Massachusetts Institute of Technology
Department of Aeronautics and Astronautics

Ph.D. Proposal Document

Advancing the Usability, Interpretability, and Practicality of Conceptual Aircraft Design Optimization with Code Transformations

Ph.D. Candidate:
Peter Sharpe

Committee:
Professor R. John Hansman (Chair), MIT
Professor Mark Drela, MIT
Professor Karen Willcox, University of Texas at Austin

External Evaluator:
Dr. Tony Tao, MIT Lincoln Laboratory

December 11, 2023

Abstract

Design optimization has immense potential to improve aircraft conceptual design, yet practical industry adoption of advanced methods remains relatively limited despite decades of academic research. This thesis takes steps to address the root causes of this gap by introducing a new paradigm for computational design optimization frameworks called *code transformations*. Code transformations encompass a variety of best-practices scientific computing strategies (e.g., automatic differentiation, automatic sparsity detection, problem auto-scaling, etc.) that automatically analyze, augment, and accelerate the user's code before passing it to a modern gradient-based optimization algorithm.

This paradigm offers a compelling combination of ease-of-use, computational speed, and modeling flexibility; existing paradigms typically make sacrifices in at least one of these key areas. Because of this, code transformations offer a competitive avenue for increasing the adoption of advanced optimization techniques in industry, all without requiring significant expertise in applied mathematics or computer science from end users.

The proposed contributions of this thesis are fivefold. First, the thesis will introduce code transformations conceptually and demonstrate their feasibility through aircraft design case studies. Second, several common aircraft analyses will be newly implemented in a traceable form compatible with code transformations. Third is a novel technique to automatically trace sparsity through some kinds of external black-box functions by exploiting IEEE 754 handling of not-a-number (NaN) values. Fourth, strategies for efficiently incorporating black-box models into a code transformation framework through physics-informed machine learning surrogates are proposed; this will be demonstrated with an airfoil aerodynamics analysis case study. Finally, the thesis will show how a code transformations paradigm can simplify the formulation of other optimization-related aerospace tasks outside of design; here, an example of aircraft system identification and performance reconstruction from minimal flight data will be given. Taken holistically, these contributions aim to improve the accessibility of advanced optimization techniques for industry engineers, making large-scale conceptual multidisciplinary design optimization more practical for real-world systems.

Contents

Chapter 1

Introduction

The promise of multidisciplinary design optimization (MDO) to revolutionize engineering design has motivated pioneering research into new frameworks and methods for decades. MDO saw rapid progress beginning in the late 1980s, with early works outlining visions for integrated computational design tools that couple analyses across engineering disciplines [1–3]. However, contemporaneous warnings also highlighted the limited adoption of these methods by industry practitioners up to that point [4, 5].

Surveys assessing the state of aircraft MDO years later reveal a persisting gap between MDO's promise and practice. Documented examples of complete aircraft designed with MDO remain relatively uncommon in industry, even as exponential growth in computational power has mitigated other technical barriers. Expert commentary affirms this observation, noting that the use of advanced optimization techniques often stops at the academic proof-of-concept stage [1, 3, 4, 6–8].

The core remaining barriers to widespread adoption of conceptual-level MDO relate strongly to practical limitations at the human-optimizer interface, rather than purely on traditional metrics of computational speed [9]. The complexity intrinsic to coordinating numerous coupled models across disciplines poses inherent difficulties for design interpretation, method credibility, and managing tradeoffs [10]. Modern scientific computing capabilities enable unprecedented problem scale, but push human limitations in complexity management. The central difficulty is leveraging advanced techniques in scientific computing – which offer enormous performance improvements – without requiring end-users to be joint experts in applied mathematics and computer science on top of their engineering domain expertise [11].

The path forward lies in a deliberate focus on the practical human interface with optimization tools. Technical advances must synthesize with pragmatic methods (e.g., syntax) to address open challenges around complexity and usability. The overarching motivation

is to address this industry gap, allowing the benefits of advanced conceptual design optimization to be more readily accessed by industry. This grounds the proposed research directions on fundamentally human-centered principles in addition to technical ones.

1.1 Project Definition and Thesis Overview

To address the above challenges, this thesis proposes five novel contributions. These contributions are summarized below at a high level, and individually expanded upon in Chapter 3:

1. **Code Transformation Paradigm:** Conceptually introduce a new computational paradigm for MDO frameworks that offers most of the benefits of state-of-the-art paradigms with significantly fewer user frictions. Provide a proof-of-concept implementation of this paradigm in the open-source *AeroSandbox* [12] framework, and compare this implementation with existing frameworks on a set of practical aircraft design problems.

2. **Traceable Physics Models:** Provide the first implementations of several key aerospace physics models that are purpose-built to take advantage of code transformations. These aim to both stress-test the code transformation paradigm on common analysis code patterns and jump-start future applied research with a set of modular analyses.

3. **Sparsity Tracing via NaN-Propagation:** Conceptually introduce the novel idea of "NaN-propagation" as a method to trace sparsity through black-box numerical analyses by exploiting floating-point math handling. This opens up a path to including such models in a code transformations framework while still retaining some (but not all) of the speed advantages over current black-box optimization methods.

4. **Physics-Informed Machine Learning Surrogates for Black-Box Models:** Explore strategies to incorporate black-box models into a code transformation framework, expanding the practicality of the paradigm for users who require use of black-box code. As an example, a physics-informed machine learning surrogate model for airfoil aerodynamics analysis will be presented. This demonstrates that accurate surrogates can be constructed to stand in for complex analyses while retaining compatibility with code transformations, and without sacrificing mathematical flexibility.

5. **Aircraft System Identification from Minimal Sensor Data:** Demonstrate that the code transformation paradigm enables straightforward formulation of optimization

problems beyond just design optimization. As an example, the thesis will show an application to aircraft system identification and performance reconstruction from minimal flight data. Here, physics-based corrections are used alongside statistical inference techniques to accurately estimate aircraft performance characteristics from a single short test flight. This approach is enabled by the flexibility of the code transformation framework.

1.2 Proposal Document Organization

The latest doctoral handbook for the MIT Department of Aeronautics and Astronautics [13] requires the following elements in a PhD proposal, which, for reader convenience, are mapped to specific portions of this document:

Table 1.1: Requirement-to-section mapping for the PhD proposal document.

Proposal Document Requirement	**Section**
A clear, specific statement of the technical problem and the objectives of the proposed research	Chapter 1
A thorough, adequately referenced, summary of previous work done on the problem	Chapter 2
A plan for the initial approach to the problem	Chapter 3
An outline of the major foreseeable steps to a solution of the problem	Chapter 3 & Section 4.3
An estimate of the time that might be required	Section 4.3
A list of the facilities needed	Section 4.4

Chapter 2

A Review of Aircraft Design Optimization

2.1 Early Promises, Predictions, and Pitfalls

> *"When an aircraft designer hears that a new program will use multidisciplinary optimization, the reaction is often less than enthusiastic. Over the past 30 years, aircraft optimization at the conceptual and preliminary design levels has often yielded results that were either not believable, or might have been obtained more simply using methods familiar to the engineers. Even 5 to 20 years ago, actual industry application of numerical optimization for aircraft preliminary design was not widespread."*
>
> -Ilan Kroo, 1997 [4]

These are the opening lines of a landmark 1997 paper that reviewed the state of the art and future directions in the then-nascent field of aircraft multidisciplinary design optimization (MDO) [4]. The paper, by aircraft designer and MDO pioneer Ilan Kroo, not only reviewed the status of the field's academic research, but also took the key step of assessing whether these research advances had translated to practical industry impact. As a later review by an MDO technical organizing committee would emphasize, "the ultimate benchmark of a research field's impact is indicated by the realization of its theories into successful products throughout industry." [7]

Kroo's assessment of the field is generally optimistic, though he also hedged this with some important reservations. On one hand, he notes the auspicious progress of aircraft MDO research during the preceding decades by all traditional metrics: problem size, analysis fidelity, runtime speed, and so on. He credits these successes largely to both algorithmic advances and the exponential growth of computational power over time (Moore's

law[1]). Extrapolating these trends forward, he concludes that the field is poised to make a significant impact on the aircraft design process across both academia and industry. The promise of MDO is made clear in a remark that many aircraft designers would agree with today: "In a very real sense, preliminary design is MDO." [4]

On the other hand, Kroo notes that actual industry applications of aircraft MDO remained conspicuously limited as of the paper's 1997 publication, and "many difficulties remain in the routine application of MDO." [4] Other works from the early days of aircraft MDO corroborate this dearth of industry adoption. For example, in a 1982 AIAA lecture titled *On Making Things the Best – Aeronautical Uses of Optimization*, optimization advocate Holt Ashley notes the "keen disappointment felt by many [optimization] specialists because their theories have received so little practical application." [1] Ashley goes further by conducting an exhaustive industry-wide survey to identify "successful practical applications" of aircraft design optimization. The survey received "overwhelming" industry interest and encouragement; however, "the yield of examples which met the criterion of having been incorporated into a vehicle that actually operates in the Earth's atmosphere was painfully, perhaps shockingly small."

Perhaps the reason why Ashley's disappointment was so poignant is that, even at the time, aircraft designers in industry widely recognized the immense potential of MDO tools to fundamentally transform engineering design workflows. As two Lockheed engineers stated in 1998 [14], "The technical community knows the power of MDO and not having a cradle-to-grave example has been a continual source of frustration, as voiced by AIAA MDO technical committee members [for] years." Similarly, a 2002 Boeing paper identifies MDO as one of four key technologies poised to define the next generation of aircraft design[2] [6].

Motivated by this gap between promise and practice, Kroo and other luminaries offer several reasons for the lack of industry adoption of MDO technologies:

1. **Inadvertently-violated model assumptions:** When optimization is applied to an analysis toolchain, it acts adversarially, disproportionately seeking out the "weakest link" in the analysis chain and exploiting it. Simplified models that are acceptable in a manual sizing study are often unacceptable in an optimization study, because implied assumptions that an engineer would naturally cross-check during manual design are prone to optimizer exploitation. This can lead to unrealistic results from

[1] "Moore's law" is an empirical observation that chip transistor density (a surrogate for computational power) has tended to double roughly every two years, a trend that has held true for the past half-century.

[2] With the others being computational simulation, small uncrewed aircraft, and newly-emphasized design considerations such as environmental footprint and operations optimization.

MDO tools that degrade user trust in optimization processes. Kroo contends that the solution to this problem is to implement higher-fidelity models that account for more edge cases, an approach we explore later in Section 2.3.1.

2. **Missing models and constraints:** Critical aircraft design choices are often determined by tradeoffs spanning multiple disciplines. If an MDO tool does not incorporate relevant disciplines (e.g., it performs aerodynamic, propulsive, and structural analyses, but the true design driver is noise), then the resulting design will be fundamentally flawed with no indication to the user whatsoever. This can lead to costly design mistakes. Indeed, Kroo suggests that a flawed MDO tool is not only useless, but worse, due to the false confidence it can instill.

3. **Challenges of managing complexity:** As computational power grows, MDO tools can incorporate increasingly numerous and higher-fidelity models. To first order, when the number of models N grows, the potential number of cross-discipline couplings to manage tends to scale as $\mathcal{O}(N^2)$. Therefore, in the limit of growing computational power, the practical bottleneck is less about implementing individual analyses, but more about managing this communication overhead and architecting the optimization code itself.

Kroo primarily attributes these early barriers to practical industry adoption to the limited computational power of the era, noting: "This convergence between computational capability and computational requirements for interesting design problems is one of the reasons that MDO is considered to be such a promising technology, despite the limited acceptance of pioneering MDO efforts." A contemporaneous review by Sobieski and Haftka agreed, identifying "very high computational demands" as a "major [obstacle] to realizing the full potential of MDO". [3]

However, some works recognized that not all challenges would recede with increasing computational power. A 2002 review paper by a Boeing Technical Fellow in *Journal of Aircraft* [6] cautions: "New [MDO] strategies need to be developed...which take advantage of the assumptions and techniques that airplane designers use, rather than letting a computer churn away and come up with theoretically possible, but practically impossible, configurations." Likewise, Kroo, Sobeiski, and Haftka all cite "managing complexity" as a growing challenge of MDO [3, 4], which hints at a human-computer interface problem, rather than simply a need for more CPU cycles. Drela's aptly-titled 1998 work *Pros & Cons of Airfoil Optimization* also demonstrates a similar issue [5]. Here, Drela shows that even seemingly-simple aerospace design optimization problems will only yield practical results

if the problem formulation, assumptions, and results are all precisely understood by an expert designer. These works presciently foreshadow modern concerns around MDO interpretability and other user frictions, urging the development of human-centered MDO approaches that synthesize mathematical optimization with designer intuition.

In summary, while computational limits were a clear early barrier, foundational challenges around managing complexity and aligning optimization with real-world design constraints were already emerging. These issues would become increasingly central as MDO research rapidly expanded in scope.

2.2 A Retrospective on Aircraft Design Optimization in Industry

Current Status

With the benefit of an additional quarter-century of hindsight since the date of these early MDO assessments, we can begin to assess how these forecasts have played out. In many ways, these predictions by Kroo and others were remarkably accurate. The scale and speed of optimization problems solved today has indeed grown exponentially in the years since. This is not only due to increasing computational power, but also from algorithmic and architectural improvements in MDO and optimization more broadly[3]. Numerous high-quality aircraft MDO case studies and post-hoc design studies have been published in the years since, including the D8 transport aircraft [15, 16], the STARC-ABL aircraft [17], and the Aerion AS2 [18]. Some of these studies leverage relatively-high-fidelity models (e.g., RANS CFD) that would have been computationally-intractable for optimization studies in prior eras.

However, many of the barriers to practical industry use Kroo identified have not disappeared; to the contrary, as computational power increases, the challenges of interpretability and managing complexity ring even truer today. As a result, this gap between academic research and practical industry adoption has not closed, and in some ways is wider than ever before. A 2010 review paper [7] concludes that "the actual use of genuine MDO methods within industry at large...is still rather limited." In 2013, Hoburg and Abbeel lamented that "despite remarkable progress in MDO, the complexity and diversity of modern aerospace design tools and teams makes fully coordinated system-level optimization a monumental undertaking." [19] As recently as 2017, a team of Airbus engineers

[3] discussed later in Section 2.3

and MDO researchers concurred: "While the field of MDO techniques has tremendously grown since [the 1980s] in the scientific community, its application in industry is still often limited[...] A major challenge remains to apply MDO techniques to industrial design processes." [8]

Despite the prevalence of on-paper design studies using MDO, it remains somewhat rare to see *built-and-flown* airplanes developed with documented, significant use of MDO methodologies. Indeed, the industry norm for aircraft conceptual design remains largely unchanged: expert-driven manual design guided primarily by point analyses and parametric surveys. Here, the human designer informally fulfills an optimizer-like role, but the requirements are never explicitly translated into a formal mathematical optimization problem[4]. High-quality aircraft design case studies that implement this expert-driven manual approach successfully are those of the Joby Aviation S2 eVTOL [20] and Perdix micro-UAV [21] aircraft. Anecdotally, the *industry* conceptual designer's computational weapon of choice is still more often an Excel spreadsheet than a formalized MDO framework.

To assess the magnitude of this gulf between academic and industry use of MDO, we surveyed industry literature and design reviews to identify aircraft development programs showing three minimal criteria:

1. A design optimization problem for a complete aircraft coupling at least three disciplines (e.g., aerodynamics, structures, and propulsion) that is solved computationally.
2. Evidence that the optimization result (or at least, some insight gained from it) was used to inform the design of aircraft.
3. Evidence that the aircraft was subsequently built and flown.

It is worth pausing to discuss why this restrictive *built-and-flown* aircraft requirement is used, since it accounts for the majority of the aircraft design studies that were excluded from this survey. Successful optimization on built-and-flown aircraft forces a level of completeness, trust, and durability that may not be present in a paper study. Ashley notes that a practical MDO result "must also survive the gauntlet of ground testing, reliability, demonstration, flight verification, and the like without further special attention". [1] Myriad other practical considerations could be added to this list – certifiability, manufacturability, lifecycle costs, maintainability, robust off-design performance, and the realities of engineering culture (e.g., can the MDO tool give not only a result, but sufficient evidence that it should be believed?), to name only a few. To justify the substantial capital

[4] i.e., with a precisely specified objective, defined variables, and enumerated constraints

of building and flying a new aircraft, optimization results must withstand a much higher standard of scrutiny and reviewability than they might otherwise.

Consistent with previous surveys and reviews [1, 3, 4, 7, 8], we find that relatively few programs meet these criteria; here, we make note of some that do. The Lockheed Martin F-22 Raptor program was one of the first programs to leverage an MDO-like process for a complete, flown aircraft design, as reported by Radovcich and Layton [14]. However, the workflow documented in this 1998 work differs significantly from modern MDO processes in that it was remarkably manual – a fusion of traditional and MDO-based design processes. While some disciplines (aerodynamics, structures, and control law design) are coupled computationally, many other disciplines (low-observability, manufacturability, etc.) are coupled in by querying teams of subject-matter experts at iterates within the optimization loop itself, essentially serving as a black-box function call. When contrasted with more-typical MDO processes where all interdisciplinary communication is computational, this manual approach has some pros and cons. One one hand, intermediate optimization iterates are continuously human-reviewed, which could cause a poorly-posed problem to be identified as such more quickly. On the other hand, it also sharply increases the optimization runtime, which may take months instead of hours. Even allowing for this broader definition of MDO, however, the authors note the rarity of their experience in industry: "Documented experiences of MDO applications during the engineering, manufacturing, and design phases of fighter aircraft programs are not numerous. Documentation is even rarer for aircraft that have flown."

