Practical Appropriate Fidelity Optimization for Large-scale Multidisciplinary Aircraft Design

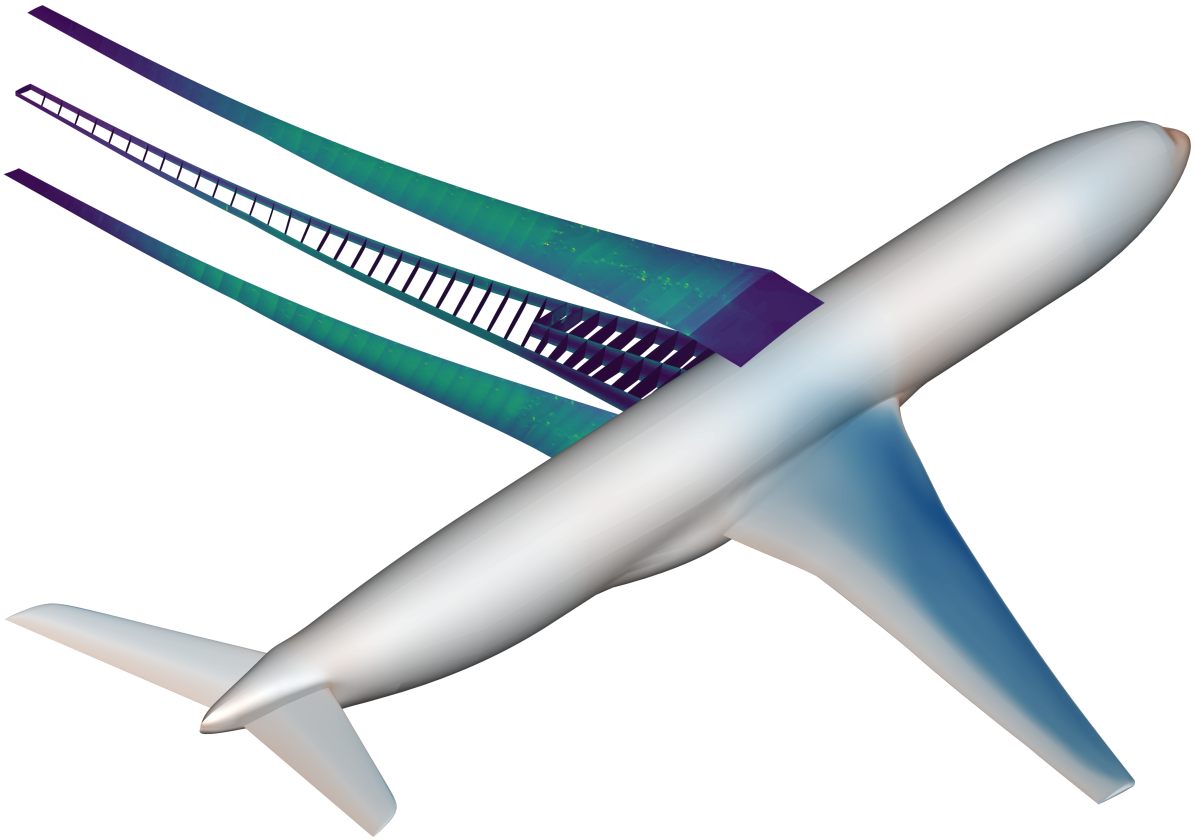
by

Neil Y. Wu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in the University of Michigan
2023

Doctoral Committee:

Professor Joaquim R. R. A. Martins, Co-Chair
Adjunct Assistant Research Scientist Charles A. Mader, Co-Chair
Professor Bogdan I. Epureanu
Professor Krzysztof J. Fidkowski



Neil Y. Wu

neilwu@umich.edu

ORCID iD: 0000-0001-8856-9661

To my parents, who have sacrificed so much to help me realize my dreams.

Acknowledgements

Doctoral research is a long and arduous journey, made possible and rewarding only through the support of many people along the way.

First and foremost, I am immensely thankful to my parents, Guodong and Xiaoli. From an early age, they instilled in me a sense of curiosity and wonder at the world. And it is that inquisitive nature that has led me to completing my PhD. Their love and unwavering support means everything to me. I also thank my aunt for looking after me, and for her loving support.

I first came to the University of Michigan as an apprehensive master's student. While I was interested in academic research, I would not have dreamed of completing a PhD here. I thank my adviser, Prof. Martins, for making that possible. I first approached him for an AE590 project, and he was happy to take me on even though I knew virtually nothing about optimization. Subsequently, he offered me a PhD position, and I am very grateful for this opportunity to perform academic research at such an esteemed department and university. Prof. Martins has been extremely helpful as an adviser and mentor, not only academically but more broadly speaking, and I aim to take many of his advices to heart. I also want to acknowledge his generosity in ensuring a positive student experience, and his work advocating for Diversity, Equity, and Inclusion (DEI) at the department.

I also thank the co-chair of my committee, Dr. Sandy Mader. His guidance has been indispensible during my PhD, and I'm grateful for all the technical discussions and insightful inquiries. I thank the other members of my committee, Prof. Krzysztof Fidkowski and Prof. Bogdan Epureanu, for their time and helpful suggestions.

My PhD would not have been possible without the support of those in the MDO Lab, both past and present. Together with Dr. Mader, Dr. Gaetan Kenway and Prof. Graeme Kennedy helped build the foundation of the code base that enabled this research. In addition, Dr. Kenway set up the initial aerostructural optimization of the XRF1 aircraft, which is greatly acknowledged. I thank Anil Yildirim, Marco Mangano, Eirikur Jonsson, Sabet Seraj, Nicholas Bons, Alasdair Gray, Alex Coppeans, and Justin Gray for technical discussions and insights that greatly benefitted this work. I also thank Shamsheer Chauhan, Saja Kaiyoom, and Xiaosong Du for being great office mates and for putting up with me over the last few years. To others in the lab: John, Ben, Mads, Sicheng, Yingqian, Gustavo, Ping, Tim, Ney,

Mohamed, Alex,* Bernardo, Galen, Andrew, Hannah, Shugo, Josh, Eytan, thank you for welcoming me into the lab and the wonderful memories.

I have had a wonderful time at the aerospace engineering department. I thank Ruthie for being a ray of sunshine in the department, Chris Wentland for being my dissertation writing buddy, members of GSAC for building such a great community, the weekly frisbee group for keeping me at least somewhat active, and the many friends I made in the department that are too numerous to list. I thank Maria, Shree, and my friends both in Michigan and back home in Toronto for all their support. To Annabelle, thank you for believing in me since the start and for supporting me.

I have also had the pleasure of working with a few external researchers. I thank my points of contact at Airbus: Anne Gazaix, Joel Brezillon, and Tom Gibson for their guidance and technical advice since the beginning of my PhD. This work was funded primarily by Airbus through the Airbus / Michigan Center for Aero-Servo-Elasticity of Very Flexible Aircraft, and I thank everyone involved for their support, especially Prof. Cesnik who led the Center. I thank the developers of SNOPT, Prof. Philip Gill and Dr. Elizabeth Wong, for helpful discussions and updates to the optimizer. Their efforts enabled large portions of this work.

This research was partly supported through computational resources and services provided by Advanced Research Computing at the University of Michigan, Ann Arbor. This research also made extensive use of the Texas Advanced Computing Center (TACC) Stampede2 supercomputer via Extreme Science and Engineering Discovery Environment (XSEDE).

* He lent me an HDD at a crucial time which allowed me to complete some of my optimizations

Table of Contents

List of Figures

List of Tables

List of Appendices

List of Acronyms

AD	automatic differentiation
ADODG	Aerodynamic Design Optimization Discussion Group
AGILE	Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts
AIAA	American Institute of Aeronautics and Astronautics
ALPSO	Augmented Lagrangian Particle Swarm Optimizer
ANK	approximate Newton–Krylov
API	application programming interface
AR	aspect ratio
ASO	aerodynamic shape optimization
AWS	Amazon Web Services
BFGS	Broyden–Fletcher–Goldfarb–Shanno
BLI	boundary-layer ingestion
CAD	computed-aided design
CFD	computational fluid dynamics
CGNS	CFD General Notation System
CONMIN	CONstrained function MINimization
CPU	central processing unit
CRM	Common Research Model

CSM	computational structural mechanics
CST	Class-Shape function Transformation
DFP	Davidon–Fletcher–Powell
DIRECT	DIviding RECTangles
DLR	German Aerospace Center
DOF	degree of freedom
DTU	Technical University of Denmark
DVC	Data Version Control
ESP	Engineer Sketch Pad
FB	fuel burn
FEM	finite-element method
FFD	free-form deformation
FOSM	first-order second-moment
FSI	fluid-structure interaction
GA	genetic algorithm
GBMS	gradient-based multistart
GCC	GNU Compiler Collection
GMRES	generalized minimal residual
HDMR	high-dimensional model representation
HF	high fidelity
HPC	high-performance computing
IDF	individual discipline feasible
IEEE	Institute of Electrical and Electronics Engineers
IFF	if and only if

IP	interior point
IPOPT	Interior Point OPTimizer
KKT	Karush–Kuhn–Tucker
KS	Kreisselmeier–Steinhauser
LE	leading edge
LGW	landing gross weight
LOC	lines of code
MACH	MDO of Aircraft Configurations with High-fidelity
MADELEINE	Multidisciplinary ADjoint-based Enablers for LargE-scale Industrial desigN in aEronautics
MAUD	modular analysis and unified derivatives
MBSE	model-based systems engineering
MDA	multidisciplinary analysis
MDF	multidisciplinary feasible
MDO	multidisciplinary design optimization
MFD	Method of Feasible Directions
MPI	Message Passing Interface
NASA	National Aeronautics and Space Administration
NK	Newton–Krylov
NLBGS	nonlinear block Gauss–Seidel
NLPQLP	NonLinear Programming with Non-Monotone and Distributed Line Search
NSGA2	Non Sorting Genetic Algorithm II
OAD	overall aircraft design
OML	outer mould line

OPENMP	Open Multi-Processing
OS	operating system
PDE	partial differential equation
PDF	probability distribution function
PETSC	Portable, Extensible Toolkit for Scientific Computation
PRNG	pseudo-random number generator
PSO	particle swarm optimization
PSQP	Preconditioned Sequential Quadratic Programming
QOI	quantity of interest
RANS	Reynolds-averaged Navier–Stokes
RCE	Remote Component Environment
RK	Runge–Kutta
SA	Spalart–Allmaras
SAND	simultaneous analysis and design
SLSQP	Sequential Least SQuares Programming
SNOPT	Sparse Nonlinear OPTimizer
SPD	symmetric positive-definite
SQP	sequential quadratic programming
SU2	Stanford University Unstructured
SUAVE	Stanford University Aerospace Vehicle Environment
SVD	singular value decomposition
TACS	Toolkit for Analysis of Composite Structures
TE	trailing edge
TOGW	take-off gross weight

TR	trust region
TRL	technology readiness level
TRMM	trust-region model management
TSFC	thrust-specific fuel consumption
UCRM	undeflected Common Research Model
UDE	unified derivatives equation
UTIAS	University of Toronto Institute for Aerospace Studies
V&V	verification and validation
VCS	version control system
VLM	vortex lattice method
WISDEM	Wind-Plant Integrated System Design and Engineering Model
XDSM	extended design structure matrix
XRF1	eXternal Research Forum 1

List of Symbols

Roman letters

$\mathbf{A}$	Mapping matrix
$\mathcal{A}$	Active set
C_D	Coefficient of drag
C_L	Coefficient of lift
c_T	Thrust-specific fuel consumption
$\mathcal{E}$	Set of Equality constraints
h	Altitude
$\mathbf{I}$	Identity matrix
$\mathcal{I}$	Set of inequality constraints; quantity of interest
$\mathbf{J}$	Jacobian
$\mathcal{L}$	Lagrangian function
ℓ	Fidelity index
M	Mach number
m	Mass
$\mathcal{N}$	Normal distribution
R	Range
$\boldsymbol{R}$	Residual
s	Design variable scaling parameter

t Thickness

$\boldsymbol{u}$ State variable

V Volume

W Weight

$\boldsymbol{x}$ Design variables

$\boldsymbol{X}_A$ Aerodynamic surface mesh coordinates

$\hat{\boldsymbol{x}}$ Mapped design variables

$\boldsymbol{X}_S$ Structural mesh coordinates

Greek letters

Δ Difference

$\boldsymbol{\Sigma}$ Covariance matrix

$\boldsymbol{\psi}$ Adjoint variables

$\boldsymbol{\alpha}$ Angle of attack

δ Discipline error

ϵ System-level error

$\epsilon_{\mathbf{mp}}$ Machine precision

η Computational speedup

θ Angle between vectors

$\boldsymbol{\lambda}$ Lagrange multiplier

μ Mean

ξ Parametric coordinate

ρ Correlation coefficient

σ Singular value; standard deviation

τ Tolerance

Subscripts and Superscripts

adj Adjoint

cr Cruise

fea Feasibility

geo Geometric

man Maneuver

opt Optimality

pr Primal

red Reduction

Accents

* Converged quantities

~ Unconverged quantities

^ Mapped quantity; normalized quantity

– Target quantity

Abstract

Numerical optimization has been successfully applied to multidisciplinary design optimizations such as aerostructural wing design. These optimizations consist of over a thousand design variables and constraints, and include expensive simulations such as computational fluid dynamics within the optimization loop. Nevertheless, by using gradient-based optimizers together with efficient gradient computation techniques, researchers have been able to tackle these challenging large-scale problems.

However, longstanding challenges remain. These optimizations tend to require direct user input, manually tuning various optimization parameters to obtain convergence. The optimizations are slow and computationally expensive, often using thousands of processors for several days at a time. As aircraft designers move toward using more expensive, higher-fidelity tools in design, the computational cost will only increase in the future.

To address these challenges, this dissertation contains three main contributions. First, a novel geometric parameterization is presented. Based on sensitivity analysis, the parameterization is also able to automatically scale the newly-generated design variables, suitably for gradient-based optimization. This approach is demonstrated on two aerodynamic shape optimizations, and is shown to perform comparably to the manual approach requiring trial and error.

Second, an adaptive error control scheme is developed to reduce the computational cost of optimizations. The function error due to convergence of the solver is estimated using an adjoint-derived approach. The error is then adapted during optimization, allowing for loose solver convergence at the beginning of the optimization, thereby reducing computational cost. The approach is demonstrated on two aerodynamic shape optimizations, showing between 30%–50% cost savings.

Third, an appropriate fidelity optimization framework is developed, suited for gradient-based multidisciplinary design optimizations. This framework uses a sequential approach, and begins by quantifying the errors present in each fidelity at the discipline level. The errors are then propagated to the system-level objective and constraints. An optimization is started using the lowest fidelity, and these errors are used to terminate the low-fidelity optimizations at an appropriate time. The errors are also used to select the next appropriate fidelity, by performing a tradeoff between error reduction and computational cost increase. A new optimization is then started using the selected fidelity, and the process is repeated until the high-fidelity optimum is reached.

The approach is then demonstrated on two aerostructural optimizations, including the XRF1 aircraft with over 900 design variables and 900 constraints. The aircraft is analyzed at four flight conditions, and there are a total of over 300 000 possible fidelity combinations. Through the use of the appropriate fidelity framework, cost savings of between 44% and 64% were realized.

Chapter 1

Introduction

Optimization problems are ubiquitous in engineering. From foundational concepts such as least squares and Lagrangian mechanics to practical problems of designing complex aerospace systems, optimization problems arise every step of the way. However, until recently such optimization problems remained out of reach, due to a lack of both computational power and advanced algorithms capable of tackling such problems.

The field of engineering science has always occupied a unique position in academia, bridging the gap between theory and application. While theory is necessary to advance the field, such developments must be kept relevant to industrial applications. On top of this, computational research requires a third component to demonstrate such applications, and the inclusion of all three is necessary to produce impact.

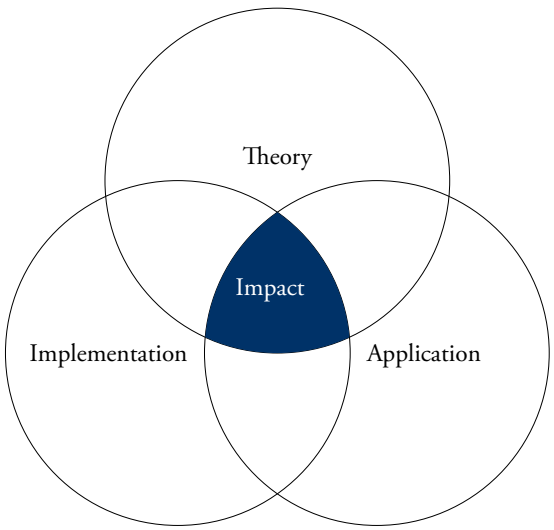


Figure 1.1: There are three components to producing impact in computational research [1].

Due to recent advances, optimization methods have been successfully applied to a range of engineering problems, particularly in the field of aerospace engineering. Nevertheless, many challenges remain. In the following sections, we give an overview of engineering design optimization, discussing the state of the art and their shortcomings.

1.1 High-fidelity aircraft design

Historically, engineering design is a manual, iterative process. Starting from an initial design, the designer uses experience to decide on changes to improve its performance. This process is iterated until the design is deemed sufficient. Figure 1.2 shows such a process.

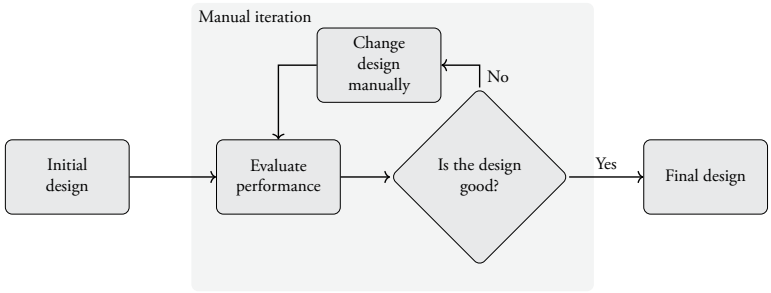


Figure 1.2: Conventional design processes, adapted from Martins and Ning [2].

In the last few decades, aircraft design has become increasingly optimization-driven. This is largely enabled by successful research outcomes in numerical optimization, efficient gradient computation, and multidisciplinary design optimization (MDO). While push-button solutions are unlikely in the near term, certain problems such as airfoil design has become relatively routine. He et al. [3], for example, were able to perform airfoil optimization robustly starting from a circle. Figure 1.3 shows the automated process using numerical optimization, where the design problem is formulated as an optimization problem, and an optimizer replaces the designer in making design changes. When using gradient-based optimizers, it is also possible to show numerically that the solution is indeed optimal for the problem posed.

[3]: He et al. (2019), *Robust aerodynamic shape optimization—from a circle to an airfoil*

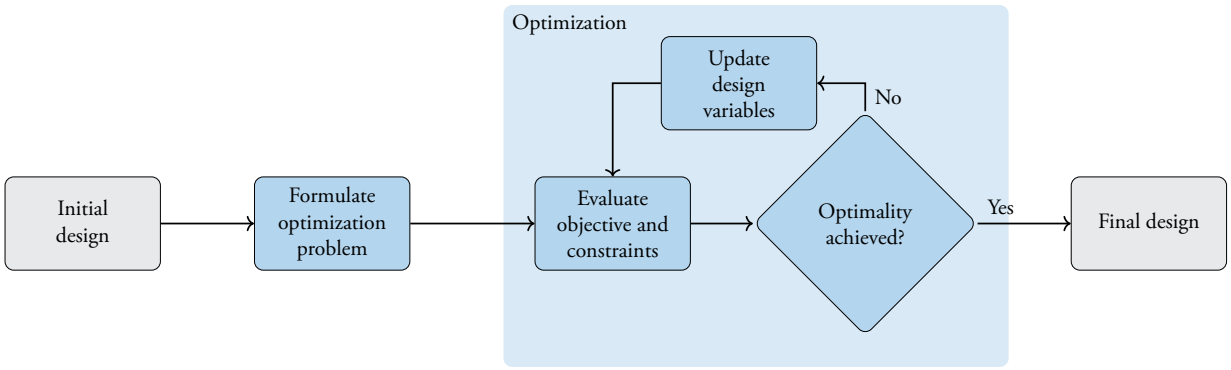


Figure 1.3: Optimization-driven design processes, adapted from Martins and Ning [2].

1.1.1 Numerical optimization

Issac Newton examined the minimal resistance problem in his *Principia* [4], where he sought the solid of revolution which minimized fluid drag. This can be considered the first true optimization problem to be solved using calculus. Not only is it still a relevant problem 300 years later, Newton invented an entirely new branch of mathematics in the process: the calculus of variations. A decade later, Johann Bernoulli posed the famous *brachistochrone problem*, which Newton solved overnight, again using the calculus of variations to arrive at the solution.

[4]: Newton (1687), *Philosophiæ Naturalis Principia Mathematica*

While mathematically interesting, most real-world optimization problems cannot be solved in infinite-dimensional space. Instead, problems are often discretized, resulting in a finite number of design variables. In Soviet Russia, Leonid Kantorovich worked on linear optimization problems in 1939, in order to optimize the industrial production of plywood. However, significant advances in general nonlinear problems did not appear until the advent of the computer, which provided the necessary computational power. Quasi-Newton methods came to prominence following the pioneering work of William Davidon, leading to the Davidon–Fletcher–Powell (DFP) method. Subsequently, the Broyden–Fletcher–Goldfarb–Shanno (BFGS) update formula was developed, which became the backbone of modern quasi-Newton methods.

1.1.2 Efficient gradient computation

Gradient-based optimization algorithms are attractive for their numerous advantages. They scale well with the number of design variables, can handle linear and nonlinear constraints directly, and demonstrate the optimality of the solution. However, the computation of gradients is not an easy task.

Finite difference methods have been known since the time of Taylor and Newton. They are simple to implement, but they suffer from subtractive cancellation [2, Sec. 6.4.2], and their costs scale linearly with the number of design variables. More recently, the complex-step method [5, 6] has been developed. It avoids subtractive cancellation errors, and offers an $\mathcal{O}(h^2)$-acurrate derivative estimate at the same linear cost.[1]

[2]: Martins et al. (2021), *Engineering Design Optimization*

[5]: Squire et al. (1998), *Using complex variables to estimate derivatives of real functions*

[6]: Martins et al. (2003), *The Complex-Step Derivative Approximation*

1: In practice, the cost is often higher than finite differences due to the overhead of performing complex arithmetic.

Given that practical engineering optimization problems can exceed a thousand design variables, a more efficient method is needed to accurately compute the derivatives. Originating in the optimal control community, the *adjoint method* was later applied to structural problems [7]. Pironneau [8] first introduced the method to fluid dynamics, which was then extended by Jameson [9] to perform ASOs using Euler equations. Further developments in numerical methods and the adoption of automatic differentiation (AD) techniques led to optimizations involving more complex PDEs, such as Reynolds-averaged Navier–Stokes (RANS) equations [10, 11], laminar-turbulence transition prediction [12], and time-spectral solver for periodic wakes [13]. Buffet analysis was added by Kenway and Martins [14], and cavitation constraint by Garg et al. [15]. See [16] for a comprehensive review of adjoint methods in computational fluid dynamics (CFD).

[7]: Arora et al. (1976), *Efficient Optimal Design of Structures by Generalized Steepest Descent Programming*

[8]: Pironneau (1973), *On Optimum Profiles in Stokes Flow*

[9]: Jameson (1988), *Aerodynamic Design via Control Theory*

[10]: Lyu et al. (2013), *Automatic Differentiation Adjoint of the Reynolds-Averaged Navier–Stokes Equations with a Turbulence Model*

[11]: Lyu et al. (2014), *Aerodynamic Design Optimization Studies of a Blended-Wing-Body Aircraft*

[12]: Shi et al. (2020), *Natural Laminar-Flow Airfoil Optimization Design Using a Discrete Adjoint Approach*

[13]: He et al. (2020), *A Time-Spectral Adjoint Approach for Aerodynamic Shape Optimization Under Periodic Wakes*

[14]: Kenway et al. (2017), *Buffet-Onset Constraint Formulation for Aerodynamic Shape Optimization*

[15]: Garg et al. (2015), *High-fidelity Hydrodynamic Shape Optimization of a 3-D Hydrofoil*

[16]: Kenway et al. (2019), *Effective Adjoint Approaches for Computational Fluid Dynamics*

1.1.3 Multidisciplinary design optimization

Aerospace systems are inherently multidisciplinary, which manifests itself in two ways. Take wing design for example, which

requires consideration of both aerodynamics and structures. First, the two disciplines are coupled, meaning that they cannot be analyzed in isolation. The aerodynamic forces on the wing will result in pressure loads for the wing structure, causing it to deflect in response. This displacement will naturally cause the wing to alter its shape, resulting in a different set of aerodynamic loads. Therefore, the true aerodynamic and structural performance of the wing must be computed in a coupled fashion, considering both disciplines simultaneously. This multidisciplinary analysis (MDA) procedure is discussed in more detail in section 2.3. Second, the performance of the wing must take both disciplines into account. Rather than performing drag minimization, it is more relevant to consider multidisciplinary metrics such as the fuel burn, which includes the structural mass. That way, the tradeoff between drag minimization and weight minimization can be exploited in a multidisciplinary fashion.

Haftka [17] first considered both aerodynamics and structures in gradient-based optimization of a fighter aircraft wing. However, the analysis was not coupled—the structural displacements had no impact on the aerodynamic coefficients, and the problem was formulated as weight minimization with an induced drag constraint.

[17]: Haftka (1977), *Optimization of Flexible Wing Structures Subject to Strength and Induced Drag Constraints*

Fully considering the coupled aerostructural system required further mathematical and algorithmic developments. For efficient gradient computation of a coupled system, Sobieszczanski–Sobieski [18] developed the coupled direct method. This was soon followed-up with the coupled adjoint method by Martins, Alonso, and Reuther [19], which provided the basis for the gradient computations used in this work. This approach was first demonstrated in an aerostructural optimization of a supersonic business jet [20] involving 97 design variables.

[18]: Sobieszczanski–Sobieski (1990), *Sensitivity of Complex, Internally Coupled Systems*

[19]: Martins et al. (2005), *A Coupled-Adjoint Sensitivity Analysis Method for High-Fidelity Aero-Structural Design*

[20]: Martins et al. (2004), *High-Fidelity Aerostructural Design Optimization of a Supersonic Business Jet*

Since then, the optimization problem has been expanded greatly. Kenway, Kennedy, and Martins [21] made several algorithmic and implementation improvements to the coupled adjoint ap-

[21]: Kenway et al. (2014), *Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Adjoint Derivative Computations*

proach to make it scalable, including using AD to compute Jacobians more accurately, improved mesh deformation, and novel numerical algorithms for the solution of the aerostructural system and adjoint. The framework, called MDO of Aircraft Configurations with High-fidelity (MACH), was demonstrated on a transport aircraft optimization involving 472 design variables, analyzed over five cruise, two maneuver, and one stability flight conditions. Similar coupled adjoint capabilities were developed by others [22]. The coupled adjoint approach, sometimes called the *flexible* adjoint by aerodynamicists, has shown significant benefits compared to the rigid adjoint [23, 24].

Within MACH, CFD analysis was extended to solving the RANS equations shortly after [25, 26, 27]. Additional technologies were analyzed and optimized, such as tow-steered composites [28, 29], morphing trailing edges [30, 31], boundary-layer ingestion (BLI) [32].

Further developments also occurred on the mathematical side. Martins and Lambe [33] performed a comprehensive survey of MDO architectures, and Tedford and Martins [34] benchmarked several architectures on MDO problems. The authors concluded that while simultaneous analysis and design (SAND), or full-space approaches can be efficient, they are less robust compared to reduced-space approaches such as multidisciplinary feasible (MDF). The efficiency of the MDF approach may also improve as the number of state or coupling variables is increased.

Later, Martins and Hwang [35] were able to unify a number of different methods for computing derivatives under a single mathematical framework called modular analysis and unified derivatives (MAUD). The complex-step method, AD, adjoint, and coupled adjoint methods were all unified into a single equation called the unified derivatives equation (UDE). This was later incorporated into the OpenMDAO framework [36] developed at NASA Glenn Research Center, and has been widely adopted in academia and industry for MDO. Mphys is a modular multi-

[22]: Zhang et al. (2016), *High-fidelity aerostructural optimization with integrated geometry parameterization and mesh movement*

[23]: Olivanti et al. (2021), *On the Benefits of Engaging Coupled-Adjoint to Perform High-Fidelity Multipoint Aircraft Shape Optimization*

[24]: Carini et al. (2021), *Towards industrial aerostructural aircraft optimization via coupled-adjoint derivatives*

[25]: Kenway et al. (2014), *Aerostructural Optimization of the Common Research Model Configuration*

[26]: Kenway et al. (2015), *High-fidelity aerostructural optimization considering buffet onset*

[27]: Brooks et al. (2018), *Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings*

[28]: Brooks et al. (2019), *High-fidelity Aerostructural Optimization of Tow-steered Composite Wings*

[29]: Brooks et al. (2020), *Aerostructural Trade-offs for Tow-steered Composite Wings*

[30]: Burdette et al. (2018), *Design of a Transonic Wing with an Adaptive Morphing Trailing Edge via Aerostructural Optimization*

[31]: Burdette et al. (2019), *Impact of Morphing Trailing Edge on Mission Performance for the Common Research Model*

[32]: Yildirim et al. (2022), *Boundary Layer Ingestion Benefit for the STARC-ABL Concept*

[33]: Martins et al. (2013), *Multidisciplinary Design Optimization: A Survey of Architectures*

[34]: Tedford et al. (2010), *Benchmarking Multidisciplinary Design Optimization Algorithms*

[35]: Martins et al. (2013), *Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models*

[36]: Gray et al. (2019), *OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization*

physics simulation package built on top of OpenMDAO, providing unified interfaces for various multiphysics solvers such that their couplings are handled automatically within OpenMDAO. It has been used in a number of applications [37, 38, 39].

[37]: Anibal et al. (2022), *Aerodynamic shape optimization of an electric aircraft motor surface heat exchanger with conjugate heat transfer constraint*
[38]: Yildirim et al. (2021), *Coupled Aeropropulsive Design Optimization of a Podded Electric Propulsor*
[39]: Jacobson et al. (2022), *Flutter-Constrained Optimization with the Linearized Frequency-Domain Approach*

There have also been major developments in Europe. Compared to research in North America, the focus has generally been on collaborative and distributed MDO architectures, mirroring the existing academic and industry approaches. Gemseo [40], developed at IRT Saint Exupéry, is an MDO framework similar to OpenMDAO, but with an emphasis on bi-level and distributed MDO architectures. The Multidisciplinary ADjoint-based Enablers for LargE-scale Industrial desigN in aEronautics (MADELEINE) project [41], funded by the European Union, is a collaboration between 15 partners in order to develop high-fidelity adjoint-based MDO capabilities and increase the technology readiness level (TRL).

[40]: Gallard et al. (2018), *GEMS: A Python Library for Automation of Multidisciplinary Design Optimization Process Generation*

[41]: Meheut (2021), *Multidisciplinary Adjoint-based Optimizations in the MADELEINE Project: Overview and Main Results*

1.2 Low and mixed-fidelity MDO

A large number of lower-fidelity MDO applications have also been developed, typically targeting the conceptual design stage. While the fidelity of the models may be lower, they offer reduced computational costs that enable the consideration of additional disciplines and load cases.

It is worth pointing out here that the definition of high and low-fidelity models have changed over the years, and there are no agreed-upon definitions. A decade or two ago, Euler simulations were considered as high-fidelity [20, 42], but now RANS CFD is typically considered [43].

Many low-fidelity MDO applications leverage OpenMDAO to couple a large number of disciplines, including FAST-OAD [44], WISDEM for wind turbine design, and OpenConcept [45, 46],

[20]: Martins et al. (2004), *High-Fidelity Aerostructural Design Optimization of a Supersonic Business Jet*
[42]: Kenway et al. (2014), *Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration*
[43]: Bravo-Mosquera et al. (2022), *Unconventional aircraft for civil aviation: A review of concepts and design methodologies*
[44]: David et al. (2021), *From FAST to FAST-OAD: An open source framework for rapid Overall Aircraft Design*
[45]: Brelje et al. (2018), *Development of a Conceptual Design Model for Aircraft Electric Propulsion with Efficient Gradients*
[46]: Adler et al. (2022), *Efficient Aerostructural Wing Optimization Considering Mission Analysis*

which can optionally use OpenAerostruct [47] to perform low-fidelity aerostructural optimizations using vortex lattice method (VLM) and a structural beam model. Other examples include Faber [48] developed at the University of Toronto, OpenAD from German Aerospace Center (DLR) [49], and Stanford University Aerospace Vehicle Environment (SUAVE) [50] developed at the Stanford University.[2]

PROTEUS [51], a low-fidelity aeroelastic optimization framework was used to perform optimizations involving hundreds of gust load cases [52]. There are also a number of MDO tools focused on dynamic aeroelastic effects such as UM/NAST [53]. Due to the high cost of predicting flutter, mixed-fidelity optimization approaches have been developed. Steady-state aerostructural solutions and performance metrics are computed using high-fidelity tools, while the flutter constraint itself is computed using lower-fidelity models [54].

The AGILE project [55] is an European Union-funded project focused on distributed MDO in aircraft design, using tools such as Remote Component Environment (RCE) [56] developed at DLR to perform optimizations where different disciplines are analyzed on different high-performance computing (HPC) systems [57]. Lower-fidelity tools are typically considered, but with many more disciplines incorporated into the MDO problem. For example, Bussemaker et al. [58] connected MDO with requirements engineering and architecture design in the context of model-based systems engineering (MBSE). Within the optimization problem itself, disciplines such as onboard systems, high-lift configuration, and mission were considered. Mandorino et al. [59] performed optimizations including a more detailed cost model, which considered—among other factors—the landing fees at Frankfurt airport.

See the review by Bravo-Mosquera, Catalano, and Zingg [43] for a more comprehensive review of MDO frameworks in aircraft design.

[47]: Jasa et al. (2018), *Open-source coupled aerostructural optimization using Python*

[48]: Chau et al. (2022), *Aerodynamic Design Optimization of a Transonic Strut-Braced-Wing Regional Aircraft*

[49]: Wöhler et al. (2020), *Preliminary aircraft design within a multidisciplinary and multifidelity design environment*

[50]: MacDonald et al. (2017), *SUAVE: An Open-Source Environment Enabling Multi-Fidelity Vehicle Optimization*

2: It is a mixed-fidelity tool that is capable of also using higher-fidelity analyses when directed.

[51]: Werter et al. (2016), *A novel dynamic aeroelastic framework for aeroelastic tailoring and structural optimisation*

[52]: Wang et al. (2022), *An aeroelastic optimisation framework for manufacturable variable stiffness composite wings including critical gust loads*

[53]: Su et al. (2010), *Nonlinear Aeroelasticity of a Very Flexible Blended-Wing-Body Aircraft*

[54]: Jonsson et al. (2022), *High-Fidelity Gradient-Based Wing Structural Optimization Including a Geometrically Nonlinear Flutter Constraint*

[55]: Ciampa et al. (2020), *AGILE Paradigm: The next generation collaborative MDO for the development of aeronautical systems*

[56]: Boden et al. (2021), *RCE: An Integration Environment for Engineering and Science*

[57]: Boden et al. (2019), *Distributed Multidisciplinary Optimization and Collaborative Process Development Using RCE*

[58]: Bussemaker et al. (2022), *Collaborative Design of a Business Jet Family Using the AGILE 4.0 MBSE Environment*

[59]: Mandorino et al. (2022), *Regional jet retrofitting design from stakeholders need and system requirements to MDAO workflow formulation*

[43]: Bravo-Mosquera et al. (2022), *Unconventional aircraft for civil aviation: A review of concepts and design methodologies*

1.3 Improvements in robustness and efficiency

Naturally, the field of MDO has been expanding to tackle more complex problems. This manifests itself in a few ways. Larger optimization problems are considered, both in terms of the number of design variables, and in terms of the number of load cases and constraints being considered. For example, Brooks, Martins, and Kennedy [28] performed a large-scale aerostructural optimization of the undeflected Common Research Model (uCRM) benchmark considering tow-steered composites, which included over 1500 design variables and 1300 constraints. The fidelity of the analyses are increased, including some unsteady problems [60, 61]. The number of disciplines has also been expanded, for example by including thermal effects in aerothermoelastic optimizations [62, 63].

[28]: Brooks et al. (2019), *High-fidelity Aerostructural Optimization of Tow-steered Composite Wings*

[60]: He et al. (2019), *Aerodynamic Shape Optimization with Time Spectral Flutter Adjoint*
[61]: Apponsah et al. (2020), *Aerodynamic shape optimization for unsteady flows: Some benchmark problems*
[62]: Guo et al. (2021), *Aero-structural optimization of supersonic wing under thermal environment using adjoint-based optimization algorithm*
[63]: Halim et al. (2022), *Aerothermoelastic Analysis and Optimization of Stiffened Thin-Walled Structures*

However, this effort exposes two fundamental issues with such approaches. With a more complex optimization problem and additional nonlinearities, its convergence rate will be further impeded, leading to suboptimal results. The introduction of higher-fidelity analyses will further increase the computational cost of such optimizations, which may not be tractable at early stages of design. These two points will be expanded in the following sections.

1.3.1 Robustness and optimization convergence

Aerodynamic and aerostructural optimizations are difficult to converge. To illustrate this further, we compiled some published aerodynamic shape optimization (ASO) problems from the American Institute of Aeronautics and Astronautics (AIAA) Aerodynamic Design Optimization Discussion Group (ADODG) benchmarks, which are listed in table 1.1. In order to ensure the results

Table 1.1: Optimization convergence achieved for some published ADODG cases.

Case	n_x	n_{con}	Major Iterations	τ_{fea}	τ_{opt}	$-\log_{10}(\tau_{opt}/\tau^0_{opt})$	Ref
1	24	—	38	—	4.0×10^{-2}	0.4	[64]
2	17	3	160	2.0×10^{-10}	6.5×10^{-6}	2.1	[65]
3	11	1	40	1.0×10^{-6}	1.1×10^{-6}	2.4	[64]
3	10	1	37	1.0×10^{-12}	2.0×10^{-8}	4.0	[65]
4.1	150	2	35	—	1.1×10^{-4}	0.8	[64]
4.1	769	753	263	—	7.0×10^{-5}	2.0	[66]
5.1	216	753	113	—	2.3×10^{-5}	1.3	[14]

are comparable, we only include those that used SNOPT, a popular gradient-based optimizer. Properties of the optimization problem such as the number of design variables n_x, the number of constraints n_{con}, achieved feasibility and optimality tolerances (τ_{fea} and τ_{opt} respectively), are listed. These metrics are formally defined in section 2.5.7.1, but they quantify how far a solution is from the true mathematical optimum. The last column is the order of magnitude reduction in optimality compared to the initial design.

While convergence is well-understood for CFD analyses, the same is not true for optimizations. There are no established metrics for determining acceptable levels of convergence for optimizations. Nevertheless, most of these results are not well-converged, including some with less than one order reduction in the optimality tolerance, despite often taking hundreds of iterations. This can lead to suboptimal results being presented as optimal, and multiple local minima appearing when the problem is in fact unimodal. For example, this effect may have affected the results presented in [67].

For example, Reist et al. [68] compared two different gradient-based ASO frameworks in order to compare optimal designs. However, the relative reduction of optimality was only between 1–2 orders of magnitude, which may partially explain the differences in the final solutions. Masters et al. [69] compared a number of geometric parameterizations for airfoil optimizations under inviscid flow conditions. They concluded that two distinct optima existed, but the relative optimality reduction was

[67]: Masters et al. (2017), *Influence of Shape Parameterization on a Benchmark Aerodynamic Optimization Problem*

[68]: Reist et al. (2020), *Cross Validation of Aerodynamic Shape Optimization Methodologies for Aircraft Wing-Body Optimization*

[69]: Masters et al. (2017), *Geometric Comparison of Aerofoil Shape Parameterization Methods*

once again between 1–2 orders of magnitude, which may be insufficient to distinguish between multiple local minima.

While simply obtaining a superior feasible design may be sufficient in an engineering setting, a deeper level of convergence is important. Without arriving at the numerical optimum, errors are introduced in both the design variables and the objective. Wu, Mader, and Martins [70] presented some aerodynamic optimizations with slow-changing design variables, and premature termination of the optimization would lead to drastically-different designs. Brooks, Kenway, and Martins [27] and Brooks, Martins, and Kennedy [28] also presented aerostructural optimizations for the uCRM, where the objective function is still decreasing noticeably at the end of the optimization.

[70]: Wu et al. (2022), *Sensitivity-based Geometric Parameterization for Aerodynamic Shape Optimization*

[27]: Brooks et al. (2018), *Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings*
[28]: Brooks et al. (2019), *High-fidelity Aerostructural Optimization of Tow-steered Composite Wings*

There are a number of reasons that can lead to poor optimization convergence. The most obvious case is simply setting loose termination criteria such that the optimization terminates prematurely, in order to reduce computational cost. If the optimization cannot proceed further, then the function values and gradients are likely inaccurate. The function values can be improved by allowing the solver to converge more tightly, and discipline-specific solvers can be effective at converging large nonlinear systems. The derivatives, on the other hand, can be difficult to compute accurately. The use of finite differences, simplifications such as frozen-turbulence, and poor convergence of the adjoint linear system can all degrade the accuracy significantly.[3] Kenway, Kennedy, and Martins [21] also cited the condition number of the structural Jacobian as the reason for reduced accuracy compared to aerodynamic results.

3: It is also common to have bugs in the gradient computations.

[21]: Kenway et al. (2014), *Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Adjoint Derivative Computations*

By carefully combining the adjoint method with AD techniques, it is possible to compute accurate derivatives. For example, Kenway et al. [16] demonstrated 11 digits of accuracy in RANS CFD using ADflow. However, even with accurate gradients, optimizations can be difficult to converge. A large amount of time is often spent on setup and tuning optimization parameters. In particular,

[16]: Kenway et al. (2019), *Effective Adjoint Approaches for Computational Fluid Dynamics*

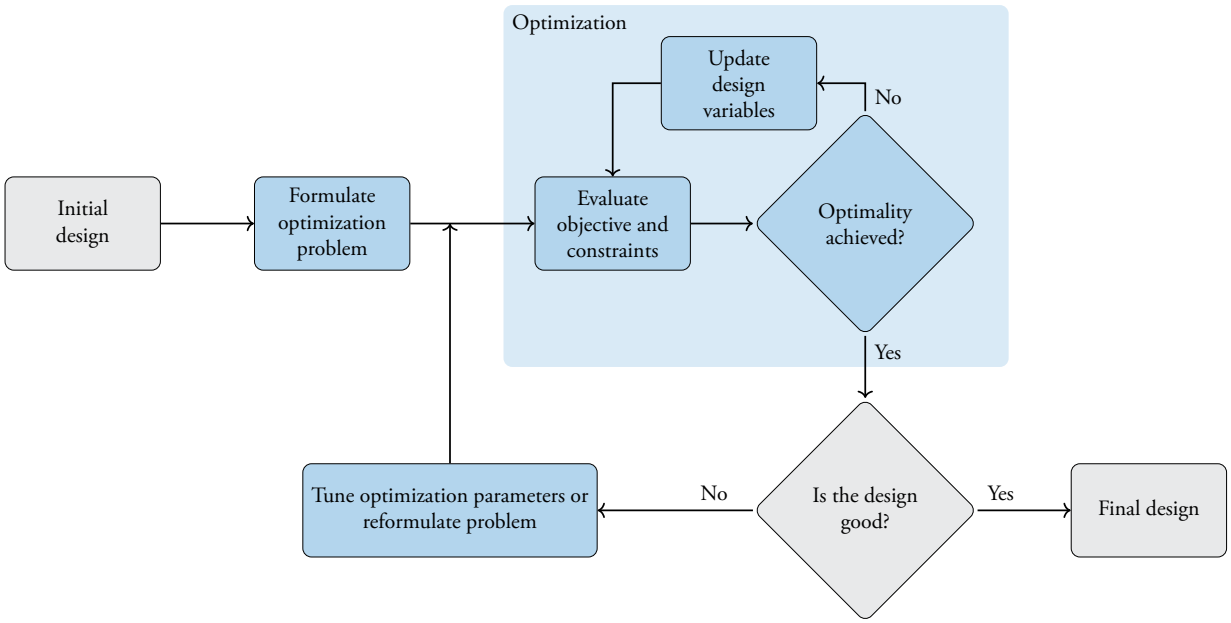


Figure 1.4: A more realistic optimization process, where human intervention is needed to modify the optimization problem. Adapted from Martins and Ning [2].

design variable scaling can significantly affect the performance of the optimization [71, 2]. For example, Bons [72] performed several large-scale aerostructural optimizations, where the scaling factors on various geometric design variables were hand-tuned, requiring considerable time. When starting an optimization, there is no guarantee that it will converge successfully, and the manual interventions necessary are not only time-consuming, but require expertise based on past experience. Figure 1.4 shows a flow chart of this process in a practical setting, which has an outer loop compared to figure 1.3.

Improvements in two aspects are necessary to enable adoption of MDO in an industrial setting. First, optimizations need to achieve a deeper level of convergence than is typically shown. Second, optimizations need to be robust, converging to the numerical optimum without requiring manual tuning or relying on past experience.

[71]: Gill et al. (1981), *Practical Optimization*
[2]: Martins et al. (2021), *Engineering Design Optimization*
[72]: Bons (2020), *High-fidelity Wing Design Exploration with Gradient-based Optimization*

1.3.2 Computational cost

Traditional aircraft design methods typically progress along two separate axes, considering the number of disciplines and the level of fidelity for each discipline. We can show this on a schematic in figure 1.5. On one hand, single disciplines can be analyzed with a large number of fidelities, from analytic models all the way to unsteady simulations requiring thousands of processors. On the other hand, overall aircraft design (OAD) analyzes a large number of disciplines but typically using low-fidelity models.

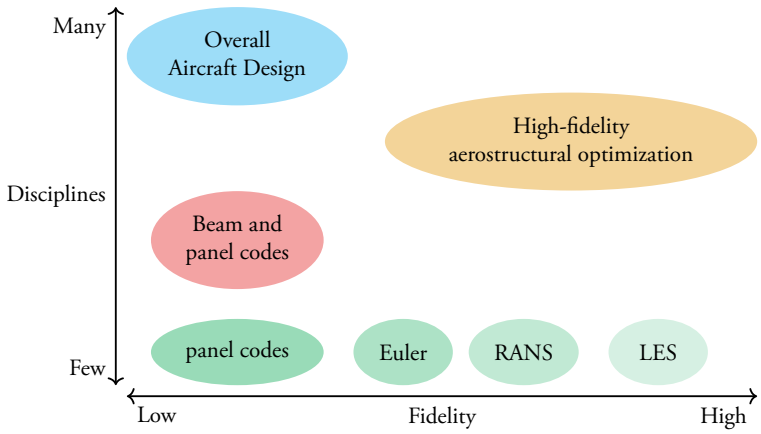


Figure 1.5: Schematic showing typical scope of MDO problems.

The goal of high-fidelity MDO has always been to progress along both axes at once, introducing high-fidelity models to a large number of disciplines. However, as mentioned in section 1.1, such approaches are becoming prohibitively expensive. For example, Brooks, Kenway, and Martins [27] performed a large-scale aerostructural optimization with over a thousand design variables and ten flight conditions, which required about 50 000 core h to complete. With the move towards higher aspect ratio (AR) wing design, scale-resolving simulations [73] and dynamic aeroelastic phenomena need to be considered. Accordingly, the cost is likely to increase dramatically in the years to come. This is particularly important as MDO is most effectively employed during the conceptual design phase, where rapid turnaround time is crucial for the design team.

[27]: Brooks et al. (2018), *Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings*

[73]: Slotnick et al. (2014), *CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences*

Multifidelity methods have emerged as a way to tackle this chal-

lenge. Low-fidelity models can arise in a number of ways, such as reduced solver convergence, coarsened grid, and simplified physics. By leveraging these models, significant cost savings can be realized.

However, existing multifidelity methods are limited to simplified optimization problems and cannot be applied directly to the MDO problems of interest. They cannot account for the multidisciplinary nature of the problem, scale to a large number of design variables and fidelities, or directly handle nonlinear constraints.

1.4 Dissertation objectives

In the previous section, we discussed a few key challenges to performing large-scale MDO for aircraft design. The objectives of this dissertation are now listed below.

1. Improve the convergence rate of gradient-based optimizations and automatically scale design variables, in order to ensure that optimizations converge more quickly and robustly to the optimum.
2. Consider solver convergence as a continuously-varying fidelity, and adaptively control the convergence error during gradient-based ASOs.
3. Incorporate multiple fidelities in a gradient-based optimization framework suitable for large-scale MDO, and automatically determine the appropriate fidelity to be used.
4. Demonstrate the appropriate fidelity MDO framework on an industrially-relevant aircraft design problem.

1.5 Dissertation outline

I start with background information in chapter 2, which contains some useful mathematical definitions. An overview of MDO follows, starting from numerical optimization and building up the mathematics necessary for multidisciplinary problems. I then introduce the computational framework—MACH—which is used throughout this work.

In chapter 3, I discuss a fundamental issue with the geometric parameterization present in MACH. I develop a sensitivity-based parameterization which addresses the shortcomings, and automatically compute appropriate design variable scaling. I then demonstrate this approach on two ASO problems.

Next, I introduce adaptive convergence error control in chapter 4. I derive and verify the adjoint-based convergence error prediction, and develop an adaptation algorithm to control the error during optimization. I then demonstrate the approach on two ASO problems.

In chapter 5, I develop the appropriate fidelity MDO framework where multiple fidelities are used to accelerate convergence of the optimization. I extend the framework in chapter 6, introducing an adjoint-based method for capturing coupled errors that arise in multidisciplinary problems. The approach is demonstrated in chapter 7 on two aerostructural optimizations.

Finally in chapter 8, I discuss the conclusions and contributions of this dissertation, and areas of future work.

Chapter 2

Background

In this chapter, we go over some topics which are foundational in understanding the rest of the dissertation. We start with numerical optimization, then introduce gradient-based methods in the context of single and multidisciplinary problems. Finally, we introduce the computational framework used in this dissertation.

2.1 Numerical optimization

We will consider general nonlinear constrained optimization problems of the form:

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) \\ \text{with respect to} \quad & \boldsymbol{x} \\ \text{subject to} \quad & c_i(\boldsymbol{x}) = 0 \qquad i \in \mathcal{E} \\ & c_i(\boldsymbol{x}) \leq 0 \qquad i \in \mathcal{I} \\ & \boldsymbol{x}_l \leq \boldsymbol{x} \leq \boldsymbol{x}_u, \end{aligned} \tag{2.1}$$

where $\boldsymbol{x}$ are the design variables, $f(\boldsymbol{x})$ is a scalar objective function, $\boldsymbol{c}$ is a vector of nonlinear constraints, and $\boldsymbol{x}_l$ and $\boldsymbol{x}_u$ are lower and upper bounds for the design variables. We follow the notations used by Nocedal and Wright [74], combining the set of equality constraints $\mathcal{E}$ and inequality constraints $\mathcal{I}$ rather than considering them separately. When inequality constraints are present, the concepts of a *feasible point* and *active set* are useful to consider.

[74]: Nocedal et al. (2006), *Numerical Optimization*

Definition 2.1.1 *A point $\boldsymbol{x}$ is deemed to be feasible if it satisfies all constraints. That is,*

$$c_i(\boldsymbol{x}) = 0, \forall i \in \mathcal{E} \tag{2.2}$$

$$c_i(\boldsymbol{x}) \leq 0, \forall i \in \mathcal{I} \tag{2.3}$$

Definition 2.1.2 *The active set $\mathcal{A}(\boldsymbol{x})$ at any feasible point $\boldsymbol{x}$ consists of the equality constraint indices $i \in \mathcal{E}$, and the inequality constraint indices for which $c_i(\boldsymbol{x}) = 0$. That is,*

$$\mathcal{A} = \mathcal{E} \cup \{i \in \mathcal{I} | c_i(\boldsymbol{x}) = 0\} \tag{2.4}$$

We also introduce the concept of the Lagrangian function, which is useful in many aspects of constrained optimization. For the sake of simplicity, we do not introduce slack variables, but instead limit the consideration of inequality constraints to those within the active set $\mathcal{A}$ defined in definition 2.1.2.

Definition 2.1.3 *The Lagrangian function of equation 2.1 is*

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(\boldsymbol{x}) \tag{2.5}$$

where $\boldsymbol{\lambda}$ are the Lagrange multipliers for both the equality and inequality constraints.

In numerical optimization, the optimum of equation 2.1 is given by the solution of a set of equations collectively known as the Karush–Kuhn–Tucker (KKT) conditions. They can be derived by differentiating the Lagrangian and setting the derivatives to zero. However, there are some nuances regarding the derivation, therefore we direct readers to references such as [74, 2].

[74]: Nocedal et al. (2006), *Numerical Optimization*
[2]: Martins et al. (2021), *Engineering Design Optimization*

Definition 2.1.4 *A point $\boldsymbol{x}^*$ is a solution of equation 2.1 if it*

satisfies the following conditions:

$$\nabla f(\boldsymbol{x}^*) + \sum_{i \in \mathcal{A}(\boldsymbol{x}^*)} \lambda_i \nabla c_i(\boldsymbol{x}^*) = 0 \tag{2.6}$$

$$c_i(\boldsymbol{x}^*) = 0, \quad \forall i \in \mathcal{E} \tag{2.7}$$

$$c_i(\boldsymbol{x}^*) \leq 0, \quad \forall i \in \mathcal{I} \tag{2.8}$$

$$\lambda_i^* \geq 0, \quad \forall i \in \mathcal{I} \tag{2.9}$$

$$\lambda_i^* c_i(\boldsymbol{x}^*) = 0, \quad \forall i \tag{2.10}$$

In this work, we will refer to equation 2.6 as the first-order condition, which is analogous to the condition $\nabla f = 0$ in the unconstrained case. Equations 2.7 and 2.8 are simply restating the constraints, and we refer to those as zeroth-order conditions due to the lack of gradients.[1] Equation 2.9 requires the Lagrange multipliers corresponding to inequality constraints to be nonnegative. Finally, equation 2.10 is referred to as the complimentarity constraint, and dictates that either the constraint i is active, or $\lambda_i^* = 0$.

1: Most texts refer to all the KKT conditions as first-order conditions, since they are derived by taking the first derivative of the Lagrangian and setting it to zero. However, here we emphasize that they are zeroth order with respect to the objective and constraint functions.

It is important to note here that while necessary, these are not sufficient conditions for $\boldsymbol{x}^*$ to be a local *minimum*. A second-order condition exists, analogous to requiring the Hessian of the objective to be symmetric positive-definite (SPD) in the unconstrained case. However, we omit such discussions here as gradient-based optimizers ensure the satisfaction of this condition through a combination of line search safeguards and inherent properties of the approximate Hessian updates. Similarly, equations 2.9 and 2.10 are also satisfied by the design of the optimization algorithm and typically not considered explicitly.

The Lagrange multipliers $\boldsymbol{\lambda}$ introduced in definition 2.1.3 have physical meaning. At the optimum, the Lagrange multipliers $\boldsymbol{\lambda}^*$ provide the sensitivity of the optimum with respect to each constraint [2, Sec. 5.3.3]:

[2]: Martins et al. (2021), *Engineering Design Optimization*

$$\lambda_i^* = -\frac{\mathrm{d}f}{\mathrm{d}c_i} \tag{2.11}$$

If a Lagrange multiplier is large, a small change in the corresponding constraint will cause a significant change in the optimum. Conversely, if a Lagrange multiplier is zero, then the constraint has no effect on the optimum and is therefore inactive by definition.

Gradient-based optimizers are not guaranteed to converge to the global optimum, and SNOPT is no exception. Nevertheless, a number of studies have investigated the possibility of multimodality in the context of ASO. Chernukhin and Zingg [78] compared a gradient-based multistart (GBMS) algorithm with a gradient-free approach on a suite of 2D and 3D ASO problems, and concluded that multimodality is rare in most cases, especially with limited geometric DOFs. In such cases, the GBMS strategy was the most efficient at finding the global optimum. Similar findings were reported by Lyu, Kenway, and Martins [79] on the Common Research Model (CRM) benchmark problem.

In reality, very few gradient-free optimizers are globally-convergent, with the notable exception of DIviding RECTangles (DIRECT) [75]. Most gradient-free optimizers fall under the classification of *metaheuristic* optimizers, and they suffer from some notable issues [76] such that certain journals are placing new requirements on publication [77]

[78]: Chernukhin et al. (2013), *Multimodality and Global Optimization in Aerodynamic Design*

[79]: Lyu et al. (2015), *Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark*

Certain ASOs do produce multiple local minima [80], and this was extensively investigated by Bons et al. [81], who concluded that most of the solutions were due to a lack of either physics in the numerical models or certain constraints. In particular, Euler solutions proved to be multimodal, but the introduction of viscous effects removed many of these local minima. This conclusion supports earlier results by Poole, Allen, and Rendall [82] who reported a number of local minima when using Euler solutions. It is also well-known that Euler solutions are not guaranteed to be unique [83, 84], but this does not appear to be a factor in these results.

[80]: Skinner et al. (2018), *State-of-the-art in aerodynamic shape optimisation methods*

[81]: Bons et al. (2019), *Multimodality in Aerodynamic Wing Design Optimization*

[82]: Poole et al. (2018), *Global Optimization of Wing Aerodynamic Optimization Case Exhibiting Multimodality*

[83]: Jameson et al. (2012), *Further Studies of Airfoils Supporting Non-Unique Solutions in Transonic Flow*

[84]: Destarac et al. (2018), *Example of a Pitfall in Aerodynamic Shape Optimization*

Using RANS, Reist et al. [68] compared two different parameterizations and two different optimizers on the same ASO problem. They used the same CFD solver for consistency, and found minor differences in the solutions that are likely due to inherent differences in the underlying geometric parameterization. Streuber and Zingg [85] performed a comprehensive study of multimodality for several ASO problems using GBMS and a RANS solver, and

[68]: Reist et al. (2020), *Cross Validation of Aerodynamic Shape Optimization Methodologies for Aircraft Wing-Body Optimization*

[85]: Streuber et al. (2021), *Evaluating the Risk of Local Optima in Aerodynamic Shape Optimization*

found that the risk of multimodality is low when performing detailed design with a good initial geometry. For exploratory studies, the risk of multimodality increases substantially. Martins [1] instead hypothesizes that the local minima found are "spurious", and are numerical artifacts rather than true local minima. Bons and Martins [86] further introduced structures into the analysis, and performed two aerostructural optimizations starting from different baselines. The authors concluded that the problem investigated was unimodal, and that the discrepancies in the structural design variables are due to poor optimization convergence. Some further discussions on this topic are provided in appendix C.3.

[1]: Martins (2022), *Aerodynamic Design Optimization: Challenges and Perspectives*

[86]: Bons et al. (2020), *Aerostructural Design Exploration of a Wing in Transonic Flow*

2.2 PDE-constrained optimization

With a reliable optimizer at our disposal, the logical next step would be to perform optimizations involving complex simulations such as CFD analyses. A common example would be ASO, where the external shape of an aerodynamic body is modified in order to improve its performance.

In the general case, we define an analysis as a function with inputs and outputs, *i.e.*,

$$\boldsymbol{\mathcal{I}} = f(\boldsymbol{x}) \tag{2.12}$$

with the design vector $\boldsymbol{x}$ as inputs and the quantity of interest (QOI) $\boldsymbol{\mathcal{I}}$ as the output.

In practice, the function is often implicit in nature, requiring a complex iterative solver which solves a discrete partial differential equation (PDE) such as a CFD solver. Therefore, it is easier to discuss the implicit form, where the residual function $\boldsymbol{R}$ is converged by varying the state variables $\boldsymbol{u}$ such that

$$\boldsymbol{R}(\boldsymbol{u}^*, \boldsymbol{x}) = 0. \tag{2.13}$$

Here the asterisk indicates that the quantity is converged. This is called the *primal* system, and is typically a system of nonlinear equations representing the discretized governing equations.

Once this is done, the function of interest $\mathcal{I}$ (also known as a *functional*) can be computed as an explicit function:

$$\mathcal{I} = f(\boldsymbol{u}^*, \boldsymbol{x}). \tag{2.14}$$

In an optimization setting, the functionals would naturally be the objective and constraint functions.

There are two approaches to tackle an optimization involving functions of this form. The first, called the full-space or one-shot method, uses the optimizer to simultaneously solve the optimization problem and equation 2.13, by setting both $\boldsymbol{x}$ and $\boldsymbol{u}$ as design variables, and letting $\boldsymbol{R}$ as a set of nonlinear equality constraints.[2] It is analogous to the SAND approach in MDO architectures. The reduced-space method, on the other hand, only lets the optimizer control the design variables $\boldsymbol{x}$, while using an existing PDE solver to converge the residual equations at every iteration of the optimization. It is analogous to the MDF approach, and an extended design structure matrix (XDSM) diagram [87] of the process is shown in figure 2.1. The XDSM diagram is an extension of the design structure matrix commonly used to depict the process and data flow in MDO.

While attractive in principle, the full-space approach is still less popular than the reduced-space approach for a number of reasons [88]. In the reduced-space approach, solution of the governing equations is handled by a dedicated solver which may be better fine-tuned for the task. In addition, it integrates well with black-box functions, without having to expose internal states of the solver to the optimizer in order to converge the residual equations. We consider the reduced-space approach in this work. Figure 2.2 shows an XDSM diagram of the reduced-space approach.

2: In some literatures, equation 2.13 is referred to as the constraints.

[87]: Lambe et al. (2012), *Extensions to the Design Structure Matrix for the Description of Multidisciplinary Design, Analysis, and Optimization Processes*

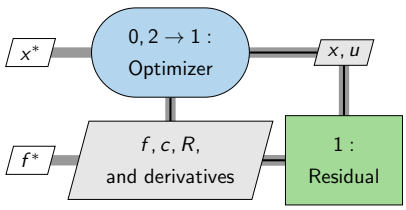


Figure 2.1: The XDSM diagram for a single-discipline optimization using the full-space approach.

[88]: Hicken et al. (2013), *Comparison of Reduced- and Full-space Algorithms for PDE-constrained Optimization*

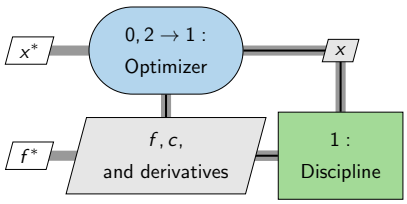


Figure 2.2: The XDSM diagram for a single-discipline optimization using the reduced-space approach.

A variety of root-finding algorithms exist to tackle equation 2.13. In CFD applications, pseudo-transient continuation techniques are popular, using either explicit or implicit time integration schemes such as Runge–Kutta (RK). Newton's method[3] is also popular for its quadratic convergence rate [2, Sec. 3.6], and the resulting linear systems are often solved using the Krylov subspace method, resulting in the Newton–Krylov (NK) method [89]. However, NK only converges in the neighbourhood of the solution, and is therefore often employed as the terminal phase of a hybrid convergence strategy involving an appropriate globalization method.

3: Also known as the Newton–Raphson method

[2]: Martins et al. (2021), *Engineering Design Optimization*

[89]: Knoll et al. (2004), *Jacobian-free Newton–Krylov methods: a survey of approaches and applications*

While many methods exist for gradient computation, only the *adjoint method* scales well with the number of inputs and is efficient for implicit functions that are typically solved iteratively [2, Sec. 6.7]. A comprehensive of adjoint methods, including its derivation and solution methods, can be found in [16]. This approach involves solving the eponymous adjoint equation

$$\left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}}\right)^{\mathsf{T}} \boldsymbol{\psi} = \left(\frac{\partial \mathcal{I}}{\partial \boldsymbol{u}}\right)^{\mathsf{T}} \tag{2.15}$$

for the adjoint vector $\boldsymbol{\psi}$.

[2]: Martins et al. (2021), *Engineering Design Optimization*

[16]: Kenway et al. (2019), *Effective Adjoint Approaches for Computational Fluid Dynamics*

This system is also known as the *dual* system, and in contrast to the *primal* system above, is a linear system. The adjoint variables $\boldsymbol{\psi}$ are also known as *dual variables*, again in contrast to the *primal variables* $\boldsymbol{u}$. This system is typically solved in a matrix-free fashion using the Krylov method, and the partial derivatives are formed using AD, called the ADjoint method [90]. However, fixed-point iterations are also popular [91]. There are important mathematical theories governing the primal-dual consistency of the solution scheme for the two systems [92, 93].

