A modular approach to large-scale design optimization of aerospace systems

by

John T. Hwang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in The University of Michigan
2015

ACKNOWLEDGMENTS

There are countless people to whom I have become indebted during my PhD, but I would like to start with my advisor, Prof. Joaquim R. R. A. Martins. I consider myself fortunate to have had Prof. Martins as my PhD advisor. I am thankful for his guidance and encouragement, especially in my frequent pursuit of side projects and ideas that were often tangent to the main research topic. I have also been inspired by Prof. Martins' work ethic and attention to detail, particularly when it comes to making effective presentations and figures.

I would like to thank the other members of my thesis committee—Prof. Krzysztof J. Fidkowski, Prof. Anthony M. Waas, and Prof. Yin Lu Young—for volunteering their time and facilitating the doctoral degree process for me on a tight schedule. Their feedback and perspective were invaluable and improved my dissertation in several areas.

Next, I would like to acknowledge many collaborators who had a large impact on this disseration: Dae Young Lee and Prof. James W. Cutler from the Michigan Exploration Laboratory (small satellite design); Satadru Roy and Prof. William A. Crossley from Purdue University (aircraft allocation problem); and Robert Haimes from Massachusetts Institute of Technology (computational geometry). I would also like to thank Justin S. Gray, Kenneth T. Moore, Bret A. Naylor, and Tristan A. Hearn from the NASA Glenn Research Center for many insightful discussions and collaborations over the years. I am particularly grateful for their patience and willingness to integrate my often poorly documented code into their software.

I want to thank Dr. Gaetan K. W. Kenway and Prof. Graeme J. Kennedy, who are recent PhD graduates of our research group. In many ways, their outstanding dissertations provided the foundation and motivation for my own dissertation. Special thanks to several students in my research group with whom I have had the pleasure to work: GaRam Jun, Song Chen, Jason Y. Kao, and Davide Ivaldi. They helped implement various aircraft-related applications that appear in my dissertation.

Finally, thank you to all my groupmates and friends over the years who have made my four-plus years as a doctoral student truly enjoyable. Last but certainly not least, I will always be grateful for the love and support from my family, and the sacrifices that my parents made to make this possible.

John Hwang
December 2014
Ann Arbor, MI

TABLE OF CONTENTS

ACKNOWLEDGMENTS ii

LIST OF FIGURES viii

LIST OF TABLES xii

LIST OF ABBREVIATIONS xiii

LIST OF SYMBOLS xv

ABSTRACT xviii

CHAPTER

1. Introduction 1

1.1 The role of large-scale optimization in design 1
1.2 The argument for gradient-based optimization 3
1.3 The limitations of gradient-based optimization 5
1.3.1 Discrete, non-smooth, and multi-modal problems . . 6
1.3.2 The computation of derivatives 6
1.4 Thesis objective and contributions 7
1.4.1 Multidisciplinary problems 7
1.4.2 Aircraft shape design 8

Part I A general computational modeling framework

2. The need for a new computational framework 10

2.1 Modularity in computational modeling 11
2.2 Existing frameworks 11
2.2.1 Existing commercial frameworks 12
2.2.2 Existing HPC frameworks 12
2.2.3 Existing MDO frameworks 13
2.3 The current framework 13

3. A monolithic formulation 16

3.1 Notation for a general numerical model 16
3.2 A monolithic formulation of the numerical model 17
3.3 Hierarchical decomposition 18
3.4 Validity of the monolithic, hierarchical formulation 19
3.5 Computation of derivatives 21

4. A unification of methods for computing derivatives 25

4.1 Monolithic differentiation 26
4.2 Algorithmic differentiation 27
4.3 Analytic methods 29
4.4 Coupled analytic methods 29

5. A compact framework design 34

5.1 Problem statement 34
5.2 Solver-driven requirements 35
5.3 A compact interface 37
5.4 Object-oriented framework design 38
5.5 Efficient data management 40
5.5.1 Data storage 41
5.5.2 Data accessing 43
5.5.3 Data transfer 43
5.6 Implementation 44
5.7 Summary 44

6. Multidisciplinary optimization of a nanosatellite 48

6.1 Optimization in nanosatellite design 48
6.2 The design problem 50
6.3 Approach 52
6.3.1 Multiple time scales: multi-point optimization . . . 53
6.3.2 Large number of constraints: constraint aggregation 54
6.3.3 Nondifferentiable models: B-spline interpolant . . . 55
6.3.4 Derivatives of ODE variables: modular Runge–Kutta solver 55
6.4 Discipline models 55
6.4.1 Orbit dynamics 56
6.4.2 Attitude dynamics 56
6.4.3 Cell illumination 58
6.4.4 Temperature 60
6.4.5 Solar power 61
6.4.6 Energy storage 61

6.4.7 Communication 63
6.5 Optimization 66
6.5.1 MDO architecture 66
6.5.2 Optimization problems 67
6.5.3 Impact of optimization 70
6.5.4 Comparison between launches 72
6.6 Summary 75

7. Allocation-mission optimization of commercial aircraft 77

7.1 A motivation from aircraft design optimization 77
7.2 Prior art in allocation optimization 79
7.3 Prior art in mission optimization 79
7.4 Methodology 80
7.4.1 The mission problem 80
7.4.2 The allocation problem 84
7.4.3 The allocation-mission problem 86
7.5 Allocation-mission optimization results 88
7.5.1 The problem 88
7.5.2 Profit increase with new aircraft 90
7.5.3 Allocation versus allocation-mission optimization . . 92
7.5.4 Local optima 93
7.5.5 Numerical performance 94
7.6 Summary 96

8. Conclusions 97

8.1 Summary and contributions 97
8.2 Significance 99
8.2.1 The unification of derivative computation methods . 99
8.2.2 The computational modeling framework 100
8.3 Recommendations 100

Part II A differentiable parametrization of aircraft geometries and structures

9. The need for a new parametrization 102

9.1 Unconventional aircraft configurations 103
9.2 High-fidelity aerostructural optimization 104
9.3 The requirements for a new parametrization 105
9.4 The solution: GeoMACH 107

10. A flexible geometry parametrization 109

10.1 Geometry representation 109
10.2 Geometry parametrization 111
10.3 Computation of Derivatives 112

11. An automated structural mesh generation algorithm 115

11.1 Linear mapping 115
11.2 Two-step process 116
11.3 Unstructured quadrilateral mesh generation 117
11.4 Mesh convergence study 122

12. Aerostructural analysis and aerodynamic shape optimization results 125

12.1 The MACH tool suite 125
12.1.1 Mesh warping: pyWarp 125
12.1.2 Flow solver: SUmb 126
12.1.3 Structural solver: TACS 126
12.1.4 Aerostructural solver: pyAeroStruct 126
12.2 Aerostructural analysis 126
12.3 Aerodynamic shape optimization 128

13. Conclusions 132

13.1 Summary and contributions 132
13.2 Significance 134
13.3 Recommendations for future work 135

BIBLIOGRAPHY 136

LIST OF FIGURES

<u>**Figure**</u>

1.1 The function evaluations used by NSGA2, a gradient-free optimizer (left), and SNOPT, a gradient-based optimizer (right), during the solution of the Rosenbrock problem. For the gradient-based optimizer, the points that deviate from the general path to the optimum correspond to function evaluations during line searches. 4
1.2 Plot showing how the number of function evaluations required to optimize the multi-dimensional Rosenbrock function scales with the number of design variables. The gradient-free optimizers (ALPSO, NSGA2) scale quadratically or worse, while the gradient-based optimizers (SNOPT, SLSQP) scale linearly with finite-difference derivatives (FD), and better than linearly with analytic derivatives (AN). 5
2.1 The role of the proposed computational modeling framework compared to that of existing frameworks in the development of a computational model. The proposed framework uses a unified formulation of the numerical model to enable features not provided by existing frameworks which do not change or reformulate the numerical model. 14
3.1 The block structure of the matrices in the left equality of Eq. (3.24). 24
4.1 Equations for computing total derivatives in a general system of equations. 26
4.2 Derivation of monolithic (black box) differentiation. 27
4.3 Derivation of algorithmic differentiation. 28
4.4 Derivation of the analytic methods: direct and adjoint. 30
4.5 Derivation of the residual form of the coupled derivative methods. . 32
4.6 Derivation of the functional form of the coupled derivative methods. 33
5.1 A class diagram showing the containment relationships between objects in a computational model implemented in the framework. . . . 39
5.2 A class diagram showing the inheritance relationships between the *System* classes. 40
5.3 Operations for the nonlinear system. 41
5.4 Operations for the linear system. 42
5.5 Operations for the transposed linear system. 46
5.6 The type of data storage for each variable in each *System* instance. 47

5.7 Runtime results for the nanosatellite (left) and aircraft allocation-mission (right) optimization problems implemented in the framework. The blue curve represents a single execution of the computational model, and the red curve represents the solution of a linear system for the adjoint method. 47
6.1 CADRE CubeSat geometry. 51
6.2 Extended design structure matrix (XDSM) diagram [1] showing all relevant disciplines in the CADRE design problem. 52
6.3 Reaction-wheel model compared with manufacturer-provided data for three torques. 58
6.4 Normalized exposed area as a function of relative Sun position for the outermost cell in an inward-facing panel. 59
6.5 Illustration of the Sun line-of-sight variable. 60
6.6 Solar cell I-V curve at different cell temperatures and exposed areas. 62
6.7 Battery-discharge curve model compared with manufacturer's data at two temperatures. 63
6.8 Transmitter gain as a function of the ground-station position in the body-fixed frame. 64
6.9 Illustration of the ground-station line-of-sight variable. 64
6.10 Extended design structure matrix (XDSM) diagram [1] for the MDO problem. 67
6.11 Convergence histories and snapshots of data and SOC at intermediate optimization iterations. 69
6.12 Division of total data downloaded over the six simulations for the three optimization problems. 71
6.13 Initial and optimized profiles of quantities of interest for the three optimization problems (continued in Fig. 6.14). 73
6.14 Initial and optimized profiles of quantities of interest for the three optimization problems (continued from Fig. 6.13). 74
6.15 Division of total data downloaded over the six simulations for the three launches. 75
6.16 Design structure matrix diagram illustrating the complexity of the problem. Each block on the diagonal represents one of 43 components that each computes one or more quantities in the computational model. Off-diagonal entries represent dependencies—variables passed from one component to another. 76
7.1 A class diagram showing the *System* instances in the aircraft mission problem. 82
7.2 Plots of a subset of the variables modeled in the aircraft mission optimization problem. The altitude profile is designed by the optimizer to minimize total fuel burn over the mission. The distances are given in nautical miles. 84
7.3 Convergence plots for a representative aircraft mission optimization. 84

7.4 Map of the cities and the routes (generated using the Great Circle Mapper: www.gcmap.com). Newark, New Jersey is chosen as the hub. 90
7.5 Comparison of profit for the four scenarios at the solution to (MINLP-a-m). The results show an increase in profit when the hypothetical airline purchases the next-generation aircraft. 91
7.6 Comparison of optimized profits. The colors represent, in order: (MINLP-a-m) not initialized to the solution to (MILP-a) (orange), (MILP-a) (red), (MINLP-a-m) initialized to the solution to (MILP-a) (blue), and (NLP-a-m) solved in the first node of the branch-and-bound algorithm. The results show clear local minima and an increase in profit in the allocation-mission optimization compared to the allocation-only optimization. 92
7.7 Optimal values of the (round trip) flights per day design variable comparing (MINLP-a-m) with and without (local) initialization to the solution to (MILP-a). The results show two distinct local minima in (MINLP-a-m). 93
7.8 Optimal values of the (round trip) flights per day design variable comparing the allocation-only solution (MILP-a), the relaxed allocation-mission solution (NLP-a-m), and the integer allocation-mission solution (MINLP-a-m). The results show that the relaxed simultaneous solution is close to the allocation-only solution and the integer simultaneous solution is close to the relaxed simultaneous solution. 94
7.9 Optimization convergence history for a 3-route, 2-new aircraft problem with 330 design variables. The tight convergence of optimality and feasibility helps verify the accuracy of the derivative computation. 95
9.1 The fuel burn per passenger per unit distance of new aircraft over time, as a percentage of the value in 1960. The last 5 decades have seen a 50 % reduction, but progress has stalled in the last 2 decades. Source: the International Council on Clean Transportation [2]. . . . 103
9.2 Renderings for NASA's HWB (left), MIT's D8 (center), and Boeing's TBW (right) concepts. Source: NASA Aeronautics Research Mission Directorate. 104
10.1 UML diagram of the parametric geometry modeler. Diamonds represent containment and open triangles represent inheritance. 110
10.2 Six aircraft configurations created in GeoMACH. 110
10.3 List of properties that parametrize Primitive components. In the shape column, n signifies the number of span-wise sections for a wing or the number of stream-wise sections for the fuselage. 111
10.4 Illustration of the parametrization of a Junction component. Bezier curves are first generated from the intersecting components, and then the interior control points are populated using the formula for a Coons patch. 112

10.5 Sequence of operations in the parametrization: (1) Design variables (2) Parameters (3) Properties (4) Primitive control points (5) Primitive and wireframe control points (6) Full control points vector ordered by component and face (7) Unique control points ordered globally. 113
10.6 Illustration of data storage and accessing. Vectors A and B (e.g. global properties vector and the global parameters vector) are both accessed via reshaped NumPy views onto sub-vectors for each component. Each vector object contains 1-D data array (in blue) and a 1-D indices array (in red) of the same size—this facilitates assembly of the sparse Jacobian. 114
11.1 Preview mesh (left), detailed final FEA mesh (center), and derivative contours with respect to a root chord parameter (right). 116
11.2 Four configurations with detailed structures. 117
11.3 The six steps of the unstructured quad meshing algorithm. 118
11.4 The global (left) and local (right) coordinate frames for the elliptical smoothing. 119
11.5 Global finite element equations for the elliptical smoothing. The K matrix represents the global finite element matrix, and P is a permutation matrix with a single entry of value 1 in each row, in the columns of nodes that are constrained. The vectors x, y represent the solution and $\bar{x}, \bar{y}$ contain the coordinates to which a node is constrained. 122
11.6 The meshes produced by the unstructured quad meshing algorithm and contours of vertical displacement. 123
11.7 Results of the mesh convergence study. The deflection is measured at the upper, outboard-most tip of the front spar, and the loading is a 1 kN point load at the same node. 124
12.1 Aerostructural analysis results for the TBW (left) and D8 (right) configurations. 127
12.2 The TBW structure shown in detail. 127
12.3 Initial (left) and optimized (right) shape and CFD solution of the truss-braced wing configuration. The pressure contours represent a slice near the root of the wing. 129
12.4 RANS-based aerodynamic shape optimization of the TBW at M=0.73. 131

LIST OF TABLES

Table

5.1 The framework's 4 operations and their implementations in each type of *System* class. 40
6.1 Data for discipline models 65
6.2 The general optimization problem. 68
6.3 Optimal design variables for for the three optimization problems. . . 70
6.4 Optimal values for the three launches. 72
7.1 The 3 routes considered in the allocation-mission optimization . . . 89
7.2 The types of aircraft considered in the allocation-mission optimization. The bottom four lines show the number of each aircraft type available in each of the four scenarios. 91
7.3 The execution times for the solution of (MINLP-a-m). 95
12.1 The optimization problem. 128

LIST OF ABBREVIATIONS

AD	Algorithmic differentiation
API	Application programming interface
BLI	Boundary layer ingestion
BPR	Bypass ratio
BSE	B-spline engine
BWB	Blended wing body
CAD	Computer-aided design
CADRE	CubeSat investigating atmospheric density response to extreme driving
CDA	Continuous descent approach
CFD	Computational fluid dynamics
CRM	Common research model
D8	Double bubble
FAP	Fleet assignment problem
FEA	Finite element analysis
FFD	Free-form deformation
FSDT	First-order shear deformation theory
GA	Genetic algorithm
GeoMACH	Geometry-centric MDO of aircraft configurations with high fidelity
GMRES	Generalized minimal residual method
GUI	Graphical user interface
fGMRES	Flexible GMRES
HPC	High-performance computing
HWB	Hybrid wing body
JFNK	Jacobian-free Newton–Krylov
KS	Kreisselmeier–Steinhauser
LOS	Line of sight
MACH	MDO of aircraft configurations with high fidelity
MDA	Multidisciplinary analysis
MDO	Multidisciplinary design optimization
MDF	Multidisciplinary feasible
OpenMDAO	Open-source multidisciplinary design, analysis, and optimization framework
OPD	Optimized profile descent

OML	Outer mold line
PDE	Partial differential equation
PETSc	Portable, extensible toolkit for scientific computation
PGM	Parametric geometry modeler
PSM	Parametric structural modeler
PSO	Particle swarm optimizer
PTC	Pseudo-transient continuation
RANS	Reynolds-averaged Navier–Stokes
RK4	4th order Runge–Kutta
SAND	Simultaneous analysis and design
SFC	Thrust specific fuel consumption
SNOPT	Sparse nonlinear optimizer
SOC	State of charge
SQP	Sequential quadratic programming
SUmb	Stanford University multi-block solver
TACS	Toolkit for the analysis of composite structures
TBW	Truss-braced wing
XDSM	Extended design structure matrix

LIST OF SYMBOLS

Chapters 3-5

$(\cdot)^*$	Solution value
D	Set of possible values of u
D_k	Set of possible values of v_k
D_k^f	Set of possible values of the kth output variable
D_k^x	Set of possible values of the kth input variable
D_k^y	Set of possible values of the kth state variable
f	All output variables
f_k	kth output variable
$\mathcal{F}$	Function that computes all the output variables
$\mathcal{F}_k$	Function that computes the kth output variable
$\mathcal{G}$	Composition of $\mathcal{F}$ and $\mathcal{Y}$
$\mathcal{I}$	Identity matrix
m	Number of input variables
n	Number of variables
N_k^f	Size of the k output variable
N_k^x	Size of the k input variable
N_k^y	Size of the k state variable
p	Number of state variables; parameter vector
q	Number of output variables
r	Output of the function R
r_k	Output of the function R_k
R	Residual function for u
R_k	Residual function for v_k
$\mathcal{R}$	Residual function for all the state variables
$\mathcal{R}_k$	Residual function for the kth state variable
S	An index set, subset of $\{1, \dots, n\}$
t_k	Variable defined by a line of code
T_k	Function computed by a line of code
u	Unknown vector of the algebraic system
v_k	kth variable
x	All input variables
x_k	kth input variable
y	All state variables

y_k	kth state variable
$\mathcal{Y}$	Function that computes all the state variables
$\mathcal{Y}_k$	Function that computes the kth state variable
δ_{ij}	Kronecker delta

Chapter 6

$\vec{(\cdot)}$	Vector
$\hat{(\cdot)}$	Unit vector
A	Area, m^2
B_r	Bit rate, Gb/s
$\boldsymbol{c}$	Vector of all constraint variables
$\boldsymbol{C}$	Vector of all constraint functions
$\boldsymbol{C}_i$	Vector of constraint functions corresponding to the ith component
d_s	Distance from satellite to Earth–Sun axis, km
d_c	Distance from satellite to Earth tangent plane at ground station, km
D	Downloaded data, Gb
G_t	Transmitter gain
I	Current, A
$\boldsymbol{J}$	Mass moment of inertia matrix, kg·m^2
$\boldsymbol{L}$	Angular momentum vector, kg·m^2/s
LOS_s	Satellite-to-Sun line of sight
LOS_c	Satellite-to-ground-station line of sight
$\boldsymbol{O}$	Orientation matrix
P	Power, W
$\dot{Q}$	Heat transfer rate, W
r	Position vector norm, km
$\vec{\boldsymbol{r}}$	Position vector, km
SOC	Battery state of charge
T	Temperature, K
$\vec{\boldsymbol{v}}$	Velocity vector, m/s
$\boldsymbol{v}$	Vector of all variables (input, state, output)
$\boldsymbol{v}_i$	Vector of variables corresponding to the ith component
$\boldsymbol{V}_i$	Vector of functions corresponding to the ith component
γ	Satellite roll angle, rad
η_s	Sun line of sight transition parameter
τ	Torque, N·m
$\boldsymbol{\omega}$	Angular velocity vector, 1/s

Chapter 7

$\tilde{(\cdot)}$	Target value
C_D	Drag coefficient

C_L	Lift coefficient
C_M	Moment coefficient
C_T	Thrust coefficient
D	Drag
g	Acceleration due to gravity
h	Altitude
I	Mass moment of inertia
L	Lift
M	Moment
SFC	Thrust-specific fuel consumption
T	Thrust
W	Weight
W_f	Fuel weight
v	Airspeed
x	Horizontal distance along route
α	Angle of attack
γ	Climb or descent angle
ρ	Atmospheric density

Chapter 11

A	Area of integration in Galerkin's method
f	A shape function in the η or ξ direction
F, F'	Matrices used in computing the local element matrix
K	Local element matrix
s	Parametric variable for line integrals
u	Discrete value of ϕ at a node
x	One of the two axes in the global frame
y	One of the two axes in the global frame
η	One of the two parametric axes in the element frame
ξ	One of the two parametric axes in the element frame
ϕ	Scalar function solved in Laplace's equation

ABSTRACT

Gradient-based optimization and the adjoint method form a synergistic combination that enables the efficient solution of large-scale optimization problems. Though the gradient-based approach struggles with non-smooth or multi-modal problems, the capability to efficiently optimize up to tens of thousands of design variables provides a valuable design tool for exploring complex tradeoffs and finding unintuitive designs. However, the widespread adoption of gradient-based optimization is limited by the implementation challenges for computing derivatives efficiently and accurately, particularly in multidisciplinary and shape design problems. This thesis addresses these difficulties in two ways.

First, to deal with the heterogeneity and integration challenges of multidisciplinary problems, this thesis presents a computational modeling framework that solves multidisciplinary systems and computes their derivatives in a semi-automated fashion. This framework is built upon a new mathematical formulation developed in this thesis that expresses any computational model as a system of algebraic equations and unifies all methods for computing derivatives using a single equation. The framework is applied to two engineering problems: the optimization of a nanosatellite with 7 disciplines and over 25,000 design variables; and simultaneous allocation and mission optimization for commercial aircraft involving 330 design variables, 12 of which are integer variables handled using the branch-and-bound method. In both cases, the framework makes large-scale optimization possible by reducing the implementation effort and code complexity.

The second half of this thesis presents a differentiable parametrization of aircraft geometries and structures for high-fidelity shape optimization. Existing geometry parametrizations are not differentiable, or they are limited in the types of shape changes they allow. This is addressed by a novel parametrization that smoothly interpolates aircraft components, providing differentiability. An unstructured quadrilateral mesh generation algorithm is also developed to automate the creation of detailed meshes for aircraft structures, and a mesh convergence study is performed to verify that the quality of the mesh is maintained as it is refined. As a demonstration, high-fidelity aerostructural analysis is performed for two unconventional configurations with detailed structures included, and aerodynamic shape optimization is applied to the truss-braced wing, which finds and eliminates a shock in the region bounded by the struts and the wing.

CHAPTER 1

Introduction

The fundamental problem that motivates this thesis is the question, how can we better utilize computational models in the design of engineering systems? Large-scale optimization is one way through which computational models can make a larger impact in engineering design. The combination of a gradient-based optimizer and the adjoint method has been shown to form a powerful tool that makes it possible to perform optimization with up to tens of thousands of design variables. However, the main drawback is that the gradient-based approach is often difficult to implement. This thesis addresses this challenge in two particular types of situations: multidisciplinary and shape design problems.

This chapter provides the motivation for this thesis and a brief overview of the contributions. Section 1.1 begins by describing how engineering design can benefit from optimization tools, and in particular, large-scale optimization. Section 1.2 explains that gradient-based optimization is required to solve a large-scale optimization problem. Section 1.3 describes the limitations of gradient-based optimization, especially the challenges of computing derivatives efficiently. Finally, Sec. 1.4 summarizes the thesis objectives and contributions.

1.1 The role of large-scale optimization in design

Engineering design is an art. It is a process that relies on the decision-making of human designers based on their knowledge and intuition, supplemented by experimental observations. While the human aspect of the design process is not likely to change, numerical experimentation using computational models is becoming an increasingly relevant tool, as advances in hardware and software enable better accuracy. In many fields, such as aircraft aerodynamics, computational models already offer a practical and useful alternative to physical experiments for certain situations because they are typically less costly to run.

Within each field that uses computational models, improving their accuracy is an ongoing research topic. Since many systems and processes are naturally governed by partial differential equations (PDEs), there are three main sources of error to target: modeling error due to the assumptions inherent in the PDE, discretization error due to the finite size of the mesh, and iteration error due to partial convergence of the solver. These types of error can be reduced with improved mathematical models, larger meshes, and tighter convergence, respectively, which are all addressed by advancements in computing power and algorithms.

Though developing more accurate computational models is a moving target, a concurrent aim of research is to utilize existing computational models to design engineering systems. This is arguably best achieved using numerical optimization, since engineering design problems are naturally expressed in such a form. Numerical optimization problems seek to minimize or maximize an *objective function* by varying *design variables* with *equality constraints* or *inequality constraints* enforced.

In a practical design problem, there are at least three ways in which numerical optimization can make an impact on the design process. First, when the computational model is sufficiently accurate, the optimization solution may be used directly, with some modifications, as the final design for the design problem. An example is the use of topology optimization in the structural design of the Airbus A380's leading edge ribs in the main wing [3, 4]. The second way numerical optimization can be an invaluable tool is by discovering a new design feature or insights that deepen the designer's understanding of the problem. For instance, an aerodynamic shape optimization algorithm applied to an aircraft wing has been used to recover the winglet from an initially flat wing [5, 6], and an aerostructural optimization algorithm recovered a raked (rear-swept) wing tip [7], which is a design feature used on the Boeing 787. These results show potential for discovering other features that can provide efficiency improvements. A third way numerical optimization can be useful is by exposing weaknesses in the computational model or in the optimization problem formulation. For instance, optimizers often converge to unphysical designs that exploit the parts of the computational model with large overprediction of performance, highlighting where the model needs to be improved. The effective use of optimization within a design process relies on the intuition of the designer, especially to separate true design improvements from performance overprediction due to model weaknesses. However, optimization results often provide feedback to the engineer, suggesting constraints to add to account for a physical assumption or design variables to include to capture an important tradeoff. The result is an iterative and interactive process that can enrich the engineer's understanding of the design problem.

Some of the benefits of optimization may be achieved by running the computational model at sufficiently many input values, using intuition to choose the values in an intelligent way. This approach—akin to manual optimization—is ineffective if there are too many variables and too much complexity for intuition to be of use. In general, numerical optimization can provide the most value when there are a large number of design variables—i.e., *large-scale optimization*—because it can yield the most unintuitive results.

1.2 The argument for gradient-based optimization

Given a large-scale optimization problem, the challenge is to solve it to an acceptable level of optimality with as few evaluations of the computational model as possible. Here, three assumptions are made regarding the problem.

The first assumption is that the optimization problem is continuous—the design variables are permitted to take on any values in a convex set—and the objective and constraint functions are continuous on this set. Though discrete variables and functions often do arise, continuous problems are more common. Moreover, treating discrete design variables continuously often yields a useful solution: e.g., for integer design variables with a sufficiently large scale, the optimized values can be rounded to the nearest integer. There are also algorithms for handling discrete design variables in optimization problems such as the branch-and-bound method [8], which is used in Ch. 7, and these solve a series of continuous optimization problems.

The second assumption is that the computational model is expensive to evaluate. The most valuable computational models capture unintuitive behavior, such as that governed by complex PDEs that cannot be solved analytically. These must be discretized into a system of nonlinear equations, which is often large in size, with $\mathcal{O}(10^5 \sim 10^7)$ unknowns.

The third assumption is that the constraints represent physical considerations or system requirements that must be satisfied for any design to be considered. Thus, it is required that the optimizer finds feasible designs that satisfy all the constraints at a high level of precision.

Given these assumptions, gradient-based optimization algorithms are better suited to solve large-scale optimization problems than gradient-free optimization algorithms.

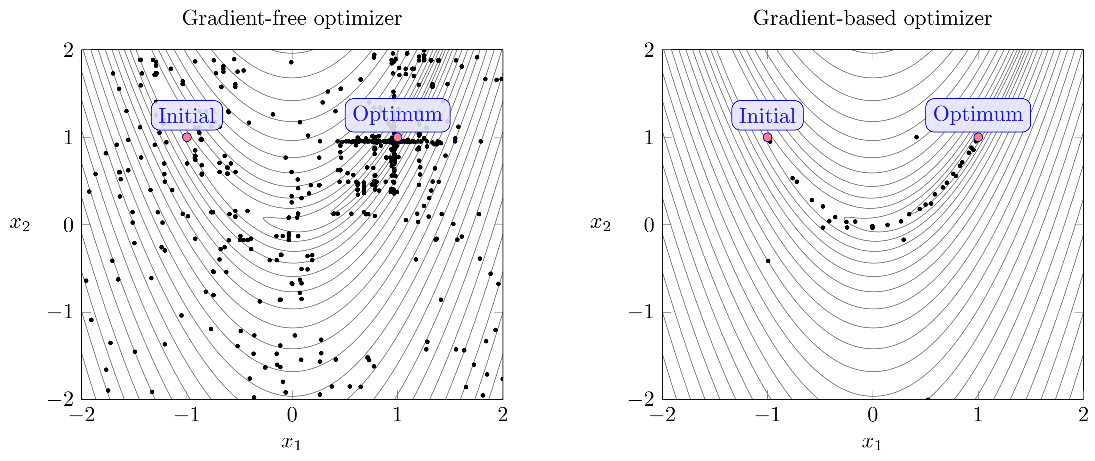


Figure 1.1: The function evaluations used by NSGA2, a gradient-free optimizer (left), and SNOPT, a gradient-based optimizer (right), during the solution of the Rosenbrock problem. For the gradient-based optimizer, the points that deviate from the general path to the optimum correspond to function evaluations during line searches.

Optimizers of the latter type do not perform well for problems with more than hundreds of design variables [9]. This is because gradient-free methods such as genetic algorithms (GAs) and particle swarm optimizers (PSOs) work with populations distributed over the design-variable range. This is illustrated in Fig. 1.1, which plots the function evaluations used by NSGA2 [10], a gradient-free optimizer, and SNOPT [11], a gradient-based optimizer, to find the minimum of the Rosenbrock function [12] with respect to two design variables. For gradient-free methods, Perez et al. [9] note that solution accuracy greatly suffers in large-scale problems due to the large number of evaluations required. Furthermore, for a constant level of convergence, the number of required function evaluations grows exponentially.

When combined with analytic derivative computation, gradient-based optimization can be a powerful tool, because the adjoint method makes it possible to compute derivatives at a cost that is nearly independent of the number of design variables [13]. In addition, the number of iterations required for the optimization typically scales only linearly with sequential quadratic programming (SQP). With SQP, an approximate Hessian of the Lagrangian is built from only first derivatives and the method is second-order convergent close to the optimum point. As shown in Fig. 1.2, the end result is that pairing gradient-based optimization with the adjoint method enables the solution of large-scale optimization problems with fewer than $\mathcal{O}(n)$ evaluations of the computational model, where n is the number of design variables.

In the aerospace field, the adjoint method has flourished in aerodynamic shape

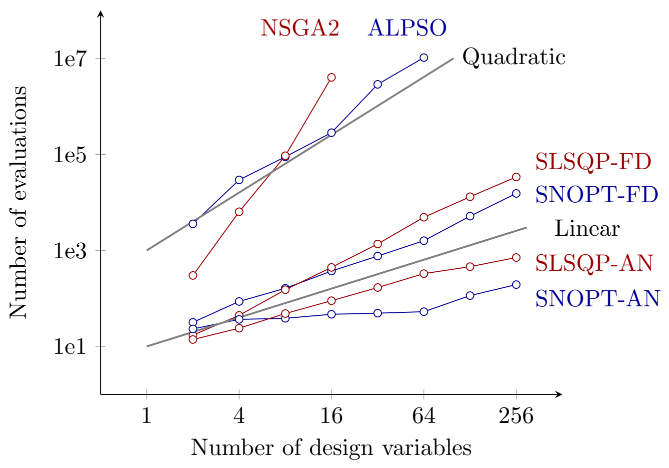


Figure 1.2: Plot showing how the number of function evaluations required to optimize the multi-dimensional Rosenbrock function scales with the number of design variables. The gradient-free optimizers (ALPSO, NSGA2) scale quadratically or worse, while the gradient-based optimizers (SNOPT, SLSQP) scale linearly with finite-difference derivatives (FD), and better than linearly with analytic derivatives (AN).

optimization, initially with the approach of Jameson [14] and later with the optimization of full aircraft configurations [15, 16, 17]. It has also been successfully applied to the multidisciplinary optimization of aircraft aerodynamics and structures simultaneously using a coupled adjoint approach [18, 19].

Gradient-based optimizers can never guarantee convergence to the global optimum. However, guaranteeing the global optimum is not a realistic goal for large-scale optimization problems. Even as a local optimizer, a gradient-based optimization algorithm is still useful because it is able to find feasible designs that can improve upon one selected using experience and human intuition, if this design is used as the initial point for optimization. The argument is that a local optimum for a problem that closely represents reality may be more useful than the global optimum of a problem based on lower-fidelity models.

1.3 The limitations of gradient-based optimization

In Sec. 1.1, it was argued that numerical optimization can maximize the value of computational models in an engineering design process, especially as the accuracy of the models improve. Large-scale optimization adds even more value because its results are likely to be more unintuitive, whether it yields a useful design or reveals insights about the problem. Section 1.2 showed that gradient-based optimization and the adjoint method form a powerful combination because they enable the solution of large-scale optimization problems much faster than evaluating the computational model once for each design variable. This section discusses the limitations of gradient-based optimization that hinder more widespread adoption.

1.3.1 Discrete, non-smooth, and multi-modal problems

The first limitation is that many optimization problems are not suited to a gradient-based method. For instance, the problem could contain discrete design variables. One potential solution is relaxation—treating them as continuous variables—with constraints imposed to force the optimization solution to be discrete. However, this approach cannot always be applied and when it can, it yields a problem in which every discrete solution candidate is a local minimum.

The optimization problem may be discontinuous, non-differentiable, or simply not continuously differentiable. Though not always the case, convergence issues can generally be expected if the objective and constraint functions have any of these properties. As with discrete variables, this problem can sometimes be addressed by artificially altering the computational model to ensure the optimization problem is continuously differentiable to eliminate the risk of convergence issues. However, this introduces additional error to the computational model.

Another potential issue is multi-modality of the combined objective and constraint functions. If this is the case, there are many local optima, so the result of gradient-based optimization would be sensitive to the selection of the initial design. Though gradient-free optimizers are more likely to find the global optimum, they do so at the expense of significantly more evaluations of the computational model, as shown in Fig. 1.2. If finding the global optimum is critical, a more efficient approach would be to run multiple gradient-based optimizations from different starting points that are uniformly sampled over the domain or randomly selected [20]. However, there is evidence that some practical problems may be convex [20], and in most cases, a local optimum is still a useful result that improves upon the initial design.

1.3.2 The computation of derivatives

The second major limitation of gradient-based optimization is that it is much more difficult to implement than gradient-free optimization, because it requires the computation of derivatives. The derivatives must be computed accurately and efficiently to enable large-scale optimization using gradient-based methods.

Accuracy is required for two reasons. First, it can reduce the required number of optimization iterations by helping the optimizer take a more direct path, which can be visualized in Fig. 1.1. This effect is significantly pronounced in large-scale optimization problems, and inaccurate derivatives represent arguably the most common cause of convergence failure, especially in large-scale problems. Second, accuracy is also required for tight convergence in terms of *optimality*, which is the norm of the gradients that are driven to zero, and *feasibility*, which represents the degree of constraint violation. In practice, inaccurate gradients cause the optimization to eventually stall as both optimality and feasibility plateau.

Gradient-free optimizers treat the computational model as a black box that is simply evaluated at various input values. The only methods for computing derivatives that also treat the computational model as a black box are the finite-difference methods, which are neither accurate nor efficient. In addition to requiring an evaluation of the computational model for each design variable, these methods suffer from subtractive cancellation error with small step sizes and truncation error with larger step sizes. The adjoint method provides both accuracy and efficiency, but it can require extensive modification of the original computational model.

1.4 Thesis objective and contributions

Given the motivation for large-scale optimization and the aforementioned obstacles, my objective in this thesis is to lower the entry barrier for gradient-based optimization. Specifically, I focus on two types of problems in which large-scale optimization problems commonly arise—multidisciplinary problems and shape design problems.

1.4.1 Multidisciplinary problems

Many engineering design problems are multidisciplinary. However, the computational model for each discipline is often developed separately, resulting in multiple pieces of software that must be joined together. Moreover, coupling between disciplines may be present, requiring a global solver to converge the multidisciplinary computational model. Computing the derivatives of the outputs of the multidisciplinary model poses an even greater challenge.

In Part I of this thesis, I address these challenges via a new computational modeling framework that facilitates the solution of multidisciplinary systems and implements adjoint-type methods for computing derivatives. This framework automates much of the work involved in developing computational models with multiple disciplines or at least multiple components. For example, the framework centrally handles: parallel data passing between components, implementing solvers, and forming and solving the linear systems required for computing derivatives.

In Ch. 2, I motivate the need for a new framework in more detail, by reviewing existing frameworks and highlighting the unique features of the new framework. Chapter 3 presents a new mathematical formulation I developed that represents all computational models as a system of algebraic equations. This monolithic formulation provides the foundation for the framework, and it is what enables the automated computation of the simulation and the derivatives with a compact and simple framework design. In Ch. 4, I show how I used this monolithic formulation to develop an equation that unifies all methods for computing discrete derivatives. For instance, the black-box finite-difference method, algorithmic differentiation, the chain rule, and the adjoint method can all be derived from this single equation by making the appropriate choice of variables in the algebraic system. Chapter 5 presents a compact algorithmic framework design that I developed based on the monolithic mathematical formulation and the derivative unification equation. In this framework design, the algebraic system is hierarchically decomposed, and all components and solvers are homogeneously treated as instances of a general *System* class that has an interface consisting of only 4 primary operations—computing the residuals, driving them to zero, computing the Jacobian of the residuals, and applying the inverse of the Jacobian as an operator. In Ch. 6, I apply the framework to the optimization of a nanosatellite involving over 25,000 design variables. In the process, I demonstrate that it is possible to simultaneously optimize all 7 disciplines of the satellite design problem, and that smoothing the discontinuities and discrete data in the problem allows gradient-based methods to be applied. In Ch. 7, I present the application of the framework to another problem, the simultaneous allocation and mission optimization of commercial aircraft. This implementation tests the framework's built-in hierarchical nonlinear and linear solvers, as it uses the framework's solvers exclusively. Finally, Ch. 8 presents a summary and contributions, significance, and recommendations for future work.

1.4.2 Aircraft shape design

Another large class of engineering design problems involves shape design. Geometry plays an important role in engineering design, as evidenced by the widespread commercial success of computer-aided design (CAD) packages, thanks to their versatility and ease of use. However, one disadvantage of CAD is that the mapping between parameters and the final geometry is not always differentiable, making parametric CAD unsuitable for gradient-based optimization. In aircraft design, gradient-based optimization is commonly used in aerodynamic, structural, and aerostructural optimization problems. Thus, there is a need for a parametrization for the outer mold line (OML) and structural mesh that maintains the benefits of CAD and is differentiable.

In Part II of this thesis, I address this need with a methodology for parametrizing the aircraft geometry and structural mesh in a differentiable way. I have implemented these methods in GeoMACH, an open-source high-fidelity aircraft parametrization tool suite. GeoMACH supports unconventional configurations, and it allows for the definition of high-level aircraft design parameters such as span and sweep, in addition to low-level parameters that provide fine control of the shapes of airfoils and cross sections. It automatically generates unstructured quadrilateral meshes of aircraft structures, and computes the mapping from the geometry parameters to the structural nodes. GeoMACH also computes sparse Jacobians of the mappings from the parameters to the OML and to the full structural mesh.

Chapter 9 motivates high-fidelity aircraft design optimization, and explains how the new parametrization of aircraft geometries and structures addresses a need. In Ch. 10, I describe the geometry parametrization and explain how it maintains differentiability by using Bezier and bilinear interpolation to define the junctions between aircraft components. This contrasts with existing geometry tools that run intersection algorithms to combine aircraft components together, which does not yield a smooth mapping. Chapter 11 presents the novel unstructured quadrilateral mesh generation algorithm that I developed to enable automatic generation of meshes for the airframe structure. In Ch. 12, I provide aerostructural analysis results as well as aerodynamic shape optimization results for unconventional aircraft configurations to demonstrate the capabilities of GeoMACH. Chapter 13 concludes with a summary and contributions, significance, and recommendations for future work.

Part I

A general computational modeling framework

CHAPTER 2

The need for a new computational framework

In Part I of this thesis, I address the problem of facilitating gradient-based optimization in multidisciplinary problems by developing a computational modeling framework that automates the solution of multidisciplinary systems and the computation of derivatives.

In Ch. 3, I describe the mathematical foundation for the framework, which is the formulation of all the variables in a computational model as the solution of a single system of algebraic equations. I show in Ch. 4 that all the methods for computing the derivatives of the computational model can be derived from a single equation I developed using the algebraic system. In Ch. 5, I present a compact and minimalistic framework design that implements the theory from Ch. 3 and Ch. 4 using a single type of object with an interface consisting of only 5 methods. As a demonstration, Ch. 6 and Ch. 7 present two engineering problems implemented and executed entirely within the framework: the optimization of a nanosatellite and the simultaneous allocation-mission optimization of commercial aircraft. Finally, Ch. 8 provides a summary of contributions and recommendations from Part I.

The current chapter provides a motivation for Part I. Section 2.1 describes the role of frameworks in computational modeling, Sec. 2.2 surveys existing frameworks, and Sec. 2.3 introduces the current framework.

2.1 Modularity in computational modeling

Computational models have a ubiquitous presence in many scientific fields. They are widely used in the natural sciences, engineering, medicine, finance, and many other fields involving systems and processes that are complex and difficult to fully understand. Computational models are employed to assist in design (e.g. of an airplane), making decisions (e.g. about an investment portfolio), enhancing understanding (e.g. of a cell), and forecasting (e.g. a weather system). They are useful because they provide an inexpensive and practical alternative to real-world experimentation and observation.

The development of a computational model begins with the *mathematical model*, defined here as the continuous set of equations obtained by applying the relevant physical laws and assumptions for the problem. Mathematical models often involve field quantities that vary over space, time, or both, along with the relationships between them, which are the governing equations that must be enforced at every point. Discretizing by applying the appropriate numerical methods yields the *numerical model*, consisting of the finite-dimensional vectors and discrete algebraic equations. The *computational model* is formally defined as the computer program that implements the necessary solution algorithms and software design to compute the variables in the numerical model. The computational model can be made up of *components*—useful, standalone pieces of code with clear inputs and outputs— and can itself be seen as one of the components that make up another computational model.

There are significant challenges to performing large-scale simulation and optimization because of the implementation effort involved. These challenges are compounded in multidisciplinary computational models made up of components corresponding to different types of physics. One strategy for managing the resulting code complexity is the use of a *computational modeling framework*. This refers to a software library that facilitates the integration of the components that constitute a computational model.

2.2 Existing frameworks

There are many existing frameworks developed in the industry and in academia, but they do not formulate problems in a sufficiently general way that provides the efficiency required for large-scale simulation while computing the derivatives required for large-scale optimization using the analytic methods.

2.2.1 Existing commercial frameworks

Many commercial frameworks have been very successful, especially among engineers in the industry. These include Phoenix Integration's ModelCenter/CenterLink, Dassault Systemes' Isight/SEE, TechnoSoft's AML suite, MathWorks' MATLAB/Simulink, Noesis Solutions' Optimus, and Vanderplaats' VisualDOC. Underlying their differences, they have four fundamental features in common: a graphical user interface (GUI), software libraries, interfaces to external software, and grid computing management. The first feature is a GUI that allows the construction of computational models from smaller components through a drag-and-drop interface and provides tools for post-processing and visualization of results. Software libraries are suites of reusable components for general-purpose use—e.g. optimizers, stochastic modeling tools, and surrogate models. The third feature is a set of built-in interfaces to existing commercial computer-aided design and engineering software. Finally, grid computing management typically involves a simple graphical interface that automates the handling of parallel computing jobs running on remote clusters.

These features are all valuable and have contributed to the widespread use of these frameworks in the engineering industry, but they have seen limited adoption for high-performance computing (HPC) applications in the research community. HPC applications involve large systems of linear and nonlinear equations that are solved using advanced numerical algorithms and parallel computing. The aforementioned frameworks are primarily designed to integrate either smaller scale components or already mature HPC codes in a minimally invasive way, effectively treating them as black boxes.

2.2.2 Existing HPC frameworks

The modular development of HPC codes is addressed by many framework designs presented in the literature. For instance, CACTUS [21] is an open-source framework originally developed in the physics field to facilitate the solution of partial differential equations (PDEs) with parallel computing. Another open-source framework is SCIRun [22], which originated from medical applications, but also aims to facilitate a modular approach to parallel scientific computing, especially in multidisciplinary problems. NWChem [23] is a third example of an open-source framework, designed for efficient, parallel solution of large systems of equation arising in computational chemistry. MOOSE [24] is another open-source framework for the parallel solution of coupled, multi-physics systems of equations with finite-element discretizations.

All of these HPC-focused frameworks provide distributed data passing and various PDE-specific utilities, in addition to some of the features the commercial frameworks provide—visualization tools, software libraries, and grid computing management. The downside is that the provided features are often specialized to the types of PDEs and PDE solvers for which the framework is designed. An example that is more general is the common component architecture (CCA) [25], a model for component-based implementation of HPC software, but it is a general set of specifications rather than a framework with built-in features.

2.2.3 Existing MDO frameworks

Many frameworks have also been developed in the context of multidisciplinary design optimization (MDO). Padula and Gillian [26] review these and note that modularity, data handling, parallel processing, and user interface are the most important features of a framework that facilitates optimization with multiple disciplines. Prior to this, Salas and Townsend [27] had performed a similar survey and listed more detailed requirements for an MDO framework, categorized into the framework's architectural design, problem construction, problem execution, and data access. The existing MDO frameworks are susceptible to the same missing features as the aforementioned commercial and HPC-focused frameworks. One notable exception is pyMDO [28], which automatically computes derivatives using analytic methods [29] though in a less general form than proposed in this thesis. Moreover, pyMDO focuses on facilitating the implementation of MDO architectures, so it lacks several other features such as built-in solvers for HPC and parallel data transfer.