The Lockheed Martin X-59 Quiet Supersonic Transport (QueSST) is another program that leveraged an MDO-based design process throughout aircraft development, unifying aerodynamics, acoustics, and structural analyses for outer mold line design and composite ply scheduling [22–24]. (Although the X-59 has not yet met the "flown" criterion at the time of writing, industry reports credibly suggest this is imminent with no remaining obvious barriers.) The X-59 program builds upon several decades of successful MDO research for sonic-boom-minimization problems, a topic where computational shape optimization has proven particularly useful due to the non-intuitive and sensitive nature of the design problem [25].

The Airbus *Vahana*, a single-seat eVTOL demonstrator flown in 2018, is perhaps one of the most recent public examples of an MDO-based process used to develop a flown aircraft [26–28]. The initial sizing study considered aerodynamics, structures, propulsion, and cost analysis to drive conceptual trade studies between various vehicle configurations. This study included practical constraints and margins, such as reserve mission energy, battery cycle life, and engine-out safety (by enforcing either autorotation capability, motor

redundancy, or a mass allocation for a ballistic parachute, depending on configuration).

Several other programs have built and flown uncrewed research aircraft with MDO use, such as the Facebook *Aquila* solar-powered UAV [29], the MIT *Jungle Hawk Owl* long-endurance UAV [30], and the X-48B blended-wing-body demonstrator [31–33]. Other programs document MDO usage for narrower subsystem- or component-level design, as in the detailed design of the Boeing 787 Dreamliner [7].

Outside of these cases, documented *industrial* use of MDO for complete-aircraft conceptual design remains rare, relative to the vast number of industry programs conducted in the past few decades. This observation is especially surprising in light of the widely-recognized potential utility of MDO in industry [6, 34].

One compelling partial explanation for this lack of use is a lag effect stemming from long timelines of new aircraft development programs – it takes time for new tools (such as MDO methods) to proliferate throughout industry, due to sunk-costs on design pipelines for existing programs. Indeed, although the present literature review finds that documented industry use of MDO is scarce, the situation appears somewhat less dire than older surveys indicate [1, 3, 4, 7, 8]; this suggests that industry use may be gradually increasing. However, considering that aircraft MDO research and industry interest in optimization has existed for over forty years [1, 2], this lag effect alone does not constitute a complete explanation for the lack of adoption.

Ashley considers another possible explanation for the lack of visible MDO use in industry: that of "military classification or company proprietary considerations." [1] However, after exhaustive correspondence with dozens of aircraft design industry contacts, Ashley concludes that this limitation was only occasional and "not to an extent that would affect the principal conclusions." Therefore, it seems that the limited observations of formal mathematical optimization processes in industrial aircraft design are genuine, representing a missed opportunity for the field to make an practical impact.

2.3 Pivotal Advances in Design Optimization Research

On a more optimistic note, MDO's substantial academia-industry gap is equally attributable to academia's remarkable advances in design optimization research over recent decades. This progress is readily apparent from analyzing trends in academic literature. As shown in Figure 2.1, the fraction of aircraft design publications directly referencing MDO has grown from near-zero in 1985 to 10% today.

Moreover, this statistic alone understates the true magnitude of MDO's impact. Ar-

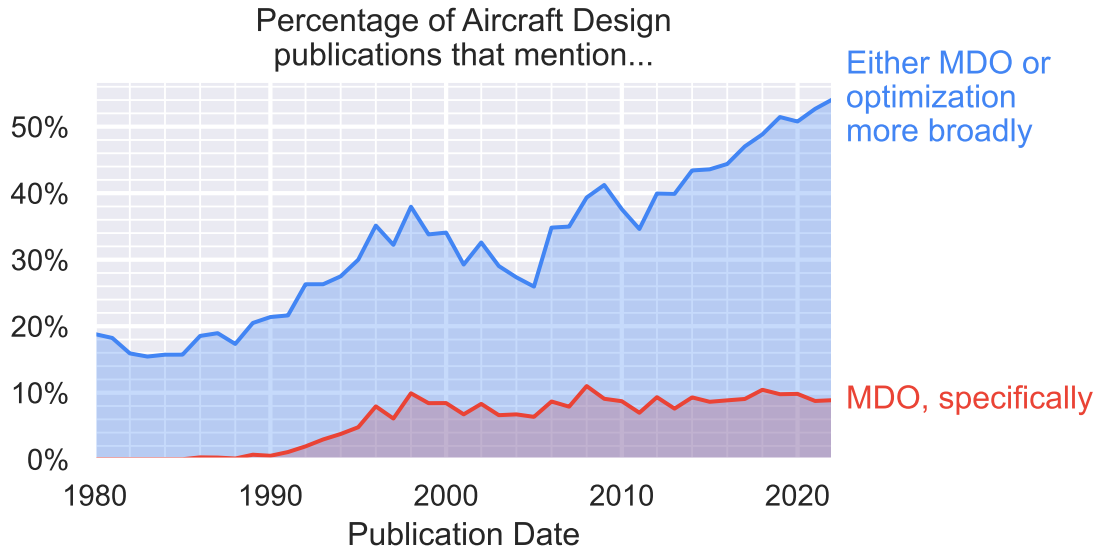


Figure 2.1: Prevalance of optimization-related keywords in academic literature with the keyword "aircraft design". Data from Google Scholar; includes industry-standard texts such as AIAA journals and conference proceedings. MDO-specific keywords (red line) include any of: "multidisciplinary design optimization", "MDO", and "MDAO". The blue line adds the keyword "optimization" to this list.

guably the most pivotal contribution of MDO research to aircraft design has been to catalyze a shift towards an *optimization mindset*: the recognition that optimization is not only a useful tool, but also a principled mathematical framework that can represent complete design problems. Figure 2.1 shows that this mindset is a relatively recent development – the fraction of aircraft design publications mentioning optimization in any form has tripled since the advent of MDO, reaching a majority share today.

These optimization-based design processes have shown several benefits when used to augment traditional design methods such as point analyses, parametric surveys, and carpet plots [34]. Most obviously, formal optimization methods can lead to improved design results and allow the consideration of many more design variables. Optimization can also help human designers discover clever cross-discipline synergies that might otherwise be overlooked [5], and its rigor can reduce the likelihood of biased decisions [34]. In cases where designer intuition is exhausted, such as with unconventional configurations, optimization provides a means to identifying useful directions for further exploration [5]. Torenbeek argues that an optimization-based design process can respond more quickly to unexpected changes in program requirements, whereas manual processes often require substantial redesign effort [34]. Finally, the optimization mindset itself has benefits, even

beyond the results of an optimization study. for example, discussions about problem formulation (how to translate given requirements into a quantified optimization problem) can help a team of engineers align on design goals and expose subtle discrepancies in perceived requirements[5].

Another important area where MDO research has made significant progress is in defining the relationship between the human designer and the optimizer. The prevailing view in the early days of aircraft design optimization was that optimization would eventually advance to the point that computational tools could yield complete, production-ready designs with minimal human oversight – as evidenced by encouraging titles such as "Automating the Design Process" [2, 35, 36]. Over time, this has largely given way to a more balanced view: although the optimization solve itself may be well-served by computational means, this forms only one small part of the larger design process. Inputting the engineer's design intent into the optimizer accurately ("problem specification") and extracting intuition from the results ("interpretation") are challenges in their own right, and best addressed by iterative human-computer teams. As described by Drela in a 1998 optimization study [5], "[Engineering optimization] is still an iterative cut-and-try undertaking. But compared to [traditional] techniques, the cutting-and-trying is not on the geometry, but rather on the precise formulation of the optimization problem."

When considering this full design process (i.e., from initial requirements to product delivery), the scope of challenges becomes even broader. Indeed, industry adoption of MDO is often limited by non-computational user frictions [9, 10, 37]. In the present work, we deliberately term these user frictions "costs", borrowing optimization nomenclature to emphasize that minimizing these frictions is an implied part of the objective function of any optimization-driven design process. Figure 2.2 summarizes a variety of design literature to present a holistic view of this complete process as applied to aircraft design [10, 34, 37–41]. Frictions that a designer may experience from a computational optimization framework at each step are shown in red.

In the aircraft design process model of Figure 2.2, the initial point of departure is a set of high-level requirements [34, 39]. The first design step is to develop a set of candidate concepts ("whiteboard sketches"), a qualitative and configuration-focused process that largely leverages designer creativity and experience [38, 40]. Concepts and the associated requirements are then translated into a formal mathematical design optimization problem consisting of an objective function, design variables, and constraints. Physics models are invoked as needed to support this process; this modeling process incurs a cost that de-

[5]For example, is the objective to minimize fuel burn, or to maximize range? Is operating cost a quantity that should be strictly capped (a constraint), or instead only penalized (an objective)?

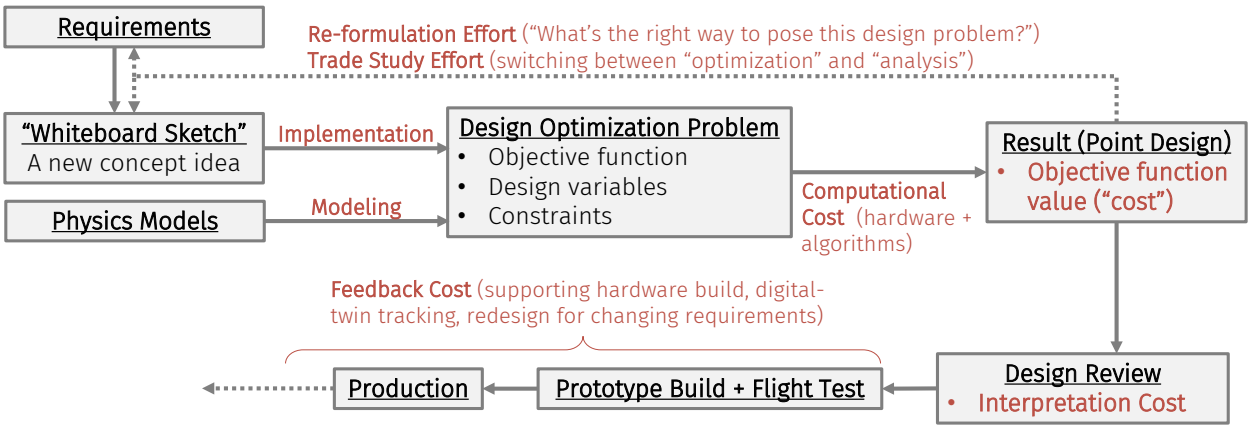


Figure 2.2: A high-level overview of the optimization-driven design process, as applied to aircraft design. "Costs" (user frictions to be minimized) associated with each step are shown in red.

pends on the modeling flexibility of the chosen MDO paradigm [12]. The optimization problem is then solved, which incurs a computational cost and yields a point design.

This process of mapping potential concepts to problems and problems to optimized point designs is often iterative. Even for expert users, it is rare that a design optimization process will fully reflect design intent on the first attempt [5]. In addition, post-optimality studies may be used to assess performance robustness to off-design conditions. The total time required to close this re-formulation loop is a critical driver of how the human interacts with the MDO tool, as it directly rate-limits how much design exploration can be performed.

For a practical aircraft development program, however, this is only the beginning. The point design must survive external validation and design review by a team of subject-matter-experts; the cost of this process is a direct function of the tools an MDO framework supplies to aid result interpretability. Even beyond this, in prototyping, flight testing, production, and deployment, the design process still imposes framework requirements. For example, an analysis and optimization framework that supports a wide range of modeling fidelities has the potential to can smoothly transition from a conceptual design tool to a digital thread [42, 43]. Likewise, a design tool can be used during the production process to support manufacturing decisions, such as whether an unexpected change in a component supplier still produces a feasible design with desired margins.

The remainder of this chapter will discuss some of the most important advances in optimization, design processes, and scientific computing in recent decades, with a focus on how these advances can reduce the costs incurred at various steps of this holistic process.

2.3.1 Two Directions: "Wide" vs. "Deep" MDO

Before proceeding further, it is important to clarify the scope of aircraft design optimization problems that are of interest in this thesis, as the definition of "MDO" is somewhat overloaded. More precisely, the modern field of MDO can be decomposed into two related but distinct subfields which have developed different strategies and mindsets to address different design needs. Figure 2.3 illustrates this distinction, showing how the term "MDO" can refer to separate tasks at different stages of the aircraft design process.

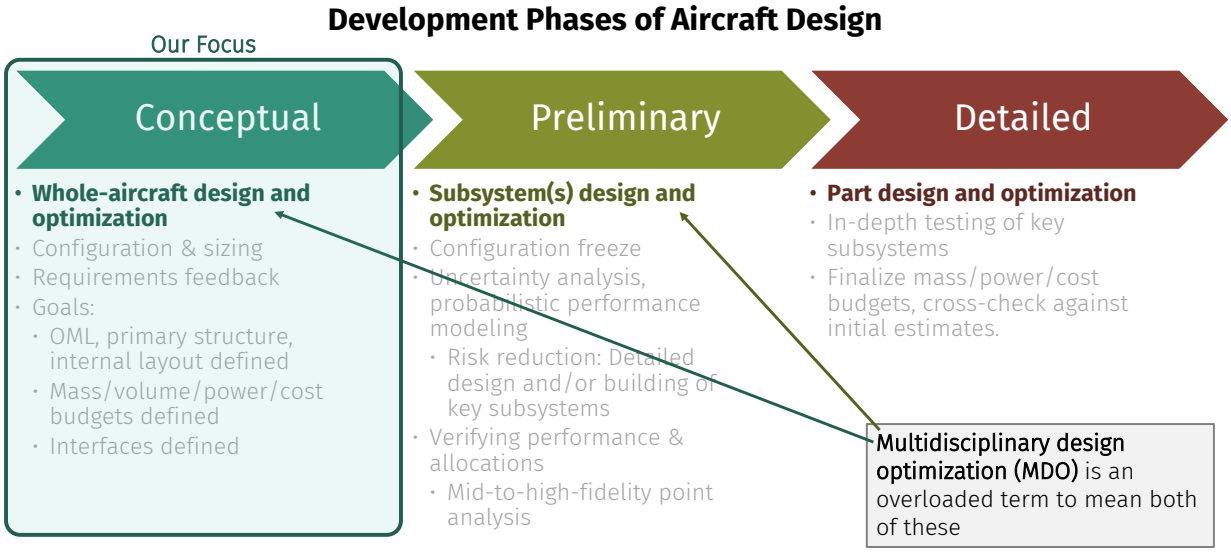


Figure 2.3: Multiple definitions of the term "multidisciplinary design optimization". Here, conceptual, or "wide" MDO is contrasted against preliminary, or "deep" MDO.

The first subfield, and the subject of the present work, is that of **conceptual**, or **"wide" MDO**. This is essentially a formalization of the conceptual sizing problem and typically involves casting a wide net to capture as many first-order dependencies as possible. The intended use case is often clean-sheet design of a complete aircraft, and the end goals are often initial identification of strong design drivers and assessment of concept feasibility. Sensitivity analysis is also a key desired output of a wide MDO process. Frameworks that tend to specialize on this class of wide MDO problems are TASOPT [16] and GPkit [9].

These goals requires an enormous breadth of models to be considered, as practical aircraft designs are shaped by an enormous number of constraints that do not appear in first-principles performance analysis (e.g., the Breguet range equation). For example, a drag reduction of the vertical stabilizer is of little interest if engine-out performance constraints preclude certification; likewise, a more efficient engine is of little interest if acoustic constraints preclude customer acceptance. This need for analysis breadth can lead to de-

sign problems with many hundreds of models spanning anywhere from five to twenty relevant disciplines. To illustrate this breadth, consider the following non-exhaustive list of example disciplines that might be included:

- Aerodynamics
- Structures and weights estimation
- Propulsion
- Power systems and thermal management
- Flight dynamics (trim, stability, and control)
- Internal geometry and packaging
- Mission design (concept of operations) and trajectory optimization
- Takeoff and landing field length analysis
- Certifiability, safety, and engine-out performance
- Life-cycle emissions
- Cost modeling and manufacturability
- Acoustic signature
- Ride quality
- Survivability and stealth

Each of these disciplines might include dozens of submodels [16, 34, 39, 44]. To keep problems tractable despite their large scope, these models are typically low-fidelity; often, they are either derived from first principles physics with appropriate simplifications or regressions to statistical data. Despite the reduced accuracy of these models, they fulfill two critical functions. First, they apply *optimization pressure* to steer the optimizer towards realistic designs, often in subtle, interdisciplinary ways. For example, increasing fuselage length requires increased landing gear length and mass to maintain a proper takeoff rotation angle, reducing the attractiveness of such design changes. Secondly, these low-fidelity models provide information on which constraints might be active near the optimum, allowing computational resources to be devoted to fidelity improvement only where it is needed. Often, these wide MDO problems are posed as large single-level optimization problems, which allows the large number of cross-disciplinary constraints to be computationally represented in a shared namespace [19].

The second subfield of MDO is that of **preliminary**, or **"deep" MDO**. Here, the goal is often to provide detailed design refinement of a very close initial guess. Because the low-hanging fruit has often been picked by this stage, the design problem is often more local in nature, and the number of models is typically smaller than in wide MDO. Likewise, continued improvement from a good initial guess requires high-fidelity models, since the objective function tends to be relatively flat near the optimum and hence highly sensitive to model inaccuracy. For example, RANS-based CFD[6] models are often used for aerodynamics, at significant cost: optimization runtimes of 1,000 to 100,000 CPU-hours are

[6]Reynolds-averaged Navier-Stokes; Computational fluid dynamics

not uncommon [45]. To maintain tractability, relatively few disciplines are considered – often only aerodynamics and structures. The intended use case is often detailed design of a single component or subsystem, such as a wing. Due to computational challenges that tend to be more prevalent in high-fidelity models (e.g., PDE-constrained optimization, adjoint-based gradients, numerical stiffness of models, and convergence of models that use iterative solvers), the subfield of "deep" MDO has placed a large focus on MDO architectures that allow multi-level decomposition of the original optimization problem [46]. Frameworks that tend to specialize on this deep MDO subfield are MACH-Aero [47], and, to some extent, OpenMDAO [48].