[90]: Mader et al. (2008), *ADjoint: An Approach for the Rapid Development of Discrete Adjoint Solvers*

[91]: Albring et al. (2016), *Efficient aerodynamic design using the discrete adjoint method in SU2*

[92]: Peter et al. (2010), *Numerical Sensitivity Analysis for Aerodynamic Optimization: A Survey of Approaches*

[93]: Gomes et al. (2022), *Pitfalls of Discrete Adjoint Fixed-Points Based on Algorithmic Differentiation*

Once solved, the adjoint vector can then be used to compute

the total derivative:

$$\frac{\mathrm{d}\mathcal{I}}{\mathrm{d}\boldsymbol{x}} = \frac{\partial \mathcal{I}}{\partial \boldsymbol{x}} - \boldsymbol{\psi}^\mathsf{T} \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}. \tag{2.16}$$

The key benefit of this approach is that the design variables do not appear in equation 2.15, which is solved once per $\mathcal{I}$. The overall cost is proportional to the number of functions of interest, independent of the size of the design variables.

Comment 2.2.1

It is by no means a coincidence that equations 2.6 and 2.16 share the same form. If we take the view of the one-shot approach, then the residuals $\boldsymbol{R}$ are equality constraints of the optimization problem, with the corresponding Lagrangian as

$$\mathcal{L} = f(\boldsymbol{x}) + \boldsymbol{\lambda}^\mathsf{T} \boldsymbol{R}(\boldsymbol{u}, \boldsymbol{x}) \tag{2.17}$$

The corresponding KKT condition would be exactly the same as equation 2.6, with the adjoint vector $\boldsymbol{\psi}$ as the Lagrange multipliers. The difference in the sign is due to our definition of the adjoint vector, so perhaps it is more accurate to state that the adjoint vector is the negative of the Lagrange multipliers. There are some who use a different convention, where the adjoint vector is defined as the negative of what is defined here [94].

A more realistic XDSM diagram is shown in figure 2.3 for an ASO problem. It contains a number of additional components to enable necessary tasks such as the volume mesh warping.

2.3 Multidisciplinary design optimization

So far, we have discussed single-disciplinary problems, *i.e.*, problems that consider a PDE solver arising from a single discipline.

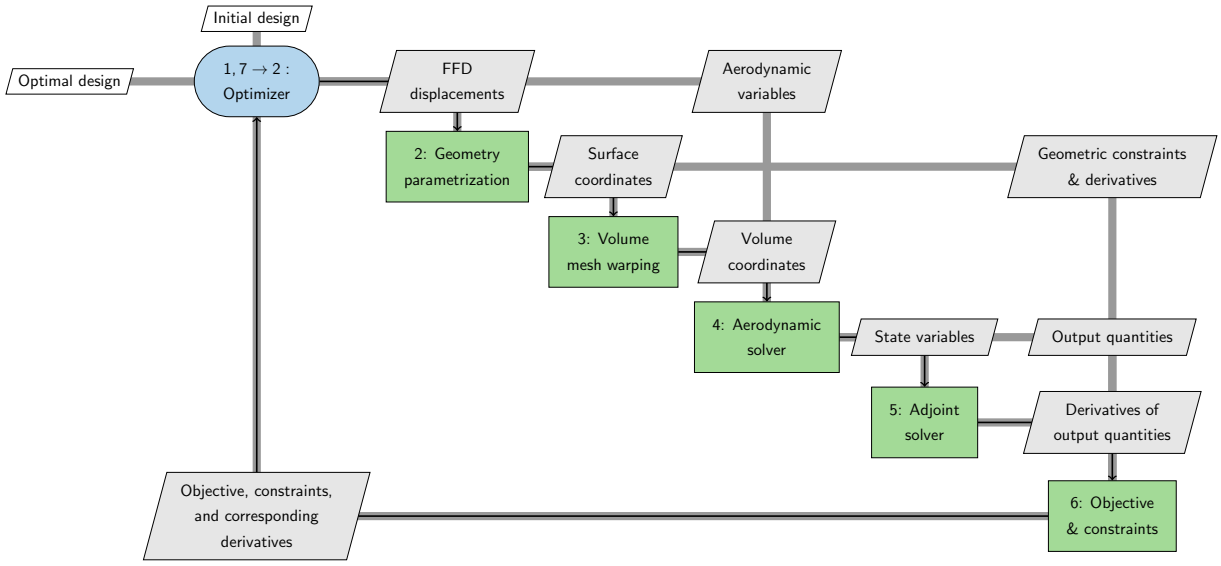


Figure 2.3: An example XDSM diagram of an ASO problem. In this case the process within MACH is given, but the process is similar for other frameworks.

However, many engineering problems of practical interest involve multiple disciplines that interact with each other. A common example is the aerostructural system,[4] involving aerodynamics and structural mechanics.

4: Sometimes it is referred to as fluid-structure interaction (FSI).

While it is possible to consider a multidisciplinary system simply as a larger PDE with an expanded set of residuals, this is often not done. Maintaining the individual disciplinary structure allows us to reuse these single-discipline solvers in its entirety, and the existing preconditioners can also be reused in some cases [21]. We use the term *discipline* here, but they can be more generally called *components* [2, Sec. 13.2.1], for example in the context of OpenMDAO [36].

[21]: Kenway et al. (2014), *Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Adjoint Derivative Computations*

[2]: Martins et al. (2021), *Engineering Design Optimization*

[36]: Gray et al. (2019), *OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization*

With the introduction of multiple disciplines, we need to compute both the coupled solution (known as MDA) as well as the coupled derivatives. We will illustrate the problem using a model with two disciplines. In order to keep the discussion general, we have opted for a generic description based on past work on MDO architectures [33].

[33]: Martins et al. (2013), *Multidisciplinary Design Optimization: A Survey of Architectures*

First there are the governing equations for each discipline, writ-

ten in residual form

$$\boldsymbol{R}_1(\boldsymbol{x}, \boldsymbol{u}_1, \boldsymbol{y}_2) = 0 \tag{2.18}$$

$$y_1 = \mathcal{G}_1(\boldsymbol{x}, \boldsymbol{u}_1) \tag{2.19}$$

$$\boldsymbol{R}_2(\boldsymbol{x}, \boldsymbol{u}_2, \boldsymbol{y}_1) = 0 \tag{2.20}$$

$$y_2 = \mathcal{G}_2(\boldsymbol{x}, \boldsymbol{u}_2) \tag{2.21}$$

Here the subscript denotes the discipline a given variable belongs to, $\boldsymbol{R}_i$ are the residual equations, $\boldsymbol{u}_i$ are the state variables, $\boldsymbol{x}$ is the design vector, and $\boldsymbol{y}$ are the coupling variables.

These equations are coupled via $\boldsymbol{y}$, and require an iterative solution strategy. Two popular approaches are nonlinear block Gauss–Seidel (NLBGS) (also known as fixed-point iterations), and coupled Newton [2, Sec.13.2.5].

[2]: Martins et al. (2021), *Engineering Design Optimization*

In a typical NLBGS iteration, we start with discipline 1, make an initial guess on the coupling variables y_2, and solve for the states u_1 that would satisfy equation 2.18. Then, the coupling variables from discipline 1 are computed using equation 2.19, yielding y_1. This is passed along to discipline 2, where the states u_2 are solved, and the coupling variables y_2 computed. These values are then passed back to discipline 1, and the entire process is repeated until the changes in the coupling variables are minimal, at which point the coupled system is deemed to be converged. Note that within each iteration, the state variables u_i which are solved only satisfy the governing equations for its own discipline, and not the multidisciplinary system. With each NLBGS iteration, the converged states will change due to the updated coupling variables, eventually settling into a converged value. This process is illustrated in figure 2.4.

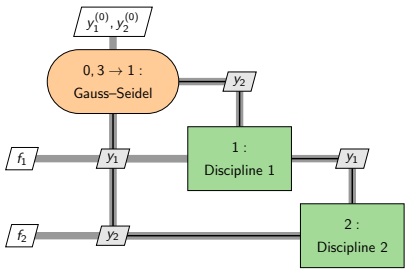


Figure 2.4: An XDSM diagram of the NLBGS process.

In reality, there is an additional step of mapping the coupling variables from one discipline to another, but here we incorporate that step as part of functions $\mathcal{G}_i$ for simplicity. Furthermore, it is common for each discipline to work with a subset of the overall design vector x. In their review paper, Martins and Lambe

[33] made the distinction between local and global design vectors. However, for simplicity we do not make such distinctions here.

[33]: Martins et al. (2013), *Multidisciplinary Design Optimization: A Survey of Architectures*

Once the entire system is converged, the functional outputs can then be computed. These are then combined into system-level objective and constraints, and passed to the optimizer along with their gradients.

The sensitivities can be computed using the coupled adjoint method [19], the multidisciplinary extension of the adjoint method above. In essence, equation 2.15 becomes

[19]: Martins et al. (2005), *A Coupled-Adjoint Sensitivity Analysis Method for High-Fidelity Aero-Structural Design*

$$\begin{bmatrix} \dfrac{\partial \boldsymbol{R}_1}{\partial \boldsymbol{u}_1} & \dfrac{\partial \boldsymbol{R}_1}{\partial \boldsymbol{u}_2} \\ \dfrac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{u}_1} & \dfrac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{u}_2} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \boldsymbol{\psi}_1 \\ \boldsymbol{\psi}_2 \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \mathcal{I}}{\partial \boldsymbol{u}_1} & \dfrac{\partial \mathcal{I}}{\partial \boldsymbol{u}_2} \end{bmatrix}^{\mathsf{T}} \tag{2.22}$$

The corresponding total derivative equation is:

$$\frac{\mathrm{d}\mathcal{I}}{\mathrm{d}\boldsymbol{x}} = \frac{\partial \mathcal{I}}{\partial \boldsymbol{x}} - \begin{bmatrix} \boldsymbol{\psi}_1^{\mathsf{T}} & \boldsymbol{\psi}_2^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} \dfrac{\partial \boldsymbol{R}_1}{\partial \boldsymbol{x}} \\ \dfrac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{x}} \end{bmatrix}. \tag{2.23}$$

2.4 Analysis fidelities and sources of error

In an engineering setting, the same analysis can often be performed using several different tools varying in model fidelity. Some tools are known to be more accurate but more computationally expensive, while others are cheap but inaccurate. Relatively speaking, higher-fidelity tools have higher accuracy.

We define accuracy as the lack of *systematic* errors, where precision is the lack of *random* errors (often treated as noise). In the field of uncertainty quantification, those are often referred to as *epistemic* and *aleatoric* uncertainties, respectively. Figure 2.5 shows a schematic of these two effects. The concept of fidelity is

usually related to the systematic or *epistemic* errors due to the inadequacy of the model, and other sources of error are discussed in appendix C.2.

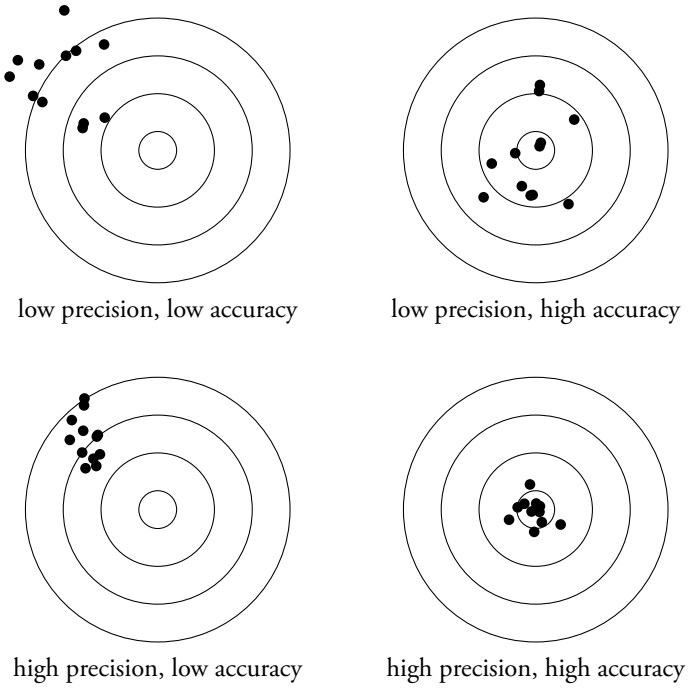


Figure 2.5: Schematic showing the difference between accuracy and precision.

Compared to the high-fidelity model, low-fidelity models can contain a number of systematic errors [95] arising from sources such as:

- Model error due to simplified physics
- Numerical error
  - Discretization error
  - Convergence error
- Simplified geometry or boundary conditions

[95]: Giselle Fernández-Godino et al. (2019), *Issues in Deciding Whether to Use Multifidelity Surrogates*

In contrast to [96], we do not consider surrogate models as low-fidelity models, since they typically require a large set of training data that are expensive to generate, and the cost of training is often not considered. Instead, we require all fidelities to be physics-based.

[96]: Peherstorfer et al. (2018), *Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization*

Out of these sources of error, convergence error is often treated as a continuously-varying error in the field of variable fidelity

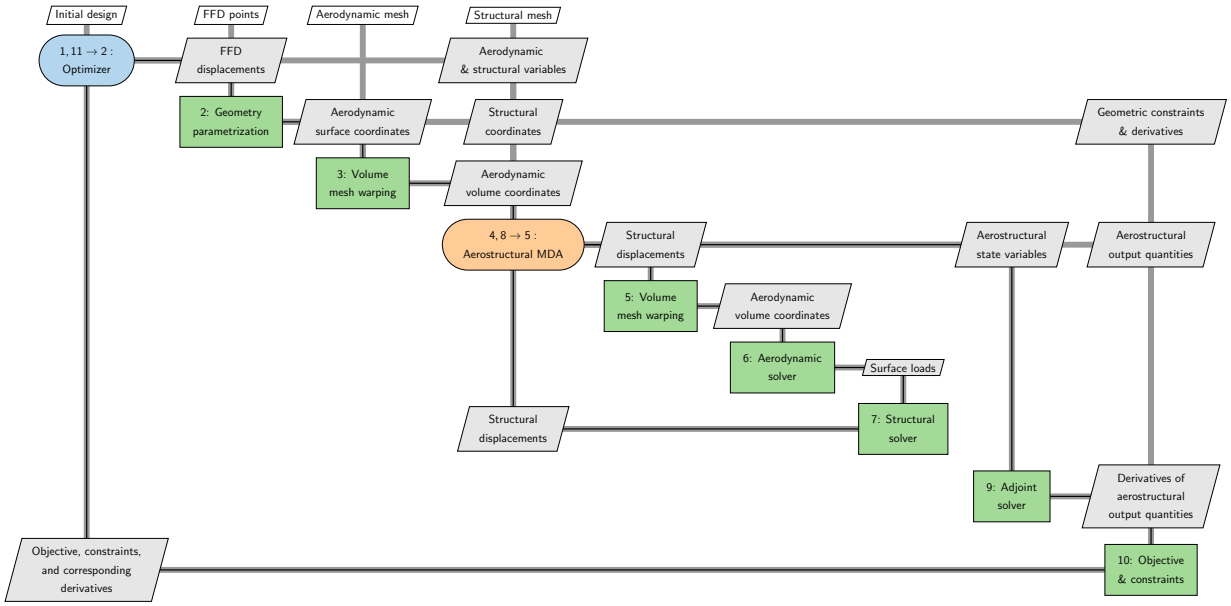


Figure 2.6: An XDSM diagram of the MACH framework applied to a single-point aerostructural optimization.

optimization. This is explored in chapter 4 where the convergence error is adapted during optimization. The rest of the errors are typically discrete, and chapters 5 to 7 focus on the development of the appropriate fidelity framework for managing these fidelities during optimization.

2.5 Computational framework

We use the MACH framework [21, 42] to perform aerostructural optimizations. It uses the MDF formulation of the optimization problem and gradient-based optimizers to perform aerostructural optimizations. An XDSM diagram of the framework is given in figure 2.6.

Some of the historical developments and capabilities of MACH have been given already in section 1.1.3. To date, MACH has been applied successfully to a wide range of problems. ASOs have been performed on aircraft [97, 98, 99, 100] and wind turbine applications [101, 102]. Similarly, aerostructural optimizations have been performed for aircraft [103, 104, 105], wind tur-

[21]: Kenway et al. (2014), *Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Adjoint Derivative Computations*
[42]: Kenway et al. (2014), *Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration*

[97]: Secco et al. (2019), *RANS-based Aerodynamic Shape Optimization of a Strut-braced Wing with Overset Meshes*
[98]: Mangano et al. (2021), *Multipoint Aerodynamic Shape Optimization for Subsonic and Supersonic Regimes*
[99]: Chauhan et al. (2021), *RANS-Based Aerodynamic Shape Optimization of a Wing Considering Propeller-Wing Interaction*
[100]: Seraj et al. (2022), *Predicting the High-Angle-of-Attack Characteristics of a Delta Wing at Low Speed*
[101]: Dhert et al. (2017), *Aerodynamic Shape Optimization of Wind Turbine Blades Using a Reynolds-Averaged Navier–Stokes Model and an Adjoint Method*
[102]: Madsen et al. (2019), *Multipoint high-fidelity CFD-based aerodynamic shape optimization of a 10 MW wind turbine*

[103]: Bons et al. (2020), *High-fidelity Aerostructural Optimization Studies of the Aerion AS2 Supersonic Business Jet*
[104]: Brelje et al. (2021), *Aerostructural Wing Optimization for a Hydrogen Fuel Cell Aircraft*
[105]: Bons et al. (2022), *Aerostructural wing optimization of a regional jet considering mission fuel burn*

bines [106], and hydrofoils [107, 108]. Aeropropulsive [109, 110, 32] and conjugate heat transfer [37] applications have also been explored.

We now describe the components of MACH in detail.

2.5.1 Geometric parameterization

We use pyGeo [111][5] for geometric parameterization. pyGeo uses free-form deformation (FFD) volumes to deform both the aerodynamic surface mesh and the structural mesh. We start by embedding a cloud of points called a "point set", which typically includes the surface mesh coordinates for aerodynamics or the finite element mesh nodes for structures. These points are manipulated via geometric operations on the FFD nodes, which then apply the deformation to the embedded points.

However, instead of allowing each FFD node to move arbitrarily in 3-dimensional space, we instead group them into intuitive sets of design variables. These operations can grow to be rather complicated and include nested geometric operations. Some examples include twist variables, which are rotations of a slice of FFD nodes at a spanwise section around a pre-defined rotation axis, and shape variables, which are vertical displacements of individual FFD nodes. This approach allows for global and local shape control, and the design variables represent quantities familiar to designers.

2.5.2 Volume mesh warping

Once the aerodynamic surface mesh is deformed by pyGeo, the volume mesh must be updated accordingly. Instead of re-extruding the volume mesh, we deform it in a continuous fashion to maintain smooth gradients. While this can be done using the FFD once more—by adding farfield control points and embedding the entire volume mesh [112, Sec. 5]—this approach is

[106]: Mangano et al. (2022), *Towards Passive Aeroelastic Tailoring of Large Wind Turbines Using High-Fidelity Multidisciplinary Design Optimization*

[107]: Liao et al. (2021), *3-D High-Fidelity Hydrostructural Optimization of Cavitation-Free Composite Lifting Surfaces*

[108]: Liao et al. (2022), *RANS-based Optimization of a T-shaped Hydrofoil Considering Junction Design*

[109]: Gray et al. (2018), *Modeling Boundary Layer Ingestion Using a Coupled Aeropropulsive Analysis*

[110]: Gray et al. (2019), *Coupled Aeropropulsive Design Optimization of a Boundary-Layer Ingestion Propulsor*

[32]: Yildirim et al. (2022), *Boundary Layer Ingestion Benefit for the STARC-ABL Concept*

[111]: Kenway et al. (2010), *A CAD-Free Approach to High-Fidelity Aerostructural Optimization*

5: `https://github.com/mdolab/pygeo`

[112]: Madsen (2020), *High-Fidelity CFD-based Shape Optimization of Wind Turbine Blades*

inferior at maintaining mesh orthogonality and becomes less effective with large geometric changes.

Algebraic methods [113], linear elasticity-based methods [114, 115], and hybrid methods [111] have all been developed to deform the volume mesh smoothly while maintaining mesh quality. In this work, we use IDWarp [116],[6] uses an inverse-distance algorithm based on the work of Luke, Collins, and Blades [117].

[113]: Reuther et al. (1996), *Aerodynamic Shape Optimization of Complex Aircraft Configurations via an Adjoint Formulation*

[114]: Dwight (2009), *Robust Mesh Deformation using the Linear Elasticity Equations*

[115]: Biedron et al. (2009), *Recent Enhancements to the FUN3D Flow Solver for Moving-Mesh Applications*

[111]: Kenway et al. (2010), *A CAD-Free Approach to High-Fidelity Aerostructural Optimization*

[116]: Secco et al. (2021), *Efficient Mesh Generation and Deformation for Aerodynamic Shape Optimization*

6: `https://github.com/mdolab/idwarp`

[117]: Luke et al. (2012), *A Fast Mesh Deformation Method Using Explicit Interpolation*

2.5.3 CFD solver

The aerodynamic analysis is performed using ADflow [118],[7] a finite-volume CFD solver with an efficient adjoint implementation suitable for ASO [16]. ADflow can solve both the Euler and RANS equations with various turbulence models, and the Spalart–Allmaras (SA) turbulence model is used for all RANS-based analyses in this work. To solve the governing equations, we use approximate Newton–Krylov (ANK) [119] to globalize the terminal NK iterations. For Euler simulations, we also compute a viscous drag correction based on a flat-plate estimate with form factor corrections [42]. ADflow contains an efficient adjoint solver to compute the derivatives necessary for gradient-based optimization.

[118]: Mader et al. (2020), *ADflow: An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization*

7: `https://github.com/mdolab/adflow`

[16]: Kenway et al. (2019), *Effective Adjoint Approaches for Computational Fluid Dynamics*

[119]: Yildirim et al. (2019), *A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations*

[42]: Kenway et al. (2014), *Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration*

2.5.4 CSM solver

The structure is analyzed using the Toolkit for Analysis of Composite Structures (TACS) [120], a finite-element method (FEM) solver designed for gradient-based optimizations of thin-walled structures. In order to tackle the poorly conditioned linear systems arising from thin shell structures, it uses a parallel direct factorization method. It also has an efficient adjoint implementation to facilitate gradient computations.

[120]: Kennedy et al. (2014), *A Parallel Finite-Element Framework for Large-Scale Gradient-Based Design Optimization of High-Performance Structures*

2.5.5 Aerostructural solver

We use the approach outlined by Kenway, Kennedy, and Martins [21] to solve the coupled aeroelastic system and compute the coupled adjoint.

[21]: Kenway et al. (2014), *Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Adjoint Derivative Computations*

First, the two disciplines are coupled via the load and displacement scheme. We use rigid links [121, 20], which can be shown to be consistent and conservative. The same volume mesh warping scheme from section 2.5.2 is used to update the aerodynamic volume mesh based on structural displacements. The MDA is solved using NLBGS with Aitken acceleration [122]. The coupled adjoint is solved using the coupled Krylov method, with the linear algebra package Portable, Extensible Toolkit for Scientific Computation (PETSc). By leveraging the matrix-free AD routines, the generalized minimal residual (GMRES) iterations are very memory-efficient.

[121]: Brown (1997), *Displacement extrapolations for CFD+CSM aeroelastic analysis*
[20]: Martins et al. (2004), *High-Fidelity Aerostructural Design Optimization of a Supersonic Business Jet*

[122]: Irons et al. (1969), *A version of the Aitken accelerator for computer iteration*

2.5.6 Optimization framework

We use pyOptSparse [123],[8] an optimization framework designed for constrained nonlinear optimization of large sparse problems. It provides a unified interface for various gradient-free and gradient-based optimizers, which are listed in table 2.1 By using an object-oriented approach, the software maintains independence between the optimization problem formulation and the implementation of the specific optimizers. The code is Message Passing Interface (MPI)-wrapped to enable execution of expensive parallel analyses and gradient evaluations, such as when using CFD simulations, which can require hundreds of processors. Aside from MACH, pyOptSparse has been incorporated in several other design frameworks such as OpenMDAO [36] and SUAVE [50].

[123]: Wu et al. (2020), *pyOptSparse: A Python framework for large-scale constrained nonlinear optimization of sparse systems*

8: `https://github.com/mdolab/pyoptsparse`

[36]: Gray et al. (2019), *OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization*

[50]: MacDonald et al. (2017), *SUAVE: An Open-Source Environment Enabling Multi-Fidelity Vehicle Optimization*

Optimizer	Type	Gradient	Sparsity
ALPSO [124]	PSO		
CONMIN [125]	MFD	✓	
IPOPT [126]	IP	✓	✓
NLPQLP [127, 128]	SQP	✓	
NSGA2 [129]	GA		
ParOpt [130]	IP or TR	✓	✓
PSQP	SQP	✓	
SLSQP [131]	SQP	✓	
SNOPT [132]	SQP	✓	✓

Table 2.1: Brief overview of available optimizers in pyOptSparse and their capabilities.

2.5.6.1 Selecting an appropriate optimizer

The well-known No Free Lunch Theorem [133] states that, when considering all possible optimization problems, all optimization algorithms perform equally well on average. However, as pointed out by Adam et al. [134], this does not imply that the same is true for a specific subset of optimization problems, where certain optimizers can be more effective. To date, a large number of optimizers have been developed, and some are tailored to specific characteristics of certain classes of problems in order to be more efficient. For example, ParOpt [130] can take advantage of the sparsity structure in constraint Jacobians that arise from topology optimization problems.

[133]: Wolpert et al. (1997), *No free lunch theorems for optimization*

[134]: Adam et al. (2019), *No free lunch theorem: A review*

[130]: Chin et al. (2019), *A Scalable Framework for Large-Scale 3D Multimaterial Topology Optimization with Octree-based Mesh Adaptation*

Due to the difficulties in performing optimizer benchmarks [135], we focus on applications to relevant real-world engineering problems. While it is common [136, 137] to benchmark optimization algorithms against analytical benchmarks such as CUTEr [138] and CUTEst [139], aerodynamic and aerostructural optimizations—the type of optimizations of interest in this work—have several defining features. First, they are general nonlinear programming problems as described by equation 2.1, exhibiting nonlinearities in both the objective and constraints, and with no guarantees of convexity. Second, they are expensive functions that dwarf the cost of the optimizer itself.[9] Third, they often have efficient gradients available through the implementation of the adjoint or coupled-adjoint method. Last, the functions are often noisy, due to a number of effects which are

[135]: Beiranvand et al. (2017), *Best practices for comparing optimization algorithms*

[136]: Mittelmann (**year**), *AMPL-NLP Benchmark*
[137]: Gill et al. (2015), *On the performance of SQP methods for nonlinear optimization*
[138]: Gould et al. (2004), *CUTEr (and SifDec), a Constrained and Unconstrained Testing Environment, revisited*
[139]: Gould et al. (2014), *CUTEst: A Constrained and Unconstrained Testing Environment with safe threads for mathematical optimization*

9: This is one reason why analytic benchmarks are problematic, often the runtime is dominated by the optimizer cost which in reality would be negligible.

discussed in more detail in appendix C.2.

To date, a number of studies have investigated the effectiveness of different optimization algorithms on a variety of engineering design problems. Zingg, Nemec, and Pulliam [140], Lyu, Xu, and Martins [141], and Yu et al. [142] investigated the choice of optimization algorithm on a suite of aerodynamic optimization problems, and concluded that gradient-based optimizers are significantly more efficient than gradient-free optimizers when the problem dimension is large. Following that comparison, they also benchmarked several popular gradient-based optimizers, and concluded that SNOPT was the most efficient optimizer. Similar conclusions have been drawn by Wendorff, Botero, and Alonso [143] when performing aircraft conceptual design through SUAVE, and Baker et al. [144] in the context of wind farm layout optimization. It has been the optimizer of choice within MACH, leading to several seminal works [79, 42, 27]. Furthermore, SNOPT has been extremely popular within the broader field of aerodynamic and aerostructural optimizations, being used by frameworks such as Jetstream at University of Toronto Institute for Aerospace Studies (UTIAS) [145], Stanford University Unstructured (SU2) at Stanford University [67, 146],[10] and EllipSys3D from Technical University of Denmark (DTU) [112]. A large number of cases developed by ADODG at AIAA [11] have also been produced using SNOPT as the optimizer [147, 148]. Based on these results, we choose to use SNOPT for all optimizations presented here, although many optimizer-relevant developments are not specific to SNOPT.

[140]: Zingg et al. (2008), *A Comparative Evaluation of Genetic and Gradient-Based Algorithms Applied to Aerodynamic Optimization*

[141]: Lyu et al. (2014), *Benchmarking Optimization Algorithms for Wing Aerodynamic Design Optimization*

[142]: Yu et al. (2018), *On the Influence of Optimization Algorithm and Starting Design on Wing Aerodynamic Shape Optimization*

[143]: Wendorff et al. (2016), *Comparing Different Off-the-Shelf Optimizers' Performance in Conceptual Aircraft Design*

[144]: Baker et al. (2019), *Best Practices for Wake Model and Optimization Algorithm Selection in Wind Farm Layout Optimization*

[79]: Lyu et al. (2015), *Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark*

[42]: Kenway et al. (2014), *Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration*

[27]: Brooks et al. (2018), *Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings*

[145]: Hicken et al. (2010), *Aerodynamic Optimization Algorithm with Integrated Geometry Parameterization and Mesh Movement*

[67]: Masters et al. (2017), *Influence of Shape Parameterization on a Benchmark Aerodynamic Optimization Problem*

[146]: Hoogervorst et al. (2017), *Wing aerostructural optimization using the Individual Discipline Feasible Architecture*

10: SU2 does not support SNOPT out of the box, but researchers have developed their own couplings to the optimizer.

[112]: Madsen (2020), *High-Fidelity CFD-based Shape Optimization of Wind Turbine Blades*

11: `sites.google.com/view/mcgill-computational-aerogroup/adodg`

[147]: Bisson et al. (2015), *Adjoint-based aerodynamic optimization of benchmark problems*

[148]: Anderson et al. (2015), *Aerodynamic shape optimization benchmarks with error control and automatic parameterization*

[132]: Gill et al. (2005), *SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization*

2.5.7 SNOPT: the optimizer of choice

SNOPT [132] is an sequential quadratic programming (SQP)-based optimizer designed for large-scale nonlinear constrained problems in the form given in equation 2.1, and where the gradients of $f(\boldsymbol{x})$ and $c(\boldsymbol{x})$ are readily available. In particular, it is effective when the problem has a certain sparsity structure,

and when the number of degrees of freedom at the optimum is moderate. It uses the active-set method to handle inequality constraints, an augmented Lagrangian merit function for line search, and maintains a quasi-Newton approximate Hessian via the BFGS update. As with most established optimizers, there are a large number of options available [149]. Here we discuss those that are relevant for the rest of the work.

[149]: Gill et al. (2007), *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*

2.5.7.1 Termination criteria

As with typical gradient-based optimizers, SNOPT uses two different criteria to determine the termination of the optimization, corresponding to residuals of the KKT conditions given by equations 2.6 to 2.8. Equations 2.7 and 2.8 are referred to as the *feasibility* criteria, and equation 2.6 the *optimality* criteria.

Definition 2.5.1 *For a candidate design $\boldsymbol{x}$, its feasibility, also known as primal infeasibility, is computed as*

$$\tau_{fea} \triangleq \max_i \frac{v_i(\boldsymbol{x})}{\|\boldsymbol{x}\|_2} \tag{2.24}$$

where $v_i(\boldsymbol{x})$ is the violation for constraint i, given as

$$v_i(\boldsymbol{x}) = \begin{cases} |c_i(\boldsymbol{x})|, & i \in \mathcal{E} \\ \max\{0, c_i(\boldsymbol{x})\}, & i \in \mathcal{I} \end{cases} \tag{2.25}$$

Definition 2.5.2 *For a candidate design $\boldsymbol{x}$, its optimality, also known as dual infeasibility, is computed as*

$$\tau_{opt} \triangleq \max_j \frac{\mathsf{Comp}_j}{\|\boldsymbol{\lambda}\|_2} \tag{2.26}$$

where the complimentarity slackness Comp_j is computed as

$$\mathsf{Comp}_j = \begin{cases} d_j \min\{x_j - l_j, 1\} & \text{if } d_j \geq 0 \\ -d_j \min\{u_j - x_j, 1\} & \text{if } < 0 \end{cases}$$

The quantity d_j is called the reduced gradient, and is computed as

$$d_j = \frac{\mathrm{d}f(\boldsymbol{x})}{\mathrm{d}x_j} + \lambda_j \frac{\mathrm{d}\boldsymbol{c}(\boldsymbol{x})}{\mathrm{d}x_j}$$

It is straightforward to verify that $\tau_{\text{fea}} = \tau_{\text{opt}} = 0$ iff the KKT conditions equations 2.6 to 2.8 are satisfied. In cases where the conditions are not satisfied, these tolerances give a measure of how far away $\boldsymbol{x}$ is from the optimal solution $\boldsymbol{x}^*$, and are therefore also referred to as optimization metrics. Compared to the KKT conditions, some additional scaling factors have been applied. The zeroth order conditions are scaled by the L_2 norm of the design vector $\boldsymbol{x}$, and the first-order condition is scaled by both the dimension of the design space as well as the L_2 norm of the Lagrange multipliers. This results in tolerances that are more comparable across problems of different dimensions, and where the magnitudes of either the function values or gradients are wildly different. Nevertheless, these tolerances are not invariant to problem scaling, and it is often more meaningful to discuss relative convergence, as was done in table 1.1.

2.5.7.2 Optimization restarts

Optimizations can be initialized in three ways. In a cold start, the optimization is initialized with no prior knowledge of the problem, so all the state variables are initialized to default values, except for the initial design variables provided by the user. This is the most common starting strategy taken. In a warm start, partial state information is provided by the user. This could be an initial Hessian approximation or a guess for the active set $\mathcal{A}$. In a hot start, the optimizer is initialized with the full state. For a deterministic optimizer, this means that we have all the information to exactly retrace an optimization.

We define optimizer states as variables that completely determine the state at each optimization iteration. Given the same

set of states, a deterministic optimizer will always produce the same design for the next iteration and follow the same path within the design space (within machine precision). For an SQP-based optimizer, these states can include Lagrange multipliers, the active set, the approximate Hessian, and many more. The exact make-up depends on the implementation of a particular optimizer.

As part of the appropriate fidelity work detailed in chapter 5, the developers of SNOPT implemented the hot start in addition to the cold and warm starts already available.

2.5.7.3 Hessian update strategy

SNOPT uses the BFGS Hessian update, with either a limited-memory or full-memory implementation [149]. The default behaviour in SNOPT is to use the full-memory Hessian when $n_x \leq 75$, and the limited-memory Hessian otherwise. In the limited-memory approach, the approximate Hessian matrix itself is not stored. Instead, only the diagonal is stored, along with the vector pairs in the rank-2 BFGS update. As long as the number of update vectors (*i.e.* the number of major iterations) is less than the dimensionality of the problem,[12] memory savings can be realized [74, Sec. 7.2].

[149]: Gill et al. (2007), *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*

12: Since the Hessian is SPD, a Cholesky factorization is used to further reduce the storage requirements of the full memory Hessian.

[74]: Nocedal et al. (2006), *Numerical Optimization*

However, in practice the number of vector pairs stored is much less than n_x, necessitating Hessian *resets*. In such cases, the vector pairs (and therefore the accumulated Hessian updates) are all discarded, leaving only the diagonal of the approximate Hessian. This will inevitably slow down the optimization since crucial information is being discarded, but the reduced memory requirement is sometimes important with large problems (say $n_x \sim \mathcal{O}(10^6)$).

In this work, since the problem size is moderate ($n_x \sim \mathcal{O}(10^3)$), we use the full memory Hessian in order to have easy access to

the approximate Hessian for post-optimization analyses. However, we still choose to periodically reset the Hessian on problems involving structural variables, as we have found that to be beneficial when the design space varies significantly between the initial and final designs.

Chapter 3

Sensitivity-Based Geometric Parameterization

3.1 Introduction to geometric parameterizations

There have been considerable advancements in the ASO of wings and aircraft, accelerated by improvements in the analysis and optimization algorithms. It is now routine to perform optimizations based on RANS equations involving hundreds of geometric design variables, subject to multiple flight conditions and a large number of geometric constraints.

To this end, several geometric parameterizations have been developed and used in ASOs. While some early efforts directly used mesh points as design variables, this approach is not practical for large-scale optimizations due to the large number of DOFs, resulting in smoothness issues. Instead, Hicks-Henne bump functions and Class-Shape function Transformation (CST) were developed for two-dimensional problems [150, 69]. For three-dimensional problems, the most popular approach is to use FFD [151]. Instead of parameterizing the geometry directly, FFD parameterizes the geometric *deformation*, which offers some significant benefits [152]. Several popular ASO suites such as SU2 [153], Jetstream [145, 154], and MACH [111] all use FFD for geometric parameterization.

In recent years, computed-aided design (CAD) has been directly incorporated into the optimization process. By directly parameterizing the underlying geometry, the final design can be modi-

[150]: Castonguay et al. (2007), *Effect of Shape Parameterization on Aerodynamic Shape Optimization*
[69]: Masters et al. (2017), *Geometric Comparison of Aerofoil Shape Parameterization Methods*

[151]: Sederberg et al. (1986), *Free-form Deformation of Solid Geometric Models*

[152]: Zhang et al. (2018), *A review of parametric approaches specific to aerodynamic design process*

[153]: Economon et al. (2016), *SU2: An Open-Source Suite for Multiphysics Simulation and Design*

[145]: Hicken et al. (2010), *Aerodynamic Optimization Algorithm with Integrated Geometry Parameterization and Mesh Movement*
[154]: Gagnon et al. (2015), *Two-Level Free-Form and Axial Deformation for Exploratory Aerodynamic Shape Optimization*

[111]: Kenway et al. (2010), *A CAD-Free Approach to High-Fidelity Aerostructural Optimization*

fied and manufactured without additional post-processing steps. Some popular CAD packages used in gradient-based optimization include Engineer Sketch Pad (ESP) [155] used by Brelje and Martins [104], and OpenVSP [156] used by Yildirim et al. [32].

[155]: Haimes et al. (2013), *The Engineering Sketch Pad: A Solid-Modeling, Feature-Based, Web-Enabled System for Building Parametric Geometry*

[104]: Brelje et al. (2021), *Aerostructural Wing Optimization for a Hydrogen Fuel Cell Aircraft*

[156]: McDonald et al. (2022), *Open Vehicle Sketch Pad: An Open Source Parametric Geometry and Analysis Tool for Conceptual Aircraft Design*

[32]: Yildirim et al. (2022), *Boundary Layer Ingestion Benefit for the STARC-ABL Concept*

However, optimization performances are degraded significantly by including complex geometric variables. For example, Lyu, Kenway, and Martins [79] performed an ASO on the CRM with 720 shape variables, and the initial optimization on a coarser mesh required over 600 iterations to converge. Even then, the best optimality achieved was only around 10^{-4}. Similar trends have also been observed in other works, such as those by Koo and Zingg [157], Streuber and Zingg [158], and Kedward, Allen, and Rendall [159]. This suggests that the scaling of the design variables could be improved or that the design variables themselves cause poor conditioning of the optimization problem.

[79]: Lyu et al. (2015), *Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark*

[157]: Koo et al. (2018), *Investigation into Aerodynamic Shape Optimization of Planar and Nonplanar Wings*

[158]: Streuber et al. (2021), *Dynamic Geometry Control for Robust Aerodynamic Shape Optimization*

[159]: Kedward et al. (2020), *Gradient-Limiting Shape Control for Efficient Aerodynamic Optimization*

To address this issue, a class of adaptive geometric parameterizations has emerged. Bons and Martins [86] employed a manual approach, where a sequence of optimizations is performed using successively refined FFD control volumes defined *a priori*. This way, a significant amount of geometric changes can be achieved with the smaller design space before switching to the denser FFD volume for finer geometric adjustments. A class of refinement approaches has been developed based on the same principle, fitting into two broad categories. On the one hand, progressive refinement is based on a sequence of progressively refined geometry parameterizations that are determined *a priori* [160, 148]. On the other hand, adaptive approaches refine the geometric parameterization during the optimization process [158, 161, 162]. By updating the definition of the geometric design variables during refinement, both computational speedup and superior optima were obtained compared to the progressive approach.

[86]: Bons et al. (2020), *Aerostructural Design Exploration of a Wing in Transonic Flow*

[160]: Masters et al. (2017), *Multilevel Subdivision Parameterization Scheme for Aerodynamic Shape Optimization*

[148]: Anderson et al. (2015), *Aerodynamic shape optimization benchmarks with error control and automatic parameterization*

[158]: Streuber et al. (2021), *Dynamic Geometry Control for Robust Aerodynamic Shape Optimization*

[161]: Han et al. (2014), *An adaptive geometry parametrization for aerodynamic shape optimization*

[162]: Anderson et al. (2015), *Adaptive Shape Control for Aerodynamic Design*

However, another challenge that remains unsolved is determining appropriate optimization scaling factors for geometric de-

sign variables [2, Tip 4.4]. While design variable scaling is a general problem in optimization, geometric design variables are of particular interest because they comprise most of the design variables in ASO. Furthermore, in MDO, such as aerostructural applications [27, 42], geometric variables affect both disciplines simultaneously and appear as dense sub-blocks in the constraint Jacobian. Historically, design variable scaling has been hand-tuned through trial and error, and the success and performance of an optimization can be heavily dependent on this process. Given that typical geometric parameterizations consist of groups of disparate variables responsible for local and global shape control, finding suitable scaling for all design variables is a significant challenge.

[2]: Martins et al. (2021), *Engineering Design Optimization*

[27]: Brooks et al. (2018), *Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings*

[42]: Kenway et al. (2014), *Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration*

This work aims to develop an approach that addresses numerical issues caused by these geometric design variables from an optimization perspective. We do not attempt to adaptively move or add geometric design variables but work with the given parameterization. In this way, we do not alter the optimization problem or the design space of interest. Instead, we use singular value decomposition (SVD) to compute design variable mapping and scaling such that the optimization problem becomes easier to solve while remaining mathematically identical.

While there are similarities with modal approaches that also use SVD [163, 164, 165, 166], there are some key differences. In the modal approach, SVD is applied to a library of candidate designs. For airfoil applications, the UIUC airfoil database [167] is often used, which is akin to a database of optimal airfoil designs. In addition, dimension reduction is often applied to reduce the design space to a handful of design variables. While effective for a range of applications, the modal approach requires some *a priori* knowledge of the design space through the database. It operates on a modal design space that is a subspace of the full geometric design space permitted by the FFD approach. As a result, the optimum found is often inferior to the full-space optimum but at a reduced cost. On the other hand, the proposed

[163]: Ghoman et al. (2012), *A POD-based Reduced Order Design Scheme for Shape Optimization of Air Vehicles*

[164]: Li et al. (2019), *Data-based Approach for Fast Airfoil Analysis and Optimization*

[165]: Li et al. (2021), *Adjoint-Free Aerodynamic Shape Optimization of the Common Research Model Wing*

[166]: Poole et al. (2022), *Efficient aeroelastic wing optimization through a compact aerofoil decomposition approach*

[167]: Selig (1996), *UIUC airfoil data site*

approach does not alter the design space. It should yield the same solution when using a gradient-based optimizer on a unimodal problem. Wing ASO problems have been shown to be largely unimodal [142, 3]

[142]: Yu et al. (2018), *On the Influence of Optimization Algorithm and Starting Design on Wing Aerodynamic Shape Optimization*
[3]: He et al. (2019), *Robust aerodynamic shape optimization—from a circle to an airfoil*

Kedward et al. [168] proposed an approach based on SVD that does not rely on a geometry library. The authors were initially motivated by the introduction of shape gradient constraints. They used SVD to convert those linear constraints into bound constraints and generate orthogonal design variables in the process. In contrast, this work does not require a set of linear inequality constraints of a particular form, instead operating on generic geometry parameterizations independent of the specific implementation or constraints present.

[168]: Kedward et al. (2022), *Generic Modal Design Variables for Efficient Aerodynamic Optimization*

There are also some notable similarities with the active subspace method [169, 170]. However, the active subspace method is concerned with a single scalar output, and the SVD is applied to a covariance matrix computed from its gradients sampled within the design space. In contrast, the proposed approach is purely geometric in nature. It is cheap to compute and does not require a large number of gradient evaluations. Furthermore, the active subspace method is a form of dimension reduction, where the geometric approach does not alter the dimensionality of the optimization problem.

[169]: Lukaczyk et al. (2014), *Active Subspaces for Shape Optimization*
[170]: Grey et al. (2018), *Active Subspaces of Airfoil Shape Parameterizations*

In the following sections, we outline the motivation behind our approach and explain the methodology itself. We then demonstrate the approach in an ASO problem, showing that the modified optimization problem results in improved optimizer performance.

3.2 Motivating analyses

3.2.1 Impact of geometric design variables

To further highlight the issue with geometric design variables, we perform two ASOs where we minimize the total drag of a wing under transonic flight conditions. We perform the two RANS-based optimizations starting from the same baseline design; they differ only in the number of geometric design variables and constraints considered in the optimization problem. In one case, we consider section twist variables, while additional shape variables are included in the other. These shape variables are allowed to move in the vertical direction individually, and are typically used to alter the sectional airfoil shape. The geometry and FFD box used are shown in figure 3.1, and the CFD simulations are performed on a medium-density mesh with 193 536 cells. The flight condition is at Mach 0.8 and an altitude of 10 000 m. We use SNOPT [132] as the optimizer and converge the optimization tightly by setting both the feasibility and optimality tolerances to 10^{-6}. We also ensure that the primal and adjoint solutions are converged tightly, using a relative convergence criteria of 10^{-14}. The optimization problem is detailed in table 3.1, where `T` refers to the twist-only optimization, and `T+S` is the case with twist and shape variables.

[132]: Gill et al. (2005), *SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization*

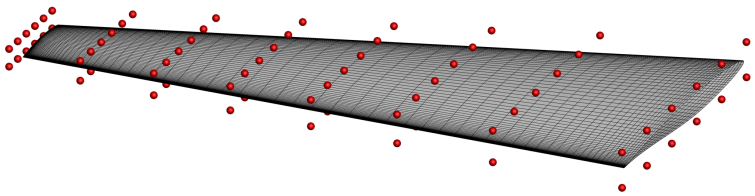


Figure 3.1: The standalone wing considered, together with the FFD box.

The optimization performance is shown in figure 3.2. As expected, case `T` converged quickly. In particular, once the approximate Hessian became sufficiently accurate, we observed the rapid convergence trends expected of quasi-Newton optimizations reminiscent of Newton's methods. The last four iterations

Table 3.1: Aso problem for twist design variables (T) and twist and shape design variables (T+S). "(L)" denotes that the constraint is linear.

	Function/variable	Description	Quantity T	Quantity T+S
minimize	C_D	Drag coefficient		
with respect to	x_{twist}	Section twist	7	7
	x_{shape}	Shape	—	96
	α	Angle of attack	1	1
		Total design variables	**8**	**104**
subject to	$C_L = 0.5$	Lift constraint	1	1
	$t \geq t_0$	Thickness constraint	—	100
	$V \geq V_0$	Volume constraint	—	1
	$\Delta z_{\text{LE, upper}} = -\Delta z_{\text{LE, lower}}$	Fixed leading edge constraint (L)	—	8
	$\Delta z_{\text{TE, upper}} = -\Delta z_{\text{TE, lower}}$	Fixed trailing edge constraint (L)	—	8
		Total constraints	**1**	**118**

decreased the optimality by about an order of magnitude per iteration, dropping from 3×10^{-4} to 4×10^{-7}.

On the other hand, the T+S case took significantly longer, mirroring the results from Lyu, Kenway, and Martins [79]. Although it is expected that an optimization with more design variables will take more iterations, it took over 300 iterations to reach the same target optimality. Beyond iteration 100, progress is painfully slow, with a slight improvement in either optimality or the merit function. The same behaviour was reported by Lee, Koo, and Zingg [171], where the twist-only optimization converged well but more complex problems had only one or two orders of reduction in optimality.

[79]: Lyu et al. (2015), *Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark*

[171]: Lee et al. (2017), *Comparison of B-Spline Surface and Free-Form Deformation Geometry Control for Aerodynamic Optimization*

3.2.2 Impact of orthogonality

Given the stark contrast in performance between the two optimizations, we suspect that part of the issue lies in the geometric parameterization, where certain geometric design variables are not *orthogonal* to other design variables. This means that some subsets of geometric design variables, while linearly independent, are similar. Changing them would affect the outputs similarly—the embedded aerodynamic surface mesh nodes in this case. This

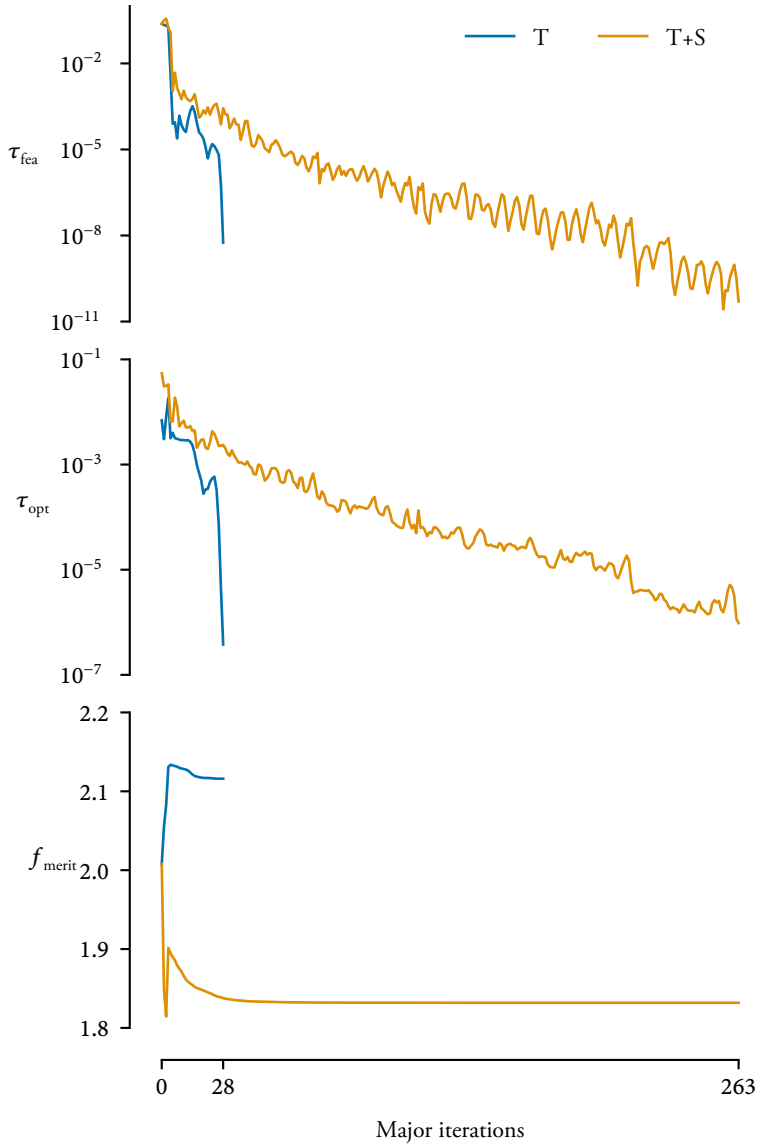


Figure 3.2: Comparison of ASOs with and without shape variables, showing optimization metrics. The horizontal lines show the optimization termination criteria, set at 10^{-6} for both feasibility and optimality.

would harm the optimization because it leads to poor conditioning for the optimization problem.

The earlier work by Lyu, Kenway, and Martins [79] examined the effect of changing the number of FFD nodes in both spanwise and chordwise directions. They found that the addition of nodes in the spanwise direction had little effect on the convergence rate of the optimization. However, there is a significant impact when adding nodes in the chordwise direction. The authors concluded that "the coupled effects between design variables are much stronger between variables within an airfoil than between

[79]: Lyu et al. (2015), *Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark*

variables in different airfoils".

To investigate this further, we analyze the T+S optimization problem performed earlier. We first compute the geometric Jacobian of surface sensitivities

$$\mathbf{J}_{\text{geo}} \triangleq \frac{\mathrm{d}\boldsymbol{X}_A}{\mathrm{d}\boldsymbol{x}_{\text{geo}}}, \tag{3.1}$$

where $\boldsymbol{X}_A$ is the flattened 1-D vector of surface mesh coordinates corresponding to the surface mesh of the wing used in CFD, and $\boldsymbol{x}_{\text{geo}}$ are the geometric design variables. Each column of this Jacobian matrix represents the sensitivity of all surface mesh coordinates with respect to one geometric design variable. We examine the geometric sensitivity of surface mesh coordinates because they are the points embedded in the FFD. The volume mesh coordinates are then deformed by propagating these surface deformations using an inverse-distance approach, which was described in section 2.5.

From this, we can compute the angle θ_{ij} between pairs of Jacobian columns $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ using equation 3.2, and look for pairs of design variables that result in near parallel or antiparallel gradients, that is, angles near 0° or 180°.

$$\theta_{ij} = \arccos\left(\frac{\boldsymbol{v}_i^{\mathsf{T}}\boldsymbol{v}_j}{|\boldsymbol{v}_i| \cdot |\boldsymbol{v}_j|}\right), \quad i \neq j \tag{3.2}$$

where $\boldsymbol{v}_i$ is the i-th column of the Jacobian $\mathbf{J}$.

The distribution of angles is shown in figure 3.3. Although most design variable pairs are close to being orthogonal, some design variables are close to being "parallel" to each other, with the smallest angle at just 3.4°. A further check revealed that these correspond to two local shape variables that control the pair of upper and lower FFD nodes at the TE of the wing tip, shown in figure 3.4. The top 18 pairs of non-orthogonal design variables correspond to upper and lower FFD nodes at the same planform location along the wing, with the ones most parallel

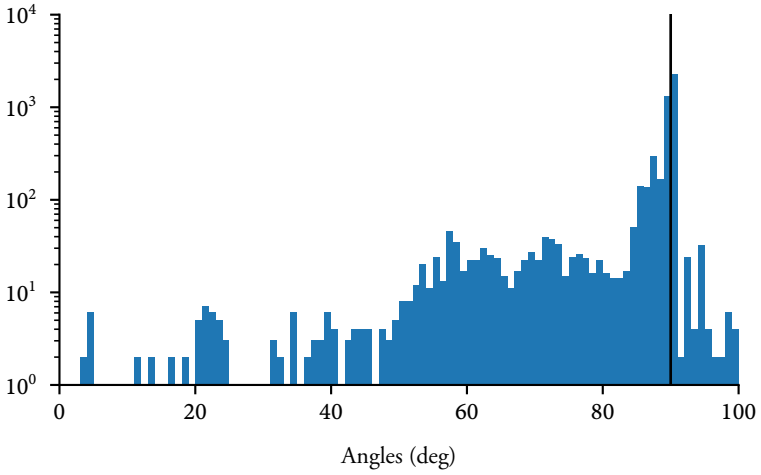


Figure 3.3: Distributions of angles between pairwise geometric design variable gradients for the T+S case. The y-axis uses a logarithmic scale, and the vertical line denotes 90°.

at the TE locations. This is expected, as these design variables are restricted to move only in the vertical direction. Moving the upper node would have a similar effect to moving the lower node, effectively displacing surface nodes near the FFD node in the vertical direction. These pairs of nodes are also more likely to be located at the TE because, at those locations, more of the surface mesh nodes are roughly equidistant to both the upper and lower FFD nodes. As a result, the surface sensitivities from either node are roughly equal.

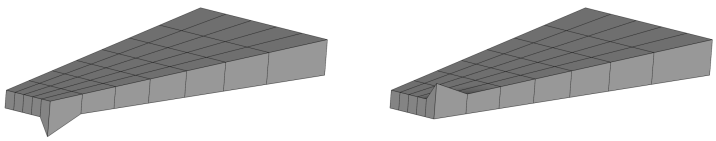


Figure 3.4: Shape variables 6 and 7, which have the smallest gradient angle at just 3.4°.

The distribution of design variable angles is in sharp contrast to that from the T optimization, shown in figure 3.5. In this case, the angles are much more concentrated around 90°, with the smallest angle at 51°.

In ASO, we employ a set of leading-edge and trailing-edge constraints that precisely link these pairs of design variables at the leading and trailing edges to prevent those nodes from emulating section twist via shear deformations. These are listed in table 3.1, effectively eliminating one degree of freedom (DOF) in each pair. Unfortunately, some pairs of design variables in the

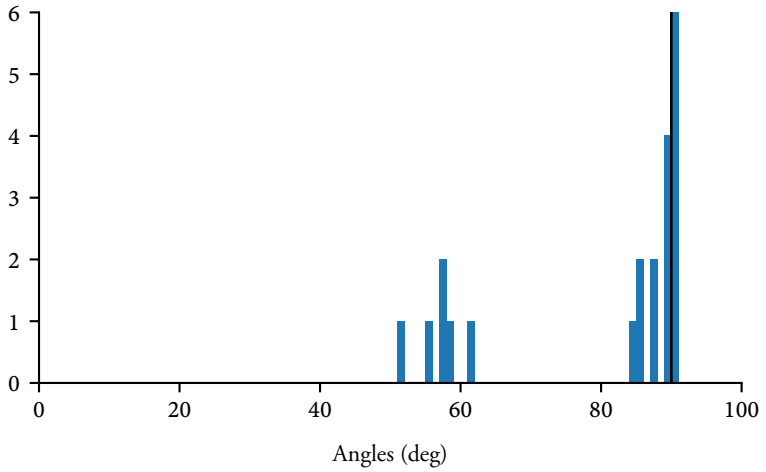


Figure 3.5: Distribution of angles between pairwise geometric design variable gradients involving only twist. The y-axis uses a logarithmic scale, and the vertical line denotes 90°.

wing interior are still rather non-orthogonal. Out of these, the most non-orthogonal pair is shown in figure 3.6. Lastly, there are also non-orthogonalities between specific shape and twist design variables; an example is shown in figure 3.7.

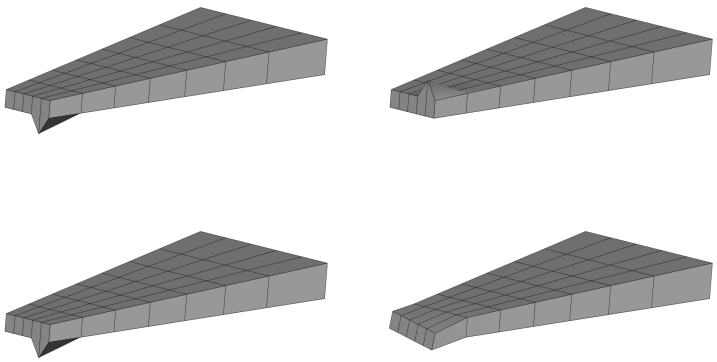


Figure 3.6: Shape variables 19 and 23, which have a gradient angle of 12°.

Figure 3.7: Shape variable 19 and twist variable 6, which have a gradient angle of 22°.

We then cross-check these potentially problematic pairs of design variables against their optimization progress in figure 3.8. Since these variables are at the leading or trailing edge, the linear constraint ensures that the pairs have equal and opposite values because SNOPT satisfies all linear constraints at each iteration. However, these design variables have noticeably slower convergence than the rest, which typically arrive close to their final value within the first 100 iterations. Instead, these are still changing after 200 iterations, and these variables are likely partly responsible for the slow overall convergence rate.

On the other hand, figure 3.9 shows the optimization history

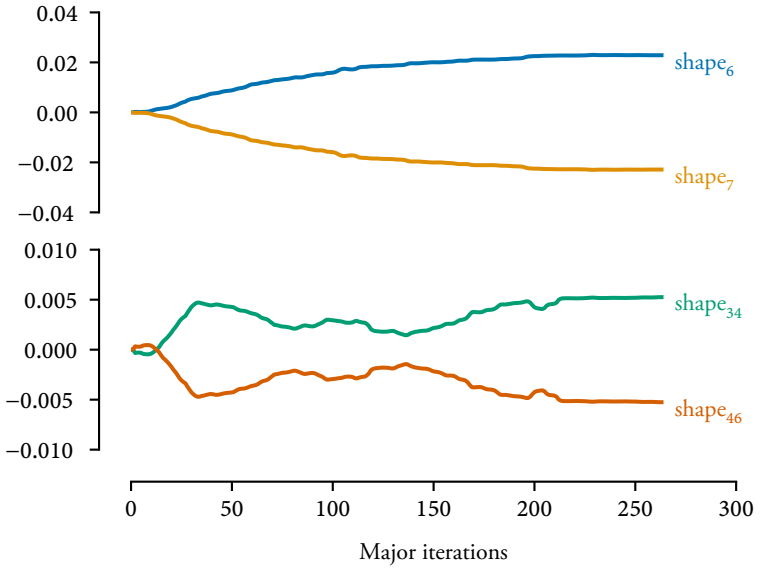


Figure 3.8: Optimization history for some shape variables at the leading and trailing edges.

for an orthogonal pair of design variables.

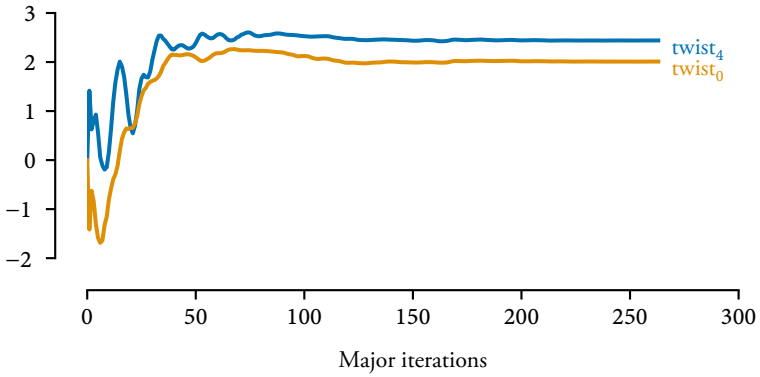


Figure 3.9: Optimization history for a pair of orthogonal design variables.

Although the linear constraints remove some DOFs from the design space, these design variables are still challenging for the optimization. To illustrate, suppose that we have two variables, x_1 and x_2, that have similar gradients, that is,

$$\frac{\partial \mathcal{I}}{\partial x_1} \approx \frac{\partial \mathcal{I}}{\partial x_2}$$

for some objective $\mathcal{I}(x_1, x_2)$. Then, suppose we add a linear constraint $x_1 + x_2 = 0$, meaning the two variables must be opposite of each other. This creates two opposing effects that render the design space relatively flat. If moving along x_1 decreases the objective by $\delta\mathcal{I}$, the corresponding increase in x_2 required by the linear constraint would increase the objective by a similar

amount. As a result, the sensitivity within the feasible space is close to zero, and the optimizer will have trouble finding the optimum. Fundamentally, the issue is caused by the fact that these design variables are not orthogonal to each other, resulting in similar gradients that worsen the optimization problem conditioning.

Mathematically, we can show this by introducing a variable ξ that is used to parameterize the linear constraint:

$$x_1 = \xi, \quad x_2 = -\xi.$$

Then, the sensitivity along the constraint becomes

$$\begin{aligned} \frac{\mathrm{d}\mathcal{I}}{\mathrm{d}\xi} &= \frac{\partial \mathcal{I}}{\partial x_1}\frac{\mathrm{d}x_1}{\mathrm{d}\xi} + \frac{\partial \mathcal{I}}{\partial x_2}\frac{\mathrm{d}x_2}{\mathrm{d}\xi} \\ &= \frac{\partial \mathcal{I}}{\partial x_1} - \frac{\partial \mathcal{I}}{\partial x_2} \\ &\approx 0. \end{aligned}$$

Optimization histories for some design variables that are not linearly constrained are shown in figure 3.10. Even though no linear constraints exist, the fact that the gradients are similar means that the optimizer moves them in either equal or opposite directions, resulting in slow convergence.

Finally, we revisit the pairwise angles shown in figure 3.3. For each pair of geometric design variables $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, we compute the deviation from orthogonality as

$$\delta_{ij} = \left|90° - \theta_{ij}\right|, \tag{3.3}$$

where θ_{ij} is computed from equation 3.2.

By definition, $\delta_{ij} = 0$ iff $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are orthogonal, while $\delta_{ij} = 90°$ if $i = j$. We plot δ_{ij} in figure 3.11. If all the design variables are orthogonal to each other, this should look like an identity matrix, with all off-diagonal entries equal to 0°. Unfortunately,

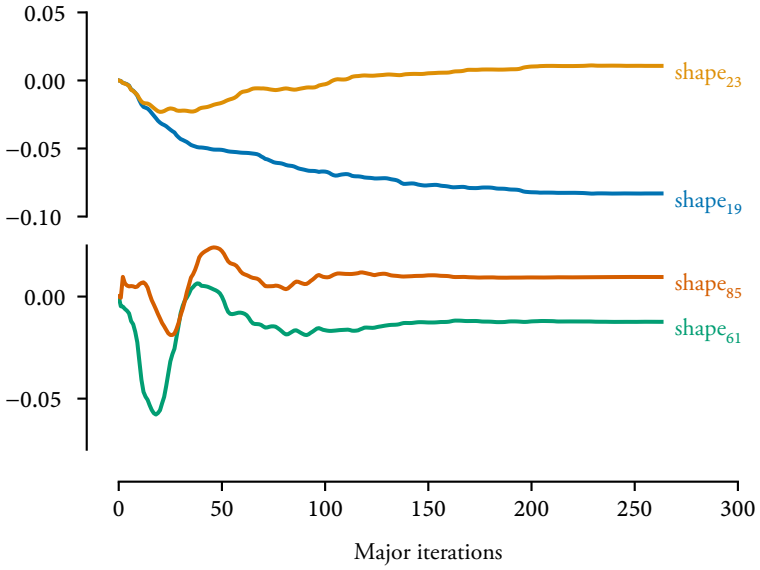


Figure 3.10: Optimization history for some shape variables at the wing interior.

this is not the case, and many pairs of design variables are not orthogonal to each other.

