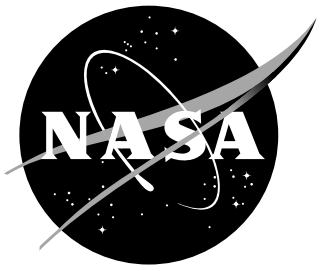
NAS Technical Report: NAS-2015-02



Adaptive Shape Parameterization for Aerodynamic Design

George R. Anderson
Stanford University, CA

Michael J. Aftosmis
NASA Ames, NASA Advanced Supercomputing Division, Moffett Field, CA

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, CA 94035

May 2015

Acknowledgments

Funding for this project was fully provided by a NASA ARMD Seedling Fund grant. The authors also thank Marian Nemec for many helpful discussions and for his development of the static-parameterization design framework used in this study. The aeroelastic design studies are the work of David Rodriguez, who has also contributed to development of the geometry platform. This work was supported by a Phase II Seedling effort through NASA's Aeronautics Mission Directorate, and computational support was provided by the NASA Advanced Supercomputing Division.

Available from:

NASA Advanced Supercomputing Division
Mail Stop 258-5
Moffett Field, CA 94025-1000

Abstract

This report concerns research performed in fulfillment of a 2.5-year NASA Seedling Fund grant to develop an adaptive shape parameterization approach for aerodynamic optimization of discrete geometries. The overarching motivations for this work were the potential to radically reduce manual setup time and achieve faster and more robust design improvement, especially for problems where many design variables are required. The primary objective was to develop a fully-functional prototype system that automatically and adaptively parameterizes discrete geometries for aerodynamic shape optimization. In support of this, we also matured the discrete geometry engine that underlies the work. Various applications are presented that demonstrate broad support and uptake for the discrete geometry tools developed in this work. This report outlines our theoretical approach to adaptive parameterization, provides detailed, practical implementation notes, and contains several verification studies and design examples. These studies demonstrate substantial wall-clock time computational savings, smoother shapes, and superior final designs compared to a standard static-parameterization approach. They also demonstrate that the adaptive system can autonomously discover the parameters necessary to solve an optimization problem. As a whole, this work is an important step towards greater automation in solving the unfamiliar aerodynamic shape design problems of the future.

Nomenclature

Z: Scalar
$\mathbf{Z}$: Vector
Z: Continuous
$\mathcal{Z}$: Function

C, $\mathbf{C}$	Shape control description
$\mathcal{C}$, $\boldsymbol{\mathcal{C}}$	Constraint functional(s)
D	Design description (shape plus design conditions)
$\mathbf{D}$	Diagonal scaling matrix or approximation of Hessian diagonal
$\mathcal{D}(\mathbf{X})$	Shape deformation function
$\mathcal{F}$	Design functional (objective or constraint)
$g, \mathbf{g}$	Growth rate(s) in number of parameters
I	Importance indicator
$\mathcal{J}$	Objective functional
$N_{(\cdot)}$	Number of $(\cdot)$
$\mathcal{O}$	Asymptotic order of computational complexity
$\mathcal{P}(\mathbf{C})$	Shape parameterization function
$\mathbf{Q}$	Flow solution
r	Slope reduction factor for trigger
S	Shape
$\mathbf{S}$	Discrete (tesselated) surface
w	Window width
$X, \mathbf{X}$	Design variable value(s)
$\boldsymbol{\Theta}$	Operating conditions
ψ_o, ψ_j	Adjoint solutions for objective and constraints

Subscripts

$(\cdot)_c$	Candidate shape control
$(\cdot)_G$	Gradient
$(\cdot)_H$	Hessian
$(\cdot)_\times$	Static

Abbreviations

BFGS	Broyden-Fletcher-Goldfarb-Shanno algorithm
DV	Design variable
KKT	Karush-Kuhn-Tucker conditions

1 Introduction

SHAPE optimization is fast becoming an indispensable component of aerodynamic design. Most current approaches to aerodynamic shape optimization rely heavily on user training and experience with a range of specialized subtools. To a certain degree this is essential; design decoupled from understanding and experience is inconceivable. Nevertheless, manual setup of these tools typically involves a large number of *non*-design-related details which can require deep expertise to master. Unfortunately, these non-design tools can still significantly impact the validity, robustness and cost of optimization.

Consider the four most universal optimization subtools, in the order they typically appear in design: (1) the geometry modeler, (2) meshing tools, (3) the flow solver, and (4) the optimizer. Encouraging strides toward automation have been made with respect to the last three tools. Regarding the geometry modeler, automated shape deformation tools are increasingly commonplace. The major missing link, however, is the relative lack of automatic shape *parameterization* tools. This constitutes a major obstacle to robustness and consistency, as the parameterization sets and limits the range of reachable shapes. Different designers will craft distinct parameterizations and thus obtain different optima, even to otherwise identically-posed problems. Expert-crafted parameterizations strike a compromise between the speed of the optimization (favoring a compact parameterization) and the capacity for design improvement (detailed parameterizations and a high dimensional design space). Indeed, this fundamental tradeoff between cost and capacity is the primary motivation for manually constructing shape parameterizations in the first place. Regardless of the cost balance struck, the final design is certain to be suboptimal – the design space was limited by the parameterization. This often leads to subsequent, expert-guided, redesigns wherein additional subsets of parameters are added in an attempt to further advance the objective function. Furthermore, a designer must be aware of additional potential pitfalls of manual parameterizations, such as bias towards familiar designs and repetitive and error-prone setup procedures.

This work demonstrates that, to a large degree, the construction of the shape parameterization can be automated. Sufficient information is unearthed during optimization to make intelligent, automated decisions about the shape parameterization. During optimization, this information can be used to automatically refine and evolve the shape parameterization, both in terms of the number of design variables and their distribution. The primary goals of this approach are to reduce manual setup time, to alleviate the need for user-in-the-loop reparameterizations, and to reduce the dependence of the result on the skill of the designer at choosing the best possible parameterization. An appealing side-effect is that the system tends to achieve faster and more robust design improvement compared to "obvious", but non-expert-crafted parameterizations. Ultimately, the hope is to approach expert-performance with an automated system, thereby rendering parametric shape optimization more fruitful to aerodynamic designers who are not necessarily highly trained in the various codes, frameworks and tools involved.

1.1 Limitations of Static Parameterizations

Under a traditional *static* parameterization approach, the space of all reachable shapes is prescribed before each optimization begins. This can restrict the design space in unnecessary ways, needlessly hindering the discovery of superior designs outside this envelope. One recourse is to use a very large number of design variables. However, as shown in Figure 1, while finer parameterizations can reach superior designs, they take longer to converge, even with BFGS gradient-based optimizers, which typically converge in $\mathcal{O}(N_{DV})$ design iterations. The designer strives to find an optimal balance in the number of design variables. Using too many leads to inefficient navigation, while using too few restricts adequate exploration of the design space. In practice, frequently a designer will typically perform an optimization and then manually refine the parameterization and restart the design. This is clearly a time-consuming and labor-intensive approach.

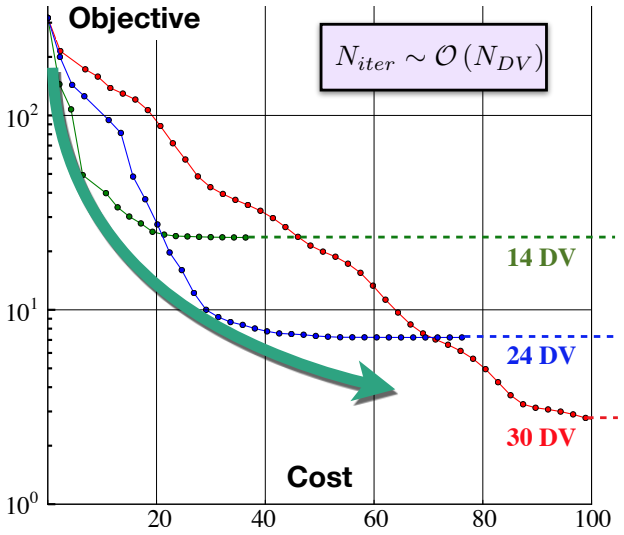


Figure 1: BFGS-style optimization converges in $\mathcal{O}(N_{DV})$ search directions. A progressive parameterization can follow the "inside track", making rapid gains early, while still approaching the continuous optimal shape.

1.2 Progressive Parameterization

In this work, we adopt a *progressive* parameterization approach, where optimization begins in a coarse search space, and then automatically transitions to finer parameterizations at strategic moments. The basic idea is to first optimize in low-dimensional search spaces focusing on the manipulation of gross features. We then automatically introduce more dimensions with finer shape control to drive towards the optimal shape. Importantly, this approach can be both efficient and complete, by eliminating the arbitrary tradeoff caused by using a static parameterization. It constitutes a single process that drives the shape towards the local optimum of the continuous problem, while remaining efficient early on.

Figure 2 illustrates the basic approach. The designer first specifies an initial low-dimensional shape parameterization. As the design evolves, higher-resolution shape control is automatically added, removing the restrictions of optimizing within a static parameterization. Rapid design improvement is encouraged by using compact search spaces early on; more degrees of freedom are introduced only when necessary to further improve the design. In the limit of uniform shape control refinement, the full design space of the continuous problem gradually becomes available for exploration.

The primary benefit of a variable shape control approach is that the designer is no longer burdened with predicting, *a-priori*, how many and which design variables will be

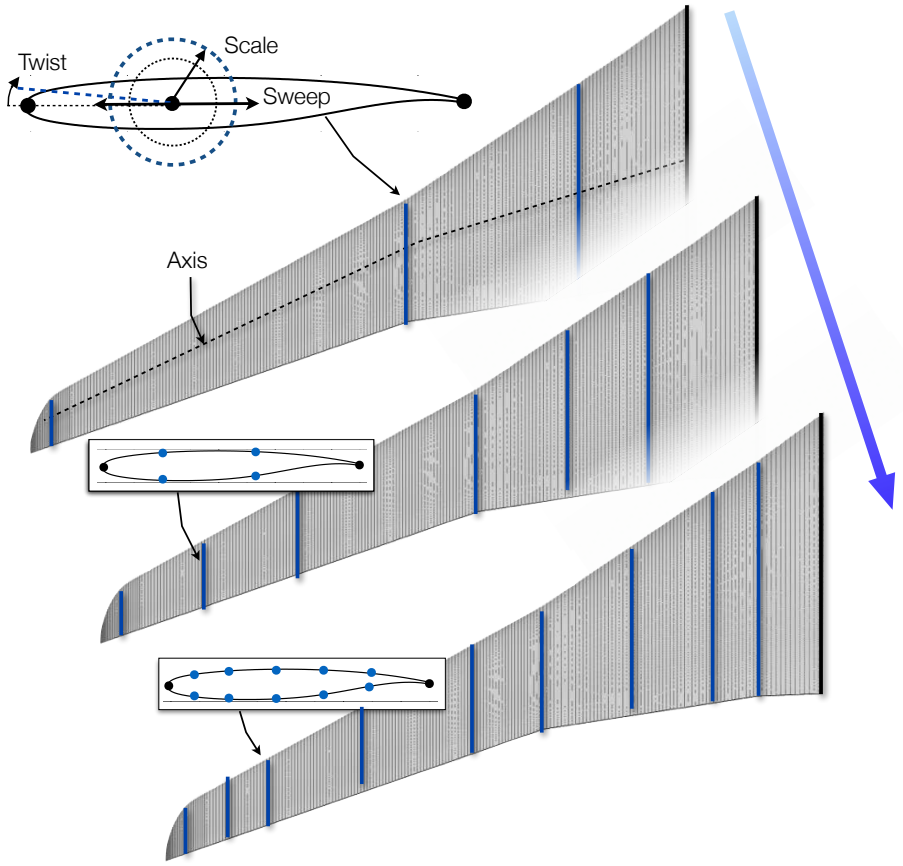


Figure 2: Search space refinement for wing design. Airfoil control is refined independently on each section.

appropriate for a particular design problem. It also radically reduces user setup time, as design variables are automatically created and bounded. Furthermore, it constitutes a single hierarchical process that progressively drives the shape towards the true continuous optimal shape, instead of an approximation in an arbitrarily fixed design space. Although our primary goal is to streamline and automate design tools, using a variable approach is also motivated by a growing body of evidence that substantial design acceleration can be achieved by using a hierarchical parameterization approach.[1–4]

1.3 Enabling Technologies

1.3.1 Adjoint Sensitivity Information

A key enabling technology for our approach is the incorporation of sensitivity information provided by the adjoint equations. Since its introduction to the aerodynamic community 25 years ago,[5] the adjoint method has revolutionized gradient-based shape optimization, rendering it computationally feasible to optimize on very large numbers of design variables. The adjoint approach allows *all* of the objective gradients to be computed for a fixed cost of roughly two PDE solutions, instead of the $2N$ solutions required under a finite difference

formulation.

This work employs the adjoint in an additional and novel role. Sensitivities in the adjoint encode much more information than traditional parametric shape optimization makes use of. This information can be extracted at trivial cost, and, if harnessed, can accelerate the rate of design improvement. Specifically, we use the adjoint to compute gradients not only with respect to existing shape design variables, but also with respect to *potential* design variables. We can then determine whether these candidate design variables would drive the design forward more effectively, and if so, inject them into the active set. Different design problems may call for different shape control. Our goal is to automatically discover the necessary shape control as the design evolves. Thus we expand the traditional role of the adjoint from efficiently computing objective gradients to include adaptive refinement of the search space.

1.3.2 Discrete Geometry Manipulation Tools

Geometry manipulation is central to shape optimization, and even more so when moving to automatic parameterization. One key objective of this work was to develop and mature a tool that allows discrete surfaces to be parameterized with design variables automatically placed in any location. This supports both manual and automatic parameterization approaches. To modify geometries during optimization, and to reparameterize them on the fly, we leverage discrete geometry tools from the computer graphics (CG) industry and customize them for aerospace design. The CG industry has invested for decades in general geometry manipulation techniques. These tools are designed to serve as automated geometry deformation engines and offer access to a wealth of flexible deformation techniques and efficient surface manipulation routines. They are also highly extensible via back-end scripting interfaces.

Section §2 discusses our geometry platform in more detail. This platform can incorporate the many deformation techniques developed by the aerospace community in recent years as plugins. We can rapidly prototype new deformation techniques on the standard, unified geometry manipulation platform it offers. Mature graphical user interfaces (GUIs) enable designers to interactively parameterize discrete geometries during preparation for an automated shape optimization study. Section §2.3 presents several applications of this toolset outside the immediate context of the present work.

1.4 Outline

In Section §2 we discuss the geometry platform that underlies the entire progressive parameterization system. In addition to describing the shape parameterization methods used for this study, we also highlight various spin-off applications that have successfully leveraged this discrete geometry platform. Thereafter we cover the basic algorithm and approach to progressive design (Section §3) and discuss the details of our particular implementation (Section §4). Section §5 gives an approach to adaptive shape parameterization, where we seek to add only the most important shape control. Finally, evaluation studies and design examples are given in Sections 6 and 7.

2 Discrete Geometry Platform

Throughout this work 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.[6,7] It allows Blender to serve as a geometry engine for shape optimization, providing on-demand deformations, along with analytic shape sensitivities. For this work we further extended and improved this platform in several ways. First, we developed additional custom shape parameterization plugins, which are described in the following sections. Plugins to support aerostructural analysis and design were also developed.[8,9] Finally, support has been added for adaptive parameterization.[10,11]

2.1 Curve Deformer

To deform curves (such as airfoils or cross-sections of wings or fuselages), we use a "direct manipulation" approach, illustrated in Figure 3. The deformation of certain "pilot points" placed along the curve are explicitly specified. These points serve as the design variables. Deformation of the remainder of the curve is smoothly interpolated using radial basis functions (RBF). Derivations of the RBF deformation technique as applied to aerodynamic design are given by several authors.[12–15] Each parameter has a bump-shaped deformation mode that is mostly confined to the region between its neighboring points, while maintaining smoothness. We chose the basis function $\phi = r^3$ here, primarily because it requires no local tuning parameters, making it more amenable to automation.

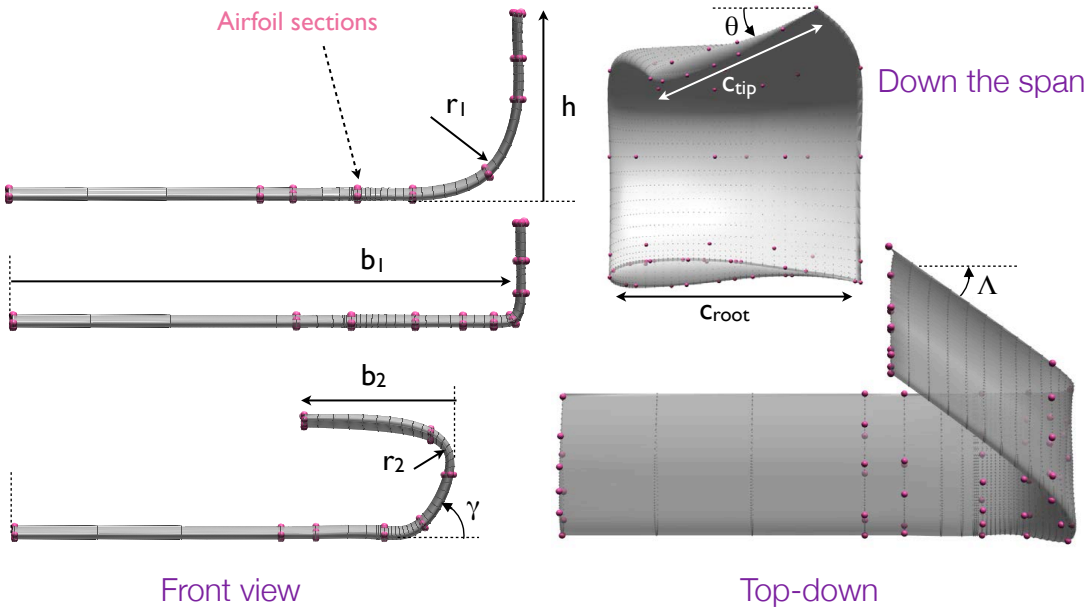


Figure 3: Parametric wingtip deformation using constraint-based deformation.

2.2 Wing Deformer

For parametric manipulation of wings, we use the technique illustrated in Figure 2, which linearly interpolates twist, sweep, chord and airfoil shapes between spanwise stations. At each station a curve deformer (described in the previous section) deforms the airfoil shape, after which the twist, sweep, and chord modifications are applied. Interpolation of twist, sweep, chord and airfoil shape happen independently, and so the shape control may be refined anisotropically, to give greater resolution of one type of control than another. Similarly, each airfoil control station can offer different streamwise shape control resolution. To refine the shape control, more spanwise stations may be added, or new degrees of freedom may be added to individual curve parameterizations.

2.3 Applications

Before turning the main thrust of this work, namely the development of *automatic* parameterization tools, we find it encouraging to note that the discrete geometry tools developed as part of this work have already proved useful in various real-world design settings. In fact, an important goal of the present work was to enable automated shape optimization while simultaneously improving the geometry tools available in existing user-driven design environments. Two examples of the latter are briefly presented here.