While both subfields share broad similarities and provide useful design insight, they are fundamentally attacking different problems: wide MDO is aimed at clean-sheet conceptual design, while deep MDO tends to aim at detailed refinement of a close initial design. Historically, this schism became increasingly apparent roughly in the 1990s and 2000s, as various research groups began "spending" their increasing computational budgets in different ways: either model breadth or model depth. Examples of wide MDO research efforts in the time since this divergence include PASS [49] by Kroo, TASOPT [16] by Drela, and various GPkit-based [19] aircraft design codes. Modern ideas around deep MDO research can trace their origins to high-fidelity adjoint-based aerodynamic shape optimization studies by Alonso, Martins, and other contemporaries [25, 50, 51].

2.3.2 Advances in Optimization Algorithms

The current academic state-of-the-art for MDO paradigms has largely centered on two approaches: gradient-based methods with analytical gradients, and disciplined optimization methods. The former tends to be more popular in deep MDO applications, while the latter tends to be more popular in wide MDO applications; however, exceptions exist. These two existing approaches are discussed in turn below. Pros and cons of these existing approaches, as compared to the proposed new paradigm, are summarized in Table 3.1 of the subsequent chapter and discussed fully in Appendix A.

Gradient-Based Methods with Analytical Gradients

Gradient-based optimization offer fundamental scaling advantages over gradient-free methods, and are hence preferred for large-scale design optimization when model characteristics allow [37, 52]. The effectiveness of gradient-based methods is highly contingent on the speed and precision of gradient calculations. To illustrate this, Kirschen and Hoburg show that gradient-based optimization gives inadequate performance on even modest en-

gineering design problems if simple finite-differencing is used [53]. Likewise, Adler et al. show that the inaccuracy of finite-differenced gradients can prevent accurate optimization [54].

Because of these considerations, a leading academic approach to gradient-based MDO involves manually supplying exact analytical gradients to the optimizer for each submodel. Frameworks like OpenMDAO and MACH-Aero [48] provide example implementations of this manual-gradients approach, forming the basis for several interesting case studies [17, 18, 55, 56]. Although this paradigm is most often applied to deep MDO problems, it has been applied to wide MDO problems in examples such as OpenConcept [55] and OpenAeroStruct [56].

Disciplined Optimization Methods

Another MDO paradigm that has become popular in recent years is that of disciplined optimization methods, borrowing nomenclature from Grant [57], Agrawal [58], and Boyd [59, 60]. These methods are considered *disciplined* as they impose a strict set of rules on the optimization problem formulation, which allows the optimizer to make assumptions about the problem structure (such as convexity, or log-convexity). These assumptions allow the optimizer to make more efficient use of computational resources and provide guarantees of global optimality. Geometric programming in particular has proven to be a useful disciplined MDO paradigm for aircraft design applications, as originally observed by Hoburg [19]. GPkit [9, 19, 53], a framework that demonstrates this geometric programming concept targeting engineering design applications, has led to successful case studies such as the Jungle Hawk Owl aircraft [30] and Hyperloop system studies [9].

2.3.3 Other Advances in Design Optimization Practice

A recent notable shift in aircraft design optimization perspectives has been a renewed focus on geometry representation and parameterization. For example, Haimes and Drela [61] contended in 2012 that "constructive solid geometry (CSG) is the natural foundation for [aircraft design optimization]". This publication extended prior work by Lazzara, Haimes, and Willcox [62] that introduced the concept of "multifidelity geometry". Together, these works advocated for accurate 3D outer mold line representations from the earliest stages of conceptual design, with degenerate representations of this central geometry used to drive individual discipline analyses. Furthermore, they recognized the fundamental limitations of general-purpose (e.g., Parasolid-based) CAD tools in aircraft design optimization: because aerospace geometries tend to use complex, lofted surfaces,

the resulting CAD models can be brittle with respect to parameter changes. This observation, along with later developments such as differentiable discretization techniques [63], led to renewed research on how aircraft geometry should be parameterized for optimization. For example, in recent years aircraft-specific geometry tools like OpenVSP [64] and Engineering Sketch Pad [63] have been developed for design optimization workflows, a trend that arguably stems from this line of research. In a broader sense, another notable area where computational methods for aircraft design have long made inroads to industry is in inverse design tools. (An example of successful industry adoption in aerodynamics the inverse approach used by the XFoil airfoil design code [65] and others [32] to recover a shape from a specified pressure distribution.) Like optimization methods, these inverse methods can aid designers because they offer fundamentally different capabilities than traditional design methods (e.g., carpet plots). While traditional methods focus on the "forward problem" (design $\rightarrow$ performance), inverse methods and optimization methods both focus on the "inverse problem" (performance $\rightarrow$ design). Drela [5] shows that this more-manual inverse approach can lead to more robust designs than optimization, if the user is not familiar with the risks of an improperly-specified optimization problem.

2.3.4 Advances in Numerical Methods

In the past two decades several notable scientific computing techniques have matured to the point of practical use in engineering design optimization. Two of the most significant advances are automatic differentiation and automatic sparsity detection, which are discussed in detail below. A review focusing on these two techniques is available in work from Andersson et al. [66] and Rackauckas [67]. Review of several other notable scientific computing advances, such as GPU-accelerated computing, probabilistic programming, deep learning, and uncertainty quantification, are omitted here for brevity. A wider review of state-of-the-art scientific computing techniques is available from Lavin et al. [68]

Automatic Differentiation

Here, we review literature on *automatic differentiation*, an advanced technique borne out of the machine learning community that fundamentally accelerates gradient-based optimization algorithms. A comprehensive modern survey is available by Baydin et al. [69]. A 2008 textbook by Griewank and Walther [70] provides a more in-depth review of the mathematical foundations of automatic differentiation.

Gradient-based optimization methods are computationally bottlenecked by the cost of computing gradients [37, 52]. Existing state-of-the-art workflows have users manually de-

rive and implement highly-efficient gradients for each model, but this is time-consuming and thus impractical for early-stage design. Automatic differentiation (AD) is an efficient and accurate way to automatically compute the gradients of a function *as represented in code* at runtime. In many cases, the computational complexity of this technique is fundamentally better than traditional approaches like finite-differencing; Griewank provides concrete upper bounds on the worst-case time complexity [71].

Automatic differentiation exploits the fact that any function can be broken into a series of simple mathematical operators, which can then be represented as a computational graph. As a practical matter, this computational graph is often created by tracing program execution with an overloaded numerics library [72]. This graph construction process is illustrated in Figure 2.4. The chain rule can then be applied to this graph to compute the Jacobian of the function. Various modes of automatic differentiation effectively allow this Jacobian construction to occur either column-by-column (forward-mode) or row-by-row (reverse-mode) [37, 66, 73]; depending on the shape of this Jacobian, one strategy may be enormously more efficient than the other.

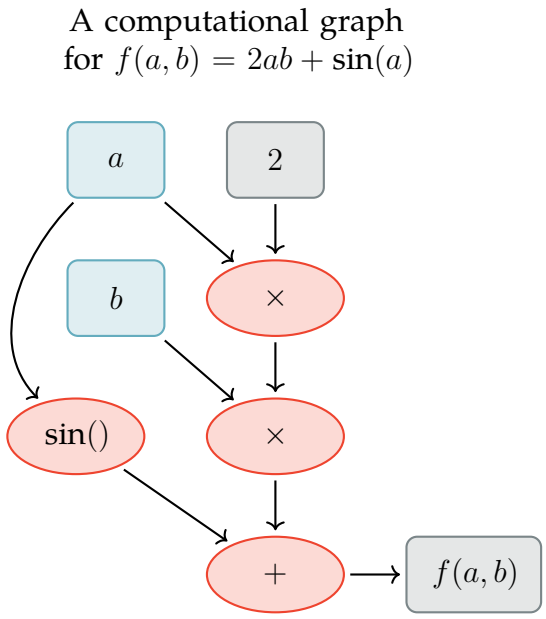


Figure 2.4: A computational graph, as would be used for automatic differentiation. Reproduced from [12].

Advances in automatic differentiation (and in particular, combining this with GPU computing) are largely responsible for the revolution of practical advances in machine learning in the past decade [69]. Rackauckas in 2021 [67] states: "[Automatic differentiation] has become the pervasive backbone behind all of the machine learning libraries. If

you ask what PyTorch or Flux.jl is doing that's special, the answer is really that it's doing automatic differentiation over some functions." The goal is that similar practical advances can be brought to bear in engineering design optimization, a field that has similar potential to benefit from this scientific computing technique.

Automatic Sparsity Detection and Jacobian Coloring

The process of gradient computation for optimization can be further accelerated if the sparsity of the constraint Jacobian is known a priori. This is because sparsity can guarantee structural independence of multiple columns[7] of the Jacobian, so the respective elements of the Jacobian can be evaluated simultaneously. Sparse Jacobians arise frequently in the modeling of physical systems, especially multidisciplinary ones [74]; because of this, this represents a significant opportunity in engineering design. The process of reducing the number of Jacobian evaluations by simultaneously evaluating independent columns is known as *Jacobian compression* and is illustrated in Figure 2.5. To enable compression, independent columns of the Jacobian must be grouped. This process is known as *Jacobian coloring*, referencing mathematical similarities to the graph coloring problem if the Jacobian's sparsity pattern is represented as a bipartite graph [75]. This coloring problem does not readily admit exact solution for Jacobian sizes of interest; however, Gebremedhin et al. provide a comprehensive review of graph coloring heuristics that work well in practice [76]. Martins and Ning provide a more concise summary that contextualizes this process around the backdrop of design optimization [37].

[7] or rows, if using reverse-mode automatic differentiation

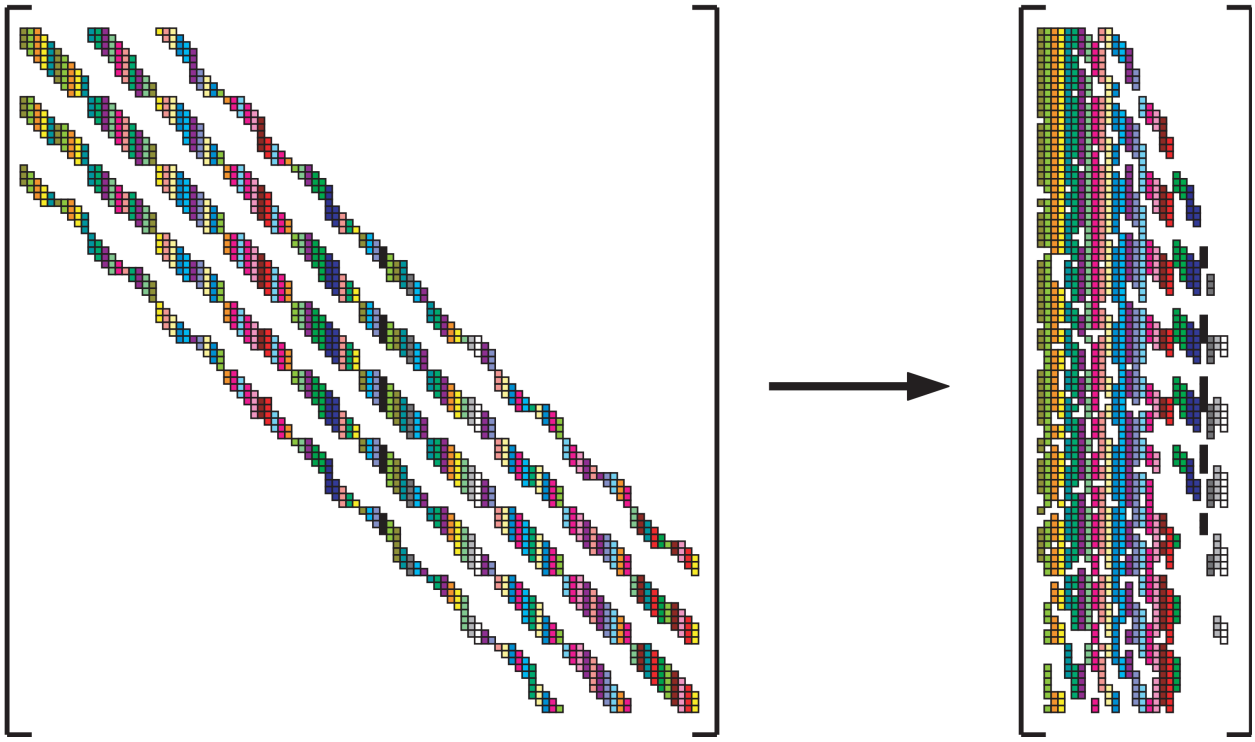


Figure 2.5: A sparse Jacobian can be evaluated more quickly by simultaneously evaluating derivatives that are structurally independent, a process known as Jacobian coloring and compression. On the left of the figure is the original Jacobian; on the right is its compressed form. Figure reproduced from Gebremedhin et al. [76]

Before Jacobian coloring and compression can be performed, the sparsity pattern of the Jacobian must be known. This is a nontrivial task and an area of active research, with the major complications summarized by Rackauckas [67]. In recent years, automated sparsity detection has enabled automatic construction of sparsity patterns using a tracing approach that is quite similar to that of automatic differentiation. This bypasses many of the potential edge-case failure modes that can occur when trying to infer sparsity purely by inspecting the dense Jacobian, such as conditional branching or cancellation. Indeed, the same computational graph can be used for this process, making this a computationally-efficient addition to a code base that already leverages automatic differentiation. Andersson et al. [66] provide an example implementation of this automated sparsity detection within the CasADi framework. Gowda et al. provide a more complete review on state-of-the-art methods in automated sparsity detection [77], with state-of-the-art implementations of this by Ma et al. [11].

2.4 Motivations for Improving Industry Access to Design Optimization

This thesis focuses on improving the accessibility of advanced optimization methods for industry practitioners, with a particular focus on techniques applicable to conceptual ("wide") MDO for aircraft. This focus is motivated by several factors.

Motivation 1: A Majority of Performance, Risk, and Cost is Committed Early

First, the vast majority of the performance of the final design is determined by early-stage conceptual design decisions. This can manifest in both a positive and a negative sense: after the conceptual design is frozen, most design opportunities cannot be captured later and most design regrets cannot be fixed. Figures 2.6 and 2.7 illustrates the two sides to this coin with practical examples.

The first example, shown in Figure 2.6, reproduces conceptual design work on the D8 "Double Bubble" transport aircraft configuration by Drela [15, 78]. Among other technology improvements, the configuration leverages a strategy of improving passenger loading/unloading speed, which allows for a reduced cruise Mach number while retaining comparable door-to-door travel times (a surrogate for passenger acceptance). This synergistic strategy creates a feedback loop which dramatically reduces fuel burn compared to designs that are optimized on a per-component decomposition basis. For example, one of Drela's main observations from a 2010 work [78] is that "multi-discipline optimization considerably increases the fuel savings compared to single-discipline optimization."

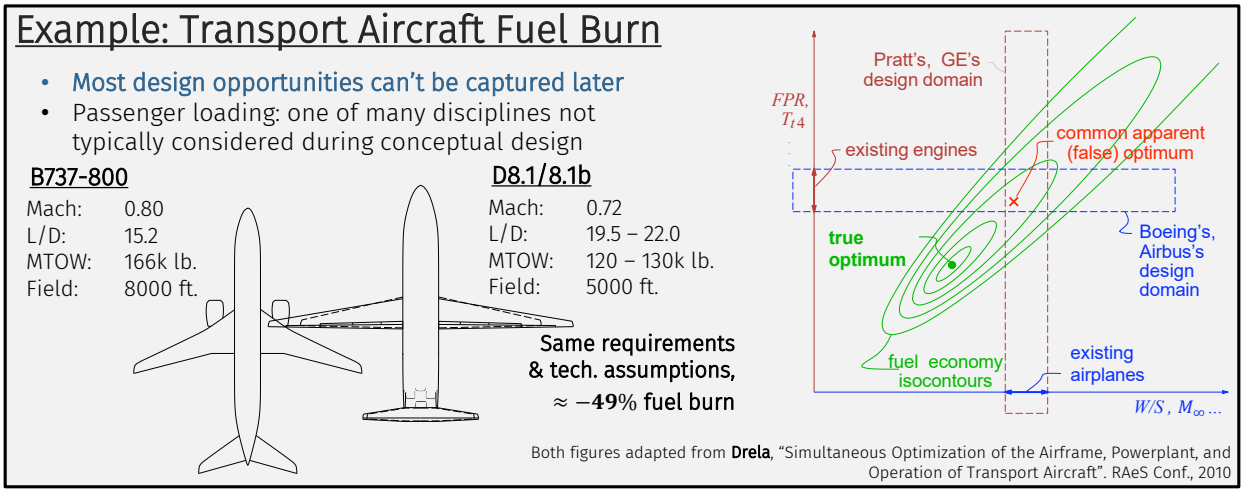


Figure 2.6: In an example from Drela [15], large potential fuel savings are available for transport aircraft if cross-discipline couplings are considered early in the conceptual design process. Figure elements reproduced from Drela [78].

In a second example, shown in Figure 2.7, the other side of this effect is shown. Here, various electric vertical-takeoff-and-landing (eVTOL) aircraft design concepts are compared. Given the significant energy limitations of battery-powered aircraft, much of the focus of eVTOL conceptual design in the early 2010s was on traditional unit-economics performance metrics of range and payload fraction. However, as these vehicles begin to approach FAA certification and initial deployment, significant regulatory concerns have been raised about the community noise impact of such vehicles. Aeroacoustic noise, to first order, is a strong function of propulsor disk loading and blade tip speed [79], factors which require substantial redesign to modify later. Hence, manufacturers have had varying levels of difficulty in adapting to this renewed focus on acoustics, as this metric is largely locked in from conceptual design decisions made many years ago.