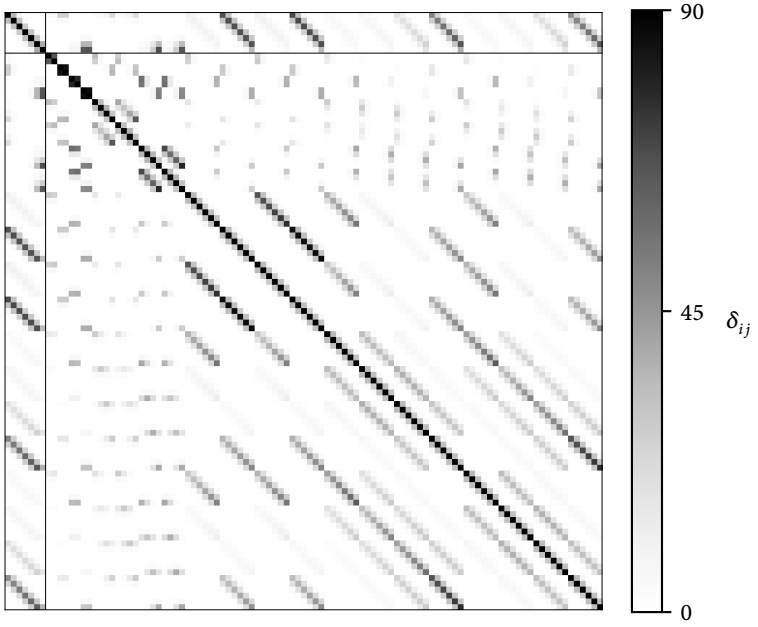


Figure 3.11: A contour plot of δ_{ij}. The diagonal entries are 90° by definition, and any nonzero entries in the off-diagonal indicate design variable pairs that are not orthogonal.

3.3 Generating design variables

We aim to map the existing geometric design variables $\boldsymbol{x}_{\text{geo}}$ into alternative variables more suitable for optimization. We do this through a linear mapping of the form

$$\hat{\boldsymbol{x}}_{\text{geo}} = \mathbf{A}\boldsymbol{x}_{\text{geo}}, \tag{3.4}$$

where $\boldsymbol{x}_{\text{geo}}$ are the original design variables, and $\hat{\boldsymbol{x}}_{\text{geo}}$ the new design variables. For the remainder of this section, we omit the subscript "geo" with the implicit understanding that we are only performing this linear mapping on geometric design variables. The full optimization problem will likely have other design variables that remain unchanged. This is expanded further in 3.3.2

3.3.1 Methodology

This matrix $\mathbf{A}$ is chosen such that the surface sensitivity Jacobian $\hat{\mathbf{J}}$ in the new design space, computed as

$$\hat{\mathbf{J}} \triangleq \frac{\mathrm{d}\boldsymbol{X}_A}{\mathrm{d}\hat{\boldsymbol{x}}}, \tag{3.5}$$

has orthogonal columns. This means that for two distinct design variables x_i and x_j, the dot product of the gradients should be zero:

$$\frac{\mathrm{d}\boldsymbol{X}_A}{\mathrm{d}\hat{\boldsymbol{x}}_i}^{\mathsf{T}} \frac{\mathrm{d}\boldsymbol{X}_A}{\mathrm{d}\hat{\boldsymbol{x}}_j} = 0 \quad \forall i \neq j.$$

To orthogonalize the design variables, we perform SVD on the geometric Jacobian $\mathbf{J}$. We can now rewrite the Jacobian as a multiplication of three matrices,

$$\mathbf{J} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathsf{T}},$$

where $\mathbf{U}$ and $\mathbf{V}$ are the left and right singular vectors, and $\boldsymbol{\Sigma}$ is a diagonal matrix of the singular values. By construction, $\mathbf{V}$ is

an orthonormal matrix, and we use its inverse as the mapping matrix as follows:

$$\mathbf{A} = \mathbf{V}^{\mathsf{T}}. \tag{3.6}$$

The proposed methodology is a purely geometric approach based on the geometric sensitivities of embedded points and therefore is parameterization-independent. We implement and demonstrate this methodology using FFD, but the same could be done for other parameterizations as long as the geometric Jacobian $\mathbf{J}_{\text{geo}}$ is available. However, the approach does not fully capture the nonlinearity present in the optimization problem. For example, from the perspective of the optimizer, the objective sensitivity (in this case, the drag coefficient C_D) is given by

$$\frac{\mathrm{d}C_D}{\mathrm{d}\boldsymbol{x}} = \frac{\mathrm{d}C_D}{\mathrm{d}\boldsymbol{X}_A}\frac{\mathrm{d}\boldsymbol{X}_A}{\mathrm{d}\boldsymbol{x}},$$

and simply orthogonalizing the second term will not capture the effect of the CFD analysis. However, it should still offer some benefits for the optimization problem, especially since geometric design variables are often numerous and affect many optimization quantities simultaneously. This is true for multipoint problems where the same geometry is analyzed at different flight conditions and for multidisciplinary problems where multiple disciplines share the same geometry. The proposed approach can be easily extended to aerostructural problems by considering the embedded structural points in $\boldsymbol{X}_A$ when computing the SVD.

Because the method is based on a linearization of the geometric parameterization, it only orthogonalizes the design variables in the neighborhood of the design point about which SVD was computed. The orthogonalization may be less effective for more significant geometric design changes because the geometric Jacobian changes with the design variables. As a result, it may be useful to perform this orthogonalization multiple times during optimization to capture this nonlinearity better.

3.3.2 Reformulating the Optimization

Once the mapping matrix $\mathbf{A}$ has been computed, the optimization problem has to be re-written in these new design variables $\hat{\boldsymbol{x}}_{\text{geo}}$. This involves mapping four parts of the optimization problem:

- design variables
- derivatives of objectives and nonlinear constraints
- linear constraints
- design variable bounds

Additionally, since the mapping is done on the geometric design variables $\hat{\boldsymbol{x}}_{\text{geo}}$ only, care must be taken to use the correct indices such that the rest of the design variables remain intact. For example, the full design vector could look like the following:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ x_{i+1} \\ \vdots \\ x_n \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \boldsymbol{x}_{\text{geo}} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \boldsymbol{x}_{\text{other}} \end{matrix}$$

3.3.2.1 Design Variables

Firstly, the design variable mapping is straightforward since we already constructed the mapping:

$$\hat{\boldsymbol{x}}_{\text{geo}} = \mathbf{A}\boldsymbol{x}_{\text{geo}} \tag{3.7}$$

$$\boldsymbol{x}_{\text{geo}} = \mathbf{A}^{-1}\hat{\boldsymbol{x}}_{\text{geo}} \tag{3.8}$$

Due to the use of SVD to determine the mapping matrix $\mathbf{A}$, it is orthonormal by construction. This means that, conveniently, its inverse is just its transpose:

$$\mathbf{A}^{-1} = \mathbf{A}^{\mathsf{T}} \tag{3.9}$$

which is handy since we do not have to compute its inverse separately.

3.3.2.2 Nonlinear Gradient and Jacobian

Next, mapping the derivatives of nonlinear outputs are discussed.

First, in the case of the objective, we have a single scalar output f, and its gradient is given by:

$$\begin{aligned}\frac{\mathrm{d}f}{\mathrm{d}\hat{\boldsymbol{x}}_{\text{geo}}} &= \frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}_{\text{geo}}}\frac{\mathrm{d}\boldsymbol{x}_{\text{geo}}}{\mathrm{d}\hat{\boldsymbol{x}}_{\text{geo}}} \\ &= \frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}_{\text{geo}}}\mathbf{A}^{-1}\end{aligned}$$

where $\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}_{\text{geo}}}$ is the original gradient.

For a vector of constraints $\boldsymbol{c}$, their sensitivities with respect to the new design variables are:

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{c}}{\mathrm{d}\hat{\boldsymbol{x}}_{\text{geo}}} &= \frac{\mathrm{d}\boldsymbol{c}}{\mathrm{d}\boldsymbol{x}_{\text{geo}}}\frac{\mathrm{d}\boldsymbol{x}_{\text{geo}}}{\mathrm{d}\hat{\boldsymbol{x}}_{\text{geo}}} \\ &= \mathbf{J}\mathbf{A}^{-1}\end{aligned}$$

where $\mathbf{J}$ is the original nonlinear Jacobian.

Again, care must be taken to map the Jacobian only for the design variables of interest, i.e. geometric variables. In reality, the full Jacobian is actually composed of several smaller sub-blocks:

$$\mathbf{J} = \begin{bmatrix}\mathbf{J}_{gg} & \mathbf{0} \\ \mathbf{J}_{og} & \mathbf{J}_{oo}\end{bmatrix}$$

where $\mathbf{J}_{gg}$ is the Jacobian of geometric constraints with respect to geometric design variables, $\mathbf{J}_{og}$ the Jacobian of other constraints with respect to geometric design variables, and $\mathbf{J}_{oo}$ the Jacobian of other constraints with respect to other design variables. By

construction, $\mathbf{J}_{go}$, the Jacobian of geometric constraints with respect to other design variables, is zero. For example, the volume constraint is not affected by the angle of attack.

In our case, we need to map the geometric sub-blocks $\mathbf{J}_{gg}$ and $\mathbf{J}_{og}$, while leaving the rest untouched. This is done by right-multiplying those blocks by $\mathbf{A}^{-1}$.

$$\hat{\mathbf{J}} = \begin{bmatrix} \hat{\mathbf{J}}_{gg} & \mathbf{0} \\ \hat{\mathbf{J}}_{og} & \mathbf{J}_{oo} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{gg}\mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{J}_{og}\mathbf{A}^{-1} & \mathbf{J}_{oo} \end{bmatrix}$$

3.3.2.3 Linear Jacobian

The linear constraints are handled separately since they are never evaluated during the optimization process. Instead, the bounds and the linear Jacobian is passed directly to the optimizer. SNOPT [132] handles linear constraints of the form

[132]: Gill et al. (2005), *SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization*

$$\boldsymbol{h}_L \leq \mathbf{J}_{\text{lin}}\boldsymbol{x} \leq \boldsymbol{h}_U \tag{3.10}$$

where $\mathbf{J}_{\text{lin}}$ is the linear constraint Jacobian.

Luckily, the linear constraints which involve geometric constraints are only related to geometric constraints. In the new design space, we have:

$$\boldsymbol{h}_L \leq \mathbf{J}_{\text{lin}}\mathbf{A}^{-1}\hat{\boldsymbol{x}}_{\text{geo}} \leq \boldsymbol{h}_U \tag{3.11}$$

3.3.2.4 Design variable bounds

Lastly, the design variable bounds require some modification to the optimization problem itself. The bounds in the original space $\boldsymbol{x}$ are usually given as

$$\boldsymbol{x}_L \leq \boldsymbol{x} \leq \boldsymbol{x}_U. \tag{3.12}$$

However, in the new design space, these bound constraints become linear inequality constraints:

$$\boldsymbol{x}_{L,g} \leq \mathbf{A}^{-1}\hat{\boldsymbol{x}}_{\text{geo}} \leq \boldsymbol{x}_{U,g}. \tag{3.13}$$

Because the matrix $\mathbf{A}$ is orthonormal, geometrically, the design space mapping corresponds to a rotation and possibly a reflection. The original bound constraints form a rectangular feasible space, but after rotation, the boundary is no longer aligned with the coordinate axes of the new design space, as shown in figures 3.12 and 3.13. Therefore, they cannot be expressed as design variable bounds, and must be replaced by explicit linear constraints with $\mathbf{A}^{-1}$ as the linear Jacobian. We only do this for the geometric design variables, and the rest of the design variable bounds are unchanged. While this technically increases the dimension of the optimization problem by introducing additional constraints, it should not cause any difficulties for gradient-based optimizers because the constraints are linear.

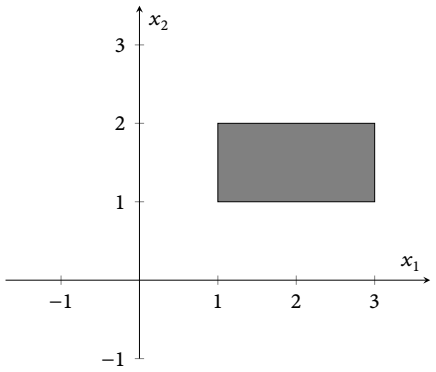


Figure 3.12: Original design space bounds.

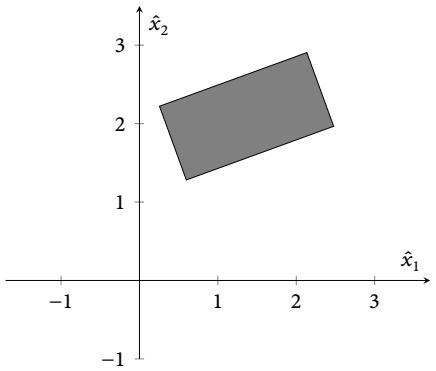


Figure 3.13: The same bounds in the new design space.

3.3.2.5 Alternative Implementation

Another way to look at these mappings is by working with the global design vector $\boldsymbol{x}$ and Jacobian $\mathbf{J}$ instead of partitioning into sub-blocks. We can construct a global mapping matrix $\tilde{\mathbf{A}}$:

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{3.14}$$

corresponding to the design variable ordering which we assume to be

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{\text{geo}} \\ \boldsymbol{x}_o \end{bmatrix}$$

Then, we can apply the matrix transformation to the full design

variable vectors and Jacobians:

$$\hat{\boldsymbol{x}} = \tilde{\mathbf{A}}\boldsymbol{x}$$
$$\hat{\mathbf{J}} = \mathbf{J}\tilde{\mathbf{A}}^{-1}$$

which is the same result as above.

Note that the code implementation uses the former approach for ease of implementation and better bookkeeping of the various indices.

3.3.3 Verification

As before, we compute the angles θ_{ij} between pairs of columns of $\hat{\mathbf{J}}$ using equation 3.2, but in the new design space $\hat{\boldsymbol{x}}$. Recall that in the original design space $\boldsymbol{x}$, the worst alignment has an angle of $\theta_{ij} = 3.4°$. Through the SVD, we expect all the gradient vectors to be orthogonal to each other. Therefore we compute the maximum deviation from orthogonality, computed as

$$\delta_{\max} = \max_{i \neq j} \delta_{ij}, \tag{3.15}$$

where δ_{ij} is computed from equation 3.3.

As shown in table 3.2, the linear transformation yields an orthogonal Jacobian, up to machine precision. This means that the matrix δ_{ij} as computed by equation 3.3 would be an identity matrix in the new design space, with 90° along the diagonal and zero elsewhere.

Table 3.2: Verification of the orthogonality of geometric sensitivities.

	$\delta_{\max}$ (deg)
$\boldsymbol{x}$	86.58
$\hat{\boldsymbol{x}}$	2.48×10^{-11}

3.3.4 Generated design variables

Finally, we show the first eight design variables in figure 3.14, sorted by decreasing singular values. These variables are typical of modal decompositional approaches, where the design variables

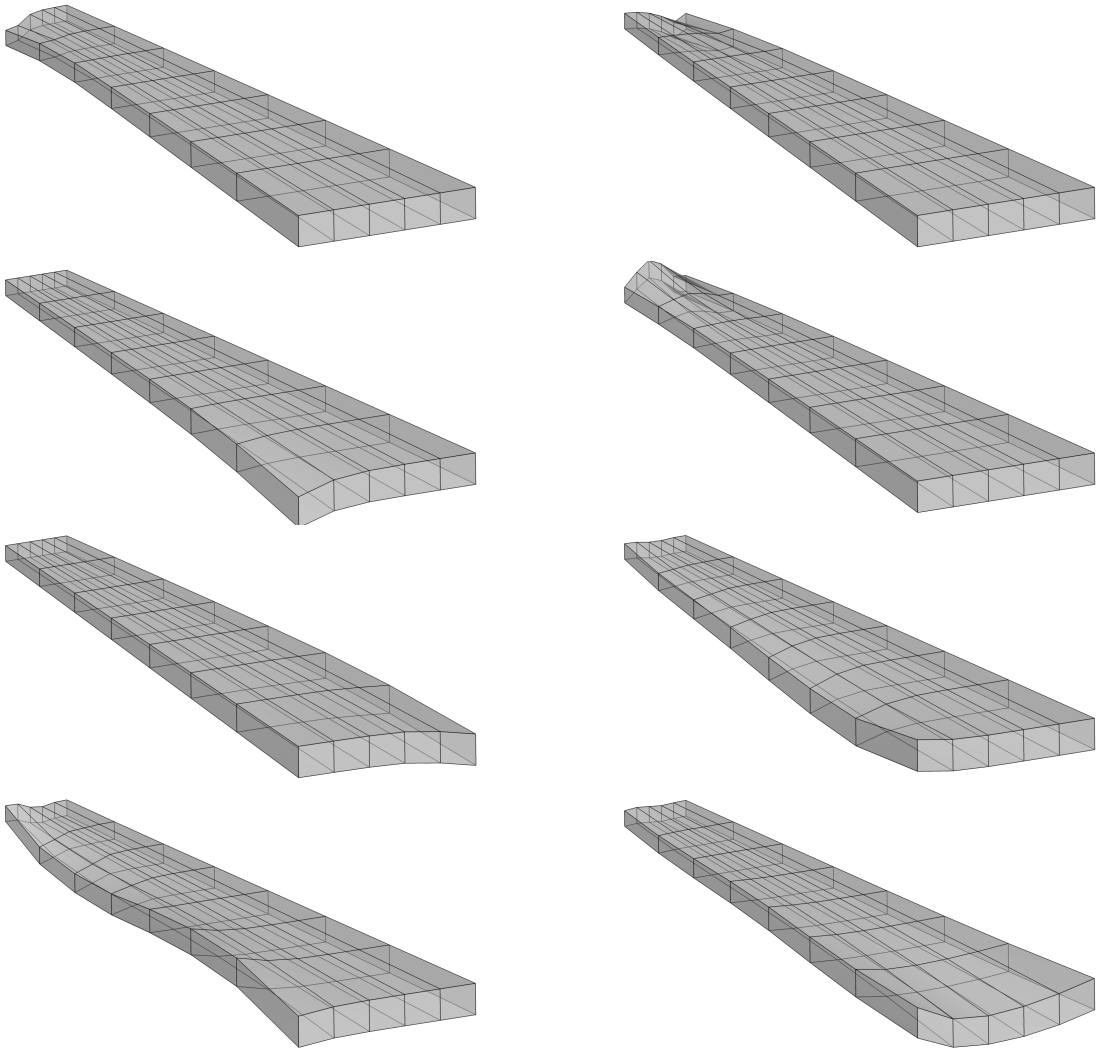


Figure 3.14: The first eight newly constructed design variables.

corresponding to larger singular values have lower frequency modes. Some design variables are recognizable; for example, the third and fifth design variables show root camber deformations. However, later design variables are less discernable, which is expected in any decompositional approach. Since the transformation is linear, we can always map the design variables back and forth with relative ease, and users can monitor optimization progress using more physically-intuitive design variables.

3.4 Design variable scaling

In gradient-based optimization, design variable scaling is an important aspect that can significantly impact the performance of nonlinear optimizers [2, Tip. 4.4]. Variable scaling refers to using multipliers s_i that are applied to each design variable, such that the optimizer operates in the scaled design space where $x_i' = s_i x_i$. This is mathematically equivalent to a diagonal preconditioner applied to the entire optimization problem. A common strategy is to scale the design variables so that their bounds are in the $[0, 1]$ range. Physical intuitions can also be used to deduce, for example, that the scaling for the angles of attack should be the same as for the spanwise twist variables on the wing because they have the same units and roughly the same impact on the flow solution.

[2]: Martins et al. (2021), *Engineering Design Optimization*

However, there is no general approach to scaling design variables. Scaling design variables to be within the unit hypercube is not always reliable, and physical intuition cannot be applied across disparate design groups, such as shape or sweep, where their impact on the optimization is complex. Users must rely largely on trial and error to produce well-behaved optimizations, which can consume a significant amount of time.

3.4.1 Methodology

When using design variable mapping, neither approach can be used to scale the design variables as they no longer have bounds or physical meaning. However, with the additional insight provided by the SVD, we can use that information to automatically determine the appropriate scaling for these design variables. In particular, the singular values σ_i corresponding to each geometric design variable x_i are a natural choice. For a given unit of change, the design variables with larger singular values have a more significant impact on the surface mesh coordinates and are thus more sensitive. The scaling factors are only applied to

geometric design variables where there is sensitivity information. The remainder of the design variables are left unscaled, and the same factor is used for all optimizations.

We seek a scaling function $s(\sigma)$ that gives the appropriate scaling for each design variable based on its singular value. Logically, we would like to reduce the sensitivity of more impactful design variables and vice versa. This would ensure that all design variables have similar sensitivities and converge equally quickly. From a numerical optimization perspective, this should also ensure a better-conditioned Hessian with similar curvatures along different axes [71].

[71]: Gill et al. (1981), *Practical Optimization*

In this work, we try four choices for s_i: 1, σ_i, $\sqrt{\sigma_i}$, and $1/\sigma_i$. Because a larger scaling value *decreases* the sensitivity of a design variable, we expect monotonically increasing choices for $s(\sigma)$, such as $s = \sigma$ or $s = \sqrt{\sigma}$, to perform well.

Together with the design variable mapping, this approach can be considered mathematically as a preconditioning step for the optimization problem. Together with design variable mapping, this can make the Hessian of the Lagrangian more diagonally dominant, and the diagonal elements may be of similar magnitudes. In such cases, the initial Hessian approximation from the optimizer can be more accurate, leading to more rapid optimization convergence.

3.4.2 Impact of mesh density

The singular vectors and singular values change depending on the mesh density. As the mesh density increases, the geometric Jacobian $\mathbf{J}_{\text{geo}}$ will contain more entries, resulting in larger singular values. However, the overall trends of the singular values and singular vectors should be similar across different mesh densities. To study this effect in more detail, we perform the same SVD process on a family of meshes with `R2` as the coarsest mesh and `R0` as the finest mesh. We also introduce a mesh `E1` with a

slightly different TE to study the impact of different underlying surface definition.

The top of figure 3.15 shows the distribution of singular values σ_i for each mesh density. As expected, the finest grid, `R0`, has much larger singular values than the rest. The lower figure of figure 3.15 shows the normalized singular values $\tilde{\sigma}$, computed as

$$\tilde{\sigma}_i = \frac{\sigma_i}{\sum_j \sigma_j},$$

such that $\sum_i \tilde{\sigma}_i = 1$.

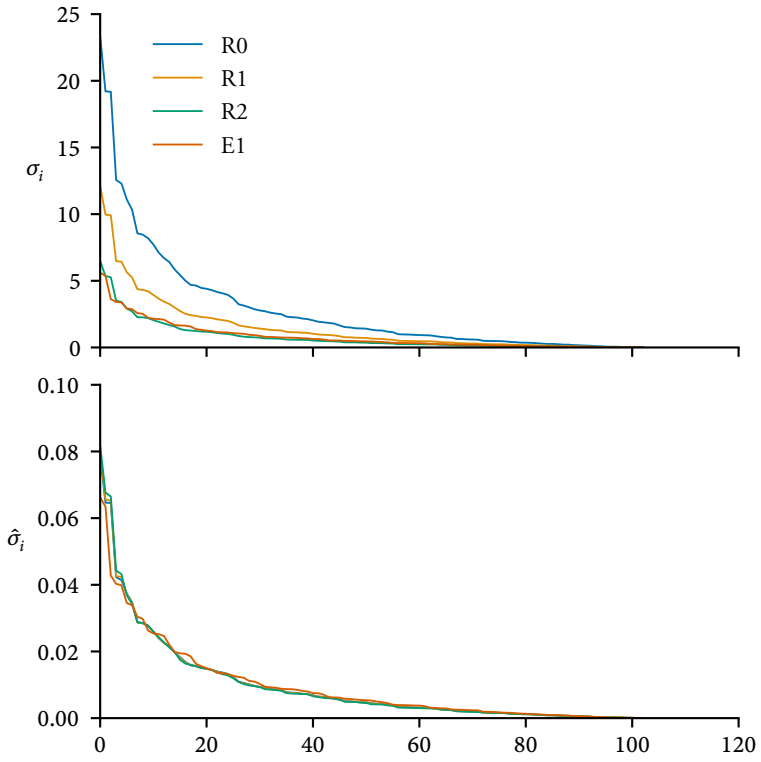


Figure 3.15: Comparison of unscaled and scaled singular values computed from several different grids.

The distributions across the mesh family remain near-identical, except for the mesh `E1`. This is somewhat expected given the different trailing edge geometry, and the effect is only present for the first few singular values. Therefore, after the SVD, we normalize all singular values before applying the scaling function $s(\sigma)$.

3.5 Optimization results

We solve two optimization problems: the T+S problem presented above, and one with an additional span variable which we call T+S+S. The optimizations involving span have an extra constraint such that the projected planform area of the wing must be greater than or equal to the initial planform area.

We perform five optimizations for each of these optimization problems to compare the two parameterizations and explore the different scaling options. We first perform a reference optimization using the existing parameterization and four optimizations using the sensitivity-based parameterization with various design variable scaling.

For this to be a fair comparison, we converge all CFD primal and adjoint solutions tightly, using the tolerances $\tau_{\text{pr}} = \tau_{\text{adj}} = 10^{-14}$. This way, we ensure that we can reach the target feasibility and optimality tolerances without the optimizer wasting extra evaluations due to inaccurate function values or gradients. We also set the SNOPT feasibility and optimality tolerances to 10^{-6}. The only difference between these optimizations is the design variable mapping and scaling, which affects the optimizer's performance. This way, we can check first if the four scaling options converge to the same reference optimum and subsequently analyze their performance. Table 3.3 lists the different optimization problems, showing that the new parameterization has the same number of design variables, and extra linear constraints are in place to replace the bound constraints in the original design space.

3.5.1 Twist and shape

Table 3.4 shows the outcomes of those optimizations. As explained above, the mapping does not modify the optimization problem in any way, and we would expect all the optimizations to arrive at the same optimum.

Table 3.3: Optimization problem statements highlighting the differences in problem size between the original and new parameterizations.

	Function/variable	Description	Quantity (T+S) Original	Quantity (T+S) New	Quantity (T+S+S) Original	Quantity (T+S+S) New
minimize	C_D	Drag coefficient	1	1	1	1
with respect to	x_{twist}	Section twist	7	0	7	0
	x_{shape}	Shape	96	0	96	0
	x_{span}	Span	—	—	1	0
	x_{user}	User	0	103	0	104
	α	Angle of attack	1	1	1	1
		Total design variables	**104**	**104**	**105**	**105**
subject to	$C_L = 0.5$	Lift constraint	1	1	1	1
	$t \geq t_0$	Thickness constraint	100	100	100	100
	$A \geq A_0$	Area constraint	—	—	1	1
	$V \geq V_0$	Volume constraint	1	1	1	1
	$\Delta z_{\text{LE,upper}} = -\Delta z_{\text{LE,lower}}$	Fixed leading edge constraint (L)	8	8	8	8
	$\Delta z_{\text{TE,upper}} = -\Delta z_{\text{TE,lower}}$	Fixed trailing edge constraint (L)	8	8	8	8
	$x_L \leq x \leq x_U$	Design variable bounds (L)	0	103	0	104
		Total constraints	**118**	**221**	**119**	**223**

Table 3.4: Summary of optimization results for the T+S problem.

Scaling	Objective (counts)	Major itns	Feasibility	Optimality	Time (hr)	η (%)
Reference	183.188	263	5.0×10^{-11}	9.6×10^{-7}	2.61	—
$s = 1$	183.231	1000	9.1×10^{-9}	6.4×10^{-4}	9.94	–281
$s = \sigma$	183.188	309	9.7×10^{-12}	4.7×10^{-7}	3.27	–26
$s = \sqrt{\sigma}$	183.188	284	1.5×10^{-11}	9.8×10^{-7}	2.83	–8
$s = 1/\sigma$	183.198	577	6.4×10^{-13}	8.3×10^{-7}	5.98	–129

However, this was not the case. While most optimizations converged to the same optimum as the reference case without geometric mapping, the cases with $s = 1$ and $s = 1/\sigma$ did not. Although feasible, these two optimizations took significantly more iterations and terminated with worse objective values. In the case of $s = 1$, the optimization took the maximum allowable major iterations of 1000 and terminated before reaching the final optimality tolerance of 10^{-6}. This is somewhat consistent with our expectations, as those two scaling options did not reduce the sensitivity of more impactful design variables. This highlights the importance of design variable scaling, which can significantly impact the optimization performance.

The performance of the different optimizations are characterized by the *computational speedup*:

Definition 3.5.1 *The speedup, η, for an optimization is computed relative to a reference optimization as*

$$\eta = 1 - \frac{c}{c_{ref}} \tag{3.16}$$

where c is the computational cost of the optimization, and c_{ref} is the cost for the reference optimization.

Out of the three optimizations that converged to the same optimum, design variable mapping did not offer a clear advantage over the existing parameterization. The best-performing scaling option, $s = \sqrt{\sigma}$, took 21 more major iterations and 8% longer wall time to converge. However, it is worth pointing out that the reference optimization was performed with design variable scaling tuned through prior experience. Achieving comparable performance is still advantageous because it reduces the cost of setup and tuning, which is not included in the comparison above.

Figure 3.16 shows a few design variables throughout the optimization. Although the shape and twist variables were not used directly, we can easily apply the inverse mapping to obtain their values throughout the optimization. These plots clearly show that the two optimizations with $s = 1$ and $s = 1/\sigma$ did not converge. Their design variable values stayed near the initial design for much of the optimization, and the final value did not match the reference optimum. For the remaining three optimizations, some variables converged more rapidly than the reference, while others exhibited more oscillations.

The optimization metrics are shown in figure 3.17. In addition to the previously-mentioned metrics, we also show the merit function, which is an augmented Lagrangian composed of three terms: the objective, the constraints weighted by the Lagrange multipliers, and a quadratic penalty term on the constraint violation [132]. This metric helps gauge optimization progress, as it combines the objective value with the constraints. Again,

[132]: Gill et al. (2005), *SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization*

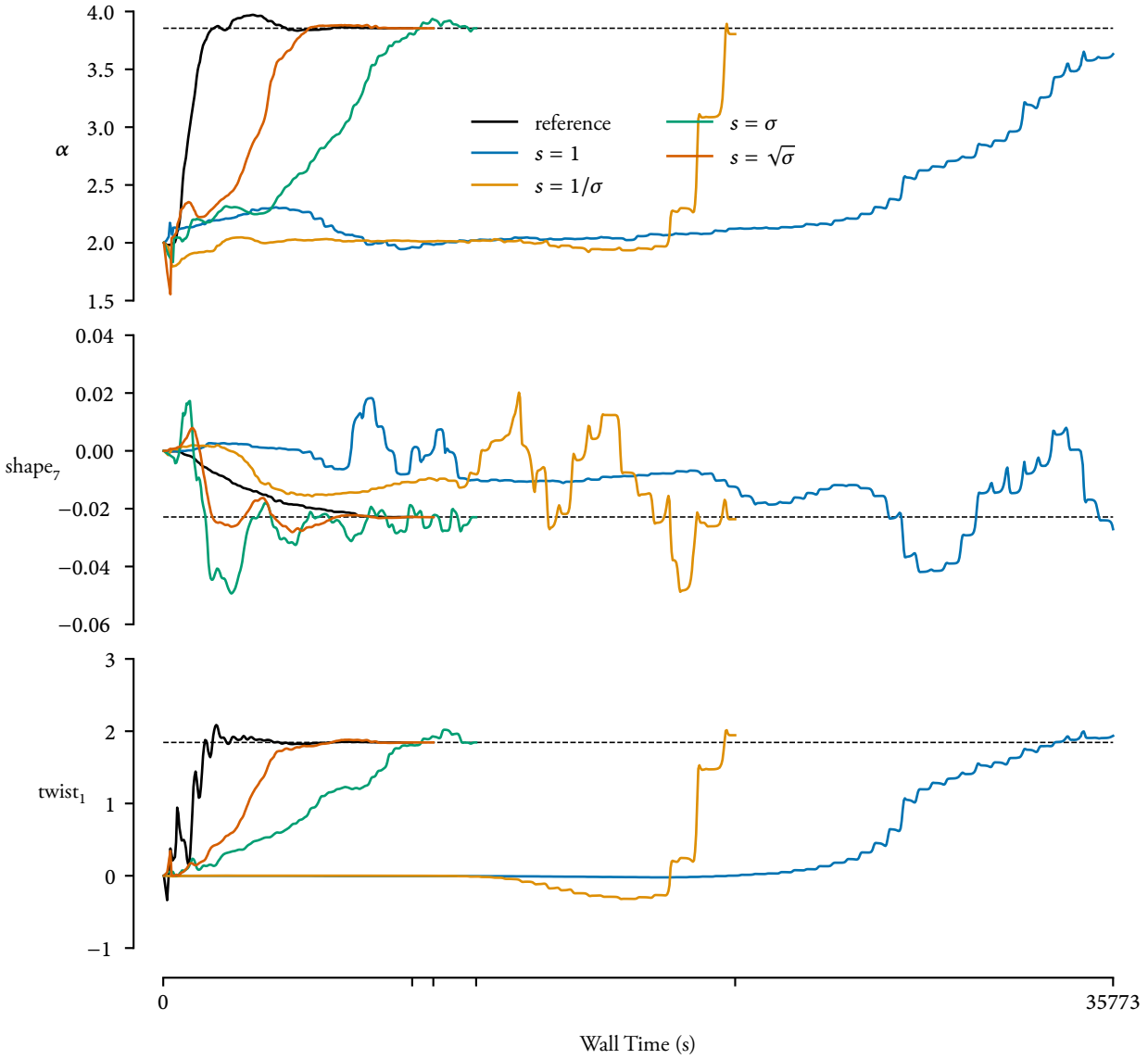


Figure 3.16: The design variable history for each optimization. The dashed line corresponds to the final value obtained from the reference optimization without any design variable mapping.

we can see that the two unconverged optimizations were much slower at reducing the merit function than the others. They also had large oscillations for feasibility and optimality, indicating difficulties in obtaining an accurate approximate Hessian.

On the other hand, the optimization with $s = \sigma$ converged rapidly at the end of its optimization, reminiscent of the twist-only case presented earlier. After close to 300 iterations with relatively little change to optimality, it decreased it from 10^{-3} to 10^{-6} in 6 iterations. This rapid convergence rate is an indication

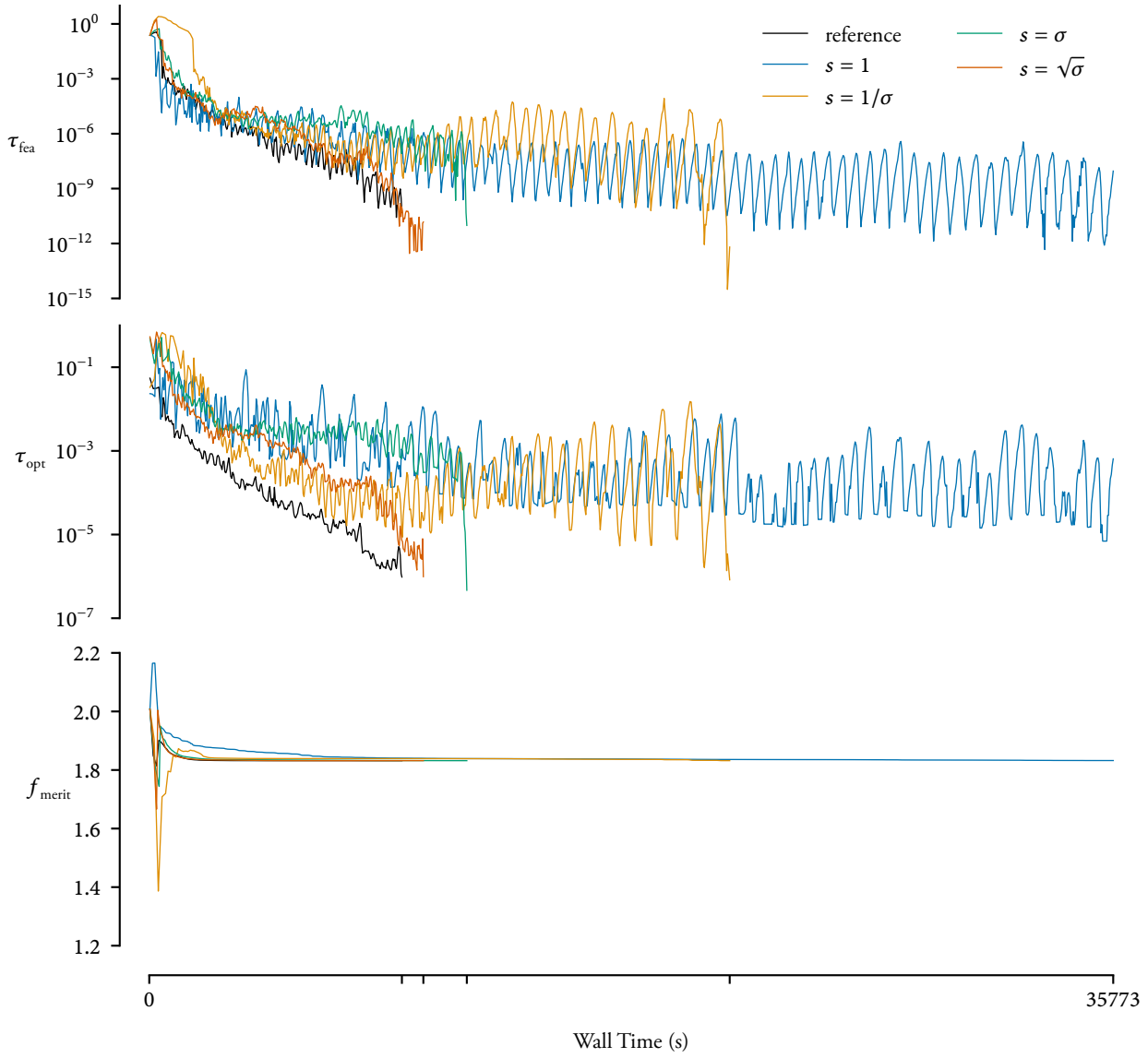


Figure 3.17: The optimization metrics for the T+S case. The thin horizontal black line indicates the termination criteria of 10^{-6} for feasibility and optimality.

that it was able to accumulate a sufficiently-accurate approximate Hessian to be within the quadratic convergence region, likely due to a combination of orthogonal design variables and suitable scaling. Despite taking longer than the reference optimization, the rapid terminal convergence means that for a tighter termination criterion, it could conceivably obtain an optimum in fewer iterations.

Lastly, we extract the approximate Hessian at the final iteration and show them in figure 3.18. We ensure that the Hessian is

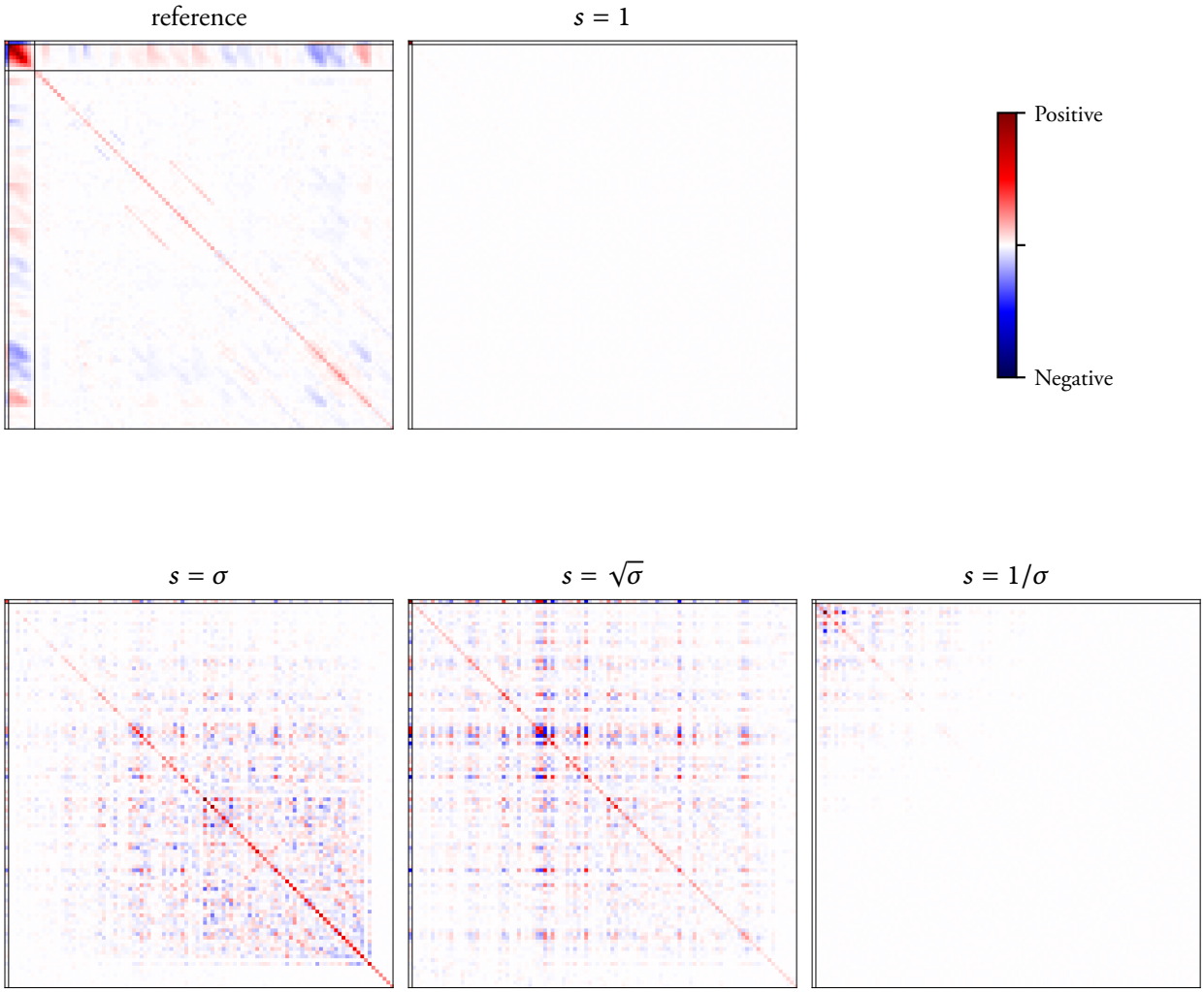


Figure 3.18: The approximate Hessian at the final optimization iteration. While a symmetric diverging colormap is used, the maximum value is adjusted for each plot. Therefore, entries with the same color do not have the same magnitude across different subplots.

never reset during the optimization, either periodically or when the optimizer encounters numerical difficulties so that these plots are representative of the design space as perceived by the optimizer. In addition, the optimizations presented here took a large number of major iterations, on the order of several times the number of design variables. Therefore, they represent a reasonable approximation to the Hessian of the Lagrangian and give us an indication of the curvature of the design space.

In these figures, the horizontal and vertical thin black lines are used to show different design variable groups. For the reference optimization, the first row and column correspond to the angle of attack variable, followed by the seven twist variables. The remaining entries are the shape variables. For the other optimizations, the design variable mapping groups all geometric design variables into a single sub-block, and the only other variable is α.

The structure of the approximate Hessian is visible in the reference optimization, with the twist variables having the largest entries, particularly along the diagonal. Significant coupling across design variables can also be seen in the structure of the off-diagonal terms. Many of these off-diagonal entries are precisely the design variable pairs with near-parallel alignment, not just between the shape variables but also between twist variables. This is demonstrated by comparing figures 3.11 and 3.18, where the design variables with poor alignment match the same off-diagonal structure of the approximate Hessian very well. This provides further evidence of the hypothesis in section 3.2.1 that by orthogonalizing the geometric design variables based purely on their sensitivity, we can obtain a design space better suited for gradient-based optimization.

When mapping is applied, the structured pattern in the Hessian is eliminated, and different scaling options affect the magnitudes of the entries. For the two optimizations that did not converge, it is clear that the approximate Hessian was far from accurate. For the remaining optimizations, the magnitudes of the diagonal terms are informative.

In the ideal scenario, a well-scaled optimization should have entries of similar magnitudes along the diagonal, indicating similar curvatures across design variables. With the scale option $s = \sigma$, there is an over-emphasis on the later design variables, whereas $s = \sqrt{\sigma}$ had a more uniform distribution. This could be why this particular scaling performed the closest to the reference

Table 3.5: Summary of optimization results for the T+S+S problem.

Scaling	Objective (counts)	Major itns	Feasibility	Optimality	Time (hr)	η (%)
Reference	171.924	373	8.0×10^{-12}	9.5×10^{-7}	3.75	—
$s = 1$	171.934	1000	2.4×10^{-12}	3.8×10^{-6}	10.30	–176
$s = \sigma$	171.924	353	2.7×10^{-15}	7.8×10^{-7}	4.05	–8
$s = \sqrt{\sigma}$	171.924	314	1.9×10^{-14}	8.6×10^{-7}	3.49	7
$s = 1/\sigma$	171.936	547	1.7×10^{-14}	7.0×10^{-7}	5.81	–55

optimization.

3.5.2 Span

Similar trends were observed for the optimization involving twist, shape, and span variables. The optimization results are summarized in table 3.5.

The design variable histories are shown in figure 3.19, and the optimization metrics are shown in figure 3.20. As before, the scalings $s = 1$ and $s = 1/\sigma$ did not converge, and $s = \sqrt{\sigma}$ performed the best, beating the reference optimization by 7%. Interestingly, both $s = \sigma$ and $s = \sqrt{\sigma}$ took fewer major iterations to converge than the reference optimization, and the rapid terminal convergence stage was observed again in figure 3.20.

We know from the earlier analysis that at the initial design, the reference parameterization has a deviation from orthogonality of 86.6°, and that after orthogonalization the deviation is zero. However, it was not clear whether these values would persist throughout the optimization. Therefore, at each major iteration, we compute the maximum deviation across all pairs of design variables:

$$\max_{ij} \delta_{ij}.$$

These values are plotted for the reference and the $\sqrt{\sigma}$ optimizations, and shown in figure 3.21.

Inevitably, the δ_{ij} worsened for the orthogonal parameterization over the course of the optimization. Since the parameterization

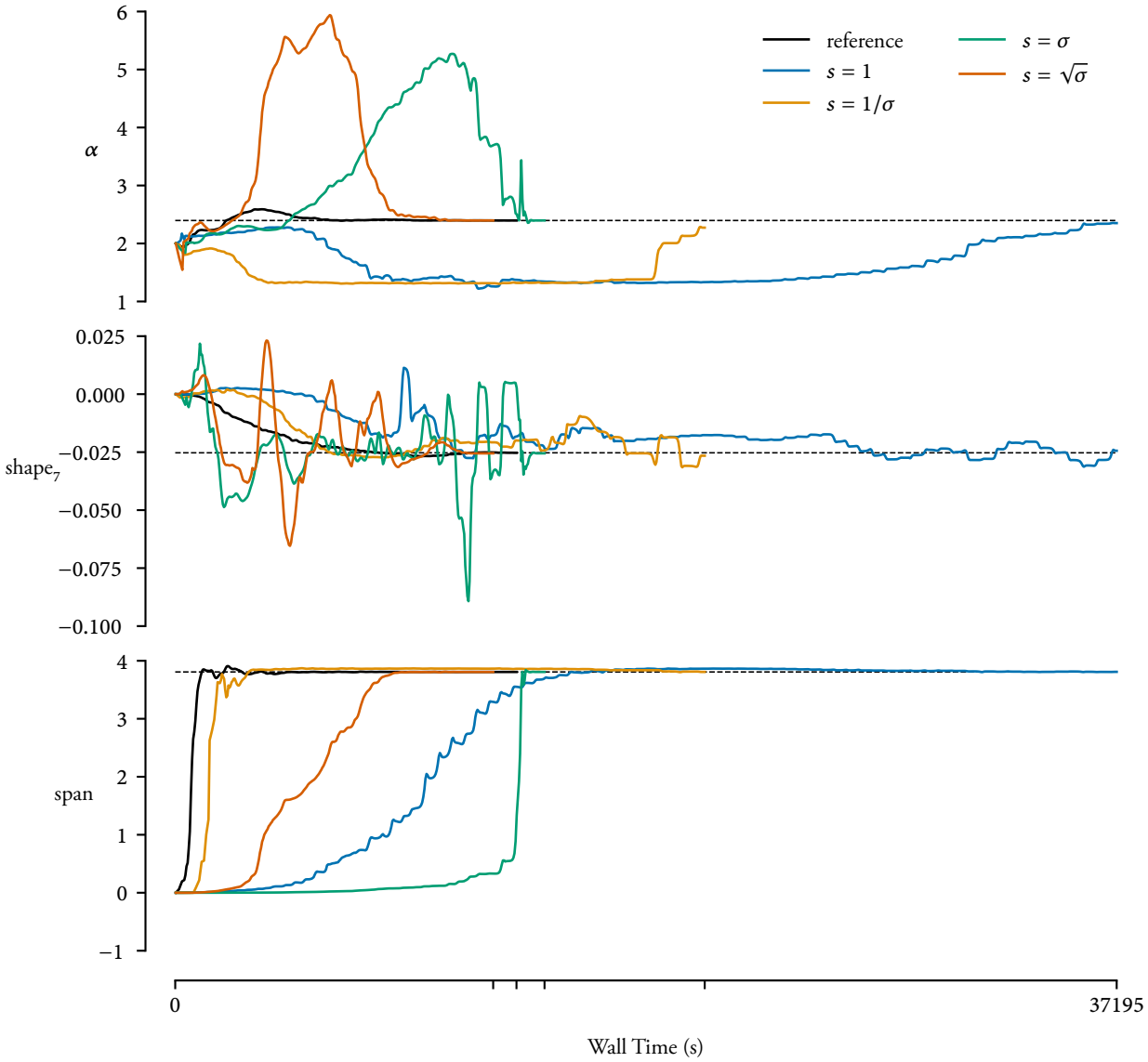


Figure 3.19: The design variable history for the T+S+S case.

was computed about the initial point, any nonlinear geometric deformation will introduce non-orthogonality. However, at the optimum, the maximum deviation was only 15°, indicating that for this particular optimization problem, the parameterization was likely valid throughout and no re-orthogonalization was necessary. Of course, for an optimization with larger deformations, the same may not be true. On the other hand, the reference parameterization stayed near constant. This is due to the fact that this metric is the maximum deviation, taken over all possible design variable pairs—numbering over 5400. From the histogram

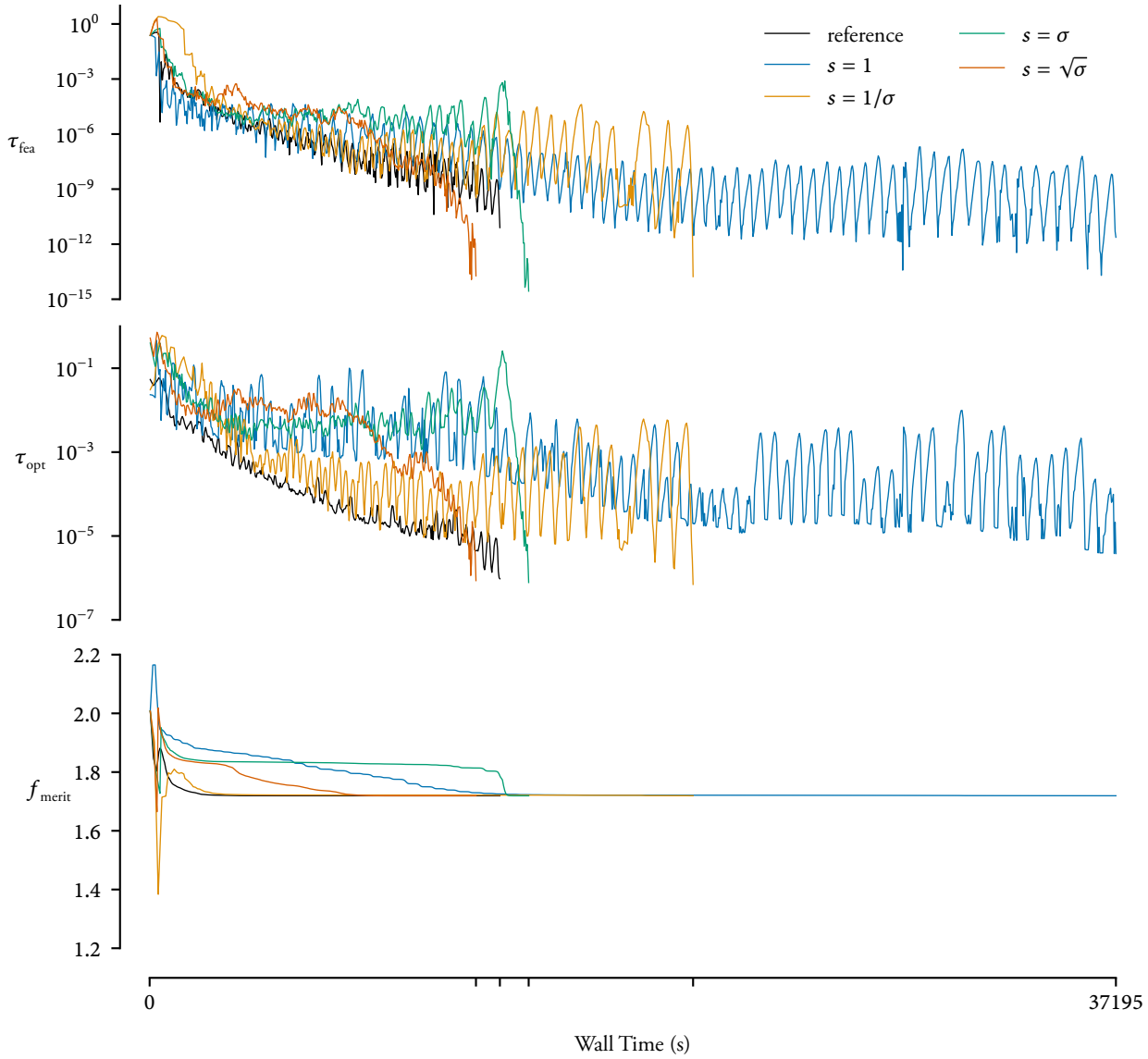


Figure 3.20: The optimization metrics for the T+S+S case.

shown in figure 3.3, the culprit was a single pair of design variables identified in figure 3.4. With small geometric changes in the region, the amount of deviation was not altered, and the same value remained throughout the optimization, negatively impacting the optimization at every iteration.

The final approximate Hessians are shown in figure 3.22, and once again, the $s = \sqrt{\sigma}$ case had a more uniform distribution of entries along the diagonal. In retrospect, this outcome makes

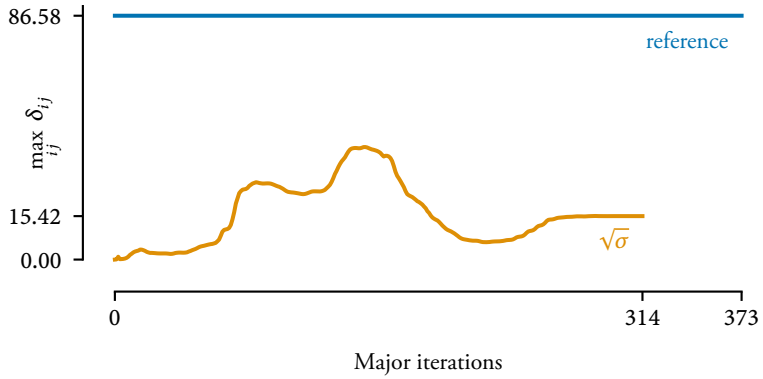


Figure 3.21: The maximum deviation from orthogonality for the reference and $\sqrt{\sigma}$ optimizations.

sense. When we scale variables by a factor s,

$$\hat{x} = sx.$$

This causes the derivatives to be scaled by the same factor s. Since the Hessian is the matrix of second derivatives, this causes the entries of the Hessian to be scaled by s^2. If the aim is to scale those Hessian entries by σ, then scaling the design variables by their square root would be the logical choice.

The differences in performance between these four identical optimizations show the importance of design variable scaling. Together with the construction of the new design variables, we demonstrate that despite examining only the geometric properties of the optimization problem, we can improve the performance of the optimization.

3.6 Summary

In this chapter, we showed that the existing geometric design variables based on intuitive parameters are not orthogonal to each other. In some cases, a design variable affects the embedded surface mesh in a similar way to another, which could lead to issues with the optimizer. To address this, we constructed a new set of design variables based on an SVD of the geometric Jacobian, such that the Jacobian in the new design space is orthogonal.

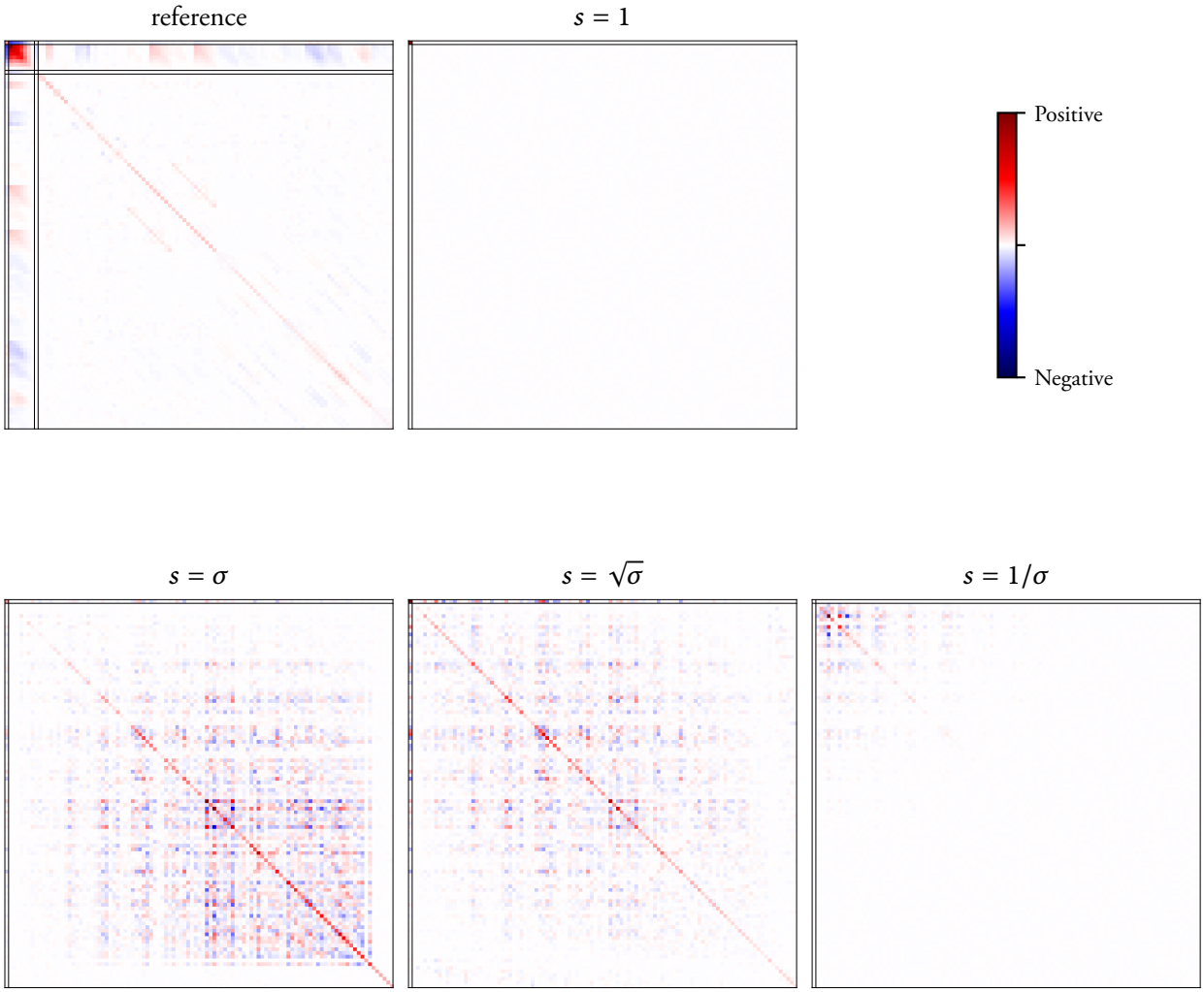


Figure 3.22: The approximate Hessian at the final iteration for the T+S+S case. The additional span variable is visible in the reference optimization at the ninth index, highlighted by the thin vertical and horizontal lines.

At the same time, we use the singular values from the SVD to automatically scale these constructed design variables.

We then perform two sets of ASOs, first with twist and shape variables and later including span. For each optimization problem, we compared the original parameterization against the newly developed parameterization with various scaling approaches. We found that with the appropriate design variable scaling, the proposed methodology improved the optimization convergence

while converging to the same optimum. Unlike the reference optimization, we eliminated the long optimization tail and recovered the rapid terminal convergence expected of quasi-Newton methods. Although the overall computational costs were comparable to the reference optimization, the automatic scaling eliminates the need for manual tuning, which is a significant advantage over the traditional approach.

In addition, the proposed methodology is not limited to aerodynamic optimization problems. It can be easily extended to aerostructural problems where the same geometric parameterization describes both the aerodynamic and structural mesh points. The automatic scaling approach could also be more effective when the parameterization lacks an initial physically informed scaling, such as when using CST parameterizations.

Chapter 4

Adaptive Convergence Error Control

4.1 Background

In many optimizations, the objective and constraint functions are computed from an expensive, iterative solution process, as discussed in section 2.2. One approach to reduce the computational cost is to only partially converge the solution and adjoint equations during optimization, via the tolerances defined in equations 4.2 and 4.3. By loosely converging the solution initially, and gradually tightening the tolerance as the optimizer moves towards the optimum, significant cost savings could be realized. This is conceptually not dissimilar from the NLBGS algorithm proposed by Kenway, Kennedy, and Martins [21], or the Eisenstat–Walker algorithm [172] for choosing the linear residual. The convergence level can also be thought of as a continuously-varying and user-adjustable level of fidelity, in the context of multifidelity optimization [173].

[21]: Kenway et al. (2014), *Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Adjoint Derivative Computations*

[172]: Eisenstat et al. (1996), *Choosing the Forcing Terms in an Inexact Newton Method*

[173]: Peherstorfer et al. (2018), *Multifidelity Monte Carlo estimation for large-scale uncertainty propagation*

However, the solution and gradient accuracy will impact the optimization convergence. Without careful consideration of the adaptive approach, it is possible to converge to an incorrect *apparent* optimum, or not converge at all. Several papers have covered the development of *inexact* SQP algorithms with guaranteed convergence [174, 175], where the accuracy requirements of the function and gradients are determined by the optimizer, and the errors are managed in such a way that convergence properties are retained. While mathematically rigorous, these approaches require the modification of the optimization algorithm itself,

[174]: Heinkenschloss et al. (2001), *Analysis of inexact trust-region SQP algorithms, SIAM Journal on Optimization*

[175]: Gu et al. (2017), *A new inexact SQP algorithm for nonlinear systems of mixed equalities and inequalities*

which is not applicable to typical engineering applications where the solvers and optimizer are treated as independent black boxes with only input parameters such as convergence tolerances to adjust.

In general, the development of any adaptive error control requires two key ingredients. First, an error estimation must be available for the function value and possibly the gradient, based on the current solver convergence tolerances. Second, an adaptation algorithm will modify the solver tolerances based on the current accuracy requirements of the optimizer, such that the solvers are converged to the necessary tolerances but no further.

Adjoint-based error estimation is a relatively well-established field, where the adjoint is used to estimate the discretization error in CFD simulations [94]. Based on this error estimation, the computational mesh can then be adapted to reduce discretization error, via either p or h-refinement. This approach has been successfully employed in a number of CFD solvers such as FUN3D [176] and Cart3D [177].

[94]: Fidkowski et al. (2011), *Review of output-based error estimation and mesh adaptation in computational fluid dynamics*

[176]: Biedron et al. (2019), *FUN3D Manual: 13.6*

[177]: Nemec et al. (2008), *Adjoint-Based Adaptive Mesh Refinement for Complex Geometries*

It is natural to extend this approach to convergence error, which has been done in the past by Lu and Darmofal [178] and Lozano and Ruiz Juretschke [179], and which we have independently derived in section 4.2. Lu and Darmofal [178] further attempted to combine this together with discretization error control, to adaptively change both the solution tolerance and the mesh during optimization. While the authors showed some computational gains of their methodology, the application was fairly basic and cannot be applied directly to the present problems of interest. Furthermore, they developed a concurrent primal-dual solution strategy in order to perform error estimation on-the-fly, something that is infeasible with the current computational framework.

