SHAPE OPTIMIZATION IN ADAPTIVE SEARCH SPACES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF AERONAUTICS & ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

George R. Anderson
December 2015

Abstract

Design space dimensionality is a persistent obstacle to high-fidelity aerodynamic shape optimization. The burden typically falls to the designer to select the shape design variables, which restricts the design space and impacts computational efficiency. In this work, I develop a system that automatically adapts the shape parameterization to efficiently solve the given design problem. Optimization begins with a small set of design variables, which is then progressively refined to enable the discovery of superior designs. The adaptation is goal-oriented and is driven by a novel refinement indicator that focuses shape control resolution on regions with the highest potential to improve the design. This indicator is computed from adjoint sensitivity information and a quasi-Newton Hessian approximation, which are both readily available in many design frameworks.

To verify that the system can autonomously discover parameters sufficient to solve a problem, three inverse design problems are examined. For these cases, the procedure robustly converges to the continuous optimum from different starting points and with different refinement strategies. Several airfoil and wing shape optimization problems are then performed. The adaptive approach generally yields smoother design trajectories, at substantially lower cost than under fixed or uniformly-refined parameterizations. Finally, I consider the combinatorial problem of finding an optimal layout for an adaptive, trailing edge flap system. In fewer than 100 simulations, the adaptive procedure discovers a flap layout that outperforms naive layouts. I conclude with a discussion of the tremendous potential this approach offers for many other disciplines and design environments.

ACKNOWLEDGMENTS

This work is deeply indebted to:

- Mike Aftosmis, mentor, advocate, tireless and inspiring researcher, trove of idiomatic speech, and deeply human being.
- Antony Jameson, compassionate advisor and teacher.
- The NASA ARMD NARI Seedling Fund, lenders of critical support for this research.
- Marian Nemec and David Rodriguez, patient colleagues and co-authors.
- Bradley and Susan Anderson, progenitors extraordinaire; Elizabeth and Thomas Anderson, siblings resplendent.
- My dearest friends: Dorian Jackman, expleter and relentless friend; Thomas Marnin, observer of things; Alessandra Aquilanti, standpartner for life; Anna Polishchuk, fairy friend.
- Elan Dagenais, my companion, helpmeet, and little one.

Nomenclature

Z — Scalar
$\mathbf{Z}$ — Vector
$\overline{\mathbf{Z}}$ — Matrix
Z — Continuous surface vector
$\mathcal{Z}$ — Function

A	Area footprint of deformation mode
$\mathbf{b}$	Design variable bounds
$\overline{\mathbf{B}}$	Quasi-Newton Hessian approximation
$C, \mathbf{C}$	Shape control description
$C_{D/L/M}$	Drag/lift/moment coefficient
$\mathcal{C}, \boldsymbol{\mathcal{C}}$	Constraint functional(s)
d	Tree search depth
$\mathcal{D}$	Shape deformation function
$\mathcal{D}_{\mathcal{H}}$	Differential part of $\mathcal{H}$
$\mathcal{F}$	Design functional ($\mathcal{J}$ or $\mathcal{C}$)
g	Growth rate in N_{DV}
h	Height of deformation mode
$\overline{\mathbf{H}}/\mathcal{H}$	Objective Hessian matrix/operator
$\overline{\mathbf{I}}/\mathcal{I}$	Identity matrix/operator
I	Importance indicator
$\mathcal{J}$	Objective functional
K	Static part of $\mathcal{H}$
M	Hessian prolongation scale term
$N_{(\cdot)}$	Number of $(\cdot)$
$\mathcal{O}$	Asymptotic complexity
$\mathcal{P}$	Shape parameterization function
r	Slope reduction factor for trigger
S	Shape
$\mathbf{S}$	Discrete surface
w	Window
$X, \mathbf{X}$	Design variable value(s)
α	Angle of attack
$\boldsymbol{\lambda}$	Lagrange multipliers
$\boldsymbol{\Theta}$	Operating conditions
ψ_o/ψ_j	Adjoint solutions (objective/constraints)

Subscripts

$(\cdot)_i$	Design iteration
$(\cdot)_\star$	Optimal design
$(\cdot)_G$	First-order (gradients only)
$(\cdot)_H$	Second-order (Hessian)
$(\cdot)_D$	Second-order, diagonal only

Superscripts

$(\cdot)^a$	Active constraints
$(\cdot)^c$	Candidate shape control
$(\cdot)^k$	Shape parameterization level
$(\cdot)^\infty$	Continuous shape control
$(\cdot)^\times$	Static shape control
$(\cdot)^*$	Target value
$(\cdot)^{\cancel{\mathcal{C}}}$	No constraints
$(\cdot)^{\cancel{\mathcal{H}}}$	Ignoring Hessian

Abbreviations

BFGS	Broyden-Fletcher-Goldfarb-Shanno
CAD	Computer-aided design
DV	Design variable
GSM	Geometric shape matching
KKT	Karush-Kuhn-Tucker conditions
OSD	Optimal shape design
RBF	Radial basis function

Contents

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

Modern engineering design increasingly employs numerical optimization, which aims to find the best design for a given set of requirements. In aerodynamic design endeavors, this usually involves optimizing a surface, whose shape directly determines its performance. The full shape optimization problem is continuous, usually without an analytic solution, and is solved numerically by projecting it into a finite set of design variables via *parameterization* of the shape modifications. This choice of design variables impacts the robustness, completeness and efficiency of the subsequent optimization. In standard approaches, whether the parameterization is specified by the designer or automatically generated, it remains *fixed* throughout optimization. This work examines automating the construction and refinement of the shape parameterization, resulting in a substantially more autonomous, robust, and efficient design system.

1.1 Motivation

Optimal Shape Design (OSD) is typically phrased as finding the shape that maximizes some measure of performance. In most design settings, however, OSD is used with somewhat less ambitious, more practical goals in mind:

1. Maximally improving the performance of a design *within limited resources*.

2. Deepening the designer's understanding of the design space.

The shape parameterization is intimately intertwined with each of these goals. Consider the first goal. Except for certain academic cases, a designer rarely expects to find the true continuously optimal shape. Instead, a finite approximation of the problem is created via selection of a set of design variables. The credibility of the resulting pseudo-optimal shape is uncertain. At the end of optimization, there is no indication of how much more performance might be attainable if other degrees of freedom were used.

The computational efficiency of this process is also suspect. The rate of design improvement per simulation depends on two competing factors. To achieve better designs, more flexibility must be added. However, as the dimensionality increases, the search space generally becomes more costly to navigate. Either extreme (too few or too many design variables) can lead to sluggish design improvement. Historically, this has led researchers to develop a spectrum of optimization approaches, from rapid, low-dimensional explorations (e.g. [1–4]) to high-dimensional approaches that approximate the continuous problem by brute force (e.g. [5, 6]). In virtually all approaches, the choice of shape control resolution (and perhaps distribution) is relegated to the designer, who must strike a compromise between capacity for overall design improvement and speed.

A key observation of this thesis is that this tradeoff between completeness and efficiency is an artificial conflict arising from the use of *static* parameterizations. This thesis develops a progressive approach to parameterization, where the spectrum of dimensionality is automatically traversed. In addition to enabling rapid design improvement early in optimization, this approach is convergent to the *continuous* locally optimal shape, not merely to some finite approximation. It also removes the need for user-in-the-loop reparameterization, and it helps reduce the dependence of the final result on the designer's skill at crafting a parameterization.

Now consider Goal (2). All-encompassing, fully-automated aircraft design systems are a lofty but distant dream. Instead, OSD is viewed more as a tool for helping the designer better comprehend the design space. As a goal-oriented process, OSD can typically uncover non-obvious trends much more rapidly than impartial parametric studies. In pursuit of comprehensible feedback, many designers constrain the design space through the use of "intuitive" design variables. By exercising meticulous control over the parameterization, they frame the problem such that the results are more relatable to previous experience.

The obvious criticism of this approach is that it biases the optimization and exacerbates the tendency to produce familiar but suboptimal designs. This is only partially a fair criticism. Any given *distribution* of design variables does arbitrarily restrict the optimization. Nevertheless, it can be highly fruitful to restrict optimization to a certain intuitive shape control *class*. This system developed in this thesis allows the designer to specify a basis for shape modifications, while the actual construction and refinement of the design variables within that basis is automated. This preserves the comprehensibility of the results, while alleviating the bias and restrictions of choosing an arbitrary set of design variables.

Beyond merely controlling the dimension of the search space, I also examine localized, adaptive refinement of the search space. I show that sufficient information can be extracted during optimization to enable an autonomous system to intelligently enrich the shape parameterization and improve its spatial distribution. This has the potential to achieve faster and more robust design improvement compared to "obvious", but non-expert-crafted parameterizations. Moreover, it provides yet more feedback in the form of the refinement pattern. Ultimately, the hope is to approach (or perhaps surpass) expert performance with an automated system. Like any automated system, however, the goal of this approach is primarily to make OSD simpler, more robust, and more informative.

1.2 Background

This work focuses on detailed, high-fidelity parametric shape optimization. As one important goal is to support convergence to the continuously optimal shape, which necessarily ultimately requires very high-dimensional search spaces, I will make use of gradient-based methods and adjoint approaches. Optimization will generally be driven by high-fidelity simulations, or with variable fidelity simulations to save cost. The primary thrust of the thesis is related to controlling the resolution and distribution of shape control, which will motivate the use of adaptive, progressive parameterization. An important goal of the implementation is to support a wide variety of design frameworks, simulation tools, and geometry modelers. Therefore, I will aim to minimize modifications to existing design codes. The following sections review the various components of this approach, as well as some alternatives.

1.2.1 Parametric Shape Optimization

From the earliest days [7], the most prevalent approaches to OSD have been "parametric", i.e. involving higher level shape parameters, often ones that are intuitive to designers. By decoupling the shape control from the simulation meshes, parametric approaches can simplify the development of modular optimization architectures. Shape manipulation is performed by a geometry modeler, which provides the optimizer with a set of adjustable shape parameters to be used as design variables. During optimization, the geometry modeler interprets each requested set of parameter values and instantiates a fully-realized surface geometry that is appropriate for high-fidelity analysis. Examples of geometry modelers include CAD systems [8], parametric constructive solid modelers like EGADS [9] and OpenCSM [10], aerospace-focused modelers such as RAGE [11], OpenVSP [12], and MICADO [13], plus innumerable in-house codes. These tools are built upon libraries of low-level shape constructors (e.g. polynomials and splines), boolean operations, and deformation techniques (e.g. lattices, bump functions).

A major drawback to parametric design is that it introduces additional user decisions: how to construct the shape parameters, how many variables to use, and where to place them. This process can be labor intensive, involving trial-and-error predictions of the degrees of freedom that most efficiently span the relevant region of the design space. Manually constructed parameterizations permit only limited improvement, and the final result is sensitive to the initial choice of design variables. The designer must therefore frequently manually re-parameterize the shape, either by naive uniform refinement [14–16] or by selective refinement that reflects emerging knowledge about the problem [17]. This is a difficult task, because there is no direct indication of how best to redirect optimization effort to further improve the design. This work aims to automate this process of constructing a shape parameterization, while remaining within the paradigm of parametric design. To allow the designer to specify custom shape control bases, I aim to keep the system as "modeler-agnostic" as possible.

1.2.2 The Adjoint Method

The demonstrations in this work will involve aerodynamic design subject to the inviscid flow equations, although the arguments and framework apply equally to other disciplines. In this context, adjoint-derived objective and constraint gradients are essential for efficiency. The adjoint approach removes the need for finite-differenced flow solutions. The sensitivity of the design functionals to the continuous surface is obtainable for the cost of only one additional linear PDE solution per functional.

Stemming from early theoretical works in optimal control [18, 19], adjoint methods are now widely used in fields as disparate as meteorology [20], geophysics [21], environmental monitoring [22], computer graphics [23, 24], and finance [25]. Pironneau first investigated optimal shape design with respect to Stokes flow [26], Navier-Stokes flow [27] and elliptic systems [28]. Other early applications of the adjoint approach involved structural sizing optimization (e.g. [29]). Jameson applied the technique to Euler and Navier-Stokes flow conditions, and demonstrated the tremendously reduced cost of aerodynamic shape optimization [30, 31], initiating an extremely productive period of research in shape design techniques (see e.g. [32]). The adjoint approach led to the development of efficient, monolithic one-shot methods [33–35], in which all the subcomponents of the optimization loop are tightly coupled. It also dramatically accelerated parametric shape optimization approaches, which previously had been severely limited in scope by the cost of finite-differenced gradients. Under the adjoint approach, arbitrary numbers of design variables can be used, with negligible increase in cost. Adjoint solvers are now a standard component of many aerodynamic design codes in research and industry, where they enable efficient gradient-based shape optimization with large numbers of design variables both for purely aerodynamic [36, 37] and for multidisciplinary problems [38]. Adjoint methods are also used to compute estimates of discretization error and to drive goal-oriented adaptive mesh refinement [39].

Although not previously used in this manner, the adjoint can be leveraged to guide the construction of efficient search spaces for optimization. In typical parametric shape optimization, only a small subset of all the sensitivity information encoded in the adjoint solution is used. In Chapter 3, I show how to use adjoint solutions in a novel manner to compare hypothetical shape parameters and determine the *most important* ones to solve the given problem. This enables automatic adaptation of the shape control to target the particular goals of the optimization.

1.2.3 Optimization Approaches

As the goal is to perform shape optimization driven by expensive simulations and in search spaces of arbitrarily high dimension (approaching continuous), it is essential to minimize the number of functional evaluations. Therefore for this work I use a gradient-based approach, which is dramatically more efficient than gradient-free approaches, at least for smooth problems. For popular gradient-based approaches (e.g. BFGS), the number of simulations required for optimization is formally linearly proportional to the number of design variables [40, 41]. Gradient-free methods perform much worse (see e.g. [42, 43]) and are not generally recommended for high-dimensional optimization [44]. A consequence of using gradient-based optimization is that the approach developed here guarantees only local optimality for non-convex design spaces.

This work focuses on reduced-space, quasi-Newton optimization — perhaps the most widely used approach in gradient-based aerodynamic design. One reason for the broad uptake of quasi-Newton approaches is the availability of many black-box optimizers, which makes the simulation tools more interchangeable. Other gradient-based approaches to optimization include full-space methods [45] and inexact Newton methods [46], each with particular advantages. Because these approaches involve substantial invasive modification of simulation tools, I do not consider them in this work. However, Chapter 7 briefly discusses how adaptive parameterization might still be useful in those settings.

1.2.4 Order Reduction

The number of degrees of freedom in the parameterization and their distribution directly dictate the best achievable design. Although this has long been recognized (e.g. [33]), it is unfortunately frequently ignored in applied design settings, though for understandable reasons. In analysis, accurate analysis or quantification of error in an aerospace vehicle's performance may quite literally be a matter of life and death. In optimization, there is less impetus for accuracy; any design improvement is valuable, and it is not usually critical that the final result be optimal in a global, continuous sense. Nevertheless, there has recently accrued a substantial body of studies examining the dependence of the final design's performance (and the optimization efficiency) on both the number of degrees of freedom [4, 46–56][1] and (less commonly) the shape control distribution [5, 47, 58]. These studies are primarily concerned with empirical trends. Direct estimation of the degree of suboptimality appears to be confined to simpler model PDES (especially elliptic problems [59, 60]).

The goal of order-reduction is to reduce the cost of optimization, while retaining the ability to gain the majority of the possible design improvement. Many approaches to order-reduction are guided by a preliminary analysis of a sampling of the design space. Examples include proper orthogonal decomposition [61, 62], principal components analysis [63], or active subspaces [2]. In these approaches, a higher-dimensional search space is sampled, from which a lower-dimensional manifold is extracted, accounting for the majority of the variation in the design space. In tandem with surrogate models (e.g. [2, 64–67]), these approaches can be effective at capturing low-order trends. However, their cost rapidly balloons for high-dimensional design, making them inappropriate in the context of seeking a continuous optimum. In this work, we are seeking a procedure that is both efficient and able to converge to a continuous optimal solution.

[1] Also in other fields, such as computer graphics [57].

1.2.5 Progressive Parameterization

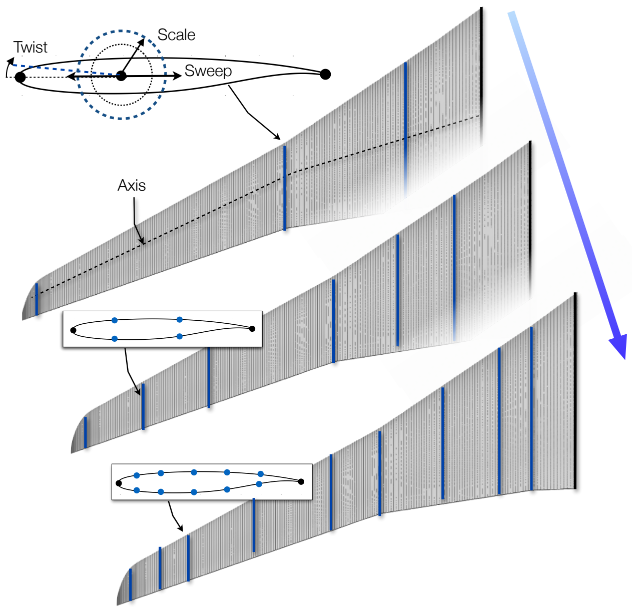


Figure 1.1: Notional parameterization refinement for wing design.

This work adopts a *progressive* parameterization approach, where optimization begins in a low-dimensional search space. As the design evolves, higher-resolution shape control is gradually added, as illustrated in Figure 1.1. The basic idea is inspired by *h*-refinement in multilevel methods. Rapid design improvement is encouraged by using compact search spaces early on; the search space is then enriched only when necessary to further improve the design, and if resources permit. In the limit of shape control refinement, the full, continuous design space becomes available for exploration. In contrast to a static parameterization approach, which strikes a compromises between efficiency and completeness, a progressive approach constitutes a single design process that drives the shape towards a local optimum of the continuous problem, while still retaining efficiency. Additionally, this approach ensures that the final optimized shape depends only on the problem specification, while being robust with respect to the initial shape parameterization.[2]

Progressive shape parameterization has its deepest roots in structural optimization. Early work focused on adaptation of the *solution* grid to accelerate optimization [68–70] but soon also considered refinement of the parameterization [71]. The first multilevel parameterization technique for aerodynamic shape optimization was developed at INRIA in 1993 by Marco, Beux, and Dervieux [72, 73]. A series of subsequent papers from the same group demonstrated that substantial design acceleration can be achieved with nested Bézier curves or free-form deformation [47, 74–79]. Multiresolution subdivision surfaces have also been used for progressive parameterization [80]. Coarse-to-fine sequencing is also a very natural technique even for designer-in-the-loop approaches [14–16].

Hwang and Martins developed a progressive approach where Hessian information is exactly transferred when refining the parameterization, which avoids repeating the initial

[2]The presence of local optima can affect this.

Hessian build-up time at each transition [81]. However, to do so, they require that the *finest* parameterization be provided a priori. Although they likewise observe computational acceleration over uniform refinement, many of the other advantages of progressive parameterization are lost. The designer must construct the finest permissible parameterization, which limits the ability to approach the continuous problem. Second, the number of shape sensitivity computations and gradient projections is always high, which can increase wall-clock time and storage requirements for very large numbers of design variables.

Sequencing and Multigrid

Multilevel shape optimization is closely related to work on multilevel PDE solution techniques. Certain authors argue that optimization is inherently an "anti-smoothing" process [82] and that design space sequencing is analogous to preconditioning the optimization [48, 76]. Making an analogy to grid sequencing and multi-grid techniques in PDE solutions, they find that a sequence of refined design spaces performs substantially better than a fine, fixed parameterization [48, 79].

Multilevel optimization can be performed either using a straightforward sequencing approach (which I use in this work) or using a more involved multigrid algorithm. Results are inconclusive on whether a more involved multigrid parameterization approach is appreciably faster than sequencing [48, 78, 83], although results for simpler model PDES are promising [84, 85].

1.2.6 Adaptive Refinement

Most research on progressive parameterization involves a simple predetermined sequence of parameterizations, with uniformly distributed shape control. The results universally show design acceleration, which strongly justifies the use of parameterization sequencing for optimization. A much smaller number of groups have investigated adaptive refinement, where the goal is to achieve a more optimal *distribution* of the shape control. Adaptive refinement seeks to add only the most important design variables to solve the given problem, thus reducing the overall dimension of the search. It is conjectured that this should accelerate quasi-Newton (and slower) optimization approaches, which generally converge faster with fewer degrees of freedom, so long as those degrees of freedom have high potential. There have been various approaches to adaptive shape control refinement, but moderate additional reduction in computational cost is typically observed [47, 50, 75, 77, 79, 81, 86]. More detail on these approaches will be given in Chapter 3.

Refinement vs. Redistribution

An important alternative (or supplement) to progressive refinement is to seek a better distribution of the existing shape control, which I will call "*r*-refinement" by analogy to adaptive meshing. The results in [77] suggest that redistribution can be more effective than naive enrichment. However, as demonstrated in [58], making this approach goal-oriented requires tedious and extensive differentiation of the parameterization function itself (not just the deformation function), which may not be feasible for many geometry modelers.

Regardless, under a given shape control basis, *r*-refinement alone is insufficient to converge to the continuous optimal shape. At some point, the capacity of the fixed-dimension search space will be exhausted, and the number of design variables must be increased to continue design improvement.[3] In this work I develop an *h*-refinement approach, because it is both necessary (excluding the possibility of adapting the shape control basis itself) and sufficient to converge to the continuous optimal shape. However, further investigation of redistribution is certainly warranted, as it offers significant complementary potential.

1.3 Contributions

The central goal of this work is to develop a system that automatically discovers the necessary shape control to solve a problem. The broad contributions of this thesis are twofold:

1. A complete system for solving aerodynamic shape optimization problems using goal-oriented adaptive shape control, with (nearly) arbitrary geometry modelers.

2. Validation of the system on problems with known solutions, and evaluation of its performance on practical design problems.

Taken together, these aim to establish adaptive shape control as a robust, automated, and efficient approach to aerodynamic shape design. The major specific contributions of this work are outlined below.

I have developed novel approaches to several independent, essential components of any adaptive shape control system:

[3]Discrete surface-based deformational modelers could technically bypass this restriction, by periodically reestablishing the current shape as the new baseline. However, even in this case, the discrepancies between the current shape and the (unknown) optimal shape consist of higher and higher frequencies, suggesting that redistribution of a small number of low-frequency controllers would cause the asymptotic convergence to the continuous optimum to severely stagnate.

1. A goal-oriented adaptation criterion based on "expected design improvement", that improves upon previous approaches by explicitly accounting for constraints, variation in design space curvature, and redundancy.
2. A method for estimating curvature in a hypothetical search space via prolongation of the quasi-Newton Hessian approximation from the previous space.
3. An efficient adaptation strategy, comprised of:
    - An approximate constructive algorithm for efficiently searching the combinatorial space of possible refinements.
    - Cost-benefit approaches to automatically determine strategic moments to refine the parameterization and tune the growth rate in the number of design variables.

Another important component of this thesis is a series of studies that verify the correctness of each of the system's components in isolation. I show that the adaptive system is able to independently, robustly, and efficiently navigate to the continuous optimal design from different starting points and with different refinement strategies. To my knowledge, recovery of continuously optimal shapes with parametric shape optimization techniques has not been previously demonstrated. In addition, I provide several design examples that compare the efficiency of the entire integrated system to that of traditional optimization, demonstrating major advantages to using progressive shape control.

1.4 Thesis Organization

Chapter 2 presents a basic framework for progressive shape control with discrete refinement mechanics. Chapter 3 develops an approach for shape control *adaptation*, including derivation of a novel goal-oriented refinement indicator and discussion of efficient adaptation strategies. Chapter 4 shows how the system supports nearly arbitrary geometry modelers, but also discusses the impact those choices can have on efficiency and robustness. Chapter 5 verifies the accuracy and robustness of the approach on three inverse design problems with known answers. Chapter 6 evaluates the effectiveness of adaptive parameterization on several more practical design examples. Though exercised here in the narrow capacity of aerodynamic design under inviscid conditions, this approach also has tremendous potential to accelerate and automate shape parameterization in many other design settings. Chapter 7 discusses some of these and offers recommendations for further investigation.

CHAPTER 2

Progressive Parameterization with Discrete Refinement

Consider an aerospace shape optimization problem that consists of finding a shape S and possibly a set of variable operating conditions $\boldsymbol{\Theta}$ that minimize a single objective, subject to design constraints:

$$\begin{aligned} &\min_{\mathsf{S},\boldsymbol{\Theta}} \mathcal{J}(\mathsf{S},\boldsymbol{\Theta}) \\ &\text{s.t. } a \leq \mathcal{C}_j(\mathsf{S},\boldsymbol{\Theta}) \leq b \end{aligned} \tag{2.1}$$

where $\mathcal{J}$ and $\mathcal{C}_j$ are scalar functionals that aggregate performance metrics such as lift, drag, static margin, maneuver loads, or wing volume, or more specialized concerns such as reducing sonic boom ground signatures. The functionals may involve integration over multiple flight conditions (multipoint design), which also means that Equation (2.1) encompasses certain probabilistic or robust optimization approaches [87, 88].

This work considers design only of the external, aerodynamically exposed surface. The term “shape” therefore denotes the external surface. However, the approach developed here would be equally applicable to aerostructural design, where the shape description might include internal geometry, such as spars and ribs.

The values of $\mathcal{J}$ and $\mathcal{C}_j$ (collectively $\mathcal{F}_j$) are ultimately determined by the shape and design conditions, either explicitly or implicitly via the state variables $\mathbf{Q}$, i.e. $\mathcal{F}(\mathsf{S},\boldsymbol{\Theta},\mathbf{Q}(\mathsf{S},\boldsymbol{\Theta}))$.

For gradient-based optimization, the derivatives of each output functional must be computed with respect to the shape and variable design conditions. In the case of aerodynamic functionals, these gradients are most efficiently computed using an adjoint approach.

2.1 Shape Parameterization

The variable operating conditions $\mathbf{\Theta}$ typically comprise a small discrete set of design variables (e.g. Mach number, altitude, trim or throttle settings, or angle of attack). However, the shape S is continuous, and so the full design space is infinitely dimensional. To reduce the search space to a manageable dimension, the surface modifications are usually parameterized.

A shape parameterization technique $\mathcal{P}$ is a map from a vector $\mathbf{C}$ describing the shape control to a deformation[1] function $\mathcal{D}(\mathbf{X})$:

$$\textbf{(Parameterize)} \qquad \mathcal{P} : \mathbf{C}, \mathsf{S}_0 \to \mathcal{D}(\mathbf{X}) \tag{2.2}$$

The shape parameters $\mathbf{X}$, or a subset thereof, serve as the design variables[2], which define the search space for optimization. During optimization, the deformation function takes the design variable values and generates a new surface:

$$\textbf{(Deform)} \qquad \mathcal{D} : \mathbf{X} \to \mathsf{S} \tag{2.3}$$

The range of $\mathcal{D}$ is the set of all reachable shapes, which is a subset of the continuous shape design space. The local linearization of $\mathcal{D}$ about the current shape provides the shape derivatives $\frac{\partial \mathsf{S}}{\partial \mathbf{X}}$, which describe the deformation mode of each parameter. These shape derivatives are used in gradient-based optimization, allowing projection of the adjoint-derived functional gradients with respect to the surface, $\frac{\partial \mathcal{F}}{\partial \mathsf{S}}$, into the lower-dimensional search space spanned by the shape parameters.

In standard design frameworks, $\mathcal{D}$ represents a call to an automated geometry modeling tool, while $\mathcal{P}$ is usually a manual pre-processing step.[3] In this work, $\mathcal{P}$ must now be automated as well; the geometry modeler must be able to automatically re-parameterize the deformations.

[1]In this work I happen to use shape *deformation* techniques, where modifications of an existing surface are parameterized, but this discussion is also fully applicable to constructive (CAD-like) modeling paradigms.

[2]$\mathbf{X}$ are often called "control variables" in the literature on PDE-constrained optimization. This should not be confused with the term "shape control", used here to indicate *where* the shape can be controlled.

[3]In practice, these two steps may be combined into a single function call $\mathcal{D}(\mathbf{C}, \mathsf{S}_0, \mathbf{X})$, where it is understood that only $\mathbf{X}$ is modified during optimization.

The difference between the shape *control* **C** and the shape *parameters* **X** is a subtle distinction that becomes important only in the context of progressive parameterization [58, 77, 86]. To help make this distinction more concrete, consider the following example.

Example: Wing Planform Design

Figure 2.1 illustrates a wing planform parameterization scheme where twist, sweep and chord are interpolated between spanwise stations (blue lines). The shape control **C** is a vector of spanwise coordinates of the control stations, indicating *where* twist, sweep and chord can be manipulated. The parameterization function $\mathcal{P}_{wing}$ expands this compact, high-level description into precise descriptions of the deformation modes. These are encoded in the deformation function $\mathcal{D}$, which takes the twist, sweep and chord values **X** and generates a new surface. To refine the shape control, new spanwise stations are added, introducing new degrees of freedom.

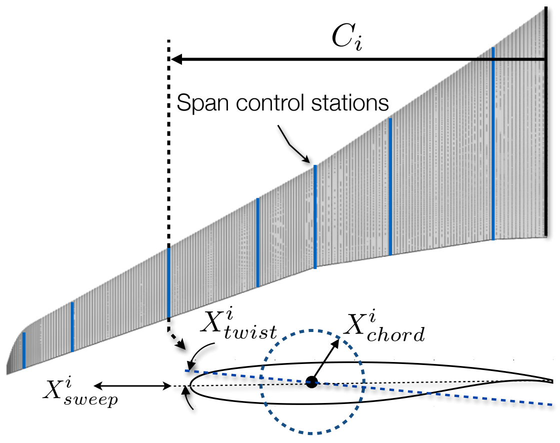


Figure 2.1: Wing planform parameterization

In the example above, each deformation mode shape is determined not only its location C_i, but also by the locations of the neighboring stations, C_{i+1} and C_{i-1}, which also affect the interpolation. In this and many other situations, there is not a one-to-one correspondence between the shape control and the shape parameters. This point will be important throughout this work.

2.1.1 Static vs. Progressive Shape Control

In standard shape optimization approaches, the shape control $\mathbf{C}^{\times}$ is pre-determined by the designer and remains static, unless the designer manually reparameterizes the shape. Optimization takes place in a fixed search space $\mathcal{D}^{\times}(\mathbf{X}^{\times})$, which may be more or less effective at improving the objective function, due to the unavoidable tradeoff between completeness and efficiency. The space of all reachable shapes is prescribed before each optimization begins, which can needlessly prevent the discovery of superior designs outside this envelope. The designer strives to find an optimal balance in the number of design variables. As shown in Figure 2.2 (and widely observed [5, 48, 55, 78]), finer parameterizations can reach superior designs but take longer to converge. Even with a quasi-Newton approach, the optimization requires $\mathcal{O}(N_{DV})$ design iterations to converge [40, 41]. In practice, a designer may need to manually refine the parameterization — a time-consuming task.

A progressive approach instead uses a sequence of shape control $\mathbf{C}^0, \mathbf{C}^1, \mathbf{C}^2 \ldots$ that permits ever more detailed design. The most important characteristic of this approach is that low-frequency deformations are handled first, and higher-frequency deformations are handled later, if necessary. From experience with multilevel methods, we would expect this approach to have several characteristics. First, it should accelerate the optimization and smooth the design trajectory. Each level is initialized much closer to the optimum than would be the case under a static parameterization, which would be expected to improve robustness by preventing high-frequency deformations until they are needed. Second, multilevel optimization constitutes a form of smoothing [79], which should alleviate the need for the type of direct gradient smoothing (or equivalently, modification of the definition of the inner product [30]) that is usually mandatory with high-resolution shape control.[4] Third, for under-constrained problems with infinite optima, such as inviscid airfoil design at low transonic Mach numbers, this approach regularizes the problem, making it well-posed during the early levels. This can be a convenient alternative or supplement to Tikhonov (L^2) regularization or "homotopy", whose weight must be carefully tuned or iteratively reduced to avoid overwhelming the actual objective function [46, 89, 90].

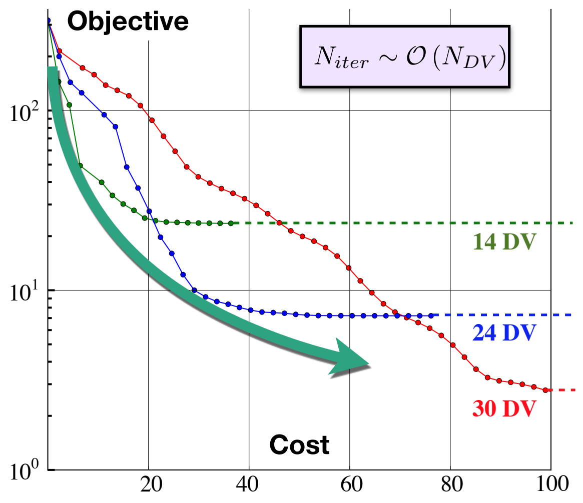


Figure 2.2: Static parameterizations compromise between efficiency and performance of the optimal design. A progressive parameterization approach dynamically controls N_{DV},allowing the process make rapid gains early, while approaching the continuous optimal shape. The goal is to follow the "inside track", notionally illustrated by the green arrow. Data corresponds to a geometric shape matching optimization.

In my approach, the designer still establishes one or more shape control bases that describe allowable classes of deformation, and provides a parameterization function $\mathcal{P}$ for each basis. However, construction of the actual shape control distribution $\mathbf{C}$ is automated. Starting from a coarse search space $\mathbf{C}^0$, the shape control is periodically refined, allowing optimization to continue in ever-higher-dimensional search spaces. In this work I do not consider removing parameters from the optimization. While it is certainly true that a parameter's potential may be exhausted in its initial context, it may become important

[4]Some degree of gradient smoothing may still be appropriate, especially as the resolution increases. However, the robustness of the optimization should be substantially less dependent on smoothing.

again as more degrees of freedom are added. Moreover, removing ineffective parameters is related to r-refinement, which can lead to stagnant convergence to the continuous optimum (cf. Footnote 3 in Chapter 1).

2.1.2 Design Spaces Admitting Progressive Parameterization

Let $\mathcal{J}_\star^k$ indicate the best attainable objective value in search space k, and let $\mathcal{J}_\star^\infty$ indicate the best attainable design in the continuous space. In general, we expect higher-dimensional search spaces to contain superior designs, i.e.

$$\mathcal{J}_\star^k \geq \mathcal{J}_\star^{k+1} \geq \ldots \geq \mathcal{J}_\star^\infty \tag{2.4}$$

This is self-evidently true for the type of nested search spaces I am considering, where $\mathbf{C}^k \subset \mathbf{C}^{k+1}$ [91],[5] and has also been observed empirically by many authors (e.g. [49, 55, 92]). Of course the mere fact that a search space *contains* a given optimal design does not imply that the optimizer can actually navigate to that design. Navigability depends on the type of optimizer and on the characteristics of the search space, including smoothness, convexity, and the presence of local extrema, but this is an orthogonal concern that can be addressed independently.

For progressive parameterization to be worthwhile, an appreciable portion of the the total design potential should be recoverable in lower-dimensional search spaces. This leads to a stronger statement than Equation (2.4):

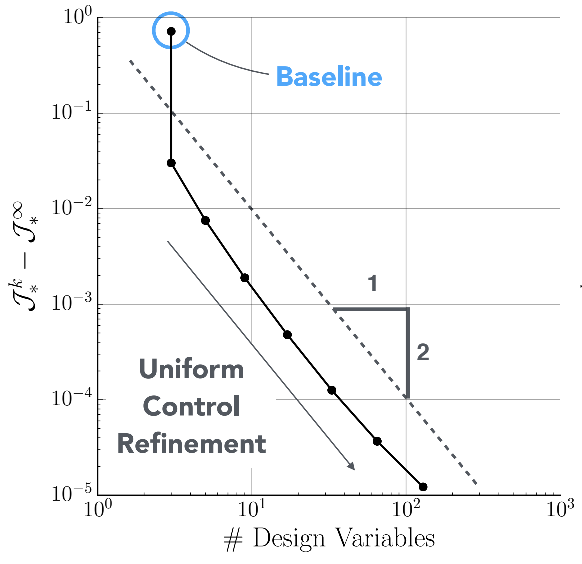
$$\mathcal{J}_\star^k - \mathcal{J}_\star^\infty \gg \mathcal{J}_\star^{k+1} - \mathcal{J}_\star^\infty \tag{2.5}$$

In other words, low-frequency shape modifications should have the largest impact on the design goals, and the influence should decay at high frequencies. For functionals typically encountered in aerodynamic shape optimization, this appears to be the case. The following sections will give one example where progressive parameterization is highly effective, followed by a counterexample that does not satisfy Equation (2.5).

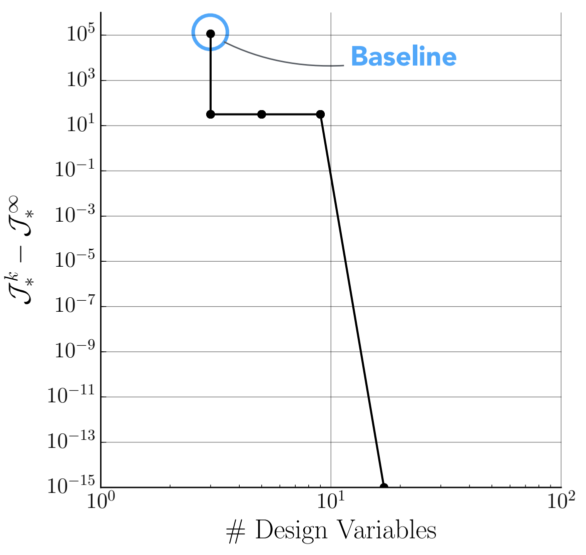
Example: Curve Matching

Consider approximating a continuous target curve $\mathsf{y}^*(x)$ on $x = [0, 1]$ with piecewise linear segments, forming a best-fit curve $\mathsf{y}(x)$. The objective is measured as the L_2-norm of the

[5]Without nesting, Equation (2.4) may not be true in the short term, due to the suboptimality of different shape control distributions. One example is given in [52], Figure 4, where moving from 8 to 9 evenly-spaced parameters actually resulted in an inferior final design. In that situation, $\mathbf{C}^k \not\subset \mathbf{C}^{k+1}$ and it so happened that the lower dimensional space contained a superior design.



(a) Function A: Curve matching



(b) Function B: 17-D Rosenbrock

Figure 2.3: Convergence to the continuous optimum with refinement of the control variables. Each data point represents a full optimization in a discretized search space.

point-wise distance between corresponding points on the curves:

$$\mathcal{J}_A = \left(\int_0^1 (\mathsf{y}(x) - \mathsf{y}^*(x))^2 dx \right)^{0.5}$$

In the continuous sense, the design variables are the y-values at every point on the curve. We discretize the search space by selecting a finite number of uniformly-spaced positions $\mathbf{x}_c$ at which to control the curve. The approximation $\mathsf{y}(x)$ is constructed by linear interpolation between the deformations $\mathbf{y}_c$ at these stations. In the notation of Chapter 2, $\mathsf{S} = \mathsf{y}(x)$, $\mathbf{C} = \mathbf{x}_c$, $\mathbf{X} = \mathbf{y}_c$, and $\mathcal{D}$ consists of linear interpolation.

Now consider multilevel optimization of $\mathcal{J}_A$, with $y^*(x) = x^2$. Starting with three design variables ($\mathbf{x}_c^0 = [0, 0.5, 1.0]$), the control resolution is gradually increased by uniform refinement. In the limit of control refinement, the continuous curve-matching problem is recovered. At each level, the curve is optimized to convergence, allowing evaluation of the maximum potential $\mathcal{J}_\star^k$ of that level (the continuous optimal value is $\mathcal{J}_\star^\infty = 0$). Figure 2.3a shows that $\mathcal{J}_\star^k$ converges at second-order rate with respect to the number of design variables. This is the expected result for approximating a smooth curve with evenly-spaced linear segments. In this design space (characterized by the shape-matching objective function and the manner of shape control refinement), Equation (2.5) holds — lower frequency shape modes capture the majority of the relevant information about the curve. All the aerodynamic shape optimization cases considered in the work behave in this manner.

Appropriate Test Functions for Progressive Parameterization

Not every optimization test function is appropriate for benchmarking progressive parameterization. Consider, for example, the N-dimensional extension of the Rosenbrock function:

$$\mathcal{J}_B = \sum_{i=1}^{N-1} \left((1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2 \right)$$

with a global optimum of $\mathcal{J}_B = 0$ at $x_i = (1, 1, \ldots)$.

Whereas the curve parameterization was spatially organized, the Rosenbrock design variables lack a coherent structure. Consider naively reducing the N-dimensional design space in the same way as before. We can arrange the coordinates x_i along a line, and drive their values using a coarser set of control coordinates, with linear interpolation between. This is technically a valid multilevel optimization approach, in the sense that in the limit of refinement, optimization will take place with the original N design variables. The levels are perfectly nested, so Equation (2.4) is satisfied. However, the strategy exhibits highly non-smooth convergence as shown in Figure 2.3b. The initial 3-DV parameterization is able to reduce the objective somewhat, by finding the N-D analog of the 2D Rosenbrock valley. However, no appreciable progress can be made with the next two parameterizations. Only upon recovering the full design space is any (and all) remaining potential recovered. Thus Equation (2.5) is not satisfied, and so this is clearly an inappropriate multilevel strategy.[6] Throughout this work I will therefore turn to similar shape-matching problems as more representative analytic test functions for progressive parameterization.

In this work I observe that a multilevel approach accelerates several common aerodynamic shape optimization problems. For more general problem classes, it is worth mentioning the approach of [93], which develops a general procedure for determining whether multiscale optimization is suitable for a particular combination of governing PDE, optimization problem type, and multigrid algorithm. They discuss several criteria which should be satisfied, notably the degree of nonlinearity and consistency across levels.

[6]This example does not imply that the Rosenbrock function does not admit *any* design space reduction techniques, merely that the obvious, chosen reduction is inconsistent, in that coarser levels are not representative of finer ones.

2.2 Optimization with Progressive Parameterization

The design loop is illustrated in Figure 2.4 and consists of a nested, alternating sequence: optimization within the current search space followed by refinement of the shape control. The optimization is divided into a sequence of sub-problems, each involving optimization in the fixed space spanned by $\mathbf{X}^k$. When transitioning to the next optimization level, the shape is held fixed, while the shape control is enriched.

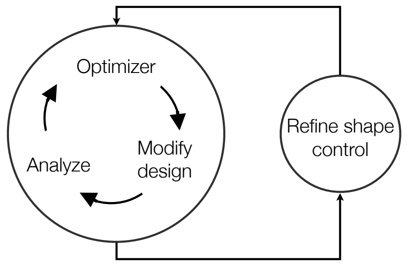


Figure 2.4: Optimization loop with periodic search space refinement

An alternative approach to refining at discrete intervals might more continuously introduce degrees of freedom, without halting the optimization. Although this might improve efficiency, it would require development of a specialized optimization routine. By refining only at discrete intervals, existing design optimization frameworks can be invoked to solve each sub-problem in a "black-box" manner. Additionally, discrete refinement moments provide convenient opportunities to safely programmatically modify the optimization problem, e.g. increasing the flow mesh resolution or farfield boundaries [94], relaxing design variable bounds, or resetting objective regularization terms [46].

This nested optimization loop is given explicitly in Algorithm A, which integrates three basic software components: (1) a geometry modeler, (2) a shape optimization framework, and (3) functions that guide search space refinement. The shape optimization framework and geometry modeler are treated as independent servers and are invoked during the outer loop over the sequence of search spaces.

2.2.1 Shape Control Basis

While the parameterization will be variable, the shape control *basis* is specified by the designer and is fixed throughout adaptation. The basis completely defines all aspects of the deformation that are fixed. Some examples of shape control bases are

- Wing twist about a given axis through a wing, linearly interpolated spanwise.
- Deformation of an airfoil by Hicks-Henne bump functions, e.g. $y = \sin^4(\pi x^{\frac{\log 0.5}{\log x_j}})$, where the locations x_j will be determined by adaptation.

The system automatically adapts the resolution and distribution of shape control within this framework.

Algorithm A: Optimization with Progressive Parameterization

```
Input: Shape parameterization function $\mathcal{P}$; initial surface $\mathsf{S}_0$,
       shape control $\mathbf{C}^0$, and DV bounds $\mathbf{b}^0$; objective and
       constraint functionals $\mathcal{J}, \mathcal{C}_j$
Result: Optimized surface $\mathsf{S}$

$\mathcal{D}^0, \mathbf{X}_0 \leftarrow \mathcal{P}(\mathsf{S}_0, \mathbf{C}^0)$ // Parameterize
$k \leftarrow 0$
repeat
    $\mathsf{S}, \psi_j, \overline{\mathbf{B}} \leftarrow$ OPTIMIZE$(\mathcal{J}, \mathcal{C}_j, \mathcal{D}^k, \mathbf{X}_0, \mathbf{b})$ until TRIGGER$(\cdot)$
    $\mathbf{C}_c \leftarrow$ GETCANDIDATES$(\mathbf{C}^k)$
    if adaptive then
        $\mathbf{C}^{k+1} \leftarrow$ ADAPTSHAPECONTROL$(\mathcal{P}, \mathbf{C}^k, \mathbf{C}_c, \mathsf{S}, \psi_j, \overline{\mathbf{B}})$
    else
        $\mathbf{C}^{k+1} \leftarrow \mathbf{C}^k + \mathbf{C}_c$ // Uniform refinement
    end
    $\mathcal{D}^{k+1}, \mathbf{X}_0 \leftarrow \mathcal{P}(\mathsf{S}, \mathbf{C}^{k+1})$ // Re-parameterize
    $\mathbf{b}^{k+1} \leftarrow$ PROLONG$(\mathbf{b}^k, \mathbf{C}^k, \mathbf{C}^{k+1})$
    $k \leftarrow k + 1$
until convergence of $\mathcal{J}^k_\star - \mathcal{J}^{k+1}_\star$ w.r.t. $\mathbf{C}$
```

Function color key:
BLUE: Refinement strategy (modeler independent)
BLACK: Standard adjoint-based design framework

As an example, Figure 2.5 illustrates the basic process for airfoil design. The designer selects a deformation technique (e.g. bump functions or splines, etc.), marks the leading and trailing edges and spar locations as "anchors" (black and orange dots in Figure 2.5), and places a single root shape controller in each region between anchors (blue dots). Thereafter, the shape control is automatically adaptively refined by the system.

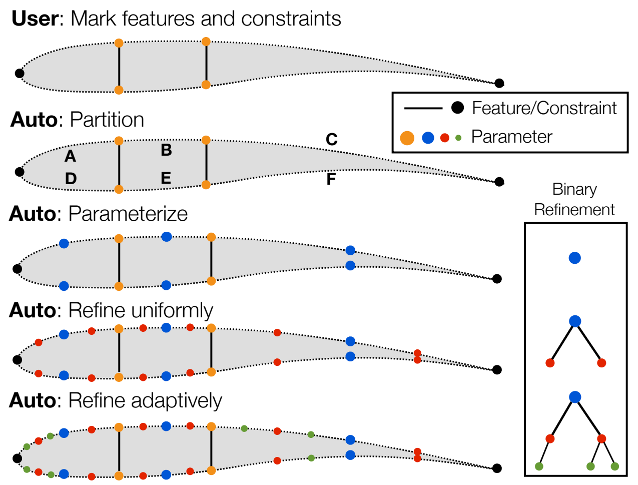

User: Mark features and constraints
Auto: Partition
A
B
C
D
E
F
Feature/Constraint
Parameter
Binary Refinement
Auto: Parameterize
Auto: Refine uniformly
Auto: Refine adaptively

Figure 2.5: Progressive parameterization with discrete, hierarchical shape control refinement

The actual vehicle for specifying the basis is a designer-provided function $\mathcal{P}$ (Equation (2.2)) that takes the current shape control $\mathbf{C}$ and prepares the modeler for deformation. This will typically involve generating setup files for deformation, pre-computing deformation modes, etc. Unlike in static optimization approaches, where this is a manual pre-processing step, here it must be automated. Further discussion of shape control bases and geometry modelers is deferred to Chapter 4.

2.2.2 Sub-optimization Solver

Each sub-level of the overall design problem is optimized under fixed shape control, a process represented by OPTIMIZE$(\cdot)$. This can be performed in a mostly black-box manner by any shape optimization framework. For reference, the typical suboptimization process is outlined in Appendix A, Function 2.

For the studies and demonstrations in this work, I use an existing design framework that uses an embedded-boundary Cartesian mesh method for inviscid flow solutions [95]. Aerodynamic objective and constraint gradients are computed using an adjoint formulation [96, 97]. These same adjoint solutions are later reused to prioritize candidate design variables when refining the search space [98]. For the numerical studies in this work, flow meshes are automatically adapted throughout optimization to reduce the discretization error in the output functionals [94, 99–101]. This adaptive meshing approach is discussed in more detail in Appendix C.

The framework supports design with arbitrary parametric geometry modelers, an important feature that I maintain also in the adaptive system. Geometric functionals and their analytic derivatives are computed by a standalone tool that operates directly on simulation-ready discrete surfaces. The design framework communicates with all geometry tools via XDDM, anXML-based protocol for design markup [95]. Each sub-optimization can be driven by any general, nonlinear, gradient-based optimizer; for this study, the SQP optimizer SNOPT [102] is used, enabling proper treatment of linear and nonlinear constraints. I use the full-memory BFGS variant, as the problems considered here involve fewer than 1000 design variables.

