MADE: A Multidisciplinary Computational Framework for Aerospace Engineering Education

Arjit Seth, Stephane Redonnet, and Rhea P. Liem

Abstract*—*Contribution: **A multidisciplinary computational framework to support undergraduate engineering education is introduced. It is posed as an open-source tool that students can use to understand and apply computational methods commonly required in engineering problems.**

Background: **This framework development is motivated by past experiences of teaching courses as segregated disciplines within an aerospace engineering curriculum. Reviewing pedagogical approaches shows that current university practices do not adequately emphasize or demonstrate the importance of interdisciplinary relationships. This framework is also motivated by the apparent need for engineering graduates to study these relationships by learning computational engineering via programming, in which proficiency is becoming increasingly important for their future careers.**

Intended Outcomes: **1) To effectively teach multidisciplinary engineering concepts to students via the integration of cross-disciplinary content from various courses and 2) To improve computational and programming literacy among undergraduate students in engineering studies.**

Application Design: **A multidisciplinary design optimization approach is adopted to develop a framework called Multidisciplinary Aircraft Design Education (MADE), which provides functionalities for studying engineering disciplines within a computational programming environment. MADE makes the related course contents more accessible and intuitive to students by using reactive computational notebooks in lectures, tutorials, and assignments.**

Findings: **MADE is assessed based on the learning outcomes and feedback from undergraduate students of two courses within an aerospace engineering curriculum. These assessments indicate an improved understanding of multidisciplinary concepts and their applications via computational programming, as reflected in the positive responses from the students.**

***Index Terms*—Aerospace engineering, engineering curriculum, multidisciplinary design, reflection.**

Manuscript received 9 February 2023; revised 22 May 2023; accepted 27 May 2023. Date of publication 21 June 2023; date of current version 1 December 2023. This work was supported in part by the Teaching and Learning Innovation Projects Initiative under Project 108E-1920 [2020–22] and Project 131E-2021 [2021–23], and in part by the Center for Education Innovation from The Hong Kong University of Science and Technology. *(Corresponding author: Rhea P. Liem.)*

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the Center for Education Innovation at the Hong Kong University of Science and Technology.

The authors are with the Department of Mechanical and Aerospace Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: ajseth@ust.hk; redonnet@ust.hk; rpliem@ust.hk).

This article has supplementary downloadable material available at https://doi.org/10.1109/TE.2023.3281825, provided by the authors.

Digital Object Identifier 10.1109/TE.2023.3281825

I. Introduction

HIGHER engineering education provides the gateway for students to develop successful careers in industry and academia. For this reason, when designing course curricula and pedagogical approaches, it is critical to consider the current trends and, ultimately, future practices in the industry. This article addresses two important aspects of the current curricula that can be strengthened to make engineering undergraduate students more "future ready."

First, modern engineering practices involve various scientific disciplines that are tightly interconnected. Due to these interdisciplinary relationships, analyzing each discipline in isolation is not sufficient. Consequently, tackling realistic problems often necessitates the deployment of multidisciplinary design optimization (MDO) methods [1]. Thus, it is essential to translate such multidisciplinary paradigms into pedagogical approaches early enough in the educational process to equip students with the necessary skills for analyzing complex engineering systems.

Second, it is becoming increasingly desirable for engineering students to develop skills in programming and computational methods in order to become more competitive in the workforce, be it in industry or academia. Indeed, the past decades have seen most engineering fields adopting computational methods as powerful and relatively inexpensive means of conducting research, development, and production. Furthermore, enriching the undergraduate engineering curricula with the teaching of computational methods may also support the introduction of MDO concepts to students, as computational methods play a pivotal role in MDO practices.

Aerospace engineering is an apt representation of a field where the above two complementary aspects are particularly relevant, as it requires extensive knowledge of multiple disciplines and scientific computing. The high complexity of this field is particularly related to the many tightly coupled disciplines involved, including but not limited to aerodynamics, structures, propulsion, and flight dynamics. Despite its multidisciplinary nature, the current paradigm for teaching it is to cover each of these "fundamental" disciplines in separate courses. This type of learning process can lead to a fragmented understanding of the whole system, particularly because multidisciplinary thinking beyond the subject matter is generally considered out of scope in each course. This pedagogical challenge is common, especially in an aircraft design course, often offered as a standalone in most curricula, which requires detailed knowledge of the above disciplines. Notably,

© 2023 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

the current foundation of undergraduate curricula has not yet adequately introduced the multidisciplinary aspects of aircraft design, despite MDO being commonly applied in its industrial and research practices [2], [3].

The slow adoption of multidisciplinary paradigms could stem from the lack of a systematic introduction to computational methods, which are known to be difficult for students to learn due to various reasons. A major issue is that performing multidisciplinary analysis usually entails using multiple disciplinary software, which is not straightforward. In aircraft design, for instance, aerodynamic and structural aspects would typically be analyzed sequentially, each one using specific software. However, the two disciplines are strongly tied to each other (e.g., aerodynamic loads cause structural deformations, which in turn alter the aerodynamic loading). Ideally, these two aspects should be analyzed together in a unified manner. Introducing an MDO framework that is catered to education could help address the underlying pedagogical complexities of such multidisciplinary analyses. Moreover, doing so alongside the introduction of computational methods and programming practices could help improve students' proficiency and understanding of the relevant computational tools.

To this end, a new pedagogical approach is proposed in this article, which aims at familiarizing undergraduate students with multidisciplinary aspects and modern computational methods of aerospace engineering. Doing so requires the development and assessment of a unified computational platform for aerospace engineering education, namely, the Multidisciplinary Aircraft Design Education (MADE) framework. The details of this initiative are presented in this article, which is organized as follows. First, Section II provides an overview of prior, noteworthy pedagogical approaches (along with relevant software) which have been used for teaching aerospace engineering disciplines. In Section III, MADE is introduced as a computational framework specifically tailored for teaching aerospace engineering disciplines at an undergraduate level. Section IV describes how the framework was applied to two specific courses offered by the Department of Mechanical and Aerospace Engineering at the Hong Kong University of Science and Technology (HKUST), along with the specific feedback and evaluation from students. Finally, some conclusions and perspectives are presented in Section V.

