University of Michigan.[5] The thicknesses for the front and rear spars, upper and lower skins, and ribs are shown in Figs. 14, 15, and 16, respectively. Note that due to the change in wing box sweep at 10% semispan, there is no corresponding rear-spar section for ribs 4-6, and rib 4 has four distinct sections, labeled in Fig. 16, in order from front spar to rear spar, as *a*, *b*, *c*, and *d*.

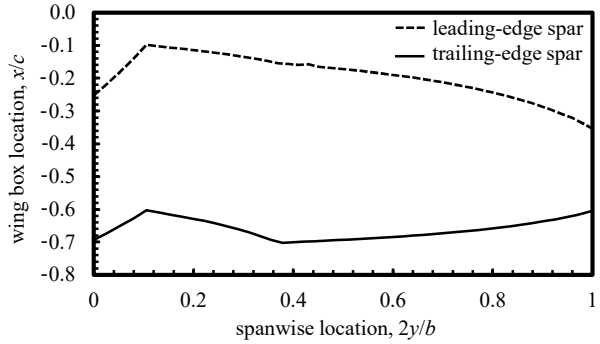


Fig. 12 Normalized chordwise location (measured from the wing leading edge) of the leading- and trailing-edge spars of the uCRM-9 wingbox, as a function of span.

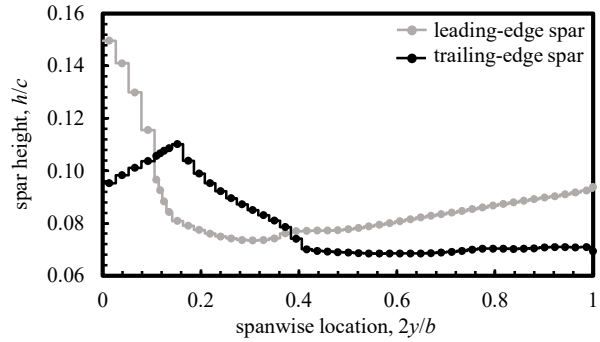


Fig. 13 Normalized spar-height distribution for the leading- and trailing-edge spars of the uCRM-9 wing box.

[5] http://mdolab.engin.umich.edu/ucrm

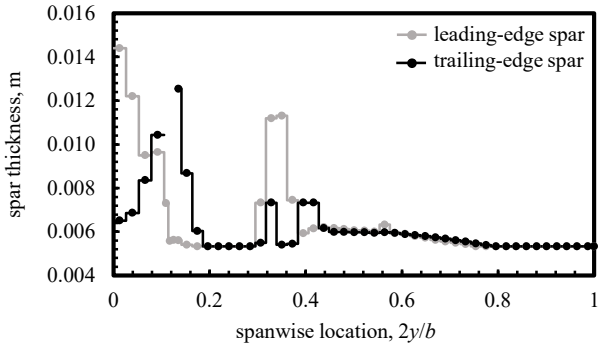


Fig. 14 Thicknesses of the leading- and trailing-edge spars for the uCRM-9 wing box.

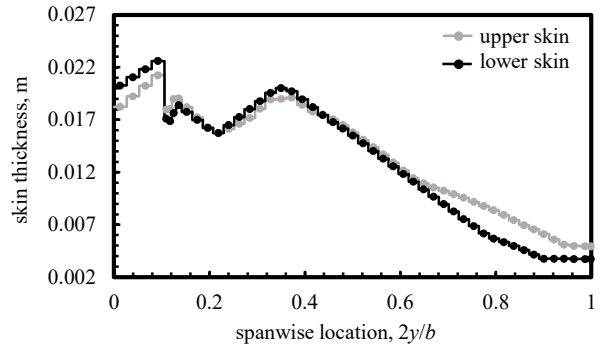


Fig. 15 Thicknesses of the upper and lower skins for the uCRM-9 wing box.

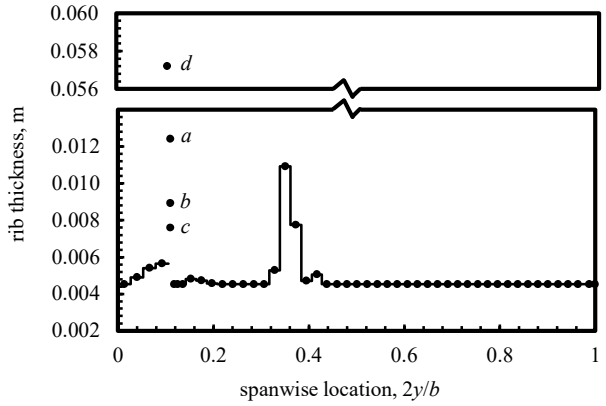


Fig. 16 Thicknesses of the ribs for the uCRM-9 wing box.

B. Locus of Centers of Gravity and Elastic Axis

The locus of section centers of gravity for the uCRM-9 wing box was obtained by extracting the center of gravity of the wing box cross-section from the finite-element model for the uCRM-9 at each spanwise

location of interest. The resulting locus of centers of gravity is shown in Fig. 17, normalized by the local chord. For convenience, the normalized data were fit to a polynomial. Because the wing-box geometry is discontinuous at the wing-body junction (10% semispan) and the break (37% semispan), the fits were performed independently on three sections spanning 0-10% semispan, 10-37% semispan, and 37-100% semispan, respectively. The result is a piecewise function of the form

$$\frac{x_{cg}}{c} = \begin{cases} a_{0,cg} + a_{1,cg}\xi & 0 \le \xi \le 0.1 \\ b_{0,cg} + b_{1,cg}\xi + b_{2,cg}\xi^2 + b_{3,cg}\xi^3 & 0.1 < \xi \le 0.37 \\ c_{0,cg} + c_{1,cg}\xi + c_{2,cg}\xi^2 & 0.37 < \xi \le 1.0 \end{cases} \tag{4}$$

where $\xi = 2y/b$ is the normalized spanwise coordinate, and $a_{i,cg}$, $b_{i,cg}$, and $c_{i,cg}$ are the fit coefficients for the center of gravity, which are given in Table 2. The fits are shown with the data in Fig. 17.

For most aerostructural studies, the elastic axis is obtained from FEM models of the wingbox. However, FEM analysis is beyond the scope of this paper. Instead, we show here results from previously-published data for the elastic axis of wings similar to the uCRM-9. The three studies considered here are from Chauhan and Martins [24], Cramer and Nguyen [41], and Stodieck et al. [42]. Chauhan and Martins [24] approximated the elastic axis of the uCRM-9 wingbox in using the weighted-average process described in the previous paragraph. The result is identical to the center of gravity estimate shown in Fig. 17. Cramer and Nguyen [41] approximated the elastic axis for an elastic wind-tunnel model as a straight line with a sweep angle of 31.5 degrees beginning at about 40% of the chord at the wing-body junction. The elastic axis presented by Stodieck et al. [42] was obtained from computational models of an aluminum wing box, designed by the authors for the CRM. The elastic axis from each of these studies is shown in Fig. 18.

Averaging the data from each of these studies gives the data points denoted by black circles in Fig. 18. Using the same wing partitions as shown in Eq. (4), the normalized average elastic-axis data were fit to a piecewise function of the form

$$\frac{x_{ea}}{c} = \begin{cases} a_{0,ea} + a_{1,ea}\xi & 0 \le \xi \le 0.1 \\ b_{0,ea} + b_{1,cg}\xi + b_{2,ea}\xi^2 + b_{3,ea}\xi^3 + b_{4,ea}\xi^4 & 0.1 < \xi \le 0.37 \\ c_{0,ea} + c_{1,ea}\xi + c_{2,ea}\xi^2 & 0.37 < \xi \le 1.0 \end{cases} \tag{5}$$

Here, the coefficients $a_{i,ea}$, $b_{i,ea}$, and $c_{i,ea}$ are fit coefficients for the elastic axis, and are given in Table 2. The resulting fit is shown alongside the data in Fig. 18.

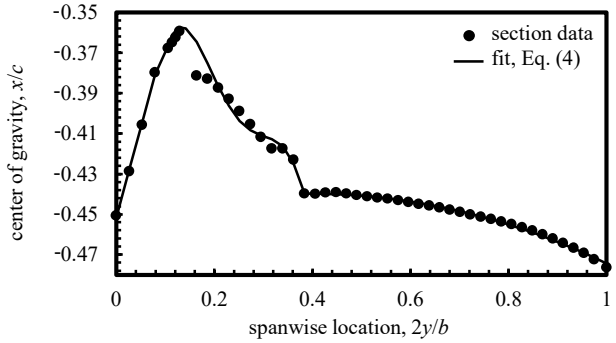


Fig. 17 Approximate normalized chordwise location (measured from the wing leading edge) of the center of gravity for the uCRM-9 wingbox.

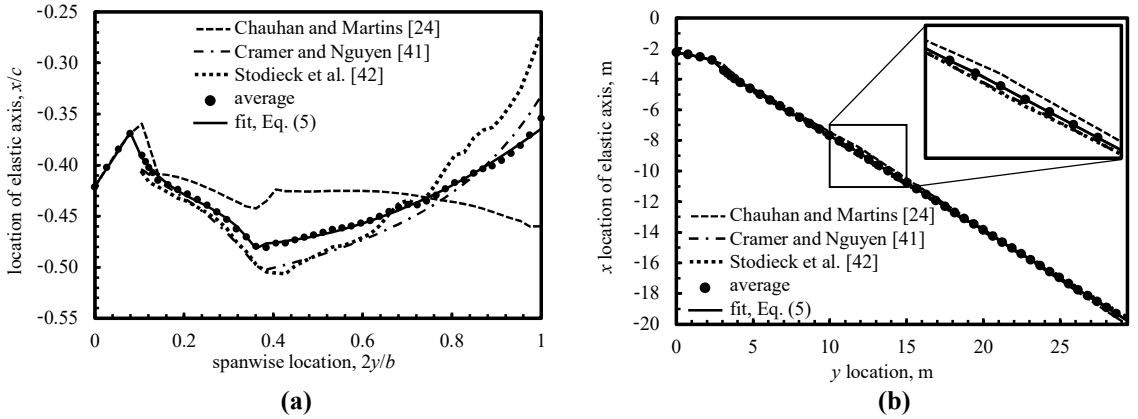


Fig. 18 Approximate locations of the uCRM-9 elastic axis in (a) normalized chordwise coordinates (measured from the wing leading edge) and (b) dimensional coordinates.

Table 2 Fit coefficients for the piecewise approximations of the locus of normlized centers of gravity and elastic axis (measured from the wing leading edge) of the uCRM-9 wing box.

center of gravity, $x_{cg}/c(\zeta)$		elastic axis, $x_{ea}/c(\zeta)$	
$a_{0,cg}$	0.4206	$a_{0,ea}$	0.4516
$a_{1,cg}$	-0.6640	$a_{1,ea}$	-0.8940
$b_{0,cg}$	0.2928	$b_{0,ea}$	0.7584
$b_{1,cg}$	1.4144	$b_{1,ea}$	-8.1124
$b_{2,cg}$	-5.1770	$b_{2,ea}$	56.8562
$b_{3,cg}$	7.4934	$b_{3,ea}$	-160.9238
		$b_{4,ea}$	162.2804
$c_{0,cg}$	0.4591	$c_{0,ea}$	0.4552
$c_{1,cg}$	0.1335	$c_{1,ea}$	-0.0771
$c_{2,cg}$	-0.2283	$c_{2,ea}$	0.0963

C. Flexural and Torsional Rigidity

The approximate flexural and torsional rigidity for the uCRM model were obtained from data presented by Fujiwara et al. [28]. In their study, Fujiwara et al. [28] presented the flexural and torsional rigidity required to produce the CRM 1-g geometry from their version of the uCRM model. The data shown in Fig. 19 were reproduced from this study. As was done for the center of gravity and elastic axis, the flexural and torsional rigidity were fit to a function. However, here, the fits were performed on the data within the range $0.1 \le \xi \le 1.0$. Below $\xi = 0.1$ the data were linearly interpolated to account for the dip shown in Fig. 19. The results are expressions for the flexural and torsional rigidity of the form

$$EI = C_b e^{a_{0,b} - a_{1,b}\xi} \tag{6}$$

$$GJ = C_t e^{a_{0,t} - a_{1,t}\xi} \tag{7}$$

where C_b, and $a_{0,b}$ and $a_{1,b}$ are fit coefficients for the flexural rigidity, and C_t, and $a_{0,t}$ and $a_{1,t}$ are fit coefficients for the torsional rigidity. Values for each of these coefficients are given in Table 3, and the fits are shown alongside the respective data in Fig. 19.

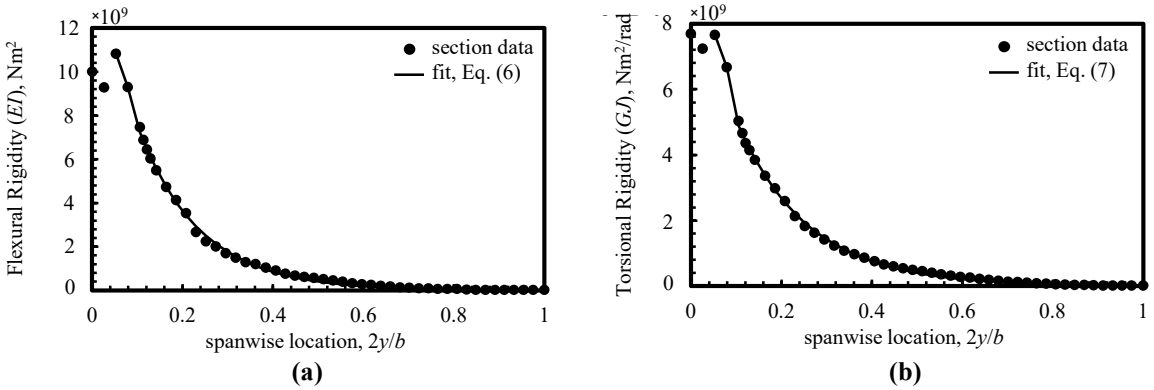


Fig. 19 Flexural Rigidity and Torsional Rigidity as a function of span for the uCRM-9 wing box. Reproduced from Fujiwara et al. [28]

Table 3 Fit coefficients for the approximate expressions for the flexural and torsional rigidity for the uCRM-9 wing structure.

Flexural Rigidity, $EI(\xi)$		Torsional Rigidity, $GJ(\xi)$	
C_b	100.3820	C_t	104.9792
$a_{0,b}$	18.8684	$a_{0,t}$	18.3235
$a_{1,b}$	7.3045	$a_{1,t}$	6.3429

IV. Weight Distribution

Key weight characteristics for the uCRM-9 can be obtained from data presented by Brooks et al. [16] and from publicly-available data for the Boeing 777-200ER [43], upon which the uCRM geometry is partially based. A summary of the weight breakdown is given in Table 4. Note that in this paper, we assume that the CRM carries one engine weighing 7,893 kg on each wing. The cruise weight is found from the nominal flight condition described at the beginning of Section II. Assuming that the CRM operates in steady level flight with 50% fuel at the nominal flight condition, the CRM weight with 50% fuel is found from the lift coefficient to be 220,240 kg. The weight with 100% fuel is then found by adding half of the maximum usable fuel weight (137,460 kg) for the Boeing 777-200ER [43] to give 288,970 kg, which is below the maximum takeoff weight (MTOW) of 297,550 kg [43]. The "net" weight in Table 4 is the total CRM weight with 100% fuel less the wing-structure weight, which is approximated using the method described below. The root weight is the net weight minus the total fuel load and the weight of both engines.

Table 4 Weight characteristics for the uCRM configuration.

Maximum Takeoff Weight (MTOW), kg	297,550
Maximum Zero-Fuel Weight (MZFW), kg	195,040
Operational Empty Weight (OEW), kg	138,100
Cruise Weight (50% Fuel), kg	220,240
Total Weight (100% Fuel), kg	288,970
Design Payload, kg	34,000
Usable Fuel Weight, kg	137,460
Root Weight, kg	105,806
Net Weight (100% Fuel), kg	259,052
Engine Weight, kg	7,893
Wing-Structure Weight, kg	29,895

For the low-fidelity CRM model, the wing-structure weight distribution was extracted from the uCRM-9 wing box finite element model. The material properties were chosen to be typical of 7000-series aluminum, as shown in Table 5. The resulting wing-structure weight distribution is shown in Fig. 20, without the weight of the ribs. Using the volume from the uCRM-9 wing box finite element model, including the ribs, and the density shown in Table 5, the total wing-structure weight is 23,916 kg, which matches the value found by Brooks et al. [16]. As suggested by Brooks et al. [16], we obtain the final wing-structure weight by multiplying this value by 1.25 to account for the weight of fasteners, overlaps, and other unmodeled structural components. The result is a final wing-structure weight of 29,895 kg, as reported in Table 4.

The net weight distribution is defined as the distribution of all non-structural components carried by the wing. Here, we assume that the majority of net weight consists of the fuel weight and the weight of the engines, which are mounted at about 32.7% of the semispan [43]. The approximate fuel model for the low-fidelity uCRM is based on publicly-available data for the Boeing 777-200ER [43,44]. As seen in Table 4, the maximum usable fuel weight is 137,460 kg. Based on fuel-tank layout diagrams for the Boeing 777-200ER [44], we assume that 57.7% of the fuel is carried in a center tank and 42.3% is carried in wing tanks. Assuming that the fuel density is 803.1 kg/m^3, and assuming that the fuel tanks fill the volume of the wing box, we find that in order to carry their respective portions of the fuel weight, the center tank must

extend to about 21% of the wing semispan, and the wing tank must extend from 21% semispan to 76% semispan. The fuel-tank layout is shown in Fig. 21. Over the course of a flight, fuel is first burned from the center tank, after which, fuel is burned from the wing tanks. Thus, as the fuel burns, the fuel-weight distribution changes, as shown in Fig. 22. Note that Fig. 22 also includes the weight of the engine in the net-weight distribution. The thrust-specific fuel consumption is estimated to be $c_T = 0.054$ kg/(N h).

Table 5 Material properties used for the low-fidelity CRM wing-structure weight estimation.

Density, kg/m^3	2780
Specific Weight, kg/(m^{2}s^2)	27,272
Modulus of Elasticity, Pa	7.31×10^{10}
Yield Strength, Pa	4.2×10^{8}
Poisson Ratio	0.33
Shear Modulus, Pa	2.75×10^{10}

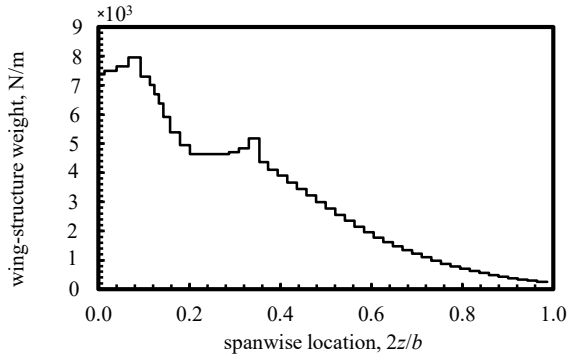


Fig. 20 Approximate wing-structure weight distribution for the CRM wing.

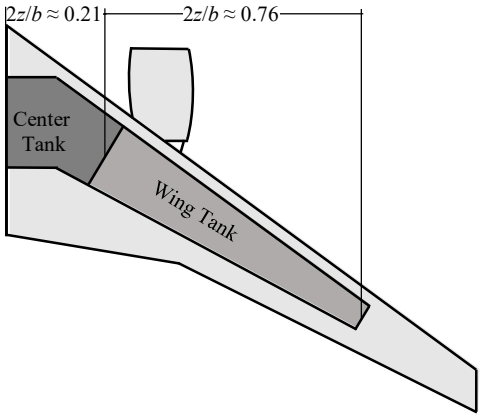


Fig. 21 Schematic of an example fuel-tank layout for the CRM.

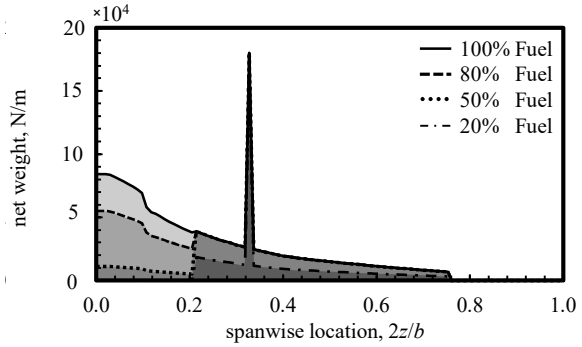


Fig. 22 Example net-weight distributions for the CRM.

V. Conclusion

The CRM was designed as a benchmark geometry for high-fidelity CFD methods, and it and its variants have been used in hundreds of high-fidelity studies throughout government, academia, and industry. The uCRM-9 geometry is an aerostructural variant of the CRM that includes a wing box model and an outer mold line representative of the 0-g geometry of the CRM. Although both the CRM and uCRM-9 are tailored for high-fidelity studies, they can also be used with low-fidelity models as a benchmark configuration for exploratory and proof-of-concept studies that require a high number of computations. However, most low fidelity methods require parameterized data of the geometry to be used. Extracting these data is often difficult and time consuming. Therefore, in this paper, we have presented a characterization of the CRM/uCRM-9 wing and the uCRM-9 wing box that includes geometric and weight data that can be used with low-fidelity aerostructural analysis tools.

The wing outer mold line geometry was extracted from CAD models of the CRM and uCRM-9 wings. The chord distribution, sweep distribution, dihedral distribution, and twist distribution are shown in Figs. 3-6. A summary of the wing properties and geometric distributions is given in Table A1 in the appendix. The airfoil profiles were also extracted from the CAD geometries and are shown in Figs. 7 and 8. Transonic data for the lift coefficient, moment coefficient, and drag coefficient for each airfoil were obtained using the transonic small-disturbance theory code TSFOIL in conjunction with an integral boundary layer method, as described in Section II.E. The data were obtained over a range of angles of attack, Reynolds numbers, and Mach numbers. Fit coefficients for the multidimensional linear fits of lift

coefficient and moment coefficient with respect to each of these variables are shown in Tables A2 and A3. Fit coefficients for the multidimensional parabolic fits of drag coefficient with respect to the same variables are given in Tables A4-A6.

The wing box geometry was extracted from CAD and finite-element models of the uCRM-9 wing box. A geometric description of the wing box, including its location within the wing and dimensions and thicknesses of the various wing box components, is given in Section III.A. The spar locations, spar heights, and wing box component thicknesses are given in Fig. 12, Fig. 13, and Figs. 14-16, respectively. The locus of aerodynamic centers was also obtained from the finite element model of the uCRM-9 wing box and is shown in Fig. 17. Obtaining the elastic axis for the uCRM-9 wing is beyond the scope of this study. Therefore, three approximations for the elastic axis from previously-published studies on wing similar to the uCRM-9 are shown in Fig. 18, along with the average of the three approximations. Similarly, approximations for the flexural and torsional rigidity were obtained from previously-published data, as shown in Fig. 19.

Section IV shows the weight breakdown of the uCRM-9 wing, based on available data on the Boeing 777-200ER and data from the University of Michigan. The structural weight distribution, without the ribs, is shown in Fig. 20. An example fuel model is also presented, based, in part, on available fuel data for the 777-200ER. The resulting net-weight distribution resulting from this model is shown in Fig. 22. It is anticipated that the low-fidelity characterization of the CRM/uCRM wing presented in this paper will be useful for low-fidelity aerostructural analysis and optimization of the CRM configuration.

Appendix

Table A1 Planform, twist, dihedral, and sweep information for the CRM and uCRM wing geometries.

		CRM					uCRM				
ξ	chord, m	twist, deg	$x_{c/4}$, m	$z_{c/4}$, m	dihedral, deg	sweep, deg	twist, deg	$x_{c/4}$, m	$z_{c/4}$, m	dihedral, deg	sweep, deg
0.00	13.6161	6.7166	0.0000	0.0000	0.0000	29.7522	6.6338	0.0000	0.0000	0.0000	29.7522
0.10	11.8976	4.4402	1.7486	-0.2084	3.6709	29.9501	4.4145	1.7482	-0.2006	3.6709	30.0793
0.15	11.0384	3.6063	2.6197	-0.2934	2.4339	31.1690	3.7304	2.6188	-0.2785	2.2160	31.0457
0.20	10.1790	3.0131	3.5216	-0.3466	2.0360	31.6678	3.3105	3.5203	-0.3199	1.4119	31.6328
0.25	9.3197	2.2419	4.4283	-0.4032	2.2941	31.5728	2.7349	4.4268	-0.3580	1.4162	31.4833
0.30	8.4604	1.5252	5.3324	-0.4628	2.3945	31.7263	2.2480	5.3310	-0.3894	1.0428	31.6504
0.35	7.6010	0.9379	6.2367	-0.5254	2.4187	31.0666	1.9361	6.2353	-0.4106	0.5650	30.9032
0.37	7.2573	0.7635	6.5982	-0.5504	2.5002	32.9494	1.8787	6.5968	-0.4152	0.3120	32.8045
0.40	7.0416	0.4285	7.2154	-0.5923	3.0022	35.6796	1.7370	7.2141	-0.4186	0.2433	35.4894
0.45	6.6821	-0.2621	8.2440	-0.6796	3.6606	34.8209	1.3592	8.2430	-0.4263	0.1905	34.6167
0.50	6.3226	-0.6782	9.2724	-0.7791	4.1806	35.0526	1.2762	9.2716	-0.4268	-0.0358	34.8329
0.55	5.9631	-0.9436	10.3009	-0.8954	4.8250	34.9914	1.3784	10.3001	-0.4249	-0.1792	34.7655
0.60	5.6035	-1.2067	11.3293	-1.0261	5.3523	35.0048	1.5129	11.3286	-0.4154	-0.5727	34.7735
0.65	5.2440	-1.4526	12.3578	-1.1722	6.0639	35.0039	1.6816	12.3570	-0.3970	-0.7911	34.7685
0.70	4.8845	-1.6350	13.3863	-1.3388	6.8534	35.0048	1.9304	13.3855	-0.3753	-0.9495	34.7653
0.75	4.5250	-1.8158	14.4147	-1.5250	7.6114	34.9974	2.1837	14.4139	-0.3474	-1.2104	34.7518
0.80	4.1654	-2.0301	15.4429	-1.7318	8.4165	34.9986	2.3788	15.4421	-0.3141	-1.3639	34.7448
0.85	3.8059	-2.2772	16.4713	-1.9586	9.1030	35.0031	2.4860	16.4706	-0.2774	-1.5117	34.7463
0.90	3.4464	-2.5773	17.4997	-2.2021	9.7474	35.0034	2.4437	17.4992	-0.2377	-1.5156	34.7378
0.95	3.0869	-3.1248	18.5280	-2.4608	10.1231	35.0020	2.0515	18.5281	-0.1998	-1.4942	34.7657
1.00	2.7274	-3.7500	19.5560	-2.7207	9.8390	34.9949	1.4465	19.5567	-0.1578	-1.8418	34.8792

Table A2 Multidimensional linear fit coefficients for the lift coefficient produced by the airfoil sections of the CRM/uCRM wing as a function of angle of attack, Mach number, and Reynolds number.

	$a_{L,000}$	$a_{L,001}$	$a_{L,010} \times 10^7$	$a_{L,011} \times 10^7$	$a_{L,100}$	$a_{L,101}$	$a_{L,110}$	$a_{L,111} \times 10^7$
$\xi = 0.0$	-0.5682	0.6200	0.0000	0.0000	17.9513	-11.5657	0.0000	-0.3167
$\xi = 0.1$	-0.5122	0.5225	0.0000	0.0000	16.6136	-9.6691	0.0000	-0.5619
$\xi = 0.15$	-0.2953	0.3401	0.0000	0.0000	15.8984	-8.5657	0.0000	-0.5623
$\xi = 0.2$	-0.1624	0.2535	0.0000	0.0000	14.5488	-6.8159	0.0000	-0.6538
$\xi = 0.25$	-0.0191	0.1564	0.0000	0.0000	14.2191	-6.4702	0.0000	-0.6588
$\xi = 0.3$	0.1917	0.0405	0.0000	0.0000	13.9930	-6.2138	0.0000	-0.8402
$\xi = 0.35$	0.3995	-0.1062	0.0000	0.0000	14.6710	-7.2700	0.0000	-1.0286
$\xi = 0.37$	0.4055	-0.0629	0.0000	0.0000	13.5411	-5.6631	0.0000	-1.1746
$\xi = 0.4$	0.3910	0.0604	0.0000	0.0000	11.6430	-2.9371	0.0000	-1.1959
$\xi = 0.45$	0.4536	0.0346	0.0000	0.0000	12.4849	-4.2587	0.0000	-1.3269
$\xi = 0.5$	0.5343	-0.0163	0.0000	0.0000	12.3558	-4.1372	0.0000	-1.3607
$\xi = 0.55$	0.6403	-0.1218	0.0000	0.0000	12.6962	-4.5681	0.0000	-1.5118
$\xi = 0.6$	0.7591	-0.4458	0.0000	0.0000	14.1389	-6.0314	0.0000	-0.3449
$\xi = 0.65$	0.6512	-0.0900	0.0000	-0.1144	12.8697	-4.8439	0.0000	-1.7401
$\xi = 0.7$	0.6401	-0.0513	0.1078	-0.1431	12.9864	-5.0611	0.0000	-1.9190
$\xi = 0.75$	0.6826	-0.0936	0.1275	-0.1692	13.0417	-5.1567	0.0000	-1.9620
$\xi = 0.8$	0.6481	-0.0800	0.1199	-0.1592	12.8176	-4.7793	0.0000	-2.1277
$\xi = 0.85$	0.5828	-0.0685	0.0000	-0.1312	12.1771	-3.9081	0.0000	-2.1490
$\xi = 0.9$	0.4479	0.0543	0.1226	-0.1573	11.8963	-3.4408	0.0000	-2.4484
$\xi = 0.95$	0.3823	-0.0065	0.1020	-0.1213	11.5509	-2.8649	0.0000	-3.3893
$\xi = 1.0$	-0.1634	0.0953	0.0000	0.0000	9.8103	-0.3978	0.0000	-2.2602

Table A3 Multidimensional linear fit coefficients for the moment coefficient produced by the airfoil sections of the CRM/uCRM wing as a function of angle of attack, Mach number, and Reynolds number.

	$a_{m,000}$	$a_{m,001}$	$a_{m,010}$	$a_{m,011}$	$a_{m,100}$	$a_{m,101}$	$a_{m,110} \times 10^7$	$a_{m,111} \times 10^7$
$\xi = 0.0$	0.0812	-0.0508	0.0000	0.0000	0.9662	-3.4029	-0.1503	0.2010
$\xi = 0.1$	0.0691	-0.0270	0.0000	0.0000	1.4133	-3.9689	-0.2155	0.2819
$\xi = 0.15$	0.0429	-0.0384	0.0000	0.0000	1.6716	-4.3351	-0.1975	0.2640
$\xi = 0.2$	0.0366	-0.0686	0.0000	0.0000	2.0827	-4.8171	-0.2060	0.2765
$\xi = 0.25$	0.0373	-0.1051	0.0000	0.0000	2.1343	-4.7903	-0.1977	0.2658
$\xi = 0.3$	0.0358	-0.1704	0.0000	0.0000	2.1116	-4.7034	-0.2533	0.3344
$\xi = 0.35$	0.0034	-0.1788	0.0000	0.0000	1.7029	-4.0720	-0.3182	0.4160
$\xi = 0.37$	0.0257	-0.2308	0.0000	0.0000	2.0679	-4.6199	-0.3125	0.4192
$\xi = 0.4$	0.0489	-0.3027	0.0000	0.0000	2.4463	-5.1866	-0.2526	0.3487
$\xi = 0.45$	0.0447	-0.3240	0.0000	0.0000	2.1015	-4.6356	-0.2864	0.3929
$\xi = 0.5$	0.0347	-0.3319	0.0000	0.0000	2.1297	-4.6489	-0.3022	0.4155
$\xi = 0.55$	0.0171	-0.3219	0.0000	0.0000	2.0244	-4.5160	-0.3584	0.4890
$\xi = 0.6$	-0.0805	-0.1333	0.0000	0.0000	1.8123	-4.4083	0.0000	0.1380
$\xi = 0.65$	0.0200	-0.3453	0.0000	0.0000	1.8812	-4.3116	-0.3905	0.5359
$\xi = 0.7$	0.0311	-0.3680	0.0000	0.0000	1.8485	-4.2271	-0.4750	0.6417
$\xi = 0.75$	0.0197	-0.3589	0.0000	0.0000	1.7854	-4.1353	-0.4755	0.6466
$\xi = 0.8$	0.0153	-0.3430	0.0000	0.0000	1.8318	-4.2358	-0.4454	0.6232
$\xi = 0.85$	0.0151	-0.3151	0.0000	0.0000	2.1351	-4.6335	-0.4886	0.6714
$\xi = 0.9$	0.0430	-0.3330	0.0000	0.0000	2.1912	-4.7459	-0.4881	0.6931
$\xi = 0.95$	0.0200	-0.2460	0.0000	0.0000	2.4578	-5.1924	-0.6767	0.9233
$\xi = 1.0$	-0.0162	0.0536	0.0000	0.0000	2.9536	-5.9138	-0.3814	0.5391

Table A4 Fit coefficients for the α^0 terms in the multidimensional parabolic fit for the drag coefficient produced by the airfoil sections of the CRM/uCRM wing as a function of angle of attack, Mach number, and Reynolds number.

	$a_{D,000}$	$a_{D,001}$	$a_{D,002}$	$a_{D,010}\times10^7$	$a_{D,011}\times10^7$	$a_{D,012}\times10^7$	$a_{D,020}$	$a_{D,021}$	$a_{D,022}$
$\xi = 0.0$	-0.3249	0.4505	0.1282	0.1428	-0.3937	0.2596	0.0000	0.0000	0.0000
$\xi = 0.1$	-0.1666	0.0736	0.3187	0.1303	-0.3663	0.2444	0.0000	0.0000	0.0000
$\xi = 0.15$	-0.0016	-0.3656	0.5776	0.1648	-0.4587	0.3072	0.0000	0.0000	0.0000
$\xi = 0.2$	0.0985	-0.6075	0.7063	0.1787	-0.4872	0.3196	0.0000	0.0000	0.0000
$\xi = 0.25$	0.1386	-0.6857	0.7275	0.1889	-0.5273	0.3541	0.0000	0.0000	0.0000
$\xi = 0.3$	0.1824	-0.7615	0.7473	0.0000	-0.2667	0.1812	0.0000	0.0000	0.0000
$\xi = 0.35$	0.2710	-0.9996	0.8964	0.0000	0.1840	-0.1239	0.0000	0.0000	0.0000
$\xi = 0.37$	0.1886	-0.7740	0.7525	0.1716	-0.4776	0.3199	0.0000	0.0000	0.0000
$\xi = 0.4$	-6.3579	18.4320	-13.0157	8.6931	-25.2829	17.9795	0.0000	0.0000	0.0000
$\xi = 0.45$	0.0336	-0.3156	0.4274	0.4223	-1.2122	0.8444	0.0000	0.0000	0.0000
$\xi = 0.5$	0.3539	-1.2428	1.0813	0.0000	0.1835	-0.1293	0.0000	0.0000	0.0000
$\xi = 0.55$	0.3909	-1.3550	1.1635	0.0000	0.2156	-0.1511	0.0000	0.0000	0.0000
$\xi = 0.6$	0.2268	-0.8302	0.7671	0.1170	-0.4489	0.3666	0.0000	0.0000	0.0000
$\xi = 0.65$	0.2699	-1.0076	0.9283	0.1245	-0.3463	0.2242	0.0000	0.0000	0.0000
$\xi = 0.7$	-0.0721	-0.0040	0.2055	0.7014	-2.0130	1.4059	0.0000	0.0000	0.0000
$\xi = 0.75$	0.0143	-0.2529	0.3815	0.6093	-1.7422	1.2097	0.0000	0.0000	0.0000
$\xi = 0.8$	0.1697	-0.7273	0.7321	0.3889	-1.0702	0.7154	0.0000	0.0000	0.0000
$\xi = 0.85$	0.2188	-0.8469	0.7993	0.3805	-1.0834	0.7451	0.0000	0.0000	0.0000
$\xi = 0.9$	0.2261	-0.8748	0.8224	0.3380	-0.9252	0.6139	0.0000	0.0000	0.0000
$\xi = 0.95$	0.2908	-1.0783	0.9584	0.3528	-1.0519	0.7431	0.0000	0.0000	0.0000
$\xi = 1.0$	0.5669	-1.8816	1.5356	-0.1123	0.2966	-0.2110	0.0000	0.0000	0.0000

Table A5 Fit coefficients for the α^1 terms in the multidimensional parabolic fit for the drag coefficient produced by the airfoil sections of the CRM/uCRM wing as a function of angle of attack, Mach number, and Reynolds number.

	$a_{D,100}$	$a_{D,101}$	$a_{D,102}$	$a_{D,110}\times10^7$	$a_{D,111}\times10^7$	$a_{D,112}\times10^7$	$a_{D,120}$	$a_{D,121}$	$a_{D,122}$
$\xi = 0.0$	3.4522	-9.5715	6.3852	-0.3702	0.7616	-0.3849	0.0000	0.0000	0.0000
$\xi = 0.1$	3.7241	-10.5048	7.1116	-0.2785	0.6081	-0.3319	0.0000	0.0000	0.0000
$\xi = 0.15$	2.3719	-6.7957	4.6910	-0.2334	0.5091	-0.2763	0.0000	0.0000	0.0000
$\xi = 0.2$	1.8475	-5.0214	3.3820	-0.5856	1.3866	-0.8163	0.0000	0.0000	0.0000
$\xi = 0.25$	1.7373	-4.3757	2.8109	-1.0672	2.6945	-1.6907	0.0000	0.0000	0.0000
$\xi = 0.3$	0.3988	-0.2622	-0.1073	0.2670	-0.7818	0.5436	0.0000	0.0000	0.0000
$\xi = 0.35$	-0.6120	3.0560	-2.5606	0.0000	0.2133	-0.1314	0.0000	0.0000	0.0000
$\xi = 0.37$	-0.5310	2.9002	-2.4828	-0.5517	1.4372	-0.9284	0.0000	0.0000	0.0000
$\xi = 0.4$	7.2507	-19.4816	13.4060	-11.4235	32.7342	-23.0312	0.0000	0.0000	0.0000
$\xi = 0.45$	-2.0364	7.5068	-5.7658	0.3047	-0.8602	0.5854	0.0000	0.0000	0.0000
$\xi = 0.5$	-2.3027	8.3959	-6.4317	-0.1408	0.4136	-0.2967	0.0000	0.0000	0.0000
$\xi = 0.55$	-2.8246	10.1164	-7.7435	0.0000	0.1733	-0.1277	0.0000	0.0000	0.0000
$\xi = 0.6$	-0.1861	2.4692	-2.5041	-0.2853	0.6275	-0.3560	0.0000	0.0000	0.0000
$\xi = 0.65$	-2.2169	8.4434	-6.5833	-1.5703	4.5294	-3.1899	0.0000	0.0000	0.0000
$\xi = 0.7$	-3.0602	10.8701	-8.2789	-0.8040	2.2813	-1.5773	0.0000	0.0000	0.0000
$\xi = 0.75$	-1.6789	6.8642	-5.4344	-3.6051	10.5826	-7.5465	0.0000	0.0000	0.0000
$\xi = 0.8$	-2.7125	9.8408	-7.5601	-0.9386	2.8840	-2.1176	0.0000	0.0000	0.0000
$\xi = 0.85$	-2.2931	8.4391	-6.5175	-0.2211	0.5277	-0.3305	0.0000	0.0000	0.0000
$\xi = 0.9$	-1.4576	5.5134	-4.2259	-1.2781	3.7967	-2.7297	0.0000	0.0000	0.0000
$\xi = 0.95$	-0.9530	4.0416	-3.2442	-0.6794	1.8443	-1.2332	0.0000	0.0000	0.0000
$\xi = 1.0$	2.1107	-6.6056	4.8527	5.5156	-14.8095	9.8225	0.0000	0.0000	0.0000

Table A6 Fit coefficients for the α^2 terms in the multidimensional parabolic fit for the drag coefficient produced by the airfoil sections of the CRM/uCRM wing as a function of angle of attack, Mach number, and Reynolds number.

	$a_{D,200}$	$a_{D,201}$	$a_{D,202}$	$a_{D,210}\times10^7$	$a_{D,211}\times10^7$	$a_{D,212}\times10^7$	$a_{D,220}$	$a_{D,221}$	$a_{D,222}$
$\xi = 0.0$	-29.3175	125.2950	-96.5283	-12.0835	31.1859	-19.7837	0.0000	0.0000	0.0000
$\xi = 0.1$	-38.2340	147.1626	-109.6929	-10.0881	26.6275	-17.1962	0.0000	0.0000	0.0000
$\xi = 0.15$	-39.8641	151.8656	-112.8863	-11.5079	30.4493	-19.7782	0.0000	0.0000	0.0000
$\xi = 0.2$	-41.8532	155.3432	-114.2371	-13.1423	34.4909	-22.2512	0.0000	0.0000	0.0000
$\xi = 0.25$	-39.9437	148.3293	-108.5422	-12.7401	33.6001	-21.7748	0.0000	0.0000	0.0000
$\xi = 0.3$	-44.7487	159.6118	-115.1512	-9.0101	24.1681	-15.8819	0.0000	0.0000	0.0000
$\xi = 0.35$	-50.0983	173.9021	-124.4604	2.0535	-5.7130	3.9051	0.0000	0.0000	0.0000
$\xi = 0.37$	-45.2050	162.1663	-117.8794	-12.1454	32.9826	-21.9460	0.0000	0.0000	0.0000
$\xi = 0.4$	-7.3546	52.8462	-40.6669	-60.1637	173.4676	-122.7107	0.0000	0.0000	0.0000
$\xi = 0.45$	-42.5019	155.0687	-113.7193	-12.5075	35.3122	-24.3604	0.0000	0.0000	0.0000
$\xi = 0.5$	-48.4733	172.0750	-125.6363	0.6943	-2.0402	1.4810	0.0000	0.0000	0.0000
$\xi = 0.55$	-52.5214	184.7772	-135.0893	1.7306	-4.9505	3.4937	0.0000	0.0000	0.0000
$\xi = 0.6$	-36.9600	140.6751	-104.1685	-10.0047	31.1780	-23.0141	0.0000	0.0000	0.0000
$\xi = 0.65$	-47.2434	170.4245	-125.6433	-11.8329	33.0339	-22.4416	0.0000	0.0000	0.0000
$\xi = 0.7$	-37.3132	141.1789	-104.6604	-29.4374	84.2729	-58.8640	0.0000	0.0000	0.0000
$\xi = 0.75$	-20.4532	91.6385	-69.2266	-62.8641	181.0406	-127.2507	0.0000	0.0000	0.0000
$\xi = 0.8$	-45.3995	165.2642	-122.0605	-17.1845	46.2002	-30.6045	0.0000	0.0000	0.0000
$\xi = 0.85$	-45.2908	163.5222	-119.9219	-12.0494	31.0498	-19.8488	0.0000	0.0000	0.0000
$\xi = 0.9$	-48.6533	172.4732	-125.8618	-8.7429	21.9033	-13.5197	0.0000	0.0000	0.0000
$\xi = 0.95$	-47.6118	169.3364	-123.4272	-18.0857	51.5001	-35.5157	0.0000	0.0000	0.0000
$\xi = 1.0$	-53.4387	183.5192	-132.0227	11.2253	-29.6558	19.8096	0.0000	0.0000	0.0000

Table A7 Location and thickness data for the leading-edge spar, trailing-edge spar, upper and lower skins, and ribs of the uCRM-9 wingbox geometry.

Sec	ξ_{LE}	x_{LE}/c	t_{LE}, mm	h_{LE}, m	ξ_{TE}	x_{TE}/c	t_{TE}, mm	h_{TE}, m	t_{US}, mm	t_{LS}, mm	t_{rib}, mm
0	0.0000	0.2524	1.4396	2.0371	0.0000	0.6936	0.6517	1.2970	1.8226	2.0274	0.4526
1	0.0264	0.2178	1.2204	1.8564	0.0264	0.6730	0.6865	1.2942	1.9227	2.1048	0.4905
2	0.0527	0.1806	0.9505	1.6504	0.0527	0.6509	0.8352	1.2858	2.0233	2.1826	0.5402
3	0.0791	0.1407	0.9639	1.4163	0.0791	0.6272	1.0439	1.2711	2.1246	2.2609	0.5652
4	0.1055	0.0978	0.7299	1.1415	0.1055	0.6019	-	1.2473	1.7945	1.7093	1.2409
5	0.1258	0.1016	0.5576	1.0797	0.1055	0.2301	-	1.2438	1.8025	1.6849	0.4526
6	0.1462	0.1056	0.5620	1.0193	0.1055	0.3596	-	1.2407	1.8990	1.7610	0.4526
7	0.1665	0.1089	0.5608	0.9603	0.1055	0.4891	1.2544	1.2372	1.9048	1.8375	0.4526
8	0.1868	0.1123	0.5406	0.9049	0.1084	0.6026	0.8677	1.2320	1.8208	1.7735	0.4812
9	0.2072	0.1157	0.5333	0.8540	0.1310	0.6090	0.6028	1.1220	1.7238	1.6970	0.4724
10	0.2275	0.1193	0.5333	0.8072	0.1535	0.6157	0.5333	1.0316	1.6285	1.6210	0.4568
11	0.2478	0.1232	0.5333	0.7642	0.1760	0.6219	0.5333	0.9573	1.5728	1.5717	0.4526
12	0.2681	0.1274	0.5333	0.7250	0.1986	0.6286	0.5333	0.8916	1.6122	1.6478	0.4526
13	0.2885	0.1319	0.5333	0.6893	0.2211	0.6356	0.5333	0.8320	1.6603	1.7243	0.4526
14	0.3088	0.1369	0.5333	0.6568	0.2436	0.6432	0.5333	0.7777	1.7169	1.8012	0.4526
15	0.3291	0.1422	0.7333	0.6273	0.2662	0.6515	0.5496	0.7271	1.8043	1.8786	0.4526
16	0.3495	0.1481	1.1200	0.6009	0.2887	0.6607	0.7334	0.6788	1.8943	1.9564	0.5286
17	0.3698	0.1545	1.1316	0.5777	0.3112	0.6707	0.5401	0.6310	1.8988	2.0018	1.0926
18	0.3901	0.1570	0.7449	0.5639	0.3338	0.6818	0.5444	0.5817	1.9125	1.9734	0.7751
19	0.4104	0.1596	0.5937	0.5514	0.3563	0.6941	0.7333	0.5303	1.8404	1.8962	0.4709
20	0.4308	0.1579	0.6146	0.5398	0.3786	0.7016	0.7333	0.4903	1.7776	1.8195	0.5069
21	0.4511	0.1650	0.6208	0.5287	0.4004	0.7002	0.6165	0.4761	1.7498	1.7432	0.4526
22	0.4714	0.1680	0.6158	0.5174	0.4223	0.6988	0.5991	0.4635	1.7042	1.6782	0.4526
23	0.4918	0.1710	0.6120	0.5064	0.4441	0.6972	0.5984	0.4514	1.6480	1.6164	0.4526
24	0.5121	0.1742	0.6059	0.4967	0.4660	0.6957	0.5970	0.4400	1.5790	1.5490	0.4526
25	0.5324	0.1775	0.6047	0.4880	0.4878	0.6941	0.5967	0.4284	1.5092	1.4767	0.4526
26	0.5528	0.1810	0.6010	0.4797	0.5096	0.6924	0.5952	0.4171	1.4390	1.4025	0.4526
27	0.5731	0.1847	0.6332	0.4715	0.5315	0.6906	0.5978	0.4065	1.3698	1.3289	0.4526
28	0.5934	0.1886	0.5927	0.4634	0.5533	0.6888	0.5942	0.3964	1.2944	1.2556	0.4526
29	0.6137	0.1927	0.5908	0.4553	0.5752	0.6868	0.5891	0.3860	1.2179	1.1828	0.4526
30	0.6341	0.1970	0.5794	0.4472	0.5970	0.6848	0.5847	0.3755	1.1438	1.1105	0.4526
31	0.6544	0.2015	0.5719	0.4385	0.6189	0.6826	0.5794	0.3653	1.0954	1.0385	0.4526
32	0.6747	0.2063	0.5631	0.4291	0.6407	0.6804	0.5746	0.3555	1.0564	0.9670	0.4526
33	0.6951	0.2114	0.5555	0.4194	0.6626	0.6779	0.5680	0.3460	1.0253	0.8960	0.4526
34	0.7154	0.2168	0.5489	0.4098	0.6844	0.6754	0.5616	0.3370	0.9923	0.8254	0.4526
35	0.7357	0.2226	0.5438	0.4002	0.7063	0.6726	0.5549	0.3284	0.9589	0.7552	0.4526
36	0.7560	0.2287	0.5333	0.3904	0.7281	0.6697	0.5470	0.3199	0.9192	0.6854	0.4526
37	0.7764	0.2353	0.5333	0.3805	0.7500	0.6666	0.5393	0.3109	0.8800	0.6162	0.4526
38	0.7967	0.2423	0.5333	0.3703	0.7718	0.6633	0.5333	0.3005	0.8378	0.5691	0.4526
39	0.8170	0.2498	0.5333	0.3599	0.7936	0.6597	0.5333	0.2894	0.7925	0.5337	0.4526
40	0.8374	0.2579	0.5333	0.3492	0.8155	0.6558	0.5333	0.2786	0.7437	0.4982	0.4526
41	0.8577	0.2666	0.5333	0.3382	0.8373	0.6516	0.5333	0.2682	0.6949	0.4574	0.4526
42	0.8780	0.2760	0.5333	0.3269	0.8592	0.6470	0.5333	0.2583	0.6566	0.4153	0.4526
43	0.8984	0.2862	0.5333	0.3154	0.8810	0.6420	0.5333	0.2487	0.6114	0.3772	0.4526
44	0.9187	0.2973	0.5333	0.3038	0.9029	0.6365	0.5333	0.2386	0.5600	0.3762	0.4526
45	0.9390	0.3094	0.5333	0.2922	0.9247	0.6303	0.5333	0.2276	0.5086	0.3751	0.4526
46	0.9593	0.3209	0.5333	0.2808	0.9466	0.6236	0.5333	0.2167	0.4971	0.3741	0.4526
47	0.9797	0.3373	0.5333	0.2691	0.9684	0.6163	0.5333	0.2062	0.4956	0.3730	0.4526
48	1.0000	0.3536	0.5333	0.2560	1.0000	0.6039	0.5333	0.1890	0.4956	0.3730	0.4526

Table A8 Approximations for the locus of centers of gravity, elastic axis, and flexural and torsional stiffness of the uCRM-9 wingbox geometry.

section	ξ	x_{cg}/c	x_{ea}/c [24]	x_{ea}/c [41]	x_{ea}/c [42]	x_{ea}/c (avg)	EI, N.m^2 ×10^{-9}	GJ, N.m^2/rad ×10^{-9}
0	0.0000	0.4507	0.4214	-	-	0.4214	10.0052	7.6883
1	0.0264	0.4287	0.4024	-	-	0.4024	9.2733	7.2357
2	0.0527	0.4057	0.3844	-	-	0.3844	10.8361	7.6534
3	0.0791	0.3798	0.3690	-	-	0.3690	9.2963	6.6712
4	0.1055	0.3677	0.3595	0.4046	0.4071	0.3904	7.4630	5.0283
5	0.1139	0.3648	0.3709	0.4070	0.4121	0.3967	6.8888	4.6576
6	0.1212	0.3622	0.3808	0.4091	0.4165	0.4021	6.4375	4.3570
7	0.1295	0.3592	0.3921	0.4115	0.4185	0.4074	6.0379	4.1393
8	0.1416	0.3305	0.4086	0.4153	0.4210	0.4150	5.4871	3.8367
9	0.1639	0.3812	0.4097	0.4216	0.4265	0.4193	4.7280	3.3576
10	0.1860	0.3829	0.4116	0.4279	0.4324	0.4240	4.1395	2.9743
11	0.2079	0.3873	0.4147	0.4345	0.4354	0.4282	3.5313	2.5800
12	0.2298	0.3928	0.4182	0.4415	0.4413	0.4337	2.6721	2.1271
13	0.2516	0.3988	0.4221	0.4490	0.4469	0.4393	2.2503	1.8254
14	0.2735	0.4053	0.4265	0.4572	0.4533	0.4457	2.0076	1.6216
15	0.2953	0.4117	0.4311	0.4662	0.4620	0.4531	1.7067	1.4146
16	0.3172	0.4174	0.4358	0.4760	0.4751	0.4623	1.4977	1.2288
17	0.3392	0.4174	0.4403	0.4867	0.4845	0.4705	1.2962	1.0718
18	0.3615	0.4229	0.4424	0.4988	0.4972	0.4794	1.2084	0.9685
19	0.3839	0.4398	0.4351	0.5022	0.5043	0.4805	1.0475	0.8555
20	0.4059	0.4399	0.4237	0.4999	0.5056	0.4764	0.9188	0.7550
21	0.4271	0.4392	0.4255	0.4975	0.5064	0.4765	0.7754	0.6575
22	0.4482	0.4392	0.4256	0.4950	0.4987	0.4731	0.6820	0.5923
23	0.4693	0.4397	0.4257	0.4924	0.4939	0.4707	0.6197	0.5360
24	0.4904	0.4404	0.4258	0.4897	0.4895	0.4683	0.5735	0.4874
25	0.5116	0.4412	0.4255	0.4868	0.4849	0.4657	0.5247	0.4450
26	0.5327	0.4418	0.4252	0.4838	0.4801	0.4630	0.4615	0.3963
27	0.5538	0.4423	0.4252	0.4807	0.4789	0.4616	0.3986	0.3541
28	0.5749	0.4431	0.4252	0.4774	0.4757	0.4594	0.3355	0.3142
29	0.5961	0.4440	0.4253	0.4739	0.4715	0.4569	0.2917	0.2778
30	0.6172	0.4449	0.4254	0.4702	0.4672	0.4542	0.2601	0.2500
31	0.6383	0.4458	0.4257	0.4663	0.4591	0.4504	0.2088	0.2111
32	0.6593	0.4468	0.4265	0.4622	0.4477	0.4454	0.1864	0.1835
33	0.6804	0.4479	0.4275	0.4578	0.4381	0.4411	0.1453	0.1533
34	0.7014	0.4489	0.4287	0.4532	0.4343	0.4387	0.1243	0.1269
35	0.7224	0.4501	0.4302	0.4482	0.4386	0.4390	0.1033	0.1146
36	0.7435	0.4513	0.4319	0.4430	0.4305	0.4351	0.0932	0.1024
37	0.7645	0.4524	0.4335	0.4374	0.4176	0.4295	0.0716	0.0796
38	0.7855	0.4537	0.4349	0.4314	0.4034	0.4232	0.0621	0.0674
39	0.8066	0.4550	0.4363	0.4250	0.3904	0.4172	0.0621	0.0536
40	0.8277	0.4564	0.4380	0.4180	0.3874	0.4145	0.0397	0.0402
41	0.8487	0.4582	0.4401	0.4105	0.3734	0.4080	0.0311	0.0327
42	0.8697	0.4600	0.4427	0.4024	0.3658	0.4036	0.0311	0.0299
43	0.8907	0.4620	0.4457	0.3936	0.3630	0.4008	0.0311	0.0170
44	0.9117	0.4642	0.4489	0.3841	0.3530	0.3953	0.0311	0.0146
45	0.9328	0.4666	0.4521	0.3736	0.3401	0.3886	0.0311	0.0134
46	0.9538	0.4692	0.4547	0.3620	0.3259	0.3809	0.0311	0.0107
47	0.9748	0.4723	0.4600	0.3493	0.3025	0.3706	0.0311	0.0095
48	1.0000	0.4764	0.4597	0.3322	0.2708	0.3542	0.0311	0.0080

Acknowledgements

This material is partially based upon work supported by NASA under Grant No. 80NSSC18K1696 issued by the Aeronautics Research Mission Directorate through the 2018 NASA Fellowship Activity with Nhan Nguyen as the NASA Technical Advisor.