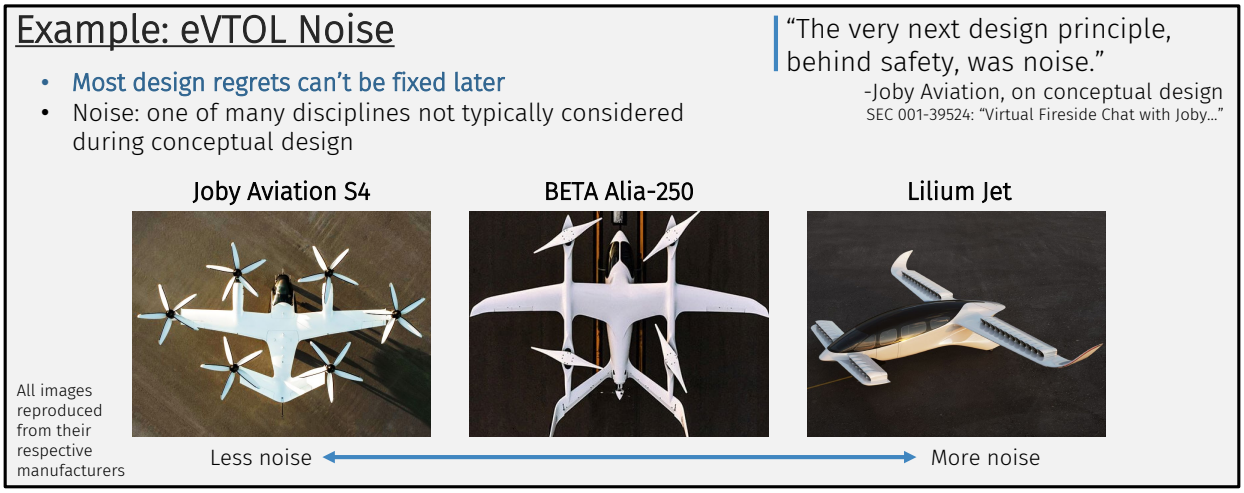


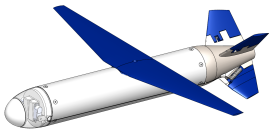
Figure 2.7: Design regrets may arise if conceptual MDO studies do not include all relevant disciplines. This example of contemporaneous eVTOL designs assesses aeroacoustic noise, a factor that is largely determined by conceptual design decisions made many years prior. Although noise is not traditionally included in conceptual aircraft design relations, it is crucial for certification and acceptance. Thus, including many disciplines in conceptual design – even non-traditional ones, like acoustics – is key for avoiding major downstream design regrets.

Motivation 2: New Technologies Reignite a Need for Early-Stage Design Exploration

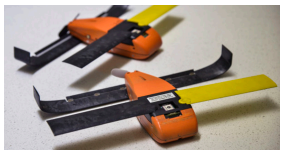
A second motivation for the focus on conceptual MDO is that recent technological shifts have significantly expanded the aircraft design space compared to previous eras, calling for new tools to aid in down-selecting design concepts.

As a first example shown in Figure 2.8, miniaturization and uncrewed aircraft have opened up new trade spaces to explore, calling for renewed focus on first-principles early-stage design optimization. These micro-drone aircraft are able to take more risks with exotic designs, due to shorter design cycles, lower cost, and reduced certification and safety risks. New disciplines, such as packaging and folding considerations, drive new trades – in many cases, volume allocation trades can be as sensitive as mass fraction trades in conventional aircraft. Another new trade is that of size, weight, and power (SWaP) allocations to autonomy and onboard computing. For small air vehicles, flight computer power requirements can equal or exceed propulsive power, creating new incentives that tilt the design space towards control simplicity [80]. New missions, particularly those that are "dull, dirty, and dangerous" are enabled by uncrewed aircraft with attritable design and cost optimization. Finally, physics scaling laws, like the square-cube law and transitional Reynolds numbers, cause unique concerns for small-scale aircraft that warrant

first-principles conceptual design optimization.



(a) MIT Firefly, a Mach 0.8, rocket-propelled micro-UAV [81, 82]



(b) MIT Perdix, an air-launched ALE-55-class ISR UAV [21]

(c) Transonic DP, a 545-mph dynamic soaring glider with 100 G turn capability (photo: Spencer Lisenby)

(d) Black Hornet Nano, an 18-gram ISR helicopter (photo: Richard Watt/MOD)

Figure 2.8: Examples of new design spaces enabled by miniaturization and uncrewed aircraft technology in recent years.

Another prominent new design space that motivates revisiting first-principles conceptual aircraft design is electric aviation, as shown in Figure 2.9. Electric propulsion offers a fundamentally different configuration trade space compared to conventional propulsion. In terms of energy storage, modern lithium batteries have realizable[8] specific energies that are roughly 1/25th that of kerosene, placing extreme sizing demands here. On the other hand, electric motors have excellent specific power across a wide range of size scales, dramatically fewer moving parts, and wider power bands compared to combustion engines. These factors make electric motors far more modular, essentially allowing designers to place propulsors at will – electric aircraft with eight or more propulsors are not uncommon. The enormous design space that electric propulsion opens up is evident in Figure 2.9, where the diversity of aircraft configurations is wholly unprecedented compared to other aerospace domains. Part of the value proposition of conceptual MDO is that it can make rigorous design within this large space more tractable.

[8] in other words, after accounting for the generally-higher powertrain efficiencies of electric propulsion due to the lack of a thermodynamic gas cycle

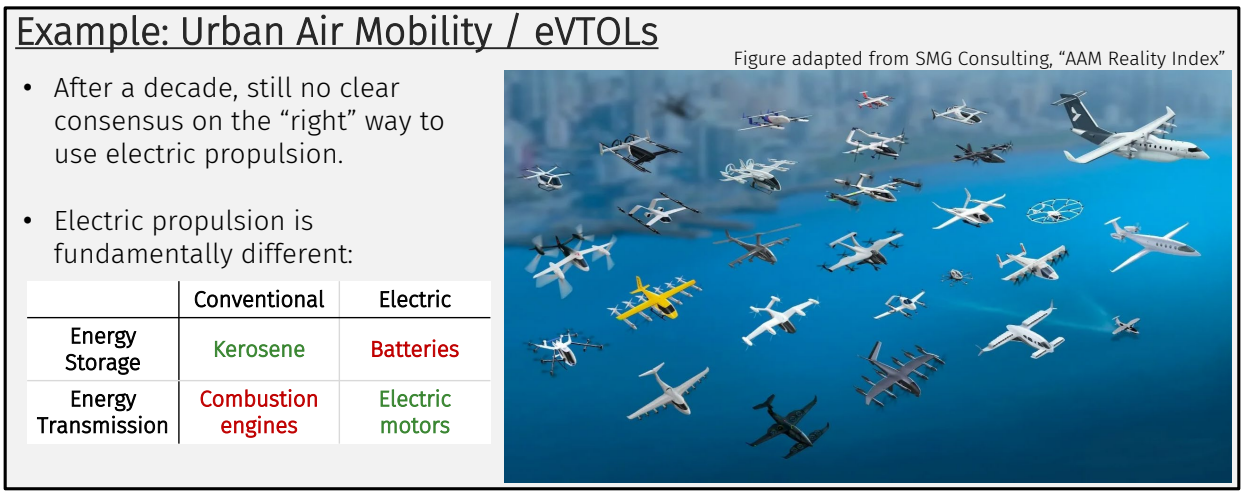


Figure 2.9: Electric propulsion opens up a much larger aircraft configuration space, since electric motors offer higher specific power and modularity than combustion engines. A focus on early-stage conceptual design is critical for down-selecting from this diversity of configuration options. Figure adapted from SMG Consulting [83].

Motivation 3: Design Space Exploration

The final and most significant motivation for this thesis' focus on early-stage MDO is that it provides value far beyond merely the point design that results from an optimization study. The true value of conceptual MDO is in determining which questions a design team should be asking. Examples of such questions that might be fueled by a design tool leveraging conceptual MDO are given in Figure 2.10. Though some of these questions can be answered with traditional methods, like post-optimality studies and parameter sweeps, modern conceptual MDO tools can supercharge this to tens, hundreds, or thousands of design variables, allowing for a much more comprehensive exploration of the design space. The inclusion of similarly large numbers of relevant constraints effectively prunes the design space, reining unrealistic designs that a parameter sweep might otherwise yield. This capability is especially important in the early stages of a program, where the design space is often poorly understood and the design team is still developing an intuition for the problem. In this sense, conceptual MDO is a tool for *design space exploration*, not just design optimization.

Figure 2.10: Conceptual MDO provides far more utility than just the point design that results from an optimization study. It can help designers develop an intuition for the problem space and determine which questions to ask next. This figure gives examples of such exploratory questions.

Chapter 3

Proposed Contributions and Approaches

The Ph.D. thesis will make five following novel conceptual contributions which to address the technical gap and motivation identified in Chapter 2. A concise list of these contributions is given in Section 1.1. Here, we expand upon each of these contributions in detail, discuss intended approaches to making advances here, and discuss the results to date.

3.1 Code Transformation Paradigm

The thesis will introduce the idea of *code transformations* as a new computational paradigm on top of which an MDO framework can be built. Here, code transformations are defined as a generalized set of computational techniques that intercept the original optimization problem posed by the user (at runtime), apply some improvement based on analysis of the code itself, and then solve a modified optimization problem instead. This encompasses a variety of recent advanced techniques in scientific computing, such as:

- Automatic differentiation [71]
- Automatic sparsity detection [75]
- Problem transformations:
    - Auto-scaling [84]
    - Log-transformation of variables, constraints, and objectives (similar to geometric programming) [53, 58]
    - Redundant constraint elimination
- Backend-agnostic programming (CPU/GPU, different math library backends, just-in-time compilation, automatic vectorization and parallelization) [73]

The benefit of introducing this abstraction is to recognize that all of these advanced

techniques essentially share one major requirement to use: the optimization framework must be able to inspect the actual code driving the design problem. This code inspection is done either via direct source code analysis or by creating a computational graph of the optimization model at runtime. The latter approach, which is called *tracing* in machine learning literature [69, 73, 85], is widely preferred to the former in modern, syntactically-rich languages[1]. Therefore, for the purposes of this work, *traceability* is effectively synonymous with code transformability.

If code transformation techniques can be applied to engineering design optimization, they offer order-of-magnitude speedups over the black-box optimization methods that form the vast majority of industry use today [37, 68]. These achievable speeds are comparable to those of state-of-the-art optimization methods in academia, such as disciplined optimization methods[2] [9, 57, 58, 60] and gradient-based methods using user-provided analytic gradients[3] [48, 87, 88].

However, code transformations offer a key advantage over these state-of-the-art methods in that they can be applied *automatically* – most of the benefits of these advanced techniques can be gained without requiring any additional effort (or mathematical expertise) from the user. This ease-of-use is critical for practicality – Grant notes that existing paradigms are hamstrung by a "expertise barrier" of existing methods, as described by Grant [57]. Table 3.1 compares code transformations to existing MDO paradigms across three key practical metrics. In short, code transformations offer the best of both worlds: the computational speed of latest academic methods with the ease-of-use of methods already accepted by industry today.

[1] Maclaurin provides a comprehensive discussion of the tradeoffs between these techniques, and why tracing is often preferable in practice [86]

[2] such as geometric programming and disciplined convex programming

[3] sometimes referred to as "adjoint methods" in reference to a common method for manually deriving these gradients for more-complex analyses

Table 3.1: A subjective comparison of tradeoffs between existing MDO framework paradigms and the proposed *code transformation* paradigm. The industrial state-of-the-art is largely *black-box optimization*. The academic state-of-the-art has two major branches: *gradient-based methods with analytical gradients*, and *disciplined optimization methods*. More detailed discussion of these assessments, including definitions and reasoning, is given in Appendix A.

MDO Framework Paradigm	**Example Frameworks and Tools**	**Ease of Implementation** (idea-to-code)	**Runtime Speed and Scalability** (code-to-result)	**Modeling Flexibility**
Black-box Optimization	SUAVE [89], OpenMDAO* [48], TASOPT [16], PASS [49], FAST [90], FLOPS [91], FBHALE [29], **almost all industry codes**	**Great**	**Limited**	**Best**
Gradient-based with Analytic Gradients	MACH-Aero [47], OpenMDAO* [48], OpenConcept [55]	**Limited**	**Best**	**Great**
Disciplined Optimization	GPkit [9], other convex methods [57, 60, 92], most algebraic modeling languages	**Good**	**Good**	**Limited**
Code Transformations	AeroSandbox† [12], JAX‡ [73], ModelingToolkit.jl‡ [11]	**Good**	**Great**	**Good**

* Can use either paradigm, depending on user's implementation
† Part of the present work, as detailed in Section 3.1
‡ These are computational tools to facilitate code transformations, rather than frameworks themselves

This leads to a compelling value proposition: if we can develop new methods for industry engineers to easily write traceable design code, then we gain access to an host of advanced scientific computing techniques that can significantly shrink the academia-industry MDO gap described in Chapter 2.

Thus, this contribution is twofold. First, the thesis will conceptually introduce code transformations as a new paradigm for engineering MDO frameworks. Secondly, the the-

sis will demonstrate strategies that allow traceability of engineering design code with minimal user effort, enabling the use of code transformations in practice.

Results-To-Date in Code Transformations

To explore and demonstrate this code transformation paradigm, we created *AeroSandbox*, a Python-based MDO framework for conceptual design. This framework serves as both a proof-of-concept implementation as well as a proving ground for testing new transformations, with specific focus on engineering design applications. On top of this code framework, many optional aircraft-design-specific tools and physics modules are included; nevertheless, the core design optimization framework is applicable for conceptual design of many large-scale engineering systems. The code is made available open-source on GitHub under the permissive MIT license in order to gather as much practical user feedback as possible.

Core components of this framework and its mathematical basis are discussed more fully by Sharpe [12]. The framework is built on top of the *CasADi* [66] and *JAX* [73] scientific computing libraries, which allow the computational graph tracing and several of the code transformation capabilities that are central to this work. The framework is designed to be modular and extensible, with a focus on ease-of-use and interpretability. The framework is also designed to be highly performant, with a focus on leveraging code transformations to achieve high computational speeds. As an optimization backend, the framework currently solves the transformed optimization problem using IPOPT, a primal-dual interior point algorithm [93].

Fundamentally, AeroSandbox is a tool for building and transforming computational graphs, and indeed these graphs can be directly inspected both computationally and visually, as Figure 3.1 shows. When constructing this graph, AeroSandbox deliberately makes certain framework-level decisions (e.g., to use a static, rather than dynamic graph; to use heuristics for forward- vs. reverse-mode automatic differentiation); these choices are tradeoffs that tend to work well on engineering analysis code [94].

AeroSandbox can be used to demonstrate the potential of code transformations to accelerate optimization problems with minimal effort. Here, we give an example comparison using the Rosenbrock optimization problem. This problem is a classic optimization benchmark, designed to be a stress-test as the optimum lies at the bottom of a shallow-curving valley [95]. Here, we solve an N-dimensional extension of this problem, which conveniently gives a knob to dial up or down the difficulty of the problem [96]. This optimization problem is defined in Equation 3.1:

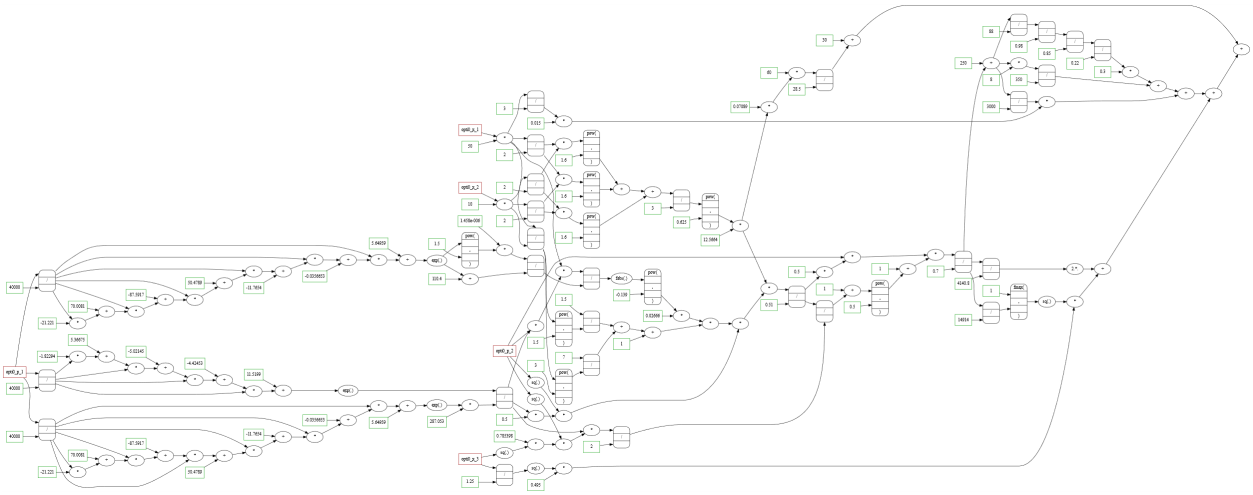


Figure 3.1: Computational graph of a mass model for an airship, as constructed by AeroSandbox. This graph is automatically constructed at runtime.

$$\underset{\vec{x}}{\text{minimize}} \quad \sum_{i=1}^{N-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + \left(1 - x_i \right)^2 \right] \tag{3.1}$$

Figure 3.2 illustrates this optimization landscape for the case where $N = 2$, showing the curved valley. For all N, the global optimum is at $\vec{x} = \vec{1}$, where the objective function evaluates to 0. This problem is chosen here because it shares many difficult aspects with engineering design optimization problems: it is nonlinear, nonconvex, and poorly-scaled. Furthermore, we deliberately choose poor initial guesses, with each element of the vector of initial guesses drawn from a random uniform distribution in the interval $[-10, 10]$.