2.3 The current framework

As previously mentioned, one advantage of the current framework is that it handles PDE-type components efficiently and in an active way, solving the system of equations centrally. At the same time, it handles components that explicitly map inputs to outputs not as a special case, but using the same mathematical formulation. The other advantage is that derivatives of full numerical model are computed efficiently and accurately using the analytic methods.

The difference in approach between the current and all existing frameworks is illustrated in Fig. 2.1. Existing frameworks do not attempt to change the original numerical model in any fashion; rather, they focus on the implementation of the computational model. They provide software libraries that can facilitate the development

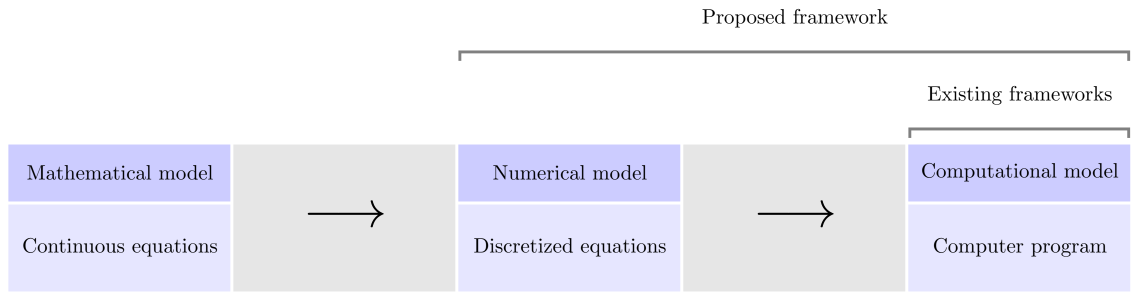


Figure 2.1: The role of the proposed computational modeling framework compared to that of existing frameworks in the development of a computational model. The proposed framework uses a unified formulation of the numerical model to enable features not provided by existing frameworks which do not change or reformulate the numerical model.

of the computational model provided that the components follow a paradigm and the application programming interface (API) dictated by the framework. In contrast, the computational modeling framework presented in this thesis goes further and applies a similar philosophy to the numerical model as well. Each component's numerical model must be reformulated to fit within a paradigm, albeit with minimal effort, and this is what enables the current framework to provide additional features.

The following chapters present a unique but simple framework design that addresses these missing features and lowers the entry barrier for large-scale simulation and optimization. It facilitates large-scale simulations by automating the parallel execution of components and parallel data passing, and by centrally solving the coupling between components and the systems of equations from discretized PDEs in an efficient way. It also enables large-scale optimization by implementing the analytic methods to centrally compute the derivatives of the overall computational models given those of the constituent components.

This framework is able to provide these automated features despite the possible heterogeneity in type and scale of the components by using a unique yet simple mathematical and algorithmic design. Mathematically, the framework formulates the entire numerical model as a set of variables and a set of corresponding residual functions that define a system of equations. Thus, the framework unifies all components as objects that define variables implicitly through residual functions, regardless of whether a component solves a complex, discretized PDE or it simply evaluates an explicit function. In terms of the algorithms, the framework applies a hierarchical solution strategy to the overall system of equations using built-in iterative nonlinear and linear solvers while incorporating any solvers the user provides. With this approach, it unifies all framework operations as part of the solution of a nonlinear or linear system. For instance, data transfers, executing components in a sequence, solving a discretized PDE in a component, or converging two coupled components using fixed-point iteration are all parts of various methods for solving nonlinear systems.

CHAPTER 3

A monolithic formulation

This chapter presents the monolithic reformulation of numerical models and the resulting benefits. Section 3.1 describes the notation for any numerical model. Sections 3.2 and 3.3 formulate the general numerical model as a single system of algebraic equations and hierarchical decompose it into smaller systems of equations by recursively partitioning the set of unknowns. Section 3.4 presents a solution existence proof to show when it is valid to form these systems of equations. Finally, Section 3.5 shows that the derivatives of interest can be computed by solving a system of linear equations which unifies the existing methods for computing derivatives.

3.1 Notation for a general numerical model

A numerical model was previously defined to consist of the discretized quantities and the finite set of algebraic equations that define their values. This section defines the nomenclature for a general numerical model.

In this context, a *variable* represents a vector of a single type of physical or abstract quantity in a numerical model. In many settings, each individual scalar is treated as a separate variable; however, in the current context, a group of scalars representing the same quantity—such as a vector comprised of temperature values at different time instances—is collectively referred to as a single variable. The only exception is design variables; each design variable is a scalar that is varied by the optimizer to minimize or maximize the objective function.

Fundamentally, numerical models capture the relationships between quantities—i.e., the response of one or more quantities to changes in others. Thus, it is useful to classify the variables in a numerical model as either input, state, or output. Input variables are independent variables whose values are set externally; output variables are the dependent variables of interest; and state variables are dependent variables computed in the process of computing the output variables.

The following notation is used to characterize a general numerical model. Let $x_1, \ldots, x_m$ denote the input variables; $y_1, \ldots, y_p$ denote the state variables; and $f_1, \ldots, f_q$ denote the output variables. Let $D^x_k \subseteq \mathbb{R}^{N^x_k}$, $D^y_k \subseteq \mathbb{R}^{N^y_k}$, and $D^f_k \subseteq \mathbb{R}^{N^f_k}$ be the sets of permitted values of the kth input variable, the kth state variable, and the kth output variable, respectively, such that the variables satisfy $x_k \in D^x_k$ for $k = 1, \ldots, m$; $y_k \in D^y_k$ for $k = 1, \ldots, p$; and $f_k \in D^f_k$ for $k = 1, \ldots, q$. Each of these sets is a Cartesian product of intervals defined by the lower and upper bounds for the variable.

Each output variable f_k is computed by a function of the state and input variables that is denoted $\mathcal{F}_k : D^x_1 \times \cdots \times D^x_m \times D^y_1 \times \cdots \times D^y_p \to D^f_k$. Each state variable y_k is computed by a function of the other state variables and the input variables, denoted $\mathcal{Y}_k : D^x_1 \times \cdots \times D^x_m \times D^y_1 \times \cdots \times D^y_{k-1} \times D^y_{k+1} \times \cdots \times D^y_p \to D^y_k$. Moreover, each state variable can be classified as one of two types, explicit and implicit. The value of an explicit state variable is defined by $\mathcal{Y}_k$ and no other information is known; the value of an implicit state variable is defined by the implicit function $\mathcal{R}_k : D^x_1 \times \cdots \times D^x_m \times D^y_1 \times \cdots \times D^y_p \to \mathbb{R}^{N^y_k}$ and $\mathcal{Y}_k$ can be interpreted as the function that solves the implicit equation. Thus, the numerical model is defined by

$$\boxed{\begin{array}{ll} y_k = \begin{cases} \mathcal{Y}_k(x_1, \ldots, x_m, y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_p), & \text{explicit} \\ a \,|\, \mathcal{R}_k(x_1, \ldots, x_m, y_1, \ldots, y_{k-1}, a, y_{k+1}, \ldots, y_p) = 0, & \text{implicit} \end{cases}, & 1 \leq k \leq p \\ f_k = \mathcal{F}_k(x_1, \ldots, x_m, y_1, \ldots, y_p), & 1 \leq k \leq q. \end{array}} \tag{3.1}$$

3.2 A monolithic formulation of the numerical model

This section presents the reformulation of the numerical model described by Eq. (3.1) in the previous section as a single system of algebraic equations. The choice of values for the input variables changes the algebraic system; thus, it is necessary to make this choice before proceeding. Let us assume x^*_k is the value of input variable x_k for all $k = 1, \ldots, m$, at the point at which the numerical model is being evaluated.

The first step is to concatenate the set of variables into one vector,

$$u = (v_1, \ldots, v_n) = (x_1, \ldots, x_m, y_1, \ldots, y_p, f_1, \ldots, f_q), \tag{3.2}$$

where $n = m + p + q$. As before, the variables are restricted to be in a set of permitted values: $v_k \in D_k$ where $D_k \subseteq \mathbb{R}^{N_k}$ is a Cartesian product of N_k intervals, for $k = 1, \ldots, n$.

Define functions $R_k : D_1 \times \cdots \times D_n \to \mathbb{R}^{N_k}$ for $k = 1, \ldots, n$ where

$$\begin{aligned} &R_k(u) = x_k - x_k^* && , 1 \leq k \leq m \\ &R_{m+k}(u) = \begin{cases} y_k - \mathcal{Y}_k(x_1, \ldots, x_m, y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_p) & \text{, explicit} \\ \quad -\mathcal{R}_k(x_1, \ldots, x_m, y_1, \ldots, y_p) & \text{, implicit} \end{cases} && , 1 \leq k \leq p \\ &R_{m+p+k}(u) = f_k - \mathcal{F}_k(x_1, \ldots, x_m, y_1, \ldots, y_p) && , 1 \leq k \leq q. \end{aligned} \tag{3.3}$$

A minus sign is present for the implicit state variables because including it yields more intuitive formulae when computing derivatives.

Let $R : D \to \mathbb{R}^n$ be defined by $R = (R_1, \ldots, R_n)$, where $D = D_1 \times \cdots \times D_n$. Then, the numerical model can be formulated as the following algebraic system of equations:

$$\boxed{\left.\begin{array}{c} R_1(v_1, \ldots, v_n) = 0 \\ \vdots \\ R_n(v_1, \ldots, v_n) = 0 \end{array}\right\} \Leftrightarrow R(u) = 0.} \tag{3.4}$$

The unified formulation of any numerical model is the algebraic system of equations $R(u) = 0$, referred to hereafter as the *fundamental system*. Its significance is due to the fact that the vector u^* that solves Eq. (3.4) satisfies Eq. (3.1); thus, the solution of the fundamental system is equivalent to the outcome of evaluating the numerical model.

3.3 Hierarchical decomposition

To enable block Gauss–Seidel-like methods for solving Eq. (3.4), it is useful to partition the set of unknowns to form smaller systems of equations. Moreover, this partitioning strategy can be recursively applied to the smaller systems of equations, resulting in a hierarchical decomposition of the original algebraic system in Eq. (3.4).

Let $S = \{i_1 + 1, \ldots, i_2\}$ be an index set where $i_1, i_2 \in \mathbb{N}$ and $i_1 < i_2$. Let $R_S : D_1 \times \cdots \times D_n \to \mathbb{R}^{N_{i_1+1}} \times \cdots \times \mathbb{R}^{N_{i_2}}$ be defined by concatenating the residual functions corresponding to the indices in S. Let $p_S = (v_1, \ldots, v_{i_1}, v_{i_2+1}, \ldots, v_n)$ and $u_S = (v_{i_1+1}, \ldots, v_{i_2})$ partition u into the *parameter vector* and *unknown vector*, respectively.

Then, the system of equations corresponding to R_S is

$$\boxed{\left.\begin{aligned} R_{i_1+1}(v_1,\dots,v_{i_1},v_{i_1+1},\dots,v_{i_2},v_{i_2+1},\dots,v_n) &= 0 \\ \vdots\qquad\qquad & \\ R_{i_2}(\underbrace{v_1,\dots,v_{i_1}}_{p_S},\underbrace{v_{i_1+1},\dots,v_{i_2}}_{u_S},\underbrace{v_{i_2+1},\dots,v_n}_{p_S}) &= 0 \end{aligned}\right\} \Leftrightarrow R_S(p_S,u_S)=0,} \tag{3.5}$$

which is referred to as an *intermediate system.*

3.4 Validity of the monolithic, hierarchical formulation

The validity of the fundamental system in Eq. (3.4) and intermediate systems in Eq. (3.5) hinges on the existence of solutions for each system. In the former case, the question concerns whether the numerical model represents a well-posed problem, while in the latter case, the question is whether it is possible to solve for a given intermediate system's unknowns as a function of its parameters. This section presents conditions that guarantee existence of solutions in both situations.

Theorem 3.1. *Let $D_k \subseteq \mathbb{R}^{N_k}$ be a Cartesian product of intervals for each $k=1,\dots,n$ and define $D = D_1\times\cdots\times D_n$. Let $R_k : D \to \mathbb{R}^{N_k}$ be a residual function for each $k=1,\dots,n$.*

If there exist continuous functions

$$F_k : D_1\times\cdots\times D_{k-1}\times D_{k+1}\times\cdots\times D_n \to D_k \tag{3.6}$$

that solve $R_k(v_1,\dots,v_N)=0$ for $k=1,\dots,n$ and at most one of $D_1,\dots,D_n$ is unbounded, then the following hold:

1. *there exists $v_k^* \in D_k$ for each $k=1,\dots,n$ such that*

$$\begin{aligned} R_1(v_1^*,\dots,v_n^*) &= 0 \\ &\vdots \\ R_n(v_1^*,\dots,v_n^*) &= 0. \end{aligned} \tag{3.7}$$

2. *for all index sets $S=\{i_1+1,\dots,i_2\}\subseteq\{1,\dots,n\}$, there exists a function*

$$F_S : D_1\times\cdots\times D_{i_1}\times D_{i_2+1}\times\cdots\times D_n \to D_{i_1+1}\times\cdots\times D_{i_2} \tag{3.8}$$

that solves $R_k(v_1,\dots,v_n)=0\ \forall k\in S$.

Proof. The proof of the first part is as follows. Without loss of generality, assume D_1 is the unbounded set if one of $D_1, \ldots, D_n$ is unbounded. Define

$$G : \begin{pmatrix} v_2 \\ \vdots \\ v_n \end{pmatrix} \mapsto \begin{pmatrix} F_2(F_1(v_2, \ldots, v_n), v_3, \ldots, v_n) \\ \vdots \\ F_n(F_1(v_2, \ldots, v_n), v_2, \ldots, v_{n-1}) \end{pmatrix} \tag{3.9}$$

The function G is a composition of continuous functions so G itself is also continuous. The domain and codomain of G are both $D_2 \times \cdots \times D_n$, which is a convex and compact subset of a Euclidean space since $D_2, \ldots, D_n$ are Cartesian products of intervals that are bounded. Given these two properties, the Brouwer fixed-point theorem guarantees the existence of a point $(v_2^*, \ldots, v_n^*)$ such that

$$\begin{aligned} v_2^* &= F_2(F_1(v_2^*, \ldots, v_n^*), v_3^*, \ldots, v_n^*) \\ &\vdots \\ v_n^* &= F_n(F_1(v_2^*, \ldots, v_n^*), v_2^*, \ldots, v_{n-1}^*). \end{aligned} \tag{3.10}$$

Defining $v_1^* = F_1(v_2^*, \ldots, v_n^*)$, it is easy to see that $(v_1^*, \ldots, v_n^*)$ solves the system of equations defined by $R_1, \ldots, R_n$:

$$\begin{aligned} R_1(v_1^*, v_2^*, \ldots, v_n^*) &= R_1(F_1(v_2^*, v_3^*, \ldots, v_n^*), v_2^*, \ldots, v_n^*) = 0 \\ R_2(v_1^*, v_2^*, \ldots, v_n^*) &= R_2(v_1^*, F_2(v_1^*, v_3^*, \ldots, v_n^*), \ldots, v_n^*) = 0 \\ &\vdots \\ R_n(v_1^*, v_2^*, \ldots, v_n^*) &= R_n(v_1^*, \ldots, v_{n-1}^*, F_n(v_1^*, v_2^*, \ldots, v_{n-1}^*)) = 0, \end{aligned} \tag{3.11}$$

as required.

The proof of the second part follows from that of the first part. Given an index set $S = \{i_1 + 1, \ldots, i_2\} \subseteq \{1, \ldots, N\}$, define $S^c = \{1, \ldots, N\} \backslash S$. Taking any $v_k \in D_k \ \forall k \in S^c$ and applying the result of part (i) with $v_k \ \forall k \in S^c$ held fixed yields a solution $(v_{i_1+1}^*, \ldots, v_{i_2}^*)$ that satisfies $R_k(v_1, \ldots, v_n) = 0 \ \forall k \in S$. This defines the required function. □

This theorem can be interpreted in two ways. First, if all but one variable in the numerical model has lower and upper bounds and each variable can be continuously solved for in terms of the others, then this theorem guarantees that the full coupled numerical model has a solution. Furthermore, any intermediate system defined by selecting any subset of the variables is guaranteed to also have a solution for each set of values for the remaining variables. This is a necessary condition for hierarchical block Gauss–Seidel or Jacobi-type methods to be used.

In the second situation, it is assumed to be known that the coupled numerical model has a solution. The remaining question is then whether an intermediate system of interest is valid—i.e., whether the variables that constitute its unknown vector can be solved for in terms of the remaining variables. This theorem guarantees that this is the case if all or all but one of the variables in the unknown vector has lower and upper bounds respected by the function defining the variable.

3.5 Computation of derivatives

The derivatives of interest are those of the output variables with respect to the input variables. Let $x = (x_1, \ldots, x_m)$, $y = (y_1, \ldots, y_p)$, and $f = (f_1, \ldots, f_q)$ be vectors and let $\mathcal{Y} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_p)$ and $\mathcal{F} = (\mathcal{F}_1, \ldots, \mathcal{F}_q)$ be functions. Then, the numerical model is encapsulated in the function $\mathcal{G} : x \mapsto \mathcal{F}(x, \mathcal{Y}(x))$, which maps the inputs x to the outputs f and the derivatives of interest are contained in $\partial\mathcal{G}/\partial x$.

Methods for computing derivatives can be divided into 4 categories: finite-step methods, chain rule, analytic methods, and algorithmic differentiation. The simplest form of the first type is the finite-difference method, which is the simplest method to implement but has limited accuracy because one of either the discretization error or the subtractive cancellation error dominates, depending on the step size. The complex-step method fixes the accuracy issue by eliminating the subtractive cancellation error, but like the finite-difference method, the computation cost scales linearly with the number of input variables, which can be large in some applications. The second class of methods computes derivatives of components and combines them using the chain rule, though this method can only be applied when there is no feedback among the components. When residuals are available for the state variables, analytic methods can be applied to accurately compute derivatives of coupled systems, potentially at a cost independent of the number of the input variables. Algorithmic differentiation can have this property as well, but it requires automatic source-code modification, can have a large memory usage, and is less efficient than the analytic methods.

It is possible to derive, from the fundamental system, an equation that unifies all four types of methods. The method that is used depends on the choice of the state variables—on one extreme, including no state variables results in a finite-step method and on the other, including the variable from every line of code in a computational model effectively results in algorithmic differentiation. This is explained in detail in the next chapter.

The analytic methods are significantly more efficient than the others for many types of problems. To use this class of methods, the fundamental system must be defined with the original residuals of implicit state variables used; for explicit state variables, residuals are defined as the value of the state variables minus the output of the function, as before. The fundamental system has the form,

$$u = \begin{pmatrix} x \\ y \\ f \end{pmatrix} \quad \text{and} \quad R(u) = \begin{pmatrix} x - x^* \\ -\mathcal{R}(x, y) \\ f - \mathcal{F}(x, y) \end{pmatrix}. \tag{3.12}$$

The following proposition shows how $\partial \mathcal{G}/\partial x$ can be computed from the fundamental system defined Eq. (3.12).

Proposition 3.2. *Let R and u be defined as in Eq. (3.12) and define $\mathcal{G} : x \mapsto \mathcal{F}(x, \mathcal{Y}(x))$. If $\partial R/\partial u$ is invertible and the inverse is defined as*

$$\frac{\partial R}{\partial u}^{-1} = \begin{bmatrix} A^{[N_x \times N_x]} & A^{[N_x \times N_y]} & A^{[N_x \times N_f]} \\ A^{[N_y \times N_x]} & A^{[N_y \times N_y]} & A^{[N_y \times N_f]} \\ A^{[N_f \times N_x]} & A^{[N_f \times N_y]} & A^{[N_f \times N_f]} \end{bmatrix}, \tag{3.13}$$

then the following identity holds.

$$\frac{\partial \mathcal{G}}{\partial x} = A^{[N_f \times N_x]}, \tag{3.14}$$

where $\partial R/\partial u$ is evaluated at u^* satisfying $R(u^*) = 0$.

Proof. By construction, the following holds:

$$\begin{bmatrix} \mathcal{I} & 0 & 0 \\ -\frac{\partial \mathcal{R}}{\partial x} & -\frac{\partial \mathcal{R}}{\partial x} & 0 \\ -\frac{\partial \mathcal{F}}{\partial x} & -\frac{\partial \mathcal{F}}{\partial x} & \mathcal{I} \end{bmatrix} \begin{bmatrix} A^{[N_x \times N_x]} \\ A^{[N_y \times N_x]} \\ A^{[N_f \times N_x]} \end{bmatrix} = \begin{bmatrix} \mathcal{I} \\ 0 \\ 0 \end{bmatrix}. \tag{3.15}$$

Block forward substitution yields

$$\begin{bmatrix} A^{[N_x \times N_x]} \\ A^{[N_y \times N_x]} \\ A^{[N_f \times N_x]} \end{bmatrix} = \begin{bmatrix} \mathcal{I} \\ -\frac{\partial \mathcal{R}}{\partial y}^{-1} \frac{\partial \mathcal{R}}{\partial x} \\ \frac{\partial \mathcal{F}}{\partial x} - \frac{\partial \mathcal{F}}{\partial y} \frac{\partial \mathcal{R}}{\partial y}^{-1} \frac{\partial \mathcal{R}}{\partial x} \end{bmatrix}. \tag{3.16}$$

Now, $\mathcal{G}$ is a composition of functions mapping $x \mapsto (x, \mathcal{Y}(x))$ and $(x, y) \mapsto \mathcal{F}(x, y)$, so applying the chain rule yields

$$\frac{\partial \mathcal{G}}{\partial x} = \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial x} & \frac{\partial \mathcal{F}}{\partial y} \end{bmatrix} \begin{bmatrix} \mathcal{I} \\ \frac{\partial \mathcal{Y}}{\partial x} \end{bmatrix}. \tag{3.17}$$

Since $\partial \mathcal{R}/\partial y$ is invertible, the implicit function theorem states

$$\frac{\partial \mathcal{Y}}{\partial x} = -\frac{\partial \mathcal{R}}{\partial y}^{-1} \frac{\partial \mathcal{R}}{\partial x}. \tag{3.18}$$

Combining the two equations above yields

$$\frac{\partial \mathcal{G}}{\partial x} = \frac{\partial \mathcal{F}}{\partial x} - \frac{\partial \mathcal{F}}{\partial y} \frac{\partial \mathcal{R}}{\partial y}^{-1} \frac{\partial \mathcal{R}}{\partial x}. \tag{3.19}$$

Therefore,

$$\frac{\partial \mathcal{G}}{\partial x} = A^{[N_f \times N_x]}, \tag{3.20}$$

as required. $\square$

The application of the inverse function theorem sheds insight as to why the lower left $N_f \times N_x$ block of $\partial R/\partial u$ is equal to the matrix of derivatives commonly understood as df/dx in practice. Assuming $\partial R/\partial u$ is invertible, the inverse function theorem guarantees the existence of a locally defined inverse function $R^{-1} : r \mapsto u|R(u) = r$ which satisfies

$$\frac{\partial (R^{-1})}{\partial r} = \left[\frac{\partial R}{\partial u}\right]^{-1}. \tag{3.21}$$

The concept of a total derivative is used in many settings, but it is difficult to find a clear definition in the literature—total derivatives are usually defined in terms of other total derivatives as in

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\partial F}{\partial x} + \frac{\partial F}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}x}, \tag{3.22}$$

staying with the convention that functions are capitalized.

Total derivatives are useful for distinguishing direct and indirect dependence on a variable. In the example above, the total derivative df/dx captures both the explicit dependence of F on the argument x and the indirect dependence of F on x via y.

The Jacobian $\partial(R^{-1})/\partial r$ captures a similar relationship because the (i, j)th entry of the matrix $\partial(R^{-1})/\partial r$ captures the dependence of the ith component of u on the

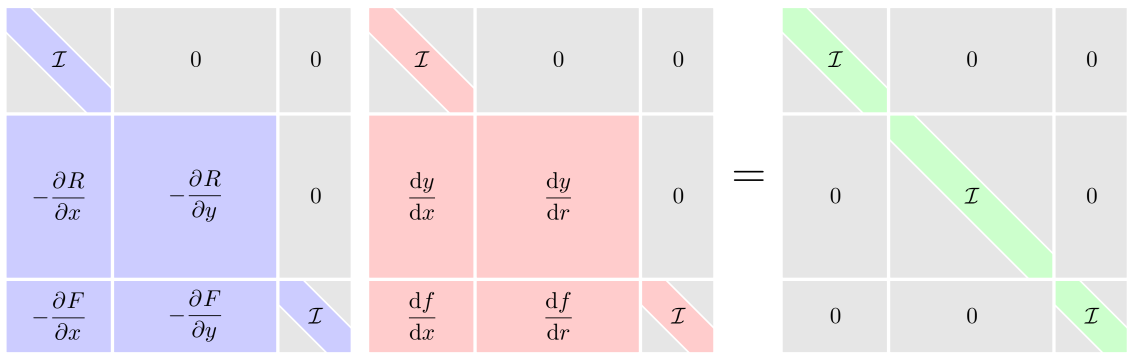

$\mathcal{I}$
0
0
$\mathcal{I}$
0
0
$\mathcal{I}$
0
0
$-\frac{\partial R}{\partial x}$
$-\frac{\partial R}{\partial y}$
0
$\frac{\mathrm{d}y}{\mathrm{d}x}$
$\frac{\mathrm{d}y}{\mathrm{d}r}$
0
=
0
$\mathcal{I}$
0
$-\frac{\partial F}{\partial x}$
$-\frac{\partial F}{\partial y}$
$\mathcal{I}$
$\frac{\mathrm{d}f}{\mathrm{d}x}$
$\frac{\mathrm{d}f}{\mathrm{d}r}$
$\mathcal{I}$
0
0
$\mathcal{I}$

Figure 3.1: The block structure of the matrices in the left equality of Eq. (3.24).

jth component of r directly, but also indirectly via the other components of u since they also change when the jth component of r changes. This motivates the following definition of the total derivative.

Definition 3.3. Given the algebraic system of equations $R(u) = 0$, assume $\partial R/\partial u$ is invertible at the solution of this system. The matrix of total derivatives du/dr is defined to be

$$\frac{\mathrm{d}u}{\mathrm{d}r} = \frac{\partial (R^{-1})}{\partial r}, \tag{3.23}$$

where $\partial(R^{-1})/\partial r$ is evaluated at $r = 0$.

Following from Eq. (3.21), the matrix du/dr is also equal to the inverse of $\partial R/\partial u$, leading to

$$\boxed{\frac{\partial R}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}r} = \mathcal{I} = \frac{\partial R}{\partial u}^T \frac{\mathrm{d}u}{\mathrm{d}r}^T}, \tag{3.24}$$

which is referred to as the *unifying chain rule equation* [13]. The left and right equalities are denoted the *forward mode* and the *reverse mode*, respectively, drawing inspiration from terminology in algorithmic differentiation.

For independent and explicit variables, the r in the denominator of du/dr can be replaced with the symbol for the variable itself, as shown in Fig. 3.1. The derivatives of interest are df/dx, which is a sub-block of du/dr. Computing df/dx involves solving a linear system with multiple right-hand sides, so this is more efficient with the left or right equality in Eq. (3.24), depending on the relative sizes of f and x.

CHAPTER 4

A unification of methods for computing derivatives

This chapter shows that all discrete methods for computing derivatives can be derived from the unifying chain rule (3.24). Each method corresponds to a specific choice of the variables in u and the residuals R. To illustrate this, I present these choices and the derivation from the chain rule in figures with a common layout. Figure 4.1 shows the layout for the general case, with no specific choice of variables or residuals.

The top box shows the definition of the variables and residuals on the right, and a diagram showing the dependence of the residual functions on the variables on the left. The diagram uses the extended design structure matrix (XDSM) standard developed by Lambe and Martins [1], which enables the representation of both the data dependency and the procedure for a given algorithm. The diagonal entries are the functions in the process, and the off-diagonal entries represent the data. Here, only the data dependency information is relevant, which is expressed by the thick gray lines. The XDSM diagram in Figure 4.1 shows the residuals or vectors of residuals along the diagonal, and variables or vectors of variables in the off-diagonal positions. The off-diagonal entry in row i and column j expresses the dependence of the jth residual on the ith variable.

The middle box is used for the derivation of the method from the unifying chain rule (3.24). The two boxes at the bottom show the forward and reverse forms of the method.

The sections that follow present one such table for each of the methods, where the variables and residuals are specified differently depending on the method, and the application of the unifying chain rule to those variables and residuals yields the method.

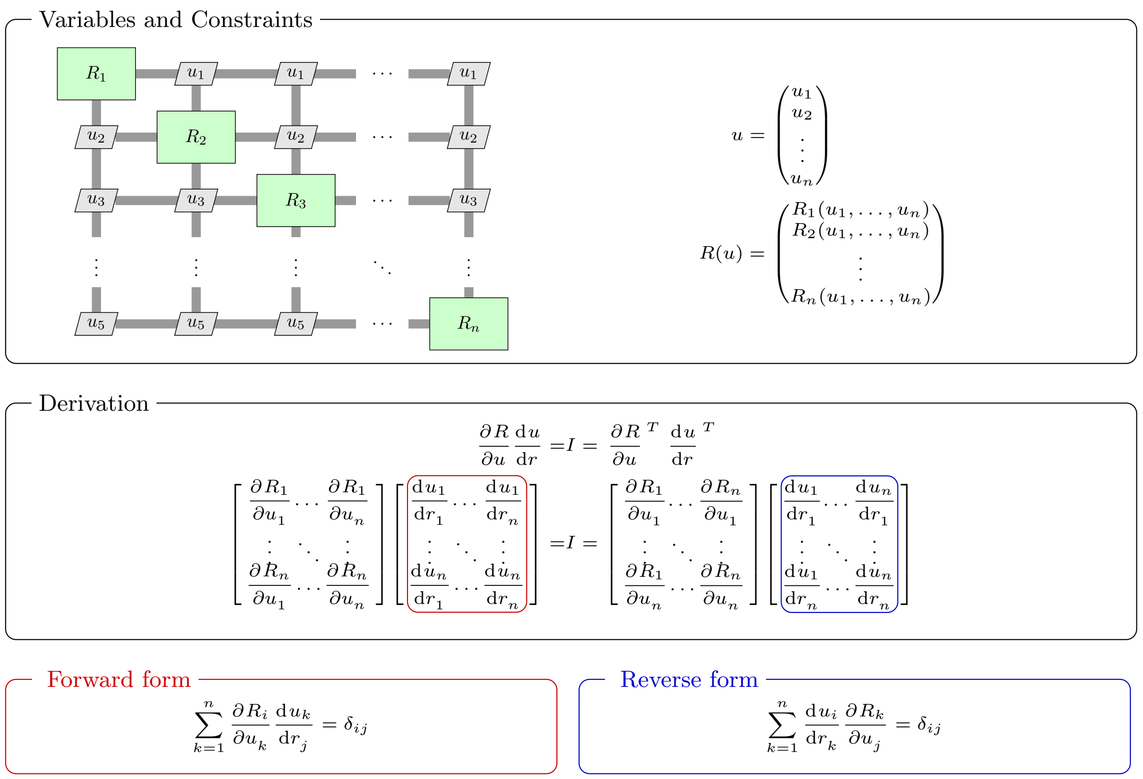

Variables and Constraints
R_1
R_2
R_3
R_n
u_1
u_2
u_3
u_5
$u = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$
$R(u) = \begin{pmatrix} R_1(u_1,\dots,u_n) \\ R_2(u_1,\dots,u_n) \\ \vdots \\ R_n(u_1,\dots,u_n) \end{pmatrix}$
Derivation
$\frac{\partial R}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}r} = I = \frac{\partial R}{\partial u}^T \frac{\mathrm{d}u}{\mathrm{d}r}^T$
Forward form
$\sum_{k=1}^{n} \frac{\partial R_i}{\partial u_k}\frac{\mathrm{d}u_k}{\mathrm{d}r_j} = \delta_{ij}$
Reverse form
$\sum_{k=1}^{n} \frac{\mathrm{d}u_i}{\mathrm{d}r_k}\frac{\partial R_k}{\partial u_j} = \delta_{ij}$

Figure 4.1: Equations for computing total derivatives in a general system of equations.

4.1 Monolithic differentiation

In monolithic differentiation, the entire computational model is treated as a "black box." This may be the only option in cases for which the source code is not available, or if it is not deemed worthwhile to implement more sophisticated approaches for computing the derivatives.

In monolithic differentiation, the only variables that are tracked are the inputs x and the outputs f. Thus, the variables are defined as $u = [x^T, f^T]^T$, as shown in Fig. 4.2. The residuals are just the residuals of the inputs and outputs, i.e., the differences between the actual values of the variables and the corresponding functions. Thus, the input variables x are simply forced to match the specified values x^*, and the output variables f are forced to match the results of the computational model, $\mathcal{F}$.

The result of applying these definitions of the variables and residuals is rather

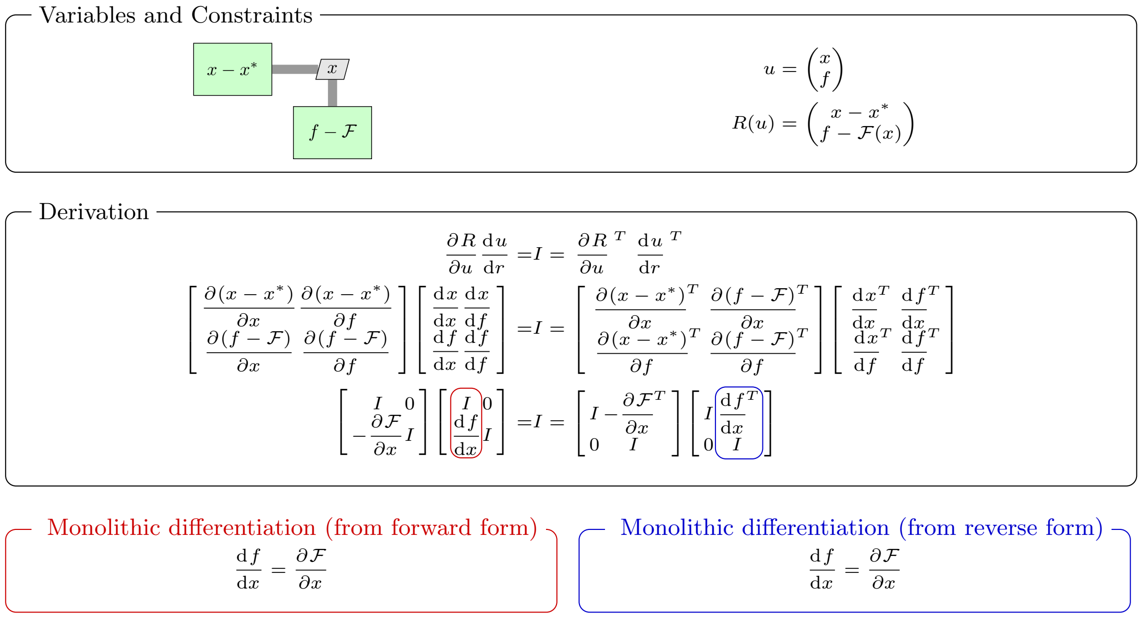

Variables and Constraints
$x - x^*$
x
$f - \mathcal{F}$
$u = \begin{pmatrix} x \\ f \end{pmatrix}$
$R(u) = \begin{pmatrix} x - x^* \\ f - \mathcal{F}(x) \end{pmatrix}$
Derivation
$\frac{\partial R}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}r} = I = \frac{\partial R}{\partial u}^T \frac{\mathrm{d}u}{\mathrm{d}r}^T$
$\begin{bmatrix} I & 0 \\ -\frac{\partial \mathcal{F}}{\partial x} & I \end{bmatrix}\begin{bmatrix} I & 0 \\ \frac{\mathrm{d}f}{\mathrm{d}x} & I \end{bmatrix} = I = \begin{bmatrix} I & -\frac{\partial \mathcal{F}}{\partial x}^T \\ 0 & I \end{bmatrix}\begin{bmatrix} I & \frac{\mathrm{d}f}{\mathrm{d}x}^T \\ 0 & I \end{bmatrix}$
Monolithic differentiation (from forward form)
$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\partial \mathcal{F}}{\partial x}$
Monolithic differentiation (from reverse form)
$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\partial \mathcal{F}}{\partial x}$

Figure 4.2: Derivation of monolithic (black box) differentiation.

trivial:

$$\frac{\mathrm{d}f_i}{\mathrm{d}x_j} = \frac{\partial \mathcal{F}_i}{\partial x_j} \tag{4.1}$$

for each input x_j and output variable f_i. This relationship is simply stating that the total derivatives of interest are the partial derivatives of the model when considered in this context.

4.2 Algorithmic differentiation

Algorithmic differentiation (AD)—also known as computational differentiation or automatic differentiation—is a well-known method based on the systematic application of the differentiation chain rule to computer programs [30, 31]. Although this approach is as accurate as an analytic method, it is potentially much easier to implement since the implementation can be done automatically.

In the AD perspective, the independent variables x and the quantities of interest f are assumed to be in the vector of variables t. To make clear the connection to the

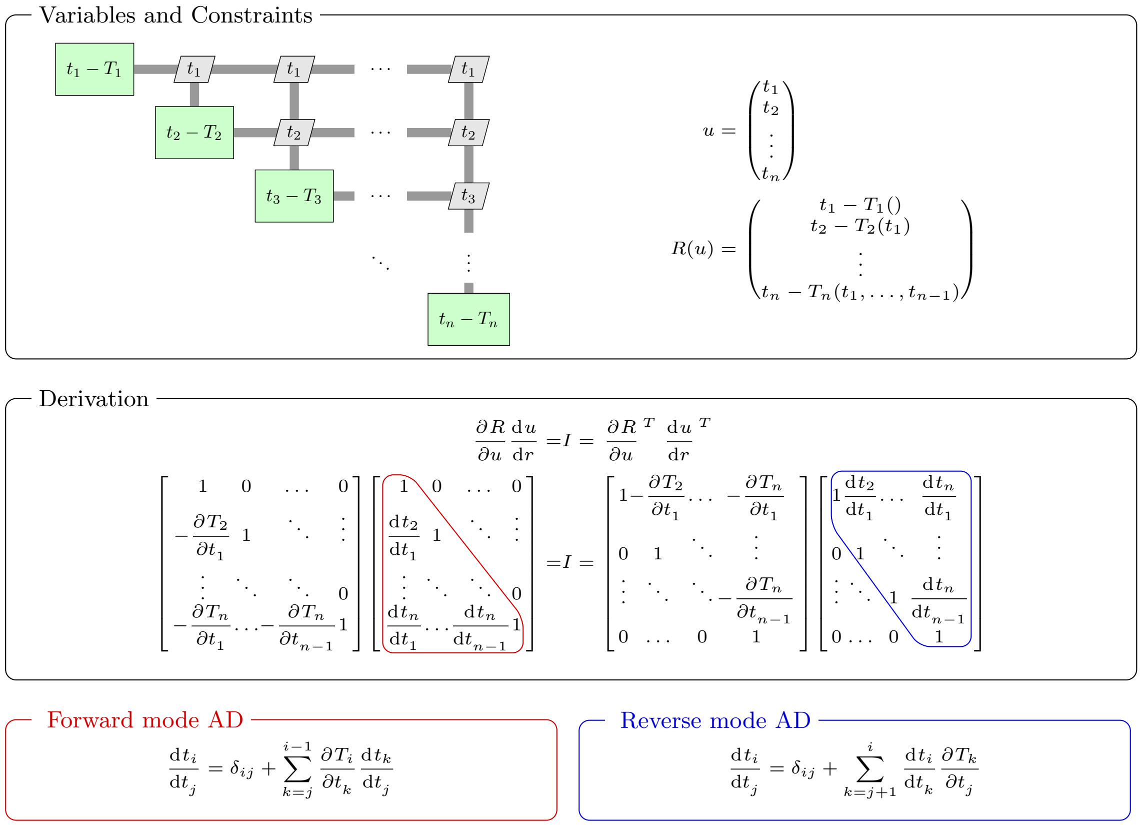

Variables and Constraints
$t_1 - T_1$
$t_2 - T_2$
$t_3 - T_3$
$t_n - T_n$
$u = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix}$
$R(u) = \begin{pmatrix} t_1 - T_1() \\ t_2 - T_2(t_1) \\ \vdots \\ t_n - T_n(t_1, \dots, t_{n-1}) \end{pmatrix}$
Derivation
$\frac{\partial R}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}r} = I = \frac{\partial R}{\partial u}^T \frac{\mathrm{d}u}{\mathrm{d}r}^T$
Forward mode AD
$\frac{\mathrm{d}t_i}{\mathrm{d}t_j} = \delta_{ij} + \sum_{k=j}^{i-1} \frac{\partial T_i}{\partial t_k}\frac{\mathrm{d}t_k}{\mathrm{d}t_j}$
Reverse mode AD
$\frac{\mathrm{d}t_i}{\mathrm{d}t_j} = \delta_{ij} + \sum_{k=j+1}^{i} \frac{\mathrm{d}t_i}{\mathrm{d}t_k}\frac{\partial T_k}{\partial t_j}$

Figure 4.3: Derivation of algorithmic differentiation.

other derivative computation methods, we group these variables as follows:

$$u = [\underbrace{t_1, \dots, t_{n_x}}_{x}, \dots, t_j, \dots, t_i, \dots, \underbrace{t_{(n-n_f)}, \dots, t_n}_{f}]^T. \tag{4.2}$$

Figure 4.3 shows this definition and the resulting derivation. Note that the XDSM diagram shows that all variables are above the diagonal, indicating that there is only forward dependence, because of the unrolling of all loops. The residuals just enforce that the variables must be equal to the corresponding function values. Using these definitions in the unifying chain rule, we obtain a matrix equation, where the matrix that contains the unknowns (the total derivatives that we want to compute) is either lower triangular or upper triangular. The lower triangular system corresponds to the forward mode and can be solved using forward substitution, while the upper triangular system corresponds to the reverse mode of AD and can be solved using back substitution.

These matrix equations can be rewritten as shown at the bottom of Fig. 4.3. The equation on the left represents forward-mode AD. In this case, we choose one t_j and keep j fixed. Then we work our way forward in the index $i = 1, 2, \ldots, n$ until we get the desired total derivative. In the process, we obtain a whole column of the lower triangular matrix, i.e., the derivatives of all the variables with respect to the chosen variable.

Using the reverse mode, shown on the bottom right of Fig. 4.3, we choose a t_i (the quantity we want to differentiate) and work our way backward in the index $j = n, n-1, \ldots, 1$ all of the way to the independent variables. This corresponds to obtaining a column of the upper triangular matrix, i.e., the derivatives of the chosen quantity with respect to all other variables.

4.3 Analytic methods

Like AD, the numerical precision of analytic methods is the same as that of the original algorithm. In addition, analytic methods are usually more efficient than AD for a given problem. However, analytic methods are much more involved than the other methods, since they require detailed knowledge of the computational model and a long implementation time.

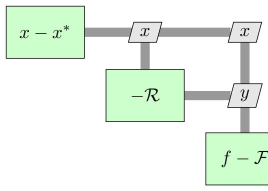
Figure 4.4 shows the derivation of the analytic methods from the unifying chain rule. In this case, the variables are the independent variables x, the state variables y, and the outputs of interest, f. The residuals force the independent variables to be equal to the specified values, and the residuals and functions of interest to be equal to the values resulting from the computational model.

Substituting these definitions in the unifying chain rule yields the familiar direct and adjoint methods, whose equations are shown at the bottom of Fig. 4.4. As we can see, the direct method comes from the forward chain rule, while the adjoint method is derived from the reverse chain rule. For the traditional derivation of the direct and adjoint methods, the reader is referred to Martins and Hwang [13].