References

[1] Vassberg, J. C., DeHaan, M. A., Rivers, S. M., and Wahls, R. A., "Development of a Common Research Model for Applied CFD Validation Studies," AIAA 2008-6919, 26th AIAA Applied Aerodynamics Conference, Honolulu, HI, 18-21 August 2008. (doi:10.2514/6.2008-6919)

[2] Vassberg, J. C., DeHaan, M. A., Rivers, S. M., and Wahls, R. A., "Retrospective on the Common Research Model for Computational Fluid Dynamics Validation Studies," *Journal of Aircraft*, Vol. 55, No. 4, 2018, pp. 1325-1337. (doi:10.2514/1.C034906)

[3] Rivers, M. B. and Dittberner, A., "Experimental Investigation of the NASA Common Research Model (Invited)," AIAA 2010-4218, 28th AIAA Applied Aerodynamics Conference, Chicago, IL, 28 June-1 July 2010. (doi:10.2514/6.2010-4218)

[4] Rivers, M. B., Quest, J., and Rudnik, R., "Comparison of the NASA Common Research Model European Transonic Wind Tunnel Test Data to NASA Test Data (Invited)," AIAA 2015-1093, 53rd AIAA Aerospace Sciences Meeting, Kissimmee, FL, 5-9 January 2015. (doi:10.2514/6.2015-1093)

[5] Rivers, M. B., "NASA Common Research Model: A History and Future Plans," AIAA 2019-3725, AIAA Aviation 2019 Forum, Dallas, TX, 17-21 June 2019. (doi:10.2514/6.2019-3725)

[6] Development of the High Lift Common Research Model (HL-CRM): A Representative High Lift Configuration for Transonic Transports," AIAA 2016-0308, 24th AIAA Aerospace Sciences Meeting, San Diego, CA, 4-8 January 2016. (doi:10.2514/6.2016-0308)

[7] Lynde, M. N. and Campbell, R. L., "Computational Design and Analysis of a Transonic Natural Laminar Flow Wing for a Wind Tunnel Model," AIAA 2017-3058, 35th AIAA Applied Aerodynamics Conference, Denver, CO, 5-9 June 2017. (doi:10.2514/6.2017-3058)

[8] Rivers, M. B., Lynde, M. N., Campbell, R. L., Viken, S. A., Chan, D. T., Watkins, A. N., and Goodliff, S. L., "Experimental Investigation of the NASA Common Research Model with a Natural Laminar Flow Wing in the NASA Langley National Transonic Facility," AIAA 2019-2189, AIAA SciTech 2019 Forum, San Diego, CA, 7-11 January 2019. (doi:10.2514/6.2019-2189)

[9] Lynde, M. N., Campbell, R. L., and Viken, S. A., "Additional Findings from the Common Research Model Natural Laminar Flow Wind Tunnel Test," AIAA 2019-3292, AIAA Aviation 2019 Forum, Dallas, TX, 17-21 June 2019. (doi:10.2514/6.2019-3292)

[10] Atinault, O. and Hue, D., "Design of a vertical tail for the CRM configuration," ONERA RT 1/21960 GMT/DAAP, June 2014.

[11] Cartieri, A., Hue, D., Chanzy, Q., and Atinault, O., "Experimental Investigations on the Common Research Model at ONERA-SIMA – Comparison with DPW Numerical Results," AIAA 2017-0964, 55th AIAA Aerospace Sciences Meeting, Grapevine, TX, 9-13 January 2017, (doi:10.2514/6.2017-0964)

[12] Ueno, M., Kohzai, T., Koga, S., Kato, H., Nakakita, K., and Sudani, N., "80% Scaled NASA Common Research Model Wind Tunnel Tests in JAXA," AIAA 2013-0963, 51st AIAA Aerospace Sciences Meeting, Grapevine, TX., 7-10 January 2013. (doi:10.2514/6.2013-0963)

[13] Broughton, C. A., Bendmeddour, A., Mebarki, Y., and Rivers, M. B., "Experimental Investigations of the NASA common Research Semispan Model in the NRC 5-Foot Trisonic Wind Tunnel," AIAA 2018-4285, 2018 Aerodynamic Measurement Technology and Ground Testing Conference, Atlanta, GA, 25-29 June 2018. (doi:10.2514/6.2018-4285)

[14] Klimmek, T., "Development of a Structural Model of the CRM Configuration for Aeroelastic and Loads Analysis," *International Forum on Aeroelasticity and Structural Dynamics*, Royal Aeronautical Society., Bristol, England, U.K., June 2013, Paper 1794.

[15] Kennedy, G. J., Kenway, G. W., and Martins, J. R. R. A., "High Aspect Ratio Wing Design: Optimal Aerostructural Tradeoffs for the Next Generation of Materials," AIAA 2014-0596, 52nd Aerospace Sciences Meeting, National Harbor, MD, 13-17 January 2014. (doi:10.2514/6.2014-0596)

[16] Brooks, T. R., Kenway, G. K. W., and Martins, J. R. R. A., "Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings," *AIAA Journal*, Vol. 56, No. 7, July 2018 pp. 2840- 2855. (doi:10.2514/1.J056603)

[17] Keye, S. Brodersen, O., and Rivers, M. B., "Investigation of Aeroelastic Effects on the NASA Common Research Model," *Journal of Aircraft*, Vol, 51, No. 4, July-August 2014, pp. 1323-1330. (doi:10.2514/1.C032598)

[18] Vassberg, J. C., Tinoco, E. N., Mani, M., Rider, B., Zickuhr, T., Levy, D. W., Brodersen, O. P., Eisfeld, B., Crippa, S., Wahls, R. A., Morrison, J. H., Mavriplis, D. J., and Murayama, M., "Summary of the Fourth AIAA Computational Fluid Dynamics Drag Prediction Workshop," *Journal of Aircraft*, Vol. 51, No. 4, July-August 2014, pp. 1070-1089. (doi:10.2514/1.C032418)

[19] Keye, S. and Brodersen, O., "Investigations of Fluid-Structure Coupling and Turbulence Model Effects on the DLR Results of the Fifth AIAA CFD Drag Prediction Workshop", AIAA 2013-2509, 31st AIAA Applied Aerodynamics Conference, San Diego, CA, 24-27 June 2013. (doi:10.2514/6.2013-2509)

[20] Morrison, J., "Statistical Analysis of CFD Solutions From the Fifth AIAA Drag Prediction Workshop." AIAA 2013-47, 51st AIAA Aerospace Sciences Meeting, Grapevine, TX, 7-10 January 2013. (doi:10.2514/6.2013-47)

[21] Tinoco, E. N., Brodersen, O. P., Keye, S., Laflin, K. R., Feltrop, E., Vassberg, J. C., Mani, M., Rider, B., Wahls, R. A., Morrison, J. H.., Hue, D., Roy, C. J., Mavriplis, D. J., and Murayama, M., "Summary Data from the Sixth AIAA CFD Drag Prediction Workshop: CRM Cases," *Journal of Aircraft*, Vol. 55, No. 4, July-August 2018, pp. 1352-1379. (doi:10.2514/1.C034409)

[22] Keye, S. and Mavriplis, D., "Summary of Case 5 from Sixth Drag Prediction Workshop: Coupled Aerostructural Simulation," *Journal of Aircraft*, Vol. 55, No. 4, July-August 2018, pp. 1380-1387. (doi:10.2514/1.C034427)

[23] Hue, D., Chanzy, Q., and Landier, Sâm, "DPW-6: Drag Analyses and Increments Using Different Geometries of the Common Research Model Airliner," *Journal of Aircraft*, Vol. 55, No. 4, July-August 2018, pp. 1509-1521. (doi:10.2514/1.C034139)

[24] Chauhan, S. S. and Martins, J. R. R. A., "Low-fidelity aerostructural optimization of aircraft wings with a simplified wingbox model using OpenAeroStruct," *Proceedings of the 6th International Conference on Engineering Optimization*, EngOpt 2018, pp. 418-431. (doi:10.1007/978-3-319-97773-7_38)

[25] Wiberg, B. D., Fujiwara, G. E. C., Woodard, B., and Bragg, M., "Large-Scale Swept-Wing Icing Simulations in the NASA Glenn Icing Research Tunnel Using LEWICE3D," AIAA 2014-2617, 6th AIAA Atmospheric and Space Environments Conference, Atlanta, GA, 16-20 June 2014. (doi:10.2514/6.2014-2617)

[26] Koivisto, P., Soinne, E., and Kivekäs, J., "Anti-Icing Fluid Secondary Wave and Its Role in Lift Loss During Takeoff," *Journal of Aircraft*, Vol. 55, No. 6, November-December 2018, pp. 2298-2306. (doi:10.2514/1.C034694)

[27] Stanford, B. K. and Massey, S. J., "Uncertainty Quantification of the FUN3D-Predicted NASA CRM Flutter Boundary," AIAA 2017-1816, 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Grapevine TX, 9-13 January 2017. (doi:10.2514/6.2017-1816)

[28] Fujiwara, G. E. C., Nguyen, N. T., Livne, E., and Bragg, M. B., "Aerostructural Design Optimization of a Flexible Wing Aircraft with Continuous Morphing Trailing Edge," AIAA 2018-3517, 2018 Multidisplinary Analysis and Optimization Conference, Atlanta, GA, 25-29 June 2018. (doi:10.2514/6.2018-3517)

[29] 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)

[30] 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)

[31] Gallay, S., and Laurendeau, E., "Preliminary-Design Aerodynamic Model for Complex Configurations Using Lifting-Line Coupling Algorithm," *Journal of Aircraft*, Vol. 53, No. 4, 2016, pp. 1145–1159. (doi:10.2514/1.C033460)

[32] Phillips, W. F., and Hunsaker, D. F., "Lifting-Line Predictions for Induced Drag and Lift in Ground Effect," *Journal of Aircraft*, Vol. 50, No. 4, 2013, pp. 1226–1233. (doi:10.2514/1.C032152)

[33] Wickenheiser, A., and Garcia, E., "Aerodynamic Modeling of Morphing Wings Using an Extended Lifting-Line Analysis," *Journal of Aircraft*, Vol. 44, No. 1, 2007, pp. 10–16. (doi:10.2514/1.18323)

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

[35] Rasmussen, M. L., and Smith, D. E., "Lifting-Line Theory for Arbitrarily Shaped Wings," *Journal of Aircraft*, Vol. 36, No. 2, 1999, pp. 340–348. (doi:10.2514/2.2463)

[36] Bera, R. K., "Some remarks on the solution of the lifting line equation," *Journal of Aircraft*, Vol. 11, No. 10, 1974, pp. 647–648. (doi:10.2514/3.44397)

[37] Prandtl, L., "Tragflügel Theorie," *Nachricten von der Gesellschaft der Wissenschaften zu Göttingen*, Geschäeftliche Mitteilungen, Klasse, 1918, pp. 451–477.

[38] Prandtl, L., "Applications of Modern Hydrodynamics to Aeronautics," NACA TR-116, June 1921.

[39] Fujiwara, G. E. C., Chaparro, D., and Nguyen, N., "An Integral Boundary Layer Direct Method Applied to 2D Transonic Small-Disturbance Equations," AIAA 2016-3568, 34th AIAA Applied Aerodynamics Conference, Washington, D. C., 13-17 June 2016. (doi:10.2514/6.2016-3568)

[40] Ullah, A. H., Fabijanic, C., Estevadeordal, J., Montgomery, Z. S., Hunsaker, D. F., Staiger, J. M., and Joo, J. J., "Experimental and Numerical Evaluation of the Performance of Parabolic Flaps," AIAA 2019-2916, AIAA Aviation 2019 Forum, Dallas, Texas, 17-21 June 2019. (doi:10.2514/6.2019-2916)

[41] Cramer, N. B. and Nguyen, N. T., "Development of an Aeroservoelastic Model for Gust Load Alleviation of the NASA Common Research Model Wind Tunnel Experiment," AIAA 2020-0211, AIAA SciTech 2020 Forum, Orlando, Florida, 6-10 January 2020. (doi:10.2514/6.2020-0211)

[42] Stodieck, O., Cooper, J. E., and Weaver, P. M., "Interpretation of Bending/Torsion Coupling for Swept, Nonhomogenous Wings," *Journal of Aircraft*, Vol. 53, No. 4, 2016, pp. 892-899. (doi:10.2514/1.C033186)

[43] "777-200/300 Airplane Characteristics for Airport Planning", D6-58329, Boeing Commercial Airplanes, July, 1998.

[44] "Boeing 777 Airplane Rescue and Fire Fighting Information," Boeing Commercial Airplanes, October, 2018.

[45] Taylor, J. D. and Hunsaker, D. F., "Numerical Method for Rapid Aerostructural Design and Optimization," AIAA 2020-3175, AIAA Aviation 2020 Virtual Forum, 15-19 June, 2020. (doi:10.2514/6.2020-3175)

[46] Reid, J. T., and Hunsaker, D. F., "A General Approach to Lifting-Line Theory, Applied to Wings with Sweep," AIAA 2020-1287, AIAA SciTech 2020 Forum, Orlando, Florida, 6-10 January, 2020. (doi:10.2514/6.2020-1287)

[47] Lyu, Z., Kenway, G. K., and Martins, J. R. R. A., "RANS-based Aerodynamic Shape Optimization Investigations of the Common Research Model Wing," AIAA 2014-0567, 52nd Aerospace Sciences Meeting, National Harbor, Maryland, 13-17 January, 2014. (doi:10.2514/6.2014-0567)

APPENDIX B

Simplified Trajectory Optimization Formulations Using Optimal Control Theory and the Calculus of Variations

B.1 Mission Profile Optimization*

Consider an aircraft having an engine with the thrust-specific fuel consumption c. The thrust-specific fuel consumption is defined as

$$c = \frac{\Delta \dot{W}_f}{T} \tag{B1}$$

where $\Delta \dot{W}_f$ is the total weight of fuel consumed per unit time and T is the engine thrust. Rearranging Eq. (B1) to solve for $\Delta \dot{W}_f$ and integrating from time $t = 0$ to $t = t_f$ gives the total weight of fuel consumed over the interval $0 \le t \le t_f$, i.e.,

$$\Delta W_f = \int_0^{t_f} cT dt \tag{B2}$$

In general, c depends on the altitude h, the throttle setting τ, and the airspeed V, i.e., $c = c(h, \rho, \tau)$. If we wish to minimize the fuel burn, then we consider the minimum-fuel optimal control problem

* The formulations in this section are based on notes from Dr. Nhan T. Nguyen, Senior Research Scientist and Technical Group Lead of the Advanced Control and Evolvable Systems Group in the Intelligent Systems Division at NASA Ames Research Center.

$$\min J = \Delta W_f = \int_0^{t_f} cT dt \tag{B3}$$

Here, J is the functional which we wish to minimize subject to dynamic constraints based on the equations of motion.

The equations of motion can be expressed according to the point-mass model in terms of the time rates of change of the climb angle $\dot{\gamma}$, velocity $\dot{V}$, altitude $\dot{h}$, and the weight $\dot{W}$. The climb angle is the angle between the velocity vector and the horizontal. Therefore, the time rate of change of the climb angle is related to the velocity V, the time rate of change of velocity in the direction perpendicular to V, the aerodynamic moment M, and the moment due to engine thrust, i.e.,

$$\dot{\gamma} = \frac{\dot{V}_t}{V} = \frac{L - W\cos\gamma + T\sin\alpha_T}{mV} + \frac{M + Tz_e}{m} \tag{B4}$$

where m is the aircraft mass, $\dot{V}_t$ is the time rate of change of the velocity in the direction perpendicular to V, L is the aircraft lift, W is the aircraft weight, α_T is the engine thrust angle, and z_e is the vertical distance from the engine centerline to the aircraft center of gravity. The time rate of change of velocity can be expressed as

$$\dot{V} = \frac{T\cos\alpha_T - D - W\sin\gamma}{m} \tag{B5}$$

The time rate of change of altitude is related to the velocity and the climb angle according to

$$\dot{h} = V\sin\gamma \tag{B6}$$

The time rate of change of weight is simply the fuel-consumption rate, i.e.,

$$\dot{W} = -cT \tag{B7}$$

For most aircraft, the thrust angle is small. Assuming that the thrust angle $\alpha_T = 0$, the thrust is aligned with the center of gravity ($z_e = 0$), and the aerodynamic moment $M = 0$, we can rewrite Eqs. (B4) and (B5) to give

$$\dot{\gamma} = \frac{L - W\cos\gamma}{mV} \tag{B8}$$

$$\dot{V} = \frac{T - D - W\sin\gamma}{m} \tag{B9}$$

Equations (B6)-(B9) are the equations of motion. Using the method of Lagrange multipliers, they are included as dynamic constraints to the minimization problem shown in Eq. (B3) to give the functional

$$\begin{aligned} J = \int_0^{t_f} \Bigg[& cT + \lambda_h (V\sin\gamma - \dot{h}) + \lambda_\gamma \left(\frac{L - W\cos\gamma}{mV} - \dot{\gamma} \right) \\ & + \lambda_V \left(\frac{T - D - W\sin\gamma}{m} - \dot{V} \right) + \lambda_W (-cT - \dot{W}) \Bigg] dt \end{aligned} \tag{B10}$$

B.1.1 Lift and Drag Relationships

Consider an aircraft equipped with a distributed wing-flap system. Assuming that the aerodynamic center lies at or very near the center of gravity, we can trim the aircraft by enforcing the condition

$$I_{zz}\dot{q} = M + Tz_e = 0 \tag{B11}$$

where I_{zz} is the second moment of inertia about the z axis, q is the pitch rate, M is the aerodynamic pitching moment, and z_e is the vertical offset of the engine thrust from the aircraft center of gravity. For a trimmed aircraft with zero initial pitch rate, this implies that the flight path angle θ, which is related to the angle of attack and climb angle according to

$$\gamma = \theta - \alpha \tag{B12}$$

is constant.

The lift coefficient is assumed to be linear with respect to angle of attack α, the wing flap deflections $\boldsymbol{\delta}$, and elevator deflection δ_e, i.e.,

$$C_L = C_{L_0} + C_{L,\alpha}\alpha + \mathbf{C}^{\mathrm{T}}_{\mathbf{L},\boldsymbol{\delta}}\boldsymbol{\delta} + C_{L,\delta_e}\delta_e \tag{B13}$$

where $\boldsymbol{\delta}$ is a vector containing the individual flap deflections. The pitching moment is also assumed to be linear with respect to α, $\boldsymbol{\delta}$, and δ_e, which gives

$$C_m = C_{m_0} + C_{m,\alpha}\alpha + \mathbf{C}^{\mathrm{T}}_{\mathbf{m},\boldsymbol{\delta}}\boldsymbol{\delta} + C_{m,\delta_e}\delta_e \tag{B14}$$

The drag coefficient is assumed to be parabolic with respect to α, $\boldsymbol{\delta}$, and δ_e. This gives the relation

$$\begin{aligned} C_D = C_{D_0} + C_{D,\alpha}\alpha + \mathbf{C}^{\mathrm{T}}_{\mathbf{D},\boldsymbol{\delta}}\boldsymbol{\delta} + C_{D,\delta_e}\delta_e + C_{D,\alpha^2}\alpha^2 + \boldsymbol{\delta}^{\mathrm{T}}\mathbf{C}_{\mathbf{D},\boldsymbol{\delta}^2}\boldsymbol{\delta} \\ + C_{D,\delta_e^2}\delta_e^2 + \mathbf{C}^{\mathrm{T}}_{\mathbf{D},\boldsymbol{\delta},\,\mathbf{e}}\boldsymbol{\delta}\delta_e + \mathbf{C}^{\mathrm{T}}_{\mathbf{D},\boldsymbol{\delta},}\boldsymbol{\delta}\alpha + C_{D,\alpha\delta_e}\alpha\delta_e \end{aligned} \tag{B15}$$

Using Eq. (B13), the trim condition from Eq. (B11) can be rewritten to give

$$\bar{q}S\left(C_{m_0}+C_{m,\alpha}\alpha+\mathbf{C}_{\mathbf{m,\delta}}^{\mathrm{T}}\boldsymbol{\delta}+C_{m,\delta_e}\delta_e\right)+Tz_e=0 \tag{B16}$$

where $\bar{q}$ is the dynamic pressure and S is the wing reference area. Solving for δ_e in Eq. (B16) gives the elevator deflection required to trim an aircraft with a given angle of attack and known wing flap deflections

$$\delta_e=-\frac{\frac{Tz_e}{\bar{q}S}+C_{m_0}+C_{m,\alpha}\alpha+\mathbf{C}_{\mathbf{m,\delta}}^{\mathrm{T}}\boldsymbol{\delta}}{C_{m,\delta_e}} \tag{B17}$$

Using Eq. (B17) with Eqs. (B13) and (B15), the trim lift and drag can be written as

$$L=\bar{q}SC_L=\bar{q}S\left[C_{L_0}+C_{L,\alpha}\alpha+\mathbf{C}_{\mathbf{L,\delta}}^{\mathrm{T}}\boldsymbol{\delta}+C_{L,\delta_e}\left(\frac{\frac{Tz_e}{\bar{q}S}+C_{m_0}+C_{m,\alpha}\alpha+\mathbf{C}_{\mathbf{m,\delta}}^{\mathrm{T}}\boldsymbol{\delta}}{C_{m,\delta_e}}\right)\right] \tag{B18}$$

$$\begin{aligned}D=\bar{q}SC_D=\bar{q}S\Big[&C_{D_0}+C_{D,\alpha}\alpha+\mathbf{C}_{\mathbf{D,\delta}}^{\mathrm{T}}\boldsymbol{\delta}+C_{D,\alpha^2}\alpha^2+\boldsymbol{\delta}^{\mathrm{T}}\mathbf{C}_{\mathbf{D,\delta^2}}^{\mathrm{T}}\boldsymbol{\delta}+\mathbf{C}_{\mathbf{D,\delta\alpha}}^{\mathrm{T}}\boldsymbol{\delta}\alpha\\&+\frac{C_{D,\delta_e}+C_{D,\alpha\delta_e}\alpha+\mathbf{C}_{\mathbf{D,\delta\delta_e}}^{\mathrm{T}}\boldsymbol{\delta}}{C_{m,\delta_e}}\left(\frac{Tz_e}{\bar{q}S}+C_{m_0}+C_{m,\alpha}\alpha+\mathbf{C}_{\mathbf{m,\delta}}^{\mathrm{T}}\boldsymbol{\delta}\right)\\&+\frac{C_{D,\delta_e^2}}{C_{m,\delta_e}^2}\left(\frac{Tz_e}{\bar{q}S}+C_{m_0}+C_{m,\alpha}\alpha+\mathbf{C}_{\mathbf{m,\delta}}^{\mathrm{T}}\boldsymbol{\delta}\right)^2\Big]\end{aligned} \tag{B19}$$

The derivatives of lift with respect to h, V, α, and $\boldsymbol{\delta}$ are

$$\frac{\partial L}{\partial h}=\left(\frac{\partial\bar{q}}{\partial h}+\frac{1}{C_L}\frac{\partial C_L}{\partial h}\right)L \tag{B20}$$

$$\frac{\partial L}{\partial V}=\left(\frac{1}{C_L}\frac{\partial C_L}{\partial V}+\frac{2}{V}\right)L \tag{B21}$$

$$\frac{\partial L}{\partial \alpha} = \frac{1}{C_L}\frac{\partial C_L}{\partial \alpha} L \tag{B22}$$

$$\frac{\partial L}{\partial \boldsymbol{\delta}} = \frac{1}{C_L}\frac{\partial C_L}{\partial \boldsymbol{\delta}} L \tag{B23}$$

The derivatives of drag with respect to h, V, α, and $\boldsymbol{\delta}$ are

$$\frac{\partial D}{\partial h} = \left(\frac{\partial \overline{q}}{\partial h} + \frac{1}{C_D}\frac{\partial C_D}{\partial h}\right) D \tag{B24}$$

$$\frac{\partial D}{\partial V} = \left(\frac{1}{C_D}\frac{\partial C_D}{\partial V} + \frac{2}{V}\right) D \tag{B25}$$

$$\frac{\partial D}{\partial \alpha} = \frac{1}{C_D}\frac{\partial C_D}{\partial \alpha} D \tag{B26}$$

$$\frac{\partial D}{\partial \boldsymbol{\delta}} = \frac{1}{C_D}\frac{\partial C_D}{\partial \boldsymbol{\delta}} D \tag{B27}$$

The partial derivatives of C_L and C_D with respect to h and V can be approximated using a finite difference scheme, i.e.,

$$\frac{\partial C_L}{\partial h} = \frac{C_L(h+\Delta h) - C_L(h)}{\Delta h} + \frac{C_{L,\delta_e}}{C_{m,\delta_e}}\frac{Tz_e}{\overline{q}^2 S}\frac{\partial \overline{q}}{\partial h} \tag{B28}$$

$$\frac{\partial C_L}{\partial V} = \frac{C_L(V+\Delta V) - C_L(V)}{\Delta V} + \frac{C_{L,\delta_e}}{C_{m,\delta_e}}\frac{Tz_e}{\overline{q} S}\frac{2}{V} \tag{B29}$$

$$\frac{\partial C_D}{\partial h} = \frac{C_D(h+\Delta h) - C_D(h)}{\Delta h} + \left(C_{D,\delta_e} + 2C_{D,\delta_e^2}\delta_e + \mathbf{C}^{\mathrm{T}}_{\mathbf{D,\delta\delta_e}}\boldsymbol{\delta} + C_{D,\alpha\delta_e}\alpha\right)\frac{1}{C_{m,\delta_e}}\frac{Tz_e}{\bar{q}^2 S}\frac{\partial \bar{q}}{\partial h} \tag{B30}$$

$$\frac{\partial C_D}{\partial V} = \frac{C_D(V+\Delta V) - C_D(V)}{\Delta V} + \left(C_{D,\delta_e} + 2C_{D,\delta_e^2}\delta_e + \mathbf{C}^{\mathrm{T}}_{\mathbf{D,\delta\delta_e}}\boldsymbol{\delta} + C_{D,\alpha\delta_e}\alpha\right)\frac{1}{C_{m,\delta_e}}\frac{Tz_e}{\bar{q}S}\frac{2}{V} \tag{B31}$$

The partial derivatives of C_L and C_D with respect to α and $\boldsymbol{\delta}$ are

$$\frac{\partial C_L}{\partial \alpha} = C_{L,\alpha} - \frac{C_{m,\alpha}}{C_{m,\delta_e}}C_{L,\delta_e} \tag{B32}$$

$$\frac{\partial C_L}{\partial \boldsymbol{\delta}} = \mathbf{C_{L,\delta}} - \frac{\mathbf{C_{m,\delta}}}{C_{m,\delta_e}}C_{L,\delta_e} \tag{B33}$$

$$\frac{\partial C_D}{\partial \alpha} = C_{D,\alpha} + C_{D,\alpha^2}\alpha + \mathbf{C}^{\mathrm{T}}_{\mathbf{D,\delta\alpha}}\boldsymbol{\delta} + C_{D,\alpha\delta_e}\delta_e - \frac{C_{m,\alpha}}{C_{m,\delta_e}}\left(C_{D,\delta_e} + 2C_{D,\delta_e^2}\delta_e + \mathbf{C}^{\mathrm{T}}_{\mathbf{D,\delta\delta_e}}\boldsymbol{\delta} + C_{D,\alpha\delta_e}\alpha\right) \tag{B34}$$

$$\begin{aligned}\frac{\partial C_D}{\partial \boldsymbol{\delta}} &= \mathbf{C_{D,\delta}} + 2\mathbf{C_{D,\delta^2}}\boldsymbol{\delta} + \mathbf{C_{D,\delta\delta_e}}\delta_e + \mathbf{C_{D,\alpha\delta}}\alpha\frac{\partial C_D}{\partial \boldsymbol{\delta}} \\ &= \mathbf{C_{D,\delta}} + 2\mathbf{C_{D,\delta^2}}\boldsymbol{\delta} + \mathbf{C_{D,\delta\delta_e}}\delta_e \\ &\quad - \frac{\mathbf{C_{m,\delta}}}{C_{m,\delta_e}}\left(C_{D,\delta_e} + 2C_{D,\delta_3^2}\delta_e + \mathbf{C}^{\mathrm{T}}_{\mathbf{D,\delta\delta_e}}\boldsymbol{\delta} + C_{D,\alpha\delta_e}\alpha\right)\end{aligned} \tag{B35}$$

B.1.2 General Trajectory Optimization

The Hamiltonian corresponding to the functional given in Eq. (B10) can be written as

$$H = cT + \lambda_h (V \sin\gamma) + \lambda_V \left(\frac{T - D - W \sin\gamma}{m} \right) + \lambda_\gamma \left(\frac{L - W \cos\gamma}{mV} \right) + \lambda_W (-cT) \quad \text{(B36)}$$

where λ_h, λ_V, λ_γ, and λ_W are the costate variables and are given by

$$\dot{\lambda}_h = -\frac{\partial H}{\partial h} = -T(1 - \lambda_W)\frac{\partial c}{\partial h} + \frac{\lambda_V}{m}\frac{\partial D}{\partial V} - \frac{\lambda_\gamma}{mV}\frac{\partial L}{\partial h} \quad \text{(B37)}$$

$$\dot{\lambda}_V = -\frac{\partial H}{\partial V} = -T(1 - \lambda_W)\frac{\partial c}{\partial V} - \lambda_h \sin\gamma + \frac{\lambda_V}{m}\frac{\partial D}{\partial V} - \frac{\lambda_\gamma}{mV^2}\left(W\cos\gamma - L + \frac{\partial L}{\partial V}V \right) \quad \text{(B38)}$$

$$\dot{\lambda}_\gamma = -\frac{\partial H}{\partial \gamma} = \lambda_h V \cos\gamma + \frac{\lambda_V}{m}\left(\frac{\partial D}{\partial \gamma} - W\cos\gamma \right) + \frac{\lambda_\gamma}{mV}\left(W\sin\gamma - \frac{\partial L}{\partial \gamma} \right) \quad \text{(B39)}$$

$$\dot{\lambda}_W = -\frac{\partial H}{\partial W} = \lambda_V g\left(\frac{T - D}{W^2} \right) + \frac{\lambda_\gamma g}{V}\left(\frac{L}{W^2} \right) \quad \text{(B40)}$$

The costate variables are bounded by the transversality condition, such that

$$\dot{\lambda}_h(t_f) = \dot{\lambda}_V(t_f) = \dot{\lambda}_\gamma(t_f) = \dot{\lambda}_W(t_f) = 0 \quad \text{(B41)}$$

The optimal flap deflections can be found from

$$\frac{\partial H}{\partial \boldsymbol{\delta}} = -\frac{\lambda_V}{m}\frac{\partial D}{\partial \boldsymbol{\delta}} + \frac{\lambda_\gamma}{mV}\frac{\partial L}{\partial \boldsymbol{\delta}} = 0 \quad \text{(B42)}$$

Using Eqs. (B23), (B27), (B33), and (B35), Eq. (B42) can be rewritten as

$$0 = -\bar{q}S\frac{\lambda_V}{m}\Bigg[\mathbf{C_{D,\delta}} + 2\mathbf{C_{D,\delta^2}}\boldsymbol{\delta} + \mathbf{C_{D,\delta,\,e}}\delta_e + \mathbf{C_{D,\alpha,}}\,\alpha - \frac{\mathbf{C_{m,\delta}}}{C_{m,\delta_e}}\left(C_{D,\delta_e} + 2C_{D,\delta_3^2}\delta_e + \mathbf{C}^{\mathrm{T}}_{\mathbf{D,\delta,\,e}}\boldsymbol{\delta} + C_{D,\alpha\delta_e}\alpha\right)\Bigg] - \bar{q}S\frac{\lambda_\gamma}{mV}\left(\mathbf{C_{L,\delta}} - \frac{C_{L,\delta_e}}{C_{m,\delta_e}}\mathbf{C_{m,\delta}}\right) \tag{B43}$$

Solving for $\boldsymbol{\delta}$ in Eq. (B43) gives

$$\boldsymbol{\delta} = \left[2\mathbf{C_{D,\delta^2}} - \frac{\mathbf{C_{m,\delta}}}{C_{m,\delta_e}}\mathbf{C}^{\mathrm{T}}_{\mathbf{D,\delta\delta_e}}\right]^{-1}\Bigg[\frac{\mathbf{C_{m,\delta}}}{C_{m,\delta_e}}\left(C_{D,\delta_e} + 2C_{D,\delta_3^2}\delta_e + C_{D,\alpha\delta_e}\alpha\right) - \mathbf{C_{D,\delta}} - \mathbf{C_{D,\delta\delta_e}}\delta_e - \mathbf{C_{D,\alpha\delta}}\alpha - \frac{\lambda_\gamma V}{\lambda_V}\left(\mathbf{C_{L,\delta}} - \frac{\mathbf{C_{m,\delta}}}{C_{m,\delta_e}}C_{L,\delta_e}\right)\Bigg] \tag{B44}$$

The optimal thrust can be found from

$$\frac{\partial H}{\partial T} = c(1-\lambda_W) + \frac{\lambda_V}{m} = 0 \tag{B45}$$

However, because Eq. (B36) is linear in T and T is bounded by $T_{\min} \le T \le T_{\max}$, the optimal thrust has a singular-arc solution of the form

$$T = \begin{cases} T_{\max} & \text{for } c(1-\lambda_W) + \frac{\lambda_V}{m} < 0 \\ T^* & \text{for } c(1-\lambda_W) + \frac{\lambda_V}{m} = 0 \\ T_{\min} & \text{for } c(1-\lambda_W) + \frac{\lambda_V}{m} > 0 \end{cases} \tag{B46}$$

where T^* is the singular-arc solution, which can be found by taking the time derivative of Eq. (B45), i.e.,

$$\dot{c}\left(1-\lambda_W\right)-c\dot{\lambda}_W+\frac{g}{W^2}\left(\dot{\lambda}_V W-\dot{W}\lambda_V\right)=0 \tag{B47}$$

Using Eqs. (B7), (B38), and (B40) in Eq. (B47) and simplifying gives

$$\begin{aligned}0=m\dot{c}(1-\lambda_W)+\lambda_V\left(\frac{cD}{W}+\frac{1}{m}\frac{\partial D}{\partial V}\right)+\frac{\lambda_\gamma}{V}\left(\frac{L}{mV}-\frac{cL}{W}-\frac{W\cos\gamma}{mV}-\frac{1}{m}\frac{\partial L}{\partial V}\right)\\ -T^*\left(1-\lambda_W\right)\frac{\partial c}{\partial V}-\lambda_h\sin\gamma\end{aligned} \tag{B48}$$

Equation (B48) can be solved for T^* to give

$$\begin{aligned}T^*=m\dot{c}\left(\frac{\partial c}{\partial V}\right)^{-1}+\left[\left(1-\lambda_W\right)\frac{\partial c}{\partial V}\right]^{-1}\left[\lambda_V\left(\frac{cD}{W}+\frac{1}{m}\frac{\partial D}{\partial V}\right)\right.\\ \left.+\frac{\lambda_\gamma}{V}\left(\frac{L}{mV}-\frac{cL}{W}-\frac{W\cos\gamma}{mV}-\frac{1}{m}\frac{\partial L}{\partial V}\right)-\lambda_h\sin\gamma\right]\end{aligned} \tag{B49}$$

From Eqs. (B44) and (B49), we see that solving for the optimum flap deflections and the optimum thrust requires that Eqs. (B37)-(B40) be solved for the costate variables, subject to Eq. (B41) and Eqs. (B6)-(B9). The states are generally known at the initial time t_0. Thus, obtaining the optimum control inputs and trajectory for the Hamiltonian in Eq. (B36) requires the solution of a two-point boundary-value problem. The primary challenge with solving this two-point boundary-value problem is that we do not know when the thrust switches between maximum thrust, zero thrust, and the singular-arc solution.

B.1.3 Constant Mach Number Mission Profile

Consider the case of an aircraft with distributed wing flaps operating with a constant Mach number. The Mach number M is related to the aircraft velocity V and the speed of sound a according to the relationship

$$M = \frac{V}{a} \tag{B50}$$

If the Mach number is constant, Eq. (B50) can be differentiated in time and rearranged to give the time rate of change of velocity $\dot{V}$, i.e.,

$$\dot{V} = M \frac{\partial a}{\partial h} \dot{h} \tag{B51}$$

Using Eq. (B6) in Eq. (B51) gives

$$\dot{V} = M \frac{\partial a}{\partial h} V \sin\gamma \tag{B52}$$

Comparing Eq. (B52) with Eq. (B5), we have

$$M \frac{\partial a}{\partial h} V \sin\gamma = \frac{T - D - W \sin\gamma}{m} \tag{B53}$$

Assuming that the climb angle γ is small, $\sin\gamma \approx \gamma$ and $\cos\gamma \approx 1$. Using the small angle approximation, Eq. (B53) can be rearranged to give

$$\gamma = \frac{T - D}{W + M \frac{\partial a}{\partial h} mV} \tag{B54}$$

Because we have known expressions for velocity and climb angle, constraints on these two state variables can be integrated directly into the Hamiltonian, without the need to include the dynamic constraints for flight path angle and velocity. The Hamiltonian is therefore given by

$$H = cT + \lambda_h V\gamma + \lambda_W(-cT) \tag{B55}$$

where V and γ are given by Eqs. (B50) and (B54). The costate variables λ_h and λ_W are found from

$$\dot{\lambda}_h = -\frac{\partial H}{\partial h} = -T(1-\lambda_W)\frac{\partial c}{\partial h} - \lambda_h\left(\frac{\partial V}{\partial h}\gamma + \frac{\partial \gamma}{\partial h}V\right) \tag{B56}$$

$$\dot{\lambda}_W = -\frac{\partial H}{\partial W} = \lambda_h V\frac{\partial \gamma}{\partial W} \tag{B57}$$

The optimal flap deflections $\boldsymbol{\delta}$ are found from

$$\frac{\partial H}{\partial \boldsymbol{\delta}} = \lambda_h V\frac{\partial \gamma}{\partial D}\frac{\partial D}{\partial \boldsymbol{\delta}} = 0 \tag{B58}$$

Using Eqs. (B27), (B35), (B50), and (B54), Eq. (B58) can be rewritten as

$$\begin{aligned} 0 = \frac{\lambda_h a\bar{q}S}{W + M\frac{\partial a}{\partial h}mV}\Bigg[&\mathbf{C_{D,\delta}} + 2\mathbf{C_{D,\delta^2}}\boldsymbol{\delta} + \mathbf{C_{D,\delta\delta_e}}\delta_e + \mathbf{C_{D,\alpha\delta}}\alpha \\ &- \frac{\mathbf{C_{m,\delta}}}{C_{m,\delta_e}}\left(C_{D,\delta_e} + 2C_{D,\delta_3^2}\delta_e + \mathbf{C^{T}_{D,\delta\delta_e}}\boldsymbol{\delta} + C_{D,\alpha\delta_e}\alpha\right)\Bigg] \end{aligned} \tag{B59}$$

Equation (B59) is solved for $\boldsymbol{\delta}$ to give

$$\boldsymbol{\delta} = \left[2\mathbf{C}_{\mathbf{D,\delta^2}} - \frac{\mathbf{C}_{\mathbf{m,\delta}}}{C_{m,\delta_e}} \mathbf{C}^{\mathrm{T}}_{\mathbf{D,\delta\delta_e}} \right]^{-1} \left[\frac{\mathbf{C}_{\mathbf{m,\delta}}}{C_{m,\delta_e}} \left(C_{D,\delta_e} + 2C_{D,\delta_3^2}\delta_e + + C_{D,\alpha\delta_e}\alpha \right) - \mathbf{C}_{\mathbf{D,\delta}} - \mathbf{C}_{\mathbf{D,\delta\delta_e}}\delta_e - \mathbf{C}_{\mathbf{D,\alpha\delta}}\alpha \right] \tag{B60}$$

which are the flap deflections that also minimize drag.

The optimal thrust is found from

$$\frac{\partial H}{\partial T} = c(1-\lambda_W) + \frac{\lambda_h Ma}{W + M\frac{\partial a}{\partial h}mV} = 0 \tag{B61}$$

again, the optimal thrust is a bang-singular-bang control solution of the form

$$T = \begin{cases} T_{\max} & \text{for } c(1-\lambda_W) + \dfrac{\lambda_h}{W + M\frac{\partial a}{\partial h}mV} < 0 \\ T^* & \text{for } c(1-\lambda_W) + \dfrac{\lambda_h}{W + M\frac{\partial a}{\partial h}mV} = 0 \\ T_{\min} & \text{for } c(1-\lambda_W) + \dfrac{\lambda_h}{W + M\frac{\partial a}{\partial h}mV} > 0 \end{cases} \tag{B62}$$

The singular-arc control T^* can be found from the time derivative of Eq. (B62), i.e.,

$$\begin{aligned} 0 &= \dot{c}(1-\lambda_W) - c\dot{\lambda}_W \\ &+ \frac{\left(W + M\frac{\partial a}{\partial h}mV\right)\left(\dot{\lambda}_h Ma + \lambda_h M\frac{\partial a}{\partial h}\dot{h}\right) - \lambda_h\left[\dot{W} + M\frac{\partial^2 a}{\partial h^2}\dot{h}mV + M\frac{\partial a}{\partial h}\left(\frac{\dot{W}}{g}V + m\dot{V}\right)\right]}{\left(W + M\frac{\partial a}{\partial h}mV\right)^2} \end{aligned} \tag{B63}$$

Using Eqs. (B6), (B7), (B50), (B51), (B54), (B56), and (B57), along with a small-climb-angle assumption, Eq. (B63) can be solved to give the singular-arc thrust T^*, i.e.,

$$T^* = \frac{\dot{c}W}{Mga\frac{\partial c}{\partial h}} - \frac{cD\lambda_h}{W\frac{\partial c}{\partial h}(1-\lambda_W)} - \frac{Ma\frac{\partial D}{\partial h}g\lambda_h}{W\frac{\partial c}{\partial h}\left(M^2 a\frac{\partial a}{\partial h} + g\right)(\lambda_W - 1)} \tag{B64}$$

Thus, solving for the optimal flap deflections and optimal thrust for an aircraft operating with constant Mach number requires the solution of the two-point boundary-value problem given by Eqs. (B6), (B7), (B56), (B57), and the transversality condition $\lambda_h(t_f) = \lambda_W(t_f) = 0$. Again, a principal challenge in solving this problem is that we do not know when the optimal thrust switches between maximum thrust, zero, thrust, and the singular-arc solution.

B.1.4 Constant Velocity Mission Profile

Consider an aircraft with distributed flaps operating with a constant velocity. For this case, $\dot{V} = 0$, and Eq. (B5) can be rewritten to give

$$\frac{T - D - W\sin\gamma}{m} = 0 \tag{B65}$$

Assuming that the climb angle is small, Eq. (B65) can be rearranged to give

$$\gamma = \frac{T - D}{W} \tag{B66}$$

Therefore, the velocity and climb angle are known functions, and the dynamic constraints for V and γ can be eliminated from the Hamiltonian. The Hamiltonian is then given by

$$H = cT + \lambda_h V\gamma + \lambda_W(-cT) \tag{B67}$$

which is identical to Eq. (B55). The costate variables are found from

$$\dot{\lambda}_h = -\frac{\partial H}{\partial h} = -T(1 - \lambda_W)\frac{\partial c}{\partial h} + \frac{\lambda_W}{W}\frac{\partial D}{\partial h} \tag{B68}$$

$$\dot{\lambda}_W = -\frac{\partial H}{\partial W} = \frac{\lambda_h}{W^2}(T - D) \tag{B69}$$

The optimal flap deflections $\boldsymbol{\delta}$ are found from

$$\frac{\partial H}{\partial \boldsymbol{\delta}} = \frac{\lambda_h}{W}\frac{\partial D}{\partial \boldsymbol{\delta}} = 0 \tag{B70}$$

Using Eqs. (B27) and (B35), Eq. (B70) is rewritten as

$$\begin{aligned} 0 = \frac{\lambda_h \bar{q} S}{W}\Big[&\mathbf{C_{D,\delta}} + 2\mathbf{C_{D,\delta^2}}\boldsymbol{\delta} + \mathbf{C_{D,\delta\delta_e}}\delta_e + \mathbf{C_{D,\alpha\delta}}\alpha \\ &- \frac{\mathbf{C_{m,\delta}}}{C_{m,\delta_e}}\left(C_{D,\delta_e} + 2C_{D,\delta_3^2}\delta_e + \mathbf{C^{T}_{D,\delta\delta_e}}\boldsymbol{\delta} + C_{D,\alpha\delta_e}\alpha\right)\Big] \end{aligned} \tag{B71}$$

which can be solved for $\boldsymbol{\delta}$ to give

$$\begin{aligned} \boldsymbol{\delta} = \left[2\mathbf{C_{D,\delta^2}} - \frac{\mathbf{C_{m,\delta}}}{C_{m,\delta_e}}\mathbf{C^{T}_{D,\delta\delta_e}}\right]^{-1}\Bigg[&\frac{\mathbf{C_{m,\delta}}}{C_{m,\delta_e}}\left(C_{D,\delta_e} + 2C_{D,\delta_3^2}\delta_e + + C_{D,\alpha\delta_e}\alpha\right) \\ &- \mathbf{C_{D,\delta}} - \mathbf{C_{D,\delta\delta_e}}\delta_e - \mathbf{C_{D,\alpha\delta}}\alpha\Bigg] \end{aligned} \tag{B72}$$

Note that Eq. (B72) is identical to Eq. (B60) and is also the flap setting that produces minimum drag. The optimal thrust is found from

$$\frac{\partial H}{\partial T} = c(1 - \lambda_W) + \frac{\lambda_h V}{W} = 0 \tag{B73}$$

The optimal thrust is a bang-singular-bang solution of the form

$$T = \begin{cases} T_{\max} & \text{for } c(1-\lambda_W) + \dfrac{\lambda_h V}{W} < 0 \\ T^* & \text{for } c(1-\lambda_W) + \dfrac{\lambda_h V}{W} = 0 \\ T_{\min} & \text{for } c(1-\lambda_W) + \dfrac{\lambda_h V}{W} > 0 \end{cases} \tag{B74}$$

The singular-arc control is found from

$$0 = \dot{c}(1-\lambda_W) - c\dot{\lambda}_W + \frac{V}{W^2}\left(W\dot{\lambda}_h - \lambda_h \dot{W}\right) \tag{B75}$$

Using Eqs. (B7), (B68), and (B69), Eq. (B75) is rewritten as

$$0 = \dot{c}(1-\lambda_W) - c\frac{\lambda_h}{W^2}(T^* - D) + \frac{V}{W^2}\left[\lambda_W \frac{\partial D}{\partial h} - T^* W(1-\lambda_W)\frac{\partial c}{\partial h} + \lambda_h c T^*\right] \tag{B76}$$

Equation (B76) is solved for T^* to give

$$T^* = \frac{\dot{c}W^2(1-\lambda_W) + \lambda_h cD + V\lambda_W \frac{\partial D}{\partial h}}{c\lambda_h(1-V) + VW(1-\lambda_W)\frac{\partial c}{\partial h}} \tag{B77}$$

Thus, solving for the optimal flap deflections and optimal thrust for an aircraft operating with a constant velocity requires the solution of the two-point boundary-value problem given by Eqs. (B6), (B7), (B68), (B69), along with the transversality condition, i.e., $\lambda_h(t_f) = \lambda_W(t_f) = 0$.

B.1.5 Constant Altitude Mission Profile

Consider the case of an aircraft operating at a constant altitude. For constant-altitude flight, the time rate of change of altitude is zero, i.e.,

$$\dot{h} = V \sin\gamma = 0 \tag{B78}$$

The velocity V cannot be zero in forward flight. Therefore, Eq. (B78) implies that

$$\gamma = 0, \quad \dot{\gamma} = 0 \tag{B79}$$

The time rate of change of velocity from Eq. (B5) can then be rewritten as

$$\dot{V} = \frac{T - D}{m} \tag{B80}$$

and, using Eq. (B79) and the small-angle approximation for γ, Eq. (B8) results in

$$L = W \tag{B81}$$

which is typical for level (constant altitude) flight. Since the altitude and climb angle are zero, the altitude and climb-angle dynamic constraints can be neglected, and the Hamiltonian becomes

$$H = cT + \lambda_V \frac{T - D}{m} + \lambda_W (-cT) \tag{B82}$$

where the costate equations are given by

$$\dot{\lambda}_V = -\frac{\partial H}{\partial V} = -T(1 - \lambda_W)\frac{\partial c}{\partial V} + \frac{\lambda_V}{m}\frac{\partial D}{\partial V} \tag{B83}$$

$$\dot{\lambda}_W = -\frac{\partial H}{\partial W} = \frac{\lambda_V g}{W^2}(T - D) \tag{B84}$$

The optimal flap deflection is obtained from

$$\frac{\partial H}{\partial \boldsymbol{\delta}} = -\frac{\lambda_V}{m}\frac{\partial D}{\partial \boldsymbol{\delta}} = 0 \tag{B85}$$

which can be rewritten using Eqs. (B27) and (B35) to give

$$\begin{aligned} 0 = \frac{\lambda_V \bar{q} S}{m}\Big[& \mathbf{C_{D,\delta}} + 2\mathbf{C_{D,\delta^2}}\boldsymbol{\delta} + \mathbf{C_{D,\delta\delta_e}}\delta_e + \mathbf{C_{D,\alpha\delta}}\alpha \\ & - \frac{\mathbf{C_{m,\delta}}}{C_{m,\delta_e}}\left(C_{D,\delta_e} + 2C_{D,\delta_3^2}\delta_e + \mathbf{C}^{\mathrm{T}}_{\mathbf{D,\delta\delta_e}}\boldsymbol{\delta} + C_{D,\alpha\delta_e}\alpha\right)\Big] \end{aligned} \tag{B86}$$

and solved for $\boldsymbol{\delta}$ to give

$$\begin{aligned} \boldsymbol{\delta} = \left[2\mathbf{C_{D,\delta^2}} - \frac{\mathbf{C_{m,\delta}}}{C_{m,\delta_e}}\mathbf{C}^{\mathrm{T}}_{\mathbf{D,\delta\delta_e}}\right]^{-1}\Bigg[& \frac{\mathbf{C_{m,\delta}}}{C_{m,\delta_e}}\left(C_{D,\delta_e} + 2C_{D,\delta_3^2}\delta_e + + C_{D,\alpha\delta_e}\alpha\right) \\ & - \mathbf{C_{D,\delta}} - \mathbf{C_{D,\delta\delta_e}}\delta_e - \mathbf{C_{D,\alpha\delta}}\alpha\Bigg] \end{aligned} \tag{B87}$$

Equation (B87) is identical to Eqs. (B72) and (B60), and it is the flap deflection that also minimizes drag. The optimal thrust is found from

$$\frac{\partial H}{\partial T} = c(1-\lambda_W) + \frac{\lambda_V}{m} = 0 \tag{B88}$$

which gives a bang-singular-bang solution, i.e.,

$$T = \begin{cases} T_{\max} & \text{for } c(1-\lambda_W) + \frac{\lambda_V}{m} < 0 \\ T^* & \text{for } c(1-\lambda_W) + \frac{\lambda_V}{m} = 0 \\ T_{\min} & \text{for } c(1-\lambda_W) + \frac{\lambda_V}{m} > 0 \end{cases} \tag{B89}$$

Note that Eq. (B89) is identical to Eq. (B46). The singular solution T^* is found by taking the time derivative of Eq. (B88), i.e.,

$$0 = \dot{c}(1-\lambda_W) - c\dot{\lambda}_W + \frac{g}{W^2}\left(\dot{\lambda}_V W - \dot{W}\lambda_V\right) \tag{B90}$$

Using Eqs. (B7), (B83), and (B84), Eq. (B90) is rewritten as

$$0 = \dot{c}\left(1-\lambda_W\right) - c\frac{\lambda_V g}{W^2}(T^* - D) + \frac{g}{W^2}\left[\lambda_V g + \lambda_V c T^* - T^* W(1-\lambda_W)\frac{\partial c}{\partial V}\right] \tag{B91}$$

which can be solved to find the singular-arc solution, i.e.,

$$T^* = \frac{W^2\dot{c}(\lambda_W - 1) + \lambda_V g\left(cD - g\frac{\partial D}{\partial V}\right)}{Wg(1-\lambda_W)\frac{\partial c}{\partial V}} \tag{B92}$$

Thus, in order to obtain the optimal thrust and optimal flap deflection for an aircraft operating at a constant altitude, we must solve the two-point boundary-value problem given by Eqs. (B7), (B78)-(B80), (B83), (B84), and the transversality conditions $\dot{\lambda}_V(t_f) = \dot{\lambda}_W(t_f) = 0$.

B.2 Trajectory Optimization for a Battery-Powered Elliptic Wing

Consider a wing with an elliptic planform in climbing flight, with velocity V, weight W, climb angle γ, and thrust T aligned in the direction of flight, as shown in Fig. B1. If the thrust is generated by a battery-powered motor, we can assume that the weight W is constant.

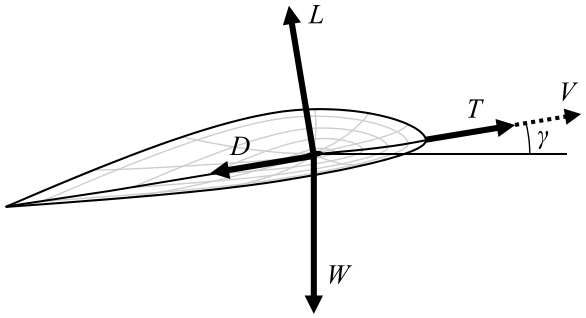


Fig. B1 Elliptic wing in climbing flight

B.2.1 Equations of Motion

The equations of motion for this wing can be found using a point mass model. We assume that the wing is in pseudo-trim state with zero aerodynamic moment. If the wing is in climbing flight with climb angle γ, the time rate of change of downrange position x is simply the horizontal component of velocity V, i.e.,

$$\dot{x} = V \cos\gamma \tag{B93}$$

The time rate of change in vertical position h is the vertical component of the velocity, i.e.,

$$\dot{h} = V \sin\gamma \tag{B94}$$

If the thrust is aligned with the direction of flight, the time rate of change of the climb angle can be written as

$$\dot{\gamma} = \frac{L - W\cos\gamma}{mV} \tag{B95}$$

where L is the aircraft lift, and m is the aircraft mass, which can be rewritten in terms of the weight W and acceleration g due to gravity as $m = W/g$. The time rate of change of velocity can be written

$$\dot{V} = \frac{T - D - W\sin\gamma}{m} \tag{B96}$$

where T is the thrust, and D is the drag.