2.2.3 Refinement Procedure

The parameterization refinement is governed by several new procedures. Each will be developed in detail in later sections — this section briefly outlines the process flow. During each suboptimization level, design progress is monitored by TRIGGER$(\cdot)$, which determines when to terminate this level and refine the shape control. Next, GETCANDIDATES$(\cdot)$ generates a list of possible locations where the shape control may be refined. By analogy to mesh adaptation, Algorithm A uses an *h*-refinement approach to adding design variables.

Optimal continuous positioning of the shape controllers (r-refinement) is not considered. This decision was motivated by the desire to have a simple, yet sufficient system. As will be shown in Section 2.3, managing a discrete set of candidate shape control locations leads to a straightforward method for adaptively refining the parameterization.

Finally, some subset of these candidates are marked for refinement. The simplest approach is to add all of them; I will call this "uniform" refinement. Alternatively, the system can predict which subset of the candidates would best enrich the search space. This adaptive procedure is represented by ADAPTSHAPECONTROL$(\cdot)$. Note that exploring different shape control bases is outside the scope of this function. While this could be considered in future work, it is a much more algorithmically challenging problem.

Once again, $\mathcal{P}$ invokes the geometry modeler to refine the parameterization, which expands the search space available to the optimizer. Upon refinement, PROLONG$(\cdot)$ transfers the design variable bounds from the previous search space into the new one. It could also potentially transfer approximate Hessian information to initialize the optimization in the new search space. I do not attempt that in this work, but instead cold-start each suboptimization level with $\overline{\mathbf{B}}_0 = \overline{\mathbf{I}}$. This is done in order to isolate the acceleration due solely to reduction in N_{DV} from the relatively unknown factor of Hessian preconditioning. This approach is conservative; the substantial design acceleration demonstrated in this work can be further accelerated by preserving Hessian information from previous design spaces.

2.2.4 Convergence

Convergence of Algorithm A happens as the discrete shape control $\mathbf{C}$ approaches continuous shape control[7]. The process would be terminated when the objective (or merit function) has converged sufficiently to the continuous local optimal value $\mathcal{J}_\star^\infty$ with respect to shape control refinement. We do not usually know $\mathcal{J}_\star^\infty$, except for attainable inverse design problems. However, after sufficient refinement, if Equation (2.5) holds, then $\mathcal{J}_\star^k - \mathcal{J}_\star^{k+1}$ might be expected to enter a region of asymptotic convergence to zero (see Figure 2.3a). Its magnitude could thus be used as a rough convergence criterion [103].

More rigorously, in Chapter 3 I develop a refinement indicator that directly estimates the potential for design improvement in a hypothetical refined search space. When this potential falls below a certain threshold, it could be determined automatically that further refinement and optimization is not worthwhile. An alternative approach might directly estimate proximity to the optimal solution, by analogy to error estimation in adaptive PDE

[7]If the designer-specified shape control bases cover all possible deformation "directions", then $\mathbf{C}^\infty \equiv \mathsf{S}$, direct optimization of every point on the surface.

solvers. One approach might be to use second-order adjoints, similar to the approach in [104] for optimization with progressive meshing. In practical settings, however, the optimization is most likely be terminated based on computational resource limits. The main goal of Algorithm A is to improve the design as rapidly as possible, and will not usually be allowed to converge deeply, except in verification cases.

If for some reason Algorithm A encounters a design space that does not admit multilevel acceleration, it will simply continue to refine the shape control until the original design space is recovered (as happened in Figure 2.3b). In this hypothetical case only efficiency would be compromised, not robustness. Anecdotally, none of the examples and studies in Chapters 5 and 6 encounter this situation, which strongly suggests that progressive parameterization is appropriate for most aerodynamic shape design problems that involve well-structured shape control.

2.3 Discrete Shape Control Refinement

In this work, I adopt a discrete refinement approach, similar to [50]. The conjecture is that by localizing refinement of the shape control, many problems can be solved with far fewer degrees of freedom than required with uniform shape control. As shown in Figure 2.5, the shape control is partitioned into independently-controlled regions (labeled **A**-**F**) between important design features, which are to be preserved or directly manipulated.

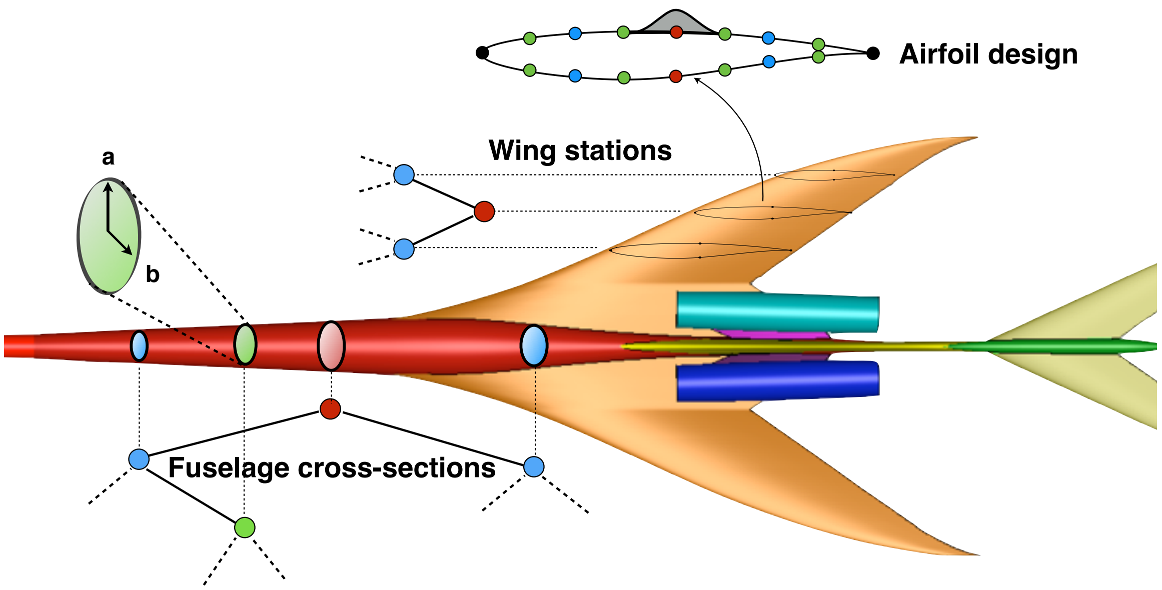


Figure 2.6: Progressive parameterization of an aircraft configuration

Under a discrete approach, it is natural to organize the shape control into binary trees. Each region has a "root" controller, and higher resolution control is introduced through binary refinement. In the limit of refinement, this sequence converges to continuous shape control that completely "fills" the surface being designed. This scheme easily generalizes to encompass parameterization of a full aircraft configuration, as illustrated in Figure 2.6. Each component has its own set of shape control trees, allowing its parameterization to be refined independently. Each control station may enact several modes of deformation, which can each be represented by independent subtrees. In wing design, for example, twist, thickness, dihedral, and airfoil control are each represented by a separate tree.

As shown in Figure 2.7, each shape controller can be uniquely identified by a string of L's and R's, indicating a path down the tree. These identifiers provide information about the structure of the parameterization that is used to determine refinement candidates and to transfer information between search spaces. Interpreting these identifiers is the role of the parameterization function $\mathcal{P}$, which generates the specific shape deformation modes from this generic description.

The binary tree structure can equivalently be represented as a Cartesian grid, as depicted at the bottom of Figure 2.7. For certain purposes this can be a more useful representation, enabling straightforward analogies to meshing and repurposing of familiar algorithms for searching, traversal and refinement [105]. For a parameterization with only one principal direction (e.g. twist along a wing or airfoil deformation), the tree- and grid-based views are functionally equivalent. For a 2D parameterization, however, one or the other may be more appropriate to define the refinement rules.

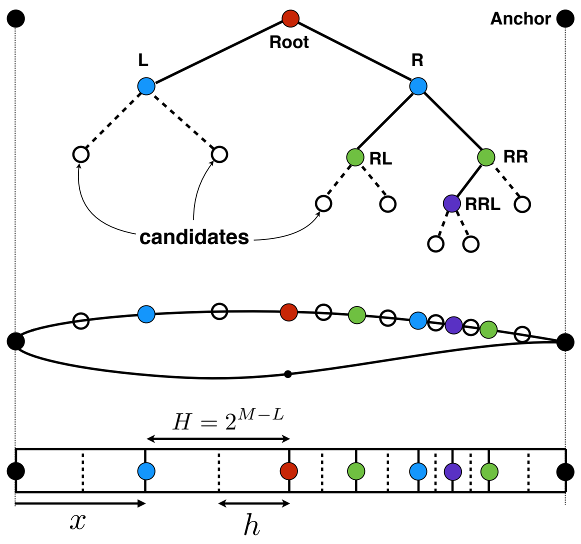


Figure 2.7: Binary tree and grid representations of an airfoil shape parameterization, showing candidate refinement locations computed by Function 3 in Appendix A

2.3.1 Shape Control Refinement

Consecutive shape control levels are constructed by binary subdivision of the leaves in the current tree. A specific procedure is given in Appendix A, Function 3 and is illustrated in

Figure 2.7. Under uniform refinement, this process leads to complete "coverage" of the curve or surface and in the limit of refinement it leads to continuous control of the shape (within the given shape control basis). New shape controllers may be placed only at the midpoints between existing shape controllers, but the refinement could be directionally biased, e.g. to cluster parameters towards the leading edge of a wing. In other words, the tree-representation of the shape control logically partitions the surface, but the geometry modeler may map this to physical space via any function.

By recursion, Function 3 can refine more than one level at a time. The number of new parameters generated is roughly $2^{dD-1}N_{DV}$, where N_{DV} is the current number of design variables, d is the search depth, and D is the dimension of the parameterization (e.g. $D = 1$ for curves, $D = 2$ for 2D surface parameterization). Many parameterization techniques support infinitely-refinable shape control resolution, but it is usually appropriate to specify a maximum refinement depth d_{max}, which maintains a minimum spacing between adjacent parameters and imposes smoothness constraints on the shape control.

2.3.2 Prolongation of Design Variable Bounds

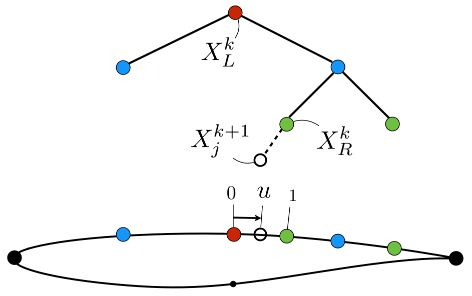


Figure 2.8: Prolongation

Each design variable in search space k may have upper and lower bounds. When transitioning to search space $k+1$, bounds must be set for the newly added variables. Because the binary tree shape control representation enforces a high degree of spatial organization, we can define a "prolongation" operator that transfers the bounds to the new search space. This process is denoted by the function PROLONG$(\cdot)$ in Algorithm A. A complementary "restriction" operator is not required, as we are using a progressive, sequencing approach.

Let the upper and lower design variable bounds for shape control level k at parameter i be designated $b_i^k \equiv (l_i^k, u_i^k)$. Say that on transitioning from search space k to $k+1$, a new parameter j is being added between parameters L and R, its nearest neighboring existing parameters in the same tree, as illustrated in Figure 2.8. To determine bounds for this new design variable, I linearly interpolate the bounds between the existing controllers:

$$b_i^{k+1} = (1-u)b_L^k + ub_R^k \tag{2.6}$$

where u is a mapping of the fraction of the distance between L and R at which j is located. If the shape control is refined at midpoints instead of being skewed or biased in one direction, then $u = 0.5$.

This is a crude method, but consistent with the typical nature of design variable bounds as crude restrictions of the search space. Linear interpolation correctly handles certain common uses of design variable bounds. For example, in a thickness/camber parameterization, positive thickness can be enforced via non-negative design variable bounds on the thickness parameters. In general, however, Equation (2.6) is not guaranteed to avoid invalid designs, even if the original design variable bounds did. An example of this is given in Section 6.1. This capability is included largely as a stopgap measure to handle common use cases. It is generally preferable to codify the actual design requirements using geometric constraints defined with respect to the surface itself.

There are many other types of data that may be associated with each design variables, such as scale factors, finite difference step sizes, geometry modeler-specific parameters, etc. An approach similar to Equation (2.6) can be used to transfer each of these to refined parameterizations. Chapter 3 will develop a special Hessian prolongation operator to estimate curvature in hypothetical search spaces.

2.4 Refinement Pacing

In typical static parameterization approaches, convergence is measured by monitoring an optimality criterion based on the Karush-Kuhn-Tucker (KKT) conditions. With progressive parameterization, however, we are more concerned about convergence with respect to the shape control — the outer loop of Algorithm A. Within each suboptimization level, the design is improved by exploiting the current search space. After a certain point, the existing design variables offer negligible potential for improvement, and the shape control must be refined.

The precise timing of this transition has a major impact on efficiency, as shown in Figure 2.9. A foolproof approach is to fully converge each suboptimization, to achieve maximal design improvement within each search space (as in [50]). However, this can be extremely slow. Over-optimizing on the initial parameterization leads to long periods of negligible design improvement, while triggering earlier can lead to much faster design improvement. Similar observations have been made, both in this context [77] and regarding optimization with progressively refined flow meshes [101].

In certain special cases, the objective may be to determine an optimal set of physical degrees of freedom to build into a system — an example of this is given in Section 6.4, which considers layout of a flap system. In such cases, each level should be allowed to fully exploit the existing parameters before adding new ones. However, in most cases, the desire is to improve the the shape as rapidly as possible, and coarse parameterizations are used merely to accelerate the rate of improvement. In these cases it is appropriate to only partially converge each suboptimization.

2.4.1 Automatic Trigger

The designer can always manually invoke a refinement of the parameterization. However, to avoid designer-in-the-loop actions, I use an automatic signal to initiate a refinement. This is represented by TRIGGER$(\cdot)$ in Algorithm A. I ruled out certain simplistic signals, based on a pre-determined number of major iterations (as in [77]). This would demand prior knowledge of the rate of convergence for a problem, which is not conducive to an automated system.

A less problem-dependent signal is based on sufficient reduction in the order of magnitude of the KKT optimality criterion O, relative to the baseline:

$$\frac{\log O_i}{\log O_0} < r \tag{2.7}$$

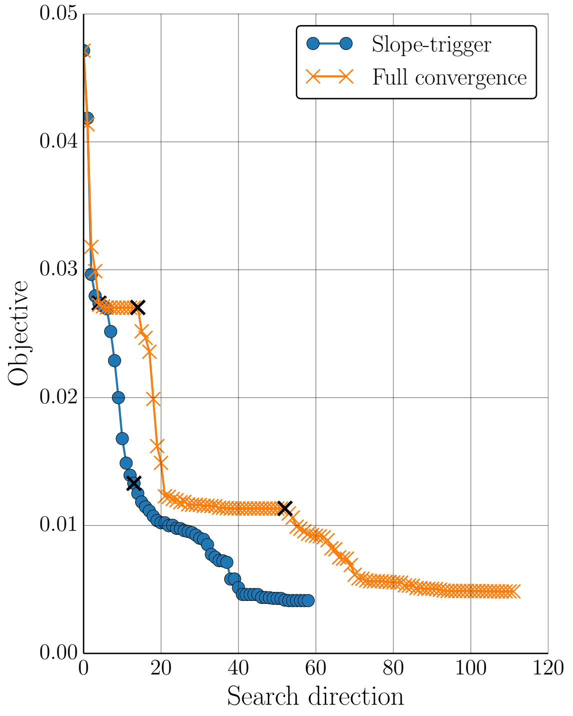


Figure 2.9: *Orange*: Optimizing to convergence on each level leads to slow design improvement. *Blue*: Using aggressive slope-based transitions permits much faster design improvement. $\times$-marks denote parameterization refinements.

Unfortunately, gradient reduction is only tenuously related to the actual rate of design improvement. I found establishing an efficient cutoff r to be challenging and unintuitive. A more intuitive approach is based on monitoring the slope of the objective convergence history for a diminishing rate of design improvement with respect to a suitable measure of computational cost.[8] The optimization is terminated when this slope falls below some fraction r of the maximum slope that has occurred so far:

$$\frac{\Delta \mathcal{J}_i}{\max_k(\Delta \mathcal{J}_k)} < r \tag{2.8}$$

[8]The slope is evaluated at major search iterations, which is monotonically decreasing. In the case of constrained optimization, I monitor convergence of SNOPT's "merit function" [102]. For attainable inverse design problems, the slope is measured in log-space to better reflect the problem.

where $\Delta\mathcal{J}_i = \mathcal{J}_{i-1} - \mathcal{J}_i$. This is a pertinent choice from a practical perspective, where the desire is usually to maximally improve a design within a resource budget. In this case, setting the cutoff parameter r is more intuitive, as it constitutes a simple cost-benefit analysis. For simple problems a fairly aggressive trigger can be used (I have used as high as $r = 0.2$). For more challenging problems, especially ones with initially violated constraints, it can be more effective to allow deeper convergence on the coarser parameterizations before proceeding.

The denominators in Equations (2.7) and (2.8) normalize the triggers by accounting for the widely differing scales (and units) that can occur in different objective functions and their gradients. For example, drag is often $\mathcal{O}\left(10^{-2}\right)$ while mission range may be $\mathcal{O}\left(10^{5}\right)$. In Equation (2.8), the normalization is by the maximum slope observed so far — this accounts for the fact that quasi-Newton optimizers often require several iterations to develop the Hessian approximation before they begin to make substantial progress. If the normalization were instead by the initial slope, the trigger could be excessively delayed. The objective and gradient histories can be non-smooth, which can cause false triggering. This can be ameliorated by using running averages over a small window w, which smooths the history and prevents premature triggering. Because this delays the trigger for w major iterations, w should be as small as possible.

The main limitation of both Equations (2.7) and (2.8) is that they are myopic. They tacitly assume that diminishing design improvement or gradient reduction indicates that the search space is nearly fully exploited. This assumption is not always valid: the optimizer could be simply navigating a highly nonlinear or poorly-scaled region of the design space, after which faster design improvement would continue. Thus these triggers may introduce shape parameters earlier than strictly necessary. However, under an adjoint formulation, the cost of computing additional objective and constraint gradients is usually negligible compared to the cost of over-converging in a coarse search space.

CHAPTER 3

Goal-Oriented Adaptation of the Shape Control

Although uniform shape control refinement is guaranteed to approach the continuous optimization problem, indiscriminate introduction of design variables can quickly lead to unwieldy search spaces that are slow to navigate. Moreover, evenly-spaced shape control does not provide feedback regarding important regions of the surface. This chapter investigates selective, localized adaptation of the shape control. To place my approach in context, I first examine previous approaches to shape control adaptation and discuss their limitations. I then introduce a more accurate, low-cost, goal-oriented refinement criterion that favors design variables that offer the greatest potential for design improvement. I then develop an adaptation strategy, which is paramount for overall efficiency.

3.1 Previous Approaches

Data-driven Methods

One possible approach to shape control adaptation involves preliminary sampling and analysis of the design space to identify the most important degrees of freedom. [2, 61–63]. However, building a shape control adaptation process on this paradigm could be cost-effective only in quite low-dimensional spaces. This is partly due to the nature of the fitting [44], but also due to their “top-down” approach — sampling takes place in a *finer* search space than the final parameterization.

Another notable approach is that of Olhofer *et al.* [86], who use a genetic optimization approach to simultaneously optimize the location of the shape parameters **C** along with their values **X**. Under this approach, optimization is simultaneously taking place under several different populations, each with a different parameterization. The main concern with this approach is that it does not predict the best shape control a priori, but rather seeks to fortuitously chance upon it. For optimization driven by high-fidelity flow simulations, this is extremely expensive.[1]

These approaches would be more effective for determining a compact, well-spanning search space in which to perform a large number of optimizations. In that case, the high cost of a pilot data-driven optimization study could be amortized over all the future optimizations. The present work, however, aims for efficient one-time optimizations in (ultimately) very high-dimensional search spaces. In this context, belatedly discovering an effective parameterization is not useful.

Geometric Adaptation Criteria

In another class of adaptive approaches, the shape control is periodically redistributed during optimization to improve its geometrical regularity, which is observed to moderately accelerate the optimization [47, 75, 77, 79, 81]. Special definitions of "regularity" must be defined for each parameterization technique (e.g. Bézier curves [75], free-form deformation [47], and B-splines [81]). It is not clear how to derive a universal, modeler-neutral approach.

From analogous work in adaptive flow meshing, it is clear that such geometric or feature-based refinement criteria tend to be effective only on a restricted set of problems. For example, in [47, 75] an adaptation cost function is defined, which measures the total variation of the deformed control point locations. They take this cost function as a surrogate for the "ineffectiveness of the current parameterization" [77]. Despite its label, this cost function measures ineffectiveness due only to poor conditioning, as measured via spectral analysis [79]. It does not directly measure the parameters' actual potential to improve the objective function.

[1]However, when combined with more predictive methods, parallel evolution of parameterizations is not without value, in the same way that a multistart strategy is a useful in the context of gradient-based design.

Goal-Oriented Adaptation Criteria

Inspired by similar developments in adaptive meshing of PDES, a *goal-oriented* approach might be expected to be more universally appropriate. In this approach, the parameterization is adapted precisely in order to more efficiently solve the optimization problem at hand. A natural way to directly target the optimization problem is to leverage information on the sensitivity of the objective and constraints to the shape control. To my knowledge, the first such goal-oriented shape control refinement criterion in the literature is that of Han and Zingg [106] (and later briefly examined in [83]). After disqualifying some candidate parameters based on heuristics related to constraints and parameter spacing regularity, they add the one parameter with the largest objective gradient, leading to the following refinement indicator:

$$I(C_i) = \left| \frac{\partial \mathcal{J}}{\partial X_i} \right| \tag{3.1}$$

A similar approach was used by Poole *et al.* for reconstructing airfoil databases with compact parameterizations [58].

Equation (3.1) assumes that large objective gradients are directly correlated with future design improvement. Several factors can erode the validity of this assumption:

1. The presence of active constraints
2. Variation in design space curvature
3. Redundancy among parameters

The following sections briefly discuss these issues.

Constraints

Aerospace designers often account for the requirements of unmodeled disciplines by including surrogate design constraints. For example, wing thickness constraints are often used to enforce structural requirements. Because these surrogate constraints represent important design-driving disciplines, the constraints are typically active. A candidate shape parameter is not useful if it must violate a constraint to improve the objective. It is therefore important to prioritize shape parameters that enable *feasible* objective reduction.

For localized constraints (e.g. wing thickness), one could simply exclude any candidate shape control stations located near the active constraints, as done in [50]. However, this does not work for important non-local constraints, such as lift, pitching moment or wing volume. Alternatively, any constraint can always be incorporated into the objective via a penalty term:

$$\hat{\mathcal{J}} := \mathcal{J} + \sum_j w_j(\mathcal{C}_j - c_j^*)^2$$

if one accepts the attendant drawbacks: user-defined weights, inexact constraint satisfaction, and possible ill-conditioning.[2] The indicator I develop will instead handle general linear and nonlinear constraints, including design variable bounds, using an approach based on the Karush-Kuhn-Tucker (KKT) optimality conditions [107].

Design Space Curvature and Redundancy

The second derivative of the objective reveals and quantifies two important factors for ranking candidates. First, large curvature indicates that a candidate design variable is not as effective as its gradient would otherwise suggest. Secondly, while a pair of candidates may individually offer high potential, the potential may be mutually exclusive — adding both might not be helpful. The opposite is true as well; one shape controller can become more effective in the presence of another. These factors can both be seen in the Hessian matrix $\overline{\mathbf{H}} := \frac{\partial^2 \mathcal{J}}{\partial \mathbf{X}^2}$, by decomposing it into diagonal and off-diagonal terms:

$$\overline{\mathbf{H}} = a(\overline{\mathbf{D}} + \overline{\mathbf{O}})$$

The first factor is encoded by variation along the diagonal $\overline{\mathbf{D}}$, while redundancy is revealed by the presence of large off-diagonal entries $\overline{\mathbf{O}}$. Note that variation in the constant factor a does not change the *relative* ranking of candidates. It is not design space curvature *per se* that matters, but rather variation in curvature.

In a typical *static* parameterization approach, Hessian information is not typically available a priori, short of prohibitively costly finite-differencing. However, at a refinement stage during progressive parameterization, a substantial degree of optimization has already taken place in coarser search spaces. With spatially structured shape control, it is possible to use Hessian information from previous coarser spaces to help guide adaptation, with negligible overhead.

[2]Also note that this cannot simply be used as a temporary surrogate functional for adaptation. Assuming the constraints are satisfied, their gradients are zero, and so their influence could be accounted for only through their Hessians. This penalty formulation would have to be used for the shape optimization itself, so that the quasi-Newton Hessian approximation would reflect the constraints.

Non-Separability

Another subtle issue with Equation (3.1) is that it examines only the gradient of the single candidate parameter being considered. However, under most parameterization techniques, a parameter's deformation mode depends also on where its neighboring shape parameters are located.

To see this, consider Figure 3.1, which depicts shape modes that interpolate prescribed deformation between control points. The addition of a new parameter (green dot) shrinks the deformation mode of its neighbor (red dot). This is true for any technique involving interpolation between controllers (e.g. spanwise lofting or radial basis functions) and is also true for any technique with "compact support", such as B-splines and free-form deformation. Similarly, with global modes such as Bernstein polynomials or class/shape transformation methods [108], raising the degree of the polynomial basis changes *all* the shape functions.

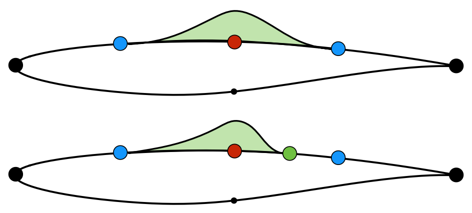


Figure 3.1: Under an interpolation-based deformation model, the red parameter's deformation mode shape (and thus the potential it offers) depends on the location of its neighbors.

The implication is that, in general, a given shape control candidate C_i does not necessarily define a unique corresponding deformation mode X_i.[3] Rather, the parameterization function $\mathcal{P}(\mathbf{C})$ is a nonlinear function of the *ensemble* of shape control. A refinement indicator that is expected to work with arbitrary parameterization techniques must likewise treat the shape control as an ensemble, unlike Equation (3.1).

3.2 Refinement Indicator — Maximum Potential

To explicitly address the previous issues, I now develop a more general class of refinement indicators. Simply put, the goal of this indicator is to prioritize design variables that maximize the *potential* for design improvement in the next search space. This potential is defined as the objective reduction that would be possible in the candidate search space defined by $\mathbf{C}$, without violating any constraints:

$$\Delta\mathcal{J}_\star(\mathbf{C}) = \mathcal{J}_0 - \mathcal{J}(\mathbf{X}_\star) \tag{3.2}$$

[3]For certain specific techniques, including algebraic bump functions [7] this is not an issue. The point is that we cannot rely on this for a system that is expected to work with arbitrary parameterization techniques.

where $\mathcal{J}_0$ is the current objective value, i.e. the optimum achieved in the previous search space. The local refinement indicator for a candidate parameter C^c is the additional potential that it would offer over the existing shape control:

$$I(C^c) := \Delta\mathcal{J}_\star(\mathbf{C} + C^c) - \Delta\mathcal{J}_\star(\mathbf{C}) \tag{3.3}$$

To compute this indicator, we must estimate the optimal objective value $\mathcal{J}(\mathbf{X}_\star)$ in a *hypothetical* search space.

Figure 3.2 gives a rough illustration of the approach to estimating $\mathcal{J}_\star(\mathbf{C})$, the potential of the nonlinear candidate search space spanned by $\mathbf{X}$. I take a local quadratic fit of the objective and local linear fits of the active constraints about the current design $\mathbf{X}_0$. The quadratic fit is computed from the objective's current value $\mathcal{J}_0$, gradients $\frac{\partial \mathcal{J}}{\partial \mathbf{X}}$, and Hessian matrix $\overline{\mathbf{H}}$. I take the analytically known minimal objective value of this local fit as an estimate of $\mathcal{J}_\star$.[4] For reasons that will become apparent later, I begin the derivation from the viewpoint of continuous shape optimization and then show how it projects into a finite search space.

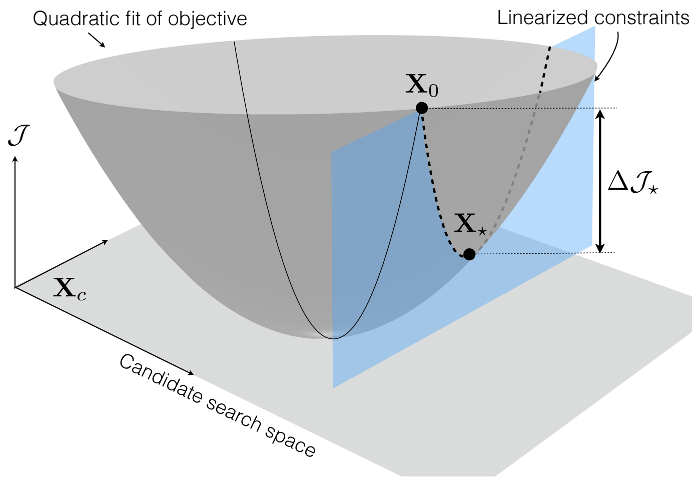


Figure 3.2: Estimating feasible design improvement in a candidate search space using a local quadratic fit of the objective and local linearization of the active constraints.

3.2.1 Potential of the Continuous Design Space

Denote the inner product of two continuous surface vectors as

$$\langle \mathsf{a}, \mathsf{b} \rangle := \int_{\mathsf{S}} \mathsf{a}(\mathbf{x})\mathsf{b}(\mathbf{x}) d\mathsf{A}(\mathbf{x}) \tag{3.4}$$

[4] One might propose stepping straight to the predicted optimum, but there is no way to do so robustly. Even with an exact Hessian, if higher-order derivatives are large, then the proposed step could actually be arbitrarily worse than the current design. Here I use local, approximate derivatives to predict the best shape *control* but leave navigation to a robust optimizer.

where $\mathbf{x}$ is some surface coordinate on S. If S is a surface, then $d\mathsf{A}(\mathbf{x})$ represents a differential area. If S is a curve, then $d\mathsf{A}(\mathbf{x})$ is a differential segment of the curve. The design functionals $\mathcal{F}_j$ must not depend on the shape control. The following derivations preclude simultaneous optimization of the design variables $\mathbf{X}$ and the shape control $\mathbf{C}$, as in [86]. This is satisfied under the approach developed in Chapter 2, which explicitly decouples $\mathbf{X}$ and $\mathbf{C}$. To avoid cluttering the derivation, I assume that $\mathcal{F}_j$ are functions of S only, taking any operating conditions $\mathbf{\Theta}$ constant. The extension to include operating conditions is straightforward, as $\mathbf{\Theta}$ is usually a finite, fixed set of variables.

Denote the potential for feasible objective reduction in the continuous design space as

$$\Delta\mathcal{J}_\star^\infty := \mathcal{J}_0 - \mathcal{J}(\mathsf{S}_\star) \tag{3.5}$$

where $\mathsf{S}_\star$ is the unknown optimal shape. To proceed, I will derive a second-order estimate of this potential. A local quadratic fit by Taylor expansion about the current design gives

$$\mathcal{J}(\mathsf{S}_0 + \delta\mathsf{S}) = \mathcal{J}(\mathsf{S}_0) + \left\langle \frac{\partial\mathcal{J}}{\partial\mathsf{S}}, \delta\mathsf{S} \right\rangle + \frac{1}{2}\langle \delta\mathsf{S}, \mathcal{H}\delta\mathsf{S}\rangle + h.o.t. \tag{3.6}$$

where $\mathcal{H}$ is the continuous symmetric Hessian operator of the objective:

$$\mathcal{H} : \delta\mathsf{S} \to \delta\frac{\partial\mathcal{J}}{\partial\mathsf{S}} \tag{3.7}$$

In other words, $\mathcal{H}$ returns a second-order sensitivity indicating how the surface objective gradient would change in response to unit deformation in the direction $\delta\mathsf{S}$. Combining Equation (3.5) and Equation (3.6), and neglecting higher-order terms we have

$$\Delta\mathcal{J}_\star^\infty \approx -\left\langle \frac{\partial\mathcal{J}}{\partial\mathsf{S}}, \delta\mathsf{S}_\star \right\rangle - \frac{1}{2}\langle \delta\mathsf{S}_\star, \mathcal{H}\delta\mathsf{S}_\star\rangle \tag{3.8}$$

where $\delta\mathsf{S}_\star := \mathsf{S}_\star - \mathsf{S}$, the step to the optimal design. I estimate $\delta\mathsf{S}_\star$ as the step to the minimizer of the local, constrained quadratic fit; this is the classic Newton iteration applied in a continuous setting.

The currently active constraints $\boldsymbol{\mathcal{C}}^a \subseteq \boldsymbol{\mathcal{C}}$ are linearized about the current design:

$$\boldsymbol{\mathcal{C}}^a(\mathsf{S}_0 + \delta\mathsf{S}) = \mathbf{c}_0 + \left\langle \frac{\partial\boldsymbol{\mathcal{C}}^a}{\partial\mathsf{S}}, \delta\mathsf{S} \right\rangle$$

where $\mathbf{c}_0$ are the current constraint values. I make two simplifying assumptions:

1. All active constraints are currently satisfied ($\mathbf{c}_0 = \mathbf{c}^*$).

2. Currently *inactive* constraints remain inactive at the optimum, so that $\boldsymbol{\mathcal{C}}^{a,k+1} \subseteq \boldsymbol{\mathcal{C}}^{a,k}$.

Although this derivation can be extended to not require Assumption (1), it is usually safe to assume, assuming the problem has been posed correctly. As long as the constraints are initially satisfied, or at least driven to satisfaction within the first sub-optimization level, the optimizer will be able to keep them satisfied as more degrees of freedom are added. If for some reason a constraint is not satisfied, or perhaps fundamentally unable to be satisfied, then the refinement indicator will consider it active and will seek to make it no *less* satisfied than it currently is.

Assumption (2) could be more problematic in certain cases. If the full Newton step subject to the active constraints would violate a currently inactive constraint, then the potential of the design space would be lower than expected. Predicting the correct active set of constraints at the optimum is a challenging problem even for sophisticated optimizers, so I leave this more general case for future work. The goal here is to make a prediction about the optimum without actually expending the optimization effort to navigate there.

The constrained minimizer of the quadratic fit is found by solving the following (continuous) system of equations, called the KKT system[5]:

$$\begin{bmatrix} \mathcal{H} & \frac{\partial \boldsymbol{\mathcal{C}}^a}{\partial \mathsf{S}} \\ \left(\frac{\partial \boldsymbol{\mathcal{C}}^a}{\partial \mathsf{S}}\right)^\mathsf{T} & \bar{\mathbf{0}} \end{bmatrix} \begin{pmatrix} \delta \mathsf{S}_\star \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\frac{\partial \mathcal{J}}{\partial \mathsf{S}} \\ \mathbf{0} \end{pmatrix} \tag{3.9}$$

where $\boldsymbol{\lambda}$ are the KKT multipliers.[6] The zero on the right hand side of Equation (3.9) is the result of Assumption (1) above. The top half of the KKT system gives the Newton step to the optimum of the constrained quadratic problem:

$$\delta \mathsf{S}_\star = -\mathcal{H}^{-1} \left(\frac{\partial \mathcal{J}}{\partial \mathsf{S}} + \boldsymbol{\lambda} \frac{\partial \boldsymbol{\mathcal{C}}^a}{\partial \mathsf{S}} \right) \tag{3.10}$$

To be clear, $\delta \mathsf{S}_\star$ is a non-infinitesimal deformation of the continuous surface. To solve for the KKT multipliers $\boldsymbol{\lambda}$, we substitute Equation (3.10) into the bottom half of Equation (3.9). This yields the following system of equations (one per active constraint):

$$\left\langle \frac{\partial \mathcal{C}^a_j}{\partial \mathsf{S}}, \mathcal{H}^{-1} \left(\frac{\partial \mathcal{J}}{\partial \mathsf{S}} + \lambda_i \frac{\partial \mathcal{C}^a_i}{\partial \mathsf{S}} \right) \right\rangle = 0$$

[5]For a general derivation of this system see, e.g. [41].

[6]For constraints of the form $\mathcal{C}_j(\mathsf{S}) = c_j$, $\mathcal{C}_j(\mathsf{S}) \leq c_j$, and $\mathcal{C}_j(\mathsf{S}) \geq c_j$, the KKT multipliers must satisfy $\lambda_j \neq 0$, $\lambda_j \geq 0$, $\lambda_j \leq 0$, respectively. In the absence of inequality constraints, these are called Lagrange multipliers.

This system is satisfied when

$$\lambda_i \frac{\partial \mathcal{C}_i^a}{\partial \mathsf{S}} = -\frac{\partial \mathcal{J}}{\partial \mathsf{S}} \tag{3.11}$$

Equation (3.11) is (infinitely) over-determined and is only exactly true at the optimum, so we can only approximate $\boldsymbol{\lambda}$.

Substituting Equation (3.10) into Equation (3.8) yields

$$\Delta \mathcal{J}_\star^\infty = \frac{1}{2} \left\langle \left(\frac{\partial \mathcal{J}}{\partial \mathsf{S}} - \boldsymbol{\lambda} \frac{\partial \boldsymbol{\mathcal{C}}^a}{\partial \mathsf{S}} \right), \mathcal{H}^{-1} \underbrace{\left(\frac{\partial \mathcal{J}}{\partial \mathsf{S}} + \boldsymbol{\lambda} \frac{\partial \boldsymbol{\mathcal{C}}^a}{\partial \mathsf{S}} \right)}_{\mathbf{A}} \right\rangle \tag{3.12}$$

This is an estimate of the potential for feasible objective reduction in the continuous space. Term $\mathbf{A}$ in Equation (3.12) is related to the KKT stationarity condition. Optimization drives $\mathbf{A} \to 0$, and thus $\Delta \mathcal{J}_\star^\infty \to 0$, which is an indication that the design is approaching a continuous local optimum. If there are no constraints,

$$\Delta \mathcal{J}_\star^{\infty, \not{\mathcal{C}}} = \frac{1}{2} \left\langle \frac{\partial \mathcal{J}}{\partial \mathsf{S}}, \mathcal{H}^{-1} \frac{\partial \mathcal{J}}{\partial \mathsf{S}} \right\rangle$$

Because this is in quadratic form, if $\mathcal{H}$ is positive definite (indicating local convexity), then $\Delta \mathcal{J}_\star^\infty \geq 0$, which makes sense — the potential for improvement certainly cannot be negative. Except for within a vicinity of the optimum, the true Hessian is not necessarily positive definite. This will be resolved by using a quasi-Newton approximation of the Hessian, which is positive definite by construction.

Equation (3.12) indicates *whether* further optimization is warranted, but not does not yet indicate how best to focus the shape control. Moreover, Equation (3.12) is not computable, both because it is expressed in the continuous space and also because we do not generally have the exact Hessian for aerodynamic problems. The next step is to project this into a finite search space, which will yield a computable estimate of remaining potential and also provide a refinement indicator.

3.2.2 Potential of a Finite Search Space

Now we return to the original aim: computing the expected design improvement in a *finite* search space defined by the shape control $\mathbf{C}$. The goal is to compute this using only the gradient vectors $\frac{\partial \mathcal{F}_j}{\partial \mathbf{X}}$ and a quasi-Newton Hessian approximation from the previous search space $\overline{\mathbf{B}}^k$. This will require a discretization of the continuous shape control.

Discretization

The discrete approximation of Equation (3.4) is the area-weighted dot product of two finite vectors defined at discrete points on the surface:

$$\langle \mathsf{a}, \mathsf{b} \rangle \approx \sum_{i=1}^{N_{verts}} a(\mathbf{v}_i) b(\mathbf{v}_i) A(\mathbf{v}_i)$$

Then the discrete estimate of Equation (3.12) is:

$$\Delta \mathcal{J}_\star^\infty \approx \frac{1}{2} \sum_i \left(\frac{\partial \mathcal{J}}{\partial \mathsf{S}} - \lambda_j \frac{\partial \mathcal{C}_j^a}{\partial \mathsf{S}} \right)_i (-\delta \mathsf{S}_\star)_i A_i \tag{3.13}$$

The discrete locations i correspond to the locations of the shape parameters X_i, which includes both existing and candidate design variables. For simplicity, I will assume that the shape modes have compact support, so that each is localized to a region of the surface.[7] This will be sufficient for the purposes of this thesis, but would require slight modification to be appropriate for highly-overlapping shape modes.

For design variable X_i, let A_i indicate its characteristic "footprint" area (in 2D, A is a width)[8]. Let h represent the unit deformation magnitude, corresponding to the units of the deformation mode. It is reasonable to assume that the magnitude of a unit deformation remains the same within each shape control tree, regardless of location or depth in the tree. If the units of X_i are the same as the units of $\delta\mathsf{S}$ then $h = 1$. By describing the deformation modes in these rough terms, the claim is that the specific *shape* of the mode is not important, as long as it remains consistent with refinement. This assumption is validated in Appendix B.3.1.

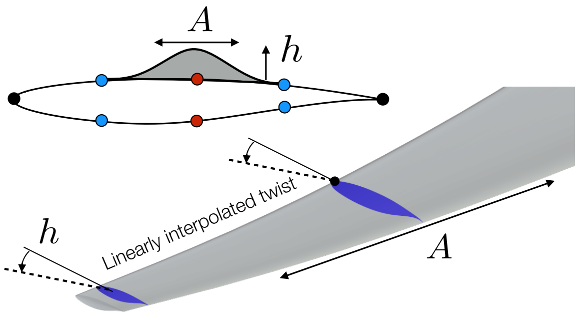


Figure 3.3: Shape mode characteristic dimensions A and h for two parameterization techniques. For the twist case, h might indicate, e.g. 1° twist.

The gradient vectors are obtained by projecting the continuous gradient density $\frac{\partial \mathcal{F}}{\partial \mathsf{S}}$ into

[7]Technically, I assume that the shape modes do not overlap, but the results show this to be sufficiently accurate even with modest overlap.

[8]For many parameterizations it is obvious how to define A_i, but it can also be computed directly as an integration over the surface: $A_i = \frac{1}{h} \int_{\mathsf{S}} \left| \frac{\partial \mathsf{S}}{\partial X_i} \right| dA$

the finite search space defined by the deformation modes $\frac{\partial \mathsf{S}}{\partial \mathbf{X}}$:

$$\frac{\partial \mathcal{F}}{\partial X_i} = \left\langle \frac{\partial \mathcal{F}}{\partial \mathsf{S}}, \frac{\partial \mathsf{S}}{\partial X_i} \right\rangle \tag{3.14}$$

Taking $\frac{\partial \mathcal{F}}{\partial \mathsf{S}}$ to be its average value within the region corresponding to shape mode i, Equation (3.14) can be approximated as

$$\left(\frac{\partial \mathcal{F}}{\partial \mathsf{S}} \right)_i \approx \frac{1}{A_i h} \frac{\partial \mathcal{F}}{\partial X_i} \tag{3.15}$$

This clarifies the fact that the continuous gradient with respect to the surface is in fact a gradient *density*, with units $\frac{[\mathcal{F}]}{[\mathrm{X}] \cdot L^2}$. Equation (3.15) is a valid approximation for *any* design functional (see Figure B.1) but only applies to localized, non-overlapping deformation modes. Using Equation (3.15), Equation (3.13) becomes

$$\Delta \mathcal{J}_\star^\infty \approx \frac{1}{2} \sum_i \frac{1}{h} \left(\frac{\partial \mathcal{J}}{\partial X_i} - \lambda_j \frac{\partial \mathcal{C}_j^a}{\partial X_i} \right) (-\delta \mathsf{S}_\star)_i \tag{3.16}$$

First-order Indicator

If we ignore the Hessian, we can now derive a first-order estimate of design improvement. From Equation (3.16) and Equation (3.10), taking $\mathcal{H} = \mathcal{I}$, and again using Equation (3.15), we obtain

$$\Delta \mathcal{J}_\star^{\mathcal{H}} = \frac{1}{2} \sum_{i=1}^{N_{DV}} \frac{1}{A_i h^2} \left(\frac{\partial \mathcal{J}}{\partial X_i} - \lambda_j \frac{\partial \mathcal{C}_j^a}{\partial X_i} \right) \left(\frac{\partial \mathcal{J}}{\partial X_i} + \lambda_j \frac{\partial \mathcal{C}_j^a}{\partial X_i} \right) \tag{3.17}$$

To compute the KKT multipliers, we project Equation (3.11) into $\frac{\partial \mathsf{S}}{\partial \mathbf{X}}$:

$$\lambda_i \frac{\partial \mathcal{C}_i^a}{\partial \mathbf{X}} = - \frac{\partial \mathcal{J}}{\partial \mathbf{X}} \tag{3.18}$$

This is a set of N_{DV} equations that form an over-determined system, which is only exactly satisfied at the optimum in the finite search space. In a refined candidate search space, the shape will certainly not be optimal. Therefore, I compute a first-order estimate of the KKT multipliers using an iterative bounded least-squares solver.[9] Higher-order approximations of λ_i are possible, but it is challenging to guarantee that they are actually more accurate than this first-order estimate [110].

[9] I used `scipy.optimize.leastsq` from the SciPy library [109], with high-weighted quadratic penalties on bounds violations.

In the absence of active constraints, Equation (3.17) becomes

$$\Delta\mathcal{J}_{\star}^{\mathcal{H},\not{\mathcal{C}}} = \frac{1}{2}\sum_{i=1}^{N_{DV}} \frac{1}{A_i h^2}\left(\frac{\partial\mathcal{J}}{\partial X_i}\right)^2 \tag{3.19}$$

Like Equation (3.1) (the indicator from [111]), this indicator uses only objective gradients. However, unlike Equation (3.1), this one accounts for the gradients for *all* of the design variables.

Second-order Indicator

As we will see in the verification studies (Chapter 5), the first-order indicator can make grossly inaccurate predictions of performance, even on fairly simple problems. This is because it omits Hessian information, which is essential both when redundancy is present and when comparing different classes of shape control. For these reasons, I now develop a second-order estimate of design improvement potential. This is substantially more challenging, and so a detailed discussion is relegated to Appendix B, with the salient results are restated here. The Newton step $\delta\mathsf{S}_{\star} = -\mathcal{H}^{-1}\left(\frac{\partial\mathcal{J}}{\partial\mathsf{S}} + \boldsymbol{\lambda}\frac{\partial\boldsymbol{\mathcal{C}}^{a}}{\partial\mathsf{S}}\right)$ must be projected into the discrete space, so that it can be expressed in terms of an approximation of the Hessian matrix $\overline{\mathbf{H}}$. Due to overlap of shape modes, nonlinearity, and the nature of the objective function itself, $(\delta\mathsf{S}_{\star})_i$ is not necessarily equivalent to the step taken by a black-box quasi-Newton optimizer. As before, the deformation modes are assumed to be localized. For global modes, the following derivations would require some modification. For general objectives, the problem is quite broad, and so I make the following restrictions:

1. The surface deformation is sufficiently linear with respect to the design variables.
2. The objective Hessian operator $\mathcal{H}$ is a local differential operator.
3. Within a given shape control tree, the deformation modes have consistent shapes and characteristic magnitudes (or units).

These are not fundamental restrictions on the general idea, but they greatly simplify the derivation and are sufficient for the present purposes. More detailed discussion is given in Appendix B.

Using these assumptions, Appendix B.2 derives discretized approximations of the Newton step $\delta\mathsf{S}_{\star}$ for two specific objective functions. The first is geometric shape matching:

$$\mathcal{J}^{GSM} := \frac{1}{2}\int_{\mathsf{S}} (\mathsf{y}(\mathbf{x}) - \mathsf{y}^{*}(\mathbf{x}))^2 d\mathsf{A}(\mathbf{x}) \tag{3.20}$$

where $\mathbf{x}$ is a coordinate on the surface, $\mathsf{y}(\mathbf{x})$ is the surface deformation and $\mathsf{y}^*(\mathbf{x})$ is the target shape. For this objective, using $\delta\mathsf{S}_\star$ from Equation (B.12), Equation (3.16) becomes

$$\Delta\mathcal{J}_\star^{GSM} = \frac{1}{2}\sum_{i=1}^{N_{DV}} \left(\frac{\partial\mathcal{J}}{\partial X_i} - \lambda_j \frac{\partial\mathcal{C}_j^a}{\partial X_i}\right) \frac{1}{H_{i,i}} \left(\frac{\partial\mathcal{J}}{\partial X_i} + \lambda_j \frac{\partial\mathcal{C}_j^a}{\partial X_i}\right) \tag{3.21}$$

Now consider a 2D inverse surface pressure-matching functional

$$\mathcal{J}^P = \frac{1}{2}\int_{\mathsf{S}} (p(\mathbf{x}) - p^*(\mathbf{x}))^2 d\mathsf{A}(\mathbf{x}) \tag{3.22}$$

where $p^*(\mathbf{x})$ is the target profile. For this objective

$$\Delta\mathcal{J}_\star^{P} = \frac{1}{2}\sum_{i=1}^{N_{DV}} \left(\frac{\partial\mathcal{J}}{\partial X_i} - \lambda_j \frac{\partial\mathcal{C}_j^a}{\partial X_i}\right) \frac{1}{A_i}\frac{1}{H_{i,i}}\frac{1}{4} \left(\frac{Z_{i-1}}{A_{i-1}} + 2\frac{Z_i}{A_i} + \frac{Z_{i+1}}{A_{i+1}}\right) \tag{3.23}$$

where $Z_i := \frac{\partial\mathcal{J}}{\partial X_i} + \lambda_j \frac{\partial\mathcal{C}_j^a}{\partial X_i}$.