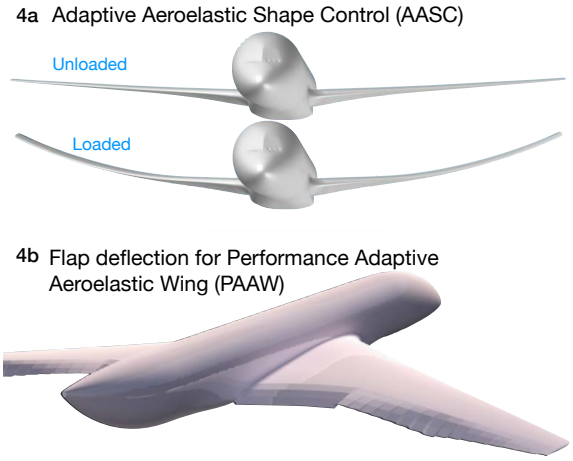


Figure 4: Use of the discrete geometry deformation platform developed in this work for support of ARMD initiative on airframe efficiency

This work integrates directly into NASA's Cart3D Design Framework via an XML-based protocol, which enables it to plug in to any other design system that subscribes to this protocol. This integration has fed into several projects related to aeroelastic analysis[8] and design.[9] Further, it supported a twist optimization study regarding the design of a highly flexible wind tunnel model tested at the University of Washington's Aerospace Laboratory in 2013 and 2014. Figure 4 illustrates two design initiatives supported by our geometry platform as part of ARMD's initiative on airframe efficiency. The lower frame in this figure highlights work done for analysis of a Performance Adaptive Aeroelastic Wing (PAAW) in which a Variable Camber Continuous Trailing Edge Flap (VCCTEF) was used to actively modulate the lift distribution of the wing throughout the mission profile. This was a complex optimization task involving not only aerostatic deformation of the flexible wing but also articulation and deflection of the 43-segment VCCTEF. Reference [9] presents further detail of this work.

Figure 5 illustrates an application of the geometry platform performed in support of ARMD's Commercial Supersonic Transport project to assess aeroelastic effects on sonic boom signatures. This work was task T3.32 (milestone #33213) under the High Fidelity Analysis & Validation (HiFAV) project element. The constraint-based deformation pack-

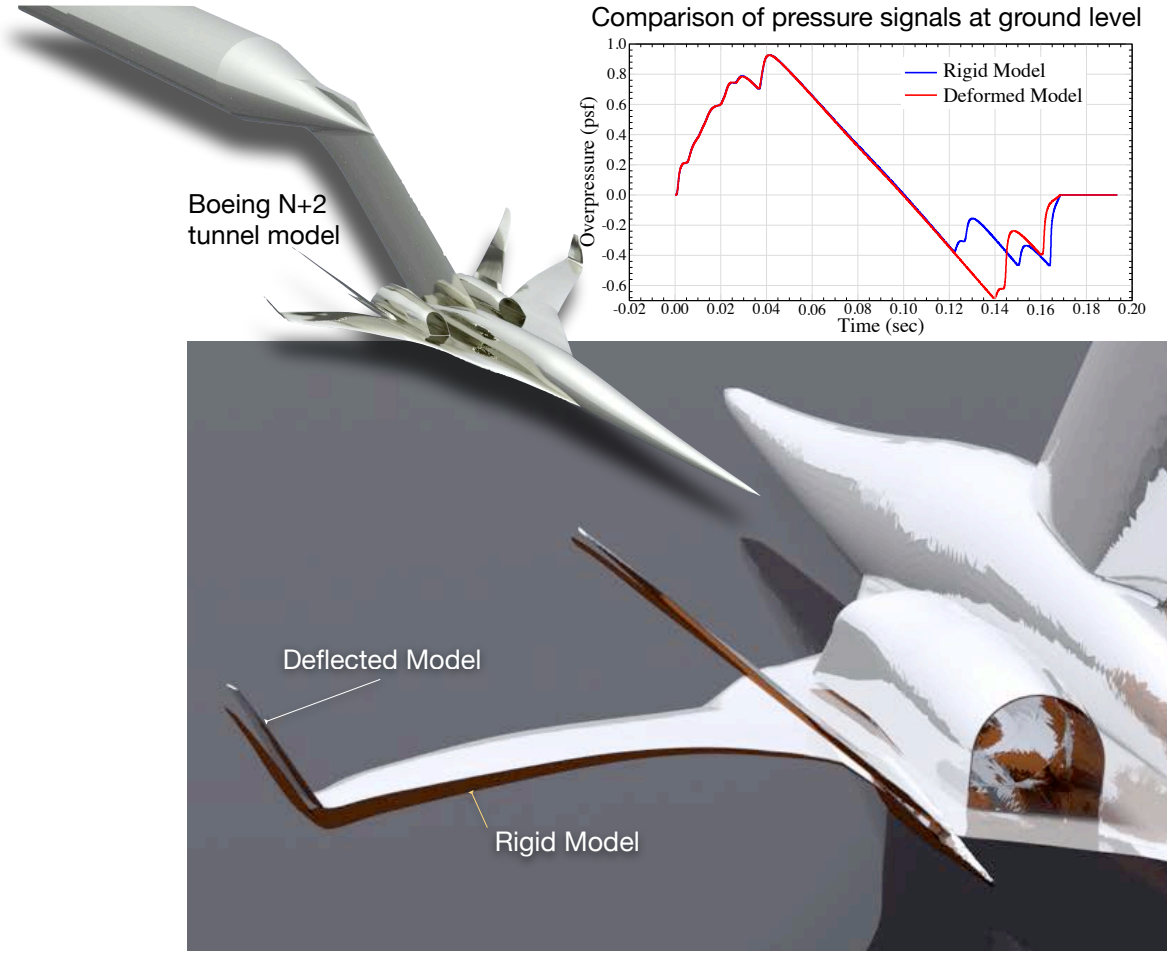


Figure 5: Support for ARMD initiative on sonic-boom control for development of a Commercial Supersonic Transport, (HiFAV project, Task T3.3.2, milestone #33213).

age developed as part of the geometry platform of the preceding section was used to apply measured tunnel deflections to CFD-ready CAD models used for simulations. After applying the deformation, the model was re-analyzed yielding quantitative prediction of the effects of aeroelastic deformation on the boom footprint.

Figure 6 shows a final example done as part of the Advanced Air Transport Technology Project's work on element AATT.02.01. This work further quantified opportunities afforded by the VCCTEF on elastically shaped aircraft. In this effort, the geometry platform outlined in the preceding section was used to install and articulate the multi-segment flap on an aircraft with a truss-braced wing. This optimization exercise orchestrated flap deflection to minimize the combination of both wave drag and induced drag at cruise conditions and use of the VCCTEF to for load alleviation. The work involved autonomously calling the discrete geometry engine from within the design cycle to control and coordinate flap deflection. This milestone was reported to the AATT program in March 2015 by the authors of [9].

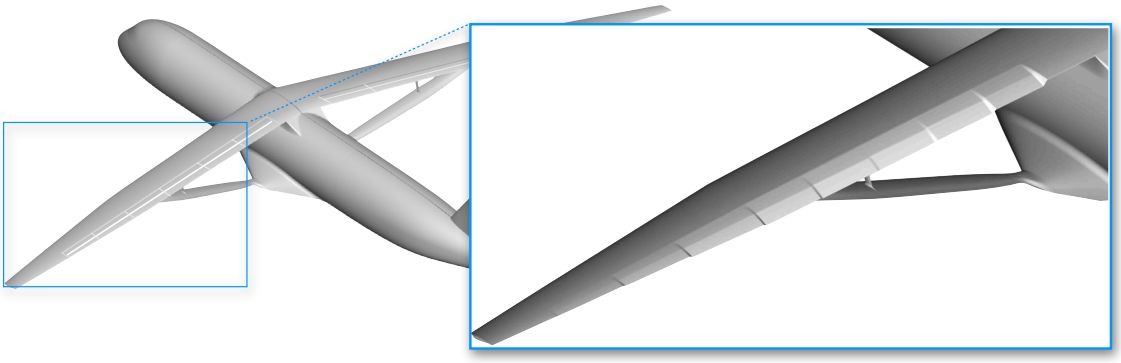


Figure 6: Application of the discrete geometry platform to support NASA Advanced Air Transportation Technologies Project program element AATT.02.01. This work examined design and optimization of VCCTE flap system on a truss-braced-wing aircraft to achieve minimum cruise drag & active load alleviation.

3 Optimization with Variable Shape Control

The aerodynamic shape optimization problem we consider consists of finding a design D that minimizes a single objective $\mathcal{J}$, subject to design constraints $\mathcal{C}_j$:

$$\begin{aligned} &\min_{\mathsf{D}} \mathcal{J}(\mathsf{D}) \\ &\text{s.t. } a \leq \mathcal{C}_j(\mathsf{D}) \leq b \end{aligned} \tag{1}$$

where $\mathcal{J}$ and $\mathcal{C}_j$ are scalar functionals that involve specific performance metrics such as lift, drag, range, stability margins, maneuver loads, or wing volume, possibly integrated over multiple flight conditions. They may also relate to higher level metrics like operating range or cost, or to more specialized concerns such as reducing sonic boom ground signatures or reducing environmental impact. The design $\mathsf{D} = (\mathsf{S}, \boldsymbol{\Theta})$ consists of a shape S and a set of operating conditions $\boldsymbol{\Theta}$ that are independent of the shape, such as angle of attack, Mach number, altitude, or throttle settings.

In this work, all design functionals considered depend only on the external surface. This includes aerodynamic functionals such as lift or drag, which depend only on the wetted surface or "outer mold line", and also geometric functionals, such as wing volume or thickness, that can be measured with only an external surface description. For our purposes, the term "shape" will denote the external surface, while internal layout details or material properties will be ignored. The approach developed here is, however, applicable to other disciplines. In aerostructural design, for example, the "shape" description would include both the external surface and internal geometry, such as spars and ribs.

Whether directly or indirectly, the values of all functionals are ultimately determined by the shape. In the case of aerodynamic functionals, $\mathcal{J}$ and $\mathcal{C}_j$ are evaluated after solving for the flow variables $\mathbf{Q}$.[a] For gradient-based optimization, the derivative of each output functional must be computed with respect to the design. In the case of aerodynamic functionals, the functional gradients are most efficiently computed using an adjoint approach.

[a] $\mathbf{Q}$ can just as easily represent other PDE solutions, such as structural deformation.

3.1 Shape Parameterization

While the operating conditions $\boldsymbol{\Theta}$ typically comprise a small discrete set of design variables, 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 function[b], $\mathcal{D}$, of a set of shape parameters $\mathbf{X}$:

$$\textbf{(Parameterize)} \qquad \mathcal{P} : \mathbf{C} \longrightarrow \mathcal{D}(\mathbf{X}) \tag{2}$$

This deformation function defines the search space for optimization. It determines the span of reachable designs, which is (intentionally) only a subset of the full shape design space supported by the continuous surface. The shape parameters $\mathbf{X}$, or a subset thereof, serve as the design variables for optimization. During optimization, the deformation function takes the design variable values and generates a new surface:

$$\textbf{(Modify Shape)} \qquad \mathcal{D} : \mathbf{X} \longrightarrow \mathsf{S} \tag{3}$$

The local linearization of $\mathcal{D}$ provides the shape derivatives $\frac{\partial \mathsf{S}}{\partial \mathbf{X}}$, which describe the deformation modes of each parameter. These shape derivatives are used in gradient-based optimization, allowing projection of the functional gradients with respect to the surface, $\frac{\partial \mathcal{F}}{\partial \mathsf{S}}$, into the compact search space spanned by the shape parameters.

In short, Equation (2) describes how the shape control induces a set of shape parameters, while Equation (3) describes how those shape parameters deform the surface. In standard optimization approaches, only Equation (3) is automated. One of the core differences of the present work is that Equation (2) is also automated.

The distinction drawn here between the shape *control* $\mathbf{C}$ and the shape *parameters* $\mathbf{X}$ is important. Each optimization level involves a search in the space spanned by $\mathbf{X}$, while holding $\mathbf{C}$ fixed. When transitioning to the next search space, the shape control $\mathbf{C}$ is modified, while the shape is held fixed. For many modelers, there is not always a one-to-one correspondence between the shape control and the shape parameters, a point that is important in the following section.

3.2 Progressive Shape Control

In standard shape optimization approaches, the shape control $\mathbf{C}_\times$ is static and predetermined by the designer. This induces a static 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. A progressive approach uses instead a sequence of shape control resolutions $\mathbf{C}^0, \mathbf{C}^1, \mathbf{C}^2 \ldots$, which generate a sequence of search spaces $\mathcal{D}_0(\mathbf{X}_0), \mathcal{D}_1(\mathbf{X}_1), \mathcal{D}_2(\mathbf{X}_2) \ldots$ that permit ever more detailed shape control. The designer provides only the initial shape control $\mathbf{C}^0$. After an optimization in this design space, the shape control is automatically refined, and optimization continues in the

[b] In this work we use shape *deformation* techniques, where modifications of an existing surface are parameterized. However, the following discussion is fully applicable to constructive parameterized surface generation as well.

higher-dimensional search space.[c]

The purpose of this approach is to automate the generation of nested search spaces of increasing resolution. However, the designer is still responsible for establishing a basic framework for this process. Specifically, the designer selects a *class* of shape control (e.g. twist vs. airfoil deformation), specifies an initial coarse parameterization, and indicates how the shape control may be refined. Automatic refinement is performed within this user-defined framework.

For example, Figure 7 illustrates nested search space refinement as applied to airfoil design. Instead of providing a static set of design variables, the designer establishes a general shape control framework. This may involve establishing important design features as parameters or constraints, such as the leading and trailing edges or spar locations (black and orange dots in Figure 7). These features partition the curve into several regions. Initially, a single shape controller is placed in each region (blue dots). Thereafter, the shape control is automatically refined when and where necessary.

3.3 Binary Refinement

In this work, we adopt a nested, hierarchical search space refinement technique, with a discrete approach to adding design variables, akin to h-refinement in mesh adaptation. (In other words, optimal continuous positioning of the shape controllers is not considered.) The shape control can thus be encoded as a binary tree, as depicted in Figure 7, with deeper levels corresponding to higher resolution shape control. Starting from a single "root" controller, finer shape control is gradually introduced through binary refinement of each "leaf". In the limit of refinement, this sequence converges to continuous shape control that covers the shape being designed.

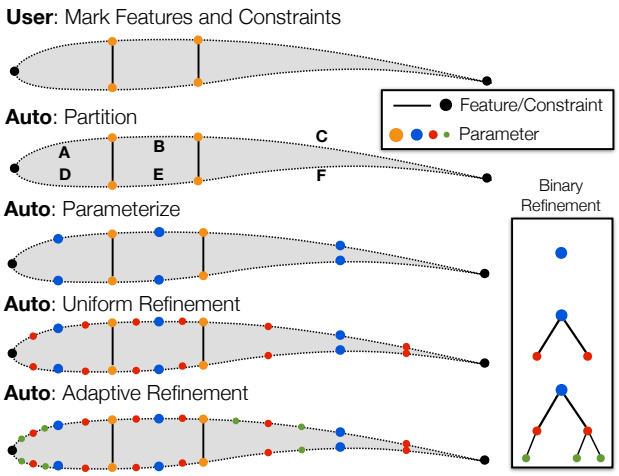


Figure 7: Progressive parameterization with discrete, hierarchical shape control refinement

Note that Figure 7 depicts design of a simple curve; the shape control can thus be represented with a single tree. For design of a 3D surface, the shape control tree has at least two dimensions. Each dimension can be refined independently, leading to anisotropic control. Additionally, different deformation modes may each have their own independent trees.

[c] Although we do not consider it in this work, *removing* design parameters that are no longer useful is also a possibility.

3.4 Uniform vs. Adaptive Refinement

One simple and robust approach to refinement is to simply uniformly refine the parameterization by binary subdivision of each leaf on the tree. However, this raises the issue of the rate of growth in the number of design variables, which has a critical impact on efficiency. Excessively high growth rates introduce large numbers of design variables, leading to search spaces that are slow to navigate. As will be shown, even uniform refinement greatly accelerates design improvement compared to static parameterizations. However, uniform distribution of shape control is generally suboptimal, implying that even more gains in efficiency are possible.

This immediately raises the possibility of "adaptive" refinement, where we selectively refine only certain regions, as shown at the bottom of Figure 7. The goal of adaptive refinement is to add only the most important shape control to solve the given problem, maximizing design improvement for a fixed number of design variables. This often reduces the total number of design variables required to find the optimum.

3.5 Optimization Loop

The design loop now consists of a nested sequence: optimize within the current search space, and then refine the shape control. This process is illustrated in Figure 8, where the inner loop represents a standard parametric shape optimization framework that optimizes a shape in a static search space. This inner-outer loop is detailed more explicitly in Algorithm A. The function $Optimize(\cdot)$ represents a standard parametric shape optimization framework. $Parameterize(\cdot)$ is the modeler-dependent implementation of Equation (2), which generates a search space (i.e. a deformation function and design variables) from the shape control description. Unlike in static optimization approaches, where this is a manual pre-processing step, here it is automated.

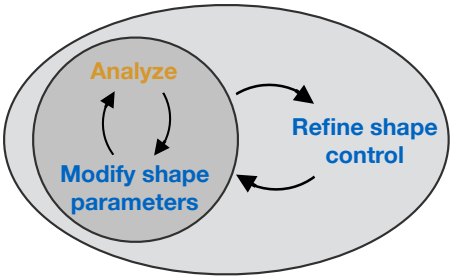


Figure 8: Optimization loop with periodic search space refinement

The refinement strategy is governed by three new functions, each of which will be discussed in more detail in the following chapters. First, the $Trigger(\cdot)$ monitors the optimization to determine *when* to refine the shape control. Next, the modeler-dependent $GetCandidateShapeControl(\cdot)$ generates a list of possible locations *where* the shape control may be refined. Finally, some or all of these candidates are marked for refinement. The simplest approach is to add all the candidates to the active set, which will be called "uniform" refinement. Alternatively, the system can try to predict which subset of the candidates would best enrich the search space. This adaptive process is represented by $AdaptShapeControl(\cdot)$, a search procedure that chooses an effective subset of the candidates. Finally, $Parameterize(\cdot)$ generates the refined search space. It also manages the transfer of design variable bounds and scale factors from the previous design space, and if possible, ensures that the new shape is identical to the final previous shape.

The ultimate convergence criterion of Algorithm A is based on objective convergence as

the discrete shape control $\mathbf{C}$ approaches continuous shape control (direct optimization of $\mathbf{S}$) such as used in one-shot optimization methods. In practical settings, the optimization may be terminated well before this degree of convergence is attained.

4 Implementation

The nested optimization procedure of Algorithm A integrates three basic software components: (1) a geometry modeler, (2) a shape optimization framework, and (3) scripts to guide search space refinement. In this chapter, we discuss the approach to each component in the context of developing a system to orchestrate them. 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.