[178]: Lu et al. (2004), *Adaptive Precision Methodology for Flow Optimization via Discretization and Iteration Error Control*

[179]: Lozano et al. (2009), *Adjoint-Based Correction of Non-Converged CFD Solutions*

[178]: Lu et al. (2004), *Adaptive Precision Methodology for Flow Optimization via Discretization and Iteration Error Control*

To date, Brown and Nadarajah [180, 181] were the most successful at tackling this topic. They developed error estimates and

[180]: Brown et al. (2017), *Inexactly constrained discrete adjoint approach for steepest descent-based optimization algorithms*

[181]: Brown et al. (2021), *Effect of inexact adjoint solutions on the discrete-adjoint approach to gradient-based optimization*

tolerance adaptation based on the convergence of the primal system,[1] and then extended it to the adjoint system in the subsequent paper. These are rigorous mathematical developments governing the adaptation algorithm such that optimization convergence is guaranteed. However, a number of simplifications prevented direct adoption of this approach. First, a steepest-descent algorithm without line search is assumed, in order to arrive at the guaranteed convergence properties. It is well known that such optimization algorithms converge linearly, and are significantly slower than more advanced algorithms such as SQP that converge superlinearly [74, Sec. 18.7]. Second, an unconstrained optimization problem is assumed, again to simplify the derivation of the adaptation algorithm. However, optimization problems of engineering relevance typically have a number of nonlinear constraints. Constraint handling in the context of multifidelity optimization is a challenging topic, as the use of low-fidelity models to compute the constraints will modify the feasible region, and may lead to a different active set $\mathcal{A}$. This is also the reason why the error estimation of Brown and Nadarajah [180, 181] is only focused on the error in the gradient, while the error in the function value itself is equally important when constraints are present. This is perhaps the reason why the authors, as well as Lu and Darmofal [178] demonstrated their approaches on inverse design problems which did not require any constraints.

1: The authors called the primal system the *constraints*, in the context of PDE-constrained optimization.

[74]: Nocedal et al. (2006), *Numerical Optimization*

[180]: Brown et al. (2017), *Inexactly constrained discrete adjoint approach for steepest descent-based optimization algorithms*

[181]: Brown et al. (2021), *Effect of inexact adjoint solutions on the discrete-adjoint approach to gradient-based optimization*

[178]: Lu et al. (2004), *Adaptive Precision Methodology for Flow Optimization via Discretization and Iteration Error Control*

In practice, established optimizers are algorithmically highly complex. IPOPT [126], for example, has over 50 000 lines of code (LOC). While there are immense values to analyzing algorithmic behaviour and convergence properties, the simplifications prevent direct adoption of such approaches to larger, industrially-relevant optimization problems. Therefore, the aim of this work is to develop an alternative approach, based on a synthesis of rigorous mathematical analysis and heuristics to tackle larger problems.

[126]: Wächter et al. (2006), *On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming*

As before, the proposed approach involves two main steps. First,

a reliable error estimate needs to be computed, so that we know how much error there are in the outputs of interest. Second, an adaptive algorithm needs to be designed to robustly adjust the convergence tolerance of the solver, such that we still arrive at the same numerical optimum but using less computational resources. We now describe the two steps in more detail.

4.2 Adjoint-based convergence error estimation

In an iterative solution scheme for nonlinear equations, we aim to drive a set of residual equations $\boldsymbol{R}(\boldsymbol{u})$ to zero, where $\boldsymbol{R}$ are the residuals and $\boldsymbol{u}$ are the states. In the context of CFD, this is typically accomplished using Newton-type methods with an appropriate globalization technique [119]. However, even with efficient solvers, it is rarely necessary to obtain the fully-converged solution $\boldsymbol{u}^*$ such that $\boldsymbol{R}(\boldsymbol{u}^*) = 0$.[2] Instead, we use the residual norm

$$\|\boldsymbol{R}(\boldsymbol{u})\|_2 \tag{4.1}$$

as the metric, and terminate the iterative scheme once a sufficient reduction has been achieved. In other words, the termination criterion is a relative tolerance of the form:

$$\frac{\|\boldsymbol{R}(\boldsymbol{u})\|_2}{\|\boldsymbol{R}(\boldsymbol{u}_0)\|_2} \leq \tau_{\text{pr}}, \tag{4.2}$$

where $\boldsymbol{u}_0$ are the initial states,[3] and τ_{pr} is the primal convergence tolerance.

We use a relative metric because the L_2 norm used here is sensitive to the number of entries in the vector. For a denser mesh, the L_2 norm will be naturally larger. To compensate for some of this effect, we use a relative norm measure.

[119]: Yildirim et al. (2019), *A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations*

2: Or at least $\boldsymbol{R}(\boldsymbol{u}^*) \approx \epsilon_{\text{mp}}$ when considering finite precision arithmetic.

3: Typically freestream states are used to initialize the flow.

Comment 4.2.1

The L_2 norm used here is a convenient and commonly-used norm, but it suffers from one issue: the norm scales with the size of the vector. As a result, it is impossible to discuss "comparative levels of convergence" between solutions on different mesh densities, since the norm of the residual would be far higher on a finer mesh. The use of the residual reduction relative to the initial residual is an attempt at addressing this, but it may not be the best solution. In reality, we may want to compute the volume integral in the continuous sense:

$$\int |\boldsymbol{R}| \, \mathrm{d}V.$$

Since the residuals are already volume weighted, the discrete analog would be to apply the L_1 norm:

$$\sum_i |\boldsymbol{R}_i|$$

and use an absolute convergence tolerance irrespective of the initial residual. This addresses another issue with the present approach. The initial freestream residual will be nonzero only at the boundary cells adjacent to the surface of the wing, where the no-slip boundary condition is applied. Therefore, the initial residual is not a true measure of the amount of violation of the governing equations, in terms of the error on the state variables, *i.e.* $|\tilde{\boldsymbol{u}} - \boldsymbol{u}^*|$.

For the adjoint, we use PETSc to solve the linear system, *i.e.* equation 2.15. Similarly to the primal system, we typically prescribe τ_{adj} as the relative residual norm reduction as computed by PETSc. For a linear system of the form $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, we define the linear residual as $\boldsymbol{r} = \boldsymbol{b} - \mathbf{A}\boldsymbol{x}$, and terminate the linear solver based on the following criteria from PETSc:

$$\frac{\|\boldsymbol{r}\|_2}{\|\boldsymbol{b}\|_2} \leq \tau_{\text{adj}}, \tag{4.3}$$

where the denominator can be interpreted as the initial residual vector $\boldsymbol{r}_0$ if the initial guess is the zero vector.

4.2.1 Derivation

Consider a partially-converged intermediate state $\tilde{\boldsymbol{u}}$ where $\boldsymbol{R}(\tilde{\boldsymbol{u}}) \gg \epsilon_{\text{mp}}$, in contrast to the fully-converged solution $\boldsymbol{u}^*$ where $\boldsymbol{R}(\boldsymbol{u}^*) = 0$. The states $\tilde{\boldsymbol{u}}$ can be obtained more quickly than $\boldsymbol{u}^*$, but they do not accurately compute functional outputs such as lift or drag. However, with the help of the adjoint, we can compute a linearized error estimate. We first start by writing a Taylor series expansion of the residual about the partially-converged state:

$$\boldsymbol{R}(\boldsymbol{u}^*) = \boldsymbol{R}(\tilde{\boldsymbol{u}}) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}}\right|_{\tilde{\boldsymbol{u}}} (\boldsymbol{u}^* - \tilde{\boldsymbol{u}}) + \dots \tag{4.4}$$

$$= 0 \tag{4.5}$$

Similarly, we can expand the function of interest $\boldsymbol{\mathcal{I}}$ about the partially-converged state as

$$\mathcal{I}(\boldsymbol{u}^*) = \mathcal{I}(\tilde{\boldsymbol{u}}) + \left.\frac{\partial \mathcal{I}}{\partial \boldsymbol{u}}\right|_{\tilde{\boldsymbol{u}}} (\boldsymbol{u}^* - \tilde{\boldsymbol{u}}) + \dots . \tag{4.6}$$

We can solve for the term $(\boldsymbol{u}^* - \tilde{\boldsymbol{u}})$ by rearranging equation 4.4 and ignoring high-order terms to get

$$\boldsymbol{u}^* - \boldsymbol{u} = - \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}}\right|_{\tilde{\boldsymbol{u}}}^{-1} \boldsymbol{R}(\tilde{\boldsymbol{u}}). \tag{4.7}$$

This term can then be plugged into equation 4.6 to get

$$\mathcal{I}(\boldsymbol{u}^*) = \mathcal{I}(\tilde{\boldsymbol{u}}) - \left.\frac{\partial \mathcal{I}}{\partial \boldsymbol{u}}\right|_{\tilde{\boldsymbol{u}}} \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}}\right|_{\tilde{\boldsymbol{u}}}^{-1} \boldsymbol{R}(\tilde{\boldsymbol{u}}). \tag{4.8}$$

We now define the adjoint vector $\tilde{\boldsymbol{\psi}}$ to be the solution of the linear system

$$\left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}}\right|_{\tilde{\boldsymbol{u}}}^{\mathsf{T}} \tilde{\boldsymbol{\psi}} = \left.\frac{\partial \mathcal{I}}{\partial \boldsymbol{u}}\right|_{\tilde{\boldsymbol{u}}}^{\mathsf{T}}, \tag{4.9}$$

which is simply the adjoint vector $\boldsymbol{\psi}$ computed at the partially-converged states $\tilde{\boldsymbol{u}}$. While the adjoint vector is typically solved at the converged state, it does not have to be. It is simply a linearization, and in this case we are linearizing about the states $\tilde{\boldsymbol{u}}$.

Plugging into equation 4.8, we have

$$\mathcal{I}(\boldsymbol{u}^*) = \mathcal{I}(\tilde{\boldsymbol{u}}) - \tilde{\boldsymbol{\psi}}^\mathsf{T} \boldsymbol{R}(\tilde{\boldsymbol{u}}) \tag{4.10}$$

In other words, the error in a functional output due to poor convergence is given by the dot product of the residual with the adjoint vector, computed about the same point:

$$\epsilon_\mathcal{I} \triangleq \mathcal{I}(\boldsymbol{u}^*) - \mathcal{I}(\tilde{\boldsymbol{u}}) \tag{4.11}$$

$$= -\tilde{\boldsymbol{\psi}}^\mathsf{T} \boldsymbol{R}(\tilde{\boldsymbol{u}}) \tag{4.12}$$

This result can be further improved to be third-order accurate by considering the dual of the above error estimation, *i.e.*, the error due to poor convergence of the adjoint variables $\boldsymbol{\psi}$. Since the linear system equation 4.9 may not be fully solved, the values of $\tilde{\boldsymbol{\psi}}$ introduces an additional source of error. Becker and Rannacher [182] first derived this term as

[182]: Becker et al. (2001), *An optimal control approach to* a posteriori *error estimation in finite element methods*

$$\boldsymbol{R}_{\boldsymbol{\psi}}^\mathsf{T} \boldsymbol{u}, \tag{4.13}$$

where $\boldsymbol{R}_{\boldsymbol{\psi}}$ is the linear residual of equation 4.9. Compared to equation 4.11, this is now the *primal*-weighted (adjoint) residual, which is clearly its dual. Fidkowski and Darmofal [94] showed that by averaging those two error estimates, a more accurate prediction can be made. In this work we do not consider this term, but it can certainly be incorporated in the future.

[94]: Fidkowski et al. (2011), *Review of output-based error estimation and mesh adaptation in computational fluid dynamics*

As an additional topic of investigation, it is possible to use the error estimate to correct for the partially-converged function values. Instead of using $\mathcal{I}(\tilde{\boldsymbol{u}})$ for optimization, we can instead pass the value $\mathcal{I}(\tilde{\boldsymbol{u}}) - \tilde{\boldsymbol{\psi}}^\mathsf{T} \boldsymbol{R}(\tilde{\boldsymbol{u}})$ to the optimizer. While the gradients

would be inconsistent with the function itself, the more accurate function estimate may be beneficial during line search. This is investigated in section 4.4.1.

4.2.2 Verification

To verify the error estimation, we stopped an airfoil analysis at various convergence tolerances. At each level, we converged the adjoint to a tolerance of $\tau_{\text{adj}} = 10^{-12}$, and computed the error estimation.

First, we plot the actual errors and the adjoint-based error estimation against the residual norm. The actual errors are computed using

$$\epsilon_{\text{actual}} = |\tilde{\mathcal{I}} - \mathcal{I}^*|. \tag{4.14}$$

where the final converged values $\tilde{\mathcal{I}}$ are computed with a convergence tolerance of $\tau_{\text{pr}} = 10^{-15}$. These are shown in figure 4.1 in log-log scale. Here, we see that for both lift and drag, the adjoint error estimation is able to accurately capture the function error.

Naturally, by omitting the higher-order terms, we are using the adjoint variables to linearly project the current solution $\tilde{\boldsymbol{u}}$ to $\boldsymbol{u}^*$. While accurate for the range of residual reductions shown in figure 4.1, there is a limit due to the nature of linear extrapolation. For example, computing the error estimate using a relative convergence of 10^{-1} is unlikely to be accurate.

Next, the semi-converged lift and drag values are plotted against the residual norm in figure 4.2. We also show the error-corrected values, *i.e.*, $\mathcal{I}(\tilde{\boldsymbol{u}}) - \tilde{\boldsymbol{\psi}}^{\mathsf{T}}\boldsymbol{R}(\tilde{\boldsymbol{u}})$. Since the error estimation was accurate, the corrected output is much closer to the final converged value, and does not suffer from significant oscillations as seen for c_l.

Of course, the rate at which the error decreases as the residual norm decreases is going to be dependent on the solution

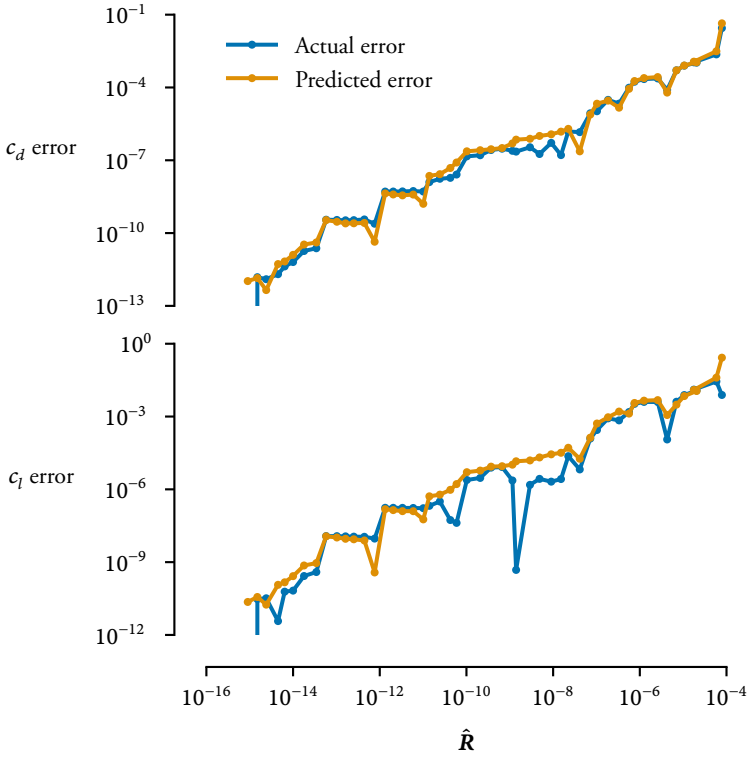


Figure 4.1: Actual and computed errors for different convergence levels.

algorithm used. For example, explicit time-stepping schemes will have a drastically different trend than the results depicted, which were generated using an approximate Newton–Krylov scheme [119]. Nevertheless, the adjoint-based approach will account for those factors and produce accurate error estimates.

[119]: Yildirim et al. (2019), *A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations*

4.3 Convergence tolerance adaptation algorithm

From section 4.2, we see that it is possible to estimate the error on any functional output, at the cost of one adjoint solution each. In the most straightforward and naive approach, we would compute the error estimation and monitor it during the convergence process. Then, once we reach the target error, we stop the primal solution process. However, this would be prohibitively expensive as each error estimation requires solving an adjoint—often as expensive as the entire primal solution itself. Therefore, it would

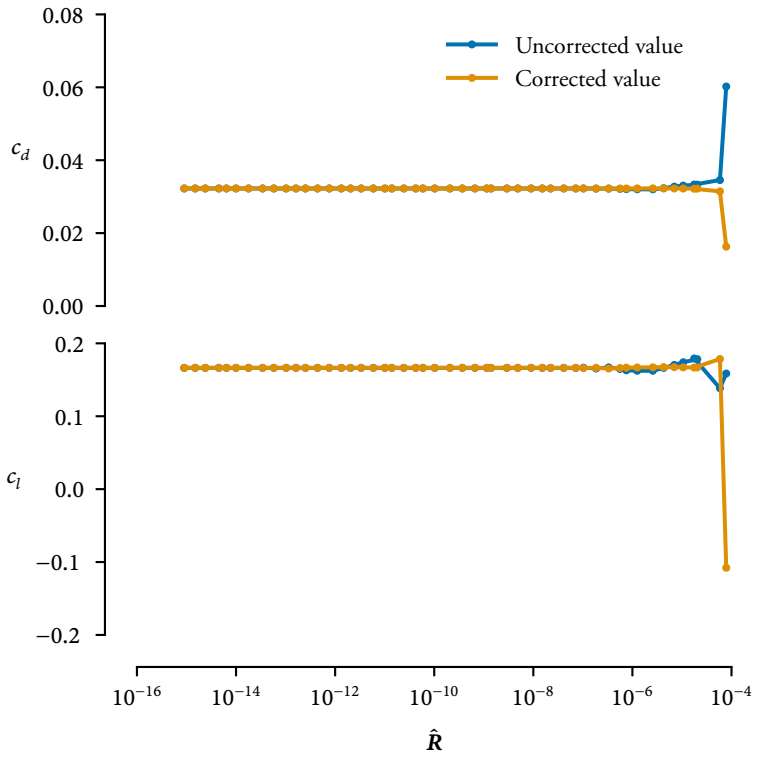


Figure 4.2: Convergence of lift and drag values, without and with correction.

never be worthwhile to estimate the error when we could simply let it fully converge.

Lu and Darmofal [178] bypassed this issue by developing a joint primal-dual solution scheme, where the primal states are solved in conjunction with the adjoint. Unfortunately implementing such a scheme within ADflow would be infeasible. Instead, we propose a lagged approach. Given that we need to solve the adjoint anyhow in order to compute the gradient, we opt to re-use this adjoint solution for error estimation. This means that we cannot adaptively terminate the current primal solution, but instead use the adjoint solution to update the termination criteria for the *next* primal solver. That way, the cost of error estimation is essentially free.

[178]: Lu et al. (2004), *Adaptive Precision Methodology for Flow Optimization via Discretization and Iteration Error Control*

At every major iteration, we retrieve the feasibility τ_{fea} and optimality τ_{opt} from the optimizer SNOPT. Recall from section 2.5.7.1 that *feasibility* corresponds to the residual of the zeroth order KKT conditions, and *optimality* corresponds to the first order KKT conditions. While these residuals are scaled in-

ternally, we assume that the optimization problem has already been suitably scaled outside the optimizer. In such cases, we can take these as absolute tolerances. Then, it is natural to use the feasibility tolerance as an indicator of the appropriate primal convergence, and optimality for the adjoint convergence.

In the simplest case, we let the target error $\bar{\epsilon}$ in computing each of the constraints equal to the current feasibility tolerance achieved, multiplied by a factor:

$$\bar{\epsilon} = k_{\text{pr}} \tau_{\text{fea}} \tag{4.15}$$

where k_{pr} is typically less than one—it does not make sense to have the error of a functional output be larger than the constraint violation. A range of values for k_{pr} were examined in this work. Obviously, the smaller the value, the tighter the convergence tolerance, so there is a tradeoff between robustness and performance.

This error on the constraints still needs to be translated into errors on functionals, which can be done via either analytic differentiation or complex step. For example, suppose we have a lift constraint $g(\boldsymbol{x}) = c_l(\boldsymbol{x}) - 0.5$. Then, a feasibility tolerance of 10^{-2} means we should be computing $g(\boldsymbol{x})$ with an error of 10^{-2}, which then translates to an error on the functional c_l of 10^{-2}. In the case of multiple constraints, each will have a target error and resulting convergence tolerance. Naturally, the tightest tolerance is used.

Once we have the error, we then need to determine the residual norm that would yield such an error. We assume a linear relationship between the residual norm and the error, and that the slope is one. See appendix A for some numerical experiments that justify this choice.

$$\frac{\epsilon}{\bar{\epsilon}} = \frac{\|\boldsymbol{R}\|_2}{\|\bar{\boldsymbol{R}}\|_2} \tag{4.16}$$

Given the current error ϵ, target error $\bar{\epsilon}$, and current residual norm $\|\boldsymbol{R}\|_2$, we can easily compute the target residual norm $\|\bar{\boldsymbol{R}}\|_2$ that would yield the target error. From equation 4.16:

$$\begin{aligned}\frac{\epsilon}{\bar{\epsilon}} &= \frac{\|\boldsymbol{R}\|_2/\|\boldsymbol{R}_0\|_2}{\|\bar{\boldsymbol{R}}\|_2/\|\boldsymbol{R}_0\|_2} \\ &= \frac{\tau_{\text{pr}}^k}{\tau_{\text{pr}}^{k+1}} \\ \tau_{\text{pr}}^{k+1} &= \frac{\bar{\epsilon}}{\epsilon}\tau_{\text{pr}}^k \end{aligned} \tag{4.17}$$

where the superscripts k and $k+1$ indicate the major iteration number.

In other words, if we need to reduce the error by an order of magnitude, then the residual tolerance should also be reduced by an order of magnitude. We chose to use this relative measure (*i. e.* based on the current value of τ_{pr}) because the linear relationship between ϵ and $\|\boldsymbol{R}\|_2$ has an undetermined y-intercept. Instead of relying on an *a priori* curve-fit of this linear relationship, we simply adjust τ_{pr} based on its current value, and the current and expected errors.

One obvious issue with this approach is that it does not take the accuracy of the objective function into account. As the objective value is not part of the KKT system—only the derivatives—there is no easy way to incorporate it in a mathematically-rigorous fashion. It is possible to use a similar approach to what is done in SLSQP [131], and take successive objective function decrease as an alternative optimality metric. This is in fact used by Lu and Darmofal [178]. However, given the tight coupling between the objective and constraints in most CFD applications—tighter convergence will typically improve the accuracy of both simultaneously—we have opted not to specifically include the objective error.

[131]: Kraft (1988), *A software package for sequential quadratic programming*

[178]: Lu et al. (2004), *Adaptive Precision Methodology for Flow Optimization via Discretization and Iteration Error Control*

For the adjoint convergence tolerance, we simply set it to $\tau_{\text{adj}} = k_{\text{adj}}\tau_{\text{opt}}$, which is mathematically equivalent to the results of

[181]: Brown et al. (2021), *Effect of inexact adjoint solutions on the discrete-adjoint approach to gradient-based optimization*

Brown and Nadarajah [181] in the unconstrained case. In this work, we use a fixed value of $k_{\text{adj}} = 10^{-6}$.

In addition to these, there are some safeguards to ensure benign convergence behaviour. First, we place upper and lower bounds on τ_{pr} and τ_{adj}, so that they are never too loose or tight. We also limit the change in these tolerances across optimization iterations, such that no positive feedback occurs—if an iterate does not do well, the feasibility may increase, which would induce a relaxed τ_{pr} that further hinders the optimization. Finally, if we are close to the final optimum, then both τ_{pr} and τ_{adj} are set to their lower bound. This ensures that by the end of the optimization, we are fully converging our solutions in order to arrive at the same numerical optimum. There are also times where the optimizer may struggle to find a new point, likely due to the poor convergence of the solver. We detect those cases, and set τ_{pr} and τ_{adj} to their lower bound so that SNOPT may recover. A pseudocode of the process is detailed in algorithm 1.

Algorithm 1: Adaptive error control

```
while optimization not converged do
    set design variables and update geometry;
    solve primal problem to tolerance τ_pr;
    solve adjoint problem to tolerance τ_adj;
    compute ϵ using adjoint-based error estimation;
    if major iteration then
        compute τ_pr and τ_adj;
        set updated values for τ_pr and τ_adj;
    compute the next design vector from the optimizer;
```

4.4 Results

We examine the performance on two optimizations, a 2D airfoil problem and a 3D wing problem. In both cases, we compare the adaptive approach with a reference optimization, and use $\tau_{\text{pr}} = \tau_{\text{adj}} = 10^{-12}$ for all iterations.

Table 4.1: The airfoil optimization problem. "(L)" denotes that the constraint is linear.

	Function/variable	Description	Quantity
minimize	c_d	Drag coefficient	1
with respect to	α	Angle of attack	1
	x_{shape}	Shape	20
subject to	$c_l = 0.5$	Lift constraint	1
	$V \geq V_0$	Volume constraint	1
	$t \geq 0.1t_0$	Thickness constraint	100
	$\Delta z_{\text{LE,upper}} = -\Delta z_{\text{LE,lower}}$	Fixed leading edge (L)	1
	$\Delta z_{\text{TE,upper}} = -\Delta z_{\text{TE,lower}}$	Fixed trailing edge (L)	1

4.4.1 Adaptive airfoil optimization

The first test case is a single-point airfoil drag minimization problem. The flight condition is at an altitude of 10 000 m and a Mach number of 0.75. The initial geometry is the RAE2822 airfoil, and the cell count is 18 816.

The design variables are 20 FFD control points, plus the angle of attack. The lift coefficient is constrained at 0.5. In addition, there are volume, thickness, and LE/TE constraints. The full optimization problem is shown in table 4.1.

A reference optimization was first performed, with a constant convergence tolerance set to 10^{-12} for both the primal and adjoint systems. In order to investigate the effect of k_{pr}, as well as whether function correction was beneficial, eight additional optimizations were performed at a range of four k_{pr} values. These results are shown in table 4.2.

	Adaptive		Adaptive + Correction	
k_{pr}	η (%)	Iters	η (%)	Iters
Reference	—	108	—	—
1	—*	27	—*	27
10^{-1}	26	139	26	139
10^{-2}	41	109	40	109
10^{-3}	41	106	40	106
10^{-4}	37	108	37	108

Table 4.2: Airfoil optimization results. Note that the reference optimization was performed without any adaptation, and serves as the reference for the speedup shown in the columns labelled η. Optimizations marked with an asterisk did not converge successfully.

The first thing to note is that both optimizations with $k_{\text{pr}} = 1$

failed to converge. This is somewhat expected, since k_{pr} directly affects the primal convergence tolerance, and indicates that the solution converged too poorly at early stages for the optimizer to make sufficient progress.

For those that did converge, we see that as k_{pr} decreased, an interesting tradeoff develops. On one hand, each iteration is more expensive as tighter convergence is required. On the other hand, the number of optimization iterations is decreased as a result of more accurate function values. Combined, these two effects result in a relative insensitivity to k_{pr} in the range of 10^{-2} to 10^{-4}, where cost savings are all around 40%.

The last observation is that in the cases tested, function correction was not beneficial. This was somewhat unexpected, as we had hoped that it would allow us to successfully converge optimizations with a relatively large k_{pr}, and thereby gain additional performance benefits. Unfortunately, this did not prove to be the case, and the behaviour was almost identical to the case without correction. In retrospect, this was perhaps unsurprising, as the errors are rather small and quickly drop below the function precision expected by SNOPT. Nevertheless, it may be helpful to further investigate why the optimizations with $k_{\text{pr}} = 1$ failed to converge.

The feasibility and optimality of the optimizations (without function correction) are plotted in figure 4.3, together with the primal and adjoint tolerances via the adaptive algorithm. Firstly, it is clear that the optimization with $k_{\text{pr}} = 1$ failed because τ_{pr} was consistently around its upper bound of 10^{-6}. The adaptive algorithm attempted to recover at the end, by setting both tolerances to their lower bounds. Unfortunately this did not prove effective. With $k_{\text{pr}} = 10^{-1}$, the optimization suffered a similar setback just before iteration 40, where the line search failed. However, the recovery this time was effective, and still yielded about 25% cost savings. The rest of the optimizations had similar performance when we examine their feasibility and optimality

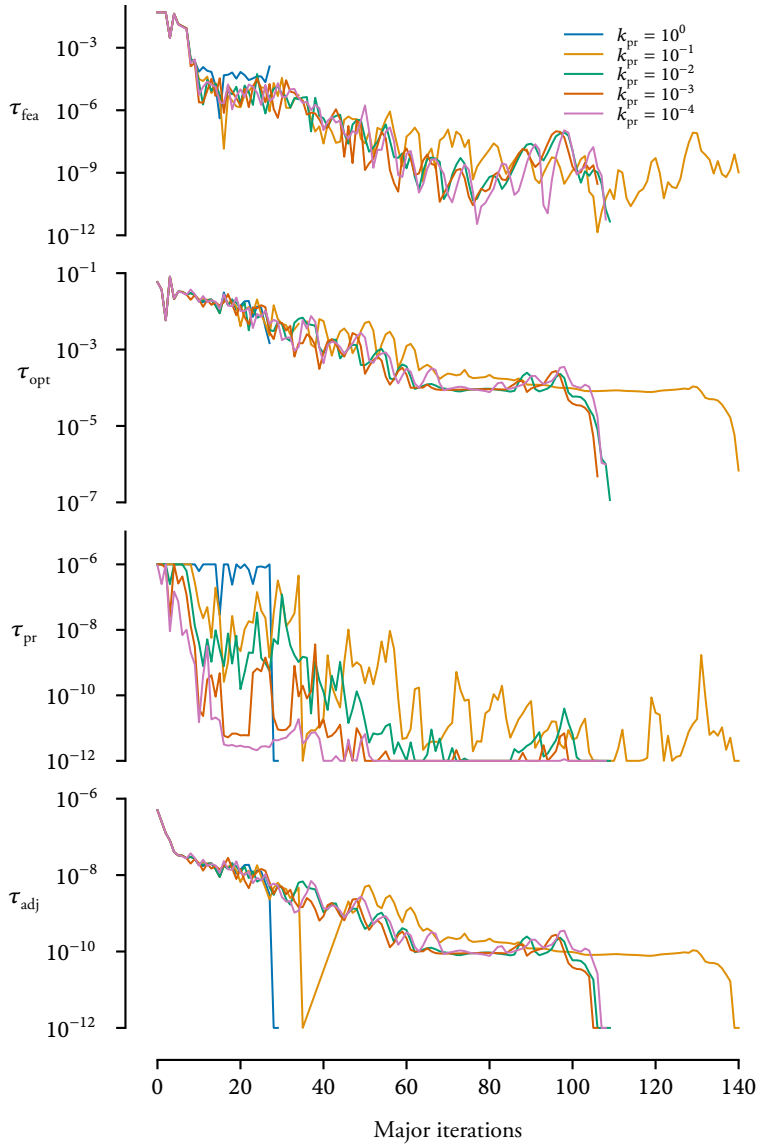


Figure 4.3: Optimization and solution tolerances for the airfoil problem.

tolerances, despite large differences in the primal solution tolerance. The effect of k_{pr} can be clearly seen, where optimizations with a lower value consistently having lower primal convergence tolerances.

In a typical optimization, the initial optimizations typically take the longest time, for two reasons. First, the initial solution is started from a uniform free-stream, which is a poor initial point. Subsequent iterations are always restarted from the previous solution, which offers some speedup. Second, the designs tend to change drastically during the initial stages of the optimization.

Therefore, the restart feature is less efficient early on. Due to these effects, it is expected that loosely converging the early solutions will offer the most cost savings. In addition, ADflow has an efficient implicit solution algorithm. This means that changing τ_{pr} by an order of magnitude will not perhaps save too much computation. The adaptive approach could offer even more benefit when applied to a different solver, perhaps one that uses explicit time-stepping for globalization of the Newton–Krylov solver.

As a final investigation, we examine whether setting *function precision* within SNOPT helps with convergence when k_{pr} is large. The function precision parameter is used by SNOPT to determine the minimum expected improvement during line search, in order to prevent excessive function evaluations when the expected improvement is less than the precision of the function. Naturally, we should set this value to the error computed. This was explored in a series of optimizations listed in table 4.3.

Table 4.3: The effect of adaptively changing the function precision parameter within SNOPT, without and with function value correction. Optimizations marked with an asterisk did not converge successfully.

	Adaptive FP		Adaptive FP + Correction	
k_{pr}	η (%)	Iters	η (%)	Iters
1	—*	27	—*	27
10^{-1}	27.33	139	26.99	139

Once again, we did not find any benefits in changing the function precision, either on its own or in conjunction with the function correction approach. This is likely due to the fact that this is a relatively well-behaving optimization problem, where line searches that result in a step length of less than unity is rare. In these cases, the function precision does not play a role, and therefore does not impact the overall optimization performance.

4.4.2 Adaptive wing optimization

To further examine the effects of k_{pr}, we perform a 3D wing optimization. This is the same twist-only optimization problem

Table 4.4: The wing twist-only optimization problem.

	Function/variable	Description	Quantity
minimize	C_D	Coefficient of drag	
with respect to	x_{twist}	Sectional twist	7
	x_{alpha}	Angle of attack	1
subject to	$C_L = 0.5$	Lift constraint	1

that was investigated in chapter 3. The full optimization problem is given in table 4.4.

We perform a reference optimization, together with five adaptive runs with varying k_{pr}. We do not correct the function values, nor do we adjust the function precision, since both were found to be ineffective in the previous tests. The results are given in table 4.5. Interestingly, all optimizations converged successfully, and the case with $k_{\text{pr}} = 1$ had the most computational gains at over 50%.

Table 4.5: Optimization results for the 3D wing case, tested with a variety of k_{pr} values.

k_{pr}	Walltime (s)	η (%)	Iters
Reference	674	—	26
1	320	53	26
10^{-1}	380	44	26
10^{-2}	420	38	26
10^{-3}	444	34	26
10^{-4}	486	28	26

The optimization histories are shown in figure 4.4. As this optimization problem had fewer design variables than the 2D airfoil case, fewer iterations were taken. All optimizations had fairly similar trajectory, and the effect of k_{pr} can be clearly seen in the third plot. As mentioned before, due to the efficient solver algorithm, the impact of k_{pr} on the computational cost is less significant than one might expect. The performance of the adaptive algorithm on both optimization cases are plotted in figure 4.5.

Based on these results, k_{pr} values between 10^{-2} and 10^{-3} offer good balance between robustness and performance. Of course, this would be both problem-dependent and solver-dependent, as the norm and tolerances used here are specific to ADflow.

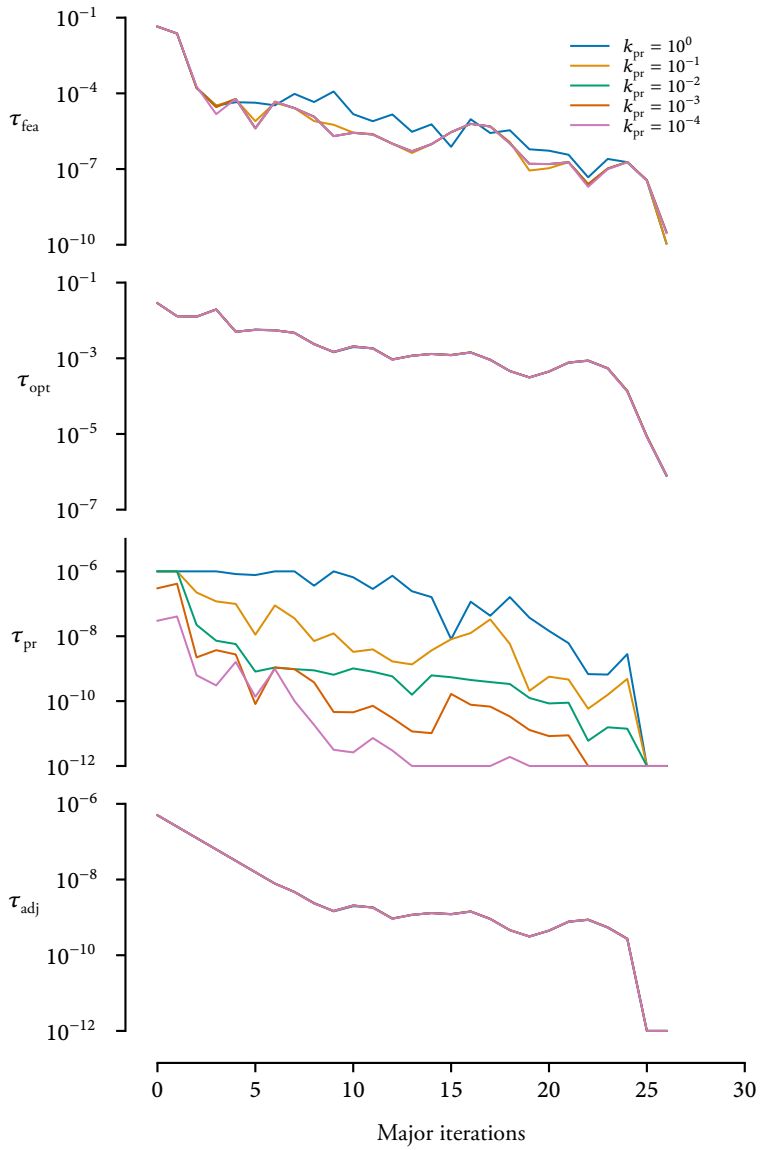


Figure 4.4: Optimization and solution tolerances for the wing problem.

4.5 Summary

In this work, we have developed an adjoint-based approach to estimate the functional errors due to poor convergence. We then developed an adaptive algorithm to control the analysis convergence tolerance during optimization, in order to reduce computational costs. We tested this on two ASO problems, and found that in most cases, savings of between 30% and 50% were observed.

We also investigated two other possible extensions to the ap-

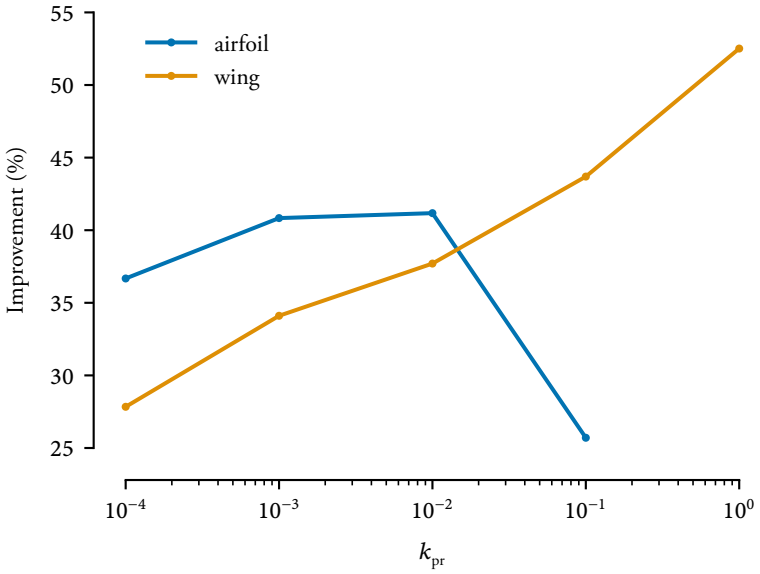


Figure 4.5: The effect of k_{pr} on the relative improvement of the two optimization problems.

proach. One involved correcting the poorly-converged function value with the error estimate, but this proved to be ineffective. Another involved adaptively setting the "function precision" parameter within SNOPT, in the hopes of reducing the number of line search iterations. This also had little effect on the overall performance. Nevertheless, the approach still yields significant computational cost savings.

Chapter 5

Appropriate Fidelity MDO Framework

5.1 Background

Ever since the pioneering work of Haftka [17] on aerostructural optimization, there have been significant efforts towards improving modeling capabilities within MDO, where increasingly complex analyses are being integrated into an optimization framework. However, more complex modeling incurs a drastically increased computational cost. As we add further analysis capabilities to aircraft MDO, computational costs will likely become prohibitive if only high-fidelity models are used.

[17]: Haftka (1977), *Optimization of Flexible Wing Structures Subject to Strength and Induced Drag Constraints*

In recent years there has been growing interest in multifidelity approaches, where different models with varying levels of accuracy, or fidelities, are combined within the optimization process to reduce overall computational time. Low-fidelity models are cheaper to evaluate but provide less accurate predictions. Together with a model management framework, they form the basis of a multifidelity optimization framework.[1]

1: This is in contrast to *mixed-fidelity* methods, where different fidelities are used to compute different outputs. For example, using CFD for aerodynamics but panel method for flutter prediction.

In the comprehensive review of multifidelity methods, Peherstorfer, Willcox, and Gunzburger [96] distinguished between global and local optimization methods for the first time. In global methods, a search is conducted over the entire feasible domain. In contrast, local methods terminate when a local optimum is found. Typically, global methods do not require gradients, and when supplied with gradient information, local methods can rapidly

[96]: Peherstorfer et al. (2018), *Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization*

converge to a local optimum. We use the same characterization below and provide a short review focusing on optimization, particularly for large-scale problems. We then discuss multifidelity methods in the context of multidisciplinary problems and highlight some shortcomings of existing approaches.

5.1.1 Global and local methods

In global methods, a globally-accurate multifidelity surrogate model is constructed and used during optimization. These approaches are based on Bayesian optimization techniques, relying on the construction of surrogate models. Giselle Fernández-Godino et al. [95] surveyed 178 papers using multifidelity methods and found 73% of them use a multifidelity surrogate model. Unfortunately, surrogate models all suffer from the curse of dimensionality [183]. This has been apparent in the multifidelity optimization field, where applications typically have fewer than 20 design variables [184, 185, 186, 187]. This is in direct contrast to high-fidelity aerostructural optimizations with upwards of a thousand design variables [27], which are precisely the type of expensive optimizations that would benefit the most from multifidelity methods. Furthermore, most surrogate-based optimization techniques do not leverage gradients, nor are they equipped to handle general nonlinear constraints. As such, they cannot verify the satisfaction of the KKT conditions required for optimality and cannot be shown to be convergent to the high-fidelity optimum. Instead, their termination criteria are commonly based on heuristics.

[95]: Giselle Fernández-Godino et al. (2019), *Issues in Deciding Whether to Use Multifidelity Surrogates*

[183]: Viana et al. (2014), *Metamodeling in Multidisciplinary Design Optimization: How Far Have We Really Come?*

[184]: Leifsson et al. (2010), *Multi-fidelity design optimization of transonic airfoils using physics-based surrogate modeling and shape-preserving response prediction*

[185]: Choi et al. (2008), *Multifidelity Design Optimization of Low-Boom Supersonic Jets*

[186]: Forrester et al. (2006), *Optimization using surrogate models and partially converged computational fluid dynamics simulations*

[187]: Nguyen et al. (2013), *Multidisciplinary unmanned combat air vehicle system design using multifidelity model*

[27]: Brooks et al. (2018), *Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings*

On the other hand, local methods are only concerned with finding a local minimum. With the help of gradient-based approaches, these methods can be very effective when dealing with many design variables. In this area, the primary method has been trust-region model management (TRMM), which is adapted from the single-fidelity trust-region SQP method by replacing the local quadratic model with the low-fidelity model. This was

investigated first by Alexandrov et al. [188], who showed that if the low-fidelity model is corrected to satisfy the first-order consistency conditions, this method is provably convergent to the high-fidelity optimum. These requirements are typically met through additive and multiplicative corrections, and the size of the trust region is updated based on the correlation between high and low-fidelity models.

[188]: Alexandrov et al. (2001), *Approximation and Model Management in Aerodynamic Optimization with Variable-Fidelity Models*

Since then, there have been several developments based on the TRMM approach. Elham and Tooren [189] adopted a Pareto filter rather than a penalty parameter when using an augmented Lagrangian approach but was unable to obtain the high-fidelity optimum, and the computation was four times slower than the single-fidelity approach. March and Willcox [190] combined the trust-region approach with Bayesian calibration to construct a provably-convergent multifidelity method that does not require high-fidelity derivatives. However, the number of function evaluations was still prohibitively high given the lack of gradient information. Gratton, Sartenaer, and Toint [191] proposed a recursive TRMM formulation that allowed for an arbitrary number of fidelities to be used. This approach is reminiscent of multigrid methods in the numerical solution of PDEs, where the algorithm switches between coarse and fine grids to accelerate convergence. This method was further extended by Olivanti et al. [192] to include both a gradient-based switching criterion as well as an asynchronous validation scheme and demonstrated on an ASO problem with 191 design variables, showing a speed-up factor of 1.3.

[189]: Elham et al. (2017), *Multi-fidelity wing aerostructural optimization using a trust region filter-SQP algorithm*

[190]: March et al. (2012), *Provably convergent multifidelity optimization algorithm not requiring high-fidelity derivatives*

[191]: Gratton et al. (2008), *Recursive trust-region methods for multiscale nonlinear optimization*

[192]: Olivanti et al. (2019), *Comparison of Generic Multi-Fidelity Approaches for Bound-Constrained Nonlinear Optimization Applied to Adjoint-Based CFD Applications*

Bryson and Rumpfkeil [193] developed an approach based on TRMM that addressed several shortcomings of the original methodology. Since conventional TRMM uses the low-fidelity model for the local surrogate, no approximate Hessian is kept. Therefore, high-fidelity evaluations are only used to calibrate the low-fidelity output to construct the local surrogate. Instead of discarding those after each step, Bryson *et al.* maintain the approximate Hessian, which is used in conjunction with line search

[193]: Bryson et al. (2018), *Multifidelity Quasi-Newton Method for Design Optimization*

to pick the next iterate. Furthermore, the optimization reverts to using only the high-fidelity model when the trust-region size is below a threshold to accelerate the optimization convergence at the final stages. This approach was demonstrated subsequently on aerostructural optimization problems [194, 195].

[194]: Bryson et al. (2019), *Aerostructural Design Optimization Using a Multifidelity Quasi-Newton Method*
[195]: Thelen et al. (2022), *Multi-Fidelity Gradient-Based Optimization for High-Dimensional Aeroelastic Configurations*

Unfortunately, TRMM-based approaches still suffer from some shortcomings. Combining multiple fidelities is a significant challenge because the constraint boundary changes based on the analysis fidelity. As such, most of the results do not include general nonlinear constraints. Instead, they either include only design variable bounds or satisfy the constraint (such as the trim constraint) internally within the solver rather than leaving it to the optimizer [192]. In other cases, a penalty approach is used to combine the objective and constraints into a single objective function, resulting in an unconstrained optimization problem [194].

[192]: Olivanti et al. (2019), *Comparison of Generic Multi-Fidelity Approaches for Bound-Constrained Nonlinear Optimization Applied to Adjoint-Based CFD Applications*
[194]: Bryson et al. (2019), *Aerostructural Design Optimization Using a Multifidelity Quasi-Newton Method*

5.1.2 Multifidelity MDO

Existing work on multifidelity optimization methods applied to multidisciplinary systems is unfortunately sparse. Several works mentioned earlier demonstrate their methodology on aerostructural systems [189, 194] that are inherently multidisciplinary. Still, the low-fidelity aerodynamic model is always coupled with the low-fidelity structural model, resulting in only two distinct fidelities for the overall aerostructural system. As a result, there is no analysis on the tradeoffs of improving the fidelity of either aerodynamics or structures independently. On the other hand, Allaire and Willcox [196] proposed a methodology for identifying and improving the discipline fidelity that contributed the most error in the objective. However, the approach does not account for the computational cost of each fidelity, nor does it consider the effect of discipline errors on constraints.

[189]: Elham et al. (2017), *Multi-fidelity wing aerostructural optimization using a trust region filter-SQP algorithm*
[194]: Bryson et al. (2019), *Aerostructural Design Optimization Using a Multifidelity Quasi-Newton Method*

[196]: Allaire et al. (2014), *A mathematical and computational framework for multifidelity design and analysis with computer models*

A significant difficulty with MDO problems in multifidelity optimization is the lack of hierarchy for model fidelities. For example, while it is clear that a RANS CFD solver has a higher fidelity relative to an Euler solver, it is not clear whether a RANS solver combined with a coarse structural grid would be a higher fidelity compared to an Euler solver using a finer structural grid. Due to the nature of a multidisciplinary problem, many fidelity combinations can arise, posing a challenge to multifidelity methods. Most existing methods cannot even account for more than two possible fidelities in total. They may also suffer from poor scaling when many fidelities are used; for example, they might require a surrogate model to be trained for each fidelity. In addition, there is an implicit assumption that all the given fidelities are useful, which is not true when considering multidisciplinary problems. Few existing methods can analyze all available fidelities and discard those that are not useful (*e.g.* fidelity 5 in figure 5.2).

Furthermore, many multidisciplinary methods do not capture coupled errors [196, 197], which arise due to multidisciplinary interactions between low-fidelity models. Capturing such errors is critical in making informed decisions about model selection. We focus on effectively identifying and managing fidelity combinations for multidisciplinary systems to tackle this challenge.

[196]: Allaire et al. (2014), *A mathematical and computational framework for multifidelity design and analysis with computer models*
[197]: Bayoumy et al. (2020), *A relative adequacy framework for multimodel management in multidisciplinary design optimization*

5.1.3 Aim of proposed method

We propose a gradient-based multifidelity optimization framework that leverages the existing high-fidelity MDO approach. We still perform a single gradient-based optimization, but during the process, we periodically evaluate the adequacy of the current model and switch to a better fidelity if necessary. Because we limit the available fidelities to be discipline specific, we first quantify the error in discipline outputs and then propagate them to multidisciplinary system-level outputs. Based on the errors and associated computational cost, an algorithm is proposed to select

the appropriate fidelity for the current point in the optimization. Error-based switching criteria trigger the periodic fidelity evaluation. A robust hot-start strategy allows the subsequent optimization to progress smoothly without losing information. The proposed methodology, named the *appropriate fidelity framework*, can handle large-scale optimizations with hundreds of design variables and constraints and converge to the same high-fidelity optimum as the single-fidelity approach. Many fidelities can be handled, including those that are not necessarily useful, which are automatically filtered. By leveraging low-fidelity models, we can realize significant cost savings.

We note that this type of sequential optimization approach has been employed before, notably by Lyu, Kenway, and Martins [79] and later adopted by Bons and Martins [86]. Named the "multi-level optimization acceleration technique", the authors perform a sequence of optimizations starting from the coarsest grid and switching to a finer grid at the end of each optimization to reduce the number of iterations needed on the finest grid. Similarly, Koziel and Leifsson [198] performed multi-level optimization with output space mapping applied to airfoil shape optimization. While the basic ideas are similar, they are done in an *ad hoc* fashion and lack many of the key ingredients that make the proposed methodology scalable and general to MDO problems.

[79]: Lyu et al. (2015), *Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark*

[86]: Bons et al. (2020), *Aerostructural Design Exploration of a Wing in Transonic Flow*

[198]: Koziel et al. (2013), *Multi-level CFD-based airfoil shape optimization with automated low-fidelity model selection*

The following sections introduce the high-fidelity optimization approach and some definitions and terminology, followed by an overview of the appropriate multifidelity framework. Finally, we demonstrate the framework on a multipoint aerostructural benchmark problem with over a hundred design variables.

5.2 Methodology

It is typical in engineering applications to have multiple analysis tools available that can compute the discipline outputs $\boldsymbol{y}$ to

varying degrees of accuracy. Any discussion of accuracy requires a "truth" model to act as a reference, provided by detailed simulations or even experimentation. We assume such a model exists and is known *a priori*, and can be evaluated as needed to provide the reference for lower-fidelity models. Without loss of generality, we assume this is provided by a known simulation model, which we call the high-fidelity model. Then, the high-fidelity model has zero error by construction.

Given a reference model, we define fidelity to be the accuracy of a given model for a particular input vector $\boldsymbol{x}$ and scalar output y. The fidelity of a model may vary over the design space and depend on the output QOI. For example, Euler-based CFD may be a relatively high-fidelity model for predicting lift, but it is low fidelity in drag because it ignores skin-friction drag. Although higher-fidelity models are commonly expected to be more computationally expensive, it is not always the case. For example, it has been shown that in some cases, XFOIL [199] is both cheaper and more accurate than RANS CFD when analyzing airfoils for low Reynolds number flows [200] , in which case XFOIL should always be used.

[199]: Drela (1989), *XFOIL — An analysis and design system for low Reynolds number airfoils*

[200]: Morgado et al. (2016), *XFOIL vs CFD performance predictions for high lift low Reynolds number airfoils*

In this work, we restrict fidelities to the discipline level, corresponding to a single engineering analysis. The only requirement for such fidelities is that they have the same inputs and outputs such that there is no need to perform design variable mapping between fidelities. In addition, as the framework is based on the MDF formulation used in a gradient-based setting, all available fidelities in each discipline must be fully coupled with all fidelities in the other disciplines, and the coupled derivatives must be available as well.

In a multidisciplinary setting, determining the available fidelities is a combinatorial problem. Suppose we have three aerodynamic fidelities and two structural fidelities. In that case, there are a total of six possible fidelity combinations for each MDA. In addition, the number of possible system-level fidelity combinations grows

even more in a multipoint optimization problem, where the same design is analyzed under multiple operating conditions. This requires novel techniques to handle many possible fidelities at the optimization level.

Our proposed framework is based on the single-fidelity MDF approach. We perform a sequence of such single-fidelity MDO, starting from the low-fidelity models and gradually improving the model during optimization until we reach the high-fidelity optimum. The framework tries to answer two questions: When is it time to update the fidelity choice? And what fidelity choice to update to next?

We start by quantifying the errors between the different fidelities in computing the various outputs. We then perform a single-fidelity MDO, starting from the lowest fidelity combinations and monitoring its progress. We stop the optimization when specific error-based termination criteria are met, then update the fidelity. This is a two-step process: we first propagate the errors from discipline outputs to system-level quantities, then construct error metrics to select the subsequent fidelity. We then continue the optimization with updated fidelities. This sequence of single-fidelity optimizations eventually reaches the high-fidelity case, which is allowed to continue until convergence. Figure 5.1 shows this process applied to a single-point MDO problem using the XDSM diagram [87]. In the following sections, we explain each of the steps in this process in more detail.

[87]: Lambe et al. (2012), *Extensions to the Design Structure Matrix for the Description of Multidisciplinary Design, Analysis, and Optimization Processes*

5.2.1 Error quantification

Unlike most existing methods, we do not assume that all available models are useful. Instead, we perform error quantification to assess the suitability of all available fidelities. As mentioned earlier, the fidelity of a given analysis method depends on both the input $\boldsymbol{x}$ and the output $\boldsymbol{y}$. In this work, we opted to neglect the effect of the design variables because we are focused

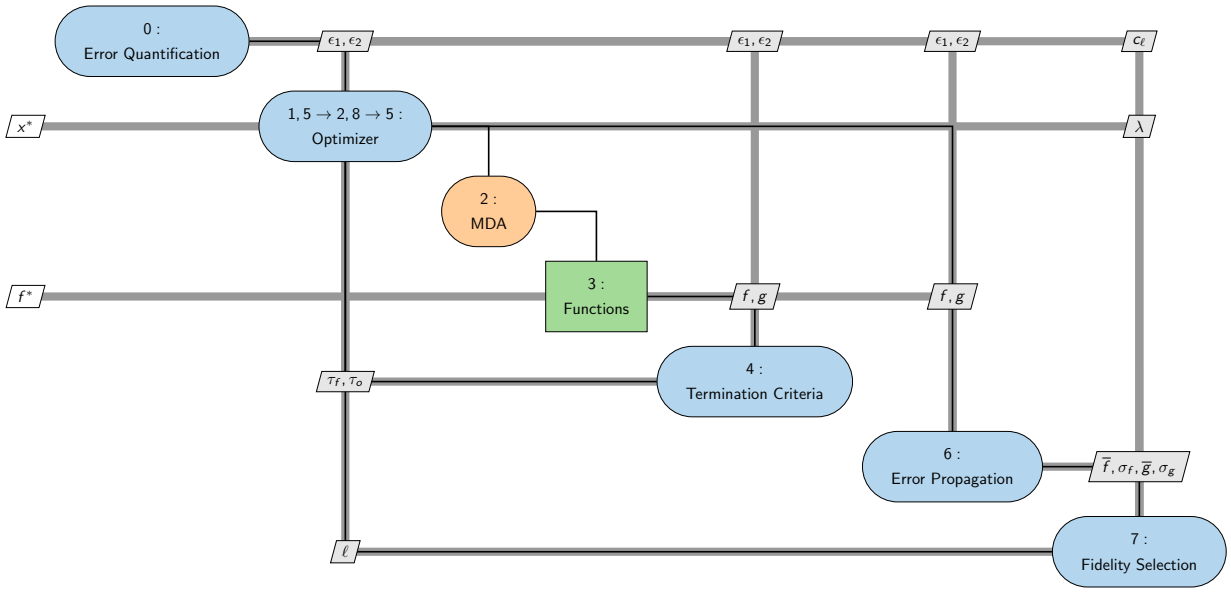


Figure 5.1: An XDSM diagram showing the appropriate fidelity framework applied to a single-point MDO problem.

on high-dimensional problems where the costs are prohibitive. Instead, we perform this error quantification process just once at the initial design point $\boldsymbol{x}_0$. Although it is possible to do this quantification every time the fidelity is updated, we found that unnecessary. In all of our test cases, the errors did not change sufficiently to cause a different sequence of fidelities to be selected.

We first perform single-discipline evaluations using each available fidelity at the initial design point and record their outputs. The disciplines need to be evaluated at each operating point for a multipoint problem. In this work, we use $\ell \in \{0, 1, 2 \ldots n_\ell - 1\}$ to denote the fidelity used to compute a given quantity, with $\ell = 0$ as the high-fidelity reference. We then compute the error for each fidelity ℓ and scalar output y_i using[2]

$$\delta_{i,\ell} = |y_{i,\ell} - y_{i,0}|. \tag{5.1}$$

By construction, $\delta_{i,0} = 0$. We also record the computational cost c_i for performing each fidelity analysis, measured, for example, in CPU-seconds.

2: By capturing the error only on the scalar outputs y_i, we do not capture other information such as the location of the error prior to constraint aggregation.

In theory, each model should fall onto the Pareto front of accuracy and computational cost—otherwise, it could be replaced by a cheaper and more accurate model. For example, in figure 5.2 we can see that fidelity 5 is not Pareto-optimal since fidelity 3 is both cheaper and more accurate. There would be no use for such fidelity in this case, and it is removed. Through this filtering process, we can identify and remove fidelities that are non-optimal. The rest of the fidelities form a Pareto front, resulting in a natural ordering. This provides the ordering for ℓ, where a higher number indicates a decrease in fidelity.

Unlike the field of variable-fidelity optimization, where the fidelity can be arbitrarily controlled, we are not concerned with constructing Pareto-optimal models in this work. We do not examine whether each physics or numerical model choice is appropriate. Instead, we take them as provided and assess them purely on their ability to predict output quantities of interest. This way, the multifidelity framework is problem-agnostic and general for arbitrary MDO problems.

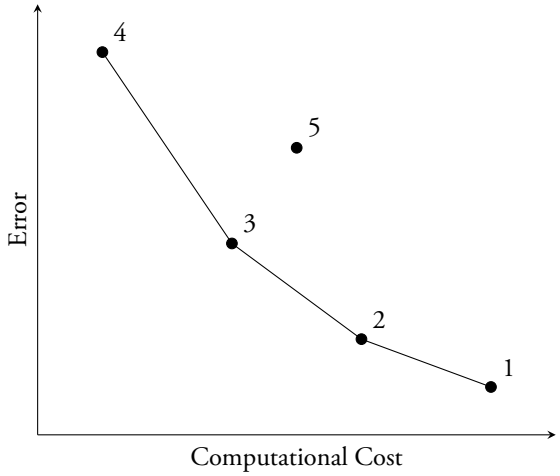


Figure 5.2: Notional diagram showing a discipline analysis involving five fidelities, where fidelities 1–4 are Pareto-optimal and 5 is not. This is for a single scalar output; the relative positions of the fidelities could be different for other outputs.

Because we do this for each scalar output y_i, the outcome may be different for different outputs. There are two possible ways to remove non-optimal fidelities. We can either remove a fidelity if at least one of its scalar outputs is not Pareto-optimal, or if all outputs are non-optimal. We choose the former in this work, but both have been implemented as software options. The remaining

fidelities now fall onto the Pareto front and can be sorted based on the cost of constructing a hierarchy of discipline fidelities.

This initially quantified error can either be assumed to be constant throughout or updated at the end of each sub-optimization so that fidelity selection is performed with the most up-to-date information. We have not found updating the error to be useful for the optimization problem demonstrated. Hence, the rest of the paper assumes that the error is fixed.

5.2.2 Error propagation

After quantifying the discipline errors, we propagate them to system-level outputs, which consist of the objective and constraint functions. This allows us to quantify the impact of each discipline's fidelity on the overall system fidelity combination. Unlike the previous step, this is performed at the end of each sub-optimization because error propagation depends on the current function values.

As hinted earlier in the section, the system-level fidelities are actually fidelity combinations composed of fidelity choices for each discipline and analysis. For example, a two-point aerostructural problem would have a fidelity combination consisting of four choices. We first identify all possible fidelity combinations from which to select the subsequent fidelity combination. In this work, they are all the possible combinations where each discipline's fidelity is better than or equal to the current fidelity. This allows for simultaneous fidelity improvements in multiple disciplines and skipping certain fidelities. For example, we could skip the medium grid and go directly from the coarse to the fine grid for CFD.

Once we identify the fidelity combinations, we then propagate their discipline errors to system-level objectives and constraints for each of those combinations. To do this, we follow the approach outlined by Allaire and Willcox [196] and

[196]: Allaire et al. (2014), *A mathematical and computational framework for multifidelity design and analysis with computer models*

model each discipline output y_i as a normal random variable $\hat{y}_{i,\ell} \sim \mathcal{N}(y_{i,\ell}, \delta^2_{i,\ell})$. That is, we model the discipline outputs as a normal random variable centred on its discipline output and use its error as the standard deviation. In reality, many of the errors are non-normal and possibly asymmetric. For example, the discretization error in CFD is typically biased such that there is significant skewness to the distribution. This has been demonstrated at the Drag Prediction Workshop [201], where the drag coefficient computed by all participating solvers were systematically over-predicted on coarser grids. Of course, if we knew the actual probability distribution function (PDF), we would use that directly instead. A normal distribution is a reasonable approximation in the absence of such information.

[201]: Levy et al. (2013), *Summary of data from the fifth AIAA CFD drag prediction workshop*

After we model the joint probability density function of all the disciplines' outputs, we must propagate them to the objective and constraints. Fortunately, these system-level outputs are typically composed of simple analytic expressions. For example, a common objective used in aerostructural optimization is fuel burn, which is given by

$$W_{\text{FB}} = m \exp\left(\frac{R\, c_T}{V(L/D)} \right), \tag{5.2}$$

where lift L, drag D, and mass m are the inputs. The range R, cruise speed V, and thrust-specific fuel consumption (TSFC) c_T are constants that remain fixed throughout the optimization.

In this sense, discipline outputs become inputs to these analytic functions that are used to compute the objective and constraints. These discipline outputs are now modeled as normal random variables through the error quantification process. As a result, the system-level outputs, such as fuel burn, also become random variables. We compute the standard deviation of these system-level outputs and use that as the error metric representing the errors introduced by discipline errors. We label system-level errors as $\epsilon_{j,\ell}$ to distinguish them from discipline-level errors $\delta_{i,\ell}$, and use j to index the system-level outputs. In particular, we

use $j = 0$ denote the objective, and number the constraints as $j \in \{1 \dots n_{\text{con}}\}$.

Since the functions of interest are analytic and relatively simple, we use the Monte Carlo method to perform the error propagation. We use an adaptive, parallel implementation, where batches of samples are drawn from the joint probability distribution of the discipline outputs $\boldsymbol{y}$, and the resulting mean and standard deviation are monitored. We terminate the Monte Carlo process when the mean and standard deviation changes are smaller than a prescribed absolute or relative tolerance. In this work, we use batches of 10 000 samples and tolerances of 10^{-3}, which results in around 10^6 samples on average. For a more detailed description of the techniques used in error propagation, see appendix B.

Once we have the errors for objective and constraints for each fidelity combination, we can again apply a Pareto filter to remove certain fidelity combinations, as we did with discipline outputs. As before, we can filter out a fidelity combination if it is non-optimal for a single output, or *all* outputs. In this case we choose the latter option, in order to reduce the number of possible fidelity combinations under consideration.

In reality, these discipline outputs are often correlated and not independent. This is manifested in two ways. First, outputs computed by the same analysis can be correlated, which we call *intradisciplinary errors*. For example, because lift and drag are both functionals computed from the same converged states, a CFD simulation with a significant error in the lift is also more likely to have a more substantial error in drag. Second, outputs computed from different discipline analyses but coupled via MDA would be correlated as well, leading to *interdisciplinary errors*.[3] Taking the aerostructural example, if a CFD simulation had a significant error in the lift, the converged aeroelastic solution would have a different shape. Consequently, the stress distributions on the wing structures would be wrong.