Unlike for the shape matching objective, Equation (3.23) contains a term proportional to $\frac{1}{A^2}$, which explicitly accounts for scale-dependence of the Hessian matrix due to discretization of the shape control. Appendix B shows that without this term, the expected improvement would falsely appear to decrease as more shape control was added. The presence of a scale-dependent term is likely to arise for other aerodynamic functionals as well, and it is important to not neglect it. While we might expect that $\mathcal{J}_\star(\mathbf{C}^{k+1}) > \mathcal{J}_\star(\mathbf{C}^k)$ (more degrees of freedom should offer more potential), $\mathcal{J}_\star(\mathbf{C})$ should not *explicitly* grow or shrink based on the resolution of shape control. If it did, then in the limit of refinement $\mathcal{J}_\star^\infty$ would tend to either zero or infinite potential, which are both clearly incorrect.

3.3 Efficient Computation of the Indicator

The estimates developed in the previous section require computation of gradients and a Hessian diagonal, both in a *hypothetical* candidate search space. This section shows how these can be computed at very low cost — no additional PDE solutions are required. The approach is based on two premises:

1. An adjoint approach is used, so that $\frac{\partial\mathcal{F}_j}{\partial\mathbf{X}}$ can be computed via inner products.

2. A quasi-Newton optimizer is used to solve each level of Algorithm A, so that a Hessian approximation $\overline{\mathbf{B}}^k$ is available.

In general, I preclude any modification of the existing design tools, so that the method can be used immediately with any standard combination of flow and adjoint solver, quasi-Newton optimizer, and geometry modeler.

Candidate Gradients

During optimization, the adjoint solutions $\Psi := \{\psi_o, \psi_j\}$ are used to efficiently compute gradients with respect to the design variables. Upon shape control refinement, assume that the shape itself does not change:

$$\mathsf{S}^k_{final} = \mathcal{D}^k(\mathbf{X}^k_{final}) = \mathcal{D}^{k+1}(\mathbf{X}^{k+1}_0) = \mathsf{S}^{k+1}_0 \tag{3.24}$$

where k and $k+1$ are the current and subsequent search spaces. Then the first iteration in the new search space is identical to the final iteration in the previous space, meaning that the adjoint solutions are identical:

$$\Psi^k_{final} \equiv \Psi^{k+1}_0$$

for *any* possible re-parameterization. This is the fundamental insight that will allow rapid computation of gradients with respect to the candidate design variables. To compute these gradients, we can project the existing sensitivities encoded in Ψ^k_{final} into the candidate deformation modes. For the purposes of estimating design improvement, Ψ^{k+1}_0 does not need to be computed.[10]

The process for computing the gradients is given in Appendix A, Function 4. Parallelism can happen at coarse and fine levels. Computation of shape derivatives and gradient projections are completely independent. Furthermore, the gradient projection tool I use supports internal fine-grained parallelism [95].

This procedure is only valid if the shape is preserved exactly when refining the shape control, which I refer to as a "neutral shape mutation", following Olhofer *et al.* [86]. All discrete geometry modelers inherently support this — the static baseline shape can simply be reset to the current shape at any time. In general, constructive (CAD-like) modelers do not seamlessly preserve the geometry when changing the parametric definition, but rather perform a refitting procedure to minimize the "jump" in the shape. Even then, any remaining discrepancy between S^k_{final} and S^{k+1}_0 casts uncertainty on the validity of using the previous adjoint solutions. Chapter 4 discusses this issue in greater detail.

[10]In fact, even after refining the parameterization, Ψ^{k+1}_0 would not need to be computed. The flow and adjoint solutions for the first iteration could be skipped entirely, assuming no other aspect of the problem definition (e.g. mesh resolution) has changed.

Candidate Hessian Matrix

Exact computation of the Hessian is typically avoided, because of the high computational cost and memory requirements. However, even approximate information on scaling and redundancy could substantially improve the identification of an effective set of shape control. A progressive parameterization approach provides a unique opportunity to extract existing Hessian information. Assuming there is some underlying continuous Hessian that remains consistent as we project it into finer and finer shape deformation modes, then it is reasonable to expect $\overline{\mathbf{H}}^k_{final}$ to be closely related to $\overline{\mathbf{H}}^{k+1}_0$, although their dimensions are different. With quasi-Newton optimization (e.g. BFGS), by the end of level k we have an approximation to the Hessian $\overline{\mathbf{B}}^k_{final}$. This suggests taking $\overline{\mathbf{H}}^{k+1}_0 \approx \mathcal{E}(\overline{\mathbf{B}}^k_{final})$ where $\mathcal{E}$ is some function that expands or "prolongs" the current estimate of the Hessian into the finer candidate search space. Naturally this only makes sense with the type of spatially organized shape control we are using here.

A convenient benefit of this approach is that the BFGS approximation is always positive definite by construction. In optimization, this is accepted to be an effective way to ensure navigability of non-convex regions of a design space. Here, it makes the adaptation procedure more robust to non-convexity by ensuring that the refinement indicator is always positive.

Appendix B.4 provides some background on a number of alternative approaches that compute exact or approximation Hessian information, including direct derivation, finite-differencing, second-order adjoints, and automatic differentiation. However, all of these approaches either (1) require at least the equivalent of $\mathcal{O}(N_{DV})$ linear PDE solutions, which is prohibitive in this case, (2) require invasive, objective-specific modification to solvers, which I want to avoid here, or (3) make insufficiently accurate assumptions. The present quasi-Newton prolongation approach is particularly advantageous in that it uses information *already* obtained during optimization, which makes the overhead cost negligible. Its major unknown factor is the veracity of the approximation $\overline{\mathbf{B}}^k \approx \overline{\mathbf{H}}^k$. However, there is very little burden on the accuracy of this approximation. As long as it is reliably superior to taking $\overline{\mathbf{H}}^k = \bar{\mathbf{I}}$, as assumed by the first-order indicator, then it is worthwhile.

Hessian Prolongation Operator

For now, I consider transferring only the Hessian diagonal, not the entire matrix. This is sufficient under the assumption of localized shape modes, but would need to be expanded for global modes. The basic form of the operator is linear interpolation. Say that on transitioning from search space k to $k+1$, a new parameter j is being added between existing

Table 3.1: Scale-dependent terms M in Hessian prolongation operator (Equation (3.25)) for two objective functions, expressed in terms of the deformation modes' footprint areas A (see Figure 3.3).

Objective	M_i^j
Geometric shape matching (Equation (3.20))	$\frac{A_j^{k+1}}{A_i^k}$
2D surface pressure matching (Equation (3.22))	$\frac{A_i^k}{A_j^{k+1}}$

parameters L and R, as illustrated in Figure 2.8. Then we can estimate the new diagonal Hessian entry for this shape controller from the old diagonal entries of its neighbors:

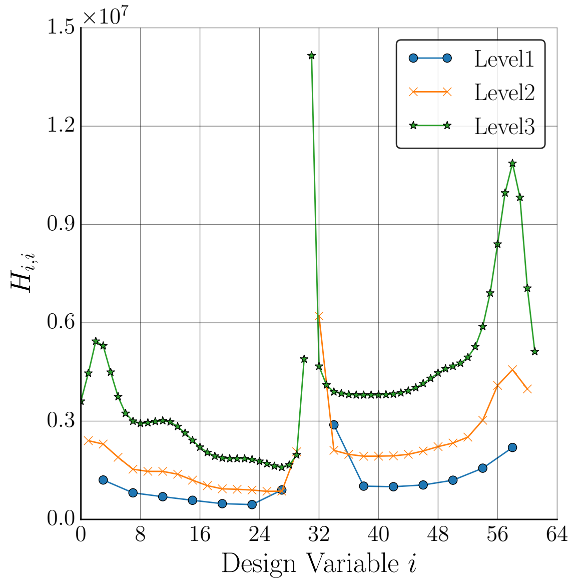
$$H_{j,j}^{k+1} = (1-u)M_L^j H_{L,L}^k + uM_R^j H_{R,R}^k \tag{3.25}$$

where u is the fraction of the distance between L and R at which j is located. If the shape control is refined at midpoints instead of being skewed or biased in one direction, then $u = 0.5$. The term M encodes dependence on the shape and size of the deformation modes. Appendix B.3 shows that its form depends on the objective function and derives it for two functionals: geometric shape matching (GSM) and surface pressure matching. Table 3.1 shows how these can be expressed approximately in terms of the shape modes' characteristic dimensions (recall Figure 3.3). For a parameterization with compact support, the deformation modes' footprints become narrower with refinement. In the limit as $A \to 0$, $M^{GSM} \to 0$ while $M^P \to \infty$. The latter is reflective of the ill-conditioning of aerodynamic optimization with high-frequency shape modes [112].

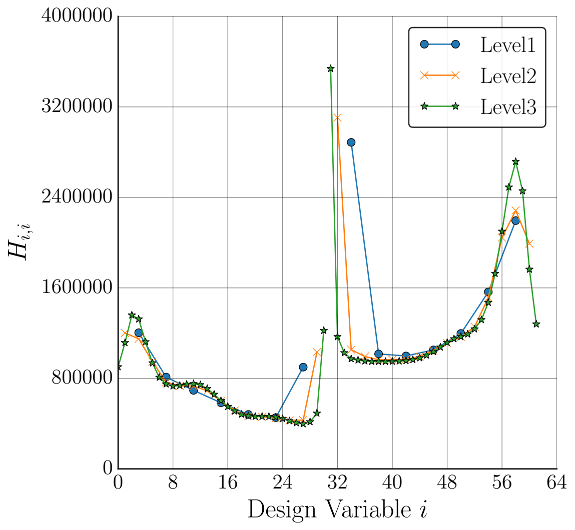
In addition to the assumptions listed in Section 3.2.2, this prolongation operator assumes that physically adjacent parameters have similar Hessian entries. This implies the existence of a continuous Hessian which varies smoothly along the surface.[11] Despite these assumptions, Equation (3.25) is reasonably accurate, as shown next.

To verify the prolongation, consider an airfoil pressure-matching problem. (This problem is examined in more detail in Section 5.2.) Note that while the indicator derived for this problem assumes that the pressure profile is sufficiently close to the target, for this

[11] At shocks, the Hessian (and gradient) can change discontinuously. In these cases, the linear interpolation "smears" through the jump. On the other hand, Telib *et al.* show that in transonic flow, oscillations in the Hessian can arise from fine discretizations [113], and that coarser discretizations of the Hessian can actually be more representative than finer ones.



(a) Unscaled



(b) Scaled using Equation (B.23)

Figure 3.4: Hessian diagonal for 2D inverse pressure matching on airfoil at various shape control resolutions. Left/right segments on each plot correspond to camber/thickness. Trailing edge is at the center, leading edge on either side.

test, the baseline is substantially far from the target (see Figure 5.11). Each curve in Figure 3.4a shows the Hessian diagonal computed at a different shape control resolution. In order to eliminate any errors due to the quasi-Newton approximation, and evaluate the prolongation operator in isolation, the Hessians were computed by finite differencing the adjoint-derived gradients. Figure 3.4b shows the same curves, but transformed by M^P from Table 3.1. As the shape control is refined, the Hessian diagonal is approaching a smooth, continuous distribution. The scale term for pressure matching technically applies only near the target. The accurate performance in this example indicates that this constraint may be relaxed. Additionally, Appendix B.3.1 shows that it is quite accurate for other 2D surface pressure-based objectives, including drag and lift, and is not unreasonable even for a 3D off-body pressure-matching functional.

3.4 Adaptation Strategy

The previous sections developed a refinement indicator to evaluate the effectiveness of any given combination of design variables. I now present a strategy for efficiently searching for an effective parameterization. The following section shows that this is a combinatorial optimization problem. Although no PDE solutions are required, an exhaustive search to find the globally ideal parameterization would still require $\mathcal{O}(N_{DV}!)$ gradient projections. To control the cost of the adaptation process, I develop an approximate constructive algorithm, which reduces the asymptotic complexity to $\mathcal{O}(N_{DV}^2)$ gradient projections.

3.4.1 Non-Separability of the Indicator

It would be convenient if each candidate parameter could be assigned a priority independently of the other parameters. The indicator would then be *local*, and we could simply add the top-ranked fraction of candidates to the active set of design variable. This type of adaptation procedure is straightforward and inexpensive and is frequently used in discrete mesh adaptation approaches (e.g. [114, 115]). For this to be possible, the indicator would need to be separable, i.e. expressible as a linear combination of the shape control:

$$I(C_0 \cup C_1 \cup ...) \overset{?}{\approx} I(C_0) + I(C_1) + ... \tag{3.26}$$

This is invalid for two reasons. First, the indicator is a nonlinear function of the set of shape modes, due to both the Lagrange multipliers (Equation (3.18)) and also the off-diagonal terms of the Hessian, which inextricably couple the shape modes. Incorporating this information on redundancy would be necessary for overlapping deformation modes. However, even under our assumption of non-overlapping modes, *candidate* design variables may be overlapping, if we are searching more than one level deep in the tree (see Section 2.3). An example of this will be seen in Section 5.1.2. The second major reason why Equation (3.26) is invalid is that the shape modes are themselves a nonlinear function of the shape control via the parameterization function $\mathcal{P}$, as illustrated in Figure 3.1. The same candidate can thus induce *different* deformation modes, depending on the location of the other shape controllers. Though it might happen that Equation (3.26) is a reasonable approximation in certain cases, we cannot rely on this for a system that is expected to work with arbitrary parameterization techniques.

In contrast to the typical localized approach to discrete mesh adaptation, finding the best ensemble of shape parameters is therefore a combinatorial optimization problem. Despite the relatively low cost of a single indicator evaluation, an exhaustive search quickly becomes prohibitive: there are $\mathcal{O}(N_{DV}!)$ possible combinations of candidates to consider. This complexity was bypassed in [111], by adding only one new parameter at a time — a severe restriction that leads to poor computational efficiency. Simple non-exhaustive procedures like random sampling are highly unlikely to find a good combination of parameters without very many samples. Metaheuristic search procedures such as evolutionary optimization [116] might be more likely to find a globally optimal combination of parameters, but would likewise require extremely large numbers of indicator evaluations.

3.4.2 Approximate Constructive Adaptation Procedure

To control the cost of the adaptation procedure, I use an approximate constructive ("greedy") selection procedure, illustrated in Figure 3.5. The goal is to find an effective parameterization for low cost, rather than spend an exorbitant amount of time to find the ideal parameterization. I believe this is a well-justified trade, especially in the context of progressive shape parameterization. At worst, a suboptimal parameterization will only temporarily reduce the efficiency until the next refinement. For applications where finding a truly optimal set of design variables is imperative, global search procedures should be investigated.

The constructive adaptation procedure is given explicitly in Appendix A, Function 5. In the first phase, an initial ranked priority queue of candidates is generated by computing the indicator value for each possible introduction of a single new parameter. Thus the priority of each candidate is the additional potential that it would add over the existing parameters. Usually this potential will be strictly increasing, except when a candidate shape controller cancels the potential due to another.

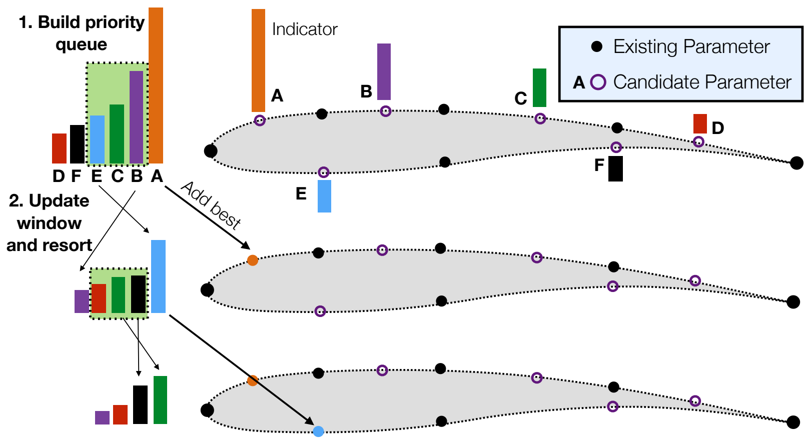


Figure 3.5: Approximate constructive refinement procedure

In the second phase, the system makes consecutive passes over the priority queue, adding the highest-ranked parameter on each pass. However, to account for the nonlinearities described earlier, the candidates in the queue must be re-evaluated in the context of the already-added parameters. To control the cost, only a window of the top few remaining candidates (N_w) is actually re-evaluated. Afterwards the queue is resorted, the top-ranked parameter is added, and the next pass begins.

Cost and Accuracy

This procedure is motivated by the expectation that similar parameterizations will offer similar potential, and is therefore most effective when the priority queue remains substantially similar from pass to pass. For cases where there is little redundancy among candidates, this is a reasonable assumption, and the procedure often finds the global optimum at a fraction of the cost of an exhaustive search. Section 5.1 gives a counterexample, where there is high redundancy among the candidates, and this procedure yields less optimal results. The choice of N_w strikes a balance between dedicating more resources to search for a more effective parameterization vs. starting to make design progress immediately, but in a less effective search space. If the initial priority queue is trustworthy, one could use a window size of $N_w = 0$, which is equivalent to accepting the approximation in Equation (3.26).

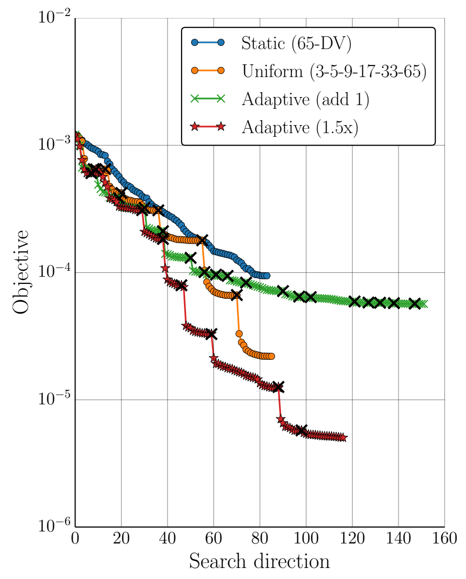
Appendix A shows that the upper bound on the cost of the entire refinement procedure is $\mathcal{O}(N_{DV}^2)$ shape sensitivity computations and $\mathcal{O}(N_{DV}^2 N_\psi)$ adjoint inner products, where N_ψ is the number of adjoint solutions involved. The cost also scales linearly with N_w and exponentially with depth of candidates considered d. The wall-clock time additionally depends on the speed of the frequently-invoked geometry modeler and gradient projection tools. Typical running times for the refinement procedure are equivalent to no more than a few design iterations, which is usually far outweighed by the savings resulting from reducing the search space dimension. However, at very large numbers of design variables, the $\mathcal{O}(N_{DV}^2)$ cost may begin to become problematic. This cost appears to be irreducible, due to the non-separability of the indicator computations. Fortunately, the process can be parallelized, and it is not uncommon for a case to be run on $\mathcal{O}(N_{DV})$ parallel processors. The parallelization can happen at several levels: first, over the candidate refinements; second, over the design variables and design functionals; and third, the gradient projection tool itself may be a parallel code.

3.4.3 Growth Rate

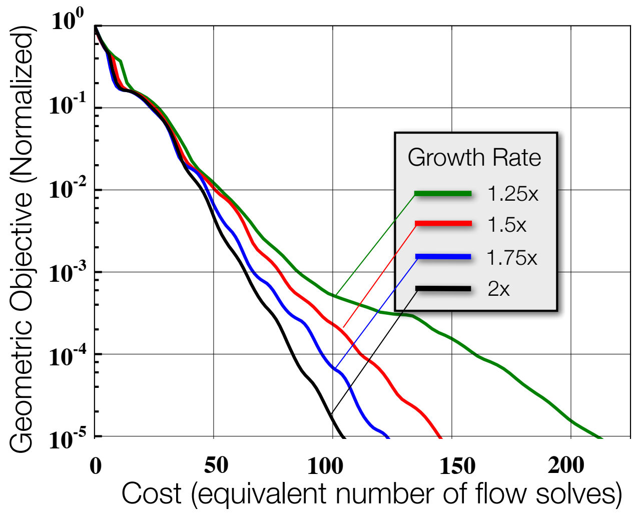
The uniform refinement strategy involved only one tuning parameter, namely the trigger, discussed in Section 2.4.1. Adaptive refinement now introduces an additional parameter, the relative growth rate in the number of design variables, which has a critical impact on performance. The optimal growth rate depends on the problem, as shown in Figure 3.6. For more complex problems, such as the 3D supersonic boom-signature matching design problem to be considered in Section 5.3, slower growth rates can prove to be faster overall. As shown in Figure 3.6a, an intermediate growth rate of 1.5× solidly outperforms both a slower growth rate (adding one parameter at a time) and uniform refinement (equivalent to 2×). The curve labeled "Adaptive (add 1)" is an important cautionary note. Even with accurate adaptation, an inappropriate growth rate can result in significantly worse performance than simple uniform refinement.

For simpler problems, higher growth rates tend to be more efficient. An example of this is shown in Figure 3.6b, which shows averaged objective convergence for a randomly generated sampling of 2D geometric shape-matching problems. Here, the well-behaved design space allows rapid and reliable design improvement, regardless of the number of design variables, and thus fast growth rates are favored. Note that these examples involve only one-dimensional refinement of the shape control. Adaptation becomes even more critical for optimization of an entire surface, where uniform refinement would increase N_{DV} by a factor of 4× at every refinement, rather than 2×.

Although the designer could manually preset the growth rates, this is an unintuitive task that nevertheless has a major impact on efficiency. This provides a strong incentive to



(a) Supersonic inverse design to off-body target pressure signature. ×-marks denote search space refinements.



(b) Geometric shape-matching objective. Each curve shows mean behavior of 10 randomized trials ($r = 0.25$).

Figure 3.6: Impact of growth rate in N_{DV} on efficiency. Each curve shows objective convergence using a different growth rate strategy.

develop a method for automatic growth rate determination. The optimal rate is difficult to know a priori and varies considerably from problem to problem, and perhaps also from level to level of shape control.

Automatic Growth Rate

To automatically determine an appropriate growth rate, I monitor for diminishing expected returns on the addition of successive new parameters. During the constructive refinement procedure (Function 5), parameters are added one at a time, gradually increasing the search space's potential for design improvement $\Delta\mathcal{J}_\star$. The indicator I (Equation (3.3)) directly measures how much potential each candidate adds. By monitoring the history of I with adaptation, we can watch for diminishing returns. Once the indicator values decrease sufficiently, it can be inferred that adding more parameters would not be worthwhile.

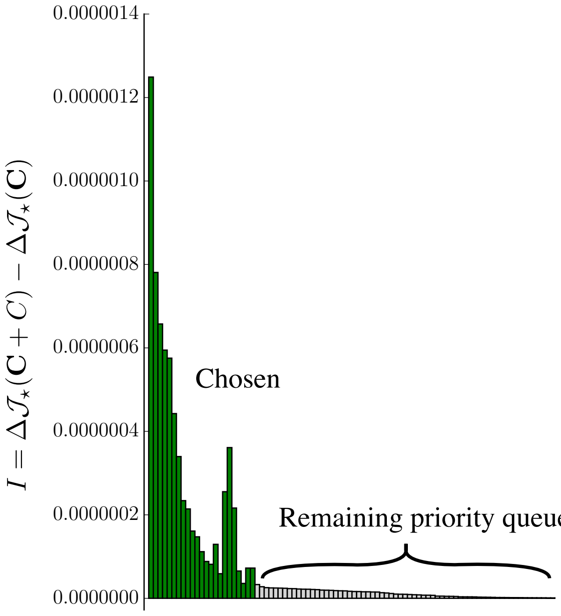


Figure 3.7: Reduction of indicator with addition of shape control. The refinement procedure has been terminated after consecutive new shape controllers (green) demonstrated consistently diminished priorities ($r = 0.1$, $w = 3$).

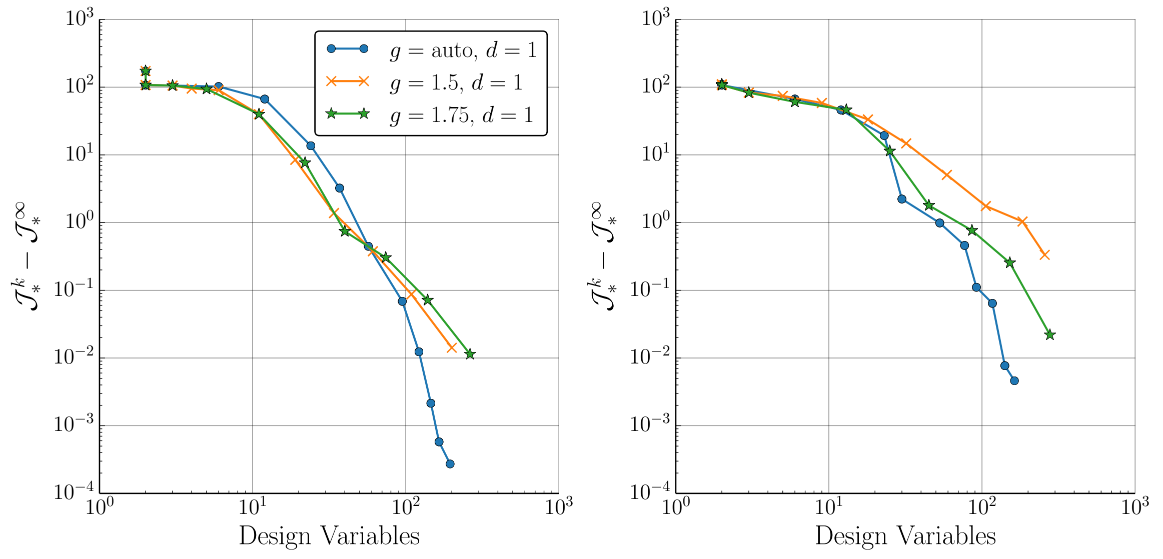
This approach is directly analogous to the TRIGGER($\cdot$) function in Section 2.4.1, and I apply the same logic here. The difference here is that $\Delta\mathcal{J}_\star$ is monotonically *increasing*,[12] so there is a sign reversal. Adaptation is terminated once

$$\frac{I_i}{I_1} < r$$

Setting the cutoff factor r for the slope-trigger constitutes a crude but effective cost-benefit analysis. Adding each new design parameter is assumed to incur some fixed cost, due to adding a dimension to the search space. When the incremental benefit of adding successive parameters becomes not worth the incremental cost of adding the parameter, then the selection process should be terminated, and optimization resumed.

Generally speaking, I_i will decrease as more parameters are added, but when there is high interdependence among the parameters, it may not decrease monotonically. This can

[12]This is because Algorithm 5 only enacts refinements that increase the potential. For an arbitrary refinement, the potential could possibly decrease.



(a) Case 1 **(b)** Case 3

Figure 3.8: Performance of automatic growth vs. fixed growth rate on 2D inverse design example (more details in Section 5.2). Each curve indicates a different refinement strategy. Each datapoint indicates the best objective value attained within a suboptimization level of Algorithm A.

be seen in Figure 3.7, where a cluster of parameters forms a second peak in the greedy search history. The mode shapes for these shape control elements were less valuable in isolation, but after the addition of other shape controllers, their mode shapes were modified into more effective shapes, and thus their priority was higher in the new context. Following the approach in Section 2.4.1, I use a smoothing window w to avoid terminating the adaptation too early. However, w should be moderately small to avoid adding many extra parameters. I found $w = 3$ or $w = 4$ to work well for most cases.

Figure 3.8 evaluates this approach on a pressure-matching objective. Each plot shows convergence with respect to the number of design variables. In each case, the automatic strategy performs comparably or substantially better than fixed growth rate strategies in terms of minimizing the number of design variables. How this translates into wall-clock time savings will be considered in the results section.

Having an automatic growth rate might also provide a solution to another important question: how many shape parameters to *start* with. Starting with a truly “minimal” search space usually leads to stunted progress early on. It is usually more efficient to start with a moderate number, though the precise number is problem-dependent. To automatically determine this, Algorithm A could be modified by adding a pre-refinement stage before the loop begins. An single initial flow solve (with adjoints) would be performed, immediately followed by an adaptive refinement to determine a good initial parameterization.

3.4.4 Regularity

Even if the indicator is computed with exact Hessian information, with no simulation error, and in a perfectly smooth design space, it is still only a local second-order expansion. It is therefore important to develop an adaptation strategy that will be robust to nonlinearity in the design space. To this end, I impose a set of heuristic refinement regularity rules, illustrated in Figure 3.9. Before adding a shape controller C, the following prerequisite controllers must also be present:

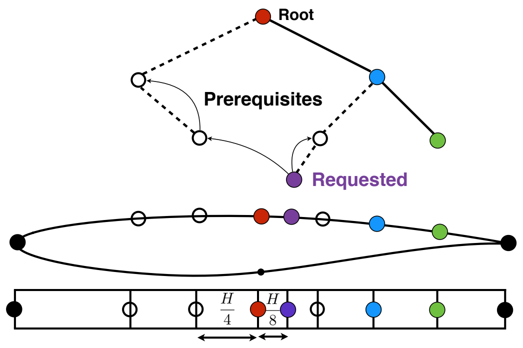


Figure 3.9: Regularity prerequisites for airfoil parameterization. Addition of the requested parameter (in purple) would lead to excessive variation in parameter spacing. The system automatically adds its two prerequisites. One of those, in turn, requires addition of a third parameter.

1. C's parent
2. The Nth "ancestor" of each of C's nearest equal-depth neighbor addresses to the left and right.[13]
3. (For certain deformers) The Mth "ancestor" of the slot opposite C.

If any are not present, they are automatically added. Alternatively, one could ignore parameters missing their prerequisites, as in [106]. However, the growth rate can severely stagnate under such an approach. Highly effective design variables can be precluded merely because the system is not able to see the importance of adding their prerequisites.

The rules are applied recursively, as shown in Figure 3.9. If there are multiple trees, then at least one new parameter should be added to each tree per cycle. Any added prerequisites are removed from the priority queue to avoid redundant indicator evaluations.

The first two prerequisites enforce N-regularity of the parameter spacing, prohibiting large discrepancies between the refinement depth at adjacent regions on the surface. For most parameterization techniques[14], irregularly spaced shape control yields irregularly shaped deformation modes, which can result in non-smooth design trajectories and slow convergence. Enforcing some degree of regularity makes the entire process more robust, although it is necessarily suboptimal from the standpoint of finding a minimal number of

[13]For example, with $N = 2$, to add shape controller `LLRR` requires `LL` and `LR`.

[14]Specifically, any technique with local support and variable mode width. This includes almost all techniques, with the exception of algebraic bump functions.

design variables. Prerequisite (3) is advisable for volumetric deformers that modify two independent sides of the same surface. For example, for airfoil parameterization, the shape control can be divided into separate trees for the top and bottom surfaces. If using free-form deformation, or radial basis functions whose distance is measured in Euclidean space (not along the surface), I apply this third prerequisite, which prevents excessive discrepancies between the resolution of the shape control of opposite regions. The precise degree of irregularity to allow depends on the robustness of the entire design suite (optimizer, flow and adjoint solver, geometry modeler); for this work, I used a strict value of $N = 1$ and, when applicable, $M = 1$.

Taken together, these rules help ensure that sufficient refinement will fill the shape control trees. Deep refinement in one region will be buffered by adding parameters to adjacent regions. This process eventually propagates through the full tree, so that the adaptation will converge to the continuous shape control problem.[15] Even for perfectly smooth flow solutions with negligible discretization error, this feature can be essential. Under the discrete refinement mechanics I use, it can easily be the case that a visible candidate would not itself be helpful, while its children would be very helpful. It is possible to imagine the pathological situation illustrated in Figure 3.10, where such a candidate might never be added, and thus the potential of its children would never be realized. The regularity rules are one way to preclude this situation.

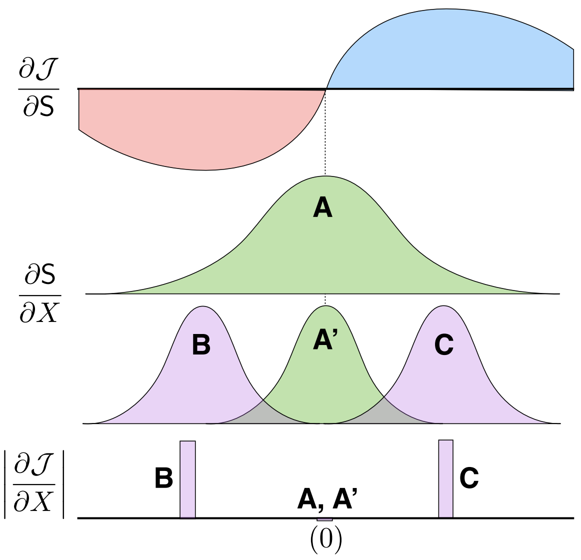

$\frac{\partial \mathcal{J}}{\partial \mathsf{S}}$
A
$\frac{\partial \mathsf{S}}{\partial X}$
B
A'
C
$\left|\frac{\partial \mathcal{J}}{\partial X}\right|$
B
A, A'
C
(0)

Figure 3.10: Parameter **A** will not be prioritized, because its net influence on the objective is zero. This conceals the potential for improvement offered by its higher resolution children, **A'**, **B** and **C**. The regularity rules alleviate this situation by ensuring that **A** will eventually be added.

[15]The regularity rules do not change the possibility that any given optimizer may not be able to actually *navigate* the design space, perhaps due to non-smoothness or inaccuracies. The rules merely avoid limitations due to myopic adaptation.

CHAPTER 4

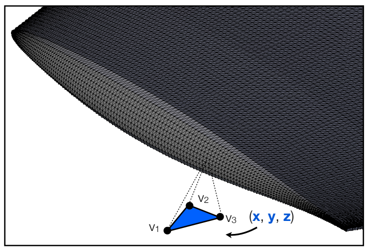
GEOMETRY MODELERS

The geometry modeler is often carefully crafted for specific design tasks and may constitute a repository of design experience and best-practices. The present work therefore aims to develop a "modeler-agnostic" system; discussion of any particular geometry modeler has therefore been deliberately avoided to this point. This chapter now addresses the most important aspects of preparing a geometry modeler for use with adaptive parameterization. It also discusses how the choice of shape control basis can impact efficiency. Finally, it introduces specific discrete deformation techniques that will be used in the results chapters. One of these involves non-invasive retrofitting of an existing modeler that was not initially designed for progressive parameterization.

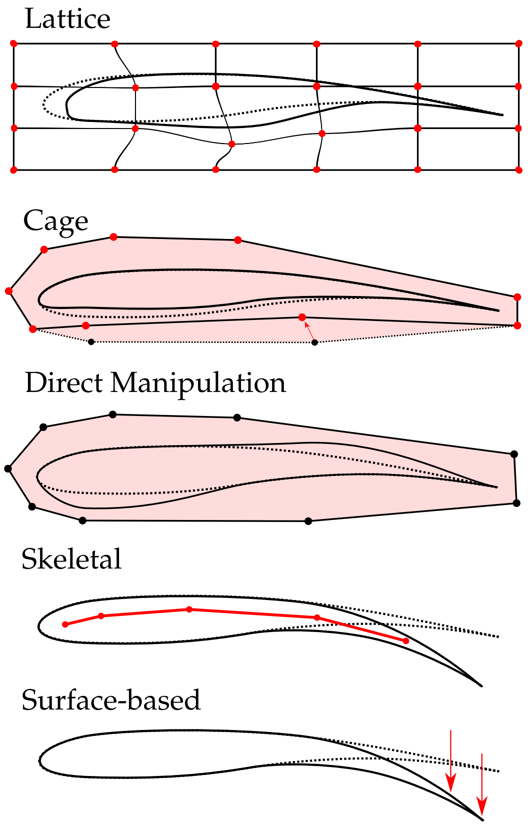
4.1 Background

At a high-level, every geometry modeler is either a constructor or a deformer. In the constructive modeling paradigm, a geometry is built from scratch according to a sequential recipe. Familiar constructive modeling elements include B-splines, Bernstein polynomials, Bézier curves, NURBS surfaces, and class/shape functions, all of which are used widely in aerospace design, either individually or as part of larger modeling libraries [47, 77, 108, 117–119]. Although their proprietary nature is often lamented, CAD modeling tools are sometimes used directly for optimization when necessary to maintain consistency with a broader design environment [117, 120–124].

Deformational approaches encode modifications to a fixed baseline geometry. In these approaches the baseline geometry is described as a discrete surface, such as a triangulation, as pictured in Figure 4.1a. Familiar examples used in aerodynamic design include bump functions [7, 125], various volumetric deformation techniques [126] including free-form deformation [5, 118, 127–134], cage-based deformation [135], plates-and-shells analogies [136], subdivision surfaces [80, 137], radial basis functions [138–141], and constraint-based deformation [130, 142]. Some of these approaches are illustrated in Figure 4.1b.



(a) Wingtip represented as a discrete triangulation



(b) Some discrete deformation techniques, depicted schematically

Figure 4.1: Deformation of discrete geometry

Except where noted, the motivating arguments and the final system are mostly compatible with either deformational or constructive modeling approaches. For this work, however, I use a discrete deformation approach. This choice is motivated by many factors. First, in deformational approaches, the complexity of the *shape* is decoupled from the complexity of the *shape control*, enabling low-dimensional optimization of geometrically complex shapes [57, 143]. Second, the shape control can be changed flexibly, while seamlessly preserving the geometry. Third, it enables study of a vast number of "legacy" geometries, where no recipe-based model exists. Finally, there are a number of powerful, scriptable, and extensible discrete geometry manipulation tools developed by the computer graphics industry. These tools can be leveraged directly for aerodynamic design, unlocking a wealth of surface manipulation techniques, and enabling rapid prototyping of new techniques. These tools also have mature graphical user interfaces that enable interactive preparation for automated shape optimization.

4.2 Requirements for Progressive Parameterization

This section discusses some of the hard requirements and services that a geometry modeler must provide, beyond those necessary in a standard design framework. For non-adaptive (uniformly-refined) progressive parameterization, the main restriction on the parameterization technique is that coarse refinement levels should be representative of finer levels, as discussed in Section 2.1.2. Most parameterization schemes satisfy this requirement. Some rare exceptions include deliberately low-dimensional fixed-parameter schemes such as PARSEC [144] or the approach in [145].

Lack of authority to modify a parameterization does not necessarily preclude the use of progressive parameterization. In these cases, we might apply dimension reduction approaches, as in [81]. However, it must be ensured that the various search spaces are consistent with each other, otherwise there may be no acceleration in design improvement, as happened with the Rosenbrock function in Section 2.1.2.

Automatic Generation and Refinement of Search Spaces

Consider a standard gradient-based shape optimization framework where the geometry modeler must:

1. ***Before optimization:*** Provide a list of available design variables, possibly with hard lower and upper bounds.

2. ***During optimization:*** Given any set of parameter values $\mathbf{X}$, generate a surface, optionally annotated with the shape derivatives $\frac{\partial \mathbf{S}}{\partial X_i}$ for each design variable.[1]

A geometry modeler integrated with a shape optimization framework likely already satisfies requirement (2). Moving to progressive parameterization requires that (1) be automated as well.

The geometry modeler must also implement one additional function:

- $\mathcal{P}(\mathbf{S}, \mathbf{C})$ — Generate a deformation function $\mathcal{D}$ and a set of design variables $\mathbf{X}$ from any set of shape control refinement locations $\mathbf{C}$ (Equation (2.2)).

This function represents automation of the traditionally manual task of re-parameterization. The details of this method depend on the geometry modeler, but typically involve modifying control parameters and settings. More concrete examples are given in Section 4.4, which discusses some specific geometry modelers.

[1]Shape derivatives can also be computed automatically by finite differencing.

It is assumed that any combination of the candidates yields a valid parameterization. If for some reason this is not true, then the modeler should silently resolve the conflict, returning the "closest" valid parameterization. As discussed in Section 3.4.4, to prevent the adaptation from stagnating, it is most efficient to err on the side of adding missing prerequisites, rather than omitting parameters without their prerequisites. A more involved system might keep track of dependencies and attempt to resolve them automatically. However, in situations where many combinations are invalid, this could lead to an algorithmically challenging space to navigate. Moreover, one would then need to explicitly codify the modeler's internal logic, which can not always be done in a straightforward manner.

Supporting Adaptation

To support localized adaptive parameterization, the shape control must be redistributable, to focus higher resolution on certain regions of the surface. Some techniques, although supporting progressive sequencing, do not support such localized adaptation. A good example is Bernstein polynomials (and thus class/shape parameters). These offer control over the number of polynomial basis functions to use, but not over their spatial distribution.

My approach also requires that the deformation modes be "spatially organized" and representable in binary tree form. It would not be particularly straightforward to define notions of adjacency and spatial organization for certain vector interpolation schemes such as [1, 4]. These exceptions aside, most modeling techniques support localized adaptation.

4.3 Implications of Shape Control Basis

In this work I leave the choice of shape control basis and geometry modeler to the designer. Nevertheless, this choice does have major implications, some of which are not obvious. Many authors have discussed various desirable qualities, including smoothness, compactness, intuitiveness, and orthogonality [4, 108, 132, 136, 146–152]. To this discussion, I add two more considerations that are specifically relevant for progressive parameterization.

4.3.1 Mode Shapes

Parameterization techniques can be roughly characterized by their "radius of support", as depicted in Figure 4.2. At one extreme, there are methods that involve highly-overlapping deformation modes that span the full extent of the shape. Examples include Hicks-Henne bump functions and Bernstein polynomials. At the other extreme are bases that consist of disjoint or slightly overlapping modes, such as radial basis functions or other forms of interpolation. These are essentially Galerkin discretizations of the shape control. Falling in between these extremes are various bases with compact support, such as free-form deformation lattices and B-splines. Hierarchical parameterization (e.g. with Fourier basis functions) can use both types at once.

This characteristic has implications for computational cost that are not immediately obvious. Chaigne and Désidéri investigate how the stiffness of the optimization depends on the shape control basis [79]. Deformations are globally smoother when using fully overlapping modes, which can substantially improve the speed and robustness of the optimization, because there are fewer evaluations of noisy design perturbations [153].

As more and more fully overlapping shape modes are added to the search space, it becomes more and more difficult to discern between the effectiveness of adjacent parameters. This is manifest in the density of the Hessian matrix. Let us assume that the continuous Hessian operator $\mathcal{H}$ is diagonal. (This appears to be the case at least for 2D aerodynamic functionals). For completely non-overlapping shape control, i.e. a Galerkin-type finite element discretization, the Hessian matrix would then be diagonal. Because introduction

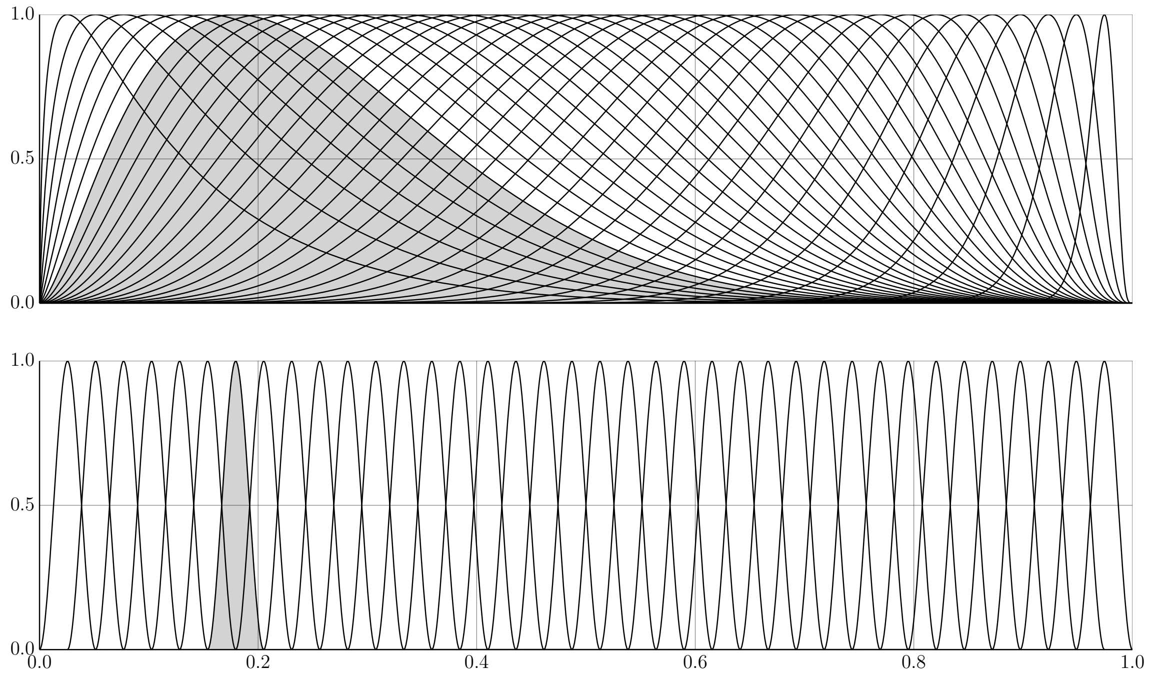
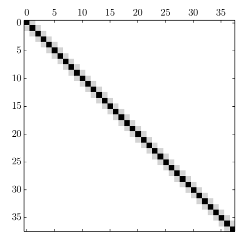


Figure 4.2: Local vs. global mode shapes. Plotted are shape modes for 18 evenly-spaced centers. One corresponding pair of modes is highlighted. *Top:* Hicks-Henne bump functions ($y = sin^4(\pi x^{\frac{log 0.5}{log x_j}})$). *Bottom:* Local cubic blending functions ($y = 2|x - x_j|^3 - 3|x - x_j|^2 + 1$).

of geometric discontinuities is almost universally undesired in aerodynamics, there is usually some amount of smooth overlap involved in interpolation. Thus, when the design variables are sorted in spatial order, the Hessian is a band matrix (see Figure 4.3a), with the width of the band determined by the interpolation stencil or radius of support. For fully overlapping modes, the Hessian matrix is dense (see Figure 4.3b).

This difference in Hessian structure has two implications for the present work. First, we often make the assumption that the diagonal of the Hessian encodes most of the information needed to make better predictions about which parameters have the most long-term design potential. Using highly overlapping modes means that the off-diagonal will also be significant. Second, even if the Hessian were computed perfectly accurately, with global modes there is also more redundancy among *candidate* parameters. In the approximate constructive refinement procedure developed in Section 3.4.2, the accuracy is degraded by high redundancy, unless the sliding window N_w is enlarged, which renders the adaptation procedure more costly.


(a) Local: cubic blending functions (all entries are zero except on 3 diagonals).

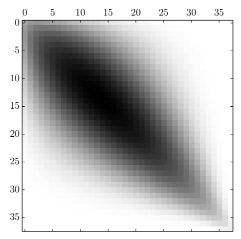

(b) Global: Hicks-Henne bump functions (all entries are non-zero).

Figure 4.3: Dependency of Hessian density on shape mode overlap. Global and local shape modes lead to denser and sparser Hessian matrices, respectively. This plot assumes that $\mathcal{H}$ is diagonal.

4.3.2 Neutral Shape Mutation

Ideally, the modeler should ensure that the shape is invariant with respect to the shape control, so that $\mathsf{S}^k_{final} \equiv \mathsf{S}^{k+1}_0$, i.e. a "neutral shape mutation" [86]. This is important firstly for efficiency. Any shape change that is not optimizer-driven is likely to hurt the performance of the design, thus incurring a temporary setback. In certain previous approaches to progressive parameterization with constructive modelers, the design variables in the refined parametric definition are approximately re-fit to at least minimize the unavoidable jump [77, 81]. In other approaches, a more efficient exact degree-elevation procedure is used [47, 50].

Neutral shape mutation is also essential for computing the refinement indicator. As discussed in Section 3.3, the efficiency of that process requires reuse of existing adjoint solutions to evaluate gradients with respect to hypothetical design variables. Any shape change invalidates the reuse of previous adjoints, as even subtle shape changes can lead to utterly different flow solutions and functional gradients. Thus for the indicator used here, neutral shape mutation should be considered mandatory.[2]

With the discrete geometry deformation approach used in this work, the shape can always be preserved exactly when refining the shape control. This is an intrinsic feature of discrete deformation; we can re-establish the previous optimal discrete surface $\mathbf{S}^k_{final}$ as the new baseline $\mathbf{S}^{k+1}_0$. For discrete geometry approaches, it is also important to consider the resolution of the surface tessellation, which

- Imposes a limit on the finest allowable level of shape control.
- Impacts the accuracy of the flow solutions.
- Affects the expense of computing surface deformations and projecting the gradients.

The simplest approach is to endow the baseline surface with enough resolution to support the entire sequence of optimizations. If it is necessary to modify the surface discretization during optimization, this must occur outside the adaptation procedure.

For the adjoint projection tool used in this study, the discrete surface must also have the same mesh connectivity. Mesh topology is intrinsically preserved by discrete deformation techniques. For constructive modelers that regenerate the discrete surface on demand, this is not necessarily guaranteed, even if the underlying shapes are geometrically identical.

4.4 Discrete Geometry Modeler

For all the evaluations and examples in Chapters 5 and 6, shape changes are made by deforming discrete surface triangulations. Shape manipulation is handled with a standalone modeler for discrete geometry, implemented as an extension to an open-source computer graphics suite called BLENDER. This platform was developed and validated in previous work [132, 142]. It allows BLENDER to serve as a geometry engine for shape optimization, providing on-demand deformations and computing analytic shape sensitivities. It has been used to support aerostructural analysis and design [154, 155] and adaptive parameterization [94, 98]. For this work I developed custom deformation plugins for this platform.

To deform curves (such as airfoils or other cross-sections), I use a "direct manipulation" approach, illustrated on the airfoils in Figure 4.4. The deformation of certain "pilot points" placed along the curve are directly manipulated and serve as the design variables. Deformation of the remainder of the curve is smoothly interpolated using radial basis functions

[2]If it were imperative to use a particular geometry modeler that cannot preserve the shape exactly, one might simply accept the errors due to using the adjoint solutions for a slightly different shape. With the regularity rules discussed in Section 3.4.4, the overall process would at least remain convergent.