II. Educational Approaches to Aerospace Engineering

Educational approaches for aerospace engineering disciplines have been previously documented in the literature. For example, Brodeur et al. [4] described problem-based learning approaches used in courses at the Department of Aeronautics and Astronautics at the Massachussetts Institute of Technology (MIT), emphasizing the need for collaboration and hands-on experiences when tackling increasingly complex engineering design problems. Concerning aircraft design education, its traditional educational practices have been assessed and tentatively improved. For instance, Young [5] explored their pedagogical aspects, stating that aircraft design is a unique subject requiring special considerations for students to develop multidisciplinary engineering skills. Scholz provided an overview of newer, computer-based educational practices for aircraft design [6], outlining that existing software at the time did not serve as efficient teaching tools. To explore a technological paradigm shift in undergraduate education, Okutsu et al. [7] investigated an approach to teaching aerospace engineering via a "virtual world" (involving a 3-D digital environment) and compared it to the traditional in-person, classroom approach. They noted that traditional teaching practices could be extended to the virtual environment, which indicates that digitalized education practices may offer a feasible alternative to the classroom environment (especially in some circumstances such as the recent COVID-19 pandemic), albeit not leading to any revolutionary paradigm shift. Cummings et al. [8] discussed how teaching aircraft design might offer a unique opportunity for introducing students to some of the key concepts pertaining to the field of optimization, such as "design space," "constraint equations," and so on. To this end, however, to the best of the authors' knowledge, most aircraft design courses rely on simplified equations that were statistically derived from historical data of aircraft, even when used along with more modern (i.e., computer-based) platforms. Such empirical equations are tailored to each discipline (aerodynamics, structures, etc.), independent of other disciplines. The statistical-based processes that are most commonly employed in undergraduate aircraft design education include the design and sizing methods of Raymer [9], Roskam [10], and Nicolai and Carichner [11]. These empirical approaches simplify the physical and mathematical formulations, thereby allowing one to perform analyses pertaining to the initial stages of the design process when information and design parameters are still scarce. However, the simplicity, while helpful, does not offer the necessary physical intuition that can help students understand the complexities of aircraft design and the inherent multidisciplinary tradeoffs in its process.

A. Computer-Driven Approaches

In more recent educational approaches, students are being taught advanced computational methods—along with programming languages—to apply the knowledge covered in the lectures to practical problems. It stands to reason that an aerospace engineering curriculum would benefit from increased use of computational frameworks. In that regard, the AeroPython [12] and CFDPython courses [13] recently proposed by Barba et al. respectively introduce concepts in fundamental aerodynamics and computational fluid dynamics, using programming languages. Over the past decades, various software have also gained popularity in the field of aerospace engineering. Some of them provide interactive graphical user interfaces (GUIs) with a variety of features for the purposes of research, which can also be beneficial for educational purposes. Popular examples include RDSwin-Student [9], ASWING,[1] OpenVSP [14], MATLAB's Aerospace Toolbox,[2]

[1] http://web.mit.edu/drela/Public/web/aswing/

[2] https://www.mathworks.com/en/products/aerospace-toolbox.html

AVL,[3] and XFLR5.[4] In addition to these software, there are also some modern engineering frameworks that emphasize writing code using programming languages instead of relying on GUIs, e.g., SUAVE [15], and OpenAeroStruct [16] created with the OpenMDAO framework [17].

However, despite the advantages they offer to research activities, all of the above computational frameworks have a common limitation for educational purposes. In particular, the scope of knowledge needed to use these tools effectively is too large to fit within a single academic semester. Most of them have also been developed in programming languages, such as C++ or Fortran, which have steep learning curves, making them less suitable as introductory programming languages for students compared to modern ones, such as Python or Julia. Furthermore, their users may also require a sophisticated understanding of advanced programming and numerical analysis concepts, such as object-oriented programming, nonlinear systems analysis, differentiation methods, and optimization architectures from relatively modern sources [1], which are neither commonly nor extensively covered in undergraduate aerospace engineering education at present. In conclusion, these software may not be suitable for introducing students to MDO practices and computational methods.

B. Current Issues and Proposed Solutions

This section summarizes the critical issues commonly faced by students in solving multidisciplinary engineering problems using the previously described approaches. For each of these issues, a possible solution is formulated.

Lack of Intuition Regarding Multidisciplinary Aspects: Students typically struggle with developing intuitive insights into the physics of an engineering system and its inherently interdisciplinary nature, even if they have sufficient domain knowledge of the separate disciplines from their coursework. To address this, demonstrating a set of realistic engineering examples while incorporating multiple disciplines would be beneficial. Additionally, providing interactive demonstrations allows students to perform their own investigations. This process would be more instructive and would assist their self-learning.

Black-Box Treatment of Software: As noted previously, multiple software exist for analyzing domain-specific disciplines. Students tend to use these software as "black boxes" without sufficiently understanding the underlying assumptions and details. To overcome this issue, it would be beneficial to have a transparent computational framework so that students can view such details and obtain relevant theoretical insights. Detailed and extensive documentation, examples, and tutorials should also be provided to support students' learning.

Complex Workflows With Iterative Processes: Assuming that some standard tools are provided for analyzing specific disciplines, then having the correct interfaces between them is critical for conducting multidisciplinary analyses. In this context, adopting a spreadsheet framework (e.g., in Microsoft Excel) to copy-paste and manipulate results generated from multiple computational tools is an error-prone process. Such issues could be mitigated by employing a framework that automatically and consistently keeps track of the results generated by those tools.

The MADE computational framework is developed to tentatively address the issues listed above by facilitating the teaching of modern engineering at an undergraduate level while easing its incorporation into current university curricula. The primary objective is to integrate the analyses of various disciplines pertaining to aerospace engineering education within a unified framework. This framework can then be used by instructors and students for their teaching and learning purposes, respectively, with special emphasis put on the computational aspects. A secondary objective is to implement a demonstrative case study (in the form of an aircraft design problem) to be used as a common, representative benchmark in single- and multidisciplinary courses. This demonstration would help enhance students' understanding of the multidisciplinary nature of engineering systems and how to tackle them properly.

III. Methodology

This section describes the methodology of the proposed MADE framework, whose paradigm and architecture are outlined in Section III-A, and computational details are described in Section III-B.