4.4 Coupled analytic methods

We now extend the analytic methods derived in the previous section to multidisciplinary systems. The direct and adjoint methods for multidisciplinary systems can

Variables and Constraints



$$u = \begin{pmatrix} x \\ y \\ f \end{pmatrix}$$

$$R(u) = \begin{pmatrix} x - x^* \\ -\mathcal{R}(x, y) \\ f - \mathcal{F}(x, y) \end{pmatrix}$$

Derivation

$$\frac{\partial R}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}r} = I = \frac{\partial R}{\partial u}^T \frac{\mathrm{d}u}{\mathrm{d}r}^T$$

$$\begin{bmatrix} \frac{\partial(x-x^*)}{\partial x} & \frac{\partial(x-x^*)}{\partial y} & \frac{\partial(x-x^*)}{\partial f} \\ \frac{\partial(-\mathcal{R})}{\partial x} & \frac{\partial(-\mathcal{R})}{\partial y} & \frac{\partial(-\mathcal{R})}{\partial f} \\ \frac{\partial(f-\mathcal{F})}{\partial x} & \frac{\partial(f-\mathcal{F})}{\partial y} & \frac{\partial(f-\mathcal{F})}{\partial f} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}x}{\mathrm{d}x} & \frac{\mathrm{d}x}{\mathrm{d}r} & \frac{\mathrm{d}x}{\mathrm{d}f} \\ \frac{\mathrm{d}y}{\mathrm{d}x} & \frac{\mathrm{d}y}{\mathrm{d}r} & \frac{\mathrm{d}y}{\mathrm{d}f} \\ \frac{\mathrm{d}f}{\mathrm{d}x} & \frac{\mathrm{d}f}{\mathrm{d}r} & \frac{\mathrm{d}f}{\mathrm{d}f} \end{bmatrix} = I = \begin{bmatrix} \frac{\partial(x-x^*)^T}{\partial x} & \frac{\partial(-\mathcal{R})^T}{\partial x} & \frac{\partial(f-\mathcal{F})^T}{\partial x} \\ \frac{\partial(x-x^*)^T}{\partial y} & \frac{\partial(-\mathcal{R})^T}{\partial y} & \frac{\partial(f-\mathcal{F})^T}{\partial y} \\ \frac{\partial(x-x^*)^T}{\partial f} & \frac{\partial(-\mathcal{R})^T}{\partial f} & \frac{\partial(f-\mathcal{F})^T}{\partial f} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}x^T}{\mathrm{d}x} & \frac{\mathrm{d}y^T}{\mathrm{d}x} & \frac{\mathrm{d}f^T}{\mathrm{d}x} \\ \frac{\mathrm{d}x^T}{\mathrm{d}r} & \frac{\mathrm{d}y^T}{\mathrm{d}r} & \frac{\mathrm{d}f^T}{\mathrm{d}r} \\ \frac{\mathrm{d}x^T}{\mathrm{d}f} & \frac{\mathrm{d}y^T}{\mathrm{d}f} & \frac{\mathrm{d}f^T}{\mathrm{d}f} \end{bmatrix}$$

$$\begin{bmatrix} I & 0 & 0 \\ -\frac{\partial\mathcal{R}}{\partial x} & -\frac{\partial\mathcal{R}}{\partial y} & 0 \\ -\frac{\partial\mathcal{F}}{\partial x} & -\frac{\partial\mathcal{F}}{\partial y} & I \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ \frac{\mathrm{d}y}{\mathrm{d}x} & \frac{\mathrm{d}y}{\mathrm{d}r} & 0 \\ \frac{\mathrm{d}f}{\mathrm{d}x} & \frac{\mathrm{d}f}{\mathrm{d}r} & I \end{bmatrix} = I = \begin{bmatrix} I & -\frac{\partial\mathcal{R}^T}{\partial x} & -\frac{\partial\mathcal{F}^T}{\partial x} \\ 0 & -\frac{\partial\mathcal{R}^T}{\partial y} & -\frac{\partial\mathcal{F}^T}{\partial y} \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} I & \frac{\mathrm{d}y^T}{\mathrm{d}x} & \frac{\mathrm{d}f^T}{\mathrm{d}x} \\ 0 & \frac{\mathrm{d}y^T}{\mathrm{d}r} & \frac{\mathrm{d}f^T}{\mathrm{d}r} \\ 0 & 0 & I \end{bmatrix}$$

Direct method

$$\frac{\partial\mathcal{R}}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}x} = -\frac{\partial\mathcal{R}}{\partial x}$$

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\partial\mathcal{F}}{\partial x} + \frac{\partial\mathcal{F}}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}x}$$

Adjoint method

$$\frac{\partial\mathcal{R}^T}{\partial y}\frac{\mathrm{d}f^T}{\mathrm{d}r} = -\frac{\partial\mathcal{F}^T}{\partial y}$$

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\partial\mathcal{F}}{\partial x} + \frac{\mathrm{d}f}{\mathrm{d}r}\frac{\partial\mathcal{R}}{\partial x}$$

Figure 4.4: Derivation of the analytic methods: direct and adjoint.

be derived by partitioning the various variables by discipline as follows:

$$\mathcal{R} = [\mathcal{R}_1^T, \ldots, \mathcal{R}_N^T]^T, \quad y = [y_1^T, \ldots, y_N^T]^T \tag{4.3}$$

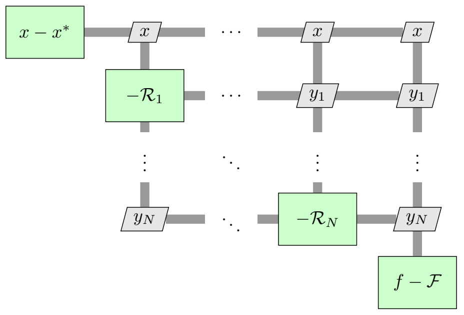
where N is the number of disciplines. All the design variables are included in x. If we substitute these vectors into the unifying chain rule, we obtain the block matrix equations shown at the bottom of Fig. 4.4. These are the coupled versions of the direct and adjoint methods, respectively. The coupled direct method was first developed by Bloebaum [32] and Sobieszczanski-Sobieski [33] while the coupled adjoint was originally developed by Martins et al. [34, 18]. Both the coupled direct and adjoint methods have since been applied to the aerostructural design optimization of aircraft wings [35, 36, 37, 7].

Figure 4.6 shows another alternative for obtaining the total derivatives of multidisciplinary systems that was first developed by Sobieszczanski-Sobieski [33] for the direct method, and by Martins et al. [18] for the adjoint method. The advantage of this approach is that we do not need to know the residuals of a given disciplinary solver but can instead use the *coupling variables*. To derive the direct and adjoint versions of this approach within our mathematical framework, we consider the same definition of the variables but replace the residuals with the residuals of the coupling variables,

$$R_i = \mathcal{Y}_i - y_i \tag{4.4}$$

where the y_i vector contains the coupling variables of the i^{th} discipline, and $\mathcal{Y}_i$ is the vector of functions that explicitly defines these variables. This leads to the equations at the bottom of Fig. 4.6, which we call the *functional* form. This contrasts with the *residual* form shown in Fig. 4.5.

Variables and Constraints



$$u = \begin{pmatrix} x \\ y_1 \\ \vdots \\ y_N \\ f \end{pmatrix}$$

$$R(u) = \begin{pmatrix} x - x^* \\ -\mathcal{R}_1(x, y_1, \dots, y_N) \\ \vdots \\ -\mathcal{R}_N(x, y_1, \dots, y_N) \\ f - \mathcal{F}(x, y_1, \dots, y_N) \end{pmatrix}$$

Derivation

$$\frac{\partial R}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}r} = I = \frac{\partial R}{\partial u}^T \frac{\mathrm{d}u}{\mathrm{d}r}^T$$

$$\begin{bmatrix} I & 0 & \dots & 0 & 0 \\ -\frac{\partial \mathcal{R}_1}{\partial x} & -\frac{\partial \mathcal{R}_1}{\partial y_1} & \dots & -\frac{\partial \mathcal{R}_1}{\partial y_N} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\frac{\partial \mathcal{R}_N}{\partial x} & -\frac{\partial \mathcal{R}_N}{\partial y_1} & \dots & -\frac{\partial \mathcal{R}_N}{\partial y_N} & 0 \\ -\frac{\partial \mathcal{F}}{\partial x} & -\frac{\partial \mathcal{F}}{\partial y_1} & \dots & -\frac{\partial \mathcal{F}}{\partial y_N} & I \end{bmatrix} \begin{bmatrix} I & 0 & \dots & 0 & 0 \\ \frac{\mathrm{d}y_1}{\mathrm{d}x} & \frac{\mathrm{d}y_1}{\mathrm{d}r_1} & \dots & \frac{\mathrm{d}y_1}{\mathrm{d}r_N} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\mathrm{d}y_N}{\mathrm{d}x} & \frac{\mathrm{d}y_N}{\mathrm{d}r_1} & \dots & \frac{\mathrm{d}y_N}{\mathrm{d}r_N} & 0 \\ \frac{\mathrm{d}f}{\mathrm{d}x} & \frac{\mathrm{d}f}{\mathrm{d}r_1} & \dots & \frac{\mathrm{d}f}{\mathrm{d}r_N} & I \end{bmatrix} = I = \begin{bmatrix} I & -\frac{\partial \mathcal{R}_1}{\partial x}^T & \dots & -\frac{\partial \mathcal{R}_N}{\partial x}^T & -\frac{\partial \mathcal{F}}{\partial x}^T \\ 0 & -\frac{\partial \mathcal{R}_1}{\partial y_1}^T & \dots & -\frac{\partial \mathcal{R}_N}{\partial y_1}^T & -\frac{\partial \mathcal{F}}{\partial y_1}^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & -\frac{\partial \mathcal{R}_1}{\partial y_N}^T & \dots & -\frac{\partial \mathcal{R}_N}{\partial y_N}^T & -\frac{\partial \mathcal{F}}{\partial y_N}^T \\ 0 & 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} I & \frac{\mathrm{d}y_1}{\mathrm{d}x}^T & \dots & \frac{\mathrm{d}y_N}{\mathrm{d}x}^T & \frac{\mathrm{d}f}{\mathrm{d}x}^T \\ 0 & \frac{\mathrm{d}y_1}{\mathrm{d}r_1}^T & \dots & \frac{\mathrm{d}y_N}{\mathrm{d}r_1}^T & \frac{\mathrm{d}f}{\mathrm{d}r_1}^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \frac{\mathrm{d}y_1}{\mathrm{d}r_N}^T & \dots & \frac{\mathrm{d}y_N}{\mathrm{d}r_N}^T & \frac{\mathrm{d}f}{\mathrm{d}r_N}^T \\ 0 & 0 & 0 & 0 & I \end{bmatrix}$$

Coupled direct: residual form

$$\begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial y_1} & \dots & \frac{\partial \mathcal{R}_1}{\partial y_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{R}_N}{\partial y_1} & \dots & \frac{\partial \mathcal{R}_N}{\partial y_N} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}y_1}{\mathrm{d}x} \\ \vdots \\ \frac{\mathrm{d}y_N}{\mathrm{d}x} \end{bmatrix} = - \begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial x} \\ \vdots \\ \frac{\partial \mathcal{R}_N}{\partial x} \end{bmatrix}$$

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\partial \mathcal{F}}{\partial x} + \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial y_1} & \dots & \frac{\partial \mathcal{F}}{\partial y_N} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}y_1}{\mathrm{d}x} \\ \vdots \\ \frac{\mathrm{d}y_N}{\mathrm{d}x} \end{bmatrix}$$

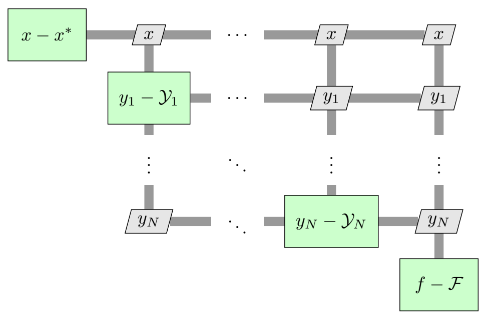
Coupled adjoint: residual form

$$\begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial y_1}^T & \dots & \frac{\partial \mathcal{R}_N}{\partial y_1}^T \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{R}_1}{\partial y_N}^T & \dots & \frac{\partial \mathcal{R}_N}{\partial y_N}^T \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}f}{\mathrm{d}r_1}^T \\ \vdots \\ \frac{\mathrm{d}f}{\mathrm{d}r_N}^T \end{bmatrix} = - \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial y_1}^T \\ \vdots \\ \frac{\partial \mathcal{F}}{\partial y_N}^T \end{bmatrix}$$

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\partial \mathcal{F}}{\partial x} + \begin{bmatrix} \frac{\mathrm{d}f}{\mathrm{d}r_1} & \dots & \frac{\mathrm{d}f}{\mathrm{d}r_N} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial x} \\ \vdots \\ \frac{\partial \mathcal{R}_N}{\partial x} \end{bmatrix}$$

Figure 4.5: Derivation of the residual form of the coupled derivative methods.

Variables and Constraints



$$u = \begin{pmatrix} x \\ y_1 \\ \vdots \\ y_N \\ f \end{pmatrix}$$

$$R(u) = \begin{pmatrix} x - x^* \\ y_1 - \mathcal{Y}_1(x, y_2, \ldots, y_N) \\ \vdots \\ y_N - \mathcal{Y}_N(x, y_1, \ldots, y_{N-1}) \\ f - \mathcal{F}(x, y_1, \ldots, y_N) \end{pmatrix}$$

Derivation

$$\frac{\partial R}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}r} = I = \frac{\partial R}{\partial u}^T \frac{\mathrm{d}u}{\mathrm{d}r}^T$$

$$\begin{bmatrix} I & 0 & \ldots & 0 & 0 \\ -\frac{\partial \mathcal{Y}_1}{\partial x} & I & \ldots & -\frac{\partial \mathcal{Y}_1}{\partial y_N} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\frac{\partial \mathcal{Y}_N}{\partial x} & -\frac{\partial \mathcal{Y}_N}{\partial y_1} & \ldots & I & 0 \\ -\frac{\partial \mathcal{F}}{\partial x} & -\frac{\partial \mathcal{F}}{\partial y_1} & \ldots & -\frac{\partial \mathcal{F}}{\partial y_N} & I \end{bmatrix} \begin{bmatrix} \boxed{\begin{matrix} I \\ \frac{\mathrm{d}y_1}{\mathrm{d}x} \\ \vdots \\ \frac{\mathrm{d}y_N}{\mathrm{d}x} \\ \frac{\mathrm{d}f}{\mathrm{d}x} \end{matrix}} & \begin{matrix} 0 \\ I \\ \vdots \\ \frac{\mathrm{d}y_N}{\mathrm{d}y_1} \\ \frac{\mathrm{d}f}{\mathrm{d}y_1} \end{matrix} & \begin{matrix} \ldots \\ \ldots \\ \ddots \\ \ldots \\ \ldots \end{matrix} & \begin{matrix} 0 \\ \frac{\mathrm{d}y_1}{\mathrm{d}y_N} \\ \vdots \\ I \\ \frac{\mathrm{d}f}{\mathrm{d}y_N} \end{matrix} & \begin{matrix} 0 \\ 0 \\ \vdots \\ 0 \\ I \end{matrix} \end{bmatrix} = I = \begin{bmatrix} I & -\frac{\partial \mathcal{Y}_1}{\partial x}^T & \ldots & -\frac{\partial \mathcal{Y}_N}{\partial x}^T & -\frac{\partial \mathcal{F}}{\partial x}^T \\ 0 & I & \ldots & -\frac{\partial \mathcal{Y}_N}{\partial y_1}^T & -\frac{\partial \mathcal{F}}{\partial y_1}^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & -\frac{\partial \mathcal{Y}_1}{\partial y_N}^T & \ldots & I & -\frac{\partial \mathcal{F}}{\partial y_N}^T \\ 0 & 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} \begin{matrix} I \\ 0 \\ \vdots \\ 0 \\ 0 \end{matrix} & \begin{matrix} \frac{\mathrm{d}y_1}{\mathrm{d}x}^T \\ I \\ \vdots \\ \frac{\mathrm{d}y_1}{\mathrm{d}y_N}^T \\ 0 \end{matrix} & \begin{matrix} \ldots \\ \ldots \\ \ddots \\ \ldots \\ 0 \end{matrix} & \begin{matrix} \frac{\mathrm{d}y_N}{\mathrm{d}x}^T \\ \frac{\mathrm{d}y_N}{\mathrm{d}y_1}^T \\ \vdots \\ I \\ 0 \end{matrix} & \boxed{\begin{matrix} \frac{\mathrm{d}f}{\mathrm{d}x}^T \\ \frac{\mathrm{d}f}{\mathrm{d}y_1}^T \\ \vdots \\ \frac{\mathrm{d}f}{\mathrm{d}y_N}^T \\ I \end{matrix}} \end{bmatrix}$$

Coupled direct: functional form

$$\begin{bmatrix} I & \ldots & -\frac{\partial \mathcal{Y}_1}{\partial y_N} \\ \vdots & \ddots & \vdots \\ -\frac{\partial \mathcal{Y}_N}{\partial y_1} & \ldots & I \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}y_1}{\mathrm{d}x} \\ \vdots \\ \frac{\mathrm{d}y_N}{\mathrm{d}x} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{Y}_1}{\partial x} \\ \vdots \\ \frac{\partial \mathcal{Y}_N}{\partial x} \end{bmatrix}$$

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\partial \mathcal{F}}{\partial x} + \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial y_1} \ldots \frac{\partial \mathcal{F}}{\partial y_N} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}y_1}{\mathrm{d}x} \\ \vdots \\ \frac{\mathrm{d}y_N}{\mathrm{d}x} \end{bmatrix}$$

Coupled adjoint: functional form

$$\begin{bmatrix} I & \ldots & -\frac{\partial \mathcal{Y}_N}{\partial y_1}^T \\ \vdots & \ddots & \vdots \\ -\frac{\partial \mathcal{Y}_1}{\partial y_N}^T & \ldots & I \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}f}{\mathrm{d}y_1}^T \\ \vdots \\ \frac{\mathrm{d}f}{\mathrm{d}y_N}^T \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial y_1}^T \\ \vdots \\ \frac{\partial \mathcal{F}}{\partial y_N}^T \end{bmatrix}$$

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\partial \mathcal{F}}{\partial x} + \begin{bmatrix} \frac{\mathrm{d}f}{\mathrm{d}y_1} \ldots \frac{\mathrm{d}f}{\mathrm{d}y_N} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathcal{Y}_1}{\partial x} \\ \vdots \\ \frac{\partial \mathcal{Y}_N}{\partial x} \end{bmatrix}$$

Figure 4.6: Derivation of the functional form of the coupled derivative methods.

CHAPTER 5

A compact framework design

This chapter describes the algorithmic and implementation details of the computational modeling framework. The framework design is characterized as compact because the framework represents both solvers and user-defined components as instances of a general *System* class. Moreover, this *System* class has an interface consisting of only 4 primary operations—computing the residuals, driving them to zero, computing the Jacobian of the residuals, and applying the inverse of the Jacobian as an operator. This minimalistic design is enabled by the monolithic formulation from Ch. 3 and derivative unification from Ch. 4.

This chapter begins with the problem statement in Sec. 5.1, followed by the requirements in Sec. 5.2, the interface for the components in Sec. 5.3, the object-oriented framework design in Sec. 5.4, data handling in the framework in Sec. 5.5, and implementation details in Sec. 5.6.

5.1 Problem statement

The problem statement for the solution algorithms in the framework follows from the unified mathematical formulation. The benefit of a monolithic formulation is that the problem to be solved can be expressed in a simple manner that is independent of the particulars of a given application. The problem statement consists of 4 systems of equations that must be solved:

(i) The nonlinear system: $R(u) = 0$.

(ii) The Newton system: $\dfrac{\partial R}{\partial u} \Delta u = -r$.

(iii) The derivatives equation (forward): $\dfrac{\partial R}{\partial u} \dfrac{\mathrm{d}u}{\mathrm{d}r} = \mathcal{I}$.

(iv) The derivatives equation (reverse): $\dfrac{\partial R}{\partial u}^T \dfrac{\mathrm{d}u}{\mathrm{d}r}^T = \mathcal{I}$.

5.2 Solver-driven requirements

The algorithmic design of the framework is driven first and foremost by the numerical solution methods currently used to solve large nonlinear and linear systems. The objective is to ensure negligible computation-time and memory overhead for implementing an existing computational model in the framework. To this end, the current state-of-the-art nonlinear and linear solvers are reviewed here.

For large systems of nonlinear equations, Newton's method is the only feasible option in the absence of problem-specific methods such as multi-grid and pseudo-transient continuation (PTC). Any alternative that only uses residual evaluations and no Jacobian evaluations would have to evaluate the residuals at least n times and most likely significantly more, where n is the number of unknowns and equations. This is prohibitively expensive because n can often be on the order of millions.

Two exceptions are quasi-Newton methods and Jacobian-free Newton–Krylov methods. Quasi-Newton methods use the Broyden class of formulas [38] to approximate the inverse of the Jacobian during each iteration based on only residual information. These methods eliminate the need to solve a linear system or even compute any derivatives, but may not provide sufficient accuracy for efficient convergence of the nonlinear system. Jacobian-free Newton–Krylov (JFNK) methods, surveyed by Knoll and Keyes [39], take advantage of the fact that a Krylov iterative method applied to the linear system in Newton's method only requires the Jacobian as a linear operator. The Jacobian-vector products are computed using a directional finite-difference approximation, so JFNK methods operate with only residual evaluations.

An important consideration when applying Newton's method is the potential lack of robustness, which necessitates a globalization strategy. One option is PTC, which time marches the unsteady PDEs towards the steady solution when the nonlinear system is the discretized form of the steady PDEs [40]. A more general approach is selecting the size of the Newton step using a line search or trust region to try to improve robustness in early iterations. Another general approach is an initial start-up stage using a method such as nonlinear block Gauss–Seidel [41]. This is possible when the unknowns can be partitioned into several sets that can each be solved for on their own, which would make use of the hierarchical systems presented earlier. Another related consideration for Newton–Krylov methods is an inexact approach. To save computational time, the tolerance for the Krylov method can start low and increase as Newton's method approaches the solution and requires more accuracy [42].

When solving large systems of linear equations, iterative solvers that use the Krylov subspace are the most efficient both from a memory and a computational-cost standpoint. The matrix in a large linear system, especially one representing a discretized PDE, is typically very sparse, with much fewer non-zeros per row than the number of columns. Matrix factorizations that are computed by direct solvers for linear systems have three issues: the factorized forms are often much less sparse and hence less memory-efficient; these methods are not as efficient in parallel as iterative methods; and in a direct comparison, they are typically slower in general than iterative methods. In other situations, the matrix is only available as an operator that, given a vector, returns the matrix-vector product, because explicitly computing it is prohibitively expensive, even in a sparse format. For a review of iterative linear solvers, the reader is referred to Kelley [43].

Similar to Newton's method, Krylov subspace methods can be extremely efficient when they do converge, but they sometimes stall in ill-conditioned problems. In these situations, it is necessary to use a preconditioner, a second matrix that approximates the original one but can be cheaply inverted or solved given a right-hand side. One option for preconditioning is a problem-specific option such as multi-grid [44] or additive Schwarz [45]. Another is linear block Gauss–Seidel or Jacobi, analogous to the corresponding nonlinear methods for globalization.

The requirements for the algorithmic design of the framework are chosen so that the aforementioned solution methods can be supported without any significant inefficiencies. There are 3 categories of requirements:

1. General architecture
    (a) Types of Jacobians: pre-computed or matrix-free, dense or sparse
    (b) Distributed-memory parallel computation
    (c) Recursive solvers for hierarchically nested systems
    (d) Inexact solution methods

2. Nonlinear solvers

    (a) Newton's method with a start-up phase

    (b) Line search or trust region

    (c) Nonlinear block Gauss–Seidel or Jacobi

    (d) Custom solvers: e.g., PTC

3. Linear solvers

    (a) Preconditioned Krylov subspace method

    (b) Direct method

    (c) Linear block Gauss–Seidel or Jacobi

    (d) Custom solvers: e.g., incomplete factorization

5.3 A compact interface

Each component in the computational model must conform to an interface specified by the framework, consisting of a set of functions that the user must implement. A small number of functions is desired, but there must be a sufficient number to account for nearly any situation in which a solution algorithm needs to initiate an operation in the component.

Each component corresponds to an intermediate system that cannot be further decomposed or partitioned—it is found at the deepest level of the hierarchy tree. The framework design enables a small and simple interface that captures all operations required from an intermediate system. For a component that corresponds to intermediate system $R_S(p_S, u_S) = 0$, the interface consists of the following operations (note: in the following list, the subscript $_S$ is omitted for brevity):

1. *apply_nonlinear*: $(p, u) \mapsto r = R(p, u)$.

    This operation computes the residual functions for the current intermediate system. It is used to compute the right-hand side of the linear system in Newton's method and to check the convergence of the current intermediate system.

2. *solve_nonlinear*: $p \mapsto u$.

This operation solves $R(p, u) = 0$, inexactly or to the required convergence criterion. It is optional to implement this method since Newton's method can be used when the user does not provide a custom solver.

3. *apply_linear*: $\begin{cases} (dp, du) \mapsto dr = \dfrac{\partial R}{\partial p} dp + \dfrac{\partial R}{\partial u} du, & \text{forward mode} \\ dr \mapsto \left(dp = \dfrac{\partial R}{\partial p}^T dr, du = \dfrac{\partial R}{\partial u}^T dr \right), & \text{reverse mode.} \end{cases}$

   This operation allows the component to implicitly make $\partial R / \partial(p, u)$ known to the framework. By implementing it as a linear operator, the framework can be unaware of how each component stores or computes the Jacobian. There is a single, simple interface whether the Jacobian is computed in a sparse, dense, or matrix-free format. There are two modes since the forward and reverse modes for computing derivatives use $\partial R / \partial(p, u)$ and $[\partial R / \partial(p, u)]^T$, respectively.

4. *solve_linear*: $\begin{cases} (dr) \mapsto du \left| \dfrac{\partial R}{\partial u} du = dr \right. , & \text{forward mode} \\ (du) \mapsto dr \left| \dfrac{\partial R}{\partial u}^T dr = du \right. , & \text{reverse mode.} \end{cases}$

   This operation implements the inverse of $\partial R / \partial u$ as a linear operator. Equivalently, it solves a linear system with $\partial R / \partial u$ as the matrix and a given right-hand side vector. Like *solve_nonlinear*, it is optional since the framework's Krylov iterative solver can be used if *apply_linear* is known.

5. *linearize*:

   This operation is present in the interface because many problems require an initial assembly and, in some cases, factorization of the Jacobian, and repeated applications of the matrix or factorization have a substantially lower cost.

5.4 Object-oriented framework design

The framework design is greatly simplified by an object-oriented approach. Within this paradigm, each intermediate system in the numerical model corresponds to an

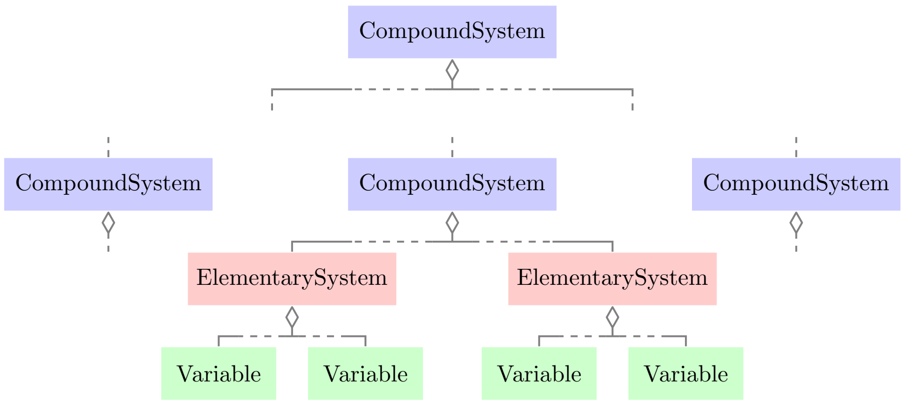


Figure 5.1: A class diagram showing the containment relationships between objects in a computational model implemented in the framework.

instance of the *System* class in the computational model, and the five operations listed above are virtual methods of this class.

Figure 5.1 contains a tree representing the hierarchical decomposition of the variables in the numerical model. Excluding the *Variable* nodes, the leaves of the tree are *ElementarySystem* objects, which implement the computational model's components. Each component 'owns' a subset of the variables; these variables form the unknown vector in the component's intermediate system. During initialization, the *ElementarySystem* object must declare its variables as well as its arguments—external variables upon which the 'owned' variable may depend. The *ElementarySystem* objects are grouped together by *CompoundSystem* objects, which can in turn be grouped by other *CompoundSystem* objects.

The *ElementarySystem* and *CompoundSystem* classes inherit from a base *System* class. The *ElementarySystem* class has 3 derived classes, reflecting whether the system contains independent, explicitly-defined, or implicitly-defined variables. The classes implementing user-defined components directly inherit from one of these three classes.

The *CompoundSystem* class has 2 derived classes that handle parallelism in different ways. In the hierarchy tree in Fig. 5.1, the root *CompoundSystem* is stored and run on all processors running the job. If it is a *SerialSystem*, it passes all of its processors to the *System* objects it contains and runs its recursive operations sequentially among the contained systems. If it is a *ParallelSystem*, it partitions its group of processors among the *System* objects it contains and runs its recursive operations concurrently among the contained systems. The same applies to all *CompoundSystem* objects, and in this manner, each *System* object is assigned a subset of or all of

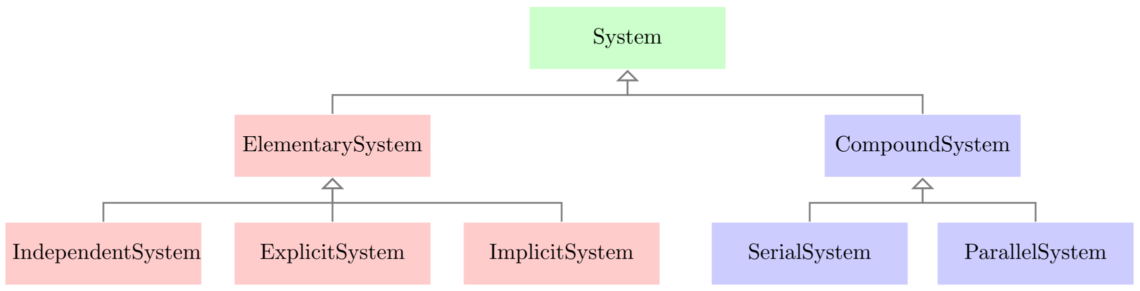


Figure 5.2: A class diagram showing the inheritance relationships between the *System* classes.

its containing *System* object's processors. A class inheritance diagram is shown in Fig. 5.2 for the various system classes.

This object-oriented design allows all of the framework's built-in solvers to be implemented as methods of one of the base system classes. Table 5.1 shows the class in which each built-in solver is implemented and pseudocode for the algorithms is shown in Figs. 5.3, 5.4, and 5.5.

Table 5.1: The framework's 4 operations and their implementations in each type of *System* class.

		System classes		
		System	*CompoundSystem*	*ElementarySystem*
Virtual methods	*apply_nonlinear*	—	Recursive	User-implemented
	apply_linear		Recursive	User-implemented or FD*
	solve_nonlinear	Newton with line search	Nonlinear block Gauss–Seidel/Jacobi	Optional
	solve_linear	Krylov-type with preconditioning	Linear block Gauss–Seidel/Jacobi	Optional

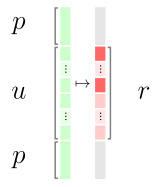
*FD: finite-difference approximation of the Jacobian.

5.5 Efficient data management

Efficient data storage, accessing, and transfer are important because the framework is designed to handle problems with large data sizes. Specifically, there are 3 important challenges: avoiding unnecessary memory overhead in data storage, simple data access for the user, and efficient parallel data transfer.

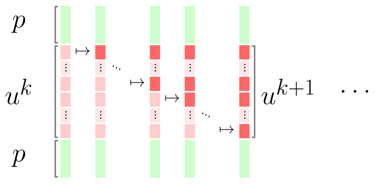
Algorithm 1. *apply_nonlinear* [recurs.]

```
input  : $(p, u)$
output: $r$
scatter $u$ to each $subsys.p$
for each subsys do
    $subsys.apply\_nonlinear$
end
```



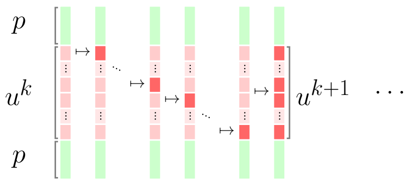
Algorithm 2. *solve_nonlinear* [GS]

```
input  : $p$
output: $u$
while not converged do
    for each subsys do
        scatter $u$ to $subsys.p$
        $subsys.solve\_nonlinear$
    end
end
```



Algorithm 3. *solve_nonlinear* [Jacobi]

```
input  : $p$
output: $u$
while not converged do
    scatter $u$ to each $subsys.p$
    for each subsys do
        $subsys.solve\_nonlinear$
    end
end
```



Algorithm 4. *solve_nonlinear* [Newton]

```
input  : $p$
output: $u$
while not converged do
    $apply\_nonlinear$
    $df \leftarrow -f$
    $solve\_linear$
    $\alpha \leftarrow line\_search(du)$
    $u \leftarrow u + \alpha \cdot du$
end
```

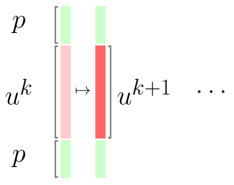
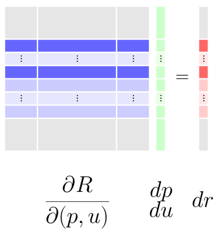


Figure 5.3: Operations for the nonlinear system.

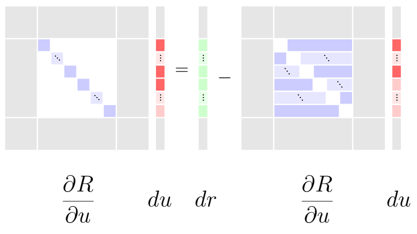
5.5.1 Data storage

For each system, six vectors must be stored: u, p, and r for the nonlinear problem and du, dp, and dr for the linear problem. The latter three can be interpreted as buffers that contain the data for the solution vector or the right-hand side vector, depending on the situation. Among these six vectors, u, du, r, and dr are instances of the *UnknownVec* class while p and dp are instances of the *ParameterVec* class.

```
ALGORITHM 1. $apply\_linear$ [recurs.]
input  : $(dp, du)$
output: $dr$
scatter $du$ to each $subsys.dp$
for each $subsys$ do
    $subsys.apply\_linear$
end
```

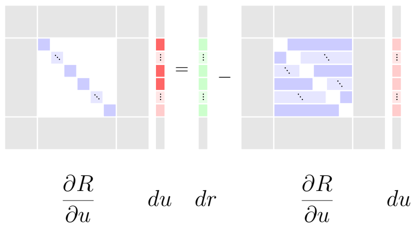


```
ALGORITHM 2. $solve\_linear$ [GS]
input  : $dr$
output: $du$
$rhs \longleftarrow dr$
while $not\ converged$ do
    for $each\ subsys$ do
        scatter $du$ to $subsys.dp$
        $subsys.apply\_linear$
        $subsys.dr \leftarrow rhs - subsys.dr$
        $subsys.solve\_linear$
    end
end
```



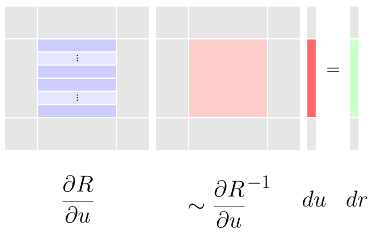
$\frac{\partial R}{\partial u}$
du
dr
=
−
$\frac{\partial R}{\partial u}$
du

```
ALGORITHM 3. $solve\_linear$ [Jacobi]
input  : $dr$
output: $du$
$rhs \longleftarrow dr$
while $not\ converged$ do
    scatter $du$ to each $subsys.dp$
    for $each\ subsys$ do
        $subsys.apply\_linear$
        $subsys.dr \leftarrow rhs - subsys.dr$
        $subsys.solve\_linear$
    end
end
```



$\frac{\partial R}{\partial u}$
du
dr
=
−
$\frac{\partial R}{\partial u}$
du

```
ALGORITHM 4. $solve\_linear$ [Krylov]
input  : $dr$
output: $du$
$rhs \leftarrow dr$
function linear_operator($x$)
    $dr \leftarrow x$
    $solve\_linear$
    $apply\_linear$
    $y \leftarrow dr$
    return $y$
$du \leftarrow krylov(rhs,$ linear_operator$)$
```



$\frac{\partial R}{\partial u}$
$\sim \frac{\partial R}{\partial u}^{-1}$
du
dr
=

Figure 5.4: Operations for the linear system.

For the *UnknownVec* instances, data is shared with contained or containing systems; that is, the full u, du, r, and dr vectors are allocated in the top-level *Com-*

poundSystem and all other systems store pointers onto sub-vectors of the global vector. Compared to allocating separate vectors in each system, this approach saves memory, but also computation time since a system's subsystems operate directly on sub-vectors of the larger system's vector.

The *ParameterVec* instances store only the variables that an *ElementarySystem* declares as its arguments and only the declared components of those variables as well. There are two reasons why it is necessary to explicitly store a separate copy of the arguments of each *ElementarySystem* as opposed to simply reading the data as needed. First, the originating data could be stored on a different processor so a buffer is needed to perform the parallel data communication. Second, the linear and nonlinear block Jacobi methods require storing a second copy of arguments to store their values from the previous Jacobi iteration. Since the framework allows for the use of block Jacobi at possibly multiple levels in the hierarchy tree, the intricacies of updating the arguments at the right times is simplified by incorporating the data transfers into the Jacobi algorithm.

5.5.2 Data accessing

The framework stores data in large vectors that concatenate multiple variables to allow operations on larger pieces of data simultaneously, which improves efficiency. However, the sub-vector corresponding to a given variable is explicitly stored with a pointer to a portion of the larger vector. This is done to allow the user to work with the variable as if it is its own vector and not to have to keep track of the global indices of the variable's components in the larger, concatenated vector. Moreover, in the Python implementation of the framework, the user accesses variables through dictionary objects whose keys and values are the individual variables' string names and data, respectively, to improve usability.

5.5.3 Data transfer

The framework automates parallel data transfer between *UnknownVec* instances and *ParameterVec* instances, providing two benefits. First, if component A depends on variable V from component B, component A does not have to know how many processors component B has and how much of variable V component B has stored on each processor. For each argument an *ElementarySystem* declares, the framework automatically determines on which processor the requested data is stored and what the local indices are, based on the global indices that were also declared. Second, the data transfer operation is implemented as a method in the *CompoundSystem* class, so it is integrated into solution algorithms. This operation performs data transfers for multiple *ElementarySystem* objects simultaneously, improving parallel performance. A given system's transfer operation transfers data to a given subsystem's *ParameterVec* instance from its remaining subsystems' *UnknownVec* instances, and vise versa for the reverse mode when solving a transposed linear system.

5.6 Implementation

The framework has been implemented using the Python programming language. It depends only the *NumPy* package for handling local vectors and the *petsc4py* package to use the portable, extensible toolkit for scientific computation (PETSc) [46], which in turn uses the message-passing interface to parallel communication. PETSc is used for all parallel data transfers, and its Krylov iterative methods are used as the framework's linear solvers as well with flexible generalized minimal residual (fGMRES) as the default solver. The entire implementation of the framework is contained in a single Python file with about 1,000 lines of code thanks to the framework's monolithic mathematical formulation and the use of PETSc.

The scaling of the framework is an important consideration especially for large problems. There is a concern that the framework adds significant overhead which increases at an unmanageable rate as the problem size increases. Figure 5.7 shows how the execution time scales with the number of total unknowns for the nanosatellite and allocation-mission optimization problems. It is difficult to separate the overhead introduced by the framework from the cost of executing the computational models. However, at least for the nanosatellite problem, what Fig. 5.7 shows is that the framework overhead is likely not significant, and if it is, it scales better than linearly with the total number of unknowns.

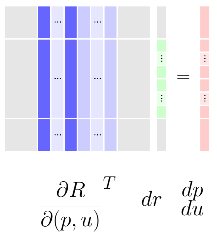
5.7 Summary

In summary, the framework automatically solves coupled systems and computes their derivatives simply by solving a nonlinear system and a linear system involving the Jacobian of the residuals or its transpose. The framework's nonlinear solvers are Newton's method with a line search, inexact nonlinear block Gauss–Seidel or Jacobi, or any problem-specific user-provided solver. The linear solvers are a Krylov iterative method with variable preconditioning, inexact linear block Gauss–Seidel or Jacobi, or any problem-specific user-provided solver. These nonlinear and linear solvers can be executed at any level of a hierarchical decomposition of the system of equations. This chapter also described the object-oriented framework design, which centers on a *System* class with an interface consisting of only five required methods used to solve the nonlinear and linear systems. User-defined components that 'own' a subset of the variables are *System* objects, as are framework-provided *System* objects that group other *System* objects together. In the interface, components provide matrices only as linear operators, and the framework centrally stores and operates on all vectors in a concatenated form, making pointers to sub-vectors available to components for easy access.

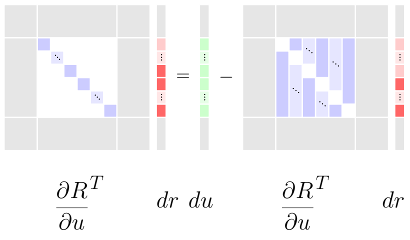
The framework's key features can be summarized as follows. First, the framework's parallel data passing greatly facilitates distributed-memory parallel computation. The framework allows components that have parallel vectors as arguments to not be aware of how those parallel vectors are distributed across their processors. Second, the framework automatically solves the nonlinear and linear systems that arise. When the user provides a custom solver, the framework uses it; otherwise, it uses its built-in solvers. The third benefit is the automated computation of derivatives given partial derivatives of each component. It uses the analytic methods for computing derivatives, which can compute a full gradient with respect to all inputs at a cost on the same order of magnitude as that of running the simulation.

The framework can be applied to any problem that deterministically computes a set of variables as a continuous function of others. It provides significantly more value when the variables are also differentiable, since the automatic derivative computation is a key feature. The derivative computation provides a significant increase in efficiency when there are implicit state variables and when there is a feedback loop among the dependencies between variables.

```
ALGORITHM 1. apply_linear [recurs.]
input  : $dr$
output: $(dp, du)$
for each subsys do
    subsys.apply_linear
end
scatter each subsys.dp to du
```

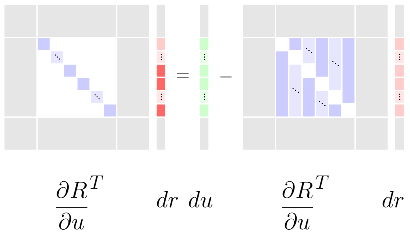


```
ALGORITHM 2. solve_linear [GS]
input  : $du$
output: $dr$
$rhs \longleftarrow du$
while not converged do
    for each subsys1 do
        for each subsys2 do
            subsys2.apply_linear
        end
        scatter each subsys2.dp to du
        $subsys1.du \leftarrow$
        $rhs - subsys1.du$
        subsys1.solve_linear
    end
end
```



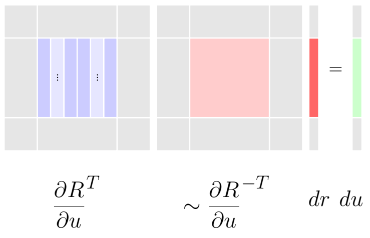
$\frac{\partial R^T}{\partial u}$
dr
du
$\frac{\partial R^T}{\partial u}$
dr

```
ALGORITHM 3. solve_linear [Jacobi]
input  : $du$
output: $dr$
$rhs \longleftarrow du$
while not converged do
    for each subsys do
        subsys.apply_linear
    end
    scatter each subsys.dp to du
    for each subsys do
        $subsys1.du \leftarrow$
        $rhs - subsys.du$
        subsys.solve_linear
    end
end
```



$\frac{\partial R^T}{\partial u}$
dr
du
$\frac{\partial R^T}{\partial u}$
dr

```
ALGORITHM 4. solve_linear [Krylov]
input  : $du$
output: $dr$
$rhs \leftarrow du$
function linear_operator($x$)
    $du \leftarrow x$
    solve_linear
    apply_linear
    $y \leftarrow du$
    return $y$
$dr \leftarrow krylov(rhs,$ linear_operator$)$
```



$\frac{\partial R^T}{\partial u}$
$\sim \frac{\partial R^{-T}}{\partial u}$
dr
du

Figure 5.5: Operations for the transposed linear system.

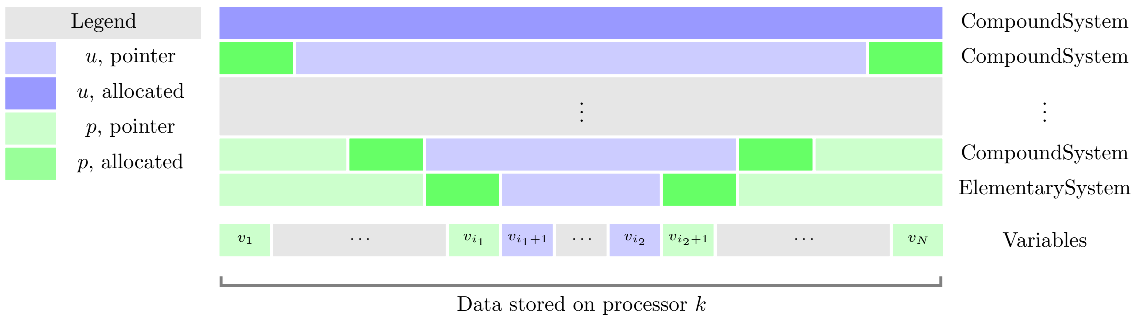


Figure 5.6: The type of data storage for each variable in each *System* instance.

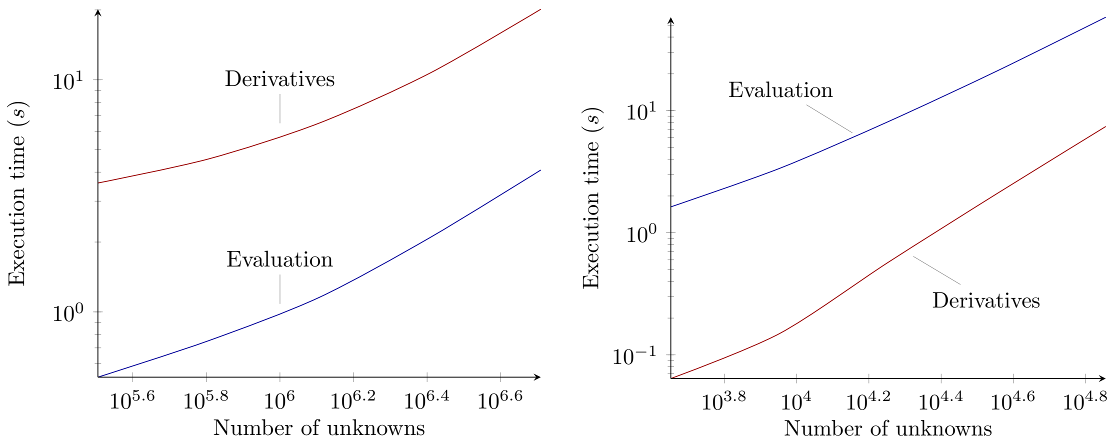


Figure 5.7: Runtime results for the nanosatellite (left) and aircraft allocation-mission (right) optimization problems implemented in the framework. The blue curve represents a single execution of the computational model, and the red curve represents the solution of a linear system for the adjoint method.

CHAPTER 6

Multidisciplinary optimization of a nanosatellite

In this chapter, I present the first of two applications implemented in the framework, the optimization of a nanosatellite. Section 6.1 provides background on nanosatellites and prior work on optimization in this field, and Sec. 6.2 presents the design problem. Section 6.3 and Section 6.4 describe the methodology and the computational models for each discipline, respectively. Finally, Sec. 6.5 present the results from several optimizations that try to quantify the effectiveness of optimization and compare the outcomes from different launch possibilities.

6.1 Optimization in nanosatellite design

Satellites serve a multitude of purposes that range from navigation and scientific research to military applications. Over the past decade, small satellites have gained increasing interest as alternatives to larger satellites because of the low time and cost required to manufacture and launch them. In particular, the CubeSat class of small satellites is becoming a common platform for education and research because it has a set of specifications that facilitates relatively frequent launches as secondary payloads.

CADRE (CubeSat investigating atmospheric density response to extreme driving) is funded by the National Science Foundation and will study the response of the Earth's upper atmosphere to auroral energy inputs [47]. This mission addresses the need for more accurate modeling of space weather effects, motivated in part by the growth of the global space-based infrastructure. To help answer some of the important scientific questions in this area, CADRE will provide critical *in situ* measurements in the ionospheric and thermospheric regions.

CADRE will inherit much of the design of the University of Michigan's Radio Aurora eXplorer (RAX) CubeSat. However, the unique scientific goals of the mission necessitate a detailed design study. Power is a driving factor because the scientific instruments are to run continuously, and large amounts of data must be transmitted to ground stations. Fortunately, there are several geometric and operational design variables whose impact can be captured with relatively inexpensive computational models, and it is possible to use these variables to satisfy the mission requirements while improving the satellite's performance. In the past, this has mostly been done via experience and human intuition aided by computational design tools that work with a relatively small number of design variables.