The first section discusses general requirements that a geometry modeler must satisfy to enable automatic generation of search spaces for shape design. (Recall that Section §2 discussed the specific discrete geometry modeler used for this work.) In Section §4.2, we discuss the use of an existing static-parameterization shape optimization framework to solve each level of the variable shape control optimization problem. Finally, Section §4.3 introduces a refinement strategy that addresses the question of when to refine the shape control to maximize efficiency.

4.1 Geometry Modelers — Universal Requirements

Consider a standard gradient-based design framework where the geometry modeler, whether constructive or deformational, generates a discrete CFD-ready surface triangulation upon

Algorithm A: Optimization with Adaptive Shape Control

Input: Initial surface $\mathbf{S}_0$ and shape control $\mathbf{C}^0$, objective $\mathcal{J}$, constraints $\mathcal{C}_j$, shape control growth rate $\mathbf{g}$

Result: Optimized surface $\mathbf{S}$

$\mathbf{C} \longleftarrow \mathbf{C}^0, \mathbf{S} \longleftarrow \mathbf{S}_0$
repeat
  $\mathcal{D}, \mathbf{X}_0 \longleftarrow \mathit{Parameterize}(\mathbf{S}, \mathbf{C})$
  $\mathbf{S} \longleftarrow \mathit{Optimize}(\mathcal{D}, \mathbf{X}_0, \mathcal{J}, \mathcal{C}_j)$ **until** $\mathit{Trigger}(\cdot)$
  $\mathbf{C}_c \longleftarrow \mathit{GetCandidateShapeControl}(\mathbf{C})$
  if adaptive **then**
    $\mathbf{C} \longleftarrow \mathit{AdaptShapeControl}(\mathbf{C}, \mathbf{C}_c, \psi, \mathbf{S}, \mathbf{g}[i])$
  else
    $\mathbf{C} \longleftarrow \mathbf{C} \cup \mathbf{C}_c$ `// Uniform refinement`
  end
until *convergence of* $\mathcal{J}$ *and* $\mathcal{C}_j$ *w.r.t.* $\mathbf{C}$

Function color key:
Parametric geometry modeler
Refinement strategy (modeler independent)

demand from the design framework, annotated with shape derivatives for each design variable. In this context, the geometry modeler must:

- ***Before optimization:*** Provide a list of available design variables with lower and upper bounds.
- ***During optimization:*** Given any set of feasible parameter values $\mathbf{X}$:
  - Generate the corresponding surface (Equation (3)).
  - Provide shape derivatives $\frac{\partial \mathbf{S}}{\partial X_i}$ for each design variable.[d]

A geometry modeler integrated with a shape optimization framework already satisfies these requirements.[e] However, moving to *progressive* parameterization places greater emphasis on automated, API-based access. Certain traditionally manual tasks must be automated in a consistent and robust manner. These tasks include setting design variable minimum and maximum bounds and scale factors, and modifying control parameters and settings in geometry modeler input files. Additionally, a small amount of modification to the geometry modeler may be required to allow the optimization system to automatically invoke refinements of the shape control. For modelers where the parameterization is specified in simple input control files, this is simply a matter of creating a script-based interface to the progressive parameterization system. For modelers that mandate GUI-based access to change settings or to re-parameterize a shape, it may involve modification of the geometry modeler itself.

4.1.1 Automatic Generation and Refinement of Search Spaces

In the context of *progressive* parameterization, the geometry modeler must implement additional script-based methods that

- ***Before refinement:*** Provide a list of possible shape control refinements — $GetCandidateShapeControl(\cdot)$ in Algorithm A.
- ***During refinement:*** Generate a deformation function $\mathcal{D}$ from any set of shape control refinement locations — $Parameterize(\cdot)$ in Algorithm A (Equation (2)).
- ***After refinement:*** Transfer bounds, scale factors and other meta-data from the old search space to the new ones.

The details of these methods depend on the geometry modeler. More concrete examples will be given in Section §2, where we discuss implementation of these functions for a specific geometry modeler. However, there are several universally relevant topics that warrant discussion.

[d] This is not technically a strict requirement; shape derivatives can be computed automatically by finite differencing.

[e] Besides these strict requirements, several authors have discussed various *desirable* qualities, including smoothness, compactness, effectiveness and intuitiveness.[16–20]

4.1.2 Exact Surface Reconstruction

It is desirable that the shape be preserved exactly when refining the parameterization. Olhofer et al. refer to this as a "neutral mutation" of the shape control.[3] In terms of the notation used in this work:

$$\mathbf{S}^k_{final} = \mathcal{P}(\mathbf{C}^k)(\mathbf{X}^k_{final}) = \mathcal{P}(\mathbf{C}^{k+1})(\mathbf{X}^{k+1}_0) = \mathbf{S}^{k+1}_0 \tag{4}$$

where k and $k+1$ are the current and subsequent search spaces. This characteristic is desirable from a computational standpoint. If it is not true, then a re-fitting procedure when re-parameterizing. This refitting is usually approximate, introducing a "jump" in the shape and therefore a setback in the design process. This has particularly been a disadvantage of certain previous attempts at progressive parameterization with constructive modelers.[1,4] In a few notable constructive modeling approaches, however, an exact refitting procedure has been used.[2,21] With discrete geometry,the shape is inherently preserved exactly.

4.1.3 Discrete Binary Shape Control Refinement

Note that at this point, we have deliberately not placed any severe restrictions on the manner of shape control refinement. However, there are certain desirable characteristics of the *structure* of the parameterization scheme:

- Hierarchical ("nested") organization of parameters
- Refinement that can be localized to particular regions of the shape
- Unlimited, or at least substantial, refinement depth
- Exact shape preservation when transferring between levels

In this work, each parameterization is viewed as a binary tree, restricting refinement to the midpoints between existing shape controllers[f], although one can search several levels deep from the current parameterization. To maintain smoothness in the spacing between parameters, we prohibit large discrepancies between the refinement depth of adjacent regions on the surface. This regularization proved to be important for robustness in many cases. Many parameterization techniques support infinitely-scalable shape control resolution, but we usually impose a minimum spacing between adjacent parameters (equivalent to a maximum depth in the binary tree). This prevents the shape control from becoming unreasonably closely spaced and keeps the shape control resolution well outside the resolution of the surface and flow mesh discretizations.

4.1.4 Transfer of Meta-data to New Parameters

Additional questions arise when determining how to propagate optimization data, including minimum and maximum bounds and scale factors. Our basic approach is to linearly interpolate these parameters between existing controllers.

[f]Refinement can also be directionally biased, for example to cluster parameters towards the leading edge of a wing.

4.2 Static Shape Optimization Framework

For the function *Optimize*(·) in Algorithm A, we use a gradient-based aerodynamic shape design framework[22] that uses an embedded-boundary Cartesian mesh method for inviscid flow solutions. Aerodynamic objective and constraint gradients are computed using an adjoint formulation. These same adjoint solutions are later reused to prioritize candidate design variables when refining the search space. Optimization can be handled with any black-box gradient-based optimizer; for this study, the SQP optimizer SNOPT[23] is used, enabling proper treatment of linear and nonlinear constraints. Geometric functionals (e.g. thickness and volume) are computed by a standalone tool that provides analytic derivatives to the functionals. The design framework communicates with all geometry tools via *XDDM*, an XML-based protocol for design markup.[22]

4.3 Trigger for Transitioning Between Search Spaces

To determine when to transition to a finer search space, we use the *Trigger*(·) function in Algorithm A, which is a stopping criterion that terminates the optimization in the current search space and initiates a parameter refinement. A timely and robust trigger is critical for efficiency, as demonstrated in Figure 9. The two branches show the performance impact of triggering at different times, for an airfoil drag minimization problem. Over-optimizing on the initial parameterization leads to long periods of negligible design improvement, while refining the search space earlier results in much faster improvement per cost. Similar observations have also been made by other authors in the context of both adaptive parameterization[1] and optimization with progressively refined PDE meshes.[24]

The approach used in this work is to only partially converge the optimization in each search space, with the goal being to move to the next parameterization at a computationally efficient moment. We ruled out simplistic triggers, such as using a number of search directions that is either fixed (as in[1]) or proportional to the number of design variables. This would demand prior knowledge of the rate of convergence for a problem, which defeats the purpose of having a general and automated system.

4.3.1 Optimality Trigger

One obvious and robust approach is to allow the optimization to converge until an optimality criterion based on the KKT conditions is sufficiently satisfied. Han and Zingg[2] used this approach to achieve maximal design improvement within each search space. However, we found that on many problems, this type of trigger often excessively delays refinement in early search spaces, as demonstrated in Figure 9. This might be remedied by choosing a looser optimality tolerance. However, the magnitudes of the gradients can vary widely, depending on the scaling of the problem. Establishing an efficient cutoff is difficult without prior experience with a particular problem, and this is unattractive in an automated setting.

4.3.2 Slope Trigger

A simple alternative approach is to trigger when the rate of design improvement starts to substantially diminish. From an engineering perspective, this is a pertinent choice, as

design improvement vs. cost is typically the most important figure of merit. To detect diminishing design improvement, the system monitors the slope of the objective (or merit function) convergence with respect to a suitable measure of computational cost. The optimization is terminated when this slope falls below some fraction r of the maximum slope that has occurred so far. This strategy proved to be less sensitive to the cutoff parameter r than the optimality criterion. The normalization by the maximum slope accounts for the widely differing scales that occur in different objective functions. For example, a drag functional is normally $\mathcal{O}\left(10^{-2}\right)$ while a functional based on mission range may be $\mathcal{O}\left(10^{5}\right)$.

The slope is evaluated at major search iterations, which is monotonically decreasing.[g] The objective slopes can be non-smooth, which can cause false triggering. This can be partially avoided by using running averages over a small window, which effectively smooths the objective history. This helps prevent premature triggering, but it causes a lag equal

[g]For attainable inverse design problems, the slope is measured in log-space to better reflect the problem.

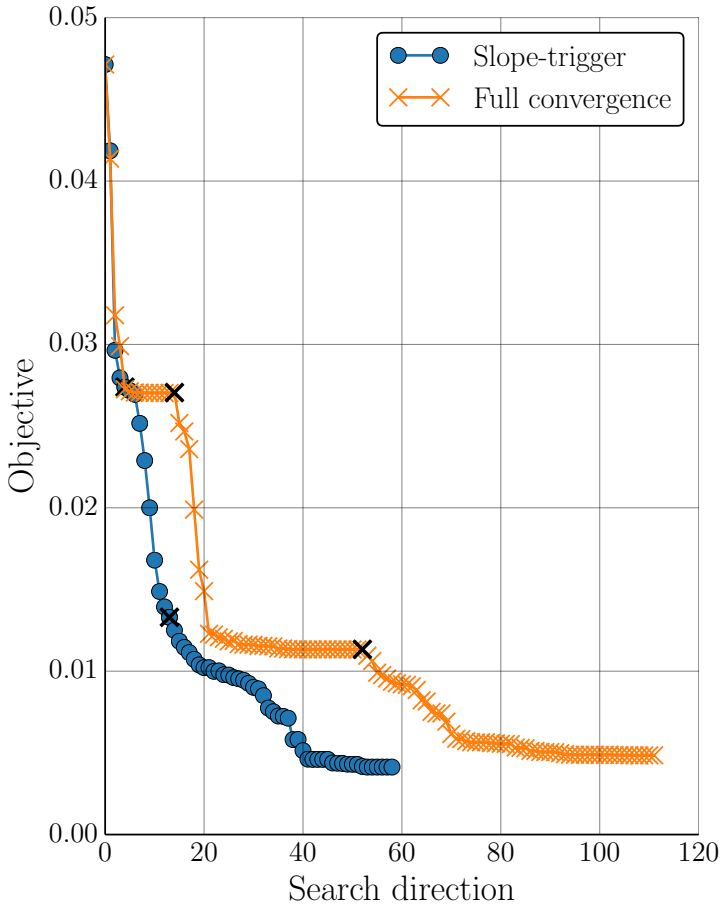


Figure 9: *Orange*: Optimizing to convergence on each level leads to slow design improvement. *Blue*: Using aggressive slope-based transitions permits much faster design improvement. ×-marks denote parameterization refinements.

to the size of the window, which delays the trigger for a few search directions. Therefore the window should be as small as possible. For relatively simple design problems, a fairly aggressive trigger can be used (as high as $r = 0.25$, with window $w = 1$). For more complex problems, especially ones with initially-violated constraints, we observe that it can be more effective to allow deeper convergence on the coarser parameterizations before proceeding.

The slope-trigger tacitly assumes that diminishing design improvement indicates a nearly fully-exploited search space. 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 an aggressive trigger 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.[h] For practical design environments we also optionally allow the designer to manually signal the framework to trigger (or delay triggering) a parameter refinement. If a signal is not sent, the automatic trigger is used.

5 Discovering the Most Effective Shape Control

At this point, we have described a system that optimizes a shape, using automatically-generated, uniformly-refined, nested search spaces. Uniform refinement is simple, robust, and consistent with the continuous optimal solution. However, uniform shape control distribution may be suboptimal for a given number of shape parameters, which can adversely impact efficiency. Minimizing the number of design variables is highly desirable.[i]

In this section, we investigate the possibility of selective refinement or adaptation. We discuss a systematic method for choosing an effective combination of refinement locations from among the myriad possibilities. In this approach, a set of candidate shape control refinements is first generated by the modeler, as described in Section §4.1. Next, the system predicts the effectiveness of each candidate by computing an "importance indicator". Finally, the system selects the shape control refinement with the highest predicted performance.

In the first section, we develop a class of importance indicators, based on "problem-aware" metrics, namely the objective and constraint gradients and Hessian information. In Section §5.2, we discuss a search algorithm for finding an effective combination of parameters. Adaptive parameterization allows setting a growth rate, which has important impacts on efficiency. We evaluate these effects in Section §5.3 and propose a possible method for automatically determining an efficient growth rate. This chapter concludes with a brief section on a few additional requirements on the geometry modeler, beyond what is required for non-adaptive (uniform) refinement.

[h] However, under a finite-difference optimization framework (i.e. without the adjoint), where the cost of each extra gradient is two flow solutions ($N + 1$ in all), it could prove more efficient to allow deeper convergence on fewer design variables.

[i] This is true even under an adjoint formulation. Although the cost of gradient computations is much less sensitive to N_{DV} than under a finite difference approach, non-negligible $\mathcal{O}(N_{DV})$ costs remain, namely the computation of geometric surface derivatives and subsequent gradient projections.

5.1 Indicators of Design Improvement Potential

The expected achievable design improvement $\Delta\mathcal{J}_{exp}$ with a given set of shape control can be estimated using the local objective and constraint gradients with respect to the candidate shape control and a Hessian approximation. As we will show in a later section, the gradients with respect to candidate design variables can be computed with minimal computational overhead, while the Hessian is less straightforward.

Consider Figure 10, which illustrates a local quadratic fit of an objective function in the candidate search space, based on the current objective value $\mathcal{J}(\mathbf{X}_0)$ (presumably the optimum achieved in the previous design space), objective gradients $\frac{\partial\mathcal{J}}{\partial\mathbf{X}_c}$, and Hessian $\frac{\partial^2\mathcal{J}}{\partial\mathbf{X}_c^2}$. Each gradient gives a local forecast of the rate at which that individual candidate parameter will help improve the design, while the second derivatives indicate how fast that rate of return will decrease.

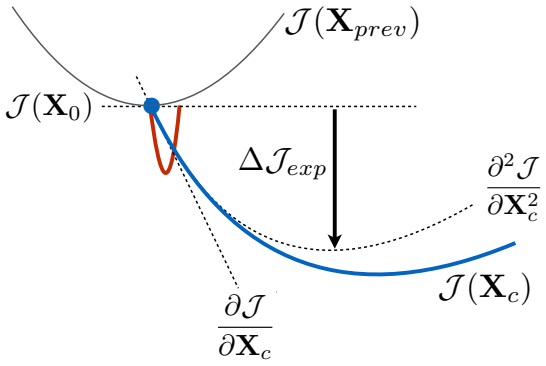


Figure 10: Local first- and second-order fits (dotted lines) of a candidate search space's actual behavior (blue). With second-derivative information, the expected improvement $\Delta\mathcal{J}_{exp}$ can be estimated. A second candidate search space with higher gradients (red) may actually offer less potential design improvement if its second derivatives are also high.

The minimizer of this fit has an analytically known location and value. Conceptually, this minimal value is an estimate of *how much design improvement is possible* under that parameterization, which serves as an intuitive importance indicator.

5.1.1 Unconstrained Case

Considering first the unconstrained case, the indicator is

$$I_H(\mathbf{C}_c) \equiv -\Delta\mathcal{J}_{exp}(\mathbf{C}_c) = \frac{1}{2}\frac{\partial\mathcal{J}}{\partial\mathbf{X}_c}^T\left(\frac{\partial^2\mathcal{J}}{\partial\mathbf{X}_c^2}\right)^{-1}\frac{\partial\mathcal{J}}{\partial\mathbf{X}_c} \tag{5}$$

which prioritizes search spaces with the highest capacity for design improvement. In the next secton, we will show that I_H performs exceptionally well, even on a highly misscaled problem. Unfortunately, for aerodynamic problems, no estimate of the Hessian for the candidate design space is readily available, without the prohibitive cost of $2N_{DV}$ finite-differenced flow and adjoint solutions. In a well-scaled problem, we might approximate the Hessian as the identity matrix, yielding an indicator that is more readily computable, as it involves only gradient information:

$$I_G(\mathbf{C}_c) = \frac{1}{2}\frac{\partial\mathcal{J}}{\partial\mathbf{X}_c}^T\frac{\partial\mathcal{J}}{\partial\mathbf{X}_c} = \frac{1}{2}\sum_{i=1}^{N_{DV}}\left(\frac{\partial\mathcal{J}}{\partial X_c^i}\right)^2 \tag{6}$$

For many aerodynamic problems, however, the relative design variable scales are orders of magnitude different from each other. As we will show in Section §6.3, this can render

I_G a grossly ineffective predictor of performance. One middle ground is to approximate the Hessian using a simple diagonal scaling matrix $\mathbf{D}$:

$$I_D(\mathbf{C}_c) = \frac{1}{2}\frac{\partial \mathcal{J}}{\partial \mathbf{X}_c}^T \mathbf{D}^{-1} \frac{\partial \mathcal{J}}{\partial \mathbf{X}_c} \tag{7}$$

Ideally, $\mathbf{D}$ is the diagonal of the Hessian, which roughly encodes the relative scaling of each parameter. (The off-diagonal terms account for redundant potential among the parameters.) Although the diagonal values could theoretically be specified as scale factors by the user, as commonly done in optimization to improve the conditioning, manual intervention is not an option in an automated setting. A more subtle problem is that, unlike in quasi-Newton optimization – which self-corrects the Hessian over several iterations, here we are trying to pick good parameters without actually taking an optimization step. An inaccurate choice of scale factors will therefore have a more serious impact on the quality of the results. In a subsequently section, we will discuss some approaches to approximating the Hessian, or at least its diagonal. Nevertheless, this remains an open area of research. For most problems, we must currently limit ourselves to using only first-order information.

5.1.2 Constrained Case

If there are design constraints or design variable bounds, it is desirable to prioritize parameterizations that have the largest expected *feasible* objective reduction. A candidate shape parameter is not useful if it must violate a constraint to improve the objective. In the specialized case of *localized* constraints (for example, wing thickness), a rough approach is to simply exclude any candidate shape control stations that are located near the active constraints.[2] However, this does not extend to important non-localized constraints, such as lift, pitching moment or wing volume.