A. Conceptual Paradigm and Overall Architecture

The MADE framework is oriented toward aerospace vehicle design and analysis, as most of the fields taught in an aerospace engineering curriculum exhibit applications in this context. It primarily consists of physics-based computational models and relies on empirical equations (statistical models) only when necessary. Notably, combinations of these physical and statistical models pertain to "hybrid" methods within the framework, which are commonly utilized for applications in aircraft design.

MADE is primarily designed for performing multidisciplinary computational analyses in a modular setup following common MDO practices. Fig. 1 illustrates features of this modular framework with a typical aerospace vehicle design problem, depicted through the extended design structure matrix (XDSM) representation [18]. The purple pillbox represents the problem to be solved as an optimization process. The green and peach blocks along the diagonal depict the corresponding MADE disciplines (called modules) as covered at present, namely, geometry, aerodynamics, structures, propulsion, and flight dynamics. The off-diagonal entries (in gray) represent the coupling variables that connect these different disciplines, thus translating the multidisciplinary nature of the problem. The inputs from the top (in white) represent the relevant data specific to the problem, and the arrows (in black) depict the execution of the solving process. It should be noted that, at the current stage, the framework does not cover all the disciplines which would be necessary to complete an aerospace vehicle design, such as air traffic management operations and cost estimation, among

[3] http://web.mit.edu/drela/Public/web/avl/
[4] http://www.xflr5.tech/xflr5.htm

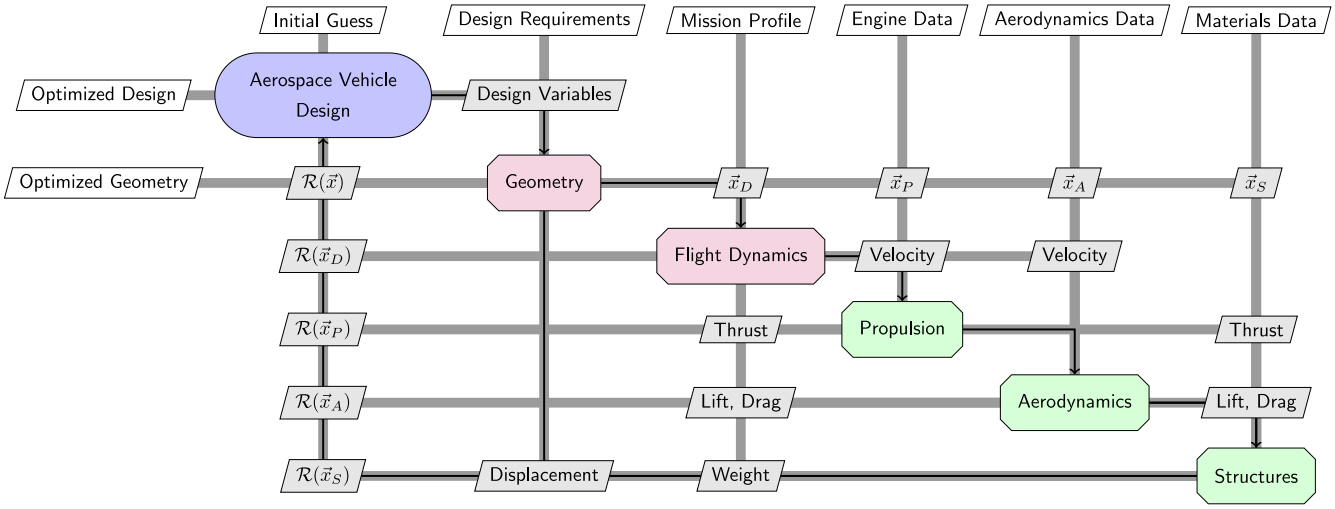


Fig. 1. XDSM for the design of an aerospace vehicle with the relevant aerospace engineering fields and their interdisciplinary couplings in the MADE framework.

many others. On that note, the aim of MADE is to satisfy the educational objectives without introducing overwhelming levels of complexity concerning real-world applications.

Using the modules together in a multidisciplinary context enables students to understand the interdisciplinary tradeoffs between different disciplines in a complex system such as an aircraft. The intuition developed via such observations can, in turn, help students make realistic design decisions. As a corollary, single-disciplinary analyses can still be performed independently thanks to the modular structure. For example, the modules pertaining to aerodynamics or structures can be used independently within their associated courses. Similarly, the flight dynamics module can be used in a course on aircraft stability, and so on. This setup makes it more convenient to adopt the framework across several courses simultaneously in the current university curricula. Such a flexible setup would also be helpful for multiple instructors to prepare exercises and assignments involving computational content related to their respective courses.

B. Implementation Details

The MADE framework is implemented using the Julia programming language [19], which has been adopted worldwide over the years by some large organizations, such as NASA [20] and NumFOCUS,[5] to name a few. Compared to more traditional languages, Julia offers compelling advantages, most of which align with the educational requirements previously listed in Section II. First, it is open source, which constitutes the best option to avoid commercial license fees that may limit its adoption. Second, Julia comprises a single programming language that allows one to conveniently write readable and efficient code. Third, it supports principal operations of scientific computing, such as linear algebra, automatic differentiation, and parallel processing, making it suitable for engineering studies. Fourth, it features convenient plotting libraries for visualization, making it particularly well-suited for academic lectures and tutorials, which often rely on graphical representations.

The kernel of the MADE framework is a Julia package named AeroFuse,[6] which comprises a set of analysis tools for aerospace engineering disciplines. Particularly, AeroFuse incorporates methods pertaining to aerospace engineering with varying fidelity to perform fast computational analyses. It combines physics-based models with statistical correlations to determine common quantities of interest in various aerospace disciplines. In aerodynamics, for instance, AeroFuse incorporates typical low-fidelity numerical approaches (inviscid panel methods) to obtain aerodynamic forces and moments [21]. To evaluate aircraft performance, AeroFuse incorporates traditional empirical methods, which are taught as part of aircraft design courses [9], [10], [11]. For aircraft structures, AeroFuse integrates finite-element methods that allow computational modeling of structural displacements [22], as commonly taught within undergraduate curricula in engineering. Regarding propulsion, AeroFuse employs methods based on the blade-element and momentum theory models [23], which are commonly covered in undergraduate courses focusing on propulsion and aircraft design (e.g., for propeller aircraft and helicopters). Finally, concerning flight dynamics, AeroFuse is equipped with the complete set of equations of motion under a rigid body assumption [24]. As mentioned, all of the methods listed can either be used independently, or combined with each other in a unified framework for conducting multidisciplinary studies based on modern research [1], [3], [16]. For accessibility, AeroFuse is hosted online on GitHub as a public repository,[7] available for anyone to download and use under an open-source license. The attributes of AeroFuse are summarized in Table I, which are compared against other common software listed in Section II-A. In summary, AeroFuse distinguishes itself from other commonly used software by offering an open source, multidisciplinary aircraft design computational environment written in a single programming language

[5]https://numfocus.org/project/julia

[6]Previously known as "AeroMDAO."