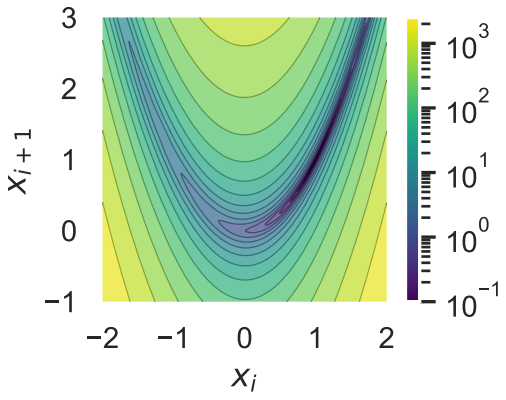


Figure 3.2: The Rosenbrock function, a classic optimization benchmark problem.

In Figure 3.3, the performance of AeroSandbox (which leverages code transformations)

is compared against existing methods using black-box optimization techniques. AeroSandbox offers faster practical and asymptotic optimization performance than existing black-box optimization methods, demonstrating the magnitude of acceleration that is possible with code transformations.

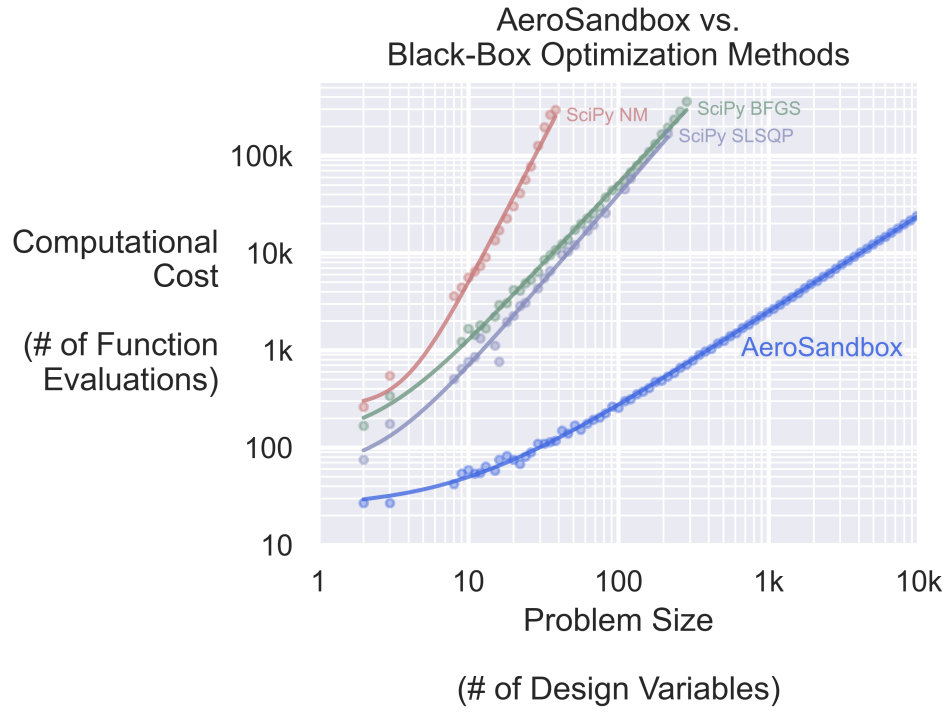


Figure 3.3: Comparison of optimization performance between AeroSandbox and existing black-box optimization methods on the N-dimensional Rosenbrock problem. Other methods are a Nelder-Mead simplex method and two gradient-based methods (SLSQP and BFGS), both using common SciPy implementations [97].

Engineering benchmarks comparing the performance of an early implementation of the code transformations paradigm to disciplined optimization methods are given in Sharpe [12]. In short, code transformations can equal or exceed the performance of disciplined optimization methods such as geometric programming, even on problems where this disciplined optimization approach is applicable. This is a promising result, as it suggests that code transformations may be able to offer the best of both worlds on engineering problems: the computational performance of disciplined optimization methods without the strict mathematical restrictions.

3.2 Traceable Physics Models

The thesis will also contribute the first computational implementations of several key physics models for aircraft design that are compatible with a code transformations paradigm (i.e., traceable).

The broad motivation stems from the observation that "ease-of-implementation" has historically proven to be one of the most important factors for determining whether an MDO paradigm can achieve use in industry. More specifically, the goals of this contribution are to:

1. Stress-test the feasibility of code transformations in practice – How much added user effort and expertise is required to bring typical engineering analyses into a code-transformations-based MDO tool? To what extent can existing code be used as-is? Finally, the thesis aims to identify any specific computational elements that cause "pain points" when attempting to make an analysis traceable.

2. Jump-start future applied research by providing a set of modular, plug-and-play analyses that can be used to quickly build a variety of aircraft design optimization problems. Since the long-term goal of this research direction is to establish whether the proposed MDO paradigm improves practicality, it follows that many practical aircraft design problems must be posed. By creating the building blocks for such problems, we aim to accelerate future research.

To achieve these goals, the thesis will contribute the first traceable implementations of the common aircraft design analyses given in Table 3.2. This set of analyses was deliberately chosen to be diverse, spanning a wide range of common computational attributes. The types of attributes that each analysis is intended to stress-test are given in the rightmost column of Table 3.2. By contributing this breadth of analyses within a traceable paradigm, we aim to cover a wide and representative gamut of engineering code patterns.

Table 3.2: A list of aircraft design analyses that the thesis will implement within a code transformations framework. The middle column lists the non-traceable tools for each analysis that are commonly used in industry today. The right-most column lists the computational attributes that each analysis is intended to stress-test.

Analysis To Contribute	**Non-Traceable Analogue**	**Tests tracing through...**
Vortex-Lattice Method Aerodynamics Analysis	AVL [98]	An aerospace geometry engine and discretization; large, vectorized matrix methods, like linear solves
Nonlinear Lifting Line Aerodynamics Analysis	Phillips & Snyder [99], Reid [100]	Nonlinear systems of equations (i.e., implicit), which often lead to value-dependent computational graphs (via convergence loops)
Workbook-style Aerodynamics Buildup	USAF Digital DATCOM [101]	Table lookups, large amounts of conditional logic (yielding a wide, branching graph), and scalar-heavy math
Rigid-Body Equations of Motion	ASWING (dynamics) [102]	Ordinary differential equations, which are often implemented in a loop-heavy way (yielding a deep graph)
Linearized Aircraft Stability Modal Decomposition	AVL [98]	More advanced matrix methods, such as an eigenvalue decomposition

These traceable implementations offer value within the context of the thesis itself (in stress-testing the practicality of code transformations), but they also have value as a standalone contribution to the aircraft design community – even outside of design optimization. For example, a traceable workbook-style aerodynamics buildup would seamlessly enable GPU-accelerated evaluations and automatic vectorization, offering significant speedups; an example application might be real-time performance estimation for model predictive controllers or flight simulation.

3.3 Sparsity Tracing via NaN-Propagation

A key challenge in applying code transformations is handling external black-box (i.e., non-traceable) function calls within the analysis graph. While code transformations can apply techniques like automatic differentiation and automated sparsity detection to accelerate traceable code, these efficiency gains are lost when opaque black-box functions are present. Here, gradient calculation methods must invariable fall back to slower techniques, such as finite-differencing or complex-step derivative computation [103]. However, significant speedups are still achievable if the sparsity pattern of dependencies between inputs and outputs can be determined, as this allows Jacobian compression via coloring [75, 76].

This thesis contribution proposes a novel technique that we call *NaN propagation* to trace these input-output dependencies through black-box functions. The approach exploits the fact that Not-a-Number (NaN) values are universally propagated through most floating-point numerical computations via the IEEE 754 standard. Because most floating-point math libraries since the 1970s follow this standard, this presents a fascinating way to potentially trace sparsity independent of math library or programming language. By systematically contaminating inputs to a black-box function with NaN values and observing which outputs become NaN, the sparsity graph can be reconstructed.

Concretely, the technique proceeds as follows:

1. Evaluate the black-box function on an initial input.
2. Re-evaluate the function with a single input replaced by NaN. Outputs that newly become NaN are downstream of that input.
3. Repeat for each input to trace all dependencies and (conservatively) compute the sparsity pattern of that function's Jacobian.

This simple procedure yields the complete bipartite graph between inputs and outputs (the *sparsity pattern*). The sparsity pattern enables gradient computation accelerations via simultaneous evaluation, since the sparsity of the Jacobian is known to be a strict subset of the result of this procedure. Crucially, no internal knowledge of the black-box function is required, which in theory compatibility with arbitrary external code.

Further accelerations are possible. One novel proposed acceleration, which we term "chunking", seeds multiple adjacent elements of the input vector with NaN values simultaneously. As an example: if pairs of adjacent inputs are seeded, this has the benefit of halving the number of function calls, at the detriment of providing overly-conservative sparsity information[4]. This exploits the fact that a conservative sparsity pattern can still provide a significant speedup, depending on the sparsity pattern. It also exploits the fact that human-written code tends to have a high degree of locality[5] (e.g., inputs 10 and 11 are much more likely to share a sparsity pattern than inputs 10 and 100), and hence these chunked sparsity patterns can be surprisingly accurate in practice. Thus, chunking may offer significant practical speedups by exploiting typical problem structure; a computational analogy would be how an A^* graph traversal algorithm outperforms Dijkstra's algorithm

[4] as one cannot determine which NaN input caused a given output to return NaN, so neither can be ruled out

[5] in other words, humans tend to code Discipline A in its entirety before implementing analyses for Discipline B

in practice despite identical worst-case complexity. This and other heuristics can significantly reduce the number of black-box function evaluations required to trace the sparsity pattern.

There are several subtleties to address for robustness across numerical code. First, some functions may internally handle NaN values via exceptions, preventing contamination and hence precluding this strategy. However, NaN propagation is common in most numerical libraries. Second, conditional logic can complicate tracing, but most libraries propagate NaNs through static conditional statements (i.e., a flattened if-else statement where both branches are fused). Overall, NaN propagation shows promise for automated sparsity detection through common black-box functions. Integrating this technique into the code transformation paradigm could expand its applicability.

3.4 Physics-Informed Machine Learning Surrogates for Black-Box Models

This contribution demonstrates a method of bringing complex black-box physics models into a code transformations paradigm using surrogate modeling via physics-informed neural networks. Although low- and mid-fidelity traceable models are often more than adequate for conceptual design, there are some cases where high-fidelity black-box models is critical. Hence, providing a method to bring these models into a code transformations paradigm is an important step towards boosting practicality.

As a case study, this will be demonstrated using NeuralFoil, a machine-learned airfoil aerodynamics analysis tool. Airfoil analysis is ubiquitous in aircraft design, but codes like XFoil exhibit attributes that make direct design optimization somewhat challenging [54]. Moreover, external codes like XFoil cannot directly be used in the AeroSandbox implementation of the code transformations paradigm, as XFoil is not written in Python.

NeuralFoil aims to match XFoil's accuracy using pure Python and NumPy code amenable to transformations, while avoiding these limitations. It is composed of feedforward neural networks trained on over 25 million XFoil runs, embedding physical knowledge like symmetry with respect to angle of attack and power-law drag behavior in the low-Reynolds limit[6]. Strategies for structurally embedding these physics constraints into neural networks are discussed by Zhang [104] and leveraged for this work. This allows accurate generalization to airfoil shapes outside its training set and more efficient use of training data. Eight different neural network models, with increasing levels of complexity, have

[6] i.e., Stokes flow

been created; this spectrum offers a tradeoff between accuracy and computational cost, as later shown in Table 3.3.

The airfoil geometry given to NeuralFoil's neural networks is parameterized as an 8-parameter-per-side CST (Kulfan) parameterization, with Kulfan's added leading-edge-modification (LEM) and trailing-edge thickness parameter [105, 106]. This gives a total of 18 parameters (corresponding to linear modes) to describe a given airfoil shape, which are illustrated in Figure 3.4. This parameterization was chosen as it is one of the most parameter-efficient representations of airfoil shape [107], is linear and relatively interpretable, and is common in existing aerospace tools [64].

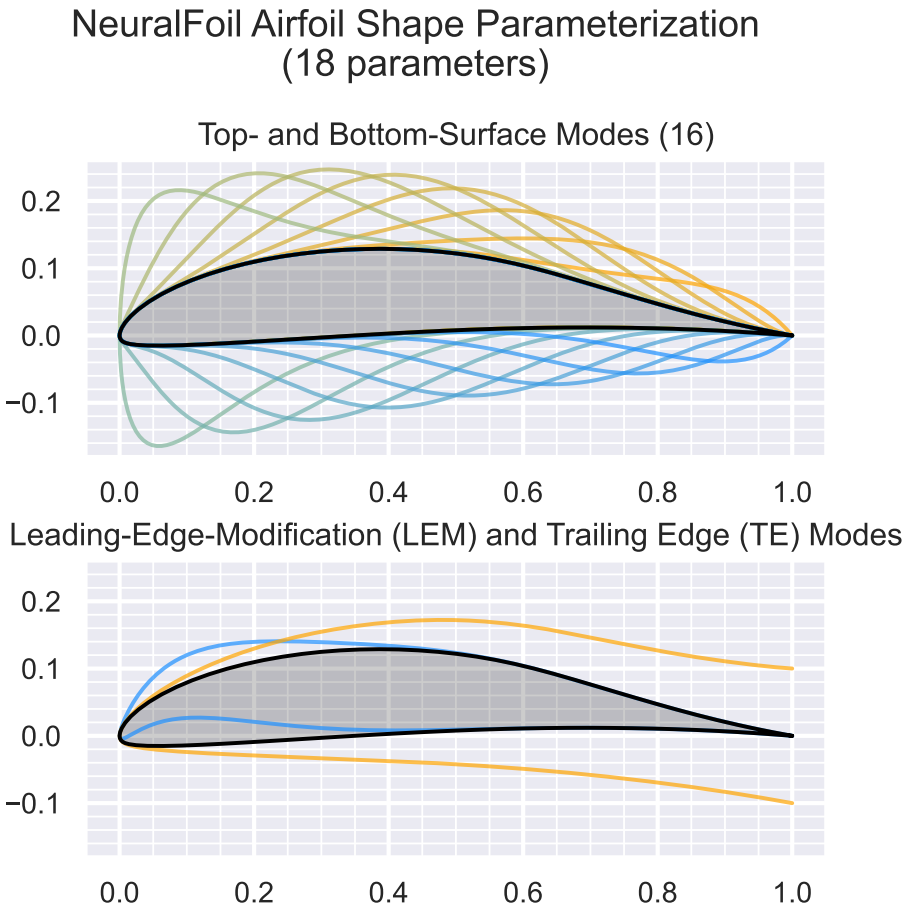


Figure 3.4: Geometry input parameterization used by NeuralFoil.

Qualitatively, NeuralFoil tracks XFoil very closely across a wide range of angles of attack and Reynolds numbers. In Figure 3.5, we compare the performance of NeuralFoil to XFoil on aerodynamic polar prediction. Notably, the airfoil analyzed here was developed from scratch for a real-world aircraft development program and is completely separate

from the airfoils used during NeuralFoil's training, so NeuralFoil isn't gaining an unfair advantage by memorizing this airfoil's performance. In this example, and in others, NeuralFoil is typically accurate to within a few percent of XFoil's predictions. NeuralFoil also has the benefit of smoothing out XFoil's "jagged" predictions (for example, near $C_L = 1.4$ and $Re = 90k$ in Figure 3.5) in cases where XFoil is not reliably converging, which would otherwise make optimization difficult. Indeed, due to the nature of the neural network structures used, NeuralFoil outputs are structurally guaranteed to be C^∞-continuous.

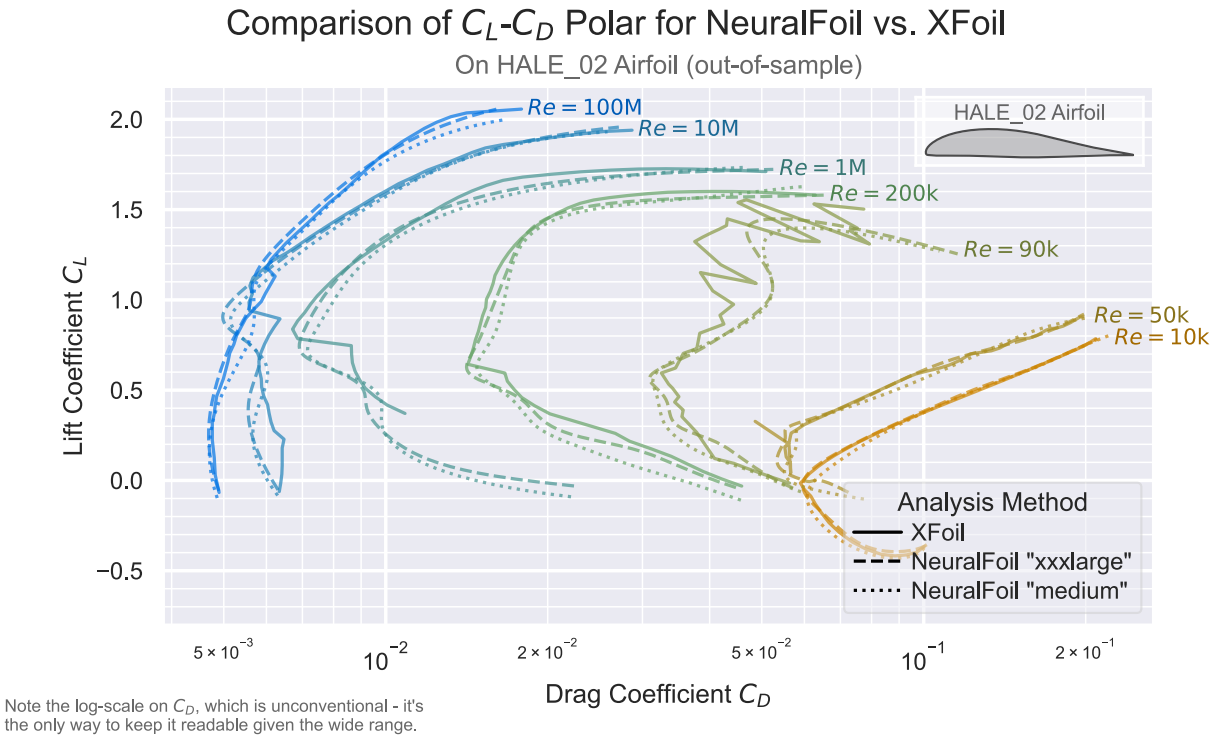


Figure 3.5: Sample validation of NeuralFoil versus XFoil.

