

# Adaptive Shape Control for Aerodynamic Design

George R. Anderson \*

Stanford University, CA

Michael J. Aftosmis †

NASA Ames Research Center, Moffett Field, CA

We present an approach to aerodynamic optimization in which the shape parameterization is progressively and automatically refined. The process consists of an alternating sequence of optimizing within the current search space, and then refining the parameterization to enable the discovery of superior designs. We show that this approach reduces computational cost by optimizing in search spaces of appropriate dimensionality. By automating time-consuming aspects of shape control refinement, it also reduces human cost and dependence on designer expertise. In addition to uniform shape control refinement, we also discuss adaptive refinement, where the goal is to selectively add only the shape control with the most potential to improve the aerodynamic performance. Potential design improvement is estimated by comparing local objective and constraint gradients, which are computed at low cost by reusing existing adjoint solutions. A priority queue of the most effective candidate shape parameters is then maintained using an efficient constructive search procedure. We first demonstrate adaptive shape control on an multipoint airfoil drag minimization problem with many constraints, where our system achieves equivalent design improvement to a fine, fixed parameterization, but in one-third of the wall-clock time. We also establish a 3D shape-matching benchmark, in which our system automatically discovers the shape parameters necessary to match a target shape. This approach is an important step towards greater automation in solving the unfamiliar aerodynamic shape design problems of the future.

## Nomenclature

$C, \mathbf{C}$	Shape control	$w$	Window width
$\mathcal{C}, \mathbf{c}$	Constraint functional(s)	$X, \mathbf{X}$	Design variable value(s)
$D(\mathbf{X})$	Deformation function	$\psi$	Adjoint solution
$g, \mathbf{g}$	Growth rate(s) in number of parameters		
$I$	Importance indicator		<i>Subscripts</i>
$\mathcal{J}$	Objective functional	$(\cdot)_c$	Candidate shape control
$N_{(\cdot)}$	Number of $(\cdot)$	$(\cdot)_G$	Gradient
$\mathcal{O}$	Asymptotic order	$(\cdot)_H$	Hessian
$P(\mathbf{C})$	Function that generates deformation modes	$(\cdot)_x$	Static shape control
$\mathbf{Q}$	Flow variables		<i>Abbreviations</i>
$r$	Slope reduction factor for trigger	$DV$	Design variable
$\mathcal{S}$	Continuous surface	$KKT$	Karush-Kuhn-Tucker
$\mathbf{S}$	Discrete tessellated surface		

## I. Introduction

**A**UTOMATED flow meshing and simulation tools play a central role in aerodynamic shape optimization. In this work, we examine the degree to which the shape parameterization might also be automatically and adaptively refined during optimization. The primary goals of this approach are to reduce manual setup time and to achieve faster and more robust design improvement, especially for problems where many design variables are required.

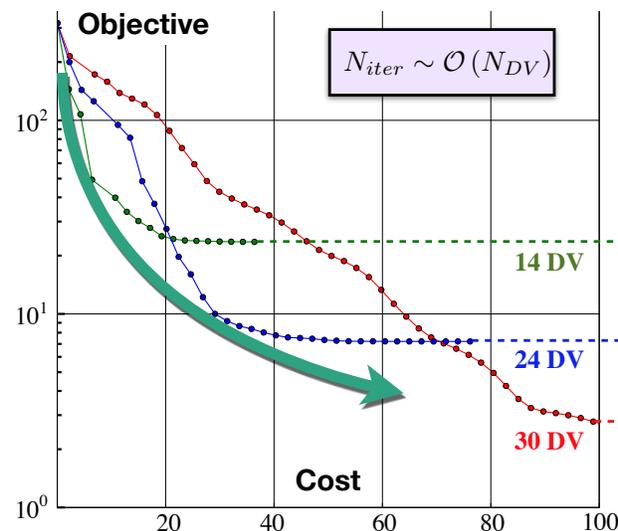
\*Ph.D. Candidate, Dept. of Aeronautics and Astronautics. george.anderson@stanford.edu. Member AIAA.

†Aerospace Engineer, Applied Modeling and Simulation Branch, MS 258-5. michael.aftosmis@nasa.gov. Assoc. Fellow AIAA.

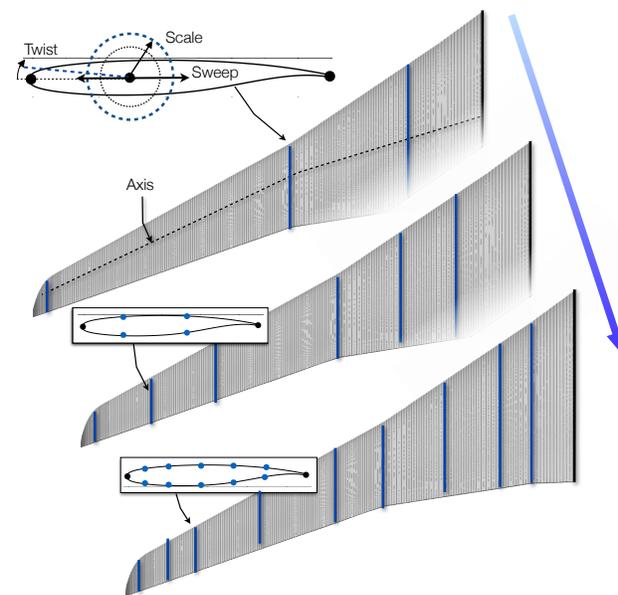
Under a traditional *static* parameterization approach, the space of all reachable shapes is prescribed before each optimization begins. This can restrict the design space in unnecessary ways, needlessly hindering the discovery of superior designs outside this envelope. One recourse is to use a very 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, a designer will typically perform an optimization and then manually refine the parameterization if necessary — a time-consuming task. In this work, we develop a *progressive* parameterization approach, where we start in a coarse search space, and then automatically transition to finer parameterizations at strategic moments. This constitutes a single optimization process that encourages rapid design improvement early on, while driving the shape towards the local optimum of the continuous problem.

Figure 2 illustrates our basic “progressive parameterization” approach. The designer 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” (fixed) 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 gradually becomes available for exploration.

This system essentially automates a practical approach to design. The designer does not need to predict the necessary shape parameters before design begins. Rather, the important shape parameters emerge as a natural *consequence* of optimization, and are clearly visible to the designer in the pattern of shape parameters. Large-scale changes are made early on, while detailed adjustments happen last, if resources permit.



**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.



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

## II. Background

Progressive shape parameterization for aerodynamic optimization was first proposed in 1993 both by Beux and Dervieux<sup>1</sup> and by Kohli and Carey.<sup>2</sup> A series of subsequent papers, primarily from researchers at INRIA, demonstrated that substantial design acceleration can be achieved with nested parameterizations.<sup>3–9</sup> Their approach is well-documented in a detailed report by Duvigneau,<sup>10</sup> which in large part motivated this work. Making an analogy to grid sequencing and multi-grid techniques in PDE solutions, they find that a sequence of refined design spaces performs substantially better than a fine, fixed parameterization.<sup>8,11</sup> Some authors

argue that optimization is an inherently “anti-smoothing” process,<sup>12</sup> and that design space sequencing is analogous to a preconditioner for the entire optimization process.<sup>5,11</sup> Their results generally suggest that redistribution of existing design variables is at least as effective as enriching the parameterization.<sup>10</sup> However, their adaptation criterion is based on a geometrical regularization operator, which is insensitive to the specific design problem.

Another early effort is that of Olhofer *et al.*,<sup>13</sup> who performed genetic optimization in adaptively refined design spaces. Because they used a gradient-free approach, they evolve several candidate parameterizations in parallel for several iterations before selecting the most promising one. Hwang and Martins developed a conceptually reversed approach that starts from an initial fine parameterization, and then uses coarsened search spaces to accelerate design improvement, analogous to grid sequencing in PDE solvers.<sup>14</sup> Their major achievement is an exact transfer of the Hessian information when switching between search spaces, avoiding the initial Hessian build-up time. The disadvantage of this approach is that the sequence of search spaces must be provided *a priori*. Other work has also demonstrated design acceleration using manually-refined parameterizations.<sup>15,16</sup>

The most functionally similar approach to ours is that of Han and Zingg,<sup>17,18</sup> who most notably introduced an adaptation criterion that is sensitive to the specific design problem, by using objective gradient information. In this work we further develop this approach, introducing a more detailed adaptation criterion that incorporates Hessian information and constraint gradients, and we present a search procedure for finding an effective ensemble of shape parameters. We also discuss an efficient triggering mechanism to determine when to transition to a new search space. Our approach focuses on discrete geometry modelers, which offer unique advantages for adaptive parameterization.

### III. Optimization with Progressive Shape Control

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

$$\min_{\mathcal{S}} \mathcal{J}(\mathcal{S}, \mathbf{Q}(\mathcal{S})) \quad (1)$$

$$\text{s.t. } a \leq \mathcal{C}_j(\mathcal{S}, \mathbf{Q}(\mathcal{S})) \leq b \quad (2)$$

where  $\mathcal{J}$  and  $\mathcal{C}_j$  are scalar functionals that involve performance metrics such as lift, drag, range, stability margins, maneuver loads, wing volume or operating costs. They may also relate to more specialized concerns such as reducing sonic boom ground signatures or environmental impact. In the case of aerodynamic functionals,  $\mathcal{J}$  and  $\mathcal{C}_j$  are evaluated after solving for the flow variables  $\mathbf{Q}$ .