3: Also known as coupled errors. We will use the two interchangeably.

To highlight this further, consider a two-point aerostructural problem with the following discipline outputs:[4]

- Lift L_{cr} and drag D_{cr} at cruise
- Lift L_{man} at maneuver
- Aggregated stress constraints $KS_{1,2,3,\text{man}}$ at maneuver

We show the structure of the discipline outputs computed at the cruise and maneuver points in figure 5.3, where the different types of correlations are colour-coded. The diagonal entries in green are self-correlations and are unity by definition. The correlations between outputs computed by the same analysis are shown in blue, and the correlations resulting from the coupled MDA are shown in red.

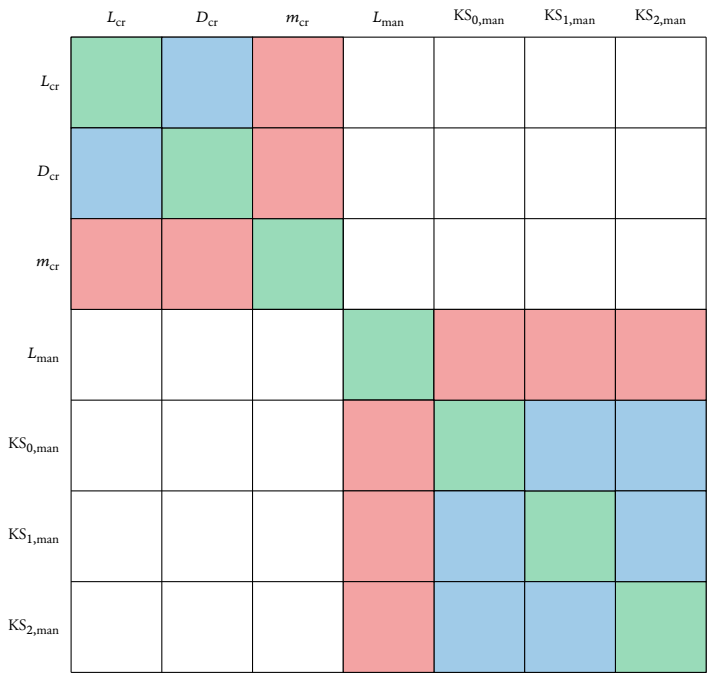


Figure 5.3: Correlation matrix for a two-point aerostructural problem, where all correlations are taken into account. Blue entries indicate correlations between outputs computed by the same analysis, and red entries indicate correlation as a result of the coupled MDA.

To capture the first effect, we extend the probability distribution to a multivariate normal distribution with correlations between specific output quantities. For a given fidelity ℓ, the multivariate normal distribution used is

$$\hat{\boldsymbol{y}} \sim \mathcal{N}(\boldsymbol{y}, \boldsymbol{\Sigma}), \tag{5.3}$$

4: This is the same problem that is considered later on in section 7.1

where the covariance matrix $\boldsymbol{\Sigma}$ is given by

$$\boldsymbol{\Sigma}_{i,j} = \rho_{i,j}\delta_i\delta_j. \tag{5.4}$$

By definition, $\rho_{i,i} = 1.0$ since each output is perfectly correlated with itself. For outputs computed from the same analysis, we use a fixed value of 0.9 for $\rho_{i,j}$ throughout this work. We selected a high correlation coefficient because these outputs are computed from the same set of converged state variables. However, we do not attempt to estimate this quantity since doing so would require many function evaluations that could instead be spent on optimization.

The treatment of the second type of coupling errors is discussed in more detail in chapter 6.

5.2.3 Fidelity selection

Finally, we construct a single scalar error metric for each fidelity, combining the objective and constraint errors and the cost of evaluating the model. Unlike common criteria used in Bayesian optimization, this metric is not used to select the next sampling point, only the fidelity of the next sub-optimization.

For a given fidelity ℓ, we first combine the error contributions from the objective and constraints with

$$\epsilon_\ell = \epsilon_{0,\ell} + \sum_{j=1}^{n_{\text{con}}} |\lambda_j| \epsilon_{j,\ell}, \tag{5.5}$$

where λ_j are the Lagrange multipliers associated with constraint j, that are accessed directly from the optimizer.

As mentioned in section 2.1, the Lagrange multipliers carry physical significance. Note the similarity in form between equations 2.5 and 5.5. In this case, they provide a natural scaling

factor when combined with the objective, such that more influential constraints have their errors weighed more heavily, and inactive constraints are automatically ignored. This is similar to the approach by Chen and Fidkowski [202] in combining errors in the objective and constraints when performing output-based mesh adaptation. We take the absolute value of the Lagrange multipliers because for equality constraints, the sign indicates whether a shift in the constraint causes a positive or negative change in the optimum. However, a change in either direction should be considered equally, and the scaling factors used to sum the errors should be nonnegative.

[202]: Chen et al. (2019), *Discretization Error Control for Constrained Aerodynamic Shape Optimization*

Lastly, we compute the error reduction $\epsilon_\ell^{\text{red}}$ relative to the current fidelity, given by

$$\epsilon_\ell^{\text{red}} = \epsilon_{\text{current}} - \epsilon_\ell. \tag{5.6}$$

This gives us a measure of how much error we expect to reduce if we switch to each of the possible fidelity combinations. We also normalize $\epsilon_\ell^{\text{red}}$ by the cost c_ℓ of each fidelity, to give us the normalized error reduction

$$\hat{\epsilon}_\ell^{\text{red}} = \frac{\epsilon_\ell^{\text{red}}}{c_\ell}. \tag{5.7}$$

The fidelity that has the highest normalized error reduction is chosen as the next fidelity combination ℓ_{next} for optimization:

$$\ell_{\text{next}} = \arg\max_\ell \hat{\epsilon}_\ell^{\text{red}}. \tag{5.8}$$

By construction, the high-fidelity combination yields zero expected error reduction. This ensures that the algorithm will always eventually select the high-fidelity combination, which will be the final optimization.

Once the subsequent fidelity is chosen, a new optimization is performed, after which the fidelity is selected again. This process is repeated until we arrive at the high-fidelity optimum.

5.2.4 Switching criteria

In addition to selecting the next fidelity to use, we also need to determine when to switch to the updated fidelity. This is accomplished using optimization tolerances to terminate the current optimization problem and trigger an update to the fidelity selected. Because we use gradient-based optimizers, the proposed criteria are based on the KKT conditions, which are the necessary conditions for the solution of a nonlinear optimization problem. The implementation of these conditions typically involves feasibility and optimality tolerances.

It is not necessary to fully converge each sub-optimization since the lower-fidelity optima may not be close to the high-fidelity optimum. Instead, we wish to terminate each sub-optimization at an appropriate point depending on the level of fidelity used—typically earlier for a lower-fidelity model. Therefore, instead of using a fixed tolerance for all sub-optimizations, we developed switching criteria that use the available error estimates of each system-level output (objective and constraints). This allows us to achieve a similar level of convergence for each fidelity relative to the accuracy of the analyses.

The first is an adaptation of the feasibility tolerance based on constraint violation $v_i(x)$, defined as

$$v_i(x) = \max\left\{0, g_i(x) - g_{U,i}, g_{L,i} - g_i(x)\right\}, \tag{5.9}$$

where $g_i(x)$ are the constraints, and $g_{U,i}$ and $g_{L,i}$ are the upper and lower bounds for each constraint.

For each constraint, we require the violation to be less than a factor τ_{fea} of the error in predicting that constraint. Recall that we denote the error in the output i for the current fidelity ℓ as $\epsilon_{i,\ell}$, which is computed from Monte Carlo error propagation. We place error bounds along each constraint curve, representing the uncertainty in computing these constraints due to the low-fidelity model. Figure 5.4 shows this graphically for an equality

constraint. In the case of an inequality constraint, only the single dashed line within the infeasible region would be present. If every constraint lies within the prescribed error bound, then we have met the feasibility criterion. Mathematically, we formulate the feasibility criteria as

$$\max_i \left\{ \frac{v_i(x)}{\epsilon_{i,\ell}} \right\} \leq \tau_{\text{fea}}. \tag{5.10}$$

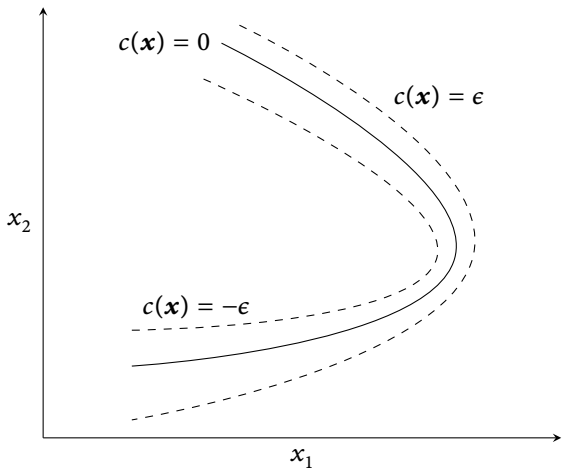


Figure 5.4: An equality constraint curve within a 2-dimensional design space. The dashed lines represent the error bound on either side of the constraint, showing the effective feasible region in between.

The second criterion is an optimality condition, typically based on the objective and constraint gradients. It would be straightforward to apply the same approach as before to the optimality criterion. However, since it is a first-order condition, error estimates of the gradients are needed for each fidelity. Unfortunately, estimating these errors is costly, as it requires adjoint solutions for every output and every existing fidelity. Instead, we replace the optimality condition with a sufficient decrease condition. We monitor the objective decrease between successive major iterations during optimization. We satisfy the optimality criterion when the decrease in the objective Δf, scaled by the error in the objective, is below a prescribed optimality tolerance τ_{opt}. Mathematically, this criterion is

$$\frac{\Delta f}{\epsilon_{0,\ell}} \leq \tau_{\text{opt}}. \tag{5.11}$$

While the values of τ_{fea} and τ_{opt} would affect the timing of each sub-optimization and alter their terminations, we have not found them to cause a large impact on the overall performance of the approach for the aerostructural optimization demonstrated here. Therefore, in this work we use $\tau_{\text{opt}} = \tau_{\text{fea}} = 10^{-2}$, but these values may need to be tuned for other optimization problems. These switching criteria are applied to every optimization except the final, high-fidelity optimization. In that case, the errors are zero by definition, and the error-based criteria would never be met. Instead, we must revert to using the termination criteria within the optimizer, as done for the single-fidelity optimizations. This ensures that for the final optimization, the optimum found is computed using the high-fidelity model and with a tight convergence tolerance. For a unimodal problem, this would guarantee convergence to the same high-fidelity optimum.

5.2.5 Optimization

In the context of the multifidelity framework developed, efficiently restarting an optimization is crucial to the overall performance. See section 2.5.7.2 for a description of different restart strategies. As we switch the fidelity from one to another, we must do so without impeding the progress of subsequent optimizations. By transferring these state variables from a lower-fidelity optimization to a higher one, we use the lower-fidelity model to learn about the optimization problem and provide better estimates for these state variables. We no longer simply use low-fidelity models to obtain a better initial guess for the subsequent optimization. Instead, we are using these models to learn about the overall characteristics of the optimization problem so that fewer iterations are required at the latter stages when more expensive fidelities are used. For example, the approximate Hessian provides curvature information with respect to the augmented Lagrangian. If the low-fidelity model is sufficiently similar to the high-fidelity model, the approximate Hessian can be used

to speed up the more expensive optimizations.

Recall from section 2.5.7.2 that there are three restart strategies available in SNOPT. For this work, we use the hot start, where the full state within the optimizer is stored at the end of each sub-optimization and loaded into SNOPT at the start of the next sub-optimization. However, because the fidelity is updated in between optimizations, a discontinuity is introduced in the function values, which may cause optimization difficulties. In particular, if the function value is higher under the updated fidelity, the optimizer may have trouble finding a feasible step during line search and may quit immediately after. To address this, we perform an additional function evaluation at the design where the optimization was paused. This updates the function value to that of the new fidelity but preserves other state variables that we want to re-use from the previous optimization.

5.3 Practical considerations

We discuss a few practical considerations in the context of multifidelity optimizations. These considerations are important when studying industrially-relevant problems, and guide the design of the appropriate fidelity framework.

5.3.1 Load balancing

A fundamental issue with all multifidelity optimizations is load balancing. The type of MDO problems that benefit most from multifidelity methods are those with expensive function evaluations. In practice, this typically means a nonlinear solver executed in parallel within an HPC environment. In multipoint problems, each analysis point typically executes an independent copy of the solver, such that all points are parallelized further to reduce wall time. Figure 5.5 is a schematic showing such a processor layout.

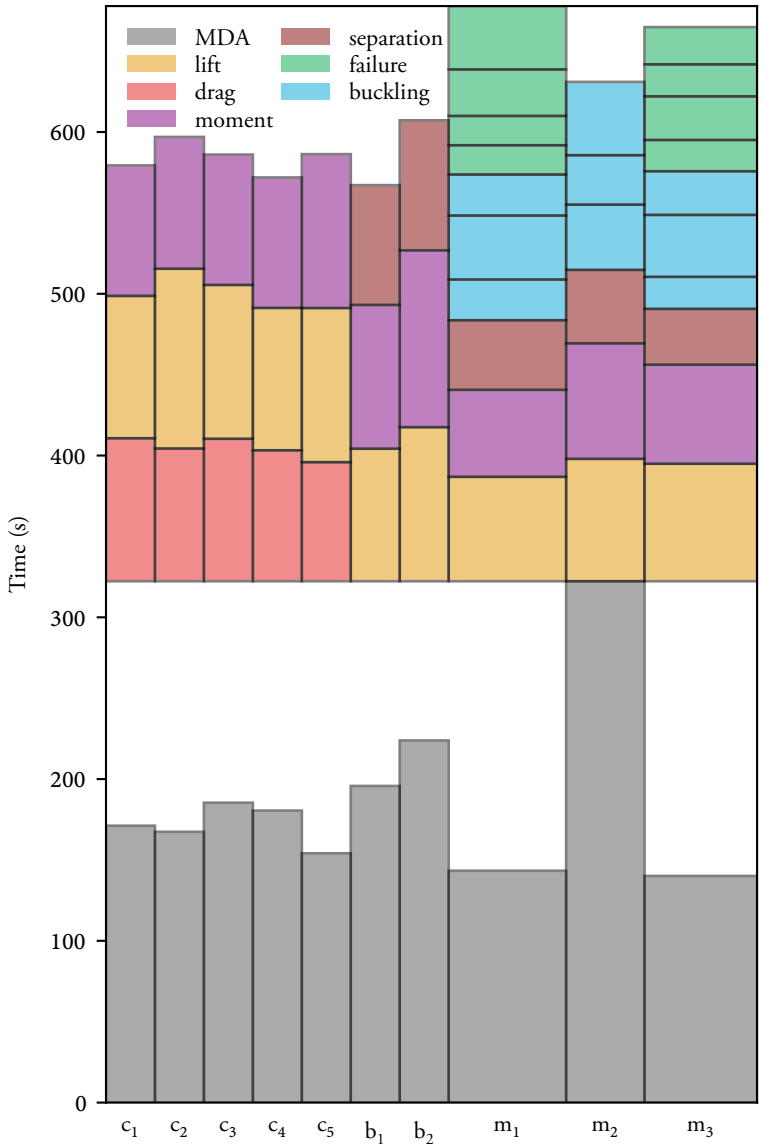


Figure 5.5: An example of a multipoint aerostructural analysis executed in parallel, including both the primal and adjoint stages. The widths of the bars are proportional to the number of processors used for each solver instance.

In such cases, each fidelity will likely require a different parallelization, and therefore a different processor arrangement. It is impractical to design a multifidelity algorithm that switches between multiple fidelities in an *ad hoc* fashion, or at a high frequency. This is due to the design of existing HPC architectures that require compute jobs to be requested for a fixed wall time and processor count. While elastic computing is available on certain cloud platforms such as Amazon Web Services (AWS), they have not been made available on traditional clusters that use job schedulers such as Slurm [203]. Therefore, the only way to perform multifidelity optimization with such approaches would

[203]: Yoo et al. (2003), *Slurm: Simple linux utility for resource management*

be to request the largest number of processors necessary, thereby achieving low processor utilization when running low-fidelity models.

Furthermore, the wall time of each function evaluation can have large variance. Figures 5.6 and 5.7 show the distribution of wall times for the MDA and the coupled adjoint, respectively, during an aerostructural optimization. Each thin vertical line is a single function evaluation, and the large spread means synchronous methods will exhibit even lower parallel utilization.

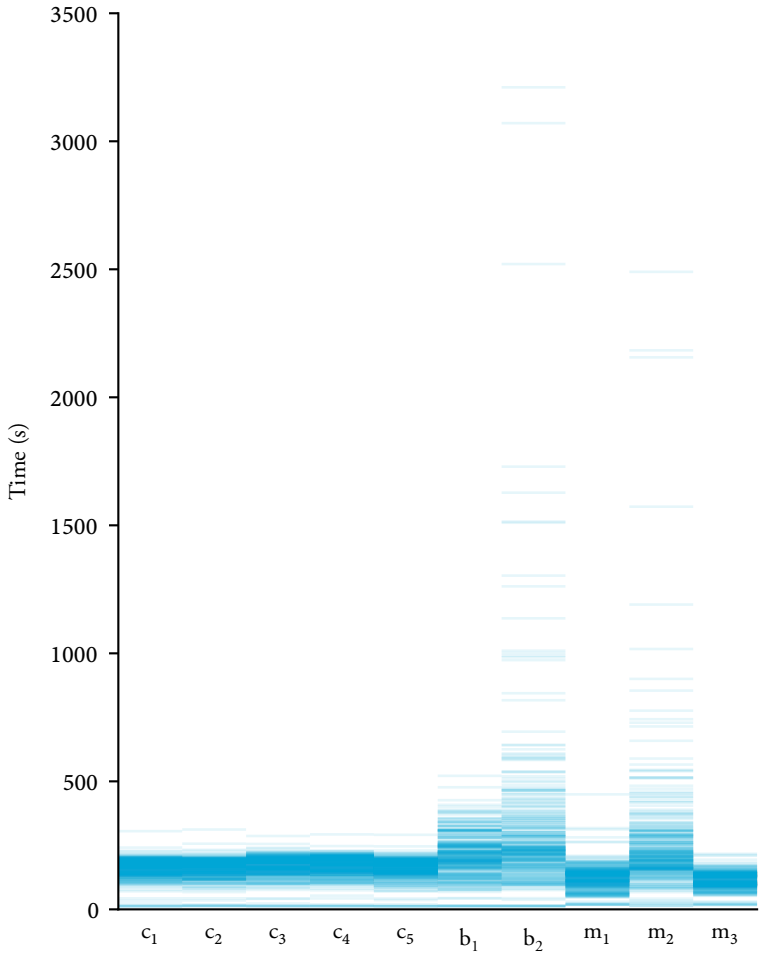


Figure 5.6: Distribution of MDA wall times for a 10-point problem.

Due to these considerations, the proposed appropriate fidelity framework does not switch between fidelities rapidly, nor does it execute multiple fidelities simultaneously in an *ad hoc* fashion. Instead, a strict sequence of fidelities is generated, each from the previous sub-optimization. Therefore, each sub-optimization

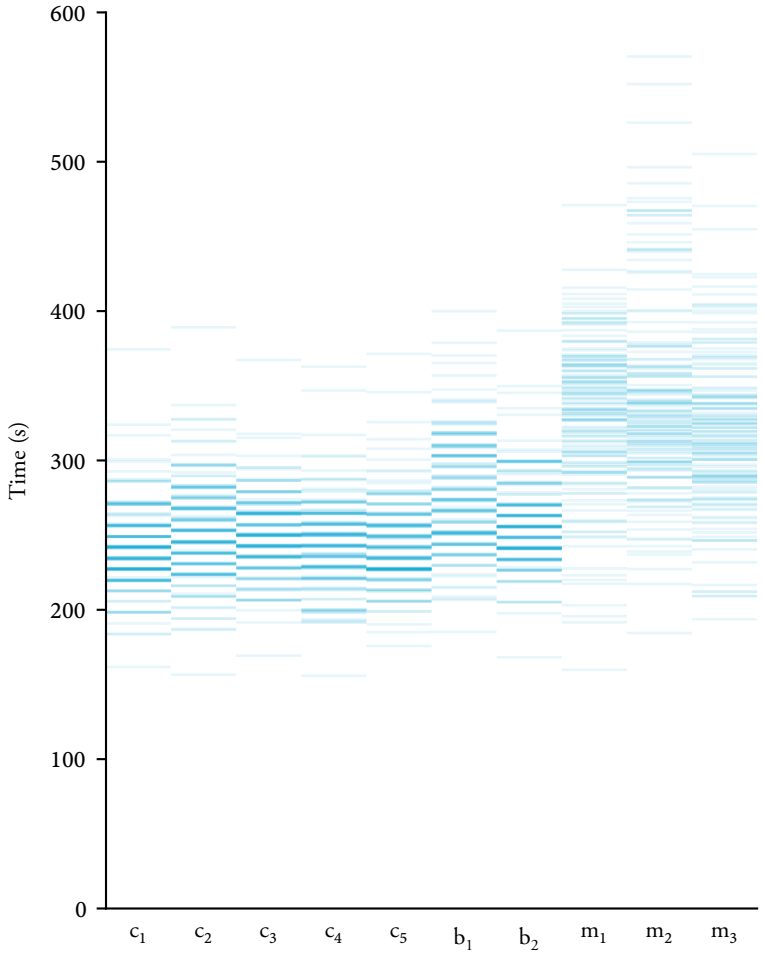


Figure 5.7: Distribution of coupled adjoint wall times for a 10-point problem.

can be executed with a fixed number of processors to maximize cluster utilization.

5.3.2 Summary

A key focus of the appropriate fidelity framework is the quantification of model discrepancy and how it impacts the optimization problem under consideration. The reason for quantifying errors at a discipline level, then propagating them to the system level is that there could be so many possible fidelity combinations that performing an MDA for each would be prohibitively expensive. For example, in the aerostructural test case used in this work, there are 400 combinations for a simple two-point formulation with five aerodynamic and four structural fidelities. An

optimization involving 5 flight conditions would have required $20^5 \approx 3.2 \times 10^6$ MDAs. Instead, we only determine the discipline errors and cheaply propagate them to the system-level outputs. The Monte Carlo evaluations take seconds to complete, compared with the computational time of several hours for an expensive MDA to converge. This approach also provides a well-defined method for selecting the next fidelity in a multidisciplinary context, based on expected error reduction.

Chapter 6

Treatment of Coupled Errors

Recall that the first step in the appropriate fidelity MDO framework from chapter 5 is *error quantification*. In this step, single-disciplinary analyses are performed at the same design point for each fidelity, in order to determine the errors in predicting the disciplinary outputs such as lift and stress. These errors are then modelled as uncertain variables to be propagated to system-level outputs, as discussed in section 5.2.2.

However, two types of couplings exist within these outputs. The first type, *intradisciplinary* errors, are discussed in section 5.2.2. This chapter is concerned with computing the second type, which are coupled errors, also known as interdisciplinary errors (*c.f. intradisciplinary errors* from section 5.2.2). As an example, in the aerostructural case, the aerodynamic fidelity has an impact on the error of the stress predictions, which are purely structural. A low-fidelity aerodynamic model would produce an incorrect lift distribution on the wing, which translates to an incorrect loading on the wingbox, which leads to errors in the stress. It is essential to capture these errors so that the fidelity selection algorithm is not misled.

6.1 Background

Similar topics have been investigated in the field of *uncertainty quantification*, which is concerned with propagating input uncertainties—analogous to errors—through an analysis. The analysis could be

explicit, *e.g.* given by equation 2.12, or implicit and given by the residual form equation 2.13.

However, in this case the inputs $\boldsymbol{x}$ are no longer deterministic. Instead, they are described by a PDF $p(\boldsymbol{x})$, and the goal is to compute the resulting probability distribution on the outputs $\boldsymbol{I}$ which are no longer deterministic.

There are numerous proposed methods for solving this class of problems, such as sensitivity-based approaches (*e.g.* first-order second-moment (FOSM)), sampling-based approaches (*e.g.* Monte Carlo), and other techniques such as polynomial chaos. A brief overview of some simple methods is presented in appendix B.

More recently, there has been some work in extending this problem to the case of coupled, multidisciplinary analyses. As before, we introduce uncertainty in the input parameters $\boldsymbol{x}$, and seek the resulting probability distribution for outputs $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$. Due to the coupling between the disciplines, this is a challenging problem to tackle. Figure 6.1 shows an example MDA for an aerostructural system.

This is an area of active development, and so far the proposed methods are all computationally expensive. First, the naive approach would be to perform system-level Monte Carlo simulations, analogous to the MDF architecture in MDO. However, this would be prohibitively expensive as it requires each Monte Carlo sample at each iteration to converge the MDA. Ghoreishi and Allaire [204] proposed using adaptive Gibbs sampling together with importance resampling, in an individual discipline feasible (IDF) fashion without needing to converge each MDA. However, the cost is still relatively high, on the order of hundreds to millions of single-disciplinary analyses depending on the acceptable error tolerance. Next, there are several proposed methods based on constructing a surrogate model to approximate the coupling between the various disciplines. Friedman et al. [205] constructed a high-dimensional model representation (HDMR)

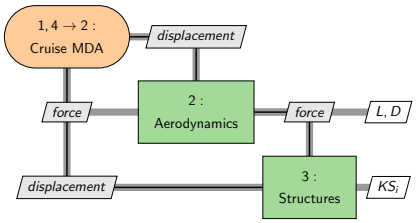


Figure 6.1: An XDSM diagram of the aerostructural MDA, solved using NLBGS iterations.

[204]: Ghoreishi et al. (2017), *Adaptive Uncertainty Propagation for Coupled Multidisciplinary Systems*

[205]: Friedman et al. (2018), *Efficient Decoupling of Multiphysics Systems for Uncertainty Propagation*

surrogate model for each coupling variable, and used it to predict the effect of coupling on the output probability distribution. Similarly, Chaudhuri, Lam, and Willcox [206] constructed Kriging surrogates, and used an alternate adaptive sampling strategy to ensure convergence while reducing the number of samples. As before, the cost of this approach is prohibitive as it requires thousands of samples to construct an accurate surrogate model. Furthermore, as it requires one surrogate model for each coupling variable, this approach does not scale well when there are large numbers of coupling variables present, which is the case with tightly-coupled engineering applications. For example, with the aerostructural MDO problems used in this work, the coupling variables are mesh-dependent and can be up to several hundred thousand.

[206]: Chaudhuri et al. (2018), *Multifidelity Uncertainty Propagation via Adaptive Surrogates in Coupled Multidisciplinary Systems*

Direct decompositional approaches also exist, which aim to perform analyses at the discipline level without requiring MDAs. Du and Chen [207] proposed an approach similar to IDF, where the uncertainty propagation is performed at the discipline level. However, a system-level sub-optimization is necessary to ensure compatibility between the various disciplines.

[207]: Du et al. (2002), *Efficient Uncertainty Analysis Methods for Multidisciplinary Robust Design*

One important distinction between these problems in uncertainty quantification, and our problem concerning coupled errors, is that the uncertainties in our case do not arise from input parameters $\boldsymbol{x}$. Instead, the uncertainties are the direct result of using a low-fidelity disciplinary analysis model. Previously, we had represented this by modelling the discipline outputs, $\boldsymbol{\mathcal{I}}$ as normally-distributed variables, with the standard deviation equal to its error relative to the high-fidelity value. This does not capture the effect of coupling between the different disciplines. In order to do so, we need to model the coupling variables y_i as random variables as well, and determine the effect of this additional uncertainty on the discipline outputs $\boldsymbol{\mathcal{I}}$. This is a problem that has not been studied in the literature, but shares many similarities with those highlighted earlier where the uncertainties are introduced via input parameters. Due to the multidisciplinary

coupling, the input uncertainties will cause the coupling variables y_i to become nondeterministic also, and the PDF of these coupling variables will also need to be converged as part of the MDA process in order to obtain the resultant PDF on the outputs $\mathcal{I}$.

6.2 Methodology

We seek a method for estimating these coupling errors which is relatively cheap to compute, and scalable to a large number of coupling variables commonly found in aerostructural problems. As we only rely on these error values for selecting the appropriate fidelity, we do not need accurate estimations, and would rather spend the vast majority of the computational power on optimization instead.

We propose to use the FOSM method to estimate the coupled errors, leveraging on the existing adjoint capabilities to compute sensitivities cheaply. Furthermore, we do not consider the fully coupled multidisciplinary system, but instead perform only a single forward pass. This is analogous to performing a single NLBGS iteration without any feedback when converging the MDA. The motivation for this is two-fold. First, the mathematics involved in converging the fully coupled aerostructural system considering coupled errors would be a substantial development, particularly given that prior works are prohibitively expensive. It is not yet clear whether an approach exists that does not involve expensive queries to the single or multi-disciplinary analyses. Furthermore, decoupled approaches have been studied in the context of MDA convergence in the literature, and were shown to be effective [208]. By analyzing the feed-forward system, we still consider a significant portion of the coupling, similar to how a single NLBGS iteration can still yield solutions close to the converged MDA state. However, by decoupling the disciplines, the cost is greatly reduced, and a straightforward application of

[208]: Baptista et al. (2018), *Optimal Approximations of Coupling in Multidisciplinary Models*

the adjoint method allows us to compute the errors at the cost of a single adjoint solution. Since we only require these coupled errors for determining the appropriate fidelities, this approach strikes a good balance between cost and accuracy.

First a quick review of FOSM theory for error propagation. Recall that for an output $\mathcal{I}$ with the input vector x is normally distributed, *i.e.* $\boldsymbol{x} \sim \mathcal{N}(\mu_x, \sigma_x^2)$, the FOSM method gives the estimated variance of the output $\boldsymbol{f}$ as

$$\sigma_{\mathcal{I}}^2 \approx \sum_i \left(\frac{\partial \mathcal{I}}{\partial x_i} \right)^2 \sigma_{x_i}^2. \tag{6.1}$$

See appendix B for an in-depth overview.

We apply this to the single-discipline scenario, taking the coupling variables as the independent variables with an associated uncertainty. To compute the coupled errors, we need to obtain both the derivative and the error of the coupling variables.

Correspondingly, the process is broken down into two steps. First, we quantify the errors on the coupling variables for each discipline and analysis fidelity, in addition to the discipline outputs that were quantified previously. Then, we propagate the errors introduced by the coupling variables into the other discipline in a feed-forward fashion, to determine the coupling errors caused by the low-fidelity analysis of another discipline. This process is explained in detail in the next sections. As before, this process is done just once prior to the start of any optimization, and the errors are assumed to be fixed throughout the optimization.

6.2.1 Coupled error quantification

Here we want to quantify the error in the coupling variables, *i.e.* the difference between the coupling variables y_i of different disciplines after converging the discipline governing equations. For aerodynamics, this would be the differences in the surface

pressure distribution between each fidelity and the high-fidelity analyses. However, in many cases the coupling variables have varying dimensions dependent on the chosen fidelity. For example, if the aerodynamic mesh discretizations are used as different fidelities, then the number of coupling variables (*i. e.* the number of surface cells) will change between the different fidelities. This greatly complicates the quantification process, but luckily a mechanism is already in place to deal with this—the transfer scheme used to map coupling variables from one fidelity to another is already built to handle variable problem dimensions. Therefore, we opt to quantify the coupling errors *after* mapping them to the other discipline, in this case using rigid links [21] as the load and displacement transfer scheme, which are based on earlier works by Martins, Alonso, and Reuther [19] and Brown [121]. This ensures that, no matter which aerodynamic mesh is used, the resultant load is applied to the same structural mesh, and is therefore dimensionally constant.

[21]: Kenway et al. (2014), *Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Adjoint Derivative Computations*

[19]: Martins et al. (2005), *A Coupled-Adjoint Sensitivity Analysis Method for High-Fidelity Aero-Structural Design*

[121]: Brown (1997), *Displacement extrapolations for CFD+CSM aeroelastic analysis*

The overall process is as follows. We perform aerodynamic analyses for each fidelity as before, but in addition to recording the functional outputs (such as lift and drag) we also obtain the resulting pressure distributions $\boldsymbol{p}$. These are converted to structural loads $\boldsymbol{f}$ via the load and displacement transfer scheme, and the loads are recorded. The errors can then be computed by finding the difference between the loads generated by the low and high-fidelity aerodynamic analyses. Note that this needs to be done *for each combination of structural and aerodynamic fidelities*, since we need to consider the combined fidelities used for each MDA. The error $\epsilon_{\boldsymbol{f}}$ on the forces for a given structural fidelity ℓ_S and aerodynamic fidelity ℓ_A is computed as

$$\epsilon_{\boldsymbol{f}}(\ell_S, \ell_A) = \left| \boldsymbol{f}_{\ell_S}(\boldsymbol{p}_{\ell_A}) - \boldsymbol{f}_{\ell_S}(\boldsymbol{p}_0) \right|, \tag{6.2}$$

where $\boldsymbol{f}_{\ell_S}(p_{\ell_A})$ is the force on the structural fidelity ℓ_S computed from the surface pressure $\boldsymbol{p}$ of the aerodynamic fidelity ℓ_A after performing the load transfer. Because we map the coupling variables to the same structural fidelity, the dimensions are consistent

and a simple subtraction is sufficient. Recall that we denote the highest fidelity as $\ell = 0$.

Similarly, for the other direction, we perform structural analyses for the various structural fidelities, then map the structural nodal displacements $\boldsymbol{X}_S$ to a fixed aerodynamic mesh to obtain the aerodynamic surface node displacements $\boldsymbol{X}_A$. The differences in the aerodynamic surface node displacements due to different structural fidelities are then recorded, for each combination of aerodynamic and structural fidelity used. This is taken as the error on the aerodynamic surface node displacements.

$$\epsilon_{\boldsymbol{X}_A}(\ell_S, \ell_A) = \left| \boldsymbol{X}_A^{\ell_A}(\boldsymbol{X}_S^{\ell_S}) - \boldsymbol{X}_A^{\ell_A}(\boldsymbol{X}_S^{0}) \right|, \tag{6.3}$$

where $\boldsymbol{X}_A^{\ell_A}(\boldsymbol{X}_S^{\ell_S})$ are the aerodynamic surface nodes $\boldsymbol{X}_A$ on the aerodynamic fidelity ℓ_A, computed from the structural node displacements $\boldsymbol{X}_S$ on the structural fidelity ℓ_S.

In practice, these errors are never written to disk but instead stored in memory. The overall computational cost of this step is relatively low, since the single-disciplinary analyses were already necessary for quantifying the errors on the discipline outputs. The only additional cost is in performing load and displacement transfer for each combination of aerodynamic and structural fidelity, and in both directions.

6.2.2 Coupled error propagation

Because we had already performed the load and displacement transfer earlier when quantifying the errors, coupled error propagation is a self-contained process within a single discipline.

Let us first look at the structural side, where a QOI $\mathcal{I}$ now has uncertainty due to uncertainties in the coupling variables f_i, which are the structural loads. The resulting variance on $\mathcal{I}$ can

be computed following equation 6.1 as

$$\epsilon_{\mathcal{I}}^2 \approx \sum_i \left(\frac{\mathrm{d}\boldsymbol{\mathcal{I}}}{\mathrm{d}f_i} \right)^2 \epsilon_{f_i}^2. \tag{6.4}$$

Previously we had quantified the error ϵ_{f_i}, so it remains to compute the sensitivity of $\boldsymbol{\mathcal{I}}$ with respect to the forces. We use the adjoint approach to compute the sensitivities efficiently. Starting from equation 2.16, we write the total sensitivity equation with the forces as the design variables:

$$\frac{\mathrm{d}\boldsymbol{\mathcal{I}}}{\mathrm{d}\boldsymbol{f}} = \frac{\partial \boldsymbol{\mathcal{I}}}{\partial \boldsymbol{f}} - \boldsymbol{\psi}_S^\mathsf{T} \frac{\partial \boldsymbol{R}_S}{\partial \boldsymbol{f}}, \tag{6.5}$$

where $\boldsymbol{R}_S$ is the structural residual, and $\boldsymbol{\psi}_S$ is the structural adjoint vector for output $\boldsymbol{\mathcal{I}}$, computed via the solution of 2.15.

Note that this linear system has the well-known property of being independent of the design variables $\boldsymbol{f}$. Therefore only one linear solution is needed for each output $\boldsymbol{\mathcal{I}}$. Given that typically the outputs $\boldsymbol{\mathcal{I}}$ are not explicitly dependent on the coupling variables $\boldsymbol{f}$, equation 6.5 can be further simplified to

$$\frac{\mathrm{d}\boldsymbol{\mathcal{I}}}{\mathrm{d}\boldsymbol{f}} = -\boldsymbol{\psi}_S^\mathsf{T} \frac{\partial \boldsymbol{R}_S}{\partial \boldsymbol{f}}. \tag{6.6}$$

Luckily for the structural analysis, the residual is actually linear in $\boldsymbol{f}$.[1] The residual Jacobian is in fact the negative identity matrix:

1: This is only true for a *linear* structural model, which is the case for this work but not in general.

$$\begin{aligned} \boldsymbol{R}_S &= \mathbf{K}\boldsymbol{u}_S - \boldsymbol{f} \\ \frac{\partial \boldsymbol{R}_S}{\partial \boldsymbol{f}} &= -\mathbf{I}. \end{aligned}$$

Equation 6.6 simplifies to

$$\frac{\mathrm{d}\boldsymbol{\mathcal{I}}}{\mathrm{d}\boldsymbol{f}} = \boldsymbol{\psi}_S, \tag{6.7}$$

and equation 6.4 becomes

$$\epsilon_I^2 \approx \left(\boldsymbol{\psi}_S \odot \boldsymbol{\psi}_S\right)^\mathsf{T} \cdot \left(\epsilon_f \odot \epsilon_f\right). \tag{6.8}$$

Here $\odot$ denotes the Hadamard product, or element-wise multiplication. In terms of the implementation, we simply solve the structural adjoint $\boldsymbol{\psi}_S$ for each output $\boldsymbol{I}$, then perform parallel dot product with the errors on the forces following equation 6.8.

The equations are largely the same for aerodynamics. First, we write out the equation for obtaining the variance following the FOSM method:

$$\epsilon_I^2 \approx \sum_i \left(\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{X}_{A_i}} \right)^2 \epsilon_{\boldsymbol{X}_{A_i}}^2. \tag{6.9}$$

We then expand the total derivative term using the adjoint equations:

$$\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{X}_A} = \frac{\partial \boldsymbol{I}}{\partial \boldsymbol{X}_A} - \boldsymbol{\psi}_A^\mathsf{T} \frac{\partial \boldsymbol{R}_A}{\partial \boldsymbol{X}_A} \tag{6.10}$$

$$= -\boldsymbol{\psi}_A^\mathsf{T} \frac{\partial \boldsymbol{R}_A}{\partial \boldsymbol{X}_A}, \tag{6.11}$$

where $\boldsymbol{\psi}_A$ is the aerodynamic adjoint vector for output $\boldsymbol{I}$.

But unlike before, the residual Jacobian cannot be simplified further since the aerodynamic residuals are not linear with respect to the coupling variables. Nevertheless, we do not form the residual Jacobian explicitly, since it can be prohibitively expensive to compute and store in memory. Instead, we use matrix-free techniques to do this without explicitly computing or storing the Jacobian matrix. By using the existing AD code within ADflow, we can seed the reverse-mode AD routines with the corresponding vector—in this case the adjoint vector—and obtain the matrix-vector product in a single reverse pass. This process is also done in parallel, so each processor only has the portion of the matrix-vector product $\boldsymbol{\psi}_A^\mathsf{T} \, \partial \boldsymbol{R}_A / \partial \boldsymbol{X}_A$ local to itself.

The cost of this step is an additional single-disciplinary adjoint solution for each discipline and fidelity available, as the extra

linear algebra computations are negligible in comparison. Compared with methods in the literature that require hundreds or thousands of single-disciplinary analyses *for each fidelity*, the added cost of only a single adjoint solution is a significant improvement.

6.3 Verification

We perform verification tests to ensure that the implementation is numerically correct, by comparing the computed errors using the proposed method against brute-force Monte Carlo simulations. We draw independent samples of the coupling variables according to their normal distributions, and perform single-discipline analyses for each sample. The computed discipline outputs are then recorded, and the resulting standard deviation is compared against the outcome of the proposed adjoint-based method. Note that this verification process is numerically consistent with the forward-pass approach, since we are performing Monte Carlo simulations over single-disciplinary analyses. As such, the Monte Carlo results account for the multidisciplinary coupling in the same way—as a single forward pass, rather than the fully-coupled approach. For all verification tests, a total of 10^4 samples are used.

First we examine the structural discipline, in this case the errors on the `O2L3` structural fidelity caused by the `R2` aerodynamic fidelity.[2] As shown in table 6.1, we have excellent agreement between the two approaches, with relative errors at less than 0.5%. This is expected, as the structural analysis is linear so the linearization used by the FOSM approach should work very well.

In particular, we note that analytically, the errors on the mass estimations should be exactly zero since those values only depend on the design variables, and not on the coupling variables at

2: The structural fidelities are named `O#L#`, where the first number denotes the order of the elements, and the second number denotes the mesh level. Similarly, the aerodynamic fidelities begin with either `R` for RANS or `E` for Euler, followed by a number denoting the mesh level. The mesh level is always finer for smaller numbers, with 0 typically denoting the finest mesh available.

	Adjoint	Monte Carlo	Relative error (%)
m_{cr}	0.0000	1.80×10^{-12}	0.00
m_{man}	2.80×10^{-11}	1.80×10^{-12}	2.80×10^{-11}
KS_0	0.0313	0.0313	0.01
KS_1	0.0340	0.0340	0.20
KS_2	0.0357	0.0355	0.50

Table 6.1: The errors computed via Monte Carlo on the 02L3 structural fidelity due to the R2 aerodynamic fidelity show excellent agreement with the adjoint approach. Since the correct reference values for the mass errors are zero, absolute errors are shown for those two instead of relative errors

all. Because of this, the term $\frac{\partial \mathcal{I}}{\partial u}$ on the right hand side of equation 2.15 is a vector of zeros, so the adjoint vector and therefore the resulting errors are zero as well. The small values obtained for the maneuver mass are due to finite precision arithmetic and the poor conditioning of the structural system, and can be considered to be numerically zero. Because the mathematical relations between the various quantities are automatically captured in the adjoint approach, no special treatment is required for this case.

Next we perform the same verification, but on a lower-fidelity aerodynamic model to show that we indeed obtain larger errors on the structural outputs as a result. We use the 02L2 structural fidelity, together with the E1 aerodynamic fidelity. As seen in table 6.2, we do indeed obtain larger errors. The differences between the Monte Carlo simulations and the adjoint approach are also larger at between 1% and 4%, but still in very good agreement.

	Adjoint	Monte Carlo	Relative error (%)
m_{cr}	0.0000	1.80×10^{-12}	0.00
m_{man}	5.10×10^{-11}	1.80×10^{-12}	5.10×10^{-11}
KS_0	0.0454	0.0459	1.09
KS_1	0.0483	0.0488	1.01
KS_2	0.0541	0.0519	4.24

Table 6.2: The errors computed on the 02L2 structural fidelity due to the E1 aerodynamic fidelity are larger, since we used a lower-fidelity aerodynamic model.

Finally, we examine the same process on the aerodynamic side. We look at the error on the E1 aerodynamic fidelity, caused by the 02L3 structural fidelity. Here, we do not expect similar levels of agreement since the aerodynamic analysis is no longer linear. Since we linearize the residuals about the analysis point, the residual Jacobian $\partial \boldsymbol{R} / \partial \boldsymbol{X}_A$ will not be able to capture any nonlinear effects due to the coupling variables. As a

result, we observe errors of up to 67% for the cruise outputs, as listed in table 6.3. We still believe these are acceptable, since a rough order-of-magnitude estimate would likely be sufficient for fidelity selection. Furthermore, we used a fixed number of samples for the Monte Carlo simulations and did not monitor the convergence, so it is possible that the reference values are under-converged.

	Adjoint	Monte Carlo	Relative error (%)
L_{cr}	7293	4367	67
D_{cr}	387	303	27
L_{man}	36 061	205 044	—

Table 6.3: The errors computed on the `E1` aerodynamic fidelity due to the `02L3` structural fidelity.

However, for the maneuver case we encountered another difficulty during the verification process. A large portion of the Monte Carlo simulations actually failed to converge due to mesh warping failures, due to the randomly-generated samples which are not always physically valid.[3] These random surface mesh displacements can cause the volume mesh to collapse onto itself or even invert certain cells to create negative cell volumes, causing the analysis to fail. Figure 6.2 shows an example of a failed mesh during volume mesh warping. This issue did not show up for the cruise case as the errors were smaller in magnitude. It is worth noting here that since the adjoint approach requires a single adjoint solution at the design point, it is guaranteed to converge if the primal solution converged, and therefore does not suffer from this problem.

3: Realistically, the errors in the displacements are not in fact independent, and an approach similar to section 5.2.2 (with an assumed correlation coefficient ρ used to compute a covariance matrix) can be used to reduce the chance of mesh warping failure.

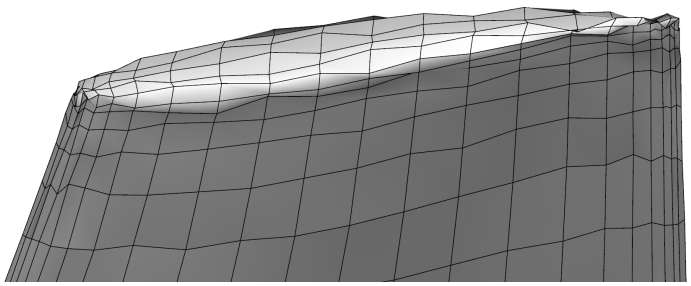


Figure 6.2: An example of a mesh failure due to negative cell volumes during volume mesh warping.

In practice, the errors are not uncorrelated as assumed. If the displacement was under-predicted at one surface node, it is likely

that the same is true for the surrounding nodes. Therefore, a more realistic approach would be to assume a large correlation coefficient for the covariances of the coupling variables, as done in section 5.2.2. This would improve both the error propagation, and increase the robustness of the Monte Carlo simulations.

6.4 Incorporation into framework

The coupled errors are computed prior to any optimization, and saved to disk. They are then used just prior to the error propagation step (discussed in section 5.2.2, not the coupled error propagation step mentioned in section 6.2.2), where they are added to the single-disciplinary error values quantified earlier. In essence, we have:

$$\delta_{i,\ell} = |y_{i,\ell} - y_{i,0}| + \epsilon_{i,\text{coupled}}. \tag{6.12}$$

In this way, the errors are now dependent not only on the fidelity of each discipline, but the fidelity of the other disciplines as well. The combined values are then passed through the error propagation step, either using Monte Carlo or FOSM methods, to obtain the error values on the system-level objective and constraints. Those error values are then used to guide the fidelity selection process.

6.5 Summary

In this chapter, we developed a methodology to effectively capture coupled errors for an aerostructural system, and implemented it within MACH. The proposed method is based on FOSM, where the sensitivities are provided using the adjoint method. By using a feed-forward approach, we could effectively decouple the disciplines while accounting for some coupled interactions,

leading to a much more cost-effective method. The method was implemented in an efficient manner, taking advantage of existing AD routines which were able to assemble and compute Jacobian-vector products in a matrix-free manner, and was verified against brute-force Monte Carlo simulations. The effect of coupled errors is further analyzed in section 7.1.3, demonstrating the importance of capturing such errors in multidisciplinary systems.

Chapter 7

Appropriate Fidelity Optimization Results

In this chapter, we demonstrate the appropriate fidelity MDO framework on two sets of aerostructural optimizations. The first is the standalone wing from chapter 3 and chapter 4, but with an added structural model. While it is not a large-scale problem, we show that the framework is able to handle fidelities arising from not just discretization, but simplified physics. We also present results to illustrate the impact of considering coupled errors in a multidisciplinary system. The second optimization involves the XRF1 aircraft configuration, and is a four-point aerostructural problem consisting of over 900 design variables and 900 constraints. In both cases, significant computational savings were obtained compared to the reference high-fidelity optimization.

7.1 Demonstration on a standalone wing

7.1.1 Benchmark problem

We use a multifidelity wing optimization problem as the benchmark. It is a two-point aerostructural optimization, where the aerodynamics and structure are analyzed using various fidelities during optimization. The geometry consists of a single swept wing with an embedded wingbox, shown in figure 7.1. This is the same outer mould line (OML) as the problems exampled in chapters 3 and 4, but extended to include a structural wingbox

model. The wingbox is made of 2024 aluminum; the material properties are listed in table 7.2.

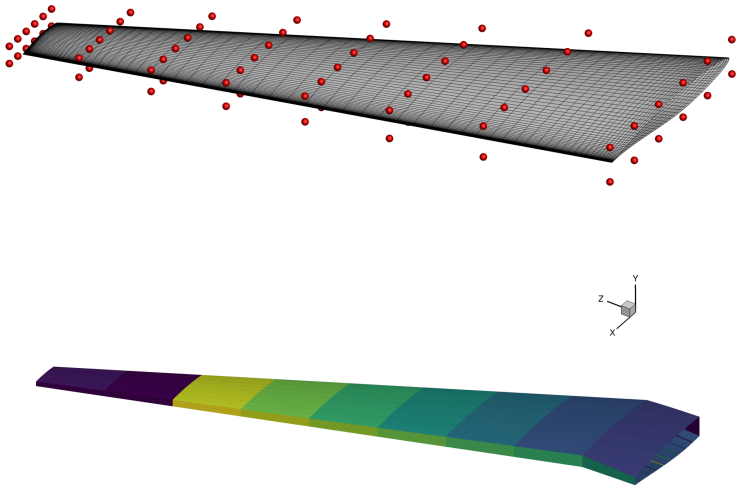


Figure 7.1: Geometric design variables are shown in the upper figure. Each red node indicates an FFD control point, and groups of nodes are manipulated together to form geometric design variables. Structural design variables are visualized in the lower figure, where each design variable controls the panel thickness of a distinct region of the wingbox.

We use the cruise flight condition to compute the wing's performance and the 2.5 g maneuver flight condition to size the structure. These flight conditions are listed in table 7.1. The design variables consist of two angles of attack, one for each flight condition and seven sectional twist variables that simultaneously warp the CFD and computational structural mechanics (CSM) meshes. However, the root twist is not a design variable because we have separate angles of attack for trimming the aircraft. There are also 108 panel thickness variables for the different wingbox components. These geometric and structural design variables are shown in figure 7.1.

The objective of the optimization is to minimize the fuel burn W_{FB} given by equation 5.2. The drag D and lift L are computed from the cruise point, and the flow speed is computed from the cruise Mach number and altitude. The range R and TSFC are given as constants, and the mass m is computed from

$$m = 2 \times m_{\text{struct}} + m_{\text{extra}}, \tag{7.1}$$

Table 7.1: Operating conditions for the cruise and maneuver points.

	Cruise	Maneuver
Altitude (m)	10 000	5000
Mach number	0.8	0.75
Load factor n_i	1.0	2.5

where m_{struct} is the structural mass of the wingbox as computed by CSM, and m_{extra} is a constant value meant to emulate the fuselage and other mass not accounted for in the structural model. All the relevant constants are listed in table 7.2.

There are also several constraints in this problem. We constrain the lift and weight for both the cruise and maneuver flight conditions, taking the load condition into account. We also have manufacturing constraints for the panel thickness variables, such that the difference in thickness between adjacent panels is less than 2.5 mm. These are sparse linear constraints that can be satisfied easily by the optimizer. Of course, yield stress constraints are enforced at the maneuver flight condition, using the von Mises failure criterion and a safety factor of 1.5. These constraints are aggregated using the Kreisselmeier–Steinhauser (KS) function [209], ultimately resulting in three constraints: one each for the ribs and spars, upper skin and stringers, and lower skin and stringers. The entire optimization problem formulation is summarized in table 7.3.

Table 7.2: Reference values for the wing-only test case.

Description	Values
Range	10^4 km
Thrust-specific fuel consumption	0.53 lb/(lbf h)
Extra mass	4×10^4 kg
Material density	2780 kg/m^3
Young's modulus	73.1 GPa
Yield stress	324 MPa
Poisson's ratio	0.33

[209]: Kreisselmeier et al. (1979), *Systematic Control Design by Optimizing a Vector Performance Index*

Table 7.3: Optimization problem formulation for the aerostructural wing-only problem.

	Function/variable	Description	Quantity
minimize	W_{FB}	Fuel burn	1
with respect to	x_{twist}	Section twist	7
	x_{alpha}	Angle of attack	2
	x_{struct}	Panel thickness	108
		Total design variables	**117**
subject to	$L = n_i W$	Lift constraint	2
	$\lvert x_{\text{struct},i} - x_{\text{struct},i+1} \rvert \leq 2.5$ mm	Adjacency constraint	72
	$\text{KS}_i \leq 1.0$	Yield stress	3
		Total constraints	77

There are several fidelities available for the aerodynamic and structural disciplines. For aerodynamics, we use both RANS with the SA turbulence model and Euler solutions with a skin-friction correction. In addition, there are different discretizations available: three meshes for RANS and two for Euler, for a total of five fidelities. For structures, there are two discretizations and two finite-element solution orders possible for a total of four fidelities. In total, there are 20 possible aerostructural fidelity

combinations for each operating point, giving a total of 400 choices. This shows that even for a small MDO problem, the number of possible fidelity combinations grows quickly. A rigorous and scalable fidelity management framework is needed to handle this increasing complexity.

The available fidelities are listed in table 7.4. For aerodynamic fidelities, `E` represents Euler simulations and `R` represents RANS. For structural fidelities, the `O` and the following number represent the structural finite-element analysis order. In both cases, the final number represents the mesh level, where increasing numbers correspond to coarser meshes.

Table 7.4: Fidelities available for aerodynamic and structural analyses, together with the number of DOFs and computational cost.

Fidelity	DOF	Cost (proc-hours)	
		Cruise	Maneuver
`E1`	14 560	0.0172	0.0009
`E0`	116 480	0.1926	0.0659
`R2`	24 192	0.0286	0.0234
`R1`	193 536	0.2290	0.2451
`R0`	1 548 288	3.3748	3.6901
`O2L0`	179 628	0.0058	0.0061
`O3L0`	725 484	0.0596	0.0653
`O2L1`	44 076	0.0006	0.0006
`O3L1`	179 628	0.0062	0.0062

7.1.2 Results and discussion

We perform two optimizations: the first uses the appropriate fidelity approach, and the second uses only the high-fidelity model. Note that the appropriate fidelity optimization presented here does *not* account for the coupled errors in the manner described in chapter 6. That optimization is presented in section 7.1.3, together with further analysis.

First, we examine the multifidelity results. The process begins with the initial error quantification phase, where we perform single-discipline evaluations to determine the errors in computing discipline outputs. For each of these outputs, we can generate a scatter plot of cost against error for each output and each fidelity shown in figures 7.2 and 7.3.

We make several observations here. First, not all available fidelities are useful fidelities. For example, the fine Euler fidelity `E0` is not Pareto-optimal for computing any aerodynamic quantities and is dominated by `R2`, the coarse RANS solution. Second, the Pareto-optimality of a fidelity depends on the QOI. For structures, the fidelity `O3L1` is not Pareto-optimal for the KS stress constraints but is optimal for computing the structural mass. In the current framework, we still filter out such fidelities as

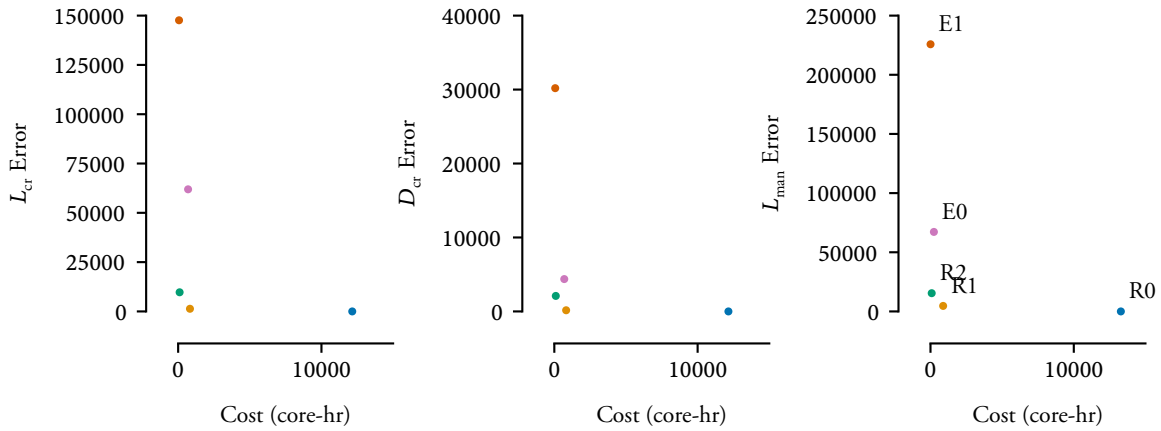


Figure 7.2: Pareto front for aerodynamic fidelities, showing that not all fidelities are optimal.

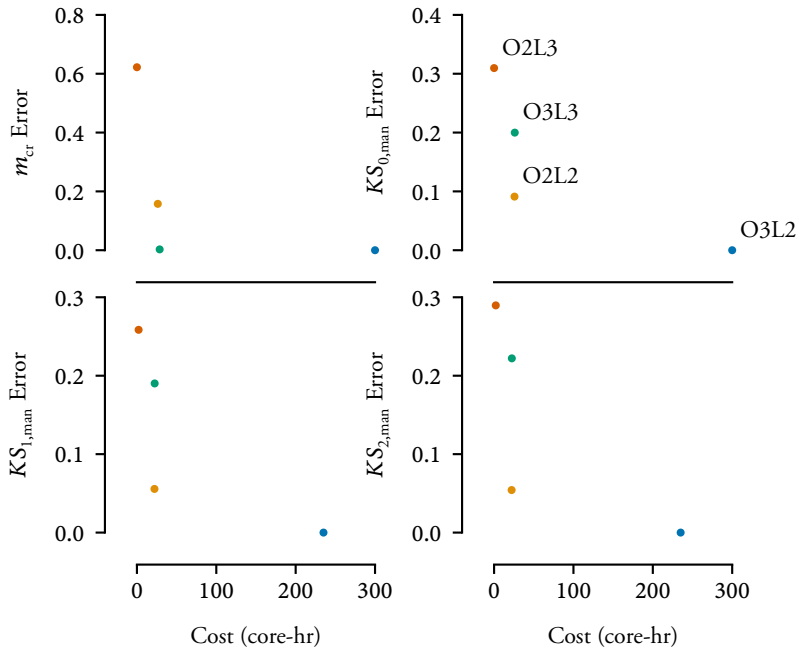


Figure 7.3: Pareto front for structural fidelities, showing that not all fidelities are optimal.

we require each potential fidelity to be Pareto-optimal for all outputs for which it is responsible. Lastly, the scatter plots could be different if analyzed at another design point, resulting in the removal of different fidelities. We do not consider such effects in this work.

After filtering out these non-optimal fidelities, we perform a single-fidelity optimization using the lowest fidelity available. Once terminated via the criteria from section 5.2.4, we propagate the discipline errors to system-level objectives and constraints.

Figure 7.4 shows the Pareto plot of cost against error, analogous to figures 7.2 and 7.3 but for system outputs. Each point corresponds to a fidelity combination, and the error is computed through Monte Carlo simulations. After applying another Pareto filter, we compute the composite metric $\hat{\epsilon}_\ell^{\text{red}}$. Finally, we select the subsequent fidelity based on this metric.

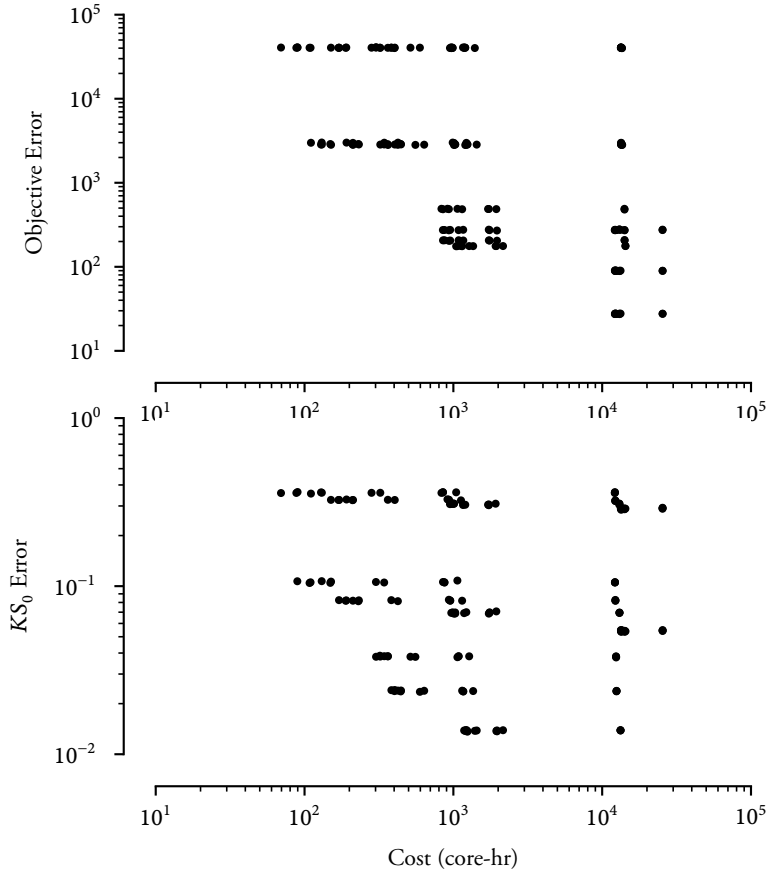


Figure 7.4: Pareto plot of objective errors $\epsilon_{0,\ell}$ against cost for all 400 possible fidelity combinations, showing that many are not Pareto-optimal. The corresponding plot for constraint KS_0 is shown on the right.

A new optimization is then performed, hot-started from the previous one. This process continues until we reach the final, high-fidelity optimization, which is allowed to continue to completion. Table 7.5 lists the sequence of fidelities taken, along with the computational costs. For comparison, we perform a reference optimization starting from the same initial design but using only the high-fidelity models. Overall, the multifidelity approach offered a speedup of 59% compared to the single-fidelity approach, while effectively finding the same numerical optimum. The difference in the objective between the two designs

Table 7.5: Sequence of fidelities for the aerostructural optimization, showing the fidelity used, the computational costs, and the number of major iterations taken. The fidelity combination is represented by four columns corresponding to the four analyses in the two-point aerostructural problem. The cost is given in proc-hours, and the relative cost is normalized by the total cost of the multifidelity approach.

	Cruise		Maneuver				
Number	Aero	Struct	Aero	Struct	Cost	% Cost	Iterations
1	E1	O2L1	E1	O2L1	0.95	0.1	80
2	R2	O2L1	E1	O2L0	12.72	0.7	43
3	R2	O2L1	R2	O2L0	0.53	0.0	16
4	R1	O2L1	R2	O2L0	0.53	0.0	14
5	R1	O2L1	R2	O3L0	1.22	0.1	33
6	R1	O2L1	R1	O3L0	32.03	1.8	35
7	R0	O3L1	R1	O3L0	275.01	15.4	15
8	R0	O3L1	R0	O3L0	643.70	36.0	16
9	R0	O3L0	R0	O3L0	821.08	45.9	26
Total					1787.79	100.0	278
High-fidelity	R0	O3L0	R0	O3L0	4379.05	244.9	116

is 2×10^{-3} kg, for a relative difference of 4×10^{-8}.

A total of nine sub-optimizations were taken, each one hot-started from the previous optimization. As expected, the early, low-fidelity optimizations took many iterations to build up a more accurate approximate Hessian. In contrast, the later optimizations converged in far fewer iterations. As a result, although the number of major iterations was twice as much as the single-fidelity optimization, the overall computational cost was far lower. This also demonstrated the robustness of the hot-start approach that enabled us to perform nine sub-optimizations without losing progress in between.

Looking at the sequence of fidelities chosen, we see that the algorithm preferred to improve the cruise aerodynamics fidelity more than the maneuver analysis. This preference makes sense because both lift and drag values are needed from the cruise point, but only lift is needed for maneuver. Because the error in the lift is typically higher than the error in the drag for a given fidelity, it was more important to improve the cruise aerodynamic fidelity. This is commonly done in single, high-fidelity optimizations, where the maneuver aerodynamics is often analyzed using a lower-fidelity model, such as by using a coarser

grid [210]. Furthermore, structural fidelity is more important for the maneuver point than cruise because it computes all the stress constraints. Naturally, the fidelity selection algorithm improves the maneuver structural fidelity more quickly than the cruise counterpart.