(RBF). Derivations of the RBF deformation technique for aerodynamic design are given by several authors [130, 138–140]. Each parameter has a bump-shaped deformation mode that is mostly confined to the region between its neighboring points, while maintaining smoothness. I chose the basis function $\phi = r^3$ here, primarily because it requires no local tuning parameters, making it more amenable to automation, but other bases may have superior properties [138]. In the notation of Section 2.1, the shape control $\mathbf{C}$ is the parametric locations of the pilot points along the airfoil curve. Function $\mathcal{P}_{RBF}(\mathbf{C})$ "binds" these pilot points to the remainder of the curve, resulting in a deformation function $\mathcal{D}(\mathbf{X})$, which takes the pilot point deflections $\mathbf{X}$ and generates a new curve. For binary shape control refinement, adjacency and midpoints are defined in terms of arc-length along the curve.

To deform wings and similar components, I interpolate twist, sweep, scale (affecting thickness and chord) and airfoil deformations between spanwise stations, as illustrated in Figure 4.4. The twist and scale are taken about a user-defined, segmented axis, usually placed either at the leading or trailing edge, quarter-chord line, or elastic axis. At each station, a curve deformer modifies the airfoil shape, after which the twist, scale, and sweep

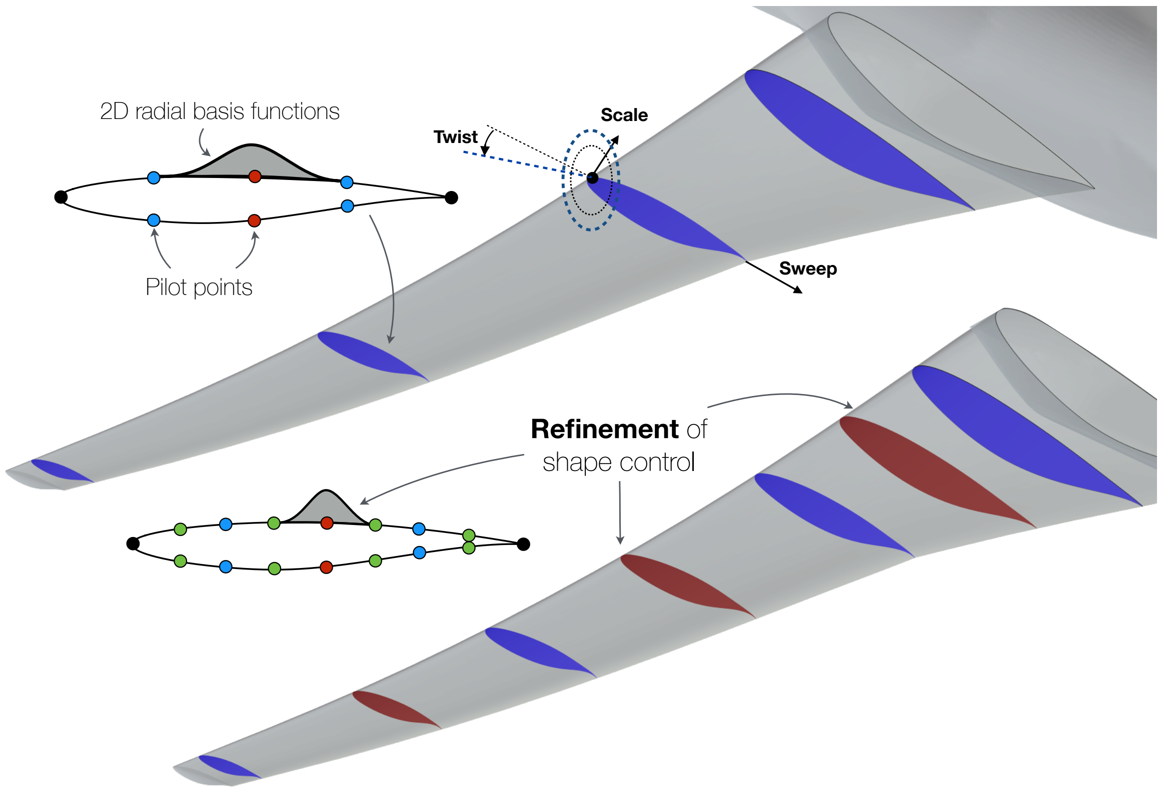


Figure 4.4: Wing deformation model

are applied (in that order).

Interpolation of each deformation class happens independently, allowing the shape control to be refined anisotropically. Similarly, each station can provide different resolution of control over the airfoil shape. To refine the shape control, more spanwise stations may be added, or new pilot points may be added at different stations.

Using the notation of Section 2.1, the shape control $\mathbf{C}$ is a vector containing

1. The spanwise location of each control station.
2. *For each control station:* the parametric location along the curve of each pilot point.

Function $\mathcal{P}_{wing}(\mathbf{C})$ generates the deformation function $\mathcal{D}(\mathbf{X})$, which takes all of the shape parameters $\mathbf{X}$ and generates a new wing shape. The shape parameters include

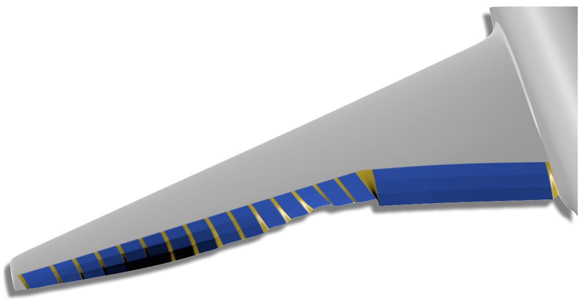
1. The twist, chord, and sweep at each station.
2. *For each control station:* the deformation of each pilot point.

The shape control is organized in multiple independent trees, corresponding to control of twist, sweep, and chord. Airfoil section control is organized as a "tree of trees". The top-level tree organizes the spanwise locations at which the airfoil can be controlled. At each of these locations, a subtree organizes the pilot points.

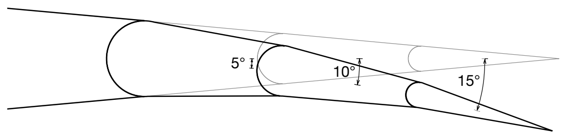
4.4.1 Wrapping an Existing Modeler

The previous two parameterization techniques were specifically developed to support adaptive parameterization. However, it is also possible to non-invasively wrap an existing modeler to support adaptive parameterization, e.g. via design variable linking. For modelers where the parameterization is specified in non-proprietary control files, this is simply a matter of creating a script-based interface to automatically modify these files.

As an example, consider the modeler in Figure 4.5a, which enacts trailing edge flap deflections on a transport wing. This system was developed by Rodriguez *et al.* to



(a) One possible flap system built using the modeler. Blue regions are flaps, gold are joining elastomer material.



(b) Each flap can be segmented, allowing more precise re-cambering.

Figure 4.5: Trailing edge flap system modeler *(figure from [155])*

assess the potential performance benefits of a variable-camber, continuous trailing edge flap system on a transport wing [155]. The modeler cuts a highly-flexible flap system out of a wing, allowing the flap deflections to be optimized for different flight conditions. The modeler supports arbitrary spanwise spacing of flaps and also allows each flap to be segmented, as shown in Figure 4.5b. Flap systems with fewer moving parts can be modeled by linking the flap deflections both spanwise and streamwise. I leverage this modeler in a later design example (Section 6.4). In that example, I start with a large set of flaps provided by the modeler. I then automatically link all of the flap segments into two monolithic flaps. Thereafter, flap refinement is performed by adaptively *unlinking* the parameters, which exposes more degrees of freedom and allows modeling of more complex flap systems. In this case, no modification to the underlying modeler was required.

CHAPTER 5

EVALUATION AND VERIFICATION

This section validates the adaptive shape control system developed in Chapter 3. Three inverse design examples of increasing complexity are considered. For each case, the continuous target is attainable only with continuous shape control. The system is initially given a naive coarse parameterization and must automatically determine parameters that are sufficient to solve each problem. In the process, the refinement indicator and the adaptation strategy are verified and evaluated in isolation. I also evaluate robustness with respect to initial conditions and the refinement strategy. After following different trajectories through the design space, all paths consistently coverge to the same fine-space optimal design.

5.1 Geometric Shape Matching

This first verification problem involves 3D geometric shape matching, where the optimizer drives a baseline shape to a target shape, using a given basis of shape control. The objective function aims to minimize the point-wise geometric deviation from the target shape $\mathbf{S}^*$ in a least-squares sense:

$$\mathcal{J}^{GSM} = \frac{1}{2} \sum_{i=1}^{N_{verts}} \|\mathbf{v}_i - \mathbf{v}_i^*\|^2 \tag{5.1}$$

where $\mathbf{v}_i$ are the current vertex coordinates on the discrete surface and $\mathbf{v}_i^*$ are the corresponding target vertex coordinates. The purpose of this verification study is to evaluate the adaptive procedure developed in Chapter 3 by answering two fundamental questions:

1. (Section 5.1.1) *Are the indicator values correlated with actual design improvement?*
2. (Section 5.1.2) *Can the system discover a compact set of parameters that enable the shape to be matched?*

Initial and Target Shapes

Figure 5.1 shows the the baseline and target shapes. The baseline is a straight wing with no twist, taper or sweep, and is represented as a discrete geometry with about 197K vertices. The target geometry is a typical transport wing with substantial twist, chord-length and sweep profiles, as shown in Figure 5.1. For this academic example, the target sweep profile is linear and the target chord-length profile is piecewise linear in two segments, while the twist profile is quadratic. Both wings have the same airfoil section, so this problem considers only planform design.

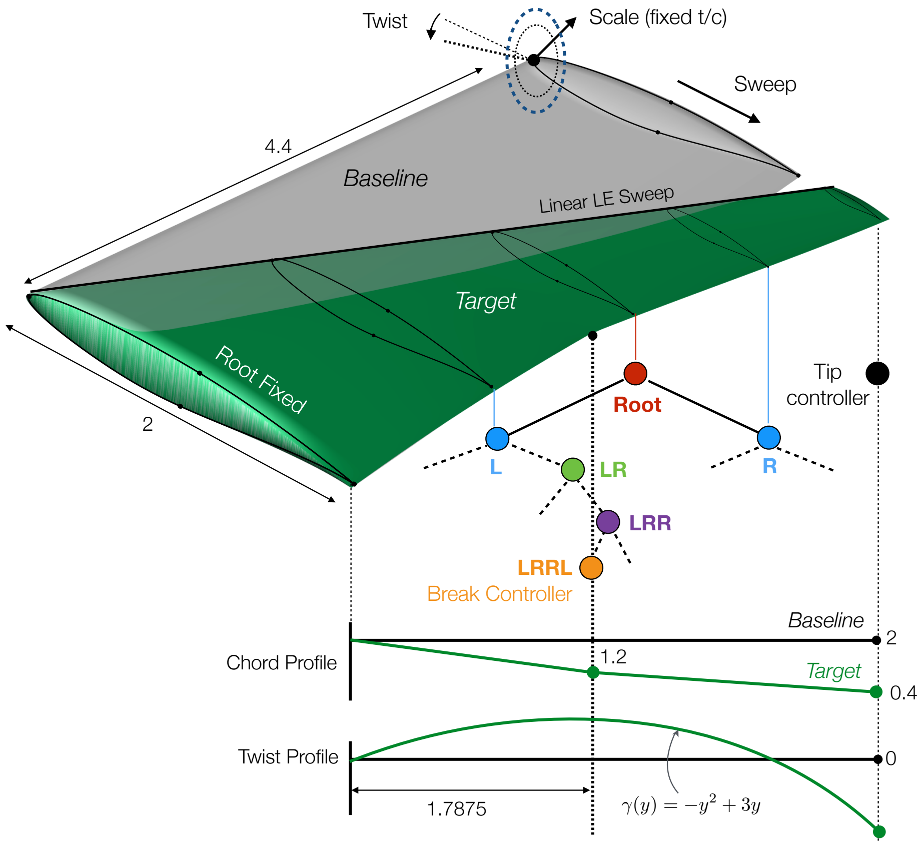


Figure 5.1: ***Shape Matching:*** Baseline and target planform profiles. Shape control is organized in spanwise tree structure. At a depth of $d = 4$, a controller can be added at the wing break.

Parameterization

The wing planform is parameterized using the technique described in Section 4.4, which linearly interpolates twist, sweep and chord between spanwise stations. Control stations can be placed anywhere along the wing, except at the root, which is fixed. The adaptive system must determine a compact set of locations for the twist, sweep and chord control stations that enable exact recovery of the target shape. This problem is very challenging for adaptive parameterization, because three different classes of shape control must be refined simultaneously. Each has different units and scaling, and there is redundancy both within each class (due to mode overlap) and among the classes (twist, sweep and scale are not geometrically orthogonal). To my knowledge this is the first example in the literature to consider simultaneous adaptation of different classes of shape control.

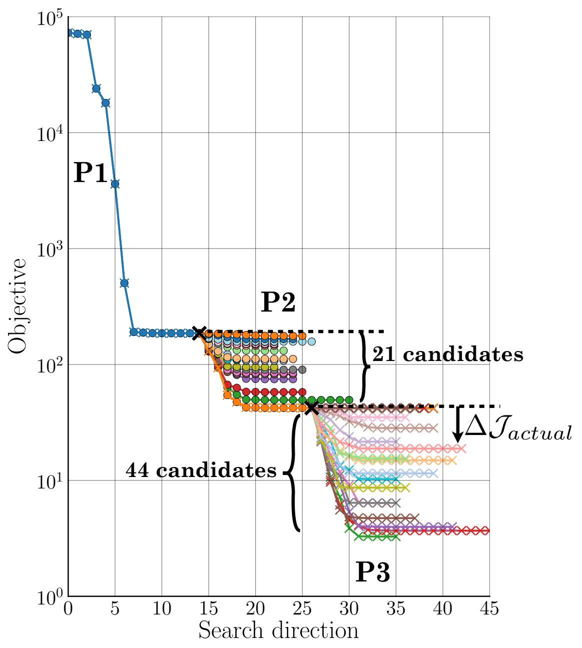


Figure 5.2: ***Shape Matching:*** *Indicator verification:* **P1:** Objective convergence under initial parameterization. **P2** and **P3:** Subsequent optimizations corresponding to addition of one of the candidates, starting from the previous best design.

Initially the optimizer is given three design variables: twist, chord and sweep at the tip station. The shape is initially optimized to convergence in this 3-DV search space, as shown by the blue curve in Figure 5.2, exhausting the potential of this search space. Further improvement is possible only after the system adds more spanwise stations through binary refinement of each class of shape control, as shown in Figure 5.1.

5.1.1 Indicator Verification

This section evaluates the ability of three versions of the refinement indicator from Section 3.2.2 to predict the actual performance of the various candidate shape parameters:

- I_H — The second-order indicator, which uses the nearly-exact Hessian[1] (Equation (3.21))

[1]The Hessian is computed analytically, but the nonlinear term **D** in Appendix B, Equation (B.3) is neglected. Here, the twist deformation modes are nonlinear with respect to the angle, although because the angles are small, the resulting error is small.

- I_D — Using only the diagonal of the Hessian
- I_G — The first-order indicator based on $\Delta\mathcal{J}_\star^{\mathcal{H}}$, which uses only gradients (Equation (3.17))

Using a search depth of $d = 3$, Function 3 (in Appendix A) returns 21 candidate parameters (seven each for twist, sweep and chord). The goal in adaptive refinement is to pick the subset of these parameters that enable the objective to converge to the lowest final value, thus maximizing $\Delta\mathcal{J}_{actual}$. For each candidate the system estimates the potential design improvement using I_H, I_D and I_G. Then 21 full optimizations are run, where only one of the candidates is added to the active set.

Figure 5.2 shows the objective convergence corresponding to addition of each one of the candidates. The top-ranked candidate was the chord design variable at station "LR" (green parameter in Figure 5.1) which was as close to the essential break station as the search depth allowed. I then added this parameter, and repeated this study once more, with a new set of candidates. For the second pass, there were 44 candidate shape parameters — 21 on each side of the newly added station, plus sweep and twist on the station where chord control was added. As before, a full optimization was run for each candidate (see Figure 5.2).

Figure 5.3 correlates the predicted potential for design improvement $\Delta\mathcal{J}_\star$ with $\Delta\mathcal{J}_{actual}$. Each data-point corresponds to a candidate design variable. In each plot, the parameters at the top right are the most effective ones. The adaptive system would prioritize them over the relatively ineffective parameters to the bottom left. The top row in Figure 5.3 corresponds to I_H and demonstrates nearly perfect predictions. The middle row corresponds to the first-order predictions (I_G). On the first pass, the ranking is reasonable, but on the second pass it is extremely poor, with the highest-ranked parameters performing the worst in reality. The bottom row corresponds to using only the Hessian diagonal (I_D). In this case the correlation is still quite reasonable, showing that accurate diagonal scaling may be sufficient to achieve good predictions of importance.

This is a striking validation of the refinement indicator, and also a display of the importance of second-order information. Previous studies have suggested using gradient information to determine the relative importance of different candidate parameters [50, 81]. This study demonstrates that unless the problem is well-scaled, examining only first-order information can sometimes lead to very poor predictions. On some later examples, this conclusion will be refined somewhat. We will see that I_G can provide reasonably good rankings when all parameters are of the same class. The present example involves simultaneous adaptation of three fundamentally distinct types of deformation, each having very different second-order effects, visible in the Hessian structure in Figure 5.4. Ignoring those effects

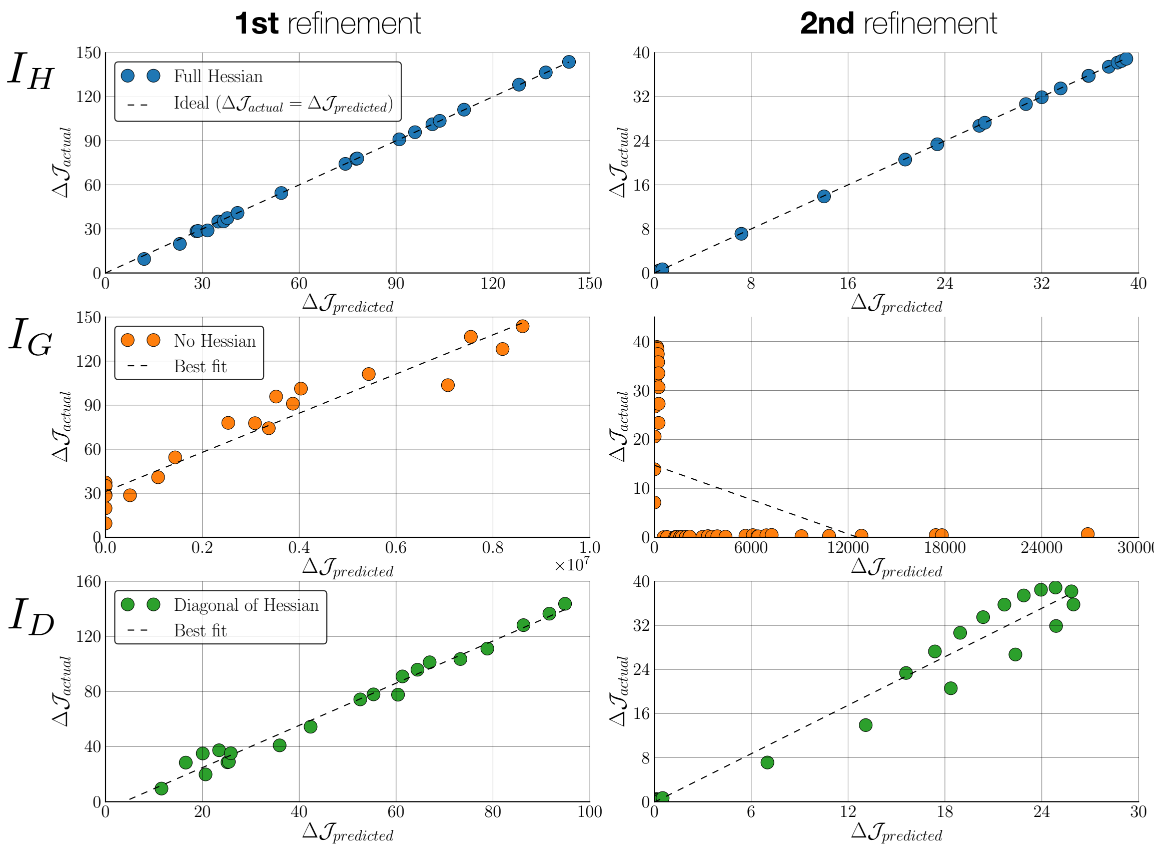


Figure 5.3: ***Shape Matching:*** *Indicator verification:* Correlation between predicted and actual design improvement for the first (*left*) and second (*right*) refinements. *Top row*: I_H, *Middle row*: I_G, *Bottom row*: I_D

might be safe *within* a given class of parameters, but not across multiple shape control trees. The Hessian naturally accounts for this scaling and corrects the predictions.

5.1.2 Evaluation of Adaptation Strategy

I now evaluate the adaptation strategy, which has major impacts on robustness and efficiency. The problem is constructed such that the parameterization that allows the closest recovery of the target with the fewest design variables is known in advance:

1. **Chord:** To recover the piecewise linear chord profile requires exactly one chord controller at the break ($\frac{13}{32}$ span).

2. **Twist:** To approximate the quadratic twist profile with piecewise linear segments, progressively finer twist control should be added. It should be evenly spaced to

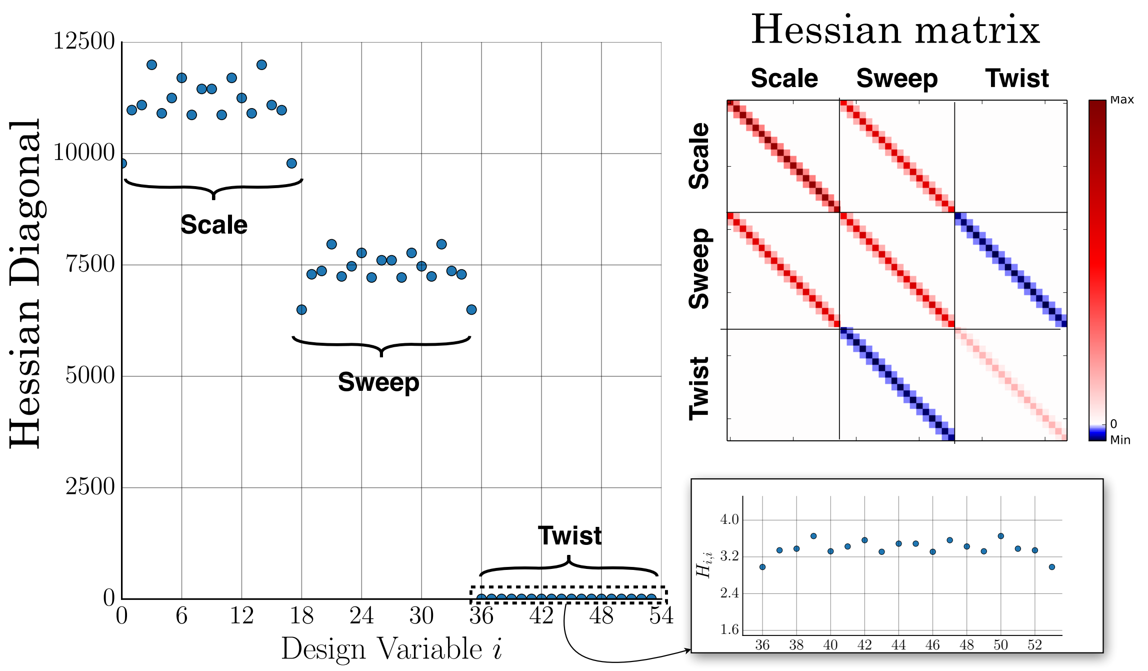


Figure 5.4: ***Shape Matching:*** Hessian structure. *Left:* Diagonal entries vary by orders of magnitude, depending on the shape control class. (Variation within each class is due to non-uniform surface discretization — $\mathcal{J}$ weights each vertex equally.) *Right:* Within each class block, $\overline{\mathbf{H}}$ is tridiagonal, due to the nearest-neighbor interpolation. Redundancy across classes is visible in the off-diagonal blocks, except for between twist and scale, which are perfectly geometrically orthogonal.

optimally clamp down the error.

3. **Sweep:** The initial controller at the tip is sufficient to recover the linear sweep distribution, so no additional control is needed.

The adaptive system is not expected to find *exactly* this ideal parameterization. However, it should use substantially fewer degrees of freedom than a uniform, isotropically refined parameterization with equivalent potential. The evaluation is split into three tests, each addressed to a specific question:

1. (Section 5.1.2) *Can the indicators identify the parameters necessary to solve the problem?*

2. (Section 5.1.2) *What impacts the efficiency of the adaptation strategy?*

3. (Section 5.1.2) *Under what situations could the refinement strategy perform poorly?*

Test 1: Identifying Important Parameters

This section performs a head-to-head benchmark of using I_G vs. I_H as the indicator, to compare their ability to discover the necessary parameters. The adaptation routine searches only one level deep ($d = 1$) and adds only one new design parameter at a time ($N_{add} = 1$). This deliberately restrictive strategy allows evaluation of the indicators independently of the approximations made by the constructive refinement procedure, which might otherwise confound the conclusions.

Distinct refinement patterns evolve under the two indicators, as shown in Figure 5.5. The right frame shows the pattern produced after 22 cycles of adaptation based on I_H. The resulting parameterization is almost identical to the ideal parameterization, which was known a priori. Sweep control was correctly ignored, chord control was correctly added at the break, and twist control was evenly spaced along the span. (The next nested level of twist control has begun to be added near the root.) Only four extraneous chord design variables were added, and even these were not mistakes. Under the binary refinement rules stipulated in Section 2.3.1, the necessary station at $\frac{13}{32}$ span was not considered a candidate until higher levels in the tree were first added (see Figure 5.1). The adaptation procedure did precisely this, and ultimately correctly identified and added the necessary chord parameter at the break.

The left half of Figure 5.5 shows the results after 25 cycles of adaptation based on I_G. The refinement pattern reveals that the procedure was quite inaccurate and failed to efficiently

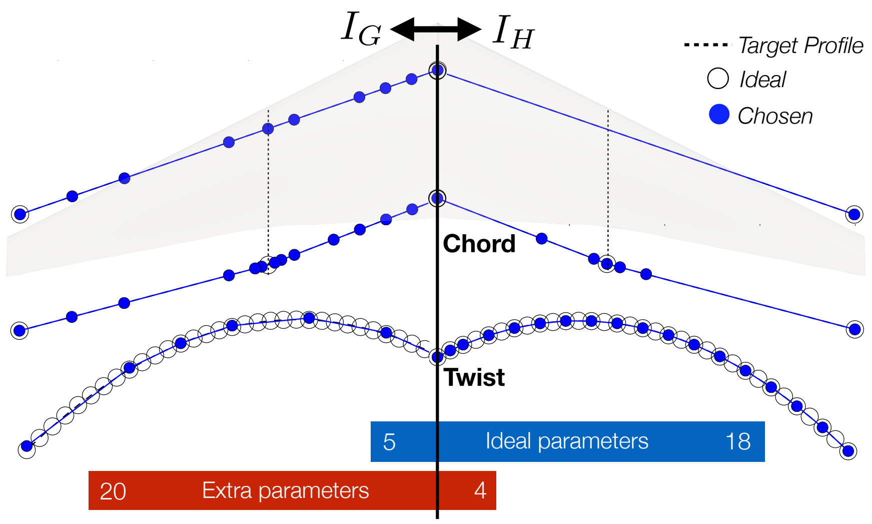


Figure 5.5: ***Shape Matching:*** *Test 1:* Final adapted parameterizations using I_G *(left)* vs. I_H *(right)*. I_H recovers the expected parameters with few extras, while I_G mostly adds extraneous parameters.

capture the important design variables. Although the shape has been matched reasonably, it is inferior to the match resulting from I_H, especially in the twist profile. As before, the reason for the relatively poor performance is related to the highly distorted scaling of the search space. This is visible in the Hessian diagonal, which is much larger for scale and sweep than for twist (Figure 5.4). Thus even when very close to the optimum, I_G strongly favored the chord and sweep variables because their gradients were much larger, even though they offered only extremely short-term potential. I_H, by contrast, revealed that they in fact had low long-term potential. This again underscores the essential role of second derivative scaling information for adaptation.

The objective convergence achieved using the two indicators is shown in Figure 5.6, labeled "I_G (add 1)" and "I_H (add 1)". I_G frequently selected parameters with almost zero potential, leading optimization to stall for several adaptation cycles. Nevertheless, it still managed to eventually reduce the objective by over 6 orders of magnitude, indicating quite

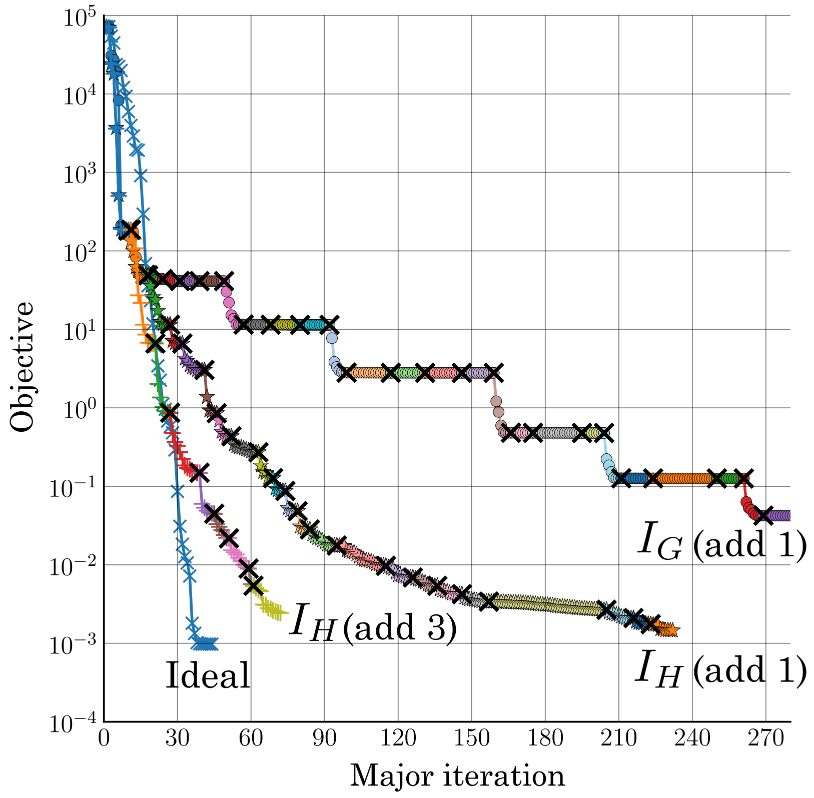


Figure 5.6: ***Shape Matching:*** Objective convergence with different indicators and adaptation strategies. Each color represents a suboptimization, and $\times$-marks denote search space refinements. The curve labeled "Ideal" represents the most efficient possible 35-DV parameterization.

close recovery of the target shape. This is important; it indicates that even with a quite inaccurate indicator, the adaptive system is still able to approach the continuous optimum. I_H, however, makes faster progress, and reaches a superior design.

Test 2: Refinement Procedure

The previous test compared two indicators, while removing the impact of the approximate refinement procedure. Compared to the "ideal" parameterization, however, performance is still relatively slow (Figure 5.6). It took many adaptation cycles to drive towards the target shape, because of the excessively slow growth rate. This second test accelerates the growth rate and search depth slightly ($d = 2$, $N_{add} = 3$, using I_H), which should lead to much faster design improvement. This test is intended to evaluate the cost and accuracy of the approximate constructive adaptation procedure develop in Section 3.4.2.

After several alternating optimizations and refinements, the process converged to a shape match equally as good as in Test 1 (see Figure 5.6, "I_H (add 3)"). Unsurprisingly, the number of iterations required is about one third of the number required for the previous case ("I_H (add 1)"). More interestingly, for the first several levels, it outperforms even the minimal parameterization, indicating that progressive parameterization can accelerate optimization even if the minimal parameterization is known in advance! The system has now added 10 non-ideal parameters, compared to four when adding only one at a time. This is still substantially better than isotropic uniform refinement, which would have added about 30 non-ideal parameters.

High Redundancy

Given the success of the previous two tests, we might wonder whether the system could immediately determine *all* of the necessary shape control upfront. To test this I use I_H and set the refinement parameters to $d = 5$ and $N_{add} = 32$, which is sufficient to allow the system to discover the minimal parameterization. With $d = 5$, there are a total of 93 candidates. An exhaustive search would involve evaluating the $\sim 8 \times 10^{24}$ possible combinations of parameters. The constructive search procedure seeks an approximate solution with only $\mathcal{O}(N_{DV})$ indicator evaluations.

The first phase of Function 5 computes an initial ranking of the 96 candidates, which is shown in Figure 5.7. The y-axis gives the estimate of the additional design potential that would be added to the search space by adding *only* the corresponding parameter. The minimal ensemble of 32 parameters is highlighted in green. I_H correctly identified the chord station at the break as the most important shape controller to add. Initially, it appears

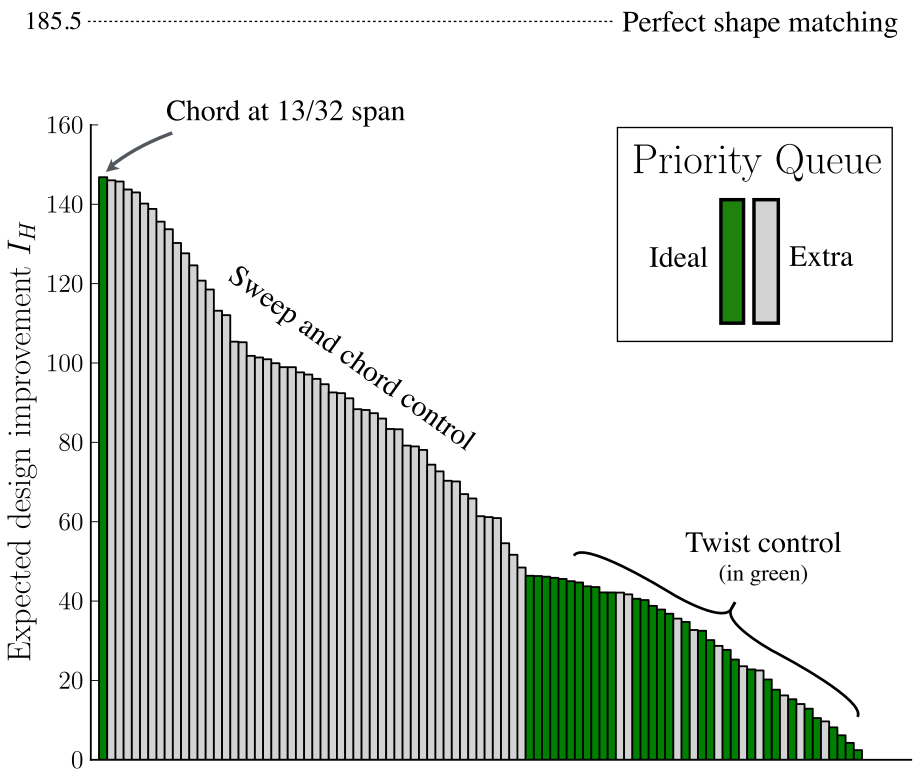


Figure 5.7: ***Shape Matching:*** *Test 3:* Initial priority queue ($d = 5$, $N_{add} = 32$, I_H). The 32 parameters that would best recover the target shape (*green*) are highlighted. After adding the top member of the queue, all of the subsequent chord and sweep controllers (*gray*) become redundant.

that the twist variables are the least important in the queue. However, after adding the break-chord parameter, the next 50 controllers all become highly ineffective. Their initial appraisal was based on the absence of the break-chord parameter; they could each have recovered much of the *same* design potential that it offered. The twist stations at the end of the priority queue offer relatively little potential, but that potential is independent of the chord control, and thus they remain useful.

The constructive procedure remains functional on this problem, but it is slow. Function 5 must work its way through all of the other chord and sweep variables until finally discovering the more important twist control. While alternate search strategies might be developed to handle this situation, from a practical standpoint the easiest recourse is to simply limit the depth of the search to $d = 1$ or $d = 2$, which eliminates most of the redundancy and yields excellent performance.

5.2 Subsonic Inverse Airfoil Design

This study examines inverse airfoil design. The goal is to deform the airfoil such that its pressure profile matches a target. The objective function is a quadratic penalization of deviations from the target profile:

$$\mathcal{J} = \frac{1}{2} \sum_{i=1}^{N_{verts}} (p_i - p_i^*)^2 \tag{5.2}$$

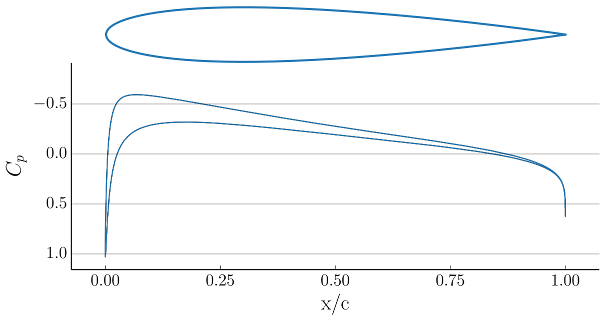


Figure 5.8: ***Inverse Airfoil Design:*** Target airfoil (NACA 0012) and pressure profile. *Mach 0.3,* $\alpha = 1°$

where p_i^* is the target pressure at vertex i on the discrete curve. The target pressure profile (shown in Figure 5.8) is that of a NACA 0012 airfoil at $M_\infty = 0.3$ and $\alpha = 1°$. While the target airfoil is symmetric, the initial shapes are asymmetrically randomly generated, and the shape parameterization is permitted to be refined asymmetrically. As before, the target profile is attainable, but not under the initial parameterization. This example is split into two parts, whose primary goals are to assess whether

1. The refinement indicator is effective for aerodynamic functionals.

2. The adaptation system is *robust*, in the sense that it converges to the same optimal design independent of both the initial shape and the refinement strategy.

5.2.1 Indicator Verification

The first study parallels the indicator evaluation study from the shape-matching study in Section 5.1.1. It is important to repeat that study, because this case introduces an aerodynamic functional. Starting with an evenly-spaced, 14-DV parameterization, the initial shape is optimized to convergence, as shown by the blue curve in Figure 5.9. The design improvement potential of this initial search space has been fully exploited, but further improvement is possible when more degrees of freedom are added.

This study compares the predictive accuracy of two indicators, I_G and I_D (Equations (3.17) and (3.23)). To compute the diagonal of the Hessian for I_D, we take the BFGS Hessian approximation from the initial search space and prolong its diagonal into each candidate search space via Equation (3.25). Using a search depth of $d = 1$, Function 3 returns 16 candidate parameters (eight each on the top and bottom surfaces).

For each candidate, the system estimates the potential design improvement using I_D and I_G. Then 16 independent optimizations are run, where only one of the candidates is added to the active set. Figure 5.9 shows the objective convergence corresponding to addition of each one of the candidates. The predicted design improvements are then correlated with the actual observed design improvements, as shown in Figure 5.10. Using approximate Hessian information strongly improves the absolute predictions, with the results being closer to "ideal" ($\Delta\mathcal{J}_\star = \Delta\mathcal{J}_{actual}$). However, note that the two indicators provided nearly identical *relative* rankings, which is more important, as it dictates which parameters to add. In contrast to Figure 5.3, here I_G proved to be reasonably effective. This is largely because in the current design example, there is only *one* class of shape control and thus less extreme

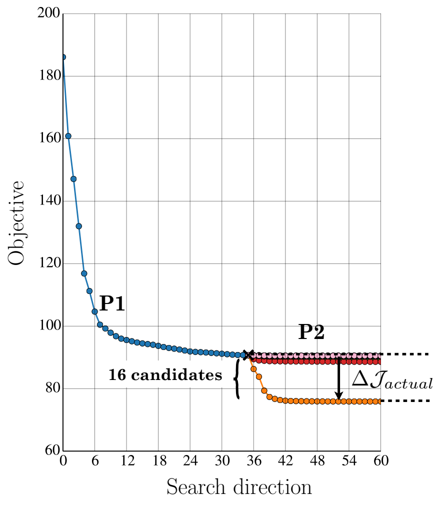


Figure 5.9: ***Inverse Airfoil Design:*** *Indicator verification:* Objective convergence under initial parameterization, followed by subsequent optimizations corresponding to addition of one of the candidates.

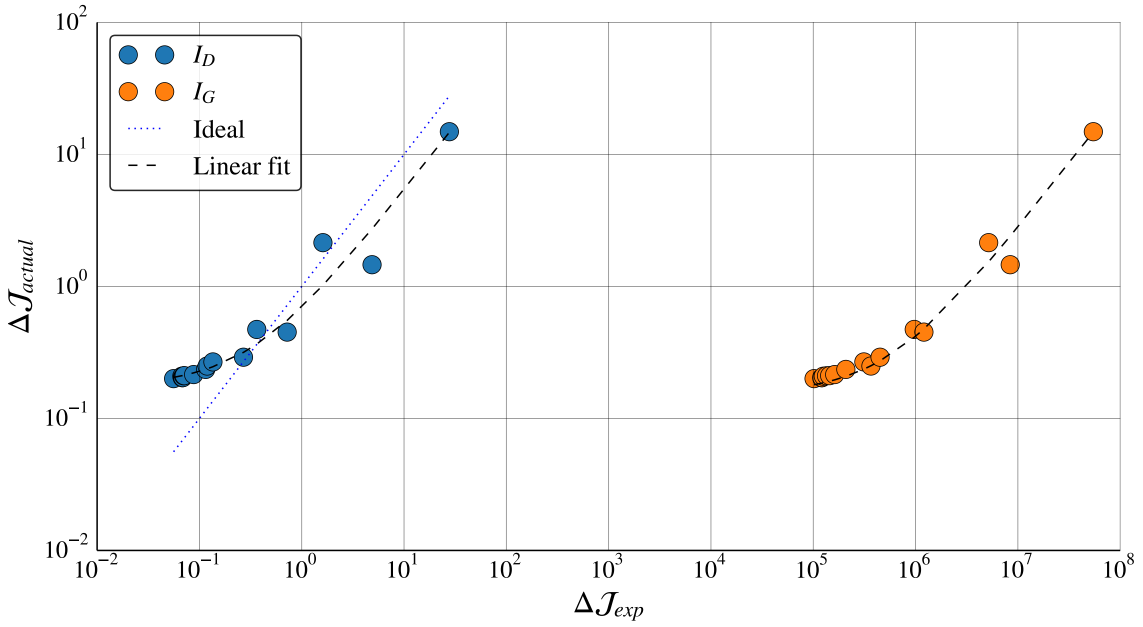


Figure 5.10: ***Inverse Airfoil Design:*** Evaluation of predictions made by indicators I_G and I_D (with transferred quasi-Newton Hessian approximation)

variation in design space curvature.

Figure 5.10 shows that the rankings are non-monotone, which indicates that they are imperfect. Based on the results from the shape-matching example in Section 5.1.1, it is likely that the ranking could be improved further with off-diagonal information. However, we would *not* expect the rankings to ever be ideal. After all, a perfectly accurate second-order indicator is still just a local expansion of the full nonlinear behavior. Nevertheless, the rankings are encouragingly good for both I_G and I_D, which motivates the application of this method to the more complex upcoming design examples.

5.2.2 Robustness to Inputs

The purpose of this section is to assess robustness with respect to (1) the initial shape and (2) the refinement strategy. I perform a suite of 15 adaptive optimizations: I test five different refinement strategies, starting from each of three initial shapes. The three starting shapes are shown in Figure 5.11. They were generated by smoothly deforming the target NACA 0012 airfoil using 15th-order Bernstein polynomials with random weights, multiplied by a Kulfan-style class function $c(x) = \sqrt{x}(1-x)$ that maintains the circular leading edge and sharp trailing edge [108]. From each starting point several refinement strategies are compared:

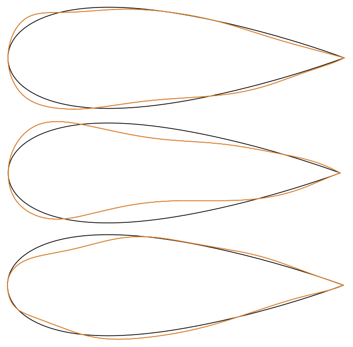


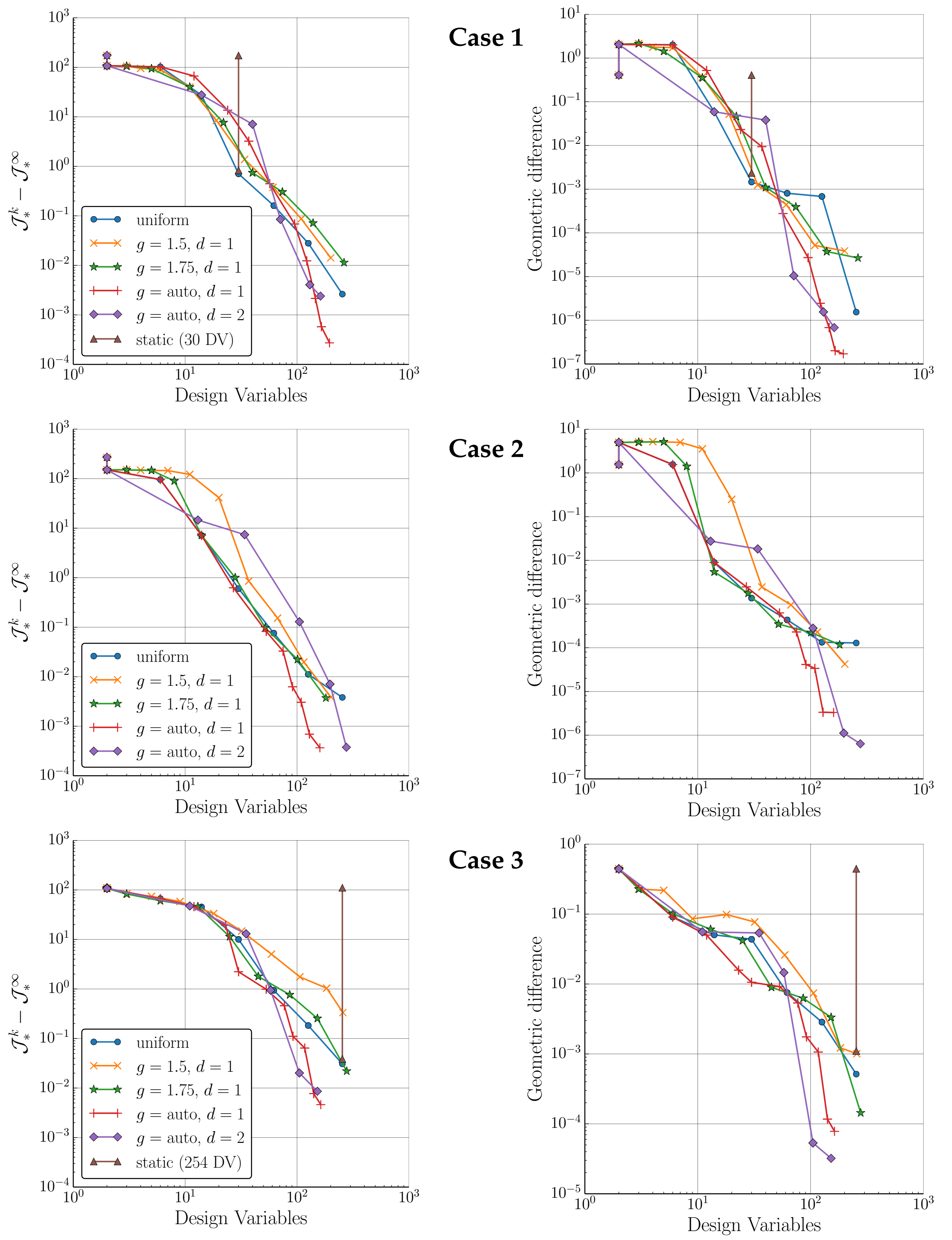
Figure 5.11: ***Inverse Airfoil Design:*** *Robustness Study:* Randomly-generated initial geometries *(1-3 from top to bottom)*

- Uniform refinement (growth rate $g = 2\times$)
- Fixed growth rates, $g = 1.5\times$ and $g = 2\times$ (both with search depth $d = 1$)
- Automatic growth rates ($r = 0.01$, $w = 4$) at two search depths, $d = 1$ and $d = 2$.

(Explanations of these settings were given in Sections 2.3.1 and 3.4.3.)

While other examples will consider wall-clock time efficiency, this study focuses more on the *consistency* of the final results and also on the asymptotic rate of convergence with respect to the number of design variables. Therefore, each suboptimization level is converged deeply.[2] Each case was initialized with a minimal 2-DV parameterization to eliminate as much initial bias as possible. In production settings, intermediate shape control levels would not be converged so deeply, and we would start with more design variables.

[2]The norm of the gradients was required to decrease by 2.5 orders of magnitude.



(a) Objective convergence

(b) Convergence of geometry to target

Figure 5.12: ***Inverse Airfoil Design:*** Convergence of the objective with respect to shape control refinement. Each curve indicates a different refinement strategy. Each point indicates the best (final) objective value attained within that level.

Table 5.1: ***Inverse Airfoil Design:*** Asymptotic convergence rate of $\mathcal{J}_\star^k - \mathcal{J}_\star^\infty$ with respect to the number of design variables, for different refinement strategies. *(Rate averaged over last four levels.)*

case	uniform	$g = 1.5$ $d = 1$	$g = 1.75$ $d = 1$	$g =$auto $d = 1$	$g =$auto $d = 2$
1	2.6	2.6	2.2	8.3	5.0
2	2.4	3.2	3.0	5.2	5.6
3	2.7	2.1	2.3	5.7	4.7
mean	**2.6**	**2.6**	**2.5**	**6.4**	**5.1**

Optimization Results

Figure 5.12, left column, plots the asymptotic convergence of $\mathcal{J}$ to $\mathcal{J}_\star^\infty = 0$, the continuous optimal value. (Recall the discussion of this type of plot in Section 2.1.2.) Each frame corresponds to a different starting point, and each line corresponds to a different adaptation strategy. To establish a point of reference, convergence is also shown for some evenly-spaced, static parameterizations.