#### A. Shape Parameterization

The surface  $\mathcal{S}$  is continuous, and so the design space is infinitely dimensional. To reduce the search space to a manageable dimension, the surface modifications are parameterized.<sup>a</sup> A shape parameterization technique,  $P$ , is a map from a vector  $\mathbf{C}$  describing the shape control to a search space, consisting of a deformation function,  $D$ , and a set of shape parameters  $\mathbf{X}$ . This deformation function takes the shape parameter values and generates a new surface  $\mathcal{S}$ :

$$\text{(Parameterize)} \quad P : \mathbf{C} \longrightarrow D \quad (3)$$

$$\text{(Deform)} \quad D : \mathbf{X} \longrightarrow \mathcal{S} \quad (4)$$

In other words,  $P$  describes how the shape control induces a set of shape parameters, while  $D$  describes how those shape parameters deform the surface. In standard optimization approaches, only Equation 4 is automated. In this work we also automate Equation 3. The shape parameters  $\mathbf{X}$ , or a subset thereof, serve as the design variables for optimization. The “search space” defined by  $D$  and  $\mathbf{X}$  is a subset of the full design space. The local linearization of  $D$  provides the shape derivatives  $\frac{\partial \mathcal{S}}{\partial \mathbf{X}}$ , which describe the deformation modes of each parameter, and which are used in gradient-based optimization. To help elucidate this somewhat terse terminology, we give a brief example.

---

<sup>a</sup>In this work we work exclusively with parameterized *modifications* of an existing surface. However, the discussion is completely accurate for constructive parameterized surface *generation* as well.

---

EXAMPLE: WING PLANFORM DESIGN

Consider the simple wing parameterization scheme illustrated in Figure 3. Twist, sweep and chord are interpolated between the spanwise stations. Here, the shape control  $\mathbf{C}$  is the vector of spanwise coordinates of the control stations (blue lines), indicating *where* twist, sweep or chord can be manipulated.  $P_{wing}$  interprets this compact, high-level description, expanding it into precise, detailed definitions of each deformation mode. This collection of deformation modes is encoded by the deformation function  $D$ , which takes the twist, sweep and chord values  $\mathbf{X}$  and generates a new surface. To refine the shape control, new stations are added, introducing new deformation modes.

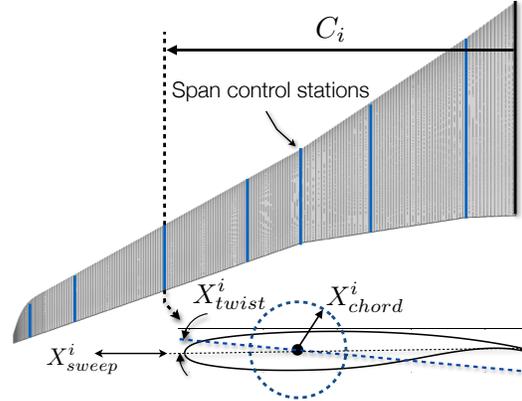


Figure 3: Wing planform parameterization

The distinction we have drawn 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 we transition to the next search space, we are modifying the shape control  $\mathbf{C}$ , while holding the shape fixed. There is not generally a one-to-one correspondence between the two. In the example above, each shape control element  $C_i$  induces three parameters (twist, thickness, sweep). Furthermore, each deformation mode is influenced not only by  $C_i$ , but also by the neighboring shape controllers,  $C_{i+1}$  and  $C_{i-1}$ , which also contribute to the interpolation.

## B. Progressive Parameterization

In standard shape optimization approaches, the shape control  $\mathbf{C}_\times$  is static and pre-determined by the designer. This induces a static search space  $D_\times(\mathbf{X}_\times)$ , which may be more or less effective at improving the objective function. In our approach, we instead use a *sequence* of shape control resolutions  $\mathbf{C}_0, \mathbf{C}_1, \mathbf{C}_2 \dots$ , which induces a sequence of search spaces  $D_0(\mathbf{X}_0), D_1(\mathbf{X}_1), D_2(\mathbf{X}_2) \dots$  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 more detailed search space.<sup>b</sup>

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.

In this work we use nested, hierarchical shape control, which implies a discrete approach to adding design variables. (In other words, we do not consider optimal continuous positioning of the shape controllers.) Figure 4 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

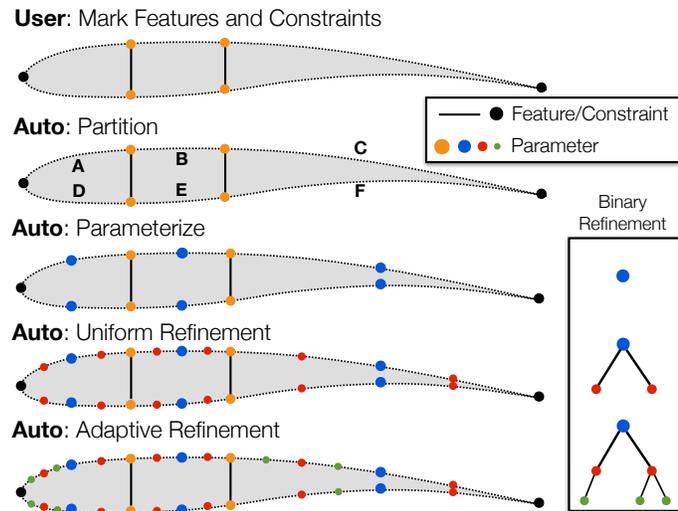


Figure 4: Progressive parameterization with discrete, hierarchical shape control refinement

<sup>b</sup>Although we do not consider it in this work, *removing* design parameters that are no longer useful is also a possibility.

leading and trailing edges or spar locations (black and orange dots in Figure 4). These features partition the curve into several regions. In each region we initially place a single shape controller (blue dots). Finer shape control is then gradually introduced as necessary through binary refinement. Conceptually, this allows the shape control to be viewed as a set of binary trees, with deeper levels corresponding to higher resolution shape control. This immediately raises the possibility of “adaptive” refinement, where we selectively refine only certain regions, as shown at the bottom of Figure 4. The goal of adaptive refinement is to add only the most important shape control to solve the given problem.

## IV. Implementation

The design loop now consists of an alternating sequence: optimize within the current search space, and then refine the shape control. This process is outlined in Algorithm A. The function  $Optimize(\cdot)$  represents a standard parametric shape optimization framework, which for reference is outlined in the Appendix.  $Parameterize(\cdot)$  is the modeler-dependent implementation of Equation 3, which generates a search space (i.e. a deformation function and design variables) from the shape control description. 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. Unlike in static optimization approaches, where this is a manual pre-processing step, here it is automated.

The refinement strategy is governed by three new functions, each of which will be discussed in more detail in the following sections. 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 we will call “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. The ultimate convergence criterion is based on objective convergence as the discrete shape control  $\mathbf{C}$  approaches continuous shape control (direct optimization of  $\mathbf{S}$  or  $\mathcal{S}$ ).

Algorithm A integrates three basic software components: (1) a geometry modeler, (2) a shape optimization framework, and (3) scripts to guide search space refinement. Conceptually, these components can be viewed as standalone tools, although in practice there is a substantial degree of communication among 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.

### A. Shape Optimization Framework

For the function  $Optimize(\cdot)$  in Algorithm A, we use a gradient-based aerodynamic shape design framework<sup>19</sup> that uses an embedded-boundary Cartesian mesh method for inviscid flow solutions. Objective and constraint gradients are computed using an adjoint formulation. We leverage these same adjoint solutions 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<sup>20</sup> is used, enabling proper treatment of linear and nonlinear constraints.

<b>Algorithm A: Optimization with Adaptive Shape Control</b>	
<b>Input:</b>	Initial surface $\mathbf{S}_0$ and shape control $\mathbf{C}_0$ , objective $\mathcal{J}$ , constraints $\mathcal{C}_j$ , shape control growth rate $\mathbf{g}$
<b>Result:</b>	Optimized surface $\mathbf{S}$
<hr/>	
$\mathbf{C} \leftarrow \mathbf{C}_0, \mathbf{S} \leftarrow \mathbf{S}_0$	
<b>repeat</b>	
$D, \mathbf{X}_0 \leftarrow Parameterize(\mathbf{S}, \mathbf{C})$	
$\mathbf{S} \leftarrow Optimize(D, \mathbf{X}_0, \mathcal{J}, \mathcal{C}_j)$ <b>until</b> $Trigger(\cdot)$	
$\mathbf{C}_c \leftarrow GetCandidateShapeControl(\mathbf{C})$	
<b>if</b> adaptive <b>then</b>	
$\mathbf{C} \leftarrow AdaptShapeControl(\mathbf{C}, \mathbf{C}_c, \psi, \mathbf{S}, \mathbf{g})$	
<b>else</b>	
$\mathbf{C} \leftarrow \mathbf{C} \cup \mathbf{C}_c$ // Uniform refinement	
<b>end</b>	
<b>until</b> convergence of $\mathcal{J}$ and $\mathcal{C}_j$ w.r.t. $\mathbf{C}$	
<hr/>	
<b>Function colors:</b>	
Parametric geometry modeler	
Refinement strategy (modeler independent)	

## B. Parametric Geometry Generation

Throughout this work we optimize shapes 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.<sup>21</sup> This extension allows Blender to serve as a geometry engine for optimization. For this work we developed custom shape parameterization plugins, which are described with the corresponding examples in Section VI. Shape sensitivities are computed analytically for each deformer. 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 these geometry tools via *XDDM*, an XML-based protocol for design markup.<sup>19</sup>

When using discrete geometry, the shape is preserved exactly when transferring between shape search spaces, which is a useful feature for our approach (and required for adaptive refinement). By contrast, constructive modelers require a re-fitting procedure when re-parameterizing. With a few notable exceptions,<sup>6, 18</sup> this refitting is usually approximate, introducing a “jump” in the shape and typically a setback in the design process. With discrete geometry, exact shape preservation means that no time is lost during these transfers.