[210]: Brooks et al. (2017), *High-fidelity Multipoint Aerostructural Optimization of a High Aspect Ratio Tow-steered Composite Wing*

However, some of the selections could be improved, particularly for the cruise structural fidelity. The algorithm waited until the final optimization to switch to the high-fidelity structural model for the cruise analysis. This is because the only output directly computed by the cruise structural solver is the structural mass. The mass is computed accurately for all structural fidelities because it is a simple linear computation. This results in an insignificant contribution in the objective error from the mass computation. These two effects, combined with the relative insensitivity of the fuel burn objective with respect to the structural mass [42], result in the selection algorithm favouring lower-fidelity models for the cruise structural analysis. The error introduced by using a low-fidelity structural model is more than just an inaccurate mass computation. A lower-fidelity structural model would yield inaccurate structural displacements because of the coupled aerostructural analysis. This would result in an inaccurate aeroelastic flying shape and, therefore, inaccurate lift and drag computations. This coupled effect corresponds to the red entries in the correlation matrix shown in figure 5.3, which are not accounted for here. See section 7.1.3 for the subsequent optimization which included these errors.

[42]: Kenway et al. (2014), *Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration*

Now, we examine the optimizations in more detail. First, we plot the intermediate designs at the end of each sub-optimization to show the sequence of optimizations. Figure 7.5 shows the structural panel thicknesses at the end of each sub-optimization. The initial design of uniform thickness is also shown. Similarly, figure 7.6 shows the corresponding structural failure when analyzed with the same fidelity as used in optimization, where a value of 1.0 indicates the yield limit. Despite the loose convergence tolerance of earlier optimizations, their final stress distributions

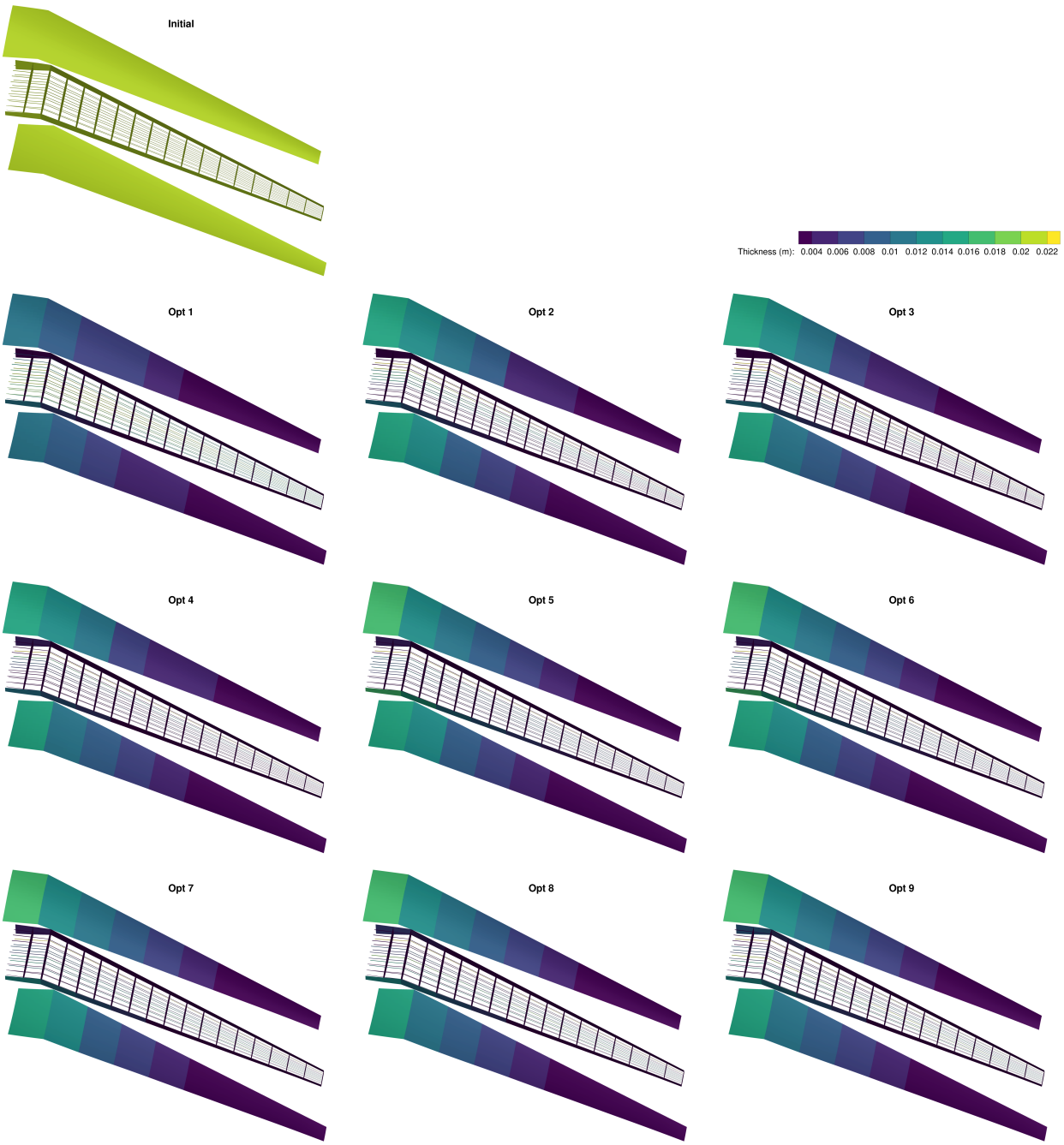


Figure 7.5: Sequence of structural panel thicknesses at the end of each sub-optimization, showing the rapid convergence in the early optimizations.

are still quite close to being optimal, with significant regions of the wingbox close to the yield limit. Since the vast majority of the design variables are these structural thicknesses, their rapid convergence is a good indication of the proposed methodology's effectiveness.

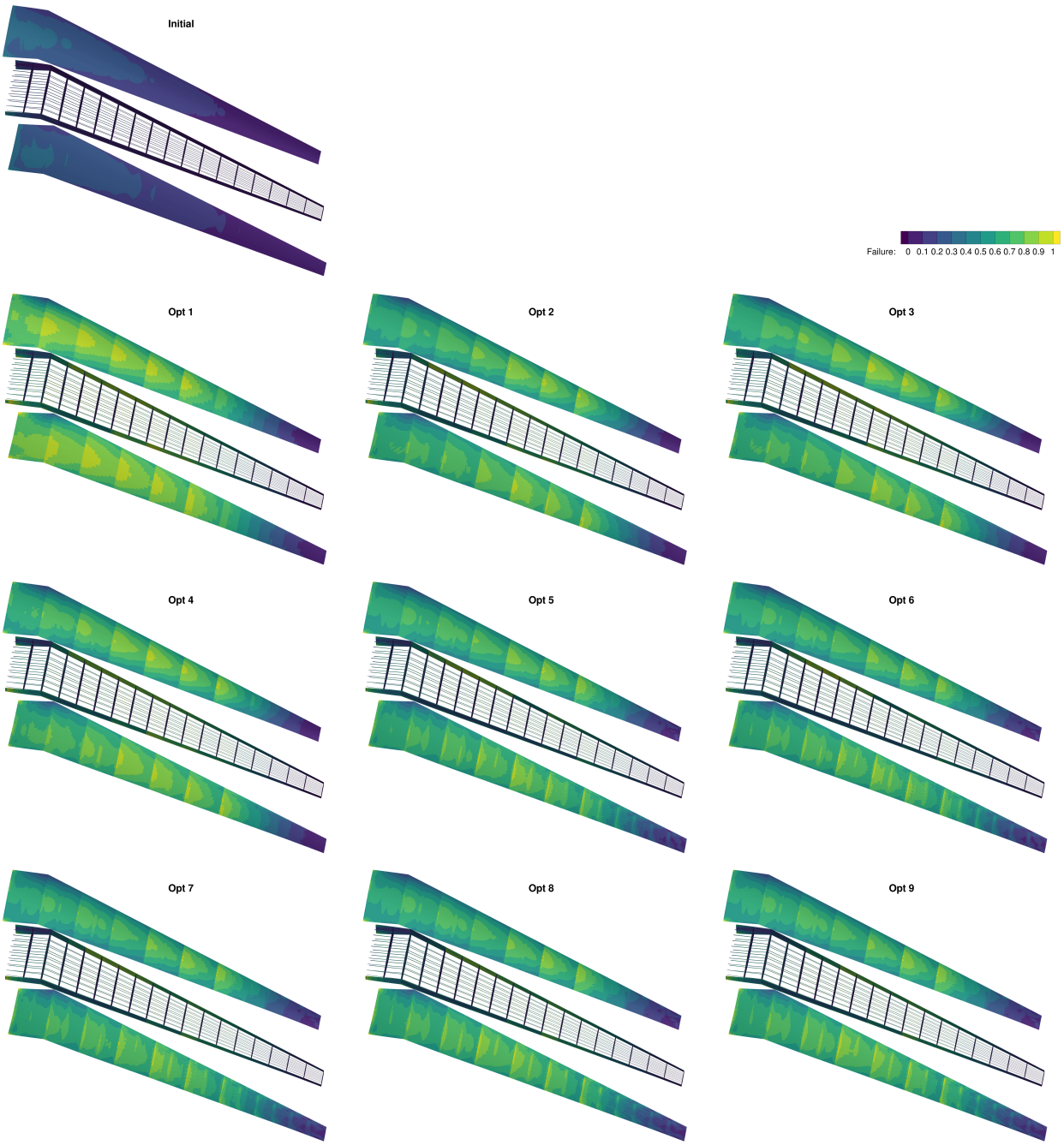


Figure 7.6: Sequence of stress failure values at the end of each sub-optimization, where a value of 1.0 indicates the yield limit.

Next, we plot some design variables over the optimization history in figure 7.7, comparing the progress made by the single and multifidelity approach. We have selected design variables plotted against major iterations throughout the optimization on the left. As expected, the single-fidelity approach took significantly fewer major iterations to arrive at the optimum. However, this

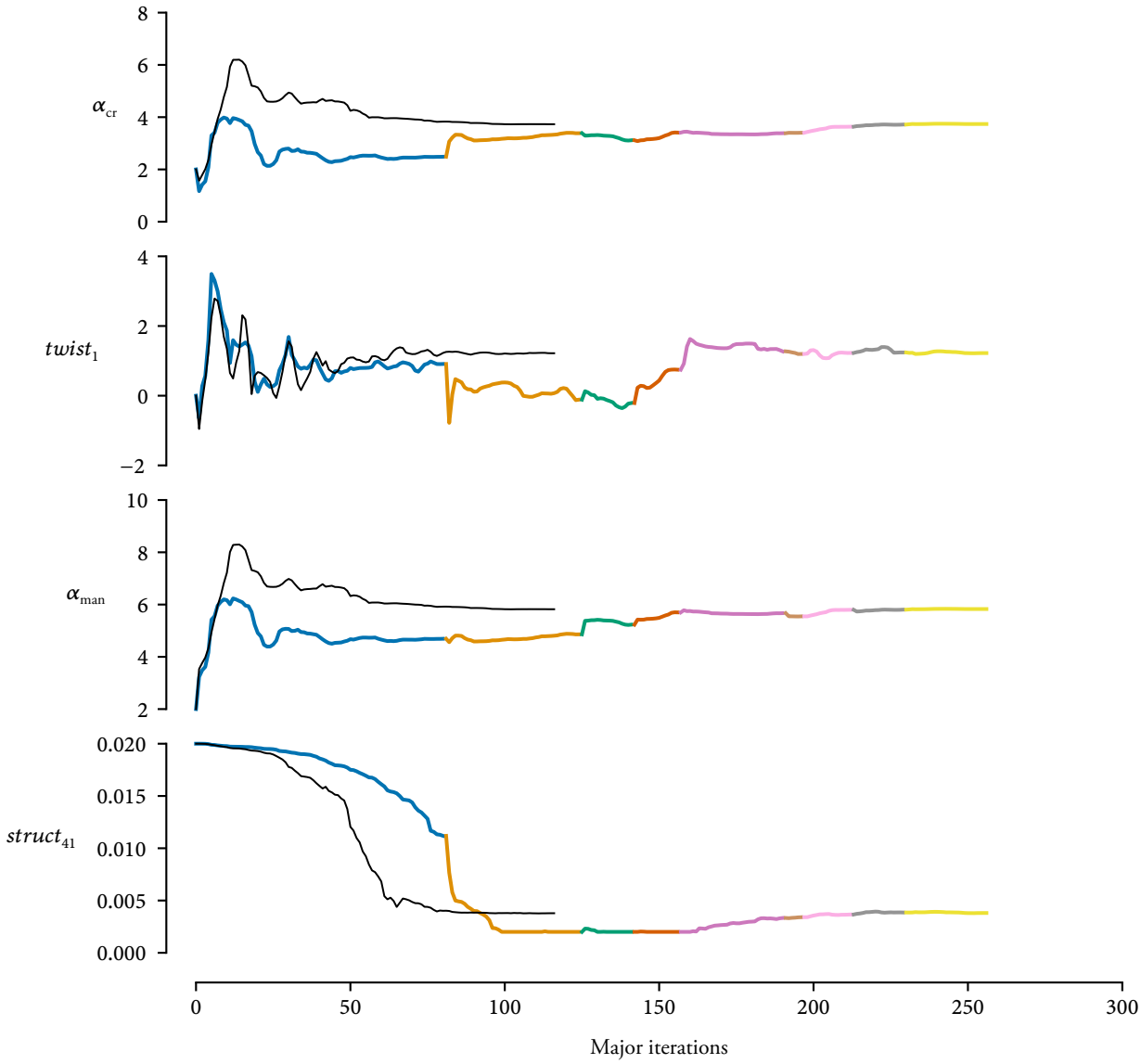


Figure 7.7: Selected design variables during the course of optimizations. Because we hot-start each optimization, these lines are continuous. Despite taking more iterations to converge, due to cheaper, lower-fidelity models, the design variables converged more quickly when measured using computational cost.

is misleading because the earlier iterations in the multifidelity approach are significantly cheaper. When adjusted for the computational cost, the multifidelity approach is much quicker, as shown in figure 7.8. The earlier, cheaper optimizations required far fewer resources. By the time we start the last few expensive optimizations, the designs are so close to the optimum that only a few iterations are needed.

Similarly, we plot the optimization history for a few representa-

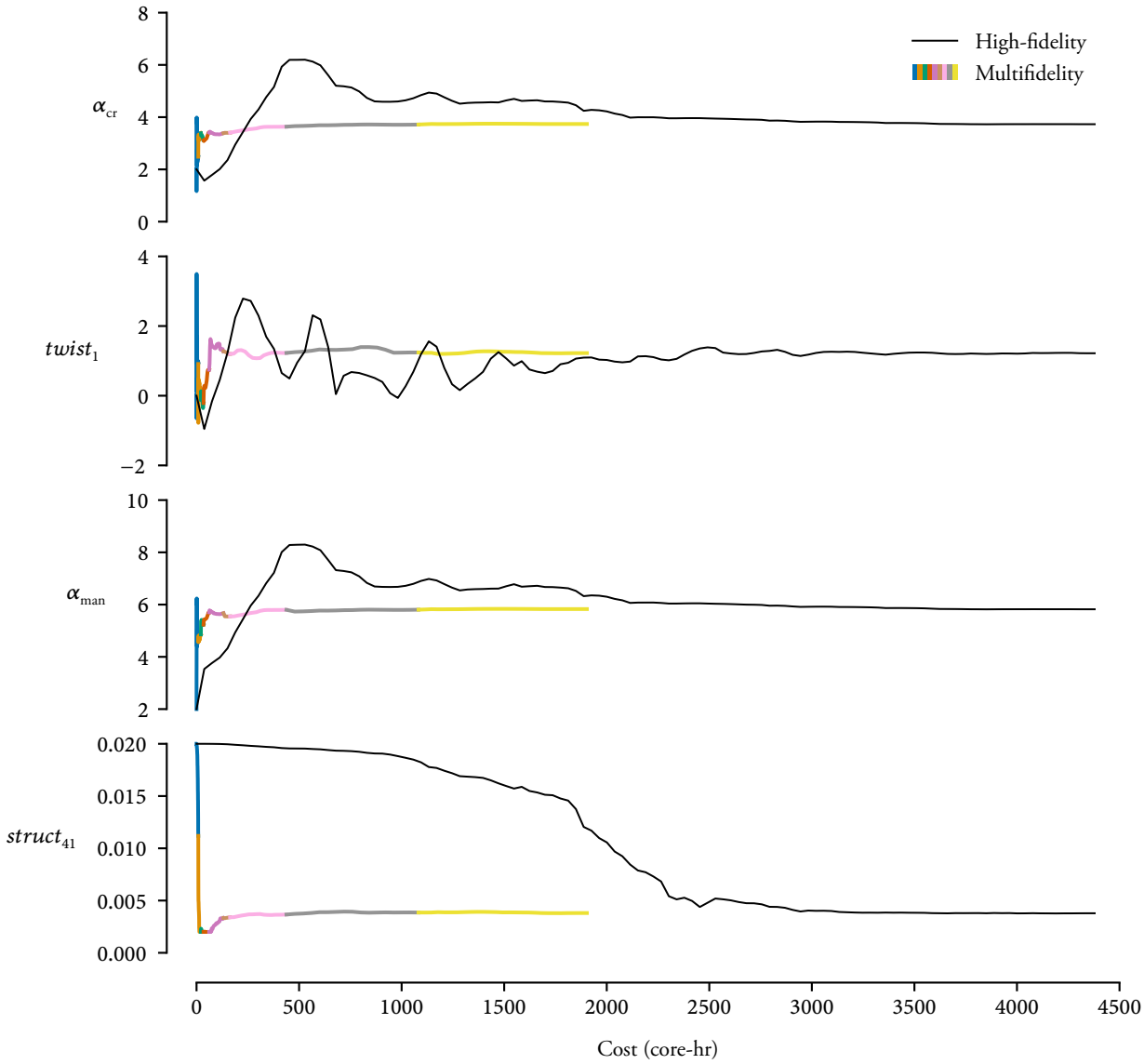


Figure 7.8: The same figure as figure 7.7, but plotted against computational cost. The use of low-fidelity models in earlier iterations is obvious.

tive outputs in figure 7.9. Unlike design variables, these outputs are not continuous across optimizations since the same design analyzed using different fidelities will yield different outputs. Nevertheless, the outputs still converge relatively quickly when plotted against computational cost.

Figure 7.10 shows the normalized distance traversed by the two optimizations, which provides a good idea of the progress made throughout the optimizations. Because the design variables vector $\boldsymbol{x}$ is composed of entries of varying magnitudes, the design

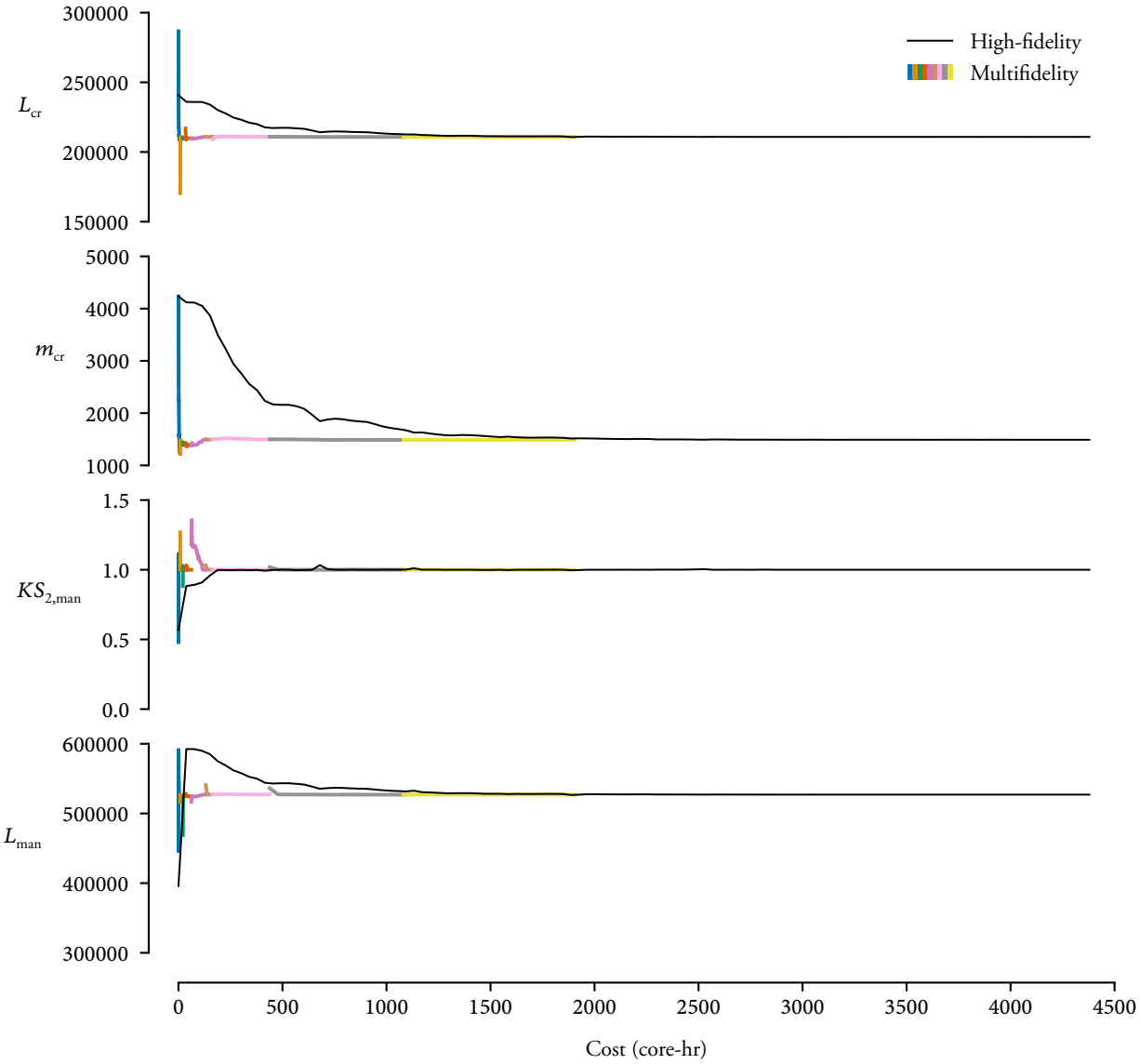


Figure 7.9: Selected function outputs during the course of optimizations. The discontinuity is due to the same design being analyzed by different fidelities.

variables are first scaled element-wise following the scaling factors in table 7.6. These scaled design variables $\hat{\boldsymbol{x}}$ are then used to compute a scalar distance metric at each optimization iteration, using

$$d(\hat{\boldsymbol{x}}_i) = \frac{\|\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_{\text{final}}\|_2}{\|\hat{\boldsymbol{x}}_{\text{initial}} - \hat{\boldsymbol{x}}_{\text{final}}\|_2}. \tag{7.2}$$

This distance metric is normalized such that the initial design vector $\boldsymbol{x}_0$ is one unit distance away from the final high-fidelity optimum. Throughout the optimizations, the design gradually

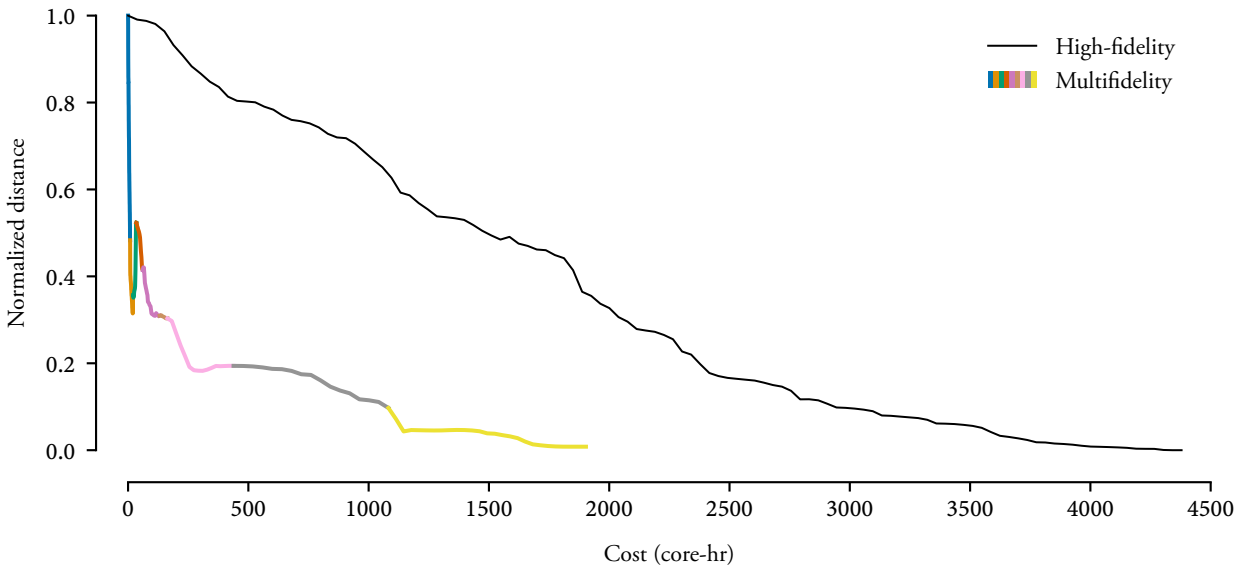


Figure 7.10: Normalized distance from the initial to the final optimum

converges to the final optimum.

From figure 7.10, the multifidelity approach uses the low-fidelity optimizations at the beginning to make significant progress towards the final optimum while using minimal computational resources. The first five optimizations cost less than 1% in total but obtained a design that is 70% of the distance to the final optimum. However, it is worth noting that not all the optimizations took the design closer to the optimum. For example, the third optimization took the design further away. This is not unexpected since there is no guarantee that the distance will reduce monotonically throughout. Due to the nature of SQP, the optimization will typically follow the constraint boundary once a feasible point is found. This may result in a circuitous path within the design space. Ultimately, as we improve the fidelities used and tighten the termination criteria, we see the optimization rapidly converging to the final optimum.

Lastly, we plot the merit function, the feasibility, and the optimality tolerances for the optimization in figure 7.11, as computed by SNOPT. The merit function is defined as the augmented Lagrangian plus a quadratic penalty term for constraint violations

Table 7.6: Normalization factors on the design variables used to compute distances within the design space.

Design variable	Scaling
α	0.1°
Twist	0.1°
Thickness	0.0001 m

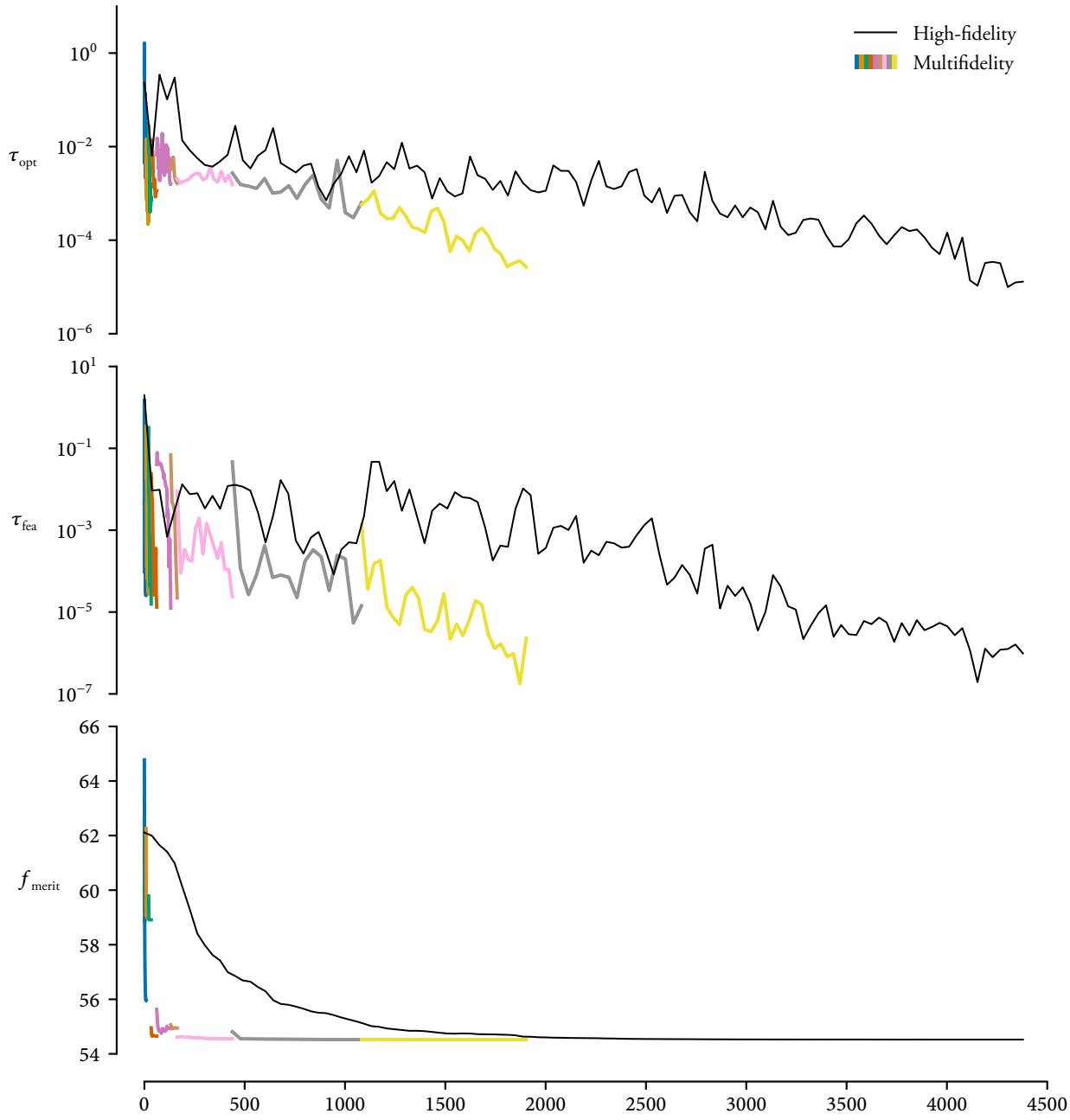


Figure 7.11: Optimization metrics as reported by SNOPT over the course of the optimizations.

and is used during line search to find an appropriate step length. The feasibility and optimality tolerances are used as termination criteria and are good metrics for judging the progress of the optimization. Because of the efficient hot start, later optimizations in the multifidelity approach can quickly reduce the feasibility and optimality compared with the single-fidelity approach

that also used the most expensive high-fidelity model. Furthermore, the feasibility and optimality plots show that the earlier optimizations were not fully converged. Instead, the error-based switching criteria terminated each optimization at an appropriate time without over-converging on the lower-fidelity models.

Overall, we see that the multifidelity approach uses more iterations compared to the single-fidelity approach, but the majority of those iterations were spent on the low-fidelity models. This allowed the optimizer to "learn" the optimization problem by building up the approximate Hessian while moving closer to the final optimum. The final few optimizations using higher-fidelity models took fewer iterations and offered significant computational savings. Ultimately, a 59% cost reduction is achieved while obtaining the same numerical optimum.

7.1.3 Impact of coupled errors

In this section, we go over an optimization that incorporates coupled errors based on the approach in chapter 6. We compare against the previous results reported in section 7.1.2, both with the multifidelity approach and the single high-fidelity optimization. We use the same optimization problem and available fidelities, the only difference is in the addition of coupled errors in the fidelity selection algorithm.

First, we perform error quantification for all disciplines. This involves a total of $(5 + 4) \times 2 = 18$ analyses and the same number of adjoint solutions, with a total cost of 8.02 core h and 20.92 core h, respectively. Relative to the cost of the high-fidelity optimization, the increased cost of quantifying coupled errors is only 0.477 %, and the total cost of error quantification is 0.6609 %, which is a negligible fraction since it only needs to be done once for a given optimization problem.

Next, we perform multifidelity MDO accounting for these coupled errors. Table 7.7 shows the result of the optimization.

	Cruise		Maneuver			
Number	Aero	Struct	Aero	Struct	Cost	% Cost
1	E1	O2L3	E1	O2L3	8.55	0.5
2	R2	O2L3	E1	O2L3	5.34	0.3
3	R2	O2L3	R2	O2L2	27.29	1.7
4	R1	O3L3	R2	O2L2	34.77	2.2
5	R1	O3L3	R2	O3L2	101.61	6.4
6	R1	O3L2	R1	O3L2	52.33	3.3
7	R0	O3L2	R1	O3L2	170.23	10.7
8	R0	O3L2	R0	O3L2	1197.23	75.0
Total					1597.36	100.0

Table 7.7: The sequence of optimizations taken when considering coupled errors.

The first thing to note is that, the multifidelity optimization considering coupled errors was 10% faster compared to the multifidelity optimization without. When compared to the high-fidelity reference optimization, the case considering coupled errors took 64% less time, compared to 59% without.

But the results are more illuminating after a closer look. Instead of requiring a sequence of nine optimizations, the framework has instead taken just eight, which largely accounts for the savings in computational time. This was accomplished primarily by increasing the structural fidelity of the cruise point, switching to the high-fidelity model O3L2 at the sixth optimization rather than all the way at the end. This has effectively allowed us to skip the eighth optimization in the previous optimization, saving computational time in the process. In contrast, in the optimization without coupled errors, the fidelity selection algorithm was hesitant to improve the cruise structural fidelity because its error contribution was only in computing the mass, which was accurate even for the low-fidelity models. That observation was the primary motivation behind this work, and now that the coupled errors are captured, the fidelity selection algorithm without any modification was able to automatically select a more appropriate sequence of fidelities. This result clearly highlights the need for incorporating coupled errors into the multifidelity MDO framework, and showed that despite not fully accounting for the two-way coupling between the disciplines, enough cross-discipline interactions were accounted for to elicit the desired

behaviour. Not only is there a reduction in computational cost, but it was done by selecting more appropriate fidelities for this particular optimization problem, and in a completely automated fashion.

7.2 Demonstration on the XRF1

7.2.1 Problem description

We perform aerostructural optimization of the XRF1 model [211], which is an aircraft research model developed by Airbus that contains both an aerodynamic and structural model. In this work, the wing-body-horizontal tail configuration is used.[1] Figure 7.12 shows a sample aerostructural solution of the configuration, where the aeroelastic deflection of the wingtip due to the aerodynamic loads can be seen.

[211]: Pattinson et al. (2013), *High Fidelity Simulation of Wing Loads with an Active Winglet Control Surface*

1: The OML is provided by Airbus, but the internal structural layout of the wingbox was constructed by Gaetan Kenway at the University of Michigan.

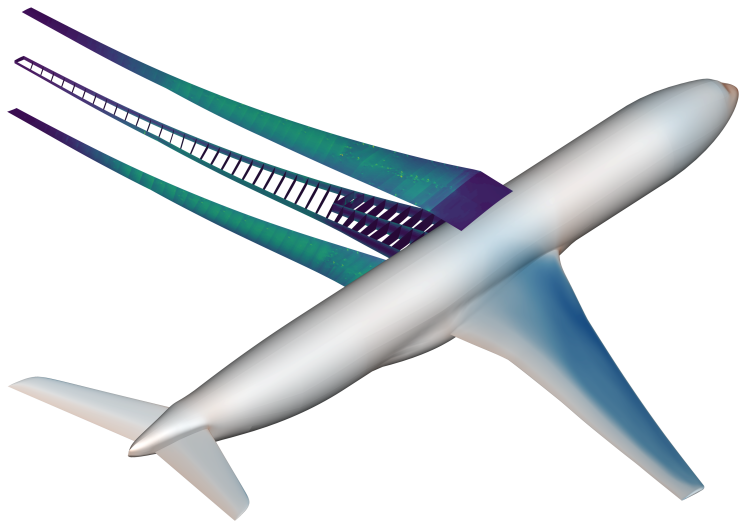


Figure 7.12: A sample aerostructural solution of the XRF1 wing-body-horizontal tail configuration.

7.2.1.1 Objective function

The objective function used for the optimization is to minimize the fuel burn at the cruise point. The fuel burn is computed based on a rearranged form of the Bréguet range equation:[2]

2: which may be more aptly named Devillers–Coffin equation [212].

$$W_{\text{TOGW}} = W_{\text{LGW}} \exp\left(\frac{R\, c_T}{V\ (L/D)}\right) \tag{7.3}$$

$$W_{\text{FB}} = W_{\text{TOGW}} - W_{\text{LGW}}, \tag{7.4}$$

where W_{TOGW} is the take-off gross weight (TOGW), W_{FB} is the fuel burn (FB), W_{LGW} is the landing gross weight (LGW), R is the mission range, c_T is the TSFC, V is the cruise speed, and L/D is the lift-to-drag ratio. The landing weight is computed using the following formula:

$$W_{\text{LGW}} = 1.25 \times W + \text{Area Weight} + \text{Fixed Weight} + \text{Payload} + \text{Reserve Fuel Weight}, \tag{7.5}$$

where W is the weight computed by the structural finite-element model. The factor of 1.25 used here accounts for additional weight associated with fasteners and other components not modeled in the idealized structural wingbox model. The "Area Weight" refers to the additional mass associated with the LE and TE, and the necessary actuation equipment. The relevant values are listed in table 7.8.

Table 7.8: The reference values used for the optimization.

Parameter	Value
Area Weight	6000 kg
Fixed Weight	90 595 kg
Payload	23 950 kg
Reserve Fuel Weight	8000 kg
Range	8000 nm
TSFC	15.57 g/k$\mathcal{N}$/s

7.2.1.2 Cruise and maneuver flight conditions

There are in total one cruise point and three maneuver points. The cruise point is used to compute the objective, while the maneuver points are used to compute the structural constraints at higher loads. The three maneuver points include symmetric 2.5 g pull-up and −1.0 g push-over maneuvers, as well as a gust case analyzed in steady-state cruise. The purpose of the gust case is to simulate a sudden load during cruise, such that the structure did not have time to respond. Therefore, it cannot rely on passive load alleviation to shift the spanwise loading inboard. This load case is simulated at 1 g and an elevated Mach number and altitude, and an increased safety factor is applied, similar to the approach by Brooks, Kenway, and Martins [27].

[27]: Brooks et al. (2018), *Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings*

Contrary to most optimizations, we do not fix the cruise altitude but set it as a design variable h. The full set of flight conditions are listed in table 7.9.

Number	Mach	Altitude	Fuel fraction	Load factor
1	0.83	h	0.50	1.0
1	0.78	16 000 ft	1.0	2.5
2	0.78	22 000 ft	1.0	−1.0
3	0.84	27 000 ft	1.0	1.0

Table 7.9: Flight conditions for the multipoint optimization.

7.2.1.3 Design variables

There are over 900 design variables for this optimization problem. First are the geometric design variables, which directly impact both the aerodynamic and structural analyses. Those include local shape variables that modify the sectional airfoil shape at each spanwise station, sectional twist for the wing, and tail rotation angles for each flight condition to trim the aircraft. Wing planform variables such as span or sweep are not considered in this work.

Next, there are aerodynamic design variables. For this optimization, they are the individual angles of attack for the different flight conditions, and the altitude h.

Lastly are the structural design variables. We use a smeared stiffness approach to model the wingbox [213], where the effects of the stiffeners are accounted for by changing the material properties of the panel rather than being explicitly modelled. As a result, instead of a single panel thickness variable for each patch of the wingbox, we have three more: stiffener pitch, thickness, and height. However, we do not allow each panel to control these variables independently. For example, we keep a single stiffener pitch variable for each of the skins, ribs, and spars. The full breakdown of the structural design variables are shown in table 7.10. In this work, we model the wingbox using 7000 series aluminum alloy with a yield stress of 420 MPa.

[213]: Kennedy et al. (2014), *High Aspect Ratio Wing Design: Optimal Aerostructural Tradeoffs for the Next Generation of Materials*

Figure 7.13 shows the geometric and structural design variables on the XRF1 model.

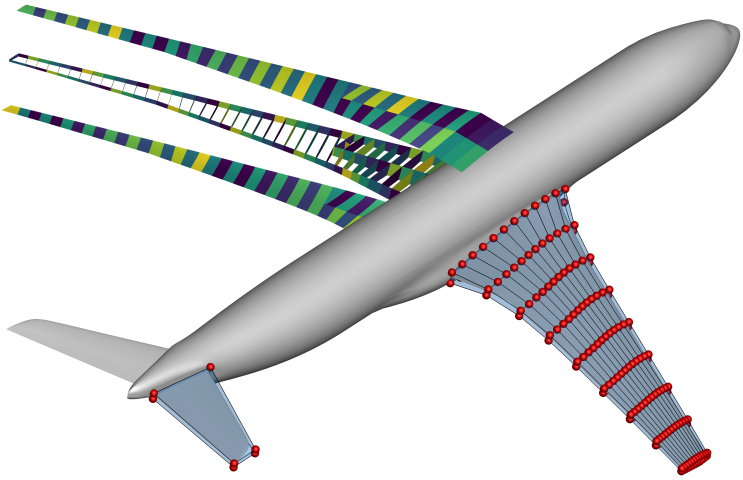


Figure 7.13: The geometric and structural parameterization for the XRF1 model, showing the component breakdown of the wingbox and the FFD volume used for shape control.

7.2.1.4 Constraints

There are also many constraints present in the optimization problem. First are the aerodynamic constraints, consisting of lift and moment constraints for each flight condition. We also enforce a separation constraint for the three maneuver cases, as the optimizer has a tendency to exploit the problem by stalling the wing in order to reduce the wing loading.

Next are the geometric constraints. We impose limits on the LE radii and TE thickness on the wing, such that their values cannot go below their initial values. We also impose a fuel volume constraint, and linear leading and trailing-edge constraints to prevent the shape design variables from emulating a twist deformation.

Finally, there are structural constraints on the wingbox. These include yield and buckling constraints computed on the maneuver flight conditions, and aggregated using the KS function [209]. We also add linear adjacency constraints to prevent significant differences in the structural design between adjacent components. The full optimization problem is given in table 7.10.

[209]: Kreisselmeier et al. (1979), *Systematic Control Design by Optimizing a Vector Performance Index*

Table 7.10: The aerostructural optimization problem for the XRF1 case. "(L)" denotes that the constraint is linear.

	Function/variable	Description	Quantity
minimize	W_{FB}	Fuel burn	1
with respect to	x_{twist}	Wing twist	9
	x_{shape}	FFD control points	216
	α_i	Angle of attack for each flight condition	4
	h	Cruise altitude	1
	x_{η_i}	Tail rotation angle for each flight condition	4
	$x_{\text{panel thick}}$	Panel thickness for skins/spars/ribs	273
	$x_{\text{skin pitch}}$	Stiffener pitch for skins	2
	$x_{\text{spar pitch}}$	Stiffener pitch for LE/TE/mid spar	3
	$x_{\text{rib pitch}}$	Stiffener pitch for ribs	1
	$x_{\text{stiff height}}$	Stiffener height for skins	116
	$x_{\text{spar height}}$	Stiffener height for LE/TE/mid spar	3
	$x_{\text{rib height}}$	Stiffener height for ribs	1
	$x_{\text{stiff thick}}$	Stiffener thickness for skins/spars/ribs	273
	Total		**906**
subject to	$L = n_i W$	Lift constraint	4
	$C_{M_y} = 0$	Trim constraint	4
	$A_{\text{sep}} \leq 4\% A_{\text{ref}}$	Amount of separation at maneuver	3
	$t_{\text{LE}}/t_{\text{LE}_{\text{Init}}} \geq 1$	Leading edge radius	20
	$t_{\text{TE}}/t_{\text{TE}_{\text{Init}}} \geq 1$	Trailing edge thickness	20
	$V_{\text{wing}} > V_{\text{fuel}}$	Minimum fuel volume	1
	$\Delta z_{\text{TE,upper}} = -\Delta z_{\text{TE,lower}}$	Fixed trailing edge (L)	8
	$\Delta z_{\text{LE,upper}} = -\Delta z_{\text{LE,lower}}$	Fixed leading edge (L)	8
	$\text{KS}_{\text{buckling}} \leq 1$	2.5 g buckling	3
	$\text{KS}_{\text{yield}} \leq 1$	2.5 g yield stress	4
	$\text{KS}_{\text{buckling}} \leq 1$	1.0 g gust buckling	3
	$\text{KS}_{\text{yield}} \leq 1$	1.0 g gust yield stress	4
	$\text{KS}_{\text{buckling}} \leq 1$	−1.0 g buckling	3
	$\lvert x_{\text{panel thick}_i} - x_{\text{panel thick}_{i+1}} \rvert \leq 0.5$ mm	Skin thickness adjacency (L)	217
	$\lvert x_{\text{stiff thick}_i} - x_{\text{stiff thick}_{i+1}} \rvert \leq 0.5$ mm	Stiffener thickness adjacency (L)	217
	$\lvert x_{\text{stiff height}_i} - x_{\text{stiff height}_{i+1}} \rvert \leq 0.5$ mm	Stiffener height adjacency (L)	217
	$\lvert x_{\text{stiff thick}} - x_{\text{panel thick}} \rvert \leq 2.5$ mm	Maximum stiffener-skin difference (L)	186
	Total		**922**

7.2.1.5 Available fidelities

There are a number of fidelities available for the multifidelity approach. For aerodynamics, there are four different RANS grids available, which we label `L3`, `L2.5`, `L2`, and `L1.5`. For structures, there are a total of six grids of varying solution order and mesh level. These fidelities are listed in table 7.11.

In total, there are 24 different fidelity combinations for each

MDA. With four different flight conditions, the number of total possible combinations is $(4\times6)^4 = 331\,776$, although the actual number is slightly lower due to the initial Pareto filter applied to the discipline outputs.

Table 7.11: The different fidelities available for both aerodynamics and structures.

Fidelity	# of DOF
L3	113 604
L2.5	282 408
L2	908 832
L1.5	2 259 264
02L3	6254
02L2	21 937
03L3	26 300
02L1	85 814
03L2	90 336
03L1	348 470

7.2.2 Results

We perform two aerostructural optimizations. The reference optimization uses just the high-fidelity model, while the multifidelity optimization will use a number of different fidelities sequentially. Since the two approaches solve the same optimization problem, we should obtain the same optimum.

Table 7.12 shows a summary of the two optimizations. Out of the 331 776 possible fidelity combinations, the appropriate fidelity framework selected a sequence of eight. Compared to the single-fidelity optimization, the multifidelity approach obtained a computational speedup of 44%. Although it took more major iterations in total, the earlier iterations were substantially cheaper since they were computed using lower-fidelity models. The final sub-optimization used close to 90% of the total cost of the appropriate fidelity optimization, but it took only 335 iterations instead of 513 for the reference optimization.

It is important to note here that the load balance efficiency of the two results are comparable to each other. The load balance efficiency is defined as the "real" computational cost (in core h) divided by the total computational cost. This factor represents the ability to balance the computational load in multipoint problems, and can be viewed as the average of the ratios of the coloured rectangles to the total rectangular area in figure 5.5. The efficiency was 74% for the multifidelity results, and 76% for the high-fidelity results. Therefore, the appropriate fidelity results did not outperform the high-fidelity optimization simply by having an improved load balancing. The speedup obtained is due to the use of the low-fidelity models to accelerate convergence.

Table 7.12: The sequence of fidelities for the aerostructural optimization, showing the fidelity used, the computational costs, and the number of major iterations taken. Here the fidelity combination is represented by four pairs of columns, corresponding to the four analysis points. The cost is given in proc-hours, and the relative cost is normalized by the total cost of the multifidelity approach.

	Cruise		Maneuver 1		Maneuver 2		Maneuver 3				
#	Aero	Struct	Aero	Struct	Aero	Struct	Aero	Struct	Cost	% Cost	# Iter
1	L3	O2L1	L3	O2L2	L3	O2L2	L3	O2L2	936	1.4	162
2	L2.5	O2L1	L3	O2L0	L3	O2L2	L3	O2L2	1146	1.7	133
3	L2	O2L0	L3	O2L0	L3	O2L2	L3	O2L1	273	0.4	15
4	L2	O3L0	L2.5	O3L0	L3	O2L2	L3	O2L1	699	1.0	39
5	L1.5	O3L0	L2.5	O3L0	L3	O2L0	L3	O3L0	2024	3.0	49
6	L1.5	O3L0	L2	O3L0	L2.5	O2L0	L2	O3L0	661	1.0	7
7	L1.5	O3L0	L1.5	O3L0	L2.5	O2L0	L2	O3L0	977	1.5	206
8	L1.5	O3L0	L1.5	O3L0	L1.5	O3L0	L1.5	O3L0	59 968	89.9	335
Total									66 683	100.0	946
HF	L1.5	O3L0	L1.5	O3L0	L1.5	O3L0	L1.5	O3L0	118 313	177.4	513

To show the fidelities in a more intuitive fashion, we take the information from table 7.12 and plot it in figure 7.14. Upon closer examination, the fidelity choices are quite intuitive. The fidelity selection algorithm preferred improving the structural fidelity of the 2.5 g maneuver point first, as that was the most structurally constrained analysis. The other structural fidelities are improved at a later stage, corresponding to their diminishing impact on the overall optimization. The −1.0 g point which was least constrained, only switched to the high-fidelity structural model at the last sub-optimization. On the other hand, although the cruise structural fidelity was only responsible for computing the structural mass, it also introduces errors into the aerodynamic outputs due to the multidisciplinary coupling. As a result, it was also improved rapidly.

In terms of the aerodynamic fidelities, naturally the cruise aerodynamic fidelity was improved first, since it is responsible for computing the objective function. The maneuver analyses, on the other hand, stayed on low-fidelity models for longer since only a few aerodynamic constraints are computed, and the aerodynamic fidelities are very expensive relative to the structural fidelities. These trends are not dissimilar to what was observed in section 7.1 for the wing-only case.

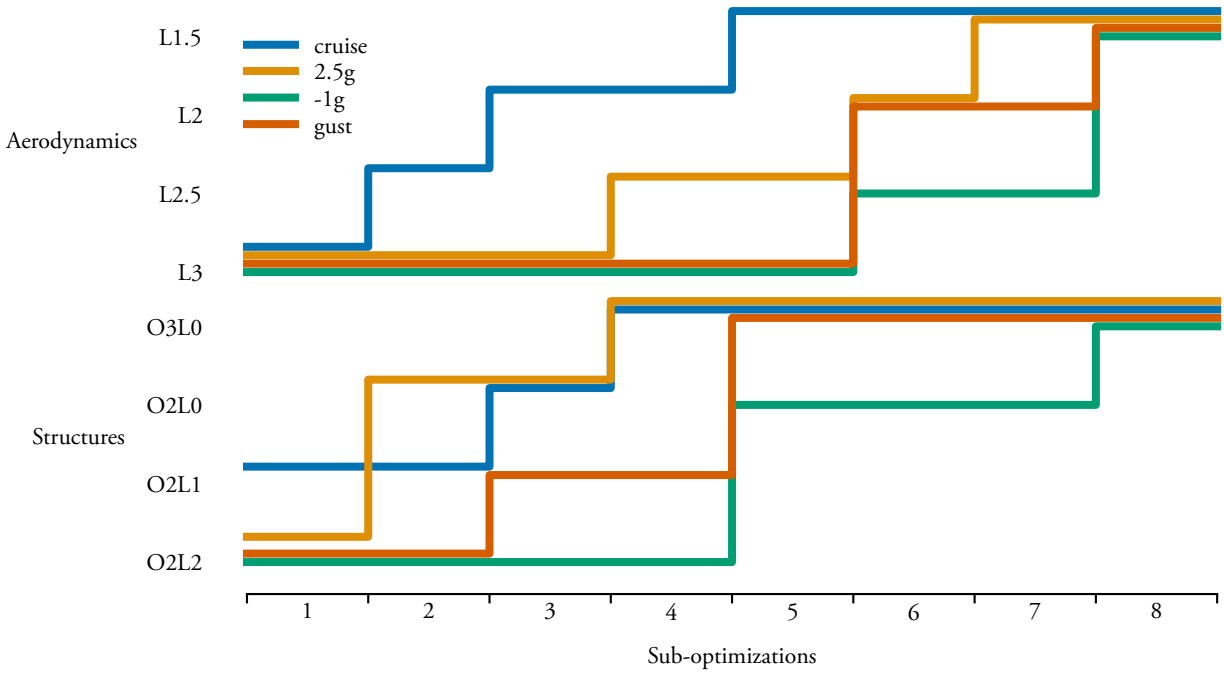


Figure 7.14: The sequence of aerodynamic and structural fidelities selected, plotted on a graph.

Since both optimizations solve the same problem, if the design space is unimodal then we expect both approaches to obtain the same numerical optimum. While there are subtle differences in the final designs, they are expected given the level of convergence for the optimizations. In terms of the objective, the difference between the fuel burn of the two optimal designs is 0.0066 kg, for a relative difference of 6.97×10^{-8}. Figure 7.15 shows the progress of the optimization, measured in terms of the normalized distance as computed using equation 7.2.

We see that at the initial sub-optimizations, the design actually moved further away from the optimum. This is partially due to the fact that over half of the design variables are structural design variables, leading to an over-emphasis of their changes when applying the L_2 norm. These structural design variables did not converge initially because the aerodynamic loads from CFD were computed using low-fidelity models, leading to inaccurate stress computations. This caused the wingbox to be significantly undersized. However, while there are no guarantees of monotonic convergence, the framework recovered quickly, and converged

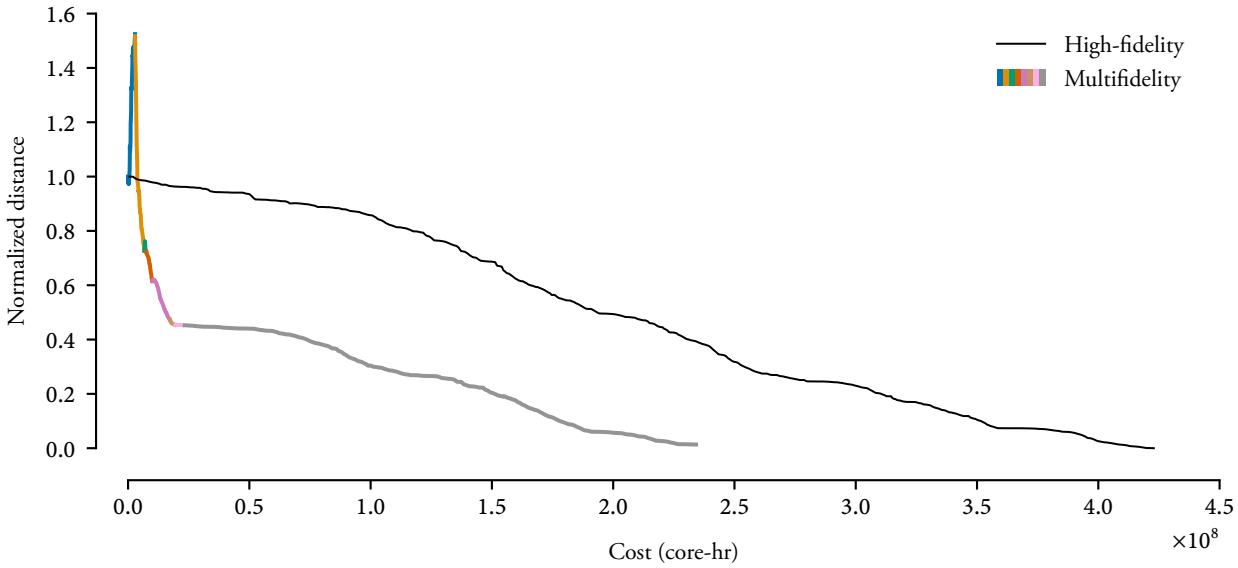


Figure 7.15: Normalized distances for the XRF1 case.

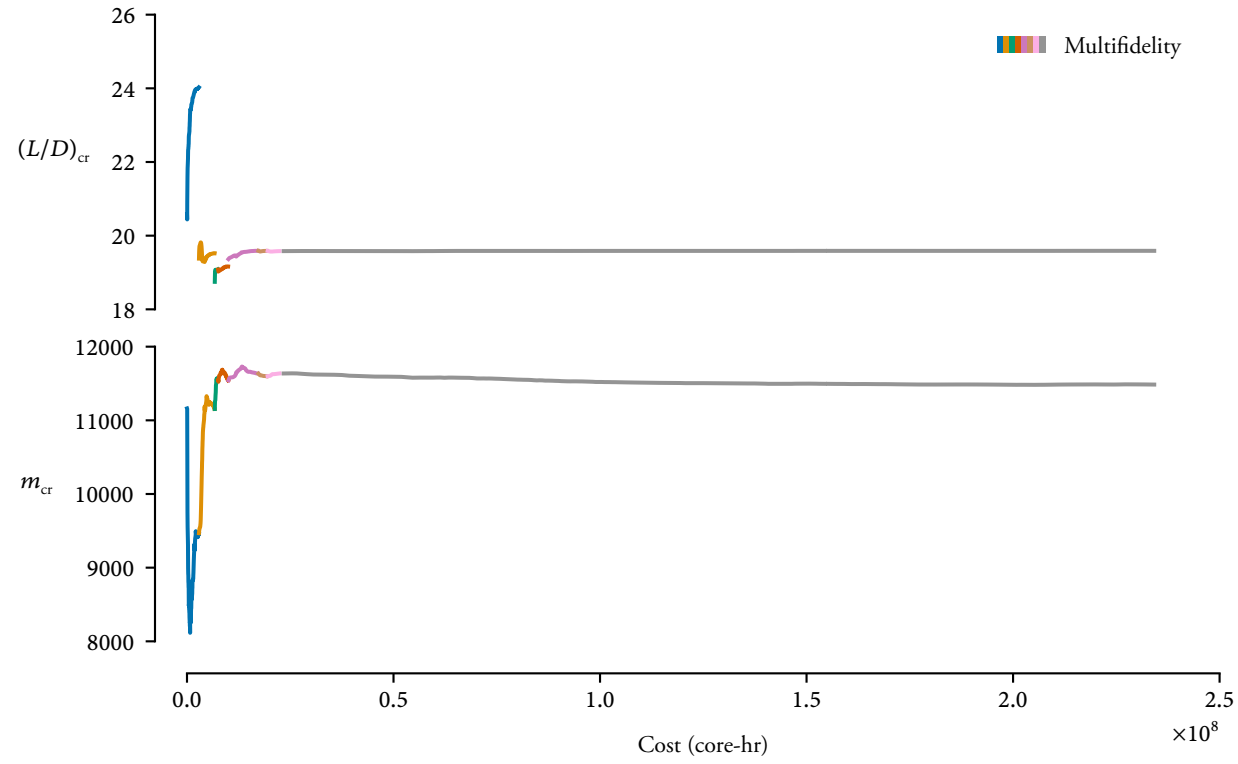


Figure 7.16: Select function outputs for the XRF1 case.

rapidly without any issues.

Next we examine the intermediate solutions in more detail. Figure 7.16 shows the two key performance metrics—the L/D ratio

at cruise and the structural mass—during the appropriate fidelity optimization. These values are computed by the respective fidelity, leading to discontinuities between sub-optimizations as the same design is analyzed on a different fidelity.[3] Therefore, these do not represent true values of those intermediate designs, but rather what is observed by the optimizer. For the lift-to-drag ratio, incremental improvements in the aerodynamic efficiency can be seen within each sub-optimization. For the structures, the initial under-sizing of the wingbox is noticeable, but recovery is swift as the fidelities are updated.

3: The discontinuities for the structural mass are imperceptible, due to the accuracy of all FEM models in computing the mass.

Figure 7.17 shows three airfoil profiles at three different spanwise stations along the wing. At the tip, the effect of the aerostructural system is evident, which caused the different tip deflections. At the root and mid-span sections, the design labelled "Opt 6" is virtually indistinguishable from the optimum, both in terms of the airfoil shape and the C_p distribution. However, it took less than 10% of the total computational cost to arrive at this intermediate design. This highlights the efficacy of the appropriate fidelity approach.

Figure 7.18 shows the normalized lift distributions for the 2.5 g load condition at three solutions during the optimization. At this high-lift maneuver case, the optimizer was able to shift more load inboard compared to the elliptical lift distribution, via aeroelastic tailoring. This reduces the root bending moment and therefore the stresses on the wingbox, resulting in a more efficient wing design. The ability of the optimizer to exploit passive load alleviation has been noted before [42, 27].

[42]: Kenway et al. (2014), *Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration*
[27]: Brooks et al. (2018), *Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings*

The lower figure also shows the twist distribution of the wing under the 2.5 g load. Compared to the initial design, the outer sections of the wing are negatively twisted, resulting in reduced lift compared to the inboard section. As before, the 6th intermediate optimum is quite close to the final design.

Next we look at the structural design in more detail. Figure 7.19 shows the initial, intermediate, and final structural solutions in

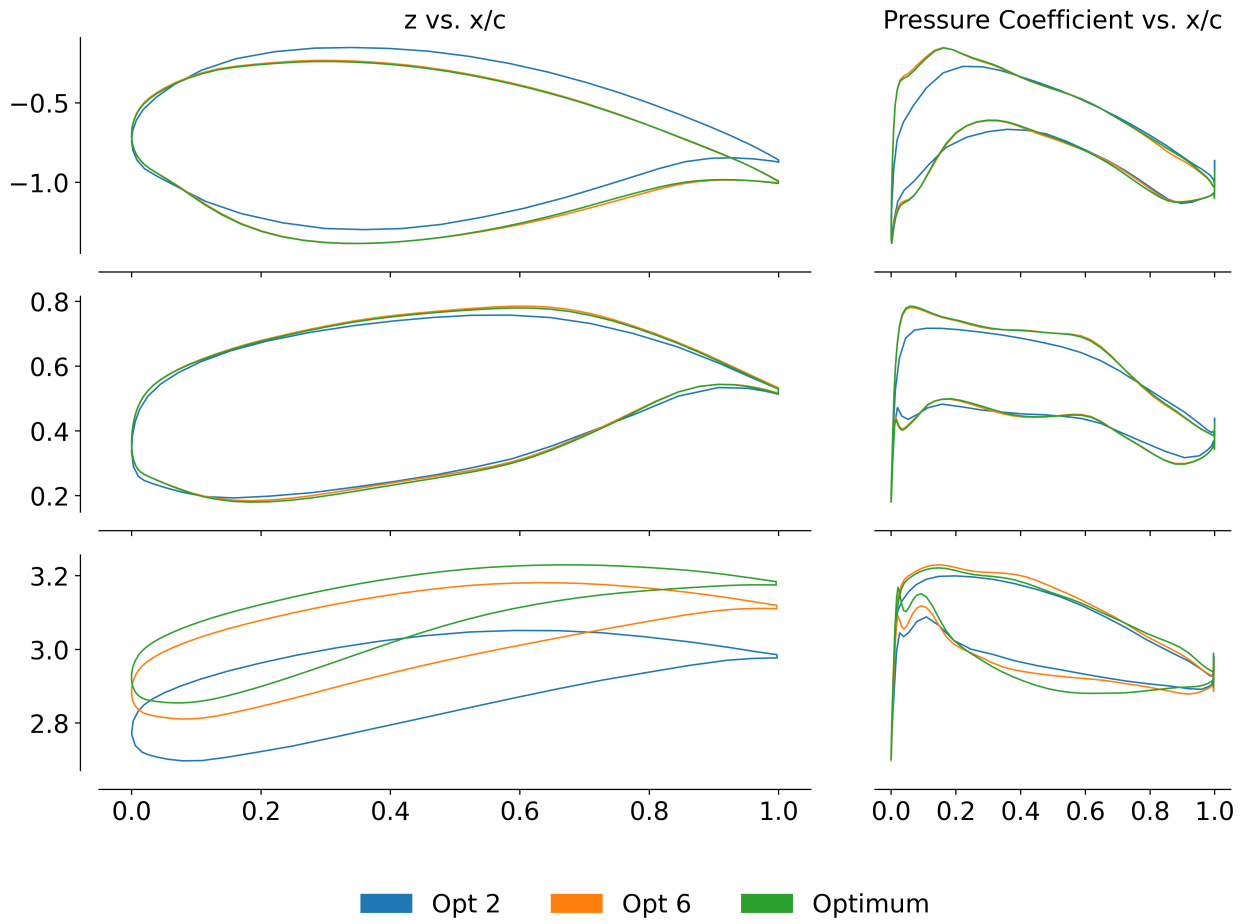


Figure 7.17: Airfoil profiles and pressure distributions along the wing.