In the literature, there are many studies in which computational modeling and optimization have been applied to satellite design. For instance, Boudjemai et al. [48] performed topology optimization on the structure of a small satellite using NASTRAN for the finite element analysis. Numerical optimization has been applied to several other disciplines as well. Galski et al. [49] optimized a thermal control system, while Jain and Simon [50] implemented real-time load scheduling optimization of a small satellite's batteries. More recently, Richie et al. [51] and Zhang et al. [52] used optimization to size the energy storage and attitude control system and to design the layout of the satellite's components, respectively. All of these single-discipline optimization studies share a common approach: with the exception of the actuator sizing optimization, they use a genetic algorithm (GA) as a simple solution to deal with the discrete design variables and the discontinuities that are often present in the models.

Other authors considered multiple disciplines simultaneously to better model the overall physical problem. Barnhart et al. [53] implemented SPIDR, a systems-engineering-based framework for satellite design with an artificial-intelligence-based optimization algorithm that incorporates user-defined rules and constraints. Fukunaga et al. [54] developed OASIS, which uses a machine-learning algorithm to adaptively select and configure a metaheuristic optimizer such as a GA to optimize a model in MIDAS [55], a satellite design framework. SCOUT [56] is another framework that uses a GA for optimization, and ATSV [57] uses a shopping paradigm to aid the design process. Recently, Ebrahimi et al. [58] and Jafarsalehi et al. [59] developed multidisciplinary design frameworks that use a particle swarm optimizer (PSO) and a GA, respectively.

With the exception of the last two efforts, all of the computational design tools cited above have graphical user interfaces (GUI) that significantly enhance usability. For these tools, the approach is to make user interaction with the framework as streamlined as possible, allowing the user's knowledge and experience to work together with the framework's optimization capability. However, as was the case with the single-discipline studies, all of these computational design tools use optimizers or design techniques that do not use gradients, which limits the number of design variables that can be considered. Without gradients, algorithms must rely on sampling the design space at a cost that grows exponentially with the number of design variables, and in practice, this becomes prohibitive when there are more than $\mathcal{O}(10)$ variables. Wu et al. [60] used a gradient-based approach to solve a satellite MDO problem with collaborative optimization (CO) [61, 62], but the cost of computing coupled derivatives limited the number of design variables to $\mathcal{O}(10)$ here as well.

Given the existing body of work, this chapter seeks to address the question whether multidisciplinary design optimization (MDO) can handle the full set of design variables in the satellite design problem simultaneously, even when there are tens of thousands of them. The high-level approach is gradient-based optimization in combination with adjoint-based derivative computation, with a modular implementation of the disciplinary models in an integrated framework. The full small-satellite design problem is simultaneously considered, including all major disciplines, multiple time scales, and tens of thousands of design variables that parametrize the variation of several quantities over time.

6.2 The design problem

CADRE is a 3U CubeSat [47], meaning its body is a square prism with dimensions of 30 cm $\times$ 10 cm $\times$ 10 cm. As with other CubeSats, CADRE's dimensions are fixed so that it can be launched as a secondary payload with a larger satellite in order to reduce costs. The satellite has four fins that are initially folded at the sides of the satellite but are permanently deployed after launch in the rear direction to a preset angle. Although the roll angle is flexible, CADRE must always be forward-facing because of the scientific requirements, so the swept-back fins provide passive attitude stabilization through aerodynamic drag. CADRE has 12 solar panels with 7 cells each: 4 panels on the sides of the body, and one on the front and back of each fin. In general, the 84 cells are only partially illuminated because the Earth or another part of the satellite can cast shadows even when a cell is facing the Sun. Since the cells cover most of the satellite, it may be beneficial to install a radiator in place of one or more of the solar cells that are often shaded to provide cooling and to improve the power generation of other cells. A rendering of the CADRE CubeSat is shown in Fig. 6.1.

Other relevant subsystems include energy storage, communication, and attitude

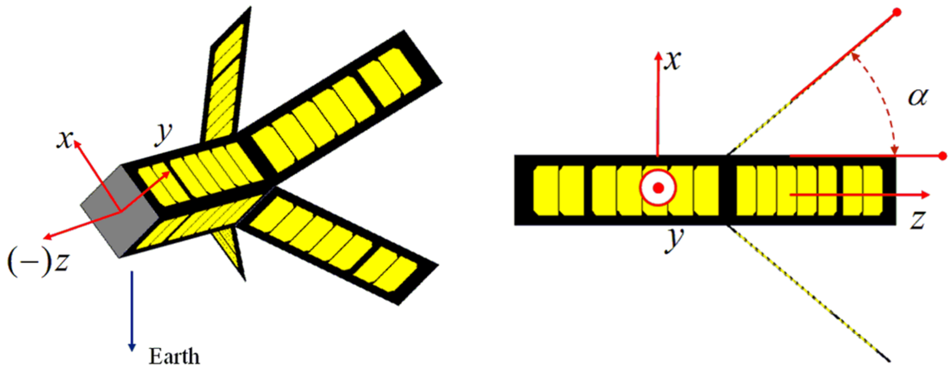


Figure 6.1: CADRE CubeSat geometry.

control. Lithium-ion batteries are installed with charge- and discharge-rate constraints, and a depth-of-discharge limit of 20% is enforced to lengthen the battery life. To transmit data to ground stations, an antenna is installed toward the rear of the satellite, and the installation angle is a parameter that can be varied, although it is constrained to be in the vertical plane. For the current purposes, data transmission to ground stations is assumed to use a UHF antenna, although the final design for CADRE may use an S-band antenna for high-speed data download. CADRE uses two types of attitude-control actuators that complement each other: magnetorquers for gross changes, and reaction wheels for more precise control. With the latter, there is potential for the rotation rates of the wheels to accumulate and grow unmanageably large, so the magnetorquers are used to counteract constant torques such as that due to solar pressure. In this problem, only the reaction wheels are modeled to capture the power requirements of the desired attitude profiles, and the cost of counteracting disturbance torques is modeled as a constant background power consumption. Fig. 6.2 shows the disciplines and how they are coupled through the state variables.

CADRE's mission is to continuously collect data and transmit as much of that data as possible to the ground stations. Therefore, the total data downloaded is the natural objective function for the CADRE design optimization problem, although generating and storing sufficient energy is a driving factor. The fin and antenna angles are important geometric design variables, because they affect the power generation and the data-transmission rate, respectively. CADRE's attitude profile over time can be designed as well, providing further flexibility that can be used to increase power generation, cool panels when necessary, and increase transmission gain during communication with a ground station. The attitude profile must be optimized simultaneously with the geometric design variables because the fin and antenna angles

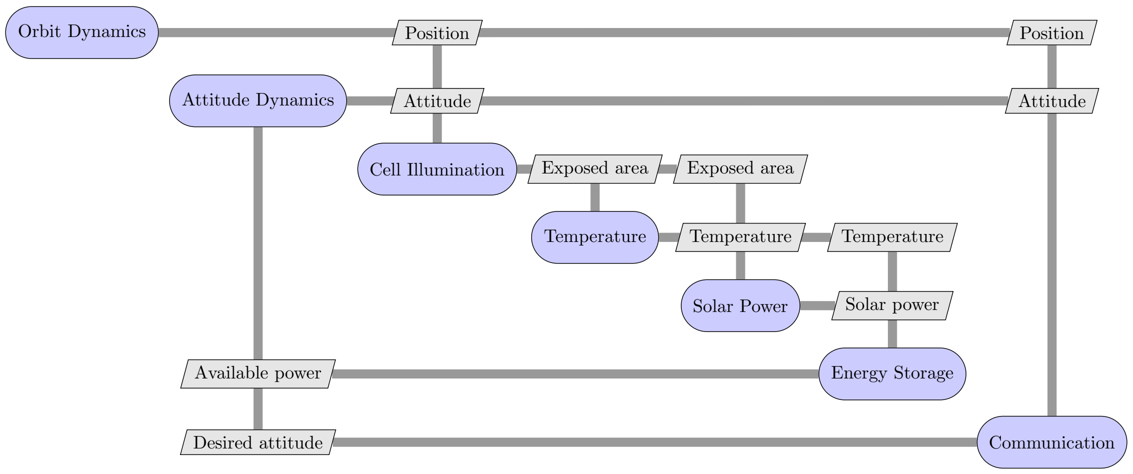


Figure 6.2: Extended design structure matrix (XDSM) diagram [1] showing all relevant disciplines in the CADRE design problem.

that are optimal for an assumed attitude profile may no longer be optimal for an attitude profile that is optimized separately. The available power must also be optimally distributed between communication and actuation, so the power-distribution profile must be considered simultaneously as well.

Optimizing these profile variables involves manipulating 2-D curves without any *a priori* knowledge of their final optimal shapes. To do this, the curves must be discretized and parametrized, and in the simulation of hours, days, or even months of the satellite's operation, the resulting number of design variables can easily reach tens of thousands. To summarize, the objective of the CADRE design problem is to maximize the total data downloaded subject to constraints on the power and energy available, with respect to the fin angle, the antenna angle, the attitude profile, the communication power profile, and the 84 binary cell-installation variables.

6.3 Approach

The advantage of the gradient-based MDO approach taken in this problem is that it can handle a problem with many disciplines, design variables, and state variables. As a result, the true design problem can be optimized with few simplifications. This section discusses the approach used in this problem by listing each of the technical challenges and their solutions.

6.3.1 Multiple time scales: multi-point optimization

The CADRE design problem involves multiple time scales. Capturing CADRE's power generation and temperature fluctuations requires a time-resolution of $\mathcal{O}(5\,\text{min})$ because its orbit has a period of roughly 90 min. Ground-station passes last $\mathcal{O}(10\,\text{min})$ and energy must be stored between sets of ground-station passes, which occur in patterns that roughly repeat each day. Assuming that the ground stations are close to the Equator, this requires a resolution of $\mathcal{O}(1\,\text{min})$ with a simulation of at least 12 h of the satellite's operation. However, depending on the launch orbit, CADRE's orbit may precess multiple times per year. This, combined with the effect of the seasons, requires a simulation of one year to model one period of oscillations in the satellite's operating conditions.

This multi-scale characteristic, combined with the ambitious scope of the design problem, presents a significant challenge. For a truly unbiased simulation, the year must be simulated with a resolution of 1 min, yielding $262,800$ discrete points. If a shorter period of time is simulated, the resulting design may be optimal in one season but not others. In some seasons the satellite may have difficulty generating sufficient power because the solar cells see much less of the Sun at the chosen fin angle.

The periodic nature of many of the variables suggests a frequency-domain approach for the state and design variables to reduce the size of the model and the optimization problems. Such an approach would capture the oscillatory behavior of the variables with a relatively small number of degrees of freedom. However, we did not adopt this approach for two reasons. First, many of the state variables, such as the Sun line-of-sight variable, have near-discontinuous jumps that cannot be accurately represented with a small number of frequencies. These effects propagate to other variables, such as the temperature, solar power, and battery current, as similar discontinuities or as nonsmoothness. Second, while other state variables do behave smoothly for the most part, they tend to have one or more spikes due to ground-station passes. The transmitter gain, battery current, and temperature are examples of variables that exhibit such spikes, although for temperature these are on a relatively small scale. These high-frequency components could potentially be represented by additional modes, but they would require *a priori* knowledge of where the passes are, and some of the automation in the computational tool would be lost. Nonetheless, it is worth noting for future work that representing some of the state variables in the frequency domain and others in the time domain would no doubt reduce the size of the problem.

Our solution is to simulate six 12-hour blocks with 0.5 min resolution, distribute them uniformly over the year, and weight each one equally in a single optimization problem. The orbit and communication time scales are captured within the 12-hour blocks, and simulating half a day every two months captures the orbit-precession time scale. The optimization constraints are applied separately to each block, and the objective functions computed from the six blocks are averaged. This approach is essentially multi-scale, multi-point optimization; the minute-level time scale is directly simulated, and the objective function for the month-level time scale is numerically integrated using the midpoint rule.

6.3.2 Large number of constraints: constraint aggregation

As previously mentioned, the coupled-derivative equations compute the gradients efficiently because they give either the full vector of derivatives with respect to a single variable (forward mode of (3.24)) or the full gradient of a variable (reverse mode of (3.24)) using a single solution of a linear system. Since our optimization has a large number of design variables, the reverse mode must be used, but it requires a linear solution for each constraint. Moreover, the battery discipline requires four inequality constraints at each time instance—maximum charge rate, maximum discharge rate, minimum state of charge, and maximum state of charge—resulting in tens of thousands of constraints.

We use constraint aggregation to reduce the number of constraints. The Kreisselmeier–Steinhauser (KS) function [63] aggregates the constraints over all the time instances into a single criterion. Constraint aggregation with KS functions has been shown to work well in combination with the adjoint method in optimization problems, for instance in structural weight minimization [64] and aerostructural optimization [7, 65]. The KS function is given by

$$KS(x) = f_{i_{\max}}(x) + \frac{1}{\rho} \ln \sum_i e^{\rho(f_i(x) - f_{i_{\max}}(x))}, \tag{6.1}$$

where f_i is the i^{th} function in the vector of functions we wish to aggregate, i_{max} is the index of the function with the largest value at x, and ρ is a parameter that is problem-dependent. In the limit, as ρ approaches infinity, the KS function approaches the maximum function because $e^{\rho \cdot 0}$ dominates in the sum, and $KS(x)$ approaches $f_{i_{\max}}(x)$. For finite ρ, the KS function is a smooth function that is dominated by the f_i with the largest values. Thus, as an inequality constraint, the KS function encourages the optimizer to resolve the largest infeasibilities first and eventually choose a point at which the KS function itself is less than or equal to zero. The optimization problems solved in this chapter use $\rho = 50$, a value which was found through numerical tests.

6.3.3 Nondifferentiable models: B-spline interpolant

Often, a discipline has a model that cannot be differentiated. The reason could be that the underlying physical phenomenon is nonsmooth, the computational model is a legacy code without source code access, or only a table of data is available. To address these situations, we implemented in Fortran a tensor-product B-spline interpolant with analytic derivatives. A model with any number of input variables can be fitted with this interpolant given a structured array of data that spans the full range of values for the input variables.

6.3.4 Derivatives of ODE variables: modular Runge–Kutta solver

Several of the disciplines in the CADRE design problem involve ordinary differential equations (ODEs), which complicates the task of computing partial and total derivatives. In particular, ODEs in time have a natural forward direction, so the unknown variable depends only on those before it in time, but the reverse mode of (3.24) must compute the total derivatives in the opposite direction. This is not possible if the values from previous time instances are discarded as the algorithm moves forward, so the CADRE MDO algorithm explicitly keeps track of the full time series as a vector and operates on the entries of this vector in sequence. Furthermore, the fourth-order Runge–Kutta method (RK4) has been implemented in Fortran as a modular solver with the time-marching scheme differentiated. For each discipline that uses this modular RK4 solver, only the derivatives of the ODE must be provided; correct indexing and application of the chain rule to combine these with the partial derivatives of the RK4 equations are automatically handled.

6.4 Discipline models

This section describes the models for all the disciplines in the CADRE MDO algorithm. For vectors, the nomenclature used in this section is as follows. Upper-case subscripts represent the frames of reference: B, R, E, and I represent the body-fixed frame, rolled body-fixed frame (explained later), Earth-fixed frame, and Earth-centered inertial (ECI) frame, respectively. Lower-case subscripts represent the origins of frames: b, e, g, and s denote the body (satellite), Earth, ground station, and Sun, respectively. For instance, $\vec{\boldsymbol{r}}_{b/e}$ signifies a vector pointing from the Earth's origin to the satellite's origin. The axes of the body-fixed frame are denoted $\hat{\boldsymbol{\imath}}_B$, $\hat{\boldsymbol{\jmath}}_B$, and $\hat{\boldsymbol{k}}_B$. The orientation matrices are represented by $\boldsymbol{O}$, e.g., $\boldsymbol{O}_{B/I}$ represents the orientation of the body-fixed frame as seen in the ECI frame.

6.4.1 Orbit dynamics

The orbit-dynamics discipline computes the Earth-to-body position vector in the ECI frame. In the orbit equation, there are terms that represent the fact that the Earth is not a perfectly spherical and homogeneous mass. These are captured in the J_2, J_3, and J_4 coefficients in the following equation:[1]:

$$\begin{aligned}\ddot{\vec{\boldsymbol{r}}} = -\frac{\mu}{r^3}\vec{\boldsymbol{r}} - \frac{3\mu J_2 R_e^2}{2r^5}&\left[\left(1-\frac{5r_z^2}{r^2}\right)\vec{\boldsymbol{r}} + 2r_z\hat{\boldsymbol{z}}\right] \\ -\frac{5\mu J_3 R_e^3}{2r^7}&\left[\left(3r_z-\frac{7r_z^3}{r^2}\right)\vec{\boldsymbol{r}} + \left(3r_z-\frac{3r^2}{5r_z}\right)r_z\hat{\boldsymbol{z}}\right] \\ +\frac{15\mu J_4 R_e^4}{8r^7}&\left[\left(1-\frac{14r_z^2}{r^2}+\frac{21r_z^4}{r^4}\right)\vec{\boldsymbol{r}} + \left(4-\frac{28r_z^2}{3r^2}\right)r_z\hat{\boldsymbol{z}}\right]. \end{aligned} \tag{6.2}$$

The values of the coefficients are listed in Table 6.1.

The J_2, J_3, and J_4 terms must be considered because their effect is to rotate the orbit plane on a scale of months. If they are ignored, a fin angle that may initially increase power generation may no longer be optimal after the orbit plane has changed. This also makes the CADRE design problem a multi-scale problem in time because much of the system's behavior occurs on the scale of minutes and hours, since the period of the satellite's orbit is roughly 90 minutes. The slow rotation of the orbit plane affects the power generation and communication as well, since the satellite's trajectory affects how much data can be transmitted as it passes over ground stations. We solve the orbit equation using the modular RK4 solver described earlier.

6.4.2 Attitude dynamics

Because of the requirements for scientific data collection, CADRE must always have a forward-facing orientation. The roll angle, γ, can change provided the maximum rate of 1 rad/min is not exceeded. The optimizer controls the roll-angle profile over time, and all the other attitudes, torques, and related quantities are computed from this. Since all the time instances are modeled simultaneously, this approach is equivalent to determining the control inputs using optimization instead of a controller.

[1]Eagle, C. David, *Orbital Mechanics with MATLAB.*
Accessed February 2013. *http://www.cdeagle.com/ommatlab/toolbox.pdf*

At any given time instance, CADRE's attitude is determined by applying the rotations from the ECI frame to what is referred to here as the *rolled* frame and then to the actual body-fixed frame. The rolled frame is an intermediate frame obtained after ensuring that CADRE is forward-facing but prior to applying the appropriate rotation from the specified roll-angle profile. For this frame, $\hat{\boldsymbol{k}}_B$ must point in the opposite direction to $\hat{\boldsymbol{v}}_{b/e}$, and the chosen convention is that $\hat{\boldsymbol{j}}_B$ is parallel to $\hat{\boldsymbol{r}}_{b/e}$. The orientation matrices that implement these two successive rotations are given by

$$\boldsymbol{O}_{R/I} = \begin{bmatrix} \hat{\boldsymbol{i}}_B^T \\ \hat{\boldsymbol{j}}_B^T \\ \hat{\boldsymbol{k}}_B^T \end{bmatrix} = \begin{bmatrix} -(\hat{\boldsymbol{r}}_{b/e} \times \hat{\boldsymbol{v}}_{b/e})^T \\ \hat{\boldsymbol{r}}_{b/e}^T \\ -\hat{\boldsymbol{v}}_{b/e}^T \end{bmatrix} \quad \text{and} \quad \boldsymbol{O}_{B/R} = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.3}$$

Once the $\boldsymbol{O}_{B/I}$ matrix is known for all time instances, its time derivative can be computed using finite differences, and the angular-velocity vector can be computed using $\vec{\boldsymbol{\omega}}_B^{\times} = \dot{\boldsymbol{O}}_{B/I} \cdot \boldsymbol{O}_{B/I}^T$.

As mentioned previously, we model only the reaction wheel for actuation. The required inputs are computed from the satellite's angular-velocity profile. We do this by applying conservation of angular momentum to the satellite and reaction-wheel system, expressed by setting the time derivative of the total angular momentum to zero:

$$\dot{\vec{\boldsymbol{L}}} = \underbrace{\boldsymbol{J}_B \cdot \dot{\vec{\boldsymbol{\omega}}}_B + \vec{\boldsymbol{\omega}}_B \times (\boldsymbol{J}_B \cdot \vec{\boldsymbol{\omega}}_B)}_{\vec{\boldsymbol{\tau}}_B} + \underbrace{\boldsymbol{J}_{RW} \cdot \dot{\vec{\boldsymbol{\omega}}}_{RW}}_{\vec{\boldsymbol{\tau}}_{RW}} + \vec{\boldsymbol{\omega}}_B \times (\boldsymbol{J}_{RW} \cdot \vec{\boldsymbol{\omega}}_{RW}) = 0. \tag{6.4}$$

Computing the required reaction-wheel torque is a three-step process. First, we can compute $\vec{\boldsymbol{\tau}}_B$ since $\vec{\boldsymbol{\omega}}_B$ is known and its time derivative can again be computed using finite differences. Next, we solve the resulting ODE to determine $\vec{\boldsymbol{\omega}}_{RW}$ over time, and finally we can compute $\vec{\boldsymbol{\tau}}_{RW}$ when the reaction wheels' angular-velocity profiles are known. The mass moment of inertia matrices for the satellite and reaction wheels are, respectively,

$$\boldsymbol{J}_B = \begin{bmatrix} 18 & 0 & 0 \\ 0 & 18 & 0 \\ 0 & 0 & 6 \end{bmatrix} \times 10^{-3} \text{ kg m}^2 \quad \text{and} \quad \boldsymbol{J}_{RW} = \begin{bmatrix} 28 & 0 & 0 \\ 0 & 28 & 0 \\ 0 & 0 & 28 \end{bmatrix} \times 10^{-6} \text{ kg m}^2. \tag{6.5}$$

Based on the manufacturer's data[2], we develop a simple equation to model the

[2]Sinclair, Doug, *10 mNm-sec power consumption curves.* Accessed June 2012. *http://www.sinclairinterplanetary.com/reactionwheels*

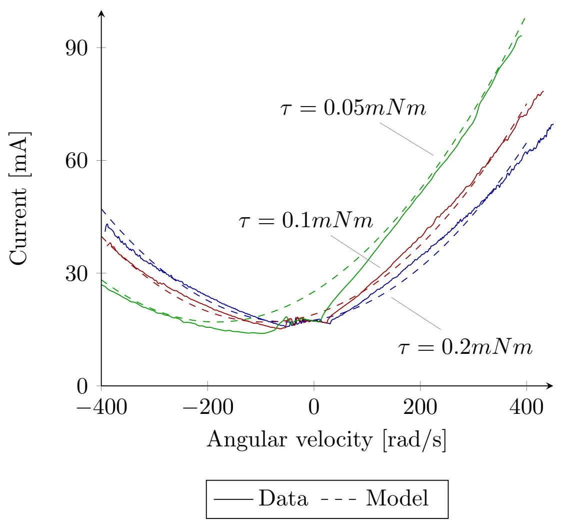


Figure 6.3: Reaction-wheel model compared with manufacturer-provided data for three torques.

dependence of the reaction wheel's current draw on its angular velocity and desired torque:

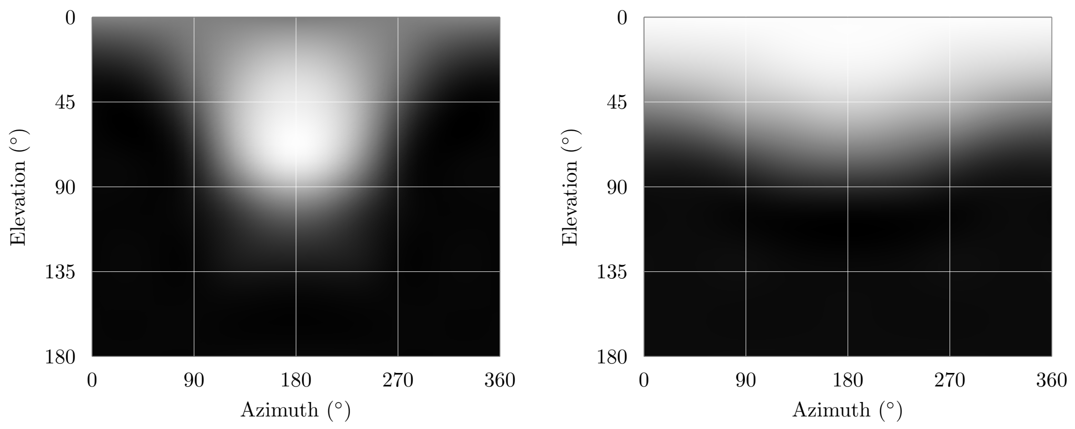
$$I = (a\omega + b\tau)^2 + I_0. \tag{6.6}$$

Given the right coefficients, this simplified model correctly captures the trends, as shown in Fig. 6.3. When both the angular velocity and desired torque are zero, there is a constant baseline current draw. As the angular velocity increases in either direction, the current draw increases roughly quadratically with the torque constant. However, the behavior is asymmetric, since we need less power to achieve a torque in the opposite direction of the angular velocity, which amounts to slowing down the wheel with the assistance of friction. This effect is reflected in both the actual data and the model, as shown in Fig. 6.3. The motor is assumed to run at 4 V.

6.4.3 Cell illumination

The cell-illumination discipline models the area of each solar cell that is exposed to the Sun, projected onto the plane normal to the Sun's incidence. The 84 exposed areas depend on the fin angle as well as the azimuth and elevation angles of the Sun in the body-fixed frame. We compute these using an OpenGL model of the geometry, in which the satellite is discretized into small rectangles.

Since this model is both discontinuous and difficult to incorporate into the framework, we generate a table of data, and we use the B-spline multi-dimensional interpolant mentioned in Sec. 6.3.3. to provide an approximation of the exposed areas in



(a) Fin angle: 30°

(b) Fin angle: 80°

Figure 6.4: Normalized exposed area as a function of relative Sun position for the outermost cell in an inward-facing panel.

terms of the three parameters. This also has the effect of smoothing the areas since the B-spline interpolant does not have a sufficient number of control points to capture the discontinuous jumps, but it does have the degrees of freedom to follow the general trends. Figure 6.4 shows the variation in the exposed area as a function of the Sun's position for the outermost cell in one of the inward-facing panels.

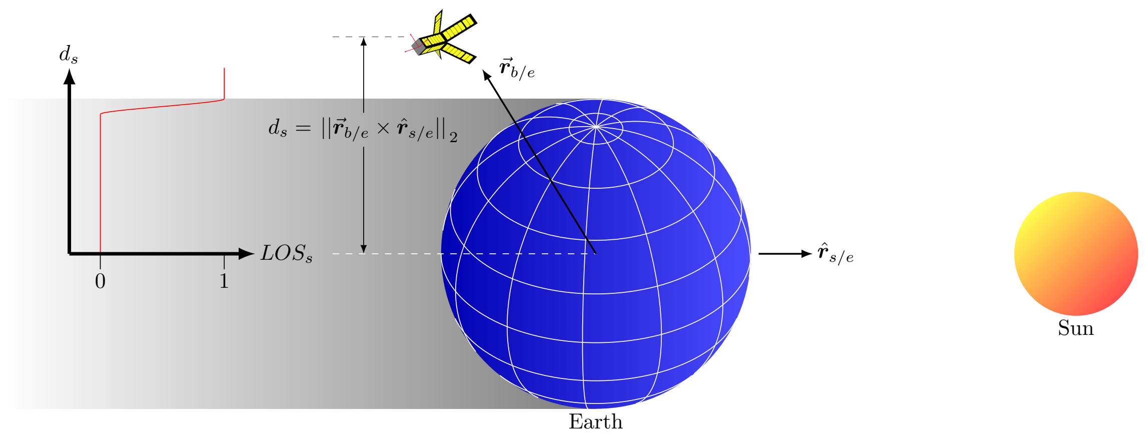
The line-of-sight variable, LOS_s, is essentially a multiplier for the exposed areas: it is 0 if the satellite is behind the Earth and 1 otherwise. To smooth this discontinuous jump, we assume that the sunlight does not decrease instantaneously as the satellite moves into the Earth's shadow, but instead, smoothly transitions to zero. This is physically the case to a certain extent because of the umbra and penumbra effects, but it is greatly exaggerated to avoid numerical difficulties in the optimization. The procedure used to compute LOS_s is illustrated in Fig. 6.5, and it is defined by

$$LOS_s = \left\{ \begin{matrix} 1 & , & \vec{r}_{b/e} \cdot \hat{r}_{s/e} \geq 0 \\ \left\{ \begin{matrix} 1 & , & d_s > R_e \\ 3\eta^2 - 2\eta^3 & , & \alpha R_e < d_s < R_e \\ 0 & , & d_s < \alpha R_e \end{matrix} \right\} & , & \vec{r}_{b/e} \cdot \hat{r}_{s/e} < 0 \end{matrix} \right\}, \tag{6.7}$$

where d_s and η are given by

$$d_s = ||\vec{r}_{b/e} \times \hat{r}_{s/e}||_2 \quad \text{and} \quad \eta = \frac{d_s - \alpha R_e}{R_e - \alpha R_e}. \tag{6.8}$$

Mathematically, we construct a simple cubic function between $d_s = \alpha R_e$ and $d_s = R_e$,



d_s
$d_s = ||\vec{\boldsymbol{r}}_{b/e} \times \hat{\boldsymbol{r}}_{s/e}||_2$
$\vec{\boldsymbol{r}}_{b/e}$
LOS_s
0
1
$\hat{\boldsymbol{r}}_{s/e}$
Sun
Earth

Figure 6.5: Illustration of the Sun line-of-sight variable.

satisfying C^1 continuity at each end point. The value of α represents how far this smoothing effect extends into the Earth's shadow; a typical value is $\alpha = 0.9$.

6.4.4 Temperature

Temperature is an important consideration that couples many disciplines: it affects solar power generation and battery performance, while both cell illumination and data transmission generate heat. The temperature is assumed to be uniform within each of the fins and the body, so there are five temperature state variables at each time instance. We use the Stefan–Boltzmann law to model the rate of heat radiation, and we use the area exposed to the Sun to compute each cell's contribution to the heating of its fin. Since communication power amplifies data transmission with an efficiency, η_p, of roughly 20%, we assume that the remaining 80% is converted to heat, which contributes to the temperature ODE for the body. The equations are

$$\dot{T} = \frac{\dot{Q}_{in} - \dot{Q}_{out} + \dot{Q}^*_{comm}}{mc_v} \tag{6.9}$$

$$\dot{Q}_{in} = \alpha q_{sol} A_{exp} LOS_s \tag{6.10}$$

$$\dot{Q}_{out} = \epsilon \left(\frac{2\pi^5 k^4}{15c^2h^3} \right) T^4 A_T \tag{6.11}$$

$$\dot{Q}_{comm} = (1 - \eta_p) P_{comm}, \tag{6.12}$$

where A_{exp} is the exposed area of the cell, T is the temperature, and $\dot{Q}$ is the rate of heat transfer. The values for all the constants are listed in Table 6.1.

6.4.5 Solar power

The cells in each solar panel are connected in series, so their output voltages are added to compute the total voltage for the panel. The voltage is set so as to maximize the power output, but the optimal voltage, and thus the optimal current, changes depending on the illumination and temperature of the cells.

Each cell has a unique I-V curve that depends on its exposed area and temperature. Our model is based on one [66] that is a nonlinear implicit equation in I given by

$$I = I_{sc} - I_{sat}\left[\exp\left\{\frac{V + R_s I}{V_T}\right\} - 1\right] - \frac{V + R_s I}{R_{sh}}, \tag{6.13}$$

where the values of the constants are listed in Table 6.1, and

$$I_{sc} = LOS_s \frac{A_{exp}}{A_T} I_{sc_0} \qquad \text{and} \qquad V_T = \frac{nkT}{q}. \tag{6.14}$$

Our model has two modifications. First, the series resistance is very small, so the two terms containing R_s can be neglected. Second, a diode is used to limit the voltage in the negative region to $V_0 = -0.6\,\text{V}$, so a shifted hyperbolic tangent function is used to model the I-V curve for negative voltages. We determine the coefficient in the argument of tanh by applying the constraint that the first derivative must be continuous at $V = 0$. Since V is still an implicit function of I in the positive voltage region, we evaluate the model for the full ranges of areas and temperatures, and we fit the B-spline interpolant discussed earlier. The model is plotted in Fig. 6.6 and the expressions are

$$\left\{\begin{array}{ll} I_{sc} - I_{sat}\left[\exp\left\{\frac{V}{V_T}\right\} - 1\right] - \frac{V}{R_{sh}} - I = 0 & , \quad I \leq I_{sc} \\ V(I) = V_0 \tanh\left[\frac{-V_T R_{sh}}{V_0(I_{sat}R_{sh} + V_T)}(I - I_{sc})\right] & , \quad I > I_{sc} \end{array}\right\}. \tag{6.15}$$

6.4.6 Energy storage

The energy-storage discipline tracks the state of charge (SOC) of the battery. We can compute the SOC by integrating the nonlinear ODE given by

$$S\dot{O}C = \frac{P_{bat}}{V_{bat} Q}, \tag{6.16}$$

where Q is the nominal discharge capacity of the battery.

We model the dependence of the voltage on the SOC as an exponential, primarily

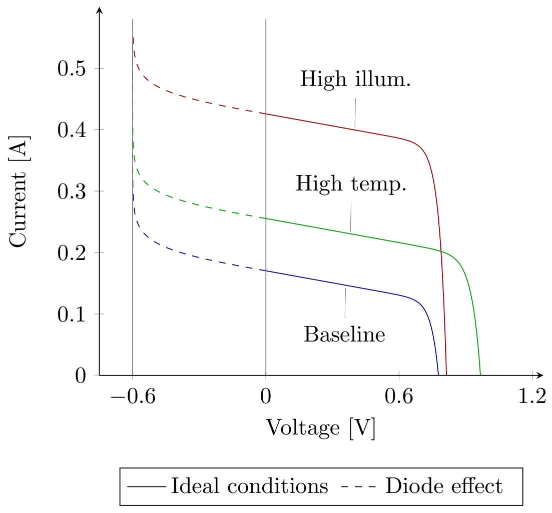


Figure 6.6: Solar cell I-V curve at different cell temperatures and exposed areas.

to ensure that the voltage is always positive. A linear relationship would have been within the scope of this work. However, a battery at a large negative SOC has a negative voltage, and drawing power from the battery would increase its SOC since the current is still positive. Negative states of charge often arise at the initial point in an optimization, when a poor baseline design point uses more power than is available. In these circumstances, the model must provide the optimizer with the correct gradient directions instead of failing. Artificially removing the drop-off in voltage does not lead to inaccuracies that affect our results, since the optimization constrains the SOC to be nonnegative, which ensures that the optimal design is never in this drop-off region.

The dependence of the voltage on the temperature is also exponential, as shown in Fig. 6.7, which compares the model to the manufacturer's data[3]. The values for the constants are listed in Table 6.1, and the expression is

$$V_{bat}(SOC) = \left(3 + \frac{e^{SOC} - 1}{e - 1}\right)\left(2 - e^{\lambda \frac{T - T_0}{T_0}}\right). \tag{6.17}$$

At any given time instance, the battery power is the sum of the loads, i.e.,

$$P_{bat} = P_{sol} - P_{RW} - P_{comm} - P_0, \tag{6.18}$$

[3]Panasonic, *Batteries & Energy Products-Lithium Ion Batteries, Cylindrical type, NCR18650.* Accessed Jan. 2013.
http://industrial.panasonic.com/www-data/pdf2/ACA4000/ACA4000CE240.pdf

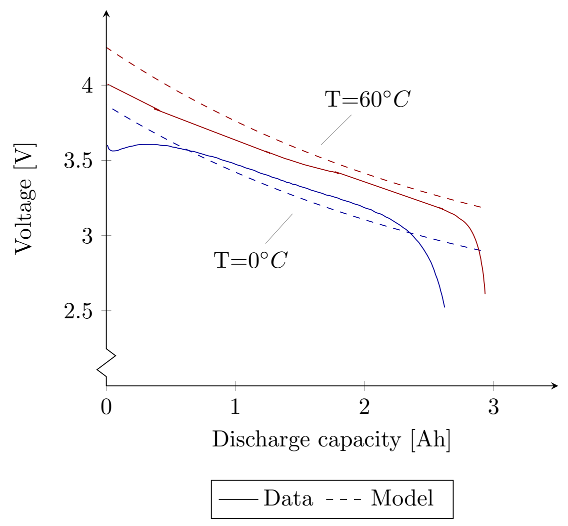


Figure 6.7: Battery-discharge curve model compared with manufacturer's data at two temperatures.

where P_0 is a 2-W constant power usage that accounts for the scientific instruments on the satellite and small actuator inputs in response to disturbance torques.

6.4.7 Communication

The communication discipline models the data-transfer bit rate as a function of several variables. We fix the signal-to-noise ratio (SNR) to a minimum acceptable value to maintain a reliable connection. A line-of-sight variable, similar to that computed in the Sun-position discipline, is used to account for the times when a link with the ground station is not possible. We compute the resulting data-download rate using the following equation [67, pp. 550–558]:

$$B_r = \frac{c^2 G_r L_l}{16\pi^2 f^2 k T_s (SNR)} \frac{\eta_p P_{comm} G_t}{S^2} LOS_c, \tag{6.19}$$

where the constants are listed in Table 6.1, S is the distance to the ground station, and G_t is the transmitter gain, which is plotted in Fig. 6.8.

We compute the LOS_c variable based on the dot product between the normalized Earth-to-ground station vector and the Earth-to-body vector in the inertial frame. We again smooth the discontinuous function, in this case by assuming that the line-of-sight variable gradually increases as the satellite comes over the horizon. This is illustrated in Fig. 6.9.

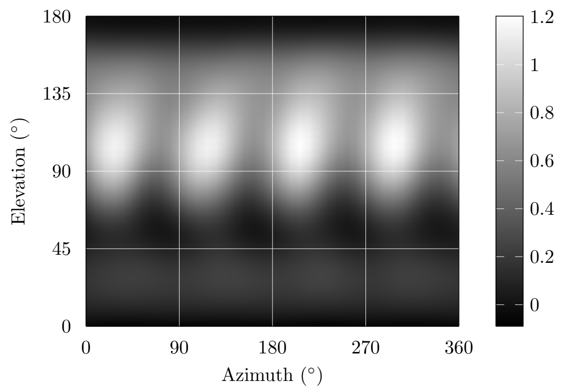
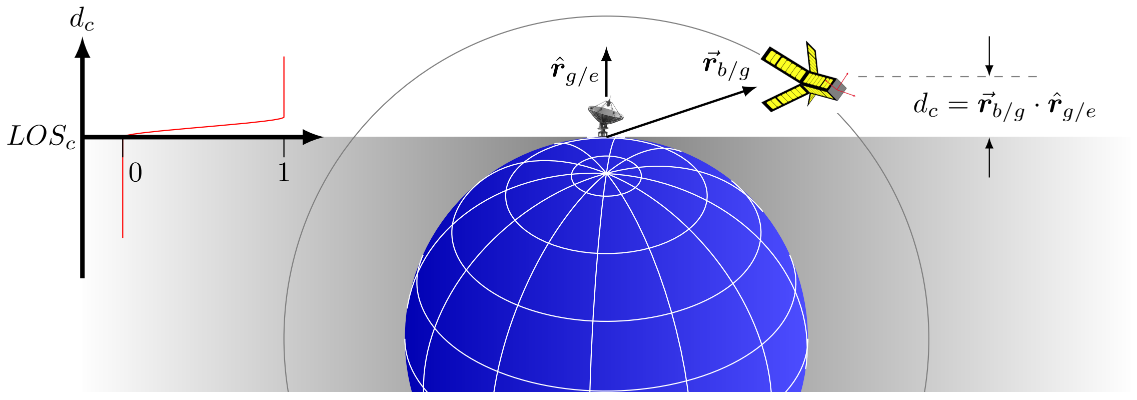


Figure 6.8: Transmitter gain as a function of the ground-station position in the body-fixed frame.



d_c
LOS_c
0
1
$\hat{\boldsymbol{r}}_{g/e}$
$\vec{\boldsymbol{r}}_{b/g}$
$d_c = \vec{\boldsymbol{r}}_{b/g} \cdot \hat{\boldsymbol{r}}_{g/e}$

Figure 6.9: Illustration of the ground-station line-of-sight variable.

Orbit Dynamics		
Earth's gravitational parameter	μ	398600.44 km^{3}s^{-2}
Earth's radius	R_e	6378.137 km
Orbit perturbation coefficients	J_2	1.08264×10^{-3}
	J_3	-2.51×10^{-6}
	J_4	-1.60×10^{-6}
Attitude Dynamics		
Model coefficients	a	4.9×10^{-4} A$^{1/2}$ s/rad
	b	4.5×10^{2} A$^{1/2}$/(Nm)
	I_0	0.017 A
Temperature		
Mass	m	0.4 (fin), 2.0 (body) kg
Specific heat capacity	c_v	0.6 (fin), 2.0 (body) kJ/kg K
Absorptivity	α	0.9 (cell), 0.2 (radiator)
Emissivity	ϵ	0.87 (cell), 0.88 (radiator)
Boltzmann constant	k	$1.3806488 \times 10^{-23}$ m^2 kg/(s^2 K)
Speed of light	c	2.99792458×10^{8} m/s
Planck's constant	h	$6.62606957 \times 10^{-34}$ m^2 kg/s
Total cell area	A_T	2.66×10^{-3} m^2
Solar constant	q_{sol}	1.36×10^{3} W/m^2
Communication efficiency	η_p	0.2
Solar Power		
Diode voltage	V_0	-0.6 V
Max. short-circuit current	I_{sc_0}	0.453 A
Saturation current	I_{sat}	2.809×10^{-12} A
Diode factor	n	1.35 V
Charge of an electron	q	$1.60217657 \times 10^{19}$ C
Shunt resistance	R_{sh}	40 Ω
Energy Storage		
Nominal capacity	Q	2900 mAh
Temperature decay coeff.	λ	$ln\left(\frac{1}{1.1^5}\right)$
Reference temperature	T_0	293 K
Max. discharge rate	I_{min}	-10 A
Max. charge rate	I_{max}	5 A
Communication		
Receiver gain	G_r	12.9 dB
Line loss factor	L_l	-2.0 dB
Transmission frequency	f	437 MHz
System noise temperature	T_s	500 K
Minimum acceptable SNR	SNR	5.0 dB

Table 6.1: Data for discipline models

6.5 Optimization

6.5.1 MDO architecture

For an optimization problem involving many disciplines, the choice of the MDO architecture is critical. We use the multidisciplinary feasible (MDF) architecture [62], which solves an MDO problem by fully resolving the coupling between all the disciplines within each optimization iteration, effectively treating the coupled analyses of all the disciplines as one monolithic analysis. The rationale is that taking a restricted path to the optimum, with the interdisciplinary coupling converged at every optimization iteration, yields a robustness that is likely necessary for a problem with such a large number of disciplines. For a review of MDO architectures, the reader is encouraged to refer to Martins and Lambe [62].

However, the approach has elements that resemble the simultaneous analysis and design (SAND) architecture [68, 62] because some of the design variables could also be state variables. The roll-angle design variables could be replaced with reaction-wheel control inputs that are computed using a control law, and the optimal solar panel current at every time instance could be computed using maximum power point tracking (MPPT). Instead, we use nonlinear optimization as the controller in the former case. In the latter case, we compute all the peak power currents simultaneously as a smooth profile over time; our goal is to avoid poor conditioning due to local maxima.

Overall, this SAND-type approach yields three benefits. First, it avoids assumptions that would be required if these design variables were implemented as state variables. For instance, the attitude-control law must assume a desired roll-angle profile based on predetermined weights for the solar panel heating and cooling, cell illumination, and communication signal strength, while the optimization considers the net effect of rolling on the objective function by way of these three criteria. Second, it eliminates the risk that a discipline may not have a feasible solution, such as the attitude controller lacking the power to satisfy the forward-facing orientation constraint for any roll angle. Allowing the optimizer to control the distribution of power and enforce the battery-power and charge-level constraints ensures that all the disciplines are feasible internally, while we allow the battery constraints to be violated during the optimization. Finally, it removes the coupling between disciplines from the multidisciplinary analysis by moving the appropriate state variables to the optimization as design variables. The optimizer resolves the coupling, allowing the MDA to become a sequential problem, as shown in Fig. 6.10.

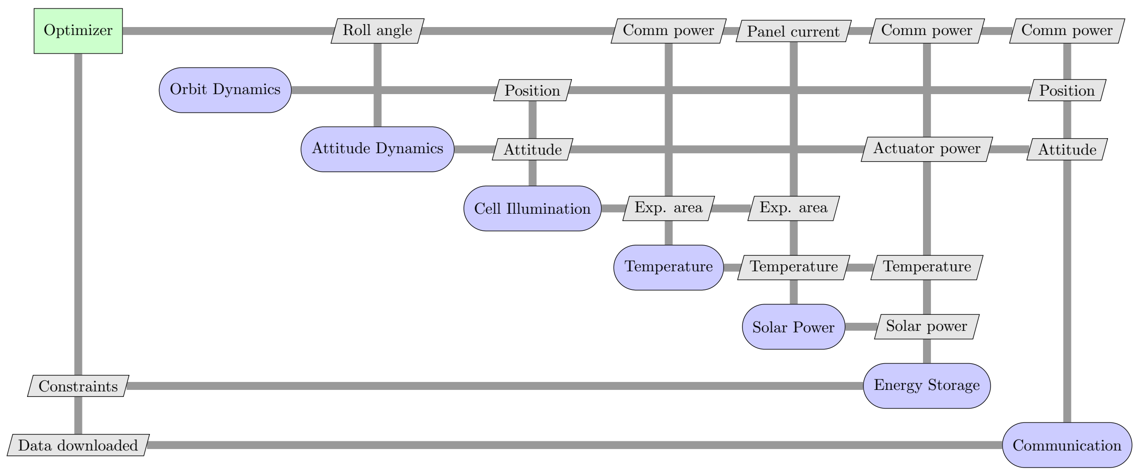


Figure 6.10: Extended design structure matrix (XDSM) diagram [1] for the MDO problem.

To avoid confusion, it is worth restating that the MDO architecture used in this problem is still MDF. The connection to SAND is limited to the fact that certain variables that could have been state variables have been implemented as design variables. However, the remaining state variables are not exposed to the optimizer, and all the variables are converged fully within every optimization iteration, which is consistent with the MDF architecture.