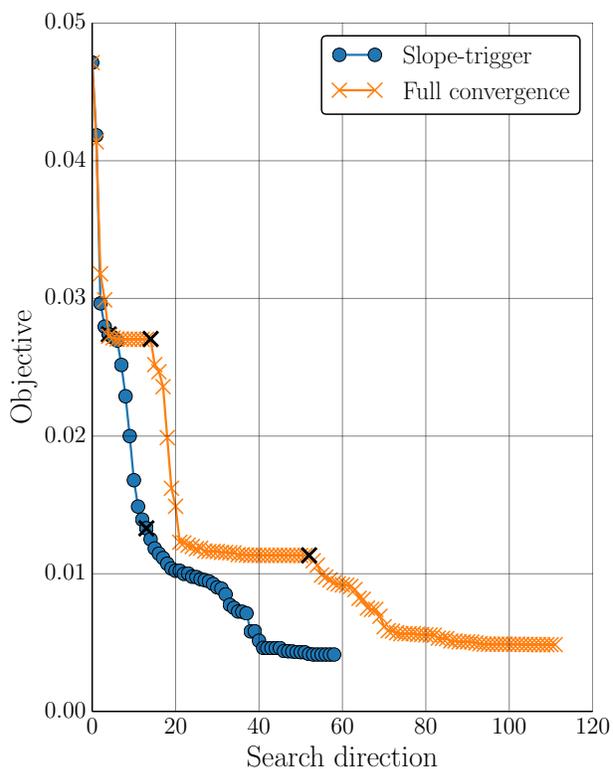
We view each parameterization as a binary tree, restricting refinement to the midpoints between existing shape controllers<sup>c</sup>, although we can also search several levels deep from the current parameterization. Additionally, we prohibit large discrepancies between the refinement depth of adjacent regions on the surface. This maintains smoothness in the spacing between parameters, which was found to be important for robustness in certain 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.

## C. Triggering Search Space Refinement

The *Trigger*( $\cdot$ ) function in Algorithm A is a stopping-condition that terminates the optimization on the current set of design variables and initiates a parameter refinement. The trigger is critical for efficiency, as demonstrated in Figure 5. 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 sluggish design improvement. 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<sup>10</sup> and optimization with progressively refined PDE meshes.<sup>22</sup> We ruled out simplistic triggers, such as setting the maximum number of search directions 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.

### 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<sup>18</sup> used this approach to achieve maximal design improvement within each search space. However, we found that on many problems, this type



**Figure 5:** *Orange:* Optimizing to convergence on each level leads to slow design improvement. *Blue:* Using aggressive slope-based transitions permits much faster design improvement. X-marks denote parameterization refinements.

<sup>c</sup>Refinement can also be directionally biased, for example to cluster parameters towards the leading edge of a wing.

of trigger often excessively delays refinement in early search spaces, as demonstrated in Figure 5. 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 efficient an cutoff is difficult without prior experience with a particular problem, which we would like to avoid for an automated approach.

## 2. Slope Trigger

We propose a simple alternative approach: to trigger when the rate of design improvement starts to substantially diminish. To detect this, we monitor the slope of the objective convergence with respect to a suitable measure of computational cost. We terminate the optimization when this slope falls below some fraction  $r$  of the maximum slope that has occurred so far. We found this strategy 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  $\mathcal{O}(10^{-2})$  while a functional based on operating range may be  $\mathcal{O}(10^5)$ .

The slope is evaluated at major search iterations, which is monotonically decreasing.<sup>d</sup> The objective slopes can be non-smooth, which can cause false triggering. To alleviate this, we use running averages over a small window, which effectively smooths the objective history. This helps prevent premature triggering, but it causes a lag equal 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 correct: 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.<sup>e</sup> 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.

## V. Adaptive Shape Control Refinement

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. Even under an adjoint formulation, where the cost of gradient computations are much less sensitive to the number of design variables  $N_{DV}$  than under a finite difference approach, minimizing the number of design variables is still generally highly desirable. Non-negligible  $\mathcal{O}(N_{DV})$  costs remain, namely the computation of geometric surface derivatives and subsequent gradient projections.

To investigate the possibility of selective adaptation, we now develop a systematic method for choosing an effective combination of refinement locations from among the myriad candidates. To do so, we first generate a set of candidate shape control refinements. Then we predict the effectiveness of each candidate by computing an “importance indicator”, which is based on problem-aware metrics, such as the objective and constraint gradients. We then select the candidate shape control refinement with the highest predicted performance.

### A. Effectiveness Indicator

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. Consider Figure 6, 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

<sup>d</sup>For attainable inverse design problems, the slope should be measured in log-space to better reflect the problem.

<sup>e</sup>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.

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.

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. Considering first the unconstrained case, the indicator is

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

which prioritizes search spaces with the highest capacity for design improvement.

In the next section, 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.<sup>f</sup> 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:

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

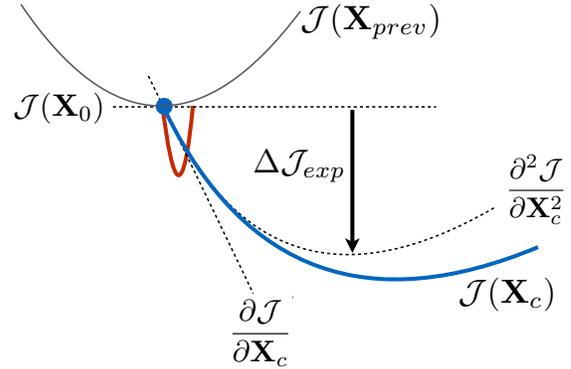
For many aerodynamic problems, however, the relative design variable scales are orders of magnitude different from each other. As we will show in the next section, this can render  $I_G$  a grossly ineffective predictor of performance. To rectify this, a simple diagonal scaling matrix  $\mathbf{D}$  can also be used:

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

Ideally,  $\mathbf{D}$  is the diagonal of the Hessian. Roughly speaking, the diagonal of the Hessian encodes the relative scaling of each parameter, while the off-diagonal terms account for redundant potential among the parameters. Although even this diagonal is often impractical to compute, the Hessian approximation from the previous design space (built up by a quasi-Newton optimizer), might be transferred to the candidate design space, although this requires information about the similarity or spatial organization of shape parameters. We leave this possibility for future work. Although the diagonal values could theoretically be specified as scale factors by the user, as commonly done in optimization to improve the conditioning, we rule that out here for two reasons. Most importantly, it effectively de-automates the process. Additionally, 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.

The objective gradients have modest additional cost. For example, Function 2 shows how  $I_G$  might be assembled. During optimization, the adjoint solution  $\psi$  is used to efficiently compute gradients with respect to arbitrary shape design variables. Now, after a search space refinement is triggered, we *reuse* the adjoint solution from the final design in the previous search space to rapidly compute gradients with respect to the

<sup>f</sup>The BFGS method builds up a Hessian approximation in the previous search space, but it is not obvious how or if one can extrapolate this information to the candidate design variables.



**Figure 6:** 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.

#### Function 2: *GradientIndicator*(·)

**Input:** Surface  $\mathbf{S}$ , shape control  $\mathbf{C}$ ,  
objective adjoint solution  $\psi$   
**Result:** Indicator  $I_G$

---

```

D, X ← Parameterize(S, C)
foreach X_c^i in X_c do
    | ∂S / ∂X_c^i ← ShapeDerivative(D, X_c^i)
    | ∂J / ∂X_c^i ← ProjectGradient(ψ, ∂S / ∂X_c^i)
end
I_G ← ∑_i (∂J / ∂X_c^i)^2

```

new candidate design variables. Reuse of the adjoint is possible only if the geometry modeler exactly preserves the shape when changing search spaces, 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.

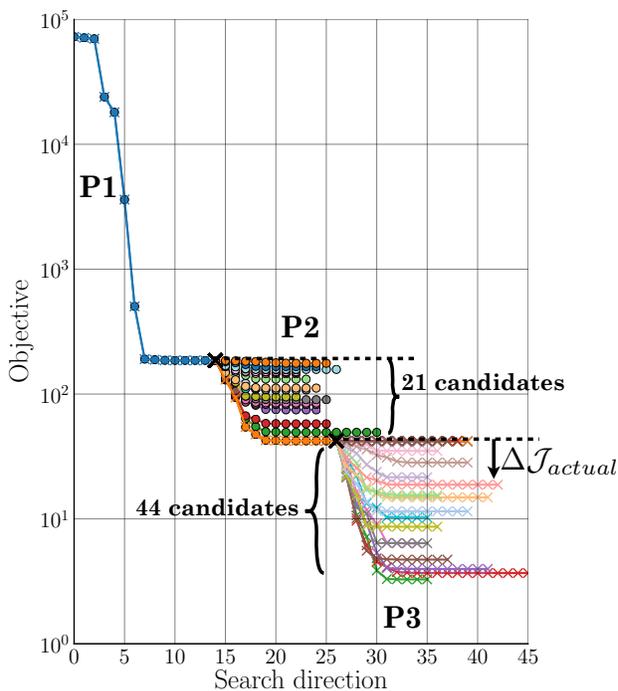
### 1. Indicator Validation

We briefly compare the performance of these three indicators on a geometric shape-matching problem. (The detailed setup for this problem is given in Section §VI.B.) 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 7. 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, 6 and 7) to predict the actual performance of the various candidate shape parameters.