[7]https://github.com/HKUST-OCTAD-LAB/AeroFuse.jl/

TABLE I
COMPARISON OF ATTRIBUTES OF RELEVANT SOFTWARE FOR ASSESSING SUITABILITY IN AEROSPACE ENGINEERING EDUCATION WITH PROGRAMMING

Software	Attributes				
	Open-Source License	Multidisciplinary Analyses	GUI	API	Programming Languages & Frameworks
Athena Vortex Lattice (AVL)	✓	✗	✓	✗	Fortran
XFLR5	✓	✗	✓	✗	C++
ASWING	✗	✓	✓	✗	Fortran
MATLAB Aerospace Toolbox	✗	✓	✗	✓	MATLAB
RDSwin Student	✗	✓	✓	✗	Unknown to the authors at this time
OpenAeroStruct	✓	✓	✗	✓	Python, OpenMDAO
SUAVE	✓	✓	✗	✓	Python, C++
OpenVSP	✓	✓	✓	✓	C++, AngelScript, Python
AeroFuse	✓	✓	✗	✓	Julia

(Julia) with a simple application programming interface (API). Its documentation is also accessible online,[8] which provides tutorials, theory, and implementation details for reference. All of this enables students to implement course content in a programming language with a gentler learning curve.

To make it easier to adopt by students and instructors, the MADE framework comes with computational notebooks. Besides the intuitive, hands-on features they offer to instructors, these notebooks allow students to quickly adapt to programming in AeroFuse using Julia. Particularly, computational notebooks are helpful for tackling complex workflows by segmenting large problems into smaller iterative steps of executable code. The computational notebook environment used is a Julia package called `Pluto.jl` [25]. These "Pluto notebooks" provide instantaneous feedback in the workflow when any code is changed, and encourage good programming practices to reduce common mistakes. These two facets greatly help beginning programmers to reason about and debug their code. Finally, Pluto offers convenient features for displaying code, documentation, text, mathematics, and plots altogether.

In a nutshell, thanks to its AeroFuse kernel and Pluto notebooks, the MADE framework provides an intuitive computational environment for students to learn single- and multidisciplinary analyses pertaining to aerospace engineering via programming.

IV. APPLICATIONS AND FINDINGS

This section discusses the application of MADE to two courses that are taught at HKUST, as part of the curriculum for undergraduate students majoring in aerospace engineering (cf. Section IV-A). The feedback from students and their learning outcomes are evaluated in Sections IV-B and IV-C, respectively, to assess the methodology of the framework.

A. Applications to Aerospace Engineering Courses

The MADE framework was deployed within two aerospace engineering courses that are taught at HKUST, as part of the curriculum for undergraduate students majoring in aerospace engineering. It was first introduced in the aircraft performance and stability (APS) course, where students were provided the framework to complete a group project. This allowed them to become familiar with a computational programming environment, whilst using the framework in a single-disciplinary manner. In a subsequent semester, the platform was further deployed in the hands-on, project-based aircraft design (AD) course. Within this AD course, the framework was extensively employed, both by the instructors for their content delivery and by the students for completing their semester-long AD group project, which involved extensive multidisciplinary analyses.

First, when initially employed within the APS course, the MADE framework made it easier for students to understand some of the more technical concepts and theories pertaining to APS, which are challenging to grasp at an undergraduate level. For instance, using AeroFuse with Pluto notebooks, students were shown how to program and perform typical operations pertaining to APS, such as manipulating coordinate transformations via rotation matrices, integrating equations of motion, determining longitudinal- and lateral-directional flight modes (phugoid, short-period, spiral, etc.). Upon this, students were asked to use MADE for performing an in-depth analysis of the performance and stability characteristics exhibited by a given aircraft (B747) throughout an entire mission (from take-off to landing), with only minimal information provided (e.g., aerodynamic coefficients and stability derivatives under specific flight conditions). Finally, they were required to document their outcomes within a detailed report whilst providing their developments using MADE in a Pluto notebook, both of which were counted equally toward the assignment's mark. Overall, students performed very well in this assignment.

When further deployed within the AD course, the framework helped students to grasp the intricate complexities underlying AD, owing to the many disciplines involved. For their projects, the students were tasked to develop their designs—based on a hypothetical set of mission requirements—where MADE was used to generate relevant technical data and plots. From the beginning of the course, the Pluto notebook environment was used in conjunction with the AeroFuse package to teach and demonstrate concepts related to AD in lectures and tutorials. In this process, the interactive features of Pluto notebooks notably helped students become familiar and comfortable with programming in a relatively short period of time. Throughout the course,