The equations of motion in Eqs. (B94)-(B96) can be rewritten in terms of range using the change of variables $\frac{\partial}{\partial t} = \frac{\partial}{\partial x}\frac{\partial x}{\partial t}$, and noting that $\frac{\partial x}{\partial t} = \dot{x} = V\cos\gamma$, as shown in Eq. (B93). Assuming that γ is small, $\cos\gamma \approx 1$, the equations of motion become

$$h' = \sin\gamma \tag{B97}$$

$$\gamma' = \frac{L - W\cos\gamma}{mV^2} \tag{B98}$$

$$V' = \frac{T - D - W\sin\gamma}{mV} \tag{B99}$$

For an aircraft in climbing flight, we can rearrange Eq. (B151) to obtain an expression for the thrust, i.e.,

$$T = mVV'\cos\gamma + D + W\sin\gamma \tag{B100}$$

Here, it is convenient to rewrite Eq. (B100) in terms of the drag coefficient, i.e.,

$$T = mVV'\cos\gamma + \tfrac{1}{2}\rho V^2 S_w C_D + W\sin\gamma \tag{B101}$$

In subsonic flight the drag coefficient can be thought of as the sum of the induced drag coefficient C_{D_i} and the parasitic drag coefficient C_{D_p}. From classical lifting-line theory, the induced drag coefficient C_{D_i} on a wing with an elliptic planform can be written in terms of the lift coefficient C_L as

$$C_{D_i} = \frac{C_L^2}{\pi R_A} \tag{B102}$$

The induced drag usually makes up around half of the total drag in cruise. Using Eq. (B102) and the definition of the drag coefficient, (B101) can be rewritten as

$$T = mVV'\cos\gamma + \tfrac{1}{2}\rho V^2 S_w\left(\frac{C_L^2}{\pi R_A} + C_{D_p}\right) + W\sin\gamma \tag{B103}$$

Assuming that the forces are balanced in the direction perpendicular to the lift, we can write the lift as

$$L = W \cos\gamma \tag{B104}$$

Using Eq. (B104), along with the relation $m = W/g$, Eq. (B103) can be simplified to give

$$T = W\left[\frac{VV'}{g}\cos\gamma + \frac{2\cos^2\gamma}{\rho V^2 \pi R_A}\frac{W}{S_w} + \rho V^2 \frac{S_w}{2W} C_{D_p} + \sin\gamma\right] \tag{B105}$$

By the small angle approximation, $\sin\gamma \approx \gamma$ and $\cos\gamma \approx 1$. Using this approximation in Eq. (B105), we have

$$T = W\left[\frac{VV'}{g} + \frac{2}{\rho V^2 \pi R_A}\frac{W}{S_w} + \rho V^2 \frac{S_w}{2W} C_{D_p} + \gamma\right] \tag{B106}$$

The climb angle can be found in terms of the velocity V and climb rate, V_c. Using the small angle approximation, the climb angle can be written as

$$\gamma = \sin^{-1}\frac{V_c}{V} \approx \frac{V_c}{V} \tag{B107}$$

The climb rate V_c is simply the time rate of change of altitude h, i.e.,

$$V_c = \frac{dh}{dt} = \dot{h} \tag{B108}$$

The climb rate can also be written in terms of the change in altitude with horizontal distance x using the relation $V\cos\gamma = dx/dt$ and the chain rule in Eq. (B108)to give

$$V_c = Vh'\cos\gamma \tag{B109}$$

Comparing Eq. (B107) to Eq. (B109), we see that for small climb angles

$$h' = \gamma \tag{B110}$$

Using Eq. (B110) in Eq. (B106), we can rewrite the thrust in terms of V, V', h, and h' to give an expression for the thrust required to maintain climbing flight for an elliptic wing in inviscid flow with small climb angle, i.e.,

$$T = W\left[\frac{VV'}{g} + \frac{2}{\rho V^2 \pi R_A}\frac{W}{S_w} + \rho V^2 \frac{S_w}{2W} C_{D_p} + h'\right] \tag{B111}$$

In many cases, we can assume that we are in quasi-steady flight, or that during flight, the velocity does not change quickly. Under this assumption, we assume that $V' = 0$, and we can obtain a simplified expression for the thrust, i.e.,

$$T = W\left[\frac{2}{\rho V^2 \pi R_A}\frac{W}{S_w} + \rho V^2 \frac{S_w}{2W} C_{D_p} + h'\right] \tag{B112}$$

B.2.2 Minimizing Power using the Calculus of Variations

In this study, we aim to minimize the power over the course of a trajectory by identifying the optimum altitude and velocity as a function of horizontal distance x. The total power over a trajectory can be written as

$$J = \int_0^t TV dt \tag{B113}$$

where t is the time of flight. If we assume that the climb angle γ is small, Eq. (B113) can be rewritten in terms of the range r using the change of variables $dx = Vdt$, i.e.,

$$J = \int_0^r T dx \tag{B114}$$

Using Eq. (B111) in Eq. (B114) gives

$$J = W \int_0^r \left[\frac{VV'}{g} + \frac{2}{\rho V^2 \pi R_A} \frac{W}{S_w} + \rho V^2 \frac{S_w}{2W} C_{D_p} + h' \right] dx \tag{B115}$$

Assuming that the density ρ changes linearly with altitude within the typical operating range of this wing, the density can be rewritten as

$$\rho = C_{\rho_0} + C_{\rho_1} h \tag{B116}$$

Using Eq. (B116) in Eq. (B115) gives

$$J = W \int_0^r \left[\frac{VV'}{g} + \frac{2}{\left(C_{\rho_0} + C_{\rho_1} h\right) V^2 \pi R_A} \frac{W}{S_w} + \left(C_{\rho_0} + C_{\rho_1} h\right) V^2 \frac{S_w}{2W} C_{D_p} + h' \right] dx \tag{B117}$$

Equation (B115) can be rewritten as a functional that is dependent on the functions $V(x)$ and $h(x)$. These functions can be found by applying the calculus of variations. From the calculus of variations, the first variation of J can be written as

$$\delta J = \frac{d}{d\epsilon} W \int_0^r \left\{ \frac{\left(V + \epsilon y_v\right)\left(V' + \epsilon y_v'\right)}{g} + \frac{2}{\left[C_{\rho_0} + C_{\rho_1}\left(h + \epsilon y_h\right)\right]\left(V + \epsilon y_v\right)^2 \pi R_A} \frac{W}{S_w} + \left[C_{\rho_0} + C_{\rho_1}\left(h + \epsilon y_h\right)\right]\left(V + \epsilon y_v\right)^2 \frac{S_w}{2W} C_{D_p} + \left(h' + \epsilon y_h'\right) \right\} dx \Bigg|_{\epsilon=0} \tag{B118}$$

where ϵ is a small number and $y_v = y_v(x)$ and $y_h = y_h(x)$ are perturbation functions on the functions $V(x)$ and $h(x)$, respectively. Equation (B118) can be simplified to give

$$\delta J = W \int_0^r \left[\frac{V y_v' + V' y_v}{g} - \frac{4W}{\pi R_A \left(C_{\rho_0} + C_{\rho_1} h\right) V^3 S_w} y_v - \frac{2W C_{\rho_1}}{\pi R_A \left(C_{\rho_0} + C_{\rho_1} h\right)^2 V^2 S_w} y_h + \frac{S_w C_{D_p}}{W} \left(C_{\rho_0} + C_{\rho_1} h\right) V y_v + \frac{S_w C_{D_p}}{2W} C_{\rho_1} V^2 y_h + y_h' \right] dx \tag{B119}$$

Note that the first term in the integrand in Eq. (B119) is the product rule expansion of

$d/dx(Vy_v)$. The last term in the integrand can be rewritten as $d/dx(y_h)$. Thus, the first and last terms can be integrated to give

$$\delta J = W \int_0^r \left[-\frac{4W}{\pi R_A \left(C_{\rho_0} + C_{\rho_1} h\right) V^3 S_w} y_v - \frac{2WC_{\rho_1}}{\pi R_A \left(C_{\rho_0} + C_{\rho_1} h\right)^2 V^2 S_w} y_h \right. \\ \left. + \frac{S_w C_{D_p}}{W} \left(C_{\rho_0} + C_{\rho_1} h\right) V y_v + \frac{S_w C_{D_p}}{2W} C_{\rho_1} V^2 y_h \right] dx + \left. \frac{V}{g} y_v \right|_0^r + \left. y_h \right|_0^r \quad \text{(B120)}$$

Because y_v and y_h are perturbation functions on $V(x)$ and $h(x)$, which have fixed boundary conditions at $x = 0$ and $x = r$, they are zero at 0 and r. Therefore, the last two terms in Eq. (B120) are zero, and Eq. (B120) can be rewritten as

$$\delta J = W \int_0^r \left[\left(\frac{S_w C_{D_p}}{W} \left(C_{\rho_0} + C_{\rho_1} h\right) V - \frac{4W}{\pi R_A \left(C_{\rho_0} + C_{\rho_1} h\right) V^3 S_w} \right) y_v \right. \\ \left. + \left(\frac{S_w C_{D_p}}{2W} C_{\rho_1} V^2 - \frac{2WC_{\rho_1}}{\pi R_A \left(C_{\rho_0} + C_{\rho_1} h\right)^2 V^2 S_w} \right) y_h \right] dx \quad \text{(B121)}$$

The functions $h(x)$ and $V(x)$ that minimize J must satisfy the relation

$$\delta J = W \int_0^r \left[\left(\frac{S_w C_{D_p}}{W} \left(C_{\rho_0} + C_{\rho_1} h\right) V - \frac{4W}{\pi R_A \left(C_{\rho_0} + C_{\rho_1} h\right) V^3 S_w} \right) y_v \right. \\ \left. + \left(\frac{S_w C_{D_p}}{2W} C_{\rho_1} V^2 - \frac{2WC_{\rho_1}}{\pi R_A \left(C_{\rho_0} + C_{\rho_1} h\right)^2 V^2 S_w} \right) y_h \right] dx = 0 \quad \text{(B122)}$$

for all possible functions y_h and y_h. Therefore, the condition in Eq. (B122) reduces to

$$\left(\frac{S_w C_{D_p}}{W} \left(C_{\rho_0} + C_{\rho_1} h\right) V - \frac{4W}{\pi R_A \left(C_{\rho_0} + C_{\rho_1} h\right) V^3 S_w} \right) y_v \\ + \left(\frac{S_w C_{D_p}}{2W} C_{\rho_1} V^2 - \frac{2WC_{\rho_1}}{\pi R_A \left(C_{\rho_0} + C_{\rho_1} h\right)^2 V^2 S_w} \right) y_h = 0 \quad \text{(B123)}$$

which requires that

$$\frac{S_w C_{D_p}}{W}\left(C_{\rho_0}+C_{\rho_1}h\right)V-\frac{4W}{\pi R_A\left(C_{\rho_0}+C_{\rho_1}h\right)V^3S_w}=0 \tag{B124}$$

and

$$\frac{S_w C_{D_p}}{2W}C_{\rho_1}V^2-\frac{2WC_{\rho_1}}{\pi R_A\left(C_{\rho_0}+C_{\rho_1}h\right)^2V^2S_w}=0 \tag{B125}$$

Note that Eqs. (B124) and (B125) both reduce to

$$\frac{S_w C_{D_p}}{W}V^2=\frac{4W}{\pi R_A\left(C_{\rho_0}+C_{\rho_1}h\right)^2V^2S_w} \tag{B126}$$

which can be solved for the velocity to give

$$V=\sqrt[4]{\frac{4}{\pi R_A C_{D_p}}}\sqrt{\frac{W}{\rho S_w}} \tag{B127}$$

Equation (B127) is equivalent to the minimum drag airspeed for an aircraft in steady level flight.

B.3 Trajectory Optimization for a Battery-Powered Aircraft

Consider an aircraft in climbing flight, with velocity V, weight W, climb angle γ, and thrust T aligned in the direction of flight, as shown in Fig. B2. If the thrust is generated by a battery-powered motor, we can assume that the weight W is constant. The aircraft equations of motion are given by Eqs. (B93)-(B99).

The lift and drag are often written in terms of a lift and drag coefficient, which are defined as

$$C_L=\frac{L}{\frac{1}{2}\rho V^2S_w} \tag{B128}$$

$$C_D=\frac{D}{\frac{1}{2}\rho V^2S_w} \tag{B129}$$

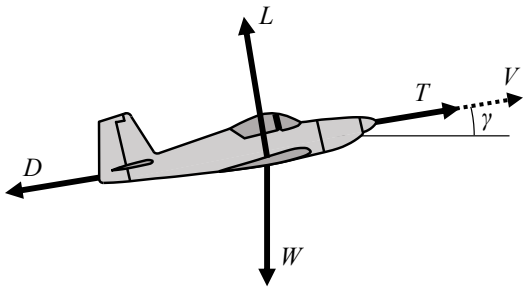


Fig. B2 Aircraft in climbing flight

where ρ is the atmospheric density, and S_w is the reference area of the main wing. Using a linear approximation, the lift coefficient can be written in terms of a lift slope, $C_{L,\alpha}$, the angle of attack α, and the zero-lift angle of attack $\alpha_{L=0}$ as

$$C_L = C_{L,\alpha}\,(\alpha - \alpha_{L=0}) \tag{B130}$$

The drag coefficient can be written in terms of the lift coefficient by assuming that the drag coefficient is nearly parabolic with respect to the lift coefficient, i.e.,

$$C_D = C_{D_0} + C_{D_1} C_L + C_{D_2} C_L^2 \tag{B131}$$

where C_{D_0}, C_{D_1}, and C_{D_2} are constant coefficients in the parabolic representation of the drag coefficient as a function of the lift coefficient.

If the climb angle is assumed to be zero and constant, the time rate of change of velocity can be simplified to give

$$\dot{V} = \frac{T}{m} - \frac{D(V)}{m} \tag{B132}$$

For this case, the lift is equal to the weight, which means that the lift coefficient C_L can be written as

$$C_L = \frac{W}{\frac{1}{2}\rho V^2 S_w} \tag{B133}$$

Using Eq. B133 with Eq. B131 gives an expression for the drag as a function of the velocity, i.e.,

$$D = \tfrac{1}{2}\rho V^2 S_w C_{D_0} + C_{D_1} W + C_{D_2} \frac{W^2}{\tfrac{1}{2}\rho V^2 S_w} \tag{B134}$$

B.3.1 Minimum Thrust with Zero Climb Angle

Consider the optimal control problem in which the integral of thrust is to be minimized over the course of a flight from an initial time $t_0 = 0$ to some final time t_f with $\gamma = 0$. As shown in the previous subsection, the aircraft is subject to equations of motion, which can be considered dynamic constraints. The thrust can take any value bounded by $0 \leq T \leq T_{\max}$, where $T_{\max}$ is the maximum available thrust. The aircraft begins at a position $x(0) = x_0$ with velocity $V(0) = V_0$, and ends at a final position $x(t_f) = x_f$ with final velocity $V(t_f) = V_f$. Thus, the optimization problem can be summarized as

$$\begin{aligned}
\text{minimize:} \quad & \int_0^{t_f} T\,dt \\
\text{with respect to:} \quad & T(t),\ V(t) \\
\text{subject to:} \quad & \dot{x} = V \\
& \dot{V} = \frac{T}{m} - \frac{1}{m}\left(\tfrac{1}{2}\rho V^2 S_w C_{D_0} + C_{D_1} W + C_{D_2} \frac{W^2}{\tfrac{1}{2}\rho V^2 S_w}\right) \\
& x(0) = x_0, \quad x(t_f) = x_f \\
& V(0) = V_0, \quad V(t_f) = V_f \\
& 0 \leq T \leq T_{\max}
\end{aligned}$$

From optimal control theory, the Hamiltonian can be written as

$$H = \lambda_0 T + \lambda_1 V + \lambda_2 \left[\frac{T}{m} - \frac{1}{m}\left(\tfrac{1}{2}\rho V^2 S_w C_{D_0} + C_{D_1} W + C_{D_2} \frac{W^2}{\tfrac{1}{2}\rho V^2 S_w}\right)\right] \tag{B135}$$

where λ_0, λ_1, and λ_2 are the costate variables. The costate variable λ_0 is a constant that can be either 0 or 1. The other two variables, λ_1 and λ_2, can be found from the differential

equations

$$\dot{\lambda_1} = -\frac{\partial H}{\partial x} = 0 \tag{B136}$$

$$\dot{\lambda_2} = -\frac{\partial H}{\partial V} = -\lambda_1 + \frac{\lambda_2}{m}\frac{\partial D}{\partial V} \tag{B137}$$

The optimal thrust T^* minimizes the Hamiltonian. This requires that T^* minimize the expression

$$\left(\lambda_0 + \frac{\lambda_2}{m}\right) T^* \tag{B138}$$

The value of T^* that minimizes this expression depends on the value of the quantity in parentheses. For the case where $\left(\lambda_0 + \frac{\lambda_2}{m}\right) > 0$, Eq. B138 is minimized with $T^* = 0$. When $\left(\lambda_0 + \frac{\lambda_2}{m}\right) < 0$, thrust is minimized with $T^* = T_{\max}$. For the case where $\left(\lambda_0 + \frac{\lambda_2}{m}\right) = 0$ the optimal thrust is said to be singular.

Suppose that $\lambda_0 = 0$. For the expression in Eq. (B138) to remain zero, then λ_2 and $\dot{\lambda_2}$ must also be zero. Using $\lambda_2 = \dot{\lambda_2} = 0$ in Eq. (B137) gives $\lambda_1 = 0$. Thus, if $\lambda_0 = 0$, then $\lambda_1 = \lambda_2 = 0$. This violates the non-triviality condition, which dictates that $(\lambda_0, \lambda_1, \lambda_2) \neq 0$. Therefore, λ_0 must be equal to 1. In the case that $\lambda_0 = 1$, the condition $\left(\lambda_0 + \frac{\lambda_2}{m}\right) = 0$ requires that $\lambda_2 = -m$, which is a constant. Therefore, $\dot{\lambda_2} = 0$, and from Eq. (B137), we find that $\lambda_1 = \frac{\partial D}{\partial V}$, which, by Eq. (B136), is also a constant. Evaluating $\frac{\partial D}{\partial V}$ from Eq. (B134) gives

$$\frac{\partial D}{\partial V} = \rho V S_w C_{D_0} - \frac{4 C_{D_2} W^2}{\rho V^3 S_w} \tag{B139}$$

For Eq. (B139) to be constant requires that

$$\frac{\partial \dot{D}}{\partial V} = \dot{V}\left(\rho S_w C_{D_0} + \frac{12 C_{D_2} W^2}{\rho V^4 S_w}\right) = 0 \tag{B140}$$

Equation (B140) can be satisfied only if V is constant, i.e, $\dot{V} = 0$. From Eq. (B132), this

gives

$$T^* = \tfrac{1}{2}\rho V^2 S_w C_{D_0} + C_{D_1} W + C_{D_2} \frac{W^2}{\tfrac{1}{2}\rho V^2 S_w} \tag{B141}$$

which is equivalent to the drag D. Thus, the optimal thrust can be written

$$T^* = \begin{cases} 0 & \text{for} \quad \left(\lambda_0 + \dfrac{\lambda_2}{m}\right) > 0 \\ T_{\max} & \text{for} \quad \left(\lambda_0 + \dfrac{\lambda_2}{m}\right) < 0 \\ D & \text{for} \quad \left(\lambda_0 + \dfrac{\lambda_2}{m}\right) = 0 \end{cases} \tag{B142}$$

B.3.2 Minimum Power with zero Climb Angle

If the power, rather than the thrust, is to be minimized over time with $\gamma = 0$, the optimization problem changes to

$$\begin{aligned} \text{minimize:} \quad & \int_0^{t_f} TV dt \\ \text{with respect to:} \quad & T(t),\ V(t) \\ \text{subject to:} \quad & \dot{x} = V \\ & \dot{V} = \frac{T}{m} - \frac{1}{m}\left(\tfrac{1}{2}\rho V^2 S_w C_{D_0} + C_{D_1} W + C_{D_2} \frac{W^2}{\tfrac{1}{2}\rho V^2 S_w}\right) \\ & x(0) = x_0, \quad x(t_f) = x_f \\ & V(0) = V_0, \quad V(t_f) = V_f \\ & 0 \le T \le T_{\max} \end{aligned}$$

Here, the Hamiltonian is written as

$$H = \lambda_0 TV + \lambda_1 V + \lambda_2 \left(\frac{T}{m} - \frac{D}{m}\right) \tag{B143}$$

The costate variables are found from

$$\dot{\lambda_1} = -\frac{\partial H}{\partial x} = 0 \tag{B144}$$

$$\dot{\lambda_2} = -\frac{\partial H}{\partial V} = -\lambda_0 T - \lambda_1 + \frac{\lambda_2}{m}\frac{\partial D}{\partial V} \tag{B145}$$

The optimal thrust minimizes the expression

$$\left(\lambda_0 V + \frac{\lambda_2}{m}\right) T^* \tag{B146}$$

When $\left(\lambda_0 V + \frac{\lambda_2}{m}\right) > 0$, the expression in Eq. (B146) is minimized with $T^* = 0$; if $\left(\lambda_0 V + \frac{\lambda_2}{m}\right) < 0$, the expression in Eq. (B146) is minimized with $T^* = T_{\max}$; when $\left(\lambda_0 V + \frac{\lambda_2}{m}\right) = 0$, the optimal thrust is singular.

Consider the case where $\left(\lambda_0 V + \frac{\lambda_2}{m}\right) = 0$ and $\lambda_0 = 0$. In order for Eq.(B146) to remain zero,λ_2 and $\dot{\lambda_2}$ must be zero. Using $\lambda_2 = \dot{\lambda_2} = 0$ in Eq. (B145) gives $\lambda_1 = 0$. Thus, $\lambda_0 = 0$ requires that $\lambda_1 = \lambda_2 = 0$, which violates the non-triviality condidition. If $\lambda_0 = 1$, then $\lambda_2 = -mV$. Differentiating gives

$$\dot{\lambda_2} = -m\dot{V} = T - \lambda_1 - V\frac{\partial D}{\partial V} \tag{B147}$$

Equating Eq. (B145) and Eq. (B147) and simplifying gives

$$\lambda_1 = -V\frac{\partial D}{\partial V} - D \tag{B148}$$

Differentiating Eq. (B148) and using Eq. (B144) gives

$$\dot{\lambda_1} = \dot{V}\left(\frac{\partial D}{\partial V} - 6C_{D_2}\frac{2W^2}{\rho V^3 S_w} - \rho V S_w C_{D_0} - 2C_{D_2}\frac{2W^2}{\rho V^2 S_w}\right) = 0 \tag{B149}$$

which implies that $\dot{V} = 0$. Using this result with Eq. (B132) gives $T^* = D$. Thus, the optimal

thrust can be written as

$$T^* = \begin{cases} 0 & \text{for} \quad \left(\lambda_0 V + \dfrac{\lambda_2}{m}\right) > 0 \\ T_{\max} & \text{for} \quad \left(\lambda_0 V + \dfrac{\lambda_2}{m}\right) < 0 \\ D & \text{for} \quad \left(\lambda_0 V + \dfrac{\lambda_2}{m}\right) = 0 \end{cases} \tag{B150}$$

B.3.3 Optimization with Free Final State and Zero Climb Angle

Consider now the case where we wish to identify the trajectory that minimizes the thrust or the power over the course of a flight phase in which the final state is not specified. In other words, we seek to minimize thrust or power, but we allow the optimization to determine the optimal final state. This sort of problem often appears in cruise trajectory optimization, in which the initial cruise state is specified, but the final cruise state is to be determined through optimization alongside the optimal trajectory. For this case, we require two boundary conditions in addition to the initial state boundary conditions. These come from the transversality conditions $\lambda_1(t_f) = \lambda_2(t_f) = 0$. The optimization problem for minimum thrust can be written as

$$\begin{aligned} \text{minimize:} \quad & \int_0^{t_f} T\,dt \\ \text{with respect to:} \quad & T(t),\ V(t) \\ \text{subject to:} \quad & \dot{x} = V \\ & \dot{V} = \frac{T}{m} - \frac{1}{m}\left(\tfrac{1}{2}\rho V^2 S_w C_{D_0} + C_{D_1} W + C_{D_2} \frac{W^2}{\tfrac{1}{2}\rho V^2 S_w}\right) \\ & x(0) = x_0, \quad \lambda_1(t_f) = 0 \\ & V(0) = V_0, \quad \lambda_2(t_f) = 0 \\ & 0 \le T \le T_{\max} \end{aligned}$$

The Hamiltonian for this problem is the same as that shown in Eq.(B135). Because $\dot{\lambda}_1 = 0$,

as shown in Eq. (B136), and $\lambda_1(t_f) = 0$, we know that $\lambda_1(t) = 0$ throughout the trajectory.

For the case where the final state is not specified, the optimization problem for minimum power can be rewritten using the transversality conditions to give

$$\begin{aligned}
\text{minimize:} \quad & \int_0^{t_f} TV dt \\
\text{with respect to:} \quad & T(t),\ V(t) \\
\text{subject to:} \quad & \dot{x} = V \\
& \dot{V} = \frac{T}{m} - \frac{1}{m}\left(\tfrac{1}{2}\rho V^2 S_w C_{D_0} + C_{D_1} W + C_{D_2} \frac{W^2}{\tfrac{1}{2}\rho V^2 S_w}\right) \\
& x(0) = x_0, \quad \lambda_1(t_f) = 0 \\
& V(0) = V_0, \quad \lambda_2(t_f) = 0 \\
& 0 \le T \le T_{\max}
\end{aligned}$$

The Hamiltonian matches that shown in Eq. (B143), and $\dot{\lambda}_1 = 0$, as shown in Eq. (B144), which means that $\lambda_1(t) = 0$ throughout the trajectory.

B.3.4 Minimum Power over a Specified Range with Zero Climb Angle

It is sometimes convenient to rewrite the problem in terms of downrange distance x. Using the conversion $\frac{\partial}{\partial t} = \frac{\partial}{\partial x}\frac{\partial x}{\partial t}$, and noting that $\frac{\partial x}{\partial t} = \dot{x} = V$, as shown in Eq. (B93), we can rewrite Eq. (B132) in terms of x as

$$V' = \frac{\dot{V}}{V} = \frac{T}{mV} - \frac{D(V)}{mV} \tag{B151}$$

The power is $P = TV$, thus, using the same conversion, we can rewrite the integral of the power over time in terms of x as

$$J = \int_0^{t_f} TV dt = \int_{x_0}^{x_f} \frac{TV}{V} dx = \int_{x_0}^{x_f} T dx \tag{B152}$$

The optimization problem for minimum power over time can then be summarized as

$$\begin{aligned}
\text{minimize:} \quad & \int_{x_0}^{x_f} T\,dx \\
\text{with respect to:} \quad & T(t),\ V(t) \\
\text{subject to:} \quad & V' = \frac{T}{mV} - \frac{1}{m}\left(\tfrac{1}{2}\rho V S_w C_{D_0} + C_{D_1}\frac{W}{V} + C_{D_2}\frac{W^2}{\tfrac{1}{2}\rho V^3 S_w}\right) \\
& V(0) = V_0, \quad V(t_f) = V_f \\
& 0 \le T \le T_{\max}
\end{aligned}$$

The Hamiltonian for this problem is

$$H = \lambda_0 T + \lambda_1\left[\frac{T}{mV} - \frac{1}{m}\left(\tfrac{1}{2}\rho V S_w C_{D_0} + C_{D_1}\frac{W}{V} + C_{D_2}\frac{W^2}{\tfrac{1}{2}\rho V^3 S_w}\right)\right] \tag{B153}$$

and the costate variable λ_1 can be found from

$$\lambda_1' = -\frac{\partial H}{\partial V} = \frac{\lambda_1}{m}\left(\frac{T}{V^2} + \frac{1}{2}\rho S_w C_{D_0} - C_{D_1}\frac{W}{V^2} - 3C_{D_2}\frac{W^2}{\tfrac{1}{2}\rho V^4 S_w}\right) \tag{B154}$$

The optimal thrust must minimize the expression

$$\left(\lambda_0 + \frac{\lambda_1}{mV}\right)T^* \tag{B155}$$

If $\left(\lambda_0 + \frac{\lambda_1}{mV}\right) > 0$, this expression is minimized with $T^* = 0$; if $\left(\lambda_0 + \frac{\lambda_1}{mV}\right) < 0$, the expression is minimized with $T^* = 0$; if $\left(\lambda_0 + \frac{\lambda_1}{mV}\right) = 0$, the optimal thrust is said to be singular.

Consider the case where $\lambda_0 = 0$. In order for $\left(\lambda_0 + \frac{\lambda_1}{mV}\right) = 0$, then λ_1 must also be zero. This violates the non-triviality condition. Therefore, since λ_0 can only take the values 0 and 1, we know that $\lambda_0 = 1$. For this case, The expression $\left(\lambda_0 + \frac{\lambda_1}{mV}\right) = 0$ requires that $\lambda_1 = -mV$. Taking the derivative of λ_1 gives $\lambda_1' = -mV'$. Using this expression and Eq.

(B151) in Eq. (B154) gives

$$-\frac{T^*}{V}+\frac{1}{2}\rho V S_w C_{D_0}+C_{D_1}\frac{W}{V}+C_{D_2}\frac{W^2}{\frac{1}{2}\rho V^3 S_w}=-\frac{T^*}{V}-\frac{1}{2}\rho V S_w C_{D_0}+C_{D_1}\frac{W}{V}+3C_{D_2}\frac{W^2}{\frac{1}{2}\rho V^3 S_w} \tag{B156}$$

Simplifying Eq. (B156) gives

$$\rho V S_w C_{D_0}-2C_{D_2}\frac{W^2}{\frac{1}{2}\rho V^3 S_w}=\frac{\partial D}{\partial V}=0 \tag{B157}$$

For the singular case, the optimal thrust can be found from

$$\frac{\partial H'}{\partial T}=\frac{mV\lambda_1'-\lambda_1 mV'}{m^2V^2}=0 \tag{B158}$$

Equation (B158) can be expanded using Eqs. (B154) and (B99) and solved for T^* to give

$$T^*=\frac{1}{2}\rho V^2 S_w C_{D_0}\frac{\left(1-\frac{1}{mV}\right)}{\left(1+\frac{1}{mV}\right)}+C_{D_1}W+C_{D_2}\frac{W^2}{\frac{1}{2}\rho V^2 S_w} \tag{B159}$$

Thus, the optimal thrust can be written as

$$T^*=\begin{cases}0 & \text{for} \quad \left(\lambda_0+\frac{\lambda_1}{mV}\right)>0\\ T_{\max} & \text{for} \quad \left(\lambda_0+\frac{\lambda_1}{mV}\right)<0\\ \frac{1}{2}\rho V^2 S_w C_{D_0}\frac{\left(1-\frac{1}{mV}\right)}{\left(1+\frac{1}{mV}\right)}+C_{D_1}W+C_{D_2}\frac{W^2}{\frac{1}{2}\rho V^2 S_w} & \text{for} \quad \left(\lambda_0+\frac{\lambda_1}{mV}\right)=0\end{cases} \tag{B160}$$

B.3.5 Minimum Thrust Over Specified Range with Nonzero Climb Angle

Consider the case where thrust is to be minimized over specified range with known initial states, and the aircraft is allowed to change altitude. For this case, the dynamic constraints are the equations of motion given in Eqs. (B97)-(B99). Here, we will assume that the angle

of attack (and therefore the lift coefficient) is constant, the density does not change with altitude, and the climb angle γ is small. Under these assumptions, the optimization problem can be summarized as

$$\begin{aligned}
\text{minimize:} \quad & \int_{x_0}^{x_f} T dx \\
\text{with respect to:} \quad & T(x),\ V(x),\ h(x),\ \gamma(x) \\
\text{subject to:} \quad & h' = \gamma \\
& \gamma' = \frac{\rho S_w C_L}{2m} - \frac{W}{mV^2} \\
& V' = \frac{T - \frac{1}{2}\rho V^2 S_w C_D - W\gamma}{mV} \\
& h(0) = h_0, \quad \lambda_h(t_f) = 0 \\
& \gamma(0) = \gamma_0, \quad \lambda_\gamma(t_f) = 0 \\
& V(0) = V_0, \quad \lambda_V(t_f) = 0 \\
& 0 \le T \le T_{\max}
\end{aligned}$$

The Hamiltonian is written as

$$H = \lambda_0 T + \lambda_h \gamma + \lambda_\gamma \left(\frac{\rho S_w C_L}{2m} - \frac{W}{mV^2} \right) + \lambda_V \left(\frac{T - \frac{1}{2}\rho V^2 S_w C_D - W\gamma}{mV} \right) \tag{B161}$$

The costate variables, λ_h, λ_γ, and λ_V are found from

$$\lambda_h' = -\frac{\partial H}{\partial h} = 0 \tag{B162}$$

$$\lambda_\gamma' = -\frac{\partial H}{\partial \gamma} = -\lambda_h + \lambda_V \frac{g}{V} \tag{B163}$$

$$\lambda_V' = -\frac{\partial H}{\partial V} = -\frac{2\lambda_\gamma g}{V^3} + \lambda_V \left(\frac{T - \frac{1}{2}\rho V^2 S_w C_D - W\gamma}{V^2} + \rho S_w C_D \right) \tag{B164}$$

The optimal thrust is found by minimizing the expression

$$\left(\lambda_0 + \frac{\lambda_V}{mV}\right) T^* \tag{B165}$$

If $\left(\lambda_0 + \frac{\lambda_V}{mV}\right) > 0$, this expression is minimized with $T^* = 0$; if $\left(\lambda_0 + \frac{\lambda_V}{mV}\right) < 0$, the expression is minimized with $T^* = 0$; if $\left(\lambda_0 + \frac{\lambda_V}{mV}\right) = 0$, the optimal thrust is said to be singular.

Consider the case where $\lambda_0 = 0$. In order for $\left(\lambda_0 + \frac{\lambda_V}{mV}\right) = 0$, then λ_V must also be zero. This violates the non-triviality condition. Therefore, since λ_0 can only take the values 0 and 1, we know that $\lambda_0 = 1$. For this case, The expression $\left(\lambda_0 + \frac{\lambda_V}{mV}\right) = 0$ requires that $\lambda_V = -mV$. Taking the derivative of λ_V gives $\lambda_V' = -mV'$. Using Eq. (B151) in this expression, and comparing the result to Eq. (B164) gives

$$-\frac{T^* - \frac{1}{2}\rho V^2 S_w CD - W\gamma}{V} = -\frac{2\lambda_\gamma g}{V^3} + \lambda_V \left(\frac{T^* - \frac{1}{2}\rho V^2 S_w C_D - W\gamma}{V^2} + \rho S_w C_D\right) \tag{B166}$$

Using the requirement $\lambda_V = -mV$, Eq. (B166) reduces to

$$\frac{2\lambda_\gamma g}{V^3} = \rho V S_w C_D \tag{B167}$$

which can be solved for λ_γ to give

$$\lambda_\gamma = -\frac{\rho V^4 S_w C_D}{2g} \tag{B168}$$

Taking the derivative of Eq. (B168) and comparing it to Eq. (B163) gives

$$-\frac{4}{2g}\rho C_D V^3 V' = -\frac{2}{g}\rho C_D V^3 \frac{T^* - \frac{1}{2}\rho V^2 S_w C_D - W\gamma}{mV} = -\lambda_h + \lambda_V \frac{g}{V} \tag{B169}$$

which can be solved for T^* to give

$$T^* = -\frac{W}{2\rho S_w C_D V^2}\left(-\lambda_h + \lambda_V \frac{g}{V}\right) + \tfrac{1}{2}\rho V^2 S_w C_D + W\gamma \tag{B170}$$

Thus, the optimal thrust can be summarized as

$$T^* = \begin{cases} 0 & \text{for} \quad \left(\lambda_0 + \dfrac{\lambda_V}{mV}\right) > 0 \\ T_{\max} & \text{for} \quad \left(\lambda_0 + \dfrac{\lambda_V}{mV}\right) < 0 \\ -\dfrac{W}{2\rho S_w C_D V^2}\left(-\lambda_h + \lambda_V \dfrac{g}{V}\right) + \tfrac{1}{2}\rho V^2 S_w C_D + W\gamma & \text{for} \quad \left(\lambda_0 + \dfrac{\lambda_V}{mV}\right) = 0 \end{cases} \tag{B171}$$

B.3.6 Minimum Drag Over Given Range with Nonzero Climb Angle

Suppose we wish to minimize the integral of the drag over a specified range with a variable climb angle, known initial conditions on the states, and no final conditions on the states. The optimization problem can be summarized for this case as follows:

$$\begin{aligned} \text{minimize:} \quad & \int_{x_0}^{x_f} D\,dx \\ \text{with respect to:} \quad & T(x),\ V(x),\ h(x),\ \gamma(x) \\ \text{subject to:} \quad & h' = \gamma \\ & \gamma' = \frac{\rho S_w C_L}{2m} - \frac{W}{mV^2} \\ & V' = \frac{T - \frac{1}{2}\rho V^2 S_w C_D - W\gamma}{mV} \\ & h(0) = h_0, \quad \lambda_h(t_f) = 0 \\ & \gamma(0) = \gamma_0, \quad \lambda_\gamma(t_f) = 0 \\ & V(0) = V_0, \quad \lambda_V(t_f) = 0 \\ & 0 \le T \le T_{\max} \end{aligned}$$

The Hamiltonian is written as

$$H = \lambda_0 \tfrac{1}{2}\rho V^2 S_w C_D + \lambda_h \gamma + \lambda_\gamma \left(\frac{\rho S_w C_L}{2m} - \frac{W}{mV^2} \right) + \lambda_V \left(\frac{T - \frac{1}{2}\rho V^2 S_w C_D - W\gamma}{mV} \right) \quad \text{(B172)}$$

The costate variables, λ_h, λ_γ, and λ_V are found from

$$\lambda_h' = -\frac{\partial H}{\partial h} = 0 \quad \text{(B173)}$$

$$\lambda_\gamma' = -\frac{\partial H}{\partial \gamma} = -\lambda_h + \lambda_V \frac{g}{V} \quad \text{(B174)}$$

$$\lambda_V' = -\frac{\partial H}{\partial V} = -\lambda_0 \rho V S_w C_D - \frac{2\lambda_\gamma g}{V^3} + \frac{\lambda_V}{m} \left(\frac{T - \frac{1}{2}\rho V^2 S_w C_D - W\gamma}{V^2} + \rho S_w C_D \right) \quad \text{(B175)}$$

The optimal thrust is found by minimizing the expression

$$\frac{\lambda_V}{mV} T^* \quad \text{(B176)}$$

If $\frac{\lambda_V}{mV} > 0$, this expression is minimized with $T^* = 0$; if $\frac{\lambda_V}{mV} < 0$, the expression is minimized with $T^* = T_{\max}$; if $\frac{\lambda_V}{mV} = 0$, λ_V must also be zero, and the optimal thrust is said to be singular.

Note that because the transversality conditions require that $\lambda_h(x_f) = 0$, and Eq. (B173) shows that $\lambda_h' = 0$, we know that $\lambda_h(x) = 0$ along any trajectory. Along the singular arc, $\lambda_V = 0$. Equation (B174) shows that when $\lambda_V = 0$ and $\lambda_h = 0$, $\lambda_\gamma' = 0$, which, by the transversality condition requires that $\lambda_\gamma(x) = 0$.

Along the singular arc, λ_V is zero, $\lambda_V' = 0$. Comparing this to Eq. (B175) gives

$$0 = -\lambda_0 \rho V S_w C_D - \frac{2\lambda_\gamma g}{V^3} + \frac{\lambda_V}{m} \left(\frac{T^* - \frac{1}{2}\rho V^2 S_w C_D - W\gamma}{V^2} + \rho S_w C_D \right) \quad \text{(B177)}$$

Because $\lambda_V = 0$ and $\lambda_\gamma' = 0$, Eq. (B177) requires that $\lambda_0 = 0$, which violates the non-triviality condition.

APPENDIX C

CODE EXAMPLES

C.1 Wing-Structure Prediction

The following example code extracts can be used to predict the wing-structure weight for a desired input aircraft, as described in Chapter 3.

C.1.1 Wing-Structure Weight Code Main

```
'''
wing_structure.py

Used to predict the wing-structure weight and its distribution for a
specified aircraft

Calls the wing_structure_m3 module, which contains most calculations for the
wing-structure weight
'''

#!/usr/bin/env python
import numpy as np
import sys
import time
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
#User-Defined Module Containing functions used for
#-Reading Input File
#-Geometry Setup
#-Discretization
#-Moment Calculations
#-Wing Structure Calculations
#-Non-Structural Weight Calculations
#-Solver
import  wing_structure_m3 as ws
from matplotlib import rc

data_format="{0:<30}{1:<24.16f}\n"
dist_header="{0:<24}{1:<24}{2:<24}{3:<24}{4:<32}{5:<24}\n"
dist_format="{0:<24.16f}{1:<24.16f}{2:<24.16f}{3:<24.16f}{4:<32.16f}{5:<24.16f}\n"

plot_flag = False
```