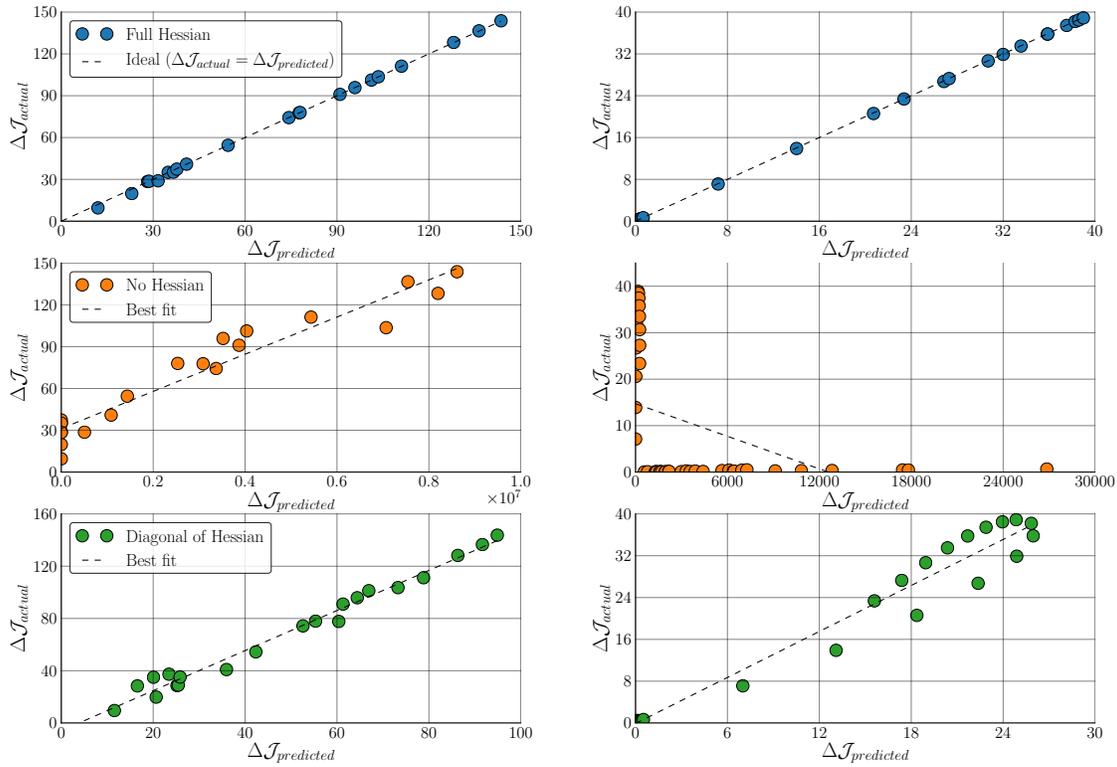
For this evaluation, we generate 21 candidate shape control refinements from the baseline parameterization. For each candidate, we run a separate optimization, where we add only that one shape parameter to the active set. (In practice, we would add several parameters at once; adding one at a time simplifies this validation study.) Figure 7 shows the objective convergence corresponding to each candidate parameter. Our 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 we predict the potential design improvement using Equations 5, 6 and 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 8 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. Our 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 8, 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 §VI.B, 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.<sup>14,18</sup> 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 7:** Indicator validation: **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 8:** Indicator validation: 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.

## 2. Constraint-Sensitive Indicator

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.<sup>18</sup> 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 constraints, which are treated as equality constraints and linearized about the current design:

$$\frac{\partial \mathbf{c}^T}{\partial \mathbf{X}_c} \mathbf{X}_c = \mathbf{b} \quad (8)$$

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, which may be inaccurate, but is an unavoidable consequence of linearizing the constraint functionals. As before, the constraint gradients with respect to the candidate design variables can be readily computed by reusing the existing constraint adjoint solution(s), following a process similar to Function 2. The constrained minimizer of the quadratic fit is the solution to the following system of equations

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

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  $\lambda$ :

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

We do not need to directly compute the minimizer  $\mathbf{X}_c^*$ . Instead, we substitute its functional form into the quadratic fit, which yields the expected *feasible* design improvement

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

Roughly speaking,  $I_{KKT}$  prioritizes parameterizations where the objective gradients are as orthogonal as possible to a linear combination of the active constraint gradients. Any available curvature information from the Hessian (middle term) corrects the prediction. Term A in Equation 11 is related to the KKT optimality metric. At the optimal design in the previous search space, Term A is zero (or nearly zero 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. In our experiments, the two indicators typically result in very similar rankings, but in some cases, Equation 11 may avoid adding ineffectual parameters. As before, if we treat the Hessian as the identity matrix, the indicator simplifies to a first-order prediction

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

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

## B. 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 parameters  $\mathbf{C}_c$ , not simply of each individual parameter  $C_c$ . 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, we 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

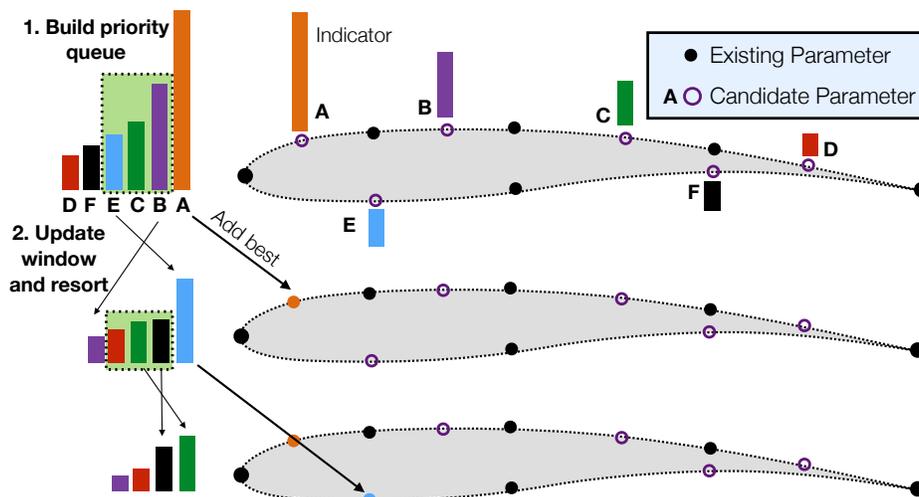


Figure 9: Constructive search algorithm for refining the shape parameterization.

if it enacts similar shape modifications. Put another way, the potential design improvement offered by two candidates may be mutually exclusive.

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 consider it, “metaheuristic” search procedures such as genetic optimization, could be used. However, these typically require large numbers of functional evaluations.

For this work we developed a “constructive” search procedure, illustrated in Figure 9. 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 §V.A. In the second phase, we make  $N_{add}$  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, we avoid adding redundant parameters.
- The cost for the entire search is bounded and  $\mathcal{O}(N_c)^g$ .

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 §VI.B.

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. In our environment, we observe highly practical running times, 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.

For certain special types of deformers (notably, Hicks-Henne bump functions<sup>23</sup> and Bernstein polynomials or Kulfan parameters<sup>24</sup>) 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.

<sup>g</sup>At most  $N_c(1 + \frac{w}{2})$  indicator evaluations are required:  $N_c$  to build the initial priority queue and  $\min(N_{add}, N_c - N_{add})$  to add the rest, because if we are adding more than half of the candidates, we can work backwards, removing one at a time.

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

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

```

// Phase 1. Build priority queue
queue ← ∅
foreach  $C_c$  in  $\mathbf{C}_c$  do
     $I$  ← ComputeIndicator( $\mathbf{S}, \mathbf{C} \cup C_c, \psi_i$ )
    queue.Add( $C_c$ , priority =  $I$ )
end

// Phase 2. Add shape control
 $N_{add}$  ←  $\text{int}(\text{len}(\mathbf{C}) \cdot \mathbf{g}[i])$ 
for  $i=1..N_{add}$  do
    foreach  $C_c$  in queue.Best( $w$ ) do
         $I$  ← ComputeIndicator( $\mathbf{S}, \mathbf{C} \cup C_c, \psi_i$ )
        queue.Update( $C_c$ , priority =  $I$ )
    end
     $C_{best}$  ← queue.pop()
     $\mathbf{C}$  ←  $\mathbf{C} \cup C_{best}$ 
end

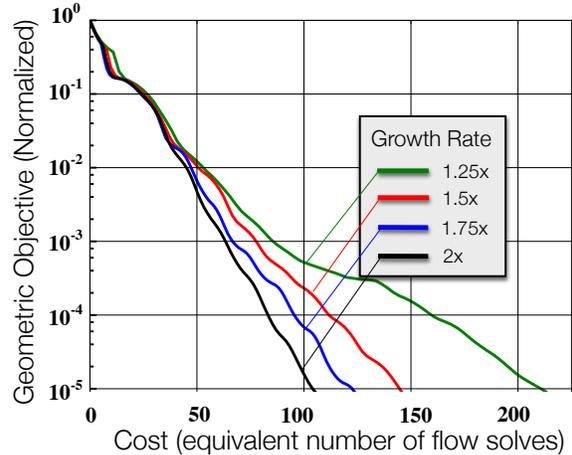
```

### C. Pacing

Setting the growth rate of the number of parameters ( $g$  in Algorithm A) 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.

For this work, we specify relative growth rates (e.g. “increase the number of design variables by 50%”). Figure 10 compares the performance of various growth factors from  $1.25\times$  –  $2\times$  on a geometric shape-matching problem. On this problem, a growth rate of  $2\times$  converges twice as fast as a growth rate of  $1.25\times$ . The optimal pace will depend on the problem. Here, the relative simplicity of the 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, we observe that slower growth rates are superior.

Other growth-setting strategies might also eventually prove useful, such as performing a cost-benefit estimation (especially with the Hessian-based indicator), or even removing some design variables from the active set for efficiency. An additional consideration is how many shape parameters to start with. As the first example will demonstrate, starting with a truly minimal search space (e.g. one or two variables) leads to stunted growth early on. We observe that it is often more effective to start with several design variables (at least 6-10), again dependent on the problem.



**Figure 10:** Performance of different growth rates on a geometric shape-matching objective. Each curve shows mean behavior over 10 randomized trials (trigger  $r = 0.25$ ).