[8]https://hkust-octad-lab.github.io/AeroFuse.jl/stable/

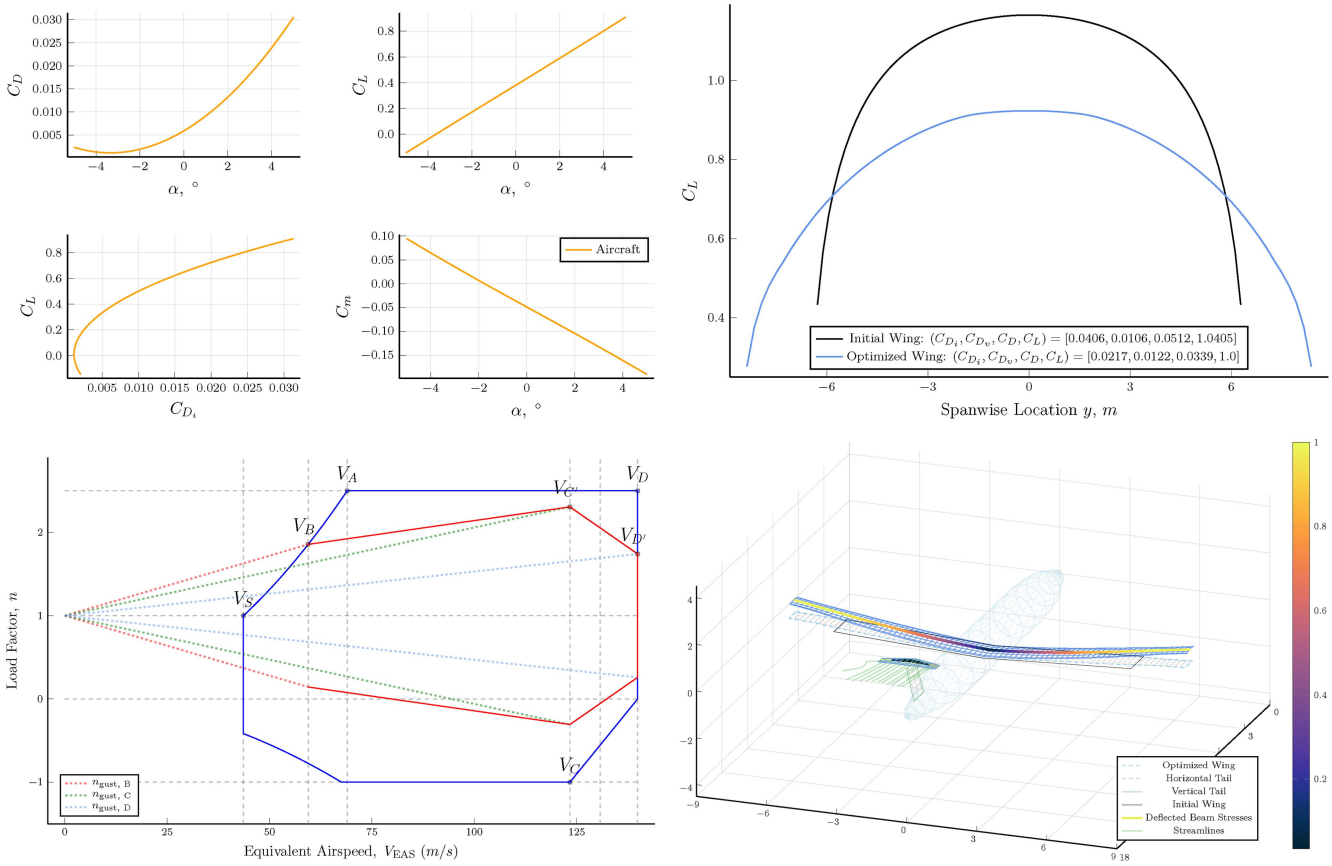


Fig. 2. Demonstrative applications of AeroFuse to preliminary AD, performance, stability analyses, and optimizations.

AeroFuse's modules (discussed in Section III-B) were used by the instructors for demonstrating aerodynamic, structural, performance, and stability analyses with a representative AD problem as a case study. These analyses were subsequently performed by the students in their projects. By observing how modifying a variable in one discipline could impact the other disciplines, students gained insight into the multidisciplinary nature of AD and its inherent interdisciplinary tradeoffs.

As an illustration of such results, Fig. 2 depicts the application of these models to the design of a business jet aircraft. It shows the performance, aerodynamic, and structural optimizations of the geometry altogether. An example of such multidisciplinary effects is evidenced by the bottom right figure, where the impact of aerodynamic loading on the wing is correctly reflected together with its structural deflection. Particularly due to this deflection that alters the wing's geometry, the aerodynamic loading—which depends on the geometry and, in turn, causes the structural deflection—is simultaneously affected. Fig. 3 shows a photograph of a group of students with the Pluto notebook used to complete their AD project with the MADE framework at HKUST.

As demonstrated, MADE serves as a unified framework for students to learn and conduct multidisciplinary engineering studies via programming within a university curriculum. The framework also provides a representative AD benchmark problem with intuitive visual representations as intended.

B. Students' Feedback

To evaluate the effectiveness of the MADE framework and its integration within the curriculum, specific feedback was sought from the students. To this end, surveys were designed and conducted at the end of each course, this being done in partnership with educational experts from the Center for Education Innovation (CEI) at HKUST. First, an assessment was conducted during the APS course to evaluate the extent to which MADE actually increased the students' confidence in solving typical problems pertaining to APS through programming. This was assessed through a paired samples t-test via surveys that were conducted before and after the students had performed the course's group project using MADE. Fig. 4 summarizes the outcomes of this two-step survey, in which 20 out of 45 students responded to the post-test. The results reveal a quasi-systematic and statistically significant increase in the level of confidence expressed by the students. Entries that did not demonstrate significant differences pertain mainly to simpler APS aspects, such as coding the coordinate transformations or the ground roll equations of motion. This may indicate that students already felt confident enough to perform these tasks even without MADE. For more complicated APS aspects they were asked about