C.1.1 Wing-Structure Weight Module (python)

```
"""
wing_structure_m3.py

Calculates the wing-structure weight required to support the bending
moments produced by a given lift distribution and payload
distribution for a
given wing geometry.

This module is intended to allow a user to output the required
wing-structure
weight given the parameters described in the title. It takes an
input file
containing the aircraft geometry, limits, and flight condition. It
calculates
the bending moments for the hard-landing and maneuvering flight
limits, for
the deflection-limited and stress-limited designs (if desired). It
returns
the bending moment distribution, wing-structure weight distribution,
total
wing-structure weight, and the induced drag. The process follows
that outlined
in "Minimizing Induced Drag for Wings with Arbitrary Planform and
Weight
Distribution" with the option to superimpose different payload
distributions.

Parameters:
----------
input.json file : the input file containing required parameters
See README in for more information.

Returns:
----------
plane.w.structure_distribution : array containing the wing-structure
weight
                                 distribution
plane.w.structure : the total wing-structure weight
plane.moments : array containing the limiting bending moments
plane.induced_drag : the induced drag produced under these conditions

Notes:
----------
```

```
---------
This is a rewrite of a previous version. This version was rewritten in order
to accommodate the superposition of several payload distributions.

Certain parameters in the input file can be specified as functions. For
example, the chord distribution, the thickness-to-chord ratio, and the payload
distributions can all be specified as functions. Some of the functions are
built in and are included in the dist_functions.py script. The user also has
the option of specifying a custom distribution, which is contained in the
user_functions.py script.

Example:
-------------------------------------------------------------------------
---------
plane = wing_structure_m2.Domain('input.json',comment_flag)
plane.solver(tolerance)

"""
import sys
import os
import numpy as np
import math as ma
import json
from collections import OrderedDict
import time
import dist_functions_2
from scipy import integrate
from scipy import interpolate
from decimal import *

getcontext().prec = 16

class Domain(object):
    """The Domain class contains all case information.

    The domain class contains all of the aircraft and flight condition values,
    both parameters and variables.

    Attributes
    ----------
    Wing : class
        Class containing all of the wing data and methods specific to the wing.
```

```
Spar : class
    Class containing all of the spar data and methods specific
    to the spar.
Weight : class
    Class containing all of the spar data and methods specific
    to the spar.
moment : Array
    Array containing the calculated bending moment for the input
    payload distribution at each spanwise location specified by
    the base
    grid.
limits : array
    Array of length 2 containing the positive and negative load
    limits
density : float
    The density of air at the flight condition
velocity : float
    The velocity of the aircraft at the flight condition.

Methods
-------
__init__:
    initializes the domain class.
initialize_constants
    initializes and stores constants from input file.
initialize_payload_distributions
    initializes and stores payload distributions from input file
initialize_distributions
    initializes all distributions excluding the payload
    distribution

"""

def __init__(self, filename, comment_flag=False):
    """Initializes the Domain class

    The __init__ method initializes the Domain class with the
    information from the input file.

    Note
    ----
    This is a private method.

    Parameters
    ----------
    filename : str
        The name of the input file, in string form.
    comment_flag: Boolean
        Flag to specify whether the init function should display
```

```
            information about the initialization of the Domain Class.

        """
        # Create instances of each sublcass
        self.wing = self.Wing()
        self.spar = self.Spar()
        self.weight = self.Weight()

        self.comment_flag = comment_flag

        # Read in the input file
        with open(filename) as input_file:
            data = json.load(input_file, object_pairs_hook=OrderedDict)

        # Read Constants from Input File
        self.read_constants(data)

        # Initialize all distributions
        self.initialize_distributions(data)

        # Set Distributions
        self.set_distributions(data)

    def read_constants(self, data):
        """Initializes and stores constants from the input file

        The initialize_constants method reads in all constants (not
        distributions) from the input file and stores them inside      ↵
        the Domain
        class. Many of the constants are stored in the sublcasses     ↵
        Wing, Spar,
        and Weight.

        Parameters
        ----------
        data : dict
            Dictionary containing information from the input file.
        comment_flag: Boolean
            Flag to specify whether the init function should display
            information about the initialization of the constants.

        """

        # Store constants general to the Domain Class
        self.density = np.float128(data["flight"]["density"])
        self.velocity = np.float128(data["flight"]["velocity"])

        self.limits = np.zeros(2, dtype=np.float128)
        self.limits[0] = np.float128(data["limits"]["maneuvering"])
```

```
        self.limits[1] = np.float128(data["limits"]["hard_landing"])

        # Store constants for the Wing subclass
        self.wing.read_constants(data)

        # Store constants for the Spar sublcass
        self.spar.read_constants(data)

        # Store constants for the Weight subclass
        self.weight.read_constants(data)

    def initialize_distributions(self, data):
        """Initializes all distributions excluding the payload distributions

        The initialize_distributions method initializes all remaining
        distributions based on information from the input file and the spanwise
        locations specified in the weight.net_distribution list of arrays.
        Specifically, it sets a lift, wing_structure weight, bending_moment,
        chord, and thickness-to-chord ratio distribution for each set of
        spanwise locations given in the weight.net_distribution list of arrays.
        It stores each of these distributions in a respective list of arrays.
        (If a distribution is given in a file, instead of by a function, values
        will be linearly interpolated to find values at the correct spanwise
        locations.)

        Parameters
        ----------
        data : dict
            Dictionary containing information from the input file.
        comment_flag: Boolean
            Flag to specify whether the init function should display
            information about the initialization of the distributions.

        """

        # Initialize moment distributions
        self.moment = np.zeros(self.wing.grid+1, dtype=np.float128)

        # Initialize Weight class distributions
        self.weight.initialize_distributions(data)
```

```
    # Initialize Wing class distributions
    self.wing.initialize_distributions(data)

    # Initialize Spar class distributions
    self.spar.initialize_distributions(data)

def set_distributions(self, data):
    """ Sets distributions.

    This function sets all distributions based on built-in functions
    in the dist_functions script, and by custom functions defined in the
    user_functions script. These functions must return values, spanwise
    locations, and angle locations for each distribution

    Parameters
    ----------
    data : dict
        dictionary containing all input files values.
    """

    self.wing.chord.set_values(self)

    # Handle fixed wing loading
    if self.wing.loading_type == 'fixed':
        self.wing.area = self.weight.total/self.wing.loading

    # Handle Max_Lift coefficient
    elif self.wing.loading_type == 'lift':
        self.wing.lift_coeffs = np.divide(self.wing.lift_distribution,
                                          self.wing.chord.values)
        if self.wing.chord.definition == 'elliptic':
            self.wing.lift_coeffs[-1] = 0.0
        maxloc = np.argmax(self.wing.lift_coeffs)
        self.wing.area = (
            self.wing.area *
            self.wing.lift_distribution[maxloc] *
            self.weight.total*np.float128(2.0)/self.density /
            self.wing.stall_velocity**np.float128(2)/self.wing.max_lift_coeff /
            self.wing.span/self.wing.chord.values[maxloc])

    self.wing.chord.set_values(self)
    # if self.wing.chord.definition == 'file':
```

```
            # ft = np.multiply(self.wing.chord.values,
                               # np.sin(self.wing.angle_location))
            # area = -self.wing.span*integrate.simps(ft,
            self.wing.angle_location)
            # self.wing.chord.values =
            self.wing.chord.values*self.wing.area/area

        self.wing.thickness_chord.set_values(self)

        self.wing.max_thickness = np.multiply(self.wing.chord.values,
                                              self.wing.thickness_chor
                                              d.values)
        self.spar.height.set_values(self)
        for i in range(1, len(self.weight.net_distributions)):

            self.weight.net_distributions[i].set_values(self)

        self.weight.net_weight(self.wing)

        self.weight.base_net_weight_distribution(self.wing)

        self.spar.set_spar(self, data)

    def update_values(self):
        """updates values according to constraints

        constraints are implied from the input file.

        Parameters
        ----------

        """
        self.wing.chord.set_values(self)

        # Update Weight
        self.weight.net_weight(self.wing)

        self.weight.total = self.weight.net+self.weight.wing_structure

        # Reset root weight, if needed
        if self.weight.root_type == 'ratio':
            self.weight.root =
            self.weight.root_total_ratio*self.weight.total

        # Handle fixed wing loading
        if self.wing.loading_type == 'fixed':
            self.wing.area = self.weight.total/self.wing.loading
```

```
# Handle Max_Lift coefficient
elif self.wing.loading_type == 'lift':
    self.wing.lift_coeffs =
    np.divide(self.wing.lift_distribution,
                                      self.wing.chord.values)
    if self.wing.chord.definition == 'elliptic':
        self.wing.lift_coeffs[-1] = 0.0
    maxloc = np.argmax(self.wing.lift_coeffs)
    self.wing.area = (
        self.wing.area *
        self.wing.lift_distribution[maxloc] *
        self.weight.total*np.float128(2.0)/self.density /
        self.wing.stall_velocity**np.float128(2)/self.wing.max
        _lift_coeff /
        self.wing.span/self.wing.chord.values[maxloc])

# print(self.wing.area)

self.wing.chord.set_values(self)
# if self.wing.chord.definition == 'file':
    # ft = np.multiply(self.wing.chord.values,
                     # np.sin(self.wing.angle_location))
    # area = -self.wing.span*integrate.simps(ft,
    self.wing.angle_location)
    # self.wing.chord.values =
    self.wing.chord.values*self.wing.area/area

self.wing.max_thickness = np.multiply(
    self.wing.chord.values, self.wing.thickness_chord.values)

self.spar.height.set_values(self)

self.spar.set_proportionality_coefficient(self.wing)

# ft = np.multiply(
    # self.wing.chord.values, np.sin(self.wing.angle_location))
# self.wing.area = -self.wing.span*integrate.simps(
    # ft, self.wing.angle_location)

self.wing.loading = self.weight.total/self.wing.area

# print(self.wing.area)

# reset net weight distributions
for k in range(1, len(self.weight.net_distributions)):

    self.weight.net_distributions[k].set_values(self)
```

```
        self.weight.base_net_weight_distribution(self.wing)


        # Update Weight
        self.weight.net_weight(self.wing)
        self.weight.total = self.weight.net+self.weight.wing_structure

        # Reset root weight, if needed
        if self.weight.root_type == 'ratio':
            self.weight.root =
            self.weight.root_total_ratio*self.weight.total

    def solver(self, tolerance, comment_flag=False):
        """Finds the wing-structure weight

        This function finds the wing-structure weight distribution
        and wing-
        structure weight, along with the bending moment distribution.

        Parameters
        ----------
        tolerance : float
            stopping criterion for the iterative solver.
        comment_flag: Boolean
            Flag to specify whether the solver function should display
            information.

        """

        # open file to write convergence data
        f = open('error.txt', 'w')
        f.write('i'+'\t'+'error'+'\t'+'Wing structure weight'+'\n')

        # initialize error and iteration counter
        error = np.float128(1.0)
        iteration = 0
        maxiter = 200

        # loop until convergence
        while error > tolerance:  # and iteration < maxiter:

            prev = self.weight.wing_structure

            # Calculate bending moments
            self.calculate_moment()

            # Update wing-structure weight
            self.weight.calculate_structure(self)
```

```
            # Update error and iteration
            if prev == 0.0:
                error = np.float128(1.0)
            else:
                error = np.absolute(self.weight.wing_structure-prev)
                # error =                                                 ⏎
                np.absolute(((self.weight.wing_structure-prev)
                                    # /                                 ⏎
                                    self.weight.wing_structure)*100.0)
            iteration += 1

            if comment_flag is True:
                print('iteration:   ', iteration, '\t', 'error: ',       ⏎
                error,
                      'Wing Structure Weight: ',                         ⏎
                      "%.16f"%self.weight.wing_structure)
            f.write(str(iteration)+'\t'+str(error)+'\t'+
                    str(self.weight.wing_structure)+'\n')

            # Update Relevant Parameters
            self.update_values()

            if iteration == maxiter:
                if comment_flag is True:
                    print('maximum number of iterations reached.')
                break
        if comment_flag is True:
            print(' ')
            print('Wing Structure Weight Converged in:                   ⏎
            '+str(iteration)+' iterations')
            print(' ')
        f.close()
        self.calculate_induced_drag()

    def calculate_moment(self):
        """calculates the bending moments on the wing

        Parameters
        ----------

        """
        ng = np.zeros(self.wing.grid+1, dtype=np.float128)
        nm = np.zeros(self.wing.grid+1, dtype=np.float128)
        for i in range(0, self.wing.grid+1):
            ft_lift_ng = np.multiply(np.multiply(np.multiply(
               self.weight.total/self.limits[1]/np.float128(2.0),
               np.sin(self.wing.angle_location[i:])),
               self.wing.lift_distribution[i:]),
               np.subtract(np.cos(self.wing.angle_location[i:]),
```

```
                ma.cos(self.wing.angle_location[i])))
ft_structure_ng = np.multiply(np.multiply(np.multiply(
   -self.wing.span/np.float128(2.0),
   np.sin(self.wing.angle_location[i:])),
   self.weight.structure_distribution[i:]),
   np.subtract(np.cos(self.wing.angle_location[i:]),
                ma.cos(self.wing.angle_location[i])))
ft_net_ng = []
for k in range(1, len(self.weight.net_distributions)):
    ft_net_ng.append(np.multiply(np.multiply(np.multiply(
       -self.wing.span/np.float128(2.0),
                                                        ↵
       np.sin(self.weight.net_distributions[k].angle_locat ↵
       ions)),
       self.weight.net_distributions[k].values),
       np.subtract(np.cos(
        self.weight.net_distributions[k].angle_locations),
                    ma.cos(self.wing.angle_location[i]))))

ft_lift_nm = np.multiply(np.multiply(np.multiply(
   -self.weight.total/np.float128(2.0),
   np.sin(self.wing.angle_location[i:])),
   self.wing.lift_distribution[i:]),
   np.subtract(np.cos(self.wing.angle_location[i:]),
                ma.cos(self.wing.angle_location[i])))
ft_structure_nm = np.multiply(np.multiply(np.multiply(
   self.wing.span/np.float128(2.0),
   np.sin(self.wing.angle_location[i:])),
   self.weight.structure_distribution[i:]),
   np.subtract(np.cos(self.wing.angle_location[i:]),
                ma.cos(self.wing.angle_location[i])))
ft_net_nm = []
for k in range(1, len(self.weight.net_distributions)):
    ft_net_nm.append(np.multiply(np.multiply(np.multiply(
       self.wing.span/np.float128(2.0),
                                                        ↵
       np.sin(self.weight.net_distributions[k].angle_locat ↵
       ions)),
       self.weight.net_distributions[k].values),
       np.subtract(np.cos(
        self.weight.net_distributions[k].angle_locations),
                    ma.cos(self.wing.angle_location[i]))))

if (i == self.wing.grid):
    self.moment[i] = np.float128(0.0)
else:
    ng[i] =                                             ↵
    -self.limits[1]*self.wing.span/np.float128(2.0)*(integ ↵
    rate.simps(
```

```
    ft_lift_ng,
    self.wing.angle_location[i:])
    + integrate.simps(
    ft_structure_ng,
    self.wing.angle_location[i:]))
nm[i] =                                                 ⤶
self.limits[0]*self.wing.span/np.float128(2.0)*(integr ⤶
ate.simps(
    ft_lift_nm,
    self.wing.angle_location[i:])
    + integrate.simps(
    ft_structure_nm,
    self.wing.angle_location[i:]))
for k in range(1, len(self.weight.net_distributions)):
    if self.wing.angle_location[i] \
      >                                                 ⤶
      self.weight.net_distributions[k].angle_locations ⤶
      [0]:

        ng[i] +=                                        ⤶
        -self.limits[1]*self.wing.span/np.float128(2.0 ⤶
        ) \
          * integrate.simps(
            ft_net_ng[k-1],
                                                        ⤶
            self.weight.net_distributions[k].angle_loc ⤶
            ations)
        nm[i] +=                                        ⤶
        self.limits[0]*self.wing.span/np.float128(2.0) ⤶
         \
            * integrate.simps(
             ft_net_nm[k-1],
                                                        ⤶
             self.weight.net_distributions[k].angle_lo ⤶
             cations)
    elif self.wing.angle_location[i] \
        <=                                              ⤶
        self.weight.net_distributions[k].angle_locatio ⤶
        ns[0]\
        and self.wing.angle_location[i] \
        >=                                              ⤶
        self.weight.net_distributions[k].angle_locatio ⤶
        ns[
         self.weight.net_distributions[k].grid]:

        start_loc = np.argmax(np.subtract(
                                                        ⤶
          self.weight.net_distributions[k].angle_locat ⤶
          ions,
```

```
                        self.wing.angle_location[i])
                          <= 1e-16)

                    ng[i] +=                                          ↵
                    -self.limits[1]*self.wing.span/np.float128(2.0 ↵
                    ) \
                        * integrate.simps(
                         ft_net_ng[k-1][start_loc:],
                                                                      ↵
                         self.weight.net_distributions[k].angle_lo ↵
                         cations[
                            start_loc:])
                    nm[i] +=                                          ↵
                    self.limits[0]*self.wing.span/np.float128(2.0) ↵
                     \
                        * integrate.simps(
                         ft_net_nm[k-1][start_loc:],
                                                                      ↵
                         self.weight.net_distributions[k].angle_lo ↵
                         cations[
                            start_loc:])

            self.moment[i] = max(np.absolute(ng[i]),                  ↵
            np.absolute(nm[i]))

    def calculate_induced_drag(self):
        """Determines the induced drag.

        Parameters
        ----------

        """
        Bsum = 0.0
        for i in range(0, len(self.wing.lift_dist_coeffs)):
                Bsum = Bsum + (i+2) *                                 ↵
                self.wing.lift_dist_coeffs[i]**np.float128(2)
        self.induced_drag =                                           ↵
        (np.float128(2.0)*(self.weight.total/self.wing.span)**np.float ↵
        128(2)) \
            /                                                         ↵
            (ma.pi*self.density*self.velocity**np.float128(2))*(1.0   ↵
            + Bsum)

    class Wing(object):
        """The Wing class contains all wing information.

        The wing class contains attributes pertaining to the wing,    ↵
        primarily
        geometric attributes.
```

```
Attributes
----------
base_grid: integer
    Grid used for visualization of data and for calculation
    of the
    lift distribution.
grids : list
    List of integers containing the number of nodes used to
    specify
    each payload distribution.
spanwise_locations : list
    List of arrays containing spanwise location data from the
    input payload distributions. These correspond to the
    distributions
    included in all distribution lists respectively.
    Generally, these
    should be cosine clustered according to the full semispan.
base_span_loc : array
    the base spanwise location used in data visualization
    and in the
    calcluation of the lift distribution.
angle_locations : list
    List of arrays containing the angles corresponding to
    the spanwise
    locations given in the input payload distributions. The
    angle
    locations should be evenly spaced for each payload
    distribution,
    making the spanwise locations cosine clustered.
lift_distribution : array
    array containing the lift distribution at locations
    specified by
    the base grid.
chord : array
    array containing the chord data corresponding to the
    spanwise
    locations given by the base grid
chord_type : string
    String specifying if the chord values are interpolated
    from an
    input file or specified using a function.
thickness_chord : array
    array containing thickness-to-chord ratio data
    corresponding to the
     spanwise locations given by the base grid
    distributions.
thickness_chord_type : string
    String specifying if the thickness-to-chord values are
```

```
    interpolated
    from an input file or specified using a function.
max_thickness : array
    array containing the maximum airfoil thickness at each
    spanwise
    location specified by the base grid.
area : float
    The total wing area, including both semispans.
span : float
    The total wingspan. Semispan is span/2
loading : float
    The wing loading, defined as Total weight/Area.
taper_ratio : float
    The taper ratio for tapered wings. Only used in the case of
    linearly tapered wings.
root_thickness_chord : float
    The thickness-to-chord ratio at the wing root. Only used
    in the
    case of linearly tapered thickness-to-chord ratio.
tip_thickness_chord : float
    The thickness-to-chord ratio at the wing tip. Only used
    in the
    case of linearly tapered thickness-to-chord ratio.

Methods
-------
initialize_constants:
    initializes the wing class constants.
initialize_distributions:
    initializes the wing class distributions.

"""

def read_constants(self, data):
    """Initializes the wing class constants.

    Parameters
    ----------
    data : dict
        dictionary containing all of the input file data

    """
    self.grid = data["wing"].get("grid", 0.0)
    self.area = np.float128(data["wing"].get("wing_area", 0.0))
    self.span = np.float128(data["wing"].get("wing_span", 0.0))
    self.loading = np.float128(data["wing"].get("loading",
    0.0))
    self.max_lift_coeff =
    np.float128(data["wing"].get("max_lift_coefficient", 0.0))
```

```
        self.stall_velocity =
        np.float128(data["wing"].get("stall_velocity", 0.0))

        temp_coeffs = data["lift_distribution"]["B"]
        self.lift_dist_coeffs =
        np.zeros(len(data["lift_distribution"]["B"]),
        dtype=np.float128)
        self.lift_dist_coeffs = temp_coeffs

        if "loading" in data["wing"]:
            self.loading_type = 'fixed'
        elif "max_lift_coefficient" in data["wing"]:
            self.loading_type = 'lift'
        else:
            self.loading_type = 'none'

        if "function" in data["wing"]["chord"]["definition"]:
            if "taper" in
            data["wing"]["chord"]["definition"]["function"]:
                self.taper_ratio =
                np.float128(data["wing"]["chord"]["definition"][
                    "function"].get("taper", 'none'))
        else:
            self.taper_ratio = 'none'

        if "function" in
        data["wing"]["thickness_chord"]["definition"]:
            if "root_tip" in
            data["wing"]["thickness_chord"]["definition"][
              "function"]:
                self.root_thickness_chord =
                np.float128(data["wing"][
                    "thickness_chord"]["definition"]["function"][
                        "root_tip"].get("root", 'none'))
                self.tip_thickness_chord =
                np.float128(data["wing"][
                    "thickness_chord"]["definition"]["function"][
                        "root_tip"].get("tip", 'none'))
        else:
            self.root_thickness_chord = 'none'
            self.tip_thickness_chord = 'none'

    def initialize_distributions(self, data):
        """Initializes the wing class distributions.

        Parameters
        ----------
        data : dict
            dictionary containing all of the input file data
```

```
        """

        # Discretize the wing according to the base grid
        self.discretize()

        # Calculate Lift Distribution
        self.calculate_lift_distribution()

        # Initialize other distributions
        self.chord = Input_Distribution(data,
        data["wing"]["chord"])

        if self.chord.definition == 'constant':
            self.loading_type = 'none'

        self.thickness_chord = Input_Distribution(
            data, data["wing"]["thickness_chord"])

        self.max_thickness = np.multiply(self.chord.values,
                                         self.thickness_chord.valu
                                         es)

    def discretize(self):
        """ Discretizes the wing according to the grid

        Parameters
        ----------
        """
        self.spanwise_location = np.zeros(self.grid,
        dtype=np.float128)
        self.angle_location = np.zeros(self.grid,
        dtype=np.float128)
        i = np.linspace(0, self.grid, self.grid+1)
        self.angle_location = np.subtract(
            np.float128(ma.pi)/np.float128(2.0),
            np.divide(np.multiply(np.float128(ma.pi)/np.float128(2
            .0), np.float128(i)),
                                  np.float128(self.grid)))
        self.spanwise_location = np.multiply(
            self.span/np.float128(2.0),
            np.cos(self.angle_location))

    def calculate_lift_distribution(self):
        """Calculates the Lift distribution

        Parameters
        ----------
```

```
        """
        self.lift_distribution = np.zeros(self.grid+1,
        dtype=np.float128)
        for i in range(0, self.grid+1):
            Bsinsum = np.float128(0.0)
            for k in range(0, len(self.lift_dist_coeffs)):
                Bsinsum = Bsinsum + self.lift_dist_coeffs[k] \
                    * ma.sin((k + np.float128(2)) *
                    self.angle_location[i])

            self.lift_distribution[i] =
            (np.float128(4.0)/np.float128(ma.pi)) \
                * (ma.sin(self.angle_location[i])+Bsinsum)

class Spar(object):
    """The Spar class contains all spar information.

    The spar class contains attributes pertaining to the spar,
    primarily
    geometric and material attributes.

    Attributes
    ----------
    beam_type : string
        String specifying the beam type, if given in the input
        file. If no
        beam type is given in the input file, then beam_type is
        set to
        'null'
    height : array
        Array containing the spar height corresponding to the
        spanwise
        locations given by the base grid
    height_type : string
        String specifying if the height is based on a ratio of
        height to
        thickness or if it is interpolated and scaled from an
        input file.
    proportionality_coefficient : array
        Array containing the proportionality coefficient
        corresponding to
        the spanwise locations given by the base grid.
    max_stress : float
        The maximum allowable stress at any point in the spar.
    max_deflection : float
        The maximum allowable deflection at any point in the spar.
    specific_weight : float
        The specific weight of the beam material.
    modulus_elasticity : float
```

```
    The modulus of elasticity of the beam material.
stress_shape_factor : float
    The shape factor for the beam cross section for the
    stress-limited
    design. Can be specified or calculated from reference
    beam values.
deflection_shape_factor : float
    The shape factor for the beam cross section for the
    deflection-limited design. Can be specified or
    calculated from
    reference beam values.
inner_width : float
    Reference value for the inner width of a box-beam cross
    section
inner_height : float
    Reference value for the inner height of a box-beam cross
    section
outer_width : float
    Reference value for the outer width of a box-beam cross
    section
flange_height : float
    Reference value for the flange height of an I-beam cross
    section
flange_width : float
    Reference value for the flange width of an I-beam cross
    section
web_width : float
    Reference value for the web width of an I-beam cross
    section

Methods
-------
initialize_constants:
    initializes the wing class constants.
initialize_distributions:
    initializes the wing class distributions.

"""

def read_constants(self, data):
    """Initializes the spar class constants.

    Parameters
    ----------
    data : dict
        dictionary containing all of the input file data

    """
    self.max_stress =
```

```
        np.float128(data["spar"].get("max_stress", 'none'))
        self.specific_weight =
        np.float128(data["spar"].get("specific_weight", 'none'))
        self.grid = np.float128(data["wing"]["grid"])
        self.stress_shape_factor =
        np.float128(data["spar"].get("C_sigma", 0.0))
        self.deflection_shape_factor =
        np.float128(data["spar"].get("C_delta", 0.0))
        self.beam_type = data["spar"].get("beam_type", 'none')
        if 'max_deflection' in data["spar"]:
            self.max_deflection =
            np.float128(data["spar"]["max_deflection"])
            self.modulus_elasticity =
            np.float128(data["spar"]["modulus_elasticity"])
            self.design_name = 'deflection'
        else:
            self.design_name = 'stress'

        if "function" in data["spar"]["height"]["definition"]:
            if "fill" in
            data["spar"]["height"]["definition"]["function"]:
                self.fill_ratio =
                np.float128(data["spar"]["height"]["definition"][
                    "function"].get("fill", 'none'))

    def initialize_distributions(self, data):
        """Initializes the spar class distributions.

        Parameters
        ----------
        data : dict
            dictionary containing all of the input file data

        """
        self.height = Input_Distribution(data,
        data["spar"]["height"])
        # self.stress_shape_factor = Input_Distribution(data,
        data["spar"]["C_sigma"])
        # self.deflection_shape_factor =
        Input_Distribution(data, data["spar"]["C_delta"])
        self.proportionality_coefficient = np.zeros(
            data["wing"]["grid"]+1,
            dtype=np.float128)

    def set_spar(self, plane, data):
        """Sets up remaining spar parameters not from the input
        file.

        Parameters
```

```
        ----------
        plane : class
            instance of the Domain class with initialization
        data : dict
            dictionary containing all of the input file data.

        """

        # set shape coefficient if not already specified.
        # if self.stress_shape_factor.definition == "none":
            # self.set_shape_coefficient(plane.wing, data,
            'stress')
            # if self.deflection_shape_factor.definition == "none":
                # self.set_shape_coefficient(plane.wing, data,
                'deflection')
        if self.stress_shape_factor == np.float128(0.0):
            self.set_shape_coefficient(plane.wing, data, 'stress')
            if self.deflection_shape_factor == np.float128(0.0):
                self.set_shape_coefficient(plane.wing, data,
                'deflection')
        # else:
            # self.stress_shape_factor.set_values(plane)
            # self.deflection_shape_factor.set_values(plane)

        # Set the Proportionality Coefficient
        self.set_proportionality_coefficient(plane.wing)

    def set_proportionality_coefficient(self, wing):
        """ Sets the Proportionality Coefficient

        Considers both the stress- and deflection-limited cases and
        selects the proportionality coefficient at each section
        that is
        lower of the two.

        Parameters
        ----------
        wing : class
            instance of the Domain.Wing class with initialized
            values.

        """
        # Stress-Limited Design
        # stress = np.multiply(
            #
            self.max_stress*np.divide(self.stress_shape_factor.val
            ues,self.specific_weight),
            # np.multiply(wing.thickness_chord.values,
            wing.chord.values))
```

```
stress = np.multiply(
    self.max_stress*self.stress_shape_factor/self.specific_weight,
    np.multiply(wing.thickness_chord.values, wing.chord.values))

# Deflection-Limited Design
# if self.design_type == 'deflection':
ft = np.zeros(wing.grid+1, dtype=np.float128)
ft1 = np.zeros(wing.grid+1, dtype=np.float128)
deflection = np.zeros(wing.grid+1, dtype=np.float128)
if self.design_name == 'deflection':
    for k in range(0, wing.grid+1):
        if wing.chord.values[k] == np.float128(0.0):
            ft[k] = np.float128(0.0)
        else:
            ft[k] = (np.float128(1.)/wing.thickness_chord.values[k]) \
                * (np.float128(1.)/wing.chord.values[k]) \
                * ma.sin(wing.angle_location[k])

    for i in range(1, wing.grid+1):
        ft1[i] = -wing.span/np.float128(2.0)*integrate.simps(
            ft[0:i+1],
            wing.angle_location[0:i+1]) \
            * ma.sin(wing.angle_location[i])

    integral = -wing.span/np.float128(2.0)*integrate.simps(
        ft1,
        wing.angle_location)

    for i in range(0, wing.grid+1):
        # deflection[i] = (self.deflection_shape_factor.values[i]
                         # * self.modulus_elasticity
                         # * (wing.thickness_chord.values[i])
                         # * wing.chord.values[i]
                         # * self.max_deflection) \
                         # / (np.float128(8.)*self.specific_weight*integral)
        deflection[i] = (self.deflection_shape_factor
                         * self.modulus_elasticity
                         * (wing.thickness_chord.values[i])
                         * wing.chord.values[i]
```

```
                                * self.max_deflection) \
                                /                                  ⤶
                                (np.float128(8.)*self.specific_we  ⤶
                                ight*integral)
            else:
                for i in range(0, wing.grid+1):
                    deflection[i] = 1e10

            # Use the smaller of the two proportionality coefficients.
            self.proportionality_coefficient = np.minimum(stress,
                                                          deflection)

            if np.array_equal(stress,                              ⤶
            self.proportionality_coefficient):
                self.design_type = 'stress'
            else:
                self.design_type = 'deflection'

        def set_shape_coefficient(self, wing, data, flag):
            """Sets the shape coefficient for stress and deflection    ⤶
            designs

            Uses reference geometry parameters given in the input      ⤶
            file to
            determine the shape coefficient.

            Parameters
            ----------
            wing : class
                instance of the Domain.Wing class with initialized     ⤶
                values.
            data : dict
                dictionary containing all of the input file data
            flag : string
                either 'stress' or 'deflection', specifying which      ⤶
                shape factor
                to calculate.

            """
            if flag == 'stress':
                if self.beam_type == 'rectangular':
                    self.rectangular_beam(wing, data, flag)
                elif self.beam_type == 'box':
                    self.box_beam(wing, data, flag)
                elif self.beam_type == 'I':
                    self.I_beam(wing, data, flag)
                else:
                    print('WARNING: Spar not fully specified. Please  ⤶
                    select a \
```

```
                valid beam type or specify the shape
                coefficient.')

        elif flag == 'deflection':
            if self.beam_type == 'rectangular':
                self.rectangular_beam(wing, data, flag)
            elif self.beam_type == 'box':
                self.box_beam(wing, data, flag)
            elif self.beam_type == 'I':
                self.I_beam(wing, data, flag)

            # self.deflection_shape_factor.values = \
                # self.stress_shape_factor.values*min(
                    # np.divide(self.height.values,
                    # wing.max_thickness))/np.float128(2.0)
            # if wing.chord.definition == 'elliptic':
                # self.deflection_shape_factor.values[-1] = 0.0
            self.deflection_shape_factor = \
                self.stress_shape_factor*min(
                    np.divide(self.height.values,
                    wing.max_thickness))/np.float128(2.0)
            if wing.chord.definition == 'elliptic':
                self.deflection_shape_factor[-1] = 0.0

    def rectangular_beam(self, wing, data, flag):
        """Sets the shape factor for a rectangular beam

        Parameters
        ----------
        wing : class
            instance of the Domain.Wing class with initialized
            values.
        data : dict
            dictionary containing all of the input file data
        flag : string
            either 'stress' or 'deflection', specifying which
            shape factor
            to calculate.

        """
        # self.stress_shape_factor.values = min(
            # np.divide(self.height.values,
            # wing.max_thickness))/np.float128(6.0)
        # if wing.chord.definition == 'elliptic':
            # self.stress_shape_factor.values[-1] = 0.0

        self.stress_shape_factor = min(
            np.divide(self.height.values,
            wing.max_thickness))/np.float128(6.0)
```

```
        if wing.chord.definition == 'elliptic':
            self.stress_shape_factor[-1] = 0.0

    def box_beam(self, wing, data, flag):
        """Sets the shape factor for a box beam

        Parameters
        ----------
        wing : class
            instance of the Domain.Wing class with initialized
            values.
        data : dict
            dictionary containing all of the input file data
        flag : string
            either 'stress' or 'deflection', specifying which
            shape factor
            to calculate.

        """
        self.inner_height = data["spar"]["beam_type"]["box"].get(
            "inner_height", 'none')
        self.inner_width = data["spar"]["beam_type"]["box"].get(
            "inner_width", 'none')
        self.outer_width = data["spar"]["beam_type"]["box"].get(
            "outer_width", 'none')

        ratio =
        ((np.float128(1.0)-self.inner_width*self.inner_height**np.
        float128(3)
                  /
                  (self.outer_width*self.height.values[0]**np.floa
                  t128(3)))) \
            /
            (np.float128(6.0)*(np.float128(1)-self.inner_width*sel
            f.inner_height
               / (self.outer_width*self.height.values[0])))

        # self.stress_shape_factor.values = min(
            # np.divide(self.height.values, wing.max_thickness))
            * ratio
        # if wing.chord.definition == 'elliptic':
            # self.stress_shape_factor.values[-1] = 0.0

        self.stress_shape_factor = min(
            np.divide(self.height.values, wing.max_thickness)) *
            ratio
        if wing.chord.definition == 'elliptic':
            self.stress_shape_factor[-1] = 0.0
```

```
def I_beam(self, wing, data, flag):
    """Sets the shape factor for a I beam

    Parameters
    ----------
    wing : class
        instance of the Domain.Wing class with initialized
        values.
    data : dict
        dictionary containing all of the input file data
    flag : string
        either 'stress' or 'deflection', specifying which
        shape factor
        to calculate.

    """
    self.flange_height = data["spar"]["beam_type"]["I"].get(
        "flange_height", 'none')
    self.web_width = data["spar"]["beam_type"]["I"].get(
        "web_width", 'none')
    self.flange_width = data["spar"]["beam_type"]["I"].get(
        "flange_width", 'none')

    ratio =
    ((np.float128(2.0)*(self.flange_height/self.height.values[
    0])**np.float128(3)
              +
              np.float128(6.0)*(self.flange_height/self.height
              .values[0])
              *
              (np.float128(1)-self.flange_height/self.height.v
              alues[0])**np.float128(2)
              + (self.web_width/self.flange_width)
              *
              (np.float128(1)-np.float128(2.0)*self.flange_hei
              ght/self.height.values[0])**np.float128(3))) \
        /
        (np.float128(6.0)*(np.float128(2.0)*self.flange_height
        /self.height.values[0]
                + (self.web_width/self.flange_width)
                *
                (np.float128(1)-np.float128(2.0)*self.flange_h
                eight/self.height.values[0])))

    # self.stress_shape_factor.values = min(
        # np.divide(self.height.values, wing.max_thickness))
        * ratio
    # if wing.chord.definition == 'elliptic':
```

```
            # self.stress_shape_factor.values[-1] = 0.0

        self.stress_shape_factor = min(
            np.divide(self.height.values, wing.max_thickness)) *
            ratio
        if wing.chord.definition == 'elliptic':
            self.stress_shape_factor[-1] = 0.0


class Weight(object):
    """The Weight class contains all weight information.

    The weight class contains attributes pertaining to the weight

    Attributes
    ----------
    total : float
        The total weight, including the weight at the wing root,
        the
        payload distribution, and the wing-strucure weight
        distribution
    root : float
        The weight carried at the wing root.
    root_type : string
        String specifying whether the root weight is fixed or if
        it is a
        function of other parameters.
    root_total_ratio : float
        The ratio of root weight to total weight, for use in a
        function
        defining the root weight.
    net : float
        The weight of all non-structural components in the wing.
        Specifically, the root weight plus the integral of the
        payload
        distribution.
    net_type : string
        String specifying whether the net weight is fixed or a
        function
        of the wing-structure weight.
    wing_structure : float
        The total weight of the wing structure
    net_distributions : List
        List of arrays containing the payload distribution values.
    net_distribution_types : List
        List of strings defining whether the net_distributions
        are given
        by a function or interpolated from specific values from
        an input
```

```
    file.
net_distribution_names : List
    List of strings giving the name of each payload
    distribution.
structure_distribution : Array
    Array containing the wing-structure weight distribution
    values.

Methods
-------
initialize_constants:
    initializes the weight class constants.
initialize_distributions:
    initializes the weight class distributions.

"""

def read_constants(self, data):
    """Initializes the weight class constants.

    Parameters
    ----------
    data : dict
        dictionary containing all of the input file data

    """

    # Initialize Structural Weight
    if 'initial_structure' in data["weight"]:
        self.wing_structure =
        np.float128(data["weight"]["initial_structure"])

    # initialize Non-structural Weight
    if 'net_weight' in data["weight"]:
        self.net = np.float128(data["weight"]["net_weight"])
        self.net_type = 'constant'
    else:
        self.net_type = 'variable'

    # initialize root weight
    if ('root_total' in data["weight"]):
        self.root_total_ratio =
        np.float128(data["weight"]["root_total"])
        self.root_type = 'ratio'
    else:
        self.root = np.float128(data["weight"]["root_weight"])
        self.root_type = 'constant'

    # initialize total weight
```

```
        if 'total_weight' in data["weight"]:
            self.total =
            np.float128(data["weight"]["total_weight"])
        else:
            self.total = self.net + self.wing_structure

        if self.root_type == 'ratio':
            self.root = self.root_total_ratio*self.total

    def initialize_distributions(self, data):
        """Initializes the weight class distributions.

        Parameters
        ----------
        data : dict
            dictionary containing all of the input file data

        """
        self.structure_distribution = np.zeros(
            data["wing"]["grid"]+1, dtype=np.float128)

        self.net_distributions = []
        self.net_distributions.append(np.zeros(
            data["wing"]["grid"]+1, dtype=np.float128))
        for key in data["weight"]["net_distribution"]:
            self.net_distributions.append(Input_Distribution(
                data, data["weight"]["net_distribution"][key]))

    def base_net_weight_distribution(self, wing):
        """Gives the total net weight distribution on the base
        grid.

        Interpolates values from the given net weight
        distributions to
        give values on the base grid and adds net weight
        distributions
        together.

        Parameters
        ----------
        """
        self.net_distributions[0].fill(0.0)
        for i in range(1, len(self.net_distributions)):
            start_loc = np.argmax(np.subtract(
                wing.spanwise_location,
                self.net_distributions[i].locations[0]) >= -1e-16)

            end_loc = np.argmax(np.subtract(
                wing.spanwise_location,
```

```
                self.net_distributions[i].locations[-1]) >= 1e-16)

            self.net_distributions[0][0:start_loc-1] +=
            np.float128(0.0)

            self.net_distributions[0][start_loc:end_loc] \
                += interpolate.griddata(
                self.net_distributions[i].locations,
                self.net_distributions[i].values,
                wing.spanwise_location[
                    start_loc:end_loc])

            self.net_distributions[0][
                end_loc:] += np.float128(0.0)

    def calculate_structure(self, plane):
        """Calculates the required wing-structure weight

        Parameters
        ----------
        plane : class
            instance of the domain class, initialized

        """

        self.structure_distribution = np.divide(
            plane.moment,
            plane.spar.proportionality_coefficient)
        if plane.wing.chord.definition == 'elliptic':
            self.structure_distribution[-1] = 0.0

        ft = np.multiply(
            self.structure_distribution,
            np.sin(plane.wing.angle_location))

        self.wing_structure = -plane.wing.span*integrate.simps(
            ft,
            plane.wing.angle_location)

    def net_weight(self, wing):
        """calculates the total net weight

        Parameters
        ----------
        wing : class
            instance of the wing class, initialized

        """
```

```
            # if self.root_type == 'ratio':
                # self.net = 0.0
                # for k in range(1, len(self.net_distributions)):
                    # ft = np.multiply(
                        # self.net_distributions[k].values,
                        #
                        np.sin(self.net_distributions[k].angle_locatio
                        ns))

                    # self.net += -wing.span*integrate.simps(
                        # ft,
                        # self.net_distributions[k].angle_locations)
                # self.net += self.root_total_ratio*self.wing_structure
                # self.net = self.net/(1.-self.root_total_ratio)
            # else:
            if self.net_type == 'constant':
                self.net = self.net
            else:
                self.net = self.root
                for k in range(1, len(self.net_distributions)):
                    ft = np.multiply(
                        self.net_distributions[k].values,
                        np.sin(self.net_distributions[k].angle_locatio
                        ns))

                    self.net += -wing.span*integrate.simps(
                        ft,
                        self.net_distributions[k].angle_locations)
                        # print(self.net_distributions[k].name,
                        self.net)



class Input_Distribution(object):
    """The Input_Distribution class contains data for any input
    distributions.

    This class is meant to be used to define a distribution, including
    information about the distribution values and locations, as well
    as the
    distribution type and name, if applicable.

    Attributes
    ----------
    locations : array
        contains the spanwise locations at which the distribution
        values are
        known.
```

```
values : array
    contains the value of the distribution at each spanwise location.
definition : string
    string that reveals how the distribution is defined.
name : string
    string giving the name of the distribution, if applicable.

Methods
-------
__init__:
    initializes the distribution value, location, and type from a function.
file_distribution:
    initializes the distribution values and locations, if given in a file.

"""

def __init__(self, data, index):
    """Determines how a distribution should be initialized

    The __init__ function determines whether a distribution is initialized
    in a file or from a function and initializes the distribution
    accordingly, including its values, locations, type, and name, if
    applicable.

    Parameters
    ----------
    data : dict
        dictionary containing all of the input file data with distributions
        defined using the variable structure:

        "category" : { "distribution" : { "(function or file)" :{...}}}
    comment_flag: Boolean
        Flag to specify whether the init function should display
        information about the initialization of the distributions.

    """

    # Set the distribution definition
    self.definition = index.get("definition", 'none')
    if type(self.definition) == OrderedDict:
        self.definition = list(
            index.get("definition", self.definition).keys())[0]
```

```
        self.definition = index["definition"].get("function",
        self.definition)
        if type(self.definition) == OrderedDict:
            self.definition = list(
               index["definition"].get("function",
               self.definition).keys())[0]

        if self.definition == 'file':
            self.file_reference = index["definition"].get("file",
            'none')

        # Set the grid number
        self.grid = index.get("grid", data["wing"]["grid"])

        # Set the distribution name, if applicable
        self.name = index.get("name", 'none')

        # Set the distribution root_location, if applicable
        self.location = index.get("location",
        np.float128(data["wing"]["wing_span"])/np.float128(4.0))

        # Set the distribution width, if applicable
        self.width = index.get("width",
        np.float128(data["wing"]["wing_span"])/np.float128(2.0))

        self.values = np.zeros(self.grid+1, dtype=np.float128)
        if self.definition == "constant":

            self.values.fill(np.float128(index["definition"]["constant
            "]))
        self.locations = np.zeros(self.grid+1, dtype=np.float128)
        self.angle_locations = np.zeros(self.grid+1, dtype=np.float128)

    def set_values(self, plane):
        # Set values and locations to zero
        if self.definition == 'constant':
            self.locations = plane.wing.spanwise_location
            self.angle_locations = plane.wing.angle_location
        elif self.definition == 'file':
            self.locations = plane.wing.spanwise_location
            self.angle_locations = plane.wing.angle_location
            self.file_init(plane)
        else:
            self.angle_locations, self.locations, self.values =
            getattr(
                dist_functions_2, self.definition)(plane, self.name)

    def file_init(self, plane):
        #Read points from a file and interpolate based on grid.
```

```
with open(self.file_reference) as dist_file:
    dist = json.load(dist_file, object_pairs_hook=OrderedDict)

file_locations = np.zeros(len(dist.keys()), dtype=np.float128)
file_angle_locations = np.zeros(len(dist.keys()),
dtype=np.float128)
file_values = np.zeros(len(dist.keys()), dtype=np.float128)

i = 0
for key in dist:
    file_locations[i] = dist[key]["c1"]*plane.wing.span/2.0
    file_values[i] = dist[key]["c2"]
    file_angle_locations[i] =
    ma.acos(-2.0*file_locations[i]/plane.wing.span)
    i += 1

self.locations = plane.wing.spanwise_location
self.angle_locations = plane.wing.angle_location
self.values = interpolate.griddata(file_locations,
file_values, self.locations)
```

C.1.2 Built-in Distributions (python)

```
'''
dist_functions_2.py

This file contains the built-in functions for wing_structure_m3.py
includes functions that define the following
distributions:
   -Non-Structural Weight
   -Thickness-to-chord ratio
   -Chord Distribution

calls:
user_functions.py, which contains additional functions defined by the user
'''
import numpy as np
import math as ma
import user_functions as uf

def even(case, name):
    for i in range(1, len(case.weight.net_distributions)):
        if case.weight.net_distributions[i].name == name:
            grid = case.weight.net_distributions[i].grid
            location = case.weight.net_distributions[i].location
            width = case.weight.net_distributions[i].width
            k=i
    z_low = location*case.wing.span/2.0-width/2.0
    z_high = location*case.wing.span/2.0+width/2.0

    theta_low = ma.acos(-2.0*z_low/case.wing.span)
    theta_high = ma.acos(-2.0*z_high/case.wing.span)

    angle = np.subtract(ma.pi, np.linspace(
        theta_low, theta_high, grid+1))
    # angle = case.weight.net_distributions[k].angle_locations

    spanwise = case.wing.span/2.0*np.cos(
        case.weight.net_distributions[k].angle_locations)
    # spanwise = case.weight.net_distributions[k].locations

    for i in range (0,grid+1) :
        #Eq. (44)
        if (case.weight.net_type=='variable'):
            net_weight =
            case.weight.total-case.weight.wing_structure#8.8
        sumodd=0.0
```

```
        sumeven=0.0
        j=1
        while j < grid :
            sumodd = sumodd+ma.sin(angle[j])
            j=j+2

        j=2
        while j < grid :
            sumeven = sumeven+ma.sin(angle[j])
            j=j+2

        values =
        -(3.0*np.real(grid)*(net_weight-case.weight.root))/(case.wing.
        span*(angle[grid]-angle[0])*(ma.sin(angle[0])+4.0*sumodd+2.0*s
        umeven+ma.sin(angle[grid])))

    return angle, spanwise, values

def hunsaker(case, name):
    for i in range(1, len(case.weight.net_distributions)):
        if case.weight.net_distributions[i].name == name:
            grid = case.weight.net_distributions[i].grid
            location = case.weight.net_distributions[i].location
            width = case.weight.net_distributions[i].width
            k=i
    z_low = location*case.wing.span/2.0-width/2.0
    z_high = location*case.wing.span/2.0+width/2.0

    theta_low = ma.acos(0.0)
    theta_high = ma.acos(-1.0)

    angle = np.subtract(ma.pi, np.linspace(
        theta_low, theta_high, grid+1))
    # angle = case.weight.net_distributions[k].angle_locations

    spanwise = case.wing.span/2.0*np.cos(
        case.weight.net_distributions[k].angle_locations)
    # spanwise = case.weight.net_distributions[k].locations
    values = np.zeros(grid+1, dtype=np.float128)
    for i in range (0, grid+1):
        # print(case.weight.total-case.weight.wing_structure)
        if case.weight.net_type == 'constant':
            values[i] =
            (case.weight.net+case.weight.wing_structure-case.weight.ro
            ot)*case.wing.lift_distribution[i]/case.wing.span-case.wei
            ght.structure_distribution[i]
            # values[i] =
            (case.weight.total-case.weight.root)*case.wing.lift_distri
            bution[i]/case.wing.span-case.weight.structure_distributio
```

```
            n[i]
        else:
            values[i] =
            (case.weight.total-case.weight.root)*case.wing.lift_distri
            bution[i]/case.wing.span-case.weight.structure_distributio
            n[i]

    return angle, spanwise, values


def root_tip(case, name) :
    wing = case.wing
    for i in range (0,wing.grid+1) :
        values[i] =
        wing.root_thickness_chord+2.0*(wing.tip_thickness_chord-wing.r
        oot_thickness_chord)/wing.span*wing.spanwise_location[i]

    angle = case.wing.angle_location
    spanwise = case.wing.spanwise_location

    return angle, spanwise, values

def taper(case, name) :
    tr=case.wing.taper_ratio

    Cr=(np.float128(2.0)*case.wing.area)/(case.wing.span*(np.float128(
    1.0)+tr))
    values = np.subtract(Cr,
    np.multiply((Cr-tr*Cr)/(case.wing.span/np.float128(2.0)),
    case.wing.spanwise_location))

    angle = case.wing.angle_location
    spanwise = case.wing.spanwise_location

    return angle, spanwise, values

def elliptic(case, name):
    Ra=case.wing.span**2/case.wing.area
    values = np.zeros(case.wing.chord.grid+1, dtype=np.float128)
    for i in range(0,case.wing.chord.grid+1) :
        values[i] =
        4.0*case.wing.span/(ma.pi*Ra)*ma.sqrt(1-(2.0*(case.wing.spanwi
        se_location[i])/case.wing.span)**2)

    angle = case.wing.angle_location
    spanwise = case.wing.spanwise_location

    return angle, spanwise, values
```

```
def rectangular(case, name):
    values = np.zeros(case.wing.chord.grid+1, dtype = np.float64)
    values.fill(case.wing.area/case.wing.span)

    angle = case.wing.angle_location
    spanwise = case.wing.spanwise_location

    return angle, spanwise, values

def custom(case, name):
    angle, spanwise, values = getattr(uf, name)(case, name)

    return angle, spanwise, values

def fill(case, name):
    values = np.zeros(case.wing.chord.grid+1, dtype = np.float64)
    for i in range (0,case.wing.grid+1):
        # print(case.wing.angle_location[i], case.wing.grid)
        values[i]=case.wing.max_thickness[i]*case.spar.fill_ratio

    angle = case.wing.angle_location
    spanwise = case.wing.spanwise_location

    return angle, spanwise, values

def min_fill(case, name):
    minval = np.min(case.wing.max_thickness)
    values = np.zeros(case.wing.chord.grid+1, dtype = np.float64)
    for i in range (0,case.wing.grid+1):
        values[i]=minval

    angle = case.wing.angle_location
    spanwise = case.wing.spanwise_location

    return angle, spanwise, values
```

C.1.3 User-Defined Distributions for Ikhana (python)

```
'''
user_functions.py

This version of the user_functions.py script contains the fuel and pod
weight distributions for Ikhana. It is used in conjunction with
wing_structure_m3.py
'''

import numpy as np
import math as ma
from scipy import integrate
from scipy import interpolate


def fuel_2000(case, name):
    # Ikhana Fuel distribution for pod configuration (2000 lbf Fuel)
    for i in range(1, len(case.weight.net_distributions)):
        if case.weight.net_distributions[i].name == name:
            grid = case.weight.net_distributions[i].grid
            location = case.weight.net_distributions[i].location
            width =
            np.float128(0.830515)*case.wing.span/np.float128(2.0)
            # print("width:   ", width)
            # case.weight.net_distributions[i].width
            # print(width/case.wing.span*2.0)
            k=i
    z_low =
    location*case.wing.span/np.float128(2.0)-width/np.float128(2.0)
    z_high =
    location*case.wing.span/np.float128(2.0)+width/np.float128(2.0)

    theta_low = ma.acos(np.float128(-2.0)*z_low/case.wing.span)
    theta_high = ma.acos(np.float128(-2.0)*z_high/case.wing.span)

    angle = np.subtract(ma.pi, np.linspace(
        theta_low, theta_high, grid+1))
    # angle = case.weight.net_distributions[k].angle_locations

    spanwise = case.wing.span/np.float128(2.0)*np.cos(
        case.weight.net_distributions[k].angle_locations)
    # spanwise = case.weight.net_distributions[k].locations


    K=(np.float128(case.weight.net-case.weight.root-1000.0))/(np.float
    128(2.0)*np.float128(0.247952)*case.wing.chord.values[0]**np.float
```

```
    128(2)*case.wing.span)
    # print(case.weight.net-case.weight.root-1000.0)
    # print(K)
    chord = interpolate.griddata(case.wing.spanwise_location,
    case.wing.chord.values, spanwise)
    chord[0] = case.wing.chord.values[0]
    values = np.zeros(grid+1, dtype=np.float128)
    for i in range(0, grid+1):
        if spanwise[i] <=
        np.float128(0.830515)*case.wing.span/np.float128(2.0)+1e-12:
            values[i] = K*chord[i]**np.float128(2)

    return angle, spanwise, values


def fuel_3000(case, name):
    # Ikhana Fuel distribution for pod configuration (3000 lbf Fuel)
    for i in range(1, len(case.weight.net_distributions)):
        if case.weight.net_distributions[i].name == name:
            grid = case.weight.net_distributions[i].grid
            location = case.weight.net_distributions[i].location
            width =
            np.float128(0.830515)*case.wing.span/np.float128(2.0)#
            case.weight.net_distributions[i].width
            # print(width/case.wing.span*2.0)
            k=i
    z_low =
    location*case.wing.span/np.float128(2.0)-width/np.float128(2.0)
    z_high =
    location*case.wing.span/np.float128(2.0)+width/np.float128(2.0)

    theta_low = ma.acos(np.float128(-2.0)*z_low/case.wing.span)
    theta_high = ma.acos(np.float128(-2.0)*z_high/case.wing.span)

    angle = np.subtract(ma.pi, np.linspace(
        theta_low, theta_high, grid+1))
    # angle = case.weight.net_distributions[k].angle_locations

    spanwise = case.wing.span/np.float128(2.0)*np.cos(
        case.weight.net_distributions[k].angle_locations)
    # spanwise = case.weight.net_distributions[k].locations


    K=(np.float128(case.weight.net-case.weight.root))/(np.float128(2.0
    )*np.float128(0.247952)*case.wing.chord.values[0]**np.float128(2)*
    case.wing.span)
    # print(K)
    chord = interpolate.griddata(case.wing.spanwise_location,
    case.wing.chord.values, spanwise)
```

```
    chord[0] = case.wing.chord.values[0]
    values = np.zeros(grid+1, dtype=np.float128)
    for i in range(0, grid+1):
        if spanwise[i] <=
        np.float128(0.830515)*case.wing.span/np.float128(2.)+1e-12:
            values[i] = K*chord[i]**np.float128(2)

    return angle, spanwise, values


def pod(case, name):
    # Weight distribution for the Ikhana instrumentation pod (500 lbf)
    for i in range(1, len(case.weight.net_distributions)):
        if case.weight.net_distributions[i].name == name:
            grid = case.weight.net_distributions[i].grid
            location = case.weight.net_distributions[i].location
            width = case.weight.net_distributions[i].width
            k=i
    z_low =
    location*case.wing.span/np.float128(2.0)-width/np.float128(2.0)
    z_high =
    location*case.wing.span/np.float128(2.0)+width/np.float128(2.0)

    theta_low = ma.acos(np.float128(-2.0)*z_low/case.wing.span)
    theta_high = ma.acos(np.float128(-2.0)*z_high/case.wing.span)

    angle = np.subtract(ma.pi, np.linspace(
        theta_low, theta_high, grid+1))
    # angle = case.weight.net_distributions[k].angle_locations

    spanwise = case.wing.span/np.float128(2.0)*np.cos(
        case.weight.net_distributions[k].angle_locations)
    # spanwise = case.weight.net_distributions[k].locations

    values = np.zeros(grid+1, dtype=np.float128)
    for i in range(0, grid+1):
        values[i] = np.float128(500.0)

    return angle, spanwise, values
```

C.1.3 User-Defined Distributions for the CRM (python)

```
'''
user_functions.py

This version of the user_functions.py script contains the fuel and engine
weight distributions for the CRM. It is used in conjunction with
wing_structure_m3.py
'''
import numpy as np
import math as ma
from scipy import integrate
from scipy import interpolate


def center_tank(case, name):
    # Fuel weight distribution for the CRM center tank
    for i in range(1, len(case.weight.net_distributions)):
        if case.weight.net_distributions[i].name == name:
            grid = case.weight.net_distributions[i].grid
            location = case.weight.net_distributions[i].location
            width =
            case.weight.net_distributions[i].width*case.wing.span/np.f
            loat128(2.0)
            k=i

    z_low =
    location*case.wing.span/np.float128(2.0)-width/np.float128(2.0)
    z_high =
    location*case.wing.span/np.float128(2.0)+width/np.float128(2.0)

    theta_low = ma.acos(np.float128(-2.0)*z_low/case.wing.span)
    theta_high = ma.acos(np.float128(-2.0)*z_high/case.wing.span)

    angle = np.subtract(ma.pi, np.linspace(
        theta_low, theta_high, grid+1))

    spanwise = case.wing.span/np.float128(2.0)*np.cos(angle)

    chord = interpolate.griddata(case.wing.spanwise_location,
    case.wing.chord.values, spanwise)
    height = interpolate.griddata(case.wing.spanwise_location,
    case.spar.height.values, spanwise)
    t_c = interpolate.griddata(case.wing.spanwise_location,
    case.wing.thickness_chord.values, spanwise)
```

```
    fuel_weight =
    (case.weight.net-case.weight.root-7893.0*2.0*9.81-58121.0*9.81)/2.
    0

    if fuel_weight < 0.0:
        fuel_weight = 0.0

    chord[0] = case.wing.chord.values[0]
    values = np.zeros(grid+1, dtype=np.float128)
    for i in range(0, grid+1):
        if spanwise[i] < 0.1*case.wing.span/2.0:
            w_c =
            0.84977679*(spanwise[i]/case.wing.span*2.0)**2+0.54561078*
            (spanwise[i]/case.wing.span*2.0)+0.43691627
        elif spanwise[i] < 0.37*case.wing.span/2.0:
            w_c =
            0.64783085*(spanwise[i]/case.wing.span*2.0)**2+0.06455231*
            (spanwise[i]/case.wing.span*2.0)+0.48770883
        else:
            w_c =
            -1.21997827*(spanwise[i]/case.wing.span*2.0)**3+1.72590810
            *(spanwise[i]/case.wing.span*2.0)**2-1.08314960*(spanwise[
            i]/case.wing.span*2.0)+0.83163447

        values[i] = 119.826427*height[i]*t_c[i]*w_c*chord[i]**2*9.81
        # print(spanwise[i]/case.wing.span*2.0, w_c)
    values = np.multiply(values, fuel_weight/integrate.simps(values,
    spanwise))

    return angle, spanwise, values


def wing_tanks(case, name):
    # Fuel weight distribution for the CRM wing tanks
    for i in range(1, len(case.weight.net_distributions)):
        if case.weight.net_distributions[i].name == name:
            grid = case.weight.net_distributions[i].grid
            location = case.weight.net_distributions[i].location
            width =
            case.weight.net_distributions[i].width*case.wing.span/np.f
            loat128(2.0)
            k=i

    z_low =
    location*case.wing.span/np.float128(2.0)-width/np.float128(2.0)
    z_high =
    location*case.wing.span/np.float128(2.0)+width/np.float128(2.0)

    theta_low = ma.acos(np.float128(-2.0)*z_low/case.wing.span)
```

```
    theta_high = ma.acos(np.float128(-2.0)*z_high/case.wing.span)

    angle = np.subtract(ma.pi, np.linspace(
        theta_low, theta_high, grid+1))

    spanwise = case.wing.span/np.float128(2.0)*np.cos(angle)

    chord = interpolate.griddata(case.wing.spanwise_location,
    case.wing.chord.values, spanwise)
    height = interpolate.griddata(case.wing.spanwise_location,
    case.spar.height.values, spanwise)
    t_c = interpolate.griddata(case.wing.spanwise_location,
    case.wing.thickness_chord.values, spanwise)

    if case.weight.net-case.weight.root-7893.0*2.0*9.81 <
    58121.0146*9.81:
        print('HERE!')
        fuel_weight =
        (case.weight.net-case.weight.root-7893.0*2.0*9.81)/2.0
    else:
        fuel_weight = (58121.01463*9.81)/2.0
    # print(fuel_weight)
    values = np.zeros(grid+1, dtype=np.float128)
    # print('-----------------------')
    for i in range(0, grid+1):
        if spanwise[i] < 0.1*case.wing.span/2.0:
            w_c =
            0.84977679*(spanwise[i]/case.wing.span*2.0)**2+0.54561078*
            (spanwise[i]/case.wing.span*2.0)+0.43691627
        elif spanwise[i] < 0.37*case.wing.span/2.0:
            w_c =
            0.64783085*(spanwise[i]/case.wing.span*2.0)**2+0.06455231*
            (spanwise[i]/case.wing.span*2.0)+0.48770883
        else:
            w_c =
            -1.21997827*(spanwise[i]/case.wing.span*2.0)**3+1.72590810
            *(spanwise[i]/case.wing.span*2.0)**2-1.08314960*(spanwise[
            i]/case.wing.span*2.0)+0.83163447
        # print(spanwise[i]/case.wing.span*2.0, w_c)
        values[i] = 119.826427*height[i]*t_c[i]*w_c*chord[i]**2*9.81

    values = np.multiply(values, fuel_weight/integrate.simps(values,
    spanwise))


    return angle, spanwise, values


def engines(case, name):
```

```
# Weight distribution for the CRM Engines
for i in range(1, len(case.weight.net_distributions)):
    if case.weight.net_distributions[i].name == name:
        grid = case.weight.net_distributions[i].grid
        location = case.weight.net_distributions[i].location
        width = case.weight.net_distributions[i].width
        k=i
z_low = location*case.wing.span/np.float128(2.0)-width/np.float128(2.0)
z_high = location*case.wing.span/np.float128(2.0)+width/np.float128(2.0)

theta_low = ma.acos(np.float128(-2.0)*z_low/case.wing.span)
theta_high = ma.acos(np.float128(-2.0)*z_high/case.wing.span)

angle = np.subtract(ma.pi, np.linspace(
    theta_low, theta_high, grid+1))
# angle = case.weight.net_distributions[k].angle_locations

spanwise = case.wing.span/np.float128(2.0)*np.cos(angle)
# spanwise = case.weight.net_distributions[k].locations

values = np.zeros(grid+1, dtype=np.float128)
for i in range(0, grid+1):
    values[i] = np.float128(15786.0)*9.81

return angle, spanwise, values
```

C.1.5 Ikhana Input File (python)

```
{
    "wing": {
        "wing_area": 267.3,
        "wing_span": 66.0,
        "chord": {
            "definition": {
                "function": {
                    "taper": 0.42105
                }
            },
            "name": "tapered_chord"
        },
        "thickness_chord": {
            "definition": {
                "constant": "0.1875"
            },
            "name": "constant_t_c"
        },
        "loading": 31.8308760988,
        "grid": 160
    },
    "spar": {
        "C_sigma": 0.165,
        "C_delta": 0.653,
        "max_stress": 3600000.0,
        "max_deflection": 3.5,
        "modulus_elasticity": 1440000000.0,
        "height": {
            "definition": {
                "function": {
                    "fill": 1.0
                }
            }
        },
        "specific_weight": 172.8,
        "beam_type": "rectangular"
    },
    "limits": {
        "maneuvering": 3.75,
        "hard_landing": 3.75
    },
    "weight": {
        "root_weight": 4500.0,
        "net_weight": 7500.0,
        "net_distribution": {
```

```
            "dist1": {
                "definition": {
                    "function": "custom"
                },
                "location": 0.4152575,
                "width": 26.499,
                "name": "fuel_3000",
                "grid": 1280
            }
        },
        "initial_structure": 0.0
    },
    "flight": {
        "density": 0.0023769,
        "velocity": 287.0
    },
    "lift_distribution": {
        "B": [
            0.0,
            0.0,
            0.0,
            0.0,
            0.0,
            0.0,
            0.0,
            0.0,
            0.0
        ]
    }
}
```

C.1.6 CRM Input File (python)

```
{
    "wing": {
        "wing_area": 412.70,
        "wing_span": 58.76,
        "chord": {
            "definition": {
                "file" : "../chord_baseline.json"
            },
            "name": "crm_chord"
        },
        "thickness_chord": {
            "definition": {
                "file": "../thickness_chord.json"
            },
            "name": "crm_t_c"
        },
        "grid": 160
    },
    "spar": {
        "C_sigma": {
        "definition" : {
        "file" : "../C_sigma.json"
         },
            "name": "crm_c_sigma"
        },
        "C_delta": {
        "definition" : {
        "file" : "../C_delta.json"
         },
            "name": "crm_c_delta"
        },
        "max_stress": 4.20e8,
        "max_deflection": 2.56295388,
        "modulus_elasticity": 7.31e10,
        "height": {
            "definition": {
                "file": "../h_t.json"
            }
        },
        "specific_weight": 27271.8
    },
    "limits": {
        "maneuvering": 2.5,
        "hard_landing": 1.0
    },
```

```
"weight": {
"root_weight" : 1037960.346, "_comment" : "in Newtons",
"net_weight" : 2541303.606, "_comment" : "in Newtons",
"80_net_weight" : 2271607.086,
"50_net_weight" : 1867062.05,
"20_net_weight" : 1462517.526,
    "net_distribution": {
        "dist1": {
            "definition": {
                "function": "custom"
            },
            "location": 0.48631355,
            "width": 0.5473729,
            "name": "wing_tanks",
            "grid": 128
        },
        "dist2": {
            "definition": {
                "function": "custom"
            },
            "location": 0.10631355,
            "width": 0.2126271,
            "name": "center_tank",
            "grid": 128
        },
        "dist3": {
            "definition": {
                "function": "custom"
            },
            "location": 0.3271489362,
            "width": 0.5,
            "name": "engines",
            "grid": 80
        }
    },
    "initial_structure": 340000.0
},
"flight": {
    "density": 0.392967845,
    "velocity": 250.3486805
},
"lift_distribution": {
    "B": [
        0.0,
    0.0,
        0.0,
    0.0,
        0.0,
    0.0,
```

```
            0.0,
        0.0,
            0.0,
        0.0,
            0.0,
        0.0
        ]
    }
}
```

C.1.7 CRM Chord Input Distribution (python)


```
{
"r1": {
    "c1": 0.0,
    "c2": 13.551661498457399
     },
"r2": {
    "c1": 0.1,
    "c2": 11.945843636459227
     },
"r3": {
    "c1": 0.15,
    "c2": 11.068259303509292
     },
"r4": {
    "c1": 0.2,
    "c2": 10.194653744575007
     },
"r5": {
    "c1": 0.25,
    "c2": 9.326566467360001
     },
"r6": {
    "c1": 0.3,
    "c2": 8.460264114971995
     },
"r7": {
    "c1": 0.35,
    "c2": 7.597275837754842
     },
"r8": {
    "c1": 0.37,
    "c2": 7.246668020930948
     },
"r9": {
    "c1": 0.4,
    "c2": 7.032628217313561
     },
"r10": {
    "c1": 0.45,
    "c2": 6.671925934980483
     },
"r11": {
    "c1": 0.5,
    "c2": 6.312443793284292
     },
```

```
"r12": {
    "c1": 0.55,
    "c2": 5.953615902712695
     },
"r13": {
    "c1": 0.6,
    "c2": 5.594338590947295
     },
"r14": {
    "c1": 0.65,
    "c2": 5.23537166265512
     },
"r15": {
    "c1": 0.7,
    "c2": 4.876434228276691
     },
"r16": {
    "c1": 0.75,
    "c2": 4.516924077882012
     },
"r17": {
    "c1": 0.8,
    "c2": 4.157293273904669
     },
"r18": {
    "c1": 0.85,
    "c2": 3.7976420583373067
     },
"r19": {
    "c1": 0.9,
    "c2": 3.4381442572718566
     },
"r20": {
    "c1": 0.95,
    "c2": 3.0785536491388363
     },
"r21": {
    "c1": 1.0,
    "c2": 2.7148385291015975
     }
 }
```

C.1.8 CRM Thickness-to-Chord Ratio Input Distribution (python)


```
{
"r1": {
    "c1": 0.0,
    "c2": 0.1542
     },
"r2": {
    "c1": 0.02636972,
    "c2": 0.14992810536
     },
"r3": {
    "c1": 0.05273944,
    "c2": 0.14565621072
     },
"r4": {
    "c1": 0.07910916,
    "c2": 0.14138431608000002
     },
"r5": {
    "c1": 0.10547888,
    "c2": 0.13690422400000002
     },
"r6": {
    "c1": 0.108402167,
    "c2": 0.13631956660000003
     },
"r7": {
    "c1": 0.130937309,
    "c2": 0.1318125382
     },
"r8": {
    "c1": 0.153472451,
    "c2": 0.127430518036
     },
"r9": {
    "c1": 0.176007594,
    "c2": 0.123734754584
     },
"r10": {
    "c1": 0.198542736,
    "c2": 0.120038991296
     },
"r11": {
    "c1": 0.221077879,
    "c2": 0.117228498762
     },
```

```
"r12": {
    "c1": 0.243613021,
    "c2": 0.114479211438
     },
"r13": {
    "c1": 0.266148163,
    "c2": 0.11224666533
     },
"r14": {
    "c1": 0.288683306,
    "c2": 0.11021850246
     },
"r15": {
    "c1": 0.311218448,
    "c2": 0.108482019328
     },
"r16": {
    "c1": 0.333753591,
    "c2": 0.107039770176
     },
"r17": {
    "c1": 0.356288733,
    "c2": 0.10574845068
     },
"r18": {
    "c1": 0.378553584,
    "c2": 0.10480083274666667
     },
"r19": {
    "c1": 0.400400271,
    "c2": 0.103784789702
     },
"r20": {
    "c1": 0.422246959,
    "c2": 0.102954615558
     },
"r21": {
    "c1": 0.444093647,
    "c2": 0.10212444141400001
     },
"r22": {
    "c1": 0.465940334,
    "c2": 0.10129426730800001
     },
"r23": {
    "c1": 0.487787022,
    "c2": 0.100464093164
     },
"r24": {
```

```
    "c1": 0.50963371,
    "c2": 0.09976879096
     },
"r25": {
    "c1": 0.531480397,
    "c2": 0.099244470472
     },
"r26": {
    "c1": 0.553327085,
    "c2": 0.0987334583
     },
"r27": {
    "c1": 0.575173773,
    "c2": 0.09829652454
     },
"r28": {
    "c1": 0.59702046,
    "c2": 0.0978595908
     },
"r29": {
    "c1": 0.618867148,
    "c2": 0.097498125632
     },
"r30": {
    "c1": 0.640713836,
    "c2": 0.097148578624
     },
"r31": {
    "c1": 0.662560523,
    "c2": 0.096799031632
     },
"r32": {
    "c1": 0.684407211,
    "c2": 0.09644948462399999
     },
"r33": {
    "c1": 0.706253898,
    "c2": 0.09614996881599999
     },
"r34": {
    "c1": 0.728100586,
    "c2": 0.095975195312
     },
"r35": {
    "c1": 0.749947274,
    "c2": 0.095800421808
     },
"r36": {
    "c1": 0.771793961,
```

```
    "c2": 0.095669236234
     },
"r37": {
    "c1": 0.793640649,
    "c2": 0.095538156106
     },
"r38": {
    "c1": 0.815487337,
    "c2": 0.095438050652
     },
"r39": {
    "c1": 0.837334024,
    "c2": 0.095350663904
     },
"r40": {
    "c1": 0.859180712,
    "c2": 0.095281638576
     },
"r41": {
    "c1": 0.8810274,
    "c2": 0.09523794520000001
     },
"r42": {
    "c1": 0.902874087,
    "c2": 0.09519425182600001
     },
"r43": {
    "c1": 0.924720775,
    "c2": 0.09515055845
     },
"r44": {
    "c1": 0.946567463,
    "c2": 0.095106865074
     },
"r45": {
    "c1": 0.96841415,
    "c2": 0.0950631717
     },
"r46": {
    "c1": 1.0,
    "c2": 0.095
     }
 }
```

C.1.9 CRM Structure-Height-to-Thickness Ratio Distribution (python)


```
{
"r1": {
    "c1": 0.0,
    "c2": 0.79397723
     },
"r2": {
    "c1": 0.02636972,
    "c2": 0.84471244
     },
"r3": {
    "c1": 0.05273944,
    "c2": 0.8509422
     },
"r4": {
    "c1": 0.07910916,
    "c2": 0.84719307
     },
"r5": {
    "c1": 0.10547888,
    "c2": 0.83153541
     },
"r6": {
    "c1": 0.108402167,
    "c2": 0.68893291
     },
"r7": {
    "c1": 0.130937309,
    "c2": 0.71319139
     },
"r8": {
    "c1": 0.153472451,
    "c2": 0.70619773
     },
"r9": {
    "c1": 0.176007594,
    "c2": 0.70155845
     },
"r10": {
    "c1": 0.198542736,
    "c2": 0.70272557
     },
"r11": {
    "c1": 0.221077879,
    "c2": 0.70234312
     },
```

```
"r12": {
    "c1": 0.243613021,
    "c2": 0.70465787
     },
"r13": {
    "c1": 0.266148163,
    "c2": 0.70666337
     },
"r14": {
    "c1": 0.288683306,
    "c2": 0.70989636
     },
"r15": {
    "c1": 0.311218448,
    "c2": 0.71344447
     },
"r16": {
    "c1": 0.333753591,
    "c2": 0.71646327
     },
"r17": {
    "c1": 0.356288733,
    "c2": 0.72285886
     },
"r18": {
    "c1": 0.378553584,
    "c2": 0.71713279
     },
"r19": {
    "c1": 0.400400271,
    "c2": 0.70507497
     },
"r20": {
    "c1": 0.422246959,
    "c2": 0.70914089
     },
"r21": {
    "c1": 0.444093647,
    "c2": 0.71416734
     },
"r22": {
    "c1": 0.465940334,
    "c2": 0.71988372
     },
"r23": {
    "c1": 0.487787022,
    "c2": 0.72719199
     },
"r24": {
```

```
    "c1": 0.50963371,
    "c2": 0.73439937
     },
"r25": {
    "c1": 0.531480397,
    "c2": 0.74113715
     },
"r26": {
    "c1": 0.553327085,
    "c2": 0.74866464
     },
"r27": {
    "c1": 0.575173773,
    "c2": 0.75637769
     },
"r28": {
    "c1": 0.59702046,
    "c2": 0.76420115
     },
"r29": {
    "c1": 0.618867148,
    "c2": 0.77155554
     },
"r30": {
    "c1": 0.640713836,
    "c2": 0.77901018
     },
"r31": {
    "c1": 0.662560523,
    "c2": 0.78638813
     },
"r32": {
    "c1": 0.684407211,
    "c2": 0.79418333
     },
"r33": {
    "c1": 0.706253898,
    "c2": 0.80250043
     },
"r34": {
    "c1": 0.728100586,
    "c2": 0.81056473
     },
"r35": {
    "c1": 0.749947274,
    "c2": 0.81927051
     },
"r36": {
    "c1": 0.771793961,
```

```
    "c2": 0.82721439
     },
"r37": {
    "c1": 0.793640649,
    "c2": 0.83364822
     },
"r38": {
    "c1": 0.815487337,
    "c2": 0.83901001
     },
"r39": {
    "c1": 0.837334024,
    "c2": 0.84473792
     },
"r40": {
    "c1": 0.859180712,
    "c2": 0.85089014
     },
"r41": {
    "c1": 0.8810274,
    "c2": 0.85756037
     },
"r42": {
    "c1": 0.902874087,
    "c2": 0.86495514
     },
"r43": {
    "c1": 0.924720775,
    "c2": 0.87200941
     },
"r44": {
    "c1": 0.946567463,
    "c2": 0.87833752
     },
"r45": {
    "c1": 0.96841415,
    "c2": 0.8856619
     },
"r46": {
    "c1": 1.0,
    "c2": 0.91714217
     }
 }
```

C.1.10 CRM Stress-Limited Shape Factor Input Distribution (python)


```
{
"r1": {
    "c1": 0.0,
    "c2": 0.43850571
     },
"r2": {
    "c1": 0.02636972,
    "c2": 0.43850571
     },
"r3": {
    "c1": 0.05273944,
    "c2": 0.43877374
     },
"r4": {
    "c1": 0.07910916,
    "c2": 0.43883692
     },
"r5": {
    "c1": 0.10547888,
    "c2": 0.43846642
     },
"r6": {
    "c1": 0.108402167,
    "c2": 0.43630839
     },
"r7": {
    "c1": 0.130937309,
    "c2": 0.43526861
     },
"r8": {
    "c1": 0.153472451,
    "c2": 0.43324681
     },
"r9": {
    "c1": 0.176007594,
    "c2": 0.43112127
     },
"r10": {
    "c1": 0.198542736,
    "c2": 0.42885859
     },
"r11": {
    "c1": 0.221077879,
    "c2": 0.42654065
     },
```

```
"r12": {
    "c1": 0.243613021,
    "c2": 0.42404501
     },
"r13": {
    "c1": 0.266148163,
    "c2": 0.42116393
     },
"r14": {
    "c1": 0.288683306,
    "c2": 0.41839396
     },
"r15": {
    "c1": 0.311218448,
    "c2": 0.41552336
     },
"r16": {
    "c1": 0.333753591,
    "c2": 0.41230608
     },
"r17": {
    "c1": 0.356288733,
    "c2": 0.40910589
     },
"r18": {
    "c1": 0.378553584,
    "c2": 0.40556033
     },
"r19": {
    "c1": 0.400400271,
    "c2": 0.40239697
     },
"r20": {
    "c1": 0.422246959,
    "c2": 0.40061331
     },
"r21": {
    "c1": 0.444093647,
    "c2": 0.39817949
     },
"r22": {
    "c1": 0.465940334,
    "c2": 0.39637273
     },
"r23": {
    "c1": 0.487787022,
    "c2": 0.39447543
     },
"r24": {
```

```
    "c1": 0.50963371,
    "c2": 0.39265232
     },
"r25": {
    "c1": 0.531480397,
    "c2": 0.3908163
     },
"r26": {
    "c1": 0.553327085,
    "c2": 0.3889846
     },
"r27": {
    "c1": 0.575173773,
    "c2": 0.38714149
     },
"r28": {
    "c1": 0.59702046,
    "c2": 0.3841807
     },
"r29": {
    "c1": 0.618867148,
    "c2": 0.38210578
     },
"r30": {
    "c1": 0.640713836,
    "c2": 0.37995042
     },
"r31": {
    "c1": 0.662560523,
    "c2": 0.37769785
     },
"r32": {
    "c1": 0.684407211,
    "c2": 0.37536771
     },
"r33": {
    "c1": 0.706253898,
    "c2": 0.37302048
     },
"r34": {
    "c1": 0.728100586,
    "c2": 0.36903638
     },
"r35": {
    "c1": 0.749947274,
    "c2": 0.36645211
     },
"r36": {
    "c1": 0.771793961,
```

```
    "c2": 0.3636975
     },
"r37": {
    "c1": 0.793640649,
    "c2": 0.36066682
     },
"r38": {
    "c1": 0.815487337,
    "c2": 0.35746334
     },
"r39": {
    "c1": 0.837334024,
    "c2": 0.35352567
     },
"r40": {
    "c1": 0.859180712,
    "c2": 0.34972583
     },
"r41": {
    "c1": 0.8810274,
    "c2": 0.34252219
     },
"r42": {
    "c1": 0.902874087,
    "c2": 0.33796082
     },
"r43": {
    "c1": 0.924720775,
    "c2": 0.33304303
     },
"r44": {
    "c1": 0.946567463,
    "c2": 0.32781819
     },
"r45": {
    "c1": 0.96841415,
    "c2": 0.32225406
     },
"r46": {
    "c1": 1.0,
    "c2": 0.3123694
     }
 }
```

C.1.11 CRM Deflection-Limited Shape Factor Input Distribution (python)


```
{
"r1": {
    "c1": 0.0,
    "c2": 1.75402283
     },
"r2": {
    "c1": 0.02636972,
    "c2": 1.75402283
     },
"r3": {
    "c1": 0.05273944,
    "c2": 1.75509497
     },
"r4": {
    "c1": 0.07910916,
    "c2": 1.75534768
     },
"r5": {
    "c1": 0.10547888,
    "c2": 1.75386569
     },
"r6": {
    "c1": 0.108402167,
    "c2": 1.74523358
     },
"r7": {
    "c1": 0.130937309,
    "c2": 1.74107444
     },
"r8": {
    "c1": 0.153472451,
    "c2": 1.73298725
     },
"r9": {
    "c1": 0.176007594,
    "c2": 1.72448509
     },
"r10": {
    "c1": 0.198542736,
    "c2": 1.71543437
     },
"r11": {
    "c1": 0.221077879,
    "c2": 1.70616259
     },
```

```
"r12": {
    "c1": 0.243613021,
    "c2": 1.69618005
     },
"r13": {
    "c1": 0.266148163,
    "c2": 1.68465572
     },
"r14": {
    "c1": 0.288683306,
    "c2": 1.67357583
     },
"r15": {
    "c1": 0.311218448,
    "c2": 1.66209344
     },
"r16": {
    "c1": 0.333753591,
    "c2": 1.64922433
     },
"r17": {
    "c1": 0.356288733,
    "c2": 1.63642354
     },
"r18": {
    "c1": 0.378553584,
    "c2": 1.62224134
     },
"r19": {
    "c1": 0.400400271,
    "c2": 1.60958786
     },
"r20": {
    "c1": 0.422246959,
    "c2": 1.60245325
     },
"r21": {
    "c1": 0.444093647,
    "c2": 1.59271796
     },
"r22": {
    "c1": 0.465940334,
    "c2": 1.58549094
     },
"r23": {
    "c1": 0.487787022,
    "c2": 1.57790172
     },
"r24": {
```

```
    "c1": 0.50963371,
    "c2": 1.57060927
     },
"r25": {
    "c1": 0.531480397,
    "c2": 1.56326521
     },
"r26": {
    "c1": 0.553327085,
    "c2": 1.55593839
     },
"r27": {
    "c1": 0.575173773,
    "c2": 1.54856594
     },
"r28": {
    "c1": 0.59702046,
    "c2": 1.5367228
     },
"r29": {
    "c1": 0.618867148,
    "c2": 1.52842311
     },
"r30": {
    "c1": 0.640713836,
    "c2": 1.51980169
     },
"r31": {
    "c1": 0.662560523,
    "c2": 1.5107914
     },
"r32": {
    "c1": 0.684407211,
    "c2": 1.50147083
     },
"r33": {
    "c1": 0.706253898,
    "c2": 1.4920819
     },
"r34": {
    "c1": 0.728100586,
    "c2": 1.47614554
     },
"r35": {
    "c1": 0.749947274,
    "c2": 1.46580845
     },
"r36": {
    "c1": 0.771793961,
```

```
    "c2": 1.45478998
     },
"r37": {
    "c1": 0.793640649,
    "c2": 1.44266726
     },
"r38": {
    "c1": 0.815487337,
    "c2": 1.42985335
     },
"r39": {
    "c1": 0.837334024,
    "c2": 1.41410269
     },
"r40": {
    "c1": 0.859180712,
    "c2": 1.39890333
     },
"r41": {
    "c1": 0.8810274,
    "c2": 1.37008877
     },
"r42": {
    "c1": 0.902874087,
    "c2": 1.35184328
     },
"r43": {
    "c1": 0.924720775,
    "c2": 1.33217212
     },
"r44": {
    "c1": 0.946567463,
    "c2": 1.31127277
     },
"r45": {
    "c1": 0.96841415,
    "c2": 1.28901625
     },
"r46": {
    "c1": 1.0,
    "c2": 1.2494776
     }
 }
```

C.2 Trajectory Optimization

The following example code extracts can be used to compute the optimum cruise trajectory for a desired input aircraft, as described in Chapter 5.