Table 3.3 quantifies the performance of NeuralFoil models with respect to XFoil more precisely. At a basic level, the figures of merit are accuracy (here, treating XFoil as a "ground truth") and computational speed. This table details both of these considerations. The first few columns show the error with respect to XFoil on the test dataset. The test dataset is completely isolated from the training dataset, and NeuralFoil was not allowed to learn from the test dataset. Thus, the performance on the test dataset gives a good idea of NeuralFoil's performance in the wild. The second set of columns gives the runtime speed of the models, both for a single analysis and for a large batch analysis.

The benefit of using NeuralFoil as an XFoil surrogate within a code transformations framework is threefold. First, the 10-100x speedup on the analysis itself reduces compu-

Table 3.3: Performance comparison of NeuralFoil ("NF") physics-informed machine learning models versus XFoil in terms of accuracy (treating XFoil as a ground truth) and speed.

Aerodynamics Model	**Mean Absolute Error (MAE) of Given Metric, on the Test Dataset, with respect to XFoil**						**Computational Cost to Run**	
	Lift Coeff.	Fractional Drag Coeff.	Moment Coeff.	Max Overspeed	Top Transition Location	Bottom Transition Location	Runtime	Total Runtime
	C_L	$\ln(C_D)^\dagger$	C_M	$u_{\max}/u_\infty^\ddagger$	$x_{tr,top}/c$	$x_{tr,bot}/c$	(1 run)	(100,000 runs)
NF "xxsmall"	0.065	0.121	0.010	0.215	0.073	0.100	3 ms	0.190 sec
NF "xsmall"	0.042	0.075	0.007	0.134	0.039	0.055	4 ms	0.284 sec
NF "small"	0.039	0.069	0.006	0.122	0.036	0.050	4 ms	0.402 sec
NF "medium"	0.027	0.051	0.004	0.088	0.022	0.033	5 ms	0.784 sec
NF "large"	0.024	0.045	0.004	0.079	0.020	0.029	6 ms	1.754 sec
NF "xlarge"	0.023	0.043	0.004	0.076	0.019	0.028	10 ms	3.330 sec
NF "xxlarge"	0.021	0.040	0.003	0.071	0.018	0.025	13 ms	4.297 sec
NF "xxxlarge"	0.020	0.039	0.003	0.070	0.016	0.024	38 ms	8.980 sec
XFoil	0	0	0	0	0	0	73 ms	42 min

† The deviation of $\ln(C_D)$ can be thought of as "the typical relative error in C_D".

‡ This "maximum overspeed" gives $C_{p,min}$, which can be used to estimate the critical Mach number M_{crit}.

tational bottlenecks. Second, smooth predictions enable reliable gradient-based optimization without getting caught in spurious local optima. Finally, augmenting NeuralFoil with code transformations permits various acceleration techniques like vectorization and parallelization, which can offer further speedups if multiple sectional aerodynamics analyses are needed (for example, in a 3D wing aerodynamics buildup or a propeller blade-element-method).

3.5 Aircraft System Identification from Minimal Sensor Data

The flexibility afforded by code transformations enables the straightforward formulation of challenging inverse optimization problems in aerospace beyond just aircraft design. As an example application, this thesis contribution will demonstrate the use of code transformations to accurately infer aircraft performance characteristics from minimal flight data.

Here, physics-based corrections are used alongside statistical inference techniques to estimate an aircraft's power curve, aerodynamic polar, and propulsive efficiency curve simultaneously from a single short test flight. The key insight is that by framing aircraft performance reconstruction as an optimization problem over an uncertain data-generating process, one can simultaneously fit models to data while also using those models to correct the raw data for unsteady effects. This allows one to embed physical constraints on the

resulting models, significantly reducing the amount of required data. This coupled data-model inference approach draws parallels to techniques used in system identification, but its practical implementation is made simpler by the flexibility of the code transformation framework.

Concretely, conservation of energy[7] is used to construct an instantaneous aircraft power balance equation. The drag polar and propulsive efficiency are modeled as parametric functions (with initially-unknown parameters) informed by physical intuition or first-principles modeling. By posing this as an optimization problem to minimize residuals within some collapsed space (e.g., the airspeed-power space representing an aircraft's power curve), we simultaneously fit these unknown functions to the data while also correcting for unsteady flight effects. The resulting model is then used to reconstruct the aircraft's performance curves. Remarkably, this physics-informed regression approach shows excellent accuracy reconstructing the true performance curves from real-world noisy data, even when the raw data has significant contamination from unsteady flight effects and is quite limited. Additionally, by wrapping this optimization problem in a bootstrap resampling loop of the flight data samples, uncertainty bounds are obtained from an unbiased estimator. Figure 3.6 broadly illustrates the toolchain for performance reconstruction used in this contribution.

In a working example, key performance information (drag polar, propulsive efficiency curve, and power curve) is reconstructed from a short test flight of the *Solar Surfer* aircraft. This aircraft, shown in Figure 3.7a, is a remote-controlled solar-electric seaplane design with a 14-foot wingspan and as-flown all-up mass of 9.4 kg. A 260-second-long test flight was flown in a racetrack-like pattern, as shown in Figure 3.7b. Altitude, airspeed, and battery voltage and current were recorded at approximately 5 Hz. No specific attempt was made to excite the system to facilitate later system identification.

To illustrate the utility of this approach, we take the aircraft power curve (i.e., required power as a function of airspeed in steady level flight) as an example performance output of interest. Figure 3.8a illustrates the raw data recorded from the *Solar Surfer* test flight in black, and also demonstrates the inability of naive fitting without physics-informed constraints to capture performance curves. Figure 3.8b shows the corrected data; this includes not only unsteady corrections but also the on-the-fly fitted models for propulsive efficiency from an inferred advance ratio. These corrections substantially collapse the data. From this, a power curve can be simultaneously estimated, along with uncertainty bounds.

The proposed technique offers substantial advantages over traditional quasi-steady flight testing, including: (1) Performance estimates from significantly less flight time/data,

[7] including both unsteady altitude and airspeed terms

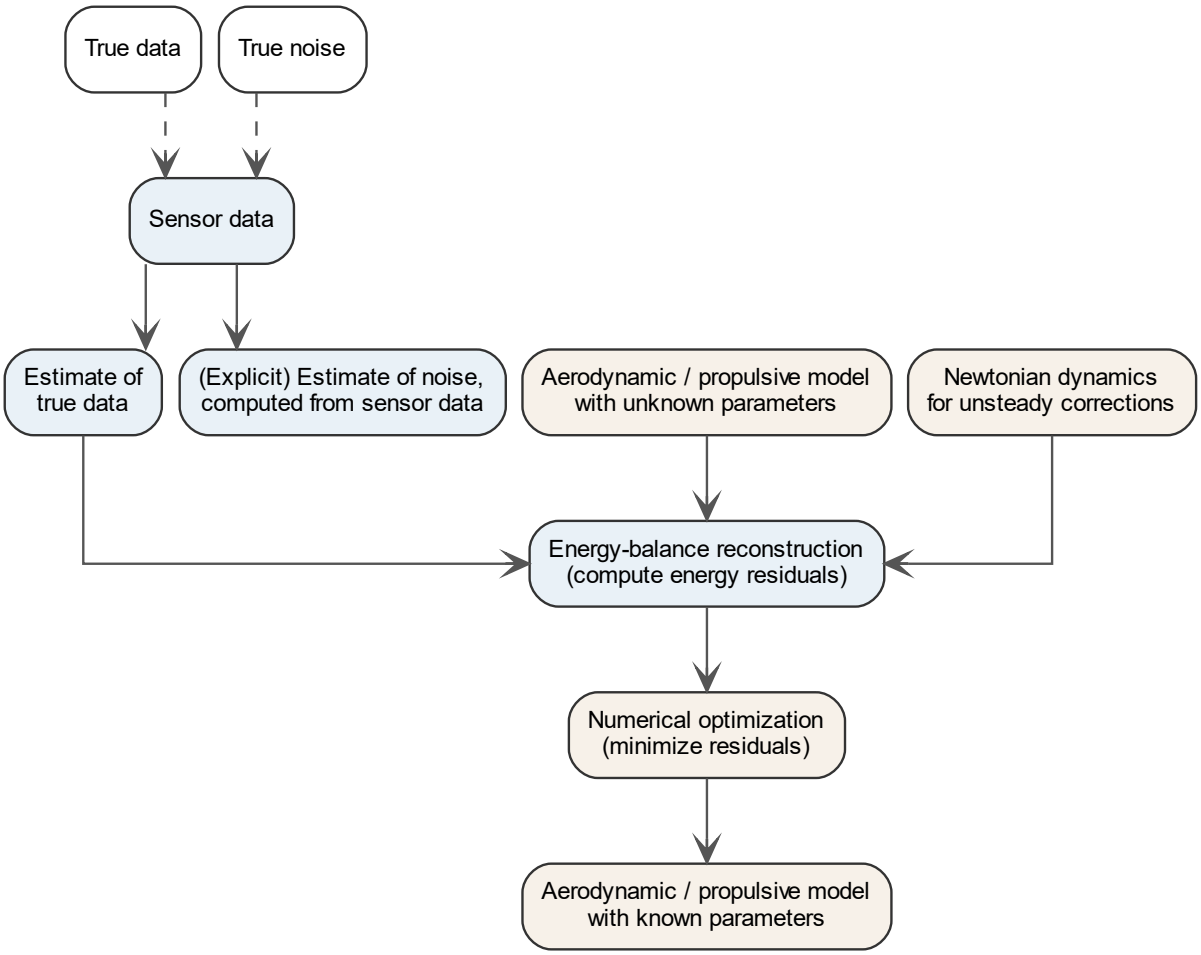
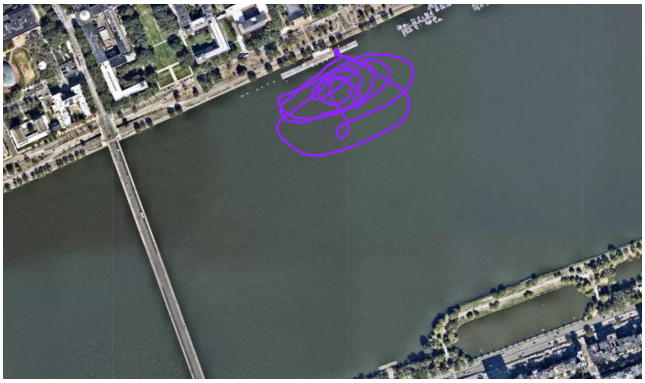


Figure 3.6: Process for inference-based performance reconstruction from flight data

(2) Rigorous uncertainty quantification, (3) The ability to incorporate data from non-steady flight segments, reducing wasted data, and (4) Reduced pilot workload. Taken together, these benefits can greatly reduce the time, instrumentation cost, and risk required to characterize the performance of an unproven aircraft design.

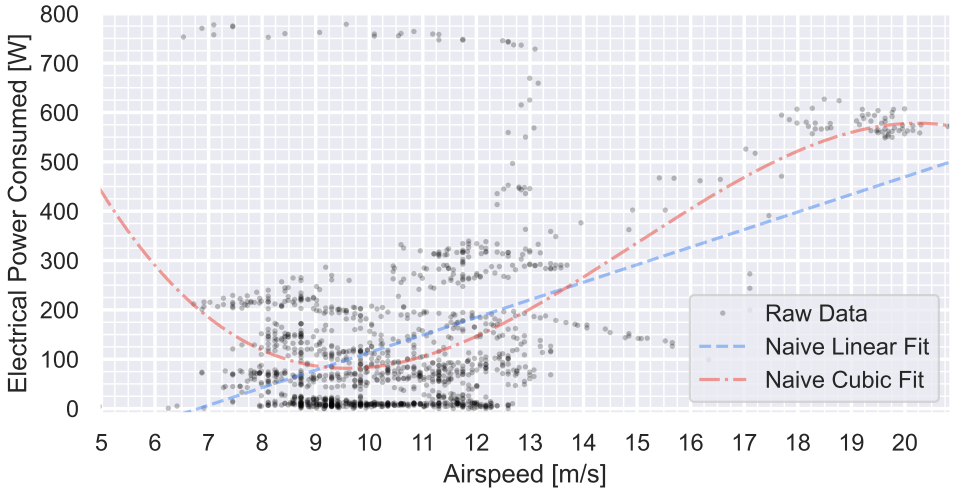
This example application demonstrates the flexibility of code transformation frameworks to tackle challenging inverse problems beyond just design optimization. It also shows the value of framing inference problems in terms of optimization when possible. These insights may have relevance even for problems not traditionally considered as candidates for optimization techniques.

(a) *Solar Surfer* in flight

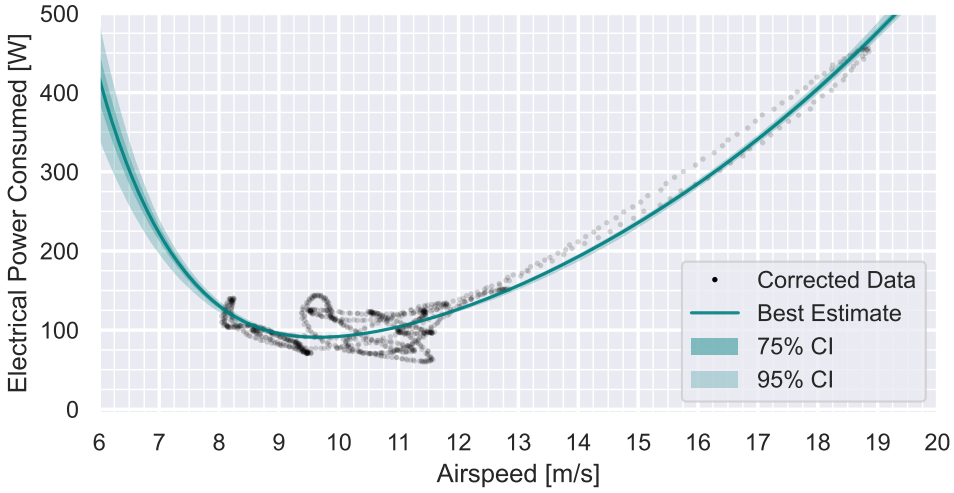


(b) Recorded GPS ground track during test flight

Figure 3.7: *Solar Surfer* aircraft test flight



(a) Raw uncorrected data for the power curve, with naive curve fitting. Uncorrected data exhibits high amounts of unexplained variance. Naive curve fits extrapolate to physically-impossible conclusions and do not capture uncertainty.



(b) Power curve with physics-informed corrections on data and model, and using noise estimator with a resampling bootstrap for uncertainty quantification.

Figure 3.8: A comparison of power curve estimation for the *Solar Surfer* airplane using naive vs. system identification methods for simultaneous data correction and performance estimation.

Chapter 4

Status and Proposed Schedule

4.1 Fields of Study and Coursework

At the first meeting with the thesis committee (Apr. 5, 2023), the following major and minor programs of study were proposed:

- Major: **Computation for Design and Optimization**
  - 16.920: Numerical Methods for Partial Differential Equations
  - 6.255: Optimization Methods
  - 16.888: Multidisciplinary Design Optimization
  - 18.337: Parallel Computing and Scientific Machine Learning
  - 16.842: Fundamentals of Systems Engineering
- Minor: **Flight Physics**
  - 16.110: Flight Vehicle Aerodynamics
  - 16.13: Aerodynamics of Viscous Fluids
  - 16.885: Aircraft Systems Engineering
- Doctoral math requirement:
  - 16.920: Numerical Methods for Partial Differential Equations
  - 6.255: Optimization Methods

The program of study was approved by the committee at the Apr. 2023 meeting, with no further coursework recommendations made. All listed classes have been completed for credit with A/A+ grades, satisfying grade requirements. The Doctoral Research and Communication Seminar course (16.995) has been completed, satisfying the department prerequisite for the thesis proposal defense.

4.2 Degree Milestones

Major degree milestones, both past and future, are listed in Table 4.1.

Table 4.1: Tentative planned timeline for PhD degree milestones.

Complete?	Date	Milestone
✓	Fall 2019	Began studies at MIT
✓	Summer 2021	S.M. thesis submitted
✓	Summer 2021	S.M. degree awarded
✓	Fall 2022	Formation of doctoral committee
✓	Spring 2023	Committee Meeting #1
	Fall 2023	Ph.D. Thesis Proposal Defense
	Spring 2024	Committee Meeting #2
	Fall 2024	Committee Meeting #3
	Fall / Winter 2024	Ph.D. Thesis Defense

4.3 Research Schedule

Table 4.2: Tentative planned timeline for PhD research milestones.