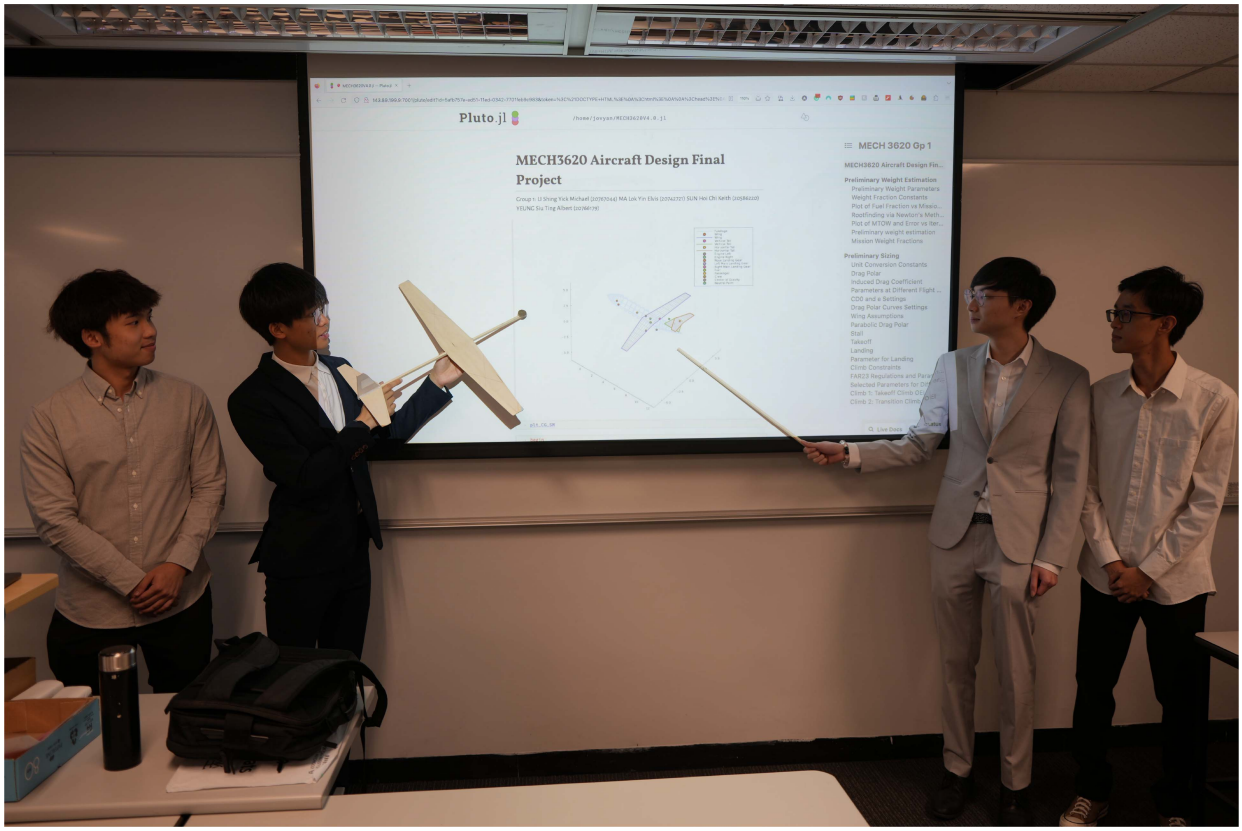


Fig. 3. Students presenting the results of their project using the MADE framework (AeroFuse with Pluto notebooks) in the AD course of Spring 2022/23 at HKUST.

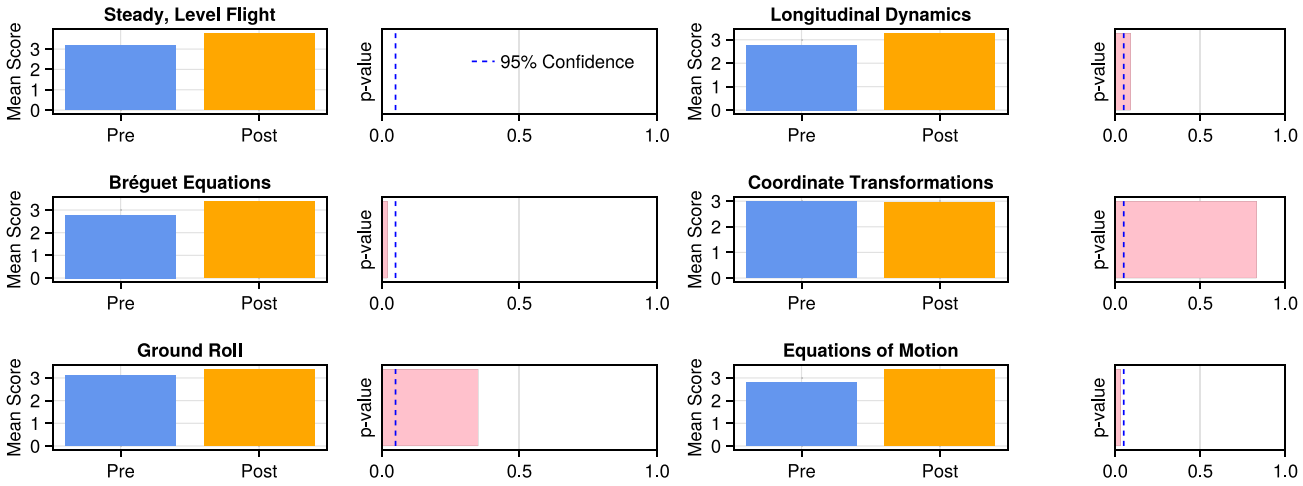


Fig. 4. Results of the survey on confidence in programming of concepts taught in the APS course, as expressed by the students before and after using MADE. The results compare the students' level of confidence, as expressed before (label "Pre") and after (label "Post") their introduction to and use of the MADE framework.

(e.g., coding the equations of motion), the students expressed increased confidence in programming, with the mean score rising from 2.78/5 (pre-MADE) to 3.39/5 (post-MADE), thereby achieving the objectives of this work.

During the subsequent AD course, a survey was conducted to evaluate the extent to which MADE improved comfort in programming. The students were also asked about their use of MADE in the group projects for gathering more subjective, end-user feedback. These two facets were particularly important for evaluation, as this course strongly relies on implementing computational methods via programming to study and develop ADs. Fig. 5 consolidates the results of this survey, which was taken by 15 out of 26 students. Notable, although not outstanding, improvements were observed, with a *minimum* rating of 4/10 regarding comfort in programming after MADE was introduced, compared to the 2/10 rating obtained prior to that. More significantly, the percentage range of the results generated using MADE in their project reports was 30%–100%, with a weighted average of 64% (the arithmetic mean weighted by the number of