6.5.2 Optimization problems

As previously mentioned, the multi-scale nature of the problem requires a multipoint optimization with 6 points, each representing a 12-hour simulation at the midpoint of every 2-month interval. We simulate half a day, 1, 3, 5, 7, 9, and 11 months after launch. This results in a multidisciplinary analysis with a total of 2,204,861 variables. The objective function is the average of the data-downloaded values of the 6 points, which is an estimate of the total annual data downloaded after a scaling factor. The battery charge rate, discharge rate, minimum SOC, maximum SOC, and periodicity constraints are enforced separately for each of the 6 points. The periodicity constraint enforces equality of the SOC at the beginning and end of each 12-hour simulation. The remaining four constraints are KS aggregation functions.

There are two scalar design variables (fin angle and antenna angle) and 84 binary variables that indicate whether or not a cell or radiator is installed. The variables that require the parametrization of their variations over time are the roll angle, the

	Variable/function	Description	Quantity
maximize	$\sum_{i=1}^{6} D_i$	Data downloaded	
with respect to	$0 \leq I_{\text{setpt}} \leq 0.4$	Solar panel current	$300 \times 12 \times 6$
	$0 \leq \gamma \leq \pi/2$	Roll-angle profile	300×6
	$0 \leq P_{\text{comm}} \leq 25$	Communication power	300×6
	$0 \leq \text{cellInstd} \leq 1$	Cell vs. radiator	84
	$0 \leq \text{finAngle} \leq \pi/2$	Fin angle	1
	$0 \leq \text{antAngle} \leq \pi$	Antenna angle	1
	$0.2 \leq SOC_i \leq 1$	Initial state of charge	6
		Total number of design variables	25292
subject to	$I_{\text{bat}} - 5 \leq 0$	Battery charge	6
	$-10 - I_{\text{bat}} \leq 0$	Battery discharge	6
	$0.2 - SOC \leq 0$	Battery capacity	6
	$SOC - 1 \leq 0$	Battery capacity	6
	$SOC_f - SOC_i = 0$	SOC periodicity	6
		Total number of constraints	30

Table 6.2: The general optimization problem.

12 solar panel currents, and the power allotted to the communication discipline for transmission. The current variable has the effect of emulating MPPT for the solar-power module, since the optimizer effectively selects the current, and indirectly the voltage, at which the maximum power can be generated from the cells in a given solar panel. Each profile variable is discretized with 1500 points, which is the number of points used in the time integrations, and they are represented using fourth-order B-splines with 300 control points. The optimization problem is summarized in Table 6.2.

As previously mentioned, we solve the optimization problem using SNOPT [69], a reduced-Hessian active-set SQP optimizer that solves nonlinear constrained problems very efficiently, particularly when derivatives are provided, as is the case here. We use the pyOpt optimization framework [9]; it provides a common interface to a suite of optimizers, including SNOPT.

Figure 6.11 plots the convergence history for the optimization. The number of function evaluations roughly corresponds to the number of SQP major iterations, and each takes about 20 min on a single processor, including the derivative computations. Overall, the algorithm requires 100 h to achieve convergence of nearly 5 orders of

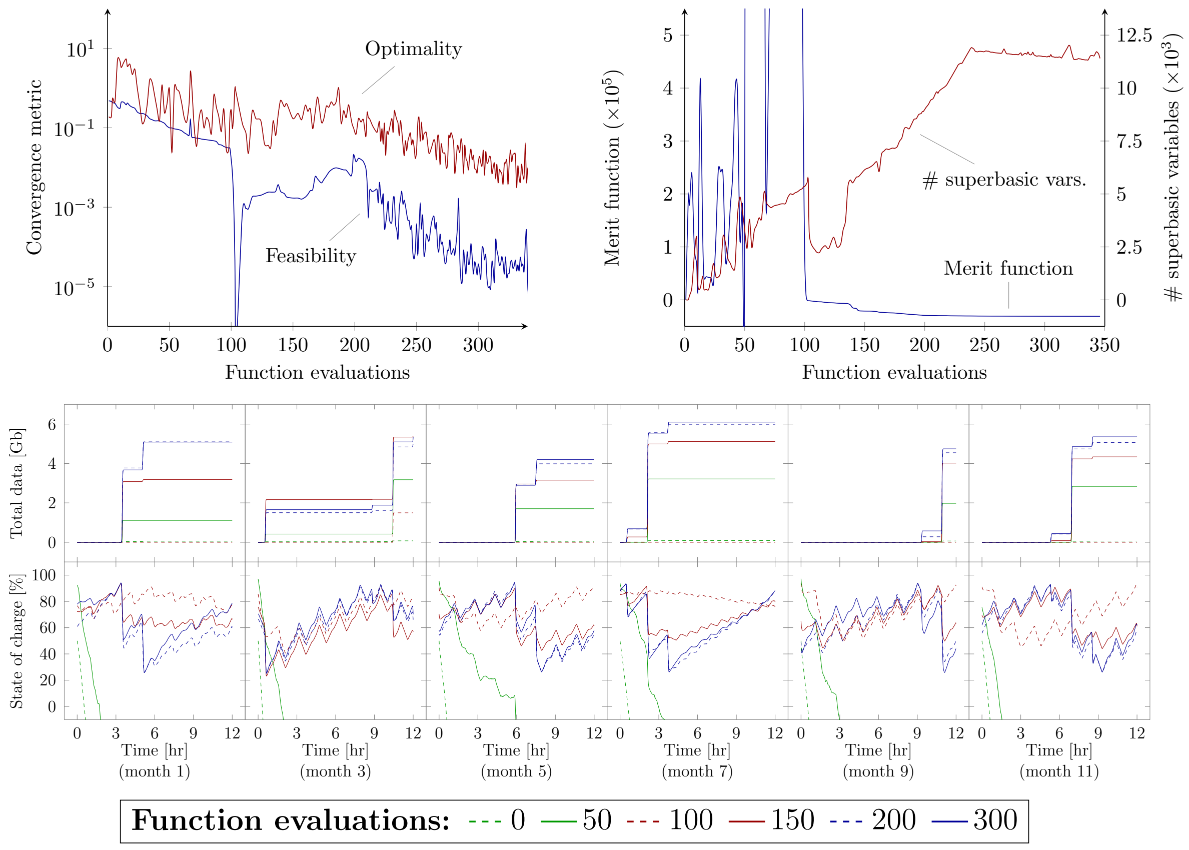


Figure 6.11: Convergence histories and snapshots of data and SOC at intermediate optimization iterations.

magnitude in feasibility and 3 orders of magnitude in optimality. Figure 6.11 shows that by the end of the optimization, nearly half of the design variables are superbasic variables in SNOPT, which are those variables that are truly free to change because they are not fixed by bounds or constraints. This indicates that near this local optimum, the dimension of the feasible design space is large, meaning that there is considerable design freedom with respect to which the design is optimal.

Figure 6.11 also illustrates the sequence in which the objective function was improved and the battery constraints were satisfied. For all six points, the initial design is clearly infeasible since the SOC curve is mostly negative. The optimizer spends most of the first 100 function evaluations trying to increase the power generation to make the SOC curves feasible. After this, it focuses on increasing the objective function.

	Fin angle	Antenna angle	Data downloaded
Baseline optimization	45°	0°	2122 Gb/yr
Geometry optimization	63.8°	−45°	2991 Gb/yr
Geometry and attitude optimization	64.4°	−45°	3758 Gb/yr

Table 6.3: Optimal design variables for for the three optimization problems.

6.5.3 Impact of optimization

To quantitatively assess the impact of the optimization, we solve three optimization problems. The first is a baseline optimization that is the same as the original optimization problem in Table 6.2, except that the fin angle, antenna angle, and roll-angle profile are removed from the set of design variables. The remaining design variables are the solar-panel current, communication power, initial SOC, and installation of cell or radiator, which provide a baseline design. The second optimization adds the geometric design variables, which are the fin and antenna angles. The third optimization adds both the geometric and attitude design variables to the baseline optimization, yielding the problem described in Table 6.2.

Table 6.3 summarizes the results. Since the constraints are satisfied in all of the optimizations, the objective function alone provides a good metric for comparison. Adding the geometric design variables yields a 40% increase in the estimate of the data downloaded, and adding the roll angle yields an additional 40% increase. Whenever the antenna angle is permitted to vary, it goes to the bound of −45°, while the fin angle converges to an interior optimum. For all the problems, the optimizer chooses to install the solar cell instead of the radiator for all 84 cells. Figure 6.12 shows how the objective-function increases are distributed among the 6 points.

These optimization results can be summarized as follows. The fin angle and roll-angle profile increase the cell illumination to provide more communication power, and the antenna angle increases the gain during the ground-station passes for higher data rates. However, an examination of each variable reveals more insight into how the optimization problems compare, as well as how they satisfy the battery constraints and increase the total downloaded data. Figures 6.13 and 6.14 plot several quantities of interest as functions of time, for each point and for each of the three optimization results.

In Fig. 6.13, the data downloaded plots demonstrate the importance of optimizing both the geometric and attitude design variables. The communication power plots

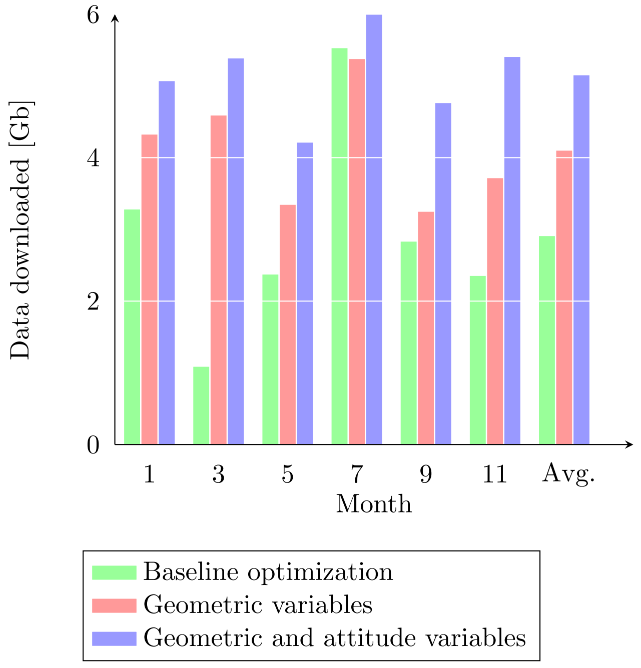


Figure 6.12: Division of total data downloaded over the six simulations for the three optimization problems.

show that the optimizer allocates power to the transmitter only during the ground-station passes, as expected, but the peaks of the spikes are limited by the available SOC and the discharge constraint. The transmitter gain plots show the highest gains for the geometry and attitude optimization, followed by the geometry optimization, then the baseline optimization. This is evidence that the fin angle provides an increase in gain, and the roll-design variables provide a further increase in gain, which translates to higher data rates. An interesting observation regarding the roll-angle profiles is that they are smooth for the most part but exhibit spikes aligned with ground-station passes. Finally, the SOC plots show that the additional power generated by the optimizations is used for a gradual build-up of energy between data transmissions, enabling short and rapid power discharges for high-bit-rate data transmissions.

In Fig. 6.14, the large increase in the solar power generation from the baseline optimization to the geometry optimization and the smaller increase from the geometry optimization to the geometry and attitude optimization indicate that the fin angle has a large effect, and the attitude profile has a smaller but still definite effect. The solar-panel current curves represent the maxima for each time instance among the twelve panels, and they correctly go to zero when the satellite is in the Earth's shadow to prevent negative voltages, while taking on optimal current values when in the sun

Launch	Fin angle	Antenna angle	Data downloaded
1	64.4°	$-45°$	3758 Gb/yr
2	49.9°	$-45°$	3829 Gb/yr
3	68.5°	$-45°$	3587 Gb/yr

Table 6.4: Optimal values for the three launches.

to maximize power. As with the solar-power plots, the total-exposed-area plots give a clear indication that the fin angle has the largest effect in increasing cell illumination, and interestingly, the exposed area is sacrificed in months 1 and 7 when the satellite is always in the sun and is not power constrained. The body temperature is weakly dependent on the roll-angle profile and it also has a smaller effect on the solar power, so periodicity constraints are not used to avoid the additional linear solutions required for the associated derivatives. The battery-current plots show that the communication power is limited by the battery-discharge constraint for many of the ground-station passes, while the remainder are energy-limited.

6.5.4 Comparison between launches

One of the strengths of our approach and implementation is that the design optimization algorithm is robust with respect to convergence. To aid decision-making, it is possible to run optimizations for various choices of parameters, such as the launch, ground-station selection, and satellite specifications, and to compare them. To illustrate, we ran two additional optimizations for different launch orbits and dates. These optimizations involve the design variables listed in Table 6.2.

The results are summarized in Table 6.4. The antenna angles converge to the same value, and the estimated data downloaded is roughly the same for the three launches. However, there is a large discrepancy in the optimal fin angle, which suggests that it could be sensitive to the launch orbit. It has been consistently observed that the fin angle has a large effect on the potential power generation, and the optimal fin angle varies significantly for different launches as the result for launch 2 shows. This observation points to the importance of carefully selecting the fin angle once the launch orbit is known. Figure 6.15 shows how the total downloaded data is divided among the points.

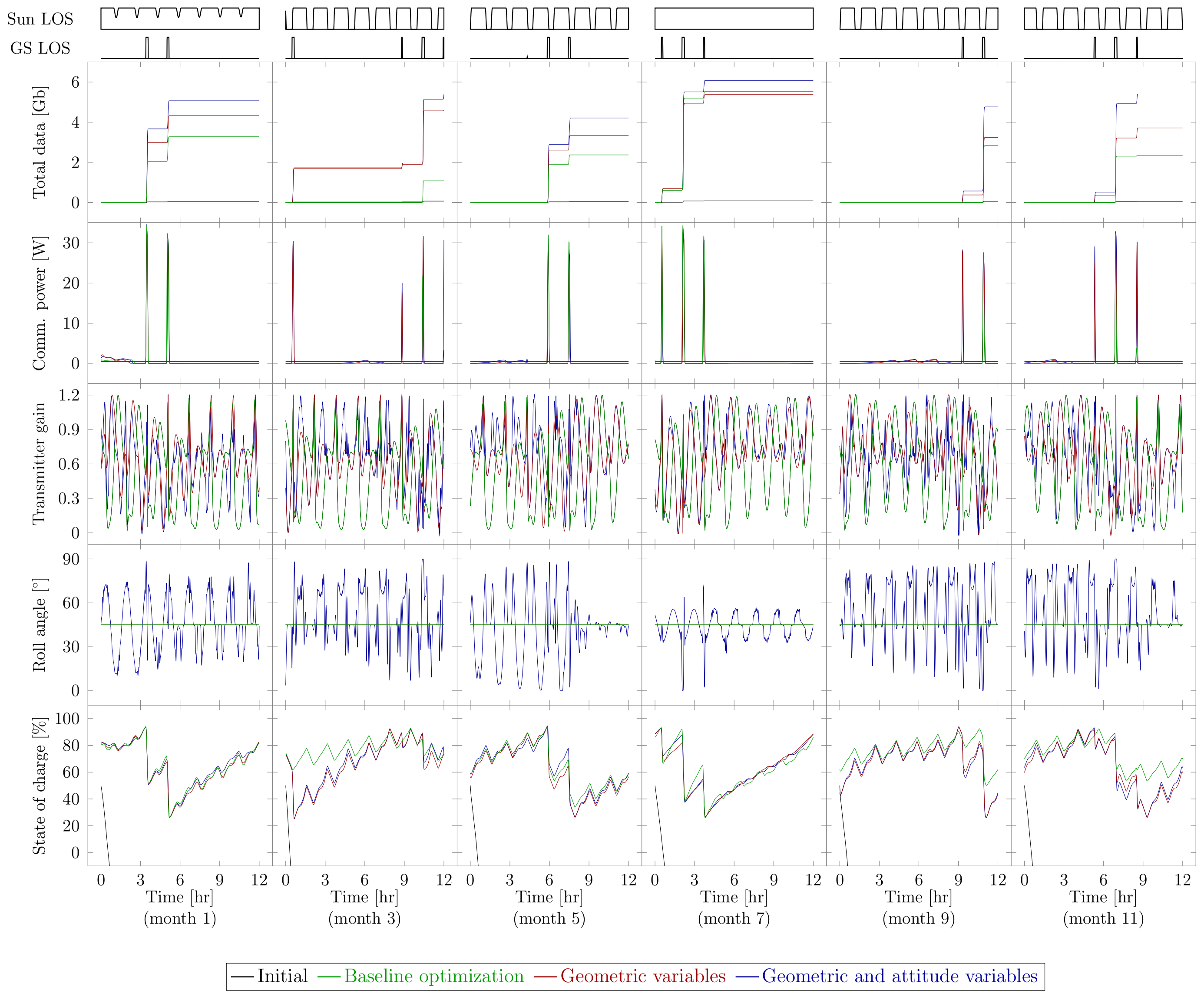


Figure 6.13: Initial and optimized profiles of quantities of interest for the three optimization problems (continued in Fig. 6.14).

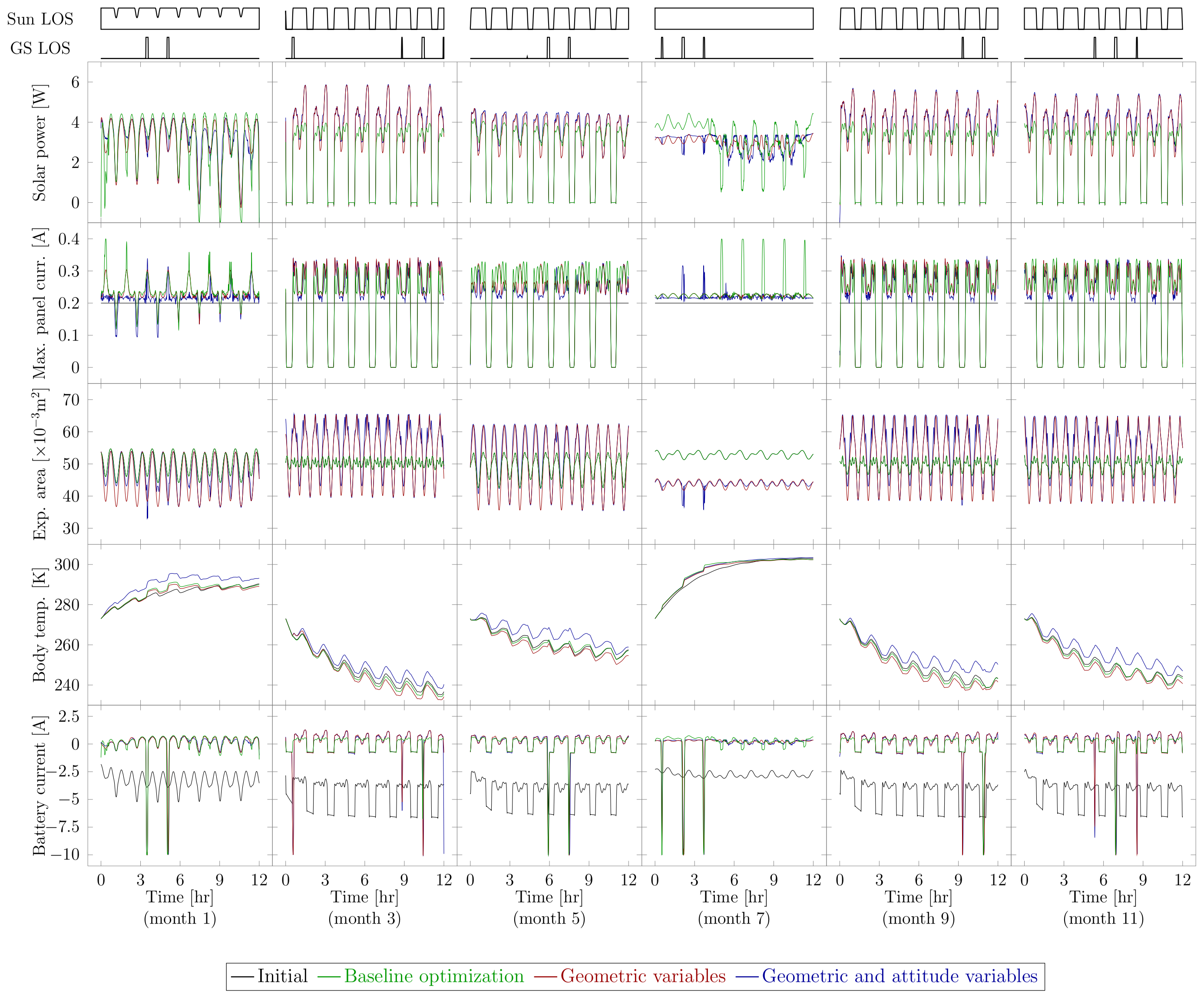


Figure 6.14: Initial and optimized profiles of quantities of interest for the three optimization problems (continued from Fig. 6.13).

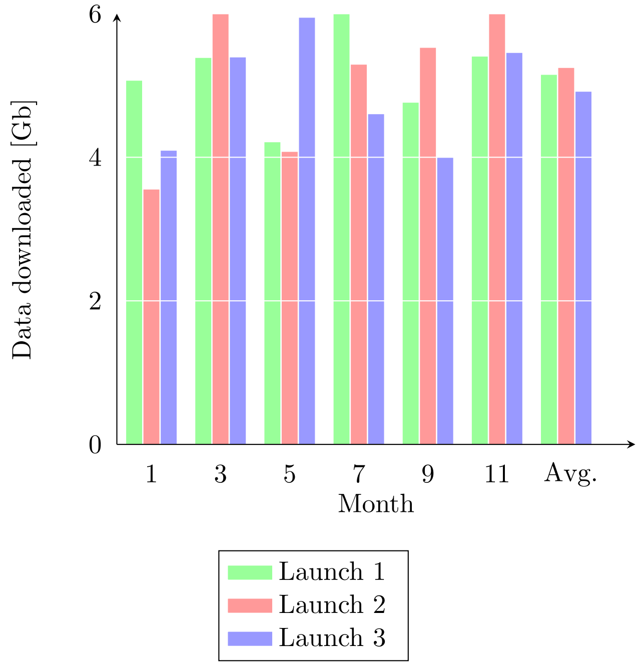


Figure 6.15: Division of total data downloaded over the six simulations for the three launches.

6.6 Summary

The objective for the work presented in this chapter was to apply large-scale multidisciplinary design optimization to a small satellite. This work demonstrated the ability to reliably solve an optimization problem with 7 disciplines, more than 25,000 design variables, and over 2.2 million state variables that represent 12 hours of the satellites operation at 6 uniformly spaced points over the year. To assess the impact of this tool, three optimization problems were solved with varying sets of design variables. The addition of geometric design variables to the satellite design problem yielded a 40 % improvement in the objective function (the total data downloaded), and the addition of operational design variables yielded a further 40 % improvement. Furthermore, changing the launch parameters changed the values of the objective function and the design variables, suggesting that this tool could be used to evaluate launch options and to tailor the design to a particular launch opportunity.

The computational modeling framework played an important role in enabling the implementation and execution of this design optimization problem. Each discipline is decomposed into separate computations to simplify and modularize the code, but this results in a large number of components. For instance, the output of the communication discipline is ultimately the total data downloaded, but it is broken down

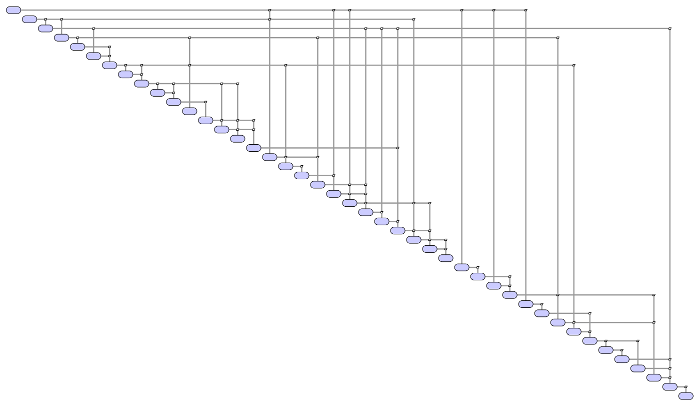


Figure 6.16: Design structure matrix diagram illustrating the complexity of the problem. Each block on the diagonal represents one of 43 components that each computes one or more quantities in the computational model. Off-diagonal entries represent dependencies—variables passed from one component to another.

into the computations of the ground station line-of-sight variable, the position vector from the satellite to the ground station, the transmitter gain as a function of this vector, the data-download rate, and the total data downloaded. In total, there are 43 components when those of all the disciplines are combined, and Fig. 6.16 shows the dependencies to illustrate the scope and complexity of this problem. The framework facilitated a modular integration of the components and automatically computed the total derivatives, which was a key feature for this problem.

CHAPTER 7

Allocation-mission optimization of commercial aircraft

The second problem is the large-scale optimization of the operation of commercial aircraft. With the objective of maximizing profit at the airline level, this problem aims to find the optimal allocation of available aircraft to routes and the optimal mission profiles on each of those routes.

This chapter begins by motivating the simultaneous consideration of the allocation and mission problems from high-fidelity aircraft design optimization, in Sec. 7.1. Next, the prior art in mission and allocation optimization is surveyed in Sec. 7.2 and Sec. 7.3, respectively. Section 7.4 presents the models and algorithms for the mission, allocation, and simultaneous allocation-mission problems. Finally, Sec. 7.5 presents allocation-mission optimization results and Sec. 7.6 summarizes the conclusions.

7.1 A motivation from aircraft design optimization

Part II of this thesis deals with high-fidelity design optimization algorithms for unconventional aircraft configurations. In the field at large, the use of computational design tools in aircraft design is growing, as computing hardware improves and methods for high-fidelity optimization become more mature. Computational fluid dynamics (CFD) and finite element analysis (FEA) enable expensive, but accurate aerodynamic and structural simulations of aircraft, respectively, and optimization provides a tool to automatically compute the aircraft design that is optimal in some sense.

This begs the questions of what the objective function should be and at what operating conditions the aircraft should be optimized or designed. For aerodynamics, one common choice is to formulate the optimization problem as lift-constrained drag minimization, assuming a required lift that is hopefully representative of how the aircraft would actually be flown. An improvement upon this would be to minimize the Breguet range equation, which approximates the fuel burn over the mission by making assumptions such as a constant lift-to-drag ratio.

For even greater accuracy, one would want to analyze the entire mission by solving for the angle of attack profile, thrust profile, etc. required to fly a desired altitude and speed profile. The mission analysis is necessary to quantify the true impact of design improvements in terms of the total fuel burn. This is particularly relevant with the ongoing research on continuous descent approach (CDA), also known as optimized profile descent (OPD).

Considering the full mission profile improves accuracy, but it raises another question—the choice of range and payload. A simplification would be to select a representative mission range and number of passengers, and let the objective function be the total fuel burn on this route. The more accurate approach would be to solve an optimization problem mimicking a profit-seeking airline that distributes limited fleets of aircraft of different types to a network of routes.

In summary, the true optimization problem would include allocation, mission, and design simultaneously. This problem would optimize allocation variables representing how many of which aircraft are flown on what routes, mission variables parametrizing the altitude profiles for each aircraft type potentially flying on each route, and design variables representing the aircraft shape and sizing. The objective function would be profit, so this simultaneous optimization would achieve two things. First, it would quantify for an airline the potential increase in profit from adding the unconventional aircraft to its fleet. Second, the outcome would be an optimized design for the unconventional configuration that is the most attractive to airlines and is likely fuel efficient, since fuel burn reduces profit.

This simultaneous optimization problem is ambitious in scope, particularly because of the large number of aircraft simulations required. One simulation is required for each discretized point in the mission, multiplied by the number of iterations in the solution of the mission equations, multiplied by the number of routes and aircraft, multiplied by the number of optimization iterations. This potential bottleneck is remedied by using a surrogate model for the aircraft simulation with Mach number, altitude, angle of attack, and tail rotation angle as input variables. Each optimization iteration, this surrogate model would be re-trained given the design corresponding to the current design variable iterates. A modular approach is needed because of the scope and complexity of this problem, so this problem would take advantage of the framework presented in Ch. 5.

7.2 Prior art in allocation optimization

One of the first applications of linear programming was the airline allocation problem, first published in 1956 [70], less than 10 years after the development of the simplex method in 1947 [71]. The original problem aimed to find the optimal allocation of aircraft to routes given uncertain demand [70], assuming a fixed cost incurred by flying a given route with a given type of aircraft.

Beginning in 1966 [72], researchers began applying optimization to schedule design for flights, and later, the fleet assignment problem (FAP)—assigning aircraft to scheduled flights. In 1989, Abara showed that it is possible to use linear programming to solve FAPs of representative sizes [73]. Recent research has focused on improvements upon these formulations with for instance improved revenue models [74, 75], homogeneity of aircraft types at airports [76] or on legs [77], and simultaneous optimization of scheduling and fleet assignment [78]. More information can be found in survey articles by Clarke and Smith [79] and Barnhart et al. [80].

The fruits of research in this field have been used by airlines to improvement their operations and increase profit. For instance, algorithms for solving fleet assignment problems have been used by American Airlines [73], Delta Airlines [81], and US Airways [82].

For the current problem, the motivation for considering the allocation problem differs from that of the airlines. The objective is to solve the allocation problem as a tool to evaluate potential new aircraft designs by quantifying their airline-level benefits. It must be possible to simultaneously optimize the variables parametrizing the mission profile and aircraft design, so simplifying assumptions are used. Specifically, the original allocation problem proposed in 1956 [70] is used, and the considerations for scheduling are ignored.

7.3 Prior art in mission optimization

The prior art in mission profile optimization divides broadly into two categories, direct and indirect [83]. The direct approach first discretizes the equilibrium equations and applies the optimality conditions on the discretized equations. The indirect approach takes the reverse order—it first differentiates the equilibrium equations to derive continuous optimality conditions and then discretizes as the second step.

Only the direct approach is reviewed as it is the one used in this chapter. Betts and Huffman performed trajectory optimization for a hypersonic re-entry vehicle using the direct approach [84]. They used a custom active-set gradient-based optimization algorithm with a sparse finite differencing method. Betts and Cramer applied a similar methodology to the trajectory optimization of commercial aircraft using sparse finite differencing and gradient-based optimization, and B-spline interpolants were used as the aerodynamic model [85]. Park and Clarke solved the trajectory optimization problem using the pseudospectral method, which discretizes state and control variables at points determined by a Gaussian quadrature [86].

An application of mission profile optimization is for continuous descent approach (CDA), also known as optimized descent profile (ODP). ODP is one component of the next generation air transportation system (NextGen) currently being developed and implemented by the federal aviation administration (FAA). In addition to noise reductions, continuous descents yield fuel burn improvements compared to step descents. In the past decade, continuous and idle descent tests have been designed or implemented at Louisville [87], Denver [88], Los Angles [89], and Malta [90] international airports.

Based on the current interest in ODP, the approach taken for this work is to simultaneously optimize the mission profiles for each route while solving the allocation problem. The expected result is a rapid and smooth climb to a cruise-climb stage and then a continuous descent near idle, with the cruise altitudes determined by the optimizer. The numerical approach is similar to that of Betts and Cramer [85], with 2 key improvements. First, the derivatives needed for optimization are computed using the framework as opposed to finite differences for improved accuracy and efficiency. Second, the mission analysis is implemented in a modular way in the framework, facilitating integrating with the allocation components, and in future, with the aircraft design components.

7.4 Methodology

7.4.1 The mission problem

The main components of the mission analysis are formed by the equilibrium equations for the aircraft and the ordinary differential equation (ODE) for fuel weight. For the former, the vertical equilibrium determines the required angle of attack to roughly match lift with weight, the horizontal equilibrium determines the engine throttle setting to roughly match thrust with drag, and the moment equilibrium determines the elevator deflection to trim the aircraft. The equations are

$$L + W\cos\gamma - T\sin\alpha + \frac{W}{g}v^2\cos\gamma\frac{\mathrm{d}\gamma}{\mathrm{d}x} = 0 \tag{7.1}$$

$$T\cos\alpha + D + W\sin\gamma + \frac{W}{g}v\cos\gamma\frac{\mathrm{d}v}{\mathrm{d}x} = 0 \tag{7.2}$$

$$M - I\left(\frac{\mathrm{d}^2\gamma}{\mathrm{d}x^2}(v\cos\gamma)^2 + \frac{\mathrm{d}\gamma}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}v(\cos\gamma)^2\right) = 0, \tag{7.3}$$

where α is the angle of attack and γ is the climb or descent angle.

Assuming quasi-steady flight conditions, the terms containing derivatives with respect to x can be ignored. Dropping these terms and expressing in terms of non-dimensional coefficients yields

$$\frac{1}{2}\rho v^2 S\tilde{C_L} + W\cos\gamma - \frac{1}{2}\rho v^2 S\tilde{C_T}\sin\alpha = 0 \tag{7.4}$$

$$\frac{1}{2}\rho v^2 S\tilde{C_T}\cos\alpha + \frac{1}{2}\rho v^2 SC_D + W\sin\gamma = 0 \tag{7.5}$$

$$\tilde{C_M} = 0. \tag{7.6}$$

The system of equations is completed by adding surrogate models for the aerodynamics and propulsion, which are of the form

$$\tilde{C_L} - C_L(h, M, \alpha, \eta) = 0 \tag{7.7}$$

$$\tilde{C_M} - C_M(h, M, \alpha, \eta) = 0 \tag{7.8}$$

$$\tilde{C_T} - C_T(h, M, t) = 0, \tag{7.9}$$

where the variables with tildes indicate target variables while C_L, C_D, C_M, and C_T are functions representing the aerodynamic or propulsion surrogates.

The fuel weight is computed by solving the ODE,

$$\dot{W}_f = \frac{\{SFC\}\frac{1}{2}\rho v^2 SC_T}{v\cos\gamma}, \tag{7.10}$$

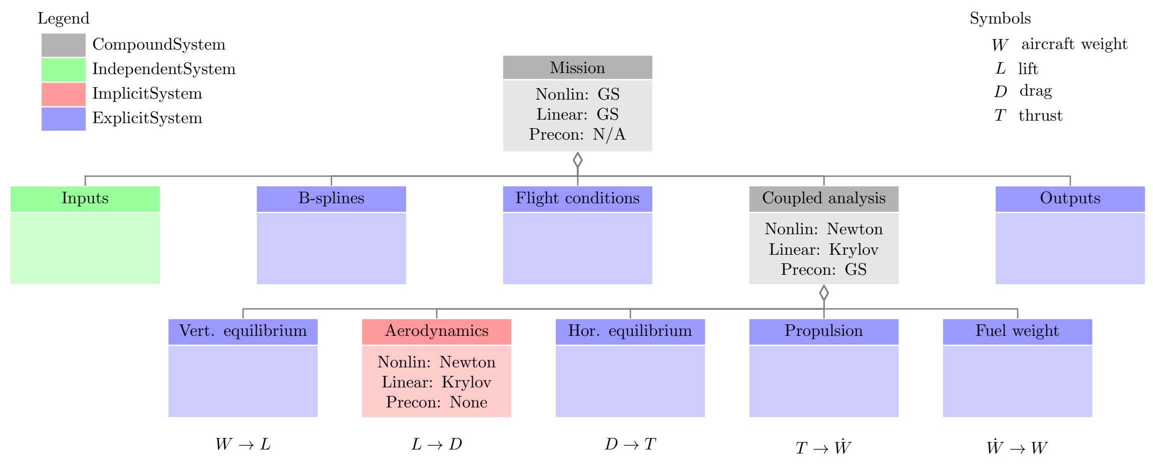


Figure 7.1: A class diagram showing the *System* instances in the aircraft mission problem.

where SFC is the thrust-specific fuel consumption, and C_T is the thrust coefficient.

As with the design of the CubeSat, this problem involves multiple disciplines that are coupled. The rate of fuel burn at a given point during the flight depends on the engine output, which is determined by the amount of aerodynamic drag that the engines must overcome. However, the amount of drag depends on how much lift must be produced to counteract the weight of the aircraft, which in turn is affected by the total amount of fuel required, completing the feedback loop.

Figure 7.1 shows a class diagram containing the components and solvers used in the computational model. The top-level *System* object, *Mission*, contains five systems that have no feedback, so nonlinear and linear block Gauss–Seidel converge in one iteration. One of the five subsystems, *Coupled analysis*, contains the aforementioned feedback loop involving fuel burn, thrust, drag, lift, and weight. For this system, Newton's method is used to solve the nonlinear system with a few iterations of nonlinear block Gauss–Seidel used as a start-up strategy, the linear systems that arise are solved using fGMRES, and the preconditioner is a few iterations of linear block Gauss–Seidel. *Coupled analysis*, in turn, contains a system that defines implicit state variables, so it also uses a Newton–Krylov solver.

The mission-only optimization is a *nonlinear programming (NLP)* problem with the objective of minimizing fuel burn and is stated below (NLP-m). The design variables are the B-spline control points of the parametrized altitude profile with n_{cp} B-spline control points and n_{pt} discretization points. Here, we use $n_{cp} = 50$ and $n_{pt} = 250$. The initial and final altitudes are constrained to be zero, and slope constraints are imposed on the altitude profile with a value of 35° for both climbing and descending. The throttle setting is constrained to be between idle (10%) and full (100%). The slope constraints are linearly mapped from the design variables, but the throttle setting is a nonlinear function of the design variables and state variables, so its derivatives are computed using the adjoint method. Since it is costly to solve one adjoint for each of the n_{pt} points, the throttle constraints are aggregated using Kreisselmeier–Steinhauser functions [63] to reduce to a single idle-throttle constraint and a single full-throttle constraint.

$$\boxed{\begin{array}{rlr} \text{minimize} & \text{fuel\_burn} & \\ \text{with respect to} & \text{altitude}_k \in [0, \text{max\_altitude}]\ , \qquad 1 \le k \le n_{cp} & \\ \text{subject to} & \text{altitude}_1 = 0 & \\ & \text{altitude}_{n_{cp}} = 0 & \text{(NLP-m)} \\ & \text{idle\_throttle} \le \text{throttle (KS)} & \\ & \text{throttle} \le \text{max\_throttle (KS)} & \\ & \text{min\_slope} \le \text{slope}_k \le \text{max\_slope}\ , \quad 1 \le k \le n_{pt} & \end{array}}$$

Figure 7.2 plots some of the simulation results at an optimized design point. The optimized altitude profile in Fig. 7.2 matches intuition—the aircraft climbs rapidly at a rate limited by the maximum thrust of the engines, there is a slow climb during the cruise segment, and it descends with the engines on idle to save fuel. It is more efficient for aircraft to fly as much as possible at its cruise altitude because drag is much lower there, and the slow climb is due to the fact that the optimal cruise altitude changes as the aircraft becomes lighter with fuel slowly being burned off. At a high level, the aircraft mission profile optimization algorithm is useful because given any aircraft design, range, and payload, it provides a best-case fuel burn estimate for the flight.

The aircraft mission optimization problem has a smaller size than the CubeSat problem, but its variables have a hierarchical structure with coupling at multiple levels. This problem takes advantage of the framework's support for hierarchical decomposition and solution of problems as well as the framework's built-in solvers. As with the CubeSat problem, this problem also benefits significantly from the framework's automated derivative computation capability, which computes gradients with respect to all design variables at a cost on the order of a single evaluation of the computational model. As before, the optimization problem is solved using SNOPT [11] via the pyOpt interface [9], and the convergence history for a representative mission optimization with $n_{cp} = 100$ design variables is shown in Fig. 7.3. More details on

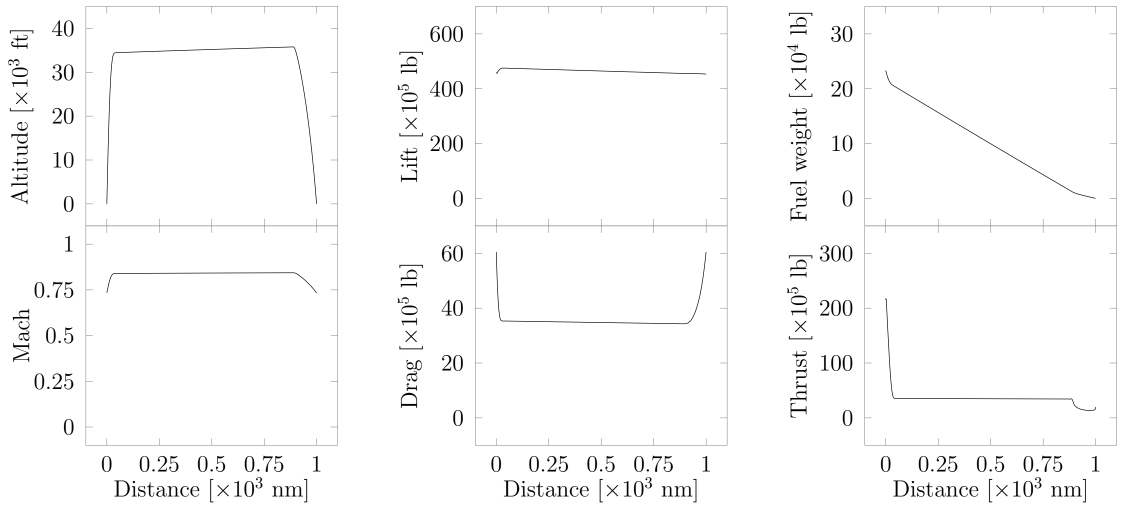


Figure 7.2: Plots of a subset of the variables modeled in the aircraft mission optimization problem. The altitude profile is designed by the optimizer to minimize total fuel burn over the mission. The distances are given in nautical miles.

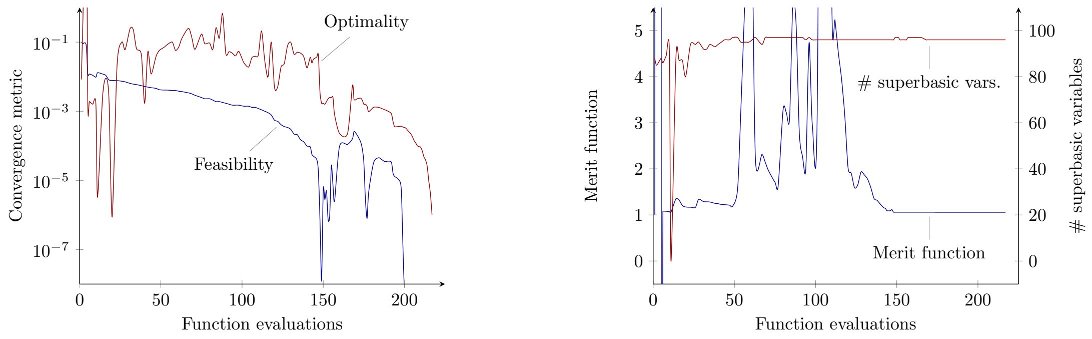


Figure 7.3: Convergence plots for a representative aircraft mission optimization.

this problem can be found in Kao et al. [91].

7.4.2 The allocation problem

The allocation problem mimics the motivation of a profit-seeking airline. Given n_{rt} routes and n_{ac} types of aircraft, the allocation problem we wish to solve seeks to maximize profit by optimally allocating the available aircraft to the routes with constraints on the number of each aircraft type available as well as the demand for each. There are 3 equations relevant to this problem. Here, i is used to index routes, and j is used to index types of aircraft. The design variables are $\text{pax\_flt}_{i,j}$, which is the number of passenger per flight, and $\text{flt\_day}_{i,j}$, which is the number of flights per day. Both design variables are arrays over routes and types of aircraft.

The first is profit, which is given by

$$\begin{aligned}\text{profit} = &\sum_{i}^{n_{rt}}\sum_{j}^{n_{ac}}\left[\text{price\_pax}_{i,j}\cdot\text{pax\_flt}_{i,j}\cdot\text{flt\_day}_{i,j}\right] && (7.11)\\ &-\sum_{i}^{n_{rt}}\sum_{j}^{n_{ac}}\left[(\text{cost\_flt}_{i,j}+\text{cost\_fuel}\cdot\text{fuel\_flt}_{i,j})\cdot\text{flt\_day}_{i,j}\right], && (7.12)\end{aligned}$$

where $\text{price\_pax}_{i,j}$ is the ticket price per flight, $\text{cost\_flt}_{i,j}$ is the total cost of operating a flight minus fuel, cost_fuel is the cost per unit fuel, and fuel_flt is the total fuel burn on a flight. In some cases, $\text{fuel\_flt}_{i,j}$ is assumed to be a constant during an allocation optimization, and in others, it is the output of the mission analysis.

The second equation is an inequality that enforces that the total number of passengers flown on a route is less than the demand on that route, and it is given by

$$\text{total\_pax}_i = \sum_{j}^{n_{ac}}\left[\text{pax\_flt}_{i,j}\cdot\text{flt\_day}_{i,j}\right] \le \text{demand}_i\ , \quad 1\le i\le n_{rt}, \tag{7.13}$$

where demand_i is the total number of passengers who are looking to fly on a given route. This inequality is enforced for each route i.

The final equation is another inequality that constrains aircraft operation time based on the total number of available aircraft for a given type, and it is given by

$$\begin{aligned}\text{total\_usage}_j = &\sum_{i}^{n_{rt}}\left[\text{flt\_day}_{i,j}\cdot(\text{time\_flt}_{i,j}(1+\text{maint}_j)+\text{turn\_flt})\right]\\ &\le 12\text{hr}\cdot\text{num\_ac}_j\ , \quad 1\le j\le n_{ac}, && (7.14)\end{aligned}$$

where $\text{time\_flt}_{i,j}$ is the block time for a flight, maint_j is the maintenance time required as a multiple of block time, turn_flt is the turnaround time between flights, and num_ac_j is the number of aircraft available for a given type. As with block fuel, block time is either a constant during optimization, or it is computed by the mission analysis. This inequality is enforced for each aircraft type j.