To handle general linear and nonlinear constraints, including design variable bounds, we propose an approach based on the Karush-Kuhn-Tucker (KKT) optimality conditions. Satisfaction of the KKT conditions indicates that no further progress is possible within the current search space. Inverting this logic, we propose to add new parameters that make the KKT conditions in the *new* search space as *un*-satisfied as possible.

As before, we assume a local quadratic fit to the objective function, but now subject to the currently active (or violated) constraints $\boldsymbol{\mathcal{C}}_a$, which are treated as equality constraints and linearized about the current design:

$$\frac{\partial \boldsymbol{\mathcal{C}}_a^T}{\partial \mathbf{X}_c}\mathbf{X}_c = \mathbf{b} \tag{8}$$

The basic idea is to ignore the currently inactive constraints, and to assume that the active (and violated) constraints will be satisfied at the optimum in the candidate design space. Equation (8) makes the assumption that the active constraint set at the current design is the same as the active set at the predicted minimizer. This may not be true, but cannot be avoided, as we are making a prediction without actually optimizing.[j] The constrained minimizer of the quadratic fit is the solution to the following system of equations

[j] Determining the active set of constraints is a very challenging problem even *during* optimization.

$$\begin{bmatrix} \frac{\partial^2 \mathcal{J}}{\partial \mathbf{X}_c^2} & \frac{\partial \boldsymbol{\mathcal{C}}}{\partial \mathbf{X}_c} \\ \frac{\partial \boldsymbol{\mathcal{C}}^T}{\partial \mathbf{X}_c} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{X}_c^* \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathcal{J}}{\partial \mathbf{X}_c} \\ \mathbf{b} \end{pmatrix} \tag{9}$$

where the leftmost term is sometimes called a KKT matrix. Solution of this system can be split into two steps. First, solve for the Lagrange multipliers $\boldsymbol{\lambda}$:

$$\left(\frac{\partial \boldsymbol{\mathcal{C}}^T}{\partial \mathbf{X}_c} \left(\frac{\partial^2 \mathcal{J}}{\partial \mathbf{X}_c^2} \right)^{-1} \frac{\partial \boldsymbol{\mathcal{C}}}{\partial \mathbf{X}_c} \right) \boldsymbol{\lambda} = \frac{\partial \boldsymbol{\mathcal{C}}^T}{\partial \mathbf{X}_c} \left(\frac{\partial^2 \mathcal{J}}{\partial \mathbf{X}_c^2} \right)^{-1} \frac{\partial \mathcal{J}}{\partial \mathbf{X}_c} - \mathbf{b} \tag{10}$$

The minimizer $\mathbf{X}_c^*$ does not need to be explicitly computed. Instead, substituting its functional form into the quadratic fit yields the expected *feasible* design improvement

$$I_{KKT}(\mathbf{C}_c) \equiv -\Delta \mathcal{J}_{exp}(\mathbf{C}_c) = \frac{1}{2} \Bigg(\underbrace{\frac{\partial \mathcal{J}}{\partial \mathbf{X}_c} - \boldsymbol{\lambda} \frac{\partial \boldsymbol{\mathcal{C}}}{\partial \mathbf{X}_c}}_{\mathbf{A}} \Bigg)^T \left(\frac{\partial^2 \mathcal{J}}{\partial \mathbf{X}_c^2} \right)^{-1} \left(\frac{\partial \mathcal{J}}{\partial \mathbf{X}_c} - \boldsymbol{\lambda} \frac{\partial \boldsymbol{\mathcal{C}}}{\partial \mathbf{X}_c} \right) \tag{11}$$

Roughly speaking, I_{KKT} prioritizes parameterizations where the objective gradients are as orthogonal as possible to a linear combination of the active constraint gradients. Curvature information from the Hessian (middle term) corrects the prediction. Term $\mathbf{A}$ in Equation (11) is related to the KKT optimality metric. At the optimal design in the previous search space, $\mathbf{A} = \mathbf{0}$ (or $\mathbf{A} \approx \mathbf{0}$ if only partially converged), but after adding new parameters it will become nonzero, indicating that there is room for further feasible reduction in the objective.

In the absence of active constraints, Equation (11) is equivalent to Equation (5). Indeed, in most experiments, the two indicators appear to yield very similar rankings, but in some cases, Equation (11) may avoid adding ineffectual parameters. As before, if the Hessian is taken to be the identity matrix, the indicator simplifies to a first-order prediction

$$I_{KKT_G}(\mathbf{C}_c) = \frac{1}{2} \sum_{i=1}^{N_{DV}} \left(\frac{\partial \mathcal{J}}{\partial X_c^i} - \sum_{j=1}^{N_{\mathcal{C}}} \lambda_j \frac{\partial \mathcal{C}_j}{\partial X_c^i} \right)^2 \tag{12}$$

which in turn is equivalent to Equation (6) when there are no active constraints.

5.1.3 Efficient Computation of Gradients for Indicators

The objective and constraint gradients have modest additional cost. During optimization, the adjoint solutions ψ_j are used to efficiently compute gradients with respect to arbitrary shape design variables. After a search space refinement is triggered, we *reuse* the adjoint solutions from the final design in the previous search space to rapidly compute gradients with respect to the new candidate design variables. This is another case where neutral mutation (Eq. (4)) of the shape control is important. Reuse of the adjoints is possible only if the geometry modeler preserves the shape exactly when refining the parameterization, so that the final shape in the previous search space is identical to the initial shape for the next search space. This is inherently true for all discrete geometry modelers, because they operate by deforming a static baseline shape, but is not generally true for constructive (CAD-like) modelers. As an example, Function 2 shows how I_G is assembled. The other indicators would be computed by a similar process.

Function 2: *GradientIndicator*$(\cdot)$

Input: Surface $\mathbf{S}$, shape control $\mathbf{C}$, objective adjoint solution ψ

Result: Indicator I_G

$\mathcal{D}, \mathbf{X} \longleftarrow \mathit{Parameterize}(\mathbf{S}, \mathbf{C})$

foreach X_c *in* $\mathbf{X}_c$ **do**

$\quad \frac{\partial \mathbf{S}}{\partial X_c} \longleftarrow \mathit{ShapeDerivative}(\mathcal{D}, X_c)$

$\quad \frac{\partial \mathcal{J}}{\partial X_c} \longleftarrow \mathit{ProjectGradient}(\psi, \frac{\partial \mathbf{S}}{\partial X_c})$

end

$$I_G \longleftarrow \sum_i \left(\frac{\partial \mathcal{J}}{\partial X_c^i} \right)^2$$

5.2 Searching for the Best Combination of Candidates

We now present a search procedure for finding an effective parameterization. Recall from Equation (5) that the expected design improvement is a function of the whole ensemble of candidate shape control $\mathbf{C}_c$, not simply of each individual controller C_c. This is a critical point, and has important consequences for efficiency. In general, the expected design improvement is a nonlinear function of the candidates. In other words, to compute the effectiveness of a collection of candidates, one cannot simply sum the incremental benefits due to each candidate parameter. The reason for this is that multiple similar shape control candidates usually have "redundant potential"; adding one of them may be useful, but adding a second may not help much if it enacts similar shape modifications. Put another way, the potential design improvement offered by two candidates may be mutually exclusive.

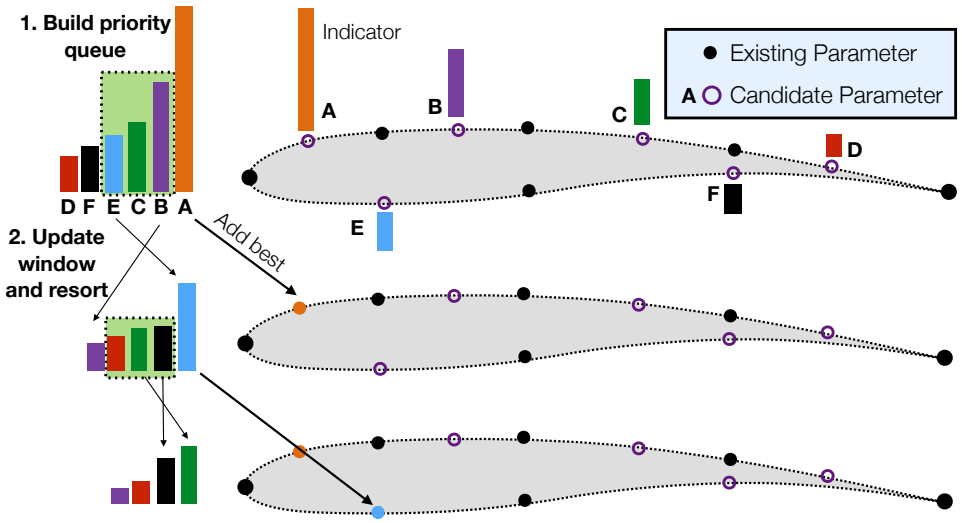


Figure 11: Constructive search algorithm for refining the shape parameterization

Function 3: *AdaptShapeControl*$(\cdot)$

Input: Surface $\mathbf{S}$, current shape control $\mathbf{C}$, candidate shape control $\mathbf{C}_c$, adjoint solutions ψ_i, growth rate g, window w

Result: Updated shape control $\mathbf{C}$

```
// Phase 1.  Build priority queue
queue ⟵ ∅
foreach C_c in C_c do
    I ⟵ ComputeIndicator(S, C ∪ C_c, ψ_i)
    queue.Add(C_c, priority = I)
end

// Phase 2.  Add shape control
N_add ⟵ int(len(C) · g)
for i=1..N_add do
    foreach C_c in queue.Best(w) do
        I ⟵ ComputeIndicator(S, C ∪ C_c, ψ_i)
        queue.Update(C_c, priority = I)
    end
    C_best ⟵ queue.pop()
    C ⟵ C ∪ C_best
end
```

Finding the best ensemble of parameters is a form of combinatorial optimization. An exhaustive search is prohibitive: choosing the best subset of A out of B candidates would require $\frac{A!}{B!(A-B)!}$ indicator evaluations. One simple "search" procedure is to randomly sample combinations of parameters. However, this is highly unlikely to find a good combination of parameters without very large numbers of samples. Although we did not explore the possibility, "metaheuristic" search procedures such as genetic optimization, could also be used. However, these typically require large numbers of functional evaluations.

For this work we developed a "constructive" search procedure, illustrated in Figure 11. In the first phase, each possible introduction of a single new parameter is analyzed. A priority queue is then formed by ranking the candidates by their indicator value, as computed by any of the methods from Section §5.1. In the second phase, the system makes N_{add} passes over the priority queue, reanalyzing only a sliding window, w, of the top few candidates remaining in the queue, resorting the queue, and adding the top-ranked parameter. The choice of the window size w is a tradeoff between the cost of evaluating more combinations and the potential benefit of finding a more effective search space. This procedure is given more explicitly in Function 3. Its important features are:

- By reanalyzing the top w candidates, redundant parameters are avoided.
- The cost for the entire search is bounded and $\mathcal{O}(N_c)$.[k]

[k] At most $w \cdot N_c(1 + \frac{w}{2})$ indicator evaluations are required: N_c to build the initial priority queue and $w \cdot \min(N_{add}, N_c - N_{add}) \leq w \cdot \frac{N_c}{2}$ to add the rest, because if we are adding more than half of the candidates,

This procedure is most effective when the initial priority queue remains a fairly accurate ranking throughout the search. We observe that for many problems this is a reasonable assumption, and the procedure often returns the same result as an exhaustive search, but at a fraction of the cost. If the initial priority queue is considered perfectly trustworthy, one can use a window size of $w = 0$, which is equivalent to immediately accepting the top N_{add} members of the queue. However, in cases with high "redundancy" among the candidate shape parameters, this procedure can yield far less optimal results. We give a detailed example of these different situations in Section §6.3.

The wall-clock time of the search procedure depends on the number of candidates being considered, the window size, and on the speed of the geometry modeler and gradient projection tools, which are invoked frequently. Typically the running times are highly practical, with cost usually equivalent to no more than a few design iterations. Naturally, there is a tradeoff between spending longer searching for a more efficient parameterization and immediately making design progress, but in a less optimal search space.

5.2.1 Simplification for Linear Deformers

For certain particular types of deformers (notably, Hicks-Henne bump functions[25] and Bernstein polynomials or Kulfan parameters[18]) each deformation mode, described by $\frac{\partial \mathbf{S}}{\partial \mathbf{X}}$, is a function of only one element of the shape control $\mathbf{C}$. In these special cases, the objective and constraint gradients can be precomputed, which greatly reduces the expense of ranking the parameters.

Unfortunately, such deformers are the exception rather than the rule. In general, the deformation mode shape of a parameter also depends on where its neighbors are located. To visualize why this is usually the case, consider interpolating deformation between consecutive control stations. By moving one station relative to its neighbor, both of their shape deformation modes are changed; the width of one shrinks, while the other expands. The presence of any form of interpolation renders this simplification invalid, ruling out almost all modelers, including spline-based approaches, CAD systems, and custom deformers like the ones used in this work.

5.3 Growth Rate

Uniform refinement involves only one tunable parameter, namely the trigger, discussed in Section §4.3. Adaptive refinement introduces one additional strategy tuning parameter: the growth rate in the number of parameters ($\mathbf{g}$ in Algorithm A). Setting the growth rate is critical for performance and involves striking a balance between flexibility and efficiency. An inflexible search space with too few design variables will quickly stagnate, requiring frequent shape control adaptation. Conversely, with too many design variables, navigation is slow. In this implementation, the designer specifies relative growth rates (e.g. 1.5×). In this work We do not yet consider the promising possibility of removing design variables from the active set.

The optimal rate of design variable introduction depends on the problem. Figure 12 compares the performance of various growth factors on a geometric shape-matching problem. On this relatively simple problem, a growth rate of 2× converges twice as fast as a

we can work backwards from uniform refinement, *removing* the least useful parameters one at a time.

growth rate of 1.25×. Here, the purely geometric objective functional allows rapid and reliable design improvement regardless of the number of design variables, thus favoring fast growth rates. In more complex problems, however, slower growth rates are often faster overall. For example, in a sonic boom pressure signature-matching verification example, we found that an intermediate growth rate of 1.5× solidly outperforms both a slower growth rate and uniform (2×) refinement.

An additional consideration is how many shape parameters to start with. As the transonic airfoil example in Section §7.1 will demonstrate, starting with a truly "minimal" search space (with, say, one or two design variables) leads to rather stunted growth early on. We have observed that it is consistently more efficient to start with at least 6-10 variables, naturally with the precise number depending on the problem.

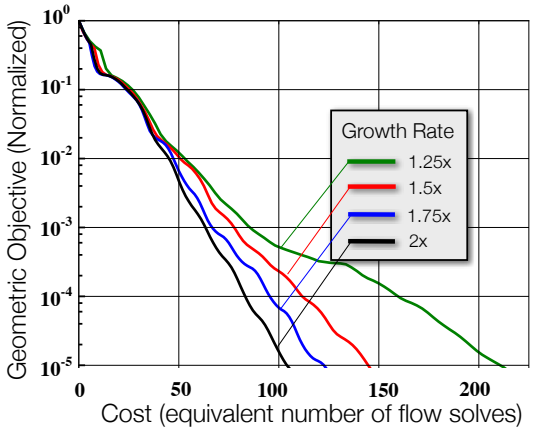


Figure 12: Effect of parameter growth rate on a geometric shape-matching objective. Each curve shows mean behavior of 10 randomized trials (trigger = 0.25).

6 Verification and Evaluation

This section evaluates the potential computational acceleration of progressive shape control and validates the adaptive approach. Three academic optimization examples are considered. The first example is a symmetric transonic airfoil design problem. On this problem we demonstrate that substantial computational acceleration is possible using even a simple uniform refinement (non-adaptive) approach.

The second verification set (Section §6.3) evaluates the adaptive shape control system. The two examples in this section demonstrate correct discovery of the parameters necessary to solve the optimization problem. In the process, we also evaluate the different indicators developed in Section §5.1 and assess the performance of the search procedure developed in Section §5.2. For both of these problems, the correct answer is known *a priori*, but the system is initially given shape control that is insufficient to solve the problem.

6.1 Symmetric Transonic Airfoil Optimization

The purpose of this first example is to demonstrate that substantial design acceleration can be achieved using progressive shape control. The test case involves 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 has been investigated by several other researchers.[26–31] The starting airfoil is a modified NACA 0012 (henceforth "N0012m"), where the trailing edge is made sharp.[1] The design Mach number is 0.85, while the angle of attack is fixed at $\alpha = 0°$. Additionally, the final airfoil shape must contain the original airfoil. This constraint is satisfied when $y \geq y_{N0012m}$ everywhere on the upper surface, and inversely on the lower surface.

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

6.1.1 Meshing and Flow Solution

Because the solution must be symmetric, the flow is solved only in the upper half of the domain with a symmetry boundary condition at $y = 0$. The farfield boundaries are placed 96 chords away in each coordinate direction. The optimization process radically increased the sensitivity of the flow to the farfield boundary distance. The initial N0012m, with its relatively confined regions of supersonic flow, is quite lenient with respect to the farfield boundary location.[m] An initial domain size study indicated that a farfield distance of 24 chords was sufficient to resolve drag to within 2 counts of the value obtained using 96-chord distances. However, the final design's carefully tuned shock structure (see Figure 14a) could not be reliably resolved with farfields nearer than about 96 chords.

6.1.2 Shape Parameterization

The airfoil is parameterized using the RBF-based direct manipulation technique described in Section §2.1. Initially a single pilot point is placed on the top surface, as shown in Figure 13 (black dot). Recall that each parameter enacts a roughly bump-shaped deformation centered on the pilot point. Having observed that it is generally 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.[n] 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.

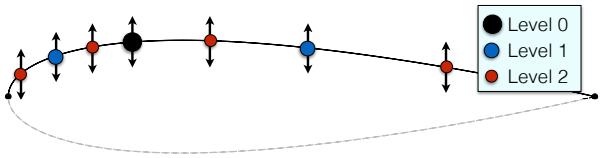


Figure 13: ***Symmetric Airfoil:*** Initial parameterization with 7 design variables, generated by twice uniformly refining a 1-DV parameterization (lower half generated by symmetry)

To handle the containment constraint, we set the lower bound of each shape parameter to the corresponding local thickness of the N0012m. The direct manipulation approach guarantees that the airfoil will exactly interpolate these pilot points. Regions between the shape control parameters may temporarily violate the containment constraint, but these violations get squeezed out as more parameters are added. Consistent with the progressive shape control paradigm, the containment constraint similarly becomes more precise as the search space is refined.

6.1.3 Optimization Results

Figure 14a shows the final optimized airfoil and its pressure profile. Notably, the leading edge has become extremely blunt. In fact, after every parameter refinement, the nose became blunter – apparently limited only by the first shape parameter's proximity to the leading edge. In fact, this is the expected optimal result for this problem.[27] The final

[m] The farfield boundary state is enforced weakly via 1-D Riemann invariants without circulation correction.