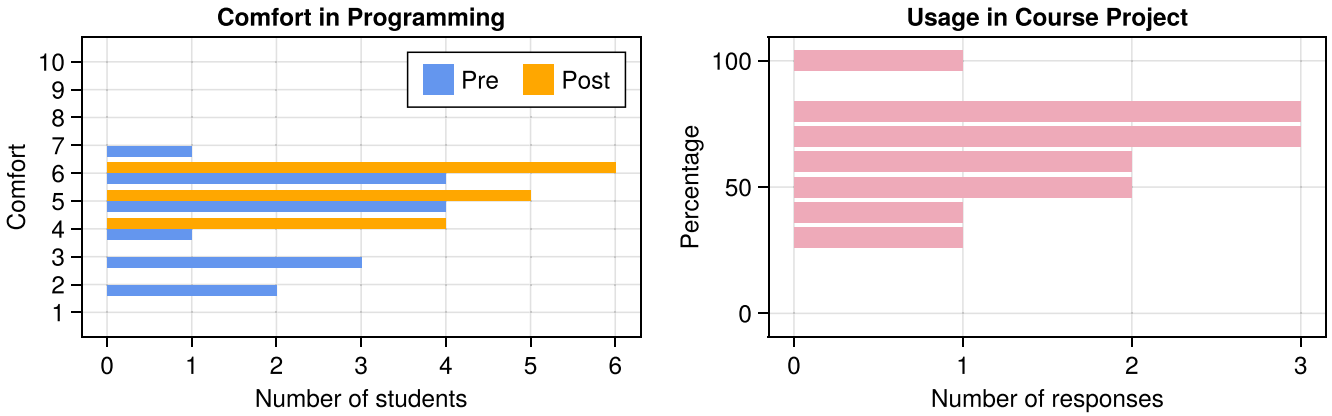


Fig. 5. *Left:* Results of the survey on students' comfort in programming before (label "Pre") and after (label "Post") their introduction to and use of the MADE framework in the AD course. *Right:* Students' usage of MADE in the AD course projects.

TABLE II
STUDENTS' REFLECTIONS ON THE MADE FRAMEWORK

Reflections	Minimum	Maximum	Mean	Standard Deviation
The platform helped me become more aware of the interdisciplinary relationships in the process of designing aircraft.	3	5	4.21	0.579
The platform helped me better realize the importance of computational methods and solvers in modern-day engineering.	3	5	4.43	0.646
The platform helped me appreciate programming more in solving real-world engineering problems, instead of relying only on Excel spreadsheets and/or pen-and-paper calculations.	1	5	4.14	1.292
The platform constituted a good intermediate solution between developing code from scratch and learning how to use a commercial solver.	2	5	4.5	0.855
I think that I would need the support of a technical person to be able to use this platform.	2	5	4.14	0.949
I found the platform user-friendly enough for a beginner to get started.	2	5	3.21	1.051
I found that the various functions in this platform were well-integrated.	4	5	4.5	0.519

responses). This showcases the effectiveness of the framework in helping students conduct multidisciplinary analyses within their projects.

It is worth noting that a preliminary assessment had been conducted prior to the initial deployment of the MADE framework to evaluate its relevance and usability. For this, a beta test was conducted with a group of 14 students who had previously completed the AD course with the help of common, single-disciplinary computational tools such as the ones introduced in Section II-A. Upon their completion of the beta test, these students were asked questions meant to compare the previous (traditional) and new (MADE-based) educational paradigms. Table II lists the most relevant questions and responses of the survey, which was designed based on the system usability scale by Brooke et al. [26]. The questions listed here correspond to those more pertinent to the framework, namely, multidisciplinary understanding and computational programming, for which most aspects received high-mean scores.

Aside from the positive feedback, the test also helped identify two aspects that required improvements at the time, namely, user-friendliness and the need for technical support. These points were subsequently addressed before the MADE platform was introduced and deployed within the APS and AD courses. Particularly, more extensive documentation and detailed step-by-step tutorials were created to ease the framework's learning curve.

C. Learning Outcomes

Regarding learning outcomes, most students were able to comfortably adopt the MADE framework. This was assessed based on the Pluto notebooks submitted by students, which detailed their computations along with explanations of the relevant multidisciplinary theory and applications covered in their respective group reports. In particular, it was observed that students had gained a better understanding of computational thinking compared to when they performed their assignments using more traditional one-off programming tools. Specifically, fewer errors were reported in assignments conducted using Pluto notebooks compared to those of previous years, which relied on Excel sheets, hand calculations, and coding tasks without any unified framework. This observation shows that MADE can facilitate the execution of complex,

multidisciplinary workflows by enforcing consistency between the results generated via the disciplinary tools.

Concerning the limitations of the MADE framework, it was observed that some students contributed more than others to the programming efforts required in their group projects. Indeed, some students may face greater challenges in acquiring programming practices (especially if they have no prior experience on the matter), which may lead to their lack of engagement. On a related note, it was also observed that some students generated incorrect results because they copied parts of reference code or examples without understanding the relevant material. Regarding these aspects, it is acknowledged that MADE cannot fully help students overcome the inertia toward the adoption of programming practices, although it can still help ease the uptake.

In terms of assessment, the solutions written in Pluto notebooks can provide a comprehensive indication of students' understanding of course content. In addition, the computational notebook can also provide more refined information to enable more flexible and fairer marking of assignments instead of just relying on the final results. This is thanks to the systematic presentation of all the work leading to the final answers, which can serve as proof verification. For instance, certain cases were observed in which computational functions correctly written by the students resulted in incorrect solutions simply due to typographical errors in the inputs. In the grading process, a student committing this error would be judged less harshly compared to a student who would repeatedly copy complex arithmetic expressions for doing computations (as in a calculator) and then make an error in the process. Correspondingly, specific feedback can be provided to students upon highlighting their common pitfalls. This kind of information would be unclear or comparably more tedious to check in traditional pen-and-paper or report-like submissions.

V. Conclusion

A new educational initiative, namely, the MADE framework, was developed, deployed, and assessed within an undergraduate curriculum of the HKUST. This framework aims at educating undergraduate students about multidisciplinary design, analysis, and optimization practices with computational methods in aerospace engineering.

MADE is implemented as software (called AeroFuse), which can be accessed online and used under an open-source license. This software introduces computational paradigms, such as numerical methods and sensitivity analysis, that are needed for handling common aerospace engineering disciplines (aerodynamics, structures, propulsion, flight dynamics, etc.), whether in isolation or altogether. Furthermore, the framework is designed to help instructors and students in their respective teaching and learning practices, this being done through "computational notebooks" used in lectures, tutorials, and assignments.

The MADE framework was initially deployed in a single-disciplinary, theory-driven course on APS, being used by the students to analyze the performance and stability of a commercial aircraft. MADE was subsequently deployed within a multidisciplinary, application-driven course on AD, being employed by the students to perform multidisciplinary analyses in completing an AD project. The outcomes of incorporating MADE in these courses were assessed all the way through dedicated tests and surveys. These assessments indicated positive learning outcomes, with students exhibiting improved multidisciplinary thinking. Although some students initially faced challenges with the user interface for programming, their surveys indicated increased confidence by the end of the courses with the continuous improvements to the framework. Overall, it was observed that students benefited from this multidisciplinary computational environment designed for education.