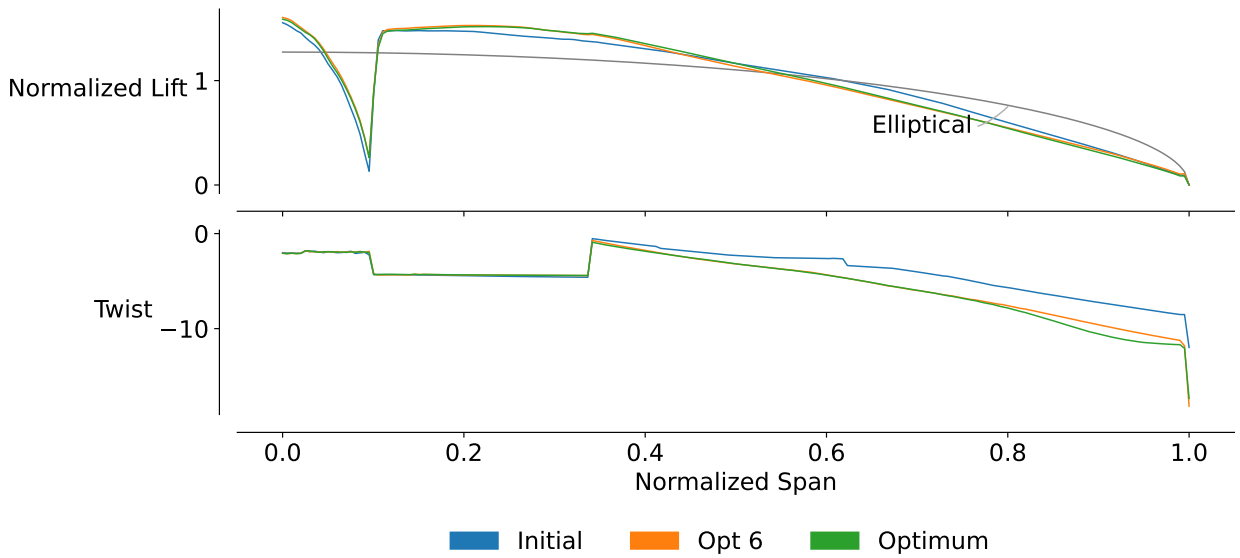


Figure 7.18: Spanwise lift distributions at three different designs.

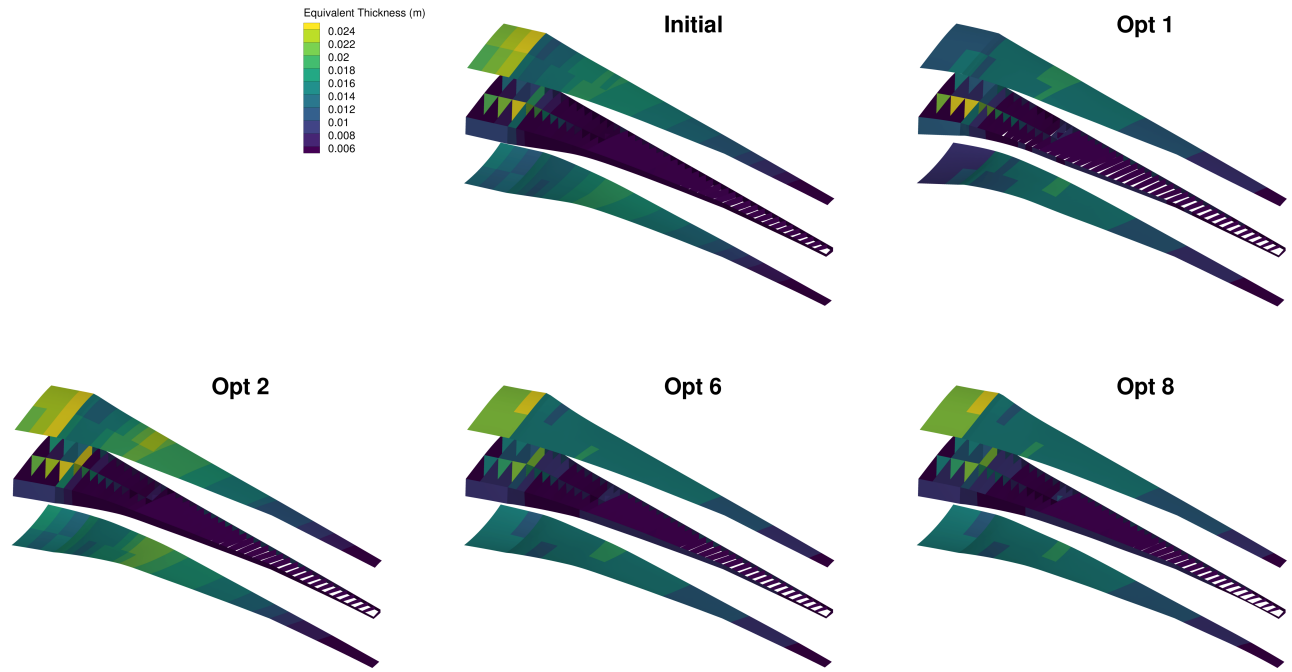


Figure 7.19: Initial, intermediate, and final structural designs for the XRF1 optimization.

terms of the mass-equivalent structural thicknesses. These structural design variables converge relatively quickly, and there are little discernable differences between the two final solutions.

Figure 7.20 shows the stress and buckling constraint values for the 2.5 g load case. The stresses are normalized such that a value of 1.0 corresponds to the yield limit. We see that the initial design was infeasible, as there are large stresses near the tip of the mid-spar. However, by the end of the first optimization, these stress concentrations have been removed and the design has become nearly feasible. For buckling, as expected the most buckling-prone regions are on the upper skin, where the compressive loading is the highest under the pull-up maneuver. The buckling constraint violation near the mid-spar is also evident, which has been removed by the end of the first optimization. However, regardless of the analysis fidelity, the optimizer converges the stress and buckling distributions quickly, and little change is observed after the second sub-optimization. This is despite having loose convergence tolerances for these low-fidelity optimizations.

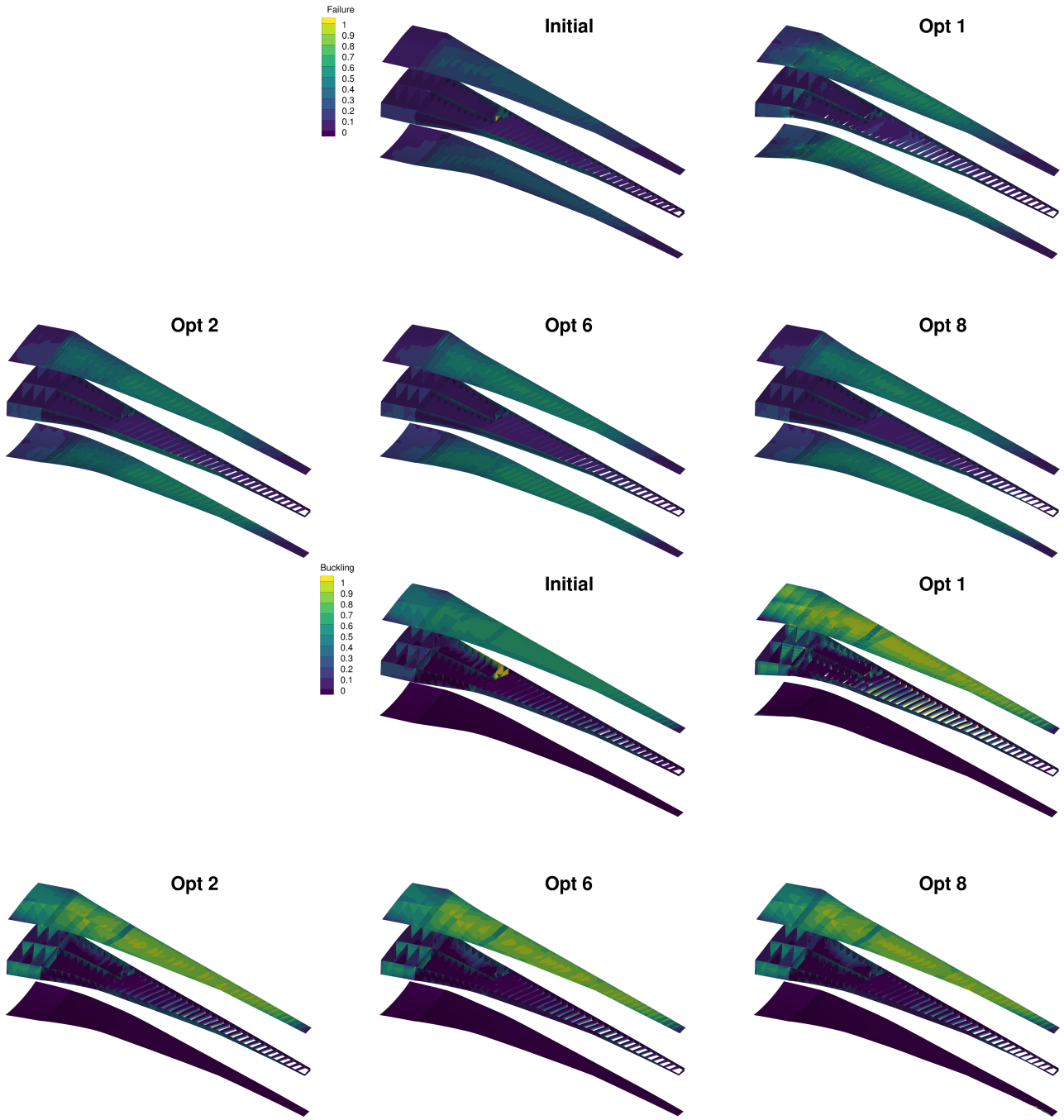


Figure 7.20: Initial, intermediate, and final stress and buckling constraint values for the XRF1 optimization.

Finally, figure 7.21 shows the optimization metrics for the two XRF1 optimizations presented here. While the rate of convergence between the reference optimization and the final multi-fidelity optimization are similar, the starting values are much lower thanks to the earlier optimizations.

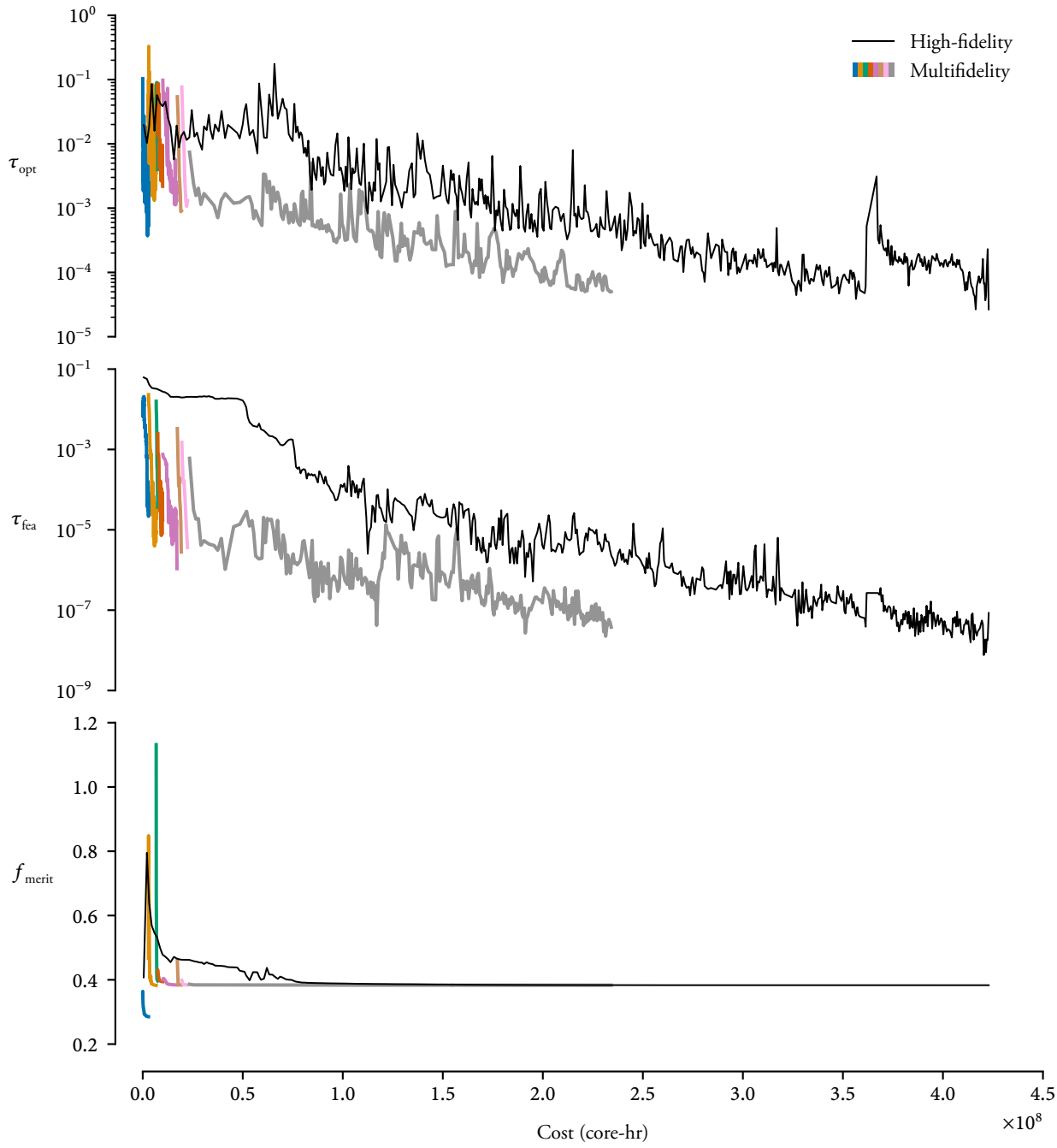


Figure 7.21: Optimization metrics for the XRF1 case.

7.3 Summary

In this chapter, we presented two aerostructural applications of the appropriate fidelity MDO framework. The first problem was a basic aircraft wing design problem with over 100 design

variables. The second problem was applied to the XRF1, which is a large-scale aircraft design problem involving over 900 design variables and 900 constraints, and the aircraft was analyzed at four different operating conditions.

We obtained cost savings of between 64% and 44%, respectively, when compared against the reference high-fidelity optimization. Both multifidelity optimizations also converged to the same numerical optimum. In both cases, the sequence of fidelities chosen were intuitive, making a tradeoff between computational cost and the reduction in the predicted error of each fidelity.

We showed that the framework is able to handle low-fidelity models arising from simplified physics, rather than just coarse discretization. We also showed that the coupled errors are important in informing the fidelity selection algorithm in picking appropriate fidelities. We demonstrated good scalability regarding the number of fidelity combinations that can be seen in practical problems. The XRF1 optimization had 331 776 possible fidelity combinations, which were filtered and down-selected to just eight sub-optimizations. The appropriate fidelity framework has shown to be efficient and reliable in tackling large-scale MDO problems.

Chapter 8

Final Remarks

8.1 Conclusions

This dissertation addressed several long-standing issues when applying MDO to large-scale problems. In chapter 2, I introduced the predominant approach to tackling such problems, using a combination of a gradient-based optimizer with efficient adjoint and coupled-adjoint computations in an MDF architecture. While such an approach is capable of finding the optimum, they suffer from high computational cost and insufficient robustness, often requiring the engineer to fine tune a number of parameters through trial and error.

In chapter 3, I identified the non-orthogonality between geometric design variables as a source of poor optimization convergence. Using SVD, I generated a set of design variables which are orthogonal to each other. Together with an automatic scaling approach, I was able to obtain a superlinear convergence rate indicative of a well-scaled problem. The approach was tested on two ASOs, and in one case the computational cost was reduced by 7%. Even with the modest performance, this result is still important as the reference optimization used a set of hand-tuned scaling parameters. Therefore, a comparable performance using automatic design variable scaling will greatly cut down the cost of having multiple optimization attempts. The approach is also general and applicable to other geometric parameterizations.

In chapter 4, I examined the use of convergence tolerance as a variable-fidelity model to reduce computational cost. I de-

rived the adjoint-based error estimation, and verified the theory and implementation through numerical experiments. I then developed an adaptive error control algorithm to adjust the convergence of the primal and adjoint systems during optimization, while ensuring convergence to the numerical optimum. The approach was demonstrated on two ASO problems, and cost savings of 30%–50%.

In chapter 5, I provided an overview of existing multifidelity methods for optimization. Unfortunately, few approaches are scalable to the problems of interest, containing hundreds of design variables and constraints. None of the methods are able to consider multidisciplinary problems, where inherent tradeoffs occur in the model fidelity at a discipline level. When considering multipoint problems, the number of possible fidelities grows exponentially, leading to millions of combinations that must be handled efficiently. I then developed the appropriate fidelity MDO framework to address these challenges. The framework uses a sequential approach, starting from the lowest fidelity and updating the fidelity between sub-optimizations. Before any optimization, it first quantifies the errors on each output at a discipline level, and propagates them to the system-level objective and constraints. These errors are used in two ways: to terminate low-fidelity optimizations earlier, and to select the next appropriate fidelity.

In chapter 6, I extended the appropriate fidelity MDO framework to capture coupled errors between coupled analyses. Coupled errors arise when errors from one discipline is propagated to another through an MDA. For example, errors in the CFD analysis will cause an error on the pressure distribution, which then induces an error on the resultant stress distribution. As a result, the aerodynamic fidelity can introduce errors on structural outputs even if the structural fidelity is at its highest level. I developed an adjoint-based method to estimate such errors considering only feed-forward coupling, which is efficient and scalable to a large number of coupling variables. The method is verified against

brute-force Monte Carlo propagation, and is shown to be both accurate and efficient.

In chapter 7, I finally demonstrated the appropriate fidelity MDO framework on two aerostructural optimizations. For the standalone wing case, computational savings of 64% was obtained relative to the high-fidelity reference optimization. I also examined the effect of capturing coupled errors, and found that only 59% cost savings were realized when the coupled errors are not considered. In addition, the sequence of fidelities that were automatically chosen by the fidelity selection algorithm were less intuitive, given the lack of information on the coupling. Next, I applied the framework to a large-scale aircraft design problem including over 900 design variables and 900 constraints. There were a total of 331 766 possible fidelities, and 44% savings were obtained, demonstrating its scalability and applicability to large-scale MDO problems.

8.2 Novel contributions

In this dissertation, I have made the following contributions:

I developed a sensitivity-based geometric parameterization and automatic design variable scaling. ASO problems often exhibit slow convergence. Rather than manually tune optimization parameters through trial and error, I developed a method which automatically generates design variables and scales them suitably for optimization. The method is comparable to the traditional approach, with 7% cost savings in one instance, and does not require any trial and error with multiple optimizations.

I developed a tolerance adaptation scheme suitable for gradient-based optimization. I developed an adjoint-based error estimation and convergence adaptation algorithm suitable for gradient-based optimization. The method does not require modification of the existing optimization algorithm, and can be directly applied

to general constrained optimization problems. I demonstrated the method on two ASO problems, with cost savings between 30%–50%.

I developed an appropriate fidelity MDO *framework suitable for large-scale problems.* The multifidelity framework represents the first instance of an approach that is able to tackle large-scale MDO problems, accounting for the different sources of error arising from the discipline fidelities, and effectively handling the large number of possible fidelities.

I efficiently estimated coupled errors in aerostructural systems. The errors arise from errors in the coupling variables, which may be high-dimensional. I developed an adjoint-based methodology to estimate the coupled errors, independent of the number of coupling variables.

I demonstrated the framework on large-scale problems. I demonstrated the framework on two aerostructural optimizations, including one which had over 900 design variables and 900 constraints. For the two cases shown, I obtain cost savings of between 44% and 64% compared to the reference high-fidelity optimization. This represented the first known case of a multifidelity optimization algorithm being succesfully applied to such large-scale multidisciplinary and multipoint problems. I also demonstrate the effectiveness of the automatic fidelity selection algorithm, which selected a logical sequence of fidelities without user input and only based on a thorough analysis of errors present in the available fidelities.

8.3 Recommendations for future work

8.3.1 General

Improve aerostructural primal and adjoint solution accuracy The existing primal and adjoint solutions are limited due to

several factors [21], which has prevented deeper optimization convergences for aerostructural problems. This was something also pointed out by Bons [72], and causes aerostructural optimizations to achieve only around three orders of magnitude reduction in the optimality before the optimizer exits. Improvements in this area will likely require a thorough analysis of the numerical stability of algorithms used, and possibly employing higher-precision linear algebra packages for solving the extremely stiff structural linear system.

[21]: Kenway et al. (2014), *Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Adjoint Derivative Computations*

[72]: Bons (2020), *High-fidelity Wing Design Exploration with Gradient-based Optimization*

Synthesize the three key developments The three developments presented in this dissertation—geometric parameterization, tolerance adaptation, and appropriate fidelity MDO—were all presented in isolation. It would be great to explore applications that took advantage of all three, examine the performance of such approaches and explore potential synergies. For example, the geometric parameterization may accelerate the final suboptimization in the appropriate fidelity framework beyond what may be obtained in the high-fidelity case.

8.3.2 Geometric parameterization

Extension to aerostructural problems The method itself is not limited to aerodynamic problems. The straightforward approach would be to simply combine all pointsets into one collection, and perform SVD on that collection. Further investigation is needed to determine whether such an approach is necessary or sufficient for aerostructural problems.

Consider dimension reduction methods As with other SVD-based approaches, either using a geometry database or active subspaces, it is natural to consider the approach for dimension reduction. It is straightforward to limit ourselves to the top x number of mode shapes when performing optimization, but design variable bounds and directly design variable constraints

must be handled correctly to retain a well-posed optimization problem. A natural investigation would be to determine the tradeoff between the efficiency afforded by a reduced optimization problem, and the inferiority of the objective when certain DOFs are eliminated.

8.3.3 Tolerance adaptation

Extension to larger problems While the theory extends easily to problems containing multiple flight conditions or constraints, demonstrating the methodology on such a problem would be interesting. In particular, load balancing becomes an issue if the solvers are executed in parallel, which would require novel methods to be developed to address such scenario.

Further algorithmic development The algorithm does not currently account for the error in the objective. To address this, we can take the approach from the appropriate fidelity framework, and consider the error in the Lagrangian rather than in individual constraints. Furthermore, the primal and adjoint errors are in fact coupled, and it may be more effective to couple their tolerances together.

Extension to multidisciplinary problems It is natural to extend the convergence error prediction to multidisciplinary systems, via the coupled adjoint variables. Similarly, the MDA and coupled adjoint tolerances can be adapted during optimization. However, in a hierarchical approach, the individual solver tolerances also need to be adapted, and managing all the possible tolerances in a unified fashion can be challenging.

8.3.4 Appropriate fidelity MDO

Improved scalability The current approach, while demonstrably scalable to over 300 000 fidelities, can still struggle against larger problems. One bottleneck is the Pareto filtering, where the current implementation operates at $\mathcal{O}(n^2)$. Even with the FOSM method, it can still be prohibitive when considering billions of fidelities.

8.3.5 Coupled error propagataion

Consider covariance of coupling variables The covariances of the coupling variables are not currently considered. Similarly to the single-discipline coupled errors, a correlation matrix can be assumed *a priori* which will likely provide more realistic error distributions than treating them as independent variables.

Fully-coupled methodology While the feed-forward approach is effective, a fully-coupled approach can be significantly more accurate. However, the existence of the MDA makes the system a Markov chain, and the high dimensionality poses a challenge for existing methods.

Appendix A

CFD Convergence Characterization

We perform several airfoil analyses on the NACA0012 airfoil in order to characterize how the outputs converges as the primal solution converges. We first generated a family of three meshes, with the cell counts given in table A.1.

Table A.1: The cell count of the meshes used.

Level	Cell count
`L0`	350 208
`L1`	87 552
`L2`	21 888

We ran the NACA0012 airfoil at three different flow conditions, listed in table A.2.

Table A.2: The three flow conditions examined.

	Mach number	Angle of attack
1	0.8	1.25°
2	0.4	2°
3	0.4	0°

The $\alpha = 0°$ case captures how the lift converges for a symmetric airfoil and symmetric flowfield where the lift is expected to be 0 at the first and last iteration. The plots of error in the output relative to the converged value for each test case is shown in figures A.1 to A.3.

The convergence of the error in lift and drag as the primal solution converges shows that while that while there are some oscillations, the trend is generally linear on a log-log plot. Futhermore, the slopes of the curve is roughly one, as indicated by an $y = x$ dashed line. Over the course of convergence, if the residual norm drops an order of magnitude, so does the output error. In effect,

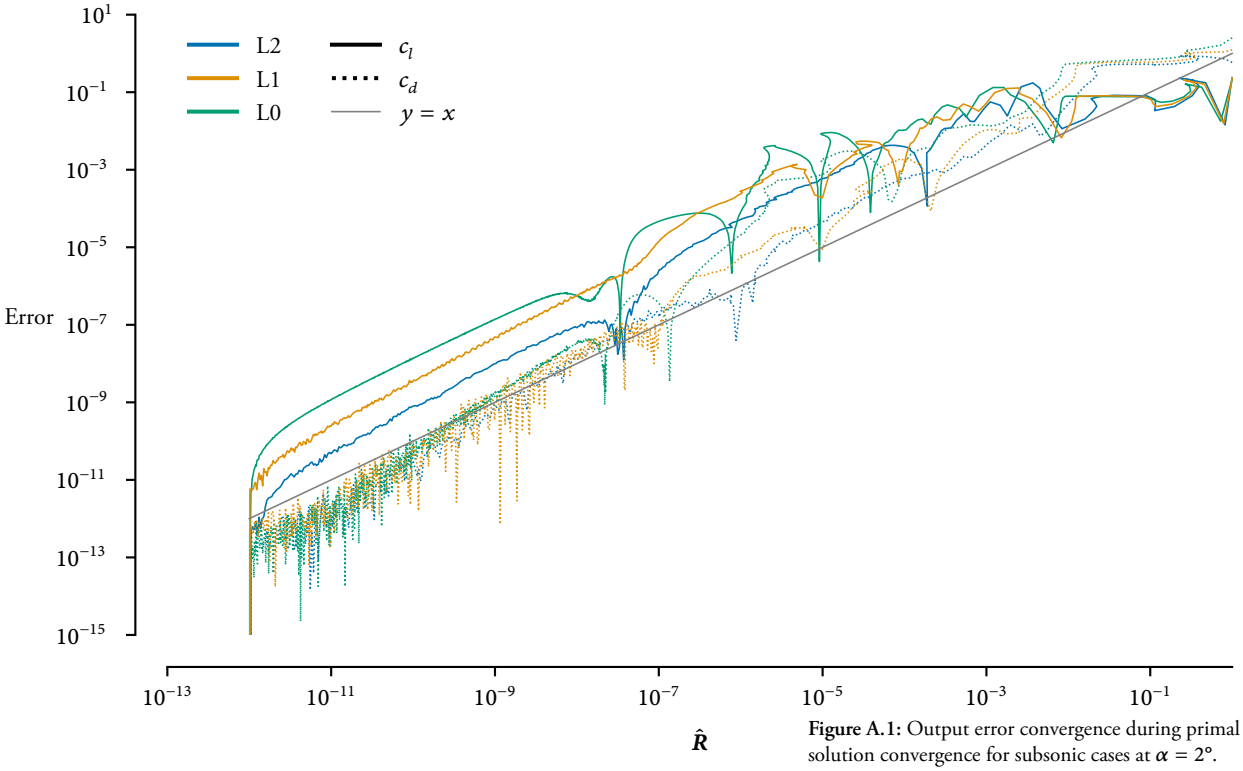


Figure A.1: Output error convergence during primal solution convergence for subsonic cases at $\alpha = 2°$.

$\Delta\epsilon \approx \Delta||\boldsymbol{R}||_2$. The linear relationship between residual norm and error also indicates that the linearized error estimate with the adjoint should provide accurate and useful error estimates. This result guides our adaptation strategy.

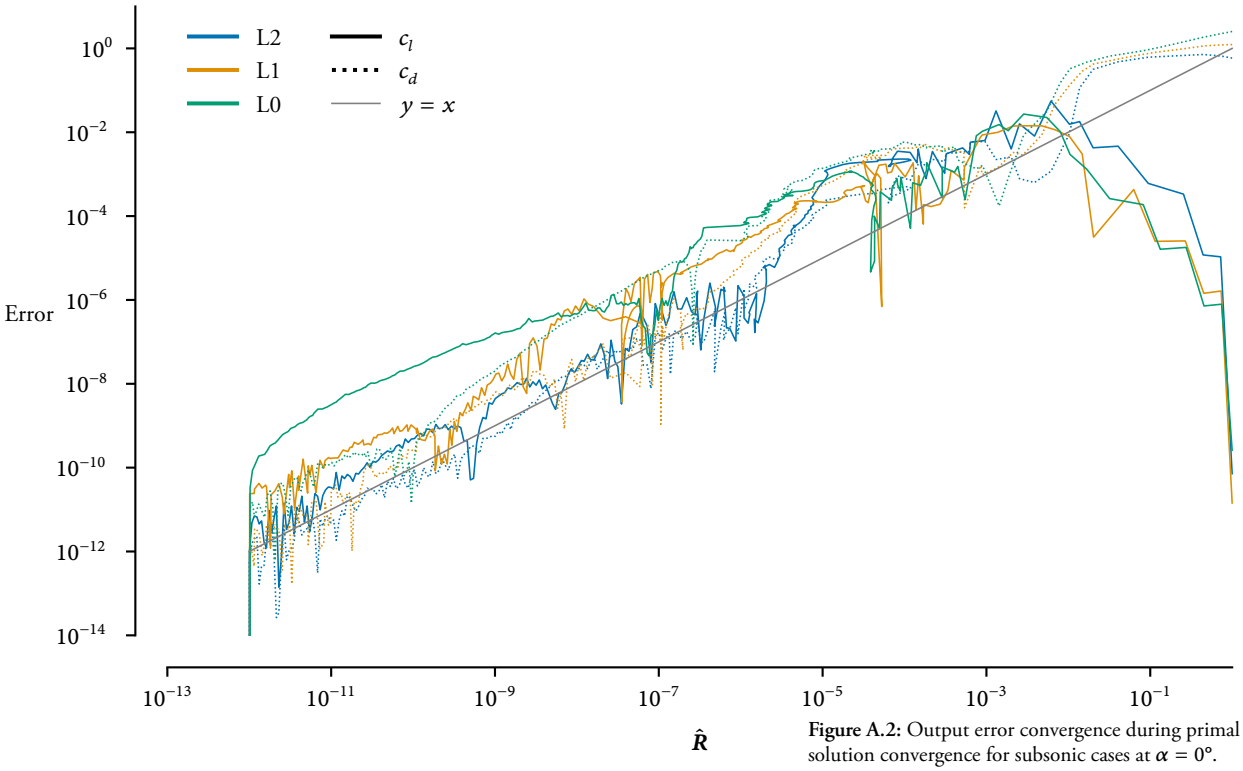


Figure A.2: Output error convergence during primal solution convergence for subsonic cases at $\alpha = 0°$.

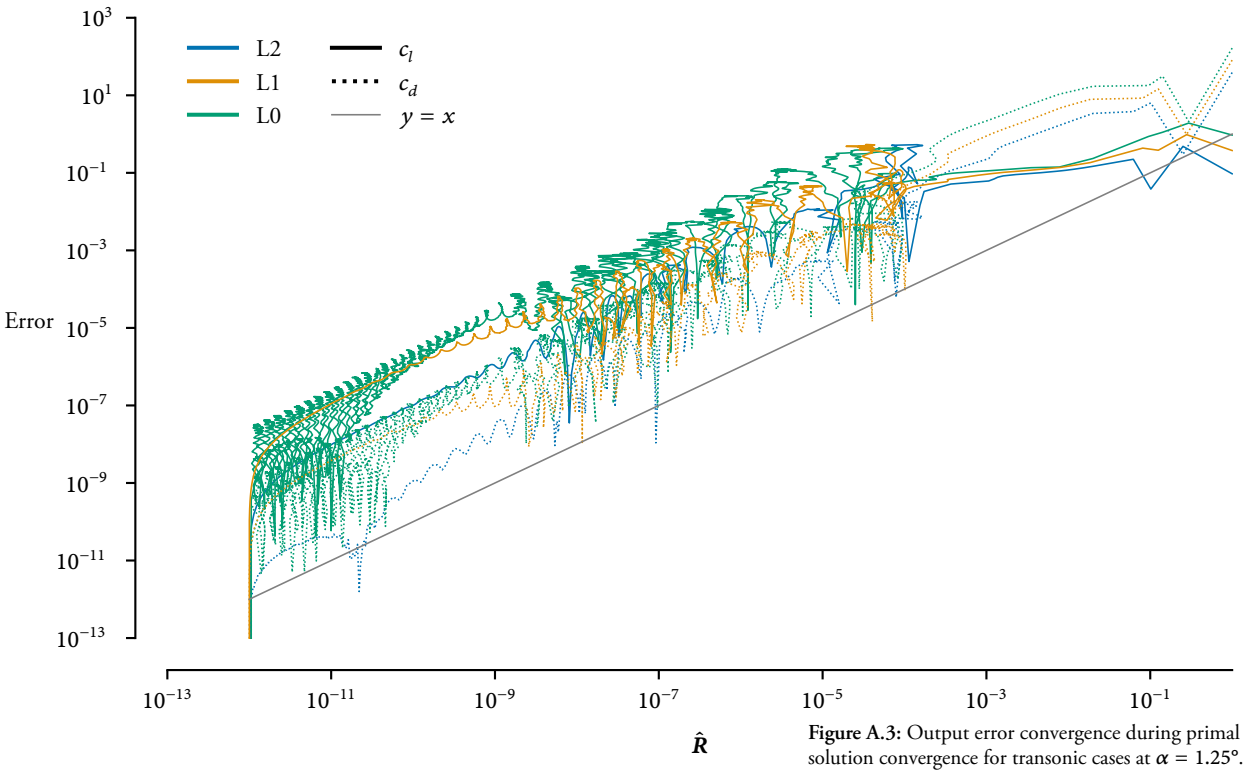


Figure A.3: Output error convergence during primal solution convergence for transonic cases at $\alpha = 1.25°$.

Appendix B

Error Propagation Techniques

The goal of error propagation is to propagate probability distributions of input parameters to output parameters. To be more precise, we start with a function

$$y = f(\boldsymbol{x}) \tag{B.1}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_x}$, n_x is the number of inputs, and y is a scalar output quantity. Suppose first that each x_i is independent, and prescribed with a PDF $g_i(x)$. The goal is to find the resulting PDF on y. A schematic of this process is shown in figure B.1.

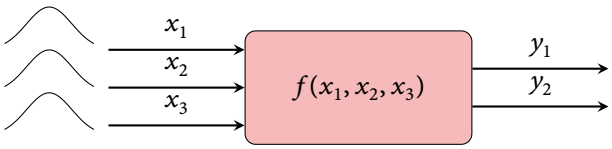


Figure B.1: A schematic showing the error propagation process. A black-box function f receives probabilistic inputs x_i, resulting in a probability distribution on the output y. The goal is to find this probability distribution given the distributions on $\boldsymbol{x}$.

In practice, the components of the input $\boldsymbol{x}$ are often correlated, with a covariance matrix Σ.

B.1 Monte Carlo methods

The most obvious approach would be to draw independent samples $\boldsymbol{x}_j$ according to $g_i(\boldsymbol{x})$ for each x_i, and compute the $y_j = f(\boldsymbol{x}_j)$. We use i to denote each component of the input vector $\boldsymbol{x}$, and j to denote each sample through the Monte Carlo process. After sufficient samples (say n_s samples) have been collected, we can then compute the various statistical moments such as mean and variance on the population y_j. Naturally,

sampling-based methods can easily deal with non-Gaussian distributions or correlated inputs, by simply sampling from the desired distribution.

It is well known that Monte Carlo methods converge independently of n_x, and therefore commonly used for problems with large dimensions [214]. The method is also applicable to any general probability distribution, and easily extended to the case where x_i are correlated by sampling from the joint probability distribution $g(\boldsymbol{x})$. We also require no knowledge of f itself, simply treating it as a black-box function which is repeatedly evaluated. However, the convergence rate of Monte Carlo is proportional to $1/\sqrt{n_s}$, and a large number of samples is typically needed [214]. The choice of n_s has significant impact on the quality of the output, and it is difficult to determine *a priori* the number of samples required [215]. In recent years, the field of *importance sampling* has been developed to address some of these concerns, by intelligently selecting the sampling points to reduce the error on the statistical outputs while using significantly fewer samples.

[214]: Yao et al. (2011), *Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles*

[214]: Yao et al. (2011), *Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles*

[215]: (2008), *Evaluation of measurement data - Supplement 1 to the "Guide to the expression of uncertainty in measurement" – Propagation of distributions using a Monte Carlo method*

B.1.1 Adaptive parallel implementation

In this work we use an adaptive approach, where an initial $n_{s,0} = 10^4$ samples are used first, and the statistics on y are computed. Then, batches of $n_{s,1} = 10^3$ samples are added and the cumulative statistics are computed. If the values of mean or variance changed by less than either an absolute or relative tolerance of 10^{-3}, then we deem the Monte Carlo sampling converged. Otherwise, additional batches of $n_{s,1}$ samples are added until convergence. In order to improve efficiency, the sampling is vectorized such that a batch of $n_{s,1}$ samples requires a single function call.

B.2 Fosm method

Fosm methods are a special case of Taylor series-based methods, where the function $f(\boldsymbol{x})$ is approximated by its Taylor series. By representing f as a polynomial, analytic solutions can be used to compute the various moments on y. In the simplest case, we use the first-order Taylor series expansion to compute the first and second moments of y from those of $\boldsymbol{x}$, hence the name fosm. In this case, both the inputs x_i and the output y is assumed to be Gaussian, so we only need to work with the first two statistical moments. Let $\boldsymbol{x}_0$ be the mean of $\boldsymbol{x}$, and $\sigma_{x_i}^2$ the variance of each x_i. The Taylor series expansion about the mean is

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + \sum_{i=1}^{n_x} \left(\frac{\partial f}{\partial x_i} \right)_{\boldsymbol{x}_0} (x_i - x_{i,0}) \tag{B.2}$$

which results in the following mean and variance values for y:

$$\mu_y = f(\boldsymbol{x}_0) \tag{B.3}$$

$$\sigma_y^2 = \sum_{i=1}^{n_x} \left(\frac{\partial f}{\partial x_i} \right)_{\boldsymbol{x}_0}^2 \sigma_{x_i}^2 \tag{B.4}$$

This approach can be extended in several ways. In the case where the inputs are correlated, the variance can be corrected by incorporating the covariance [214, 216]:

[214]: Yao et al. (2011), *Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles*
[216]: Seiler (1987), *Error Propagation for Large Errors*

$$\begin{aligned} \sigma_y^2 &= \sum_{i=1}^{n_x} \sum_{j=1}^{n_x} \left(\frac{\partial f}{\partial x_i} \right)_{\boldsymbol{x}_0} \left(\frac{\partial f}{\partial x_j} \right)_{\boldsymbol{x}_0} \operatorname{cov}(x_i, x_j) \\ &= \sum_{i=1}^{n_x} \left(\frac{\partial f}{\partial x_i} \right)_{\boldsymbol{x}_0}^2 \sigma_{x_i}^2 + 2 \sum_{i=1}^{n_x - 1} \sum_{j=i+1}^{n_x} \left(\frac{\partial f}{\partial x_i} \right)_{\boldsymbol{x}_0} \left(\frac{\partial f}{\partial x_j} \right)_{\boldsymbol{x}_0} \operatorname{cov}(x_i, x_j) \end{aligned} \tag{B.5}$$

where $\operatorname{cov}(x_i, x_j)$ is the covariance between the various components of $\boldsymbol{x}$. It is also possible to extend the method to higher-order Taylor approximations, which would also require higher moments of x_i. A derivation of second and third-order expressions without accounting for correlations is presented in [217].

[217]: Martins et al. (2011), *Generalized expressions of second and third order for the evaluation of standard measurement uncertainty*

Unfortunately, the expressions quickly grow in complexity and become intractable, especially when the inputs are correlated.

Compared with the Monte Carlo method, the FOSM method has some advantages. In its simplest form, it requires just a single function and gradient evaluation. If Hessians and higher-order derivatives are available, they also only need to be evaluated at a single point. However, the Taylor series expansion is only valid in a local neighbourhood of the function, and therefore the accuracy of this method deteriorates when the errors become large. If the inputs do not follow a Gaussian distribution, then it is difficult to incorporate higher-order statistical moments. Lastly, sometimes it may be difficult or expensive to compute higher-order derivatives of f.

B.2.1 Implementation

In this work, we use equation B.5 which accounts for the covariance of the inputs. The complex-step method [6] is used to compute the first-order derivatives, with a step size of 10^{-40}.

[6]: Martins et al. (2003), *The Complex-Step Derivative Approximation*

Note that these derivatives are constant across different fidelities—only the covariance changes. Therefore, in order to improve efficiency, we only compute these derivatives once and cache them.

Appendix C

Best Practices for Research in Scientific Computing

As shown in figure 1.1, implementation, *i.e.* software development, is one of three pillars of computational research. As such, proper software development practices should be adhered to. Unfortunately, in research this is often neglected. Too often, students write spaghetti code to meet deadlines, and the code is simply discarded when they graduate. Coupled with a generally poor background in computer science [218], and it is a surefire recipe for disaster. Johanson and Hasselbring [219] highlights many of the systemic challenges facing computational research.

[218]: Hannay et al. (2009), *How do scientists develop and use scientific software?*

[219]: Johanson et al. (2018), *Software Engineering for Computational Science: Past, Present, Future*

Luckily, these issues are well understood, and we can look to existing research and industry practices for guidance. For example, Wilson et al. [220] wrote an excellent reference containing some broad recommendations, and the authors later supplemented it with a subsequent paper which is aptly named [221]. In this section, I will briefly discuss some recommendations and lessons learned during the course of my dissertation. Some examples of projects that have focused on this aspect include [222], and the HPCMP CREATE Program [223, 224], and FUN3D [225].

[220]: Wilson et al. (2014), *Best Practices for Scientific Computing*

[221]: Wilson et al. (2017), *Good enough practices in scientific computing*

[222]: Galbraith et al. (2015), *A Verification Driven Process for Rapid Development of CFD Software*

[223]: Kendall et al. (2021), *The HPCMP CREATE™ Program management Model-Part I*

[224]: Bergeron et al. (2021), *The HPCMP CREATE™ Management Model – Part II, DevOps Principles and Practices in HPCMP CREATE™*

[225]: Zhang et al. (2022), *Component-Based Development of CFD Software FUN3D*

C.1 Code development and maintenance

While many of the topics covered here are related to the concept of "DevOps", it is a loosely-defined term [226, 227]. Instead, we will focus on some specific aspects of code development and maintenance.

[226]: Leite et al. (2019), *A Survey of DevOps Concepts and Challenges*
[227]: Jabbari et al. (2016), *What is DevOps? A Systematic Mapping Study on Definitions and Practices*

C.1.1 Version control and workflow

It should go without saying that a modern version control system (VCS) such as *Git* is imperative in the software development cycle. However, in practice this is often not done properly. For example, developers should *never* directly commit to the main branch. Instead, a development branch should be used, which allows code changes to occur in parallel and without disruption to others. It also allows for the possibility of collaboration with others. Once ready, a pull request is used to merge the code, subject to review and passage of tests. This process is sometimes called *GitHub flow*.[1]

1: `https://docs.github.com/en/get-started/quickstart/github-flow`

One key aspect of this process is the adoption of the code review process. This not only improves code quality by having additional reviewers check the code, but also facilitates broader understanding of the code. In the context of scientific computing, collaborators—often other PhD students—can use this opportunity to learn about the code. This knowledge transfer is particularly important as many codes live well past the duration of doctoral studies for the students.

C.1.2 Testing, testing, testing

Software verification and validation (V&V) is a well-discussed topic. In general, *verification* refers to whether a software is implemented correctly, while *validation* refers to the appropriateness of the computer software in modeling a real-world process [228].

[228]: Oberkampf et al. (2010), *Verification and Validation in Scientific Computing*

Here we focus on software verification, which is achieved primarily through testing.

As scientific software is commonly hierarchical, the accompanying tests should reflect the same structure. Broadly speaking, there are two types of software tests [228]. At the basic level, *unit tests* are used to verify the implementation of single functions. A common example given is the Sutherland's viscosity law given by Kleb and Wood [229], or Roe's flux to check that upwinding is used correctly for supersonic inputs. The key aspects of unit testing are:

[228]: Oberkampf et al. (2010), *Verification and Validation in Scientific Computing*

[229]: Kleb et al. (2006), *Computational Simulations and the Scientific Method*

- The code being tested is simple and singular in function
- The inputs and expected outputs can be verified easily, typically consisting of analytic values

Regression tests, on the other hand, are used to compare code outputs to that from a previous version of the code [228]. As a result, *failing a regression test does not necessarily indicate the presence of a bug*, merely that the output has changed over time. It is equally likely that the reference values, trained from an earlier version, were wrong. Regression tests are commonly used on larger pieces of code, which due to more complex interactions, cannot have simple analytic test cases. They can also be used on the entire code, for example tracking the drag value of a wing.

[228]: Oberkampf et al. (2010), *Verification and Validation in Scientific Computing*

Since code outputs are subject to various sources of numerical noise (further discussed in appendix C.2), comparisons with reference values must be made using tolerances. A common approach is to employ both an absolute tolerance τ_{abs} and relative tolerance τ_{rel}, such that a value v is approximately equal to the reference value v_{ref} if

$$|v - v_{\text{ref}}| <= \tau_{\text{abs}} + \tau_{\text{rel}} \times |v_{\text{ref}}|. \tag{C.1}$$

By using both tolerances, this works well no matter if the reference value is very large or near zero. Typically, we use the same

value for both τ_{abs} and τ_{rel}. Note that this expression is *not* symmetric about v and v_{ref}, since the relative tolerance is treated relative to the reference value.

Naturally, the choice of test tolerances is a crucial aspect. Too restrictive, and tests become *flaky*, failing for spurious reasons which can add overhead during software development. Too loose, and the tests become less useful as true bugs may be masked by the numerical noise, leading to test passage. Some of these challenges are discussed further in appendix C.2 and appendix C.3.

Once tests are in place, the *test coverage* can be computed as the percentage of LOC that are tested. Practically speaking, only unit tests should be included in this count, as regression tests do not indicate the code is bug-free. The coverage is also merely an upper bound, since not every executed LOC is tested. Nevertheless, it gives good indication of how many LOC is *definitely* not being tested.

During the course of software development, it is also common to manually perform more complex software verification tasks. For example, when solving PDEs numerically, the achieved order of convergence can be compared to theoretical results. This can be done more rigorously using the method of manufactured solutions, commonly used in verification of high-order CFD methods [230].

[230]: Wang et al. (2013), *High-order CFD methods: current status and perspective*

While tests can be written after the fact, as done in the V&V stage of software development, it is often more effective to write tests during, or even before any "real" code is written. This approach, called *test-driven development*, can be a more effective way to develop software [231].

[231]: Nanthaamornphong et al. (2015), *Test-Driven Development in scientific software: A survey*

C.1.3 Static code analysis and formatting

In addition to unit and regression tests, others tests can be performed by dedicated tools. In contrast to dynamic analysis (*i.e.* at runtime), static analysis tools comb through the source code to look for potential errors. Often called linters, these tools are especially important for languages such as Python that do not have an explicit compilation step, where many errors can be caught. We use two popular Python linters, flake8[2] and pylint.[3]

2: https://flake8.pycqa.org
3: https://pylint.org/

Code formatting can be a contentious topic, as different developers have their own preferences. However, developers should not spend time arguing over stylistic changes. A software project should have—from its inception—an established convention for code style. For example, PEP 8[4] is a popular choice for many Python projects. To simplify matters and to avoid spending time manually adjusting the formatting, automatic code formatters should be used. We use black[5] for Python, ClangFormat[6] for C/C++, and fprettify[7] for Fortran. Many linters are also capable of raising warnings when names do not follow a particular naming convention.

4: https://peps.python.org/pep-0008/

5: https://black.readthedocs.io
6: https://clang.llvm.org/docs/ClangFormat.html
7: https://github.com/pseewald/fprettify

C.1.4 Continuous integration

Tests are fairly useless if not executed often. While it is useful to have these tests available for developers, their utility is only fully realized when employed in an automated testing framework. Continuous integration is typically done via automated testing at a high frequency. We run the full test suite on every pull request to a repository, including the following tests:

- Unit and regression tests, both in serial and parallel
- Linters
- Code formatters
- Coverage testing, to ensure code changes and additions have associated tests

Table C.1: An example of the build and test matrix used for continuous integration of MACH. The versions of other dependencies are given in the corresponding column of table C.2

	1	2	3	4	5	6
OS	Centos 7	Ubuntu 22.04	Ubuntu 22.04	Ubuntu 20.04	Ubuntu 20.04	Ubuntu 18.04
compilers	Intel 2018	GCC 11.2.0	GCC 11.2.0	GCC 9.3.0	GCC 9.3.0	GCC 7.4.0
MPI	Intel 19.0.7	OpenMPI 4.0.5	OpenMPI 3.1.6	OpenMPI 4.0.5	OpenMPI 3.1.6	Intel 19.0.7
other	stable	latest	stable	latest	stable	stable

- AD tests, to ensure the automatically differentiated code is consistent with the source code.[8]

8: We track AD code in the repository such that the AD tool is not needed for end users to compile the package

In addition, the entire code base of MACH, consisting of about 20 packages, is tightly integrated. Testing each package in isolation may not catch bugs downstream. Therefore, we also test the entire codebase nightly on dedicated hardware. These include the unit and regression tests above, and several larger regression tests that perform aerodynamic and aerostructural optimizations.

On top of this, we support multiple versions of dependencies, such as different MPI implementations and compilers. The combinatorial nature of dependency management can quickly grow to be intractable. Therefore, we limit ourselves to a handful of relevant combinations that may be found on our local workstations or HPC systems. Table C.1 illustrates the complexity of this task.

The definition of the "stable" and "latest" versions of other dependencies are given in table C.2.

It is easy to see that without careful consideration, the number of combinations can quickly grow out of hand. We use Docker containers to facilitate dependency management, and to improve reproducibility (further discussed in appendix C.3). Each of the images listed in table C.1 are built in Docker, using jinja-templated Dockerfiles. The same containers are then published online, and can be used directly by end-users on either a local workstation, or converted to Singularity containers [232] for use on HPC systems.

	stable	latest
Python	3.8.9	3.9.6
NumPy	1.19.2	1.21.5
SciPy	1.5.4	1.7.3
CGNS	4.2.0	4.3.0
PETSc	3.14.6	3.15.5
SNOPT	7.7.1	7.7.7
OpenMDAO	3.18.0	3.20.0

Table C.2: Versions of additional dependencies.

[232]: Kurtzer et al. (2017), *Singularity: Scientific containers for mobility of compute*

C.1.5 Documentation

Knowledge only exists if it is written down. Without adequate documentation, software cannot attract users, and ongoing development becomes increasingly difficult until it falls into obsolescence. However, documentation is a catch-all term that encompasses different things with different goals in mind, catering to both users and developers. Here, we roughly follow the classification system from Divio,[9] which is broadly split into the following sections.

9: https://documentation.divio.com/

Tutorials is an introductory guide, similar to a *Quick Start* document that guides through a basic use case.

How-to guides describe how to accomplish specific tasks, assuming the reader is familiar with the basic concepts already.

Reference guides cover the code base comprehensively. Typically this involves application programming interface (API) documentation in the form of parsed docstrings.

Explanation provides the bigger picture, often in the form of a theory guide or developer's guide. They can describe fundamental concepts, the software architecture, and why things are set up a certain way. These texts may not be necessary to use the code, but give a deeper understanding for advanced users or collaborators. They're also a great way for developers to document the code in a holistic way.

Documentation is one of the hardest aspects of software to do well, since there are no metrics or tests to enforce, apart from tools such as `pydocstyle`.[10] But if done poorly, and those responsible for the code move away from the project, then the chances of resurrecting the code and keeping it relevant are slim.

10: http://www.pydocstyle.org/

C.2 Sources of numerical errors

In scientific computing, numerical errors can come from a number of sources [233]. As discussed in section 2.4, this includes both precision and accuracy. In the context of scientific computing, precision refers to the random noise in outputs when executing the program multiple times—either sequentially, or on multiple computer systems. In contrast, accuracy refers to the systematic error between the outputs and the expected or true numerical output.

[233]: Kennedy et al. (2001), *Bayesian Calibration of Computer Models*

The first and most obvious source of error is the use of floating point arithmetic, which allows digital representation of floating point numbers up to machine precision ϵ_{mp}. Typical double precision variables are stored using 64 bits, leading to $\epsilon_{\text{mp}} \sim 10^{-16}$. However, in many cases, this can lead to significant losses of accuracy much beyond ϵ_{mp}. For example, if a linear system has a condition number of 10^9—not uncommon in FEM with shell elements [21]—then the solution may be accurate to only 10^7 at most. Certain iterative algorithms may also lead to rapid loss of accuracy, such as the well-known classical Gram-Schmidt algorithm [234]. However, IEEE-compliant arithmetic should not cause any loss of precision, meaning every invocation of the program should still return byte-identical output.

[21]: Kenway et al. (2014), *Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Adjoint Derivative Computations*

[234]: Trefethen et al. (1997), *Numerical Linear Algebra*

Next are the compiler optimizations. In many cases, the compiler is able to speed up the code significantly if it is allowed to be non-compliant when dealing with floating point arithmetic. Some of these optimizations include unrolling loops or taking advantage of newer vectorization instruction sets, which introduce parallelism at the cost of precision and accuracy. The amount of noise introduced here depends largely on the specific code, but it can add up significantly.[11] For example, for GCC the compiler option `-Ofast` will introduce a number of unsafe operations, and Corden and Kreitzer [235] discuss many similar issues for Intel compilers.

11: `https://simonbyrne.github.io/notes/fastmath/`

[235]: Corden et al. (2009), *Consistency of floating-point results using the intel compiler or why doesn't my application always give the same answer*

Further errors are introduced using higher-level parallelism, such as MPI for distributed-memory or Open Multi-Processing (OpenMP) for shared memory parallelism. This will often introduce noise when running the same program with different numbers of processors or threads, even considering simple operations such as reductions [236, 237]. In more complex programs, domain decomposition is often required, using tools such as METIS [238]. In such cases, changing the number of processors will fundamentally alter the resultant linear and nonlinear systems to be solved, further introducing errors.

[236]: Balaji et al. (2013), *On the Reproducibility of MPI Reduction Operations*
[237]: Collange et al. (2015), *Numerical reproducibility for the parallel reduction on multi- and many-core architectures*
[238]: Karypis et al. (1998), *A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs*

Next is the stochasticity introduced into the code as part of the numerical algorithm, often in the form of pseudo-random number generator (PRNG). Many algorithms that are central to computational science, such as Monte Carlo or stochastic optimizers rely on random numbers. Of course, the results can be made deterministic by fixing the seed, but this is not commonly done.

Finally, there are some factors that may be outside the control of the end user. For example, the operating system (OS), compiler version, and versions of all dependencies may change over time, or from one HPC system to another. Any change in the entire software stack may result in differences in software outputs, and the root cause is difficult if not impossible to track down. Since scientific computing software are highly complex, small changes can propagate and cause bigger changes downstream. For example, a small change in a single floating point variable can cause the program to go down a different path due to a conditional statement.

There are two main ways to address this issue. The first is by managing and controlling the amount of error introduced throughout the code, such that the final error remains acceptable to the level required for engineering applications. For example, while noise may be present in the model, leading the optimizer to trace a different path in the design space, the final optimum should

still agree reasonably well for a unimodal problem if the various tolerances are sufficiently small. This is oftentimes achievable for ASO, but as model complexity increases, the problem may become too difficult to tackle. It is also generally difficult to control the computational environment beyond a certain time frame, making reproducibility difficult after a long period of time.

The second approach is to attempt to make the entire simulation and optimization environment *reproducible*. While this approach does not address the fundamental issue with not knowing how much error was introduced into the solution, the output will at least become deterministic, even across systems. This approach is discussed in the next section, and of course both approaches can be combined to improve the overall behaviour of the program.

C.3 Reproducibility

There are many definitions of reproducibility. Here we borrow from [239], and consider it as an independent recreation using the same dataset and tools. Figure C.1 shows the full matrix of possible combinations.

[239]: Ezer et al. (2019), *Data science for the scientific life cycle*

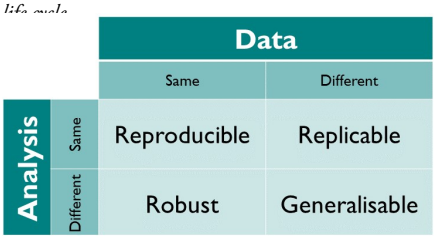


Figure C.1: Possible definitions of reproducibility, taken from [239] and made available under the Creative Commons Attribution 4.0 International (CC BY 4.0) license.

While there is strong motivation and desire for reproducible science [240], this is difficult to accomplish in practice. In the context of computational research, many factors impede such attempts, which are discussed by Ivie and Thain [241].

[240]: Munafò et al. (2017), *A manifesto for reproducible science*

[241]: Ivie et al. (2018), *Reproducibility in Scientific Computing*

Here, I discuss some practical strategies to improve the current state of computational research.[12] This is achieved by keeping track of the following components:

12: Some of these are aspirational, and have not yet been put to use in this dissertation.

- Computational environment
- Workflow
- Inputs and outputs

To reproduce the computational environment, we use a container to isolate it from the host system. These containers can then be archived and reused at a later date to regenerate the same dataset. Many container technologies exist, such as Docker [242] and Singularity [232]. Singularly has the advantage of being compatible with many HPC systems, offering a seamless computing experience. Now, this does require the generation and maintenance of containers as a deployment step, but this is often not too difficult since there are benefits to using containers for software testing, particularly different combinations of dependencies and build tools as discussed in appendix C.1.4.

[242]: Merkel et al. (2014), *Docker: Lightweight linux containers for consistent development and deployment*

[232]: Kurtzer et al. (2017), *Singularity: Scientific containers for mobility of compute*

For workflow and data, several frameworks have been developed to address precisely this need, such as Data Version Control (DVC) [243], ReproZip [244], DataLad [245], and signac [246]. However, in addition to this, a long-term archival system is needed to preserve all the data necessary, including the containers. Many journals are now starting to require both data and software to be made publicly-available when publishing papers. This goes beyond open sourcing the respective software and is a good first step towards true reproducible computational science.

[243]: Kuprieiev et al. (2022), *DVC: Data Version Control - Git for Data & Models*

[244]: Rampin et al. (2016), *ReproZip: The Reproducibility Packer*

[245]: Halchenko et al. (2021), *DataLad: Distributed system for joint management of code, data, and their relationship*

[246]: Adorf et al. (2018), *Simple data and workflow management with the signac framework*

C.3.1 RepoState: a package to facilitate reproducible computational research in Python

While a production-ready image can be used to generate computational result for a paper, the same cannot be done in the midst of active development. Therefore, to facilitate reproducibility—particularly on HPC environments, the package RepoState was developed. The original author was Gaetan Kenway, but I have since modified and updated a significant portion of the code.

In contrast to many other approaches that attempt to generate metadata regarding the computing environment, RepoState accomplishes this by directly inspecting the Python package that

is imported. This is a key distinction from methods that query the package manager, for example parsing the output of `pip list`. Since Python allows for in-place installations—in fact the preferred methods for many developers—the version from `pip` may not match the version of the code which is executed. On top of this, the version specifier may not be sufficient for actively-developed code, which can have either a commit which is not versioned, or even uncommitted changes. Therefore, RepoState will inspect each imported package and check if it belongs in a git repository. If so, it will also record the current git commit, and a patch file if any uncommitted changes are present. Crucially, this step also includes the runscript itself if it is tracked by git. This provides an easy way to identify changes in the runscript between subsequent executions.

In addition to this information for each imported Python package, additional metadata is recorded. These include the current system time, the list of loaded modules in the compute environment, and the list of environment variables.[13] The full output of `pip freeze` is also saved, in order to track the versions of secondary dependencies. All the metadata generated should be stored along with the outputs, and archived appropriately.

Now, say it is a year later and we wish to reproduce the same results. The stored metadata can now be read back by RepoState. For each git-tracked package, it is then able to roll back to the precise version of the code used, applying any patches as necessary. The same is also done for the runscript itself. The user can then compile and install any code as necessary, and rerun the script to regenerate the output.

13: For many HPC systems, the environment variables from the MPI implementation will contain useful information such as the number of processors requested, and even the command executed.

Bibliography

[1] J. R. R. A. Martins. "Aerodynamic Design Optimization: Challenges and Perspectives". In: *Computers & Fluids* 239 (May 2022), p. 105391. DOI: 10.1016/j.compfluid.2022.105391 (cited on pages 1, 20).

[2] J. R. R. A. Martins and A. Ning. *Engineering Design Optimization*. Cambridge, UK: Cambridge University Press, 2021 (cited on pages 2–4, 12, 17, 18, 22, 24, 25, 40, 59).

[3] X. He, J. Li, C. A. Mader, A. Yildirim, and J. R. R. A. Martins. "Robust aerodynamic shape optimization—from a circle to an airfoil". In: *Aerospace Science and Technology* 87 (Apr. 2019), pp. 48–61. DOI: 10.1016/j.ast.2019.01.051 (cited on pages 2, 41).

[4] I. Newton. *Philosophiæ Naturalis Principia Mathematica*. Londini: Jussu Societatis Regiæ ac Typis Josephi Streater. Prostat apud plures Bibliopolas, 1687 (cited on page 3).

[5] W. Squire and G. Trapp. "Using complex variables to estimate derivatives of real functions". In: *SIAM Review* 40.1 (1998), pp. 110–112. DOI: 10.1137/S003614459631241X (cited on page 4).

[6] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. "The Complex-Step Derivative Approximation". In: *ACM Transactions on Mathematical Software* 29.3 (Sept. 2003), pp. 245–262. DOI: 10.1145/838250.838251 (cited on pages 4, 178).

[7] J. S. Arora and E. J. Haug. "Efficient Optimal Design of Structures by Generalized Steepest Descent Programming". In: *International Journal for Numerical Methods in Engineering* 10 (1976), pp. 747–766 (cited on page 4).

[8] O. Pironneau. "On Optimum Profiles in Stokes Flow". In: *Journal of Fluid Mechanics* 59.01 (1973), pp. 117–128. DOI: 10.1017/S002211207300145X (cited on page 4).

[9] A. Jameson. "Aerodynamic Design via Control Theory". In: *Journal of Scientific Computing* 3.3 (Sept. 1988), pp. 233–260. DOI: 10.1007/BF01061285 (cited on page 4).

[10] Z. Lyu, G. K. Kenway, C. Paige, and J. R. R. A. Martins. "Automatic Differentiation Adjoint of the Reynolds-Averaged Navier–Stokes Equations with a Turbulence Model". In: *21st AIAA Computational Fluid Dynamics Conference*. San Diego, CA, July 2013. DOI: 10.2514/6.2013-2581 (cited on page 4).

[11] Z. Lyu and J. R. R. A. Martins. "Aerodynamic Design Optimization Studies of a Blended-Wing-Body Aircraft". In: *Journal of Aircraft* 51.5 (Sept. 2014), pp. 1604–1617. DOI: 10.2514/1.C032491 (cited on page 4).

[12] Y. Shi, C. A. Mader, S. He, G. L. O. Halila, and J. R. R. A. Martins. "Natural Laminar-Flow Airfoil Optimization Design Using a Discrete Adjoint Approach". In: *AIAA Journal* 58.11 (Nov. 2020), pp. 4702–4722. DOI: 10.2514/1.J058944 (cited on page 4).

[13] P. He, A. J. Luder, C. A. Mader, K. J. Maki, and J. R. R. A. Martins. "A Time-Spectral Adjoint Approach for Aerodynamic Shape Optimization Under Periodic Wakes". In: *AIAA SciTech Forum*. AIAA. Orlando, FL, Jan. 2020. DOI: 10.2514/6.2020-2114 (cited on page 4).

[14] G. K. W. Kenway and J. R. R. A. Martins. "Buffet-Onset Constraint Formulation for Aerodynamic Shape Optimization". In: *AIAA Journal* 55.6 (June 2017), pp. 1930–1947. DOI: 10.2514/1.J055172 (cited on pages 4, 10).

[15] N. Garg, G. K. W. Kenway, Z. Lyu, J. R. R. A. Martins, and Y. L. Young. "High-fidelity Hydrodynamic Shape Optimization of a 3-D Hydrofoil". In: *Journal of Ship Research* 59.4 (Dec. 2015), pp. 209–226. DOI: 10.5957/JOSR.59.4.150046 (cited on page 4).

[16] G. K. W. Kenway, C. A. Mader, P. He, and J. R. R. A. Martins. "Effective Adjoint Approaches for Computational Fluid Dynamics". In: *Progress in Aerospace Sciences* 110 (Oct. 2019), p. 100542. DOI: 10.1016/j.paerosci.2019.05.002 (cited on pages 4, 11, 22, 30).

[17] R. T. Haftka. "Optimization of Flexible Wing Structures Subject to Strength and Induced Drag Constraints". In: *AIAA Journal* 15.8 (1977), pp. 1101–1106. DOI: 10.2514/3.7400 (cited on pages 5, 95).

[18] J. Sobieszczanski–Sobieski. "Sensitivity of Complex, Internally Coupled Systems". In: *AIAA Journal* 28.1 (1990), pp. 153–160. DOI: 10.2514/3.10366 (cited on page 5).

[19] J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther. "A Coupled-Adjoint Sensitivity Analysis Method for High-Fidelity Aero-Structural Design". In: *Optimization and Engineering* 6.1 (Mar. 2005), pp. 33–62. DOI: 10.1023/B:OPTE.0000048536.47956.62 (cited on pages 5, 26, 124).

[20] J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther. "High-Fidelity Aerostructural Design Optimization of a Supersonic Business Jet". In: *Journal of Aircraft* 41.3 (May 2004), pp. 523–530. DOI: 10.2514/1.11478 (cited on pages 5, 7, 31).