## VI. Results

In a concurrently published applications paper,<sup>25</sup> we demonstrate our progressive (uniformly-refined) parameterization approach on four benchmark shape optimization problems. The benchmark problems are posed by the AIAA Aerodynamic Design Optimization Discussion Group and include two airfoil design problems, twist optimization for minimum induced drag, and a transonic wing design case. In that paper, we also discuss the importance of controlling discretization error in achieving robust design improvement.

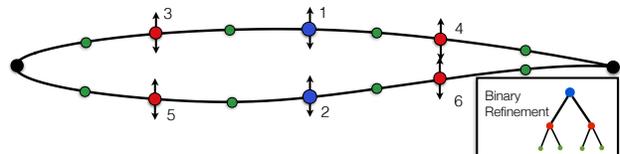
In this section, we demonstrate specifically *adaptive* shape control on two additional cases. The first example examines transonic airfoil design with two design points and many constraints, where optimization convergence is difficult. The second example establishes a benchmark geometric shape-matching problem that exercises our indicator and search procedure.

### A. Transonic Airfoil Design

In this example we consider multipoint transonic airfoil design. The purpose is to demonstrate our approach on a challenging 2D problem. We show that progressive shape control leads to a smoother design trajectory and accelerates the optimization.

#### 1. Problem Statement

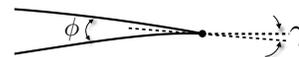
The baseline geometry is a unit-chord RAE 2822 airfoil, shown in Figure 11. The objective is to minimize an equally-weighted sum of drag at two flight conditions, Mach 0.79 and 0.82. Lift-matching and minimum pitching moment constraints are imposed at both design points. Because we are using an inviscid solver, we constrain the camber line angle  $\gamma$  at the trailing edge (see Figure 12) to prevent exces-



**Figure 11:** Baseline geometry for transonic airfoil design, showing first three levels of uniformly refined shape control (2-DV, 6-DV and 14-DV)

sive cambering that would result in poor viscous performance. We also specify a minimum and maximum geometric closing angle  $\phi$  at the trailing edge. Finally we require 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  $A_{RAE}$ . The complete optimization statement is

$$\begin{aligned}
& \text{minimize } \mathcal{J} = C_{D_1} + C_{D_2} \\
& \text{s.t. } C_{L_1} = C_{L_2} = 0.75 \\
& \quad C_{M_1} \geq -0.18 \text{ (V)} \\
& \quad C_{M_2} \geq -0.25 \text{ (V)} \\
& \quad 9^\circ \leq \phi \leq 13^\circ \\
& \quad \gamma \leq 6^\circ \text{ (V)} \\
& \quad A \geq A_{RAE} \approx 0.07787 \\
& \quad t_i \geq 0.9t_{RAE_i} \forall i
\end{aligned}$$



**Figure 12:** Geometric constraints at the trailing edge

where (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.

## 2. Shape Parameterization

The airfoil is parameterized using a *direct manipulation* technique. As shown in Figure 4, we explicitly specify the deformation of a set of “pilot points” along the curve, which serve as the design variables. Deformation of the remainder of the curve is interpolated using radial basis functions.<sup>9,26–28</sup> We choose the cubic basis function  $\phi = r^3$ , primarily because it requires no local tuning parameters, making it more amenable to automation. Shape control refinement is binary, with adjacency and midpoints defined in terms of arclength along the curve. In the language of Section III.A, the shape control  $\mathbf{C}$  is the parametric locations of the pilot points along the airfoil curve. Function  $P_{RBF}$  “binds” these locations to the surface, resulting in a deformation function  $D$ , which takes the control point deflections  $\mathbf{X}$  and generates a new surface.

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 2-DVs: (1) nested uniform refinement and (2) adaptive refinement. We set a maximum tree depth equivalent to the 62-DV parameterization, 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.

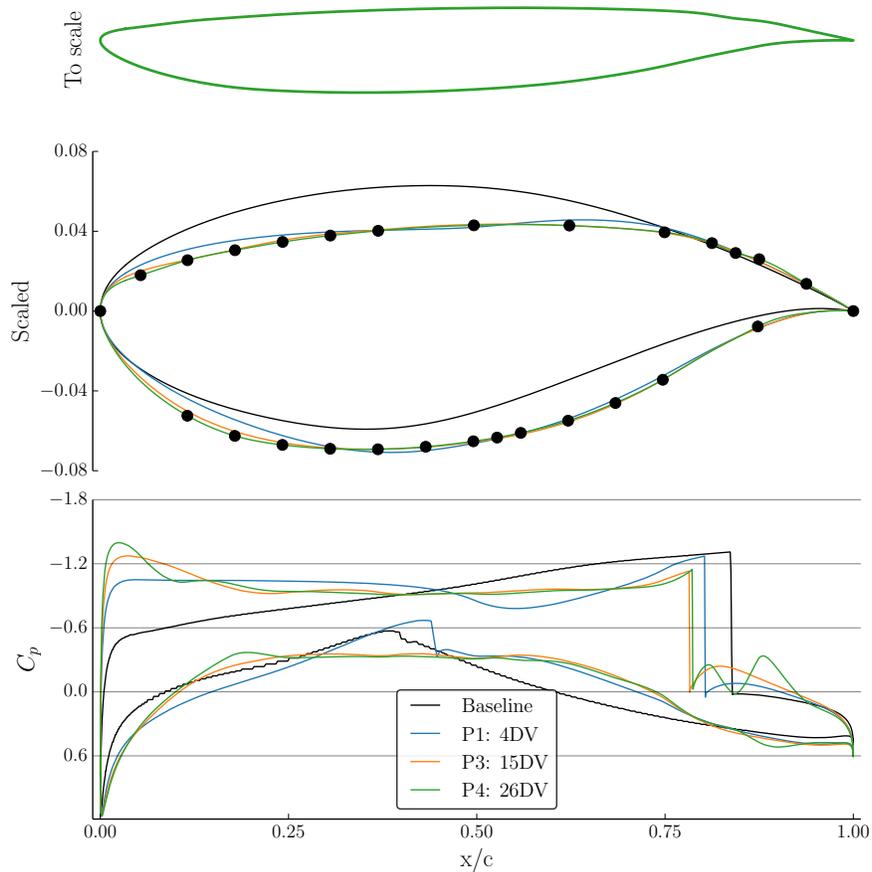
## 3. Adaptive Strategy

The trigger for the progressive and adaptive approaches was based on slope reduction, with a reduction factor of  $r = 0.01$ . We used a large window of  $w = 6$  for the first 3 levels, while the constraints are being driven to satisfaction. In the presence of violated constraints, SNOPT’s merit function undergoes large fluctuations, which can cause early slope-based triggering. Subsequently, we reduced the window to  $w = 2$  for efficiency. For the adaptive approach, we used a target growth rate<sup>h</sup> of  $1.75\times$  and used the constructive search algorithm (Function 3), 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 we do not currently have an affordable way to compute aerodynamic functional Hessians for each candidate refinement.

## 4. Optimization Results

Figure 13 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

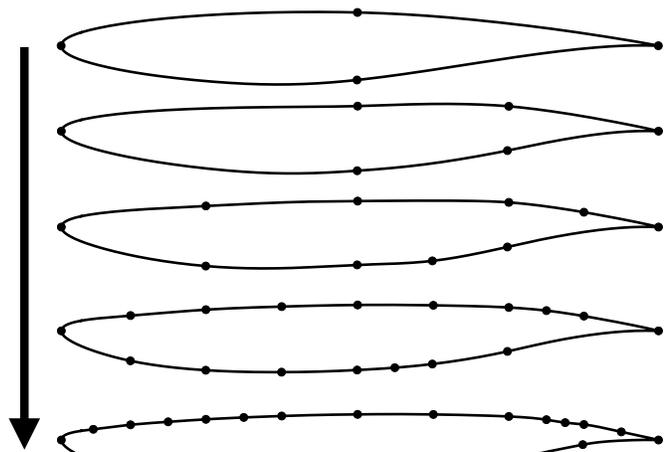
<sup>h</sup>Actual growth rates are also affected by regularity rules.



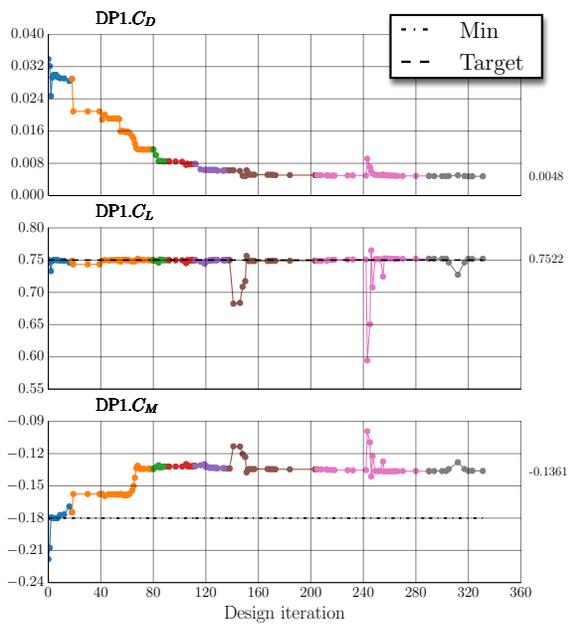
**Figure 13:** Transonic airfoil optimization results for the adaptive parameterization approach. *Top:* Optimized airfoil to scale. *Middle:* Final airfoil from three intermediate search spaces (4-DV, 15-DV, 26-DV), showing 26-DV adapted parameterization. *Bottom:* Corresponding pressure profiles for the Mach 0.79 design point.

counts to 66 counts. Similarly, at Mach 0.82, the drag is reduced from about 600 counts to 276 counts. Figure 13 also shows the non-uniformly refined 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 14.