It is envisioned to deploy the MADE framework within other courses at HKUST. Considering the demonstrated benefits, MADE would be further integrated into courses offered to freshmen at the beginning of the undergraduate curriculum, such as introduction to aerospace engineering. In the long term, the framework could be readily extended and tailored to other fields beyond aerospace engineering, such as aero-structural design for bridges in civil engineering. As such, it is planned to encourage and facilitate its adoption to successfully prepare students for future careers in academia and industry.

Acknowledgment

The authors are grateful for the support from the Center for Education Innovation members of the Hong Kong University of Science and Technology (HKUST), specifically Mansurbek Kushnazarov, Eric Yeung, Ritchie Wong, and Beatrice Chu, for their help in designing and conducting the surveys pertaining to the feedback from students. The authors are also grateful to the staff from the Department of Mechanical and Aerospace Engineering at HKUST, specifically Matthew K. K. Lai, Saviour T. Y. Leung, and Elton Wu for their support in integrating the framework within the classroom environments. Finally, the authors are thankful to the students Yeung Siu Ting, Ma Lok Yin, Sun Hoi Chi, and Li Shing Yick from the AD course of Spring 2022/23 at HKUST for featuring their work in Fig. 3.

References

[1] J. R. Martins and A. Ning, *Engineering Design Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2021.

[2] E. M. Greitzer et al., "N+3 aircraft concept designs and trade studies, final report, volumes 1–2," Nat. Aeronaut. Space Admin, (NASA), Glenn Res. Center, Cleveland, OH, USA, Rep. CR–2010-216794, 2010.

[3] M. Drela, "TASOPT 2.00," Massachusetts Inst. Technol., Cambridge, MA, USA, Rep., 2010. [Online]. Available: http://web.mit.edu/drela/Public/N+3/TASOPT_doc.pdf

[4] D. R. Brodeur, P. W. Young, and K. B. Blair, "Problem-based learning in aerospace engineering education," in *Proc. Amer. Soc. Eng. Educ. Annu. Conf. Expo.*, 2002, pp. 16–19.

[5] T. Young, "Aircraft design education at universities: Benefits and difficulties," *Aircraft Design*, vol. 3, pp. 207–215, Dec. 2000.

[6] D. Scholz and J. Thorbeck, "Computer based training in aircraft design education," in *Proc. Int. Congr. Aeronaut. Sci.*, 2000, pp. 1–2.

[7] M. Okutsu, D. DeLaurentis, S. Brophy, and J. Lambert, "Teaching an aerospace engineering design course via virtual worlds: A comparative assessment of learning outcomes," *Comput. Educ.*, vol. 60, no. 1, pp. 288–298, 2013.

[8] R. M. Cummings, S. Reed, A. J. Rolling, and S. A. Brandt, "Aircraft design as a tool in achieving educational objectives for engineering concepts in a core curriculum," in *Proc. 54th AIAA Aerosp. Sci. Meeting*, 2016, pp. 6–7. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2016–0537

[9] D. Raymer, *Aircraft Design: A Conceptual Approach*. Reston, VA, USA: Amer. Inst. Aeronaut. Astronaut. Inc., 2012.

[10] J. Roskam, "Rapid sizing method for airplanes," *J. Aircraft*, vol. 23, no. 7, pp. 554–560, 1986. [Online]. Available: https://doi.org/10.2514/3.45343

[11] L. M. Nicolai and G. E. Carichner, *Fundamentals of Aircraft and Airship Design, Volume 1–Aircraft Design*. Reston, VA, USA: Amer. Inst. Aeronaut. Astronaut., 2010.

[12] L. Barba and O. Mesnard, "Aero python: Classical aerodynamics of potential flow using python," *J. Open Source Educ.*, vol. 2, no. 15, p. 45, 2019. [Online]. Available: https://doi.org/10.21105/jose.00045

[13] L. Barba and G. Forsyth, "CFD python: The 12 steps to Navier-Stokes equations," *J. Open Source Educ.*, vol. 2, no. 16, p. 21, 2019. [Online]. Available: https://doi.org/10.21105/jose.00021

[14] R. A. McDonald and J. R. Gloudemans, "Open vehicle sketch pad: An open source parametric geometry and analysis tool for conceptual aircraft design," in *Proc. AIAA SCITECH Forum*, 2022, pp. 1–32. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2022–0004

[15] T. W. Lukaczyk et al., "SUAVE: An open-source environment for multi-fidelity conceptual vehicle design," in *Proc. 16th AIAA/ISSMO Multidiscipl. Anal. Optim. Conf.*, 2015, p. 3087.

[16] J. P. Jasa, J. T. Hwang, and J. R. R. A. Martins, "Open-source coupled aerostructural optimization using python," *Struct. Multidiscipl. Optim.*, vol. 57, no. 4, pp. 1815–1827, Apr. 2018.

[17] 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," *Struct. Multidiscipl. Optim.*, vol. 59, pp. 1075–1104, Apr. 2019.

[18] A. B. Lambe and J. R. Martins, "Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes," *Struct. Multidiscipl. Optim.*, vol. 46, no. 2, pp. 273–284, 2012.

[19] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Rev.*, vol. 59, no. 1, pp. 65–98, 2017. [Online]. Available: https://doi.org/10.1137/141000671

[20] R. Sells, "Julia programming language benchmark using a flight simulation," in *Proc. IEEE Aerosp. Conf.*, 2020, pp. 1–8.

[21] J. Katz and A. Plotkin, *Low-Speed Aerodynamics, Second Edition*. Cambridge, U.K.: Cambridge Univ. Press, 2001.

[22] D. L. Logan, *A First Course in the Finite Element Method*. Boston, MA, USA: Cengage Learn., 2016.

[23] J. M. Seddon and S. Newman, *Basic Helicopter Aerodynamics*. New York, NY, USA: Wiley, 2011.

[24] M. Drela, *Flight Vehicle Aerodynamics*. Cambridge, MA, USA: MIT Press, 2014.

[25] F. van der Plas et al. "Fonsp/Pluto.jl: V0.19.17." Dec. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7411680

[26] J. Brooke, "SUS: A 'Quick and dirty' usability scale," *Usability Eval. Ind.*, vol. 189, no. 194, pp. 4–7, 1996.