C.2.1 Trajectory Optimization for Ikhana (python)

```
'''
ikhana_range_master.py

This file performs trajectory optimization for an aircraft having an
engine model defined using a second-order multidimensional polynomial
fit to obtain the power-specific fuel consumption and power
available with
respect to altitude and velocity

'''

import math as ma
import numpy as np
import scipy.integrate as integrate
import scipy.optimize as optimize
import scipy.interpolate as interpolate
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import standard_atmosphere as std
import json
from collections import OrderedDict
import time
from progress.bar import IncrementalBar
import multiprocessing
from itertools import repeat
from functools import partial
import sys
import sys
sys.path.append('./Ikhana_structure')
import  wing_structure_m4 as ws
sys.path.append('..')


class Aircraft(object):

    def __init__(self, filename):
        with open(filename) as input_file:
            data = json.load(input_file, object_pairs_hook=OrderedDict)
```

```
        self.input_airplane(data)
        self.grid_setup()
        self.array_initialize()

    def input_airplane(self, data):

        #Units
        if data["units"] == 'English':
            self.english_units = True
        else:
            self.english_units = False

        # CRM Constant Properties
        self.iweight = data["MTOW"]
        self.fweight = data["M15FW"]
        self.Sw = data["wing_area"]
        self.CD0 = data["CD0"]
        self.CD1 = data["CD1"]
        self.CD2 = data["CD2"]
        self.CM0 = data["CM0"]
        self.CM1 = data["CM1"]
        self.CM2 = data["CM2"]
        self.r = data["range"]*5280.0
        self.a00 = data["a00"]
        self.a01 = data["a01"]
        self.a02 = data["a02"]
        self.a10 = data["a10"]
        self.a11 = data["a11"]
        self.a12 = data["a12"]
        self.a20 = data["a20"]
        self.a21 = data["a21"]
        self.a22 = data["a22"]
        self.b00 = data["b00"]
        self.b01 = data["b01"]
        self.b02 = data["b02"]
        self.b10 = data["b10"]
        self.b11 = data["b11"]
        self.b12 = data["b12"]
        self.b20 = data["b20"]
        self.b21 = data["b21"]
        self.b22 = data["b22"]
        self.n = data["grid"]
        self.g = 32.174
        self.name = data["name"]
        self.run_type = 'none'
        self.run_direction = data["direction"]
        self.s_runtime = 0.0
        self.f_runtime = 0.0
        self.results_path = '.'
```

```
        self.RA = 16.5
        self.e = 1.

    def grid_setup(self):
        self.x = np.linspace(0, self.r, self.n)
        self.xm = self.x/5280.0

    def array_initialize(self):
        self.W = np.zeros(self.n)
        if self.run_direction == "f":
            self.W[0] = self.iweight
        elif self.run_direction == "b":
            self.W[-1] = self.fweight
        self.CL = np.zeros(self.n)
        self.CD = np.zeros(self.n)
        self.V = np.zeros(self.n)
        self.h = np.zeros(self.n)
        self.t = np.zeros(self.n)
        self.esfc = np.zeros(self.n)
        self.rho = np.zeros(self.n)
        self.a = np.zeros(self.n)
        self.Temp = np.zeros(self.n)
        self.T = np.zeros(self.n)

    def opt_h_V(self, y, i):
        h = y[0]*100.
        V = y[1]
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)
        # ~ print(Temp, TSL)

        #Lift coefficient
        CL = self.W[i]/(0.5*rho*V**2*self.Sw)

        #Mach number
        M = V/a

        #Drag coefficient
        CD =                                                              ↵
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se ↵
```

```
        lf.CM2)

        #Thrust
        T = 0.5*rho*V**2*self.Sw*CD

        #Power Required
        PR = T*V

        #Power Available
        if h <36131:
            m=0.7
        else:
            m=1.0
        PA =                                                              ⤶
        ((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+self. ⤶
        b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)*V/1.6878 ⤶
        1*V/1.68781)*550
        # ~ print(i, self.W[i])

        climb = (PA-PR)/self.W[i]*60
        # ~ print(h, V)

        #ESFC
        esfc =                                                            ⤶
        ((self.a00+self.a10*h+self.a20*h*h)+(self.a01+self.a11*h+self. ⤶
        a21*h*h)*V/1.68781+(self.a02+self.a12*h+self.a22*h*h)*V/1.6878 ⤶
        1*V/1.68781)/550/(32.174*3600)

        if i==self.n-1:
            dx = self.x[i]-self.x[i-1]
        else:
            dx = self.x[i+1]-self.x[i]

        t = dx/V

        if climb>100.:
            # ~ print(esfc*PR)
            return esfc*PR*self.g*t
        else:
            # ~ print('yo',esfc*PR*self.g*t+10*(climb-100)**2)
            return esfc*PR*self.g*t+10*(climb-100)**2

    def set_state(self, y, i):
        h = y[0]*100.
        V = y[1]
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
```

```
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)

        #Lift coefficient
        CL = self.W[i]/(0.5*rho*V**2*self.Sw)

        #Mach number
        M = V/a

        #Drag coefficient
        CD =                                                          ↵
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se ↵
        lf.CM2)

        #Thrust
        T = 0.5*rho*V**2*self.Sw*CD

        #ESFC
        esfc =                                                        ↵
        ((self.a00+self.a10*h+self.a20*h*h)+(self.a01+self.a11*h+self. ↵
        a21*h*h)*V/1.68781+(self.a02+self.a12*h+self.a22*h*h)*V/1.6878 ↵
        1*V/1.68781)/550/(32.174*3600)

        self.h[i] = h
        self.V[i] = V
        self.rho[i] = rho
        self.a[i] = a
        self.Temp[i] = Temp
        self.CL[i] = CL
        self.CD[i] = CD
        self.T[i] = T
        self.esfc[i] = esfc

    def breguet(self, y, flag='obj'):
        h = y[0]
        V = y[1]
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
```

```
Temp = statmos.T(h)
TSL = statmos.T(0)
rhoSL = statmos.rho(0)
# ~ print(Temp, TSL)

#Lift coefficient
if self.run_type == 'f':
    CL = self.W[0]/(0.5*rho*V**2*self.Sw)
else:
    CL = self.W[-1]/(0.5*rho*V**2*self.Sw)

#Mach number
M = V/a

#Drag coefficient
CD =
(self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
lf.CM2)


#TSFC
esfc =
((self.a00+self.a10*h+self.a20*h*h)+(self.a01+self.a11*h+self.
a21*h*h)*V/1.68781+(self.a02+self.a12*h+self.a22*h*h)*V/1.6878
1*V/1.68781)/550/(32.174*3600)
c = esfc*V

if self.run_direction == 'f':
    Wf = self.W[0]/np.exp(self.r*self.g*c/(V*CL/CD))

    self.hb = h
    self.Vb = V
    self.L_Db = CL/CD
    self.esfcb = esfc
    self.Wfb = Wf
    self.W[-1] = self.Wfb
    self.Mb = M
    self.tb = self.r/V

    #Power Req.
    PR = 0.5*rho*V**3*self.Sw*CD

    #Power Available
    if h <36131:
        m=0.7
    else:
        m=1.0
    PA =
    ((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+s
```

```
            self.b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)* ↵
            V/1.68781*V/1.68781)*550

            climb = (PA-PR)/self.W[0]*60

            if climb>100.:
                return self.W[0]-Wf
            else:
                return (self.W[0]-Wf)+10*(climb-100)**2

        elif self.run_direction == 'b':
            # ~ print(V, CL, CD)
            Wi = self.W[-1]*np.exp(self.r*self.g*c/(V*CL/CD))

            self.hb = h
            self.Vb = V
            self.L_Db = CL/CD
            self.esfcb = esfc
            self.iweight = Wi
            self.W[0] = self.iweight
            self.Mb = M
            self.tb = self.r/V

            # ~ print('fuel_burn', c*T*self.g*t)
            # ~ print('velocity: ', V)
            # ~ print('altitude: ', h)
            # ~ print('Mach: ', M)
            # ~ print('Drag Coefficient: ', CD)
            # ~ print('Thrust: ', T)
            # ~ print('TSFC: ', c)
            # ~ print('Weight: ', self.W[i])
            # ~ print('-----------------------------------')

            #Power Req.
            PR = 0.5*rho*V**3*self.Sw*CD

            #Power Available
            if h <36131:
                m=0.7
            else:
                m=1.0
            PA =                                                    ↵
            ((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+s ↵
            self.b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)* ↵
            V/1.68781*V/1.68781)*550

            climb = (PA-PR)/self.W[-1]*60

            if climb>100.:
```

```
            return Wi-self.W[-1]
        else:
            return (Wi-self.W[-1])+10*(climb-100)**2

def breguet_opt(self):
    y0 = [20000., 200.0]
    bnds = ((100., 80000.), (135.0, 1200.))
    cons = [{"type" : "ineq",
             "fun" : self.breguet}]
    ans = optimize.minimize(self.breguet,
                            y0,
                            method='SLSQP',
                            bounds = bnds,
                            # ~ constraints = [{"type" : "ineq",
                                        # ~ "fun" : self.breguet,
                                        # ~ "args" : ("cons",)}],
                            options={'disp' : False,
                                     'ftol' : 1e-16})

def service_ceiling(self, y, i):
    # ~ print(y, i)
    h = y[0]

    #atmospheric properties
    if self.english_units == True:
        statmos = std.StandardAtmosphere('English')
    else:
        statmos = std.StandardAtmosphere('SI')

    rho = statmos.rho(h)
    a = statmos.a(h)
    Temp = statmos.T(h)
    TSL = statmos.T(0)
    rhoSL = statmos.rho(0)

    if self.run_type == 'Vmf':
        V = optimize.newton(self.Vmfp, 800.0, args=(rho, a, i))
    elif self.run_type == 'Vmd':
        V = optimize.newton(self.Vmd, 800.0, args=(rho, a, i))
    else:
        V = y[1]

    #Lift coefficient
    CL = self.W[i]/(0.5*rho*V**2*self.Sw)

    #Mach number
    M = V/a

    #Drag coefficient
```

```
        CD =
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
        lf.CM2)

        #Power Req.
        PR = 0.5*rho*V**3*self.Sw*CD

        #Power Available
        if h <36131:
            m=0.7
        else:
            m=1.0
        PA =
        ((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+self.
        b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)*V/1.6878
        1*V/1.68781)*550

        climb = (PA-PR)/self.W[i]*60

        return climb-100.+400.

    def trajectory_opt(self):
        opts = {'disp' : False,
                'maxiter' : 500}#,
                # ~ 'eps' : 1e-10,
                # ~ 'ftol' : 1e-11}
        tolerance = 1e-14
        iguess = [200., 250.]

        if self.run_direction == 'b':

            if self.run_type == 'full_opt':
                #Full Optimization

                #-------------------------------------------------------
                ---------------------------------
                y0 = iguess
                bnds = ((1., 800.), (100.0, 600.))
                # ~ cons = [{"type" : "ineq",
                            # ~ "fun" : self.service_ceiling}]
                for i in range(1, self.n+1):
                    j = self.n-i
                    ans = optimize.minimize(self.opt_h_V,
                                            y0, args=(j),
                                            method='SLSQP',
                                            bounds = bnds,
                                            tol = tolerance,
                                            options=opts)
```

```
                    self.set_state(ans.x, j)


                    if j>0:
                        self.W[j-1] =                                    ⤶
                        self.W[j]+self.esfc[j]*self.T[j]*self.V[j]*sel ⤶
                        f.g*(self.x[j]-self.x[j-1])/self.V[j]
                    if ans.success == False:
                        print('optimization failed', ans.message)
                        y0 = ans.x
                    #print(y0)
                for i in range(0, self.n):
                    if i<self.n-1:
                        self.t[i+1] =                                    ⤶
                        self.t[i]+(self.x[i+1]-self.x[i])/self.V[i]
                    #print(y0)

        self.L = np.multiply(self.Sw*0.5*self.rho,                       ⤶
        np.multiply(np.multiply(self.V, self.V), self.CL))
        self.D = np.multiply(self.Sw*0.5*self.rho,                       ⤶
        np.multiply(np.multiply(self.V, self.V), self.CD))
        self.L_D = np.divide(self.L, self.D)

    def trajectory_cases(self):
        data_format="{0:<30}{1:<30.16f}\n"
                                                                         ⤶
        dist_header="{0:<30}{1:<30}{2:<30}{3:<30}{4:<32}{5:<30}{6:<30} ⤶
        {7:<32}{8:<30}{9:<30}{10:<32}{11:<30}{12:<30}\n"
                                                                         ⤶
        dist_format="{0:<30.16f}{1:<30.16f}{2:<30.16f}{3:<30.16f}{4:<3 ⤶
        2.16f}{5:<30.16f}{6:<30.16f}{7:<32.16f}{8:<30.16f}{9:<30.16f}{ ⤶
        10:<30.16f}{11:<30.16f}{12:<30.16f}\n"

        self.fig_path = self.results_path+'figs/'
        if self.run_type == 'breguet':
            self.breguet_opt()
            print('Run Type: ', self.run_type)
            if self.run_direction == 'f':
                print('Total Fuel Burn: ', self.W[0]-self.Wfb, 'lbf')
            else:
                print('Total Fuel Burn: ', self.iweight-self.W[-1],     ⤶
                'lbf')
            print('Total Time: ', self.tb/3600.0, 'hours')
            print('L/D: ', self.L_Db)
            print('V: ', self.Vb, 'ft/s')
            print('h: ', self.hb, 'ft')
            print('TSFC: ', self.esfcb, 'slugs/lbf/s')
            print('Mach: ', self.Mb)
            print('total run time: ', time.time()-self.s_runtime)
```

```
        print('-------------------------------------------------
        ')


    else:
        self.trajectory_opt()
        print('Run Type:', self.run_type)
        print('Total Fuel Burn: ', self.W[0]-self.W[-1], 'lbf')
        print('Total Time: ', self.t[-1]/3600.0, 'hours')
        print('total run time: ', time.time()-self.s_runtime)
        print('-------------------------------------------------
        ')

        '''
        data_file =
        open(self.results_path+str(self.RA)+'_'+str(self.e)+'_data
        .txt', 'w')
        data_file.write(data_format.format('Total Fuel Burn:',
        self.W[0]-self.W[-1]))
        data_file.write(data_format.format('Total Cruise Time:',
        self.t[-1]/3600.0))
        data_file.write(data_format.format('Run Time:',
        time.time()-self.s_runtime))
        data_file.close()

        dist_file =
        open(self.results_path+str(self.RA)+'_'+str(self.e)+'_dist
        s.txt', 'w')
        dist_file.write(dist_header.format('x[ft]',
        'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/
        D','T[lbf]','ESFC[slugs/lbf ft/s /s]','W[lbf]'))
        for i in range(0, len(self.x)):
            dist_file.write(dist_format.format(self.x[i],
            self.x[i]/5280., self.t[i], self.t[i]/3600.0,
            self.h[i], self.V[i], self.V[i]/self.a[i],
            self.CL[i], self.CD[i], self.CL[i]/self.CD[i],
            self.T[i], self.esfc[i], self.W[i]))
        dist_file.close()

        plot_range_multiplier = 100.

        #Altitude
        plt.figure(1)
        plt.plot(self.xm, self.h)
        plt.xlabel('x [mi]')
        plt.ylabel('h [ft]')
```

```
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_h
.png')

#Velocity
plt.figure(2)
plt.plot(self.xm, self.V)
plt.xlabel('x [mi]')
plt.ylabel('V [ft/s]')

plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_V
.png')

#Mach
plt.figure(3)
plt.plot(self.xm, np.divide(self.V,self.a))
plt.xlabel('x [mi]')
plt.ylabel('M')

plt.ylim(np.amin(np.divide(self.V,self.a))-(np.amax(np.div
ide(self.V,self.a))-np.amin(np.divide(self.V,self.a)))*plo
t_range_multiplier,
np.amax(np.divide(self.V,self.a))+(np.amax(np.divide(self.
V,self.a))-np.amin(np.divide(self.V,self.a)))*plot_range_m
ultiplier)

plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_M
.png')


#CL
plt.figure(4)
plt.plot(self.xm, self.CL)
plt.xlabel('x [mi]')
plt.ylabel('CL')

plt.ylim(np.amin(self.CL)-(np.amax(self.CL)-np.amin(self.C
L))*plot_range_multiplier,
np.amax(self.CL)+(np.amax(self.CL)-np.amin(self.CL))*plot_
range_multiplier)

plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_C
L.png')

#CD
plt.figure(5)
plt.plot(self.xm, self.CD)
plt.xlabel('x [mi]')
plt.ylabel('CD')