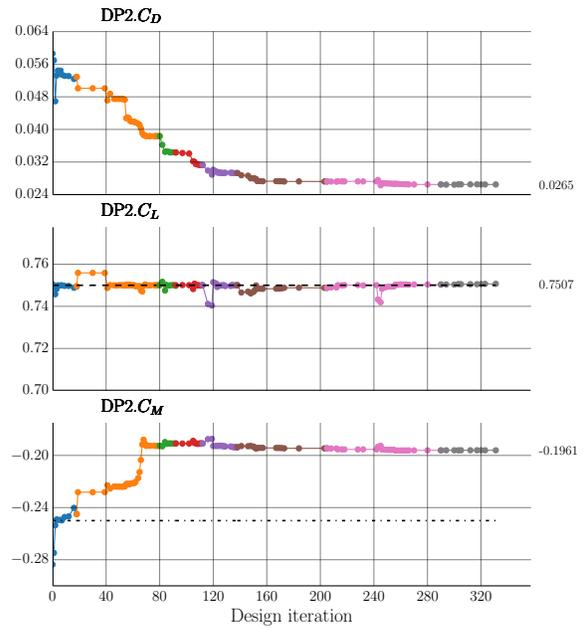
Figure 15 shows the evolution of the lift, drag and pitching moment functionals. The constraints are rapidly met and held throughout the optimization, while the drag is gradually reduced. The thickness constraints are satisfied at every design. The area and trailing edge constraints are all active but 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 we have not yet reached the continuous limit of design improvement.



**Figure 14:** *Transonic airfoil:* History of adaptive refinement, showing best airfoils attained under the first several parameterizations.

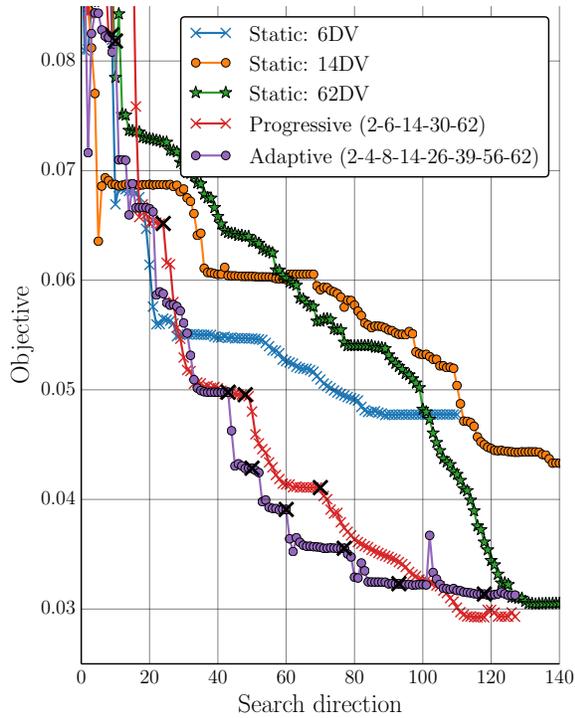


(a) Mach 0.79 design point

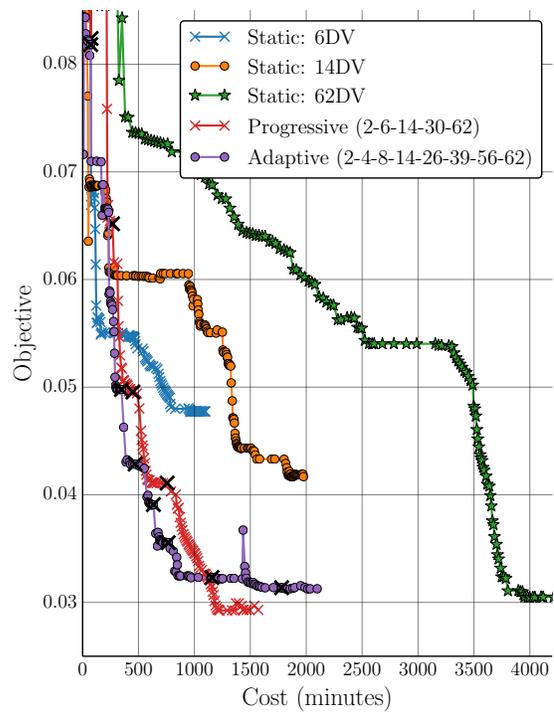


(b) Mach 0.82 design point

**Figure 15:** *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.

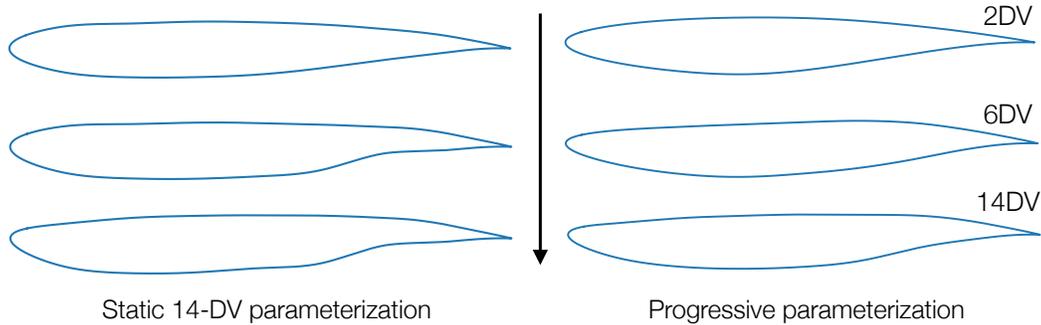


(a) Search direction history



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

**Figure 16:** *Transonic airfoil:* Convergence of combined drag value ( $C_{D1} + C_{D2}$ ) (ignoring satisfaction of constraints) for each parameterization method.  $\times$ -marks denote search space refinements.



**Figure 17:** *Transonic airfoil:* shapes encountered during optimization under a progressive parameterization (*right*) are consistently much smoother than airfoils encountered under a static parameterization (*left*).

### 5. Comparison to Static Parameterizations

The left frame of Figure 16 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.

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 our 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,<sup>i</sup> 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 16, which shows objective improvement vs. an estimate of wall-clock time<sup>j</sup>. 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,<sup>22, 29</sup> 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 16 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 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 17 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.

## B. Geometric Shape Matching Benchmark

In this example, we demonstrate the ability of our system to *discover* the parameters necessary to solve an optimization problem. We also use this example to evaluate the different indicators developed in Section §V.A and to assess the performance of our search procedure developed in §V.B.

The problem involves geometric shape-matching on a typical transport wing. In shape matching, we examine the convergence from a baseline geometry to an attainable target shape. The objective function

<sup>i</sup>We performed cold restarts when the static parameterizations stalled, to verify that no further progress could be made.

<sup>j</sup>Rough timings on 24 Intel Haswell cores

aims to minimize the deviation between the current shape and the target shape  $\mathbf{S}^*$  in a least-squares sense:

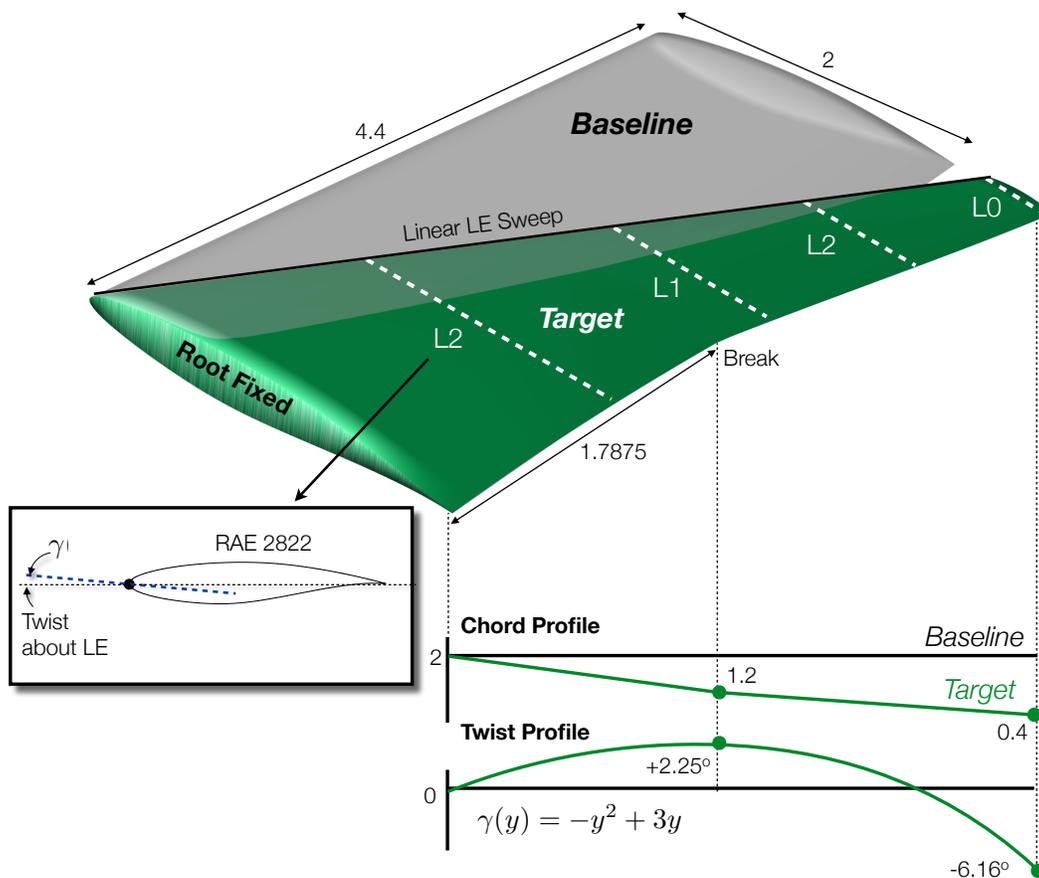
$$\mathcal{J} \equiv \|\mathbf{S} - \mathbf{S}^*\|^2 = \sum_{i=1}^{N_{verts}} \|\mathbf{v}_i - \mathbf{v}_i^*\|^2 \quad (13)$$

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 not only know the optimal shape, but we also know the *minimal* shape parameterization that can achieve that design. The goal of this exercise is to efficiently discover a parameterization that enables the optimizer to exactly match the target shape.