All of the convergence curves are initially almost flat. This indicates that the coarse parameterizations are not yet in the region of asymptotic convergence. As adaptation continues, all refinement strategies begin to converge asymptotically to the target pressure profile. The right column of Figure 5.12 verifies that the shape itself is also converging to the NACA 0012 airfoil, not to some hypothetical different shape with an identical pressure profile. Such robust convergence to the continuous optimum is due to two factors: (1) generally accurate predictions of the most important parameters to add, and (2) the imposition of shape control regularity rules (see Section 3.4.4), which prevent the process from stalling in cases where the predictions are less accurate.

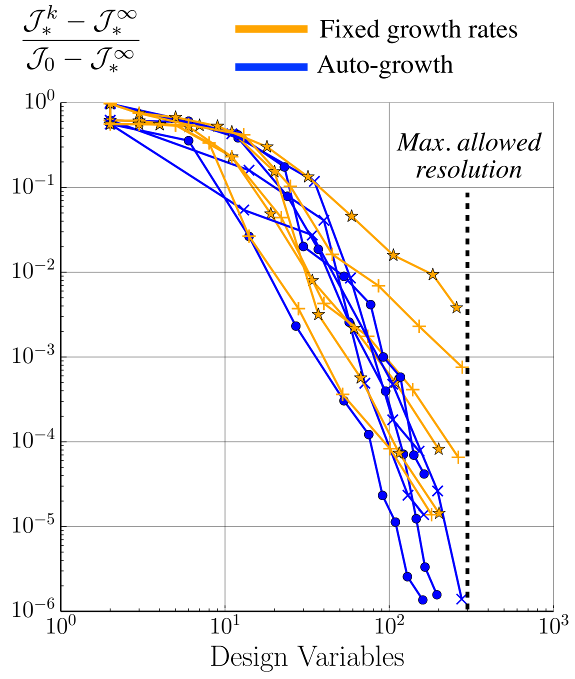


Figure 5.13: ***Inverse Airfoil Design:*** Convergence to continuous optimum using both fixed growth rates (*orange*) and automatically determined growth rates (*blue*). Objective values are normalized to begin at $\mathcal{J}_0 = 1$, to remove the discrepancy in $\mathcal{J}_0$ for the different starting points.

Different strategies take different routes to the optimum, and have different convergence rates (Table 5.1). On average, all adaptive strategies eventually converge at least as fast as uniform refinement, and usually much faster. This is the expected result, and supports the validity of the refinement indicator and the efficiency of the refinement strategy. Because the adaptation is approximate, it naturally makes imperfect predictions. This is evident in the variation in the rates of convergence of different strategies. (If the indicator were a perfect prediction, then the rate of convergence with respect to be N_{DV} should be independent of the strategy.) Dividing the adaptive strategies into fixed growth rates and automatic growth rates reveals an interesting trend. Figure 5.13 compares convergence under the two approaches. While the method for automatically setting growth rates was initially intended to simplify the process, it appears to also consistently improve the asymptotic convergence.

It is not surprising that adaptive refinement achieves substantially faster asymptotic convergence with respect to N_{DV}. However, efficient use of design variables does not directly imply a reduction in wall-clock time. (Figure 5.12 conceals the cost of each suboptimization.) Devising refinement strategies that are more computationally efficient than uniform refinement is examined in later examples. However, even here we can observe computational acceleration. Figure 5.14 shows convergence of the objective with respect to major iteration for three strategies. When given a more aggressive trigger ($r = 0.05$), uniform refinement outperforms a fine static parameterization ($d = 6$, i.e. 256 DVS) for most of the design trajectory.

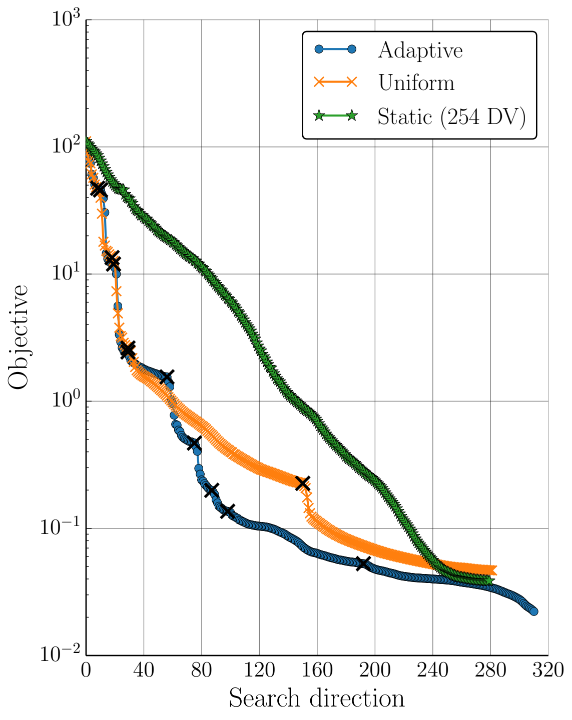
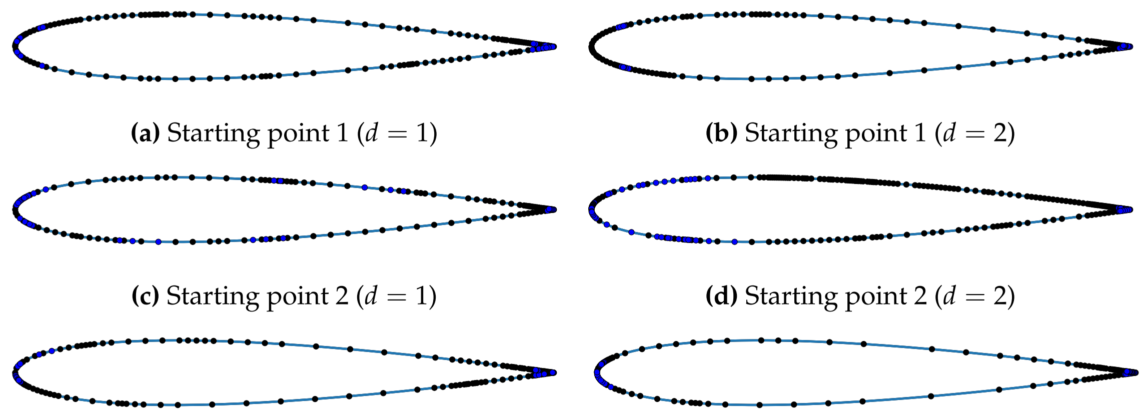


Figure 5.14: ***Inverse Airfoil Design:*** Objective convergence with respect to design iteration. When given an aggressive trigger, progressive schemes tend to exhibit faster convergence than a static parameterization. Adaptive refinement substantially outperforms uniform refinement.

The adaptive approach is faster yet, and also achieves a superior design with 90 fewer design variables (166 total). It does so by radically varying the density of shape control across the surface. The local refinement depth ranges from as coarse as $d = 2$ in some regions to as fine as $d = 10$ in others. Specifically, it clustered the shape control at



(a) Starting point 1 ($d = 1$)
(b) Starting point 1 ($d = 2$)
(c) Starting point 2 ($d = 1$)
(d) Starting point 2 ($d = 2$)
(e) Starting point 3 ($d = 1$)
(f) Starting point 3 ($d = 2$)

Figure 5.15: ***Inverse Airfoil Design:*** Final adapted parameterizations for each starting point, using two auto-growth strategies.

the leading and trailing edges, as shown in Figure 5.15. This is a common technique used by aerodynamic designers, because of the high sensitivity of flow profiles to those regions — here it was discovered automatically. From each starting point, the system used somewhat different parameter distributions to recover the target profile. For example, in the middle region of the airfoil, starting point 2 used much denser control than starting point 3. Each parameterization was generated in a path-dependent process; the discrepancies are partly due to that. This example demonstrates that even slightly different problems can require different parameterizations. Problems that are more starkly different would be expected to display even more divergent and less predictable refinement patterns.

5.3 Supersonic Off-body Pressure Signature Matching

This third verification example involves reshaping a supersonic body to match a target off-body pressure signal. This problem undergirds one effective approach to low-boom design [156, 157]. As with the previous examples, the goal of this study is to evaluate whether the adaptive parameterization system can approach the continuously optimal design. This example introduces a more complex and realistic aerodynamic functional involving a 3D flowfield and an off-body pressure signal.

Inverse Design Approach

The objective function seeks to minimize the discrepancy along a sensor between an off-body pressure signal and a target signal:

$$\mathcal{J} = \frac{1}{2p_\infty^2} \int (p - p^*)^2 dS \tag{5.3}$$

where p^* is the target signal and dS indicates integration along the length of the sensor. This discrepancy is depicted by the shaded region in Figure 5.16. There are no constraints except for design variable bounds to maintain physical, positive-radius geometries throughout design. These are mostly inactive, including at the optimum.

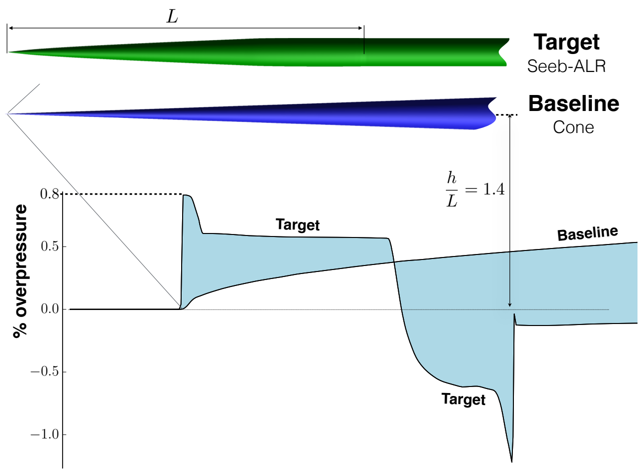


Figure 5.16: ***Supersonic Forebody:*** Baseline and target geometries and corresponding off-body pressure signatures *(off-body distance not to scale)*. Shaded region indicates the signal deviation to be driven to zero by shape optimization.

Figure 5.16 shows the target geometry, a thin body of revolution based on work by Darden [158] and George and Seebass [159] that has been used in numerous sonic boom studies [160, 161]. Here I use a modified version with more aft lift relaxation that was studied in the First AIAA Sonic Boom Workshop [161]. The Mach number is 1.6 and the angle of attack is 0°. Figure 5.16 shows the on-track target pressure signal at 21.2 model units directly below the forebody axis. The signal is clipped on either end so that it contains only the active portion generated by the forebody. This clipped signal serves as the target for inverse design.

Meshing and Flow Solutions

At each design iteration, the mesh is adapted to control discretization error in the objective function. An example is shown in Figure 5.17, which shows the mesh adapted to accurately compute the target pressure signature. The approach to meshing used for this problem is based on [101], which considered this problem from the point of view of progressive adaptive meshing. The mesh adaptation functional is subtly different from the objective:

$$\mathcal{F}_{adapt} = \frac{1}{2p_\infty^2} \int (p - p_\infty)^2 dS \tag{5.4}$$

This functional targets discrepancies between the signature and the *free-stream* pressure, which has two advantages. First, unlike Equation (5.3), Equation (5.4) avoids quadratically tending to zero as the signal approaches the target. This circumvents numerical problems with mesh refinement accuracy near the optimum. Second, it focuses mesh adaptation on the strongest regions of the signal, which most impact the loudness in decibels after the boom has coalesced and propagated to the ground. For optimization, the meshes were adapted over 10 levels. To resolve the strongly varying flow fields encountered during optimization, the actual cell counts ranged from 7 million to 16 million.

Geometry and Parameterization

The baseline geometry is a cone. The parameterization allows the radius to be specified at various control stations, with linear lofting in between. The initial parameterization (shown in black in Figure 5.18) consists of three radius stations evenly distributed along the region of the forebody that can affect the pressure along the sensor. Subsequent refinements of the shape control follow the binary structure illustrated in Figure 5.18.

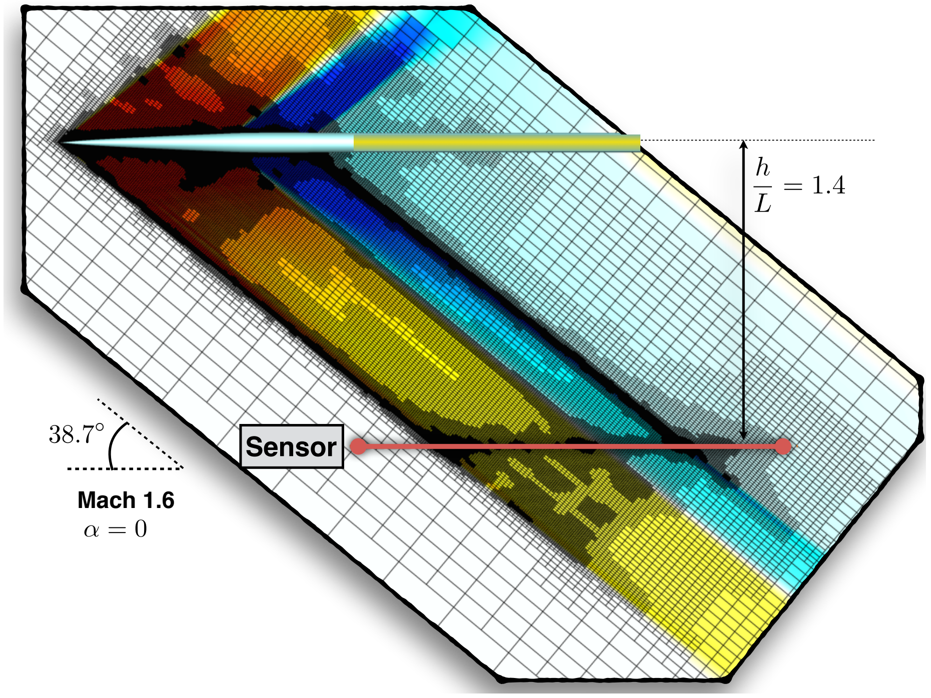


Figure 5.17: ***Supersonic Forebody:*** Example of an adapted flow mesh. Cells are distributed to accurately compute the pressure signature along the sensor *(orange line)*. The mesh shown is for the target shape and has 1.9M cells.

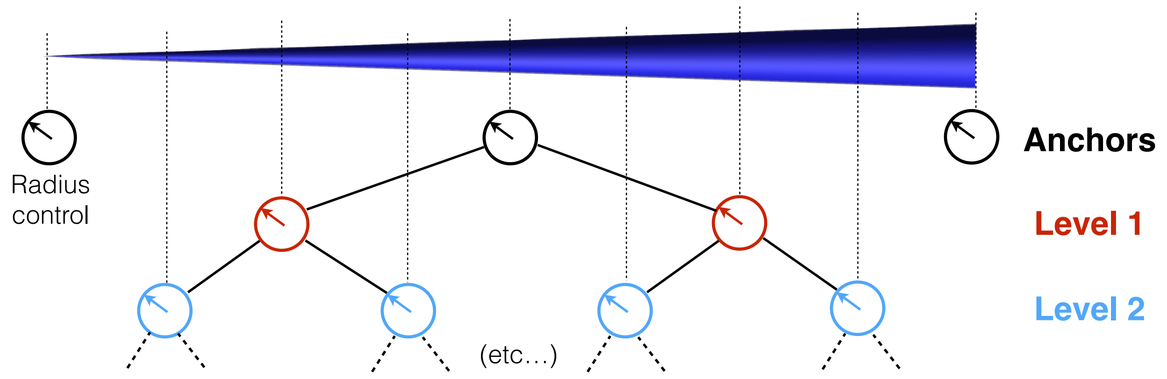


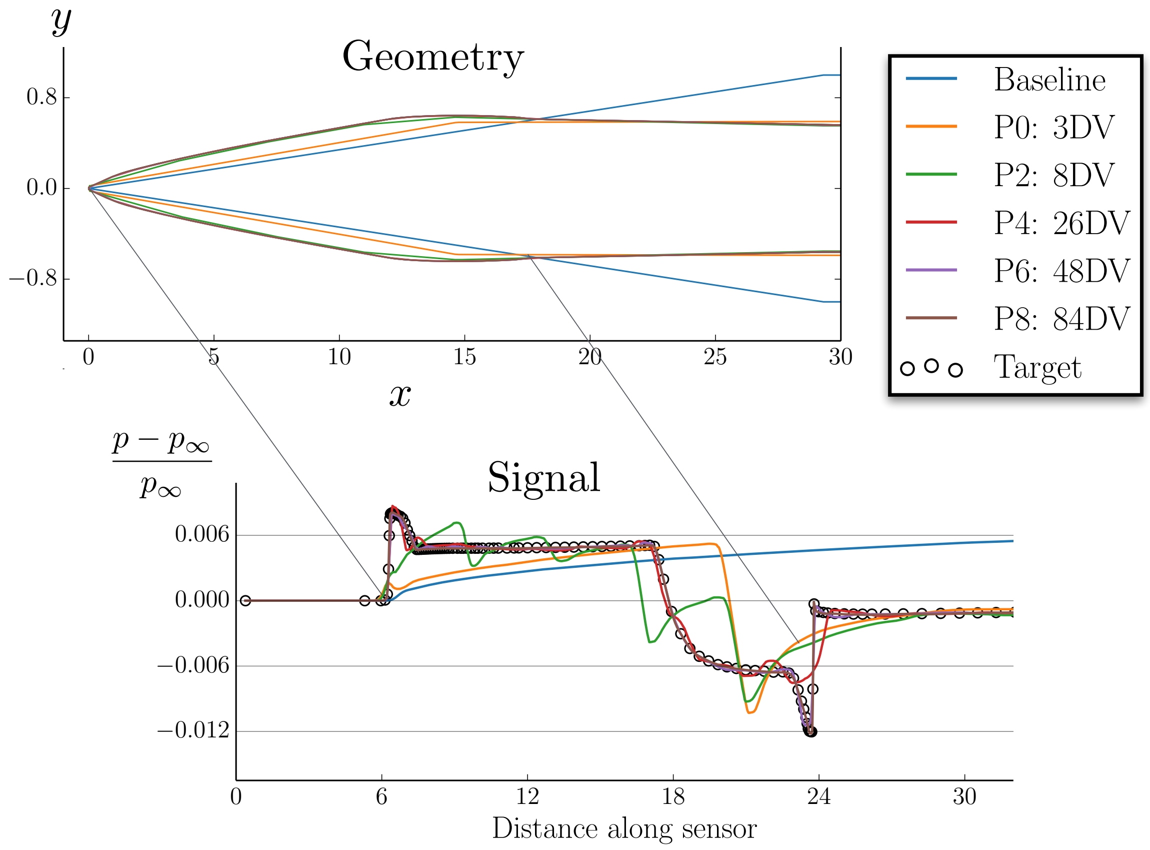
Figure 5.18: ***Supersonic Forebody:*** Hierarchical shape parameterization of forebody radius

Adaptation Strategy

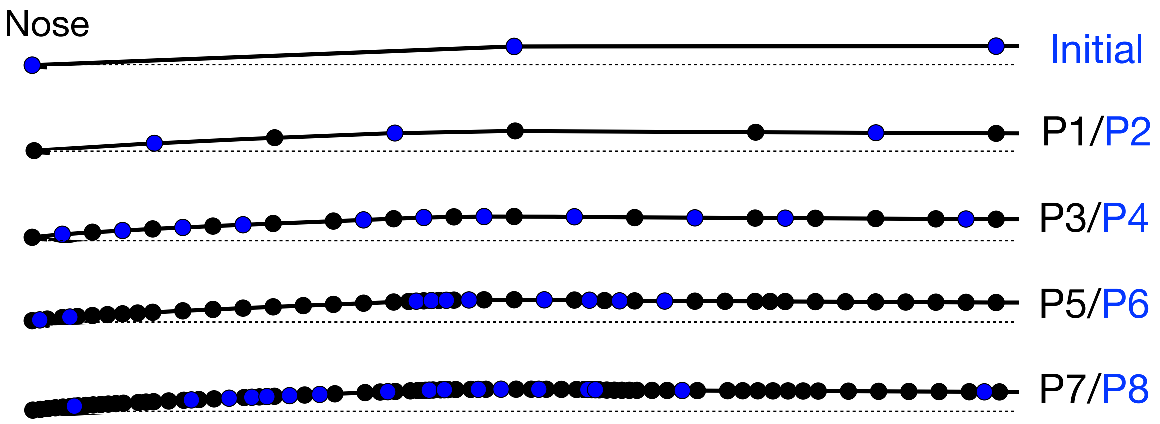
I first test that the adaptive procedure can solve the problem in a continuous sense. For this test, a gradient-reduction trigger is used with $r = 0.01$ and $w = 2$. One level of candidates is considered at a time ($d = 1$), and the regularity rules from Section 3.4.4 are applied. The constructive adaptation procedure is used with $N_w = 3$ and auto-growth settings $w = 4$, $r = 0.05$. Like most aerodynamic functionals, Equation (5.3) does not have an analytic Hessian available. As in Section 5.2, the indicator I_D is computed by prolonging the BFGS Hessian diagonal approximation from the previous search space into each candidate search space, using Equation (3.25). Not knowing the scale term M for this functional, I used the term for inverse surface pressure matching. This was motivated by Figure B.3 in Appendix B.3.1, which suggests that this at least improves the predictions, despite clearly being incomplete.

Results

Figure 5.19a shows several snapshots of the convergence to the target signal. By the end, the signal is nearly indistinguishable from the target. The evolution of the shape parameterization is shown in Figure 5.19b. Generally the system has clustered more shape control in regions corresponding to higher curvature in the signal. Linear supersonic theory predicts that these correspond to high-curvature regions of the geometry itself. Some previous approaches to adaptive parameterization have adapted directly to geometric curvature [81]. For this problem such indirect techniques might perform well, although it is simple to construct problems where that would not be the case.



(a) Signature recovery throughout adaptive parameterization *(for clarity, not all intermediate signals are displayed)*



(b) Parameterization refinement history

Figure 5.19: ***Supersonic Forebody:*** Results of optimization with adaptive parameterization

Figure 5.20 shows the objective smoothly converging by over four orders of magnitude. On the finest level, the gradients are so small that even the small amount of remaining discretization error begins to affect the convergence. These results are a further demonstration that the adaptive strategy can robustly approach the continuous optimal design, at least as closely as the simulation accuracy permits.

Next, I compared the performance of several other refinement strategies on this problem:

- Static parameterizations (17-DV, 33-DV, 65-DV)
- Uniform refinement starting from 9 DVS
- Adaptive refinement starting from 9 DVS (driven by I_D vs. I_G)

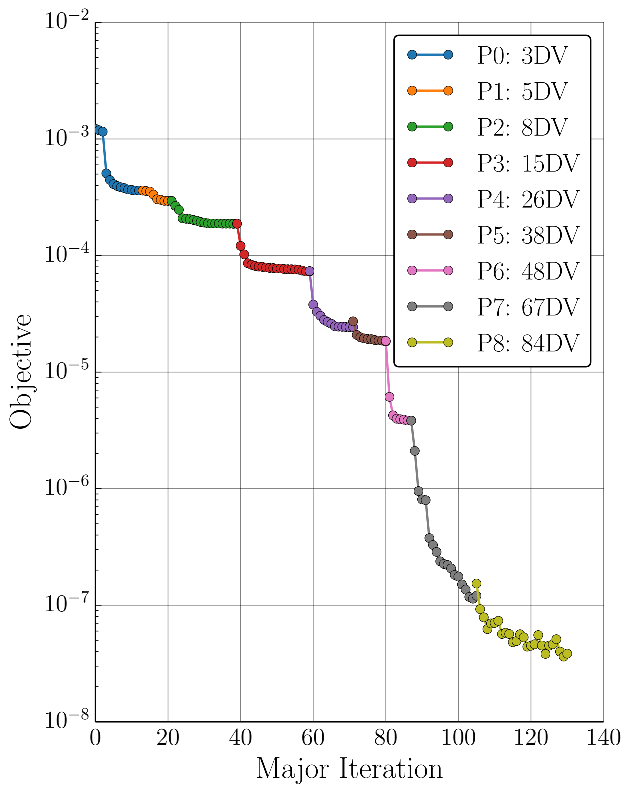
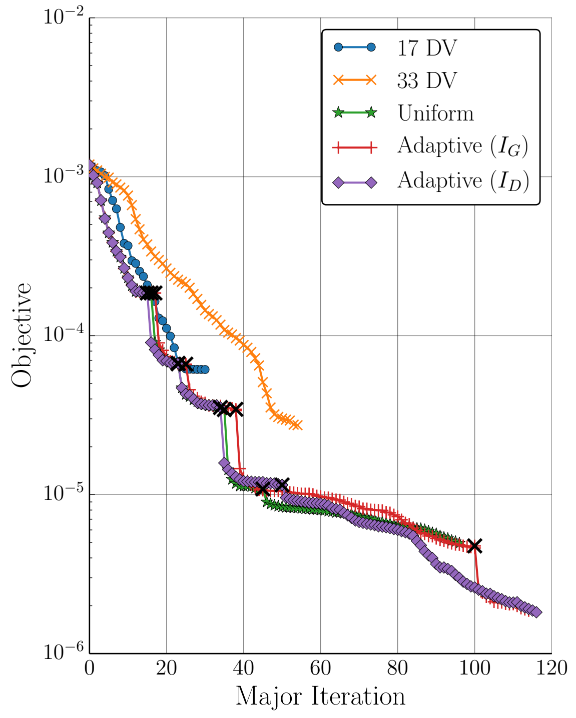


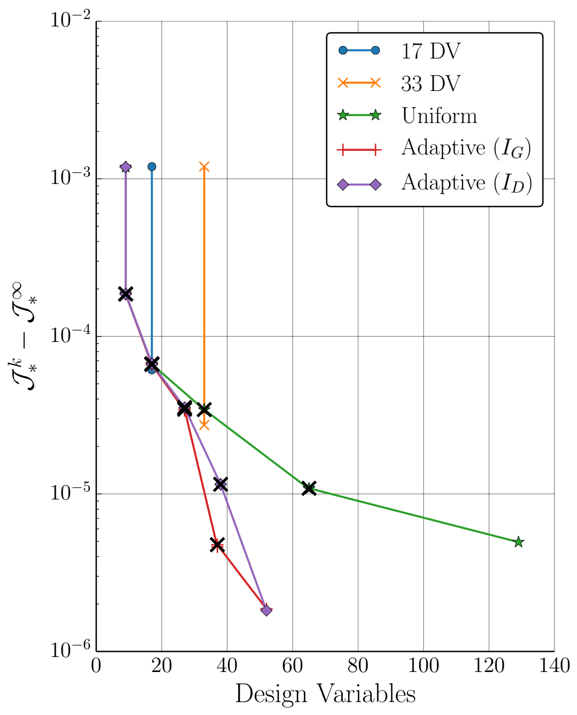
Figure 5.20: ***Supersonic Forebody:*** Objective convergence (adaptive parameterization strategy)

Figure 5.21 compares the objective convergence rates for these strategies. As expected, coarser static parameterizations support only limited matching of the target signal. When attempting to use a 63-DV static parameterization, early high-frequency geometric deformations were enacted, causing the optimization to stall while still quite far from the target. This is a commonly-observed characteristic for optimization with high-resolution shape control (e.g. [162]), and gradient smoothing is typically advocated (e.g. [30]). All progressive parameterization approaches followed qualitatively smoother design trajectories. This obviates the need for such gradient smoothing, results in more realistic designs throughout the early phases of optimization, and ultimately allows the optimizer to drive ever closer to the target signal.

Throughout the process, the progressive approaches achieved equivalent objective reduction with fewer simulations than the static parameterizations. The adaptive strategies required about the same number of simulations as a uniform refinement strategy. However, Figure 5.21b shows that they can solve the problem with substantially fewer design variables, which has numerous benefits. There was not a substantial difference between using I_G and I_D on this problem. Like Section 5.2 this problem involves only one class of shape control, which lessens the need for second-order information. However, I also used a scale-term that is not rigorously true for this objective functional — correcting that might improve the performance of I_D.



(a) Convergence with respect to major iteration. ×-marks denote search space refinements.



(b) Convergence with respect to N_{DV}

Figure 5.21: ***Supersonic Forebody:*** Objective convergence under different refinement strategies

CHAPTER 6

DESIGN EXAMPLES

This chapter evaluates the progressive parameterization system on several drag minimization problems in 2D and 3D, under subsonic and transonic conditions. The first two examples were posed by the AIAA Aerodynamic Design Optimization Discussion Group[1] and have been the subject of numerous studies [51–54, 56, 94, 162, 163]. For these examples, evaluation focuses primarily on the basic progressive system developed in Chapter 2, using uniform shape control refinement. The last two examples examine the potential of the adaptive system developed in Chapter 3 to further improve efficiency and provide the designer with feedback.

6.1 Symmetric Transonic Airfoil

This example considers drag minimization for a symmetric airfoil under inviscid conditions. The problem was posed as part of the AIAA Aerodynamic Design Optimization Discussion Group, where it was investigated by several researchers [51–54, 162, 163]. The initial airfoil is a modified NACA 0012 (henceforth "N0012m"), where the trailing edge is made sharp.[2] The design Mach number is 0.85, while the angle of attack is fixed at $\alpha = 0°$. Additionally, the final airfoil shape is required to contain the original airfoil. This constraint is satisfied when $y \geq y_{N0012m}$ everywhere on the upper surface, and conversely on the lower surface.

[1]Cases I and III from the discussion group.

[2]Via modification of the x^4 coefficient: $y = \pm 0.6\left(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.10\underline{\mathbf{36}}x^4\right)$

Geometry and Parameterization

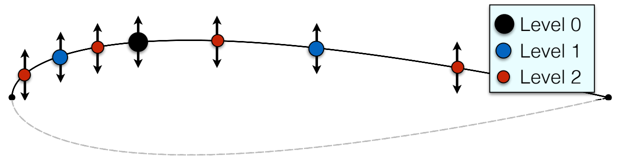


Figure 6.1: ***Symmetric Airfoil:*** Initial parameterization with 7 design variables, generated by twice uniformly refining a 1-DV parameterization (lower half generated by symmetry)

The airfoil is parameterized using the RBF-based direct manipulation technique described in Section 4.4. Initially a single pilot point is placed on the top surface, as shown in Figure 6.1 (black dot). Each parameter enacts a roughly bump-shaped deformation centered on the pilot point (see Figure 4.4). Having observed that it is usually more efficient to start with several design variables rather than a truly minimal set, two uniform refinements are performed before commencement of optimization, yielding seven initial design variables. The shape control is clustered towards the leading edge by transforming the arc-length parametric space.[3] During shape control refinement, new pilot points are placed at the midpoints between existing ones. The midpoint is also measured in the transformed space, so that in physical space, new parameters are biased towards the leading edge.

To handle the containment constraint, I set the lower bound of each shape parameter to the corresponding local thickness of the N0012m. The direct manipulation approach guarantees that the containment constraint will be satisfied exactly at the pilot points. Regions of the surface between them may temporarily violate the constraint, but these violations are clamped down as more parameters are added. This is an example of a situation where it would be incorrect to use the design variable bounds prolongation operator (Equation (2.6)), which is based on linear interpolation. For this case, the containment constraint must be set by directly sampling the equation that generates the N0012m.

Meshing

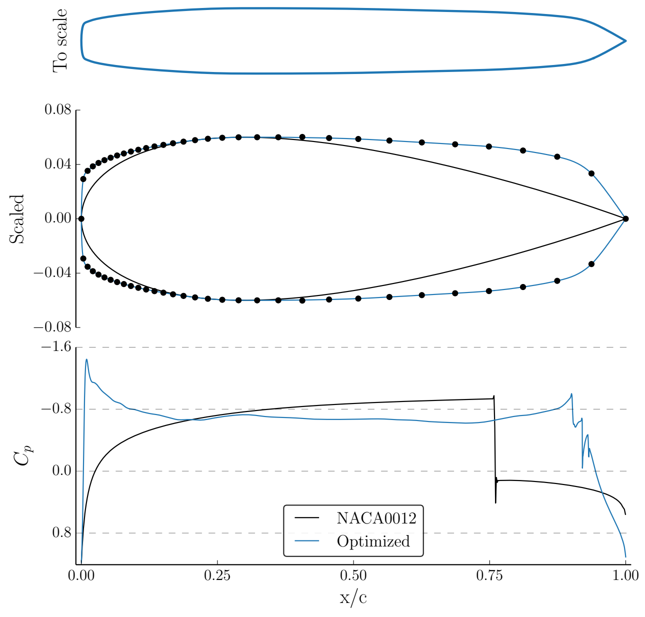
The solution must be symmetric, so the flow is solved only in the upper half of the domain with a symmetry boundary condition at $y = 0$. Due to the extreme sensitivity of the optimal design, forcing symmetry also strongly improves convergence, as observed in [56]. The farfield boundaries are placed 96 chords away in each coordinate direction, which an initial domain size study determined was necessary to resolve the final design's carefully tuned shock structure (see Figure 6.2a). More detail about the sensitivity to the farfield distance was given in a previous paper [94]. The mesh was adapted using the adjoint-based approach described in Appendix C.

[3]Transformation function is $u^{new} = u - 0.15\sin(2\pi u)$.

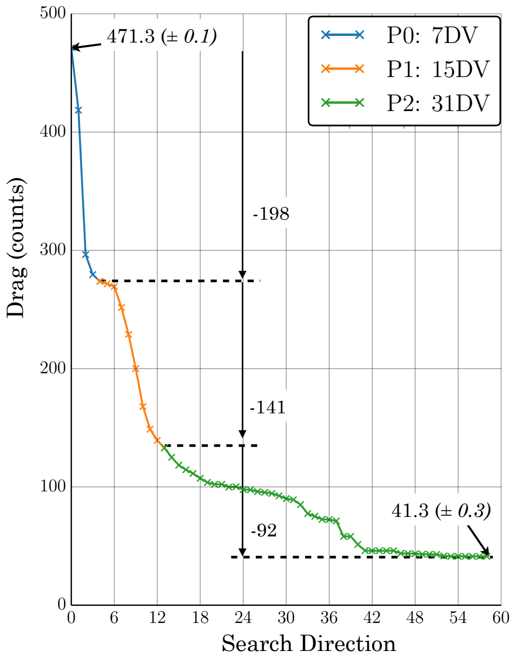
Optimization Results

Algorithm A was invoked, with a uniform refinement strategy, and a fairly aggressive slope-based trigger ($r = 0.2$, $w = 1$). Figure 6.2a shows the final optimized airfoil and its pressure profile. Unable to slim the profile at the thickest point, the optimizer instead substantially increased the thickness at the leading and trailing edges, causing the leading edge to become extremely blunt. Although unintuitive, this is roughly the expected optimal result for this inviscid, symmetric, single-point design problem. Meheut *et al.* directly compare this resulting shape to those obtained by various research groups [56]. The result obtained here, which was discovered via automatic parameterization refinement, performed favorably in comparison to the other results. The final design satisfies the containment constraint over the entire airfoil surface (not just at the interpolation points).

Figure 6.3 compares the initial and final meshes, which were automatically adapted to reduce error in drag. The refinement patterns reflect movement of the shock and changes in the width of the supersonic region. The adjoint-based mesh adaptation process exhibited smooth, asymptotic convergence of the error estimate (see e.g. Figure C.1 in Appendix C). The error level was roughly constant throughout the optimization (see Table 6.1), giving



(a) Final airfoil shape and pressure profile. Black dots indicate final locations of the shape control parameters.



(b) Objective convergence

Figure 6.2: ***Symmetric Airfoil:*** Progressive parameterization optimization results

Table 6.1: ***Symmetric Airfoil:*** Inviscid drag reduction with optimization. Drag and error expressed in counts ($C_D \cdot 10^4$)

	Baseline	7-DV	15-DV	31-DV
C_D	471.3	273.8	133.0	41.3
Error est.	±0.1	±0.1	±0.1	±0.35
Cells	26 K	49 K	50 K	61 K

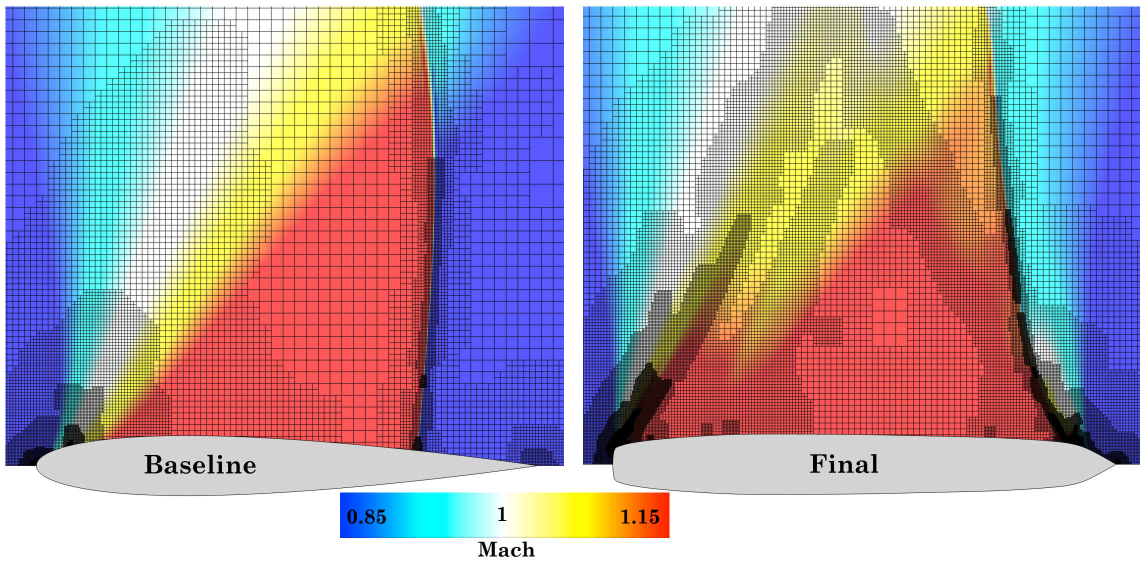


Figure 6.3: ***Symmetric Airfoil:*** Comparison of baseline and final meshes. The mesh refines the regions most important for computing drag, primarily focusing on the leading edge expansion and shock. To achieve the same error tolerance for both designs, the baseline mesh required only 26K cells (upper half only), while the final design required 61K cells.

high credibility to the final design. For the final design, the estimated error in drag was less than 0.3 counts ($< 3 \cdot 10^{-5}$ in C_D).

Figure 6.2b shows the convergence of the objective function over 60 major iterations, and over 3 parameterization levels. By the final 31-DV parameterization, the inviscid drag has been reduced by a factor of 10, from the baseline 471 counts to 41.3 counts. An additional refinement to 63 DVS proved unable to further improve the design. Though some further improvement is evidently possible [56], it is clear that the design is beginning to approach the continuous optimum; Figure 6.2b shows that each additional refinement is resulting in diminishing improvement.

6.1.1 Static, Progressive, and Adaptive Parameterizations

To give a rough performance comparison, Figure 6.4 plots objective convergence with respect to wall-clock time. Each curve corresponds to a different refinement strategy. The progressive refinement schemes (labeled "Progressive" and "Adaptive") achieved faster and deeper overall design improvement than any static parameterization, regardless of its resolution. As expected, low-dimensional static search spaces support limited design improvement, while high-dimensional static spaces take much longer to navigate to the optimum. On the finest (63-DV) static parameterization, which stalled quite early, the optimizer may simply be unable to navigate the design space, as also reported by Carrier *et al.* on this problem [162]. Starting in a coarse design space appears to smooth the navigation early on, leading to a more robust search process.

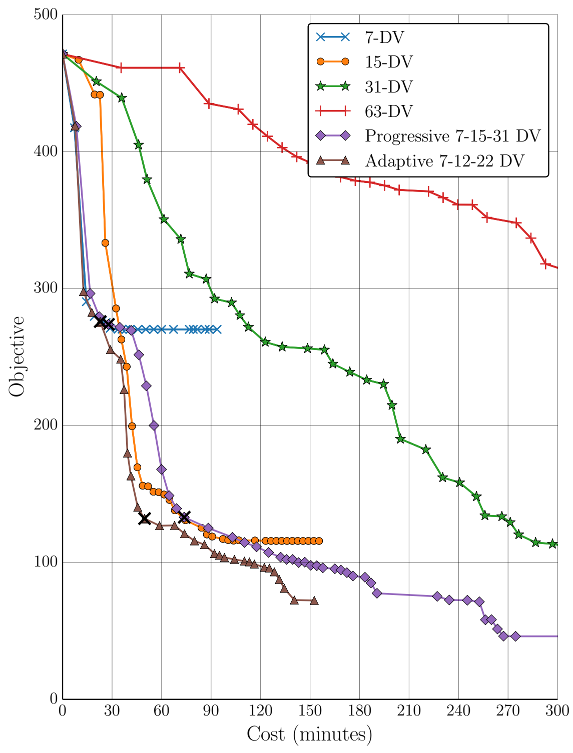


Figure 6.4: ***Symmetric Airfoil:*** Cost-effectiveness of each parameterization scheme, showing design improvement vs. wall-clock time (using late 2013 MacBook Pro, 4× 2.6GHz Intel Core i7, 16GB of memory). ×-marks indicate search space refinements on the progressive and adaptive methods. All cases used identical settings for meshing, solving, error control and triggering.

The adaptive approach achieved similar design improvement but with fewer design variables than uniform refinement (22 vs. 31 DVS). It also performed slightly faster in wall-clock time than uniform refinement throughout most of the optimization. This speedup is largely due to the smaller number of shape derivative calls to the geometry modeler and gradient projections, and also partly due to the somewhat lower dimensional design space. For slow geometry modelers, this advantage could be even more significant. However, other factors such as the trigger (Section 2.4.1), rate of variable introduction (Section 3.4.3), choice of importance indicator (Section 3.2.2), design variable scaling, and the fundamental path-dependence of optimization each strongly impact the efficiency.

6.2 Subsonic Wing Twist Optimization

This example considers wing twist optimization for a straight wing at Mach 0.5. The objective is to minimize inviscid drag at fixed lift ($C_L = 0.375$). The main goal of this example is to demonstrate how an adaptive parameterization system can be combined with adaptive goal-oriented flow meshing with progressive error targets.

Geometry and Parameterization

The baseline geometry is a straight, unswept, untwisted wing, generated by extruding the N0012m section three chord lengths and capping the tip by a simple revolution. The design variables are wing twist at various spanwise stations. The airfoil section and planform remain unmodified. The twist is in the streamwise plane about the trailing edge and is linearly interpolated between successive stations. The global angle of attack is variable while the wing root twist is fixed. The first parameterization ("P0") has two twist stations, located at the tip and mid-span. To generate the second level ("P1"), new twist stations are added at the midpoints between existing ones.

Adaptive Meshing

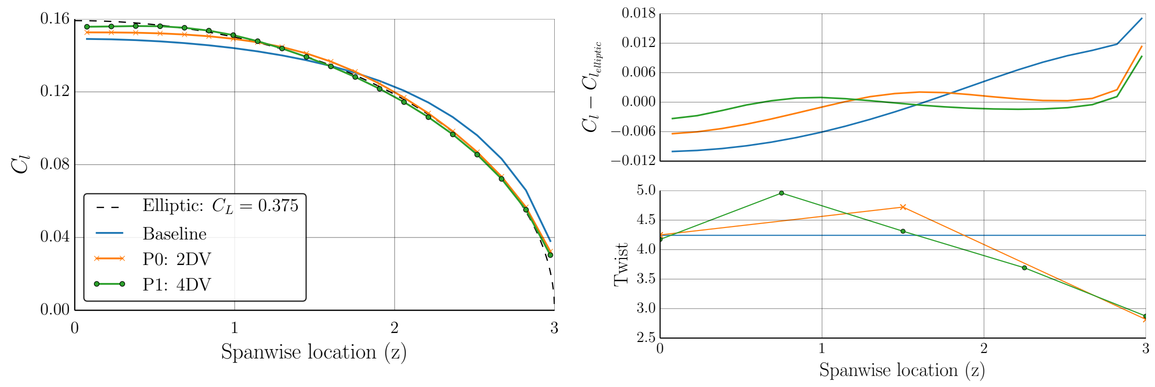
The baseline design has about 76.7 counts of inviscid drag. Unlike the previous example where the objective was reduced by a factor of ten, here the possible improvements are very small. Assuming the span efficiency factor e cannot exceed 1.0, as non-planar deformations are minimal with the twist applied about the trailing edge, the lowest possible inviscid drag is

$$C_{D_{min}} = \frac{C_L^2}{\pi e_0 \text{Æ}} = \frac{0.375^2}{6.0\pi} = 74.6 \text{ counts} \tag{6.1}$$

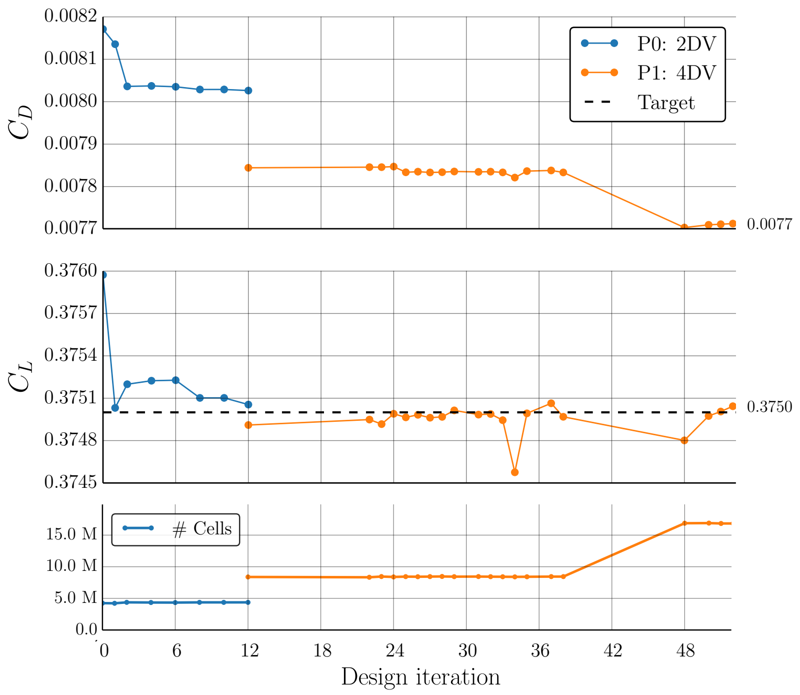
However, a very small shock was observed on the wing tip near the trailing edge, where the flow accelerates around the tip to the top surface, which may further erode the possible drag reduction. Adjoints are solved for the drag and lift functionals to compute their gradients, allowing the nonlinear lift constraint to be treated exactly by SNOPT. The farfield boundaries were placed at 48 chords away. The mesh was automatically adapted to minimize error in span efficiency factor:

$$\mathcal{F}_{adapt} = \frac{C_L^2}{\pi C_D \text{Æ}}$$

For this type of problem, there are high demands on the accuracy of the flow solutions [164]. To accelerate optimization, in the first search space the error targets were set such that the meshes contained about 5 million cells. For the second search space, the tolerance was tightened, resulting in meshes with 10-15 million cells.



(a) *Left*: Sectional lift distribution profiles. *Top right*: Deviation from elliptic distribution. *Bottom right*: Twist distribution (twist is in degrees and indicates local angle of attack ($\alpha_{global} + \gamma$))



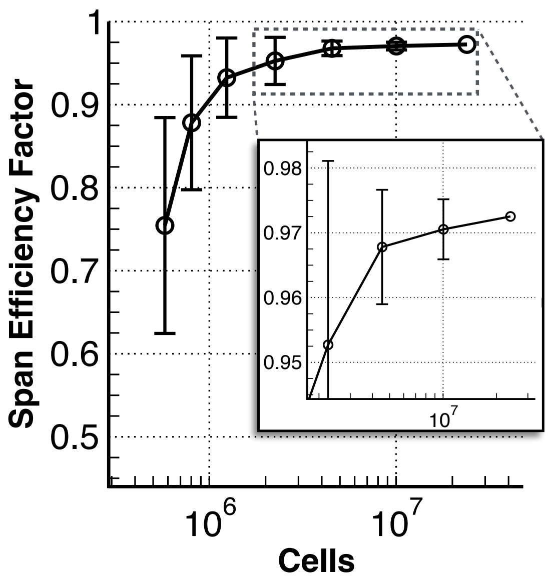
(b) Functional convergence history, *Top*: Inviscid drag, *Middle*: Lift, *Bottom*: cell count

Figure 6.5: ***Twist optimization:*** Results

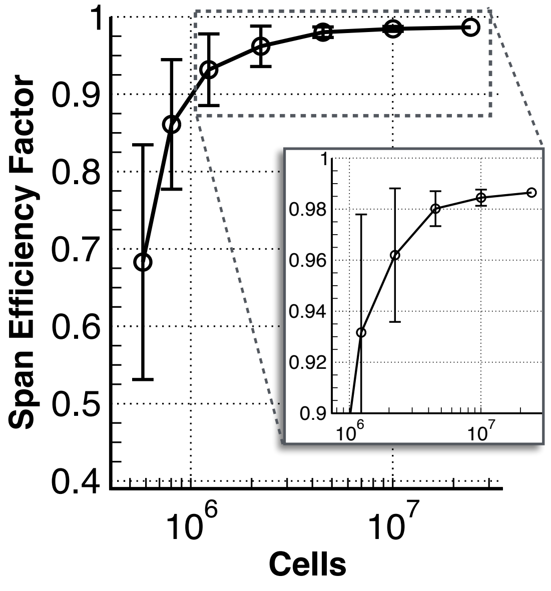
Optimization Results

Figure 6.5a shows the main results of the optimization. The lift distribution rapidly approaches an elliptical shape. The only small discrepancies are at the tip, due to the untapered section, and at the root, which must compensate to exactly match lift. Figure 6.5b shows the convergence of the lift and drag functionals. Because a coarser mesh was used in the initial design space, there is a jump in functional values when transitioning to the finer design space. Further refinement of the parameterization beyond five twist stations proved unable to substantially reduce the drag further. This may be partly due to losses stemming from the shock at the tip.