```

```
plt.ylim(np.amin(self.CD)-(np.amax(self.CD)-np.amin(self.CD))*plot_range_multiplier,
np.amax(self.CD)+(np.amax(self.CD)-np.amin(self.CD))*plot_range_multiplier)

plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_CD.png')


#L/D
plt.figure(6)
plt.plot(self.xm, self.L_D)
plt.xlabel('x [mi]')
plt.ylabel('L/D')

plt.ylim(np.amin(self.L_D)-(np.amax(self.L_D)-np.amin(self.L_D))*plot_range_multiplier,
np.amax(self.L_D)+(np.amax(self.L_D)-np.amin(self.L_D))*plot_range_multiplier)

plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_L_D.png')


#T
plt.figure(7)
plt.plot(self.xm, self.T)
plt.xlabel('x [mi]')
plt.ylabel('T [lbf]')

plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_T.png')

#TSFC
plt.figure(8)
plt.plot(self.xm, self.esfc)
plt.xlabel('x [mi]')
plt.ylabel('ESFC [slugs/lbf ft/s /s]')

plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_ESFC.png')

#W
plt.figure(9)
plt.plot(self.xm, self.W)
plt.xlabel('x [mi]')
plt.ylabel('W [lbf]')

plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_W
```

```
            .png')
            # ~ plt.show()

            plt.figure(1).clear()
            plt.figure(2).clear()
            plt.figure(3).clear()
            plt.figure(4).clear()
            plt.figure(5).clear()
            plt.figure(6).clear()
            plt.figure(7).clear()
            plt.figure(8).clear()
            plt.figure(9).clear()
            '''

class Aircraft_h(object):

    def __init__(self, filename):
        with open(filename) as input_file:
            data = json.load(input_file, object_pairs_hook=OrderedDict)

        self.input_airplane(data)
        self.grid_setup()
        self.array_initialize()

    def input_airplane(self, data):

        #Units
        if data["units"] == 'English':
            self.english_units = True
        else:
            self.english_units = False

        # CRM Constant Properties
        self.iweight = data["MTOW"]
        self.fweight = data["M15FW"]
        self.Sw = data["wing_area"]
        self.CD0 = data["CD0"]
        self.CD1 = data["CD1"]
        self.CD2 = data["CD2"]
        self.CM0 = data["CM0"]
        self.CM1 = data["CM1"]
        self.CM2 = data["CM2"]
        self.r = data["range"]*5280.0
        self.a00 = data["a00"]
        self.a01 = data["a01"]
        self.a02 = data["a02"]
        self.a10 = data["a10"]
        self.a11 = data["a11"]
        self.a12 = data["a12"]
```

```
        self.a20 = data["a20"]
        self.a21 = data["a21"]
        self.a22 = data["a22"]
        self.b00 = data["b00"]
        self.b01 = data["b01"]
        self.b02 = data["b02"]
        self.b10 = data["b10"]
        self.b11 = data["b11"]
        self.b12 = data["b12"]
        self.b20 = data["b20"]
        self.b21 = data["b21"]
        self.b22 = data["b22"]
        self.n = data["grid"]
        self.g = 32.174
        self.name = data["name"]
        self.run_type = 'none'
        self.run_direction = data["direction"]
        self.s_runtime = 0.0
        self.f_runtime = 0.0
        self.results_path = '.'
        self.RA = 16.5
        self.e = 1.

    def grid_setup(self):
        self.x = np.linspace(0, self.r, self.n)
        self.xm = self.x/5280.0

    def array_initialize(self):
        self.W = np.zeros(self.n)
        if self.run_direction == "f":
            self.W[0] = self.iweight
        elif self.run_direction == "b":
            self.W[-1] = self.fweight
        self.CL = np.zeros(self.n)
        self.CD = np.zeros(self.n)
        self.V = np.zeros(self.n)
        self.h = np.zeros(self.n)
        self.t = np.zeros(self.n)
        self.esfc = np.zeros(self.n)
        self.rho = np.zeros(self.n)
        self.a = np.zeros(self.n)
        self.Temp = np.zeros(self.n)
        self.T = np.zeros(self.n)
        self.climb = np.zeros(self.n)

    def opt_h_V(self, y, h, i):
        V = y
        #atmospheric properties
        if self.english_units == True:
```

```
    statmos = std.StandardAtmosphere('English')
else:
    statmos = std.StandardAtmosphere('SI')

rho = statmos.rho(h)
a = statmos.a(h)
Temp = statmos.T(h)
TSL = statmos.T(0)
rhoSL = statmos.rho(0)
# ~ print(Temp, TSL)

#Lift coefficient
CL = self.W[i]/(0.5*rho*V**2*self.Sw)

#Mach number
M = V/a

#Drag coefficient
CD =                                                                  ↵
(self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se ↵
lf.CM2)

#Thrust
T = 0.5*rho*V**2*self.Sw*CD

#Power Required
PR = T*V

#Power Available
if h <36131:
    m=0.7
else:
    m=1.0
PA =                                                                  ↵
((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+self. ↵
b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)*V/1.6878 ↵
1*V/1.68781)*550
# ~ print(i, self.W[i])

climb = (PA-PR)/self.W[i]*60
# ~ print(h, V)

#ESFC
esfc =                                                                ↵
((self.a00+self.a10*h+self.a20*h*h)+(self.a01+self.a11*h+self. ↵
a21*h*h)*V/1.68781+(self.a02+self.a12*h+self.a22*h*h)*V/1.6878 ↵
1*V/1.68781)/550/(32.174*3600)

if i==self.n-1:
```

```
            dx = self.x[i]-self.x[i-1]
        else:
            dx = self.x[i+1]-self.x[i]

        t = dx/V

        if climb>100.:
            # ~ print(esfc*PR*self.g*t)
            return esfc*PR*self.g*t
        else:
            # ~ print('yo',esfc*PR*self.g*t)#+10*(climb-100)**2)
            return esfc*PR*self.g*t+10*(climb-100)**2

    def set_state(self, y, h, i):
        V = y
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)

        #Lift coefficient
        CL = self.W[i]/(0.5*rho*V**2*self.Sw)

        #Mach number
        M = V/a

        #Drag coefficient
        CD =
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
        lf.CM2)

        #Thrust
        T = 0.5*rho*V**2*self.Sw*CD

        #Power Required
        PR = T*V

        #Power Available
        if h <36131:
            m=0.7
        else:
            m=1.0
```

```
        PA =                                                    ↵
        ((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+self. ↵
        b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)*V/1.6878 ↵
        1*V/1.68781)*550
        # ~ print(i, self.W[i])

        climb = (PA-PR)/self.W[i]*60

        #ESFC
        esfc =                                                  ↵
        ((self.a00+self.a10*h+self.a20*h*h)+(self.a01+self.a11*h+self. ↵
        a21*h*h)*V/1.68781+(self.a02+self.a12*h+self.a22*h*h)*V/1.6878 ↵
        1*V/1.68781)/550/(32.174*3600)

        self.h[i] = h
        self.V[i] = V
        self.rho[i] = rho
        self.a[i] = a
        self.Temp[i] = Temp
        self.CL[i] = CL
        self.CD[i] = CD
        self.T[i] = T
        self.esfc[i] = esfc
        self.climb[i] = climb

    def breguet(self, y, flag='obj'):
        h = y[0]
        V = y[1]
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)
        # ~ print(Temp, TSL)

        #Lift coefficient
        if self.run_type == 'f':
            CL = self.W[0]/(0.5*rho*V**2*self.Sw)
        else:
            CL = self.W[-1]/(0.5*rho*V**2*self.Sw)

        #Mach number
        M = V/a
```

```
#Drag coefficient
CD = (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**self.CM2)


#TSFC
esfc = ((self.a00+self.a10*h+self.a20*h*h)+(self.a01+self.a11*h+self.a21*h*h)*V/1.68781+(self.a02+self.a12*h+self.a22*h*h)*V/1.68781*V/1.68781)/550/(32.174*3600)
c = esfc*V

if self.run_direction == 'f':
    Wf = self.W[0]/np.exp(self.r*self.g*c/(V*CL/CD))

    self.hb = h
    self.Vb = V
    self.L_Db = CL/CD
    self.esfcb = esfc
    self.Wfb = Wf
    self.W[-1] = self.Wfb
    self.Mb = M
    self.tb = self.r/V

    # ~ print('fuel_burn', c*T*self.g*t)
    # ~ print('velocity: ', V)
    # ~ print('altitude: ', h)
    # ~ print('Mach: ', M)
    # ~ print('Drag Coefficient: ', CD)
    # ~ print('Thrust: ', T)
    # ~ print('TSFC: ', c)
    # ~ print('Weight: ', self.W[i])
    # ~ print('-----------------------------------')
    #Power Req.
    PR = 0.5*rho*V**3*self.Sw*CD

    #Power Available
    if h <36131:
        m=0.7
    else:
        m=1.0
    PA = ((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+self.b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)*V/1.68781*V/1.68781)*550

    climb = (PA-PR)/self.W[0]*60
```

```
            if climb>100.:
                return self.W[0]-Wf
            else:
                return (self.W[0]-Wf)+10*(climb-100)**2

        elif self.run_direction == 'b':
            # ~ print(V, CL, CD)
            Wi = self.W[-1]*np.exp(self.r*self.g*c/(V*CL/CD))

            self.hb = h
            self.Vb = V
            self.L_Db = CL/CD
            self.esfcb = esfc
            self.iweight = Wi
            self.W[0] = self.iweight
            self.Mb = M
            self.tb = self.r/V

            # ~ print('fuel_burn', c*T*self.g*t)
            # ~ print('velocity: ', V)
            # ~ print('altitude: ', h)
            # ~ print('Mach: ', M)
            # ~ print('Drag Coefficient: ', CD)
            # ~ print('Thrust: ', T)
            # ~ print('TSFC: ', c)
            # ~ print('Weight: ', self.W[i])
            # ~ print('-----------------------------------')

            #Power Req.
            PR = 0.5*rho*V**3*self.Sw*CD

            #Power Available
            if h <36131:
                m=0.7
            else:
                m=1.0
            PA =                                                    ↵
            ((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+s ↵
            elf.b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)* ↵
            V/1.68781*V/1.68781)*550

            climb = (PA-PR)/self.W[-1]*60

            if climb>100.:
                return Wi-self.W[-1]
            else:
                return (Wi-self.W[-1])+10*(climb-100)**2
```

```
def breguet_opt(self):
    y0 = [20000., 200.0]
    bnds = ((100., 80000.), (135.0, 1200.))
    cons = [{"type" : "ineq",
             "fun" : self.breguet}]
    ans = optimize.minimize(self.breguet,
                            y0,
                            method='SLSQP',
                            bounds = bnds,
                            # ~ constraints = [{"type" : "ineq",
                                        # ~ "fun" : self.breguet,
                                        # ~ "args" : ("cons",)}],
                            options={'disp' : False,
                                     'ftol' : 1e-16})

def service_ceiling(self, y, i):
    # ~ print(y, i)
    h = y[0]

    #atmospheric properties
    if self.english_units == True:
        statmos = std.StandardAtmosphere('English')
    else:
        statmos = std.StandardAtmosphere('SI')

    rho = statmos.rho(h)
    a = statmos.a(h)
    Temp = statmos.T(h)
    TSL = statmos.T(0)
    rhoSL = statmos.rho(0)

    if self.run_type == 'Vmf':
        V = optimize.newton(self.Vmfp, 800.0, args=(rho, a, i))
    elif self.run_type == 'Vmd':
        V = optimize.newton(self.Vmd, 800.0, args=(rho, a, i))
    else:
        V = y[1]

    #Lift coefficient
    CL = self.W[i]/(0.5*rho*V**2*self.Sw)

    #Mach number
    M = V/a

    #Drag coefficient
    CD =
    (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
    lf.CM2)
```

```
        #Power Req.
        PR = 0.5*rho*V**3*self.Sw*CD

        #Power Available
        if h <36131:
            m=0.7
        else:
            m=1.0
        PA =
        ((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+self.
        b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)*V/1.6878
        1*V/1.68781)*550

        climb = (PA-PR)/self.W[i]*60

        return climb-100.+400.

    def trajectory_opt(self, r):
        h = r*10000.
        opts = {'disp' : False,
                'maxiter' : 500}#,
                # ~ 'eps' : 1e-10,
                # ~ 'ftol' : 1e-11}
        tolerance = 1e-14
        iguess = [250.]

        if self.run_direction == 'b':

            if self.run_type == 'full_opt':
                #Full Optimization

                #-------------------------------------------------------
                ---------------------------------
                y0 = iguess
                bnds = ((10.0, 1800.),)
                for i in range(1, self.n+1):
                    j = self.n-i
                    ans = optimize.minimize(self.opt_h_V,
                                            y0, args=(h, j),
                                            method='SLSQP',
                                            bounds = bnds,
                                            tol = tolerance,
                                            options=opts)

                    # print(i, self.x[i], ans.x[0],
                    ans.x[1])
                    self.set_state(ans.x[0], h, j)

                    if j>0:
```

```
                    self.W[j-1] =
                    self.W[j]+self.esfc[j]*self.T[j]*self.V[j]*sel
                    f.g*(self.x[j]-self.x[j-1])/self.V[j]
                if ans.success == False:
                    print('optimization failed', ans.message)
                    y0 = ans.x
                #print(y0)
            for i in range(0, self.n):
                if i<self.n-1:
                    self.t[i+1] =
                    self.t[i]+(self.x[i+1]-self.x[i])/self.V[i]
                #print(y0)

    self.L = np.multiply(self.Sw*0.5*self.rho,
    np.multiply(np.multiply(self.V, self.V), self.CL))
    self.D = np.multiply(self.Sw*0.5*self.rho,
    np.multiply(np.multiply(self.V, self.V), self.CD))
    self.L_D = np.divide(self.L, self.D)

    print(' ')
    print('altitude: ', h)
    print(self.W[0]-self.W[-1])
    print(np.min(self.climb))

    # ~ return(self.W[0]-self.W[-1])
    # ~ if np.isnan(self.W[0]-self.W[-1]):
        # ~ return 1e16
    if np.min(self.climb) > 100:
        return (self.W[0]-self.W[-1])
    else:
        return
        ((self.W[0]-self.W[-1])+10*(np.min(self.climb)-100)**2)

def trajectory_cases(self):
    data_format="{0:<30}{1:<30.16f}\n"

    dist_header="{0:<30}{1:<30}{2:<30}{3:<30}{4:<32}{5:<30}{6:<30}
    {7:<32}{8:<30}{9:<30}{10:<32}{11:<30}{12:<30}\n"

    dist_format="{0:<30.16f}{1:<30.16f}{2:<30.16f}{3:<30.16f}{4:<3
    2.16f}{5:<30.16f}{6:<30.16f}{7:<32.16f}{8:<30.16f}{9:<30.16f}{
    10:<30.16f}{11:<30.16f}{12:<30.16f}\n"

    self.fig_path = self.results_path+'figs/'
    if self.run_type == 'breguet':
        self.breguet_opt()
        print('Run Type: ', self.run_type)
        if self.run_direction == 'f':
            print('Total Fuel Burn: ', self.W[0]-self.Wfb, 'lbf')
```

```
        else:
            print('Total Fuel Burn: ', self.iweight-self.W[-1],
            'lbf')
        print('Total Time: ', self.tb/3600.0, 'hours')
        print('L/D: ', self.L_Db)
        print('V: ', self.Vb, 'ft/s')
        print('h: ', self.hb, 'ft')
        print('TSFC: ', self.esfcb, 'slugs/lbf/s')
        print('Mach: ', self.Mb)
        print('total run time: ', time.time()-self.s_runtime)
        print('-------------------------------------------------
        ')

        # ~ data_file =
        open(self.results_path+str(self.RA)+'_'+str(self.e)+'_data
        _b.txt', 'w')
        # ~ if self.run_direction == 'f':
            # ~ data_file.write(data_format.format('Total Fuel
            Burn:', self.W[0]-self.Wfb))
            # ~ self.W[-1] = self.Wfb
        # ~ else:
            # ~ data_file.write(data_format.format('Total Fuel
            Burn:', self.iweight-self.W[-1]))
            # ~ self.W[0] = self.iweight
        # ~ data_file.write(data_format.format('Total Cruise
        Time:', self.tb/3600.0))
        # ~ self.t[-1] = self.tb
        # ~ data_file.write(data_format.format('Run Time:',
        time.time()-self.s_runtime))
        # ~ data_file.close()



else:
    h0 = 0.0001
    bnds = ((0.0001, 8.),)
    tolerance = 1e-14
    opts = {'disp' : True,
        'maxiter' : 500,
        'eps' : 1e-10,
        'ftol' : 1e-11}
    ans = optimize.minimize(self.trajectory_opt,
                                    h0,
                                    method='SLSQP',
                                    bounds = bnds,
                                    tol = tolerance,
                                    options=opts)
    self.h[:] = ans.x[0]*10000.
```

```
        print('Run Type:', self.run_type)
        print('Total Fuel Burn: ', self.W[0]-self.W[-1], 'lbf')
        print('Total Time: ', self.t[-1]/3600.0, 'hours')
        print('total run time: ', time.time()-self.s_runtime)
                                                                    ↵
        print('-------------------------------------------------- ↵
        ')

        # ~ for i in range(0, len(self.x)):
            # ~ print(self.rho[i])
                                                                    ↵
    #-------------------------------------------------------------- ↵
    -------------------------
        '''
        data_file =                                                 ↵
        open(self.results_path+str(self.RA)+'_'+str(self.e)+'_data ↵
        .txt', 'w')
        data_file.write(data_format.format('Total Fuel Burn:',      ↵
        self.W[0]-self.W[-1]))
        data_file.write(data_format.format('Total Cruise Time:',    ↵
        self.t[-1]/3600.0))
        data_file.write(data_format.format('Run Time:',             ↵
        time.time()-self.s_runtime))
        data_file.close()

        dist_file =                                                 ↵
        open(self.results_path+str(self.RA)+'_'+str(self.e)+'_dist ↵
        s.txt', 'w')
        dist_file.write(dist_header.format('x[ft]',                 ↵
        'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/ ↵
        D','T[lbf]','ESFC[slugs/lbf ft/s /s]','W[lbf]'))
        for i in range(0, len(self.x)):
            dist_file.write(dist_format.format(self.x[i],           ↵
            self.x[i]/5280., self.t[i], self.t[i]/3600.0,           ↵
            self.h[i], self.V[i], self.V[i]/self.a[i],              ↵
            self.CL[i], self.CD[i], self.CL[i]/self.CD[i],          ↵
            self.T[i], self.esfc[i], self.W[i]))
        dist_file.close()

        plot_range_multiplier = 100.

        #Altitude
        plt.figure(1)
        plt.plot(self.xm, self.h)
        plt.xlabel('x [mi]')
        plt.ylabel('h [ft]')
                                                                    ↵
        plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_h ↵
        .png')
```

```
#Velocity
plt.figure(2)
plt.plot(self.xm, self.V)
plt.xlabel('x [mi]')
plt.ylabel('V [ft/s]')
                                                                    ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_V ↵
.png')

#Mach
plt.figure(3)
plt.plot(self.xm, np.divide(self.V,self.a))
plt.xlabel('x [mi]')
plt.ylabel('M')
                                                                    ↵
plt.ylim(np.amin(np.divide(self.V,self.a))-(np.amax(np.div ↵
ide(self.V,self.a))-np.amin(np.divide(self.V,self.a)))*plo ↵
t_range_multiplier,                                                 ↵
np.amax(np.divide(self.V,self.a))+(np.amax(np.divide(self. ↵
V,self.a))-np.amin(np.divide(self.V,self.a)))*plot_range_m ↵
ultiplier)
                                                                    ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_M ↵
.png')


#CL
plt.figure(4)
plt.plot(self.xm, self.CL)
plt.xlabel('x [mi]')
plt.ylabel('CL')
                                                                    ↵
plt.ylim(np.amin(self.CL)-(np.amax(self.CL)-np.amin(self.C ↵
L))*plot_range_multiplier,                                          ↵
np.amax(self.CL)+(np.amax(self.CL)-np.amin(self.CL))*plot_ ↵
range_multiplier)
                                                                    ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_C ↵
L.png')

#CD
plt.figure(5)
plt.plot(self.xm, self.CD)
plt.xlabel('x [mi]')
plt.ylabel('CD')
                                                                    ↵
plt.ylim(np.amin(self.CD)-(np.amax(self.CD)-np.amin(self.C ↵
D))*plot_range_multiplier,                                          ↵
```

```
np.amax(self.CD)+(np.amax(self.CD)-np.amin(self.CD))*plot_
range_multiplier)

plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_C
D.png')


#L/D
plt.figure(6)
plt.plot(self.xm, self.L_D)
plt.xlabel('x [mi]')
plt.ylabel('L/D')

plt.ylim(np.amin(self.L_D)-(np.amax(self.L_D)-np.amin(self
.L_D))*plot_range_multiplier,
np.amax(self.L_D)+(np.amax(self.L_D)-np.amin(self.L_D))*pl
ot_range_multiplier)

plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_L
_D.png')


#T
plt.figure(7)
plt.plot(self.xm, self.T)
plt.xlabel('x [mi]')
plt.ylabel('T [lbf]')

plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_T
.png')

#TSFC
plt.figure(8)
plt.plot(self.xm, self.esfc)
plt.xlabel('x [mi]')
plt.ylabel('ESFC [slugs/lbf ft/s /s]')

plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_E
SFC.png')

#W
plt.figure(9)
plt.plot(self.xm, self.W)
plt.xlabel('x [mi]')
plt.ylabel('W [lbf]')

plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_W
.png')
# ~ plt.show()
```

```
            plt.figure(1).clear()
            plt.figure(2).clear()
            plt.figure(3).clear()
            plt.figure(4).clear()
            plt.figure(5).clear()
            plt.figure(6).clear()
            plt.figure(7).clear()
            plt.figure(8).clear()
            plt.figure(9).clear()
            '''

class Aircraft_fixed_h(object):

    def __init__(self, filename):
        with open(filename) as input_file:
            data = json.load(input_file, object_pairs_hook=OrderedDict)

        self.input_airplane(data)
        self.grid_setup()
        self.array_initialize()

    def input_airplane(self, data):

        #Units
        if data["units"] == 'English':
            self.english_units = True
        else:
            self.english_units = False

        # CRM Constant Properties
        self.iweight = data["MTOW"]
        self.fweight = data["M15FW"]
        self.Sw = data["wing_area"]
        self.CD0 = data["CD0"]
        self.CD1 = data["CD1"]
        self.CD2 = data["CD2"]
        self.CM0 = data["CM0"]
        self.CM1 = data["CM1"]
        self.CM2 = data["CM2"]
        self.r = data["range"]*5280.0
        self.a00 = data["a00"]
        self.a01 = data["a01"]
        self.a02 = data["a02"]
        self.a10 = data["a10"]
        self.a11 = data["a11"]
        self.a12 = data["a12"]
        self.a20 = data["a20"]
        self.a21 = data["a21"]
```

```
        self.a22 = data["a22"]
        self.b00 = data["b00"]
        self.b01 = data["b01"]
        self.b02 = data["b02"]
        self.b10 = data["b10"]
        self.b11 = data["b11"]
        self.b12 = data["b12"]
        self.b20 = data["b20"]
        self.b21 = data["b21"]
        self.b22 = data["b22"]
        self.n = data["grid"]
        self.g = 32.174
        self.name = data["name"]
        self.run_type = 'none'
        self.run_direction = data["direction"]
        self.s_runtime = 0.0
        self.f_runtime = 0.0
        self.results_path = '.'
        self.RA = 16.5
        self.e = 1.
        self.hf = 19720.0

    def grid_setup(self):
        self.x = np.linspace(0, self.r, self.n)
        self.xm = self.x/5280.0

    def array_initialize(self):
        self.W = np.zeros(self.n)
        if self.run_direction == "f":
            self.W[0] = self.iweight
        elif self.run_direction == "b":
            self.W[-1] = self.fweight
        self.CL = np.zeros(self.n)
        self.CD = np.zeros(self.n)
        self.V = np.zeros(self.n)
        self.h = np.zeros(self.n)
        self.t = np.zeros(self.n)
        self.esfc = np.zeros(self.n)
        self.rho = np.zeros(self.n)
        self.a = np.zeros(self.n)
        self.Temp = np.zeros(self.n)
        self.T = np.zeros(self.n)
        self.climb = np.zeros(self.n)

    def opt_h_V(self, y, h, i):
        V = y*100.
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
```

```
else:
    statmos = std.StandardAtmosphere('SI')

rho = statmos.rho(h)
a = statmos.a(h)
Temp = statmos.T(h)
TSL = statmos.T(0)
rhoSL = statmos.rho(0)
# ~ print(Temp, TSL)

#Lift coefficient
CL = self.W[i]/(0.5*rho*V**2*self.Sw)

#Mach number
M = V/a

#Drag coefficient
CD =
(self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
lf.CM2)

#Thrust
T = 0.5*rho*V**2*self.Sw*CD

#Power Required
PR = T*V

#Power Available
if h <36131:
    m=0.7
else:
    m=1.0
PA =
((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+self.
b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)*V/1.6878
1*V/1.68781)*550
# ~ print(i, self.W[i])

climb = (PA-PR)/self.W[i]*60
# ~ print(h, V)

#ESFC
esfc =
((self.a00+self.a10*h+self.a20*h*h)+(self.a01+self.a11*h+self.
a21*h*h)*V/1.68781+(self.a02+self.a12*h+self.a22*h*h)*V/1.6878
1*V/1.68781)/550/(32.174*3600)

if i==self.n-1:
    dx = self.x[i]-self.x[i-1]
```

```
        else:
            dx = self.x[i+1]-self.x[i]

        t = dx/V

        # ~ print('fuel_burn', c*T*self.g*t)
        # ~ print('velocity: ', V)
        # ~ print('altitude: ', h)
        # ~ print('Mach: ', M)
        # ~ print('Drag Coefficient: ', CD)
        # ~ print('Thrust: ', T)
        # ~ print('TSFC: ', c)
        # ~ print('Weight: ', self.W[i])
        # ~ print('-----------------------------------')

        if climb>100.:
            # ~ print(esfc*PR*self.g*t)
            return esfc*PR*self.g*t
        else:
            # ~ print('yo',esfc*PR*self.g*t)#+10*(climb-100)**2)
            return esfc*PR*self.g*t+10*(climb-100)**2

    def set_state(self, y, h, i):
        V = y*100.
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)

        #Lift coefficient
        CL = self.W[i]/(0.5*rho*V**2*self.Sw)

        #Mach number
        M = V/a

        #Drag coefficient
        CD =                                                              ↵
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se ↵
        lf.CM2)

        #Thrust
        T = 0.5*rho*V**2*self.Sw*CD
```

```
        #Power Required
        PR = T*V

        #Power Available
        if h <36131:
            m=0.7
        else:
            m=1.0
        PA =                                                    ⤶
        ((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+self. ⤶
        b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)*V/1.6878 ⤶
        1*V/1.68781)*550
        # ~ print(i, self.W[i])

        climb = (PA-PR)/self.W[i]*60

        #ESFC
        esfc =                                                  ⤶
        ((self.a00+self.a10*h+self.a20*h*h)+(self.a01+self.a11*h+self. ⤶
        a21*h*h)*V/1.68781+(self.a02+self.a12*h+self.a22*h*h)*V/1.6878 ⤶
        1*V/1.68781)/550/(32.174*3600)

        self.h[i] = h
        self.V[i] = V
        self.rho[i] = rho
        self.a[i] = a
        self.Temp[i] = Temp
        self.CL[i] = CL
        self.CD[i] = CD
        self.T[i] = T
        self.esfc[i] = esfc
        self.climb[i] = climb

    def breguet(self, y, h, flag='obj'):
        # ~ h = y[0]
        V = y[0]*100.
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)
        # ~ print(Temp, TSL)
```

```
#Lift coefficient
if self.run_type == 'f':
    CL = self.W[0]/(0.5*rho*V**2*self.Sw)
else:
    CL = self.W[-1]/(0.5*rho*V**2*self.Sw)

#Mach number
M = V/a

#Drag coefficient
CD = 
(self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
lf.CM2)


#TSFC
esfc = 
((self.a00+self.a10*h+self.a20*h*h)+(self.a01+self.a11*h+self.
a21*h*h)*V/1.68781+(self.a02+self.a12*h+self.a22*h*h)*V/1.6878
1*V/1.68781)/550/(32.174*3600)
c = esfc*V

if self.run_direction == 'f':
    Wf = self.W[0]/np.exp(self.r*self.g*c/(V*CL/CD))

    self.hb = h
    self.Vb = V
    self.L_Db = CL/CD
    self.esfcb = esfc
    self.Wfb = Wf
    self.W[-1] = self.Wfb
    self.Mb = M
    self.tb = self.r/V

    #Power Req.
    PR = 0.5*rho*V**3*self.Sw*CD

    #Power Available
    if h <36131:
        m=0.7
    else:
        m=1.0
    PA = 
    ((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+s
    elf.b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)*
    V/1.68781*V/1.68781)*550

    climb = (PA-PR)/self.W[0]*60
```

```
            if climb>100.:
                return self.W[0]-Wf
            else:
                return (self.W[0]-Wf)+10*(climb-100)**2

        elif self.run_direction == 'b':
            # ~ print(V, CL, CD)
            Wi = self.W[-1]*np.exp(self.r*self.g*c/(V*CL/CD))

            self.hb = h
            self.Vb = V
            self.L_Db = CL/CD
            self.esfcb = esfc
            self.iweight = Wi
            self.W[0] = self.iweight
            self.Mb = M
            self.tb = self.r/V

            #Power Req.
            PR = 0.5*rho*V**3*self.Sw*CD

            #Power Available
            if h <36131:
                m=0.7
            else:
                m=1.0
            PA =                                                    ↵
            ((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+s ↵
            elf.b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)* ↵
            V/1.68781*V/1.68781)*550

            climb = (PA-PR)/self.W[-1]*60

            if climb>100.:
                return Wi-self.W[-1]
            else:
                return (Wi-self.W[-1])+10*(climb-100)**2

    def breguet_opt(self, h):
        y0 = [2.000]
        bnds = ((1.350, 12.00),)
        cons = [{"type" : "ineq",
                 "fun" : self.breguet}]
        ans = optimize.minimize(self.breguet,
                                y0, args=(h),
                                method='SLSQP',
                                bounds = bnds,
                                options={'disp' : False,
```

```
                                        'ftol' : 1e-16})

    def service_ceiling(self, y, i):
        # ~ print(y, i)
        h = y[0]

        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)

        if self.run_type == 'Vmf':
            V = optimize.newton(self.Vmfp, 800.0, args=(rho, a, i))
        elif self.run_type == 'Vmd':
            V = optimize.newton(self.Vmd, 800.0, args=(rho, a, i))
        else:
            V = y[1]

        #Lift coefficient
        CL = self.W[i]/(0.5*rho*V**2*self.Sw)

        #Mach number
        M = V/a

        #Drag coefficient
        CD =                                                          ↵
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se ↵
        lf.CM2)

        #Power Req.
        PR = 0.5*rho*V**3*self.Sw*CD

        #Power Available
        if h <36131:
            m=0.7
        else:
            m=1.0
        PA =                                                          ↵
        ((self.b00+self.b10*h+self.b20*h*h)+(self.b01+self.b11*h+self. ↵
        b21*h*h)*V/1.68781+(self.b02+self.b12*h+self.b22*h*h)*V/1.6878 ↵
        1*V/1.68781)*550
```

```
        climb = (PA-PR)/self.W[i]*60

        return climb-100.+400.

    def trajectory_opt(self, r):
        h = r
        opts = {'disp' : False,
                'maxiter' : 500}#,
                # ~ 'eps' : 1e-10,
                # ~ 'ftol' : 1e-11}
        tolerance = 1e-14
        iguess = [2.50]

        if self.run_direction == 'b':

            if self.run_type == 'full_opt':
                #Full Optimization
                #------------------------------------------------------
                ----------------------------------
                y0 = iguess
                bnds = ((.100, 18.00),)
                for i in range(1, self.n+1):
                    j = self.n-i
                    ans = optimize.minimize(self.opt_h_V,
                                            y0, args=(h, j),
                                            method='SLSQP',
                                            bounds = bnds,
                                            tol = tolerance,
                                            options=opts)

                    self.set_state(ans.x[0], h, j)

                    if j>0:
                        self.W[j-1] =
                        self.W[j]+self.esfc[j]*self.T[j]*self.V[j]*sel
                        f.g*(self.x[j]-self.x[j-1])/self.V[j]
                    if ans.success == False:
                        print('optimization failed', ans.message)
                        y0 = ans.x

                for i in range(0, self.n):
                    if i<self.n-1:
                        self.t[i+1] =
                        self.t[i]+(self.x[i+1]-self.x[i])/self.V[i]

        self.L = np.multiply(self.Sw*0.5*self.rho,
        np.multiply(np.multiply(self.V, self.V), self.CL))
        self.D = np.multiply(self.Sw*0.5*self.rho,
```

```
        np.multiply(np.multiply(self.V, self.V), self.CD))
        self.L_D = np.divide(self.L, self.D)

        print(' ')
        print('altitude: ', h)
        print(self.W[0]-self.W[-1])
        print(np.min(self.climb))

        if np.min(self.climb) > 100:
            return (self.W[0]-self.W[-1])
        else:
            return
            ((self.W[0]-self.W[-1])+10*(np.min(self.climb)-100)**2)

    def trajectory_cases(self):
        data_format="{0:<30}{1:<30.16f}\n"

        dist_header="{0:<30}{1:<30}{2:<30}{3:<30}{4:<32}{5:<30}{6:<30}
        {7:<32}{8:<30}{9:<30}{10:<32}{11:<30}{12:<30}\n"

        dist_format="{0:<30.16f}{1:<30.16f}{2:<30.16f}{3:<30.16f}{4:<3
        2.16f}{5:<30.16f}{6:<30.16f}{7:<32.16f}{8:<30.16f}{9:<30.16f}{
        10:<30.16f}{11:<30.16f}{12:<30.16f}\n"

        self.fig_path = self.results_path+'figs/'
        if self.run_type == 'breguet':
            h0 = self.hf
            self.breguet_opt(h0)
            print('Run Type: ', self.run_type)
            if self.run_direction == 'f':
                print('Total Fuel Burn: ', self.W[0]-self.Wfb, 'lbf')
            else:
                print('Total Fuel Burn: ', self.iweight-self.W[-1],
                'lbf')
            print('Total Time: ', self.tb/3600.0, 'hours')
            print('L/D: ', self.L_Db)
            print('V: ', self.Vb, 'ft/s')
            print('h: ', self.hb, 'ft')
            print('TSFC: ', self.esfcb, 'slugs/lbf/s')
            print('Mach: ', self.Mb)
            print('total run time: ', time.time()-self.s_runtime)

            print('-------------------------------------------------
            ')


        else:
            h0 = self.hf
            self.trajectory_opt(h0)
```

```
print('Run Type:', self.run_type)
print('Total Fuel Burn: ', self.W[0]-self.W[-1], 'lbf')
print('Total Time: ', self.t[-1]/3600.0, 'hours')
print('total run time: ', time.time()-self.s_runtime)
print('-------------------------------------------------
')


'''
data_file =
open(self.results_path+str(self.RA)+'_'+str(self.e)+'_data
.txt', 'w')
data_file.write(data_format.format('Total Fuel Burn:',
self.W[0]-self.W[-1]))
data_file.write(data_format.format('Total Cruise Time:',
self.t[-1]/3600.0))
data_file.write(data_format.format('Run Time:',
time.time()-self.s_runtime))
data_file.close()

dist_file =
open(self.results_path+str(self.RA)+'_'+str(self.e)+'_dist
s.txt', 'w')
dist_file.write(dist_header.format('x[ft]',
'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/
D','T[lbf]','ESFC[slugs/lbf ft/s /s]','W[lbf]'))
for i in range(0, len(self.x)):
    dist_file.write(dist_format.format(self.x[i],
    self.x[i]/5280., self.t[i], self.t[i]/3600.0,
    self.h[i], self.V[i], self.V[i]/self.a[i],
    self.CL[i], self.CD[i], self.CL[i]/self.CD[i],
    self.T[i], self.esfc[i], self.W[i]))
dist_file.close()

plot_range_multiplier = 100.

#Altitude
plt.figure(1)
plt.plot(self.xm, self.h)
plt.xlabel('x [mi]')
plt.ylabel('h [ft]')
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_h
.png')

#Velocity
plt.figure(2)
```

```
plt.plot(self.xm, self.V)
plt.xlabel('x [mi]')
plt.ylabel('V [ft/s]')
                                                                    ⤶
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_V      ⤶
.png')

#Mach
plt.figure(3)
plt.plot(self.xm, np.divide(self.V,self.a))
plt.xlabel('x [mi]')
plt.ylabel('M')
                                                                    ⤶
plt.ylim(np.amin(np.divide(self.V,self.a))-(np.amax(np.div      ⤶
ide(self.V,self.a))-np.amin(np.divide(self.V,self.a)))*plo      ⤶
t_range_multiplier,                                             ⤶
np.amax(np.divide(self.V,self.a))+(np.amax(np.divide(self.      ⤶
V,self.a))-np.amin(np.divide(self.V,self.a)))*plot_range_m      ⤶
ultiplier)
                                                                    ⤶
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_M      ⤶
.png')


#CL
plt.figure(4)
plt.plot(self.xm, self.CL)
plt.xlabel('x [mi]')
plt.ylabel('CL')
                                                                    ⤶
plt.ylim(np.amin(self.CL)-(np.amax(self.CL)-np.amin(self.C      ⤶
L))*plot_range_multiplier,                                      ⤶
np.amax(self.CL)+(np.amax(self.CL)-np.amin(self.CL))*plot_      ⤶
range_multiplier)
                                                                    ⤶
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_C      ⤶
L.png')

#CD
plt.figure(5)
plt.plot(self.xm, self.CD)
plt.xlabel('x [mi]')
plt.ylabel('CD')
                                                                    ⤶
plt.ylim(np.amin(self.CD)-(np.amax(self.CD)-np.amin(self.C      ⤶
D))*plot_range_multiplier,                                      ⤶
np.amax(self.CD)+(np.amax(self.CD)-np.amin(self.CD))*plot_      ⤶
range_multiplier)
                                                                    ⤶
```

```
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_C
D.png')


#L/D
plt.figure(6)
plt.plot(self.xm, self.L_D)
plt.xlabel('x [mi]')
plt.ylabel('L/D')
                                                          
plt.ylim(np.amin(self.L_D)-(np.amax(self.L_D)-np.amin(self
.L_D))*plot_range_multiplier,                             
np.amax(self.L_D)+(np.amax(self.L_D)-np.amin(self.L_D))*pl
ot_range_multiplier)
                                                          
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_L
_D.png')


#T
plt.figure(7)
plt.plot(self.xm, self.T)
plt.xlabel('x [mi]')
plt.ylabel('T [lbf]')
                                                          
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_T
.png')

#TSFC
plt.figure(8)
plt.plot(self.xm, self.esfc)
plt.xlabel('x [mi]')
plt.ylabel('ESFC [slugs/lbf ft/s /s]')
                                                          
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_E
SFC.png')

#W
plt.figure(9)
plt.plot(self.xm, self.W)
plt.xlabel('x [mi]')
plt.ylabel('W [lbf]')
                                                          
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_W
.png')
# ~ plt.show()

plt.figure(1).clear()
plt.figure(2).clear()
```

```
            plt.figure(3).clear()
            plt.figure(4).clear()
            plt.figure(5).clear()
            plt.figure(6).clear()
            plt.figure(7).clear()
            plt.figure(8).clear()
            plt.figure(9).clear()
            '''

def general(Ikhana_fo, Ikhana_b, sdata_file, sdist_file,
sdata_format, sdist_format, sdist_header):


    for RA in [4,6,8,10,12,14,16,18,20]:
        for e in
        [0.2,0.25,0.3,0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.8
        5,0.9,0.95,1.]:

            Ikhana_fo.RA = RA
            Ikhana_fo.e = e
            Ikhana_fo.CD2 = 1./(ma.pi*RA*e)
            Ikhana_fo.s_runtime = time.time()
            Ikhana_fo.trajectory_cases()
            Ikhana_fo.f_runtime = time.time()

            Ikhana_b.RA = RA
            Ikhana_b.e = e
            Ikhana_b.CD2 = 1./(ma.pi*RA*e)
            Ikhana_b.s_runtime = time.time()
            Ikhana_b.trajectory_cases()
            Ikhana_b.f_runtime = time.time()

            sdata_file.write(sdata_format.format(Ikhana_fo.RA,
            Ikhana_fo.e, 1./(ma.pi*Ikhana_fo.RA*Ikhana_fo.e),
            Ikhana_fo.W[0]-Ikhana_fo.W[-1],
            Ikhana_b.W[0]-Ikhana_b.W[-1], Ikhana_fo.t[-1]/3600.,
            Ikhana_b.tb/3600., np.mean(Ikhana_fo.h),
            np.mean(Ikhana_fo.V),
            np.mean(np.divide(Ikhana_fo.V,Ikhana_fo.a)),
            np.mean(Ikhana_fo.CL), np.mean(Ikhana_fo.CD),
            np.mean(Ikhana_fo.L_D), np.mean(Ikhana_fo.T),
            np.mean(Ikhana_fo.esfc)))


            sdist_file.write(str(Ikhana_fo.RA)+'\t'+str(Ikhana_fo.e)+'
            \n')
            sdist_file.write(sdist_header.format('x[ft]',
            'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/
            D','T[lbf]','ESFC[slugs/lbf ft/s /s]','W[lbf]'))
```

```
            for i in range(0, len(Ikhana_fo.x)):
                sdist_file.write(sdist_format.format(Ikhana_fo.x[i],
                Ikhana_fo.x[i]/5280., Ikhana_fo.t[i],
                Ikhana_fo.t[i]/3600.0, Ikhana_fo.h[i],
                Ikhana_fo.V[i], Ikhana_fo.V[i]/Ikhana_fo.a[i],
                Ikhana_fo.CL[i], Ikhana_fo.CD[i],
                Ikhana_fo.CL[i]/Ikhana_fo.CD[i], Ikhana_fo.T[i],
                Ikhana_fo.esfc[i], Ikhana_fo.W[i]))
            sdist_file.write('\n')

    sdata_file.close()
    sdist_file.close()

def wingspan(Ikhana_fo, Ikhana_b, sdata_file, sdist_file,
sdata_format, sdist_format, sdist_header):

    CD2p = 0.017
    B3 =
    [-0.333333333,-0.3,-0.275,-0.25,-0.225,-0.2,-0.175,-0.15,-0.125,-0
    .1,-0.075,-0.05,-0.025,0.]
    # ~ B3 = [-0.1476]
    b =
    [76.47728384,75.14525373,74.20467605,73.30948942,72.45599409,71.64
    090659,70.86130084,70.11455917,69.39833135,68.71050012,68.04915205
    ,67.41255273,66.7991256,66.20743376]
    # ~ b = [70.04465]
    for k in range(0, len(B3)):

        es = 1./(1.+3.*B3[k]*B3[k])

        Ikhana_fo.RA = b[k]**2/Ikhana_fo.Sw
        Ikhana_fo.CD2 = CD2p+1./(ma.pi*Ikhana_fo.RA*es)
        Ikhana_fo.e = 1./(ma.pi*Ikhana_fo.RA*Ikhana_fo.CD2)
        Ikhana_fo.s_runtime = time.time()
        Ikhana_fo.trajectory_cases()
        Ikhana_fo.f_runtime = time.time()

        Ikhana_b.RA = b[k]**2/Ikhana_b.Sw
        Ikhana_b.CD2 = CD2p+1./(ma.pi*Ikhana_b.RA*es)
        Ikhana_b.e = 1./(ma.pi*Ikhana_b.RA*Ikhana_b.CD2)
        Ikhana_b.s_runtime = time.time()
        Ikhana_b.trajectory_cases()
        Ikhana_b.f_runtime = time.time()

        sdata_file.write(sdata_format.format(Ikhana_fo.RA,
        Ikhana_fo.e, 1./(ma.pi*Ikhana_fo.RA*Ikhana_fo.e),
        Ikhana_fo.W[0]-Ikhana_fo.W[-1],
        Ikhana_b.W[0]-Ikhana_b.W[-1], Ikhana_fo.t[-1]/3600.,
        Ikhana_b.tb/3600., np.mean(Ikhana_fo.h),
```

```
            np.mean(Ikhana_fo.V),                                          ↵
            np.mean(np.divide(Ikhana_fo.V,Ikhana_fo.a)),                   ↵
            np.mean(Ikhana_fo.CL), np.mean(Ikhana_fo.CD),                  ↵
            np.mean(Ikhana_fo.L_D), np.mean(Ikhana_fo.T),                  ↵
            np.mean(Ikhana_fo.esfc)))

            sdist_file.write(str(Ikhana_fo.RA)+'\t'+str(Ikhana_fo.e)+'\n')
            sdist_file.write(sdist_header.format('x[ft]',                  ↵
            'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/D',' ↵
            T[lbf]','ESFC[slugs/lbf ft/s /s]','W[lbf]'))
            for i in range(0, len(Ikhana_fo.x)):
                sdist_file.write(sdist_format.format(Ikhana_fo.x[i],       ↵
                Ikhana_fo.x[i]/5280., Ikhana_fo.t[i],                      ↵
                Ikhana_fo.t[i]/3600.0, Ikhana_fo.h[i], Ikhana_fo.V[i],     ↵
                Ikhana_fo.V[i]/Ikhana_fo.a[i], Ikhana_fo.CL[i],            ↵
                Ikhana_fo.CD[i], Ikhana_fo.CL[i]/Ikhana_fo.CD[i],          ↵
                Ikhana_fo.T[i], Ikhana_fo.esfc[i], Ikhana_fo.W[i]))
            sdist_file.write('\n')

            if np.isnan(Ikhana_fo.W[0]-Ikhana_fo.W[-1]):
                continue

        sdata_file.close()
        sdist_file.close()

    def weight(Ikhana_fo, Ikhana_b, sdata_file, sdist_file,                ↵
    sdata_format, sdist_format, sdist_header):
        W0t = Ikhana_fo.iweight
        W0f = Ikhana_fo.fweight
        CD2p = 0.017
        CD00 = Ikhana_fo.CD0
        CD10 = Ikhana_fo.CD1
        Ws0 = 1008.0
        W_S = 31.874
        b = 66.20743376
        S0 = Ikhana_fo.Sw
        Ws =                                                               ↵
        [675.00,700.00,725.00,750.00,775.00,800.00,825.00,850.00,875.00,90 ↵
        0.00,925.00,950.00,975.00,1008.00]
        # ~ Ws = [936.19]
        B3 =                                                               ↵
        [-0.338444,-0.313888,-0.289193,-0.264360,-0.239388,-0.214278,-0.18 ↵
        9030,-0.163643,-0.138119,-0.112455,-0.086654,-0.060714,-0.034635,0 ↵
        .0]
        # ~ B3 = [-0.07505]

        for k in range(0, len(Ws)):
            es = 1./(1.+3.*B3[k]*B3[k])
```

```
Ikhana_fo.iweight = W0t-(Ws0-Ws[k])
Ikhana_fo.fweight = W0f-(Ws0-Ws[k])
Ikhana_fo.Sw = Ikhana_fo.iweight/W_S
Ikhana_fo.RA = b**2/Ikhana_fo.Sw
Ikhana_fo.CD2 = CD2p*Ikhana_fo.Sw/S0+1./(ma.pi*Ikhana_fo.RA*es)
Ikhana_fo.CD0 = CD00*Ikhana_fo.Sw/S0
Ikhana_fo.CD1 = CD10*Ikhana_fo.Sw/S0
Ikhana_fo.e = 1./(ma.pi*Ikhana_fo.RA*Ikhana_fo.CD2)
Ikhana_fo.s_runtime = time.time()
Ikhana_fo.trajectory_cases()
Ikhana_fo.f_runtime = time.time()

Ikhana_b.iweight -= (Ws0-Ws[k])
Ikhana_b.fweight -= (Ws0-Ws[k])
Ikhana_b.Sw = Ikhana_b.iweight/W_S
Ikhana_b.RA = b**2/Ikhana_b.Sw
Ikhana_b.CD2 = CD2p*Ikhana_b.Sw/S0+1./(ma.pi*Ikhana_b.RA*es)
Ikhana_b.e = 1./(ma.pi*Ikhana_b.RA*Ikhana_b.CD2)
Ikhana_b.s_runtime = time.time()
Ikhana_b.trajectory_cases()
Ikhana_b.f_runtime = time.time()

sdata_file.write(sdata_format.format(Ikhana_fo.RA,                ↵
Ikhana_fo.e, 1./(ma.pi*Ikhana_fo.RA*Ikhana_fo.e),                 ↵
Ikhana_fo.W[0]-Ikhana_fo.W[-1],                                   ↵
Ikhana_b.W[0]-Ikhana_b.W[-1], Ikhana_fo.t[-1]/3600.,              ↵
Ikhana_b.tb/3600., np.mean(Ikhana_fo.h),                          ↵
np.mean(Ikhana_fo.V),                                             ↵
np.mean(np.divide(Ikhana_fo.V,Ikhana_fo.a)),                      ↵
np.mean(Ikhana_fo.CL), np.mean(Ikhana_fo.CD),                     ↵
np.mean(Ikhana_fo.L_D), np.mean(Ikhana_fo.T),                     ↵
np.mean(Ikhana_fo.esfc)))

sdist_file.write(str(Ikhana_fo.RA)+'\t'+str(Ikhana_fo.e)+'\n')
sdist_file.write(sdist_header.format('x[ft]',                     ↵
'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/D',' ↵
T[lbf]','ESFC[slugs/lbf ft/s /s]','W[lbf]'))
for i in range(0, len(Ikhana_fo.x)):
    sdist_file.write(sdist_format.format(Ikhana_fo.x[i],          ↵
    Ikhana_fo.x[i]/5280., Ikhana_fo.t[i],                         ↵
    Ikhana_fo.t[i]/3600.0, Ikhana_fo.h[i], Ikhana_fo.V[i],        ↵
    Ikhana_fo.V[i]/Ikhana_fo.a[i], Ikhana_fo.CL[i],               ↵
    Ikhana_fo.CD[i], Ikhana_fo.CL[i]/Ikhana_fo.CD[i],             ↵
    Ikhana_fo.T[i], Ikhana_fo.esfc[i], Ikhana_fo.W[i]))
sdist_file.write('\n')

if np.isnan(Ikhana_fo.W[0]-Ikhana_fo.W[-1]):
    continue
```

```
    sdata_file.close()
    sdist_file.close()

def morphing(Ikhana_fo, Ikhana_b, sdata_file, sdist_file,
sdata_format, sdist_format, sdist_header):

    CD2p = 0.017
    B3 =
    [-0.333333333,-0.3,-0.275,-0.25,-0.225,-0.2,-0.175,-0.15,-0.125,-0
    .1,-0.075,-0.05,-0.025,0.]
    b =
    [76.47728384,75.14525373,74.20467605,73.30948942,72.45599409,71.64
    090659,70.86130084,70.11455917,69.39833135,68.71050012,68.04915205
    ,67.41255273,66.7991256,66.20743376]
    for k in range(0, len(B3)):

        es = 1.0

        Ikhana_fo.RA = b[k]**2/Ikhana_fo.Sw
        Ikhana_fo.CD2 = CD2p+1./(ma.pi*Ikhana_fo.RA*es)
        Ikhana_fo.e = 1./(ma.pi*Ikhana_fo.RA*Ikhana_fo.CD2)
        Ikhana_fo.s_runtime = time.time()
        Ikhana_fo.trajectory_cases()
        Ikhana_fo.f_runtime = time.time()

        Ikhana_b.RA = b[k]**2/Ikhana_b.Sw
        Ikhana_b.CD2 = CD2p+1./(ma.pi*Ikhana_b.RA*es)
        Ikhana_b.e = 1./(ma.pi*Ikhana_b.RA*Ikhana_b.CD2)
        Ikhana_b.s_runtime = time.time()
        Ikhana_b.trajectory_cases()
        Ikhana_b.f_runtime = time.time()

        sdata_file.write(sdata_format.format(Ikhana_fo.RA,
        Ikhana_fo.e, 1./(ma.pi*Ikhana_fo.RA*Ikhana_fo.e),
        Ikhana_fo.W[0]-Ikhana_fo.W[-1],
        Ikhana_b.W[0]-Ikhana_b.W[-1], Ikhana_fo.t[-1]/3600.,
        Ikhana_b.tb/3600., np.mean(Ikhana_fo.h),
        np.mean(Ikhana_fo.V),
        np.mean(np.divide(Ikhana_fo.V,Ikhana_fo.a)),
        np.mean(Ikhana_fo.CL), np.mean(Ikhana_fo.CD),
        np.mean(Ikhana_fo.L_D), np.mean(Ikhana_fo.T),
        np.mean(Ikhana_fo.esfc)))

        sdist_file.write(str(Ikhana_fo.RA)+'\t'+str(Ikhana_fo.e)+'\n')
        sdist_file.write(sdist_header.format('x[ft]',
        'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/D','
        T[lbf]','ESFC[slugs/lbf ft/s /s]','W[lbf]'))
        for i in range(0, len(Ikhana_fo.x)):
            sdist_file.write(sdist_format.format(Ikhana_fo.x[i],
```

```
            Ikhana_fo.x[i]/5280., Ikhana_fo.t[i],
            Ikhana_fo.t[i]/3600.0, Ikhana_fo.h[i], Ikhana_fo.V[i],
            Ikhana_fo.V[i]/Ikhana_fo.a[i], Ikhana_fo.CL[i],
            Ikhana_fo.CD[i], Ikhana_fo.CL[i]/Ikhana_fo.CD[i],
            Ikhana_fo.T[i], Ikhana_fo.esfc[i], Ikhana_fo.W[i]))
        sdist_file.write('\n')

        if np.isnan(Ikhana_fo.W[0]-Ikhana_fo.W[-1]):
            continue

    sdata_file.close()
    sdist_file.close()

def contour(Ikhana_fo, Ikhana_b, sdata_file, sdist_file,
sdata_format, sdist_format, sdist_header, runtype):
    W0t = Ikhana_fo.iweight
    W0f = Ikhana_fo.fweight
    CD2p = 0.017
    Ws0 = 1008.0
    W_S = 31.874
    S0 = Ikhana_fo.Sw
    B3 =
    [-0.333333333,-0.3,-0.275,-0.25,-0.225,-0.2,-0.175,-0.15,-0.125,-0
    .1,-0.075,-0.05,-0.025,0]
    b =
    [50.0,52.0,54.0,56.0,58.0,60.00,62.00,64.00,66.00,68.00,70.00,72.0
    0,74.00,76.00,78.00,80.00,82.,84.,86.,88.,90.,92.,94.,96.,98.,100.
    ]
    Ws = np.zeros((len(B3), len(b)),)
    S = np.zeros((len(B3), len(b)),)
    e = np.zeros((len(B3), len(b)),)
    RA = np.zeros((len(B3), len(b)),)
    Wf = np.zeros((len(B3), len(b)),)
    ctime = np.zeros((len(B3), len(b)),)
    havg = np.zeros((len(B3), len(b)),)
    Vavg = np.zeros((len(B3), len(b)),)
    Mavg = np.zeros((len(B3), len(b)),)
    CLavg = np.zeros((len(B3), len(b)),)
    CDavg = np.zeros((len(B3), len(b)),)
    L_Davg = np.zeros((len(B3), len(b)),)
    Tavg = np.zeros((len(B3), len(b)),)
    esfcavg = np.zeros((len(B3), len(b)),)
    CD2 = np.zeros((len(B3), len(b)),)

    filename = 'Ikhana_Ws_range'

    outname = 'Ikhana_Ws_range'

    plane=ws.Domain(filename+'.json',True)
```

```
for k in range(0, len(B3)):
    for l in range(0, len(b)):

        plane.wing.lift_dist_coeffs[1] = B3[k]
        plane.wing.span = b[l]

        plane.wing.discretize()
        plane.wing.calculate_lift_distribution()
        plane.set_distributions(filename+'.json')

        plane.solver(1e-9, False)

        Ws[k,l] = plane.weight.wing_structure
        # ~ print(Ws[k,l])


        if runtype == 'contour_m':
            es = 1.
        else:
            es = 1./(1.+3.*B3[k]*B3[k])

        Ikhana_fo.iweight = W0t-(Ws0-Ws[k,l])
        Ikhana_fo.fweight = W0f-(Ws0-Ws[k,l])
        Ikhana_fo.Sw = Ikhana_fo.iweight/W_S
        Ikhana_fo.CD2 =
        CD2p*Ikhana_fo.Sw/S0+1./(ma.pi*Ikhana_fo.RA*es)
        Ikhana_fo.e = 1./(ma.pi*Ikhana_fo.RA*Ikhana_fo.CD2)
        Ikhana_fo.s_runtime = time.time()
        Ikhana_fo.trajectory_cases()
        Ikhana_fo.f_runtime = time.time()
        Ikhana_fo.RA = b[l]**2/(Ikhana_fo.Sw)

        # ~ CRM_b.iweight -= (Ws0-Ws[k])
        # ~ CRM_b.fweight -= (Ws0-Ws[k])
        # ~ CRM_b.Sw = CRM_b.iweight/W_S
        # ~ CRM_b.CD2 = CD2p*CRM_b.Sw/S0+1./(ma.pi*CRM_b.RA*es)
        # ~ CRM_b.e = 1./(ma.pi*CRM_b.RA*CRM_b.CD2)
        # ~ CRM_b.s_runtime = time.time()
        # ~ CRM_b.trajectory_cases()
        # ~ CRM_b.f_runtime = time.time()

        S[k,l] = Ikhana_fo.Sw
        e[k,l] = Ikhana_fo.e
        RA[k,l] = Ikhana_fo.RA
        Wf[k,l] = Ikhana_fo.W[0]-Ikhana_fo.W[-1]
        ctime[k,l] = Ikhana_fo.t[-1]/3600.
        havg[k,l] = np.mean(Ikhana_fo.h)
        Vavg[k,l] = np.mean(Ikhana_fo.V)
```

```
            Mavg[k,l] = np.mean(np.divide(Ikhana_fo.V,Ikhana_fo.a))
            CLavg[k,l] = np.mean(Ikhana_fo.CL)
            CDavg[k,l] = np.mean(Ikhana_fo.CD)
            L_Davg[k,l] = np.mean(Ikhana_fo.L_D)
            Tavg[k,l] = np.mean(Ikhana_fo.T)
            esfcavg[k,l] = np.mean(Ikhana_fo.esfc)
            CD2[k,l] = Ikhana_fo.CD2
            # ~ sdata_file.write(sdata_format.format(Ikhana_fo.RA,
            Ikhana_fo.e, 1./(ma.pi*Ikhana_fo.RA*Ikhana_fo.e),
            Ikhana_fo.W[0]-Ikhana_fo.W[-1], CRM_b.W[0]-CRM_b.W[-1],
            Ikhana_fo.t[-1]/3600., CRM_b.tb/3600.,
            np.mean(Ikhana_fo.h), np.mean(Ikhana_fo.V),
            np.mean(np.divide(Ikhana_fo.V,Ikhana_fo.a)),
            np.mean(Ikhana_fo.CL), np.mean(Ikhana_fo.CD),
            np.mean(Ikhana_fo.L_D), np.mean(Ikhana_fo.T),
            np.mean(Ikhana_fo.c)))

            sdist_file.write(str(B3[k])+'\t'+str(b[l])+'\n')
            sdist_file.write(sdist_header.format('x[ft]',
            'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/
            D','T[lbf]','ESFC[slugs/lbf ft/s /s]','W[lbf]'))
            for i in range(0, len(Ikhana_fo.x)):
                sdist_file.write(sdist_format.format(Ikhana_fo.x[i],
                Ikhana_fo.x[i]/5280., Ikhana_fo.t[i],
                Ikhana_fo.t[i]/3600.0, Ikhana_fo.h[i],
                Ikhana_fo.V[i], Ikhana_fo.V[i]/Ikhana_fo.a[i],
                Ikhana_fo.CL[i], Ikhana_fo.CD[i],
                Ikhana_fo.CL[i]/Ikhana_fo.CD[i], Ikhana_fo.T[i],
                Ikhana_fo.esfc[i], Ikhana_fo.W[i]))
            sdist_file.write('\n')

            if np.isnan(Ikhana_fo.W[0]-Ikhana_fo.W[-1]):
                continue


            # ~ fig1.gca().plot(Ikhana_fo.xm, Ikhana_fo.h)
            # ~ fig2.gca().plot(Ikhana_fo.xm, Ikhana_fo.V)
            # ~ fig3.gca().plot(Ikhana_fo.xm,
            np.divide(Ikhana_fo.V,Ikhana_fo.a))
            # ~ fig4.gca().plot(Ikhana_fo.xm, Ikhana_fo.CL)
            # ~ fig5.gca().plot(Ikhana_fo.xm, Ikhana_fo.CD)
            # ~ fig6.gca().plot(Ikhana_fo.xm, Ikhana_fo.L_D)
            # ~ fig7.gca().plot(Ikhana_fo.xm, Ikhana_fo.T)
            # ~ fig8.gca().plot(Ikhana_fo.xm, Ikhana_fo.esfc)
            # ~ fig9.gca().plot(Ikhana_fo.xm, Ikhana_fo.W)

sdata_file = open(results_path+'sdata.txt', 'w')
sdata_file.write('Wing-Structure Weight'+'\n'+',')
for j in range(0, len(b)):
```

```
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(Ws[i,j])+',')

sdata_file.write('\n'+'\n'+'Aspect Ratio'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(RA[i,j])+',')

sdata_file.write('\n'+'\n'+'Oswald efficiency                    ↵
Factor'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(e[i,j])+',')

sdata_file.write('\n'+'\n'+'CD2'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(CD2[i,j])+',')

sdata_file.write('\n'+'\n'+'Fuel Burn'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(Wf[i,j])+',')

sdata_file.write('\n'+'\n'+'Cruise Time'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(ctime[i,j])+',')

sdata_file.write('\n'+'\n'+'average altitude'+'\n'+',')
for j in range(0, len(b)):
```

```
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(havg[i,j])+',')

sdata_file.write('\n'+'\n'+'average velocity'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(Vavg[i,j])+',')

sdata_file.write('\n'+'\n'+'average Mach'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(Mavg[i,j])+',')

sdata_file.write('\n'+'\n'+'average CL'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(CLavg[i,j])+',')

sdata_file.write('\n'+'\n'+'average CD'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(CDavg[i,j])+',')

sdata_file.write('\n'+'\n'+'average L/D'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(L_Davg[i,j])+',')

sdata_file.write('\n'+'\n'+'average Thrust'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
```

```
    for i in range(0, len(B3)):
        sdata_file.write('\n'+str(B3[i])+',')
        for j in range(0, len(b)):
            sdata_file.write(str(Tavg[i,j])+',')

    sdata_file.write('\n'+'\n'+'average ESFC'+'\n'+',')
    for j in range(0, len(b)):
        sdata_file.write(str(b[j])+',')
    for i in range(0, len(B3)):
        sdata_file.write('\n'+str(B3[i])+',')
        for j in range(0, len(b)):
            sdata_file.write(str(esfcavg[i,j])+',')

    sdata_file.close()
    sdist_file.close()

def single(Ikhana_fo, Ikhana_b, sdata_file, sdist_file,
sdata_format, sdist_format, sdist_header):

    b = 66.20743376

    Ikhana_fo.RA = b**2/Ikhana_fo.Sw
    Ikhana_fo.e = 1./(ma.pi*Ikhana_fo.RA*Ikhana_fo.CD2)
    Ikhana_fo.s_runtime = time.time()
    Ikhana_fo.trajectory_cases()
    Ikhana_fo.f_runtime = time.time()

    Ikhana_b.RA = b**2/Ikhana_b.Sw
    Ikhana_b.e = 1./(ma.pi*Ikhana_b.RA*Ikhana_b.CD2)
    Ikhana_b.s_runtime = time.time()
    Ikhana_b.trajectory_cases()
    Ikhana_b.f_runtime = time.time()


    sdata_file.write(sdata_format.format(Ikhana_fo.RA, Ikhana_fo.e,
    1./(ma.pi*Ikhana_fo.RA*Ikhana_fo.e),
    Ikhana_fo.W[0]-Ikhana_fo.W[-1], Ikhana_b.W[0]-Ikhana_b.W[-1],
    Ikhana_fo.t[-1]/3600., Ikhana_b.tb/3600., np.mean(Ikhana_fo.h),
    np.mean(Ikhana_fo.V),
    np.mean(np.divide(Ikhana_fo.V,Ikhana_fo.a)),
    np.mean(Ikhana_fo.CL), np.mean(Ikhana_fo.CD),
    np.mean(Ikhana_fo.L_D), np.mean(Ikhana_fo.T),
    np.mean(Ikhana_fo.esfc)))

    sdist_file.write(str(Ikhana_fo.RA)+'\t'+str(Ikhana_fo.e)+'\n')
    sdist_file.write(sdist_header.format('x[ft]',
    'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/D','T[lb
    f]','ESFC[slugs/lbf ft/s /s]','W[lbf]'))
    for i in range(0, len(Ikhana_fo.x)):
```

```
            sdist_file.write(sdist_format.format(Ikhana_fo.x[i],
            Ikhana_fo.x[i]/5280., Ikhana_fo.t[i], Ikhana_fo.t[i]/3600.0,
            Ikhana_fo.h[i], Ikhana_fo.V[i],
            Ikhana_fo.V[i]/Ikhana_fo.a[i], Ikhana_fo.CL[i],
            Ikhana_fo.CD[i], Ikhana_fo.CL[i]/Ikhana_fo.CD[i],
            Ikhana_fo.T[i], Ikhana_fo.esfc[i], Ikhana_fo.W[i]))
        sdist_file.write('\n')

        sdata_file.close()
        sdist_file.close()

    def run(runtype, runfile):
        # general
        # wingspan
        # weight
        # morphing
        # general_h
        # wingspan_h
        # weight_h
        # morphing_h
        # contour
        # contour_m
        if runtype == 'single':
            results_path = './results/constrained/Ikhana_range/'+runfile
        else:
            results_path = './results/constrained/Ikhana_range/'+runtype

        sdata_header="{0:<30}{1:<30}{2:<30}{3:<30}{4:<30}{5:<30}{6:<30}{7:
        <30}{8:<30}{9:<30}{10:<30}{11:<30}{12:<30}{13:<30}{14:<30}\n"

        sdata_format="{0:<30.16f}{1:<30.16f}{2:<30.16f}{3:<30.16f}{4:<30.1
        6f}{5:<30.16f}{6:<30.16f}{7:<30.16f}{8:<30.16f}{9:<30.16f}{10:<30.
        16f}{11:<30.16f}{12:<30.16f}{13:<30.16f}{14:<30.16f}\n"

        sdist_header="{0:<30}{1:<30}{2:<30}{3:<30}{4:<32}{5:<30}{6:<30}{7:
        <32}{8:<30}{9:<30}{10:<32}{11:<30}{12:<30}\n"

        sdist_format="{0:<30.16f}{1:<30.16f}{2:<30.16f}{3:<30.16f}{4:<32}{
        5:<30.16f}{6:<30.16f}{7:<32}{8:<30.16f}{9:<30.16f}{10:<32}{11:<30.
        16f}{12:<30.16f}\n"

        if runtype!='contour' or runtype!='contour_m':
            sdata_file = open(results_path+'/sdata.txt', 'w')
            sdata_file.write(sdata_header.format('Aspect Ratio', 'oswald
            efficiency', 'CD2', 'Fuel Burn [lbf]', 'Fuel Burn (breguet)
            [lbf]', 'time [hr]', 'time (breguet) [hr]', 'average
            altitude [ft]', 'average velocity [ft/s]', 'average Mach',
            'average CL', 'average CD', 'average L/D', 'average T
            [lbf]', 'average ESFC [slugs/lbf ft/s /s]'))
```

```
sdist_file = open(results_path+'/sdists.txt', 'w')

if runtype[-1] == 'h':
    Ikhana_fo = Aircraft_h(runfile+'.json')
    Ikhana_b = Aircraft_h(runfile+'.json')
elif runtype[-1] == 'f':
    Ikhana_fo = Aircraft_fixed_h(runfile+'.json')
    Ikhana_b = Aircraft_fixed_h(runfile+'.json')
else:
    Ikhana_fo = Aircraft(runfile+'.json')
    Ikhana_b = Aircraft(runfile+'.json')

Ikhana_fo.run_type = 'full_opt'
Ikhana_b.run_type = 'breguet'

Ikhana_fo.results_path = results_path
Ikhana_b.results_path = results_path

# ~ fig1=plt.figure(1)
# ~ fig1.gca().set_xlabel('x [mi]')
# ~ fig1.gca().set_ylabel('h [ft]')

# ~ fig2=plt.figure(2)
# ~ fig2.gca().set_xlabel('x [mi]')
# ~ fig2.gca().set_ylabel('V [ft/s]')

# ~ fig3=plt.figure(3)
# ~ fig3.gca().set_xlabel('x [mi]')
# ~ fig3.gca().set_ylabel('M')

# ~ fig4=plt.figure(4)
# ~ fig4.gca().set_xlabel('x [mi]')
# ~ fig4.gca().set_ylabel('CL')

# ~ fig5=plt.figure(5)
# ~ fig5.gca().set_xlabel('x [mi]')
# ~ fig5.gca().set_ylabel('CD')

# ~ fig6=plt.figure(6)
# ~ fig6.gca().set_xlabel('x [mi]')
# ~ fig6.gca().set_ylabel('L/D')

# ~ fig7=plt.figure(7)
# ~ fig7.gca().set_xlabel('x [mi]')
# ~ fig7.gca().set_ylabel('T [lbf]')

# ~ fig8=plt.figure(8)
# ~ fig8.gca().set_xlabel('x [mi]')
```

```
# ~ fig8.gca().set_ylabel('ESFC [slugs/lbf ft/s /s]')

# ~ fig9=plt.figure(9)
# ~ fig9.gca().set_xlabel('x [mi]')
# ~ fig9.gca().set_ylabel('W [lbf]')
if runtype == 'general' or runtype == 'general_h' or runtype ==
'general_hf':
    general(Ikhana_fo, Ikhana_b, sdata_file, sdist_file,
    sdata_format, sdist_format, sdist_header)
if runtype == 'wingspan' or runtype == 'wingspan_h' or runtype
== 'wingspan_hf':
    wingspan(Ikhana_fo, Ikhana_b, sdata_file, sdist_file,
    sdata_format, sdist_format, sdist_header)
if runtype == 'weight' or runtype == 'weight_h' or runtype ==
'weight_hf':
    weight(Ikhana_fo, Ikhana_b, sdata_file, sdist_file,
    sdata_format, sdist_format, sdist_header)
if runtype == 'morphing' or runtype == 'morphing_h' or runtype
== 'morphing_hf':
    morphing(Ikhana_fo, Ikhana_b, sdata_file, sdist_file,
    sdata_format, sdist_format, sdist_header)
if runtype == 'contour' or runtype == 'contour_m':
    contour(Ikhana_fo, Ikhana_b, sdata_file, sdist_file,
    sdata_format, sdist_format, sdist_header, runtype)
if runtype == 'single' or runtype == 'single_hf':
    single(Ikhana_fo, Ikhana_b, sdata_file, sdist_file,
    sdata_format, sdist_format, sdist_header)

# ~ fig1.savefig(results_path+'/h.png')
# ~ fig2.savefig(results_path+'/V.png')
# ~ fig3.savefig(results_path+'/M.png')
# ~ fig4.savefig(results_path+'/CL.png')
# ~ fig5.savefig(results_path+'/CD.png')
# ~ fig6.savefig(results_path+'/L_D.png')
# ~ fig7.savefig(results_path+'/T.png')
# ~ fig8.savefig(results_path+'/ESFC.png')
# ~ fig9.savefig(results_path+'/W.png')
# ~ plt.show()
        # ~ fig1.gca().plot(Ikhana_fo.xm, Ikhana_fo.h)
        # ~ fig2.gca().plot(Ikhana_fo.xm, Ikhana_fo.V)
        # ~ fig3.gca().plot(Ikhana_fo.xm,
        np.divide(Ikhana_fo.V,Ikhana_fo.a))
        # ~ fig4.gca().plot(Ikhana_fo.xm, Ikhana_fo.CL)
        # ~ fig5.gca().plot(Ikhana_fo.xm, Ikhana_fo.CD)
        # ~ fig6.gca().plot(Ikhana_fo.xm, Ikhana_fo.L_D)
        # ~ fig7.gca().plot(Ikhana_fo.xm, Ikhana_fo.T)
        # ~ fig8.gca().plot(Ikhana_fo.xm, Ikhana_fo.esfc)
        # ~ fig9.gca().plot(Ikhana_fo.xm, Ikhana_fo.W)
```