[n] Transformation function is $s^* = s - 0.15sin(2\pi s)$.

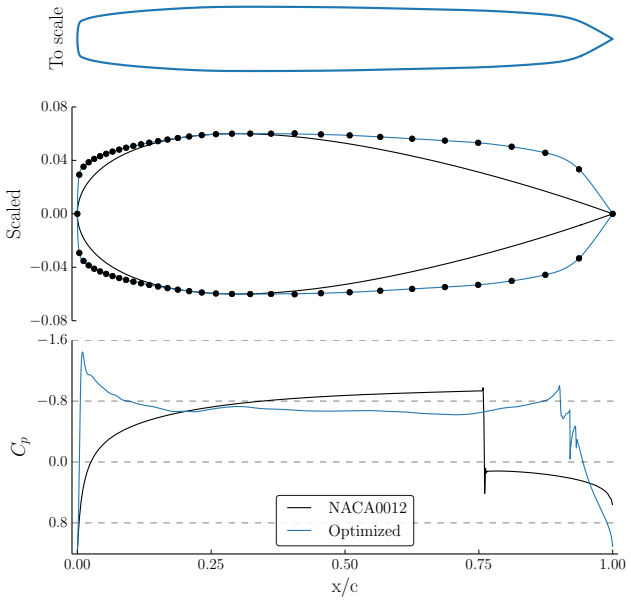
Table 1: ***Symmetric Airfoil:*** Drag reduction with optimization. Drag given in counts ($C_D \cdot 10^4$)

	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

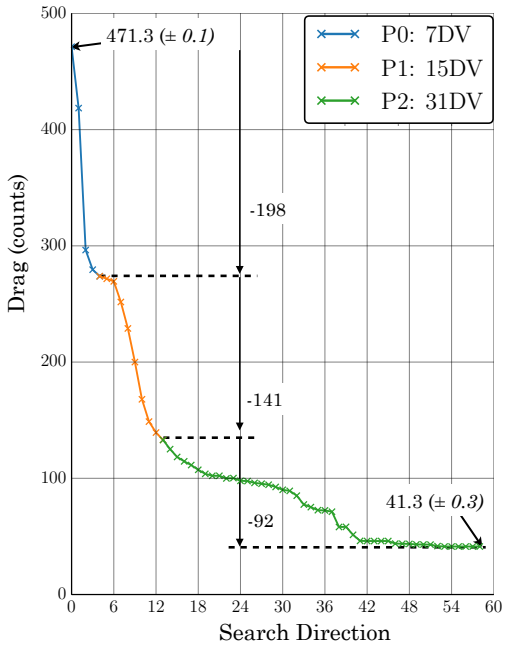
design also satisfies the containment constraint over the entire surface of the airfoil (not just at the interpolation points).

Figure 14b shows the convergence of the objective function over 60 search directions, and over 3 parameterization levels. The parameterization was automatically refined (i.e. with no user intervention) when the objective slope tapered to 20% of its maximum slope. After each transition, a new optimization is started; no transfer of Hessian information from the previous design space is attempted. The final parameterization has 31 design variables. The drag was reduced by a factor of 10, from the baseline 471 counts down to 41.3 counts. An additional refinement to 63 DVs proved unable to further improve the design. The final design is probably close to optimal, as demonstrated by the diminishing return on each additional parameter refinement visible in Figure 14b. Some further improvement is likely possible, but even the small amount of remaining discretization error combined with the very high-dimensional design space makes further improvement extremely difficult.

Figure 15 compares the initial and final meshes, which were automatically adapted to



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



(b) Objective convergence

Figure 14: ***Symmetric Airfoil:*** Progressive parameterization optimization results

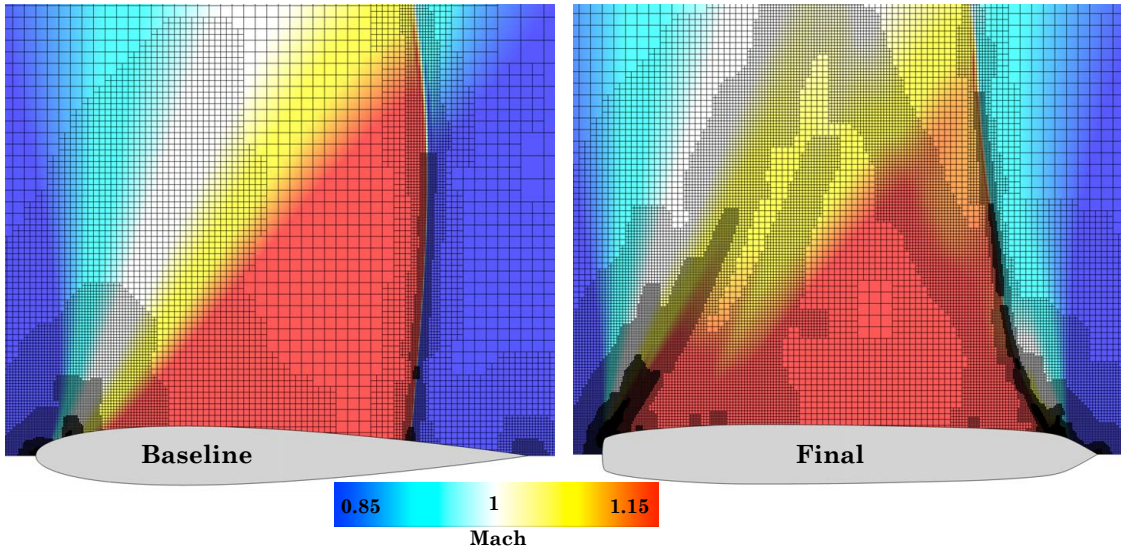


Figure 15: ***Symmetric Airfoil:*** Comparison of baseline and final meshes. The mesh refines the regions most important for computing drag, primarily focusing on the leading edge expansion and shock. To achieve the same error tolerance for both designs, the baseline mesh required only 26K cells (upper half only), while the final design required 61K cells.

reduce error in drag. Intermediate designs generated radically different mesh refinement patterns. The refinement patterns reflect movement of the shock and changes in the width of the supersonic region. For the final design, the adjoint-based mesh adaptation process provided an estimate of the remaining error in drag of about 0.3 counts ($< 3 \cdot 10^{-5}$ in C_D). The output-based mesh adaptation performed a mesh refinement study at each design iteration, yielding tight error bounds that were roughly constant throughout the optimization (see Table 1), giving high credibility to the final design. The cell count required to meet the error tolerance gradually increased throughout optimization. This indicates that the optimization drove the design to become more sensitive to the mesh discretization. This is not surprising since the final design has a much larger zone of dependence and weaker shocks which make the effects of numerical dissipation more noticeable.

6.1.4 Static vs. Progressive Search Spaces

It is worth pausing to evaluate computational performance. The main observation is that the progressive parameterization approach strongly outperforms any fixed search space. To give a rough sense of performance, Figure 16 plots design improvement versus wall-clock time for solving this case with various parameterizations on four cores of a laptop[o]. Both the uniform refinement scheme (labeled "progressive") and the adaptive approach (which resulted in fewer design variables) achieved faster and deeper overall design improvement than any static parameterization, regardless of its resolution. As expected, low-dimensional search spaces support limited design improvement, while high-dimensional spaces take much longer to navigate. On the finest (63-DV) static parameterization, which stalled quite early, the optimizer may simply be unable to navigate the design space,

[o]2013 MacBook Pro with a 2.6GHz Intel Core i7 and 16GB of memory.

as also reported by Carrier *et al.* on this problem.[27] Starting in a coarse design space appears to smooth the navigation early on, leading to a more robust search process, an observation we examine in more detail in a later example (Section §7.1).

6.1.5 Uniform vs. Adaptive Refinement

The adaptive refinement approach (which results in fewer design variables) performed slightly faster 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 §4.3), rate of variable introduction (Section §5.3), choice of importance indicator (Section §5.1), design variable scaling, and the fundamental path-dependence of optimization each have a major impact on the efficiency. It is difficult to draw firm conclusions about the potential computational advantage of adaptive refinement vs. uniform refinement from such a cursory study. In the following examples, we investigate the adaptive approach in more detail.

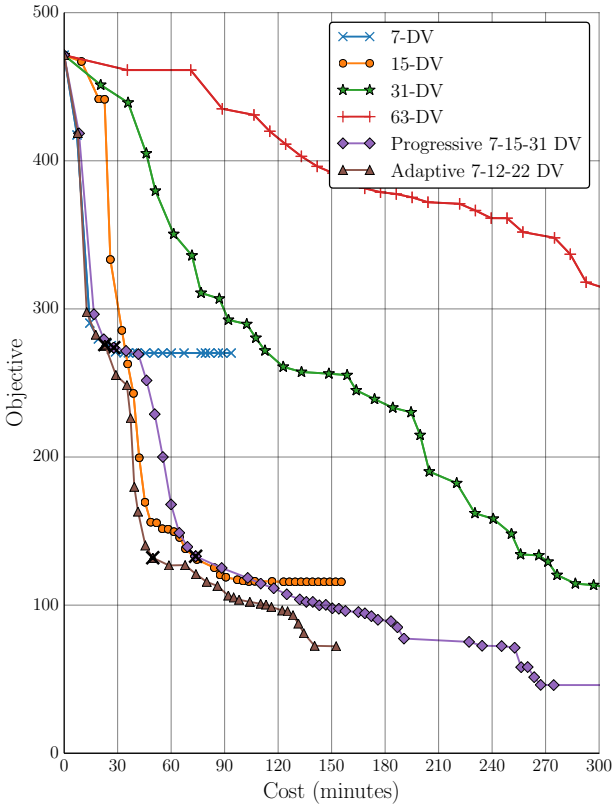


Figure 16: ***Symmetric Airfoil:*** Cost-effectiveness of each parameterization scheme, showing design improvement vs. wall-clock time. ×-marks indicate search space refinements on the progressive and adaptive methods. All cases used identical settings for meshing, solving, error control and (where relevant) triggering.

6.2 Subsonic Wing Twist Optimization

Next we consider a wing twist optimization problem, where the airfoil section and planform remain unmodified. The objective is to minimize (induced) drag at fixed lift ($C_L = 0.375$) at a flight Mach number of 0.5.

6.2.1 Shape 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. For this problem we use a deformer that interpolates twist between arbitrary spanwise stations. The twist is in the streamwise plane about the trailing edge and is linearly interpolated between successive stations. Control stations can be arbitrarily spaced along the span, but

for this problem we maintain strict regularity by refining only at the midpoints between consecutive stations. We allow the global angle of attack to vary and therefore hold the twist fixed at the wing root. 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.

6.2.2 Mesh and Error Control

The baseline design has about 76.7 counts of drag. Unlike the previous example, where the objective was reduced by a factor of ten, here the possible improvements are very small, which places high demands on the accuracy of the flow solution.[32] 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 minimum possible drag is roughly

$$C_{D_{min}} = \frac{C_L^2}{\pi e_0 \mathcal{AR}} = \frac{0.375^2}{6.0\pi} = 74.6 \text{ counts} \tag{13}$$

However, as the wing is untapered, and twist is about the trailing edge, we do not expect that the optimal design will recover a precisely elliptical lift distribution. Additionally, careful inspection of the flow reveals a small shock on the wing tip near the trailing edge, where the flow accelerates around the tip to the top surface. Losses associated with this shock further erode the potential for drag reduction.

We compute adjoint solutions for the drag and lift functionals to compute their gradients, allowing the nonlinear lift constraint to be treated exactly by SNOPT. The error control scheduling was set to coincide with the parameterization refinements, and the farfield boundaries were placed at 48 chords away. In the first search space, the resulting adapted meshes contained about 5 million cells, while for the second search space, the meshes contained roughly 10-15 million cells to meet the tighter error tolerance.

6.2.3 Optimization Results

Figure 17 shows the main results of the optimization. The lift distribution rapidly approaches an elliptical shape, with only very small discrepancies at the tip, due to the untapered section, and at the root, which compensates to exactly match lift. Figure 18

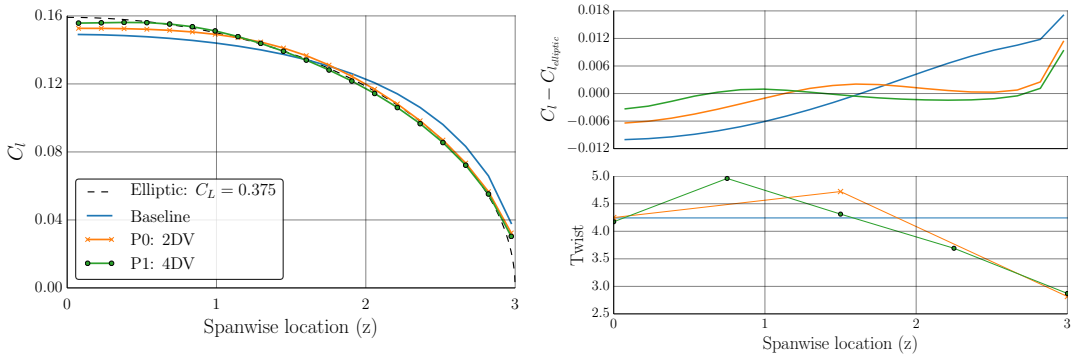


Figure 17: Twist optimization results. *Left*: Sectional lift distribution profiles. *Top right*: Deviation from elliptic distribution. *Bottom right*: Twist distribution

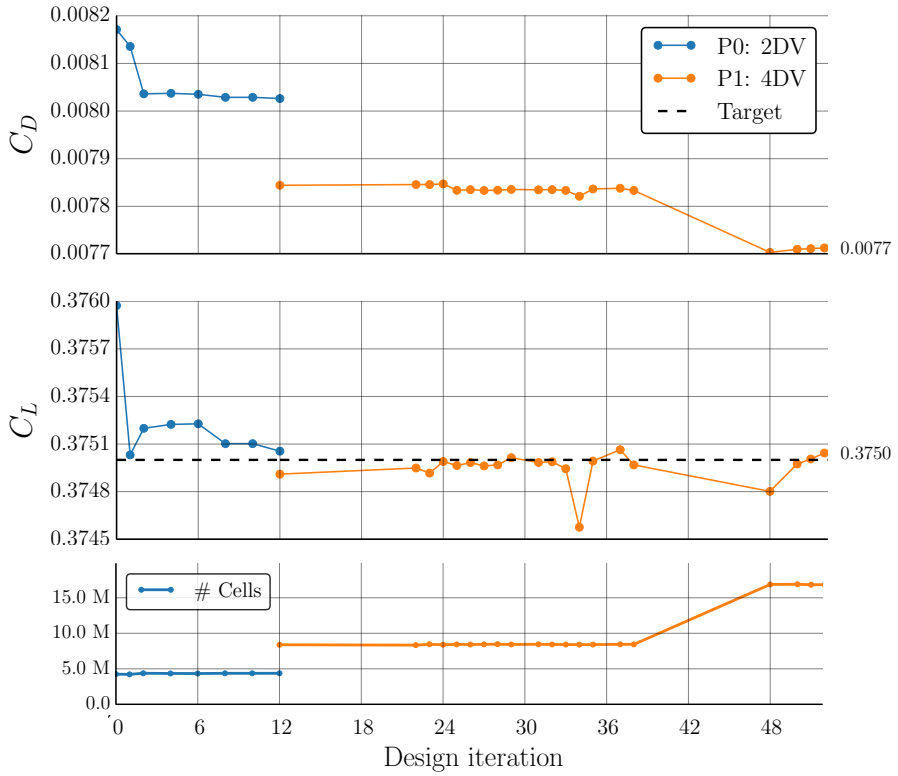
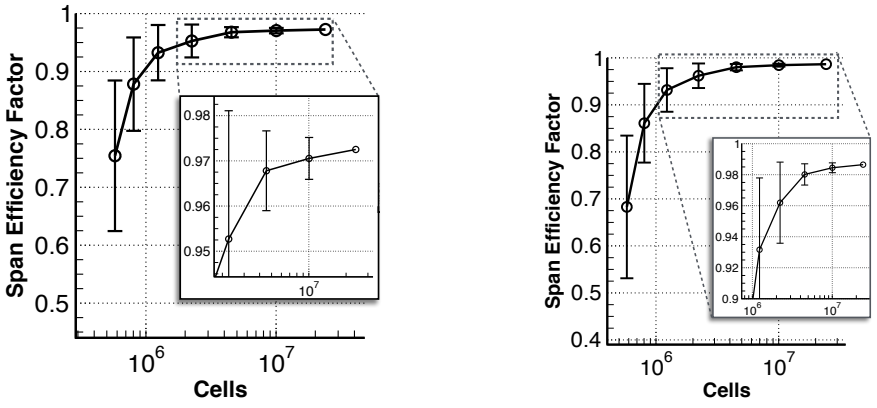


Figure 18: Twist optimization: *Top*: Convergence of drag. *Middle*: Convergence of lift. *Bottom*: Cell count history

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. By the end, lift is satisfied and drag is reduced. To accurately determine the total improvement, we performed mesh refinement studies on the initial and final designs. Figure 19 shows the convergence of span efficiency factor (e) with mesh refinement for the initial and final designs, trimmed to $C_L = 0.3750$. The initial design had $C_D = 76.7$ counts of drag ($e = 0.973 \pm 0.005$). By the final design this was improved to $C_D = 75.6$ counts ($e = 0.987 \pm 0.003$).



(a) Baseline design **(b)** Final design

Figure 19: Twist optimization: Convergence of span efficiency factor with mesh refinement

Table 2: Twist optimization results.

Chords from Root	0.0	0.6	1.2	1.8	2.4	2.7	2.85	2.97	3.0
Twist (^{o})	4.2	4.8	4.5	4.1	3.5	3.2	3.0	2.9	2.9
Sectional Lift ($2c_l/b$)	0.156	0.156	0.146	0.126	0.094	0.069	0.050	0.030	0.0

6.3 Shape Matching

This verification problem involves geometric shape-matching of a typical transport wing. In shape matching, we examine the convergence from a baseline geometry to an attainable target shape. The objective function aims to minimize the geometric deviation between the current shape and the target shape $\mathbf{S}^*$ in a least-squares sense:

$$\mathcal{J} \equiv \|\mathbf{S} - \mathbf{S}^*\|^2 = \sum_{i=1}^{N_{verts}} \|\mathbf{v}_i - \mathbf{v}_i^*\|^2 \tag{14}$$

where $\mathbf{v}_i$ are the current vertex coordinates on the discrete surface and $\mathbf{v}_i^*$ are the corresponding target vertex coordinates. This is a problem with a known solution in two senses. We know not only the optimal shape, but also the *minimal* shape parameterization that can achieve that design. The overall goal of is to efficiently discover a parameterization that enables the optimizer to exactly match the target shape.

The purpose of this verification example is to test the specifically adaptive shape control system developed in Section §5. The validation is split into four subsections, each addressed to answering a specific question:

1. (Section §6.3.2) *Are the indicator values correlated with actual design improvement?*
2. (Section §6.3.3) *Can the system discover the parameters necessary for solution?*
3. (Section §6.3.4) *How can the efficiency of the search strategy be improved?*
4. (Section §6.3.5) *Are there any situations where the strategy might fail?*

6.3.1 Initial Parameterization and Target

Figure 20 shows the the baseline and target shapes. The baseline is a straight wing with no twist, taper or sweep, represented as a discrete geometry with about 197,000 vertices. The target geometry is a wing with the same airfoil section, but substantial twist, chord-length and sweep profiles, as shown in Figure 20. 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.