To accurately determine the total improvement, additional accurate analyses were performed on the initial and final designs. Figure 6.6 shows the convergence of span efficiency factor with mesh refinement for the initial and final designs, both trimmed to $C_L = 0.3750$. The initial design had $C_D = 76.7$ counts of inviscid drag ($e = 0.973 \pm 0.005$). By the final design this was improved to $C_D = 75.6$ counts ($e = 0.987 \pm 0.003$). The error bounds come from the adjoint-derived discretization error bounds shown in Figure 6.6, which smoothly and asymptotically converge with mesh refinement and properly bracket the final functional value for the last several meshes.



(a) Baseline design



(b) Final design

Figure 6.6: ***Twist optimization:*** Convergence of span efficiency factor with mesh refinement

6.3 Multipoint Transonic Airfoil Design

This example considers multipoint transonic airfoil design under inviscid conditions. The complete optimization statement is given in Figure 6.7. The objective is to minimize an equally-weighted sum of drag at two flight conditions, Mach 0.79 and 0.82, with $C_L = 0.75$ at both conditions. A minimum pitching moment constraint is imposed at each design point. The total cross-sectional area A must be preserved, and the airfoil is required to maintain at least 90% of its initial thickness everywhere (enforced discretely at 20 chordwise locations t_i). Because the solver is inviscid, the optimization often leads to excessive trailing edge camber, which would result in poor viscous performance. To prevent this, I constrained the camber line angle γ at the trailing edge as well as the geometric closing angle ϕ (see Figure 6.7).

$$
\begin{array}{lrl}
\underset{\mathbf{x}}{\text{minimize}} & & C_{D_1} + C_{D_2} \\
\hline
\text{subject to} & & C_{L_1} = C_{L_2} = 0.75 \\
 & (\mathcal{V}) & C_{M_1} \geq -0.18 \\
 & (\mathcal{V}) & C_{M_2} \geq -0.25 \\
 & & 9^\circ \leq \phi \leq 13^\circ \\
 & (\mathcal{V}) & \gamma \leq 6^\circ \\
 & & A \geq A_{RAE} \approx 0.07787 \\
 & & t_i \geq 0.9 t_{RAE_i} \quad \forall i
\end{array}
$$

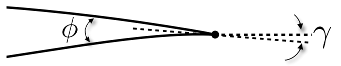


Figure 6.7: ***Transonic Airfoil:*** Problem statement. $(\mathcal{V})$ denotes constraints that are initially violated.

Gradients for the six aerodynamic functionals are computed using adjoint solutions. The 23 geometric constraints are computed on the discrete surface, with gradients derived analytically. The angle of attack at each design point is variable. At each design iteration, and for each design point, adjoint-driven mesh adaptation is performed to control discretization error (see Appendix C), culminating in a final fine-mesh flow solution and three adjoint solutions to compute gradients for drag, lift, and pitching moment.

Geometry and Parameterization

The baseline geometry is the RAE 2822 airfoil shown in Figure 6.8. The airfoil is parameterized using the RBF-based direct manipulation technique described in Section 4.4. To remove as much initial bias as possible, the optimizer was initially given only two design variables (top and bottom). Figure 6.8 shows

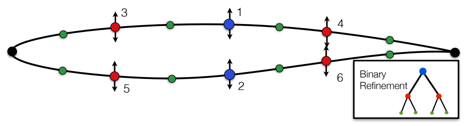


Figure 6.8: ***Transonic Airfoil:*** Baseline geometry (RAE 2822), showing first three levels of uniformly refined shape control (2-DV, 6-DV and 14-DV)

the design variable locations for the first few shape control levels. The tree depth was limited to $d_{max} = 5$, so that the finest allowed parameterization has 62 parameters.

Adaptation Strategy

In the verifications studies in Sections 5.2 and 5.3 it became clear that for problems with only a *single* class of shape control (like this problem), a first-order indicator provides sufficiently good relative rankings. Therefore, Equation (3.17) was used to rank candidate refinements. I used the slope reduction trigger from Section 2.4.1, with $r = 0.01$. In the presence of the initially violated constraints, SNOPT's merit function undergoes large fluctuations, which can cause early slope-based triggering. Therefore, I used a large window of $w = 6$ for the first three levels, and subsequently lowered the window to $w = 2$ to improve efficiency. The target growth rate was manually set to 1.75×. (Actual growth rates are also affected by regularity rules.) For the constructive refinement algorithm (Function 5) I used $w = 3$.

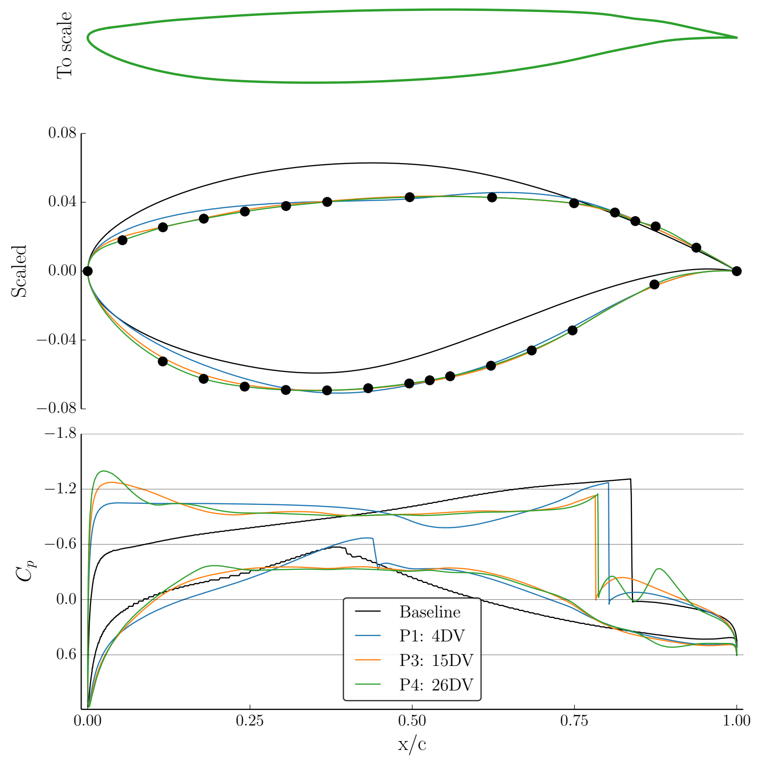


Figure 6.9: ***Transonic Airfoil:*** Optimization results for the adaptive parameterization approach. *Top*: Optimized airfoil to scale. *Middle*: Final airfoil from three intermediate search spaces (4-DV, 15-DV, 26-DV), showing 26-DV adapted parameterization. *Bottom*: Corresponding pressure profiles for the Mach 0.79 design point.

Optimization Results

Figure 6.9 shows the airfoil shapes and pressure profiles at three stages during the adaptive approach (4-DV, 15-DV, 26-DV). Examining the Mach 0.79 pressure profile, the loading is shifted forward. The reflex camber at the trailing edge is made more shallow to satisfy the camber-line angle constraint. The main shock is moved forward and weakened. A small shock temporarily appears on the lower surface while meeting the constraints, but is then eliminated by the final design. Overall the inviscid drag at Mach 0.79 is reduced from over 300 counts to 66 counts, and at Mach 0.82 from about 600 counts to 276 counts. Figure 6.9 also shows the non-uniformly refined final parameterization. The sequence of the first few adapted parameterizations is shown in Figure 6.10.

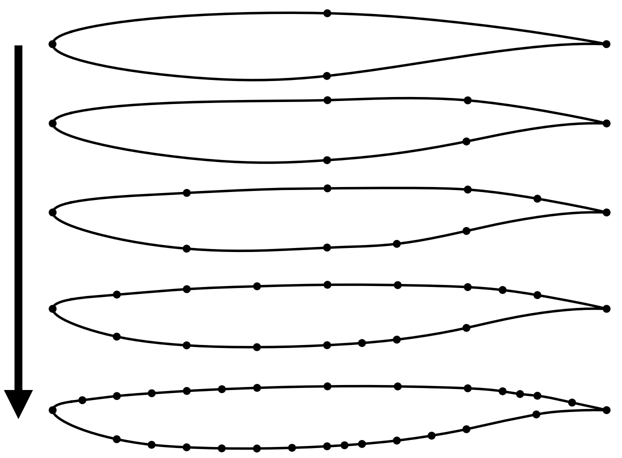


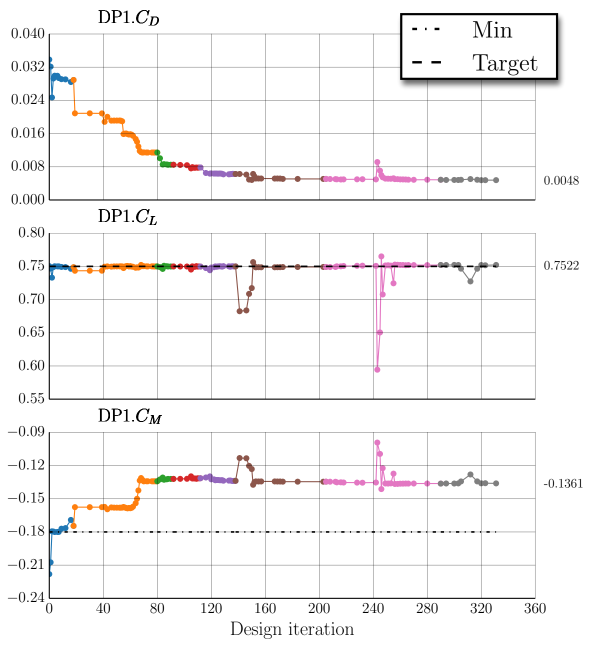
Figure 6.10: ***Transonic Airfoil:*** History of adaptive refinement, showing best airfoils attained under the first several parameterizations.

Figure 6.11 shows the evolution of the lift, drag and pitching moment functionals. The constraints are rapidly met and held throughout the optimization, while the drag is gradually reduced. The geometric constraints are all satisfied by the final design, and many are active, especially the area and trailing edge angle constraints. Because viscous effects are ignored, it is not surprising that the final shapes exhibit non-ideal behavior at the leading and trailing edges.

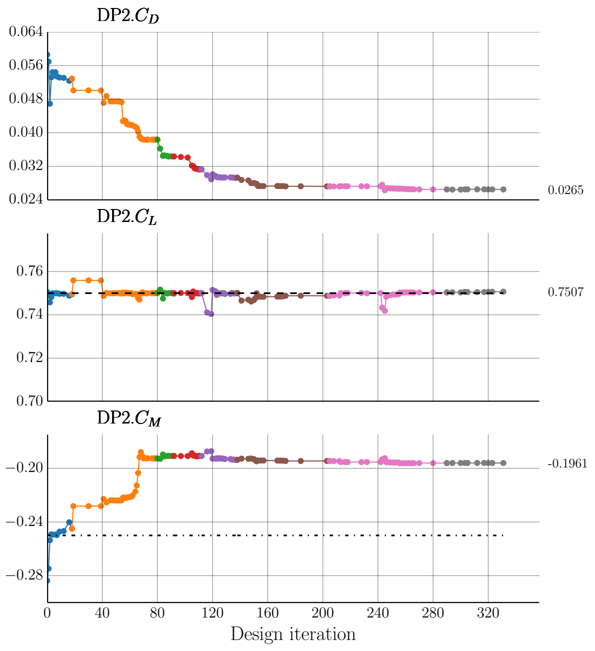
At each re-parameterization, the quasi-Newton optimizer performs a "cold restart", which resets the Hessian approximation to the identity matrix. Because of this, the lift constraints are violated for the first few search directions immediately after refining, before quickly snapping back to the targets. At the final design, the airfoil is still slowly improving. The fact that substantial gains were made even on the final parameterization indicates that the continuous limit of design improvement has not yet been reached.

Comparison to Static and Uniform Refinement

Now I compare the performance of the previous optimization to several static shape parameterizations (with 6, 14, 30 and 62 shape design variables) and to uniform refinement.



(a) Mach 0.79 design point



(b) Mach 0.82 design point

Figure 6.11: ***Transonic Airfoil:*** Convergence of aerodynamic functionals across all adaptively refined parameterization levels *(2-DV in blue, 4-DV in orange, etc.)*. Target/minimum constraint values shown in dashed lines.

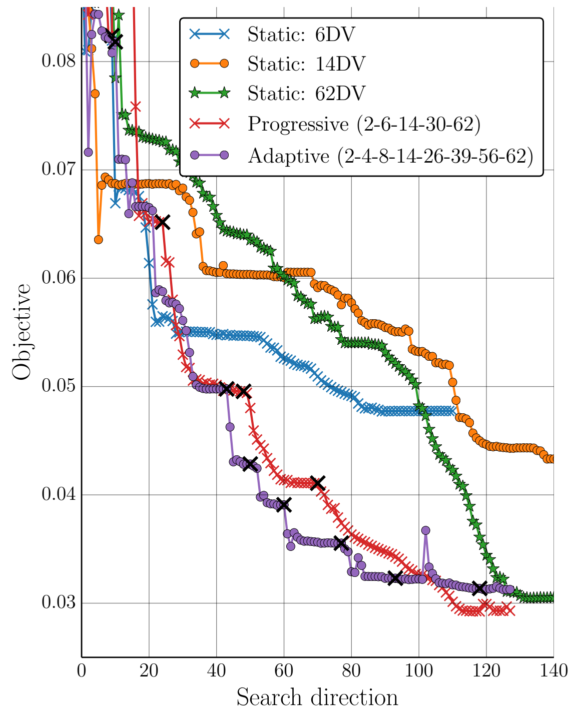
Figure 6.12a compares the convergence of the combined drag objective for the various parameterizations. On first glance, the first 10-20 major iterations appear to exhibit convoluted convergence. During this phase, the optimizer was driving the initially violated constraints to be satisfied, which incurred a severe drag penalty. Subsequent search space refinement enabled the drag objective to be reduced. Overall, the progressive and adaptive approaches strongly outperform any of the static parameterizations, achieving more consistent progress, converging far faster, and ultimately reaching equivalently good or superior designs. This is a clear confirmation of the predicted behavior, described and illustrated notionally in Figure 2.2 as following the “inside track” of the static parameterizations.

The acceleration is even more apparent in Figure 6.12b, which rescales the x-axis to show objective improvement vs. wall-clock time. The progressive and adaptive approaches reach the same objective value as the 63-DV parameterization in about one third the time. In addition to the savings to due optimizing in lower dimensional spaces (which is visible in Figure 6.12a), Figure 6.12b shows additional savings due to the $\mathcal{O}(N_{DV})$ shape derivative computations $\frac{\partial \mathbf{S}}{\partial \mathbf{X}}$ and the $\mathcal{O}(N_{DV})$ adjoint inner products to compute $\frac{\partial \mathcal{J}}{\partial \mathbf{X}}$ and $\frac{\partial \mathcal{C}_j}{\partial \mathbf{X}}$. Typically these costs are neglected in comparison to the flow solutions; here they are especially visible because of the low-cost flow solution. Adaptive refinement controls these costs by reducing the number of design variables. Figure 6.12 also includes the cost of long line searches, visible in the occasional long gaps between major iterations.

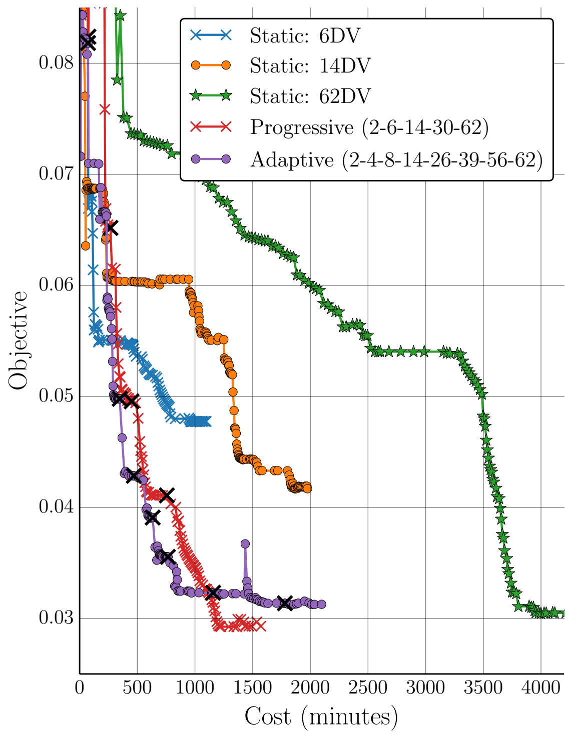
Early in design, some of the static design spaces initially outperform the extremely coarse (2-, 4- and 6-DV) progressive and adaptive search spaces. This indicates that the choice to start with a minimal 2-DV design space was not ideal. In terms of wall-clock time, it is more efficient to start with several variables. Nevertheless, by the end, the progressive approaches have still solidly outperformed the static parameterizations, which tend to stall well before reaching their theoretical potential,[4] most likely because of the relative lack of smoothness in their design trajectories.

By adjusting the progressive and adaptive strategies, even more speedup is certainly

[4]I performed cold restarts to verify that no further progress could be made.



(a) Major iteration history



(b) Design improvement vs. cost (rough timings on 24 Haswell cores) plotted only at major iterations

Figure 6.12: ***Transonic Airfoil:*** Convergence of combined drag value ($C_{D_1} + C_{D_2}$) (ignoring satisfaction of constraints) for each parameterization method. $\times$-marks denote search space refinements.

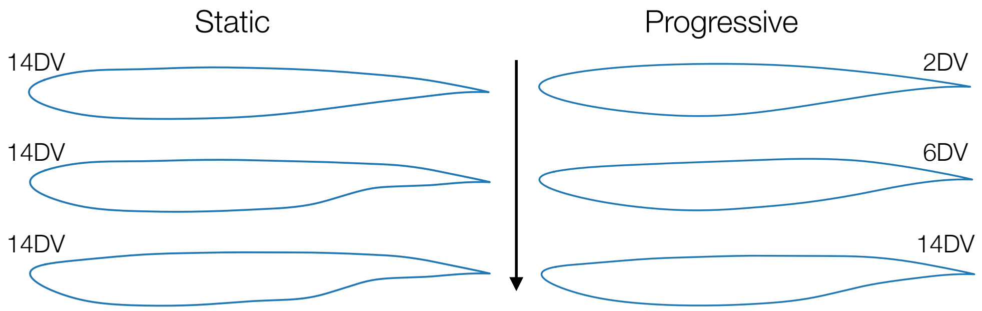


Figure 6.13: ***Transonic Airfoil:*** Shapes encountered during optimization under a progressive parameterization (*right*) are consistently much smoother than airfoils encountered under a static parameterization (*left*).

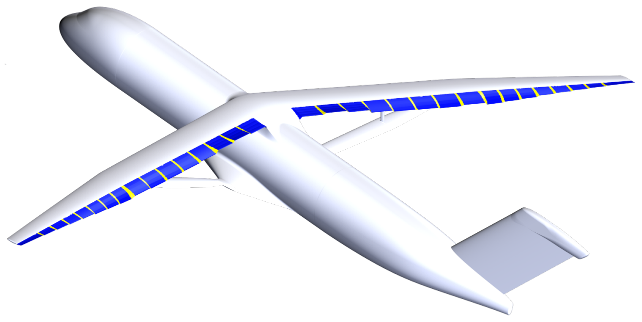
possible. For example, the relatively delayed trigger could be tightened, as it resulted in several extended periods of little design improvement. The second-order indicator might also provide some improvement, as could an auto-growth strategy. Nevertheless, even the chosen strategy demonstrates tremendous acceleration.

As a final note for this problem, Figure 6.13 shows several representative airfoils encountered during optimization. These support a general observation that with a progressive or adaptive approach, the entire design trajectory consists of qualitatively smoother, more reasonable airfoils. The early parameterizations preclude high-frequency oscillations. Only as the design approaches the optimum are higher frequency deformations permitted. This is a desirable characteristic from a robustness standpoint, and also because it makes it possible to halt the optimization at any point and have a reasonable design.

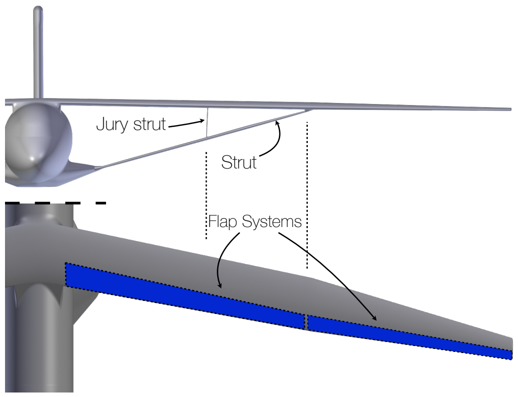
6.4 Adaptive Flap Layout for Truss-Braced Wing Configuration

This final example considers adaptive wing morphing on a truss-braced wing aircraft configuration. Adaptive wing morphing encompasses a broad class of techniques for reducing the drag penalty that is usually incurred when flying at off-design conditions. One specific morphing concept is the variable camber, continuous trailing edge flap (VCCTEF) system, where a flap system is installed at the trailing edge of the wing, as illustrated in Figure 4.5. Each flap can be subdivided into multiple segments (Figure 4.5b).

Previous studies have demonstrated that such flap systems can yield substantial drag savings at off-design flight conditions [155]. However, there are major open questions regarding the design of such as system. How many flaps should be installed, and how should they be distributed and sized? Clearly, increasing the number of flaps increases the potential aerodynamic benefits of flap adaptation — but it also increases the complexity and



(a) Trailing edge flap system. Flaps are in blue, elastomer regions depicted in gold.



(b) Strut and flap system locations

Figure 6.14: ***Truss-braced Wing:*** Configuration

weight of the system. In this example, I show that the adaptive parameterization system can provide a designer with feedback to help answer these questions.

Approach

All examples thus far have considered designing a fixed *cruise* shape. Here we take a given cruise configuration and try to determine a minimal set of flaps to install that will maximally improve off-design performance. The process consists of two alternating phases:

1. Optimize the flap deflections to minimize drag.
2. Perform the directional flap subdivision that would most improve the entire system's *potential* to further reduce the drag.

This process is equivalent to Algorithm A, and step (2) is precisely what the refinement indicator predicts. However, there is one important distinction from all previous examples. In those cases the shape control merely facilitated the efficient discovery of an optimal cruise shape. Here, the shape control has a physical manifestation. We are seeking an efficient set of active deformation modes to build into the aircraft itself. This places a much higher premium on precise adaptation. Therefore, I perform only one subdivision at a time ($N_{add} = 1$), and deeply converge the optimization of the flap deflections at each level.

Note that this procedure is effectively a "greedy" algorithm to solve a combinatorial optimization problem in one shot. One flap is added at a time, based on a local assessment of importance. This procedure is not guaranteed to find the globally optimal flap topology, but it is far less costly than exhaustive, random, or evolutionary searches. I consider only binary subdivisions of the flap structure. Clearly, allowing continuous resizing of individual flaps or merging of existing flaps would offer much more flexibility.

Aircraft Configuration

The baseline aircraft configuration involves a truss-braced wing (TBW), shown in Figure 6.14a. TBW concepts have remained a topic of interest over the years [165, 166], partly due to their ability to support a wing of much higher aspect ratio than conventional transport wings. The wing used for this test has a span of 169 feet, with an aspect ratio of about 19.4. To support such a long slender wing while controlling weight, a large strut and a vertical jury strut are installed. The flight Mach number is 0.7, with a lift coefficient of $C_L = 0.766$, with $S_{ref} = 1475\text{ft}^2$.

As a caveat, it must be noted that the baseline design had not been sufficiently optimized for drag and has a rather high inviscid drag of $C_D = 0.0169$ for the full configuration, sans horizontal tail, pylons and nacelles. There are shocks in the truss region, which incur substantial wave drag penalties. The purpose of this study is to examine a methodology for flap adaptation, but we will see that in the process of laying out the flap system, the adaptation pattern will also reveal an opportunity to improve the baseline design.

Flap System Modeling

As described in Section 4.4.1, I take an existing flap system modeler developed by Rodriguez *et al.* [155] and non-invasively wrap it to enable adaptive parameterization. Each flap segment is considered to be made of a rigid material, and is deflected by rigid rotation about a specific axis. Following [155], between adjacent flaps, I assume a narrow region of an elastomer material, which is modeled by smoothly interpolating the neighboring flap deflections using cubic interpolation.

Two independent flap systems are established, for the inboard and outboard sections of the wing, as shown in Figure 6.14b. The division between the two coincides with both the wing break and the wing-strut junction. Together, they cover the full length of the wing. Initially, each system consists of a single monolithic flap, which can be subsequently subdivided, in the streamwise and/or spanwise direction.

Flow Meshing

The farfield boundaries are placed at about 20 semi-span lengths away in each direction, and a symmetry boundary condition is applied along the fuselage centerline. The meshes were adapted to resolve a weighted sum of lift and drag on the entire configuration:

$$\mathcal{F}_{adapt} = C_L + 10C_D$$

To reduce the cost of the optimization, flow meshes of 12-13 million cells were used. Fine grid solutions of 60 million cells were performed on the initial and final designs to verify the performance improvement.

Verification

Initially, there are two geometric design variables: the deflections of the monolithic inboard and outboard flaps. The global angle of attack is also a design variable throughout this study. After optimizing those flap deflections, the system is given four possible candidate subdivisions, shown in Figure 6.15. I examine ranking these candidates using the first-order indicator, both accounting for the lift constraint ($\Delta\mathcal{J}_\star^{\mathcal{H}}$, Equation (3.17)) and ignoring it ($\Delta\mathcal{J}_\star^{\mathcal{H},\not{c}}$, Equation (3.19)). The latter case is similar to the indicator used in [50], which could not explicitly account for non-localized constraints, like lift. I did not use any Hessian information in this case due to uncertainties about the accuracy of using the 2D pressure-matching prolongation functional for this 3D case.

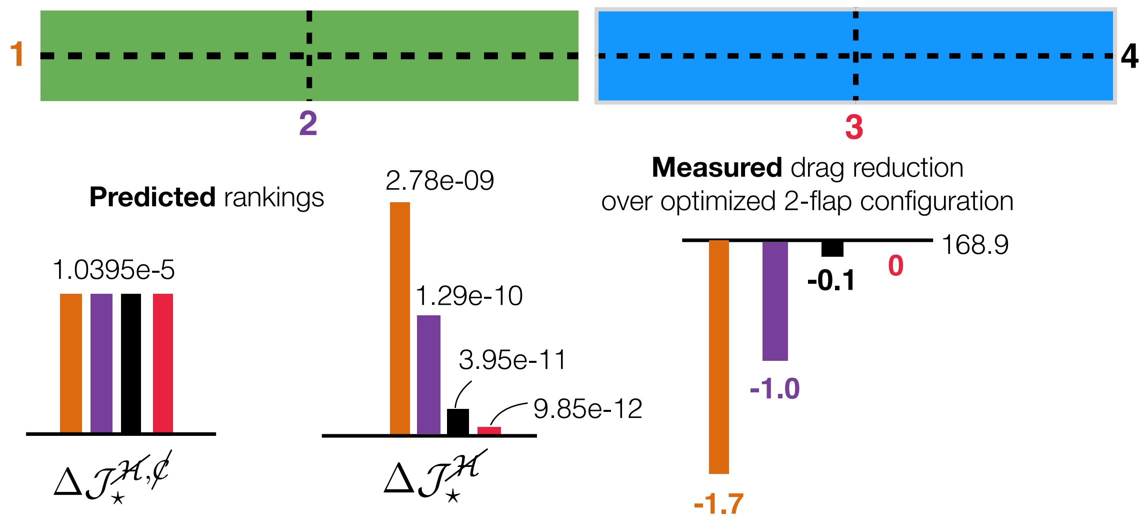
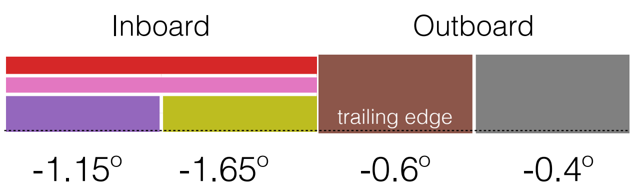
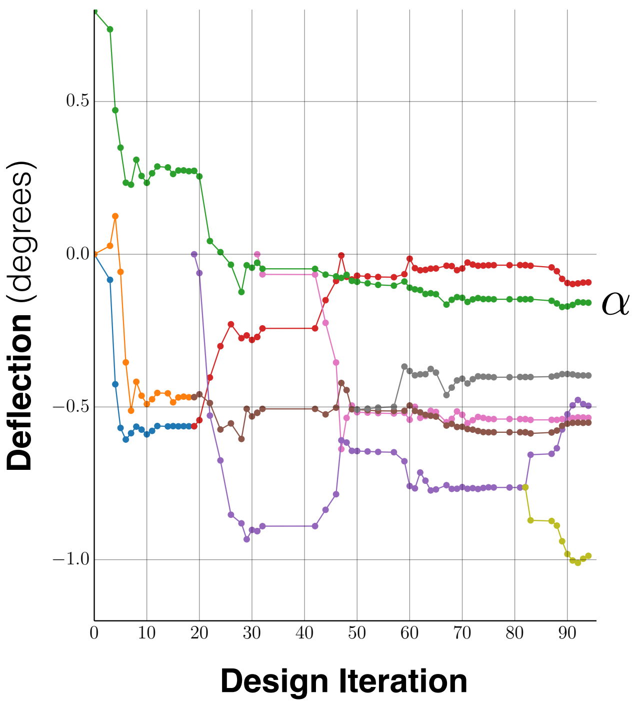


Figure 6.15: ***Truss-braced Wing:*** The adaptive system ranks the four initial candidate flap subdivisions using I_G, both accounting for and neglecting constraints. The version accounting for C_L provides a correct relative ranking of the candidates, as demonstrated by the actual drag recovery obtained by optimization. (Drag values are in counts, trimmed to C_L constraint.)

Figure 6.15 compares the rankings of potential drag reduction according to both $\Delta\mathcal{J}^{\mathcal{H}}_{\star}$ and $\Delta\mathcal{J}^{\mathcal{H},\not{c}}_{\star}$. Without the constraint terms, the indicator essentially cannot distinguish among the candidates — the drag objective is roughly equally sensitive to all the possible subdivided flap layouts. By explicitly accounting for the lift constraint in the indicator, the system assigns very different importances to each subdivision. These predictions are borne out in reality. The actual drag reduction for each candidate refinement is measured by running an independent optimization, like in Sections 5.1.1 and 5.2.1. The right side of Figure 6.15 shows that the subdivisions ranked highest by $\Delta\mathcal{J}^{\mathcal{H}}_{\star}$ indeed achieve the greatest drag reduction in practice, and vice versa.



(a) Final flap topology and cumulative flap deflections at trailing edge (negative deflection downward)



(b) History of flap component deflections (colors correspond to flap topology diagram.) Negative deflection downward.

Figure 6.16: ***Truss-braced Wing:*** Final results

Results

Next, several more flap adaptations are performed, with intervening flap optimizations. The final flap layout is shown in Figure 6.16a. The adaptive procedure has focused flap refinement on the inboard system and has substantially deflected the inboard flaps downward, while lowering global α to hold lift. By deflecting these inboard flaps down, the sectional lift in the truss region increases, relative to the outboard regions. The resulting increase in circulation slows the airflow through the truss, reducing the strength of the shocks, and thus reducing wave drag. To verify the drag savings, adaptive fine mesh solutions (60 million cells) were performed on the initial and final design, with α trimmed to meet the lift constraint. On this substantially more accurate mesh, the baseline design had $C_D = 0.0145 \pm 0.0002$. The final adapted flap system reduced the inviscid drag to $C_D = 0.0137 \pm 0.0002$, an overall savings of about 8 counts.

Figure 6.17 compares optimized inviscid drag (on the coarser mesh) for different flap layouts. Each point on the diagram corresponds to a particular flap topology. The blue curve shows the history of the adaptive refinement procedure. The other two points correspond to two reasonable hypotheses as to what might constitute an effective 8-flap topology. The adaptive procedure has discovered a layout that outperforms and these two layouts, while using fewer flap components.

These results come with several caveats. First, this flap layout was designed to improve a given baseline cruise design, which itself was far from optimal. Presented with a better baseline (which would be unlikely to suffer from so much wave drag), the adaptive system would pursue different flap layouts, perhaps focusing on improving the spanwise lift distribution. Secondly, the layout design process was goal-oriented — it sought the flaps that could most reduce the objective while meeting the constraints. With a more comprehensive optimization formulation (perhaps involving weight, structures, flutter, etc.), the resulting

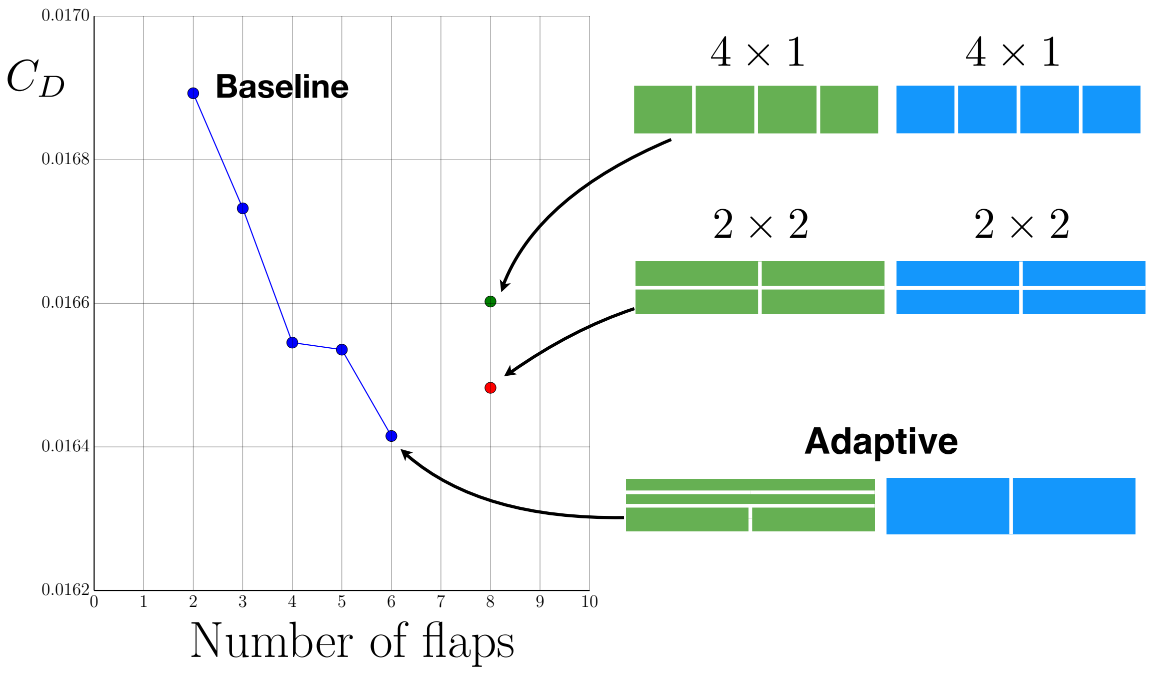


Figure 6.17: ***Truss-braced Wing:*** Inviscid drag on full configuration, on coarse 12-million cell optimization mesh. Adaptive refinement has uncovered a flap topology that outperforms two reasonable layouts, while using fewer components. In the process it has begun to reveal an approximate tradeoff between drag reduction and number of parts.

flap topology would reflect those different goals. Thirdly, the procedure was limited to binary subdivisions of flaps. If the system were provided with more options for flap layout, it could find a superior one, although the adaptation procedure would become more expensive.

The True Pareto Front

Figure 6.17 gives an approximate sense of the tradeoff between the simplicity of having fewer flap components and the greater drag reduction possible with more flaps. This is visually similar to a Pareto front, but is not truly the globally optimal . In Section 7.2 I discuss some ways to build accurate Pareto fronts using adaptive parameterization. However, in this case one of the objectives is a discrete variable, N_{flaps}, which makes the problem substantially more difficult. To compute the globally optimal layout for N components, one would need to perform a full flap optimization for each of the $\mathcal{O}(N!)$ possible layouts. The functional evaluation for this problem is itself an entire optimization. Even with genetic algorithms, the cost of this approach would be prohibitive.

Alternatively, we could use the indicator as a surrogate for expected drag reduction of a given flap layout. This would greatly reduce the cost, but the process would be even more approximate. The indicator is only a *local* prediction; it is not clear that predictions made far from the optimum would be sufficiently accurate. Under the greedy strategy I used, the indicator predictions should improve as they are evaluated closer and closer to the optimum.

CHAPTER 7

Discussion, Future Work, and Applications

In a progressive parameterization approach, the search space is enriched automatically as the optimization evolves. Recognizing that different design problems may call for different shape control, and that it may be difficult to predict the appropriate parameters a priori, I developed an *adaptive* approach that aims to discover the necessary shape control while concurrently optimizing the shape. This approach has several inherent benefits:

- **Automation**: The designer does not need to predict the necessary shape parameters before design begins, which reduces dependence of the final result on designer experience and frees the designer to focus on specifying a useful problem.
- **Credibility**: The process converges to solving the continuous shape optimization problem, instead of being restricted by the initial parameterization. The convergence of the objective with respect to shape control refinement allows assessment of proximity to the continuous optimum.
- **Feedback**: The emergent parameterization refinement pattern can convey useful information about the design problem.

Some important results of the present work are that

- The adaptive system is able to automatically discover the parameters necessary to solve a problem, without user intervention (Sections 5.1 to 5.3).

- The process is able to robustly drive the shape towards a *continuous* local optimum (Sections 5.2 and 5.3).
- Compared to using all the design variables up front, progressive parameterization accelerates the rate of design improvement (as much as $3\times$ in some cases, see especially Sections 6.1 and 6.3).
- Progressive parameterization appears to lead to smoother and more reasonable design trajectories, improving the robustness of optimization (see Sections 5.3 and 6.3). This also means that the optimum of each suboptimization level is more qualitatively reasonable, so the overall process can be safely terminated early.
- Compared to previous approaches, the new refinement indicator makes substantially better predictions of the most important shape control, especially in cases of multiple shape control classes (Section 5.1) or in the presence of important constraints (Section 6.4). For unconstrained design with single classes of shape control (Sections 5.2 and 5.3), it appears that a first-order indicator is sufficient.

My implementation involves six parameters for tuning the adaptation strategy[1]. The two with biggest performance impact are the cutoffs r for the trigger and for the auto-growth strategy. Carefully selected strategies enabled the adaptive approach to solidly outperform uniform refinement. In all cases, I was able to identify reasonable defaults that tend to lead to good performance — an experienced user can adjust these to maximize performance.

While the studies presented use an in-house discrete geometry modeler, the implementation is specifically architected to work with any geometry modeler that meets certain requirements, outlined in Chapter 4. Although some modification or wrapping of the modeler is required, the computational acceleration and reduction in setup time strongly justifies this upfront expenditure.

7.1 Future Work

Discovering Important Classes of Shape Control

In this work, the class or basis of shape control (e.g. fuselage radius, wing thickness, sweep, twist, etc.) was always specified. The adaptive system refines the resolution and

[1]d in GETCANDIDATES$(\cdot)$; r and w in the TRIGGER$(\cdot)$; N_w, r and w in ADAPTSHAPECONTROL$(\cdot)$

distribution within the given class. We might, however, consider looking beyond designer-specified shape control classes and explore automatically determining a more effective basis for optimization. This is a much harder problem algorithmically. It is not clear what constitutes a valid shape control basis, how the effectiveness of a basis would be measured, or how would one search the "space" of all shape control bases. However, the promise is staggering; the most efficient possible shape control basis can always solve the problem with one degree of freedom: the vector between the initial design and the optimal design.

Continuous Redistribution

A more tractable area for improvement involves moving outside the discrete refinement structure I impose in this work, and allowing continuous optimal redistribution (*r*-refinement) of the shape parameter locations. Some preliminary work in this area has been done [58, 77]. In general, however, *r*-refinement must be combined with *h*-refinement to guarantee convergence to the continuous optimum.

Preconditioning

For the cases considered, the optimizer was not given scaling information. In future work, the prolongation operator from Section 3.3 might be used to precondition each suboptimization, using Hessian information from the previous search space. In theory, this would accelerate the overall process by removing the partially redundant initial Hessian construction at each level. It would be interesting to compare a well-preconditioned static optimization problem to unpreconditioned (and preconditioned) progressive parameterization. Chaigne and Désidéri note that the two methods are mathematically related [79]. It is not clear whether the acceleration provided by the two methods is additive or redundant.

Generalization of Hessian Scaling

The second-order indicator is demonstrably more accurate, yet it depends on the nature of the objective function, including the governing equations. For each new objective, the Hessian must be analyzed. I derived a second-order indicator for two objective functions: geometric shape matching and pressure matching. While the latter appeared to work reasonably for a range of pressure-based functionals, it would be desirable to place the approach on firmer footing through derivation of scale-dependent terms for common functionals. Fortunately, there is a growing body of literature on this topic [89, 112, 113, 167–170].

Feasibility Constraints

In this approach, the designer no longer dictates exactly how the shape is permitted to be modified. There is consequently *more* need to use explicit constraints to preclude undesirable modifications that are otherwise invisible to the objective function. Without such explicit constraints, the adaptation procedure is likely to exploit weaknesses in the problem formulation. Although this could become a reactionary hole-plugging task, it is arguably more directly relevant to design than constructing an arbitrary parameterization. In principle it should be possible to codify and automate the specification of some common constraints, such as non-self-intersection and smoothness.

Cost-Benefit Analysis

My approach attributes all difficulty with navigation to the number of design variables (more is slower). This is evident in the selection of an automatic growth rate, that treats each additional parameter as having some effective cost, regardless of the context. Just as some parameters offer more potential than others, it is reasonable to surmise that some parameters are more quickly exploitable than others.

Adaptive Procedure

Despite not requiring any PDE solutions, the $\mathcal{O}(N_{DV}^2)$ scaling of the adaptation procedure can still become prohibitive for very large design spaces. One possibility for further reducing the cost is to reuse information from the refinement procedure during the previous adaptation cycle to reduce the number of candidates being considered in the current cycle.

Singularities

Optimization can be affected by flow singularities, e.g. at airfoil trailing edges and at shocks. Using sensitivity-driven shape control adaptation means that these singularities could also impact the evolution of the parameterization, although I did not witness any especially problematic cases. Simple solutions include geometrically constraining the trailing edge (as I do in this work) or suppressing sensitivities at known singularities [171].

7.2 Potential Applications

I have examined adaptive shape control within a relatively narrow context: aerodynamic design of discrete geometries with an adjoint-driven, gradient-based optimizer. However, adaptive parameterization offers much broader potential. Here I briefly comment on how it might improve design approaches in other settings.

Alternative Analyses and Multidisciplinary Design

Adaptive parameterization directly applies to a wide class of PDE-constrained shape optimization problems, including viscous flow, unsteady problems, structural optimization, and acoustic design. In fact, the more expensive the objective and constraint evaluations, the greater the impact it could have, assuming each analysis component has an adjoint. Multidisciplinary problems are also perfectly suited to adaptive parameterization. Because multidisciplinary problems tend to involve many classes of design variables, it is likely that the second-order indicator would prove critical for determining which variables are most important to add.

Multifidelity Analysis

Flow meshing and shape parameterization can be viewed as two orthogonal means to trade cost with accuracy. In each case, substantial acceleration can be gained by moving from a static resolution approach to a progressive approach. Higher resolution (and higher cost) are introduced only when necessary to further improve the design. In each case, output-based adaptation can be used to selectively focus computational effort on the relevant aspects of the problem. In [94], we combined the two, using a crude approach where the analysis fidelity and shape control were refined simultaneously, with manually-specified error tolerances and growth rates. For uniformly refined shape parameterizations, this proved quite successful, but some major enticing questions remain. For instance, how might the system know whether it is limited more by insufficient mesh resolution or by insufficient shape control? (Both are forms of discretization error.) How much can optimization be accelerated by a combination of the two adaptive strategies? There has been some work this direction. For example, [59] uses error estimation to combine mesh adaptation with shape control adaptation, though only for simple model problems.

Multi-Objective Optimization

Most approaches to constructing multi-objective Pareto-optimal fronts use a fixed, finite search space. An adaptive parameterization system could construct a Pareto-optimal front in the *continuous* shape design space. There are at least two ways this could be done. One approach would be to create a sample of single-objective scalarizations (weighted sums) of the competing objectives and perform completely independent optimizations for each one.[2] With this method, each independent "agent" would be driving closer and closer to one point on the continuously optimal front, independently adapting its own parameterization to get there. The advantage of this approach is that it would work directly with my existing adaptive system with no modification.

An alternative approach would be to build a sequence of Pareto fronts in progressively finer search spaces. Between each level, the parameterization would be adapted identically for all agents, so that each Pareto front would correspond to an internally consistent search space. The indicator I developed targets reduction of a single objective. In this case, if the goal is to obtain an accurate representation of the *entire* Pareto front, one should define a new refinement indicator that aims to reduce the average error between the approximate Pareto front and the Pareto front in the continuous space.

Shape Exploration and Well-Spanning Parameters

Much like output-based flow mesh adaptation, this approach seeks an effective parameterization to solve one particular optimization problem. This adapted parameterization will not likely be appropriate for other problems, perhaps not even for similar ones. Some situations require the solution of a large number of similar optimization problems. For example, consider a conceptual, multi-disciplinary aircraft design tool where airfoil optimization is treated as a sub-optimization. At each design iteration, the objectives and constraints may be different — the ideal parameterization is likely to vary as well. Rather than adapt the parameterization for each iteration, it might be more efficient to determine a single compact search space in which to solve the whole range of related problems that arise. This would also be useful for interactive shape exploration [172]. In general, solving broad classes of problems will likely require many more design variables than single-target optimization[3]. Nevertheless, upfront cost at finding a compact parameterization would be amortized over the many invocations of the optimization. For this type of situation, it would make sense

[2]There are well known caveats to this approach. It does not traverse non-convex fronts, and evenly-sampled input weights are not likely to evenly cover the output Pareto front.

[3]For example, Masters *et al.* show that achieving even 80% coverage of airfoil databases to sufficient geometric accuracy requires a great many design variables, regardless of the parameterization technique [173].

to use a different importance indicator that maximizes the ability to span a broad range of the search space, such as the approach in [58].

Modelers with Fixed Parameterizations

In this work, I focus on shape control that can, in theory, be refined without limit, allowing arbitrarily close approximation of the continuous problem. Progressive parameterization may, however, also be useful in situations with a finite, fixed set of design variables, for example when a designer has no access to the geometry modeler or authority to change the parameterization. In this case, progressive parameterization amounts to applying a series of dimension-reducing transformations to an original, large set of design variables, similar to the approach used in Hwang and Martins [81]. This approach has some drawbacks which were discussed in Section 1.2.5. Progressive parameterization is most effective when the geometry modeler is itself imbued with the ability to modify the parameterization.

For computational benefits to be reaped from a progressive parameterization approach, coarser parameterizations must be reflective of the full design space. Coarser parameterizations can be defined by linking the parameters, or by simply optimizing larger and larger subsets of them. As demonstrated in Section 2.1.2, however, naive linking methods do not always work.

Other Optimization Approaches

This work focused on accelerating optimization approaches based on reduced-space, quasi-Newton optimizers, the current workhorse for aerodynamic design. Quasi-Newton optimization is unique in that both (1) its asymptotic scaling with respect to N_{DV} is reasonable but non-constant (typically linear), and (2) it reveals Hessian information that can be leveraged to adaptively focus the search, reducing the number of design variables required. In this same vein, adaptive parameterization might help accelerate other reduced-space optimization approaches with superior but non-negligible scaling, such as inexact-Newton-Krylov methods [6, 46]. For full-space (one-shot) methods [30, 33, 45, 174], which have little or no dependence on the resolution of the shape control, the *computational* benefits of reducing the search space might not be appreciable. However, the adaptive refinement pattern would still inform the designer about which shape control is most essential for achieving the specific design goals.

For non-adjoint-based approaches (e.g. finite difference-based frameworks or gradient-free optimization approaches), it is paramount to minimize the number of design variables. In these cases, a progressive parameterization approach would tremendously effective.

However, with uniform refinement N_{DV} grows far too fast for non-adjoint-based optimization. While this would suggest that a targeted, problem-adaptive approach would be extremely valuable in this context, my approach to goal-oriented adaptive refinement relies on adjoint sensitivity information. One intriguing possibility would be to use a lower-fidelity, adjoint-based tool as a "pilot" for a higher-fidelity optimization tool without an adjoint solver. The lower-fidelity tool might determine an effective, compact shape parameterization, which would then be used by the higher-fidelity tool. However, since the ideal parameterization depends on the objective function, the low- and high-fidelity frameworks should be suitably consistent.

Bibliography

[1] Sobester, A. and Barrett, T., "Quest for a Truly Parsimonious Airfoil Parameterization Scheme," *The 26th Congress of ICAS and 8th AIAA ATIO*, 2008.

[2] Lukaczyk, T. W., Constantine, P., Palacios, F., and Alonso, J. J., "Active Subspaces for Shape Optimization," *10th AIAA Multidisciplinary Design Optimization Conference*, 2014.

[3] Sóbester, A. and Powell, S., "Design Space Dimensionality Reduction Through Physics-Based Geometry Re-Parameterization," *Optimization and Engineering*, Vol. 14, No. 1, 2013, pp. 37–59.

[4] Poole, D. J., Allen, C. B., and Rendall, T. C. S., "Metric-Based Mathematical Derivation of Efficient Airfoil Design Variables," *AIAA Journal*, Vol. 53, No. 5, 2015, pp. 1349–1361.