C.2.2 Trajectory Optimization for the CRM (python)

```
'''
CRM_range_master.py

The code in this file contains performs trajectory optimization for
aircraft having the engine model for thrust-specific fuel consumption
given by Eshelby in "Aircraft Performance: Theory and Practice" and the
power available model given by Daidzic in "Estimation of Performance
Airspeeds for High-Bypass Turbofans Equipped Transport-Category
Airplanes."

'''

import math as ma
import numpy as np
import scipy.integrate as integrate
import scipy.optimize as optimize
import scipy.interpolate as interpolate
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import standard_atmosphere as std
import json
from collections import OrderedDict
import time
from progress.bar import IncrementalBar
import multiprocessing
from itertools import repeat
from functools import partial
import sys
sys.path.append('./CRM_structure')
import  wing_structure_m3 as ws
sys.path.append('..')


class Aircraft(object):

    def __init__(self, filename):
        with open(filename) as input_file:
            data = json.load(input_file, object_pairs_hook=OrderedDict)

        self.input_airplane(data)
        self.grid_setup()
        self.array_initialize()
```

```
def input_airplane(self, data):

    #Units
    if data["units"] == 'English':
        self.english_units = True
    else:
        self.english_units = False

    # CRM Constant Properties
    self.iweight = data["MTOW"]
    self.fweight = data["M15FW"]
    self.Sw = data["wing_area"]
    self.CD0 = data["CD0"]
    self.CD1 = data["CD1"]
    self.CD2 = data["CD2"]
    self.CM0 = data["CM0"]
    self.CM1 = data["CM1"]
    self.CM2 = data["CM2"]
    self.Ctsfc = data["CTSFC"]
    self.r = data["range"]*5280.0
    self.q = data["qTSFC"]
    self.Ts = data["static_thrust_SL"]
    self.a1 = data["a1"]
    self.a2 = data["a2"]
    self.n = data["grid"]
    self.g = 32.174
    self.name = data["name"]
    self.run_type = 'none'
    self.run_direction = data["direction"]
    self.s_runtime = 0.0
    self.f_runtime = 0.0
    self.results_path = '.'
    self.RA = 9.
    self.e = 1.

def grid_setup(self):
    self.x = np.linspace(0, self.r, self.n)
    self.xm = self.x/5280.0

def array_initialize(self):
    self.W = np.zeros(self.n)
    if self.run_direction == "f":
        self.W[0] = self.iweight
    elif self.run_direction == "b":
        self.W[-1] = self.fweight
    self.CL = np.zeros(self.n)
    self.CD = np.zeros(self.n)
    self.V = np.zeros(self.n)
    self.h = np.zeros(self.n)
```

```
        self.t = np.zeros(self.n)
        self.c = np.zeros(self.n)
        self.rho = np.zeros(self.n)
        self.a = np.zeros(self.n)
        self.Temp = np.zeros(self.n)
        self.T = np.zeros(self.n)

    def opt_h_V(self, y, i):
        h = y[0]*100.
        V = y[1]
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)
        # ~ print(Temp, TSL)

        #Lift coefficient
        CL = self.W[i]/(0.5*rho*V**2*self.Sw)

        #Mach number
        M = V/a

        #Drag coefficient
        CD =
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
        lf.CM2)

        #Thrust
        T = 0.5*rho*V**2*self.Sw*CD

        #Power Required
        PR = T*V

        #Power Available
        if h <36131:
            m=0.7
        else:
            m=1.0

        PA =
        2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V)
        # ~ print(i, self.W[i])
```

```
        climb = (PA-PR)/self.W[i]*60

        #TSFC
        c = self.Ctsfc*(Temp/TSL)**0.5*M**self.q

        if i==self.n-1:
            dx = self.x[i]-self.x[i-1]
        else:
            dx = self.x[i+1]-self.x[i]

        t = dx/V


        if climb>100.:
            return c*T*self.g*t
        else:
            return c*T*self.g*t+10*(climb-100)**2

    def set_state(self, y, i):
        h = y[0]*100.
        V = y[1]
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)

        #Lift coefficient
        CL = self.W[i]/(0.5*rho*V**2*self.Sw)

        #Mach number
        M = V/a

        #Drag coefficient
        CD =
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
        lf.CM2)

        #Thrust
        T = 0.5*rho*V**2*self.Sw*CD

        #TSFC
```

```
        c = self.Ctsfc*(Temp/TSL)**0.5*M**self.q

        self.h[i] = h
        self.V[i] = V
        self.rho[i] = rho
        self.a[i] = a
        self.Temp[i] = Temp
        self.CL[i] = CL
        self.CD[i] = CD
        self.T[i] = T
        self.c[i] = c

    def breguet(self, y, flag='obj'):
        h = y[0]
        V = y[1]
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)
        # ~ print(Temp, TSL)

        #Lift coefficient
        if self.run_type == 'f':
            CL = self.W[0]/(0.5*rho*V**2*self.Sw)
        else:
            CL = self.W[-1]/(0.5*rho*V**2*self.Sw)

        #Mach number
        M = V/a

        #Drag coefficient
        CD =
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
        lf.CM2)


        #TSFC
        c = self.Ctsfc*(Temp/TSL)**0.5*M**self.q


        if self.run_direction == 'f':
            Wf = self.W[0]/np.exp(self.r*self.g*c/(V*CL/CD))
```

```
            self.hb = h
            self.Vb = V
            self.L_Db = CL/CD
            self.c = c
            self.Wfb = Wf
            self.W[-1] = self.Wfb
            self.Mb = M
            self.tb = self.r/V

            #Power Req.
            PR = 0.5*rho*V**3*self.Sw*CD

            #Power Available
            if h <36131:
                m=0.7
            else:
                m=1.0
            PA =                                                    ↵
            2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V ↵
            )

            climb = (PA-PR)/self.W[0]*60

            if climb>100.:
                return self.W[0]-Wf
            else:
                return (self.W[0]-Wf)+10*(climb-100)**2

        elif self.run_direction == 'b':
            # ~ print(V, CL, CD)
            Wi = self.W[-1]*np.exp(self.r*self.g*c/(V*CL/CD))

            self.hb = h
            self.Vb = V
            self.L_Db = CL/CD
            self.cb = c
            self.iweight = Wi
            self.W[0] = self.iweight
            self.Mb = M
            self.tb = self.r/V

            #Power Req.
            PR = 0.5*rho*V**3*self.Sw*CD

            #Power Available
            if h <36131:
                m=0.7
            else:
```

```
            m=1.0
        PA = 2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V
        )

        climb = (PA-PR)/self.W[-1]*60

        if climb>100.:
            return Wi-self.W[-1]
        else:
            return (Wi-self.W[-1])+10*(climb-100)**2

def breguet_opt(self):
    y0 = [30000., 800.0]
    bnds = ((100., 80000.), (135.0, 1200.))
    cons = [{"type" : "ineq",
             "fun" : self.breguet}]
    ans = optimize.minimize(self.breguet,
                            y0,
                            method='SLSQP',
                            bounds = bnds,
                            options={'disp' : False,
                                     'ftol' : 1e-16})

def service_ceiling(self, y, i):
    # ~ print(y, i)
    h = y[0]

    #atmospheric properties
    if self.english_units == True:
        statmos = std.StandardAtmosphere('English')
    else:
        statmos = std.StandardAtmosphere('SI')

    rho = statmos.rho(h)
    a = statmos.a(h)
    Temp = statmos.T(h)
    TSL = statmos.T(0)
    rhoSL = statmos.rho(0)

    if self.run_type == 'Vmf':
        V = optimize.newton(self.Vmfp, 800.0, args=(rho, a, i))
    elif self.run_type == 'Vmd':
        V = optimize.newton(self.Vmd, 800.0, args=(rho, a, i))
    else:
        V = y[1]

    #Lift coefficient
    CL = self.W[i]/(0.5*rho*V**2*self.Sw)
```

```
        #Mach number
        M = V/a

        #Drag coefficient
        CD =
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
        lf.CM2)

        #Power Req.
        PR = 0.5*rho*V**3*self.Sw*CD

        #Power Available
        if h <36131:
            m=0.7
        else:
            m=1.0
        PA =
        2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V)

        climb = (PA-PR)/self.W[i]*60

        return climb-100.+400.

    def trajectory_opt(self):
        opts = {'disp' : False,
                'maxiter' : 500}#,
                # ~ 'eps' : 1e-10,
                # ~ 'ftol' : 1e-11}
        tolerance = 1e-14
        iguess = [350., 850.]

        if self.run_direction == 'b':

            if self.run_type == 'full_opt':
                #Full Optimization

                #-------------------------------------------------------
                ---------------------------------
                y0 = iguess
                bnds = ((1., 800.), (100.0, 1800.))
                # ~ cons = [{"type" : "ineq",
                            # ~ "fun" : self.service_ceiling}]
                for i in range(1, self.n+1):
                    j = self.n-i
                    ans = optimize.minimize(self.opt_h_V,
                                            y0, args=(j),
                                            method='SLSQP',
                                            bounds = bnds,
```

```
                                        tol = tolerance,
                                        options=opts)

                    self.set_state(ans.x, j)


                    if j>0:
                        self.W[j-1] =                                    ↵
                        self.W[j]+self.c[j]*self.T[j]*self.g*(self.x[j ↵
                        ]-self.x[j-1])/self.V[j]
                    if ans.success == False:
                        print('optimization failed', ans.message)
                        y0 = ans.x

                for i in range(0, self.n):
                    if i<self.n-1:
                        self.t[i+1] =                                    ↵
                        self.t[i]+(self.x[i+1]-self.x[i])/self.V[i]

        self.L = np.multiply(self.Sw*0.5*self.rho,                       ↵
        np.multiply(np.multiply(self.V, self.V), self.CL))
        self.D = np.multiply(self.Sw*0.5*self.rho,                       ↵
        np.multiply(np.multiply(self.V, self.V), self.CD))
        self.L_D = np.divide(self.L, self.D)

    def trajectory_cases(self):
        data_format="{0:<30}{1:<30.16f}\n"
                                                                         ↵
        dist_header="{0:<30}{1:<30}{2:<30}{3:<30}{4:<32}{5:<30}{6:<30} ↵
        {7:<32}{8:<30}{9:<30}{10:<32}{11:<30}{12:<30}\n"
                                                                         ↵
        dist_format="{0:<30.16f}{1:<30.16f}{2:<30.16f}{3:<30.16f}{4:<3 ↵
        2.16f}{5:<30.16f}{6:<30.16f}{7:<32.16f}{8:<30.16f}{9:<30.16f}{ ↵
        10:<30.16f}{11:<30.16f}{12:<30.16f}\n"

        self.fig_path = self.results_path+'figs/'
        if self.run_type == 'breguet':
            self.breguet_opt()
            print('Run Type: ', self.run_type)
            if self.run_direction == 'f':
                print('Total Fuel Burn: ', self.W[0]-self.Wfb, 'lbf')
            else:
                print('Total Fuel Burn: ', self.iweight-self.W[-1],    ↵
                'lbf')
            print('Total Time: ', self.tb/3600.0, 'hours')
            print('L/D: ', self.L_Db)
            print('V: ', self.Vb, 'ft/s')
            print('h: ', self.hb, 'ft')
            print('TSFC: ', self.cb, 'slugs/lbf/s')
```

```
        print('Mach: ', self.Mb)
        print('total run time: ', time.time()-self.s_runtime)
                                                                    ⤶
        print('------------------------------------------------- ⤶
        ')


    else:
        self.trajectory_opt()
        print('Run Type:', self.run_type)
        print('Total Fuel Burn: ', self.W[0]-self.W[-1], 'lbf')
        print('Total Time: ', self.t[-1]/3600.0, 'hours')
        print('total run time: ', time.time()-self.s_runtime)
                                                                    ⤶
        print('------------------------------------------------- ⤶
        ')

        '''
        data_file =                                                 ⤶
        open(self.results_path+str(self.RA)+'_'+str(self.e)+'_data ⤶
        .txt', 'w')
        data_file.write(data_format.format('Total Fuel Burn:',      ⤶
        self.W[0]-self.W[-1]))
        data_file.write(data_format.format('Total Cruise Time:',    ⤶
        self.t[-1]/3600.0))
        data_file.write(data_format.format('Run Time:',             ⤶
        time.time()-self.s_runtime))
        data_file.close()

        dist_file =                                                 ⤶
        open(self.results_path+str(self.RA)+'_'+str(self.e)+'_dist ⤶
        s.txt', 'w')
        dist_file.write(dist_header.format('x[ft]',                 ⤶
        'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/ ⤶
        D','T[lbf]','TSFC[slugs/lbf/s]','W[lbf]'))
        for i in range(0, len(self.x)):
            dist_file.write(dist_format.format(self.x[i],           ⤶
            self.x[i]/5280., self.t[i], self.t[i]/3600.0,           ⤶
            self.h[i], self.V[i], self.V[i]/self.a[i],              ⤶
            self.CL[i], self.CD[i], self.CL[i]/self.CD[i],          ⤶
            self.T[i], self.c[i], self.W[i]))
        dist_file.close()

        if np.isnan(self.W[0]-self.W[-1]):
            return
        plot_range_multiplier = 100.

        #Altitude
        plt.figure(1)
```

```
plt.plot(self.xm, self.h)
plt.xlabel('x [mi]')
plt.ylabel('h [ft]')
                                                                    ⇃
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_h ⇃
.png')

#Velocity
plt.figure(2)
plt.plot(self.xm, self.V)
plt.xlabel('x [mi]')
plt.ylabel('V [ft/s]')
                                                                    ⇃
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_V ⇃
.png')

#Mach
plt.figure(3)
plt.plot(self.xm, np.divide(self.V,self.a))
plt.xlabel('x [mi]')
plt.ylabel('M')
                                                                    ⇃
plt.ylim(np.amin(np.divide(self.V,self.a))-(np.amax(np.div ⇃
ide(self.V,self.a))-np.amin(np.divide(self.V,self.a)))*plo ⇃
t_range_multiplier,                                                 ⇃
np.amax(np.divide(self.V,self.a))+(np.amax(np.divide(self. ⇃
V,self.a))-np.amin(np.divide(self.V,self.a)))*plot_range_m ⇃
ultiplier)
                                                                    ⇃
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_M ⇃
.png')


#CL
plt.figure(4)
plt.plot(self.xm, self.CL)
plt.xlabel('x [mi]')
plt.ylabel('CL')
                                                                    ⇃
plt.ylim(np.amin(self.CL)-(np.amax(self.CL)-np.amin(self.C ⇃
L))*plot_range_multiplier,                                          ⇃
np.amax(self.CL)+(np.amax(self.CL)-np.amin(self.CL))*plot_ ⇃
range_multiplier)
                                                                    ⇃
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_C ⇃
L.png')

#CD
plt.figure(5)
```

```
plt.plot(self.xm, self.CD)
plt.xlabel('x [mi]')
plt.ylabel('CD')
                                                                    ⇃
plt.ylim(np.amin(self.CD)-(np.amax(self.CD)-np.amin(self.C        ⇃
D))*plot_range_multiplier,                                         ⇃
np.amax(self.CD)+(np.amax(self.CD)-np.amin(self.CD))*plot_         ⇃
range_multiplier)
                                                                   ⇃
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_C         ⇃
D.png')


#L/D
plt.figure(6)
plt.plot(self.xm, self.L_D)
plt.xlabel('x [mi]')
plt.ylabel('L/D')
                                                                   ⇃
plt.ylim(np.amin(self.L_D)-(np.amax(self.L_D)-np.amin(self         ⇃
.L_D))*plot_range_multiplier,                                      ⇃
np.amax(self.L_D)+(np.amax(self.L_D)-np.amin(self.L_D))*pl         ⇃
ot_range_multiplier)
                                                                   ⇃
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_L         ⇃
_D.png')


#T
plt.figure(7)
plt.plot(self.xm, self.T)
plt.xlabel('x [mi]')
plt.ylabel('T [lbf]')
                                                                   ⇃
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_T         ⇃
.png')

#TSFC
plt.figure(8)
plt.plot(self.xm, self.c)
plt.xlabel('x [mi]')
plt.ylabel('TSFC [slugs/lbf/s]')
                                                                   ⇃
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_T         ⇃
SFC.png')

#W
plt.figure(9)
plt.plot(self.xm, self.W)
```

```
            plt.xlabel('x [mi]')
            plt.ylabel('W [lbf]')

            plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_W
            .png')
            # ~ plt.show()

            plt.figure(1).clear()
            plt.figure(2).clear()
            plt.figure(3).clear()
            plt.figure(4).clear()
            plt.figure(5).clear()
            plt.figure(6).clear()
            plt.figure(7).clear()
            plt.figure(8).clear()
            plt.figure(9).clear()
            '''

class Aircraft_h(object):

    def __init__(self, filename):
        with open(filename) as input_file:
            data = json.load(input_file, object_pairs_hook=OrderedDict)

        self.input_airplane(data)
        self.grid_setup()
        self.array_initialize()

    def input_airplane(self, data):

        #Units
        if data["units"] == 'English':
            self.english_units = True
        else:
            self.english_units = False

        # CRM Constant Properties
        self.iweight = data["MTOW"]
        self.fweight = data["M15FW"]
        self.Sw = data["wing_area"]
        self.CD0 = data["CD0"]
        self.CD1 = data["CD1"]
        self.CD2 = data["CD2"]
        self.CM0 = data["CM0"]
        self.CM1 = data["CM1"]
        self.CM2 = data["CM2"]
        self.Ctsfc = data["CTSFC"]
        self.r = data["range"]*5280.0
        self.q = data["qTSFC"]
```

```
        self.Ts = data["static_thrust_SL"]
        self.a1 = data["a1"]
        self.a2 = data["a2"]
        self.n = data["grid"]
        self.g = 32.174
        self.name = data["name"]
        self.run_type = 'none'
        self.run_direction = data["direction"]
        self.s_runtime = 0.0
        self.f_runtime = 0.0
        self.results_path = '.'
        self.RA = 9.
        self.e = 1.
        self.hguess = 1.0

    def grid_setup(self):
        self.x = np.linspace(0, self.r, self.n)
        self.xm = self.x/5280.0

    def array_initialize(self):
        self.W = np.zeros(self.n)
        if self.run_direction == "f":
            self.W[0] = self.iweight
        elif self.run_direction == "b":
            self.W[-1] = self.fweight
        self.CL = np.zeros(self.n)
        self.CD = np.zeros(self.n)
        self.V = np.zeros(self.n)
        self.h = np.zeros(self.n)
        self.t = np.zeros(self.n)
        self.c = np.zeros(self.n)
        self.rho = np.zeros(self.n)
        self.a = np.zeros(self.n)
        self.Temp = np.zeros(self.n)
        self.T = np.zeros(self.n)
        self.climb = np.zeros(self.n)

    def opt_h_V(self, y, h, i):
        V = y
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
```

```
rhoSL = statmos.rho(0)
# ~ print(Temp, TSL)

#Lift coefficient
CL = self.W[i]/(0.5*rho*V**2*self.Sw)

#Mach number
M = V/a

#Drag coefficient
CD =
(self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
lf.CM2)

#Thrust
T = 0.5*rho*V**2*self.Sw*CD

#Power Required
PR = T*V

#Power Available
if h <36131:
    m=0.7
else:
    m=1.0

PA =
2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V)
# ~ print(i, self.W[i])

climb = (PA-PR)/self.W[i]*60


#TSFC
c = self.Ctsfc*(Temp/TSL)**0.5*M**self.q

if i==self.n-1:
    dx = self.x[i]-self.x[i-1]
else:
    dx = self.x[i+1]-self.x[i]

t = dx/V

# ~ print('fuel_burn', c*T*self.g*t)
# ~ print('velocity: ', V)
# ~ print('altitude: ', h)
# ~ print('Mach: ', M)
# ~ print('Drag Coefficient: ', CD)
# ~ print('Thrust: ', T)
```

```
        # ~ print('TSFC: ', c)
        # ~ print('Weight: ', self.W[i])
        # ~ print('-----------------------------------')
        # ~ return c*T*self.g*t
        if climb>100.:
            return c*T*self.g*t
        else:
            return c*T*self.g*t+10*(climb-100)**2

    def set_state(self, y, h, i):
        V = y
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)

        #Lift coefficient
        CL = self.W[i]/(0.5*rho*V**2*self.Sw)

        #Mach number
        M = V/a

        #Drag coefficient
        CD =
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
        lf.CM2)

        #Thrust
        T = 0.5*rho*V**2*self.Sw*CD

        #Power Required
        PR = T*V

        #Power Available
        if h <36131:
            m=0.7
        else:
            m=1.0

        PA =
        2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V)
        # ~ print(i, self.W[i])
```

```
        climb = (PA-PR)/self.W[i]*60
        #TSFC
        c = self.Ctsfc*(Temp/TSL)**0.5*M**self.q

        self.h[i] = h
        self.V[i] = V
        self.rho[i] = rho
        self.a[i] = a
        self.Temp[i] = Temp
        self.CL[i] = CL
        self.CD[i] = CD
        self.T[i] = T
        self.c[i] = c
        self.climb[i] = climb

    def breguet(self, y, flag='obj'):
        h = y[0]
        V = y[1]
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)
        # ~ print(Temp, TSL)

        #Lift coefficient
        if self.run_type == 'f':
            CL = self.W[0]/(0.5*rho*V**2*self.Sw)
        else:
            CL = self.W[-1]/(0.5*rho*V**2*self.Sw)

        #Mach number
        M = V/a

        #Drag coefficient
        CD =                                                          ↵
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se ↵
        lf.CM2)


        #TSFC
        c = self.Ctsfc*(Temp/TSL)**0.5*M**self.q
```

```
if self.run_direction == 'f':
    Wf = self.W[0]/np.exp(self.r*self.g*c/(V*CL/CD))

    self.hb = h
    self.Vb = V
    self.L_Db = CL/CD
    self.c = c
    self.Wfb = Wf
    self.W[-1] = self.Wfb
    self.Mb = M
    self.tb = self.r/V

    #Power Req.
    PR = 0.5*rho*V**3*self.Sw*CD

    #Power Available
    if h <36131:
        m=0.7
    else:
        m=1.0
    PA =
    2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V
    )

    climb = (PA-PR)/self.W[0]*60

    if climb>100.:
        return self.W[0]-Wf
    else:
        return (self.W[0]-Wf)+10*(climb-100)**2

elif self.run_direction == 'b':
    # ~ print(V, CL, CD)
    Wi = self.W[-1]*np.exp(self.r*self.g*c/(V*CL/CD))

    self.hb = h
    self.Vb = V
    self.L_Db = CL/CD
    self.cb = c
    self.iweight = Wi
    self.W[0] = self.iweight
    self.Mb = M
    self.tb = self.r/V

    #Power Req.
    PR = 0.5*rho*V**3*self.Sw*CD
```

```
            #Power Available
            if h <36131:
                m=0.7
            else:
                m=1.0
            PA =                                                    ⇄
            2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V ⇄
            )

            climb = (PA-PR)/self.W[-1]*60

            return Wi-self.W[-1]
            if climb>100.:
                return Wi-self.W[-1]
            else:
                return (Wi-self.W[-1])+10*(climb-100)**2

    def breguet_opt(self):
        y0 = [self.hguess, 800.0]
        bnds = ((100., 80000.), (135.0, 1200.))
        cons = [{"type" : "ineq",
                 "fun" : self.breguet}]
        ans = optimize.minimize(self.breguet,
                                y0,
                                method='SLSQP',
                                bounds = bnds,
                                # ~ constraints = [{"type" : "ineq",
                                            # ~ "fun" : self.breguet,
                                            # ~ "args" : ("cons",)}],
                                options={'disp' : False,
                                         'ftol' : 1e-16})

    def service_ceiling(self, y, i):
        # ~ print(y, i)
        h = y[0]

        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)

        if self.run_type == 'Vmf':
```

```
            V = optimize.newton(self.Vmfp, 800.0, args=(rho, a, i))
        elif self.run_type == 'Vmd':
            V = optimize.newton(self.Vmd, 800.0, args=(rho, a, i))
        else:
            V = y[1]

        #Lift coefficient
        CL = self.W[i]/(0.5*rho*V**2*self.Sw)

        #Mach number
        M = V/a

        #Drag coefficient
        CD =
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
        lf.CM2)

        #Power Req.
        PR = 0.5*rho*V**3*self.Sw*CD

        #Power Available
        if h <36131:
            m=0.7
        else:
            m=1.0
        PA =
        2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V)

        climb = (PA-PR)/self.W[i]*60

        return climb-100.+400.

    def trajectory_opt(self, r):
        h=r*10000.
        opts = {'disp' : False,
                'maxiter' : 500}#,
                # ~ 'eps' : 1e-10,
                # ~ 'ftol' : 1e-11}
        tolerance = 1e-14
        iguess = [ 850.]

        if self.run_direction == 'b':

            if self.run_type == 'full_opt':
                #Full Optimization

                #-------------------------------------------------
                ---------------------------------
                y0 = iguess
```

```
                bnds = ((100.0, 1800.),)
                for i in range(1, self.n+1):
                    j = self.n-i
                    ans = optimize.minimize(self.opt_h_V,
                                            y0, args=(h, j),
                                            method='SLSQP',
                                            bounds = bnds,
                                            tol = tolerance,
                                            options=opts)

                    self.set_state(ans.x[0], h, j)

                    if j>0:
                        self.W[j-1] =                                    ↵
                        self.W[j]+self.c[j]*self.T[j]*self.g*(self.x[j ↵
                        ]-self.x[j-1])/self.V[j]
                    if ans.success == False:
                        print('optimization failed', ans.message)
                        y0 = ans.x

                for i in range(0, self.n):
                    if i<self.n-1:
                        self.t[i+1] =                                    ↵
                        self.t[i]+(self.x[i+1]-self.x[i])/self.V[i]

            self.L = np.multiply(self.Sw*0.5*self.rho,                   ↵
            np.multiply(np.multiply(self.V, self.V), self.CL))
            self.D = np.multiply(self.Sw*0.5*self.rho,                   ↵
            np.multiply(np.multiply(self.V, self.V), self.CD))
            self.L_D = np.divide(self.L, self.D)

            print(' ')
            print('altitude: ', h)
            print(self.W[0]-self.W[-1])
            print(np.min(self.climb))

            if np.min(self.climb) > 100:
                return (self.W[0]-self.W[-1])
            else:
                return                                                   ↵
                ((self.W[0]-self.W[-1])+10*(np.min(self.climb)-100)**2)

        def trajectory_cases(self):
            data_format="{0:<30}{1:<30.16f}\n"
                                                                         ↵
            dist_header="{0:<30}{1:<30}{2:<30}{3:<30}{4:<32}{5:<30}{6:<30} ↵
            {7:<32}{8:<30}{9:<30}{10:<32}{11:<30}{12:<30}\n"
                                                                         ↵
            dist_format="{0:<30.16f}{1:<30.16f}{2:<30.16f}{3:<30.16f}{4:<3 ↵
```

```
2.16f}{5:<30.16f}{6:<30.16f}{7:<32.16f}{8:<30.16f}{9:<30.16f}{
10:<30.16f}{11:<30.16f}{12:<30.16f}\n"

self.fig_path = self.results_path+'figs/'
if self.run_type == 'breguet':
    self.breguet_opt()
    print('Run Type: ', self.run_type)
    if self.run_direction == 'f':
        print('Total Fuel Burn: ', self.W[0]-self.Wfb, 'lbf')
    else:
        print('Total Fuel Burn: ', self.iweight-self.W[-1],
        'lbf')
    print('Total Time: ', self.tb/3600.0, 'hours')
    print('L/D: ', self.L_Db)
    print('V: ', self.Vb, 'ft/s')
    print('h: ', self.hb, 'ft')
    print('TSFC: ', self.cb, 'slugs/lbf/s')
    print('Mach: ', self.Mb)
    print('total run time: ', time.time()-self.s_runtime)
    print('-----------------------------------------------------
    ')


else:
    h0 = self.hguess/10000.0
    bnds = ((0.0001, 8.),)
    tolerance = 1e-14
    opts = {'disp' : True,
        'maxiter' : 500,
        'eps' : 1e-10,
        'ftol' : 1e-11}
    ans = optimize.minimize(self.trajectory_opt,
                                    h0,
                                    method='SLSQP',
                                    bounds = bnds,
                                    tol = tolerance,
                                    options=opts)
    self.h[:] = ans.x[0]*10000.
    if ans.success == False:
        return
    print('Run Type:', self.run_type)
    print('Total Fuel Burn: ', self.W[0]-self.W[-1], 'lbf')
    print('Total Time: ', self.t[-1]/3600.0, 'hours')
    print('total run time: ', time.time()-self.s_runtime)
    print('-----------------------------------------------------
    ')
```

```
'''
data_file =                                                    ⏎
open(self.results_path+str(self.RA)+'_'+str(self.e)+'_data     ⏎
.txt', 'w')
data_file.write(data_format.format('Total Fuel Burn:',         ⏎
self.W[0]-self.W[-1]))
data_file.write(data_format.format('Total Cruise Time:',       ⏎
self.t[-1]/3600.0))
data_file.write(data_format.format('Run Time:',                ⏎
time.time()-self.s_runtime))
data_file.close()

dist_file =                                                    ⏎
open(self.results_path+str(self.RA)+'_'+str(self.e)+'_dist     ⏎
s.txt', 'w')
dist_file.write(dist_header.format('x[ft]',                    ⏎
'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/     ⏎
D','T[lbf]','TSFC[slugs/lbf/s]','W[lbf]'))
for i in range(0, len(self.x)):
    dist_file.write(dist_format.format(self.x[i],              ⏎
    self.x[i]/5280., self.t[i], self.t[i]/3600.0,              ⏎
    self.h[i], self.V[i], self.V[i]/self.a[i],                 ⏎
    self.CL[i], self.CD[i], self.CL[i]/self.CD[i],             ⏎
    self.T[i], self.c[i], self.W[i]))
dist_file.close()

if np.isnan(self.W[0]-self.W[-1]):
    return
plot_range_multiplier = 100.

#Altitude
plt.figure(1)
plt.plot(self.xm, self.h)
plt.xlabel('x [mi]')
plt.ylabel('h [ft]')
                                                               ⏎
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_h     ⏎
.png')

#Velocity
plt.figure(2)
plt.plot(self.xm, self.V)
plt.xlabel('x [mi]')
plt.ylabel('V [ft/s]')
                                                               ⏎
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_V     ⏎
.png')

#Mach
```

```
plt.figure(3)
plt.plot(self.xm, np.divide(self.V,self.a))
plt.xlabel('x [mi]')
plt.ylabel('M')
                                                                    ↵
plt.ylim(np.amin(np.divide(self.V,self.a))-(np.amax(np.div ↵
ide(self.V,self.a))-np.amin(np.divide(self.V,self.a)))*plo ↵
t_range_multiplier,                                         ↵
np.amax(np.divide(self.V,self.a))+(np.amax(np.divide(self. ↵
V,self.a))-np.amin(np.divide(self.V,self.a)))*plot_range_m ↵
ultiplier)
                                                            ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_M ↵
.png')


#CL
plt.figure(4)
plt.plot(self.xm, self.CL)
plt.xlabel('x [mi]')
plt.ylabel('CL')
                                                            ↵
plt.ylim(np.amin(self.CL)-(np.amax(self.CL)-np.amin(self.C ↵
L))*plot_range_multiplier,                                  ↵
np.amax(self.CL)+(np.amax(self.CL)-np.amin(self.CL))*plot_ ↵
range_multiplier)
                                                            ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_C ↵
L.png')

#CD
plt.figure(5)
plt.plot(self.xm, self.CD)
plt.xlabel('x [mi]')
plt.ylabel('CD')
                                                            ↵
plt.ylim(np.amin(self.CD)-(np.amax(self.CD)-np.amin(self.C ↵
D))*plot_range_multiplier,                                  ↵
np.amax(self.CD)+(np.amax(self.CD)-np.amin(self.CD))*plot_ ↵
range_multiplier)
                                                            ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_C ↵
D.png')


#L/D
plt.figure(6)
plt.plot(self.xm, self.L_D)
plt.xlabel('x [mi]')
```

```
plt.ylabel('L/D')
                                                                    ↵
plt.ylim(np.amin(self.L_D)-(np.amax(self.L_D)-np.amin(self ↵
.L_D))*plot_range_multiplier,                                       ↵
np.amax(self.L_D)+(np.amax(self.L_D)-np.amin(self.L_D))*pl ↵
ot_range_multiplier)
                                                                    ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_L ↵
_D.png')


#T
plt.figure(7)
plt.plot(self.xm, self.T)
plt.xlabel('x [mi]')
plt.ylabel('T [lbf]')
                                                                    ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_T ↵
.png')

#TSFC
plt.figure(8)
plt.plot(self.xm, self.c)
plt.xlabel('x [mi]')
plt.ylabel('TSFC [slugs/lbf/s]')
                                                                    ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_T ↵
SFC.png')

#W
plt.figure(9)
plt.plot(self.xm, self.W)
plt.xlabel('x [mi]')
plt.ylabel('W [lbf]')
                                                                    ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_W ↵
.png')
# ~ plt.show()

plt.figure(1).clear()
plt.figure(2).clear()
plt.figure(3).clear()
plt.figure(4).clear()
plt.figure(5).clear()
plt.figure(6).clear()
plt.figure(7).clear()
plt.figure(8).clear()
plt.figure(9).clear()
'''
```

```
class Aircraft_fixed_h(object):

    def __init__(self, filename):
        with open(filename) as input_file:
            data = json.load(input_file, object_pairs_hook=OrderedDict)

        self.input_airplane(data)
        self.grid_setup()
        self.array_initialize()

    def input_airplane(self, data):

        #Units
        if data["units"] == 'English':
            self.english_units = True
        else:
            self.english_units = False

        # CRM Constant Properties
        self.iweight = data["MTOW"]
        self.fweight = data["M15FW"]
        self.Sw = data["wing_area"]
        self.CD0 = data["CD0"]
        self.CD1 = data["CD1"]
        self.CD2 = data["CD2"]
        self.CM0 = data["CM0"]
        self.CM1 = data["CM1"]
        self.CM2 = data["CM2"]
        self.Ctsfc = data["CTSFC"]
        self.r = data["range"]*5280.0
        self.q = data["qTSFC"]
        self.Ts = data["static_thrust_SL"]
        self.a1 = data["a1"]
        self.a2 = data["a2"]
        self.n = data["grid"]
        self.g = 32.174
        self.name = data["name"]
        self.run_type = 'none'
        self.run_direction = data["direction"]
        self.s_runtime = 0.0
        self.f_runtime = 0.0
        self.results_path = '.'
        self.RA = 9.
        self.e = 1.
        self.hf = 35000.0

    def grid_setup(self):
        self.x = np.linspace(0, self.r, self.n)
```

```
        self.xm = self.x/5280.0

    def array_initialize(self):
        self.W = np.zeros(self.n)
        if self.run_direction == "f":
            self.W[0] = self.iweight
        elif self.run_direction == "b":
            self.W[-1] = self.fweight
        self.CL = np.zeros(self.n)
        self.CD = np.zeros(self.n)
        self.V = np.zeros(self.n)
        self.h = np.zeros(self.n)
        self.t = np.zeros(self.n)
        self.c = np.zeros(self.n)
        self.rho = np.zeros(self.n)
        self.a = np.zeros(self.n)
        self.Temp = np.zeros(self.n)
        self.T = np.zeros(self.n)
        self.climb = np.zeros(self.n)

    def opt_h_V(self, y, h, i):
        V = y
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)
        # ~ print(Temp, TSL)

        #Lift coefficient
        CL = self.W[i]/(0.5*rho*V**2*self.Sw)

        #Mach number
        M = V/a

        #Drag coefficient
        CD =                                                              ↵
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se ↵
        lf.CM2)

        #Thrust
        T = 0.5*rho*V**2*self.Sw*CD
```

```
        #Power Required
        PR = T*V

        #Power Available
        if h <36131:
            m=0.7
        else:
            m=1.0

        PA =
        2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V)
        # ~ print(i, self.W[i])

        climb = (PA-PR)/self.W[i]*60


        #TSFC
        c = self.Ctsfc*(Temp/TSL)**0.5*M**self.q

        if i==self.n-1:
            dx = self.x[i]-self.x[i-1]
        else:
            dx = self.x[i+1]-self.x[i]

        t = dx/V

        if climb>100.:
            return c*T*self.g*t
        else:
            return c*T*self.g*t+10*(climb-100)**2

    def set_state(self, y, h, i):
        V = y
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
        TSL = statmos.T(0)
        rhoSL = statmos.rho(0)

        #Lift coefficient
        CL = self.W[i]/(0.5*rho*V**2*self.Sw)

        #Mach number
```

```
        M = V/a

        #Drag coefficient
        CD =
        (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
        lf.CM2)

        #Thrust
        T = 0.5*rho*V**2*self.Sw*CD

        #Power Required
        PR = T*V

        #Power Available
        if h <36131:
            m=0.7
        else:
            m=1.0

        PA =
        2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V)
        # ~ print(i, self.W[i])

        climb = (PA-PR)/self.W[i]*60
        #TSFC
        c = self.Ctsfc*(Temp/TSL)**0.5*M**self.q

        self.h[i] = h
        self.V[i] = V
        self.rho[i] = rho
        self.a[i] = a
        self.Temp[i] = Temp
        self.CL[i] = CL
        self.CD[i] = CD
        self.T[i] = T
        self.c[i] = c
        self.climb[i] = climb

    def breguet(self, y, h, flag='obj'):
        V = y[0]
        #atmospheric properties
        if self.english_units == True:
            statmos = std.StandardAtmosphere('English')
        else:
            statmos = std.StandardAtmosphere('SI')

        rho = statmos.rho(h)
        a = statmos.a(h)
        Temp = statmos.T(h)
```

```
TSL = statmos.T(0)
rhoSL = statmos.rho(0)
# ~ print(Temp, TSL)

#Lift coefficient
if self.run_type == 'f':
    CL = self.W[0]/(0.5*rho*V**2*self.Sw)
else:
    CL = self.W[-1]/(0.5*rho*V**2*self.Sw)

#Mach number
M = V/a

#Drag coefficient
CD =
(self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
lf.CM2)


#TSFC
c = self.Ctsfc*(Temp/TSL)**0.5*M**self.q


if self.run_direction == 'f':
    Wf = self.W[0]/np.exp(self.r*self.g*c/(V*CL/CD))

    self.hb = h
    self.Vb = V
    self.L_Db = CL/CD
    self.c = c
    self.Wfb = Wf
    self.W[-1] = self.Wfb
    self.Mb = M
    self.tb = self.r/V

    #Power Req.
    PR = 0.5*rho*V**3*self.Sw*CD

    #Power Available
    if h <36131:
        m=0.7
    else:
        m=1.0
    PA =
    2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V
    )

    climb = (PA-PR)/self.W[0]*60
```

```
            if climb>100.:
                return self.W[0]-Wf
            else:
                return (self.W[0]-Wf)+10*(climb-100)**2

        elif self.run_direction == 'b':
            # ~ print(V, CL, CD)
            Wi = self.W[-1]*np.exp(self.r*self.g*c/(V*CL/CD))

            self.hb = h
            self.Vb = V
            self.L_Db = CL/CD
            self.cb = c
            self.iweight = Wi
            self.W[0] = self.iweight
            self.Mb = M
            self.tb = self.r/V

            #Power Req.
            PR = 0.5*rho*V**3*self.Sw*CD

            #Power Available
            if h <36131:
                m=0.7
            else:
                m=1.0
            PA =
            2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V
            )

            climb = (PA-PR)/self.W[-1]*60

            return Wi-self.W[-1]
            if climb>100.:
                return Wi-self.W[-1]
            else:
                return (Wi-self.W[-1])+10*(climb-100)**2

    def breguet_opt(self, h):
        y0 = [800.0]
        bnds = ((135.0, 1200.),)
        cons = [{"type" : "ineq",
                 "fun" : self.breguet}]
        ans = optimize.minimize(self.breguet,
                                y0, args=(h),
                                method='SLSQP',
                                bounds = bnds,
                                options={'disp' : False,
                                         'ftol' : 1e-16})
```

```
def service_ceiling(self, y, i):
    # ~ print(y, i)
    h = y[0]

    #atmospheric properties
    if self.english_units == True:
        statmos = std.StandardAtmosphere('English')
    else:
        statmos = std.StandardAtmosphere('SI')

    rho = statmos.rho(h)
    a = statmos.a(h)
    Temp = statmos.T(h)
    TSL = statmos.T(0)
    rhoSL = statmos.rho(0)

    if self.run_type == 'Vmf':
        V = optimize.newton(self.Vmfp, 800.0, args=(rho, a, i))
    elif self.run_type == 'Vmd':
        V = optimize.newton(self.Vmd, 800.0, args=(rho, a, i))
    else:
        V = y[1]

    #Lift coefficient
    CL = self.W[i]/(0.5*rho*V**2*self.Sw)

    #Mach number
    M = V/a

    #Drag coefficient
    CD =
    (self.CD0+self.CD1*CL+self.CD2*CL**2)*(self.CM0+self.CM1*M**se
    lf.CM2)

    #Power Req.
    PR = 0.5*rho*V**3*self.Sw*CD

    #Power Available
    if h <36131:
        m=0.7
    else:
        m=1.0
    PA =
    2.0*.9*self.Ts*(rho/rhoSL)**m*(V+self.a1*V*V+self.a2*V*V*V)

    climb = (PA-PR)/self.W[i]*60

    return climb-100.+400.
```

```
def trajectory_opt(self, r):
    h=r
    opts = {'disp' : False,
            'maxiter' : 500}#,
            # ~ 'eps' : 1e-10,
            # ~ 'ftol' : 1e-11}
    tolerance = 1e-14
    iguess = [ 850.]

    if self.run_direction == 'b':

        if self.run_type == 'full_opt':
            #Full Optimization
                                                                    ↵
            #-------------------------------------------------------↵
            ---------------------------------
            y0 = iguess
            bnds = ((100.0, 1800.),)
            for i in range(1, self.n+1):
                j = self.n-i
                ans = optimize.minimize(self.opt_h_V,
                                        y0, args=(h, j),
                                        method='SLSQP',
                                        bounds = bnds,
                                        tol = tolerance,
                                        options=opts)

                self.set_state(ans.x[0], h, j)

                if j>0:
                    self.W[j-1] =                                   ↵
                    self.W[j]+self.c[j]*self.T[j]*self.g*(self.x[j ↵
                    ]-self.x[j-1])/self.V[j]
                if ans.success == False:
                    print('optimization failed', ans.message)
                    y0 = ans.x

            for i in range(0, self.n):
                if i<self.n-1:
                    self.t[i+1] =                                   ↵
                    self.t[i]+(self.x[i+1]-self.x[i])/self.V[i]

    self.L = np.multiply(self.Sw*0.5*self.rho,                      ↵
    np.multiply(np.multiply(self.V, self.V), self.CL))
    self.D = np.multiply(self.Sw*0.5*self.rho,                      ↵
    np.multiply(np.multiply(self.V, self.V), self.CD))
    self.L_D = np.divide(self.L, self.D)
```

```
        print(' ')
        print('altitude: ', h)
        print(self.W[0]-self.W[-1])
        print(np.min(self.climb))

        if np.min(self.climb) > 100:
            return (self.W[0]-self.W[-1])
        else:
            return
            ((self.W[0]-self.W[-1])+10*(np.min(self.climb)-100)**2)

    def trajectory_cases(self):
        data_format="{0:<30}{1:<30.16f}\n"

        dist_header="{0:<30}{1:<30}{2:<30}{3:<30}{4:<32}{5:<30}{6:<30}
        {7:<32}{8:<30}{9:<30}{10:<32}{11:<30}{12:<30}\n"

        dist_format="{0:<30.16f}{1:<30.16f}{2:<30.16f}{3:<30.16f}{4:<3
        2.16f}{5:<30.16f}{6:<30.16f}{7:<32.16f}{8:<30.16f}{9:<30.16f}{
        10:<30.16f}{11:<30.16f}{12:<30.16f}\n"

        self.fig_path = self.results_path+'figs/'
        if self.run_type == 'breguet':
            h0 = self.hf
            self.breguet_opt(h0)
            print('Run Type: ', self.run_type)
            if self.run_direction == 'f':
                print('Total Fuel Burn: ', self.W[0]-self.Wfb, 'lbf')
            else:
                print('Total Fuel Burn: ', self.iweight-self.W[-1],
                'lbf')
            print('Total Time: ', self.tb/3600.0, 'hours')
            print('L/D: ', self.L_Db)
            print('V: ', self.Vb, 'ft/s')
            print('h: ', self.hb, 'ft')
            print('TSFC: ', self.cb, 'slugs/lbf/s')
            print('Mach: ', self.Mb)
            print('total run time: ', time.time()-self.s_runtime)

            print('-------------------------------------------------
            ')


        else:
            h0 = self.hf
            self.trajectory_opt(h0)

            print('Run Type:', self.run_type)
            print('Total Fuel Burn: ', self.W[0]-self.W[-1], 'lbf')
```

```
print('Total Time: ', self.t[-1]/3600.0, 'hours')
print('total run time: ', time.time()-self.s_runtime)
print('-------------------------------------------------
')

'''
data_file =
open(self.results_path+str(self.RA)+'_'+str(self.e)+'_data
.txt', 'w')
data_file.write(data_format.format('Total Fuel Burn:',
self.W[0]-self.W[-1]))
data_file.write(data_format.format('Total Cruise Time:',
self.t[-1]/3600.0))
data_file.write(data_format.format('Run Time:',
time.time()-self.s_runtime))
data_file.close()

dist_file =
open(self.results_path+str(self.RA)+'_'+str(self.e)+'_dist
s.txt', 'w')
dist_file.write(dist_header.format('x[ft]',
'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/
D','T[lbf]','TSFC[slugs/lbf/s]','W[lbf]'))
for i in range(0, len(self.x)):
    dist_file.write(dist_format.format(self.x[i],
    self.x[i]/5280., self.t[i], self.t[i]/3600.0,
    self.h[i], self.V[i], self.V[i]/self.a[i],
    self.CL[i], self.CD[i], self.CL[i]/self.CD[i],
    self.T[i], self.c[i], self.W[i]))
dist_file.close()

if np.isnan(self.W[0]-self.W[-1]):
    return
plot_range_multiplier = 100.

#Altitude
plt.figure(1)
plt.plot(self.xm, self.h)
plt.xlabel('x [mi]')
plt.ylabel('h [ft]')
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_h
.png')

#Velocity
plt.figure(2)
plt.plot(self.xm, self.V)
plt.xlabel('x [mi]')
```

```
plt.ylabel('V [ft/s]')
                                                                    ⏎
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_V ⏎
.png')

#Mach
plt.figure(3)
plt.plot(self.xm, np.divide(self.V,self.a))
plt.xlabel('x [mi]')
plt.ylabel('M')
                                                                    ⏎
plt.ylim(np.amin(np.divide(self.V,self.a))-(np.amax(np.div ⏎
ide(self.V,self.a))-np.amin(np.divide(self.V,self.a)))*plo ⏎
t_range_multiplier,                                                 ⏎
np.amax(np.divide(self.V,self.a))+(np.amax(np.divide(self. ⏎
V,self.a))-np.amin(np.divide(self.V,self.a)))*plot_range_m ⏎
ultiplier)
                                                                    ⏎
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_M ⏎
.png')


#CL
plt.figure(4)
plt.plot(self.xm, self.CL)
plt.xlabel('x [mi]')
plt.ylabel('CL')
                                                                    ⏎
plt.ylim(np.amin(self.CL)-(np.amax(self.CL)-np.amin(self.C ⏎
L))*plot_range_multiplier,                                          ⏎
np.amax(self.CL)+(np.amax(self.CL)-np.amin(self.CL))*plot_ ⏎
range_multiplier)
                                                                    ⏎
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_C ⏎
L.png')

#CD
plt.figure(5)
plt.plot(self.xm, self.CD)
plt.xlabel('x [mi]')
plt.ylabel('CD')
                                                                    ⏎
plt.ylim(np.amin(self.CD)-(np.amax(self.CD)-np.amin(self.C ⏎
D))*plot_range_multiplier,                                          ⏎
np.amax(self.CD)+(np.amax(self.CD)-np.amin(self.CD))*plot_ ⏎
range_multiplier)
                                                                    ⏎
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_C ⏎
D.png')
```

```
#L/D
plt.figure(6)
plt.plot(self.xm, self.L_D)
plt.xlabel('x [mi]')
plt.ylabel('L/D')
                                                                    ↵
plt.ylim(np.amin(self.L_D)-(np.amax(self.L_D)-np.amin(self ↵
.L_D))*plot_range_multiplier,                                       ↵
np.amax(self.L_D)+(np.amax(self.L_D)-np.amin(self.L_D))*pl ↵
ot_range_multiplier)
                                                                    ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_L ↵
_D.png')


#T
plt.figure(7)
plt.plot(self.xm, self.T)
plt.xlabel('x [mi]')
plt.ylabel('T [lbf]')
                                                                    ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_T ↵
.png')

#TSFC
plt.figure(8)
plt.plot(self.xm, self.c)
plt.xlabel('x [mi]')
plt.ylabel('TSFC [slugs/lbf/s]')
                                                                    ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_T ↵
SFC.png')

#W
plt.figure(9)
plt.plot(self.xm, self.W)
plt.xlabel('x [mi]')
plt.ylabel('W [lbf]')
                                                                    ↵
plt.savefig(self.fig_path+str(self.RA)+'_'+str(self.e)+'_W ↵
.png')
# ~ plt.show()

plt.figure(1).clear()
plt.figure(2).clear()
plt.figure(3).clear()
plt.figure(4).clear()
```

```
            plt.figure(5).clear()
            plt.figure(6).clear()
            plt.figure(7).clear()
            plt.figure(8).clear()
            plt.figure(9).clear()
            '''

def general(CRM_fo, CRM_b, sdata_file, sdist_file, sdata_format,
sdist_format, sdist_header):


    for RA in [4,6,8,10,12,14,16,18,20]:
        for e in
        [0.2,0.25,0.3,0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.8
        5,0.9,0.95,1.]:

            CRM_fo.RA = RA
            CRM_fo.e = e
            CRM_fo.CD2 = 1./(ma.pi*RA*e)
            CRM_fo.s_runtime = time.time()
            CRM_fo.trajectory_cases()
            CRM_fo.f_runtime = time.time()

            CRM_b.RA = RA
            CRM_b.e = e
            CRM_b.CD2 = 1./(ma.pi*RA*e)
            CRM_b.s_runtime = time.time()
            CRM_b.trajectory_cases()
            CRM_b.f_runtime = time.time()

            sdata_file.write(sdata_format.format(CRM_fo.RA,
            CRM_fo.e, 1./(ma.pi*CRM_fo.RA*CRM_fo.e),
            CRM_fo.W[0]-CRM_fo.W[-1], CRM_b.W[0]-CRM_b.W[-1],
            CRM_fo.t[-1]/3600., CRM_b.tb/3600., np.mean(CRM_fo.h),
            np.mean(CRM_fo.V),
            np.mean(np.divide(CRM_fo.V,CRM_fo.a)),
            np.mean(CRM_fo.CL), np.mean(CRM_fo.CD),
            np.mean(CRM_fo.L_D), np.mean(CRM_fo.T),
            np.mean(CRM_fo.c)))

            sdist_file.write(str(CRM_fo.RA)+'\t'+str(CRM_fo.e)+'\n')
            sdist_file.write(sdist_header.format('x[ft]',
            'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/
            D','T[lbf]','TSFC[slugs/lbf ft/s /s]','W[lbf]'))
            for i in range(0, len(CRM_fo.x)):
                sdist_file.write(sdist_format.format(CRM_fo.x[i],
                CRM_fo.x[i]/5280., CRM_fo.t[i], CRM_fo.t[i]/3600.0,
                CRM_fo.h[i], CRM_fo.V[i], CRM_fo.V[i]/CRM_fo.a[i],
                CRM_fo.CL[i], CRM_fo.CD[i],
```

```
                CRM_fo.CL[i]/CRM_fo.CD[i], CRM_fo.T[i], CRM_fo.c[i],
                CRM_fo.W[i]))
            sdist_file.write('\n')

    sdata_file.close()
    sdist_file.close()

def wingspan(CRM_fo, CRM_b, sdata_file, sdist_file, sdata_format,
sdist_format, sdist_header):
    Sref = CRM_fo.Sw+494.7109
    CD2p = 0.03126
    B3 =
    [-0.333333333,-0.3,-0.275,-0.25,-0.225,-0.2,-0.175,-0.15,-0.125,-0
    .1,-0.075,-0.05,-0.025,0.]
    # ~ B3 = [-0.09849]
    b =
    [215.1715403,212.0891162,209.947867,207.9367975,206.0433018,204.25
    62626,202.5658758,200.963486,199.441439,197.9929502,196.6119911,19
    5.2931905,194.0317495,192.8233686]
    # ~ b = [197.9097]
    for k in range(0, len(B3)):

        es = 1./(1.+3.*B3[k]*B3[k])

        CRM_fo.RA = b[k]**2/(Sref)
        print(CRM_fo.RA)
        CRM_fo.CD2 = CD2p+1./(ma.pi*CRM_fo.RA*es)
        CRM_fo.e = 1./(ma.pi*CRM_fo.RA*CRM_fo.CD2)
        CRM_fo.s_runtime = time.time()
        CRM_fo.trajectory_cases()
        CRM_fo.f_runtime = time.time()

        CRM_b.RA = b[k]**2/(Sref)
        CRM_b.CD2 = CD2p+1./(ma.pi*CRM_b.RA*es)
        CRM_b.e = 1./(ma.pi*CRM_b.RA*CRM_b.CD2)
        CRM_b.s_runtime = time.time()
        CRM_b.trajectory_cases()
        CRM_b.f_runtime = time.time()

        sdata_file.write(sdata_format.format(CRM_fo.RA, CRM_fo.e,
        1./(ma.pi*CRM_fo.RA*CRM_fo.e), CRM_fo.W[0]-CRM_fo.W[-1],
        CRM_b.W[0]-CRM_b.W[-1], CRM_fo.t[-1]/3600., CRM_b.tb/3600.,
        np.mean(CRM_fo.h), np.mean(CRM_fo.V),
        np.mean(np.divide(CRM_fo.V,CRM_fo.a)), np.mean(CRM_fo.CL),
        np.mean(CRM_fo.CD), np.mean(CRM_fo.L_D), np.mean(CRM_fo.T),
        np.mean(CRM_fo.c)))

        sdist_file.write(str(CRM_fo.RA)+'\t'+str(CRM_fo.e)+'\n')
        sdist_file.write(sdist_header.format('x[ft]',
```

```
        'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/D',' ↵
        T[lbf]','TSFC[slugs/lbf ft/s /s]','W[lbf]'))
        for i in range(0, len(CRM_fo.x)):
            sdist_file.write(sdist_format.format(CRM_fo.x[i],          ↵
            CRM_fo.x[i]/5280., CRM_fo.t[i], CRM_fo.t[i]/3600.0,        ↵
            CRM_fo.h[i], CRM_fo.V[i], CRM_fo.V[i]/CRM_fo.a[i],         ↵
            CRM_fo.CL[i], CRM_fo.CD[i], CRM_fo.CL[i]/CRM_fo.CD[i],     ↵
            CRM_fo.T[i], CRM_fo.c[i], CRM_fo.W[i]))
        sdist_file.write('\n')

        if np.isnan(CRM_fo.W[0]-CRM_fo.W[-1]):
            continue

    sdata_file.close()
    sdist_file.close()

def weight(CRM_fo, CRM_b, sdata_file, sdist_file, sdata_format,        ↵
sdist_format, sdist_header):
    CD2p = 0.03126
    CD00 = CRM_fo.CD0
    CD10 = CRM_fo.CD1
    Ws0 = 57027.79952145
    W_S = 152.1394
    S0 = CRM_fo.Sw+494.7109
    # ~ Sref = CRM_fo.Sw+494.7109
    b = 192.8233686
    Ws =                                                               ↵
    [32000.00,34000.00,36000.00,38000.00,40000.00,42000.00,44000.00,46 ↵
    000.00,48000.00,50000.00,52000.00,54000.00,56000.00,57027.80]
    # ~ Ws = [50635.64, 50291.14, 49893.84]
    B3 =                                                               ↵
    [-0.295362599,-0.270861909,-0.246522592,-0.222343058,-0.198321739, ↵
    -0.174457087,-0.150747576,-0.127191695,-0.103787959,-0.080534896,- ↵
    0.057431057,-0.034475009,-0.01166534,5.80698E-08]
    # ~ B3 = [-0.0987877, -0.0888952, -0.0817692]

    for k in range(0, len(Ws)):
        es = 1./(1.+3.*B3[k]*B3[k])

        CRM_fo.iweight -= (Ws0-Ws[k])
        CRM_fo.fweight -= (Ws0-Ws[k])
        CRM_fo.Sw = CRM_fo.iweight/W_S
        print(CRM_fo.Sw)
        CRM_fo.RA = b**2/(CRM_fo.Sw)
        CRM_fo.CD2 = CD2p*CRM_fo.Sw/S0+1./(ma.pi*CRM_fo.RA*es)
        CRM_fo.CD0 = CD00*CRM_fo.Sw/S0
        CRM_fo.CD1 = CD10*CRM_fo.Sw/S0
        CRM_fo.e = 1./(ma.pi*CRM_fo.RA*CRM_fo.CD2)
        CRM_fo.s_runtime = time.time()
```

```
        CRM_fo.trajectory_cases()
        CRM_fo.f_runtime = time.time()

        CRM_b.iweight -= (Ws0-Ws[k])
        CRM_b.fweight -= (Ws0-Ws[k])
        CRM_b.Sw = CRM_b.iweight/W_S
        CRM_b.RA = b**2/(CRM_b.Sw)
        CRM_b.CD2 = CD2p*CRM_b.Sw/S0+1./(ma.pi*CRM_b.RA*es)
        CRM_b.e = 1./(ma.pi*CRM_b.RA*CRM_b.CD2)
        CRM_b.s_runtime = time.time()
        CRM_b.trajectory_cases()
        CRM_b.f_runtime = time.time()

        sdata_file.write(sdata_format.format(CRM_fo.RA, CRM_fo.e,      ↵
        1./(ma.pi*CRM_fo.RA*CRM_fo.e), CRM_fo.W[0]-CRM_fo.W[-1],       ↵
        CRM_b.W[0]-CRM_b.W[-1], CRM_fo.t[-1]/3600., CRM_b.tb/3600.,    ↵
        np.mean(CRM_fo.h), np.mean(CRM_fo.V),                          ↵
        np.mean(np.divide(CRM_fo.V,CRM_fo.a)), np.mean(CRM_fo.CL),     ↵
        np.mean(CRM_fo.CD), np.mean(CRM_fo.L_D), np.mean(CRM_fo.T),    ↵
        np.mean(CRM_fo.c)))

        sdist_file.write(str(CRM_fo.RA)+'\t'+str(CRM_fo.e)+'\n')
        sdist_file.write(sdist_header.format('x[ft]',                  ↵
        'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/D',' ↵
        T[lbf]','TSFC[slugs/lbf/s]','W[lbf]'))
        for i in range(0, len(CRM_fo.x)):
            sdist_file.write(sdist_format.format(CRM_fo.x[i],          ↵
            CRM_fo.x[i]/5280., CRM_fo.t[i], CRM_fo.t[i]/3600.0,        ↵
            CRM_fo.h[i], CRM_fo.V[i], CRM_fo.V[i]/CRM_fo.a[i],         ↵
            CRM_fo.CL[i], CRM_fo.CD[i], CRM_fo.CL[i]/CRM_fo.CD[i],     ↵
            CRM_fo.T[i], CRM_fo.c[i], CRM_fo.W[i]))
        sdist_file.write('\n')

        CRM_fo.iweight += (Ws0-Ws[k])
        CRM_fo.fweight += (Ws0-Ws[k])

        if np.isnan(CRM_fo.W[0]-CRM_fo.W[-1]):
            continue



    sdata_file.close()
    sdist_file.close()

def morphing(CRM_fo, CRM_b, sdata_file, sdist_file, sdata_format,      ↵
sdist_format, sdist_header):
    Sref = CRM_fo.Sw+494.7109
    CD2p = 0.03126
    B3 =                                                               ↵
```

```
[-0.333333333,-0.3,-0.275,-0.25,-0.225,-0.2,-0.175,-0.15,-0.125,-0
.1,-0.075,-0.05,-0.025,0.]
b =
[215.1715403,212.0891162,209.947867,207.9367975,206.0433018,204.25
62626,202.5658758,200.963486,199.441439,197.9929502,196.6119911,19
5.2931905,194.0317495,192.8233686]
for k in range(0, len(B3)):

    es = 1.0

    CRM_fo.RA = b[k]**2/Sref
    CRM_fo.CD2 = CD2p+1./(ma.pi*CRM_fo.RA*es)
    CRM_fo.e = 1./(ma.pi*CRM_fo.RA*CRM_fo.CD2)
    CRM_fo.s_runtime = time.time()
    CRM_fo.trajectory_cases()
    CRM_fo.f_runtime = time.time()

    CRM_b.RA = b[k]**2/Sref
    CRM_b.CD2 = CD2p+1./(ma.pi*CRM_b.RA*es)
    CRM_b.e = 1./(ma.pi*CRM_b.RA*CRM_b.CD2)
    CRM_b.s_runtime = time.time()
    CRM_b.trajectory_cases()
    CRM_b.f_runtime = time.time()

    sdata_file.write(sdata_format.format(CRM_fo.RA, CRM_fo.e,
    1./(ma.pi*CRM_fo.RA*CRM_fo.e), CRM_fo.W[0]-CRM_fo.W[-1],
    CRM_b.W[0]-CRM_b.W[-1], CRM_fo.t[-1]/3600., CRM_b.tb/3600.,
    np.mean(CRM_fo.h), np.mean(CRM_fo.V),
    np.mean(np.divide(CRM_fo.V,CRM_fo.a)), np.mean(CRM_fo.CL),
    np.mean(CRM_fo.CD), np.mean(CRM_fo.L_D), np.mean(CRM_fo.T),
    np.mean(CRM_fo.c)))

    sdist_file.write(str(CRM_fo.RA)+'\t'+str(CRM_fo.e)+'\n')
    sdist_file.write(sdist_header.format('x[ft]',
    'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/D','
    T[lbf]','TSFC[slugs/lbf/s]','W[lbf]'))
    for i in range(0, len(CRM_fo.x)):
        sdist_file.write(sdist_format.format(CRM_fo.x[i],
        CRM_fo.x[i]/5280., CRM_fo.t[i], CRM_fo.t[i]/3600.0,
        CRM_fo.h[i], CRM_fo.V[i], CRM_fo.V[i]/CRM_fo.a[i],
        CRM_fo.CL[i], CRM_fo.CD[i], CRM_fo.CL[i]/CRM_fo.CD[i],
        CRM_fo.T[i], CRM_fo.c[i], CRM_fo.W[i]))
    sdist_file.write('\n')

    if np.isnan(CRM_fo.W[0]-CRM_fo.W[-1]):
        continue

sdata_file.close()
sdist_file.close()
```

```
def contour(CRM_fo, CRM_b, sdata_file, sdist_file, sdata_format,
sdist_format, sdist_header, runtype):
    W0t = CRM_fo.iweight
    W0f = CRM_fo.fweight
    CD2p = 0.03126
    Ws0 = 57027.79952145
    W_S = 152.1394
    S0 = CRM_fo.Sw+494.7109
    B3 =
    [-0.333333333,-0.3,-0.275,-0.25,-0.225,-0.2,-0.175,-0.15,-0.125,-0
    .1,-0.075,-0.05,-0.025,0]
    b =
    [164.0770665,170.6401492,177.2032318,183.7663145,190.3293971,196.8
    924798,203.4555625,210.0186451,216.5817278,223.1448104,229.7078931
    ,236.2709758,242.8340584,249.3971411,255.9602237,262.5233064,269.0
    863891,275.6494717,282.2125544,288.775637,295.3387197,301.9018024,
    308.464885,315.0279677,321.5910503,328.154133]
    Ws = np.zeros((len(B3), len(b)),)
    S = np.zeros((len(B3), len(b)),)
    e = np.zeros((len(B3), len(b)),)
    RA = np.zeros((len(B3), len(b)),)
    Wf = np.zeros((len(B3), len(b)),)
    ctime = np.zeros((len(B3), len(b)),)
    havg = np.zeros((len(B3), len(b)),)
    Vavg = np.zeros((len(B3), len(b)),)
    Mavg = np.zeros((len(B3), len(b)),)
    CLavg = np.zeros((len(B3), len(b)),)
    CDavg = np.zeros((len(B3), len(b)),)
    L_Davg = np.zeros((len(B3), len(b)),)
    Tavg = np.zeros((len(B3), len(b)),)
    cavg = np.zeros((len(B3), len(b)),)
    CD2 = np.zeros((len(B3), len(b)),)

    filename = 'CRM_Ws_range'

    outname = 'CRM_Ws_range'

    plane=ws.Domain(filename+'.json',True)

    for k in range(0, len(B3)):
        for l in range(0, len(b)):

            plane.wing.lift_dist_coeffs[1] = B3[k]
            plane.wing.span = b[l]

            plane.wing.discretize()
            plane.wing.calculate_lift_distribution()
            plane.set_distributions(filename+'.json')
```

```
plane.solver(1e-9, False)

Ws[k,l] = plane.weight.wing_structure
# ~ print(Ws[k,l])


if runtype == 'contour_m':
    es = 1.
else:
    es = 1./(1.+3.*B3[k]*B3[k])

CRM_fo.iweight = W0t-(Ws0-Ws[k,l])
CRM_fo.fweight = W0f-(Ws0-Ws[k,l])
CRM_fo.Sw = CRM_fo.iweight/W_S
CRM_fo.CD2 = CD2p*CRM_fo.Sw/S0+1./(ma.pi*CRM_fo.RA*es)
CRM_fo.e = 1./(ma.pi*CRM_fo.RA*CRM_fo.CD2)
CRM_fo.s_runtime = time.time()
CRM_fo.trajectory_cases()
CRM_fo.f_runtime = time.time()
CRM_fo.RA = b[l]**2/(CRM_fo.Sw)


S[k,l] = CRM_fo.Sw
e[k,l] = CRM_fo.e
RA[k,l] = CRM_fo.RA
Wf[k,l] = CRM_fo.W[0]-CRM_fo.W[-1]
ctime[k,l] = CRM_fo.t[-1]/3600.
havg[k,l] = np.mean(CRM_fo.h)
Vavg[k,l] = np.mean(CRM_fo.V)
Mavg[k,l] = np.mean(np.divide(CRM_fo.V,CRM_fo.a))
CLavg[k,l] = np.mean(CRM_fo.CL)
CDavg[k,l] = np.mean(CRM_fo.CD)
L_Davg[k,l] = np.mean(CRM_fo.L_D)
Tavg[k,l] = np.mean(CRM_fo.T)
esfcavg[k,l] = np.mean(CRM_fo.esfc)
CD2[k,l] = CRM_fo.CD2
# ~ sdata_file.write(sdata_format.format(CRM_fo.RA,
CRM_fo.e, 1./(ma.pi*CRM_fo.RA*CRM_fo.e),
CRM_fo.W[0]-CRM_fo.W[-1], CRM_b.W[0]-CRM_b.W[-1],
CRM_fo.t[-1]/3600., CRM_b.tb/3600., np.mean(CRM_fo.h),
np.mean(CRM_fo.V),
np.mean(np.divide(CRM_fo.V,CRM_fo.a)),
np.mean(CRM_fo.CL), np.mean(CRM_fo.CD),
np.mean(CRM_fo.L_D), np.mean(CRM_fo.T),
np.mean(CRM_fo.c)))

sdist_file.write(str(B3[k])+'\t'+str(b[l])+'\n')
sdist_file.write(sdist_header.format('x[ft]',
```

```
        'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/
        D','T[lbf]','TSFC[slugs/lbf ft/s /s]','W[lbf]'))
        for i in range(0, len(CRM_fo.x)):
            sdist_file.write(sdist_format.format(CRM_fo.x[i],
            CRM_fo.x[i]/5280., CRM_fo.t[i], CRM_fo.t[i]/3600.0,
            CRM_fo.h[i], CRM_fo.V[i], CRM_fo.V[i]/CRM_fo.a[i],
            CRM_fo.CL[i], CRM_fo.CD[i],
            CRM_fo.CL[i]/CRM_fo.CD[i], CRM_fo.T[i], CRM_fo.c[i],
            CRM_fo.W[i]))
        sdist_file.write('\n')

        if np.isnan(CRM_fo.W[0]-CRM_fo.W[-1]):
            continue


        # ~ fig1.gca().plot(CRM_fo.xm, CRM_fo.h)
        # ~ fig2.gca().plot(CRM_fo.xm, CRM_fo.V)
        # ~ fig3.gca().plot(CRM_fo.xm,
        np.divide(CRM_fo.V,CRM_fo.a))
        # ~ fig4.gca().plot(CRM_fo.xm, CRM_fo.CL)
        # ~ fig5.gca().plot(CRM_fo.xm, CRM_fo.CD)
        # ~ fig6.gca().plot(CRM_fo.xm, CRM_fo.L_D)
        # ~ fig7.gca().plot(CRM_fo.xm, CRM_fo.T)
        # ~ fig8.gca().plot(CRM_fo.xm, CRM_fo.esfc)
        # ~ fig9.gca().plot(CRM_fo.xm, CRM_fo.W)

sdata_file = open(results_path+'sdata.txt', 'w')
sdata_file.write('Wing-Structure Weight'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(Ws[i,j])+',')

sdata_file.write('\n'+'\n'+'Aspect Ratio'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(RA[i,j])+',')

sdata_file.write('\n'+'\n'+'Oswald efficiency
Factor'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
```

```
    for j in range(0, len(b)):
        sdata_file.write(str(e[i,j])+',')

sdata_file.write('\n'+'\n'+'CD2'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(CD2[i,j])+',')

sdata_file.write('\n'+'\n'+'Fuel Burn'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(Wf[i,j])+',')

sdata_file.write('\n'+'\n'+'Cruise Time'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(ctime[i,j])+',')

sdata_file.write('\n'+'\n'+'average altitude'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(havg[i,j])+',')

sdata_file.write('\n'+'\n'+'average velocity'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
        sdata_file.write(str(Vavg[i,j])+',')

sdata_file.write('\n'+'\n'+'average Mach'+'\n'+',')
for j in range(0, len(b)):
    sdata_file.write(str(b[j])+',')
for i in range(0, len(B3)):
    sdata_file.write('\n'+str(B3[i])+',')
    for j in range(0, len(b)):
```

```
            sdata_file.write(str(Mavg[i,j])+',')

    sdata_file.write('\n'+'\n'+'average CL'+'\n'+',')
    for j in range(0, len(b)):
        sdata_file.write(str(b[j])+',')
    for i in range(0, len(B3)):
        sdata_file.write('\n'+str(B3[i])+',')
        for j in range(0, len(b)):
            sdata_file.write(str(CLavg[i,j])+',')

    sdata_file.write('\n'+'\n'+'average CD'+'\n'+',')
    for j in range(0, len(b)):
        sdata_file.write(str(b[j])+',')
    for i in range(0, len(B3)):
        sdata_file.write('\n'+str(B3[i])+',')
        for j in range(0, len(b)):
            sdata_file.write(str(CDavg[i,j])+',')

    sdata_file.write('\n'+'\n'+'average L/D'+'\n'+',')
    for j in range(0, len(b)):
        sdata_file.write(str(b[j])+',')
    for i in range(0, len(B3)):
        sdata_file.write('\n'+str(B3[i])+',')
        for j in range(0, len(b)):
            sdata_file.write(str(L_Davg[i,j])+',')

    sdata_file.write('\n'+'\n'+'average Thrust'+'\n'+',')
    for j in range(0, len(b)):
        sdata_file.write(str(b[j])+',')
    for i in range(0, len(B3)):
        sdata_file.write('\n'+str(B3[i])+',')
        for j in range(0, len(b)):
            sdata_file.write(str(Tavg[i,j])+',')

    sdata_file.write('\n'+'\n'+'average ESFC'+'\n'+',')
    for j in range(0, len(b)):
        sdata_file.write(str(b[j])+',')
    for i in range(0, len(B3)):
        sdata_file.write('\n'+str(B3[i])+',')
        for j in range(0, len(b)):
            sdata_file.write(str(esfcavg[i,j])+',')

    sdata_file.close()
    sdist_file.close()

def single(CRM_fo, CRM_b, sdata_file, sdist_file, sdata_format,        ↵
sdist_format, sdist_header):
    Sref = CRM_fo.Sw+494.7109
    b = 192.8233686
```

```
    CRM_fo.RA = b**2/Sref
    CRM_fo.e = 1./(ma.pi*CRM_fo.RA*CRM_fo.CD2)
    CRM_fo.s_runtime = time.time()
    CRM_fo.trajectory_cases()
    CRM_fo.f_runtime = time.time()

    CRM_b.RA = b**2/Sref
    CRM_b.e = 1./(ma.pi*CRM_b.RA*CRM_b.CD2)
    CRM_b.s_runtime = time.time()
    CRM_b.trajectory_cases()
    CRM_b.f_runtime = time.time()


    sdata_file.write(sdata_format.format(CRM_fo.RA, CRM_fo.e,          ↵
    1./(ma.pi*CRM_fo.RA*CRM_fo.e), CRM_fo.W[0]-CRM_fo.W[-1],           ↵
    CRM_b.W[0]-CRM_b.W[-1], CRM_fo.t[-1]/3600., CRM_b.tb/3600.,        ↵
    np.mean(CRM_fo.h), np.mean(CRM_fo.V),                              ↵
    np.mean(np.divide(CRM_fo.V,CRM_fo.a)), np.mean(CRM_fo.CL),         ↵
    np.mean(CRM_fo.CD), np.mean(CRM_fo.L_D), np.mean(CRM_fo.T),        ↵
    np.mean(CRM_fo.c)))

    sdist_file.write(str(CRM_fo.RA)+'\t'+str(CRM_fo.e)+'\n')
    sdist_file.write(sdist_header.format('x[ft]',                      ↵
    'x[mi]','t[s]','t[hr]','h[ft]','V[ft/s]','M','CL','CD','L/D','T[lb ↵
    f]','TSFC[slugs/lbf/s]','W[lbf]'))
    for i in range(0, len(CRM_fo.x)):
        sdist_file.write(sdist_format.format(CRM_fo.x[i],              ↵
        CRM_fo.x[i]/5280., CRM_fo.t[i], CRM_fo.t[i]/3600.0,            ↵
        CRM_fo.h[i], CRM_fo.V[i], CRM_fo.V[i]/CRM_fo.a[i],             ↵
        CRM_fo.CL[i], CRM_fo.CD[i], CRM_fo.CL[i]/CRM_fo.CD[i],         ↵
        CRM_fo.T[i], CRM_fo.c[i], CRM_fo.W[i]))
    sdist_file.write('\n')

    sdata_file.close()
    sdist_file.close()

def run(runtype, runfile):
    # general
    # wingspan
    # weight
    # morphing
    # general_h
    # wingspan_h
    # weight_h
    # morphing_h
    # contour
    # contour_m
    #*_hf
```

```
# single
# single_h
if runtype == 'single':
    results_path = './results/constrained/CRM_range/'+runfile
elif runtype == 'single_h':
    results_path = './results/constrained/CRM_range/'+runfile+'_h'
else:
    results_path = './results/constrained/CRM_range/'+runtype

sdata_header="{0:<30}{1:<30}{2:<30}{3:<30}{4:<30}{5:<30}{6:<30}{7:<30}{8:<30}{9:<30}{10:<30}{11:<30}{12:<30}{13:<30}{14:<30}\n"

sdata_format="{0:<30.16f}{1:<30.16f}{2:<30.16f}{3:<30.16f}{4:<30.16f}{5:<30.16f}{6:<30.16f}{7:<30.16f}{8:<30.16f}{9:<30.16f}{10:<30.16f}{11:<30.16f}{12:<30.16f}{13:<30.16f}{14:<30.16f}\n"

sdist_header="{0:<30}{1:<30}{2:<30}{3:<30}{4:<32}{5:<30}{6:<30}{7:<32}{8:<30}{9:<30}{10:<32}{11:<30}{12:<30}\n"

sdist_format="{0:<30.16f}{1:<30.16f}{2:<30.16f}{3:<30.16f}{4:<32}{5:<30.16f}{6:<30.16f}{7:<32}{8:<30.16f}{9:<30.16f}{10:<32}{11:<30.16f}{12:<30.16f}\n"

if runtype!='contour' or runtype!='contour_m':
    sdata_file = open(results_path+'/sdata.txt', 'w')
    sdata_file.write(sdata_header.format('Aspect Ratio', 'oswald efficiency', 'CD2', 'Fuel Burn [lbf]', 'Fuel Burn (breguet) [lbf]', 'time [hr]', 'time (breguet) [hr]', 'average altitude [ft]', 'average velocity [ft/s]', 'average Mach', 'average CL', 'average CD', 'average L/D', 'average T [lbf]', 'average TSFC [slugs/lbf/s]'))

sdist_file = open(results_path+'/sdists.txt', 'w')

if runtype[-1] == 'h':
    CRM_fo = Aircraft_h(runfile+'.json')
    CRM_b = Aircraft_h(runfile+'.json')
elif runtype[-1] == 'f':
    CRM_fo = Aircraft_fixed_h(runfile+'.json')
    CRM_b = Aircraft_fixed_h(runfile+'.json')
else:
    CRM_fo = Aircraft(runfile+'.json')
    CRM_b = Aircraft(runfile+'.json')

CRM_fo.run_type = 'full_opt'
CRM_b.run_type = 'breguet'

CRM_fo.results_path = results_path
CRM_b.results_path = results_path
```

```
# ~ fig1=plt.figure(1)
# ~ fig1.gca().set_xlabel('x [mi]')
# ~ fig1.gca().set_ylabel('h [ft]')

# ~ fig2=plt.figure(2)
# ~ fig2.gca().set_xlabel('x [mi]')
# ~ fig2.gca().set_ylabel('V [ft/s]')

# ~ fig3=plt.figure(3)
# ~ fig3.gca().set_xlabel('x [mi]')
# ~ fig3.gca().set_ylabel('M')

# ~ fig4=plt.figure(4)
# ~ fig4.gca().set_xlabel('x [mi]')
# ~ fig4.gca().set_ylabel('CL')

# ~ fig5=plt.figure(5)
# ~ fig5.gca().set_xlabel('x [mi]')
# ~ fig5.gca().set_ylabel('CD')

# ~ fig6=plt.figure(6)
# ~ fig6.gca().set_xlabel('x [mi]')
# ~ fig6.gca().set_ylabel('L/D')

# ~ fig7=plt.figure(7)
# ~ fig7.gca().set_xlabel('x [mi]')
# ~ fig7.gca().set_ylabel('T [lbf]')

# ~ fig8=plt.figure(8)
# ~ fig8.gca().set_xlabel('x [mi]')
# ~ fig8.gca().set_ylabel('ESFC [slugs/lbf ft/s /s]')

# ~ fig9=plt.figure(9)
# ~ fig9.gca().set_xlabel('x [mi]')
# ~ fig9.gca().set_ylabel('W [lbf]')
if runtype == 'general' or runtype == 'general_h' or runtype ==
'general_hf':
    general(CRM_fo, CRM_b, sdata_file, sdist_file, sdata_format,
    sdist_format, sdist_header)
if runtype == 'wingspan' or runtype == 'wingspan_h' or runtype
== 'wingspan_hf':
    wingspan(CRM_fo, CRM_b, sdata_file, sdist_file,
    sdata_format, sdist_format, sdist_header)
if runtype == 'weight' or runtype == 'weight_h' or runtype ==
'weight_hf':
    weight(CRM_fo, CRM_b, sdata_file, sdist_file, sdata_format,
    sdist_format, sdist_header)
if runtype == 'morphing' or runtype == 'morphing_h' or runtype
```

```
== 'morphing_hf':
    morphing(CRM_fo, CRM_b, sdata_file, sdist_file,
    sdata_format, sdist_format, sdist_header)
if runtype == 'contour' or runtype == 'contour_m':
    contour(CRM_fo, CRM_b, sdata_file, sdist_file, sdata_format,
    sdist_format, sdist_header, runtype)
if runtype == 'single' or runtype == 'single_h' or runtype ==
'single_hf':
    single(CRM_fo, CRM_b, sdata_file, sdist_file, sdata_format,
    sdist_format, sdist_header)
# ~ fig1.savefig(results_path+'/h.png')
# ~ fig2.savefig(results_path+'/V.png')
# ~ fig3.savefig(results_path+'/M.png')
# ~ fig4.savefig(results_path+'/CL.png')
# ~ fig5.savefig(results_path+'/CD.png')
# ~ fig6.savefig(results_path+'/L_D.png')
# ~ fig7.savefig(results_path+'/T.png')
# ~ fig8.savefig(results_path+'/ESFC.png')
# ~ fig9.savefig(results_path+'/W.png')
# ~ plt.show()
        # ~ fig1.gca().plot(CRM_fo.xm, CRM_fo.h)
        # ~ fig2.gca().plot(CRM_fo.xm, CRM_fo.V)
        # ~ fig3.gca().plot(CRM_fo.xm,
        np.divide(CRM_fo.V,CRM_fo.a))
        # ~ fig4.gca().plot(CRM_fo.xm, CRM_fo.CL)
        # ~ fig5.gca().plot(CRM_fo.xm, CRM_fo.CD)
        # ~ fig6.gca().plot(CRM_fo.xm, CRM_fo.L_D)
        # ~ fig7.gca().plot(CRM_fo.xm, CRM_fo.T)
        # ~ fig8.gca().plot(CRM_fo.xm, CRM_fo.esfc)
        # ~ fig9.gca().plot(CRM_fo.xm, CRM_fo.W)
```

C.2.3 Ikhana Baseline Input File (python)

```
{
    "name" : "Ikhana",
    "units" : "English",
    "MTOW" : 8500.0,
    "M15FW" : 5950.0,
    "wing_area" : 265.625,
    "CD0" : 0.023,
    "CD1" : 0.0,
    "CD2" : 0.034257,
    "CM0" : 1.0,
    "CM1" : 3.0,
    "CM2" : 30.0,
    "range" : 3500.0,
    "a00" : 0.556861696,
    "a01" : -2.78033E-05,
    "a02" : -3.38045E-07,
    "a10" : -2.60964E-06,
    "a11" : -4.71393E-09,
    "a12" : 2.00694E-11,
    "a20" : 7.03639E-11,
    "a21" : 1.69186E-13,
    "a22" : -7.6756E-16,
    "b00" : 1004.8398,
    "b01" : 0.036841611,
    "b02" : 0.001600612,
    "b10" : -0.021716745,
    "b11" : 2.31748E-08,
    "b12" : -1.96944E-08,
    "b20" : 8.44697E-08,
    "b21" : -4.35643E-11,
    "b22" : 2.52213E-14,
    "grid" : 200,
    "direction" : "b"
}
```

C.2.4 CRM Baseline Input File (python)

```
{
    "name" : "CRM",
    "units" : "English",
    "MTOW" : 628342.1825,
    "M15FW" : 370663.8114,
    "wing_area" : 3635.289071,
    "CD0" : 0.0194411562227676,
    "CD1" : -0.0159788833189256,
    "CD2" : 0.066617,
    "CM0" : 1.0,
    "CM1" : 3.0,
    "CM2" : 30.0,
    "range" : 7725.0,
    "CTSFC" : 0.0000060706,
    "qTSFC" : 0.6,
    "static_thrust_SL" : 93000.0,
    "a1" : -9.50e-4,
    "a2" : 5e-7,
    "grid" : 200,
    "direction" : "b"
}
```

C.3 Contour Plotting

The following example code extracts can be used to produce contour plots specifically formatted like those in Chapters 2 and 3.