The allocation-only optimization problem is a *mixed-integer linear programming (MILP)* problem and is stated below (MILP-a). In this case, block fuel and block time are constants, and they are not dynamically re-computed during optimization by running mission analysis. The objective function is profit, and the two integer design variables are passengers per flight and flights per day for each route and type of aircraft. Demand constraints are enforced for each route, and aircraft availability constraints are enforced for each type of aircraft. Here, the demand is assumed symmetric and the model assumes that each aircraft type makes a round trip on each route. This partially addresses the scheduling-related constraints.

$$\begin{array}{rlr}
\text{maximize} & \text{profit} & \\
\text{with respect to} & \text{pax\_flt}_{i,j} \in [0, \text{ac\_capacity}_j] \cap \mathbb{N} \; , & 1 \le i \le n_{rt} \; , \; 1 \le j \le n_{ac} \\
 & \text{flt\_day}_{i,j} \in [0, \infty) \cap \mathbb{N} \; , & 1 \le i \le n_{rt} \; , \; 1 \le j \le n_{ac} \\
\text{subject to} & \text{total\_pax}_i \le \text{demand}_i \; , & 1 \le i \le n_{rt} \\
 & \text{total\_usage}_j \le \text{num\_ac}_j \; , & 1 \le j \le n_{ac}
\end{array} \qquad \text{(MILP-a)}$$

7.4.3 The allocation-mission problem

The main optimization problem we wish to solve is the simultaneous allocation-mission optimization with profit as the objective function. This top-level problem includes the allocation problem as well as the mission problems for each route and each type of aircraft. However, mission analyses are only run for only the new types of aircraft. The rationale is that in future work with the simultaneous design-mission-allocation optimization, the existing types of aircraft have fixed designs so the block fuel and block time only depend on the routes, which are fixed, and the weight of the passengers, which does not have a large influence. In contrast, the new types of aircraft would be simultaneously designed during the design-mission-allocation optimization, so it is necessary to re-compute the mission analyses dynamically during optimization. In light of this, let us call the number of new types of aircraft n_{nac}.

The simultaneous allocation-mission optimization is a *mixed-integer nonlinear programming (MINLP)* problem and is stated below (MINLP-a-m). To lower the difficulty of the problem, the number of passengers per flight can be relaxed as a continuous variable; however, the same assumption cannot be made for the number of flights per day because the small values signify that relaxing the integer constraints would significantly change the optimization problem.

$$
\begin{array}{rlr}
\text{maximize} & \text{profit} & \\
\text{with respect to} & \text{pax\_flt}_{i,j} \in [0, \text{ac\_capacity}_j] \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{ac} \\
& \text{flt\_day}_{i,j} \in [0, \infty) \cap \mathbb{N} \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{ac} \\
& \text{altitude}_{i,j,k} \in [0, \text{max\_altitude}] \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{nac} \\
& & 1 \le k \le n_{cp} \\
\text{subject to} & \text{total\_pax}_i \le \text{demand}_i \ , & 1 \le i \le n_{rt} \\
& \text{total\_usage}_j \le \text{num\_ac}_j \ , & 1 \le j \le n_{ac} \\
& \text{altitude}_{i,j,1} = 0 \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{nac} \\
& \text{altitude}_{i,j,n_{cp}} = 0 \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{nac} \\
& \text{idle\_throttle} \le \text{throttle}_{i,j} (\text{KS}) \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{nac} \\
& \text{throttle}_{i,j} \le \text{max\_throttle} \ (\text{KS}) \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{nac} \\
& \text{min\_slope} \le \text{slope}_{i,j,k} \le \text{max\_slope} \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{nac} \\
& & 1 \le k \le n_{pt}
\end{array}
\tag{MINLP-a-m}
$$

The MINLP problem is solved using the branch-and-bound method [8] to formulate a series of continuous NLP problems that eventually generate an integer solution, and the continuous problem is shown below (NLP-a-m). The continuous NLP problems are solved using SNOPT [92], using a modified version of the pyOpt interface [9]. SNOPT is a gradient-based optimizer that implements the sequential quadratic programming (SQP) method, and it is specifically designed for nonlinear constrained optimization problems that are large-scale and sparse.

$$
\begin{array}{rlr}
\text{maximize} & \text{profit} & \\
\text{with respect to} & \text{pax\_flt}_{i,j} \in [0, \text{ac\_capacity}_j] \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{ac} \\
& \text{flt\_day}_{i,j} \in [0, \infty) \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{ac} \\
& \text{altitude}_{i,j,k} \in [0, \text{max\_altitude}] \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{nac} \\
& & 1 \le k \le n_{cp} \\
\text{subject to} & \text{total\_pax}_i \le \text{demand}_i \ , & 1 \le i \le n_{rt} \\
& \text{total\_usage}_j \le \text{num\_ac}_j \ , & 1 \le j \le n_{ac} \\
& \text{altitude}_{i,j,1} = 0 \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{nac} \\
& \text{altitude}_{i,j,n_{cp}} = 0 \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{nac} \\
& \text{idle\_throttle} \le \text{throttle}_{i,j} (\text{KS}) \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{nac} \\
& \text{throttle}_{i,j} \le \text{max\_throttle} \ (\text{KS}) \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{nac} \\
& \text{min\_slope} \le \text{slope}_{i,j,k} \le \text{max\_slope} \ , & 1 \le i \le n_{rt} \ , \ 1 \le j \le n_{nac} \\
& & 1 \le k \le n_{pt}
\end{array}
\tag{NLP-a-m}
$$

The overall algorithm that solves (MINLP-a-m) is described in Algorithm 1. The first step is to solve (NLP-m) for each route and each aircraft type using SNOPT in order to determine the block times and block fuels to be used in the next step. We use the *linprog* function in *SciPy*'s *optimize* package via NASA's OpenMDAO framework [93] to solve (MILP-a) to generate a good initial solution for the branch-and-bound algorithm. Next, we begin the branch-and-bound phase by initializing the tree and adding the first node, which is (NLP-a-m) that uses the initial values for the allocation design variables from the solution to (MILP-a) and the initial values for the mission design variables from the solution to (NLP-m) for each mission. The remainder of the algorithm implements the depth-first branch-and-bound algorithm. An important characteristic of this algorithm is that when branching for a variable x_k, all of the initial values from the parent node are used, except for x_k itself, which is taken to be the same as the upper or lower bound. This allows (NLP-a-m) to be solved in one or a very small number of NLP iterations in many cases, yielding improved efficiency.

7.5 Allocation-mission optimization results

This section presents a suite of allocation-mission optimization results obtained using Algorithm 1. We start by describing the routes and the types of aircraft in the problem we solved, and present 4 results. First, we show the predicted increase in profit for an airline if it purchases new aircraft instead of existing aircraft. Second, we show that there is a difference between the results of allocation-only optimization (MILP-a) and allocation-mission optimization (MINLP-a-m). Next, we explore the presence of local optima in (NLP-a-m) and show that (MINLP-a-m) is highly sensitive to the starting point. Finally, we discuss the numerical performance of the allocation-mission analysis and optimization algorithm.

7.5.1 The problem

The optimization results presented here reflect a 3-route problem with ranges of roughly 7000 nmi, 5500 nmi, and 2500 nmi. The routes have been chosen to represent a network with a hub in Newark, New Jersey and the following destinations: Hong Kong; Kuwait City, Kuwait; and Quito, Ecuador. The routes are summarized in Tab. 7.1 and shown graphically in Fig. 7.4.

We consider 6 aircraft types, 4 existing ones and 2 hypothetical next-generation

Algorithm 1 Solution of the allocation-mission optimization problem, (MINLP-a-m)

```
1: for $i \leftarrow 1, n_{rt}$ do
2:     for $j \leftarrow 1, n_{ac}$ do
3:         Solve (NLP-m) to determine block fuel and time for the $(i, j)$th mission
4:     end for
5: end for
6: Solve (MILP-a)
7: Create the branch and bound tree
8: Add the first node for an (NLP-a-m) starting from (MILP-a) and (NLP-m) solutions
9: while Nodes exist do
10:     Solve (NLP-a-m) in the deepest node (depth-first strategy)
11:     if Solution is better than the best integer solution then
12:         if Solution is integer then
13:             Store solution—this is the new solution
14:         else
15:             $k \leftarrow$ index of integer variable furthest from an integer
16:             Add a node for an (NLP-a-m) with upper bound and initial value of $x_k$ changed to $floor(x_k)$
17:             Add a node for an (NLP-a-m) with lower bound and initial value of $x_k$ changed to $ceiling(x_k)$
18:         end if
19:     end if
20:     Delete current node
21: end while
```

Route	Newark (EWR) Hong Kong (HKG)	Newark (EWR) Kuwait City (KWI)	Newark (EWR) Quito (IQT)
Range [nmi]	6998	5546	2509
Demand	1200	550	700

Table 7.1: The 3 routes considered in the allocation-mission optimization

.

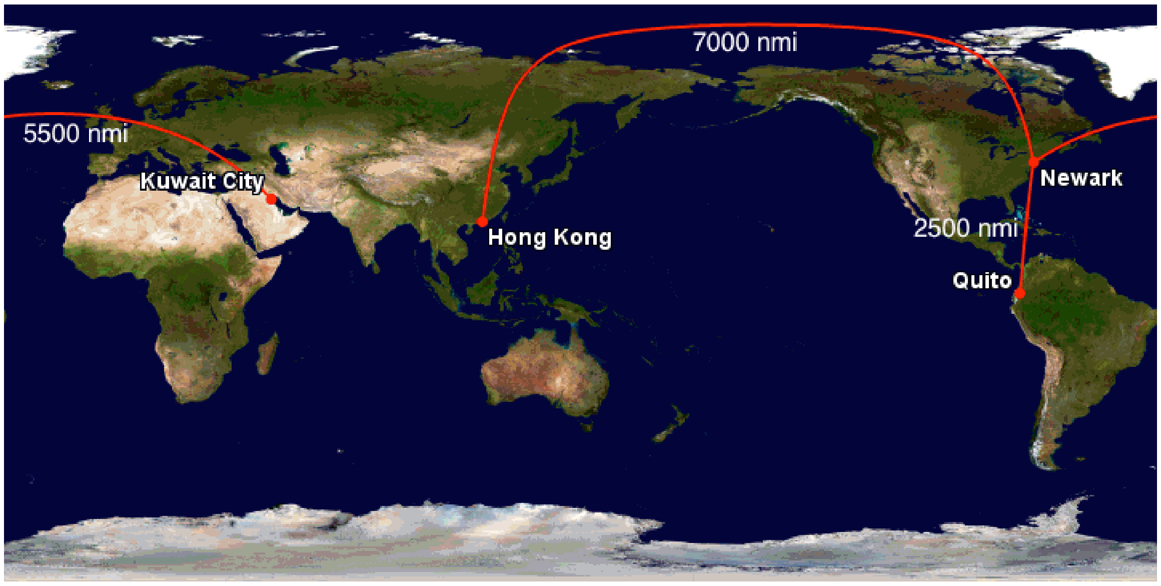


Figure 7.4: Map of the cities and the routes (generated using the Great Circle Mapper: www.gcmap.com). Newark, New Jersey is chosen as the hub.

aircraft. The Boeing 737-800 (B737), Boeing 777-200ER (B777), Boeing 747 (B747), and Boeing 787 (B787) have been chosen as the existing aircraft to cover a variety of design ranges and seating capacities. The two new aircraft are a notional advanced conventional design based on the Common Research Model [94] and a blended wing body concept based on Liebeck [95] with a reduced seating capacity. The aircraft types are summarized in Tab. 7.2 with seating capacities shown after applying an 80 % load factor. Table 7.2 also shows the 4 allocation-mission optimization problems we solved, representing different scenarios in which the hypothetical airline chooses to buy different aircraft.

The cost, ticket price and the performance data of the existing aircraft for the different routes in the network are obtained using the simulation tool FLEET [96, 97]. For the new aircraft, the ticket price, no-fuel direct operating cost and the indirect operating cost are also obtained from an equivalent aircraft modeled in FLEET. The current model do not account for airline competition and assumes the ticket prices are fixed across types of aircraft on a given route.

7.5.2 Profit increase with new aircraft

Figure 7.5 shows the profit after optimization for each of the 4 scenarios. The results agree with intuition because the CRM and BWB both represent an improvement

Aircraft	Boeing 737-800	Boeing 777-200ER	Boeing 747-400	Boeing 787-8	CRM	BWB
Category	Existing	Existing	Existing	Existing	New	New
Capacity	122	207	294	200	300	400
Scenario						
S-base	20	24	24	8		
S-CRM	20	24	24		8	
S-BWB	20	24	24			8
S-both	20	20	20		8	8

Table 7.2: The types of aircraft considered in the allocation-mission optimization. The bottom four lines show the number of each aircraft type available in each of the four scenarios.

over the existing aircraft. The CRM design is based on the B777, but it is assumed to have a larger seating capacity and higher aerodynamic efficiency since it is a next generation aircraft. The B787 has the range of the CRM but a lower seating capacity, while the B747 has the seating capacity but a lower range and lower efficiency as it is a much older design. Thus, the S-CRM scenario provides a 192 % improvement in profit over the baseline S-base, the S-BWB scenario provides a 323 % due to its larger seating capacity, and the S-both scenario provides a further improvement with 414 % compared to the baseline.

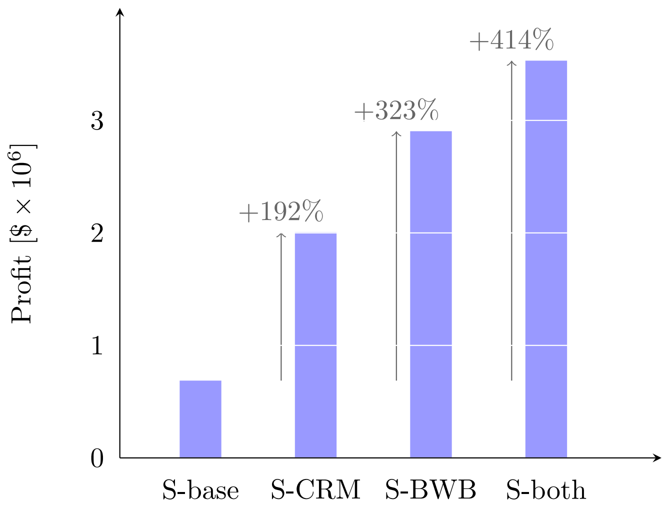


Figure 7.5: Comparison of profit for the four scenarios at the solution to (MINLP-a-m). The results show an increase in profit when the hypothetical airline purchases the next-generation aircraft.

7.5.3 Allocation versus allocation-mission optimization

Figure 7.6 compares the optimized profit among 4 cases: (MINLP-a-m) without the (MILP-a) initialization, (MILP-a), (MINLP-a-m), and (NLP-a-m). This subsection focuses on (MILP-a) versus (MINLP-a-m)—i.e., allocation-only optimization versus allocation-mission optimization. We see a small increase in profit in all three scenarios with the allocation-mission optimization because it more accurately computes the block fuel and block time for the new aircraft. In particular, the block fuel values used in (MILP-a) assume the aircraft always fly at full capacity—thus, (MILP-a) likely underpredicts profit because the actual fuel burn is lower when the aircraft flies with fewer passengers than its seating capacity. However, this is a minor difference, and the reader is reminded that the true motivation for incorporating mission analysis in the allocation problem is to enable design-mission-allocation optimization, and this work develops many of the methods needed for optimization with all three simultaneously considered.

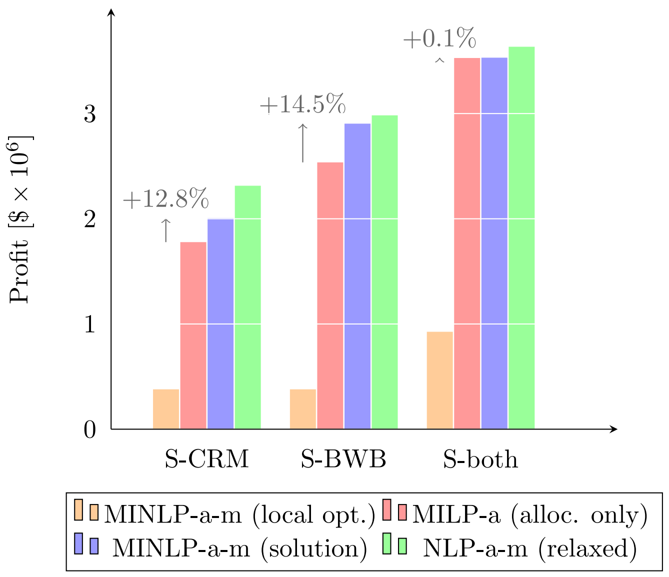


Figure 7.6: Comparison of optimized profits. The colors represent, in order: (MINLP-a-m) not initialized to the solution to (MILP-a) (orange), (MILP-a) (red), (MINLP-a-m) initialized to the solution to (MILP-a) (blue), and (NLP-a-m) solved in the first node of the branch-and-bound algorithm. The results show clear local minima and an increase in profit in the allocation-mission optimization compared to the allocation-only optimization.

7.5.4 Local optima

One significant concern when incorporating mission analysis into the allocation problem is that the problem becomes nonlinear, so there is potential for local optima that represent significantly worse profit than the global optimum. Figure 7.6 shows that there is indeed a large difference between two solutions to (MINLP-a-m): one in which Algorithm 1 is run as is and one in which Algorithm 1 is run without the (MILP-a) initialization (line 6). In Fig. 7.6, the orange bars are several times smaller than the blue bars for each scenario. Figure 7.7 confirms that the integer design variables—flights per day—consistently show large changes between the two optima.

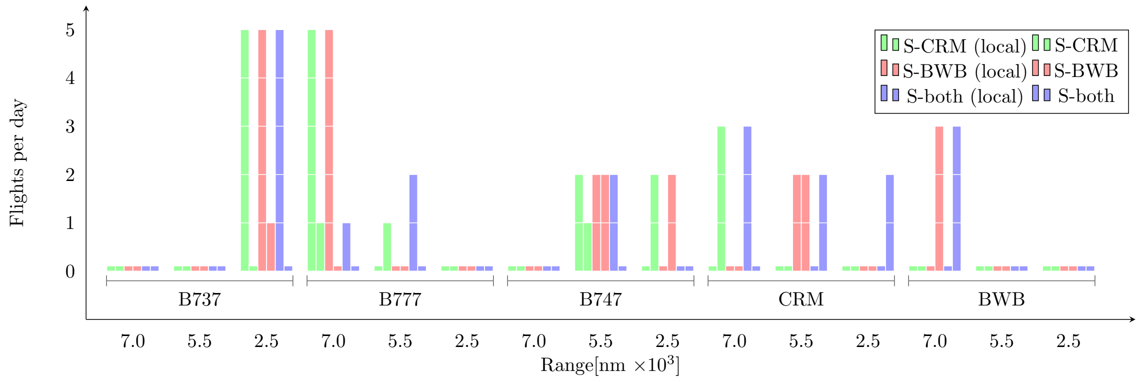


Figure 7.7: Optimal values of the (round trip) flights per day design variable comparing (MINLP-a-m) with and without (local) initialization to the solution to (MILP-a). The results show two distinct local minima in (MINLP-a-m).

These results highlight two aspects to the importance of the (MILP-a) initialization step (line 6) in Algorithm 1. First, it is not possible to guarantee convergence to a global minimum in nonlinear optimization problems that do not satisfy any convexity conditions, but Algorithm 1 provides a reasonably good local optimum by starting from the global optimum of the linear problem, (MILP-a). The initial (NLP-a-m) and the branch-and-bound method then guarantee that the solution to which it converges is no worse than the initial point provided by the linear optimization.

The second benefit is the tremendous efficiency afforded by this two-step approach. The linear optimization (MILP-a) is inexpensive, so it quickly gets close to the solution of the nonlinear optimization (MINLP-a-m), after which branch-and-bound begins from a good starting point. Fig. 7.8 shows that the continuous solution of the initial (NLP-a-m) solved in the first node of the branch-and-bound is not significantly different from the solution of (MILP-a), and the solution of (MINLP-a-m) is in turn very similar as well. In fact, the branch-and-bound only forces the integer variables to one of the nearest integers, suggesting that the (NLP-a-m) problem is locally convex in a large enough area containing the nearby integers, but multi-modal over a larger region since at least one other local optimum was found.

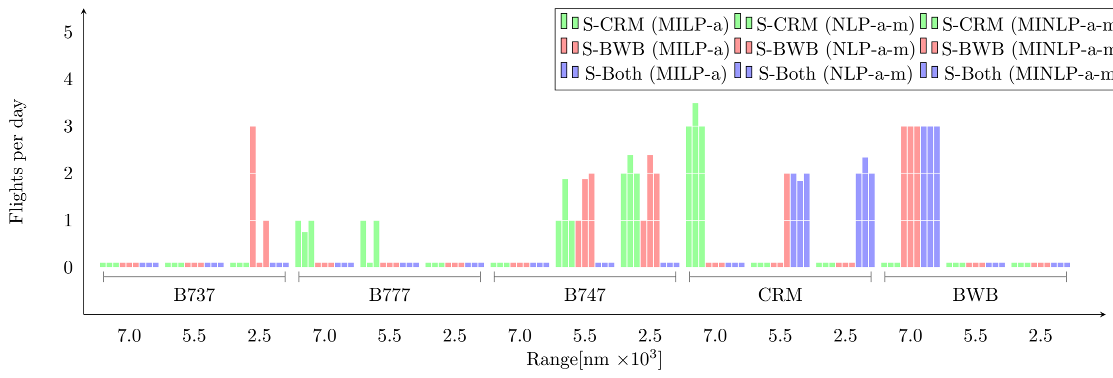


Figure 7.8: Optimal values of the (round trip) flights per day design variable comparing the allocation-only solution (MILP-a), the relaxed allocation-mission solution (NLP-a-m), and the integer allocation-mission solution (MINLP-a-m). The results show that the relaxed simultaneous solution is close to the allocation-only solution and the integer simultaneous solution is close to the relaxed simultaneous solution.

This observation is significant from a computational standpoint because the branch-and-bound method converges in a small number of iterations. For this 3-route problem, all cases converged in 5-20 node evaluations, and initial studies of 5-route problems show the same results. Furthermore, many of the (NLP-a-m) problems solved during branch-and-bound converge in one iteration or a small number of iterations because the initial point is often the solution.

7.5.5 Numerical performance

Figure 7.9 shows the convergence history for the solution of (NLP-a-m) with 3 routes and 2 new aircraft, resulting in 330 design variables in total. The fact that we can achieve such a high level of convergence provides confidence that the derivatives are computed accurately and the design variables and constraints are properly scaled. Table 7.3 shows the times for solving (MINLP-a-m) for the various cases.

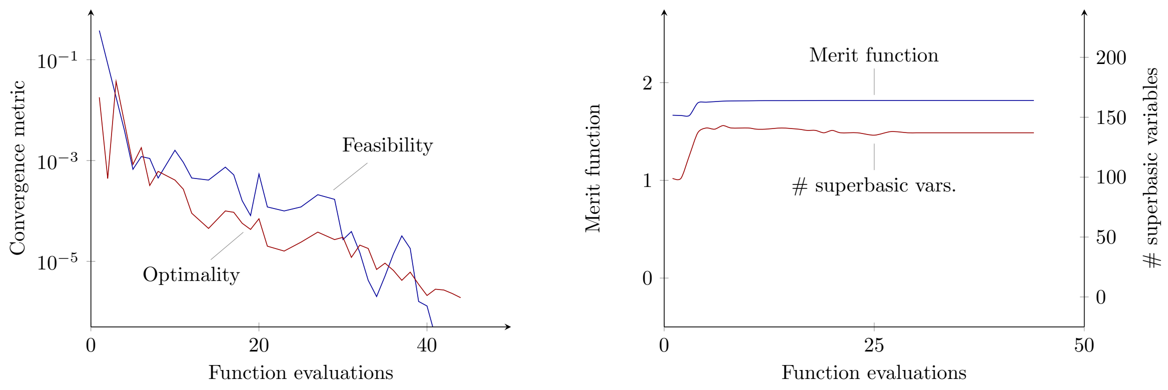


Figure 7.9: Optimization convergence history for a 3-route, 2-new aircraft problem with 330 design variables. The tight convergence of optimality and feasibility helps verify the accuracy of the derivative computation.

Case	Time [min.]
S-CRM (local)	7.7
S-CRM	20.4
S-BWB (local)	9.0
S-BWB	20.5
S-both (local)	33.3
S-both	40.6

Table 7.3: The execution times for the solution of (MINLP-a-m).

7.6 Summary

This chapter presented an algorithm for performing simultaneous allocation-mission optimization. The algorithm was applied to the allocation of 4 existing aircraft and 2 next-generation aircraft to a 3-route network and three conclusions were made.

First, the algorithm found significant profit increases if the airline chooses to purchase the next-generation aircraft. Compared to a baseline, purchasing 8 advanced conventional aircraft yielded a 192 % increase in profit, purchasing 8 blended wing body aircraft yielded a 323 % increase in profit, and purchasing 8 of each yielded a 414 % increase in profit with the optimal allocation.

Second, the allocation-mission optimization produced a higher profit than allocation-only optimization. Part of the difference is attributed to the fact that block fuel is computed dynamically during optimization using the allocated number of passengers, rather than pre-computing block fuel assuming full capacity. However, this error is minor, and the true benefit of simultaneously considering mission during the allocation problem is that this enables also incorporating design in the simultaneous optimization in future work.

Third, the algorithm found multiple local optima in the mixed-integer allocation-mission problem, and they yielded profit numbers that are 2-3 times apart. However, it was found that the algorithm efficiently finds a good local optimum by using an allocation-only optimization to find a good starting point. This is advantageous because linear programming has a low computational cost and it finds the global optimum. In practice, the solution to the first relaxed allocation-mission optimization during branch-and-bound deviates a small amount from the linear solution and the branch-and-bound method, in turn, finds one of the neighboring integer points as the solution. The significance of this is that many nodes in the branch-and-bound algorithm converge very quickly during the continuous optimization.

The significance of this work is the demonstration of simultaneous allocation-mission optimization of commercial aircraft using an approach that naturally extends to design-mission-allocation optimization. Using the framework provided two key benefits: its built-in hierarchical nonlinear and linear solvers, and its automatic implementation of the adjoint method. Moreover, the framework enables a modular approach for integrating the aerodynamic and propulsion surrogates, the flight equilibrium equations, and the fuel weight equation, all computed across multiple routes and types of aircraft, but combined into a single airline profit maximization problem.

CHAPTER 8

Conclusions

8.1 Summary and contributions

In Part I of this thesis, I presented the mathematical foundation, algorithmic design, and applications for a general computational modeling framework that automatically solves multidisciplinary systems and computes their derivatives using analytic methods. I formulated the general computational model as a system of algebraic equations, and applied the inverse function theorem to develop an equation that unifies all methods for computing derivatives. I then described the algorithmic details of the framework, and presented two engineering applications to demonstrate the effectiveness of the framework in enabling large-scale optimization.

My contributions are as follows:

- *I proposed a unified mathematical formulation of computational models as a system of equations.*

  A single vector of variables is defined, concatenating all types of variables—input, output, state, intermediate, parameter, design, objective, etc. For each variable, its value is implicitly defined by a corresponding residual function that takes as arguments all variables in the concatenated vector. When all the residual functions are combined, the system of equations that results represents the mathematical formulation of the computational model.

- *Based on this mathematical formulation, I developed an equation that unifies all discrete methods for computing derivatives.*

  With the right choice of variables and residuals in the mathematical formulation, any of the following methods can be derived from the unified equation—

monolithic differentiation, algorithmic differentiation, the direct and adjoint methods, and the chain rule.

- *Using the mathematical formulation and the derivatives unification equation, I developed a compact, minimalistic computational modeling framework that automatically solves coupled systems and computes their derivatives.*

  The framework hierarchically decomposes the variables in the problem, and applies hierarchical solution algorithms. The user defines variables in *System* objects that are grouped by other *System* objects which can in turn be part of other *System* objects and so on. The *System* object contains an API with only 4 major operations that are sufficient to interface to built-in nonlinear and linear solvers. The framework-provided *System* objects automatically perform parallel data transfer between contained *System* objects.

- *I implemented a large-scale optimization of a nanosallite within the framework, considering 7 disciplines simultaneously.*

  Based on data and equations provided by collaborators, I implemented the models in a modular way to enable gradient-based optimization using the framework. To the best of my knowledge, this is the first application of MDO to the full satellite problem considering all major disciplines and all relevant design variables simultaneously.

- *Through the nanosatellite problem, I demonstrated that an engineering design problem with many discontinuities and discrete data can be solved with gradient-based optimization.*

  I developed a multi-dimensional B-spline interpolant to develop smooth models from discrete data, and I artificially smoothed discontinuities in other models. A potential concern is that this may add local minima to the problem, but this was shown not to be the case.

- *Through the nanosatellite problem, I demonstrated that it can be effective to remove coupling from the computational model by treating the coupled variables as design variables, allowing the optimizer to converge the coupling.*

  Instead of using predetermined state variables to satisfy the power constraints, the optimizer had the freedom to optimally distribute the available power among the attitude-control actuator, communication gain, and scientific instruments, while satisfying the battery constraints. This yielded a decoupled computational model—the components can be executed in sequence with no feedback to converge.

- *I formulated and demonstrated a simultaneous optimization of allocation and mission for commercial aircraft.*

  Allocation optimization and mission profile optimization have been performed separately in the past. However, simultaneous optimization allows the aircraft design to be varied, enabling aircraft design optimization with airline profit as the objective function, considering its performance on multiple routes simultaneously. Together with collaborators, I formulated and implemented allocation-mission optimization taking advantage of the framework's built-in hierarchical solvers.

8.2 Significance

8.2.1 The unification of derivative computation methods

There are three aspects to the significance of this unification. First, it is a theoretical result that can aid in understanding methods. The unification reveals that methods that seem to be completely unrelated all have a common origin, which can be shown rigorously. Moreover, the unification provides the insight that the differences between methods for computing derivatives reflect a different level of decomposition of the computational model. For instance, including the variable from every line of code in the algebraic system yields algorithmic differentiation, while including only the input, output, and state variables yields the analytic methods.

Second, this unification provides the basis for a rigorous definition of the total derivative in the context of coupled systems. Total derivatives are often recursively defined in terms of other total derivatives using the chain rule. As part of the unification, the total derivative can be formally defined as the Jacobian of partial derivatives of an inverse function. This definition can help eliminate ambiguities in the usage of total derivatives, and provide a rigorous interpretation when the meaning is not clear.

An example is the derivative of output f_i with respect to input x_j. In a general context, this is typically written as the partial derivative $\partial f_i / \partial x_j$, but in the context of the adjoint method, the intended term is the total derivative df_i/dx_j. The monolithic formulation and the unifying chain rule equation provide the explanation that in the general context, the state variables are not included in the algebraic system so the total and partial derivatives are equal. In contrast, in the context of the adjoint method, the state variables are included so the total and partial derivatives are not equal.

Third, the unification provides homogeneity that greatly simplifies the implementation of the computational modeling framework. The automatic derivative computation feature is implemented simply by adding a linear system to solve, as opposed to *ad hoc* implementations of each type of method. Moreover, the user is not required to manually specify a method—based on the type of variables and residuals they define, the linear system naturally reduces to one of the methods for computing derivatives.

8.2.2 The computational modeling framework

The primary benefit of the framework is that it greatly shortens the development time for large-scale optimization. Its value is in the modularity it provides with minimal overhead, built-in linear and nonlinear solvers, and automatic derivative computation.

The mathematical theory, algorithmic design, and implementation of the framework have been adopted by and integrated into NASA's OpenMDAO framework. This framework design has been used for several problems including the nanosatellite optimization presented in Ch. 6, the aircraft allocation-mission optimization presented in Ch. 7, wind turbine design optimization [98], and RC aircraft design optimization.

8.3 Recommendations

There are four recommendations for future work pertaining to the framework. The unified derivatives equations could be extended to higher-order derivatives. The unified derivatives equation yields derivatives of any quantity, so we could choose to append first derivatives to the vector of variables and the expressions that define them to the vector of constraints. The solution of the resulting linear system would simultaneously yield first and second derivatives, which could in turn be selected as additional variables to compute third derivatives. This technique could be repeatedly applied to obtain expressions for derivatives of any order, though it remains to be seen how efficient such an approach could be and which method—AD, analytic, etc.—could and should be used. At each of the n steps for nth order derivatives, the forward or reverse form must be selected, yielding various combinations such as the direct-adjoint and adjoint-direct methods for second derivatives.

The framework could be extended to stochastic processes. Given probability distributions for the input variables and the functions and residuals defining all other variables, probabilities for the output variables could be computed. The unknown vector and the residual vector could be treated as random vectors, and the transformation of random vectors formula could be applied if the determinant of the Jacobian can be efficiently approximated.

It would be useful to develop an algorithm that automatically defines the hierarchy tree in the framework. This algorithm could potentially use the dependency graph of the variables, Jacobian information, and estimates for computation time for each variable to automatically determine efficient ways to hierarchically group and solve for the variables.

The unsteady adjoint method could be implemented in the framework. This would include all time instances of spatially distributed quantities as variables in the framework. Since the framework takes a matrix-free approach, the memory requirements would be on the same order as an *ad hoc* implementation of the unsteady adjoint method. However, the framework would offer the advantage that the linear systems are automatically assembled and solved.

Part II

A differentiable parametrization of aircraft geometries and structures

CHAPTER 9

The need for a new parametrization

In Part II of this thesis, I address the problem of parametrizing aircraft geometries and structures in a differentiable way, as required for high-fidelity shape optimization. Chapter 10 presents the methodology for the geometry parametrization, highlighting how it maintains differentiability unlike existing geometry tools. In Ch. 11, I describe how the aircraft structure is parametrized in relation to the geometry and present the structural mesh creation process that uses a novel unstructured quadrilateral mesh generation algorithm. As a demonstration, Ch. 12 presents aerostructural analysis results for two unconventional configurations and aerodynamic shape optimization of the truss-braced wing configuration. Finally, Ch. 13 provides a summary of contributions and recommendations from Part II.

The current chapter provides a motivation and overview for Part II of this thesis. Section 9.1 begins by discussing the current concerns over the environmental footprint of commercial aviation, and describes several promising unconventional configurations that have the potential to address these concerns. Section 9.2 introduces high-fidelity aerostructural optimization as a design tool that is useful especially for unconventional configurations. An enabling tool for this is a differentiable parametrization for aircraft geometry and structures, and Sec. 9.3 outlines the requirements for such a parametrization. Finally, Sec. 9.4 introduces GeoMACH, which is an open-source aircraft parametrization tool suite developed to meet these requirements.

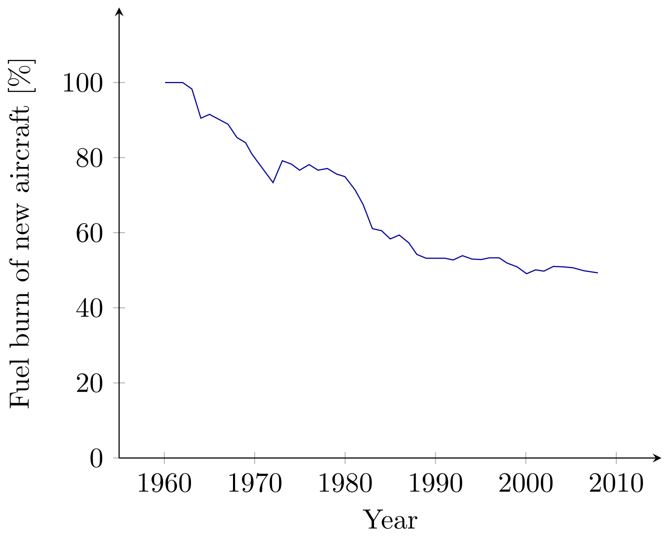


Figure 9.1: The fuel burn per passenger per unit distance of new aircraft over time, as a percentage of the value in 1960. The last 5 decades have seen a 50 % reduction, but progress has stalled in the last 2 decades. Source: the International Council on Clean Transportation [2].

9.1 Unconventional aircraft configurations

Over the last five decades, the fuel efficiency of commercial aircraft has approximately doubled through advancements in technology and design [2]. However, as Fig. 9.1 shows, this trend has stagnated in recent years, as each aircraft becomes more optimized and further improvements become more technically challenging to achieve. This concern is compounded by the fact that air traffic growth is expected to outpace efficiency improvements in the next two decades [99], in the face of rising fuel costs and growing environmental concerns.

The National Aeronautics and Space Administration (NASA) has identified the need for new concepts for commercial aircraft that significantly improve efficiency and environmental compatibility. To this end, NASA has outlined aggressive targets for the N+3 (2025) generation of aircraft, including a 71 dB noise reduction, an 80 % reduction in NOx emissions, and a 60 % reduction in fuel consumption over a mission [100]. These reduction targets are relative to a baseline represented by the Boeing 737-800, which was first flown in 1997. One approach to achieving these target metrics is through discipline-specific improvements such as increased use of composites for reduced structural weight, higher bypass ratios (BPR) for increased propulsive efficiency, and improved high-lift systems for reduced noise and drag. On the other hand, Figure 9.1 suggests that the 50-year-old tube-and-wing design may

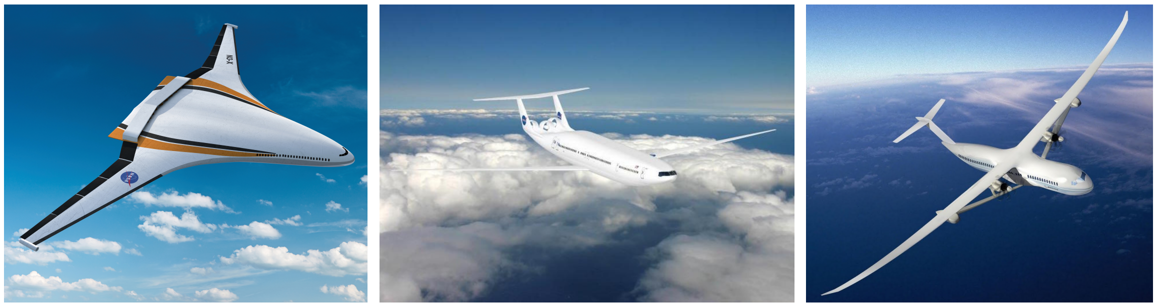


Figure 9.2: Renderings for NASA's HWB (left), MIT's D8 (center), and Boeing's TBW (right) concepts. Source: NASA Aeronautics Research Mission Directorate.

need to be significantly modified to achieve the breakthrough reductions in fuel burn, emissions, and noise that are required for sustainability.

For subsonic commercial aviation, NASA has funded research in three promising unconventional aircraft configurations in particular. First, it has supported in-house research on the hybrid wing body (HWB) concept featuring distributed turboelectric propulsion. With no fuselage or empennage, the HWB can have lower overall weight and drag, and its lifting nose can help offset the potential increase in trim drag. NASA's HWB concept includes wing tip-mounted turbogenerators that power fans distributed on the leading edge of the centerbody, and this design feature provides noise shielding for the primary propulsors, a higher effective BPR, and boundary layer ingestion (BLI) on the fuselage. Second, NASA has funded the development of the double-bubble (D8) concept by a team led by the Massachusetts Institute of Technology (MIT) [101]. The wide fuselage provides extra lift, a trimming moment on the nose, and embedded engines for noise shielding and BLI. Finally, NASA has also funded the development of a hybrid-electric truss-braced wing (TBW) concept by a team led by the Boeing Company [102]. The struts of the TBW enable wings with increased spans and aspect ratios for improved aerodynamic efficiency, and the high-wing design provides room below the cabin floor for batteries. Renderings for each of these configurations are shown in Fig. 9.2.

9.2 High-fidelity aerostructural optimization

The aircraft design process has three stages, which are summarized here based on the treatment in Raymer [103]. Conceptual design involves the design of the configuration and layout through conceptual sketching, 'first-order' sizing, decisions on technologies, and simple optimization based on estimates of takeoff weight, L/D, etc. At preliminary design, the configuration is frozen and specialists perform compartmentalized analyses within each discipline. Detail design involves analysis and design of small-scale components such as ribs and spars as well as fabrication and testing.

When considering unconventional configurations during conceptual design, the role of computational modeling is increased because the tremendous amount of knowledge and experience accumulated over decades of aircraft manufacturing does not always apply. Experimental results can be more accurate, but flight testing, wind tunnel testing, and load testing are expensive and time-consuming, which puts more emphasis on computational tools. In particular, numerical optimization has the potential to achieve the advances that were not within reach in the past, particularly by rapidly exploring revolutionary designs in which prior knowledge is limited.

At a fundamental level, aircraft design improvements target either the propulsion system by lowering thrust specific fuel consumption or the airframe by reducing structural weight and aerodynamic drag. On the airframe side, computational fluid dynamics (CFD) and finite element analysis (FEA) algorithms have advanced to the point that it is possible to efficiently optimize hundreds of aerodynamic shape and structural sizing design variables simultaneously to minimize fuel burn [7]. This problem is referred to as aerostructural optimization because the loads computed via Euler-based CFD are used to compute the displacements via FEA, which are then transferred back to CFD, and this coupling is solved each optimization iteration. A gradient-based optimizer is used with gradients computed using the coupled adjoint method [18]. This approach has been extended to aerodynamic shape optimization using the Reynolds-averaged Navier–Stokes equations (RANS) [104] and aerostructural optimization with composite design variables included [65].

9.3 The requirements for a new parametrization

The objective of Part II of this thesis is to develop the algorithms needed to enable high-fidelity aerostructural optimization in a conceptual design context in which multiple aircraft configurations are considered. What is needed is a method for quickly creating aircraft geometries and structural meshes, and manipulating them using differentiable mappings.

There are many aircraft design frameworks and geometry engines available open-source or commercially, including VSP (Vehicle Sketch Pad) [105], MICADO [106], RAGE (Rapid Geometry Engine) [107], and AVID PAGE (Parametric Aircraft Geometry Engine) [108]. These geometry engines share a similar approach. In each case, aircraft components such as the wing and fuselage are individually created, typically by lofting curves, and the intersections between components are computed as a necessary step in achieving a closed surface. Often, the underlying geometry for each component is represented as parametric surfaces such as non-uniform rational B-spline (NURBS) surfaces. To produce a discrete surface representation, triangulation is required at this point as the intersections are parametric curves that delimit surface regions that are no longer exposed. In all cases, the final tessellation is in general an unstructured mesh. For high-fidelity aerodynamic or structural analysis, the next step is to create a CFD mesh or structural mesh, respectively. With VSP for instance, there are tools for automatic generation of CFD volume [109] and structural [110] meshes. If a structured multi-block CFD solver is to be used, mesh generation is necessarily a manual process but is still possible given a geometry represented by trimmed surfaces.

The aforementioned geometry engines are useful tools that facilitate analyses and small-scale optimizations at the conceptual design level. However, because they rely on intersection algorithms to produce closed surfaces, they cannot support large-scale aerostructural optimization for several reasons. First, the lack of smoothness in the geometry parametrization can cause noisy gradients and difficulties in converging the optimization problem. In addition, since the intersections must be recomputed each optimization iteration, the CFD mesh must be regenerated as mesh movement algorithms fail in the presence of topological changes in the surface mesh. Mesh regeneration is costly and is only feasible for unstructured meshes. The same applies for the structural mesh as well; gradient-based optimization is only feasible if it is possible to warp the structural mesh instead of regenerating it each optimization iteration.

The aerodynamic and aerostructural optimization algorithms cited above use free-form deformation (FFD) blocks which morph a discrete model imported from an external source [111]. This approach can be used in conjunction with one of the aforementioned geometry engines. However, the drawback is that it is difficult to parametrize the entire aircraft including the details of junctions because FFD blocks are defined separately for each part of the geometry—one for the wing, one for the fuselage, etc.

In summary, there are three requirements for the parametrization of the aircraft geometry and structure:

- Support for unconventional configurations and large shape changes

- A continuously differentiable parametrization
- Usability and automation

9.4 The solution: GeoMACH

The solution that meets the above requirements is GeoMACH: geometry-centric MDO of aircraft configurations with high fidelity. GeoMACH is an open-source aircraft parametrization tool suite funded by NASA to facilitate high-fidelity multidisciplinary design optimization. It handles the creation and parametrization of aircraft geometries and structures, and is designed to be modular so that it can easily integrate with external CFD solvers, FEA solvers, optimizers, etc.

GeoMACH consists of 3 parts: the B-spline engine (BSE), the parametric geometry modeler (PGM), and the parametric structural modeler (PSM). GeoMACH represents geometries as a watertight union of 4-sided B-spline surfaces, which are handled by BSE. PGM maps the geometric design variables to the B-spline control points, and PSM is responsible for generating the FEA mesh.

The surface nodes of the CFD volume mesh can be interpreted as a particular discretization of the B-spline model that gets mapped to the CFD volume mesh using an external mesh warping algorithm. For the structural model, the internal nodes of the FEA mesh are defined as a linear combination of points on the geometry—let us call these the FEA surface projections. These FEA surface projections represent another discretization of the B-spline model, and PSM computes the linear mapping from the FEA surface projections to the actual FEA mesh, after which an external FEA solver is called.

The process of setting up shape optimization for CFD proceeds as follows. First, the user creates the configuration by specifying the aircraft components (wing, fuselage, etc.) and how they connect to each other. Next, the user exports the initial geometry as an IGES file to externally create the CFD mesh. The surface nodes of the CFD mesh are given to BSE to run a projection algorithm and compute the parametric coordinates of each node on the B-spline surfaces. Using these parametric coordinates, BSE computes the Jacobian that maps the control points to the CFD surface mesh.

The process of setting up the structural mesh proceeds as follows. The user defines the structural members, the coordinates of which are interpolated from points on the geometry. PSM compute the B-spline parameters of the FEA surface projections; then, the B-spline parameters are used by BSE to compute the Jacobian that maps the B-spline control points to the FEA surface projections. PSM then computes the Jacobian that maps the FEA surface projections to the actual FEA mesh.

CHAPTER 10

A flexible geometry parametrization

The design of the parametric geometry modeler is driven by the high-level objectives mentioned in Sec. 9.3: the versatility to model multiple configurations, a continuously differentiable parametrization, and usability. This chapter describes how the geometry is represented in Sec. 10.1, its parametrization in Sec. 10.2, and the computation of derivatives in Sec. 10.3.

10.1 Geometry representation

To simplify representation of multiple configurations, each is decomposed into components, and an object-oriented approach is adopted as shown in Fig. 10.1. An aircraft model is an instance of the Configuration class, which inherits from the base Configuration class for all models of that particular layout and topology. A Configuration instance contains Component objects, which fall under two categories: the Primitive class and the Interpolant class. Classes derived from Primitive represent the basic parts of the airframe—lifting surfaces, the fuselage, nacelles, pylons, struts, etc. Classes derived from Interpolant smoothly blend surfaces from Primitive components. A Junction instance forms the intersection between a Wing instance and another Primitive object, while Tip instances close wing tips and Cone instances are used for the nose cone and tail cone. Each Component instance computes the B-spline control points of its surfaces, which are then aggregated to define a global control point vector that represents a continuous description of the aircraft OML. The interpolation of Primitive components enables continuous deformation with differentiability always satisfied, and this is one of the main distinguishing features compared to other geometry engines in the literature [105, 107, 108, 106]. Figure 10.2 shows 6 aircraft configurations created in GeoMACH as a union of 4-sided B-spline surfaces.