Task		Already complete?	Spring '24				Summer '24					Fall '24		
			Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Code Transformations	Introduce concepts													
	Create proof-of-concept framework													
	Benchmark against SOTA on standard optimization test problems													
	Document end-to-end aircraft design case													
	Explore additional framework features to support interpretability													
	Provide a review of practical feasibility of code transformation paradigm													
Traceable Physics Models	Build the models specified in Table 3.2													
	Document models thoroughly													
	Document challenges encountered during model implementation and any workarounds required													
Sparsity Tracing via NaN-Propagation	Introduce concepts													
	Build a minimum-working code example to demonstrate feasibility													
	Demonstrate this code example on a simple black-box aerospace-related analysis													
Physics-Informed ML Surrogates for Black-Box Models	Create ML surrogates for XFoil													
	Demonstrate optimization through these surrogates in a code transformations framework													
Aircraft System Identification	Demonstrate application of code transformations framework to system identification													
Thesis Writing														

4.4 Facilities Required

No special facilities are required for this work beyond those already available at MIT AeroAstro.

Appendix A

Comparison of MDO Framework Paradigms

This appendix aims to expand on the comparisons made in Table 3.1 by providing more details on the pros and cons of various MDO paradigms. In this table, three qualitative metrics are defined over which MDO framework paradigms are evaluated. These metrics represent framework needs derived from the barriers to industry adoption that are identified in Chapter 2. Here, we define those metrics more precisely:

1. **Ease of implementation:** How much effort is required to implement a typical aircraft design problem (from "concept idea" to "working code") in this paradigm? How much optimization or programming expertise is required, beyond the basics needed to write engineering analysis code?

2. **Computational speed and scalability:** How fast is the resulting design optimization problem to run, and how does this scale with problem size and number of disciplines? Are there other fundamental limits to scaling up analysis fidelity?

3. **Mathematical flexibility:** What kinds of restrictions are present on the mathematical form of the optimization problem? This has important follow-on effects for backwards-compatability, as highly-restrictive frameworks preclude the use of existing engineering code and instead require from-scratch rewrites.

From here, we can provide some notes on how various MDO paradigms are assessed:

Black-Box Optimization

Black-box optimization refers to the common industry approach of taking an existing performance analysis toolchain and wrapping it in an optimizer. An example of this is shown in Figure A.1. Here, the user has a "black box" function that takes in a vector of design variables and outputs a scalar objective and a vector of constraints. The user then wraps this function in an optimizer, without providing any other information (e.g., gradient or sparsity) about the function. The moniker "black box" refers to the fact that the internal workings of this analysis function are essentially opaque to the optimizer.

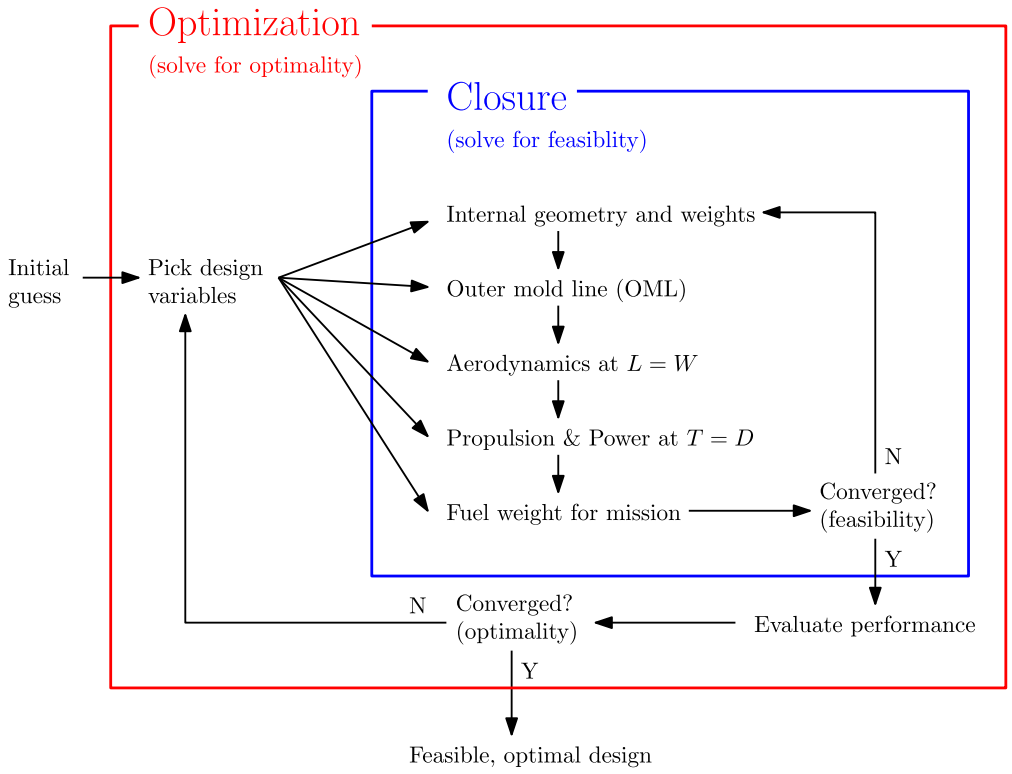


Figure A.1: A representation of a traditional "black-box" wrapped-optimization approach, which remains the dominant MDO paradigm used by industry practitioners today. Figure adapted from [78].

The main benefit of this approach is its conceptual simplicity: a black-box optimization paradigm tends to map most directly onto the "mental model" of optimization that most practicing engineers have. In this mental model, the forward problem (i.e., analysis) is the basis, and an inverse solve (i.e., optimization) is bolted on afterwards to wrap this. Because of this, black-box optimization is assessed to have the easiest path to implementation in industry. Modeling flexibility is likewise excellent due to the minimal information required by the optimizer.

However, this approach has some drawbacks, which we enumerate here:

1. Computational performance invariably degrades rapidly as the number of design variables is increased. If a gradient-free optimizer is used, this is due to the curse of dimensionality, where the size of the feasible set of the design space grows exponentially with the number of design variables. with a gradient-based optimizer, scaling is better, but it still falls sharply behind more advanced paradigms due to the bottleneck of gradient computation via finite differencing.

2. Convergence issues may exist if the wrapped analysis requires internal closure loops to be satisfied. If a wrapped analysis cannot achieve closure (i.e., feasibility) with a given set of inputs, the optimizer receives essentially no information that allows it to recover from this; this can render the optimization process brittle. Likewise, these closure loops can be inefficient – lots of computational effort is spent closing iterative feasibility loops early in the design process, when the design is far from optimal.

3. Finally, the black-box optimization approach typically forces a functional coding style (i.e., a callable data structure with defined inputs and outputs) in order to interface with an external optimizer. This can be unnatural to read and write, as engineers typically write analysis code in a procedural style. While simple analyses may allow easy conversion between functional and procedural coding styles, this task becomes substantially harder for larger codebases split across multiple files or modules.

Because of these reasons, the runtime speed and scalability of black-box optimization methods is deemed relatively poor.

Gradient-based with Analytic Gradients

Gradient-based optimization methods that are manually augmented with user-provided analytic gradients offer substantial runtime performance improvements over black-box optimization methods – indeed, to-date this is the dominant approach in deep MDO methods where runtime speed is the primary limitation [51]. However, deriving and implementing derivatives for each model can be a Herculean effort that is challenging to scale to wide MDO methods with hundreds of constituent models [55]. Moreover, this process requires significant end-user expertise and is often tedious and error-prone. Indeed, erroneous gradient calculations can be some of the most persistent and difficult errors to

detect and debug [37]. Due to the user expertise and engineering time required to implement each model in such a framework, this paradigm faces an uphill battle for conceptual design in industry.

Modeling flexibility is approaches the total freedom that black-box optimization affords, although the requirement for differentiability (more precisely, C^1-continuity) can preclude certain types of conditional logic that do not preserve this. In practice, limitations due to this restriction are rare and can usually be worked around.

Disciplined Optimization Paradigm

Disciplined optimization methods, such as geometric programming or convex programming methods, offer the ease-of-use of black-box optimization methods with the runtime speed of gradient-based methods with analytic gradients [9, 53]. Moreover, they carry notable additional benefits by virtue of their convex formulation: no initial guesses are required, and any optimum must be a global one.

These benefits are achieved by restricting the space of mathematical operators, which limits the models that can be implemented into such a framework. Because of this restriction, model flexibility is scored low. Fortunately, geometric programming maps relatively well onto many sizing relations found in aircraft conceptual design, since many of these are power-law-like relations [19]. However, even in conceptual aircraft design relations, a substantial number of exceptions to GP-compatibility exist, and this can be labor-intensive to resolve [108]. Furthermore, this model inflexibility leaves little ability to scale up the level of fidelity to include common mid-fidelity analysis elements like nonlinear systems of equations or integrators. For this reason, disciplined optimization methods are scored lower on scalability, although they are quite competitive on runtime speed when the nature of the problem allows it to be compatible with such a framework.

Code Transformations

On the metric of ease-of-implementation, code transformations are scored as identical to disciplined optimization paradigm. Both approaches allow a procedural modeling-language-like approach that tends to map closely onto existing engineering analysis code. Optimization components, like the variables, constraints, and objective, can be specified in natural-language syntax. Both are scored slightly below black-box optimization methods, since they both require understanding of certain paradigm-specific concepts. (In the case of disciplined optimization methods, this is convexity or GP-compatibility. In the case of

code transformations, this is traceability.)

Code transformations offer runtime speeds that equals or exceed those of dedicated disciplined optimization solvers on relevant problems. This paradigm also offers a pathway to medium-fidelity analysis that is often not possible with disciplined optimization methods. However, code transformations are relatively memory-hungry due to the storage of a complete computational graph; this precludes the high-fidelity analyses (e.g., RANS CFD) that are possible in an analytic-gradient paradigm. Likewise, hand-derived gradients can implement accelerations that are not yet possible with code transformations, such as more aggressive sparsity accounting, subtractive cancellation, and common subexpression elimination [37, 66, 103]. For these reasons, runtime speed and scalability falls somewhere between that of disciplined optimization methods and gradient-based methods with analytic gradients.

Finally, modeling flexibility is vastly improved over disciplined optimization methods, as the only fundamental requirement is that of C^1-continuity, similar to that of other gradient-based methods. However, mathematical operators should be drawn from a set of pre-defined primitives [67], which has the potential to reduce modeling flexibility if the numerics framework does not make appropriate syntax choices to mitigate this [73]. For these reasons, flexibility is assessed as slightly below that of gradient-based methods with analytic gradients.

Bibliography

[1] H. Ashley, "On Making Things the Best-Aeronautical Uses of Optimization," *Journal of Aircraft*, vol. 19, pp. 5–28, Jan. 1982. https://arc.aiaa.org/doi/10.2514/3.57350.

[2] G. Vanderplaats, "Automated optimization techniques for aircraft synthesis," in *Aircraft Systems and Technology Meeting*, (Dallas,TX,U.S.A.), American Institute of Aeronautics and Astronautics, Sept. 1976. https://arc.aiaa.org/doi/10.2514/6.1976-909.

[3] T. Haftka, "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments," *Springer, Structural Optimization*, Aug. 1997.

[4] I. Kroo, "Multidisciplinary Optimization Applications in Preliminary Design - Status and Directions," in *38th Structures, Structural Dynamics, and Materials Conference*, (Kissimmee, Florida), American Institute of Aeronautics and Astronautics, Apr. 1997. https://arc.aiaa.org/doi/10.2514/6.1997-1408.

[5] M. Drela, *Pros & Cons of Airfoil Optimization*, pp. 363–381. World Scientific, Nov. 1998. http://www.worldscientific.com/doi/abs/10.1142/9789812815774_0019.

[6] J. H. McMasters and R. M. Cummings, "Airplane Design - Past, Present, and Future," *Journal of Aircraft*, vol. 39, pp. 10–17, Jan. 2002. https://arc.aiaa.org/doi/10.2514/2.2919.

[7] J. Agte, O. De Weck, J. Sobieszczanski-Sobieski, P. Arendsen, A. Morris, and M. Spieck, "MDO: Assessment and direction for advancement—an opinion of one international group," *Structural and Multidisciplinary Optimization*, vol. 40, pp. 17–33, Jan. 2010. http://link.springer.com/10.1007/s00158-009-0381-5.

[8] A. Gazaix, F. Gallard, V. Gachelin, T. Druot, S. Grihon, V. Ambert, D. Guénot, R. Lafage, C. Vanaret, B. Pauwels, N. Bartoli, T. Lefebvre, P. Sarouille, N. Desfachelles, J. Brézillon, M. Hamadi, and S. Gurol, "Towards the Industrialization of New MDO Methodologies and Tools for Aircraft Design," in *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, (Denver, Colorado), American Institute of Aeronautics and Astronautics, June 2017. https://arc.aiaa.org/doi/10.2514/6.2017-3149.

[9] E. Burnell, N. B. Damen, and W. Hoburg, "GPkit: A human-centered approach to convex optimization in engineering design," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020.

[10] A. Salas and J. Townsend, "Framework Requirements for MDO Application Development," in *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, (St. Louis,MO,U.S.A.), American Institute of Aeronautics and Astronautics, Sept. 1998. https://arc.aiaa.org/doi/10.2514/6.1998-4740.

[11] Y. Ma, S. Gowda, R. Anantharaman, C. Laughman, V. Shah, and C. Rackauckas, "ModelingToolkit: A Composable Graph Transformation System for Equation-Based Modeling," *arXiv:2103.05244 (cs)*, 2021.

[12] P. D. Sharpe, "AeroSandbox: A Differentiable Framework for Aircraft Design Optimization," Master's thesis, Massachusetts Institute of Technology, 2021.

[13] MIT, "The Doctoral Program in Aeronautics and Astronautics." https://www.dropbox.com/sh/5dvic6i3m1py4q0/AABOaKVwmbtu2gz66Hg1MMKTa?dl=0&preview=NewDoCBookletOctober2021.pdf, 2021.

[14] N. Radovcich and D. Layton, "The F-22 structural/aeroelastic design process with MDO examples," in *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, (St. Louis,MO,U.S.A.), American Institute of Aeronautics and Astronautics, Sept. 1998. https://arc.aiaa.org/doi/10.2514/6.1998-4732.

[15] M. Drela, "Development of the D8 Transport Configuration," in *29th AIAA Applied Aerodynamics Conference*, (Honolulu, Hawaii), American Institute of Aeronautics and Astronautics, June 2011. https://arc.aiaa.org/doi/10.2514/6.2011-3970.

[16] M. Drela, "TASOPT: Transport Aircraft System Optimization. Technical Description." http://web.mit.edu/drela/Public/web/tasopt/TASOPT_doc.pdf, Mar. 2010.

[17] A. Yildirim, J. S. Gray, C. A. Mader, and J. R. Martins, "Performance Analysis of Optimized STARC-ABL Designs Across the Entire Mission Profile," in *AIAA Scitech 2021 Forum*, (VIRTUAL EVENT), American Institute of Aeronautics and Astronautics, Jan. 2021. https://arc.aiaa.org/doi/10.2514/6.2021-0891.

[18] N. Bons, Joaquim R. R. A. Martins, Charles A. Mader, M. McMullen, and M. Suen, "High-fidelity Aerostructural Optimization Studies of the Aerion AS2 Supersonic Business Jet," in *AIAA Aviation 2020 Forum*, 2020.

[19] W. Hoburg and P. Abbeel, "Geometric Programming for Aircraft Design Optimization," *AIAA Journal*, vol. 52, pp. 2414–2426, Nov. 2014. https://arc.aiaa.org/doi/10.2514/1.J052732.

[20] A. M. Stoll, E. V. Stilson, J. Bevirt, and P. P. Pei, "Conceptual Design of the Joby S2 Electric VTOL PAV," in *14th AIAA Aviation Technology, Integration, and Operations Conference*, (Atlanta, GA), American Institute of Aeronautics and Astronautics, June 2014. https://arc.aiaa.org/doi/10.2514/6.2014-2407.

[21] T. S. Tao, *Design and Development of a High-Altitude, In-Flight-Deployable Micro-UAV*. S.M., Massachusetts Institute of Technology, 2012. https://dspace.mit.edu/handle/1721.1/76168.

[22] K. Pitta, "Using NAS-developed tools to quiet the boom of supersonic flight." https://www.nas.nasa.gov/pubs/stories/2021/feature_X-59.html, March 2021. Accessed 10-25-2023.

[23] M. Nemec and M. Aftosmis, "Minimizing sonic boom through simulation-based design: The X-59 airplane." https://www.nas.nasa.gov/SC19/demos/demo20.html, November 2019. Accessed 10-25-2023.

[24] H. Mason, "Digital design, multi-material structures enable a quieter supersonic NASA X-plane." https://www.compositesworld.com/articles/digital-design-multi-material-structures-enable-a-quieter-supersonic-nasa-x-plane, May 2022. Accessed 10-25-2023.

[25] S. Choi, J. J. Alonso, I. M. Kroo, and M. Wintzer, "Multifidelity Design Optimization of Low-Boom Supersonic Jets," *Journal of Aircraft*, vol. 45, pp. 106–118, Jan. 2008. https://arc.aiaa.org/doi/10.2514/1.28948.

[26] Z. Lovering, "Vahana configuration trade study – part I." https://acubed.airbus.com/blog/vahana/vahana-configuration-trade-study-part-i/, 12 2016. Accessed 11-03-2023.