C.2.1 Contour Plotting Module (python)

```
"""
contour.py

Produces formatted contour plots for publication in .emf format.

This module is intended to allow a user to produce journal-quality contour
plots. It takes an input file containing many of the formatting options
in one location. The input file is meant to streamline the formatting of
the plot, and allows the inclusion of several data series. It is meant to
be run in conjunction with the bash script contour_save.sh, which allows
the user to save the figure in enhnaced windows metafile (.emf) format
and specify the desired name of the .emf file.


Parameters
----------
contour_plot_settings.json : input .json file
    Input file containing the following:
{
    "General_Format":{Contains general formatting parameters
        "N" : Integer, number of data points per axis, i.e. N=100 means 100x100
              grid.
        "Font_Size":, float, Size of the default figure font, in pt
        "Figure_Size":, array Size of the figure, in inches in the order
                        [width,height]
        {"Grid": Contains grid parameters
            "is_present":, {0,1} integer, 0 for grid off, any other integer
                          for grid on
            "Which":, string, either 'major' for gridlines on major ticks
                      only, or 'minor' for gridlines on minor ticks
            "Alpha":, float between 0 and 1 specifying transparency of
```

```
                gridlines
        "Line_Color":, string, color of the gridlines, accepts any
                        valid python color string
        "Line_Style":, string, style of the gridlines, accepts any
                        valid python linestyle string
        "Dash_Style":, tuple, (Optional) accepts dash/space
                        specification, i.e. (0, (10.0, 5.0))
        "Line_Width": float, width of the gridlines, in pt.
    }
},
"Axis_Format":{ Contains axis formatting parameters
    "Minor_Ticks":, integer, 0 for no minor ticks, any other
                integer to include minor ticks
    "Tick_Label_Size":, float, fontsize of the tick labels, in pt
    "X_Axis":{ Contains formatting parameters for the x axis.
                Repeat for Y_Axis
        "Limits":, array, contains lower and upper bounds for the x
                    axis in the order [lower, upper]
        "Tick_Density":, array of values specifying tick values
        "Tick_Label_Format": string, format string specifying the
                            output format of the tick labels.
                            Accepts
                            any valid python format string.
    }
},
"Data_Series_Format":{ Contains formatting parameters for the
data series
    "Series_1":{ formatting parameters for the first series
                duplicate for any subsequent series, i.e.
                Series_2, etc.
        "index":, order that the data series is listed in
        contour_data
        "Name":, string, name of the data series
        "Line_Width":, float, width of the data series, in pt
        "Line_Color":, string, color of the data series, accepts
        any
                        valid python color string
        "Line_Style": string, style of the gridlines, accepts any
                        valid python linestyle string
        "Dash_Style":, tuple, (Optional) accepts dash/space
                        specification, i.e. (0, (10.0, 5.0))
        "Contour_Density":, float specifying contour density or
                            array of floats specifying contour
                            levels
        "Labels":{ contains formatting parameters for contour
        labels
            "is_present":, {0,1} integer, 0 for labels off, any
            other
                        integer for labels on
```

```
                "Fontsize":, float, fontsize of the labels, in pt
                "Label_Format":, string, format string specifying the
                                 output format of the tick labels.
                                 Accepts
                                 any valid python format string.
            }
        },
    }
}

Returns
-------------------------------------------------------------------------
--
Figure.emf
    Returns a figure in .emf format, with name specified by the user.

Notes
-------------------------------------------------------------------------
--
Data is imported into the conotour.py module via contour_data.py.
contour_data.py contains the function data_import, which returns the
arrays
A, B, and C. A and B are 1-D arrays containing the X and Y values,
respectively. C is a 1-D array of 2-D arrays specifying each data
series.

Example
-------------------------------------------------------------------------
--
contour.run_plot(input.json)

"""
import sys
import numpy as np
import matplotlib

matplotlib.use('TKAgg')

import matplotlib.pyplot as plt
import matplotlib.cm as cm
from matplotlib import rc
import contour_data as cdat
import json
from collections import OrderedDict
from matplotlib.ticker import FormatStrFormatter
from ast import literal_eval

rc('font', **{'family':'serif', 'serif':['Times']})
```

```
def run_plot(input_file):
    """ The run_plot function creates the contour plot and allows
    the user
    the option to save the file.

    Parameters:

    ----------
    --
    input_file: dinput file containing formatting data

    Returns:

    ----------
    --
    matplotlib figure in .svg format.

    """
    # Read in settings
    format_data = _settings_read_(input_file)

    # Create Plot with General Formatting
    fig, ax = _create_baseplot_(format_data)

    # Format Axis
    ax = _format_axis_(format_data, ax)

    # Plot Data
    ax = plot_data(format_data, ax)

    # Save Plot
    saveflag = _plot_save_(input_file)

    # End Terminal Operation if figure is not to be saved
    if(saveflag is False):
        sys.exit(0)


def _settings_read_(input_file):
    """ The settings_read function reads in the input file and stores
    the values in the data dict.

    Parameters:

    ----------
    --
    input_file: .json file containing the formatting data as
    described above.
```

```
    Returns:
                                                                        ↵
    ------------------------------------------------------------------- ↵
    --
    format_data: dictionary containing the formatting data from the     ↵
    input file.

    """
    with open(input_file+'.json') as file:
        format_data = json.load(file, object_pairs_hook=OrderedDict)

    return format_data


def _create_baseplot_(format_data):
    """ The create_baseplot function creates the baseline plot with     ↵
    general
    formatting.

    Parameters:
                                                                        ↵
    ------------------------------------------------------------------- ↵
    --
    format_data: dictionary containing the formatting data from the     ↵
    input file.

    Returns:
                                                                        ↵
    ------------------------------------------------------------------- ↵
    --
    fig: matplotlib figure object
    ax: matplotlib axis object

    """
    # Create Figure
    fig =                                                               ↵
    plt.figure(figsize=(format_data["General_Format"]["Figure_Size"][0  ↵
    ],
                                                                        ↵
                               format_data["General_Format"]["Figure_Si ↵
                               ze"][1]))
    ax = fig.add_subplot(111)

    # Format Grid
    if(format_data["General_Format"]["Grid"]["is_present"] != 0):
        # Set Line Style
        line_style =                                                    ↵
        format_data["General_Format"]["Grid"]["Line_Style"]
        if(line_style == 'dashed'):
```

```
            line_style =
            literal_eval(format_data["General_Format"]["Grid"]["Dash_S
            tyle"])

        ax.grid(which=format_data["General_Format"]["Grid"]["Which"],
                alpha=format_data["General_Format"]["Grid"]["Alpha"],
                color=format_data["General_Format"]["Grid"]["Line_Colo
                r"],
                linestyle=line_style,
                linewidth=format_data["General_Format"]["Grid"]["Line_
                Width"])

    # Set Border
    ax.spines['top'].set_linewidth(1.3)
    ax.spines['right'].set_linewidth(1.3)
    ax.spines['bottom'].set_linewidth(1.3)
    ax.spines['left'].set_linewidth(1.3)

    return fig, ax


def _format_axis_(format_data, ax):
    """ The format_axis function applies axis-specific formatting.

    Parameters:

    ----------
    --
    format_data: dictionary containing the formatting data from the
    input file.
    ax: matplotlib axis object

    Returns:

    ----------
    --
    ax: matplotlib axis object

    """
    # Set X and Y Limits
    ax.set_xlim(format_data["Axis_Format"]["X_Axis"]["Limits"])
    ax.set_ylim(format_data["Axis_Format"]["Y_Axis"]["Limits"])

    # Set Tick Density
    ax.set_xticks(format_data["Axis_Format"]["X_Axis"]["Tick_Density"])
    ax.set_yticks(format_data["Axis_Format"]["Y_Axis"]["Tick_Density"])
```

```
    # Toggle Minor Ticks
    if(format_data["Axis_Format"]["Minor_Ticks"] != 0):
        ax.minorticks_on()

    # Format Tick Marks
    ax.tick_params(which='major',
                                                                    ↵
                   labelsize=format_data["Axis_Format"]["Tick_Label_Si ↵
                   ze"],
                   direction='in',
                   width=0.75,
                   length=3.0,
                   top=True,
                   right=True,
                   pad=7.25)
    ax.tick_params(which='minor',
                                                                    ↵
                   labelsize=format_data["Axis_Format"]["Tick_Label_Si ↵
                   ze"],
                   direction='in',
                   width=0.25,
                   length=1.75,
                   top=True,
                   right=True,
                   pad=7.25)

    # Format Tick Labels
                                                                    ↵
    ax.xaxis.set_major_formatter(FormatStrFormatter(format_data["Axis_ ↵
    Format"]["X_Axis"]["Tick_Label_Format"]))
                                                                    ↵
    ax.yaxis.set_major_formatter(FormatStrFormatter(format_data["Axis_ ↵
    Format"]["Y_Axis"]["Tick_Label_Format"]))

    return ax


def plot_data(format_data, ax):
    """ The plot_data function plots and formats the data series

    Parameters:
                                                                    ↵
    ------------------------------------------------------------------ ↵
    --
    format_data: dictionary containing the formatting data from the   ↵
    input file.
    ax: matplotlib axis object

    Returns:
```

```
                                                                    ↵
------------------------------------------------------------------- ↵
--
ax: matplotlib axis object

"""
# Import data from contour_data.py
A, B, C = cdat.data_import(format_data["General_Format"]["N"],
                                                                    ↵
                           format_data["Axis_Format"]["X_Axis"]["L ↵
                           imits"][0],
                                                                    ↵
                           format_data["Axis_Format"]["X_Axis"]["L ↵
                           imits"][1],
                                                                    ↵
                           format_data["Axis_Format"]["Y_Axis"]["L ↵
                           imits"][0],
                                                                    ↵
                           format_data["Axis_Format"]["Y_Axis"]["L ↵
                           imits"][1])

X, Y = np.meshgrid(A, B)

CS = []
# Data Series Formatting and Storage
for series in format_data["Data_Series_Format"].keys():
    CS.append(ax.contour(X, Y,                                      ↵
    C[format_data["Data_Series_Format"][series]["index"]],
                                                                    ↵
                         format_data["Data_Series_Format"][series] ↵
                         ["Contour_Density"],
                                                                    ↵
                         colors=format_data["Data_Series_Format"][ ↵
                         series]["Line_Color"],
                                                                    ↵
                         linewidths=format_data["Data_Series_Forma ↵
                         t"][series]["Line_Width"],
                                                                    ↵
                         linestyles=format_data["Data_Series_Forma ↵
                         t"][series]["Line_Style"]))

    # Dash Formatting
    if(CS[-1].linestyles == 'dashed'):
        for c in CS[-1].collections:
                                                                    ↵
            c.set_dashes([literal_eval(format_data["Data_Series_Fo ↵
            rmat"][series]["Dash_Style"])])

    # Data Series Labeling
```

```
            if(format_data["Data_Series_Format"][series]["Labels"]["is_pre
            sent"] != 0):
                ax.clabel(CS[-1], inline=1,
                          fontsize=format_data["Data_Series_Format"][serie
                          s]["Labels"]["Fontsize"],
                          fmt=format_data["Data_Series_Format"][series]["L
                          abels"]["Label_Format"])

        return ax


    def _plot_save_(input_file):
        """ The plot_save function allows the user to save the figure as
        .svg
        It is meant to interface with the contour_save.sh bash script to
        save
        the figure as .emf and allow the user to specify the desired
        name of
        the figure.

        """
        plt.show(block=False)
        choiceflag = False
        saveflag = False
        while(choiceflag is False):
            print(' ')
            saving = input("Save Figure? (y/n) ")

            if(saving == 'y'):
                plt.savefig(input_file+'tempfile.svg',
                bbox_inches='tight', transparent=True)
                choiceflag = True
                saveflag = True
            elif(saving == 'n'):
                saveflag = False
                choiceflag = True
            else:
                print(saving+' is not a valid response. Try again.')

        return saveflag
```

C.2.2 Contour Data Formatting (python)

```
"""
contour_data.py
Used to generate data in a format useful for contour plotting with
contour.py

The data_import function is meant to allow the user of the contour
module
to calculate data however the user would like and return it in the
correct form
for use in the contour module.

Parameters
----------
N: number of data points per axis. i.e. N=100 gives 100 x 100 grid
xl: lower bound of x-axis data
xu: upper bound of x-axis data
yl: lower bound of y-axis data
yu: upper bound of y-axis data

Returns
-------
A: 1-D array of length N containing x-axis values
B: 1-D array of length N containing y-axis values
C: 1-D array of 2-D arrays containing z-axis values for each x-y
pair for each
   data series. size of each 2-D array: NxN size of C: number of
   desired data
   series. Note: The order of the data series in the array C should
   match the
   order of the data series formatting in the input file to the
   contour module.
"""

import numpy as np
import math as ma

def data_import(N,xl,xu,yl,yu):

    # Calculate Data here

    # Example Data
    A=np.linspace(xl, xu, N)
    B=np.linspace(yl, yu, N)
```

```
C1=np.zeros((N, N), dtype=np.float64)
C2=np.zeros((N, N), dtype=np.float64)
for i in range(0, N):
    for j in range(0, N):
        C1[i, j] = A[i]*B[j]
        C2[i, j] = A[i]**2*B[j]

C=[C1, C2]

return A,B,C # where C=[C1, C2, C3, etc...]
```

C.2.3 Running and Saving the Contour Plot (bash)

```
#!/bin/bash
# Saves Contour plot as .svg
# Opens .svg in inkscape
# Resizes inkscape document
# Saves inkscape document as .emf for windows applications

#Contour Plot Script
SCRIPT='contour_plot.py'
TEMPFILE='tempfile.svg'

echo 'Please Enter Input Filename (without Extension): '
read INPUTFILE

python $SCRIPT $INPUTFILE

if [ -e $INPUTFILE$TEMPFILE ]
then
    #Ask for Final Filename
    echo ' '
    echo 'Please Enter .emf Filename (without Extension): '
    #Read user input to the EMFFILE variable
    read EMFFILE
    #open inkscape
    inkscape $INPUTFILE$TEMPFILE --export-emf=$EMFFILE'.emf'
    #delete temporary file
    rm -r $INPUTFILE$TEMPFILE
    echo ' '
    echo 'File '$EMFFILE'.emf was saved successfully!'
else
    echo ' '
    echo 'File was not saved!'
    exit 1
fi
```

JEFFREY D. TAYLOR

1199 W. 1780 S. Logan, Utah 84321
801·510·9461
jeffdtaylor3891@gmail.com
linkedin.com/in/jtaylor17
researchgate.net/profile/Jeffrey-Taylor-7

EDUCATION

Ph.D. | **Utah State University** | Mechanical Engineering, **Anticipatory Intelligence Certificate** *July 2022*
M.S. | **Utah State University** | Mechanical Engineering *December 2017*
B.S. | **Utah State University** | Mechanical Engineering *Cum Laude*, *May 2016*

ACADEMIC & PROFESSIONAL POSITIONS

NASA Graduate Research Fellow, NASA Ames Research Center & Utah State University *Sep 2018-May 2022*
Research Assistant, Utah State University, Center for Anticipatory Intelligence *Sep 2021*
Research Assistant, Utah State University, Mechanical & Aerospace Engineering *Feb 2016-Sep 2018*
Teaching Assistant, Flight Mechanics, Utah State University *Jan-May 2017 & 2018*
Graduate Student Summer Faculty Fellow, Air Force Research Laboratory *May-Aug 2017*
Production Engineering Intern, YESCO Electronics *Oct 2015-Feb 2016*
New Products Engineering Intern, Futura Industries *May-Aug 2015 & 2016*

PROJECTS & EXPERIENCE

Performance Adaptive Aeroelastic Wing, NASA Ames Research Center

- Performed analyses to determine optimum trajectory and wing design characteristics, including lift distribution and flap schedule, to minimize fuel consumption on aircraft wings with distributed multi-segment flaps.
- Created computational framework for combined trajectory and aerostructural optimization of aircraft wings.
- Characterized fuel-burn benefits of distributed flap systems for typical transonic commercial aircraft.

Variable Camber Compliant Wing, Air Force Research Laboratory

- Created low-fidelity fluid-structure interaction software in Python to predict optimum aerostructural design characteristics for wings with arbitrary planform and payload distribution.
- Derived analytic expressions for the optimum structural weight of wings with tapered and elliptic planforms.
- Identified several theoretical optimum aerostructural solutions that show excellent agreement with high-fidelity results using analytic methods based on lifting-line theory.

Numerical Lifting-Line Method for Swept Wings, Utah State University

- Isolated mathematical source of grid-resolution errors for swept-wing numerical lifting-line algorithm solutions.
- Performed preliminary research that enabled the development of a general implementation of lifting-line theory capable of aerodynamic predictions for swept wings.

Aerodynamic Consulting & Modeling Support

- Identified flutter mitigation strategies for the design of lightweight automobile rooftop cargo carrier.
- Provided modeling support for America's Cup AC75 yacht design team using numerical lifting-line methods.

Hypersonic Missile Defense, Space Dynamics Laboratory

- Identified key motivations for Russian and Chinese hypersonic weapons development through strategic analysis.
- Analyzed Russian and Chinese hypersonic weapons capabilities, development programs, and research infrastructure through open-source review.
- Presented 4 recommendations for hypersonic missile defense development based on Russian and Chinese hypersonic motivations.

New Products Engineering, Futura Industries

- Authored 2 proposals to management that secured funding for the creation of 2 new automated workcells.
- Led project to design and build tooling that reduced carpet metal sample production times by over 50 percent.

RESEARCH

Research Interests

General: aerodynamics, multidisciplinary design & optimization, flight mechanics & performance, bio-inspired flight, numerical methods, optimization, air vehicle stability & control, aerospace & defense policy

Applications: morphing aircraft, aeroelasticity, efficient airframe design and optimization, novel aircraft configurations, hypersonic vehicle design & performance, sUAS design & control

Research Projects

- **Research Assistant,** "The Future of Human governance: Assessing China's Ambitions to Proliferate AI-Enabled Authoritarianism," **U.S. Department of Defense Minerva Research Initiative**, PI: Jeannie Johnson, USU Center for Anticipatory Intelligence, July 2022-July 2025, Total: $1,444,954. (RA portion $5,000)
- **Student Fellow,** "Continuous Flight Optimization of Morphing-Wing Aircraft," **NASA ARMD Fellowship Grant**, PI: Douglas Hunsaker, Utah State University, September 2018-May 2022, $165,000.
- **PI,**"Strategic Review of US Hypersonic Missile Defense Development Considering Russian and Chinese Hypersonic Motivations," **Center for Anticipatory Intelligence/Space Dynamics Laboratory Capstone Project**, January-October 2021.
- **Research Assistant,** "Methods for the Aerostructural Design and Optimization of Wings with Arbitrary Planform and Payload Distribution," **AeroLab Internal Funding**, April 2017-June 2018.
- **Student Fellow,** "A Method for Evaluating the Optimum Lift Distribution and Wing Structural Requirements for a Given Payload," **AFRL Summer Faculty Fellowship**, Faculty Fellow: Douglas Hunsaker, Utah State University, May-August 2017.
- **Research Assistant,** "Grid Resolution Study of a Modern Numerical Lifting-Line Model," **AeroLab Start-up Funding**, February 2016-April 2017.

Grant Proposal Authorship & Support

- "Energy-Directable Adaptive Control of Self-Excited Aeroelastic Systems," **NSF Cyber-Physical Systems Program**, February 2022, *Proposal Authorship & Review Support.*
- "The Future of Human Governance: Assessing Chinas Ambitions to Proliferate AI-Enabled Authoritarianism," **DoD Minerva Research Initiative**, September 2021, *Proposal Research & Authorship Support.*
- "Establishing Cooperative Flight Control Over a Range of Flight Conditions," **2020 NASA Fellowship Activity**, February 2021, *Project Consultant & Proposal Writing Support.*
- "Bridging the Cyber-Physical Boundary Between Aeroelasticity and Learning-Driven Adaptive Control," **NSF Cyber-Physical Systems Program**, June 2020, *Proposal Research & Authorship Support.*
- "Continuous Flight Optimization of Morphing-Wing Aircraft," **2018 NASA Fellowship Activity**, February 2018, *Primary Author.*
- "Flight Optimization and Control of Morphing Wing Aircraft," **ASEE National Defense Science & Engineering Graduate Fellowship (NDSEG)**, December 2017, *Primary Author.*
- "Flight Optimization and Control of Morphing Wing Aircraft," **NSF Graduate Research Fellowship Program (GRFP)**, October 2017, *Primary Author.*
- "Continuous Flight Optimization of Morphing-Wing Aircraft," **2018 NASA Space Grant Consortium**, May 2018, *Primary Author.*
- "Determining Optimal Lift Distributions for Wings with General Planform and Arbitrary Weight Distribution," **2017 NASA Space Grant Consortium**, May 2017, *Primary Author.*
- "Grid Resolution for Numerical Lifting-Line Method," **2016 NASA Space Grant Consortium**, May 2016, *Primary Author.*

PUBLICATIONS

Journal Publications (6 published, 1 accepted, 2 in preparation)

9. **Taylor, J. D.** and Hunsaker, D. F., "Effects of Wing Morphing on Fuel Consumption along Fuel-Optimal Cruise Trajectories," ***in preparation***.
8. **Taylor, J. D.**, "China's Motivations for Hypersonic Weapons Development," ***in preparation***.
7. **Taylor, J. D.**, "Contextualizing Russia's Hypersonic Threat: Motivations, Threat Perceptions, and Strategic Stability," *Space & Defense*, Vol. 13, No. 1, ***accepted***.
6. **Taylor, J. D.**, "Deterring Russian Nuclear Threats with Low-Yield Nukes May Encourage Limited Nuclear War," *Journal of Advanced Military Studies*, Special Issue on Strategic Culture, January 2022. (doi:10.21140/mcuj.2022Istratcul012)
5. **Taylor, J. D.**, and Hunsaker, D. F., "Comparison of Theoretical and Multi-Fidelity Optimum Aerostructural Solutions for Wing Design," *Journal of Aircraft*, Articles in Advance, 6 September 2021. (doi:10.2514/1.C036374)
4. **Taylor, J. D.**, and Hunsaker, D. F., "Low-Fidelity Method for Rapid Aerostructural Optimization and Design-Space Exploration of Planar Wings," *The Aeronautical Journal*, Vol. 125, No. 1289, July 2021, pp. 1209-1230. (doi:10.1017/aer.2021.14)
3. **Taylor, J. D.**, and Hunsaker, D. F., "Minimum Induced Drag for Tapered Wings Including Structural Constraints," *Journal of Aircraft*, Vol. 57, No. 4, July-August 2020, pp. 782-786. (doi:10.2514/1.C035757)
2. Phillips, W. F., Hunsaker, D. F., and **Taylor, J. D.**, "Minimising Induced Drag with Lift Distribution, Wingspan, and Wing Weight," *The Aeronautical Journal*, Vol. 124, No. 1278, August 2020, pp. 1208-1235. (doi:10.1017/aer.2020.24)
1. Hunsaker, D. F., Pope, O. D., **Taylor, J. D.**, and Hodson, J., "Aerodynamic Centers of Arbitrary Airfoils Below Stall," *Journal of Aircraft*, Vol. 56, No. 6, November 2019, pp. 2158-2171. (doi:10.2514/1.C035579)

Meeting Publications (9 published, 2 in review)

11. **Taylor, J. D.** and Hunsaker, D. F., "Effects of Wing Morphing on Fuel Burn along Fuel-Optimal Trajectories," AIAA SciTech 2023 Forum, National Harbor, MD, 23-27 January 2023, submitted 1 June 2022, ***in review***.
10. Schoenfeld, J., **Taylor, J. D.**, and Hunsaker, D. F., "Multi-Fidelity Predictions for Control Allocation on the Common Research Model to Minimize Drag," AIAA SciTech 2023 Forum, National Harbor, MD, 23-27 January 2023, submitted 1 June 2022, ***in review***.
9. **Taylor, J. D.**, "Deterring Russian Nuclear Threats with Low-Yield Nukes May Encourage Limited Nuclear War," Strategic Cultures & Collective Mentalities Virtual Conference, 28-29 June 2021.
8. **Taylor, J. D.**, "Contextualizing Russia's Hypersonic Threat: Motivations, Threat Perceptions, and Strategic Stability," US Strategic Command Academic Alliance Virtual Conference, 30 March-1 April 2021.
7. **Taylor, J. D.**, and Hunsaker, D. F., "Comparison of Theoretical and High-Fidelity Aerostructural Solutions," AIAA 2021-0067, AIAA SciTech 2021 Virtual Forum, 11-15 & 19-21 January 2021. (doi:10.2514/6.2021-0067)
6. **Taylor, J. D.**, and Hunsaker, D. F., "Characterization of the Common Research Model Wing for Low-Fidelity Aerostructural Analysis," AIAA 2021-1591, AIAA SciTech 2021 Virtual Forum, 11-15 & 19-21 January 2021. (doi:10.2514/6.2021-1591)
5. **Taylor, J. D.**, and Hunsaker, D. F., "Numerical Method for Rapid Aerostructural Design and Optimization," AIAA 2020-3175, AIAA Aviation 2020 Virtual Forum, 15-19 June 2020. (doi:10.2514/6.2020-3175)
4. **Taylor, J. D.**, and Hunsaker, D. F., "Minimum Induced Drag for Tapered Wings Including Structural Constraints," AIAA 2020-2113, AIAA SciTech 2020 Forum, Orlando, FL, 6-10 January 2020. (doi:10.2514/6.2020-2113)

3. Phillips, W. F., Hunsaker, D. F., and **Taylor, J. D.**, "Minimizing Induced Drag with Weight Distribution, Wingspan, Lift Distribution, and Wing-Structure Weight," AIAA 2019-3349, AIAA Aviation 2019 Forum, Dallas, TX, 17-21 June 2019. (doi:10.2514/6.2019-3349)

2. Abdel-Motaleb, S., **Taylor, J. D.**, and Hunsaker, D. F., "Optimization of Lift Distribution for Minimum Induced and Parasitic Drag Component," AIAA 2019-2120, AIAA SciTech 2019 Forum, San Diego, CA, 7-11 January 2019. (doi:10.2514/6.2019-2120)

1. **Taylor, J. D.**, Hunsaker, D. F., and Joo, J. J., "Numerical Algorithm for Wing-Structure Design," AIAA 2018-1050, AIAA SciTech 2018 Forum, Kissimmee, FL, 8-12 January 2018. (doi:10.2514/6.2018-1050)

Theses, Dissertations, & Capstone Publications (2 published, 1 in review)

3. **Taylor, J. D.**, "Multidisciplinary Reference Solutions for Performance-Optimized Aircraft Wings with Tailored Aerodynamic Load Distributions," Ph.D. Dissertation, Utah State University, Logan, UT, 2022, Submitted 14 June 2022, ***in review***.

2. **Taylor, J. D.**, "Strategic Considerations for U.S. Missile-Defense Development Based on Russian, Chinese Hypersonic Motivations," CAI Capstone Report, Space Dynamics Laboratory/Utah State University, Logan, UT, 13 September 2021.

1. **Taylor, J. D.**, "Methods for the Aerostructural Design and Optimization of Wings with Arbitrary Planform and Payload Distribution," M.S. Thesis, Utah State University, Logan, UT, 2017.

INVITED PRESENTATIONS

· "Hypersonic Delivery Systems," 1st National Intelligence University/National Counterproliferation Center Conference on Weapons of Mass Destruction, *Invited Panelist*, Virtual, 7 June 2022.

· "The Role of Hypersonic Weapons in Russian and Chinese Military Thought," National Intelligence University, *Invited Seminar*, Virtual, 25 March 2022.

· "Report on NASA Fellowship Activity 2018-2021," Utah State University AeroLab, *Review with Air Force Research Laboratory*, Virtual, 26 August 2020.

· "Continuous Flight Optimization of Aircraft Wings with Wing Shaping Controls," Advanced Controls and Evolvable Systems Group, NASA Ames Research Center, *Invited Seminar*, Virtual, 13 August 2020.

· "Recent Developments in Aerodynamic and Structural Optimization Techniques for Morphing Wings," Air Force Research Laboratory, *Summer Faculty Fellow Seminar*, Wright-Patterson AFB, 15 August 2017.

TEACHING & ACADEMIC SUPPORT

Utah State University, Flight Mechanics, *Senior Elective Course* (3 Semesters - Teaching Assistant)

· Recorded nearly 6 hours of online instructional material on aircraft performance.
· Planned and taught 9 lectures on aircraft stability and trim.
· Provided individualized tutoring and support to more than 90 senior-level undergraduate students.

Utah State University, Potential Flow, *Graduate Course*

· Prepared and taught 3 class lectures on elementary potential flows.

Utah State University, Center for Anticipatory Intelligence, *Minor & Graduate Certificate* (2 Semesters - Cabinet)

· Served as peer mentor to incoming students in developing writing skills and project ideas.
· Helped plan and carry out simulations, speaker events, symposia, and student socials.
· Advanced student and professional recruitment of new students.
· Assisted in the development of program curriculum.

SERVICE & INVOLVEMENT

Reviewer of manuscripts in the following journals:
Journal of Aircraft
Part G: Journal of Aerospace Engineering

USU Center for Anticipatory Intelligence Graduate Mentor	*Aug 2021- Present*
Utah Representative, AIAA U.S. Congressional Visits	*Mar 2021 & 2022*
USUSA Engineering Council Graduate Student VP	*Aug 2017 - May 2018*
USUSA Graduate Student Council Member	*Aug 2017 - May 2018*
USU College of Engineering Student Mentor	*Aug 2017 - May 2018*

HONORS & AWARDS

NASA ARMD Fellowship	*Sept 2018-May 2022*
College of Engineering Outstanding PhD Scholar	*Mar 2021*
Mechanical & Aerospace Engineering Department PhD Researcher of the Year	*Mar 2021*
Utah State University Achievement of the Year Robins Award	*Apr 2019*
Outstanding Masters Student Researcher Departmental Award Nominee	December 2017
Utah State University Presidential Scholar	*Aug 2010 - May 2016*
WAC/MWC Academic All-Conference Track & Field	*Jan - May 2011 & 2014*
Whitesides Scholar Athlete	*May 2011*
Robert C. Byrd Scholar	*Aug 2010 - May 2011*

Reference Style and Format

Text Citations

References must be limited to readily accessible published material; i.e., those available from libraries, databases, and other public sources. All references must be numbered and cited in numerical order in the text. The list should be a complete and comprehensive representation of available literature, reflecting the state of the art as it pertains to the scope of the paper. Authors must reference the original source of a work, not a secondary source. Classified or export-restricted references, private communications, personal websites, and websites where there is no commitment to archiving may be mentioned parenthetically in the text or in a footnote but should not be cited in the reference list. A reference that is not reliably available is of no use to readers.

Each numbered reference citation in the text is enclosed in brackets as in the following examples:

It is shown by Smith [4] that ...
The effect of ... should be taken into account [5].
For example, see Refs. [6, 7].
Further documentation can be found in [8-10].

On the rare occasion when a specific page number or page range needs to accompany a citation within the text, include it within the brackets in the following manner:

This procedure was proposed by Gelb [11, p. 250]
The solution can be found in Rogers [12, pp.14-18].

When multiple author names are cited in the text, list up to two authors, as in “Walsh and Jones,” but use “et al.” with three or more authors, as in “Walsh et al.”

List of References

To avoid production delays, all references must be complete prior to acceptance of a manuscript. The following list gives examples of commonly lacking information:

- All authors to a reference (do not use “et al.” in the reference list)
- Full journal or book titles and publisher names (no abbreviations)
- Inclusive page numbers for journal articles and page or chapter numbers for books, when appropriate
- Journal volume and issue numbers (or months)
- Locations of report sponsors (e.g., companies and universities)
- Complete publishing information for proceedings (i.e., the same as for other books)

A URL that includes the DOI (Digital Object Identifier) should be incorporated into every reference for which it is available. A DOI is an identifier of intellectual property entities on digital networks. DOIs provide a system for persistent identification and interoperable exchange of managed information, and they are commonly assigned to journal articles, ebooks, research reports, and data sets. A DOI forms a permanent URL that begins https://doi.org/ and is the preferred link; see examples below for how to include a URL in a reference. For more information on DOIs and their value, visit **www.doi.org (http://www.doi.org/)** or **www.crossref.org (http://www.crossref.org/)**.

For **all** references: Spell out **everything** except AIAA, NASA, NACA, AGARD, and NATO; months may be abbreviated.

Follow these examples for format; double-space the reference list in your manuscript:

Chapter in a Book

[1] Turner, M. J., Martin, H. C., and Leible, R. C., “Further Development and Applications of Stiffness Method,” *Matrix Methods of Structural Analysis*, 1st ed., Vol. 1, Wiley, New York, 1963, pp. 6–10.

Journal Article

[2] Johnson, J. E., Lewis, M. J., and Starkey, R. P., “Multi-Objective Optimization of Earth-Entry Vehicle Heat Shields,” *Journal of Spacecraft and Rockets*, Vol. 49, No. 1, 2012, pp. 38–50. [Note: Month acceptable if number is not available.]
https://doi.org/xx.xxxx/x.xxxxx

[Note that in the URL the doi.org prefix precedes the DOI generated by Crossref, with the publisher/content identifiers represented by the x's.]

or (if published only electronically and not yet available in a particular journal issue):

[3] Alyanak, E. J., and Pendleton, E., "Aeroelastic Tailoring and Active Aeroelastic Wing Impact on a Lambda Wing Configuration," *Journal of Aircraft*, published online 10 Nov. 2016.

https://doi.org/xx.xxxx/x.xxxxx

[4] von Hippel, T., "Contribution of White Dwarfs to Cluster Masses," Astronomical Journa.l, Vol. 114, No. 4, 1998, pp. 1536–1542.
http://www.journals.uchicago.edu/AJ/journal/contents/v115n4.html

[Note that a direct URL link may not include DOI data.]

Journal Article Not Yet Published

[5] Devasia, S., and Lee, A., "Scalable Low-Cost Unmanned-Aerial-Vehicle Traffic Network," *Journal of Air Transportation* (not yet published).

Update the publication status of unpublished works at the author proof stage.

Proceedings/Transactions Articles

[6] Blottner, F. G., "Prediction of Electron Density in the Boundary Layer of Entry Vehicles with Ablation," *The Entry Plasma Sheath and Its Effects on Space Vehicle Electromagnetic Systems*, NASA SP-252, Vol. 1, Oct. 1970, pp. 219–240.

[7] Wirin, W. B., "Space Debris 1989," *Proceedings of the Thirty-Second Colloquium on the Law of Outer Space*, AIAA, Washington, DC, 1990, pp. 184–196.

Company or NASA Report

[8] Bhutta, B. A., and Lewis, C. H., "PNS Predictions of External/Internal Hypersonic Flows for NASP Propulsion Applications," VRA, Inc., VRA-TR-90-01, Blacksburg, VA, June 1990.

[9] Miner, E. W., and Lewis, C. H., “Hypersonic Ionizing Air Viscous Shock-Layer Flows over Nonanalytic Blunt Bodies,” NASA CR-2550, May 1975.

Meeting Paper

[10] Neifeld, A., and Ewert, R., “Jet Mixing Noise from Single Stream Jets using Stochastic Source Modeling,” AIAA Paper 2011-2700, June 2011.
https://doi.org/xx.xxxx/x.xxxxx

[11] Hobbs, D. E., “Experimental Investigation of Compressor Cascade Wakes,” American Society of Mechanical Engineers, Paper 82-GT-299, April 1982.

Update meeting paper references if they have been published in a journal.

AIAA Book Series

[12] Sutton, K., “Air Radiation Revisited,” *Thermal Design of Aeroassisted Orbital Transfer Vehicles*, edited by H. F. Nelson, Vol. 96, Progress in Astronautics and Aeronautics, AIAA, New York, 1985, pp. 419–441.

Electronic Databases

[13] Tseng, K., “Nonlinear Green’s Function Method for Transonic Potential Flow,” Ph.D. Dissertation, Aeronautics and Astronautics Dept., Boston Univ., Boston, MA, 1983, ProQuest Ebrary.

[14] Vickers, A., “10–110 mm/hr Hypodermic Gravity Design A,” Rainfall Simulation Database, retrieved 15 March 1998.
http://www.geog.le.ac.uk/bgrg/lab.htm (http://www.geog.le.ac.uk/bgrg/lab.htm)

[Include access or retrieval date when an online source is not dated.]

Anonymous Report

[15] “Equations, Tables, and Charts for Compressible Flow,” NACA Rept. 1135, 1953. [Note: Include month if available.]

American Institute of Aeronautics and Astronautics
12700 Sunrise Valley Drive, Suite 200
Reston, VA 20191-5807
800-639-AIAA (2422)

THE AERONAUTICAL JOURNAL

ROYAL AERONAUTICAL SOCIETY

Cambridge University Press Style Guide: For Chapter 3

The Aeronautical Journal – Instructions for Authors

The Review Process

All stages of the peer review process for *The Aeronautical Journal*, from receipt to approval, are carried out via the online Editorial Manager system. This is a familiar software platform across academic publishing which is user-friendly and intuitive.

The review process typically lasts 3-6 months depending on how quickly reviews are obtained and authors can revise their papers.

When a paper is received, it is assigned to an appropriate Associate Editor who invites a minimum of two referees to review the manuscript. Once two reviews have been obtained, the referees' comments are sent to the corresponding author and, if changes are recommended, they are invited to revise the paper as suggested, unless it is rejected in the first instance.

The authors should upload their revised paper with highlighted or tracked changes and include a separate list of changes or a rebuttal against each point which has been raised. This will enable the referees to see easily where changes have been made and speed up the review process.

The revised manuscript will be sent back to the original referees. If the referees suggest further improvements to the paper, these will be passed back to the authors and the paper revised again. If the Associate Editor feels, having considered the second reviews, that the authors have not responded adequately to the original reviews of the referees, then the paper will be rejected. Thus, it is essential that all comments are addressed properly by authors. A third referee may be approached if the Associate Editor feels this is appropriate. The Editor ultimately reserves the right to reject a paper on grounds of quality or lack of co-operation from authors.

A paper may also be rejected immediately without review if the subject of a paper is deemed outside the remit of *The Aeronautical Journal,* there is an obvious lack of novelty or the quality of English grammar and spelling is below an acceptable level.

Once a paper is accepted, the authors will be invited to upload the final files (the approved version of the text and individual files for any accompanying figures and tables) on to Editorial Manager.

New submissions

Manuscripts should be submitted by authors via Editorial Manager at www.editorialmanager.com/aeroj. They are accepted for consideration on the strict condition that they are based on original research and have not previously been published or submitted for publication to another published journal. Research papers submitted at conferences, and unpublished, would be considered for publication.

The receipt of papers will be acknowledged by email with a reference number which should be used in all correspondence.

Submission form

When submitting a manuscript, authors will be required to complete an online submission form. This includes:

Full title: The title should be kept short and concise.

Article type: This will be Regular or Survey. Most papers will be regular papers. The purpose of a survey paper (sometimes known as a review paper) is to succinctly review recent research in a particular area and summarise the current state of knowledge of the topic.

Keywords: A minimum of two keywords describing the main subject area(s) covered by the paper should also be given. This will appear in the published paper.

Corresponding author: This is the author responsible for handling correspondence during the review and production stages. Full name, institution and email address are required. Correspondence will be through email. If the corresponding author changes or their email address changes during the process, then they should update their contact details on Editorial Manager or inform the publications staff at aerojournal@aerosociety.com.

Order of authors: Names should be presented in the order they should appear on the published paper. Each author's organisation to which they are associated should be included with accompanying address. Please note the authors' organisations and email addresses will be published unless requested not to.

Abstract: An abstract of around 150 words is required which summarises the paper and contains no references.

Suggested reviewers: The author will be invited to suggest some appropriate referees and a suitable Associate Editor if they wish, though this is not mandatory. A list of Associate Editors can be found on *The Aeronautical Journal* website at Editorial board (cambridge.org)

The Editor reserves the right not to use the suggested Associate Editors or referees and referees' identities are not revealed during or after the review process.

Opposed reviewers: The author can name referees they would prefer not to be used, giving a reason, eg conflict of interest, previous dispute. This section is not mandatory.

Supplementary material: Supplementary material may be added, such as additional images, extensive data sets or filmed footage. Authors are encouraged to make use of this facility for any supporting material which is not essential to the main argument of their articles.

Special issues: *The Aeronautical Journal* publishes special issues from time to time. If the author has been contacted to submit their paper for a special issue they should specify the name of the special issue.

Open access: The 'green' version of Open Access is offered to all authors of approved papers by default for no fee. This allows *The Aeronautical Journal* to publish the paper exclusively under licence for six months, a period in which only it can present the paper to the general public. In this period the author is also allowed to host a non typeset version of the paper on the website of the university or organisation at which they work, research or teach.

Alternatively, authors can opt for the 'gold' version of Open Access which allows the paper to be made available either in print or online to the general public immediately after it is first published with no delay. Costs can be found here:

https://www.cambridge.org/core/services/open-access-policies/open-access-journals/hybrid-open-access-faqs

Manuscript guidelines

English: Papers must be written in British English (not American English).

Authors, particularly those whose first language is not English, are encouraged to have their manuscripts critically read by a native English speaker before submission. This may help to ensure that the academic content of the paper is fully understood by the Associate Editor and reviewers. A list of third-party services specialising in language editing and/or translation is available at: https://journals.cambridge.org/action/stream?pageId=8728&level=2&menu=Authors&page Id=3608

Please note that the use of any of these services is voluntary and at the author's own expense. Use of these services does not guarantee that the manuscript will be accepted for publication, nor does it restrict the author to submitting to a journal published by Cambridge University Press.

File types: For authors using MS Word, please note that equations must NOT be converted to picture format and the file must be saved with the option 'make equation editable'. When using special character sets a PDF should be made from the final accepted version of the Word Doc.
Authors using LaTeX are strongly encouraged to compose their papers, using the AER.cls style _file and supporting files provided with the AER-instructions.tex file serving as a template. A PDF of the LaTeX file should then be generated and submitted via the submission site to accompany the LaTeX files. For initial submission there is no need to submit the LaTeX source file alongside the PDF, but upon acceptance of the paper, the LaTeX source file, along with individual figure files and a PDF of the final version, will need to be submitted for typesetting purposes.

ORCID: *The Aeronautical Journal* strongly encourages all corresponding authors to identify themselves using ORCID when submitting a manuscript to the journal. ORCID provides a unique identifier for researchers and, through integration in key research workflows, such as publication and grant applications, provides the following benefits:

1. Discoverability: ORCID increases the discoverability of your publications, by enabling smarter publisher systems and by helping readers to reliably find work that you have authored.

2. Convenience: As more organisations use ORCID, providing your ID or using it to register for services will automatically link activities to your ORCID profile, and will save you re-keying information multiple times.

3. Keeping track: Your ORCID profile is a neat place to record and display (if you choose) validated information about your research activities.

If you do not already have an ID, you can create one during submission to *The Aeronautical Journal.* You can register for one directly from your user account on Editorial Manager or via https://orcid.org/register. If you already have an ID, please use this when submitting by linking it to your Editorial Manager User account.

Permissions and copyright: Authors must also obtain permission, where necessary, to use any material in a paper which is copyright or the property of any other persons or entity, including their employers. Any fees incurred are the sole responsibility of the author(s). Please note that *The Aeronautical Journal* is marketed and sold internationally.

The policy of The *Aeronautical Journal* is that authors (or in some cases their employers) retain copyright and grant Cambridge University Press a licence to publish their work. In the case of 'gold' Open Access articles this is a non-exclusive licence. Authors must complete and return an author publishing agreement form as soon as their article has been accepted for publication; the journal is unable to publish without this.

Fees: There are no page charges for *The Aeronautical Journal* – please see Open Access.

Colour*:* All papers appear in colour on *The Aeronautical Journal* website.

Format: The document should be in single column format.

Length: Papers should preferably not exceed 10,000 words.

Nomenclature: Papers should contain a nomenclature. A list of all symbols and abbreviations used in the text and figures, whether familiar or not, should be given in alphabetical order with, for example, c before C and all English letters listed before Greek symbols. Subscripts and superscripts should be listed separately where possible. SI units are preferable.

Introduction: Discuss the raison d'être of the work, including previous work by others and how the work presented aims to advance or complement this.

Equations: Equations must be numbered in brackets ...(1). Each equation should be produced electronically in Word preferably using either Equation Manager or MathType. Variables should be in italics. Constants should be in plain text. Vectors and matrices should be in plain text but bold. Cos, sin, tan should begin in lower case and be in plain text.

Conclusions: This section should be very concise and bullet points are recommended for clarity. The degree to which the aims have been achieved should be portrayed clearly to the reader. Suggestions for future work or comments on work in progress are encouraged.

References: References should be numbered sequentially in the text as they occur and placed at the end of the manuscript. The examples below show how to format article(1), report(2), review(3) and book(4) references. Authors' names should be in small Caps. DOI numbers should be included with references if possible.

Article

1. MILLER, P. and WILSON, M. Wall jets created by single and twin high pressure jet impingement, *Aeronaut J, March 1993, 97, (963), pp 87-100.*

Report

2. GREEN, J.E., WEEKS, D.J. and BROOMAN, J.W.F. Prediction of turbulent boundary layers and wakes in compressible flow, ARC R&M No 3791, 1979.

Review

3. VINCENTY, M. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations, 1975, Technical Report XXIII, No 176, Survey Review.

Books

4. KING-HELE, D. Satellite Orbits in an Atmosphere, Blackie, Glasgow, UK, 1987.

Appendices: If no suitable reference is available appendices may be used to clarify certain points, such as a step in the theoretical analysis. Appendices may be hosted online as supplementary material if they are not a core part of the submitted paper.

Tables: Tables should have a number and a caption. Each table should be cited in numerical order in the text. See table examples below.

Table 1
Smart materials and their characteristics[13]

Material	Max strain (%)	Max. stress (MPa)	Elastic energy density (J/g)	Max. effic. (%)	Relative speed
Electrostrictor Polymer P (VDF-TrFE)	4	15	0.17	–	Fast
Piezoelectric Ceramic (PZT)	0.2	110	0.013	>90	Fast
Single Crystal (PZN-PT)	1.7	131	0.13	>90	Fast
Polymer (PVDF)	0.1	4.8	0.0013	n/a	Fast
SMA (TiNi)	>5	>200	>15	<10	Slow

Table 2
Derivatives and standard errors (clean)

$\hat{\theta}$	Linmod	Dif	$s(\hat{\theta})$	Rec	$s(\hat{\theta})$
z_w	–4.139	–4.173	0.020	–4.115	0.009
z_q	24.33	24.44	0.091	24.27	0.027
z_η	–2.361	–1.726	0.253	–2.352	0.072
m_w	–4.289	–4.420	0.005	–4.427	0.008
m_q	–6.035	–6.237	0.015	–6.311	0.025
m_η	–32.54	–33.27	0.038	–33.30	0.064

Table 3
Derivatives and standard errors (noise)

$\hat{\theta}$	Linmod	Dif	$s(\hat{\theta})$	Rec	$s(\hat{\theta})$
z_w	–4.139	–4.489	0.174	–3.594	0.297
z_q	24.33	24.17	0.599	23.80	0.622
z_η	–2.361	–2.835	1.327	–3.163	1.419
m_w	–4.289	–4.844	0.250	–4.296	0.389
m_q	–6.035	–2.110	0.935	–3.179	1.081
m_η	–32.54	–20.58	2.315	–21.44	2.462

Table 4
Frequencies (rad/s) and damping ratios

Method	ω_{sp}	ω	ζ
Linmod	11.37	10.17	0.4473
Differentiated	11.57	10.34	0.4495
Differentiated noise	11.25	10.75	0.2932
Reconstructed	11.55	10.30	0.4513
Reconstructed noise	10.66	10.11	0.3176

Figures: Text files of the manuscript should clearly state the preferred position of the figures. All figures must be cited within the manuscript and for users of Word, figure captions should be included at the end of the manuscript after the references. References to figures in the text should be referred to as: Fig. 1, Fig. 2 and Fig. 3 or Figure 1 if at the start of a sentence. Figures in an appendix must be numbered separately from those in the main text, eg 'Fig. A1'. To ensure that figures are reproduced to the highest possible standards, Cambridge University Press recommends the following formats and resolutions for supplying electronic figures:

Photographs

Formats: TIFF, EPS, JPEG or PN
Colour mode: RGB
Resolution: 200 dpi

Tables, graphs and lineart

Format: EPS, JPEG or PNG
Colour mode: RGB or grayscale
Resolution: 150dpi

Authors should, where necessary, obtain permission for online reproduction. Publications staff can be consulted if there are any queries about this. For full instructions on how to prepare figures go to: http://journals.cambridge.org/action/stream?pageId=7848&level=2&menu=Authors&pageId=3608]

Proofs

Authors are sent proofs for the purposes of checking and making typographical and minor changes only. All final changes are at the Editor's discretion. Any corrections made against journal style will not be made. The Editor of *The Aeronautical Journal* reserves the right to publish a paper after just one set of corrections.

PDF of published version

Authors will receive a link to a PDF file of their final typeset article upon publication. This version can not be displayed or published anywhere by the author. Only the submitted final Word version of the final approved version should be used by the author after six months to publish elsewhere.

Please forward any questions or queries to: aerojournal@aerosociety.com