[5] Lyu, Z., Kenway, G. K. W., and Martins, J. R. R. A., "Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark," *AIAA Journal*, 2014.

[6] Dener, A., Kenway, G. K., Hicken, J. E., and Martins, J., "Comparison of Inexact- and Quasi-Newton Algorithms for Aerodynamic Shape Optimization," *53rd AIAA Aerospace Sciences Meeting*, 2015.

[7] Hicks, R. M. and Henne, P. A., "Wing Design by Numerical Optimization," *J. Aircraft*, Vol. 15, No. 7, July 1978.

[8] Brock, W., Burdyshaw, C., Karman, S., Betro, V., Hilbert, B., Anderson, K., and Haimes, R., "Adjoint-Based Design Optimization Using CAD Parameterization Through CAPRI," *50th AIAA Aerospace Sciences Meeting*, 2012.

[9] Haimes, R. and Drela, M., "On The Construction of Aircraft Conceptual Geometry for High-Fidelity Analysis and Design," *50th AIAA Aerospace Sciences Meeting*, 2012.

[10] Dannenhoffer, J., "OpenCSM: An Open-Source Constructive Solid Modeler for MDAO," *51st AIAA Aerospace Sciences Meeting*, 2013.

[11] Rodriguez, D. L. and Sturdza, P., "A Rapid Geometry Engine for Preliminary Aircraft Design," *44th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 2006.

[12] Hahn, A., "Vehicle Sketch Pad: A Parametric Geometry Modeler for Conceptual Aircraft Design," *48th AIAA Aerospace Sciences Meeting*, 2010.

[13] Risse, K., Anton, E., Lammering, T., Franz, K., and Hoernschemeyer, R., "An Integrated Environment for Preliminary Aircraft Design and Optimization," *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2012.

[14] Wu, H.-Y., Yang, S., Liu, F., and Tsai, H.-M., "Comparison of Three Geometric Representations of Airfoils for Aerodynamic Optimization," *16th AIAA Computational Fluid Dynamics Conference*, Orlando, FL, June 2003.

[15] Morris, A. M., Allen, C. B., and Rendall, T. C. S., "High-Fidelity Aerodynamic Shape Optimization of Modern Transport Wing using Efficient Hierarchical Parameterization," *International Journal for Numerical Methods in Fluids*, Vol. 63, No. 3, 2010, pp. 297–312.

[16] Sherar, P. A., Thompson, C. P., Xu, B., and Zhong, B., "A Novel Shape Optimization Method using Knot Insertion Algorithm in B-spline and its Application to Transonic Airfoil Design," *Scientific Research and Essays*, Vol. 6, No. 27, November 2011, pp. 5696–5707.

[17] Wintzer, M. and Ordaz, I., "Under-Track CFD-Based Shape Optimization for a Low-Boom Demonstrator Concept," *33rd AIAA Applied Aerodynamics Conference*, 2015.

[18] Bryson, A. E. and Ho, Y. C., *Applied Optimal Control*, Hemisphere, New York, 1975.

[19] Lions, J. L., *Optimal Control of Systems Governed by Partial Differential Equations*, Springer-Verlag, Berlin, 1971.

[20] Errico, R. M., "What Is an Adjoint Model?" *Bulletin of the American Meteorological Society*, Vol. 78, No. 11, 1997, pp. 2577–2591.

[21] Plessix, R. E., "A Review of the Adjoint-State Method for Computing the Gradient of a Functional with Geophysical Applications," *Geophysical Journal International*, Vol. 167, No. 2, 2006, pp. 495–503.

[22] Skiba, Y., "Direct and Adjoint Oil Spill Estimates," *Environmental Monitoring and Assessment*, Vol. 59, No. 1, 1999, pp. 95–109.

[23] Christensen, P. H., "Adjoints and Importance in Rendering: An Overview," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 9, No. 3, July 2003, pp. 329–340.

[24] McNamara, A., Treuille, A., Popović, Z., and Stam, J., "Fluid Control Using the Adjoint Method," *ACM Trans. Graph.*, Vol. 23, No. 3, Aug. 2004, pp. 449–456.

[25] Giles, M. and Glasserman, P., "Smoking Adjoints: Fast Monte Carlo Greeks," *Risk Magazine*, 2006.

[26] Pironneau, O., "On Optimum Profiles in Stokes Flow," *Journal of Fluid Mechanics*, Vol. 59, 1973, pp. 117–128.

[27] Pironneau, O., "On Optimum Design in Fluid Mechanics," *Journal of Fluid Mechanics*, Vol. 64, 1974, pp. 97–110.

[28] Pironneau, O., "Optimal shape design for elliptic systems," *System Modeling and Optimization*, edited by R. Drenick and F. Kozin, Vol. 38 of *Lecture Notes in Control and Information Sciences*, Springer Berlin Heidelberg, 1982, pp. 42–66.

[29] Haug, E. J. and Arora, J. S., "Design Sensitivity Analysis of Elastic Mechanical Systems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 15, No. 1, 7 1978, pp. 35–62.

[30] Jameson, A., "Aerodynamic Design via Control Theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988.

[31] Jameson, A., Pierce, N., and Martinelli, L., "Optimum Aerodynamic Design Using the Navier-Stokes Equations," *35th Aerospace Sciences Meeting and Exhibit*, 1997.

[32] Giles, M. B. and Pierce, N. A., "An Introduction to the Adjoint Approach to Design," *Flow, Turbulence and Combustion*, Vol. 65, No. 3-4, 2000, pp. 393–415.

[33] Ta'asan, S., Kuruvila, G., and Salas, M., "Aerodynamic Design and Optimization in One Shot," *30th Aerospace Sciences Meeting and Exhibit*, 1992.

[34] Borggaard, J., Burkardt, J., Gunzburger, M., Peterson, J., Arian, E., and Ta'asan, S., "Progress in Systems and Control Theory," *Optimal Design and Control*, Vol. 19, Birkhäuser Boston, 1995, pp. 23–40.

[35] Hazra, S. and Schulz, V., "Simultaneous Pseudo-Timestepping for PDE-Model Based Optimization Problems," *BIT Numerical Mathematics*, Vol. 44, No. 3, 2004, pp. 457–472.

[36] Anderson, W. K. and Venkatakrishnan, V., "Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation," *Computers and Fluids*, Vol. 28, No. 4–5, 1999, pp. 443–480.

[37] Nielsen, E. J. and Diskin, B., "Discrete Adjoint-Based Design for Unsteady Turbulent Flows on Dynamic Overset Unstructured Grids," *AIAA Journal*, Vol. 51, No. 6, 2015/10/26 2013, pp. 1355–1373.

[38] Martins, J., Alonso, J., and Reuther, J., "A Coupled-Adjoint Sensitivity Analysis Method for High-Fidelity Aero-Structural Design," *Optimization and Engineering*, Vol. 6, No. 1, 2005, pp. 33–62.

[39] Fidkowski, K. J. and Darmofal, D. L., "Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics," *AIAA Journal*, Vol. 49, No. 4, 2011, pp. 673–694.

[40] Liu, D. C. and Nocedal, J., "On the Limited Memory BFGS Method for Large Scale Optimization," *Mathematical Programming*, Vol. 45, No. 1-3, 1989, pp. 503–528.

[41] Nocedal, J. and Wright, S. J., *Numerical Optimization*, Springer, New York, 2nd ed., 2006.

[42] Zingg, D. W., Nemec, M., and Pulliam, T. H., "A Comparative Evaluation of Genetic and Gradient-Based Algorithms Applied to Aerodynamic Optimization," *European Journal of Computational Mechanics*, Vol. 17, 2008, pp. 103–126.

[43] Lyu, Z., Xu, Z., and Martins, J. R. R. A., "Benchmarking Optimization Algorithms for Wing Aerodynamic Design Optimization," *8th International Conference on Computational Fluid Dynamics (ICCFD8)*, Chengdu, China, July 2014, ICCFD8 2014-0203.

[44] Conn, A., Scheinberg, K., and Vicente, L., *Introduction to Derivative-Free Optimization*, Society for Industrial and Applied Mathematics, 2009.

[45] Biros, G. and Ghattas, O., "Parallel Lagrange–Newton–Krylov–Schur Methods for PDE-Constrained Optimization. Part I: The Krylov–Schur Solver," *SIAM Journal on Scientific Computing*, Vol. 27, No. 2, 2005, pp. 687–713.

[46] Hicken, J. and Alonso, J., "Comparison of Reduced- and Full-space Algorithms for PDE-constrained Optimization," *51st AIAA Aerospace Sciences Meeting*, 2013.

[47] Désidéri, J.-A., Majd, B. A. E., and Janka, A., "Nested and Self-Adaptive Bezier Parameterizations for Shape Optimization," *Journal of Computational Physics*, Vol. 224, 2007, pp. 117–131.

[48] Martinelli, M. and Beux, F., "Multi-Level Gradient-Based Methods and Parametrization in Aerodynamic Shape Design," *European Journal of Computational Mechanics*, Vol. 17, No. 1-2, 2008, pp. 169–197.

[49] Hicken, J. E. and Zingg, D. W., "Induced-Drag Minimization of Nonplanar Geometries Based on the Euler Equations," *AIAA Journal*, Vol. 48, No. 11, 2010, pp. 2564–2575.

[50] Han, X. and Zingg, D., "An Adaptive Geometry Parametrization for Aerodynamic Shape Optimization," *Optimization and Engineering*, Vol. 15, No. 1, 2013, pp. 69–91.

[51] Bisson, F., Nadarajah, S., and Shi-Dong, D., "Adjoint-Based Aerodynamic Optimization of Benchmark Problems," *52nd Aerospace Sciences Meeting*, National Harbor, MD, January 2014.

[52] Telidetzki, K., Osusky, L., and Zingg, D. W., "Application of Jetstream to a Suite of Aerodynamic Shape Optimization Problems," *52nd Aerospace Sciences Meeting*, National Harbor, MD, January 2014.

[53] Poole, D. J., Allen, C. B., and Rendall, T. C. S., "Application of Control Point-Based Aerodynamic Shape Optimization to Two-Dimensional Drag Minimization," *52nd Aerospace Sciences Meeting*, National Harbor, MD, January 2014.

[54] Amoignon, O., Navratil, J., and Hradil, J., "Study of Parameterizations in the Project CEDESA," *52nd Aerospace Sciences Meeting*, National Harbor, MD, January 2014.

[55] Vassberg, J. and Jameson, A., "Influence of Shape Parameterization on Aerodynamic Shape Optimization," *Lectures at the Von Karman Institute*, April 2014.

[56] Méheut, M., Destarac, D., Carrier, G., Anderson, G. R., Nadarajah, S., Poole, D., Vassberg, J., and Zingg, D. W., "Gradient-Based Single and Multi-point Aerodynamic Optimizations with the elsA Software," *53rd AIAA Aerospace Sciences Meeting*, Kissimmee, FL, January 2015.

[57] Musialski, P., Auzinger, T., Birsak, M., Wimmer, M., and Kobbelt, L., "Reduced-order Shape Optimization Using Offset Surfaces," *ACM Trans. Graph.*, Vol. 34, No. 4, July 2015, pp. 102:1–102:9.

[58] Poole, D. J., Allen, C. B., and Rendall, T., "Optimal Domain Element Shapes for Free-Form Aerodynamic Shape Control," *53rd AIAA Aerospace Sciences Meeting*, 2015.

[59] Becker, R., "Estimating the Control Error in Discretized PDE-constrained Optimization," *Journal of Numerical Mathematics*, March 2006, pp. 163–185.

[60] Benedix, O. and Vexler, B., "A Posteriori Error Estimation and Adaptivity for Elliptic Optimal Control Problems with State Constraints," *Computational Optimization and Applications*, Vol. 44, No. 1, 2009, pp. 3–25.

[61] Toal, D. J. J., Bressloff, N. W., Keane, A. J., and Holden, C. M. E., "Geometric Filtration Using Proper Orthogonal Decomposition for Aerodynamic Design Optimization," *AIAA Journal*, Vol. 48, No. 5, 2010, pp. 916–928.

[62] Ghoman, S., Wang, Z., Chen, P., and Kapania, R., "A POD-based Reduced Order Design Scheme for Shape Optimization of Air Vehicles," *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2012.

[63] Ghisu, T., Parks, G., Jarrett, J., and Clarkson, P., "Accelerating Design Optimization via Principal Components Analysis," *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2008.

[64] Simpson, T., Mistree, F., Korte, J., and Mauery, T., "Comparison of Response Surface and Kriging Models for Multidisciplinary Design Optimization," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1998.

[65] Chung, H. S. and Alonso, J., "Design of a Low-Boom Supersonic Business Jet Using Cokriging Approximation Models," *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2002.

[66] Han, Z., Zimmerman, R., and Görtz, S., "Alternative Cokriging Method for Variable-Fidelity Surrogate Modeling," *AIAA Journal*, Vol. 50, No. 5, 2012, pp. 1205–1210.

[67] Yamazaki, W. and Mavriplis, D. J., "Derivative-Enhanced Variable Fidelity Surrogate Modeling for Aerodynamic Functions," *AIAA Journal*, Vol. 51, No. 1, 2013, pp. 126–137.

[68] Bennett, J. A. and Botkin, M. E., "Structural Shape Optimization with Geometric Description and Adaptive Mesh Refinement," *AIAA Journal*, Vol. 23, No. 3, 1985, pp. 458–464.

[69] Kikuchi, N., Chung, K. Y., Torigaki, T., and Taylor, J. E., "Adaptive Finite Element Methods for Shape Optimization of Linearly Elastic Structures," *The Optimum Shape*, Springer, 1986, pp. 139–169.

[70] Zienkiewicz, O. C., Craig, A. W., Zhu, J. Z., and Gallagher, R. H., "Adaptive Analysis Refinement and Shape Optimization — Some New Possibilities," *General Motors Research Laboratories Symposia Series*, edited by J. A. Bennett and M. E. Botkin, chap. The Optimum Shape, Springer US, 1986, pp. 3–27.

[71] Kohli, H. S. and Carey, G. F., "Shape Optimization using Adaptive Shape Refinement," *International Journal for Numerical Methods in Engineering*, Vol. 36, No. 14, 1993, pp. 2435–2451.

[72] Marco, N. and Beux, F., "Multilevel Optimization: Application to One-Shot Shape Optimum Design," Tech. Rep. Report 2068, INRIA, 1993.

[73] Beux, F. and Dervieux, A., "A Hierarchical Approach for Shape Optimisation," Research Report RR-1868, INRIA, 1993.

[74] Desideri, J.-A. and Janka, A., "Hierarchical Parameterization for Multilevel Evolutionary Shape Optimization with Application to Aerodynamics," *International Congress on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, 2003.

[75] Majd, B. A. E., Duvigneau, R., and Désidéri, J.-A., "Aerodynamic Shape Optimization using a Full and Adaptive Multilevel Algorithm," *ERCOFTAC Design Optimization: Methods and Application*, Gran Canaria, Canaria Island, Spain, April 2006.

[76] Courty, F. and Dervieux, A., "Multilevel Functional Preconditioning for Shape Optimisation," *International Journal of Computational Fluid Dynamics*, Vol. 20, No. 7, 2006, pp. 481–490.

[77] Duvigneau, R., "Adaptive Parameterization using Free-Form Deformation for Aerodynamic Shape Optimization," Tech. Rep. 5949, INRIA, 2006.

[78] Majd, B. A. E., Desideri, J.-A., and Duvigneau, R., "Multilevel Strategies for Parametric Shape Optimization in Aerodynamics," *REMN*, 2008.

[79] Chaigne, B. and Désidéri, J.-A., "Convergence of a Two-Level Ideal Algorithm for a Parametric Shape Optimization Model Problem," Research Report 7068, INRIA, Sophia Antipolis, France, September 2009.

[80] Dubé, J.-F., Guibault, F., Vallet, M.-G., and Trépanier, J.-Y., "Turbine Blade Reconstruction and Optimization using Subdivision Surfaces," *44th AIAA Aerospace Sciences Meeting and Exhibit*, 2006.

[81] Hwang, J. T. and Martins, J. R. R. A., "A Dynamic Parametrization Scheme for Shape Optimization Using Quasi-Newton Methods," *50th AIAA Aerospace Sciences Meeting*, Nashville, TN, January 2012.

[82] Désidéri, J.-A. and Dervieux, A., "Hierarchical Methods for Shape Optimization in Aerodynamics. I: Multilevel Algorithms for Parametric Shape Optimization," *Introduction to Optimization and Multidisciplinary Design*, edited by J. Périaux and H. Deconinck, 2006-3, Von Karman Institute for Fluid Dynamics, March 2006.

[83] Hradil, J., *Adaptive Parameterization For Aerodynamic Shape Optimization In Aeronautical Applications*, Ph.D. thesis, Brno University Of Technology, 2015.

[84] Jameson, A. and Vassberg, J. C., "Studies of Alternative Numerical Optimization Methods Applied to the Brachistochrone Problem," *Computational Fluid Dynamics Journal*, Vol. 9, No. 3, October 2000.

[85] Lewis, R. and Nash, S., "Model Problems for the Multigrid Optimization of Systems Governed by Differential Equations," *SIAM Journal on Scientific Computing*, Vol. 26, No. 6, 2015/06/09 2005, pp. 1811–1837.

[86] Olhofer, M., Jin, Y., and Sendhoff, B., "Adaptive Encoding for Aerodynamic Shape Optimization using Evolution Strategies," *Congress on Evolutionary Computation*, Korea, 2001, pp. 576–583.

[87] Huyse, L., Padula, S. L., Lewis, R. M., and Li, W., "Probabilistic Approach to Free-Form Airfoil Shape Optimization Under Uncertainty," *AIAA Journal*, Vol. 40, No. 9, 2002, pp. 1764–1772.

[88] Croicu, A.-M., Hussaini, M. Y., Jameson, A., and Klopfer, G., "Robust Airfoil Optimization Using Maximum Expected Value and Expected Maximum Value Approaches," *AIAA Journal*, Vol. 50, No. 9, 2012, pp. 1905–1919.

[89] Eppler, K. and Harbrecht, H., "A Regularized Newton Method in Electrical Impedance Tomography using Shape Hessian Information," *Control and Cybernetics*, Vol. 34, No. 1, 2005, pp. 203–225.

[90] Mang, A. and Biros, G., "An Inexact Newton-Krylov Algorithm for Constrained Diffeomorphic Image Registration," *ArXiv e-prints*, Aug. 2014.

[91] Burger, M., "Infinite-Dimensional Optimization and Optimal Design," Lecture Notes, UCLA, course 285J, 2003.

[92] Vassberg, J., Harrison, N., Roman, D., and Jameson, A., "A Systematic Study on the Impact of Dimensionality for a Two-Dimensional Aerodynamic Optimization Model Problem," *29th AIAA Applied Aerodynamics Conference*, 2011.

[93] Nash, S. G. and Lewis, R. M., "Assessing the Performance of an Optimization-Based Multilevel Method," *Optimization Methods and Software*, Vol. 26, No. 4-5, 2011, pp. 693–717.

[94] Anderson, G. R., Aftosmis, M. J., and Nemec, M., "Aerodynamic Shape Optimization Benchmarks with Error Control and Automatic Parameterization," *53rd AIAA Aerospace Sciences Meeting*, Kissimmee, FL, January 2015.

[95] Nemec, M. and Aftosmis, M. J., "Parallel Adjoint Framework for Aerodynamic Shape Optimization of Component-Based Geometry," *49th AIAA Aerospace Sciences Meeting*, Orlando, FL, January 2011.

[96] Nemec, M., Aftosmis, M. J., Murman, S. M., and Pulliam, T. H., "Adjoint Formulation for an Embedded-Boundary Cartesian Method," *43rd AIAA Aerospace Sciences Meeting*, Reno, NV, January 2005.

[97] Nemec, M. and Aftosmis, M. J., "Adjoint Sensitivity Computations for an Embedded-Boundary Cartesian Mesh Method," *J. Comp. Phys.*, Vol. 227, 2008, pp. 2724–2742.

[98] Anderson, G. R. and Aftosmis, M. J., "Adaptive Shape Control for Aerodynamic Design," *53rd AIAA Aerospace Sciences Meeting*, Kissimmee, FL, January 2015.

[99] Nemec, M. and Aftosmis, M. J., "Adjoint Error Estimation and Adaptive Refinement for Embedded-Boundary Cartesian Meshes," *18th AIAA Computational Fluid Dynamics Conference*, Miami, FL, June 2007.

[100] Nemec, M., Aftosmis, M. J., and Wintzer, M., "Adjoint-Based Adaptive Mesh Refinement for Complex Geometries," *46th AIAA Aerospace Sciences Meeting*, Reno, NV, January 2008.

[101] Nemec, M. and Aftosmis, M. J., "Output Error Estimates and Mesh Refinement in Aerodynamic Shape Optimization," *51st AIAA Aerospace Sciences Meeting*, Grapevine, TX, January 2013.

[102] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006.

[103] Hemez, F. M. and Tippetts, T. B., "Successes and Failures of Verifying the Convergence of Discrete Solutions," *IMAC-XXV: Conference and Exposition on Structural Dynamics*, 2007.

[104] Hicken, J. and Alonso, J., "PDE-Constrained Optimization with Error Estimation and Control," *Journal of Computational Physics*, Vol. 263, 2014, pp. 136–150.

[105] Aftosmis, M. J., "Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries," *Lectures at the Von Karman Institute*, March 1997.

[106] Han, X. and Zingg, D. W., "An Evolutionary Geometry Parametrization for Aerodynamic Shape Optimization," *49th AIAA Aerospace Sciences Meeting*, Honolulu, HI, June 2011.

[107] Kuhn, H. W. and Tucker, A. W., "Nonlinear Programming," *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, Berkeley, Calif., 1951, pp. 481–492.

[108] Kulfan, B. M., "Universal Parametric Geometry Representation Method," *J. Aircraft*, Vol. 45, No. 1, January 2008, pp. 142–158.

[109] Jones, E., Oliphant, T., and Peterson, P., "Scipy: Open Source Scientific Tools for Python," 2001–.

[110] Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic Press Inc., 1981.

[111] Han, X., *An Evolutionary Geometry Parametrization for Aerodynamic Shape Optimization*, Master's thesis, University of Toronto, 2011.

[112] Arian, E. and Ta'asan, S., "Analysis of the Hessian for Aerodynamic Optimization: Inviscid Flow," *Computers and Fluids*, Vol. 28, No. 7, 1999, pp. 853–877.

[113] Telib, H., Arian, E., and Iollo, A., "The Effect of Shocks on Second Order Sensitivities for the Quasi-One-Dimensional Euler Equations," *Journal of Computational Physics*, Vol. 230, No. 23, 2011, pp. 8603 – 8618.

[114] Kallinderis, Y. G. and Baron, J. R., "Adaptation Methods for a New Navier-Stokes Algorithm," *AIAA Journal*, Vol. 27, No. 1, 1989, pp. 37–43.

[115] Aftosmis, M. J. and Berger, M. J., "Multilevel Error Estimation and Adaptive h-Refinement for Cartesian Meshes with Embedded Boundaries," *40th AIAA Aerospace Sciences Meeting*, Reno, NV, January 2002.

[116] Blum, C. and Roli, A., "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," *ACM Comput. Surv.*, Vol. 35, No. 3, Sept. 2003, pp. 268–308.

[117] Fudge, D., Zingg, D., and Haimes, R., "A CAD-Free and a CAD-Based Geometry Control System for Aerodynamic Shape Optimization," *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005.

[118] Hicken, J. E. and Zingg, D. W., "Aerodynamic Optimization Algorithm with Integrated Geometry Parameterization and Mesh Movement," *AIAA Journal*, Vol. 48, No. 2, 2010, pp. 400–413.

[119] Hwang, J. T., Kenway, G. K. W., and Martins, J. R. R. A., "Geometry and Structural Modeling for High-Fidelity Aircraft Conceptual Design Optimization," *Proceedings of the 15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Atlanta, GA, June 2014.

[120] Nemec, M., Aftosmis, M. J., and Pulliam, T. H., "CAD-Based Aerodynamic Design of Complex Configurations Using a Cartesian Method," *AIAA*, , No. 0113, January 2004.

[121] Nemec, M. and Aftosmis, M., "Aerodynamic Shape Optimization Using a Cartesian Adjoint Method and CAD Geometry," *24th AIAA Applied Aerodynamics Conference*, 2006.

[122] Martín, M., Andrés, E., Widhalm, M., Bitrián, P., and Lozano, C., "CAD-Based Aerodynamic Shape Design Optimization With The DLR Tau Code," *27th International Congress Of The Aeronautical Sciences (ICAS)*, 2010.

[123] Thompson, P. M., Robinson, T. T., and Armstrong, C., "Efficient CAD-based Aerodynamic Design Optimization with Adjoint CFD Data," *21st AIAA Computational Fluid Dynamics Conference*, 2013.

[124] Xu, S., Jahn, W., and Müller, J.-D., "CAD-based shape optimisation with CFD using a discrete adjoint," *International Journal for Numerical Methods in Fluids*, Vol. 74, No. 3, 2014, pp. 153–168.

[125] Cliff, S. E., Thomas, S. D., and Hawke, V. M., "Swing-Wing Inline-Fuselage Transport Design Studies at Supersonic Flight Conditions," *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, Hilton Head, SC, September 2009.

[126] Gain, J. and Bechmann, D., "A Survey of Spatial Deformation from a User-Centered Perspective," *ACM Transactions on Graphics*, Vol. 27, No. 4, October 2008.

[127] Sederberg, T. W. and Parry, S. R., "Free-Form Deformation of Solid Geometric Models," *ACM SIGGRAPH*, Vol. 20, No. 4, August 1986.

[128] Samareh, J. A., "Multidisciplinary Aerodynamic-Structural Shape Optimization using Deformation (MASSOUD)," *8th AIAA Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, September 2000.

[129] Samareh, J. A., "Aerodynamic Shape Optimization Based on Free-Form Deformation," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY, July 2004.

[130] Yamazaki, W., Mouton, S., and Carrier, G., "Geometry Parameterization and Computational Mesh Deformation by Physics-Based Direct Manipulation Approaches," *AIAA Journal*, Vol. 48, No. 8, August 2010, pp. 1817–1832.

[131] Kenway, G. K. W., 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, September 2010, AIAA 2010-9231.

[132] Anderson, G. R., Aftosmis, M. J., and Nemec, M., "Parametric Deformation of Discrete Geometry for Aerodynamic Shape Design," *50th AIAA Aerospace Sciences Meeting*, Nashville, TN, January 2012.

[133] Li, J., Gao, Z., Huang, J., and Zhao, K., "Aerodynamic design optimization of nacelle/pylon position on an aircraft," *Chinese Journal of Aeronautics*, Vol. 26, No. 4, 2013, pp. 850 – 857.

[134] Palacios, F., Economon, T. D., Wendorff, A. D., and Alonso, J. J., "Large-scale Aircraft Design using SU2," *53rd AIAA Aerospace Sciences Meeting*, Kissimmee, FL, January 2015.

[135] Anderson, W. K., Karman, S. L., and Burdyshaw, C., "Geometry Parameterization Method for Multidisciplinary Applications," *AIAA Journal*, Vol. 47, No. 6, 2009, pp. 1568–1578.

[136] Berkenstock, D. C. and Aftosmis, M. J., "Structure-Preserving Parametric Deformation of Legacy Geometry," *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, BC, Canada, September 2008.

[137] Persson, P.-O., Aftosmis, M. J., and Haimes, R., "On the Use of Loop Subdivision Surfaces for Surrogate Geometry," *15th Annual Meshing Roundtable*, October 2007.

[138] Jakobsson, S. and Amoignon, O., "Mesh Deformation using Radial Basis Functions for Gradient-based Aerodynamic Shape Optimization," *Computers and Fluids*, Vol. 36, No. 6, July 2007, pp. 1119–1136.

[139] Morris, A. M., Allen, C. B., and Rendall, T. C. S., "Domain-Element Method for Aerodynamic Shape Optimization Applied to a Modern Transport Wing," *AIAA Journal*, Vol. 47, No. 7, 2009.

[140] Rendall, T. C. S. and Allen, C. B., "Unified Fluid-Structure Interpolation and Mesh Motion using Radial Basis Functions," *Int. J. Numer. Meth. Eng.*, Vol. 74, 2008, pp. 1519–1559.

[141] Allen, C. and Rendall, T. S., "CFD-based optimization of hovering rotors using radial basis functions for shape parameterization and mesh deformation," *Optimization and Engineering*, Vol. 14, No. 1, 2013, pp. 97–118.

[142] Anderson, G. R., Aftosmis, M. J., and Nemec, M., "Constraint-Based Shape Parameterization for Aerodynamic Design," *7th International Conference on Computational Fluid Dynamics*, Big Island, Hawaii, July 2012.

[143] Olhofer, M., Bihrer, T., Menzel, S., Fischer, M., and Sendhoff, B., "Evolutionary Optimisation of an Exhaust Flow Element with Free Form Deformation," *4th European Automotive Simulation Conference*, edited by K. Seibert, ANSYS, Munich, Germany, July 2009.

[144] Sobieczky, H., "Parametric Airfoils and Wings," *Recent Development of Aerodynamic Design Methodologies*, edited by K. Fujii and G. S. Dulikravich, Vol. 65 of *Notes on Numerical Fluid Mechanics (NNFM)*, Vieweg+Teubner Verlag, 1999, pp. 71–87.

[145] Aidala, P., W. Davis, J., and Mason, W., "Smart Aerodynamic Optimization," *Applied Aerodynamics Conference*, Danvers, Massachusetts, 1983.

[146] Samareh, J. A., "A Survey of Shape Parameterization Techniques," *CEAS/AIAA/ICASE/NASA Langley Int'l Forum on Aeroelasticity and Structural Dynamics*, June 1999, pp. 333–344.

[147] Samareh, J. A., "Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization," *AIAA Journal*, Vol. 39, No. 5, May 2001.

[148] Castonguay, P. and Nadarajah, S., "Effect of Shape Parameterization on Aerodynamic Shape Optimization," *45th AIAA Aerospace Sciences Meeting and Exhibit*, 2007.

[149] Mousavi, A., Nadarajah, S., and Castonguay, P., "Survey of Shape Parameterization Techniques and its Effect on Three-Dimensional Aerodynamic Shape Optimization," *18th AIAA Computational Fluid Dynamics Conference*, 2007.

[150] Sripawadkul, V., Padulo, M., and Guenov, M., "A Comparison of Airfoil Shape Parameterization Techniques for Early Design Optimization," *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, 2010.

[151] Straathof, M. H., *Shape Parameterization in Aircraft Design: A Novel Method, Based on B-Splines*, Ph.D. thesis, Technische Universiteit Delft, Netherlands, February 2012.

[152] Sobester, A., "Four Suggestions for Better Parametric Geometries," *10th AIAA Multidisciplinary Design Optimization Conference*, 2014.

[153] Drela, M., "Pros and Cons of Airfoil Optimization," *Frontiers of Computational Fluid Dynamics 1998*, edited by E. Murman and D. Caughey, World Scientific, 1998.

[154] Rodriguez, D. L., Aftosmis, M. J., Nemec, M., and Smith, S. C., "Static Aeroelastic Analysis with an Inviscid Cartesian Method," *52nd Aerospace Sciences Meeting*, National Harbor, MD, January 2014.

[155] Rodriguez, D. L., Aftosmis, M. J., Nemec, M., and Anderson, G. R., "Optimized Off-Design Performance of Flexible Wings with Continuous Trailing-Edge Flaps," *53rd AIAA Aerospace Sciences Meeting*, Kissimmee, FL, January 2015.

[156] Wintzer, M., Kroo, I., Aftosmis, M. J., and Nemec, M., "Conceptual Design of Low Sonic Boom Aircraft Using Adjoint-Based CFD," *7th International Conference on Computational Fluid Dynamics (ICCFD7)*, July 2012.

[157] Aftosmis, M. J., Nemec, M., and Cliff, S. E., "Adjoint-Based Low-Boom Design with Cart3D," *AIAA Paper 2011-3500*, Honolulu, HI, June 2011.

[158] Darden, C. M., "Sonic-Boom Minimization with Nose-Bluntness Relaxation," Tech. Pub. NASA-TP 1348, NASA, January 1979.

[159] George, A. R. and Seebass, R., "Sonic Boom Minimization Including Both Front and Rear Shocks," *AIAA Journal*, Vol. 9, No. 10, 1971, pp. 2091–2093.

[160] Morgenstern, J., Norstrud, N., Sokhey, J., Martens, S., and Alonso, J. J., "Advanced Concept Studies for Supersonic Commercial Transports Entering Service in the 2018 to 2020 Period," Contractor Report 2013-217820, NASA, February 2013.

[161] Park, M. A. and Morgenstern, J. M., "Summary and Statistical Analysis of the First AIAA Sonic Boom Prediction Workshop," *32nd AIAA Applied Aerodynamics Conference*, Atlanta, GA, June 2014.

[162] Carrier, G., Destarac, D., Dumont, A., Meheut, M., Salah El Din, I., Peter, J., Khelil, S. B., Brezillon, J., and Pestana, M., "Gradient-Based Aerodynamic Optimization with the elsA Software," *52nd Aerospace Sciences Meeting*, National Harbor, MD, January 2014.

[163] Lyu, Z., Kenway, G. K. W., and Martins, J. R. R. A., "RANS-based Aerodynamic Shape Optimization Investigations of the Common Research Model Wing," *AIAA Journal*, 2014.

[164] Wintzer, M., "Span Efficiency Prediction Using Adjoint-Driven Mesh Refinement," *Journal of Aircraft*, Vol. 47, No. 4, July-August 2010, pp. 1468–1470.

[165] Gundlach, J. F., Tétrault, P.-A., Gern, F. H., Nagshineh-Pour, A. H., Ko, A., Schetz, J. A., Mason, W. H., Kapania, R. K., and Grossman, B., "Conceptual Design Studies of a Strut-Braced Wing Transonic Transport," *Journal of Aircraft*, Vol. 37, No. 6, 2015/08/24 2000, pp. 976–983.

[166] Chakraborty, I., Nam, T., Gross, J. R., Mavris, D. N., Schetz, J. A., and Kapania, R. K., "Comparative Assessment of Strut-Braced and Truss-Braced Wing Configurations Using Multidisciplinary Design Optimization," *Journal of Aircraft*, 2015, pp. 1–12.

[167] Eppler, K., Schmidt, S., Schulz, V., and Ilic, C., "Preconditioning the Pressure Tracking in Fluid Dynamics by Shape Hessian Information," *Journal of Optimization Theory and Applications*, Vol. 141, No. 3, 2009, pp. 513–531.

[168] Arian, E. and Iollo, A., "Analytic Hessian Derivation for the Quasi-One-Dimensional Euler Equations," *Journal of Computational Physics*, Vol. 228, No. 2, 2009, pp. 476 – 490.

[169] Schmidt, S. and Schulz, V., "Impulse Response Approximations of Discrete Shape Hessians with Application in CFD," *SIAM Journal on Control and Optimization*, Vol. 48, No. 4, 2009, pp. 2562–2580.

[170] Yang, S., Stadler, G., Moser, R., and Ghattas, O., "A Shape Hessian-Based Boundary Roughness Analysis of Navier–Stokes Flow," *SIAM Journal on Applied Mathematics*, Vol. 71, No. 1, 2011, pp. 333–355.

[171] Palacios, F., Economon, T. D., and Alonso, J. J., "Large-scale Aircraft Design using SU2," *53rd AIAA Aerospace Sciences Meeting*, 2015.

[172] Yang, Y.-L., Yang, Y.-J., Pottmann, H., and Mitra, N. J., "Shape Space Exploration of Constrained Meshes," *ACM Trans. Graph.*, Vol. 30, No. 6, Dec. 2011, pp. 124:1–124:12.

[173] Masters, D. A., Taylor, N. J., Rendall, T., Allen, C. B., and Poole, D. J., "Review of Aerofoil Parameterisation Methods for Aerodynamic Shape Optimisation," *53rd AIAA Aerospace Sciences Meeting*, 2015.

[174] Feng, D. and Pulliam, T. H., "Aerodynamic design optimization via reduced Hessian SQP with solution refining," Tech. rep., RIACS, 1995.

[175] Schmidt, S. and Schulz, V., "Shape Derivatives for General Objective Functions and the Incompressible Navier-Stokes Equations," *Control and Cybernetics*, Vol. 39, No. 3, 2010, pp. 677–713.

[176] Schmidt, S., Ilic, C., Schulz, V., and Gauger, N. R., "Three-Dimensional Large-Scale Aerodynamic Shape Optimization Based on Shape Calculus," *AIAA Journal*, Vol. 51, No. 11, 2013, pp. 2615–2627.

[177] Arian, E. and Vatsa, V. N., "A Preconditioning Method for Shape Optimization Governed by the Euler Equations," Tech. rep., Institute for Computer Applications in Science and Engineering, 1998.

[178] Ghate, D. and Giles, M., "Efficient Hessian Calculation Using Automatic Differentiation," *25th AIAA Applied Aerodynamics Conference*, 2007.

[179] Rumpfkeil, M. P. and Mavriplis, D. J., "Efficient Hessian Calculations using Automatic Differentiation and the Adjoint Method with Applications," *AIAA Journal*, Vol. 48, No. 10, 2010, pp. 2406–2417.

[180] Chalot, F., Dinh, Q., Herbin, E., Martin, L., Ravachol, M., and Roge, G., "Estimation of the Impact of Geometrical Uncertainties on Aerodynamic Coefficients Using CFD," *10th AIAA Non-Deterministic Approaches Conference*, 2008.

[181] Christianson, B., "Automatic Hessians by Reverse Accumulation," *IMA Journal of Numerical Analysis*, Vol. 12, No. 2, 1992, pp. 135–150.

[182] Floudas, C. A., Pardalos, P. M., and Dixon, L., "Automatic Differentiation: Calculation of the Hessian," *Encyclopedia of Optimization*, Springer US, 2009, pp. 133–137.

[183] Fike, J., Jongsma, S., Alonso, J., and Weide, E. V. D., “Optimization with Gradient and Hessian Information Calculated using Hyper-Dual Numbers,” *29th AIAA Applied Aerodynamics Conference*, 2011.

[184] Sherman, L. L., III, A. C. T., Green, L. L., Newman, P. A., Hou, G. W., and Korivi, V. M., “First- and Second-Order Aerodynamic Sensitivity Derivatives via Automatic Differentiation with Incremental Iterative Methods,” *Journal of Computational Physics*, Vol. 129, No. 2, 1996, pp. 307–331.

[185] Haug, E. J., “Second-Order Design Sensitivity Analysis of Structural Systems,” *AIAA Journal*, Vol. 19, No. 8, 1981, pp. 1087–1088.

[186] Haftka, R. T., “Second-Order Sensitivity Derivatives in Structural Analysis,” *AIAA Journal*, Vol. 20, No. 12, 1982, pp. 1765–1766.

[187] Wang, Z., Navon, I. M., Le Dimet, F. X., and Zou, X., “The Second Order Adjoint Analysis: Theory and Applications,” *Meteorology and Atmospheric Physics*, Vol. 50, No. 1-3, 1992, pp. 3–20.

[188] Papadimitriou, D. I. and Giannakoglou, K. C., “Aerodynamic Shape Optimization using First and Second Order Adjoint and Direct Approaches,” *Archives of Computational Methods in Engineering*, Vol. 15, No. 4, 2008, pp. 447–488.

[189] Papadimitriou, D. and Giannakoglou, K., “The Continuous Direct-Adjoint Approach for Second Order Sensitivities in Viscous Aerodynamic Inverse Design Problems,” *Computers and Fluids*, Vol. 38, No. 8, 2009, pp. 1539 – 1548.

[190] Papadimitriou, D. I. and Giannakoglou, K. C., “One-Shot Shape Optimization Using the Exact Hessian,” *ECCOMAS CFD*, 2010, pp. 14–17.

[191] Borzì, A. and Schulz, V., *Computational Optimization of Systems Governed by Partial Differential Equations*, Society for Industrial and Applied Mathematics, 2011.

[192] Jou, W., Huffman, W., Young, D., Melvin, R., Bieterman, M., Hilmes, C., and Johnson, F., “Practical Considerations in Aerodynamic Design Optimization,” *12th Computational Fluid Dynamics Conference*, 1995.

[193] Nemec, M. and Aftosmis, M. J., “Toward Automatic Verification of Goal-Oriented Flow Simulations,” Tech. Memorandum TM-2014-218386, NASA, 2014.

APPENDIX A

FUNCTIONS

This appendix provides pseudo-code for several of the routines referred to in Algorithm A and elsewhere. The following color scheme is used:

- GREEN: Parametric geometry modeler
- BLACK: Adjoint-based aerodynamic design framework
- ORANGE: Black-box optimizer
- BLUE: Adaptive routine

The first three are expected to be present in existing adjoint-based shape optimization frameworks. The last category (BLUE) comprises new routines that are specifically required for adaptive parameterization.

Function 2: OPTIMIZE$(\cdot)$

(with generic quasi-Newton optimizer)

Input: Objective $\mathcal{J}$ and constraints $\mathcal{C}_j$, shape deformation function $\mathcal{D}$, initial DV values $\mathbf{X}_0$ and bounds $\mathbf{b}$
Result: Optimized surface $\mathbf{S}$, adjoint solutions ψ_j, Hessian approximation $\overline{\mathbf{B}}$

```
$\mathbf{X} \leftarrow \mathbf{X}_0$
$\overline{\mathbf{B}} \leftarrow \overline{\mathbf{I}}$
InitializeOptimizer$(\mathcal{J}, \mathcal{C}_j, \mathbf{X}_0, \mathbf{b})$
repeat
    $\mathbf{S} \leftarrow$ GenerateSurface$(\mathcal{D}, \mathbf{X})$
    $\mathbf{M} \leftarrow$ GenerateFlowMesh$(\mathbf{S})$
    $\mathbf{Q} \leftarrow$ SolveFlow$(\mathbf{M})$
    $\mathcal{J}, \mathcal{C}_j \leftarrow$ EvaluateFunctionals$(\mathbf{Q}, \mathbf{S})$
    $\psi_j \leftarrow$ SolveAdjoints$(\mathbf{M}, \mathbf{Q})$
    foreach $X_i$ in $\mathbf{X}$ do
        $\frac{\partial \mathbf{S}}{\partial X_i} \leftarrow$ ShapeDerivative$(\mathcal{D}, X_i)$
        $\frac{\partial \mathcal{J}}{\partial X_i} \leftarrow$ ProjectGrad$(\psi_o, \frac{\partial \mathbf{S}}{\partial X_i})$
        foreach $\mathcal{C}_j$ in $\boldsymbol{\mathcal{C}}$ do
            $\frac{\partial \mathcal{C}_j}{\partial X_i} \leftarrow$ ProjectGrad$(\psi_j, \frac{\partial \mathbf{S}}{\partial X_i})$
        end
    end
    $\mathbf{X}, \overline{\mathbf{B}} \leftarrow$ OptStep$(\mathbf{X}, \mathcal{J}, \mathcal{C}_j, \frac{\partial \mathcal{J}}{\partial \mathbf{X}}, \frac{\partial \mathcal{C}_j}{\partial \mathbf{X}}, \overline{\mathbf{B}})$
until convergence of $\mathcal{J}$ and $\mathcal{C}_j$
```

Function 3: GETCANDIDATES$(\cdot)$

Input: Current shape control tree $\mathbf{C}$, depth d, maximum depth d_{max}
Result: Candidate refinement locations $\mathbf{C}^c$

```
$\mathbf{C}^c \leftarrow \emptyset$
foreach $C$ in $\mathbf{C}$ do
    if $depth(C) < d_{max}$ then
        if $C + L \notin \mathbf{C}^c$ then
            Add $C + L$ to $\mathbf{C}^c$
        end
        if $C + R \notin \mathbf{C}^c$ then
            Add $C + R$ to $\mathbf{C}^c$
        end
    end
end
// Recurse to deeper levels
if $d > 1$ then
    $\mathbf{C}^d \leftarrow$ GetCandidates$(\mathbf{C}^c, d-1, d_{max})$
    Add $\mathbf{C}^d$ to $\mathbf{C}^c$
end
```

Function 4: POTENTIAL$(\cdot)$

Input: Parameterization function $\mathcal{P}$; surface S, candidate shape control $\mathbf{C}$, previous DV bounds $\mathbf{b}$ and quasi-Newton Hessian approximation $\overline{\mathbf{B}}$, adjoint solutions ψ_o, ψ_j
Result: Estimated potential for objective reduction $\Delta\mathcal{J}_\star$

```
$\mathcal{D}^c, \mathbf{X} \leftarrow \mathcal{P}(\mathrm{S}, \mathbf{C})$
foreach $X$ in $\mathbf{X}$ do
    $\frac{\partial \mathrm{S}}{\partial X} \leftarrow$ ShapeDerivative$(\mathcal{D}, X)$
    $\frac{\partial \mathcal{J}}{\partial X} \leftarrow$ ProjectGrad$(\psi_o, \frac{\partial \mathrm{S}}{\partial X})$
    foreach $\mathcal{C}_j$ in $\boldsymbol{\mathcal{C}}^a$ do
        $\frac{\partial \mathcal{C}_j}{\partial X} \leftarrow$ ProjectGrad$(\psi_j, \frac{\partial \mathrm{S}}{\partial X})$
    end
end
$\mathbf{b} \leftarrow$ Prolong$(\mathcal{D}^{k-1}, \mathcal{D}^c, \mathbf{b})$
$\overline{\mathbf{B}} \leftarrow$ Prolong$(\mathcal{D}^{k-1}, \mathcal{D}^c, \overline{\mathbf{B}})$
$\Delta\mathcal{J}_\star \leftarrow f\left(\frac{\partial \mathcal{J}}{\partial \mathbf{X}}, \frac{\partial \boldsymbol{\mathcal{C}}^a}{\partial \mathbf{X}}, \mathbf{b}, \overline{\mathbf{B}}\right)$
```

Function 5: ADAPTSHAPECONTROL$(\cdot)$

Input: Parameterization function $\mathcal{P}$, current and candidate shape control $\mathbf{C}$, $\mathbf{C}^c$, surface $\mathbf{S}$, adjoint solutions ψ_j, Hessian approximation $\overline{\mathbf{B}}$, DV bounds $\mathbf{b}$, auto-growth criteria r and w.
Result: Selected shape control $\mathbf{C}$

```
// Phase 1. Build priority queue
$q \leftarrow \emptyset$
$\Delta\mathcal{J}^0_\star \leftarrow$ Potential$(\mathcal{P}, \mathbf{S}, \mathbf{C}, \psi_j, \mathbf{b}, \overline{\mathbf{B}})$
foreach $C$ in $\mathbf{C}^c$ do
    $\Delta\mathcal{J}_\star \leftarrow$ Potential$(\mathcal{P}, \mathbf{S}, \mathbf{C} + C, \psi_j, \mathbf{b}, \overline{\mathbf{B}})$
    $I \leftarrow \Delta\mathcal{J}_\star - \Delta\mathcal{J}^0_\star$
    $q$.Add$(C, I)$
end
// Phase 2. Greedy adaptation
$C^{best}, I^1 \leftarrow q$.Pop$()$
$\mathbf{C} \leftarrow \mathbf{C} + C^{best}$
$\Delta\mathcal{J}^0_\star \leftarrow \Delta\mathcal{J}^0_\star + I^1$
repeat
    q.MarkAllStale()
    repeat
        foreach $C$ in $q$.Best$(N_w)$ do
            $\Delta\mathcal{J}_\star \leftarrow$ Potential$(\mathcal{P}, \mathbf{S}, \mathbf{C} + C, \psi_j, \mathbf{b}, \overline{\mathbf{B}})$
            $I \leftarrow \Delta\mathcal{J}_\star - \Delta\mathcal{J}^0_\star$
            $q$.Update$(C, I)$
        end
    until $q$.Best(1) is fresh
    $C^{best}, I^{best} \leftarrow q$.Pop$()$
    $\mathbf{C} \leftarrow \mathbf{C} + C^{best}$
    $\Delta\mathcal{J}^0_\star \leftarrow \Delta\mathcal{J}^0_\star + I^{best}$
until for $w$ consecutive passes $\frac{I^{best}}{I^1} < r$
```

Cost of ADAPTSHAPECONTROL($\cdot$)

Let N_{DV} be the number of design variables before adaptation begins, N_c the number of candidates, N_{add} the number of parameters that will be added, and N_w the number of the top-ranked candidates to be re-evaluated on each pass. Clearly $N_{add} \leq N_c$, and from Section 2.3.1 $N_c \approx 2^{Dd-1} N_{DV}$. From Algorithm 4, each indicator evaluation costs

$$C_I = N_{param}(C_{ss} + N_\psi C_{proj}) \tag{A.1}$$

where N_ψ is the number of adjoint projections involved, N_{param} is the current number of design variables, C_{ss} is the cost of a shape sensitivity computation and C_{proj} is the cost of an adjoint inner product. For the first phase, N_c indicator evaluations are required to build the initial priority queue, during which $N_{param} = N_{DV} + 1$. The cost for the first phase is thus

$$C_1 = 2^{Dd-1} N_{DV}(N_{DV} + 1)(C_{ss} + N_\psi C_{proj}) \tag{A.2}$$

During the second phase, the indicator is computed $N_w N_{add}$ times, and on average, $N_{param} = N_{DV} + \frac{N_{add}}{2}$. Then the cost of the second phase it at most

$$C_2 \leq \frac{1}{2} N_w N_{DV}^2 2^{Dd-1}(2 + 2^{Dd-1})(C_{ss} + N_\psi C_{proj}) \tag{A.3}$$

Thus the total cost of the adaptation procedure is bounded by

$$C \leq N_{DV}^2 2^{Dd-1} \left(1 + \frac{1}{2} N_w (2 + 2^{Dd-1})\right) (C_{ss} + N_\psi C_{proj}) \tag{A.4}$$

Asymptotically, this cost is $\mathcal{O}(N_{DV}^2)$, but would have been $\mathcal{O}(N_{DV}^3)$ if the entire queue were re-evalauted on every pass ($N_w = N_c$). This also demonstrates that the cost depends linearly on the window size N_w and exponentially on the search depth d.