### 1. Initial Parameterization and Target

Figure 18 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 18. 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 illustrated in Figure 3, 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 18:** Baseline and target planform profiles. Initial shape control station is labeled L0, after which L1 is added, followed by the L2 stations, etc.

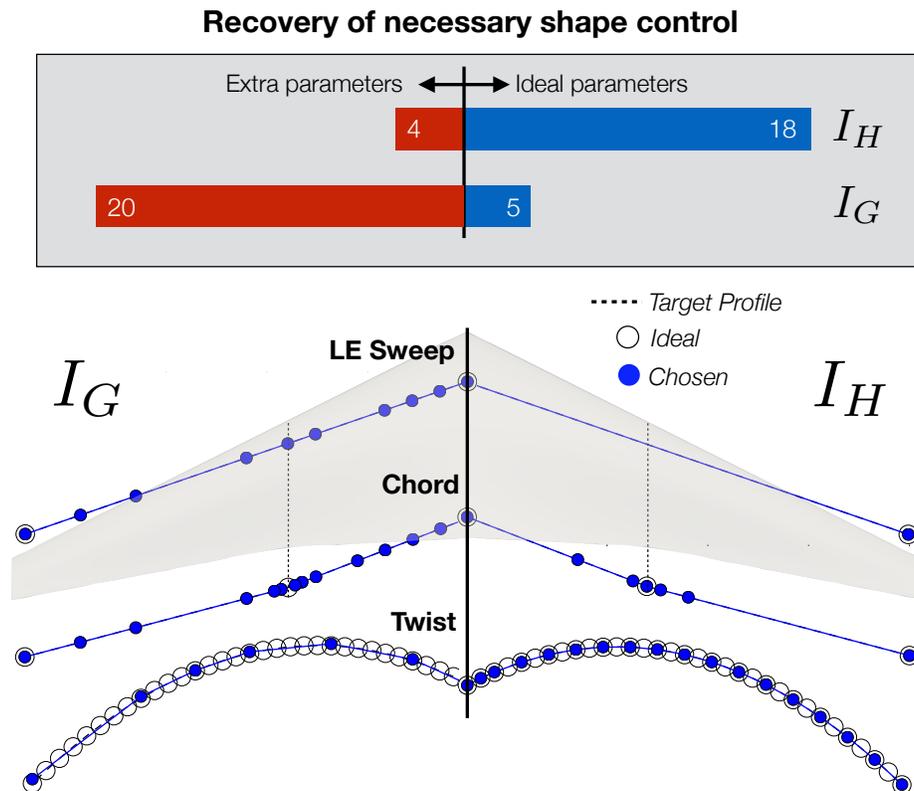
The target shape is unattainable under this initial parameterization. Only through sufficient and correct search space refinement can the target be reached. The problem is constructed such that we know in advance the necessary and sufficient refinement pattern, i.e. the one that will allow the closest 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, progressively finer twist control should be added to approximate the quadratic twist profile with piecewise linear segments. The initial 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 our system can recover or approximate this “ideal” parameterization.

## 2. Test 1: Indicator Comparison

Our 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 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.<sup>k</sup>

Figure 19 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 stipulated in Section §IV.B, 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

<sup>k</sup>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 19: 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).

span, optimally clamping down the error between the quadratic profile and the linear segments. It has also begun to add the next nested level of control near the root.

Now compare the left half of Figure 19, 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.

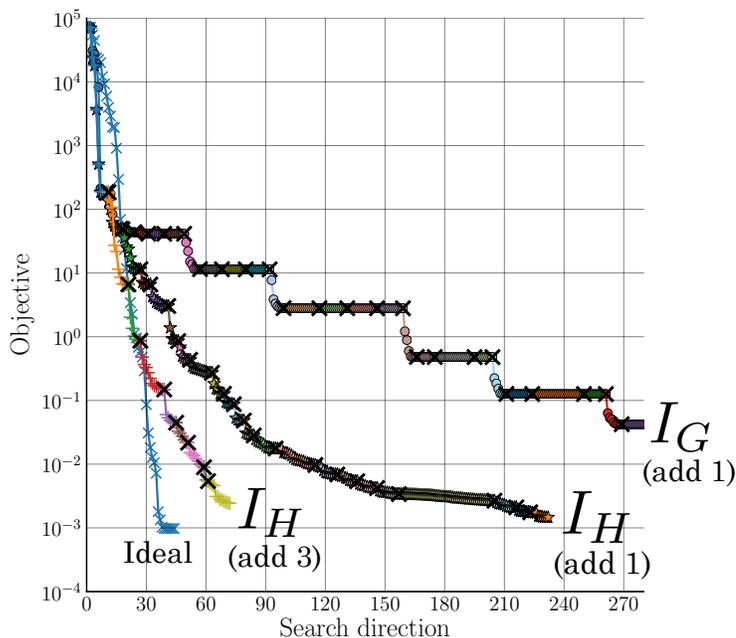
The reason for the relatively poor performance of  $I_G$  is that the chord and sweep objective gradients had much larger magnitudes than the twist gradients, even when very close to their optimal values. Thus chord and sweep were favored, even though they offered only extremely short-term potential. This is a concrete example of the idea illustrated notionally by the red and blue curves in Figure 6.  $I_H$ , by contrast, 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 20 compares the objective convergence under the two indicators (labeled “ $I_G$  (add 1)” and “ $I_H$  (add 1)”). The gradient indicator frequently adds parameters with almost no potential, leading it to stall for several adaptation cycles. Nevertheless, it still managed to reduce 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 20, 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 we searched only one level deep in the parameter tree. As mentioned in Section §V.C, much higher growth rates should lead to much faster design improvement.

### 3. Test 2: Search Procedure Evaluation

As a second test, we try 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 evaluate all combinations, as this becomes prohibitive. Instead, we use our 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 21. Figure 20 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, we have accurately matched the shape using only 30 parameters.



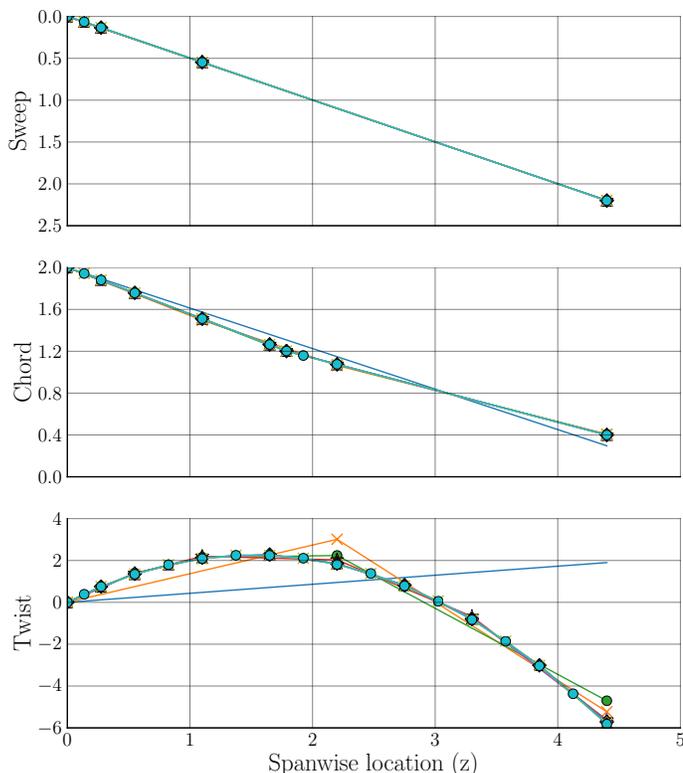
**Figure 20:** Shape-matching objective convergence for different indicators and search strategies. Solid blue line shows the “best possible” convergence, using the *a priori* known best possible 35-DV parameterization. Each color represents a different adaptively-refined parameterization and  $\times$ -marks denote search space refinements.

#### 4. Test 3: Immediate Discovery

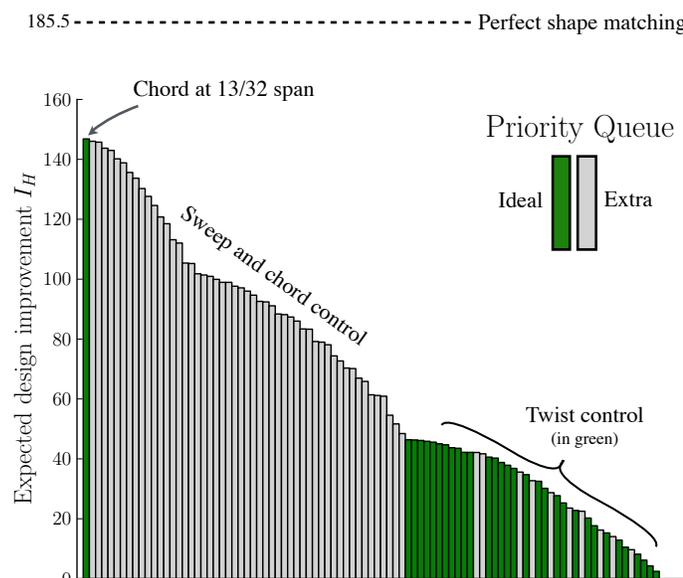
As a final experiment, we try to predict *all* of the necessary shape control, based only on information at the baseline design. In other words, there will be no intermediate optimizations. We immediately look five levels deep, and request the addition of 32 design variables at once, without any prior 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 22 shows the initial priority queue for the 96 candidates, which is formed by analyzing each candidate shape control element independently, using the Hessian indicator. 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.