The wing planform deformation is parameterized using the technique described in Section §2.2, which linearly interpolates twist, sweep and chord between spanwise stations, while exactly preserving airfoil cross-sections. The initial parameterization has three design variables: twist, chord and sweep at the tip station (marked "L0"), while the root is fixed. To refine the shape control, more spanwise stations are added (e.g. "L1", "L2", etc.), opening up new degrees of freedom. Control over twist, sweep and chord can happen at different stations, allowing for anisotropically refined shape control.

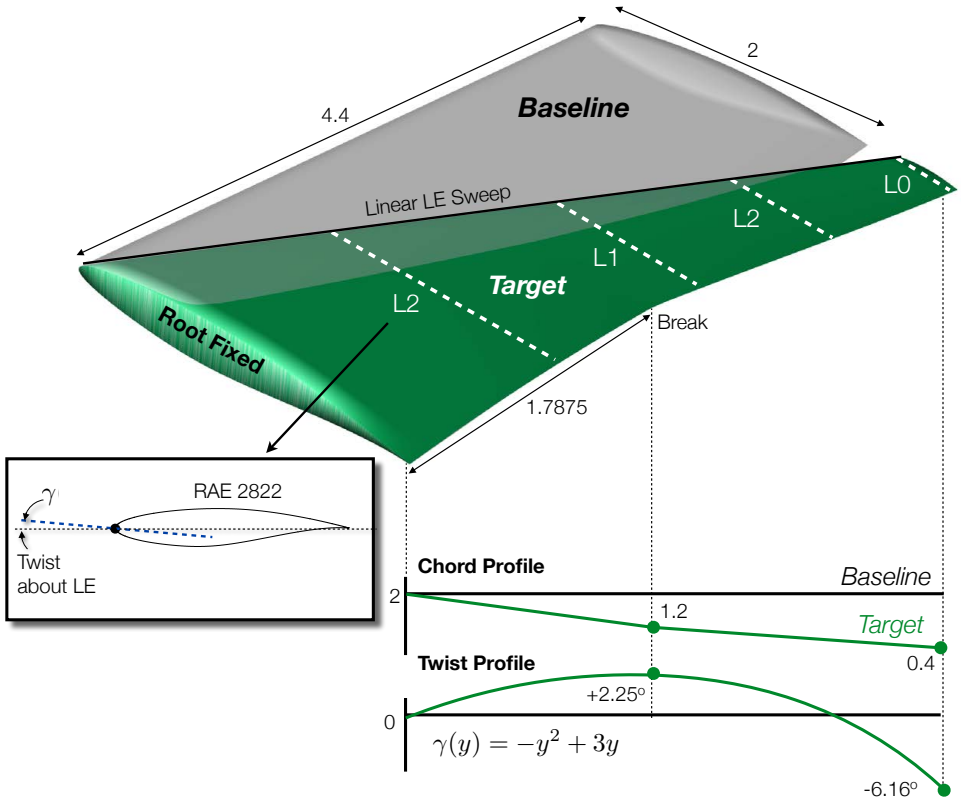


Figure 20: ***Shape Matching:*** Baseline and target planform profiles. Initial shape control station is labeled L0, after which L1 is added, followed by the L2 stations, etc.

The target shape is unattainable under this initial parameterization. Only through sufficient and correct search space refinement can the target be reached. The problem is constructed such that we know in advance the necessary and sufficient refinement pattern, i.e. the one that will allow the closest recovery of the target with the fewest design variables. Namely, chord control at the break is required to recover the piecewise linear chord profile. Next, progessively finer twist control should be added to approximate the quadratic twist profile with piecewise linear segments. The intial sweep controller at the tip is sufficient to recover the linear sweep distribution, so no additional sweep control should be added. We now test the degree to which the adaptive system can recover or approximate this "ideal" parameterization.

6.3.2 Indicator Verification

We first compare the performance of these three indicators. We start with a baseline 3-DV shape parameterization, under which the shape has been optimized to convergence, as shown by the blue curve in Figure 21. All design improvement possible under the initial parameterization has been attained, but further improvement is possible when more degrees of freedom are added. The goal of this example is to evaluate the ability of the indicators (Equations (5) to (7)) to predict the actual performance of the various candidate

shape parameters.

For this evaluation, the geometry modeler generated 21 candidate shape control refinements from the baseline parameterization. For each candidate, We ran a separate optimization, where we added only that one shape parameter to the active set. (In practice, the system will add several parameters at once; adding one at a time simplifies this verification study.) Figure 21 shows the objective convergence corresponding to each candidate parameter. The goal in adaptive refinement is to pick the parameters that enable the objective to converge to the lowest final value, thus maximizing $\Delta\mathcal{J}_{actual}$. Next, for each candidate the system predicts the potential design improvement using Equations (5) to (7), which we then correlate with the actual design improvements. We repeated this study once more, by invoking a second refinement and evaluating 44 more candidate shape parameters.

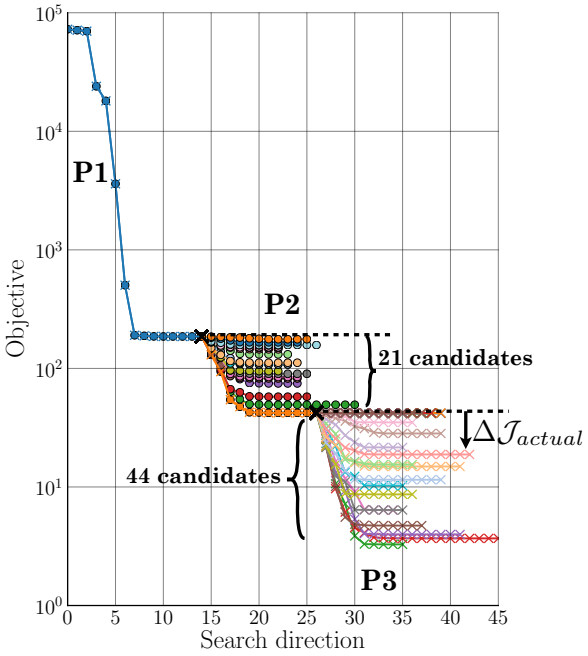


Figure 21: ***Shape Matching:*** *Indicator verification:* **P1:** Objective convergence under initial parameterization. **P2** and **P3:** Subsequent optimizations corresponding to addition of one of the candidates, starting from the previous best design.

Figure 22 shows the correlation between the predicted design improvement and the actual observed design improvement for each of the candidates. The left side corresponds to the 21 candidates tested in the first refinement, while the right side is for the subsequent 44 candidates considered in the second refinement. In each plot, the parameters at the top right are the most effective ones. The automated system would choose to add only the N rightmost parameters, while ignoring the ineffective parameters at the left. Exactly how many parameters to add will depend on the refinement strategy. The top frames in Figure 22, corresponding to a full Hessian approximation (Equation (5)), demonstrate nearly perfect performance predictions for every candidate. The middle frames show that if the Hessian is assumed to be the identity matrix (Equation (6)), the correlation is extremely poor, especially during the second refinement, where the highest-ranked parameters in fact perform the worst, and vice versa. As we discuss in Section §6.3, this happens because of poor scaling among the design parameters, which the Hessian naturally accounts for.

Previous studies have suggested using adjoint-derived gradient information to determine the relative importance of different candidate parameters.[2,4] This study demonstrates that unless the problem is well-scaled, examining only first-order information can lead to very poor predictions. As a possible remedy, consider the bottom frames, where only the diagonal of the Hessian is used. In this case the correlation is quite reasonable. Although the diagonal of the Hessian is not readily available for aerodynamic problems, this nevertheless shows that some form of simple diagonal scaling may be sufficient to achieve good predictions of importance for adaptive shape control refinement.

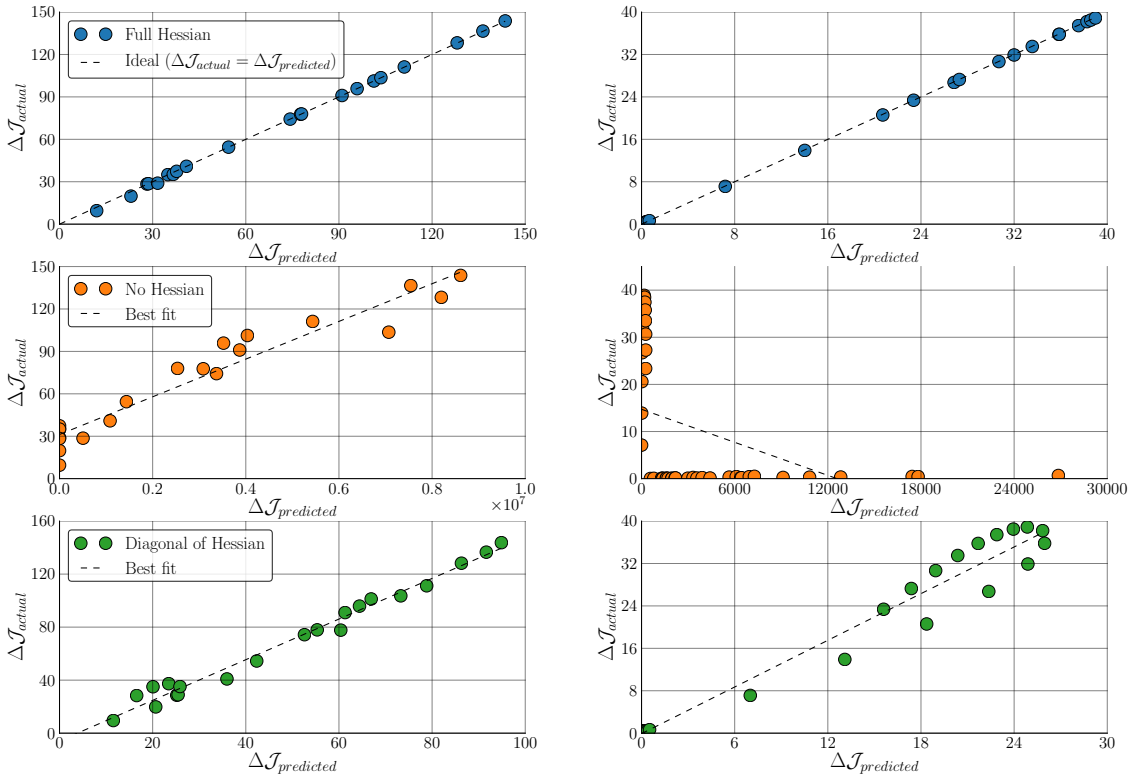


Figure 22: ***Shape Matching:*** *Indicator verification:* Correlation between predicted and actual design improvement for the first (*left*) and second (*right*) refinements. *Top row*: Full Hessian approximation. *Middle row*: Assuming the Hessian is the identity matrix. *Bottom row*: Using only the diagonal of the Hessian.

6.3.3 Test 1: Indicator Comparison and Automatic Parameter Discovery

The first goal is to investigate the indicator's ability to accurately guide the search space construction and to discover the necessary parameters. In this exercise we sequentially add one new design parameter at a time, followed by a brief optimization. we then compare the predictive power of the two effectiveness indicators, one based on gradients, I_G (Equation (6)) and one using full Hessian information, I_H (Equation (5)), which is accurately computable for this analytic objective.[P]

Figure 23 shows the resulting adaptation patterns that evolved. The right frame shows the pattern produced by the Hessian indicator after 22 adaptation cycles. Sweep control is correctly ignored. Chord control was correctly added at the break ($\frac{13}{32}$ span). Four extra chord variables were added, but this was not a mistake. Under the binary refinement rules we used, the necessary station at $\frac{13}{32}$ span was not considered a candidate until the stations at $\frac{1}{2}$, $\frac{1}{4}$, $\frac{3}{8}$, and $\frac{7}{16}$ span were all first added. The adaptation procedure did precisely this, and correctly identified the necessary parameter once the adaptation was deep enough. Examining the twist profile, the system correctly added evenly spaced stations along the span, optimally clamping down the error between the quadratic profile and the linear

[P]The Hessian is accurate but not exact, because the twist deformation modes are nonlinear with respect to the angle. The error due to this effect is small, but it explains some slightly imperfect predictions.

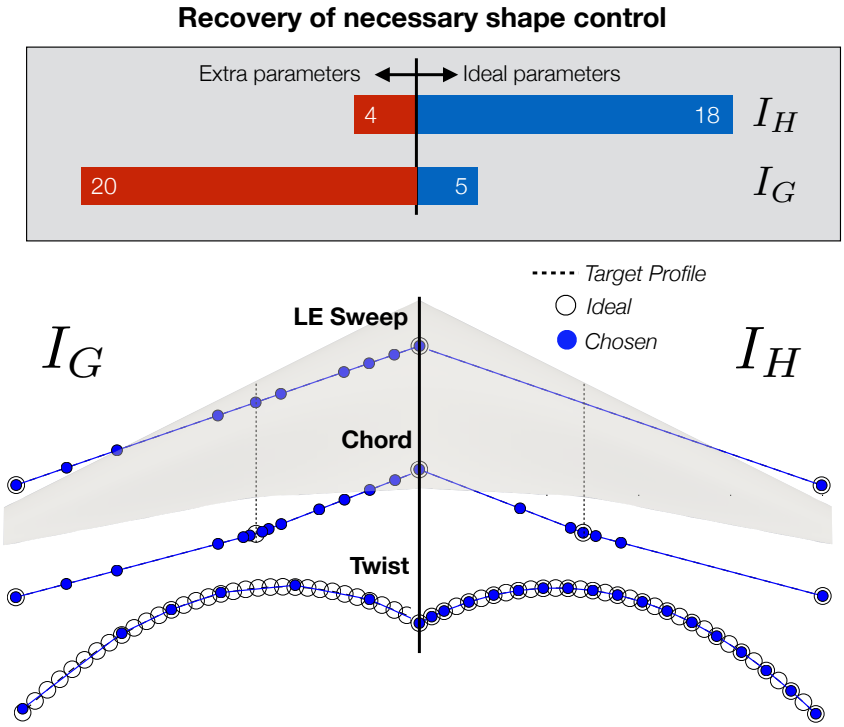


Figure 23: ***Shape Matching:*** *Test 1:* Performance of the gradient indicator I_G vs. the Hessian indicator I_H. *Top*: I_H recovers the expected parameters with few extras, while I_G mostly adds extraneous parameters. *Bottom:* Refinement patterns and optimized planform distributions with I_G (*left*) and I_H (*right*).

segments. It has also begun to add the next nested level of control near the root.

Now compare the left half of Figure 23, which shows the results using the gradient-only indicator I_G after 25 adaptation cycles. Qualitatively, the shape matching is reasonable, but the refinement pattern reveals that the procedure was quite inaccurate and failed to efficiently capture the important design variables, resulting in a somewhat inferior match, especially in the twist profile.

The reason for the relatively poor performance of I_G is that the chord and sweep objective gradients had much larger magnitudes than the twist gradients, even when very close to their optimal values. Thus chord and sweep were favored, even though they offered only extremely short-term potential. This is a concrete example of the idea illustrated notionally by the red and blue curves in Figure 10. I_H, by constrast, was intrinsically sensitive to the high *second* derivative of the objective with respect to the chord and sweep parameters, revealing that they in fact had low long-term potential. While computing I_H for aerodynamic functionals is not currently feasible, this study highlights the essential role of second derivative scaling information when predicting relative performance.

Figure 24 compares the objective convergence of the two indicators (labeled "I_G (add 1)" and "I_H (add 1)"). The gradient indicator frequently adds parameters with limited potential, leading it to stall for several adaptation cycles. Nevertheless, it still reduced the objective by over 6 orders of magnitude, indicating quite close recovery of the target shape. The Hessian indicator, however, achieves good progress at every cycle and reaches

a superior design.

Compared to the "ideal" parameterization, shown in Figure 24, performance is still relatively slow. It took many adaptation cycles to drive towards the target shape, because we deliberately permitted only one parameter to be added at a time and because the system searched only one level deep in the parameter tree. As mentioned in Section §5.3, higher growth rates should lead to much faster design improvement.

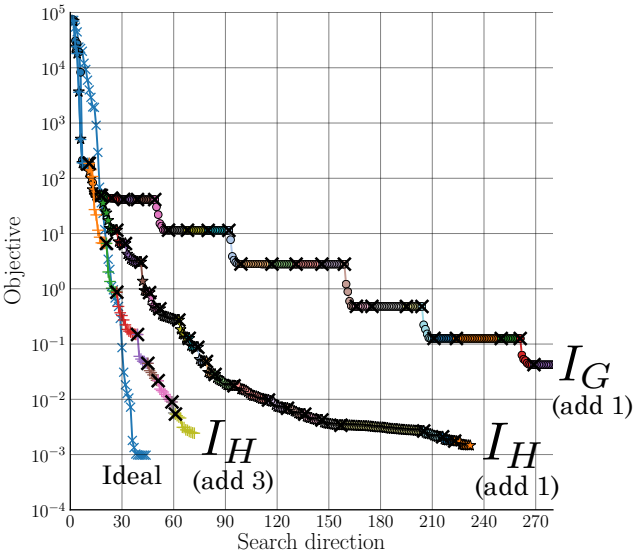


Figure 24: ***Shape Matching:*** Objective convergence for different indicators and search strategies. Solid blue line shows the "best possible" convergence, using the *a priori* known best possible 35-DV parameterization. Each color represents a different adaptively-refined parameterization and ×-marks denote search space refinements.

6.3.4 Test 2: Search Procedure Evaluation

As a second test, we then tried searching deeper for candidates (two levels deep), and specify a faster growth rate (adding three parameters per refinement). For this test, we no longer exhaustively evaluated all combinations, as this becomes prohibitive. Instead, we used the constructive search procedure (Function 3) to seek a good, if not perfect, ensemble of shape parameters, by evaluating a small number of candidates. After several alternating optimizations and refinements, the process converged to nearly perfect matching, as shown in Figure 25. Figure 24 shows that the convergence rate for this strategy (labeled "I_H (add 3)") is much faster, approaching the performance of the ideal parameterization. Although there are now a few more unnecessary parameters than before, the shape recovery is excellent. To achieve this close of a match using *uniform* refinement of the shape control would have required 48 shape parameters. By using adaptive shape control, despite adding some extraneous parameters, the system has accurately matched the shape using only 30 parameters.

6.3.5 Test 3: Immediate Discovery

As a final experiment, the system attempts to predict *all* of the necessary shape control, based only on information at the baseline design. In other words, there will be no intermediate optimizations. We allow the system to immediately look five levels deep, and request the addition of 32 design variables at once, without any prior optimization. There are a total of 93 candidates, equivalent to all the parameters that would be added under uniform refinement. An exhaustive search would involve evaluating all $\sim 8 \cdot 10^{24}$ possible combinations of the parameters, making an efficient search procedure mandatory.

Figure 26 shows the initial priority queue for the 96 candidates, which is formed by analyzing each candidate shape control element independently, using the Hessian indicator.