[27] G. Bower, "Vahana configuration trade study – part II." https://acubed.airbus.com/blog/vahana/vahana-configuration-trade-study-part-ii/, 2 2017. Accessed 11-03-2023.

[28] "Vahana trade study." https://github.com/VahanaOpenSource/vahanaTradeStudy, 2017. GitHub repository. Accessed 11-03-2023.

[29] N. Roberts, V. Samuel, and D. Colas, "FBHALE." https://github.com/facebookarchive/FBHALE/wiki, 2018. GitHub repository.

[30] B. Ozturk, W. Hoburg, N. Burnell, and C. Karcher, "Jungle Hawk Owl." https://github.com/convexengineering/jho, 2016. GitHub repository.

[31] S. Wakayama, "Blended-wing-body optimization setup," in *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, (Long Beach, CA), 2000.

[32] R. Liebeck, M. Page, and B. Rawdon, "Blended-Wing-Body Subsonic Commercial Transport," in *36th AIAA Aerospace Sciences Meeting and Exhibit*, (Reno,NV,U.S.A.), American Institute of Aeronautics and Astronautics, Jan. 1998. https://arc.aiaa.org/doi/10.2514/6.1998-438.

[33] R. Liebeck, "Design of blended wing body subsonic transport," *Journal of Aircraft*, pp. 10–25, January-February 2004.

[34] E. Torenbeek, *Advanced Aircraft Design: Conceptual Design, Analysis, and Optimization of Subsonic Civil Airplanes*. Aerospace Series, Chichester: Wiley, 2013.

[35] R. Heldenfels, "Automating the design process - Progress, problems, prospects, potential," in *14th Structures, Structural Dynamics, and Materials Conference*, (Williamsburg,VA,U.S.A.), American Institute of Aeronautics and Astronautics, Mar. 1973. https://arc.aiaa.org/doi/10.2514/6.1973-410.

[36] R. R. Heldenfels, "Automation of the Aircraft Design Process," in *International Council of the Aeronautical Sciences Congress*, (Haifa, Israel), NASA, Aug. 1974. https://www.icas.org/ICAS_ARCHIVE/ICAS1974/Page%20617%20Heldenfels.pdf.

[37] J. R. R. A. Martins and A. Ning, *Engineering Design Optimization*. Cambridge University Press, 1 ed., Nov. 2021. https://www.cambridge.org/core/product/identifier/9781108980647/type/book.

[38] M. C. Yang, "Observations on concept generation and sketching in engineering design," *Research in Engineering Design*, vol. 20, pp. 1–11, Mar. 2009. http://link.springer.com/10.1007/s00163-008-0055-0.

[39] E. Torenbeek, *Synthesis of Subsonic Aircraft Design*. Delft: Delft University Press, 1976.

[40] J. Roskam, *Airplane Design*. Ottawa, Kan.: Roskam Aviation and Engineering Corp., 1989.

[41] L. M. Nicolai and G. Carichner, *Fundamentals of Aircraft and Airship Design. Volume 1: Aircraft Design*, vol. 1 of *AIAA Educational Series*. Reston, VA: American Institute of Aeronautics and Astronautics, 2010.

[42] S. A. Niederer, M. S. Sacks, M. Girolami, and K. Willcox, "Scaling digital twins from the artisanal to the industrial," *Nature Computational Science*, vol. 1, pp. 313–320, May 2021. https://doi.org/10.1038/s43588-021-00072-5.

[43] V. Singh and K. Willcox, "Engineering Design with Digital Thread," *AIAA Journal*, vol. 56, Nov. 2018.

[44] J. R. Cruz, "Weight Analysis of the Daedalus Human Powered Aircraft," 1989.

[45] G. K. W. Kenway and J. R. R. A. Martins, "Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration," *Journal of Aircraft*, vol. 51, pp. 144–160, Jan. 2014. https://arc.aiaa.org/doi/10.2514/1.C032150.

[46] J. R. R. A. Martins and A. B. Lambe, "Multidisciplinary Design Optimization: A Survey of Architectures," *AIAA Journal*, vol. 51, pp. 2049–2075, Sept. 2013. https://arc.aiaa.org/doi/10.2514/1.J051895.

[47] P. He, C. A. Mader, J. R. Martins, and K. J. Maki, "An aerodynamic design optimization framework using a discrete adjoint approach with OpenFOAM," *Computers & Fluids*, vol. 168, pp. 285–303, May 2018. https://linkinghub.elsevier.com/retrieve/pii/S0045793018302020.

[48] J. S. Gray, J. T. Hwang, J. R. R. A. Martins, K. T. Moore, and B. A. Naylor, "OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization," *Structural and Multidisciplinary Optimization*, vol. 59, pp. 1075–1104, Apr. 2019. http://link.springer.com/10.1007/s00158-019-02211-z.

[49] N. E. Antoine and I. M. Kroo, "Framework for Aircraft Conceptual Design and Environmental Performance Studies," *AIAA Journal*, vol. 43, pp. 2100–2109, Oct. 2005. https://arc.aiaa.org/doi/10.2514/1.13017.

[50] J. Alonso, P. LeGresley, E. Van Der Weide, J. R. R. A. Martins, and J. Reuther, "pyMDO: A Framework for High-Fidelity Multi-Disciplinary Optimization," in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, (Albany, New York), American Institute of Aeronautics and Astronautics, Aug. 2004. https://arc.aiaa.org/doi/10.2514/6.2004-4480.

[51] J. R. Martins, J. J. Alonso, and J. J. Reuther, "A Coupled-Adjoint Sensitivity Analysis Method for High-Fidelity Aero-Structural Design," *Optimization and Engineering*, vol. 6, pp. 33–62, Mar. 2005. http://link.springer.com/10.1023/B:OPTE.0000048536.47956.62.

[52] Z. Lyu and Z. Xu, "Benchmarking Optimization Algorithms for Wing Aerodynamic Design Optimization," p. 18, 2014.

[53] P. Kirschen and W. Hoburg, "The power of log transformation: A comparison of geometric and signomial programming with general nonlinear programming techniques for aircraft design optimization," 2018.

[54] E. J. Adler, A. C. Gray, and J. R. Martins, "To CFD or not to CFD? Comparing RANS and viscous panel methods for airfoil shape optimization," in *Congress of the International Council of the Aeronautical Sciences (ICAS 2022)*, (Stockholm, Sweden), Sept. 2022.

[55] B. J. Brelje, *Multidisciplinary Design Optimization of Electric Aircraft Considering Systems Modeling and Packaging*. PhD thesis, University of Michigan, Ann Arbor, MI, 2021. https://deepblue.lib.umich.edu/handle/2027.42/169658.

[56] S. S. Chauhan and J. R. R. A. Martins, "Low-fidelity aerostructural optimization of aircraft wings with a simplified wingbox model using OpenAeroStruct," in *Proceedings of the 6th International Conference on Engineering Optimization, EngOpt 2018*, (Lisbon, Portugal), pp. 418–431, Springer, 2018.

[57] M. Grant, S. Boyd, and Y. Ye, "Disciplined Convex Programming," in *Global Optimization* (L. Liberti and N. Maculan, eds.), vol. 84, pp. 155–210, Boston: Kluwer Academic Publishers, 2006. http://link.springer.com/10.1007/0-387-30528-9_7.

[58] A. Agrawal, S. Diamond, and S. Boyd, "Disciplined geometric programming," *Optimization Letters*, vol. 13, pp. 961–976, July 2019. http://link.springer.com/10.1007/s11590-019-01422-z.

[59] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi, "A tutorial on geometric programming," *Optimization and Engineering*, vol. 8, pp. 67–127, May 2007. http://link.springer.com/10.1007/s11081-007-9001-7.

[60] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK ; New York: Cambridge University Press, 2004.

[61] R. Haimes and M. Drela, "On The Construction of Aircraft Conceptual Geometry for High-Fidelity Analysis and Design," in *50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, (Nashville, Tennessee), American Institute of Aeronautics and Astronautics, Jan. 2012. https://arc.aiaa.org/doi/10.2514/6.2012-683.

[62] D. S. Lazzara, R. Haimes, and K. Willcox, "Multifidelity geometry and analysis in aircraft conceptual design," in *19th AIAA Computational Fluid Dynamics Conference*, (San Antonio, TX), American Institute of Aeronautics and Astronautics, June 2009.

[63] R. Haimes and J. Dannenhoffer, *The Engineering Sketch Pad: A Solid-Modeling, Feature-Based, Web-Enabled System for Building Parametric Geometry*.

[64] R. A. McDonald and J. R. Gloudemans, "Open vehicle sketch pad: An open source parametric geometry and analysis tool for conceptual aircraft design," in *AIAA SCITECH 2022 Forum*, 2022. https://arc.aiaa.org/doi/abs/10.2514/6.2022-0004.

[65] M. Drela, "XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils," in *Low Reynolds Number Aerodynamics* (C. A. Brebbia, S. A. Orszag, and T. J. Mueller, eds.), vol. 54, (Indiana, USA), pp. 1–12, Springer, 1989. http://link.springer.com/10.1007/978-3-642-84010-4_1.

[66] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[67] C. Rackauckas, "Generalizing Automatic Differentiation to Automatic Sparsity, Uncertainty, Stability, and Parallelism." https://www.stochasticlifestyle.com/generalizing-automatic-differentiation-to-automatic-sparsity-uncertainty-stability-and-parallelism/, Mar. 2021.

[68] A. Lavin, D. Krakauer, H. Zenil, J. Gottschlich, T. Mattson, J. Brehmer, A. Anandkumar, S. Choudry, K. Rocki, A. G. Baydin, C. Prunkl, B. Paige, O. Isayev, E. Peterson, P. L. McMahon, J. Macke, K. Cranmer, J. Zhang, H. Wainwright, A. Hanuka, M. Veloso, S. Assefa, S. Zheng, and A. Pfeffer, "Simulation Intelligence: Towards a New Generation of Scientific Methods." http://arxiv.org/abs/2112.03235, Nov. 2022.

[69] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey." http://arxiv.org/abs/1502.05767, Feb. 2018.

[70] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Philadelphia: SIAM, 2. ed ed., 2008.

[71] A. Griewank, "On Automatic Differentiation," in *Mathematical Programming: Recent Developments and Applications*, Argonne, Illinois: Argonne National Laboratory, 1988. https://softlib.rice.edu/pub/CRPC-TRs/reports/CRPC-TR89003.pdf.

[72] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Autograd: Effortless Gradients in Numpy," in *AutoML Workshop*, p. 3, 2015.

[73] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[74] A. B. Lambe and J. R. R. A. Martins, "Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes," *Structural and Multidisciplinary Optimization*, vol. 46, pp. 273–284, Aug. 2012. http://link.springer.com/10.1007/s00158-012-0763-y.

[75] A. H. Gebremedhin, A. Tarafdar, A. Pothen, and A. Walther, "Efficient Computation of Sparse Hessians Using Coloring and Automatic Differentiation," *INFORMS Journal on Computing*, vol. 21, pp. 209–223, May 2009. https://pubsonline.informs.org/doi/10.1287/ijoc.1080.0286.

[76] A. H. Gebremedhin, F. Manne, and A. Pothen, "What Color Is Your Jacobian? Graph Coloring for Computing Derivatives," *SIAM Review*, vol. 47, pp. 629–705, Jan. 2005. http://epubs.siam.org/doi/10.1137/S0036144504444711.

[77] S. Gowda, V. Churavy, A. Edelman, Y. Ma, and C. Rackauckas, "Sparsity Programming: Automated Sparsity-Aware Optimizations in Differentiable Programming." https://openreview.net/pdf?id=rJlPdcY38B, 2019.

[78] M. Drela, "Simultaneous Optimization of the Airframe, Powerplant, and Operation of Transport Aircraft," in *RAeS Aircraft Structural Design Conference*, Oct. 2010.

[79] J. E. Marte and D. W. Kurtz, "A Review of Aerodynamic Noise From Propellers, Rofors, and Lift Fans," Technical Report 32-1462, NASA Jet Propulsion Laboratory, 1970.

[80] S. Sudhakar, S. Karaman, and V. Sze, "Balancing Actuation and Computing Energy in Motion Planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, (Paris, France), pp. 4259–4265, IEEE, May 2020. https://ieeexplore.ieee.org/document/9197164/.

[81] P. Mechanics, "MIT developing mach 0.8 rocket drone for the Air Force." https://www.popularmechanics.com/military/aviation/a13938789/mit-developing-mach-08-rocket-drone-for-the-air-force/, 2017.

[82] K. J. Mathesius and R. J. Hansman, "Manufacturing methods for a solid rocket motor propelling a small, fast flight vehicle," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 6 2019.

[83] SMG Consulting, "AAM reality index." https://aamrealityindex.com/aam-reality-index, 2023.

[84] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer Series in Operations Research, New York: Springer, 2nd ed ed., 2006.

[85] R. Frostig, M. J. Johnson, and C. Leary, "Compiling machine learning programs via high-level tracing," in *SYSML '18*, (Stanford, CA, USA), Feb. 2018. https://mlsys.org/Conferences/2019/doc/2018/146.pdf.

[86] D. Maclaurin, *Modeling, Inference, and Optimization With Composable Differentiable Procedures*. PhD thesis, Harvard University, Cambridge, Massachusetts, 2016. https://dash.harvard.edu/bitstream/handle/1/33493599/MACLAURIN-DISSERTATION-2016.pdf.

[87] G. K. Kenway, C. A. Mader, P. He, and J. R. Martins, "Effective adjoint approaches for computational fluid dynamics," *Progress in Aerospace Sciences*, vol. 110, Oct. 2019. https://linkinghub.elsevier.com/retrieve/pii/S0376042119300120.

[88] M. Innes, "Don't Unroll Adjoint: Differentiating SSA-Form Programs," *arXiv:1810.07951 [cs]*, Mar. 2019. http://arxiv.org/abs/1810.07951.

[89] T. MacDonald, M. Clarke, E. M. Botero, J. M. Vegh, and J. J. Alonso, *SUAVE: An Open-Source Environment Enabling Multi-Fidelity Vehicle Optimization*.

[90] ISAE-SUPAERO and ONERA, "FAST-GA: Future aircraft sizing tool - overall aircraft design - general aviation," 2021. GitHub repository. Accessed 11-13-2023.

[91] L. A. Mccullers, "Aircraft configuration optimization including optimized flight profiles," in *Recent Experiences in Multidisciplinary Analysis and Optimization, Part 1*, (Hampton, VA, United States), NASA Langley Research Center, january 1984.

[92] C. Karcher and R. Haimes, "A method of sequential log-convex programming for engineering design," *Optimization and Engineering*, vol. 24, pp. 1719–1745, Sept. 2023. https://doi.org/10.1007/s11081-022-09750-3.

[93] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, Mar. 2006. http://link.springer.com/10.1007/s10107-004-0559-y.

[94] C. Rackauckas, "Engineering Trade-Offs in Automatic Differentiation: From TensorFlow and PyTorch to Jax and Julia." https://www.stochasticlifestyle.

com/engineering-trade-offs-in-automatic-differentiation-from-tensorflow-and-pytorch-to-jax-and-julia/, Dec. 2021.

[95] H. H. Rosenbrock, "An Automatic Method for Finding the Greatest or Least Value of a Function," *The Computer Journal*, vol. 3, pp. 175–184, 01 1960.

[96] S. Kok and C. Sandrock, "Locating and characterizing the stationary points of the extended rosenbrock function," *Evolutionary Computation*, vol. 17, pp. 437–453, 09 2009.

[97] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[98] M. Drela and H. Youngren, "AVL: Athena vortex lattice." https://web.mit.edu/drela/Public/web/avl/, 1989–2023. Accessed 11-09-2023.

[99] W. F. Phillips and D. O. Snyder, "Modern Adaptation of Prandtl's Classic Lifting-Line Theory," *Journal of Aircraft*, vol. 37, pp. 662–670, July 2000. https://arc.aiaa.org/doi/10.2514/2.2649.

[100] J. T. Reid, *A General Approach to Lifting-Line Theory, Applied to Wings with Sweep*. PhD thesis, Utah State University, Aug. 2020. https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=8982&context=etd.

[101] R. Fink, "USAF Stability and Control DATCOM," Tech. Rep. AFWAL-TR-83-3048, McDonnell Douglas Corporation, Douglas Aircraft Division, 1960. for the Flight Controls Division, Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio. October 1960, revised November 1965, revised April 1978.

[102] M. Drela, "ASWING user guide." https://web.mit.edu/drela/Public/web/aswing/, 2012. Accessed: 11-09-2023.

[103] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso, "The complex-step derivative approximation," *ACM Transactions on Mathematical Software*, vol. 29, pp. 245–262, Sept. 2003. https://dl.acm.org/doi/10.1145/838250.838251.

[104] S. Zhang, *Three-Dimensional Integral Boundary Layer Method for Viscous Aerodynamic Analysis*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2022. https://dspace.mit.edu/handle/1721.1/147502.

[105] B. M. Kulfan, "Universal Parametric Geometry Representation Method," *Journal of Aircraft*, vol. 45, pp. 142–158, Jan. 2008. https://arc.aiaa.org/doi/10.2514/1.29958.

[106] B. Kulfan, "Modification of CST Airfoil Representation Methodology." https://www.researchgate.net/publication/343615711_Modification_of_CST_Airfoil_Representation_Methodology, Aug. 2020.

[107] D. A. Masters, N. J. Taylor, T. C. S. Rendall, C. B. Allen, and D. J. Poole, "Geometric Comparison of Aerofoil Shape Parameterization Methods," *AIAA Journal*, vol. 55, pp. 1575–1589, May 2017. https://arc.aiaa.org/doi/10.2514/1.J054943.

[108] T. S. Tao, *Design, Optimization, and Performance of an Adaptable Aircraft Manufacturing Architecture*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, Feb 2018.