Our 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.



**Figure 21: Test 2:** Final recovered planform distribution (search depth 2, add 3 parameters at a time, Hessian indicator).



**Figure 22: Test 3:** The priority queue after Phase I of the constructive algorithm (search depth of 5, adding 32 parameters at once, Hessian-based indicator). The 32 parameters that would best recover the target shape are highlighted in green. After adding the first parameter in the queue, all of the subsequent gray parameters (chord and sweep controllers) become redundant.

## VII. 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. We showed 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. We also showed that the optimization often achieves faster design improvement (as much as  $3\times$  in some cases) over using all the design variables up front. Additional important benefits of this approach include:

- **Automation:** By automating search space refinement, this approach greatly reduces user time, and also reduces dependence on designer expertise.
- **Completeness:** The full design space can be explored more thoroughly, as it is not restricted by the initial parameterization.
- **Feedback:** The refinement pattern conveys useful information about the design problem.

Our implementation is architected to work with arbitrary geometry modelers. Development is required to prepare an existing modeler for adaptive use. However, the computational acceleration and the amount of manual setup time eliminated from each optimization strongly justifies this expenditure. With modest tailoring, our system could also invoke different aerodynamic or multi-disciplinary design frameworks.

## VIII. Future Work

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 weak spots in the problem formulation. Many of these constraints, such as non-self-intersection, smoothness, or limits on excessive curvature, can in theory be codified, which would help regularize the optimization and likely lead to superior results.

The efficiency of our approach is highly dependent on the adaptation strategy, including the trigger, growth rate, indicator and search algorithm. All told, our implementation added only about 10 new parameters to tune the adaptation strategy. In the future, we hope to determine the degree to which we can robustly automate some of these choices. Investigations are also underway to examine whether any Hessian information (in a new candidate search space) can be approximated for aerodynamic objectives, which is expected to substantially improve the predictions of our adaptive system. 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 would be more effective, a question we are actively investigating. Finally, we also hope to demonstrate and evaluate the technique on more large-scale design problems, such as wing-body-nacelle integration or low-boom design.

## Acknowledgments

The authors are deeply indebted to Marian Nemeč for development of and support with the use of the design optimization framework used in this work. We gratefully acknowledge insightful discussions with David Rodriguez, Joshua Leffell and Susan Cliff, and with the manuscript reviewers, Tom Pulliam and Marco Ceze. Support for this work was generously provided by a 2.5-year NASA ARMD Seedling award.

## References

- <sup>1</sup>Beux, F. and Dervieux, A., “A Hierarchical approach for shape optimisation,” Research Report RR-1868, INRIA, 1993.
- <sup>2</sup>Kohli, H. S. and Carey, G. F., “Shape optimization using adaptive shape refinement,” *International Journal for Numerical Methods in Engineering*, Vol. 36, No. 14, 1993, pp. 2435–2451.
- <sup>3</sup>Desideri, J.-A. and Janka, A., “Hierarchical Parameterization for Multilevel Evolutionary Shape Optimization with Application to Aerodynamics,” *International Congress on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, 2003.

- <sup>4</sup>Majd, B. A. E., Duvigneau, R., and Désidéri, J.-A., “Aerodynamic Shape Optimization using a Full and Adaptive Multilevel Algorithm,” ERCOFTAC Design Optimization: Methods and Application, Gran Canaria, Canaria Island, Spain, April 2006.
- <sup>5</sup>Courty, F. and Dervieux, A., “Multilevel functional preconditioning for shape optimisation,” *International Journal of Computational Fluid Dynamics*, Vol. 20, No. 7, 2013/09/06 2006, pp. 481–490.
- <sup>6</sup>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.
- <sup>7</sup>Majd, B. A. E., Desideri, J.-A., and Duvigneau, R., “Multilevel Strategies for Parametric Shape Optimization in Aerodynamics,” *REMN*, 2008.
- <sup>8</sup>Chaigne, B. and Désidéri, J.-A., “Convergence of a Two-Level Ideal Algorithm for a Parametric Shape Optimization Model Problem,” Research Report 7068, INRIA, Sophia Antipolis, France, September 2009.
- <sup>9</sup>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.
- <sup>10</sup>Duvigneau, R., “Adaptive Parameterization using Free-Form Deformation for Aerodynamic Shape Optimization,” Tech. Rep. 5949, INRIA, 2006.
- <sup>11</sup>Martinelli, M. and Beux, F., “Multi-level gradient-based methods and parametrization in aerodynamic shape design,” *European Journal of Computational Mechanics*, Vol. 17, No. 1-2, 2008, pp. 169–197.
- <sup>12</sup>Désidéri, J.-A. and Dervieux, A., “Hierarchical Methods for Shape Optimization in Aerodynamics. I: Multilevel Algorithms for Parametric Shape Optimization,” *Introduction to Optimization and Multidisciplinary Design*, edited by J. Périaux and H. Deconinck, 2006-3, Von Karman Institute for Fluid Dynamics, March 2006.
- <sup>13</sup>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.
- <sup>14</sup>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.
- <sup>15</sup>Wu, H.-Y., Yang, S., Liu, F., and Tsai, H.-M., “Comparison of three geometric representations of airfoils for aerodynamic optimization,” *AIAA Paper 2003-4095*, Orlando, FL, June 2003.
- <sup>16</sup>Sherar, P. A., Thompson, C. P., Xu, B., and Zhong, B., “A Novel Shape Optimization Method using Knot Insertion Algorithm in B-spline and its Application to Transonic Airfoil Design,” *Scientific Research and Essays*, Vol. 6, No. 27, November 2011, pp. 5696–5707.
- <sup>17</sup>Han, X. and Zingg, D. W., “An Evolutionary Geometry Parametrization for Aerodynamic Shape Optimization,” *AIAA Paper 2011-3536*, Honolulu, HI, June 2011.
- <sup>18</sup>Han, X. and Zingg, D., “An adaptive geometry parametrization for aerodynamic shape optimization,” *Optimization and Engineering*, Vol. 15, No. 1, 2013, pp. 69–91.
- <sup>19</sup>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.
- <sup>20</sup>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.
- <sup>21</sup>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.
- <sup>22</sup>Nemec, M. and Aftosmis, M. J., “Output Error Estimates and Mesh Refinement in Aerodynamic Shape Optimization,” *AIAA Paper 2013-0865*, Grapevine, TX, January 2013.
- <sup>23</sup>Hicks, R. M. and Henne, P. A., “Wing Design by Numerical Optimization,” *J. Aircraft*, Vol. 15, No. 7, July 1978.
- <sup>24</sup>Kulfan, B. M., “Universal Parametric Geometry Representation Method,” *J. Aircraft*, Vol. 45, No. 1, January 2008, pp. 142–158.
- <sup>25</sup>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.
- <sup>26</sup>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.
- <sup>27</sup>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.
- <sup>28</sup>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.
- <sup>29</sup>Nemec, M. and Aftosmis, M. J., “Toward Automatic Verification of Goal-Oriented Flow Simulations,” Tech. Rep. TM-2014-218386, NASA, 2014.

## Appendix A. Static Shape Optimization Algorithm

Adjoint-based parametric shape optimization frameworks typically follow the iterative loop outlined in Function 4. A discrete tessellated surface  $\mathbf{S}$  is generated by a geometry modeler, based on the shape parameter values  $\mathbf{X}$ . The solution domain is then meshed and the PDE is solved (for our purposes, the fluid flow equations), enabling evaluation of the objective function  $\mathcal{J}$ . Next, the adjoint equations are solved, which allows rapid computation of the objective gradients  $\frac{\partial \mathcal{J}}{\partial \mathbf{X}}$  to each design variable. Finally a gradient-based optimizer determines an update to the design variables  $\mathbf{X}$ , and the loop is continued.

<p><b>Function 4:</b> <i>Optimize</i>(<math>\cdot</math>)  <b>Parametric Geometry Engine</b>  <b>PDE Solver Functions</b></p> <p><b>Input:</b> Shape deformation function <math>D</math> with initial design variable values <math>\mathbf{X}_0</math>, objective function <math>\mathcal{J}</math>, constraints <math>C_j</math>  <b>Result:</b> Optimized surface <math>\mathbf{S}</math>, adjoint solution <math>\psi</math>  <math>\mathbf{X} \leftarrow \mathbf{X}_0</math>  <b>repeat</b>            <math>\mathbf{S} \leftarrow \text{GenerateSurface}(D, \mathbf{X})</math>            <math>\mathbf{M} \leftarrow \text{DiscretizePDE}(\mathbf{S})</math>            <math>\mathbf{Q} \leftarrow \text{SolvePDE}(\mathbf{M})</math>            <math>\mathcal{J} \leftarrow \text{ComputeObjective}(\mathbf{Q}, \mathbf{S})</math>            <math>\psi \leftarrow \text{SolveAdjoint}(\mathbf{M}, \mathbf{Q})</math>            <b>foreach</b> <math>X_i</math> <b>in</b> <math>\mathbf{X}</math> <b>do</b>                <math>\frac{\partial \mathbf{S}}{\partial X_i} \leftarrow \text{ShapeDerivative}(D, X_i)</math>                <math>\frac{\partial \mathcal{J}}{\partial X_i} \leftarrow \text{ProjectGradient}(\psi, \frac{\partial \mathbf{S}}{\partial X_i})</math>            <b>end</b>            <math>\mathbf{X} \leftarrow \text{NextDesign}(\mathbf{X}, \frac{\partial \mathcal{J}}{\partial \mathbf{X}})</math>                   // Optimizer  <b>until</b> <math>\text{Stop}(\mathcal{J}, \frac{\partial \mathcal{J}}{\partial \mathbf{X}})</math>  <b>return</b> <math>\mathbf{S}, \psi</math></p>
---