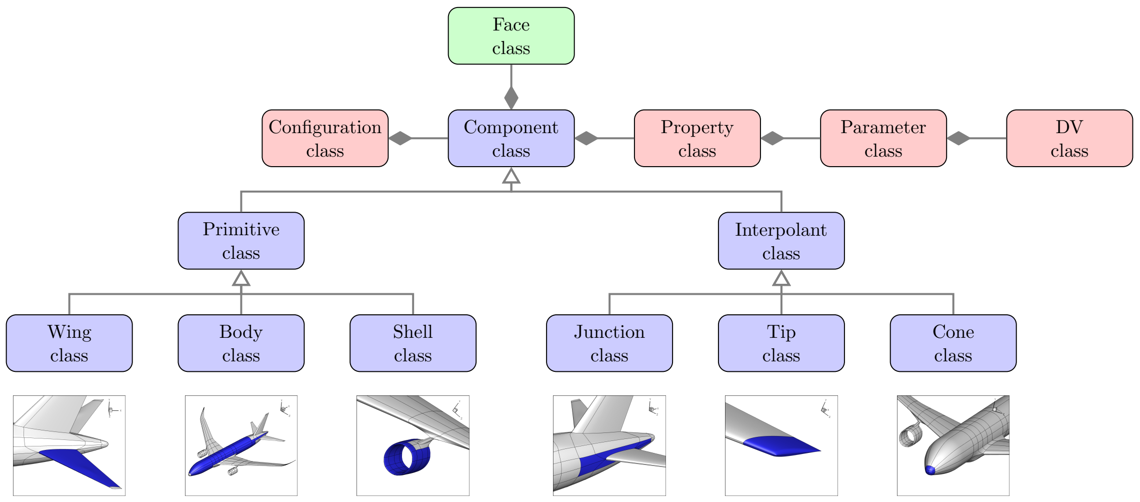


Figure 10.1: UML diagram of the parametric geometry modeler. Diamonds represent containment and open triangles represent inheritance.

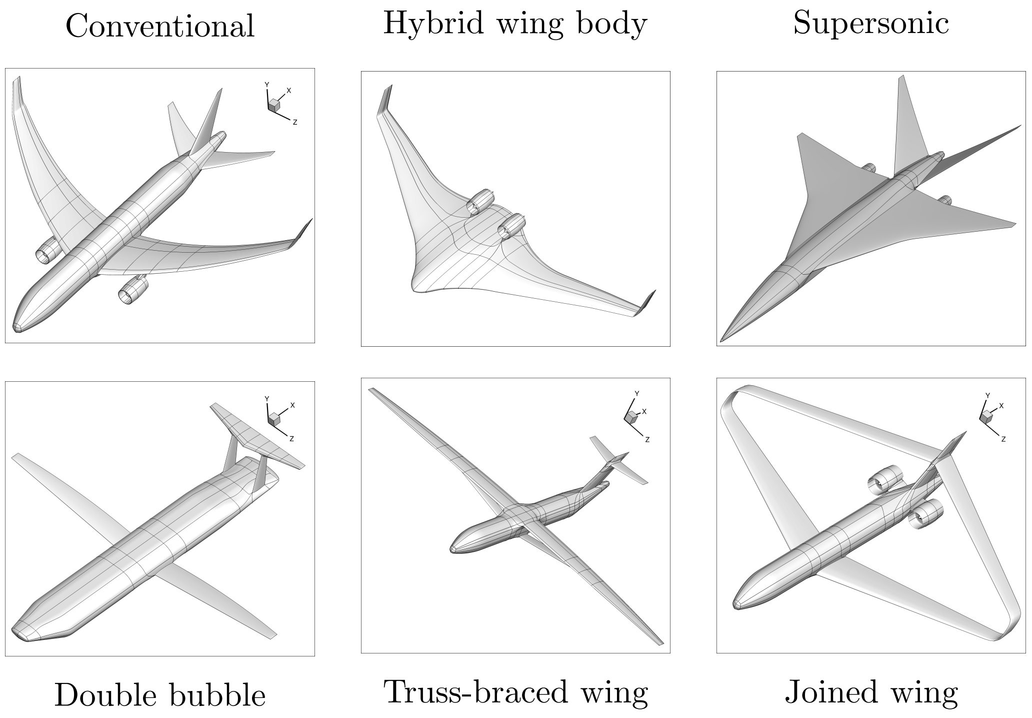


Figure 10.2: Six aircraft configurations created in GeoMACH.

Property	Shape	Description	Usage (e.g. Wing)
position	$(n, 3)$	Global origin coordinates	Sweep, dihedral, span
rotation	$(n, 3)$	Local to global frame	Twist, rotated sweep
scaling	$(n, 3)$	Stretching of local frame	Chord, thickness
origin	$(n, 3)$	Local origin coordinates	
orthogonality	$(n, 3)$	Bool; normal to anchor points	Winglets, vert. stab.
shape	(n_i, n_j)	Shape variables	

Figure 10.3: List of properties that parametrize Primitive components. In the shape column, n signifies the number of span-wise sections for a wing or the number of stream-wise sections for the fuselage.

10.2 Geometry parametrization

The Primitive components are parametrized in a manner similar to lofted sections. The B-spline control points are grouped by sections that represent airfoils for Wing components and cross-sections for Body and Shell components. The shape of the airfoil or cross-section is locally defined for a given section, and the bulk translation, rotation, and stretching of the section is controlled separately. The shape for a Primitive component is uniquely defined by six *properties* listed in Fig. 10.3.

The properties are, in turn, parametrized by *parameters* with another B-spline mapping. This allows for instance the coordinates of the origins of the n sections—i.e., the *position* property—to be defined by $m \times 3$ degrees of freedom instead of $n \times 3$, where $m < n$. Thus, when n is large, the number of degrees of freedom can be reduced to a much smaller number while smoothly parametrizing the n sections. This second layer of parametrization offers two advantages. First, it allows the resolution of the geometry representation and manipulation to be independent—the number of span-wise design variables is not fixed to the number of span-wise sections. Second, this parametrization spans the spectrum from high-level aircraft design parameters to local shape variables. If a constant chord is desired, only a single chord value needs to be specified implying a 1st order B-spline, or at the other extreme, n B-spline chord values can be specified. Alternatively, any number in between 1 and n is also possible.

After the B-spline control points belonging to Primitive components are computed, the same is done for the Interpolant components. A wireframe is first computed by interpolating the two components being attached using 3rd degree Bezier curves. The interior of the four-sided domains are then interpolated from the boundary curves as Coons patches. This process is illustrated in Fig. 10.4.

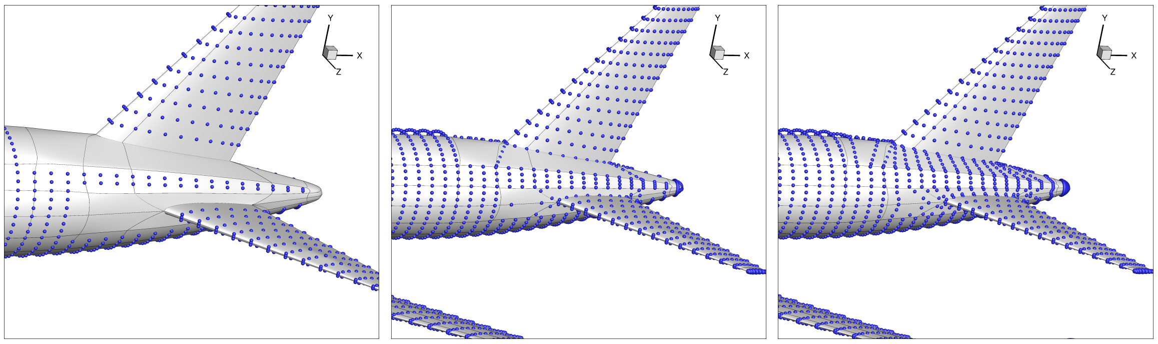


Figure 10.4: Illustration of the parametrization of a Junction component. Bezier curves are first generated from the intersecting components, and then the interior control points are populated using the formula for a Coons patch.

10.3 Computation of Derivatives

With a fine surface discretization and thousands of shape design variables, the Jacobian matrix can potentially be very large, necessitating its assembly as a sparse matrix. Computing derivatives as a sparse matrix is made difficult by the fact that there is a sequence of operations connecting the input to the output, and each intermediate quantity is distributed across aircraft components with coupling among them.

Two aspects of the adopted approach greatly facilitate the computation of derivatives. First, for each quantity, the parts corresponding to each Component are concatenated into a single vector. The parametrization is executed as an explicit sequence, as shown in Fig. 10.5, and the nonlinearity in the parametrization is contained in only two of the steps. The remaining steps are sparse matrices; therefore, no additional implementation of derivatives is necessary because the Jacobian matrix is the same matrix that defines the mapping itself. Having the sparse Jacobian available also makes it easy to provide the transpose of the matrix, which is needed for the adjoint method.

The second solution addresses storing and accessing the large, concatenated vectors. To simplify global indexing into the concatenated vectors, each quantity in the sequence in Fig. 10.5 allocates a pair of vectors, one containing the data and one containing the global indices. During initialization, NumPy views are created for subvectors of the concatenated vectors, and then reshaped before given to Component and Face instances. This allows Primitive components to work with the control-point

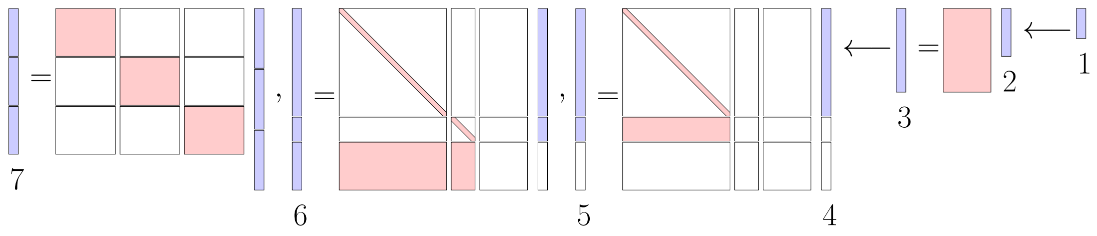


Figure 10.5: Sequence of operations in the parametrization: (1) Design variables (2) Parameters (3) Properties (4) Primitive control points (5) Primitive and wireframe control points (6) Full control points vector ordered by component and face (7) Unique control points ordered globally.

arrays in their true, 2-D shapes, instead of a flattened part of a global vector, as shown in Fig 10.6. Furthermore, if the sparse Jacobian is assembled as a coordinate list, the row and column indices are readily available as in Fig 10.6.

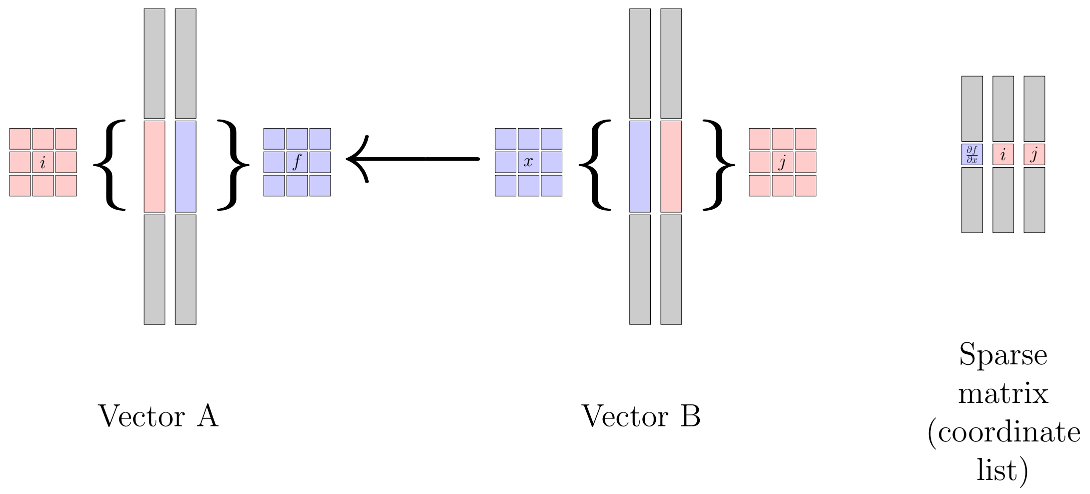


Figure 10.6: Illustration of data storage and accessing. Vectors A and B (e.g. global properties vector and the global parameters vector) are both accessed via reshaped NumPy views onto sub-vectors for each component. Each vector object contains 1-D data array (in blue) and a 1-D indices array (in red) of the same size—this facilitates assembly of the sparse Jacobian.

CHAPTER 11

An automated structural mesh generation algorithm

This chapter describes how GeoMACH generates and parametrizes structural meshes. Section 11.1 describes the overall approach, focusing on how all the structural nodes are parametrized by a linear mapping from the B-spline control points describing the geometry. Section 11.2 describes the two-step process by which the initial structural mesh is created based on the user's description of the desired structural members. Section 11.3 focuses on the second stage of this two-step process, in which a novel unstructured mesh generation algorithm is used. Finally, Section 11.4 presents the results of a mesh convergence study to verify that the overall mesh generation approach maintains the quality of the mesh as it is refined.

11.1 Linear mapping

The parametric structural modeler (PSM) has the ability to model detailed airframes including skins, spars, ribs, stringers, frames, longerons, and the cabin floor, among other structural members. The geometry of the internal structure is driven by the OML as the nodal positions are defined in terms of the aircraft OML points, allowing the entire structural mesh to be computed from the B-spline control points of the OML as a linear transformation. This is possible because the internal structure inside wing-type components is embedded in a parametric volume controlled by the upper and lower surfaces of the wing, and likewise, the internal structure inside a component such as a fuselage is projected in a cylindrical volume controlled by the fuselage skin. These internal structures warp, following changes to the fuselage and wing, and because the mapping is linear, updating the full structural mesh takes on the order of only tens of milli-seconds.

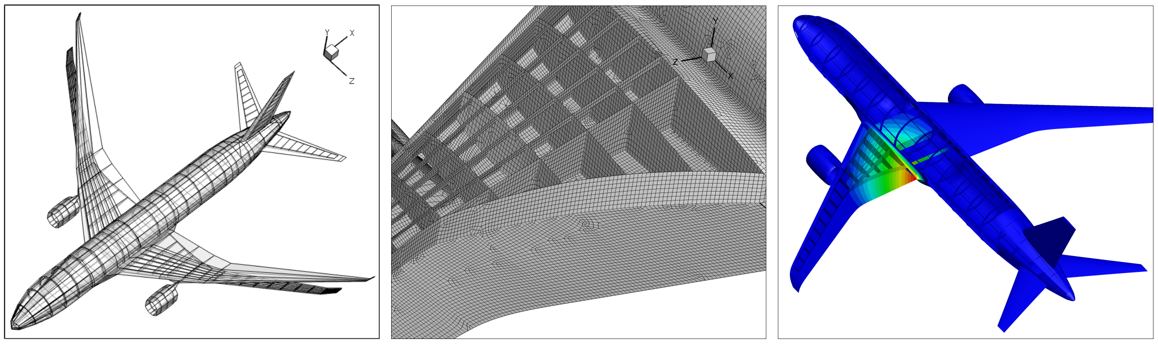


Figure 11.1: Preview mesh (left), detailed final FEA mesh (center), and derivative contours with respect to a root chord parameter (right).

Once the structural layout is computed during initialization, the structural mesh coordinates are defined by a linear mapping from the B-spline control points of the geometry description. Therefore, the derivatives of the structural mesh coordinates can be computed from the known derivatives of the control points with respect to high-level shape design variables in an efficient, accurate, and simple way since the Jacobian is a sparse matrix that does not change. Figure 11.1 plots derivative contours of the geometry of the OML and structure with respect to a notional design variable.

11.2 Two-step process

The parametric structural modeler uses a two-step process that first generates a coarsened preview mesh (shown in Fig 11.1) for two reasons. First, it provides a quick preview for the user to provide visual feedback on the airframe they have defined. This feature addresses the high-level objective of a fast turnaround time, as the preview mesh takes $\mathcal{O}(\sim \text{sec})$ to compute while the full mesh takes $\mathcal{O}(\sim \text{min})$ for a fine discretization. When the user is interactively designing the structure, the preview mesh is sufficient, so they can make changes and receive feedback in seconds. Furthermore, estimates for the dimensions of the structural members are computed from the preview mesh, and this information is later used to help ensure the quad elements have aspect ratios close to one and angles close to 90°.

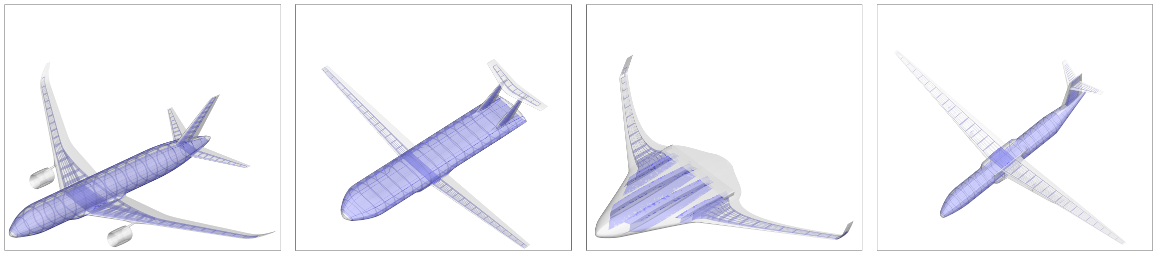


Figure 11.2: Four configurations with detailed structures.

11.3 Unstructured quadrilateral mesh generation

Automatic computation of a quadrilateral mesh for the entire airframe is challenging for several reasons. The user is allowed to specify any arrangement of structural members, and the intersection of spars, ribs, and stringers can create triangular or other non-four-sided patches on the skin that must be meshed with quad elements. Moreover, the meshes for all components must be connected together so an additional span-wise edge created on the wing by a spar must propagate through the wing-body junction and into the fuselage. Another challenge is that features such as taper make it more difficult to have high-quality, isotropic elements since a naive implementation would have the same number of chord-wise elements at the root and at the tip, which creates an imbalance with high resolution at the tip and low resolution at the root of the wing. The structural model for three configurations are shown in Fig. 11.2.

Two-dimensional quad meshing algorithms fall under three general categories: domain-decomposition [112, 113], advancing-front [114], and triangulation-based methods [115]. The first two—recursively splitting the domain through heuristic algorithms and marching out from boundaries— are unsuited to the current problem because of the line constraints imposed by the structural members intersecting the skin. Two additional ideas that have been successful are topology clean-up [116, 117, 118] and smoothing [119].

There has been work dealing with line constraints in structural mesh generation for marine engineering. Jang et al. use stiffener lines to decompose the domain into regions [120], while Lee et al. use an advancing-front approach on a background triangulation [121]. Park et al. also takes an advancing-front approach, but with topological intersection and clean-up operations [122].

The unique aspect of the current problem is that there are multiple non-planar domains that are connected to each other. The algorithm addresses this by first computing intersection points and discretizing the boundaries of each domain so that

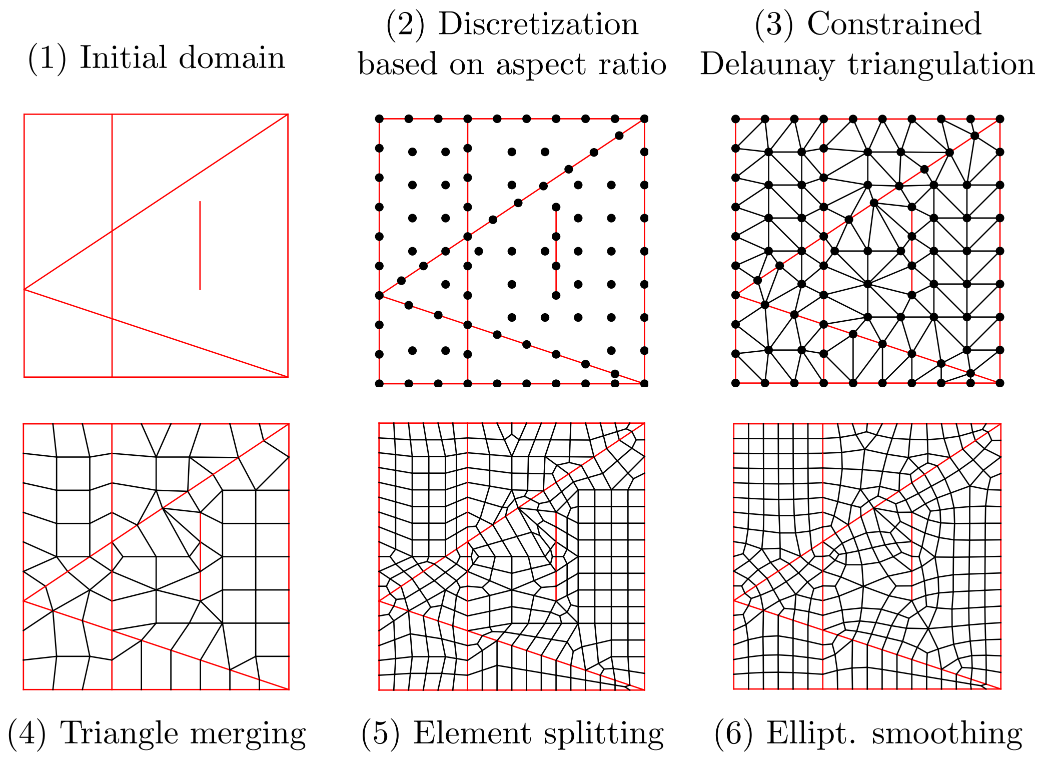


Figure 11.3: The six steps of the unstructured quad meshing algorithm.

each B-spline surface can be quad-patched separately, decoupled from the others.

Interior vertices are added, after which a constrained Delaunay triangulation is computed to ensure the intersection lines are respected. For the constrained Delaunay triangulation, the implementation in TRIPACK [123] is used. A quad-dominant mesh is then produced after ranking all potential merges of adjacent triangles, then a fully quad mesh is produced by splitting each quad and triangle. The final step is to improve mesh quality using elliptical smoothing. Finally, the mesh quality is improved using Laplacian smoothing with a second-order finite-element discretization. This procedure is illustrated in Fig. 11.3.

Elliptical smoothing There are several types of general mesh smoothing algorithms. One type solves optimization problems to maximize element quality [124] quantified using some combination of aspect ratios, angles, and areas, but nonlinear optimization is not always robust and can be inefficient. Another type formulates elliptical partial differential equations (PDEs) with derivatives taken with respect to the physical coordinates of the elements [125], but the resulting nonlinearity causes similar drawbacks to the optimization-based approach. Laplacian smoothing [126]

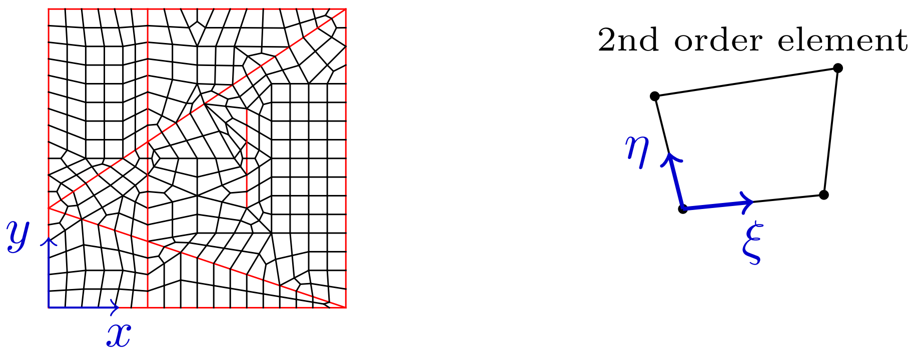


Figure 11.4: The global (left) and local (right) coordinate frames for the elliptical smoothing.

computes an explicit formula and avoids solving a system of equations, but it can at times make the mesh worse.

Here, Laplace's equation is formulated in a manner such that the resulting system of equations is linear, but the method is more robust than traditional Laplacian smoothing. The smoothing step computes the x and y coordinates of the nodes in the quad mesh by solving Laplace's equation, in the form,

$$\frac{\partial^2 \phi}{\partial \xi^2} + \frac{\partial^2 \phi}{\partial \eta^2} = 0, \tag{11.1}$$

where ϕ represents either x or y, and ξ and η are the local coordinates in the element frame, as shown in Fig. 11.4.

Laplace's equation is solved using a finite element discretization and the Galerkin method of weighted residuals. Assuming n is the order, the scalar field for x or y within an element is given by

$$\phi(\xi, \eta) = \sum_{k=1}^{n} \sum_{l=1}^{n} u_{kl} \cdot f_k(\xi) f_l(\eta), \tag{11.2}$$

where u_{kl} is a nodal value and f_k or f_l represents a basis function. Since the elements are bivariate, it is necessary to refine the mesh if elements of higher than 2nd order are desired.

The Galerkin method of weighted residuals is applied to Laplace's equation (11.1) and the definition of ϕ (11.2) is inserted, yielding

$$\int\int_A \left[\frac{\partial^2 \phi}{\partial \xi^2} + \frac{\partial^2 \phi}{\partial \eta^2} \right] f_i(\xi) f_j(\eta) dA = 0 \tag{11.3}$$

Prior to inserting the approximate form for ϕ, it is beneficial to apply a step similar to integration by parts to reduce the order of the integrand. Using the product rule, the two terms in the integrand can be written as

$$f_i(\xi)f_j(\eta)\frac{\partial^2\phi}{\partial\xi^2} = \frac{\partial}{\partial\xi}\left(f_i(\xi)f_j(\eta)\frac{\partial\phi}{\partial\xi}\right) - \frac{\partial}{\partial\xi}\left(f_i(\xi)f_j(\eta)\right)\frac{\partial\phi}{\partial\xi} \tag{11.4}$$

$$f_i(\xi)f_j(\eta)\frac{\partial^2\phi}{\partial\eta^2} = \frac{\partial}{\partial\eta}\left(f_i(\xi)f_j(\eta)\frac{\partial\phi}{\partial\eta}\right) - \frac{\partial}{\partial\eta}\left(f_i(\xi)f_j(\eta)\right)\frac{\partial\phi}{\partial\eta}, \tag{11.5}$$

and inserting the expressions in Eqs. (11.4) and (11.5) into Eq. (11.3) yields

$$\begin{aligned}&\iint\limits_A \left[\frac{\partial}{\partial\xi}\left(f_i(\xi)f_j(\eta)\frac{\partial\phi}{\partial\xi}\right) + \frac{\partial}{\partial\eta}\left(f_i(\xi)f_j(\eta)\frac{\partial\phi}{\partial\eta}\right)\right] dA - \\ &\iint\limits_A \frac{\partial}{\partial\xi}\left(f_i(\xi)f_j(\eta)\right)\frac{\partial\phi}{\partial\xi} dA - \iint\limits_A \frac{\partial}{\partial\eta}\left(f_i(\xi)f_j(\eta)\right)\frac{\partial\phi}{\partial\eta} dA = 0.\end{aligned} \tag{11.6}$$

In the first term of Eq. (11.6), the integrand is the divergence of a vector field and the integration occurs over a compact subset, so Gauss's theorem can be applied to establish the equality,

$$\iint\limits_A \nabla\cdot\begin{bmatrix} f_i(\xi)f_j(\eta)\frac{\partial\phi}{\partial\xi} \\ f_i(\xi)f_j(\eta)\frac{\partial\phi}{\partial\eta}\end{bmatrix} dA = \oint\limits_{\partial A}\begin{bmatrix} f_i(\xi)f_j(\eta)\frac{\partial\phi}{\partial\xi} \\ f_i(\xi)f_j(\eta)\frac{\partial\phi}{\partial\eta}\end{bmatrix}\cdot\hat{n}\,ds, \tag{11.7}$$

but $f_i(\xi)$ and $f_j(\eta)$ are zero on ∂A for appropriately chosen shape functions, so the entire line integral on the right-hand side is zero.

With the divergence term equal to zero, inserting the expression for ϕ into Eq. (11.6) yields

$$\iint \frac{\partial}{\partial\xi}\left(f_i(\xi)f_j(\eta)\right)\frac{\partial}{\partial\xi}\left(\sum_{k=1}^{n}\sum_{l=1}^{n} u_{kl}f_k(\xi)f_l(\eta)\right)d\xi d\eta + \tag{11.8}$$

$$\iint \frac{\partial}{\partial\eta}\left(f_i(\xi)f_j(\eta)\right)\frac{\partial}{\partial\eta}\left(\sum_{k=1}^{n}\sum_{l=1}^{n} u_{kl}f_k(\xi)f_l(\eta)\right)d\xi d\eta = 0 \tag{11.9}$$

Rearranging, the final form of the discretized equations is

$$\sum_{k=1}^{n}\sum_{l=1}^{n}\left[\int f_i'(\xi)f_k'(\xi)d\xi\int f_j(\eta)f_l(\eta)d\eta + \int f_i(\xi)f_k(\xi)d\xi\int f_j'(\eta)f_l'(\eta)d\eta\right]u_{kl} = 0. \tag{11.10}$$

Introducing the matrices F and F' to simplify, the previous equation becomes

$$\sum_{k=1}^{n}\sum_{l=1}^{n}\left[F'_{ik}F_{jl}+F_{ik}F'_{jl}\right]u_{kl}=0 \tag{11.11}$$

or

$$\sum_{k=1}^{n}\sum_{l=1}^{n}K_{ij,kl}\,u_{kl}=0, \tag{11.12}$$

where $K_{ij,kl}$ represents the entry in the global finite element matrix located at the row corresponding to node (i,j) and the column corresponding to node (k,l) in the current element. Similarly, u_{kl} is the value of x or y for the node in the global ordering indexed as (i,j) in the current element.

Though the true basis functions vanish at their boundaries, it is valid to derive element matrices by considering only the parts of basis functions within a given element. A linear equation corresponding to an interior node includes contributions from all the adjacent elements, which together partition the true basis function associated with this node. A linear equation corresponding to a boundary or constrained interior node does not influence the finite element solution.

The basis functions and the matrices for second-order elements are

$$\begin{array}{l} f_1(t)=1-t \\ f_2(t)=t \end{array} \quad : \quad F=\frac{1}{6}\begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix} \quad \text{and} \quad F'=\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix} \tag{11.13}$$

and those for third order-elements are

$$\begin{array}{l} f_1(t)=2t^2-3t+1 \\ f_2(t)=4t-t4^2 \\ f_3(t)=2t^2-t \end{array} \quad : \quad F=\frac{1}{30}\begin{bmatrix}4 & 2 & -1\\ 2 & 16 & 2\\ -1 & 2 & 4\end{bmatrix} \quad \text{and} \quad F'=\frac{1}{3}\begin{bmatrix}7 & -8 & 1\\ -8 & 16 & -8\\ 1 & -8 & 7\end{bmatrix}. \tag{11.14}$$

With the local element matrices derived, the global finite element equations can be assembled and solved in the form presented in Fig. 11.5. All unique nodes are concatenated into a single vector, and the local element matrices contribute to a global finite element matrix, K. The linear system is solved twice, once for all the x values and once for all the y values. The rectangular matrix P in Fig. 11.5 is a permutation matrix that has one entry per row with a value of 1 in a column corresponding to a constrained node. Therefore, P is a linear transformation that selects from the global vector of nodes only those that are on the boundary or are in the interior, but are fixed. In Fig. 11.5, the $*$ components of the solution vectors

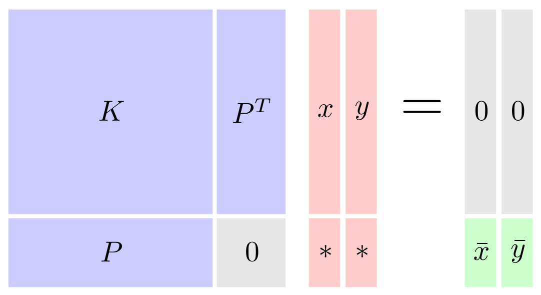


Figure 11.5: Global finite element equations for the elliptical smoothing. The K matrix represents the global finite element matrix, and P is a permutation matrix with a single entry of value 1 in each row, in the columns of nodes that are constrained. The vectors x, y represent the solution and $\bar{x}, \bar{y}$ contain the coordinates to which a node is constrained.

are ignored, and the $\bar{x}$ and $\bar{y}$ sub-vectors represent the coordinates of the constrained nodes.

11.4 Mesh convergence study

This section presents a mesh convergence study to verify that the unstructured quad meshing algorithm maintains element quality as the mesh is refined. The test case is the wing from the common research model (CRM) [94], which is a reference geometry developed for benchmarking. The CRM is based on the Boeing 777 wing, featuring a straight leading edge with the Yehudi break located at 37 % of the span.

The structure includes two main spars, stringers, ribs, and a secondary spar. The aft spar, the secondary spar, and one of the ribs form a triangle, testing the algorithm's ability to handle such cases. Even without the triangle, this case is designed to produce irregular nodes because of the wing taper—there will be far more elements in the chordwise direction at the root than at the tip of the wing.

A 1 kN point load is applied at the upper, outboard-most tip of the front spar, and the vertical deflection is measured at the same location. An aluminum structure is used with a constant thickness of 5 mm, and linear kinematics are assumed. Figure 11.6 shows six meshes of varying resolutions and the contours of vertical deflection, and Fig. 11.7 plots the results of the mesh convergence study. In Fig. 11.7, the displacement appears to converge monotonically to about 2.1 mm. This result verifies that the unstructured mesh generation algorithm at least maintains element quality as the mesh is refined.

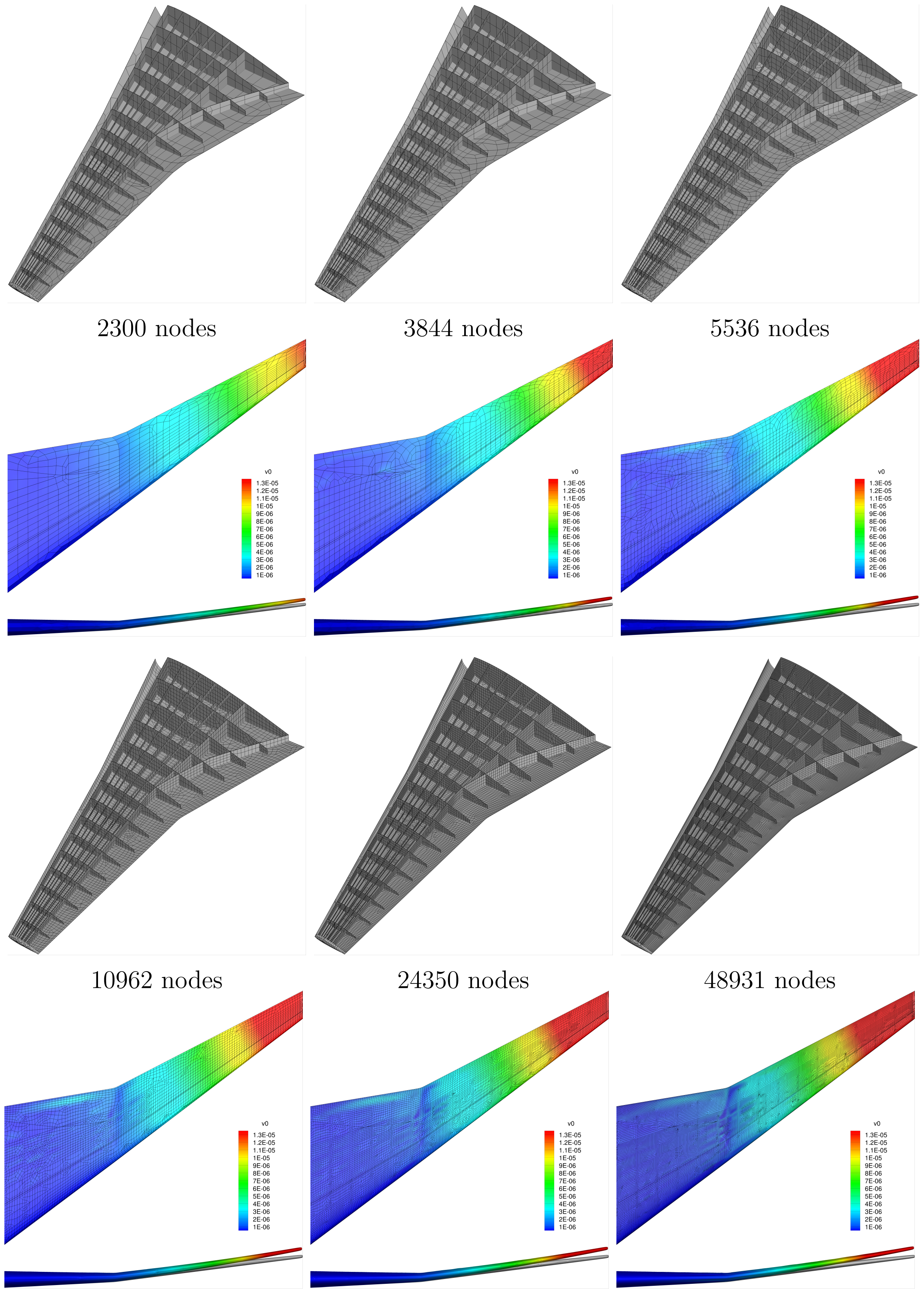


Figure 11.6: The meshes produced by the unstructured quad meshing algorithm and contours of vertical displacement.

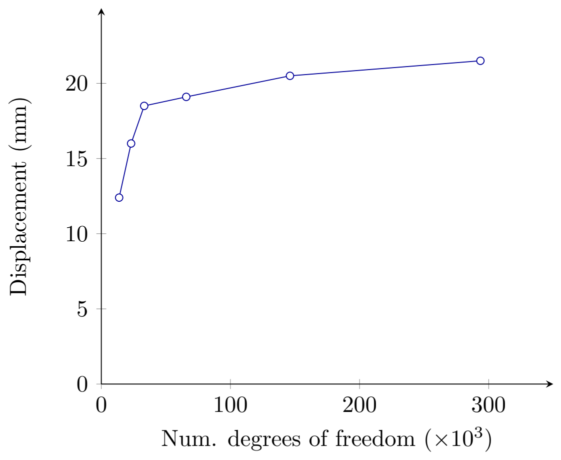


Figure 11.7: Results of the mesh convergence study. The deflection is measured at the upper, outboard-most tip of the front spar, and the loading is a 1 kN point load at the same node.

CHAPTER 12

Aerostructural analysis and aerodynamic shape optimization results

As a demonstration, GeoMACH is used to create geometries and structural meshes for the truss-braced wing and double bubble configurations for analysis and optimization. This chapter describes the software for aerostructural analysis and optimization in Sec. 12.1, presents aerostructural analysis results in Sec. 12.2, and presents aerodynamic shape optimization results in Sec. 12.3.

12.1 The MACH tool suite

The aerostructural analysis and aerodynamic optimization results shown in this chapter use the MACH tool suite, which stands for MDO of aircraft configurations with high fidelity. MACH consists of a CFD mesh warping algorithm, a flow solver, a structural solver, and a coupled aerostructural solver.

12.1.1 Mesh warping: pyWarp

The mesh warping tool, pyWarp, warps a 3-D structured multi-block CFD mesh based on geometry changes [111]. To do this, it solves the equations of elasticity with displacement constraints on a coarsened version of the multi-block mesh, and regenerates the new mesh with the original resolution using transfinite interpolation. It benefits from the robustness of the elasticity-based mesh warping approach as well as the efficiency provided by coarsening using algebraic interpolation. The elasticity equations are solved using the PETSc [46] implementation of SuperLU_DIST [127], a parallel direct linear solver.

12.1.2 Flow solver: SUmb

The Stanford University multi-block (SUmb) flow solver is used for aerodynamics. SUmb solves the Euler or Reynolds-averaged Navier–Stokes (RANS) equations using a finite volume discretization. SUmb uses multigrid during startup and during the 4th order Runge-Kutta time integration stage, after which it switches to a Newton-Krylov solver for faster convergence. The Krylov solver used is the flexible generalized minimal residual (fGMRES) method, with nested preconditioning involving the Richardson iteration, the additive Schwarz method, incomplete LU factorization, and reverse Cuthill–Mckee (RCM) reordering. The Jacobian is assembled using algorithmic differentiation. More details on the numerical methods can be found in Kenway et al. [19].

12.1.3 Structural solver: TACS

The toolkit for the analysis of composite structures (TACS) [128] is used for structural analysis. The quadrilateral shell elements in TACS use a mixed-interpolation of tensorial components (MITC) formulation and first-order shear deformation theory (FSDT). TACS computes a parallel direct factorization to solve the linear systems that arise using the Schur complement method for domain decomposition and LU decomposition locally.

12.1.4 Aerostructural solver: pyAeroStruct

Coupling the aerodynamic and structural solvers together is handled by pyAerostruct. It solves the coupled equations simultaneously using a Newton–Krylov solver with linear block Jacobi preconditioning and a nonlinear block Gauss–Seidel startup phase. Each linear block in the preconditioner reuses that from the respective solver. Rigid links are used to transfer loads from the CFD mesh to the FEA mesh and displacements in the reverse direction [129].

12.2 Aerostructural analysis

The geometries and structural meshes generated for the truss-braced wing (TBW) and double bubble (D8) configurations are used to perform aerostructural analysis. SUmb is used to solve the Euler equations at a Mach number of 0.74 in both cases. On the structures side, TACS is used with linear kinematics and a linear-elastic, aluminum structure.

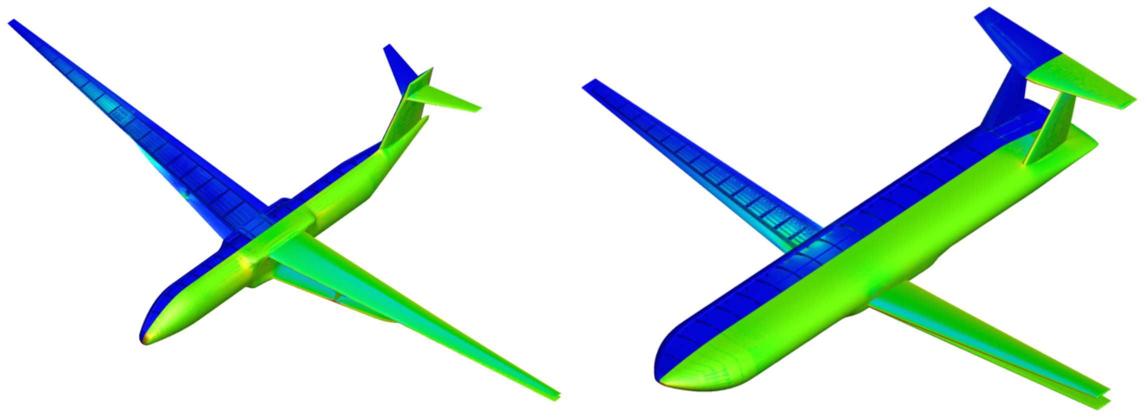


Figure 12.1: Aerostructural analysis results for the TBW (left) and D8 (right) configurations.

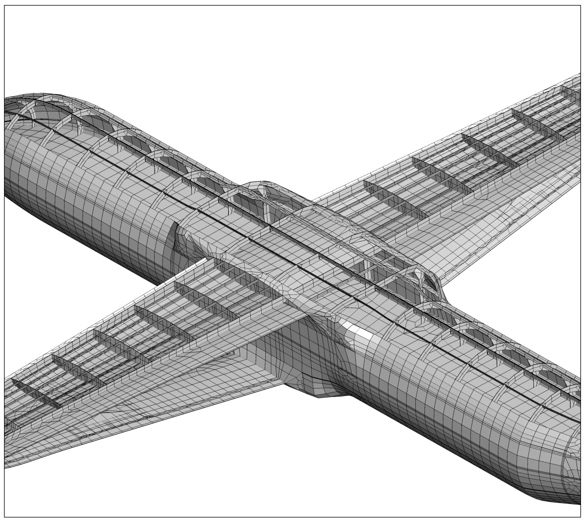


Figure 12.2: The TBW structure shown in detail.

The aerostructural analysis results are shown in Fig. 12.1. For the TBW, the CFD mesh has 1.42 million cells while the FEA mesh has 44,000 nodes. For the D8, the CFD mesh 3 million cells and the FEA mesh has 50,000 nodes. The structures for both configurations include two main spars, ribs, and stringers for the main wing, horizontal, and vertical stabilizers. Other structural features include center-body wingboxes where appropriate, and longerons and frames for the fuselages. A detailed viewed of the TBW structure is shown in Fig. 12.2.

	Variable/function	Description	Quantity
minimize	c_D	Drag coefficient	
with respect to	α	Angle of attack	1
	$0 \leq x_f \leq 1$	Fuselage shape variables	5×5
	$0 \leq x_{w,U} \leq 1$	Wing upper surf. shape variables	10×10
	$0 \leq x_{w,L} \leq 1$	Wing lower surf. shape variables	10×10
	$0 \leq x_{s,U} \leq 1$	Strut upper surf. shape variables	8×8
	$0 \leq x_{s,L} \leq 1$	Strut lower surf. shape variables	8×8
	$0 \leq x_{v,U} \leq 1$	Vert. strut upper surf. shape vars.	5×5
	$0 \leq x_{v,L} \leq 1$	Vert. strut lower surf. shape vars.	5×5
	$0 \leq x_{t,U} \leq 1$	Tail upper surf. shape variables	8×8
	$0 \leq x_{t,L} \leq 1$	Tail lower surf. shape variables	8×8
		Total	532
subject to	$c_L - 0.5 = 0$	Lift coefficient constraint	1
	$0.25t_w^0 - t_w \leq 0$	Wing thickness constraints	20×20
	$0.25t_s^0 - ts \leq 0$	Strut thickness constraints	15×15
	$0.25t_v^0 - t_v \leq 0$	Vert. strut thickness constraints	10×10
	$0.25t_t^0 - t_t \leq 0$	Tail thickness constraints	15×15
		Total	951

Table 12.1: The optimization problem.

12.3 Aerodynamic shape optimization

As a demonstration of high-fidelity optimization, aerodynamic shape optimization is applied to the TBW using GeoMACH and the MACH tool suite. Lift-constrained drag minimization is performed at a Mach number of 0.74 and c_L of 0.5. The optimization problem is given in Tab. 12.1. Angle of attack is a design variable used to satisfy the lift coefficient constraint, and 531 B-spline shape variables are used to parametrize the upper and lower surfaces of the wing, main strut, vertical strut, and horizontal stabilizer, as well as a portion of the side of the fuselage where the wing and strut attach. Nonlinear thickness constraints are included, although they are not necessary in the design problem because convergence issues forced the addition of variable bounds that only permit the control points to thicken the airfoil.