[21] G. K. W. Kenway, G. J. Kennedy, and J. R. R. A. Martins. "Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Adjoint Derivative Computations". In: *AIAA Journal* 52.5 (May 2014), pp. 935–951. DOI: 10.2514/1.J052255 (cited on pages 5, 11, 24, 28, 31, 75, 124, 169, 186).

[22] Z. J. Zhang, S. Khosravi, and D. W. Zingg. "High-fidelity aerostructural optimization with integrated geometry parameterization and mesh movement". In: *Structural and Multidisciplinary Optimization* 55.4 (Aug. 2016), pp. 1217–1235. DOI: 10.1007/s00158-016-1562-7 (cited on page 6).

[23] R. Olivanti and J. Brézillon. "On the Benefits of Engaging Coupled-Adjoint to Perform High-Fidelity Multipoint Aircraft Shape Optimization". In: *AIAA AVIATION 2021 FORUM*. Ameri-

can Institute of Aeronautics and Astronautics, July 2021, p. 3072. DOI: 10.2514/6.2021-3072 (cited on page 6).

[24] M. Carini, C. Blondeau, N. Fabbiane, M. Meheut, M. Abu-Zurayk, J. M. Feldwisch, C. Ilic, and A. Merle. "Towards industrial aero-structural aircraft optimization via coupled-adjoint derivatives". In: *AIAA AVIATION 2021 FORUM*. American Institute of Aeronautics and Astronautics, July 2021, p. 3074. DOI: 10.2514/6.2021-3074 (cited on page 6).

[25] G. K. W. Kenway, G. J. Kennedy, and J. R. R. A. Martins. "Aerostructural Optimization of the Common Research Model Configuration". In: *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. AIAA 2014-3274. Atlanta, GA, June 2014. DOI: 10.2514/6.2014-3274 (cited on page 6).

[26] G. W. K. Kenway and J. R. R. A. Martins. "High-fidelity aerostructural optimization considering buffet onset". In: *Proceedings of the 16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. Dallas, TX, June 2015. DOI: 10.2514/6.2015-2790 (cited on page 6).

[27] T. R. Brooks, G. K. W. Kenway, and J. R. R. A. Martins. "Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings". In: *AIAA Journal* 56.7 (July 2018), pp. 2840–2855. DOI: 10.2514/1.J056603 (cited on pages 6, 11, 13, 33, 40, 96, 151, 159).

[28] T. R. Brooks, J. R. R. A. Martins, and G. J. Kennedy. "High-fidelity Aerostructural Optimization of Tow-steered Composite Wings". In: *Journal of Fluids and Structures* 88 (July 2019), pp. 122–147. DOI: 10.1016/j.jfluidstructs.2019.04.005 (cited on pages 6, 9, 11).

[29] T. R. Brooks, J. R. R. A. Martins, and G. J. Kennedy. "Aerostructural Trade-offs for Tow-steered Composite Wings". In: *Journal of Aircraft* 57.5 (Sept. 2020), pp. 787–799. DOI: 10.2514/1.C035699 (cited on page 6).

[30] D. A. Burdette and J. R. R. A. Martins. "Design of a Transonic Wing with an Adaptive Morphing Trailing Edge via Aerostructural Optimization". In: *Aerospace Science and Technology* 81 (Oct. 2018), pp. 192–203. DOI: 10.1016/j.ast.2018.08.004 (cited on page 6).

[31] D. A. Burdette and J. R. R. A. Martins. "Impact of Morphing Trailing Edge on Mission Performance for the Common Research Model". In: *Journal of Aircraft* 56.1 (Jan. 2019), pp. 369–384. DOI: 10.2514/1.C034967 (cited on page 6).

[32] A. Yildirim, J. S. Gray, C. A. Mader, and J. R. R. A. Martins. "Boundary Layer Ingestion Benefit for the STARC-ABL Concept". In: *Journal of Aircraft* 59.4 (July 2022), pp. 896–911. DOI: 10.2514/1.C036103 (cited on pages 6, 29, 39).

[33] J. R. R. A. Martins and A. B. Lambe. "Multidisciplinary Design Optimization: A Survey of Architectures". In: *AIAA Journal* 51.9 (Sept. 2013), pp. 2049–2075. DOI: 10.2514/1.J051895 (cited on pages 6, 24–26).

[34] N. P. Tedford and J. R. R. A. Martins. "Benchmarking Multidisciplinary Design Optimization Algorithms". In: *Optimization and Engineering* 11.1 (Feb. 2010), pp. 159–183. DOI: 10.1007/s11081-009-9082-6 (cited on page 6).

[35] J. R. R. A. Martins and J. T. Hwang. "Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models". In: *AIAA Journal* 51.11 (Nov. 2013), pp. 2582–2599. DOI: 10.2514/1.J052184 (cited on page 6).

[36] 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". In: *Structural and Multidisciplinary Optimization* 59.4 (Apr. 2019), pp. 1075–1104. DOI: 10.1007/s00158-019-02211-z (cited on pages 6, 24, 31).

[37] J. Anibal, C. A. Mader, and J. R. R. A. Martins. "Aerodynamic shape optimization of an electric aircraft motor surface heat exchanger with conjugate heat transfer constraint". In: *International Journal of Heat and Mass Transfer* 189 (June 2022), p. 122689. DOI: 10.1016/j.ijheatmasstransfer.2022.122689 (cited on pages 7, 29).

[38] A. Yildirim, J. S. Gray, C. A. Mader, and J. R. R. A. Martins. "Coupled Aeropropulsive Design Optimization of a Podded Electric Propulsor". In: *AIAA Aviation Forum*. Aug. 2021. DOI: 10.2514/6.2021-3032 (cited on page 7).

[39] K. E. Jacobson and B. K. Stanford. "Flutter-Constrained Optimization with the Linearized Frequency-Domain Approach". In: *AIAA Science and Technology Forum and Exposition, AIAA SciTech Forum 2022*. 2022. DOI: 10.2514/6.2022-2242 (cited on page 7).

[40] F. Gallard, R. Lafage, C. Vanaret, B. Pauwels, D. Guénot, P.-.-J. Barjhoux, V. Gachelin, and A. Gazaix. "GEMS: A Python Library for Automation of Multidisciplinary Design Optimization Process Generation". In: *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. June 2018. DOI: 10.2514/6.2018-0657 (cited on page 7).

[41] M. Meheut. "Multidisciplinary Adjoint-based Optimizations in the MADELEINE Project: Overview and Main Results". In: *AIAA AVIATION 2021 FORUM*. American Institute of Aeronautics and Astronautics, July 2021, p. 3052. DOI: 10.2514/6.2021-3052 (cited on page 7).

[42] G. K. W. Kenway and J. R. R. A. Martins. "Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration". In: *Journal of Aircraft* 51.1 (Jan. 2014), pp. 144–160. DOI: 10.2514/1.C032150 (cited on pages 7, 28, 30, 33, 40, 140, 159).

[43] P. D. Bravo-Mosquera, F. M. Catalano, and D. W. Zingg. "Unconventional aircraft for civil aviation: A review of concepts and design methodologies". In: *Progress in Aerospace Sciences* 131 (May 2022), p. 100813. DOI: 10.1016/j.paerosci.2022.100813 (cited on pages 7, 8).

[44] C. David, S. Delbecq, S. Defoort, P. Schmollgruber, E. Benard, and V. Pommier-Budinger. "From FAST to FAST-OAD: An open source framework for rapid Overall Aircraft Design". In: *IOP Conference Series: Materials Science and Engineering*. Vol. 1024. 1. IOP Publishing. 2021, p. 012062 (cited on page 7).

[45] B. J. Brelje and J. R. R. A. Martins. "Development of a Conceptual Design Model for Aircraft Electric Propulsion with Efficient Gradients". In: *Proceedings of the AIAA/IEEE Electric Aircraft Technologies Symposium*. Cincinnati, OH, July 2018. DOI: 10.2514/6.2018-4979 (cited on page 7).

[46] E. J. Adler and J. R. R. A. Martins. "Efficient Aerostructural Wing Optimization Considering Mission Analysis". In: *Journal of Aircraft* (Dec. 2022). DOI: 10.2514/1.c037096 (cited on page 7).

[47] J. P. Jasa, J. T. Hwang, and J. R. R. A. Martins. "Open-source coupled aerostructural optimization using Python". In: *Structural and Multidisciplinary Optimization* 57.4 (Apr. 2018), pp. 1815–1827. DOI: 10.1007/s00158-018-1912-8 (cited on page 8).

[48] T. Chau and D. W. Zingg. "Aerodynamic Design Optimization of a Transonic Strut-Braced-Wing Regional Aircraft". In: *Journal of Aircraft* 59.1 (Jan. 2022), pp. 253–271. DOI: 10.2514/1.c036389 (cited on page 8).

[49] S. Wöhler, G. Atanasov, D. Silberhorn, B. Fröhler, and T. Zill. "Preliminary aircraft design within a multidisciplinary and multifidelity design environment". In: *Aerospace Europe Conference 2020*. 2020 (cited on page 8).

[50] T. MacDonald, M. Clarke, E. M. Botero, J. M. Vegh, and J. J. Alonso. "SUAVE: An Open-Source Environment Enabling Multi-Fidelity Vehicle Optimization". In: *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2017. DOI: 10.2514/6.2017-4437 (cited on pages 8, 31).

[51] N. P. Werter and R. De Breuker. "A novel dynamic aeroelastic framework for aeroelastic tailoring and structural optimisation". In: *Composite Structures* 158 (2016), pp. 369–386. DOI: 10.1016/j.compstruct.2016.09.044 (cited on page 8).

[52] Z. Wang, D. Peeters, and R. De Breuker. "An aeroelastic optimisation framework for manufacturable variable stiffness composite wings including critical gust loads". In: *Structural and Multidisciplinary Optimization* 65.10 (Sept. 2022), pp. 1–20. DOI: 10.1007/s00158-022-03375-x (cited on page 8).

[53] W. Su and C. E. S. Cesnik. "Nonlinear Aeroelasticity of a Very Flexible Blended-Wing-Body Aircraft". In: *Journal of Aircraft* 47.5 (2010), pp. 1539–1553. DOI: 10.2514/1.47317 (cited on page 8).

[54] E. Jonsson, C. Riso, B. B. Monteiro, A. C. Gray, J. R. R. A. Martins, and C. E. S. Cesnik. "High-Fidelity Gradient-Based Wing Structural Optimization Including a Geometrically Nonlinear Flutter Constraint". In: *AIAA SciTech Forum*. Jan. 2022. DOI: 10.2514/6.2022-2092 (cited on page 8).

[55] P. D. Ciampa and B. Nagel. "AGILE Paradigm: The next generation collaborative MDO for the development of aeronautical systems". In: *Progress in Aerospace Sciences* 119 (Nov. 2020), p. 100643. DOI: 10.1016/j.paerosci.2020.100643 (cited on page 8).

[56] B. Boden, J. Flink, N. Först, R. Mischke, K. Schaffert, A. Weinert, A. Wohlan, and A. Schreiber. "RCE: An Integration Environment for Engineering and Science". In: *SoftwareX* 15 (July 2021), p. 100759. DOI: 10.1016/j.softx.2021.100759 (cited on page 8).

[57] B. Boden, J. Flink, R. Mischke, K. Schaffert, A. Weinert, A. Wohlan, C. Ilic, T. Wunderlich, C. M. Liersch, S. Goertz, P. D. Ciampa, and E. Moerland. "Distributed Multidisciplinary Optimization and Collaborative Process Development Using RCE". In: *AIAA Aviation 2019 Forum*. American Institute of Aeronautics and Astronautics, June 2019, p. 2989. DOI: 10.2514/6.2019-2989 (cited on page 8).

[58] J. H. Bussemaker, P. D. Ciampa, J. Singh, M. Fioriti, C. Cabaleiro De La Hoz, Z. Wang, D. Peeters, P. Hansmann, P. Della Vecchia, and M. Mandorino. "Collaborative Design of a Business Jet Family Using the AGILE 4.0 MBSE Environment". In: *AIAA AVIATION 2022 Forum*. American Institute of Aeronautics and Astronautics, June 2022, p. 3934. DOI: 10.2514/6.2022-3934 (cited on page 8).

[59] M. Mandorino, P. D. Vecchia, F. Nicolosi, S. Corcione, V. Trifari, G. Cerino, M. Fioriti, C. Cabaleiro, and T. Lefebvre. "Regional jet retrofitting design from stakeholders need and system requirements to MDAO workflow formulation". In: *33rd Congress of the International Council of the Aeronautical Sciences*. Sept. 5, 2022. DOI: 10.5281/zenodo.7071016 (cited on page 8).

[60] S. He, E. Jonsson, C. A. Mader, and J. R. R. A. Martins. "Aerodynamic Shape Optimization with Time Spectral Flutter Adjoint". In: *2019 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. San Diego, CA: American Institute of Aeronautics and Astronautics, Jan. 2019. DOI: 10.2514/6.2019-0697 (cited on page 9).

[61] K. P. Apponsah and D. W. Zingg. "Aerodynamic shape optimization for unsteady flows: Some benchmark problems". In: *Proceedings of the AIAA Scitech 2020 Forum*. 2020. DOI: 10.2514/6.2020-0541 (cited on page 9).

[62] J. Guo, Y. Li, M. Xu, X. An, and G. Li. "Aero-structural optimization of supersonic wing under thermal environment using adjoint-based optimization algorithm". In: *Structural and Multidisciplinary Optimization* 64.1 (Mar. 2021), pp. 281–301. DOI: 10.1007/s00158-021-02888-1 (cited on page 9).

[63] L. J. Halim, S. Sahu, G. Kennedy, and M. J. Smith. "Aerothermoelastic Analysis and Optimization of Stiffened Thin-Walled Structures". In: *AIAA SCITECH 2022 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2022, p. 1612. DOI: 10.2514/6.2022-1612 (cited on page 9).

[64] K. Telidetzki, L. Osusky, and D. W. Zingg. "Application of Jetstream to a Suite of Aerodynamic Shape Optimization Problems". In: *52nd Aerospace Sciences Meeting*. Feb. 2014. DOI: 10.2514/6.2014-0571 (cited on page 10).

[65] C. Lee, D. Koo, K. Telidetzki, H. Buckley, H. Gagnon, and D. W. Zingg. "Aerodynamic Shape Optimization of Benchmark Problems Using Jetstream". In: *Proceedings of the 53rd AIAA*

Aerospace Sciences Meeting. American Institute of Aeronautics and Astronautics (AIAA), Jan. 2015. DOI: 10.2514/6.2015-0262 (cited on page 10).

[66] G. K. W. Kenway and J. R. R. A. Martins. "Multipoint Aerodynamic Shape Optimization Investigations of the Common Research Model Wing". In: *AIAA Journal* 54.1 (Jan. 2016), pp. 113–128. DOI: 10.2514/1.J054154 (cited on page 10).

[67] D. A. Masters, D. J. Poole, N. J. Taylor, T. C. S. Rendall, and C. B. Allen. "Influence of Shape Parameterization on a Benchmark Aerodynamic Optimization Problem". In: *Journal of Aircraft* 54.6 (Nov. 2017), pp. 2242–2256. DOI: 10.2514/1.c034006 (cited on pages 10, 33).

[68] T. A. Reist, D. Koo, D. W. Zingg, P. Bochud, P. Castonguay, and D. Leblond. "Cross Validation of Aerodynamic Shape Optimization Methodologies for Aircraft Wing-Body Optimization". In: *AIAA Journal* (Feb. 2020). DOI: 10.2514/1.J059091 (cited on pages 10, 19).

[69] D. A. Masters, N. J. Taylor, T. C. S. Rendall, C. B. Allen, and D. J. Poole. "Geometric Comparison of Aerofoil Shape Parameterization Methods". In: *AIAA Journal* 55.5 (May 2017), pp. 1575–1589. DOI: 10.2514/1.j054943 (cited on pages 10, 38).

[70] N. Wu, C. Mader, and J. R. R. A. Martins. "Sensitivity-based Geometric Parameterization for Aerodynamic Shape Optimization". In: *AIAA AVIATION 2022 Forum*. June 2022. DOI: 10.2514/6.2022-3931 (cited on page 11).

[71] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, 1981 (cited on pages 12, 60).

[72] N. P. Bons. "High-fidelity Wing Design Exploration with Gradient-based Optimization". PhD thesis. Ann Arbor, MI: University of Michigan, May 2020 (cited on pages 12, 169).

[73] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis. *CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences*. Tech. rep. CR–2014-218178. NASA, Mar. 2014 (cited on page 13).

[74] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2nd ed. Berlin: Springer, 2006 (cited on pages 16, 17, 36, 77).

[75] D. Jones, C. Perttunen, and B. Stuckman. "Lipschitzian optimization without the Lipschitz constant". In: *Journal of Optimization Theory and Application* 79.1 (Oct. 1993), pp. 157–181. DOI: 10.1007/BF00941892 (cited on page 19).

[76] C. Aranha, C. L. Camacho Villalón, F. Campelo, M. Dorigo, R. Ruiz, M. Sevaux, K. Sörensen, and T. Stützle. "Metaphor-based metaheuristics, a call for action: The elephant in the room". In: *Swarm Intelligence* 16.1 (Nov. 2021), pp. 1–6. DOI: 10.1007/s11721-021-00202-9 (cited on page 19).

[77] R. T. Haftka. "Requirements for papers focusing on new or improved global optimization algorithms". In: *Structural and Multidisciplinary Optimization* 54.1 (2016), pp. 1–1. DOI: 10.1007/s00158-016-1491-5 (cited on page 19).

[78] O. Chernukhin and D. W. Zingg. "Multimodality and Global Optimization in Aerodynamic Design". In: *AIAA Journal* 51.6 (June 2013), pp. 1342–1354. DOI: 10.2514/1.j051835 (cited on page 19).

[79] Z. Lyu, G. K. W. Kenway, and J. R. R. A. Martins. "Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark". In: *AIAA Journal* 53.4 (Apr. 2015), pp. 968–985. DOI: 10.2514/1.J053318 (cited on pages 19, 33, 39, 43, 44, 100).

[80] S. Skinner and H. Zare–Behtash. "State-of-the-art in aerodynamic shape optimisation methods". In: *Applied Soft Computing* 62 (Jan. 2018), pp. 933–962. DOI: 10.1016/j.asoc.2017.09.030 (cited on page 19).

[81] N. P. Bons, X. He, C. A. Mader, and J. R. R. A. Martins. "Multimodality in Aerodynamic Wing Design Optimization". In: *AIAA Journal* 57.3 (Mar. 2019), pp. 1004–1018. DOI: 10.2514/1.J057294 (cited on page 19).

[82] D. J. Poole, C. B. Allen, and T. C. S. Rendall. "Global Optimization of Wing Aerodynamic Optimization Case Exhibiting Multimodality". In: *Journal of Aircraft* 55.4 (July 2018), pp. 1576–1591. DOI: 10.2514/1.c034718 (cited on page 19).

[83] A. Jameson, J. C. Vassberg, and K. Ou. "Further Studies of Airfoils Supporting Non-Unique Solutions in Transonic Flow". In: *AIAA Journal* 50.12 (Dec. 2012), pp. 2865–2881. DOI: 10.2514/1.j051713 (cited on page 19).

[84] D. Destarac, G. Carrier, G. R. Anderson, S. Nadarajah, D. J. Poole, J. C. Vassberg, and D. W. Zingg. "Example of a Pitfall in Aerodynamic Shape Optimization". In: *AIAA Journal* 56.4 (2018), pp. 1532–1540. DOI: 10.2514/1.J056128 (cited on page 19).

[85] G. M. Streuber and D. W. Zingg. "Evaluating the Risk of Local Optima in Aerodynamic Shape Optimization". In: *AIAA Journal* 59.1 (Jan. 2021), pp. 75–87. DOI: 10.2514/1.j059826 (cited on page 19).

[86] N. P. Bons and J. R. R. A. Martins. "Aerostructural Design Exploration of a Wing in Transonic Flow". In: *Aerospace* 7.8 (Aug. 2020), p. 118. DOI: 10.3390/aerospace7080118 (cited on pages 20, 39, 100).

[87] 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". In: *Structural and Multidisciplinary Optimization* 46.2 (Aug. 2012), pp. 273–284. DOI: 10.1007/s00158-012-0763-y (cited on pages 21, 102).

[88] J. Hicken and J. Alonso. "Comparison of Reduced- and Full-space Algorithms for PDE-constrained Optimization". In: *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, Jan. 2013, p. 1043. DOI: 10.2514/6.2013-1043 (cited on page 21).

[89] D. A. Knoll and D. E. Keyes. "Jacobian-free Newton–Krylov methods: a survey of approaches and applications". In: *Journal of Computational Physics* 193.2 (2004), pp. 357–397. DOI: 10.1016/j.jcp.2003.08.010 (cited on page 22).

[90] C. A. Mader, J. R. R. A. Martins, J. J. Alonso, and E. van der Weide. "ADjoint: An Approach for the Rapid Development of Discrete Adjoint Solvers". In: *AIAA Journal* 46.4 (Apr. 2008), pp. 863–873. DOI: 10.2514/1.29123 (cited on page 22).

[91] T. A. Albring, M. Sagebaum, and N. R. Gauger. "Efficient aerodynamic design using the discrete adjoint method in SU2". In: *17th AIAA/ISSMO multidisciplinary analysis and optimization conference*. 2016, p. 3518. DOI: 10.2514/6.2016-3518 (cited on page 22).

[92] J. E. V. Peter and R. P. Dwight. "Numerical Sensitivity Analysis for Aerodynamic Optimization: A Survey of Approaches". In: *Computers and Fluids* 39.3 (Mar. 2010), pp. 373–391. DOI: 10.1016/j.compfluid.2009.09.013 (cited on page 22).

[93] P. Gomes and R. Palacios. "Pitfalls of Discrete Adjoint Fixed-Points Based on Algorithmic Differentiation". In: *AIAA Journal* 60.2 (Feb. 2022), pp. 1251–1256. DOI: 10.2514/1.j060735 (cited on page 22).

[94] K. J. Fidkowski and D. L. Darmofal. "Review of output-based error estimation and mesh adaptation in computational fluid dynamics". In: *AIAA Journal* 49.4 (2011), pp. 673–694. DOI: 10.2514/1.J050073 (cited on pages 23, 76, 81).

[95] M. Giselle Fernández-Godino, C. Park, N. H. Kim, and R. T. Haftka. "Issues in Deciding Whether to Use Multifidelity Surrogates". In: *AIAA Journal* 57.5 (May 2019), pp. 2039–2054. DOI: 10.2514/1.j057750 (cited on pages 27, 96).

[96] B. Peherstorfer, K. Willcox, and M. Gunzburger. "Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization". In: *SIAM Review* 60.3 (Jan. 2018), pp. 550–591. DOI: 10.1137/16M1082469 (cited on pages 27, 95).

[97] N. R. Secco and J. R. R. A. Martins. "RANS-based Aerodynamic Shape Optimization of a Strut-braced Wing with Overset Meshes". In: *Journal of Aircraft* 56.1 (Jan. 2019), pp. 217–227. DOI: 10.2514/1.C034934 (cited on page 28).

[98] M. Mangano and J. R. R. A. Martins. "Multipoint Aerodynamic Shape Optimization for Subsonic and Supersonic Regimes". In: *Journal of Aircraft* 58.3 (May 2021), pp. 650–662. DOI: 10.2514/1.C036216 (cited on page 28).

[99] S. Chauhan and J. R. R. A. Martins. "RANS-Based Aerodynamic Shape Optimization of a Wing Considering Propeller-Wing Interaction". In: *Journal of Aircraft* 58.3 (May 2021), pp. 497–513. DOI: 10.2514/1.C035991 (cited on page 28).

[100] S. Seraj and J. R. R. A. Martins. "Predicting the High-Angle-of-Attack Characteristics of a Delta Wing at Low Speed". In: *Journal of Aircraft* 59.4 (July 2022), pp. 1071–1081. DOI: 10.2514/1.C036618 (cited on page 28).

[101] T. Dhert, T. Ashuri, and J. R. R. A. Martins. "Aerodynamic Shape Optimization of Wind Turbine Blades Using a Reynolds-Averaged Navier–Stokes Model and an Adjoint Method". In: *Wind Energy* 20.5 (May 2017), pp. 909–926. DOI: 10.1002/we.2070 (cited on page 28).

[102] M. H. A. Madsen, F. Zahle, N. N. Sørensen, and J. R. R. A. Martins. "Multipoint high-fidelity CFD-based aerodynamic shape optimization of a 10 MW wind turbine". In: *Wind Energy Sciences* 4 (Apr. 2019), pp. 163–192. DOI: 10.5194/wes-4-163-2019 (cited on page 28).

[103] N. P. Bons, J. R. R. A. Martins, C. A. Mader, M. McMullen, and M. Suen. "High-fidelity Aerostructural Optimization Studies of the Aerion AS2 Supersonic Business Jet". In: *Proceedings of the AIAA Aviation Forum*. June 2020. DOI: 10.2514/6.2020-3182 (cited on page 28).

[104] B. J. Brelje and J. R. R. A. Martins. "Aerostructural Wing Optimization for a Hydrogen Fuel Cell Aircraft". In: *Proceedings of the AIAA SciTech Forum*. Jan. 2021. DOI: 10.2514/6.2021-1132 (cited on pages 28, 39).

[105] N. Bons, J. R. R. A. Martins, F. Odaguil, and A. P. C. Cuco. "Aerostructural wing optimization of a regional jet considering mission fuel burn". In: *ASME Open Journal of Engineering* 1 (Oct. 2022), p. 011046. DOI: 10.1115/1.4055630 (cited on page 28).

[106] M. Mangano, S. He, Y. Liao, D.-G. Caprace, and J. R. R. A. Martins. "Towards Passive Aeroelastic Tailoring of Large Wind Turbines Using High-Fidelity Multidisciplinary Design Optimization". In: *AIAA SciTech Forum*. Jan. 2022. DOI: 10.2514/6.2022-1289 (cited on page 29).

[107] Y. Liao, J. R. R. A. Martins, and Y. L. Young. "3-D High-Fidelity Hydrostructural Optimization of Cavitation-Free Composite Lifting Surfaces". In: *Composite Structures* 268 (July 2021), p. 113937. DOI: 10.1016/j.compstruct.2021.113937 (cited on page 29).

[108] Y. Liao, A. Yildirim, J. R. R. A. Martins, and Y. L. Young. "RANS-based Optimization of a T-shaped Hydrofoil Considering Junction Design". In: *Ocean Engineering* 262 (Oct. 2022), p. 112051. DOI: 10.1016/j.oceaneng.2022.112051 (cited on page 29).

[109] J. S. Gray, C. A. Mader, G. K. W. Kenway, and J. R. R. A. Martins. "Modeling Boundary Layer Ingestion Using a Coupled Aeropropulsive Analysis". In: *Journal of Aircraft* 55.3 (May 2018), pp. 1191–1199. DOI: 10.2514/1.C034601 (cited on page 29).

[110] J. S. Gray and J. R. R. A. Martins. "Coupled Aeropropulsive Design Optimization of a Boundary-Layer Ingestion Propulsor". In: *The Aeronautical Journal* 123.1259 (Jan. 2019), pp. 121–137. DOI: 10.1017/aer.2018.120 (cited on page 29).

[111] G. K. Kenway, G. J. Kennedy, and J. R. R. A. Martins. "A CAD-Free Approach to High-Fidelity Aerostructural Optimization". In: *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*. AIAA 2010-9231. Fort Worth, TX, Sept. 2010. DOI: 10.2514/6.2010-9231 (cited on pages 29, 30, 38).

[112] M. H. A. Madsen. "High-Fidelity CFD-based Shape Optimization of Wind Turbine Blades". PhD thesis. Technical University of Denmark, 2020 (cited on pages 29, 33).

[113] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders. "Aerodynamic Shape Optimization of Complex Aircraft Configurations via an Adjoint Formulation". In: *Proceedings of the 34th AIAA Aerospace Sciences Meeting and Exhibit*. AIAA 1996-0094. Reno, Nevada, Jan. 1996 (cited on page 30).

[114] R. P. Dwight. "Robust Mesh Deformation using the Linear Elasticity Equations". In: *Computational Fluid Dynamics 2006*. Springer Berlin Heidelberg, 2009, pp. 401–406. DOI: 10.1007/978-3-540-92779-2_62 (cited on page 30).

[115] R. Biedron and J. Thomas. "Recent Enhancements to the FUN3D Flow Solver for Moving-Mesh Applications". In: *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, Jan. 2009, p. 1360. DOI: 10.2514/6.2009-1360 (cited on page 30).

[116] N. Secco, G. K. W. Kenway, P. He, C. A. Mader, and J. R. R. A. Martins. "Efficient Mesh Generation and Deformation for Aerodynamic Shape Optimization". In: *AIAA Journal* 59.4 (Apr. 2021), pp. 1151–1168. DOI: 10.2514/1.J059491 (cited on page 30).

[117] E. Luke, E. Collins, and E. Blades. "A Fast Mesh Deformation Method Using Explicit Interpolation". In: *Journal of Computational Physics* 231.2 (Jan. 2012), pp. 586–601. DOI: 10.1016/j.jcp.2011.09.021 (cited on page 30).

[118] C. A. Mader, G. K. W. Kenway, A. Yildirim, and J. R. R. A. Martins. "ADflow: An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization". In: *Journal of Aerospace Information Systems* 17.9 (Sept. 2020), pp. 508–527. DOI: 10.2514/1.I010796 (cited on page 30).

[119] A. Yildirim, G. K. W. Kenway, C. A. Mader, and J. R. R. A. Martins. "A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations". In: *Journal of Computational Physics* 397 (Nov. 2019), p. 108741. DOI: 10.1016/j.jcp.2019.06.018 (cited on pages 30, 78, 83).

[120] G. J. Kennedy and J. R. R. A. Martins. "A Parallel Finite-Element Framework for Large-Scale Gradient-Based Design Optimization of High-Performance Structures". In: *Finite Elements in Analysis and Design* 87 (Sept. 2014), pp. 56–73. DOI: 10.1016/j.finel.2014.04.011 (cited on page 30).

[121] S. Brown. "Displacement extrapolations for CFD+CSM aeroelastic analysis". In: *38th Structures, Structural Dynamics, and Materials Conference*. Structures, Structural Dynamics, and Materials and Co-located Conferences. American Institute of Aeronautics and Astronautics, Apr. 1997. DOI: 10.2514/6.1997-1090 (cited on pages 31, 124).

[122] B. M. Irons and R. C. Tuck. "A version of the Aitken accelerator for computer iteration". In: *International Journal for Numerical Methods in Engineering* 1.3 (1969), pp. 275–277. DOI: 10.1002/nme.1620010306 (cited on page 31).

[123] N. Wu, G. Kenway, C. A. Mader, J. Jasa, and J. R. R. A. Martins. "pyOptSparse: A Python framework for large-scale constrained nonlinear optimization of sparse systems". In: *Journal*

of Open Source Software 5.54 (Oct. 2020), p. 2564. DOI: 10.21105/joss.02564 (cited on page 31).

[124] P. Jansen and R. Perez. "Constrained structural design optimization via a parallel augmented Lagrangian particle swarm optimization approach". In: *Computers & Structures* 89.13-14 (2011), pp. 1352–1366. DOI: 10.1016/j.compstruc.2011.03.011 (cited on page 32).

[125] G. N. Vanderplaats. *CONMIN: A Fortran program for constrained function minimization: User's manual*. Tech. rep. 1973 (cited on page 32).

[126] A. Wächter and L. T. Biegler. "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming". In: *Mathematical Programming* 106.1 (2006), pp. 25–57 (cited on pages 32, 77).

[127] Y.-H. Dai and K. Schittkowski. "A sequential quadratic programming algorithm with non-monotone line search". In: *Pacific Journal of Optimization* 4 (2008), pp. 335–358 (cited on page 32).

[128] K. Schittkowski. "A robust implementation of a sequential quadratic programming algorithm with successive error restoration". In: *Optimization Letters* 5.2 (June 2010), pp. 283–296. DOI: 10.1007/s11590-010-0207-9 (cited on page 32).

[129] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017 (cited on page 32).

[130] T. W. Chin, M. L. Leader, and G. J. Kennedy. "A Scalable Framework for Large-Scale 3D Multimaterial Topology Optimization with Octree-based Mesh Adaptation". In: *Advances in Engineering Software* 135 (Sept. 1, 2019). DOI: 10.1016/j.advengsoft.2019.05.004 (cited on page 32).

[131] D. Kraft. *A software package for sequential quadratic programming*. DFVLR-FB 88-28. Koln, Germany: DLR German Aerospace Center–Institute for Flight Mechanics, 1988 (cited on pages 32, 86).

[132] P. E. Gill, W. Murray, and M. A. Saunders. "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization". In: *SIAM Review* 47.1 (2005), pp. 99–131. DOI: 10.1137/S0036144504446096 (cited on pages 32, 33, 42, 55, 64).

[133] D. Wolpert and W. Macready. "No free lunch theorems for optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1 (Apr. 1997), pp. 67–82. DOI: 10.1109/4235.585893 (cited on page 32).

[134] S. P. Adam, S.-A. N. Alexandropoulos, P. M. Pardalos, and M. N. Vrahatis. "No free lunch theorem: A review". In: *Approximation and optimization* (2019), pp. 57–82 (cited on page 32).

[135] V. Beiranvand, W. Hare, and Y. Lucet. "Best practices for comparing optimization algorithms". In: *Optimization and Engineering* (Sept. 2017). DOI: 10.1007/s11081-017-9366-1 (cited on page 32).

[136] H. D. Mittelmann. *AMPL-NLP Benchmark*. URL: http://plato.asu.edu/ftp/ampl-nlp.html (visited on 11/01/2022) (cited on page 32).

[137] P. E. Gill, M. A. Saunders, and E. Wong. "On the performance of SQP methods for nonlinear optimization". In: *Modeling and Optimization: Theory and Applications*. Ed. by B. Defourny and T. Terlaky. Vol. 147. New York, NY: Springer, 2015, pp. 95–123. DOI: 10.1007/978-3-319-23699-5_5 (cited on page 32).

[138] N. I. M. Gould, D. Orban, and P. L. Toint. *CUTEr (and SifDec), a Constrained and Unconstrained Testing Environment, revisited*. *CERFACS Technical Report* TR/PA/01/04. 2004 (cited on page 32).

[139] N. I. M. Gould, D. Orban, and P. L. Toint. "CUTEst: A Constrained and Unconstrained Testing Environment with safe threads for mathematical optimization". In: *Computational Optimization and Applications* 60.3 (Aug. 2014), pp. 545–557. DOI: 10.1007/s10589-014-9687-3 (cited on page 32).

[140] D. W. Zingg, M. Nemec, and T. H. Pulliam. "A Comparative Evaluation of Genetic and Gradient-Based Algorithms Applied to Aerodynamic Optimization". In: *European Journal of Computational Mechanics* 17.1–2 (Jan. 2008), pp. 103–126. DOI: 10.3166/remn.17.103-126 (cited on page 33).

[141] Z. Lyu, Z. Xu, and J. R. R. A. Martins. "Benchmarking Optimization Algorithms for Wing Aerodynamic Design Optimization". In: *Proceedings of the 8th International Conference on Computational Fluid Dynamics*. ICCFD8-2014-0203. Chengdu, Sichuan, China, July 2014 (cited on page 33).

[142] Y. Yu, Z. Lyu, Z. Xu, and J. R. R. A. Martins. "On the Influence of Optimization Algorithm and Starting Design on Wing Aerodynamic Shape Optimization". In: *Aerospace Science and Technology* 75 (Apr. 2018), pp. 183–199. DOI: 10.1016/j.ast.2018.01.016 (cited on pages 33, 41).

[143] A. Wendorff, E. Botero, and J. J. Alonso. "Comparing Different Off-the-Shelf Optimizers' Performance in Conceptual Aircraft Design". In: *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, June 2016, p. 3362. DOI: 10.2514/6.2016-3362 (cited on page 33).

[144] N. F. Baker, A. P. Stanley, J. J. Thomas, A. Ning, and K. Dykes. "Best Practices for Wake Model and Optimization Algorithm Selection in Wind Farm Layout Optimization". In: *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2019, p. 0540. DOI: 10.2514/6.2019-0540 (cited on page 33).

[145] J. E. Hicken and D. W. Zingg. "Aerodynamic Optimization Algorithm with Integrated Geometry Parameterization and Mesh Movement". In: *AIAA Journal* 48.2 (Feb. 2010), pp. 400–413. DOI: 10.2514/1.44033 (cited on pages 33, 38).

[146] J. E. Hoogervorst and A. Elham. "Wing aerostructural optimization using the Individual Discipline Feasible Architecture". In: *Aerospace Science and Technology* 65 (June 2017), pp. 90–99. DOI: 10.1016/j.ast.2017.02.012 (cited on page 33).

[147] F. Bisson and Nadarajah. "Adjoint-based aerodynamic optimization of benchmark problems". In: *53nd Aerospace Sciences Meeting*. 2015. DOI: 10.2514/6.2015-1948 (cited on page 33).

[148] G. R. Anderson, M. Nemec, and M. J. Aftosmis. "Aerodynamic shape optimization benchmarks with error control and automatic parameterization". In: *53rd AIAA Aerospace Sciences Meeting*. 2015, p. 1719. DOI: 10.2514/6.2015-1719 (cited on pages 33, 39).

[149] P. E. Gill, W. Murray, and M. A. Saunders. *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*. Technical Report. Systems Optimization Laboratory. Stanford University, California, 94305-4023, 2007 (cited on pages 34, 36).

[150] P. Castonguay and S. K. Nadarajah. "Effect of Shape Parameterization on Aerodynamic Shape Optimization". In: *45th AIAA Aerospace Sciences Meeting and Exhibit*. Reno, Nevada, Jan. 2007. DOI: 10.2514/6.2007-59 (cited on page 38).

[151] T. W. Sederberg and S. R. Parry. "Free-form Deformation of Solid Geometric Models". In: *SIGGRAPH Comput. Graph.* 20.4 (Aug. 1986), pp. 151–160. DOI: 10.1145/15886.15903 (cited on page 38).

[152] T.-t. Zhang, Z.-g. Wang, W. Huang, and L. Yan. "A review of parametric approaches specific to aerodynamic design process". In: *Acta Astronautica* 145 (Apr. 2018), pp. 319–331. DOI: 10.1016/j.actaastro.2018.02.011 (cited on page 38).

[153] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso. "SU2: An Open-Source Suite for Multiphysics Simulation and Design". In: *AIAA Journal* 54.3 (Mar. 2016), pp. 828–846. DOI: 10.2514/1.j053813 (cited on page 38).

[154] H. Gagnon and D. W. Zingg. "Two-Level Free-Form and Axial Deformation for Exploratory Aerodynamic Shape Optimization". In: *AIAA Journal* 53.7 (2015), pp. 2015–2026. DOI: 10.2514/1.J053575 (cited on page 38).

[155] R. Haimes and J. Dannenhoffer. "The Engineering Sketch Pad: A Solid-Modeling, Feature-Based, Web-Enabled System for Building Parametric Geometry". In: *21st AIAA Computational Fluid Dynamics Conference*. Fluid Dynamics and Co-located Conferences. American Institute of Aeronautics and Astronautics, June 2013. DOI: 10.2514/6.2013-3073 (cited on page 39).

[156] 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*. American Institute of Aeronautics and Astronautics, Jan. 2022. DOI: 10.2514/6.2022-0004 (cited on page 39).

[157] D. Koo and D. W. Zingg. "Investigation into Aerodynamic Shape Optimization of Planar and Nonplanar Wings". In: *AIAA Journal* 56 (Aug. 2018), pp. 250–263. DOI: 10.2514/1.j055978 (cited on page 39).

[158] G. M. Streuber and D. W. Zingg. "Dynamic Geometry Control for Robust Aerodynamic Shape Optimization". In: *AIAA AVIATION 2021 Forum*. July 2021. DOI: 10.2514/6.2021-3031 (cited on page 39).

[159] L. J. Kedward, C. B. Allen, and T. C. S. Rendall. "Gradient-Limiting Shape Control for Efficient Aerodynamic Optimization". In: *AIAA Journal* 58.9 (Sept. 2020), pp. 3748–3764. DOI: 10.2514/1.j058977 (cited on page 39).

[160] D. A. Masters, N. J. Taylor, T. C. S. Rendall, and C. B. Allen. "Multilevel Subdivision Parameterization Scheme for Aerodynamic Shape Optimization". In: *AIAA Journal* 55.10 (Oct. 2017), pp. 3288–3303. DOI: 10.2514/1.j055785 (cited on page 39).

[161] X. Han and D. W. Zingg. "An adaptive geometry parametrization for aerodynamic shape optimization". In: *Optimization and Engineering* 15.1 (2014), pp. 69–91 (cited on page 39).

[162] G. R. Anderson and M. J. Aftosmis. "Adaptive Shape Control for Aerodynamic Design". In: *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2015, p. 0398. DOI: 10.2514/6.2015-0398 (cited on page 39).

[163] S. Ghoman, Z. Wang, P. Chen, and R. Kapania. "A POD-based Reduced Order Design Scheme for Shape Optimization of Air Vehicles". In: *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*. American Institute of Aeronautics and Astronautics, Apr. 2012. DOI: 10.2514/6.2012-1808 (cited on page 40).

[164] J. Li, M. A. Bouhlel, and J. R. R. A. Martins. "Data-based Approach for Fast Airfoil Analysis and Optimization". In: *AIAA Journal* 57.2 (Feb. 2019), pp. 581–596. DOI: 10.2514/1.J057129 (cited on page 40).

[165] J. Li and M. Zhang. "Adjoint-Free Aerodynamic Shape Optimization of the Common Research Model Wing". In: *AIAA Journal* 59.6 (June 2021), pp. 1990–2000. DOI: 10.2514/1.j059921 (cited on page 40).

[166] D. J. Poole, C. B. Allen, and T. Rendall. "Efficient aeroelastic wing optimization through a compact aerofoil decomposition approach". In: *Structural and Multidisciplinary Optimization* 65.3 (Feb. 2022), pp. 1–19. DOI: 10.1007/s00158-022-03174-4 (cited on page 40).

[167] M. S. Selig. *UIUC airfoil data site*. Department of Aeronautical and Astronautical Engineering University of Illinois at Urbana-Champaign, 1996 (cited on page 40).

[168] L. J. Kedward, C. B. Allen, D. J. Poole, and T. C. S. Rendall. "Generic Modal Design Variables for Efficient Aerodynamic Optimization". In: *AIAA Journal* (Nov. 2022), pp. 1–17. DOI: 10.2514/1.j061727 (cited on page 41).

[169] T. W. Lukaczyk, P. Constantine, F. Palacios, and J. J. Alonso. "Active Subspaces for Shape Optimization". In: *10th AIAA Multidisciplinary Design Optimization Conference*. American Institute of Aeronautics and Astronautics, Jan. 2014. DOI: 10.2514/6.2014-1171 (cited on page 41).

[170] Z. J. Grey and P. G. Constantine. "Active Subspaces of Airfoil Shape Parameterizations". In: *AIAA Journal* 56.5 (May 2018), pp. 2003–2017. DOI: 10.2514/1.j056054 (cited on page 41).

[171] C. Lee, D. Koo, and D. W. Zingg. "Comparison of B-Spline Surface and Free-Form Deformation Geometry Control for Aerodynamic Optimization". In: *AIAA Journal* 55.1 (2017), pp. 228–240. DOI: 10.2514/1.J055102 (cited on page 43).

[172] S. C. Eisenstat and H. F. Walker. "Choosing the Forcing Terms in an Inexact Newton Method". In: *SIAM Journal on Scientific Computing* 17.1 (Jan. 1996), pp. 16–32. DOI: 10.1137/0917003 (cited on page 75).

[173] B. Peherstorfer, P. S. Beran, and K. E. Willcox. "Multifidelity Monte Carlo estimation for large-scale uncertainty propagation". In: *2018 AIAA Non-Deterministic Approaches Conference*. American Institute of Aeronautics and Astronautics, Jan. 2018. DOI: 10.2514/6.2018-1660 (cited on page 75).

[174] M. Heinkenschloss and L. N. Vicente. "Analysis of inexact trust-region SQP algorithms, SIAM Journal on Optimization". In: *SIAM Journal on Optimization* 12 (2001), pp. 283–302 (cited on page 75).

[175] C. Gu, D. Zhu, and Y. Pei. "A new inexact SQP algorithm for nonlinear systems of mixed equalities and inequalities". In: *Numerical Algorithms* 78.4 (Sept. 2017), pp. 1233–1253. DOI: 10.1007/s11075-017-0421-y (cited on page 75).

[176] R. T. Biedron, J.-R. Carlson, J. M. Derlaga, P. A. Gnoffo, D. P. Hammond, W. T. Jones, B. Kleb, E. M. Lee-Rausch, E. J. Nielsen, M. A. Park, et al. *FUN3D Manual: 13.6*. Tech. rep. 2019 (cited on page 76).

[177] M. Nemec, M. Aftosmis, and M. Wintzer. "Adjoint-Based Adaptive Mesh Refinement for Complex Geometries". In: *46th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, Jan. 2008, p. 725. DOI: 10.2514/6.2008-725 (cited on page 76).

[178] J. Lu and D. Darmofal. "Adaptive Precision Methodology for Flow Optimization via Discretization and Iteration Error Control". In: *42nd AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, Jan. 2004, p. 1096. DOI: 10.2514/6.2004-1096 (cited on pages 76, 77, 84, 86).

[179] C. Lozano and I. Ruiz Juretschke. "Adjoint-Based Correction of Non-Converged CFD Solutions". In: *19th AIAA Computational Fluid Dynamics*. American Institute of Aeronautics and Astronautics, June 2009. DOI: 10.2514/6.2009-4144 (cited on page 76).

[180] D. A. Brown and S. Nadarajah. "Inexactly constrained discrete adjoint approach for steepest descent-based optimization algorithms". In: *Numerical Algorithms* 78.3 (Sept. 2017), pp. 983–1000. DOI: 10.1007/s11075-017-0409-7 (cited on pages 76, 77).

[181] D. A. Brown and S. Nadarajah. "Effect of inexact adjoint solutions on the discrete-adjoint approach to gradient-based optimization". In: *Optimization and Engineering* 23.3 (Sept. 2021), pp. 1643–1676. DOI: 10.1007/s11081-021-09681-5 (cited on pages 76, 77, 87).

[182] R. Becker and R. Rannacher. "An optimal control approach to *a posteriori* error estimation in finite element methods". In: *Acta Numerica* 10 (May 2001), pp. 1–102. DOI: 10.1017/s0962492901000010 (cited on page 81).

[183] F. A. C. Viana, T. W. Simpson, V. Balabanov, and V. Toropov. "Metamodeling in Multidisciplinary Design Optimization: How Far Have We Really Come?" In: *AIAA Journal* 52.4 (Apr. 2014), pp. 670–690. DOI: 10.2514/1.J052375 (cited on page 96).

[184] L. Leifsson and S. Koziel. "Multi-fidelity design optimization of transonic airfoils using physics-based surrogate modeling and shape-preserving response prediction". In: *Journal of Computational Science* 1.2 (2010), pp. 98–106. DOI: 10.1016/j.jocs.2010.03.007 (cited on page 96).

[185] S. Choi, J. J. Alonso, I. M. Kroo, and M. Wintzer. "Multifidelity Design Optimization of Low-Boom Supersonic Jets". In: *Journal of Aircraft* 45.1 (2008), pp. 106–118 (cited on page 96).

[186] A. I. Forrester, N. W. Bressloff, and A. J. Keane. "Optimization using surrogate models and partially converged computational fluid dynamics simulations". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 462 (2006), pp. 2177–2204. DOI: 10.1098/rspa.2006.1679 (cited on page 96).

[187] N.-V. Nguyen, S.-M. Choi, W.-S. Kim, J.-W. Lee, S. Kim, D. Neufeld, and Y.-H. Byun. "Multidisciplinary unmanned combat air vehicle system design using multi-fidelity model". In: *Aerospace Science and Technology* 26.1 (2013), pp. 200–210. DOI: 10.1016/j.ast.2012.04.004 (cited on page 96).

[188] N. M. Alexandrov, R. M. Lewis, C. R. Gumbert, L. L. Green, and P. A. Newman. "Approximation and Model Management in Aerodynamic Optimization with Variable-Fidelity Models". In: *Journal of Aircraft* 38.6 (2001), pp. 1093–1101 (cited on page 97).

[189] A. Elham and M. J. van Tooren. "Multi-fidelity wing aerostructural optimization using a trust region filter-SQP algorithm". In: *Structural and Multidisciplinary Optimization* 55 (2017), pp. 1773–1786. DOI: 10.1007/s00158-016-1613-0 (cited on pages 97, 98).

[190] A. March and K. Willcox. "Provably convergent multifidelity optimization algorithm not requiring high-fidelity derivatives". In: *AIAA journal* 50.5 (2012), pp. 1079–1089. DOI: 10.2514/1.J051125 (cited on page 97).

[191] S. Gratton, A. Sartenaer, and P. L. Toint. "Recursive trust-region methods for multiscale nonlinear optimization". In: *SIAM Journal on Optimization* 19.1 (2008), pp. 414–444. DOI: 10.1137/050623012 (cited on page 97).

[192] R. Olivanti, F. Gallard, J. Brézillon, and N. Gourdain. "Comparison of Generic Multi-Fidelity Approaches for Bound-Constrained Nonlinear Optimization Applied to Adjoint-Based CFD Applications". In: *AIAA Aviation 2019 Forum*. 2019. DOI: 10.2514/6.2019-3102 (cited on pages 97, 98).

[193] D. E. Bryson and M. P. Rumpfkeil. "Multifidelity Quasi-Newton Method for Design Optimization". In: *AIAA Journal* 56.10 (2018), pp. 4074–4086. DOI: 10.2514/1.J056840 (cited on page 97).

[194] D. E. Bryson and M. P. Rumpfkeil. "Aerostructural Design Optimization Using a Multifidelity Quasi-Newton Method". In: *Journal of Aircraft* 56.5 (2019), pp. 2019–2031. DOI: 10.2514/1.C035152 (cited on page 98).

[195] A. S. Thelen, D. E. Bryson, B. K. Stanford, and P. S. Beran. "Multi-Fidelity Gradient-Based Optimization for High-Dimensional Aeroelastic Configurations". In: *Algorithms* 15.4 (Apr. 2022), p. 131. DOI: 10.3390/a15040131 (cited on page 98).

[196] D. Allaire and K. Willcox. "A mathematical and computational framework for multifidelity design and analysis with computer models". In: *International Journal for Uncertainty Quantification* 4.1 (2014), pp. 1–20. DOI: 10.1615/Int.J.UncertaintyQuantification.2013004121 (cited on pages 98, 99, 105).

[197] A. H. Bayoumy and M. Kokkolaras. "A relative adequacy framework for multimodel management in multidisciplinary design optimization". In: *Structural and Multidisciplinary Optimization* 62.4 (July 2020), pp. 1701–1720. DOI: 10.1007/s00158-020-02591-7 (cited on page 99).

[198] S. Koziel and L. Leifsson. "Multi-level CFD-based airfoil shape optimization with automated low-fidelity model selection". In: *Procedia Computer Science* 18 (2013), pp. 889–898. DOI: 10.1016/j.procs.2013.05.254 (cited on page 100).

[199] M. Drela. "XFOIL — An analysis and design system for low Reynolds number airfoils". In: *Low Reynolds number aerodynamics*. Notre Dame,Germany, Federal Republic of, May 1989 (cited on page 101).

[200] J. Morgado, R. Vizinho, M. Silvestre, and J. Páscoa. "XFOIL vs CFD performance predictions for high lift low Reynolds number airfoils". In: *Aerospace Science and Technology* 52 (2016), pp. 207–214. DOI: 10.1016/j.ast.2016.02.031 (cited on page 101).

[201] D. Levy, K. Laflin, J. Vassberg, E. Tinoco, M. Mani, B. Rider, O. Brodersen, S. Crippa, C. Rumsey, R. Wahls, J. Morrison, D. Mavriplis, and M. Murayama. "Summary of data from the fifth AIAA CFD drag prediction workshop". In: *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. 2013. DOI: 10.2514/6.2013-46 (cited on page 106).

[202] G. Chen and K. J. Fidkowski. "Discretization Error Control for Constrained Aerodynamic Shape Optimization". In: *Journal of Computational Physics* 387 (2019), pp. 163–185. DOI: 10.1016/j.jcp.2019.02.038 (cited on page 110).

[203] A. B. Yoo, M. A. Jette, and M. Grondona. "Slurm: Simple linux utility for resource management". In: *Workshop on job scheduling strategies for parallel processing*. Springer. 2003, pp. 44–60 (cited on page 115).

[204] S. F. Ghoreishi and D. L. Allaire. "Adaptive Uncertainty Propagation for Coupled Multidisciplinary Systems". In: *AIAA Journal* 55.11 (Nov. 2017), pp. 3940–3950. DOI: 10.2514/1.j055893 (cited on page 120).

[205] S. Friedman, B. Isaac, S. F. Ghoreishi, and D. L. Allaire. "Efficient Decoupling of Multiphysics Systems for Uncertainty Propagation". In: *2018 AIAA Non-Deterministic Approaches Conference*. American Institute of Aeronautics and Astronautics, Jan. 2018, p. 1661. DOI: 10.2514/6.2018-1661 (cited on page 120).

[206] A. Chaudhuri, R. Lam, and K. Willcox. "Multifidelity Uncertainty Propagation via Adaptive Surrogates in Coupled Multidisciplinary Systems". In: *AIAA Journal* 56.1 (Jan. 2018), pp. 235–249. DOI: 10.2514/1.j055678 (cited on page 121).

[207] X. Du and W. Chen. "Efficient Uncertainty Analysis Methods for Multidisciplinary Robust Design". In: *AIAA Journal* 40.3 (Mar. 2002), pp. 545–552. DOI: 10.2514/2.1681 (cited on page 121).

[208] R. Baptista, Y. Marzouk, K. Willcox, and B. Peherstorfer. "Optimal Approximations of Coupling in Multidisciplinary Models". In: *AIAA Journal* 56.6 (June 2018), pp. 2412–2428. DOI: 10.2514/1.j056888 (cited on page 122).

[209] G. Kreisselmeier and R. Steinhauser. "Systematic Control Design by Optimizing a Vector Performance Index". In: *International Federation of Active Controls Symposium on Computer-Aided Design of Control Systems, Zurich, Switzerland*. 1979. DOI: 10.1016/S1474-6670(17)65584-8 (cited on pages 135, 153).

[210] T. R. Brooks, G. J. Kennedy, and J. R. R. A. Martins. "High-fidelity Multipoint Aerostructural Optimization of a High Aspect Ratio Tow-steered Composite Wing". In: *Proceedings of the 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA SciTech Forum*. Grapevine, TX, Jan. 2017. DOI: 10.2514/6.2017-1350 (cited on page 140).

[211] J. Pattinson and M. Herring. "High Fidelity Simulation of Wing Loads with an Active Winglet Control Surface". In: *IFASD 2013, International Forum on Aeroelasticity and Structrual Dynamics, 24-26 June 2013, Bristol*. Bristol, UK: Royal Aeronautical Society, June 2013 (cited on page 150).

[212] M. Cavcar. "Bréguet Range Equation?" In: *Journal of Aircraft* 43.5 (2006), pp. 1542–1544. DOI: 10.2514/1.17696 (cited on page 150).

[213] G. J. Kennedy, G. K. W. Kenway, and J. R. R. A. Martins. "High Aspect Ratio Wing Design: Optimal Aerostructural Tradeoffs for the Next Generation of Materials". In: *Proceedings of the*

AIAA Science and Technology Forum and Exposition (SciTech). National Harbor, MD, Jan. 2014. DOI: 10.2514/6.2014-0596 (cited on page 152).

[214] W. Yao, X. Chen, W. Luo, M. van Tooren, and J. Guo. "Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles". In: *Progress in Aerospace Sciences* 47.6 (Aug. 2011), pp. 450–479. DOI: 10.1016/j.paerosci.2011.05.001 (cited on pages 176, 177).

[215] *Evaluation of measurement data - Supplement 1 to the "Guide to the expression of uncertainty in measurement" – Propagation of distributions using a Monte Carlo method*. JCGM 101:2008. Sèvres, France: Joint Committee for Guides in Metrology, 2008 (cited on page 176).

[216] F. A. Seiler. "Error Propagation for Large Errors". In: *Risk Analysis* 7.4 (Dec. 1987), pp. 509–518. DOI: 10.1111/j.1539-6924.1987.tb00487.x (cited on page 177).

[217] M. A. F. Martins, R. Requião, and R. A. Kalid. "Generalized expressions of second and third order for the evaluation of standard measurement uncertainty". In: *Measurement* 44.9 (Nov. 2011), pp. 1526–1530. DOI: 10.1016/j.measurement.2011.06.008 (cited on page 177).

[218] J. E. Hannay, C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl, and G. Wilson. "How do scientists develop and use scientific software?" In: *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*. Ieee. IEEE, May 2009, pp. 1–8. DOI: 10.1109/secse.2009.5069155 (cited on page 179).

[219] A. Johanson and W. Hasselbring. "Software Engineering for Computational Science: Past, Present, Future". In: *Computing in Science & Engineering* 20.2 (2018), pp. 1–1. DOI: 10.1109/mcse.2018.108162940 (cited on page 179).

[220] G. Wilson, D. A. Aruliah, C. T. Brown, N. P. C. Hong, M. Davis, R. T. Guy, S. H. D. Haddock, K. D. Huff, I. M. Mitchell, M. D. Plumbley, B. Waugh, E. P. White, and P. Wilson. "Best Practices for Scientific Computing". In: *PLoS Biology* 12.1 (2014), e1001745. DOI: 10.1371/journal.pbio.1001745 (cited on page 179).

[221] G. Wilson, J. Bryan, K. Cranston, J. Kitzes, L. Nederbragt, and T. K. Teal. "Good enough practices in scientific computing". In: *PLOS Computational Biology* 13.6 (June 2017), e1005510. DOI: 10.1371/journal.pcbi.1005510 (cited on page 179).

[222] M. C. Galbraith, S. Allmaras, and D. L. Darmofal. "A Verification Driven Process for Rapid Development of CFD Software". In: *53rd AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, Jan. 2015, p. 0818. DOI: 10.2514/6.2015-0818 (cited on page 179).

[223] R. P. Kendall, N. S. Hariharan, R. L. Meakin, K. Bergeron, and D. A. Post. "The HPCMP CREATE™ Program management Model-Part I". In: *AIAA Scitech 2021 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2021, p. 0232. DOI: 10.2514/6.2021-0232 (cited on page 179).

[224] K. Bergeron, R. P. Kendall, N. S. Hariharan, R. L. Meakin, and D. A. Post. "The HPCMP CREATE™ Management Model – Part II, DevOps Principles and Practices in HPCMP CREATE™". In: *AIAA Scitech 2021 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2021, p. 0233. DOI: 10.2514/6.2021-0233 (cited on page 179).

[225] X. Zhang, W. T. Jones, S. L. Wood, and M. A. Park. "Component-Based Development of CFD Software FUN3D". In: *AIAA SCITECH 2022 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2022, p. 0253. DOI: 10.2514/6.2022-0253 (cited on page 179).

[226] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles. "A Survey of DevOps Concepts and Challenges". In: *ACM Comput. Surv.* 52.6 (Nov. 2019). DOI: 10.1145/3359981 (cited on page 180).

[227] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer. "What is DevOps? A Systematic Mapping Study on Definitions and Practices". In: *Proceedings of the Scientific Workshop Proceedings of XP2016*. XP '16 Workshops. New York, NY, USA: Association for Computing Machinery, 2016. DOI: 10.1145/2962695.2962707 (cited on page 180).

[228] W. L. Oberkampf and C. J. Roy. *Verification and Validation in Scientific Computing*. Cambridge University Press, Oct. 2010 (cited on pages 180, 181).

[229] B. Kleb and B. Wood. "Computational Simulations and the Scientific Method". In: *Journal of Aerospace Computing, Information, and Communication* 3.6 (June 2006), pp. 244–250. DOI: 10.2514/1.12949 (cited on page 181).

[230] Z. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, and M. Visbal. "High-order CFD methods: current status and perspective". In: *International Journal for Numerical Methods in Fluids* 72.8 (2013), pp. 811–845. DOI: 10.1002/fld.3767 (cited on page 182).

[231] A. Nanthaamornphong and J. C. Carver. "Test-Driven Development in scientific software: A survey". In: *Software Quality Journal* 25.2 (Sept. 2015), pp. 343–372. DOI: 10.1007/s11219-015-9292-4 (cited on page 182).

[232] G. M. Kurtzer, V. Sochat, and M. W. Bauer. "Singularity: Scientific containers for mobility of compute". In: *PLOS ONE* 12.5 (May 2017), pp. 1–20. DOI: 10.1371/journal.pone.0177459 (cited on pages 184, 189).

[233] M. Kennedy and A. O'Hagan. "Bayesian Calibration of Computer Models". In: *J.R. Statist. Soc. B* 63.3 (2001), pp. 425–464 (cited on page 186).

[234] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 1997 (cited on page 186).

[235] M. J. Corden and D. Kreitzer. "Consistency of floating-point results using the intel compiler or why doesn't my application always give the same answer". In: *Intel Corp., Software Solutions Group, Santa Clara, CA, USA, Tech. Rep* (2009) (cited on page 186).

[236] P. Balaji and D. Kimpe. "On the Reproducibility of MPI Reduction Operations". In: *2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing*. IEEE. IEEE, Nov. 2013, pp. 407–414. DOI: 10.1109/hpcc.and.euc.2013.65 (cited on page 187).

[237] C. Collange, D. Defour, S. Graillat, and R. Iakymchuk. "Numerical reproducibility for the parallel reduction on multi- and many-core architectures". In: *Parallel Computing* 49 (Nov. 2015), pp. 83–97. DOI: 10.1016/j.parco.2015.09.001 (cited on page 187).

[238] G. Karypis and V. Kumar. "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs". In: *SIAM Journal on Scientific Computing* 20.1 (Jan. 1998), pp. 359–392. DOI: 10.1137/s1064827595287997 (cited on page 187).

[239] D. Ezer and K. Whitaker. "Data science for the scientific life cycle". In: *eLife* 8 (Mar. 2019). DOI: 10.7554/elife.43979 (cited on page 188).

[240] M. R. Munafò, B. A. Nosek, D. V. M. Bishop, K. S. Button, C. D. Chambers, N. Percie du Sert, U. Simonsohn, E.-J. Wagenmakers, J. J. Ware, and J. P. A. Ioannidis. "A manifesto for reproducible science". In: *Nature Human Behaviour* 1.1 (Jan. 2017), pp. 1–9. DOI: 10.1038/s41562-016-0021 (cited on page 188).

[241] P. Ivie and D. Thain. "Reproducibility in Scientific Computing". In: *ACM Computing Surveys* 51.3 (July 2018), pp. 1–36. DOI: 10.1145/3186266 (cited on page 188).

[242] D. Merkel et al. "Docker: Lightweight linux containers for consistent development and deployment". In: *Linux j* 239.2 (2014), p. 2 (cited on page 189).

[243] R. Kuprieiev, skshetry, D. Petrov, P. Redzyński, P. Rowlands, C. da Costa-Luis, A. Schepanovski, I. Shcheklein, Gao, B. Taskaya, D. de la Iglesia Castro, J. Orpinel, F. Santos, R. Lamy, A. Sharma, D. Berenbaum, daniele, Zhanibek, D. Hodovic, N. Kodenko, A. Grigorev, Earl, N. Dash, G. Vyshnya, maykulkarni, M. Hora, Vera, and S. Mangal. *DVC: Data Version Control - Git for Data & Models*. Version 2.38.1. Dec. 2022. DOI: 10.5281/zenodo.7440136 (cited on page 189).

[244] R. Rampin, F. Chirigati, D. Shasha, J. Freire, and V. Steeves. "ReproZip: The Reproducibility Packer". In: *The Journal of Open Source Software* 1.8 (Dec. 2016), p. 107. DOI: 10.21105/joss.00107 (cited on page 189).

[245] Y. Halchenko, K. Meyer, B. Poldrack, D. Solanky, A. Wagner, J. Gors, D. MacFarlane, D. Pustina, V. Sochat, S. Ghosh, C. Mönch, C. Markiewicz, L. Waite, I. Shlyakhter, A. de la Vega, S. Hayashi, C. Häusler, J.-B. Poline, T. Kadelka, K. Skytén, D. Jarecka, D. Kennedy, T. Strauss, M. Cieslak, P. Vavra, H.-I. Ioanas, R. Schneider, M. Pflüger, J. Haxby, S. Eickhoff, and M. Hanke. "DataLad: Distributed system for joint management of code, data, and their relationship". In: *Journal of Open Source Software* 6.63 (July 2021), p. 3262. DOI: 10.21105/joss.03262 (cited on page 189).

[246] C. S. Adorf, P. M. Dodd, V. Ramasubramani, and S. C. Glotzer. "Simple data and workflow management with the signac framework". In: *Computational Materials Science* 146 (Apr. 2018), pp. 220–229. DOI: 10.1016/j.commatsci.2018.01.035 (cited on page 189).