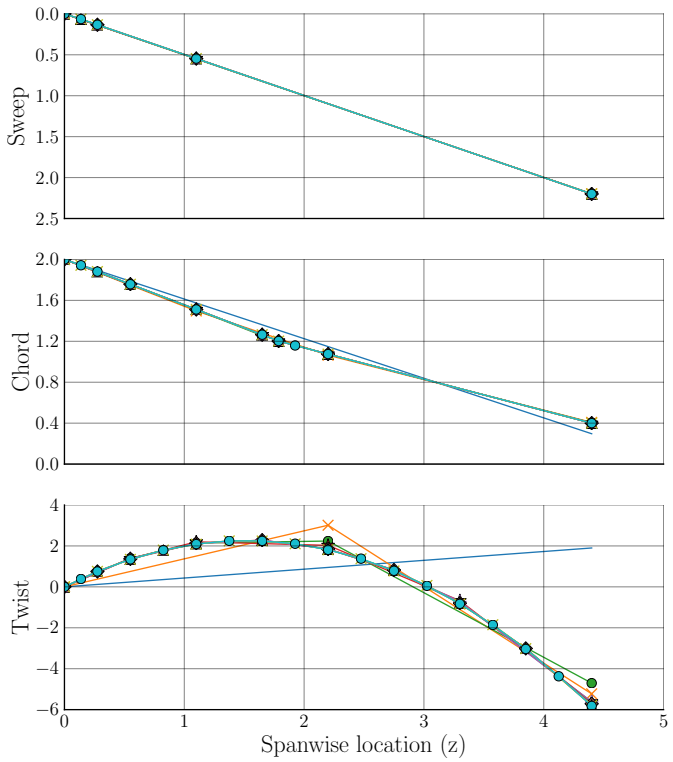


Figure 25: ***Shape Matching:*** *Test 2:* Final recovered planform distribution (search depth 2, add 3 parameters at a time, Hessian indicator).

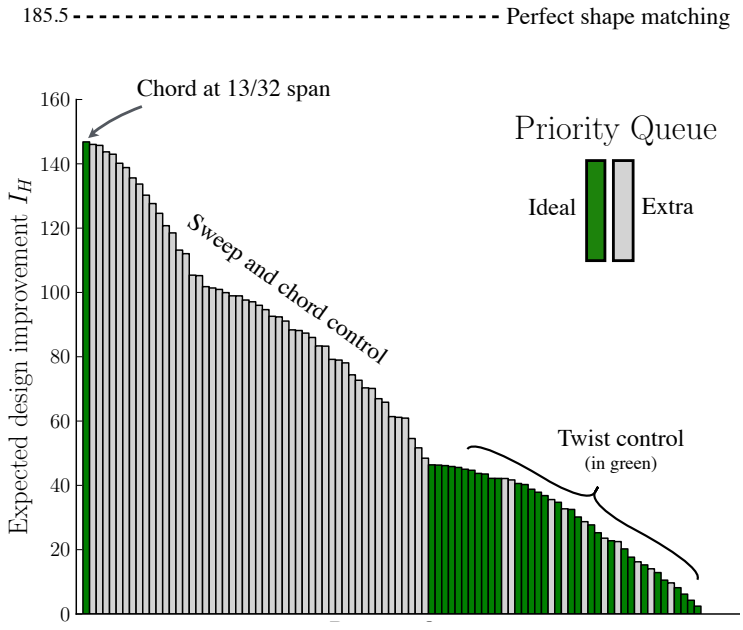


Figure 26: ***Shape Matching:*** *Test 3:* The priority queue after Phase I of the constructive algorithm (search depth of 5, adding 32 parameters at once, I_H). The 32 parameters that would best recover the target shape (*green*) are highlighted. After adding the top member of the queue, all of the subsequent chord and sweep controllers (*gray*) become redundant.

The y-axis gives the expected improvement, relative to the baseline parameterization, that can be achieved by adding the corresponding parameter. The ideal shape control ensemble of 32 parameters is highlighted in green. The first pass over the candidates correctly identified the chord station at the break as the most important shape controller to add. Initially, it appears that the twist variables are the least important in the queue. With the addition of the chord parameter, however, the next 50 elements in the queue all become highly ineffective. Their initial appraisal was based on the absence of the added 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 search procedure remains functional on this problem, but it adds many extraneous variables. Function 3 must work its way through all of the other chord and sweep variables, one window w at a time, until finally discovering the more important twist control. Studies are underway to determine whether a modification of the constructive approach can perform well on this relatively rare type of problem, or whether alternate strategies, such as a form of genetic algorithm, or specialized random search would perform better. From a practical standpoint, however, the easiest approach is to limit the depth of the search to 1-2 levels deeper than the current parameterization, which eliminates most of the redundancy and yields excellent performance.

7 Design Examples

This section demonstrates the usefulness of adaptive parameterization on two aerodynamic design examples. The first example involves multipoint design of a transonic airfoil. The second example involves design of a transonic transport wing.

7.1 Multipoint Transonic Airfoil Design with Constraints

In this example, we consider multipoint transonic airfoil design subject to numerous constraints on both geometry and aerodynamic performance. The purpose is to demonstrate adaptive shape control on a more challenging 2D problem. This example shows that using variable shape control leads to a smoother design trajectory and accelerates the optimization.

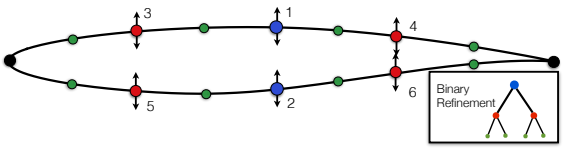


Figure 27: ***Transonic Airfoil:*** Baseline geometry, showing first three levels of uniformly refined shape control (2-DV, 6-DV and 14-DV)

The baseline geometry is a unit-chord RAE 2822 airfoil, shown in Figure 27. The objective is to minimize an equally-weighted sum of drag at two flight conditions, Mach 0.79 and 0.82. Lift-matching and minimum pitching moment constraints are imposed at both design points. Because the solver is inviscid, we constrained the camber line angle γ at the trailing edge (see Figure 28) to prevent excessive cambering that would result in poor viscous performance. A minimum and maximum geometric closing angle ϕ at the trailing edge were also specified. Finally we required that the thickness be preserved at least 90% of its initial value everywhere (enforced at 20 chordwise locations t_i), and

that the total cross-sectional area A maintain its initial value. The complete optimization statement is

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

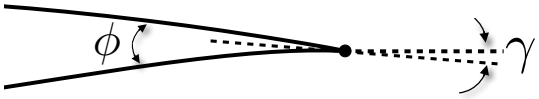


Figure 28: Geometric constraints at the trailing edge

where $(\mathcal{V})$ 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.

7.1.1 Shape Parameterization

The airfoil is parameterized using the RBF-based direct manipulation technique described in Section §2.1 and used in Section §6.1. Figure 27 shows the design variable locations for the first few shape control levels. We consider several static shape parameterizations (with 6, 14, 30 and 62 shape design variables) and compare their performance to two progressive shape control strategies starting from two design variables: (1) nested uniform refinement and (2) adaptive refinement. A maximum tree depth equivalent to the 62-DV parameterization is used, so that the two progressive approaches can, if necessary, ultimately recover the 62-DV search space, while preventing the shape control from becoming unreasonably closely spaced.

7.1.2 Adaptive Strategy

The trigger for the progressive and adaptive approaches was based on slope reduction, with a reduction factor of $r = 0.01$. A large window of $w = 6$ was used for the first 3 levels, while the constraints were being driven to satisfaction. In the presence of violated constraints, SNOPT's merit function undergoes large fluctuations, which can cause early slope-based triggering. After constraint satisfaction, the window was reduced to $w = 2$ for efficiency. For the adaptive approach, the target growth rate[q] was set to 1.75× and the constructive search algorithm (Function 3) was used, with $w = 3$. As there are many constraints in this problem, we used the first-order KKT-based indicator I_{KKT_G} (Equation (12)) to rank candidate refinements. Hessian information would be useful here, but there is not currently an affordable way to compute aerodynamic functional Hessians for each candidate refinement.

[q] Actual growth rates are also affected by regularity rules.

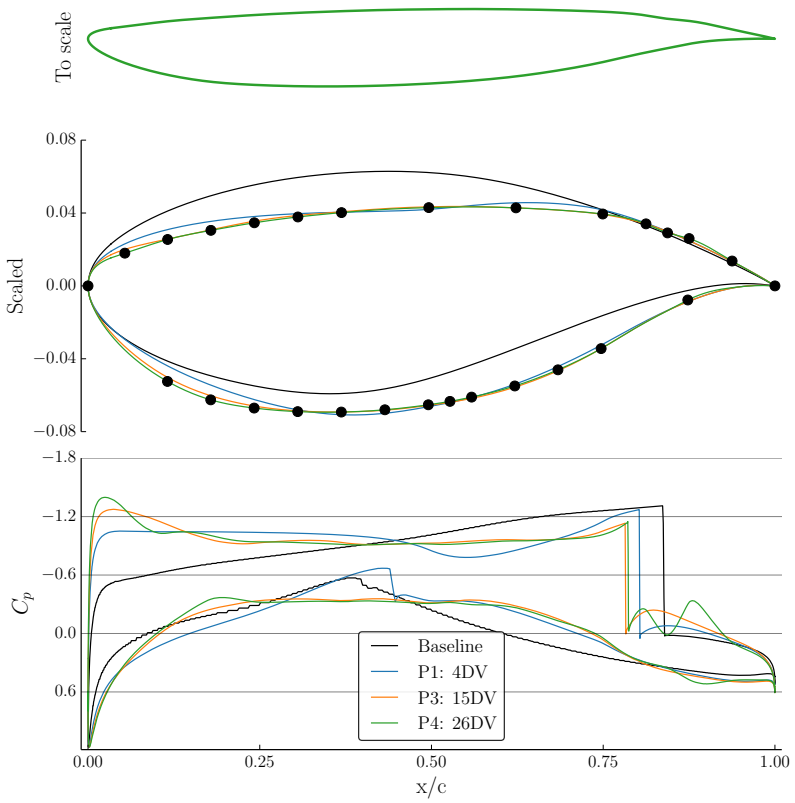


Figure 29: ***Transonic Airfoil:*** Optimization results for the adaptive parameterization approach. *Top*: Optimized airfoil to scale. *Middle*: Final airfoil from three intermediate search spaces (4-DV, 15-DV, 26-DV), showing 26-DV adapted parameterization. *Bottom*: Corresponding pressure profiles for the Mach 0.79 design point.

7.1.3 Optimization Results

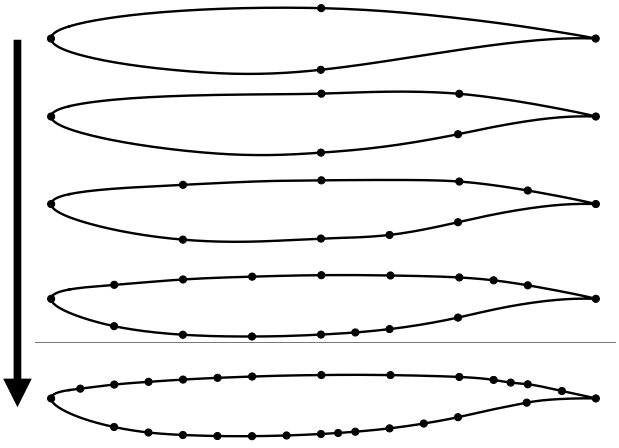
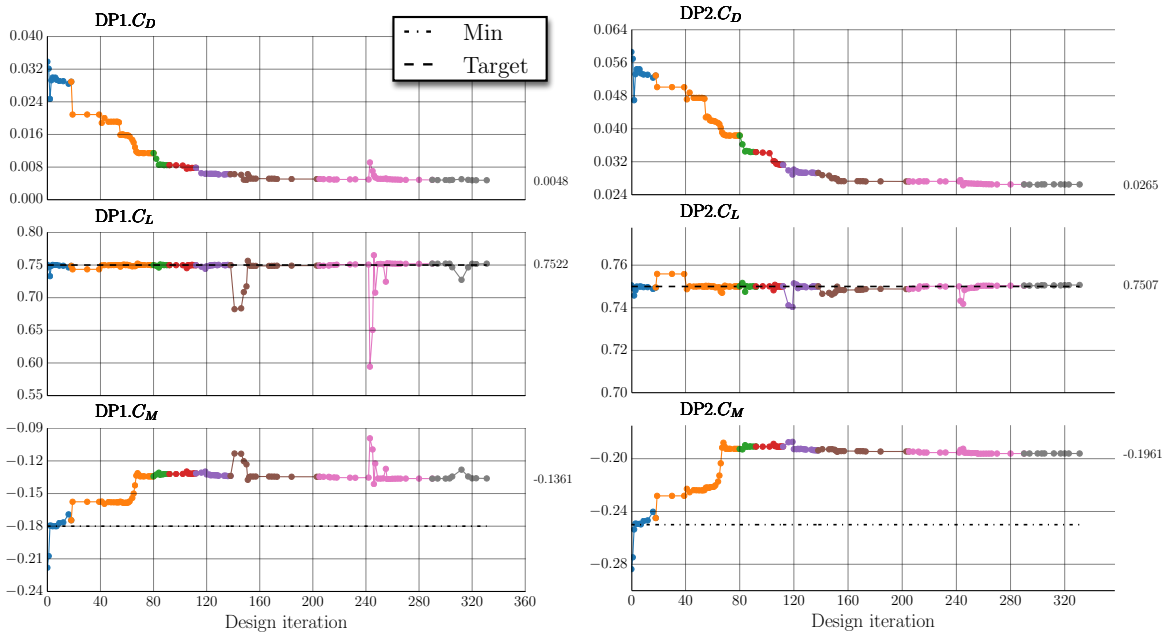


Figure 30: ***Transonic Airfoil:*** History of adaptive refinement, showing best airfoils attained under the first several parameterizations.

Figure 29 shows the airfoil shape achieved by three of the 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 camberline 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 drag at this design point is reduced from over 300 counts to 66 counts. Similarly, at Mach 0.82, the drag is reduced from about 600 counts to 276 counts. Figure 29 also shows the non-uniformly refined



(a) Mach 0.79 design point

(b) Mach 0.82 design point

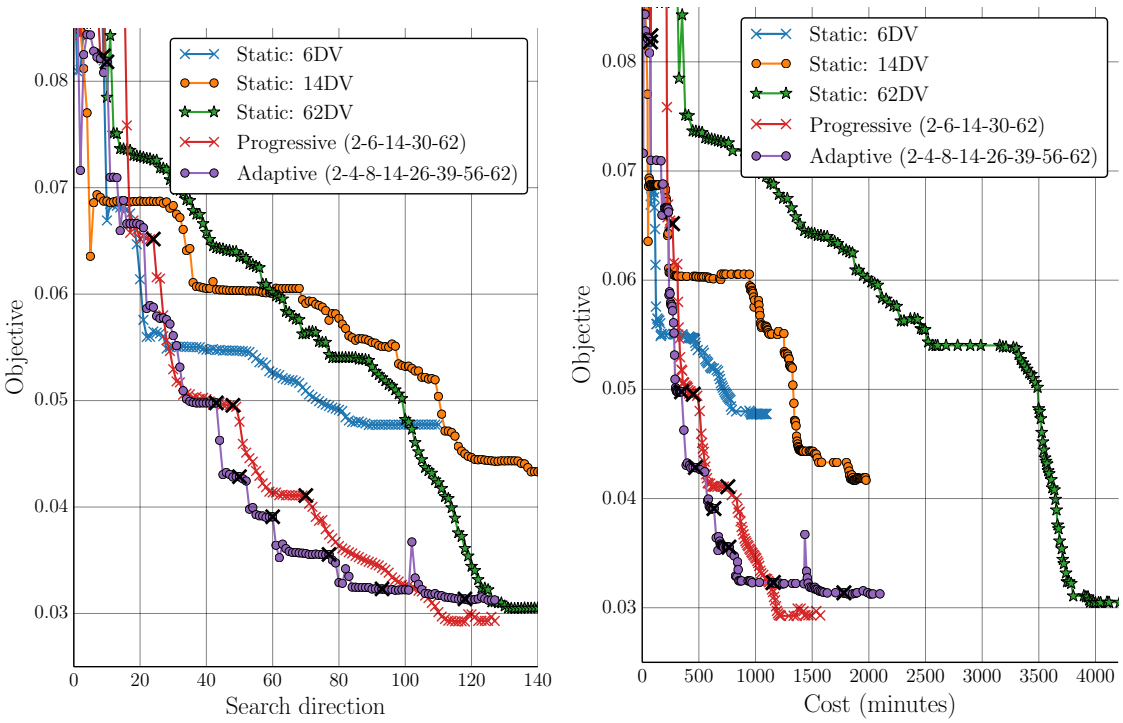
Figure 31: ***Transonic Airfoil:*** Convergence of aerodynamic functionals across all adaptively refined parameterization levels *(2-DV in blue, 4-DV in orange, etc.)*. Target/minimum constraint values shown in dashed lines.

final parameterization, which is the result of adding design variables over five levels. The sequence of the first few adapted parameterizations is shown in Figure 30.

Figure 31 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 thickness constraints are satisfied at every design. The area and trailing edge constraints are all active but satisfied by the end. At each re-parameterization, the quasi-Newton optimizer performs a "cold restart", which resets the Hessian approximation to the identity matrix. The main consequence is that the lift constraints are violated for the first few search directions immediately after refining, before snapping back to the target values. The design 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.

7.1.4 Comparison to Static Parameterizations

The left frame of Figure 32 compares the convergence of the drag objective for the various parameterizations. Initially, there is a somewhat convoluted startup period of 10-20 search directions, where the initially violated constraints were being driven to satisfaction at the expense of drag. Afterwards, 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 1 as following the "inside track" of the static parameterizations.



(a) Search direction history

(b) Design improvement vs. cost (on 24 Haswell cores) plotted only at successful search directions

Figure 32: ***Transonic Airfoil:*** Convergence of combined drag value ($C_{D_1} + C_{D_2}$) (ignoring satisfaction of constraints) for each parameterization method. ×-marks denote search space refinements.

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. Practically speaking, 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,[r] most likely because of the relative lack of smoothness in their design trajectories.

The computational savings are more stark in the right frame of Figure 32, which shows objective improvement vs. an estimate of wall-clock time[s]. The progressive and adaptive approaches reach the same objective value as the 63-DV parameterization in one-third of the time. Each design iteration included an adjoint-driven mesh adaptation to control discretization error,[24,33] a flow solution for each design point, and six adjoint solutions on the final adapted mesh to compute gradients for the aerodynamic functionals. Notably, Figure 32 includes the cost of long line searches, visible especially in the 62-DV parameterization. It also includes the usually neglected $\mathcal{O}(N_{DV})$ computational time due to computation of shape derivatives $\frac{\partial \mathbf{S}}{\partial \mathbf{X}}$ by the geometry modeler, followed by gradient

[r] We performed cold restarts when the static parameterizations stalled, to verify that no further progress could be made.