The optimization results are shown in Fig. 12.3. The initial design has a shock covering almost the entirety of the area bounded by the wing, strut, and fuselage, yielding an initial drag of 810 counts. Through shape optimization, the shock is nearly eliminated and this figure is reduced to 319 counts. The initial design also has a higher lift coefficient than required; however, even at the prescribed lift coefficient,

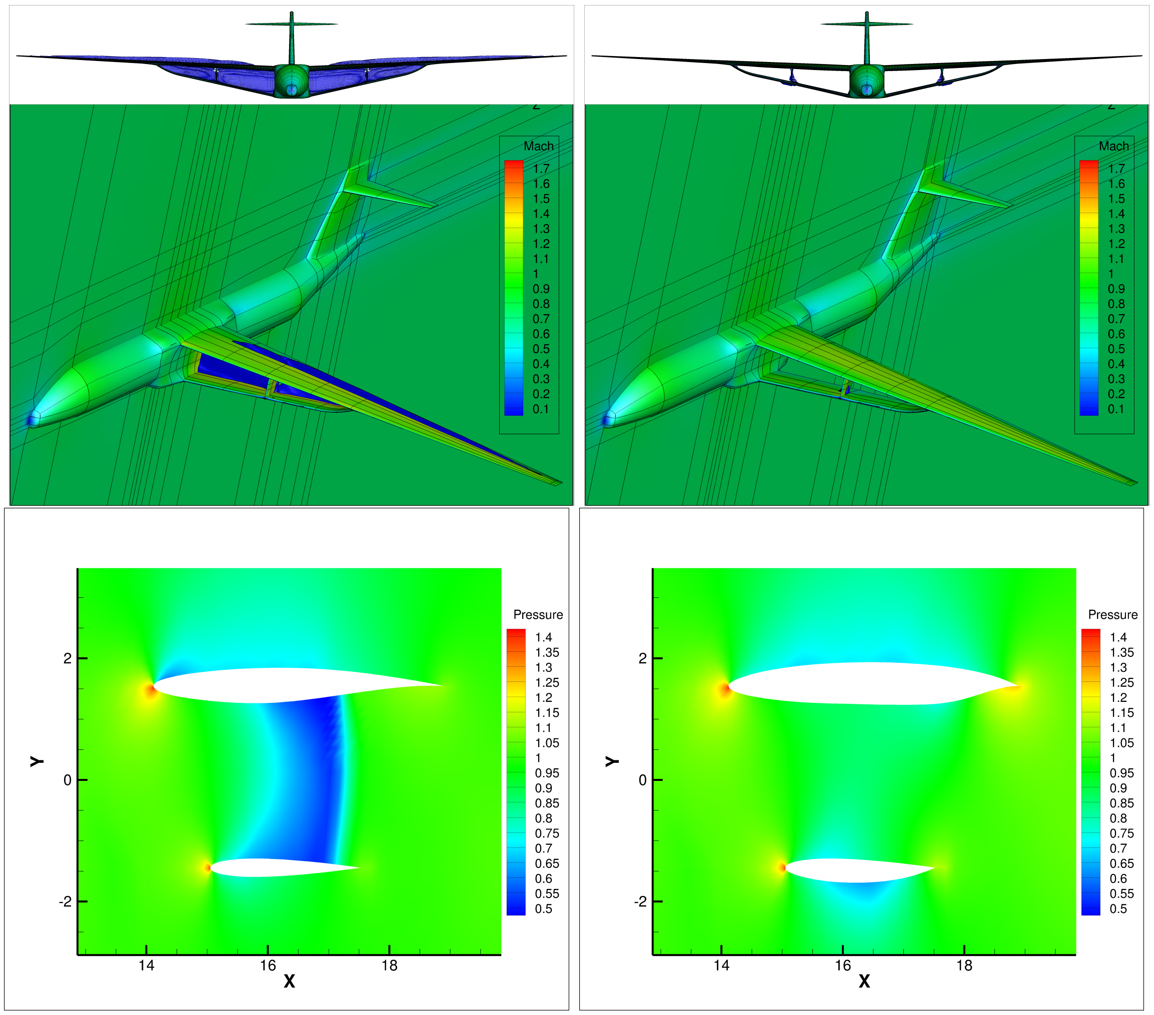


Figure 12.3: Initial (left) and optimized (right) shape and CFD solution of the truss-braced wing configuration. The pressure contours represent a slice near the root of the wing.

the initial design has a drag of roughly 750 counts. It can be seen from Fig. 12.3 that one of the biggest changes is the flattening of the lower surface of the main wing, most likely to avoid the effects of a diverging nozzle. This results in a relatively thick airfoil, but this result is interpreted as a product of the lack of freedom to thin the wing near the quarter-chord mark.

There are two aspects to the significance of these results. The first is the value of the parametric geometry modeler for enabling high-fidelity shape optimization of an unconventional configuration. To eliminate the shock, the optimizer is given fine control of the shape of the fuselage, wing, and struts, which are components that intersect with each other and must be smoothly blended with each other. Moreover, an optimization problem was solved with the intersection point of the strut and wing allowed to move span-wise, but this optimization problem proved to be insensitive to this design variable. This shows that it is possible to perform high-fidelity optimization with high-level design variables that make large geometry changes. The second aspect is the fast turnaround time for computational design demonstrated in this problem. After the initial time investment to create the grid, setting up new design variables and variants of the optimization problem takes only on the order of minutes. Furthermore, on 128 processors, an optimization problem of this size requires only a few hours to achieve the majority of convergence.

Aerodynamic shape optimization has also been run for the TBW using the Reynolds-averaged Navier–Stokes (RANS) equations. This problem is also a lift-constrained drag minimization, but at a Mach number of 0.73 and lift coefficient of 0.775. The fuselage is removed in this geometry to focus on the junction; however, as before, there are shape variables on the wing, main strut, and vertical strut.

The results are shown in Fig. 12.4. The total drag is reduced from 458 to 363 counts. As with the Euler case, the shape variables are only permitted to thicken the airfoils, and Fig. 12.4 suggests that they try to eliminate the diverging shape of the flow region between the wing and the main strut, as before.

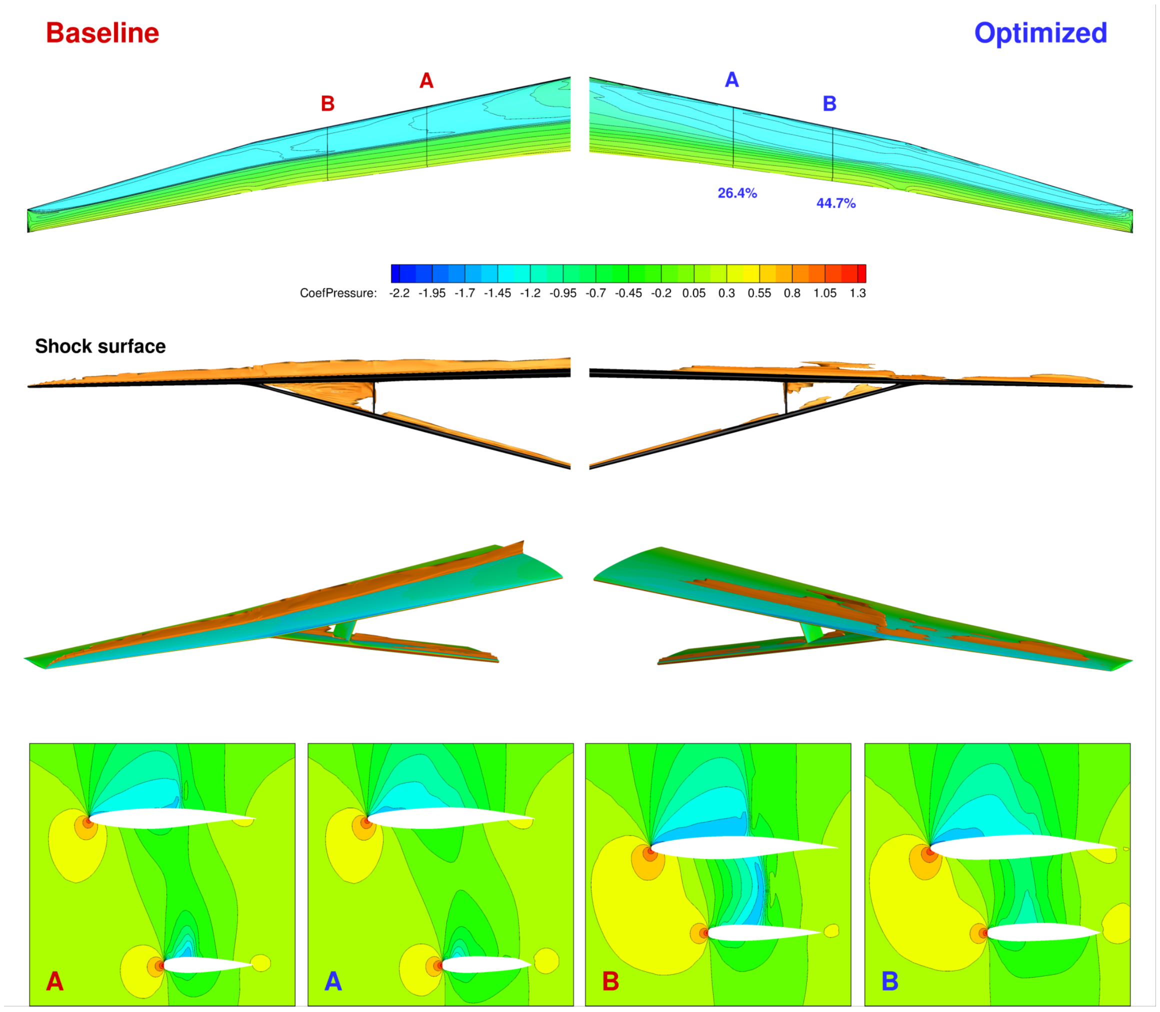

Baseline
Optimized
B
A
A
B
26.4%
44.7%
CoefPressure: -2.2 -1.95 -1.7 -1.45 -1.2 -0.95 -0.7 -0.45 -0.2 0.05 0.3 0.55 0.8 1.05 1.3
Shock surface
A
A
B
B

Figure 12.4: RANS-based aerodynamic shape optimization of the TBW at M=0.73.

CHAPTER 13

Conclusions

13.1 Summary and contributions

In Part II of this thesis, I presented a method for parametrizing the aircraft OML and structure in a differentiable way. This method takes a geometry-centric approach to high-fidelity aircraft design optimization where a central B-spline-based geometry model parametrically drives the CFD surface mesh and the FEA mesh of the airframe structure. The changes to the CFD surface mesh are subsequently propagated to the full volume mesh with an external mesh warping tool. After describing this approach, I presented a differentiable geometry parametrization that efficiently computes the Jacobian of derivatives in a sparse format. Next, I presented an unstructured quadrilateral mesh generation algorithm for automatically creating meshes of the airframe structure. These components constitute GeoMACH, an open-source aircraft parametrization tool suite funded by NASA. I presented aerostructural analysis and truss-braced wing aerodynamic shape optimization results as a demonstration of the developed tools.

My contributions are as follows:

- *I developed GeoMACH, an open-source aircraft parametrization tool suite that parametrizes the CFD surface mesh and the FEA mesh in terms of user-defined geometric parameters.*

  GeoMACH represents the geometry using B-splines, and it linearly maps the B-spline control points defining the geometry to the CFD surface mesh and the structural FEA mesh. The B-spline control points are in turn computed from user-selected geometric parameters, which can be high-level (e.g., wing span or fuselage length) or low-level (e.g., the control point positions defining the airfoil shapes). GeoMACH automatically computes the sparse Jacobians of derivatives of the CFD surface mesh and the FEA mesh with respect to the user-selected geometric parameters, which is precisely what is needed for aerodynamic shape optimization.

- *I developed a differentiable parametrization of aircraft geometry, which is enabled by using Bezier and bilinear interpolation to define the junctions between aircraft components.*

  This interpolation method is what provides differentiability while existing geometry modeling tools are discontinuous or non-smooth because they apply intersection, and in some cases, triangulation algorithms.

- *I developed an unstructured mesh generation algorithm that can automatically generate meshes for the aircraft structure.*

  This algorithm is applied on each structural member and B-spline surface on the OML to ensure the skins of different aircraft components are connected to each other as well as to internal structural members. It applies constrained Delaunay triangulation on a background Cartesian grid of points, merges triangles using heuristics, and then applies elliptical smoothing to generate high-quality elements. I demonstrated that this algorithm can be used to quickly and easily generate detailed structural meshes for unconventional configurations, including ribs, spars, stringers, fuselage frames, etc. I also performed a mesh convergence study to verify that the overall mesh generation approach maintains the quality of the mesh as it is refined.

- *I demonstrated using GeoMACH that, for the truss-braced wing, it is possible to use aerodynamic shape optimization to eliminate the shock caused by the presence of the strut to reduce drag from 750 to 319 counts using Euler-based lift-constrained drag minimization.*

  The optimization includes shape design variables placed on the wing, struts, and fuselage. The inclusion of shape variables on intersecting components is possible thanks to GeoMACH's smooth interpolation of junctions between components such as the wing-strut junction.

13.2 Significance

GeoMACH enables high-fidelity aerostructural optimization by providing a differentiable parametrization of aircraft geometries and structures. Moreover, it is an open-source tool that is modular, so it lowers the entry barrier for large-scale aircraft design optimization involving CFD, FEA, or both.

The significance of the truss-braced wing (TBW) optimization results is two-fold. First, it demonstrates that GeoMACH enables aerodynamic shape optimization to take a poor initial design for an unconventional configuration and produce a reasonable design in a short amount of time: $\mathcal{O}(1 \text{ hr})$ with Euler and $\mathcal{O}(10 \text{ hr})$ with RANS. Second, the TBW result demonstrates that it is possible to perform shape optimization of an entire configuration simultaneously because GeoMACH's junction interpolations successively handled the inclusion of shape variables from intersecting components. Moreover, since GeoMACH parametrizes the B-spline control points for the entire geometry, it is possible to perform shape optimization of a junction to minimize interference and parasitic drag by fine-tuning the shape. This is especially relevant for a configuration like the truss-braced wing because of the wing-strut junction.

The significance of GeoMACH's structural mesh generation algorithm consists of three points. First, it is an algorithm that can be easily reproduced in other applications because it is much simpler than advancing-front methods and domain-decomposition methods. Second, it enables the rapid and automated creation of detailed structural meshes, and it does not require post-processing to improve mesh topology and mesh quality. Thanks to the automation, the structural mesh generation algorithm enables parametric studies that vary, for instance, the layout or the number of ribs and spars of a wing. Finally, the structural mesh is generated as a sparse Jacobian mapping the B-spline control points describing the geometry to the structural nodes, so it is parametrically driven by the aircraft shape. Thus, GeoMACH enables structural and aerostructural optimization. In contrast, many existing methods for generating structural meshes require regeneration whenever the geometry changes, which is inefficient and produces a non-differentiable map.

With respect to unconventional configurations, the significance of GeoMACH as a whole is that it can be used to evaluate and compare configurations. For instance, it can be used to rapidly create structural meshes for multiple unconventional configurations and perform structural sizing optimizations to quantify the structural benefits of configurations like the joined wing and the truss-braced wing that rely on improvements in structural efficiency.

13.3 Recommendations for future work

Much of the significance of GeoMACH is in the types of studies it enables, so many recommendations for future work are stated in the previous section. This section discusses four more avenues for future work.

The first is the addition of micromechanical models for composites, enabling the addition of material design variables in the optimization problem. Since GeoMACH creates the structural mesh, it is aware of the indices of the quad elements belong to each structural member—therefore, it would be possible to define a set of composite design variables for each member. This would involve the implementation of micromechanical models to first map the composite design variables to the constitutive matrices and then map the constitutive matrices to the shell stiffness matrices as required by first-order shear deformation theory (FSDT).

The second recommendation for future work is the automatic generation of grids for overset CFD. Since GeoMACH divides the geometry into components, each type of component could have an associated grid that algebraically warps based on the geometry parameters for the component. Assuming the desired flow solver is an overset CFD solver, this would eliminate the need to create the CFD grid and warp it externally.

The third recommendation is to use the B-spline representation of the geometry as a tool for performing load and displacement transfer across the CFD and FEA meshes. Since GeoMACH computes Jacobians mapping the B-spline control points to the CFD and FEA surface meshes, GeoMACH could be used to interpolate the loads from the CFD surface mesh to the FEA surface mesh and the displacements from the FEA surface mesh to the CFD surface mesh. There would be some error since shocks would smoothed and the transfer scheme may not conserve virtual work, but this approach is worth investigating because of the modularity and simplicity.

The final recommendation is adding the overset grid representation or a general mesh warping algorithm to GeoMACH, and implementing all the components of GeoMACH in the computational modeling framework described in Part I of this thesis. The result would be a fully modular aircraft design tool suite in which arbitrary CFD or FEA solvers can be attached.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Lambe, A. B. and Martins, J. R. R. A., "Extensions to the Design Structure Matrix for the Description of Multidisciplinary Design, Analysis, and Optimization Processes," *Structural and Multidisciplinary Optimization*, Vol. 46, August 2012, pp. 273–284. doi:10.1007/s00158-012-0763-y.

[2] Rutherford, D. and Zeinali, M., "Efficiency Trends for New Commercial Jet Aircraft 1960 to 2008," The International Council on Clean Transportation, 2009.

[3] Grihon, S., Krog, L., Tucker, A., and Hertel, K., "A380 weight savings using numerical structural optimization," *20th AAAF colloquium on material for aerospace applications, Paris, France*, 2004, pp. 763–66.

[4] Krog, L., Tucker, A., Kemp, M., and Boyd, R., "Topology optimization of aircraft wing box ribs," *10th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2004, pp. 1–11.

[5] Hicken, J. E. and Zingg, D. W., "Aerodynamic Optimization Algorithm with Integrated Geometry Parameterization and Mesh Movement," *AIAA Journal*, Vol. 48, No. 2, Feb. 2009, pp. 400–413.

[6] Jansen, P., Perez, R. E., and Martins, J. R. R. A., "Aerostructural Optimization of Nonplanar Lifting Surfaces," *Journal of Aircraft*, Vol. 47, No. 5, 2010, pp. 1491–1503. doi:10.2514/1.44727.

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

[8] Bertsimas, D. and Tsitsiklis, J. N., "Introduction to linear optimization," 1997.

[9] Perez, R. E., Jansen, P. W., and Martins, J. R. R. A., "pyOpt: a Python-Based Object-Oriented Framework for Nonlinear Constrained Optimization," *Structural and Multidisciplinary Optimization*, Vol. 45, No. 1, January 2012, pp. 101–118. doi:10.1007/s00158-011-0666-3.

[10] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II," Tech. Rep. 200001, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology Kanpur, 2000.

[11] Gill, P., Murray, W., and Saunders, M., "SNOPT: An SQP algorithm for large–scale constraint optimization," *SIAM Journal of Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006.

[12] Rosenbrock, H. H., "An automatic method for finding the greatest or least value of a function," *The Computer Journal*, Vol. 3, No. 3, 1960, pp. 175–184.

[13] Martins, J. R. R. A. and Hwang, J. T., "Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models," *AIAA Journal*, Vol. 51, No. 11, November 2013, pp. 2582–2599. doi:10.2514/1.J052184.

[14] Jameson, A., "Aerodynamic Design via Control Theory," *Journal of Scientific Computing*, Vol. 3, No. 3, Sept. 1988, pp. 233–260.

[15] Reuther, J. J., Jameson, A., Alonso, J. J., Rimlinger, M. J., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, Part 1," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 51–60.

[16] Reuther, J. J., Jameson, A., Alonso, J. J., Rimlinger, M. J., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, Part 2," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 61–74.

[17] Lyu, Z. and Martins, J. R. R. A., "Aerodynamic Shape Optimization of a Blended-Wing-Body Aircraft," *Proceedings of the 51st AIAA Aerospace Sciences Meeting*, Grapevine, TX, Jan. 2013. doi:10.2514/6.2013-283, AIAA 2013-0283.

[18] Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., "A Coupled-Adjoint Sensitivity Analysis Method for High-Fidelity Aero-Structural Design," *Optimization and Engineering*, Vol. 6, No. 1, March 2005, pp. 33–62. doi:10.1023/B:OPTE.0000048536.47956.62.

[19] Kenway, G. K. W., Kennedy, G. J., and Martins, J. R. R. A., "Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Derivative Computations," *AIAA Journal*, Vol. 52, No. 5, May 2014, pp. 935–951. doi:10.2514/1.J052255.

[20] Lyu, Z., Kenway, G. K., and Martins, J. R. R. A., "Aerodynamic Shape Optimization Studies on the Common Research Model Wing Benchmark," *AIAA Journal*, 2014. doi:10.2514/1.J053318, (In press).

[21] Allen, G., Benger, W., Dramlitsch, T., Goodale, T., Hege, H.-C., Lanfermann, G., Merzky, A., Radke, T., Seidel, E., and Shalf, J., "Cactus Tools for Grid Applications," *Cluster Computing*, Vol. 4, No. 3, 2001, pp. 179–188. doi:10.1023/A:1011491422534.

[22] Johnson, C., Parker, S., Weinstein, D., and Heffernan, S., "Component-based, problem-solving environments for large-scale scientific computing," *Concurrency and Computation: Practice and Experience*, Vol. 14, No. 13-15, 2002, pp. 1337–1349. doi:10.1002/cpe.693.

[23] Valiev, M., Bylaska, E., Govind, N., Kowalski, K., Straatsma, T., Dam, H. V., Wang, D., Nieplocha, J., Apra, E., Windus, T., and de Jong, W., "NWChem: A comprehensive and scalable open-source solution for large scale molecular simulations," *Computer Physics Communications*, Vol. 181, No. 9, 2010, pp. 1477 – 1489. doi:http://dx.doi.org/10.1016/j.cpc.2010.04.018.

[24] Gaston, D., Newman, C., Hansen, G., and Lebrun-Grandi, D., "MOOSE: A parallel computational framework for coupled systems of nonlinear equations," *Nuclear Engineering and Design*, Vol. 239, No. 10, 2009, pp. 1768 – 1778. doi:http://dx.doi.org/10.1016/j.nucengdes.2009.05.021.

[25] Bernholdt, D. E., Armstrong, R. C., and Allan, B. A., "Managing complexity in modern high end scientific computing through component-based software engineering," *In Proceedings. of the HPCA Workshop on Productivity and Performance in High-End Computing P-PHEC 2004*, IEEE Computer Society, 2004.

[26] Padula, S. L. and Gillian, R. E., "Multidisciplinary Environments: A History of Engineering Framework Development," *In Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics, 1801 Alexander Bell Drive, Suite 500, Reston, VA, 20191-4344, USA, URL:http://www.aiaa.org, Portsmouth, Virginia, 2006.

[27] Salas, A. O. and Townsend, J. C., "Framework Requirements For Mdo Application Development," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1998, pp. 98–4740.

[28] Martins, J. R. R. A., Marriage, C., and Tedford, N. P., "pyMDO: An Object-Oriented Framework for Multidisciplinary Design Optimization," *ACM Transactions on Mathematical Software*, Vol. 36, No. 4, Aug. 2009, pp. 20:1–20:25. doi:10.1145/1555386.1555389.

[29] Marriage, C. J. and Martins, J. R. R. A., "Reconfigurable Semi-Analytic Sensitivity Methods and MDO Architectures Within the πMDO Framework," *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, BC, Sept. 2008, AIAA 2008-5956.

[30] Griewank, A., *Evaluating Derivatives*, SIAM, Philadelphia, 2000.

[31] Naumann, U., *The Art of Differentiating Computer Programs — An Introduction to Algorithmic Differentiation*, SIAM, 2011.

[32] Bloebaum, C., "Global Sensitivity Analysis in Control-Augmented Structural Synthesis," *Proceedings of the 27th AIAA Aerospace Sciences Meeting*, Reno, NV, Jan. 1989, AlAA 1989-0844.

[33] Sobieszczanski-Sobieski, J., "Sensitivity of Complex, Internally Coupled Systems," *AIAA Journal*, Vol. 28, No. 1, 1990, pp. 153–160. doi:10.2514/3.10366.

[34] Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., "Aero-Structural Wing Design Optimization Using High-Fidelity Sensitivity Analysis," *Proceedings of the CEAS Conference on Multidisciplinary Aircraft Design and Optimization*, edited by H. Hölinger, Köln, Germany, June 2001, pp. 211–226.

[35] Maute, K., Nikbay, M., and Farhat, C., "Coupled Analytical Sensitivity Analysis and Optimization of Three-Dimensional Nonlinear Aeroelastic Systems," *AIAA Journal*, Vol. 39, No. 11, 2001, pp. 2051–2061.

[36] Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., "High-Fidelity Aerostructural Design Optimization of a Supersonic Business Jet," *Journal of Aircraft*, Vol. 41, No. 3, 2004, pp. 523–530. doi:10.2514/1.11478.

[37] Nykbay, M., Öncü, L., and Aysan, A., "Multidisciplinary Code Coupling for Analysis and Optimization of Aeroelastic Systems," *Journal of Aircraft*, Vol. 46, No. 6, Nov. 2009, pp. 1938–1944. doi:10.2514/1.41491.

[38] Broyden, C., "A Class of Methods for Solving Nonlinear Simultaneous Equations," *Mathematics of Computation*, Vol. 19, 1965, pp. 577–593.

[39] Knoll, D. and Keyes, D., "Jacobian-free Newton–Krylov methods: a survey of approaches and applications," *Journal of Computational Physics*, Vol. 193, No. 2, 2004, pp. 357 – 397. doi:http://dx.doi.org/10.1016/j.jcp.2003.08.010.

[40] Kelley, C. T., David, and Keyes, E., "Convergence analysis of pseudo-transient continuation," *SIAM Journal of Numerical Analysis*, Vol. 35, 1998, pp. 508–523.

[41] Grippo, L. and Sciandrone, M., "On the convergence of the block nonlinear GaussSeidel method under convex constraints," *Operations Research Letters*, Vol. 26, No. 3, 2000, pp. 127 – 136. doi:http://dx.doi.org/10.1016/S0167-6377(99)00074-7.

[42] Eisenstat, S. C. and Walker, H. F., "Globally Convergent Inexact Newton Methods," *SIAM Journal on Optimization*, Vol. 4, No. 2, 1994, pp. 393–422.

[43] Kelley, C. T., *Iterative Methods for Linear and Nonlinear Equations*, No. 16 in Frontiers in Applied Mathematics, SIAM, 1995.

[44] Knoll, D. A. and Rider, W. J., "A Multigrid Preconditioned Newton–Krylov Method," *SIAM Journal on Scientific Computing*, Vol. 21, No. 2, 1999, pp. 691–710.

[45] Cai, X.-C. and Sarkis, M., "A Restricted Additive Schwarz Preconditioner for General Sparse Linear Systems," *SIAM Journal on Scientific Computing*, Vol. 21, No. 2, 1999, pp. 792–797.

[46] Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F., "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries," *Modern Software Tools in Scientific Computing*, edited by E. Arge, A. M. Bruaset, and H. P. Langtangen, Birkhäuser Press, 1997, pp. 163–202.

[47] Cutler, J. W., Ridley, A., and Nicholas, A., "Cubesat Investigating Atmospheric Density Response to Extreme Driving (CADRE)," *Proceedings of the 25th Small Satellite Conference*, Logan, UT, August 2011.

[48] Boudjemai, A., Bouanane, M. H., Merad, L., and Si Mohammed, A. M., "Small Satellite Structural Optimisation Using Genetic Algorithm Approach," *Proceedings of the 3rd International Conference on Recent Advances in Space Technologies*, Istanbul, Turkey, 2007, pp. 398–406. doi:10.1109/RAST.2007.4284021.

[49] Galski, R. L., De Sousa, F. L., Ramos, F. M., and Muraoka, I., "Spacecraft thermal design with the Generalized Extremal Optimization Algorithm," *Inverse Problems in Science and Engineering*, Vol. 15, No. 1, 2007, pp. 61–75. doi:10.1080/17415970600573924.

[50] Jain, S. and Simon, D., "Genetic Algorithm Based Charge Optimization of Lithium-Ion Batteries in Small Satellites," *Proceedings of the 19th Annual AIAA/USU Conference on Small Satellites*, Logan, UT, August 2005.

[51] Richie, D. J., Lappas, V. J., and Palmer, P. L., "Sizing/Optimization of a Small Satellite Energy Storage and Attitude Control System," *Journal of Spacecraft and Rockets*, Vol. 44, No. 4, July 2007, pp. 940–952. doi:10.2514/1.25134.

[52] Zhang, B., Teng, H.-F., and Shi, Y.-J., "Layout optimization of satellite module using soft computing techniques," *Appl. Soft Comput.*, Vol. 8, No. 1, Jan. 2008, pp. 507–521. doi:10.1016/j.asoc.2007.03.004.

[53] Barnhart, D. A., Kichkaylo, T., and Hoag, L., "SPIDR: Integrated Systems Engineering Design-to-Simulation Software for Satellite Build," *Proceedings of the 7th Annual Conference on Systems Engineering Research*, Loughborough, UK, 2009.

[54] Fukunaga, A., Chien, S., Mutz, D., Sherwood, R., and Stechert, A., "Automating the Process of Optimization in Spacecraft Design," *Proceedings of the 1997 IEEE Aerospace Conference*, Vol. 4, Aspen, CO, 1997, pp. 411–427. doi:10.1109/AERO.1997.577524.

[55] George, J., Peterson, J., and Southard, S., "Multidisciplinary Integrated Design Assistant for Spacecraft (MIDAS)," *Proceedings of the 36th Structures, Structural Dynamics, and Materials Conference*, New Orleans, LA, 1995. doi:10.2514/6.1995-1372.

[56] Mosher, T., "Spacecraft design using a genetic algorithm optimization approach," *Proceedings of the 1998 IEEE Aerospace Conference*, Vol. 3, Aspen, CO, 1998, pp. 123–134. doi:10.1109/AERO.1998.685783.

[57] Stump, G., Yukish, M., Simpson, T., and O'Hara, J., "Trade space exploration of satellite datasets using a design by shopping paradigm," *Proceedings of the 2004 IEEE Aerospace Conference*, Vol. 6, 2004, pp. 3885–3895. doi:10.1109/AERO.2004.1368206.

[58] Ebrahimi, M., Farmani, M. R., and Roshanian, J., "Multidisciplinary design of a small satellite launch vehicle using particle swarm optimization," *Structural and Multidisciplinary Optimization*, Vol. 44, No. 6, 2011, pp. 773–784. doi:10.1007/s00158-011-0662-7.

[59] Jafarsalehi, A., Zadeh, P. M., and Mirshams, M., "Collaborative Optimization of Remote Sensing Small Satellite Mission using Genetic Algorithms," *Transactions of Mechanical Engineering*, Vol. 36, No. 2, 2012, pp. 117–128.

[60] Wu, W., Huang, H., Chen, S., and Wu, B., "Satellite Multidisciplinary Design Optimization with a High-Fidelity Model," *Journal of Spacecraft and Rockets*, Vol. 50, No. 2, March 2013, pp. 463–466. doi:10.2514/1.A32309.

[61] Braun, R. D. and Kroo, I. M., "Development and Application of the Collaborative Optimization Architecture in a Multidisciplinary Design Environment," *Multidisciplinary Design Optimization: State of the Art*, edited by N. Alexandrov and M. Y. Hussaini, SIAM, 1997, pp. 98–116.

[62] Martins, J. R. R. A. and Lambe, A. B., "Multidisciplinary Design Optimization: A Survey of Architectures," *AIAA Journal*, Vol. 51, No. 9, September 2013, pp. 2049–2075. doi:10.2514/1.J051895.

[63] Kreisselmeier, G. and Steinhauser, R., "Systematic Control Design by Optimizing a Vector Performance Index," *International Federation of Active Controls Syposium on Computer-Aided Design of Control Systems, Zurich, Switzerland*, 1979.

[64] Poon, N. M. K. and Martins, J. R. R. A., "An Adaptive Approach to Constraint Aggregation Using Adjoint Sensitivity Analysis," *Structural and Multidisciplinary Optimization*, Vol. 34, No. 1, 2007, pp. 61–73. doi:10.1007/s00158-006-0061-7.

[65] Kennedy, G. J. and Martins, J. R. R. A., "A Comparison of Metallic and Composite Aircraft Wings Using Aerostructural Design Optimization," *14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Indianapolis, IN, Sep. 2012, AIAA-2012-5475.

[66] Kawamura, H., Naka, K., Yonekura, N., Yamanaka, S., Kawamura, H., Ohno, H., and Naito, K., "Simulation of I-V characteristics of a PV module with

shaded PV cells," *Solar Energy Materials & Solar Cells*, Vol. 75, 2003, pp. 613–621.

[67] Larson, W. and Wertz, J., *Space mission analysis and design*, Kluwer Academic Publishers, January 1991.

[68] Haftka, R. T., "Simultaneous Analysis and Design," *AIAA Journal*, Vol. 23, No. 7, 1985.

[69] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Review*, Vol. 47, No. 1, 2005, pp. 99–131. doi:10.1137/S0036144504446096.

[70] Ferguson, A. R. and Dantzig, G. B., "The allocation of aircraft to routes-an example of linear programming under uncertain demand," *Management science*, Vol. 3, No. 1, 1956, pp. 45–73.

[71] Gass, S., "George B. Dantzig," *Profiles in Operations Research*, edited by A. A. Assad and S. I. Gass, Vol. 147 of *International Series in Operations Research & Management Science*, Springer US, 2011, pp. 217–240. doi:10.1007/978-1-4419-6281-2_13.

[72] Simpson, R. W., "Computerized schedule construction for an airline transportation system," *Department of Aeronautics and Astronautics, Massachusetts Institute of Technology*, Vol. 232, 1966.

[73] Abara, J., "Applying integer linear programming to the fleet assignment problem," *Interfaces*, Vol. 19, No. 4, 1989, pp. 20–28.

[74] Barnhart, C., Farahat, A., and Lohatepanont, M., "Airline fleet assignment with enhanced revenue modeling," *Operations research*, Vol. 57, No. 1, 2009, pp. 231–244.

[75] Dumas, J., Aithnard, F., and Soumis, F., "Improving the objective function of the fleet assignment problem," *Transportation Research Part B: Methodological*, Vol. 43, No. 4, 2009, pp. 466–475.

[76] Johnson, E. L., "Robust airline fleet assignment: imposing station purity using station decomposition," *Algorithmic Applications in Management*, Springer, 2005, pp. 1–2.

[77] Bélanger, N., Desaulniers, G., Soumis, F., Desrosiers, J., and Lavigne, J., "Weekly airline fleet assignment with homogeneity," *Transportation Research Part B: Methodological*, Vol. 40, No. 4, 2006, pp. 306–318.

[78] Lohatepanont, M. and Barnhart, C., "Airline schedule planning: Integrated models and algorithms for schedule design and fleet assignment," *Transportation Science*, Vol. 38, No. 1, 2004, pp. 19–32.

[79] Clarke, M. and Smith, B., "Impact of operations research on the evolution of the airline industry," *Journal of Aircraft*, Vol. 41, No. 1, 2004, pp. 62–72.

[80] Barnhart, C., Belobaba, P., and Odoni, A. R., "Applications of operations research in the air transport industry," *Transportation science*, Vol. 37, No. 4, 2003, pp. 368–391.

[81] Subramanian, R., Scheff, R. P., Quillinan, J. D., Wiper, D. S., and Marsten, R. E., "Coldstart: fleet assignment at delta air lines," *Interfaces*, Vol. 24, No. 1, 1994, pp. 104–120.

[82] Kontogiorgis, S. and Acharya, S., "US Airways automates its weekend fleet assignment," *Interfaces*, Vol. 29, No. 3, 1999, pp. 52–62.

[83] Betts, J. T., "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, Vol. 21, No. 2, 1998, pp. 193–207.

[84] Betts, J. T. and Huffman, W. P., "Application of sparse nonlinear programming to trajectory optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 1, 1992, pp. 198–206.

[85] Betts, J. T. and Cramer, E. J., "Application of direct transcription to commercial aircraft trajectory optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 1, 1995, pp. 151–159.

[86] Park, S. G. and Clarke, J.-P., "Vertical Trajectory Optimization for Continuous Descent Arrival Procedure," *AIAA Guidance, Navigation, and Control Conference, Minneapolis (Minnesota-USA), Paper*, Vol. 4757, 2012, p. 2012.

[87] Clarke, J.-P. B., Ho, N. T., Ren, L., Brown, J. A., Elmer, K. R., Zou, K., Hunting, C., McGregor, D. L., Shivashankara, B. N., Tong, K.-O., et al., "Continuous descent approach: Design and flight test for Louisville International Airport," *Journal of Aircraft*, Vol. 41, No. 5, 2004, pp. 1054–1066.

[88] Shresta, S., Neskovic, D., and Williams, S., "Analysis of continuous descent benefits and impacts during daytime operations," *8th USA/Europe Air Traffic Management Research and Development Seminar (ATM2009), Napa, CA*, 2009.

[89] Clarke, J.-P., Brooks, J., Nagle, G., Scacchioli, A., White, W., and Liu, S., "Optimized Profile Descent Arrivals at Los Angeles International Airport," *Journal of Aircraft*, Vol. 50, No. 2, 2013, pp. 360–369.

[90] Micallef, M., Chircop, K., Zammit-Mangion, D., and Sammut, A., "Revised approach procedures to support optimal descents into Malta International Airport," *CEAS Aeronautical Journal*, 2014, pp. 1–15.

[91] Kao, J. Y., Hwang, J. T., and Martins, J. R. R. A., "A Modular Approach for Mission Analysis and Optimization," *56th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2015 (accepted).

[92] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Journal of Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006. doi:10.1137/S1052623499350013.

[93] Gray, J., Moore, K. T., Hearn, T. A., and Naylor, B. A., "Standard Platform for Benchmarking Multidisciplinary Design Analysis and Optimization Architectures," *AIAA Journal*, Vol. 51, No. 10, 2013, pp. 2380–2394. doi:10.2514/1.J052160.

[94] Vassberg, J. C., DeHaan, M. A., Rivers, S. M., and Wahls, R. A., "Development of a Common Research Model for Applied CFD Validation Studies," 2008, AIAA 2008-6919.

[95] Liebeck, R. H., "Design of the Blended Wing Body Subsonic Transport," *Journal of Aircraft*, Vol. 41, No. 1, 2004.

[96] Crossley, W. and DeLaurentis, D., "System-of-systems approach for assessing new technologies in NGATS," *Purdue University*, 2007.

[97] Govindaraju, P. and Crossley, W. A., "Profit Motivated Airline Fleet Allocation and Concurrent Aircraft Design for Multiple Airlines," *Aviation Technology, Integration, and Operations Conference*, Aug 2013, AIAA 2013-4391.

[98] Gray, J., Hearn, T., Moore, K., Hwang, J. T., Martins, J. R. R. A., and Ning, A., "Automatic Evaluation of Multidisciplinary Derivatives Using a Graph-Based Problem Formulation in OpenMDAO," *Proceedings of the 15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Atlanta, GA, June 2014. doi:10.2514/6.2014-2042.

[99] ICAO Secretariat, "Aircraft technology improvements," *ICAO Environmental Report 2010*, International Civil Aviation Organization, 2010.

[100] Del Rosario, R., Follen, G., Wahls, R., and Madavan, N., "Subsonic Fixed Wing Project Overview of Technical Challenges for Energy Efficient, Environmentally Compatible Subsonic Transport Aircraft," *Proceedings of the 50th AIAA Aerospace Sciences Meeting*, 2012.

[101] The MIT, Aurora Flight Sciences, and Pratt & Whitney Team and Greitzer, E. M., "Volume 1: N+3 Aircraft Concept Designs and Trade Studies, Final Report," Tech. rep., MIT, 2010.

[102] Bradley, M. K. and Droney, C. K., "Subsonic Ultra Green Aircraft Research Phase," Tech. rep., NASA, 2012.

[103] Raymer, D. P., *Aircraft Design: A Conceptual Approach*, AIAA, 5th ed., 2012.

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

[105] Hahn, A. S., "Vehicle Sketch Pad: A Parametric Geometry Modeler for Conceptual Aircraft Design," *Proceedings of the 48th AIAA Aerospace Sciences Meeting*, Orlando, FL, Jan. 2010.

[106] Risse, K., Anton, E., Lammering, T., Franz, K., and Hoernschemeyer, R., "An Integrated Environment for Preliminary Aircraft Design and Optimization," *Proceedings of the 53rd AIAA Structures, Structural Dynamics and Materials Conference*, Honolulu, HI, April 2012, AIAA-2012-1675.

[107] Rodriguez, D. L. and Sturdza, P., "A Rapid Geometry Engine for Preliminary Aircraft Design," *Proceedings of the 44th AIAA Aerospace Sciences Meeting*, Reno, NV, January 2006, AIAA-2006-0929.

[108] LLC, A., "AVID PAGE," `http://www.avidaerospace.com/software/avid-page`, Accessed July 11, 2012.

[109] Hahn, A. S., "Application of Cart3D to Complex Propulsion-Airframe Integration with Vehicle Sketch Pad," *Proceedings of the 50th AIAA Aerospace Sciences Meeting*, Nashville, TN, January 2012, AIAA-2012-0547.

[110] Chaput, A. J. and Rizo-Patron, S., "Vehicle Sketch Pad Structural Analysis Module Enhancements for Wing Design," *Proceedings of the 50th AIAA Aerospace Sciences Meeting*, Nashville, TN, January 2012, AIAA-2012-0546.

[111] Kenway, G. K., Kennedy, G. J., and Martins, J. R. R. A., "A CAD-Free Approach to High-Fidelity Aerostructural Optimization," *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Fort Worth, TX, Sept. 2010, AIAA 2010-9231.

[112] Talbert, J. A. and Parkinson, A. R., "Development of an automatic, two-dimensional finite element mesh generator using quadrilateral elements and Bezier curve boundary definition," *International Journal for Numerical Methods in Engineering*, Vol. 29, No. 7, 1990, pp. 1551–1567. doi:10.1002/nme.1620290712.

[113] Tam, T. and Armstrong, C., "2D finite element mesh generation by medial axis subdivision," *Advances in Engineering Software and Workstations*, Vol. 13, No. 56, 1991, pp. 313 – 324. doi:http://dx.doi.org/10.1016/0961-3552(91)90035-3.

[114] Blacker, T. D. and Stephenson, M. B., "Paving: A new approach to automated quadrilateral mesh generation," *International Journal for Numerical Methods in Engineering*, Vol. 32, No. 4, 1991, pp. 811–847. doi:10.1002/nme.1620320410.

[115] Owen, S. J., Staten, M. L., Canann, S. A., and Saigal, S., "Q-Morph: An indirect approach to advancing front quad meshing," *International Journal for Numerical Methods in Engineering*, Vol. 44, 1999, pp. 1317–1340.

[116] Bunin, G., "Non-Local Topological Clean-Up," *Proceedings of the 15th International Meshing Roundtable*, Springer Berlin Heidelberg, 2006, pp. 3–20. doi:10.1007/978-3-540-34958-7_1.

[117] Canann, S., Muthukrishnan, S., and Phillips, R., "Topological improvement procedures for quadrilateral finite element meshes," *Engineering with Computers*, Vol. 14, No. 2, 1998, pp. 168–177. doi:10.1007/BF01213591.

[118] Verma, C. and Tautges, T., "Jaal: Engineering a High Quality All-Quadrilateral Mesh Generator," *Proceedings of the 20th International Meshing Roundtable*, edited by W. Quadros, Springer Berlin Heidelberg, 2012, pp. 511–530. doi:10.1007/978-3-642-24734-7_28.

[119] Xu, H. and Newman, T., "2D FE Quad Mesh Smoothing via Angle-Based Optimization," *Computational Science ICCS 2005*, edited by V. Sunderam, G. Albada, P. Sloot, and J. Dongarra, Vol. 3514 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2005, pp. 9–16. doi:10.1007/11428831_2.

[120] Jang, B.-S., Suh, Y.-S., Kim, E.-K., and Lee, T.-H., "Automatic {FE} modeler using stiffener-based mesh generation algorithm for ship structural analysis," *Marine Structures*, Vol. 21, No. 23, 2008, pp. 294 – 325. doi:http://dx.doi.org/10.1016/j.marstruc.2007.08.001.

[121] Lee, K.-Y., Kim, I.-I., Cho, D.-Y., and wan Kim, T., "An algorithm for automatic 2D quadrilateral mesh generation with line constraints," *Computer-Aided Design*, Vol. 35, No. 12, 2003, pp. 1055 – 1068. doi:http://dx.doi.org/10.1016/S0010-4485(02)00145-8.

[122] Park, C., Noh, J.-S., Jang, I.-S., and Kang, J. M., "A new automated scheme of quadrilateral mesh generation for randomly distributed line constraints," *Computer-Aided Design*, Vol. 39, No. 4, 2007, pp. 258 – 267. doi:http://dx.doi.org/10.1016/j.cad.2006.12.002.

[123] Renka, R. J., "Algorithm 751: TRIPACK: a constrained two-dimensional Delaunay triangulation package," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 22, No. 1, 1996, pp. 1–8.

[124] Parthasarathy, V. and Kodiyalam, S., "A constrained optimization approach to finite element mesh smoothing," *Finite Elements in Analysis and Design*, Vol. 9, No. 4, 1991, pp. 309–320.

[125] Thompson, J. F., Thames, F. C., and Mastin, C., "Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies," *Journal of Computational Physics*, Vol. 15, No. 3, 1974, pp. 299 – 319. doi:http://dx.doi.org/10.1016/0021-9991(74)90114-4.

[126] Field, D. A., “Laplacian smoothing and Delaunay triangulations,” *Communications in applied numerical methods*, Vol. 4, No. 6, 1988, pp. 709–712.

[127] Li, X. S. and Demmel, J. W., “SuperLU_DIST: A Scalable Distributed-Memory Sparse Direct Solver for Unsymmetric Linear Systems,” *ACM Trans. Mathematical Software*, Vol. 29, No. 2, June 2003, pp. 110–140.

[128] Kennedy, G. J. and Martins, J. R. R. A., “A Parallel Finite-Element Framework for Large-Scale Gradient-Based Design Optimization of High-Performance Structures,” *Finite Elements in Analysis and Design*, Vol. 87, September 2014, pp. 56–73. doi:10.1016/j.finel.2014.04.011.

[129] Brown, S. A., “Displacement Extrapolation for CFD+CSM Aeroelastic Analysis,” *Proceedings of the 35th AIAA Aerospace Sciences Meeting*, Reno, NV, 1997, AIAA 1997-1090.