APPENDIX B

DISCRETIZATION OF SHAPE HESSIANS

This appendix examines Hessian matrices with respect to discretized shape control. Unsurprisingly, the Hessian entries depend on the dimensions of the shape deformation modes. It is less intuitively obvious that this dependence behaves differently for different objectives. This impacts both the refinement indicator (Section 3.2.2) and the Hessian prolongation operator (Section 3.3). The scope of this appendix is restricted to two common objective functionals, which will be sufficient for the purposes of this thesis, though far from a comprehensive treatment of the subject.

B.1 Hessian Discretization

We will consider the the objective Hessian matrix $\overline{\mathbf{H}}$ with respect to the design variables:

$$H_{i,j} := \frac{\partial}{\partial X_j}\frac{\partial \mathcal{J}}{\partial X_i} = \frac{\partial}{\partial X_j}\left\langle \frac{\partial \mathcal{J}}{\partial \mathsf{S}}, \frac{\partial \mathsf{S}}{\partial X_i} \right\rangle \tag{B.1}$$

$$= \left\langle \frac{\partial}{\partial X_j}\left(\frac{\partial \mathcal{J}}{\partial \mathsf{S}}\right), \frac{\partial \mathsf{S}}{\partial X_i} \right\rangle + \left\langle \frac{\partial \mathcal{J}}{\partial \mathsf{S}}, \frac{\partial}{\partial X_j}\frac{\partial \mathsf{S}}{\partial X_i} \right\rangle \tag{B.2}$$

$$\overset{\text{(Eq. 3.7)}}{=} \left\langle \underbrace{\mathcal{H}\frac{\partial \mathsf{S}}{\partial X_j}}_{\mathbf{A}}, \underbrace{\frac{\partial \mathsf{S}}{\partial X_i}}_{\mathbf{B}} \right\rangle + \left\langle \underbrace{\frac{\partial \mathcal{J}}{\partial \mathsf{S}}}_{\mathbf{C}}, \underbrace{\cancelto{0 \text{ if } \mathcal{D} \text{ linear}}{\frac{\partial^2 \mathsf{S}}{\partial X_i \partial X_j}}}_{\mathbf{D}} \right\rangle \tag{B.3}$$

Term **A** involves the Hessian operator of the objective $\mathcal{H}$, which depends on the type of objective function. Terms **B** and **C** are familiar from Equation (3.14). Term **D** is the second derivative of the shape with respect to the design variables, which measures the nonlinearity of the deformation function $\mathcal{D}$. To simplify, I will assume that deformation is linear with respect to the design variables, i.e. $\Delta \mathsf{S} = \mathsf{A}\Delta \mathbf{X}$, where A is constant.[1] Many common deformation functions are linear, including radial basis functions, free-form deformation, and Hicks-Henne bump functions. Examples of nonlinear deformation include any form of rotation, such as twist or dihedral, although for small rotations I observe its effects to be small (see the top frame of Figure 5.3). If term **D** is significant, it can be computed either analytically by the modeler, or by N_{DV} finite-differences of a modeler that provides $\frac{\partial \mathsf{S}}{\partial \mathbf{X}}$, or by N_{DV}^2 finite-differences of an undifferentiated modeler. However, one would then be faced with N_{DV}^2 gradient projections of $\frac{\partial \mathcal{J}}{\partial \mathsf{S}}$ into term **D**.

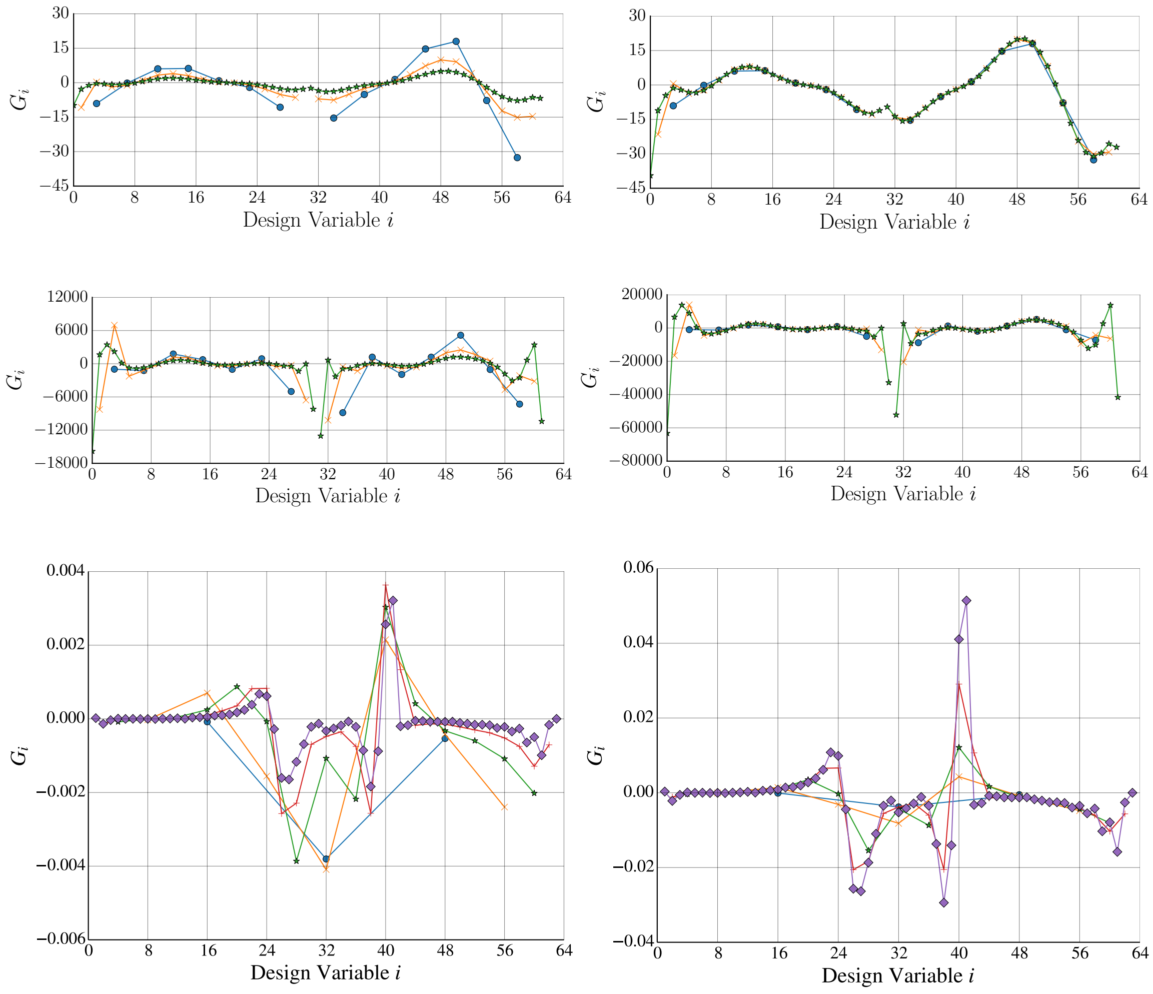
Coupling of Objective and Shape Control

In Equation (3.14), note that $\frac{\partial \mathcal{F}}{\partial \mathsf{X}_i}$ is computed via two independent terms, $\frac{\partial \mathcal{F}}{\partial \mathsf{S}}$ and $\frac{\partial \mathsf{S}}{\partial \mathbf{X}}$. The objective sensitivities are completely decoupled from the deformation mode shapes. To see this, consider how $\frac{\partial \mathcal{F}}{\partial \mathbf{X}}$ changes as the shape control is refined. In Equation (3.15) note that $\frac{\partial \mathcal{F}}{\partial \mathsf{S}}$ is constant for a given shape S. Then, taking h constant, $\frac{\partial \mathcal{F}}{\partial \mathsf{X}_i} \sim A_i$. Cutting the width of a deformation mode in half causes the corresponding gradient entry to be cut in half in 2D (or divided by four in 3D). If the shape modes are localized (have compact support), then in the limit of refinement $\frac{\partial \mathcal{F}}{\partial \mathsf{X}_i} \to 0$ at $\mathcal{O}(A_i)$.

Because gradient projection does *not* couple the objective and shape control, this scaling should apply to all functionals (for non-overlapping modes). To verify this, consider Figure B.1, which shows exact gradients evaluated at different shape control resolutions, for three completely different classes of objective (shape matching, 2D surface pressure matching, and 3D off-body pressure matching). The left column shows the unscaled versions of the gradients. As the shape control is uniformly refined, the magnitude of each individual component decreases. That this rate of decrease is exactly A_i can be seen in the right column of Figure B.1, which multiplies the gradients by $\frac{A_i^0}{A_i^k}$. As the shape control is refined, the lines more closely match each other. This indicates that the gradient *density* is consistent, and that the identified scaling factor is correct for all the functionals. Note that the largest sources of error are in regions where the gradient density varies rapidly compared to the resolution of the shape control.

[1]Note that the *parameterization* function $\mathcal{P}$ is still nonlinear. In other words, A may be a nonlinear function of **C**, but once parameterized, the deformations are linear with respect to **X**.

Figure B.1: Unscaled gradients (*left*) vs. scaled gradients (*right*). *Top row:* Shape-matching objective $\mathcal{J}_A$. *Middle row:* Pressure-matching objective $\mathcal{J}_B$. For these top two rows, the left/right groups within each plot correspond to camber/thickness variables. *Bottom row:* Inverse off-body pressure-matching objective $\mathcal{J}_D$.



The Hessian scaling is not so simple. To compute $\overline{\mathbf{H}}$ (Equation (B.3)) and $\Delta\mathcal{J}_\star$ (Equation (3.12)) involve evaluating the result of $\mathcal{H}$ or $\mathcal{H}^{-1}$ acting on a surface vector ($\mathcal{H}\frac{\partial \mathsf{S}}{\partial \mathsf{X}_j}$ and $\mathcal{H}^{-1}\left(\frac{\partial \mathcal{J}}{\partial \mathsf{S}} + \lambda_j \frac{\partial \mathcal{C}_j^a}{\partial \mathsf{S}}\right)$, respectively[2]). Thus when using the Hessian, which is involved in both the second-order refinement indicator and in the prolongation operator, we must account for the coupling between the objective function and the shape modes. This requires special treatment of the Hessian, which is the motivating purpose for this appendix.

Discretization

If $\mathcal{H}$ is a local operator, then with our assumption of a non-overlapping discretization of the shape control and linear deformation, $\overline{\mathbf{H}}$ is diagonal. Taking $\mathcal{H}\frac{\partial \mathsf{S}}{\partial \mathsf{X}_i}$ to be its average value over region i, Equation (B.3) can then be approximated as

$$\left(\mathcal{H}\frac{\partial \mathsf{S}}{\partial \mathsf{X}_i}\right)_i \approx \frac{H_{i,i}}{A_i h} \tag{B.4}$$

To proceed we need to know the form of $\mathcal{H}$. Examples for two specific objective functions are now given. Recall the simplifying assumptions that have been made thus far:

1. The surface deformation is sufficiently linear with respect to the design variables.
2. The deformation modes are localized (have compact support), so that they form a nearly non-overlapping discretization of the continuous shape control.
3. Within a given shape control tree, the deformation modes have consistent shapes and characteristic magnitudes (or units).

Geometric Shape Matching

First, consider a geometric shape-matching (GSM) objective function:

$$\mathcal{J}^{GSM} = \frac{1}{2}\int_{\mathsf{S}} (\mathsf{y}(\mathbf{x}) - \mathsf{y}^*(\mathbf{x}))^2 d\mathsf{A}(\mathbf{x}) \tag{B.5}$$

where $\mathsf{y}(\mathbf{x})$ is the shape control, and $\mathsf{y}^*(\mathbf{x})$ is the target shape. The objective is a quadratic function directly of the shape control, and so it is clear that $\mathcal{H} = \mathcal{I}$ (the identity operator). Then

$$\left(\mathcal{H}\frac{\partial \mathsf{S}}{\partial \mathsf{X}_i}\right)_i = \left(\frac{\partial \mathsf{S}}{\partial \mathsf{X}_i}\right)_i \approx h$$

[2]Note that in each case, the operand is different. This disconnect is what will give rise to a scale-dependent term in the indicator.

Then from Equation (B.4) the Hessian diagonal entries scale as $H_{i,i} \sim A_i h^2$. If the shape modes have compact support, then in the limit of refinement, $H_{i,i} \to 0$, like the gradients.

2D Surface Pressure-Based Functionals

Now consider an aerodynamic pressure-matching functional defined as a surface integral of discrepancies from a target pressure profile:

$$\mathcal{J}^P = \frac{1}{2} \int_{\mathsf{S}} (p(\mathbf{x}) - p^*(\mathbf{x}))^2 d\mathbf{x} \tag{B.6}$$

where $p^*(\mathbf{x})$ is the target profile. Using Fourier mode analysis, Arian and Ta'asan show that, for 2D inviscid flow when close to the target profile, $\mathcal{H}$ is a local differential operator of the form $\mathsf{K}\frac{\partial^2}{\partial x^2}$, where K involves the local Mach number, velocity and density [112]. This implies that the discrete Hessian matrix entries depend on the geometric curvature of the *deformation modes*.[3] Design variables enacting deformations with sharper curvature will therefore have higher second-derivatives. In this case,

$$\left(\mathcal{H}\frac{\partial \mathsf{S}}{\partial X_i}\right)_i = K_i \left(\frac{\partial^2}{\partial x^2}\frac{\partial \mathsf{S}}{\partial X_i}\right)_i \approx \frac{K_i h}{A_i^2}$$

Then with Equation (B.4) the diagonal Hessian entries scale as $H^P_{i,i} \sim \frac{h^2}{A_i}$. If the shape modes have compact support, then in the limit of refinement $H^P_{i,i} \to \infty$. This leads to ill-conditioning for high-frequency deformation modes, as noted in [112].

The most important observation of this section is given in Table B.1, which shows that the Hessian entries depend both on the type of objective function and on the rough dimensions of the deformation modes. The following sections show how this impacts the Newton step, the refinement indicator, and the Hessian prolongation operator.

Table B.1: Scaling of gradient and Hessian diagonal with shape mode dimensions for different objective functionals. The final column shows that a naively computed importance indicator can be (falsely) scale-dependent.

Objective	G_i	$H_{i,i}$	$\Delta\mathcal{J}_\star = \frac{1}{2}\mathbf{G}^\mathsf{T}\overline{\mathbf{H}}^{-1}\mathbf{G}$
$\mathcal{J}_{GSM}$	$A_i h$	$A_i h^2$	1
$\mathcal{J}_P$	$A_i h$	$\frac{h^2}{A_i}$	A_i^2

[3]Of course the Hessian also depends on the curvature of the surface itself (via the flow variables), but this is already included in K.

B.2 Newton Step Scale-Dependence

Equation (3.16) gives an estimate of the potential design improvement in a finite, non-overlapping parameterization. Computing this involves the discretized Newton step:

$$(\delta S_\star)_i = \left(\mathcal{H}^{-1} \left(\frac{\partial \mathcal{J}}{\partial \mathsf{S}} + \lambda_j \frac{\partial \mathcal{C}_j^a}{\partial \mathsf{S}} \right) \right)_i \tag{B.7}$$

Now Equation (B.7) must be expressed in terms of the gradient vector and the Hessian matrix. Table B.1 shows that naively taking the standard black-box quasi-Newton step $(\delta S_\star)_i \approx \delta X_{i\star} \equiv (\overline{\mathbf{H}}^{-1}\mathbf{G})_i$ would result in a falsely scale-dependent indicator.[4]

For an arbitrary objective, it is unclear how to proceed. However, motivated by [112], I remark that, for the two objectives considered so far, the Hessian can be decomposed into two terms:

$$\begin{aligned} \mathcal{H}(\cdot) &= \mathsf{K} \odot (\mathcal{D}_{\mathcal{H}}(\cdot)) \\ \mathcal{H}^{-1}(\cdot) &= \frac{1}{\mathsf{K}} \odot (\mathcal{D}_{\mathcal{H}}^{-1}(\cdot)) \end{aligned} \tag{B.8}$$

where $\odot$ represents element-wise (Hadamard) multiplication. K is a continuous surface vector that encodes effects due to the *static* geometry S, while $\mathcal{D}_{\mathcal{H}}$ is an operator that encodes dependence on the surface *deformation*. This decomposition appears to be valid for many common objective functions (e.g. [112, 170]), but it may not be completely general. Note that the static effects K appear in both the Hessian matrix (Equation (B.3)) and the Newton step (Equation (B.7)). I will use these two equations to eliminate K. Despite not claiming to know K accurately enough to directly compute the Hessian, we rely on the fact that its local value is consistent, in a continuous sense.

Using Equation (B.8), Equation (B.4) becomes

$$K_i \left(\mathcal{D}_{\mathcal{H}} \frac{\partial \mathsf{S}}{\partial X_i} \right)_i \approx \frac{H_{i,i}}{A_i h} \tag{B.9}$$

Combining this with Equation (B.7) and again using Equation (B.8), we obtain

$$(\delta \mathsf{S})_i \approx \frac{A_i h}{H_{i,i}} \left(\mathcal{D}_{\mathcal{H}} \frac{\partial \mathsf{S}}{\partial X_i} \right)_i \left(\mathcal{D}_{\mathcal{H}}^{-1} \left(\frac{\partial \mathcal{J}}{\partial \mathsf{S}} + \lambda_j \frac{\partial \mathcal{C}_j^a}{\partial \mathsf{S}} \right) \right)_i \tag{B.10}$$

This is the discretized Newton step for any finite search space subject to the conditions

[4]This also raises questions about the suitability of standard black-box optimizers, which use update formulas based on these aggregated gradients, for shape optimization with high-frequency modes. I will not address this concern here.

enumerated earlier, plus Equation (B.8). This is a generic expression for any objective function. Next I examine the form of $\mathcal{D}_{\mathcal{H}}$ for the two objective functions examined here.

Geometric Shape Matching

For the GSM objective (Equation (B.5)), $\mathcal{H} = \mathcal{I}$, which is trivially decomposed into $\mathcal{D}_{\mathcal{H}} = \mathcal{I}$ and $\mathsf{K} = 1$. Thus

$$\left(\mathcal{D}_{\mathcal{H}} \frac{\partial \mathsf{S}}{\partial X_i}\right)_i = \left(\frac{\partial \mathsf{S}}{\partial X_i}\right)_i \approx h \tag{B.11}$$

Then with Equation (3.15), Equation (B.10) simplifies to

$$(\delta S_\star)_i \approx \frac{h}{H_{i,i}} \left(\frac{\partial \mathcal{J}}{\partial X_i} + \lambda_j \frac{\partial \mathcal{C}^a_j}{\partial X_i}\right) \tag{B.12}$$

In this special case, it so happens that $(\delta S_\star)_i = \delta X_i$, the standard Newton step used in quasi-Newton black-box solvers.

2D Surface Pressure-Based Functionals

For the 2D pressure-matching functional (Equation (B.6)), $\mathcal{D}_{\mathcal{H}} = \frac{\partial^2}{\partial x^2}$ and K involves Mach number, velocity and density [112]. Then

$$\left(\mathcal{D}_{\mathcal{H}} \frac{\partial \mathsf{S}}{\partial X_i}\right)_i = \left(\frac{\partial^2}{\partial x^2} \frac{\partial \mathsf{S}}{\partial X_i}\right)_i \approx \frac{h}{A_i^2} \tag{B.13}$$

The inverse of the differential curvature operator is a Laplacian smoothing operator, which is a commonly used preconditioner for aerodynamic optimization [30, 175, 176].[5] For clarity, define $Z_i := \frac{\partial \mathcal{J}}{\partial X_i} + \lambda_j \frac{\partial \mathcal{C}^a_j}{\partial X_i}$. Then with Equation (3.15),

$$\left(\mathcal{D}_{\mathcal{H}}^{-1} \left(\frac{\partial \mathcal{J}}{\partial \mathsf{S}} + \lambda \frac{\partial \mathcal{C}^a_j}{\partial \mathsf{S}}\right)\right)_i \approx \frac{1}{h}\left(\frac{1}{4}\frac{Z_{i-1}}{A_{i-1}} + \frac{1}{2}\frac{Z_i}{A_i} + \frac{1}{4}\frac{Z_{i+1}}{A_{i+1}}\right) \tag{B.14}$$

Substituting Equations (B.13) and (B.14) into Equation (B.10), we have

$$(\delta \mathsf{S}_\star)_i \approx \frac{h}{A_i}\frac{1}{H_{i,i}}\left(\frac{1}{4}\frac{Z_{i-1}}{A_{i-1}} + \frac{1}{2}\frac{Z_i}{A_i} + \frac{1}{4}\frac{Z_{i+1}}{A_{i+1}}\right) \tag{B.15}$$

[5]Note that in those cases the smoothing is spatially uniform. [177] shows how to apply it with locally varying strength. By contrast to all these approaches, the present approach explicitly accounts for constraints.

This confirms that for the pressure-matching functional, $(\delta \mathsf{S}_\star)_i \neq -\overline{\mathbf{H}}^{-1}\mathbf{Z}$. Unlike in shape matching, where each station could be considered completely independently, here neighboring stations are coupled via the last term in Equation (B.15). A second distinction is the presence of scale-dependence $\sim \frac{1}{A_i^2}$.

Other Objectives

The decomposition $\mathcal{H} = \mathsf{K} \odot \mathcal{D}_{\mathcal{H}}$ appears to be valid for other objective functions as well. Lewis and Nash show that this is valid for many model problems, because the optimization Hessian operator is often elliptic, even when the governing equations themselves are hyperbolic [85]. The decomposition also appears to be valid for drag in 2D Navier-Stokes flow under certain conditions [170], potential flow [167], and drag or pressure matching in quasi-1D inviscid flow [113, 168].

In [112], it is shown to be valid for 3D inviscid pressure matching, but in that case $\mathcal{D}_{\mathcal{H}}$ is a *pseudo*-differential, non-local operator. In this case, the refinement indicator might become non-compact, i.e. involving sensitivity information from the entire domain. For the purposes of developing local preconditioners, some authors simply tune a differential operator to approximate the true behavior [169].

Quasi-Hessian-free Approach

It may be the case that we do not trust the quasi-Newton Hessian to sufficiently accurately predict the best shape control. However, in these cases, it would still be desirable for the refinement indicator to be reflect to the objective-dependent trends in $\mathcal{D}_{\mathcal{H}}$. Following Table B.1, for shape matching we should take $H_{i,i} = A_i$, while for pressure matching $H_{i,i} = \frac{1}{A_i}$. This is equivalent to assuming that K is uniform. Substituting these into Equation (B.12) and Equation (B.15), we obtain

$$(\delta S_\star)_i^{GSM,\mathscr{H}} \approx \frac{h}{A_i}\left(\frac{\partial \mathcal{J}}{\partial X_i} + \lambda_j \frac{\partial \mathcal{C}_j^a}{\partial X_i}\right) \tag{B.16}$$

$$(\delta \mathsf{S}_\star)_i^{GSM,\mathscr{H}} \approx h\left(\frac{1}{4}\frac{Z_{i-1}}{A_{i-1}} + \frac{1}{2}\frac{Z_i}{A_i} + \frac{1}{4}\frac{Z_{i+1}}{A_{i+1}}\right) \tag{B.17}$$

These are not expected to be accurate, but will at least reflect the underlying scale-dependence of the shape control's influence on second-order information.

B.3 Hessian Prolongation Operator

Section 3.3 proposes estimating the Hessian in a candidate search space from the quasi-Newton approximation in the previous search space. This requires a Hessian prolongation operator, which must also account for scale-dependent effects. Under the same assumptions as before, the Hessian diagonal entry for design variable i at shape control level k can be expressed as

$$H^k_{i,i} = \left\langle \frac{\partial \mathsf{S}}{\partial X^k_i}, K^k_i \left(\mathcal{D}_{\mathcal{H}} \frac{\partial \mathsf{S}}{\partial X^k_i} \right)_i \right\rangle \approx h K^k_i \left(\mathcal{D}_{\mathcal{H}} \frac{\partial \mathsf{S}}{\partial X^k_i} \right)_i A^k_i \tag{B.18}$$

As before, K represents effects that depend only on the static surface, not on the deformation modes. Thus $\mathsf{K}^{k+1} \equiv \mathsf{K}^k$ for *any* refinement of the shape control. To estimate the value of K^{k+1}_j for a new controller X^{k+1}_j, we can linearly interpolate between the values for the nearest neighboring controllers, X^k_L and X^k_R (see Figure 2.8):

$$K^{k+1}_j \approx (1-u)K^k_L + uK^k_R \tag{B.19}$$

where u is the fraction of the geodesic (unskewed parametric) distance between L and R at which j is located. Then the Hessian entry for this new parameter is approximated as

$$H^{k+1}_{j,j} \approx h \left((1-u)K^k_L + uK^k_R \right) \left(\mathcal{D}_{\mathcal{H}} \frac{\partial \mathsf{S}}{\partial X^{k+1}_j} \right)_j A^{k+1}_j \tag{B.20}$$

Solving Equation (B.18) for K and substituting this in for K_L and K_R in Equation (B.20), we obtain

$$H^{k+1}_{j,j} \approx \left((1-u) \frac{H^k_{L,L}}{A^k_L \left(\mathcal{D}_{\mathcal{H}} \frac{\partial \mathsf{S}}{\partial X^k_L} \right)_L} + u \frac{H^k_{R,R}}{A^k_R \left(\mathcal{D}_{\mathcal{H}} \frac{\partial \mathsf{S}}{\partial X^k_R} \right)_R} \right) \left(\mathcal{D}_{\mathcal{H}} \frac{\partial \mathsf{S}}{\partial X^{k+1}_j} \right)_j A^{k+1}_j \tag{B.21}$$

This is a generic prolongation operator, subject to the various assumptions we have made. We can then simplify it for specific objective functionals.

Geometric Shape Matching

As a trivial example, consider the GSM functional, Equation (B.5). Although $\overline{\mathbf{H}}^{k+1}_0$ could be computed exactly in this analytic case, let us momentarily pretend that it is unobtainable.

Using Equation (B.11), Equation (B.21) becomes

$$H_{j,j}^{k+1} \approx (1-u) H_{L,L}^{k} \frac{A_j^{k+1}}{A_L^k} + u H_{R,R}^{k} \frac{A_j^{k+1}}{A_R^k} \tag{B.22}$$

2D Pressure Matching

For the 2D inverse surface pressure-matching functional (Equation (B.6)), we obtain a different prolongation operator. Using Equation (B.13), Equation (B.21) becomes

$$H_{j,j}^{k+1} \approx (1-u) H_{L,L}^{k} \frac{A_L^k}{A_j^{k+1}} + u H_{R,R}^{k} \frac{A_R^k}{A_j^{k+1}} \tag{B.23}$$

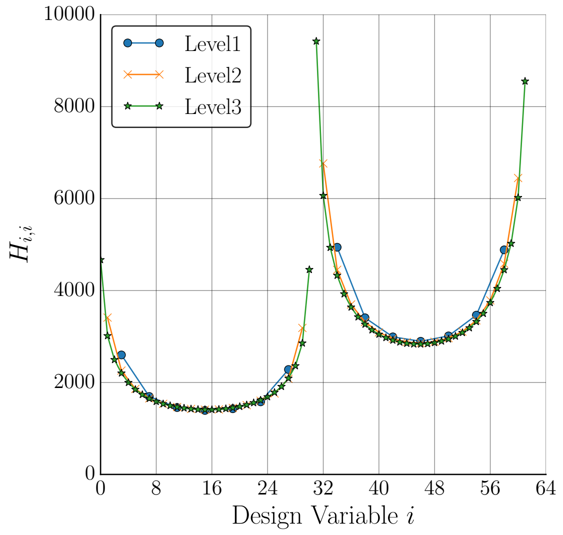
B.3.1 Verification

This section will verify that the prolongation operators are appropriate for various functionals under consideration. For each case, while holding the shape constant, I compute the Hessian diagonal across a range of uniformly-refined resolutions of shape control. Ultimately, the goal is to transfer a quasi-Newton approximation, but for verification purposes it is important to isolate the prolongation operator from any other approximations or errors. Therefore I compute the "true" Hessian diagonal by finite-differencing the adjoint-provided gradients. This proved to be sufficiently accurate for these relatively smooth cases. Four objective functionals are evaluated:

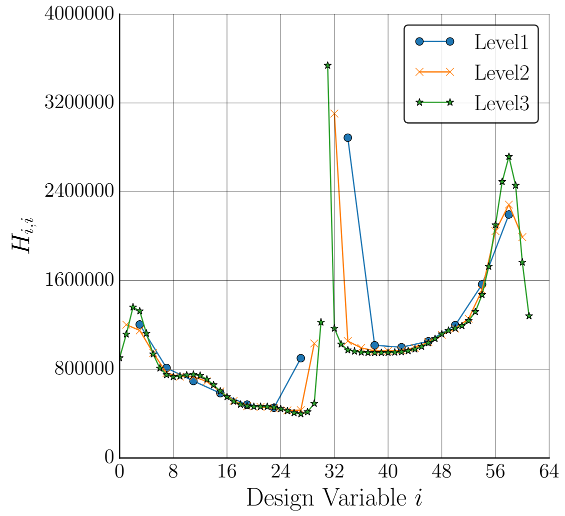
1. Geometric shape matching for an airfoil (Equation (B.5))
2. Surface pressure matching for an airfoil (Equation (B.6)) at $M_\infty = 0.3$ (see Section 5.2 for the setup)
3. Drag on the NACA 0012 airfoil at Mach 0.85
4. Drag on a diamond airfoil at Mach 2.0

To evaluate the claim that the specific details of the deformation modes are unimportant compared to the gross relative dimensions A_i, these tests use three different shape deformation bases (all local, per our prior restriction):

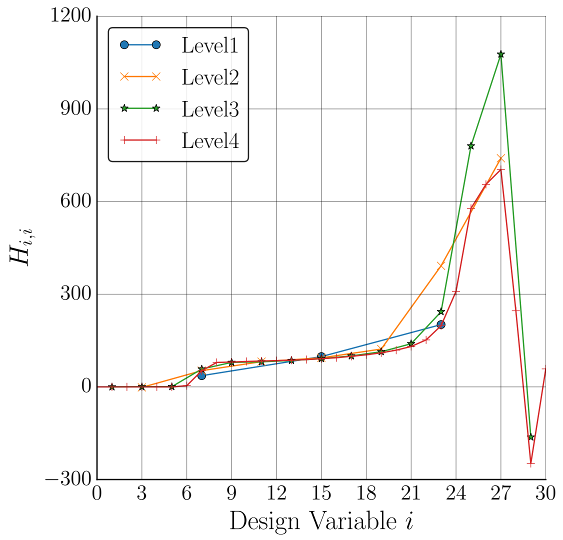
1. Cubic interpolation bumps — Cases 1 and 3
2. Radial basis functions (see Section 4.4) — Case 2
3. Linear "hat"-shaped bumps — Case 4



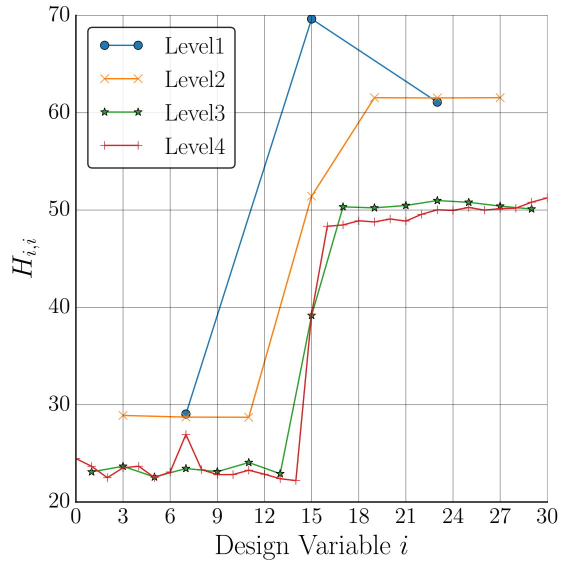
(a) Case 1: Shape Matching: Left/right sets are camber/thickness variables, trailing edge is at the center, leading edge at the sides.



(b) Case 2: Pressure Matching: Left/right sets are camber/thickness variables, trailing edge is at the center, leading edge at the sides.



(c) Case 3: Drag (Mach 0.85): Symmetric NACA 0012, top surface only, leading edge to right

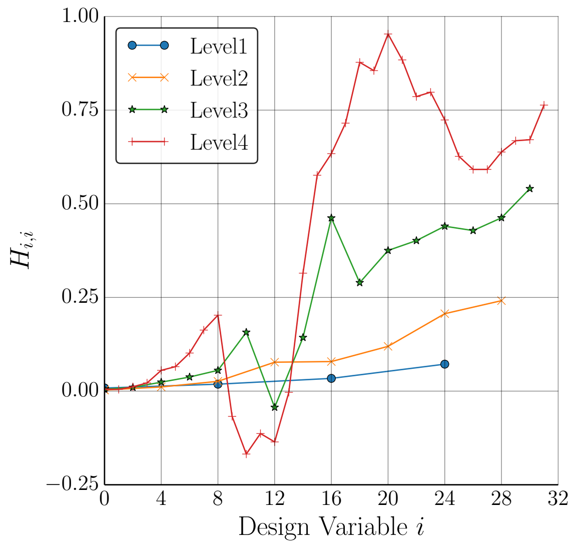


(d) Case 4: Drag (Mach 2.0): Symmetric diamond airfoil, top surface only, leading edge to right

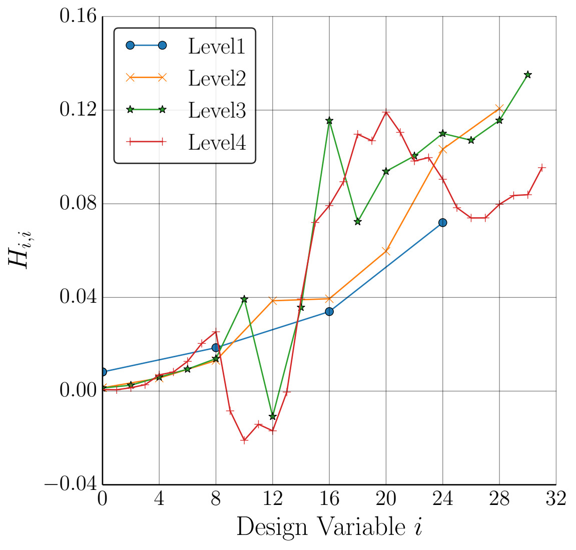
Figure B.2: Hessian diagonal for various objectives and at various shape control resolutions, transformed to be resolution-invariant, using Equation (B.22) (frame B.2a) or Equation (B.23) (other frames)

Figure B.2 shows the results of applying Equation (B.22) or Equation (B.23) to the Hessian diagonals for Cases 1-4 at different shape control resolutions. In each case, as the shape control is refined, the lines more closely match each other. This indicates that the transformations above are properly accounting for the scale dependence of the Hessian operator. It also indicates that, as long as the deformation basis remains consistent, its details can be ignored in the prolongation. Although Equation (B.23) was derived for a 2D surface pressure-matching functional, it also appears to be correct for drag functionals (Figures B.2c and B.2d). Both involve integrations of pressure over the surface; it may be that all similar 2D surface pressure-based functionals support a similar prolongation operator, but I will not attempt to address that question here.

As with the gradients (Figure B.1), the largest sources of error are in regions where the Hessian density varies rapidly compared to the resolution of the shape control. For example, at the trailing edge discontinuity we see the suboptimal effects of extrapolation (middle of Figures B.2a and B.2b). Similarly, Figure B.2d shows a sharpening discontinuity in the Hessian due to the shock anchored to the peak of the diamond airfoil. At the coarsest shape control levels, the prolonged Hessian can be noticeably different from the finer levels (e.g. in Figure B.2d). This indicates that the coarse shape control does not adequately reflect the finer levels.



(a) Unscaled



(b) Scaled with Equation (B.23)

Figure B.3: 3D off-body pressure functional: Hessian diagonal at various shape control resolutions. Left to right traverses the radius control stations from tip of forebody to the base (see Figure 5.18).

We should be cautious about extending this correction too far. Figure B.3 shows an attempt to apply this same correction (Equation (B.23)) to a 3D off-body pressure-matching functional at Mach 1.6. The details of the setup are given in Section 5.3. Comparing Figure B.3a to Figure B.3b, we see that the prolongation operator does cluster the curves better, indicating that the correction is helping somewhat. However, it does not appear that the results are converging to a consistent curve. This may indicate that there are other significant terms in the Hessian operator that are Equation (B.23) does not account for. Nevertheless, as it appears to help somewhat, I did use it for the inverse off-body design problem in Section 5.3.

B.4 Alternative Techniques for Computing Hessians

Because of the promise of increased optimization efficiency[6], many approaches to computing or estimating Hessians have been explored. Of the methods for computing either the exact Hessian itself, or Hessian-vector products, those with the lowest cost generally require the most invasive modifications to the design codes. Several authors have analytically derived the Hessian (or its most important terms) for specific problem classes[7]. Although this approach can provide insight into the behavior of a problem and can be used to develop effective optimization preconditioners [177], the procedure is difficult to generalize. One obvious general technique is to use finite-differencing, which requires $\mathcal{O}(N_{DV})$ flow and adjoint solutions. The Hessian can also be computed by double automatic differentiation [181, 182], though this is also computationally prohibitive [179]. To address concerns over numerical accuracy, hyper-dual numbers (an extension of complex-step methods) have been used to compute second derivatives to machine precision, but the cost is even higher than finite differencing [183]. These approaches can be automated somewhat, but still involve substantial modification to the design tools.

The fastest general methods for computing exact Hessians combine one of the above methods with an adjoint approach. For example, several authors combine automatic differentiation with an adjoint [178, 179, 184]. Others use various second-order adjoint approaches [185–191]. Although these approaches vary somewhat in cost, the solution of at least $\mathcal{O}(N_{DV})$ linear PDES appears unavoidable. This is generally out of the question except for very low numbers of design variables, making it unsatisfactory for adaptive parameterization. Some approaches seek to reduce the cost by solving the second-order

[6]As well as for extrapolation [178, 179] or design under uncertainty [180].

[7]Examples relevant to aerospace include pressure matching in potential flow [167], inviscid flow [112], and quasi-1D inviscid flow [113, 168]; minimizing drag in Stokes flow [169]; and minimizing drag due to wall roughness [170]. The approach is used in other fields as well, such as tomography [89].

adjoints inexactly with inexact Newton-Krylov methods [6, 46]. This can lead to payoffs when there are large numbers of design variables but is generally incompatible with a standard first-order adjoint framework.

A number of Hessian approximations are used in the literature, each motivated by the promise of obtaining substantial improvement over Hessian-free methods, while avoiding the excessive cost of the exact-Hessian approaches above. In assessing these techniques, caution must be exercised. Unlike in quasi-Newton optimization, which self-corrects the Hessian over several iterations, estimating the Hessian inaccurately could severely bias the refinement towards inappropriate regions. One can always manually set $\overline{\mathbf{H}} = \bar{\mathbf{I}} \cdot \mathbf{s}$, where $\mathbf{s}$ is a vector of scale factors, ideally approximating the diagonal of the Hessian. This is commonly done in optimization to improve conditioning, but requires manual tuning, which we want to avoid. It has long been recognized that for certain specific cases a Laplace-Beltrami smoothing operator is a reasonable surrogate of the Hessian and can be used to precondition the optimization [30, 175, 176]. However, for general problem classes this is not valid, although it may still help smooth the optimization [112, 169]. For attainable inverse functionals, where the objective is a sum of quadratic deviations of functionals from their targets

$$\mathcal{J} = \frac{1}{2}\sum_k w_k(\mathcal{F}_k(\mathbf{X}) - \mathcal{F}_k^*)^2 \tag{B.24}$$

the Hessian is

$$\frac{\partial^2 \mathcal{J}}{\partial X_i \partial X_j} = \sum_k w_k \frac{\partial \mathcal{F}_k}{\partial X_i}\frac{\partial \mathcal{F}_k}{\partial X_j} + \sum_k w_k(\mathcal{F}_k(\mathbf{X}) - \mathcal{F}_k^*)\frac{\partial^2 \mathcal{F}_k}{\partial X_i \partial X_j} \tag{B.25}$$

It has often been noted that, "sufficiently" close to the target, the second term becomes negligible [110, 112, 179, 192]. The remaining term is essentially free, since it involves only first derivatives that will have already been computed. Although any design problem could indeed be converted into this form, doing so is problematic. The optimal design is sensitive to the targets $\mathcal{F}_k^*$. To justify the approximation they must be attainable, but to ensure that the objective is sufficiently aggressive, they should be only just *barely* attainable. Second, in certain cases this approximation may rapidly become more inaccurate as the shape control is refined. As shown in Table B.1, for inviscid flow, as the width A of the deformation modes shrinks, the Hessian diagonal (term $\frac{\partial^2 \mathcal{F}_k}{\partial X_i \partial X_j}$ in Equation (B.25)) tends to infinity at $\mathcal{O}(\frac{1}{A})$. In a progressive parameterization approach with localized modes, the proximity to the targets $(\mathcal{F}_k(\mathbf{X}) - \mathcal{F}_k^*)$ would need to shrink at $\mathcal{O}(A)$ to compensate and maintain the accuracy of the assumption.

APPENDIX C

DISCRETIZATION ERROR CONTROL IN OPTIMIZATION

The primary goal of adaptive shape control is to accelerate the rate of design improvement with respect to the number of design simulations. A complementary approach is to reduce the average cost of each simulation. This can be done using a variable-fidelity approach, where the mesh is progressively refined as the optimum is approached. The bulk of the design improvement can be obtained using coarser flow meshes, with higher resolution added only when necessary to accurately resolve the design space near the optimum. It is advantageous to combine this sequencing approach with output-based mesh adaptation, which focuses resolution on the regions essential to accurately compute the aerodynamic objective and constraints. This also provides error estimates that can be used both to evaluate the credibility of an optimal design and to set accuracy targets throughout optimization.

In this work, a flow mesh is automatically generated for each design iteration. Using the method of adjoint-weighted residuals [193], the mesh is adapted to reduce error in an aerodynamic functional.[1] This also provides mesh convergence information for each design, along with an estimate of the final discretization error. An example of this process is shown in Figure C.1, which is taken from the symmetric airfoil optimization problem in Section 6.1. Various design iterations required radically different meshes, in terms of both

[1]When there are multiple aerodynamic design functionals, it is often convenient to define a combined mesh adaptation functional that seeks to simultaneously resolve all the outputs [94].

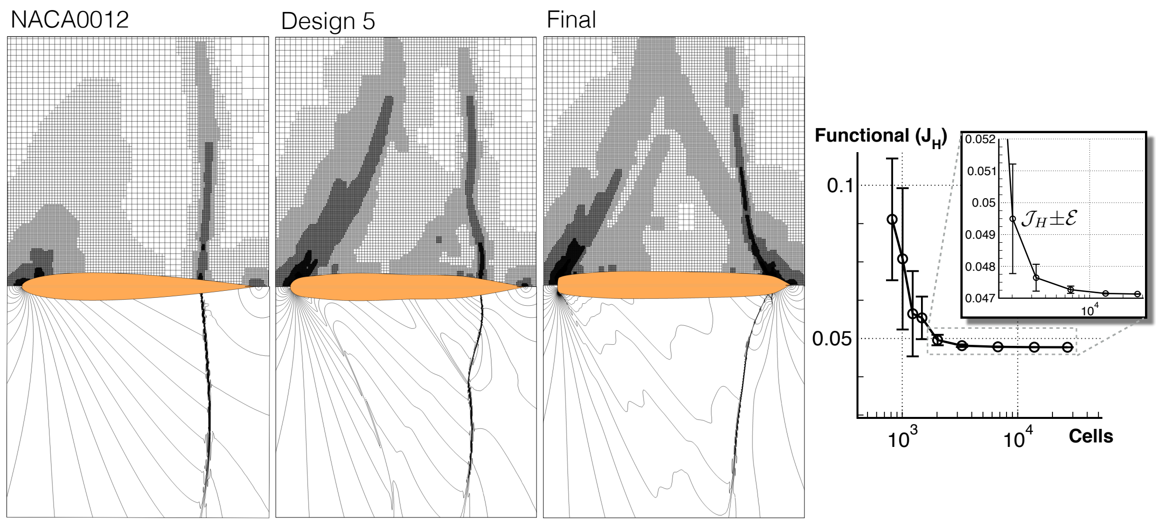
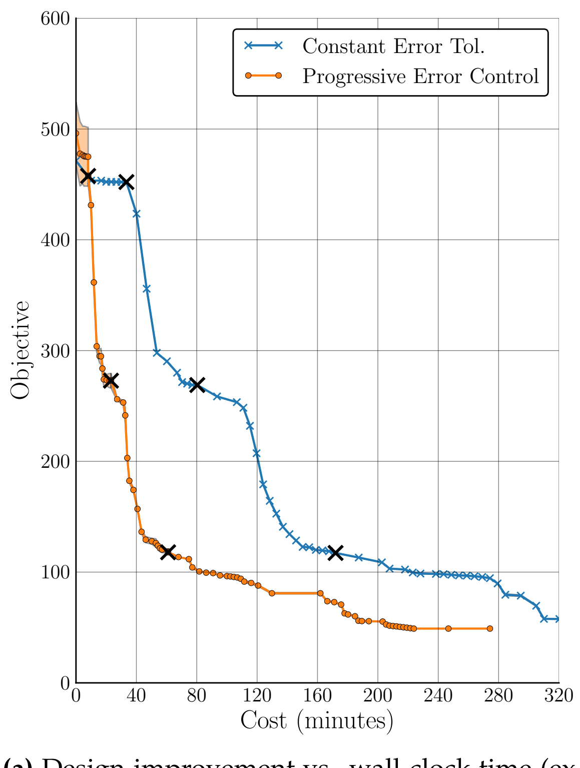


Figure C.1: ***Symmetric Airfoil:*** (See Section 6.1 for case details.) *Top*: Flow meshes adapted to accurately compute pressure drag for three airfoils encountered during optimization. *Bottom*: Mach contours. *Right*: Convergence of drag functional with mesh refinement for the baseline. Bars indicate uncertainty in drag and asymptotically bound the actual changes in the functional.

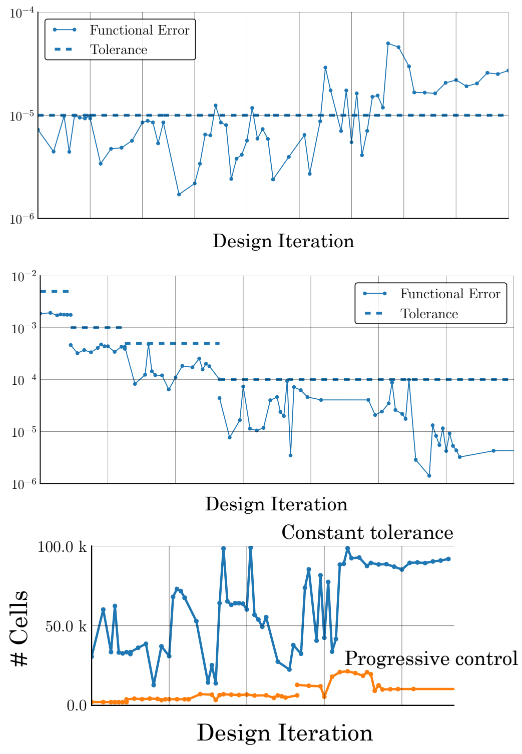
resolution and distribution. An example of convergence of the drag functional and its error bounds is shown in the right frame of Figure C.1.

One major advantage of adaptive meshing during optimization is that it removes the burden of trying to hand-craft a fixed mesh that anticipates how critical flow features will evolve as the design progresses. As the shape deviates more and more from the baseline, a fixed mesh typically becomes less appropriate, leading to higher solution error as the design evolves. This automatic adaptive meshing approach is expected to be especially advantageous for unfamiliar problems that exhibit substantial, unpredictable differences between the initial and final designs. Naturally, a naive and indiscriminate application of high-resolution adaptive meshing can greatly increase the computational expense. However, because we can monitor and control the output error, the solution accuracy can be selectively reduced during the early stages of optimization and then automatically sharpened as the design approaches optimality [101]. This reduces up-front costs (in both user and computational time) and also gives more credibility to the final design.

As an example of the potential for computational savings, consider the symmetric airfoil problem from Section 6.1, where I used a constant error target throughout the optimization. The cell count required to satisfy this tolerance gradually increased throughout optimization (Table 6.1). This indicates that the optimal design has become much more sensitive to the discretization. This is not surprising — the final design has weaker shocks and a much larger zone of dependence, which makes the effect of numerical dissipation more noticeable. The bulk of the design improvement can, however, be obtained much more efficiently by starting with a relaxed error tolerance and progressively tightening it. Figure C.2a shows that this approach results in significant cost savings over a fixed-tolerance approach. Moreover, optimization is performed at high accuracy near the optimum, which ensures the credibility of the final design.



(a) Design improvement vs. wall-clock time (excluding geometry generation). Shaded bands indicate uncertainty due to discretization error. ×-marks denote search space refinements.



(b) *Top:* Constant error tolerance and actual error estimate history. (Tolerance cannot always be satisfied within the maximum allowed mesh refinement depth.) *Center:* Progressive error control and actual error estimate history. *Bottom:* Cell count history

Figure C.2: Case I: Comparison of fixed error control vs. progressive error control. Both cases were performed with identical parameterization strategies and on identical hardware (2013 MacBook Pro with a 2.6GHz Intel Core i7 and 16GB of memory).