[s] Rough timings on 24 Intel Haswell cores

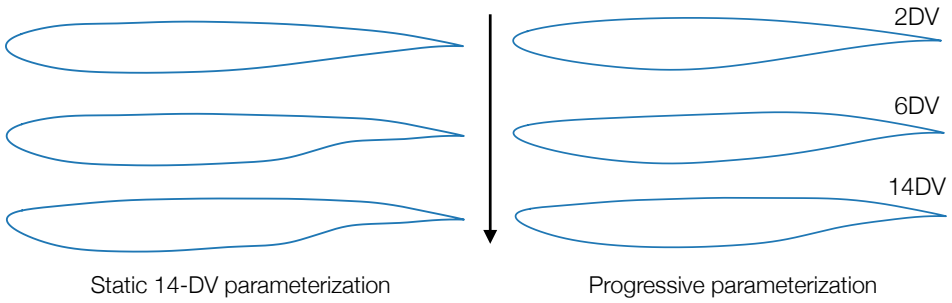


Figure 33: ***Transonic Airfoil:*** Shapes encountered during optimization under a progressive parameterization (*right*) are consistently much smoother than airfoils encountered under a static parameterization (*left*).

projections to compute $\frac{\partial \mathcal{J}}{\partial \mathbf{X}}$ and $\frac{\partial \mathcal{C}_j}{\partial \mathbf{X}}$. Adaptive refinement controls these costs by reducing the number of design variables. By adjusting the progressive and adaptive strategies, even more speedup is certainly possible. For example, the relatively delayed trigger could be tightened, as it resulted in several extended periods of little design improvement.

As a final note for this problem, Figure 33 shows several representative airfoils encountered during optimization. We observe that with a progressive or adaptive approach, the entire design trajectory involves smoother, more reasonable airfoils. This is a desirable characteristic from a robustness standpoint, and also because it makes it possible to stop at any point during optimization and have a reasonable design.

7.2 Transonic Transport Wing Design

In this example we consider the optimization of a transport wing Mach 0.85. The objective is to reduce drag subject to constraints on both lift and pitching moment.[t] The baseline geometry violates the pitching moment constraint. The wing geometry is that of the Common Research Model (CRM), scaled so that the mean aerodynamic chord has unit length. The planform is fixed, while variation in the vertical direction is permitted, including airfoil design and sectional twist. The twist is about the trailing edge and is fixed at the root, while the angle of attack is permitted to vary. The wing is required to maintain its initial volume V_0 and also to maintain at least 25% of its original local thickness t_0 everywhere. To approximate this continuous thickness constraint, we used a 10×10 grid of constraints distributed evenly across the plan-

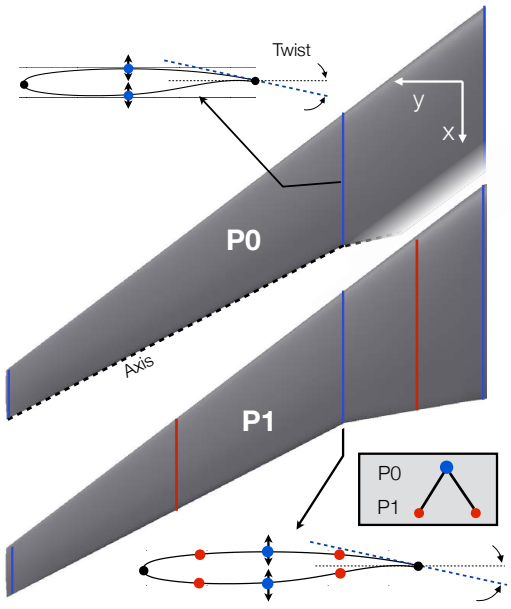


Figure 34: ***Transonic Wing:*** First two shape control levels (8-DV and 26-DV)

[t] Measured about the point (1.2077, 0, 0.007669) with the origin at the leading edge of the wing root.

form. The full optimization problem is

$$\begin{array}{lll} \underset{\mathbf{x}}{\text{minimize}} & & C_D \\ \hline \text{subject to} & & C_L = 0.5 \\ & (\mathcal{V}) & C_M \geq -0.17 \\ & & V \geq V_0 \approx 0.26291 \\ & & t_i \geq 0.25 t_{i_0} \quad \forall i \end{array}$$

7.2.1 Shape Parameterization

For this problem, we use the wing deformer described in Section §2.2, and previously used in Section §6.3. Both twist and airfoil section deformations are interpolated independently, while the planform definition is fixed. At each station a curve deformer (identical to the setup used for the previous two airfoil design examples) deforms the airfoil shape, after which the twist is applied. As before, the twist is in the streamwise plane about the trailing edge and is linearly interpolated. Control over airfoil sections and twist can happen at different stations, allowing for "anisotropic" shape control. For example, the twist control may have a higher spanwise resolution than the airfoil control. Similarly, each airfoil control station can offer different resolution for shape control.

Figure 34 shows the first two parameterizations ("P0" and "P1"). P0 allows twist at the tip and break (fixed at the root) and rough camber and thickness control (two control points each on the root, break and tip sections). There are initially eight shape design variables, plus the angle of attack. To refine the parameterization, new control stations are added at the spanwise midpoints between the existing stations, and simultaneously uniformly refine the airfoil control at each existing station. Two additional parameterization levels ("P1" and "P2") are automatically generated when needed, with 26 and 70 geometric design variables, respectively.

7.2.2 Optimization Results

Figure 35 shows the convergence of the aerodynamic functionals over the three parameterization levels. Under "P0", the initially violated pitching constraint is driven to satisfaction. To do this, large airfoil deformations are enacted, as shown in Figure 36 (blue lines), with a resulting sharp increase in drag. After adding more shape control resolution, the drag is rapidly driven down almost to the initial value, while nearly satisfying the constraints. The airfoil sections (Figure 36, orange lines) relax to more subtle changes from the baseline shape. The thickness and volume constraints are met at every design iteration.

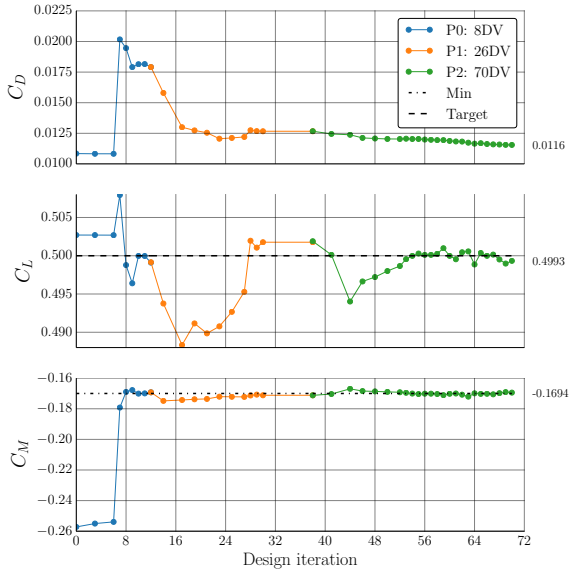
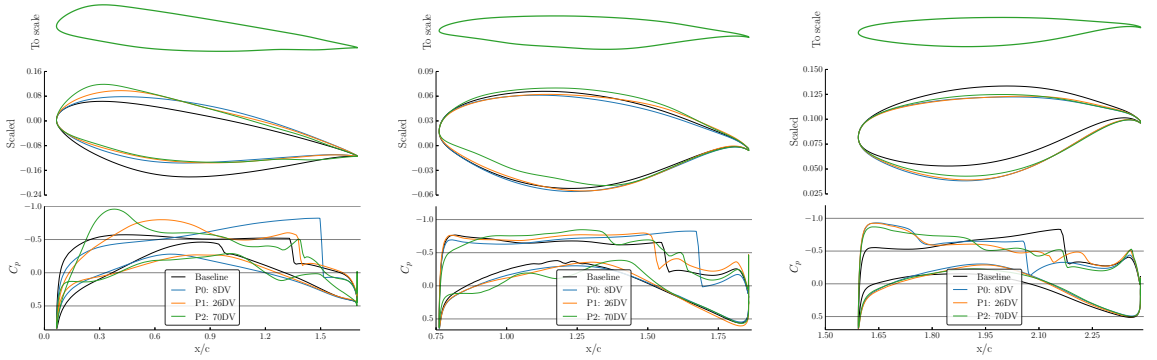
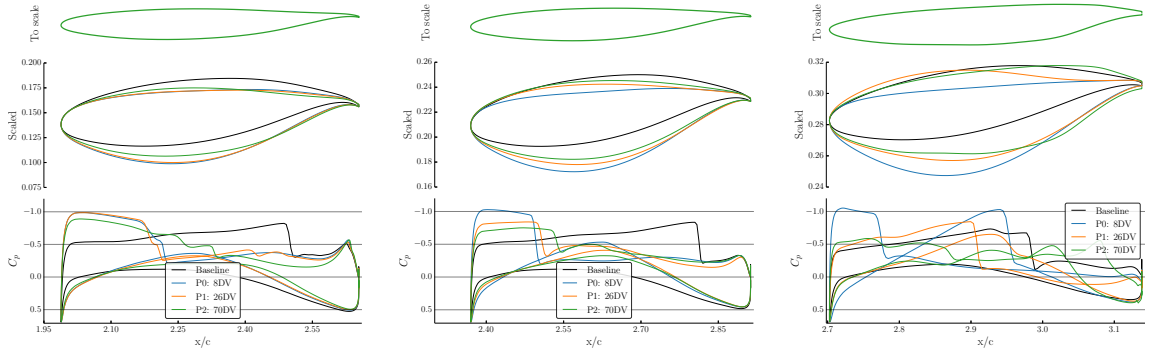


Figure 35: *Transonic Wing:* Convergence of aerodynamic functionals (only plotted at successful search directions)



(a) 2.35% **(b)** 26.7% **(c)** 55.7%



(d) 69.5% **(e)** 82.8% **(f)** 94.4%

Figure 36: *Transonic Wing:* Airfoil cuts and inviscid solution pressure profiles.

All of the constraints are nearly satisfied by the end, with only a slight drag penalty associated with having to meet the initially violated pitching moment constraint.

The flow mesh was automatically adapted for every design iteration, with about 12-18 million cells, depending on the design iteration, which was adequate to drive the optimization forward. This example demonstrates the ability of the adaptive shape optimization system to automatically solve a standard 3D aerodynamic optimization problem with constraints. Importantly, after the initial problem setup, no user intervention or problem modification was required for the remainder of the optimization.

The design improved substantially at each parameterization level, including in the finest 71-DV search space. This suggests that, although the optimization may have converged with respect to the existing shape parameters, it has not yet converged with respect to the refinement of the shape control resolution. By monitoring the amount of design improvement achieved under successive search spaces, the system is able to inform a designer about convergence towards *continuous* optimality, information that is not typically available under a static-parameterization approach.

8 Summary and Conclusions

In a progressive shape control approach, the search space is enriched automatically as the optimization evolves, eliminating a major time-consuming aspect of shape design, and freeing the designer to focus on good problem specification. Recognizing that different design problems may call for different shape control, and that for unfamiliar problems this may be difficult to predict, we developed an *adaptive* approach that aims to discover the necessary shape control while concurrently optimizing the shape. Results indicate that with progressive parameterization, the design trajectory is smoother, leading to more robust design improvement and offering the ability to stop at any point and have a reasonable design. The work also indicated that the optimization often achieves faster design improvement (as much as 3× in some cases) over using all the design variables up front. Additional important benefits of this approach include:

- **Automation**: By automating search space refinement, both user time and dependence on designer expertise is greatly reduced.
- **Completeness**: The full design space can be explored more thoroughly. The search is not restricted to the initial parameterization, and as refinement continues, the design space approaches that of the continuous problem.
- **Feedback**: The refinement pattern of the parameterization conveys useful information to a designer about the specific design problem being solved.

While the studies presented in this report use the discrete geometry modeler developed under this seedling initiative, the implementation is specifically architected to be modeler agnostic – the approach can work with any geometry modeler that meets certain requirements. While some development is required to prepare an existing modeler for adaptive use, the computational acceleration and reduction in manual setup time for each optimization strongly justifies this expenditure. With modest tailoring, the system could also invoke different aerodynamic or multi-disciplinary design frameworks.

There are two main tunable aspects of the refinement pacing: the trigger sensitivity and the growth rate in the number of parameters. We showed that these two settings are important for efficiency, and that careful strategies enable the adaptive approach to solidly outperform the standard static approach. All told, the implementation added only about 10 new parameters to tune the adaptation strategy. In the future, some of these choices can certainly be robustly automated.

Investigations are still underway to examine whether any Hessian information (in a new candidate search space) can be approximated for aerodynamic objectives. This has the potential to further improve the selection of candidate parameters. For highly redundant candidate pools, the search procedure might be accelerated by using information on the orthogonality of the deformation modes, or perhaps alternate search procedures or ranking strategies would be more effective.

8.1 Future Work

Looking to the future, automated shape optimization is certain to play an ever-increasing role in design. As the designer no longer needs to specify the exact deformation modes

by which a surface is permitted to be modified, care must be taken to explicitly specify (via constraints) how it may *not* be modified, to prevent the optimizer from taking advantage of weaknesses in the problem formulation. Many of these constraints, such as non-self-intersection, smoothness, or limits on excessive curvature, can be codified. This development would help regularize the optimization and likely lead to superior results.

A major open area for research is continuous adaptation of the shape parameter location. In this work we discussed a purely discrete refinement strategy (akin to h-refinement of flow meshes). A natural alternative would be to consider something similar to r-refinement – where parameters are redistributed, and placed at regions where the objective is most sensitive to shape modification. Other major areas of future work include using the technique with constructive solid modelers, or modelers with fixed parameter pools. Finally, nothing is unique about the aerodynamic objectives pursued within this effort, and the approach can be applied to structural, acoustic or other design disciplines. Clearly, this approach offers tremendous potential throughout the range of disciplines considered in MDAO settings, and combined aero-structural optimization is a topic of immediate interest.

References

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

[2] Han, X. and Zingg, D., "An adaptive geometry parametrization for aerodynamic shape optimization," *Optimization and Engineering*, Vol. 15, No. 1, 2013, pp. 69–91.

[3] 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.

[4] Hwang, J. T. and Martins, J. R. R. A., "A Dynamic Parametrization Scheme for Shape Optimization Using Quasi-Newton Methods," *AIAA Paper 2012-0962*, Nashville, TN, January 2012.

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

[6] Anderson, G. R., Aftosmis, M. J., and Nemec, M., "Parametric Deformation of Discrete Geometry for Aerodynamic Shape Design," *AIAA Paper 2012-0965*, Nashville, TN, January 2012.

[7] 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.

[8] Rodriguez, D. L., Aftosmis, M. J., Nemec, M., and Smith, S. C., "Static Aeroelastic Analysis with an Inviscid Cartesian Method," *AIAA Paper 2014-0836*, National Harbor, MD, January 2014.

[9] Rodriguez, D. L., Aftosmis, M. J., Nemec, M., and Anderson, G. R., "Optimized Off-Design Performance of Flexible Wings with Continuous Trailing-Edge Flaps," *AIAA Paper 2015-1409*, Kissimmee, FL, January 2015.

[10] Anderson, G. R. and Aftosmis, M. J., "Adaptive Shape Control for Aerodynamic Design," *AIAA Paper 2015-0398*, Kissimmee, FL, January 2015.

[11] Anderson, G. R., Aftosmis, M. J., and Nemec, M., "Aerodynamic Shape Optimization Benchmarks with Error Control and Automatic Parameterization," *AIAA Paper 2015-1719*, Kissimmee, FL, January 2015.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

[19] Berkenstock, D. C. and Aftosmis, M. J., "Structure-Preserving Parametric Deformation of Legacy Geometry," *AIAA Paper 2008-6026*, Victoria, BC, Canada, September 2008.

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

[21] 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.

[22] Nemec, M. and Aftosmis, M. J., "Parallel Adjoint Framework for Aerodynamic Shape Optimization of Component-Based Geometry," *AIAA Paper 2011-1249*, Orlando, FL, January 2011.

[23] 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.

[24] Nemec, M. and Aftosmis, M. J., "Output Error Estimates and Mesh Refinement in Aerodynamic Shape Optimization," *AIAA Paper 2013-0865*, Grapevine, TX, January 2013.

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

[26] Bisson, F., Nadarajah, S., and Shi-Dong, D., "Adjoint-Based Aerodynamic Optimization of Benchmark Problems," *AIAA Paper 2014-0412*, National Harbor, MD, January 2014.

[27] 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," *AIAA Paper 2014-0568*, National Harbor, MD, January 2014.

[28] Telidetzki, K., Osusky, L., and Zingg, D. W., "Application of Jetstream to a Suite of Aerodynamic Shape Optimization Problems," *AIAA Paper 2014-0571*, National Harbor, MD, January 2014.

[29] Poole, D. J., Allen, C. B., and Rendall, T. C. S., "Application of Control Point-Based Aerodynamic Shape Optimization to Two-Dimensional Drag Minimization," *AIAA Paper 2014-0413*, National Harbor, MD, January 2014.

[30] Amoignon, O., Navratil, J., and Hradil, J., "Study of Parameterizations in the Project CEDESA," *AIAA Paper 2014-0570*, National Harbor, MD, January 2014.

[31] 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.

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

[33] Nemec, M. and Aftosmis, M. J., "Toward Automatic Verification of Goal-Oriented Flow Simulations," Tech. Memorandum NASA/TM-2014-218386, National Aeronautics and Space Administration, Ames Research Center, 2014, http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150000864.pdf, visited Feb 2015.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704–0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01-05-2015	Technical Memorandum	

4. TITLE AND SUBTITLE
Adaptive Shape Parameterization for Aerodynamic Design

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
George R. Anderson and Michael J. Aftosmis

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Ames Research Center
Moffett Field, CA 94035

8. PERFORMING ORGANIZATION REPORT NUMBER
L–

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)
NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
NASA/TM–2015–Seedling Phase 2 Final Report

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 02
Availability: NASA CASI (443) 757-5802

13. SUPPLEMENTARY NOTES
An electronic version can be found at http://ntrs.nasa.gov.

14. ABSTRACT

This report concerns research performed in fulfillment of a 2.5-year NASA Seedling Fund grant to develop an adaptive shape parameterization approach for aerodynamic optimization of discrete geometries. The overarching motivations for this work were the potential to radically reduce manual setup time and achieve faster and more robust design improvement, especially for problems where many design variables are required. The primary objective was to develop a fully-functional prototype system that automatically and adaptively parameterizes discrete geometries for aerodynamic shape optimization. In support of this, we also matured the discrete geometry engine that underlies the work. Various applications are presented that demonstrate broad support and uptake for the discrete geometry tools developed in this work. This report outlines our theoretical approach to adaptive parameterization, provides detailed, practical implementation notes, and contains several verification studies and design examples. These studies demonstrate substantial wall-clock time computational savings, smoother shapes, and superior final designs compared to a standard static-parameterization approach. They also demonstrate that the adaptive system can autonomously discover the parameters necessary to solve an optimization problem. As a whole, this work is an important step towards greater automation in solving the unfamiliar aerodynamic shape design problems of the future.

15. SUBJECT TERMS
design optimization, parametric, adaptive, refinement

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
U	U	U	UU		19b. TELEPHONE NUMBER *(Include area code)* (443) 757-5802