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 control is adaptively parameterized. Starting from a coarse set of design variables, a sequence of higher-dimensional nested search spaces is automatically generated. Refinement can be either uniform or adaptive, in which case only the most important shape control is added. The relative importance of candidate design variables is determined by comparing objective and constraint gradients, computed at low cost via adjoint solutions. A search procedure for finding an effective ensemble of shape parameters is also given. We first demonstrate this system on a multipoint drag miminization problem in 2D with many constraints, showing that an adaptive parameterization approach consistently achieves smoother, more robust, and faster design improvement than fixed parameterizations. We also establish a 3D shape-matching benchmark, where we demonstrate that our approach automatically discovers the necessary parameters to match a target shape. By largely automating shape parameterization, this work also aims to remove a time-consuming aspect of shape optimization.

Nomenclature

$\mathcal{S}$	Continuous surface	$\mathcal{C}$	Constraint functional
$\mathbf{S}$	Discrete tesselated surface	w	Window width
$\mathbf{C}$	Shape control	r	Reduction factor for trigger
$\mathbf{C}_c$	Candidate shape control	$\mathbf{g}$	Vector of growth rates in # of parameters
P	Function parameterizing shape deformation	ψ	Adjoint solution
D	Deformation function	I	Importance indicator
$\mathbf{X}$	Vector of design variable values	KKT	Karush-Kuhn-Tucker conditions
$\mathcal{J}$	Objective functional	DV	Design variable

I. Introduction

AUTOMATED meshing and flow 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 generated to meet the particular demands of a given problem. The primary goal is to radically reduce user setup time and to increase robustness. Additionally, we aim to avoid the most common pitfalls of a manually constructed shape design space: Excessive numbers of design variables (which leads to inefficient navigation), shape control that permits only inadequate exploration of the design space, and unintentional bias towards familiar designs.

Figure 1 illustrates our basic approach. Starting from a low-dimensional search space, higher-resolution shape control is added as the design evolves. In the limit of uniform refinement, this process approaches the continuous shape optimization problem of the full design space. Additionally, it encourages rapid improvement up front, introducing complexity only later if necessary and if resources permit. Although automated, this progressive approach also reflects the natural design process:

- Large-scale changes are made early, while detailed adjustments happen last.

*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.

- The optimization ultimately explores the full design space, not just a restricted subspace defined by a static parameterization.
- The important design variables do not need to be predicted before design begins. Rather the important shape parameters are discovered as a natural consequence of optimization, and are clearly visible to the designer in the emergent pattern of shape parameters.

In addition, a growing body of evidence indicates that substantial design acceleration can be achieved by using variable shape control,[1–6] an observation we corroborate in this study.

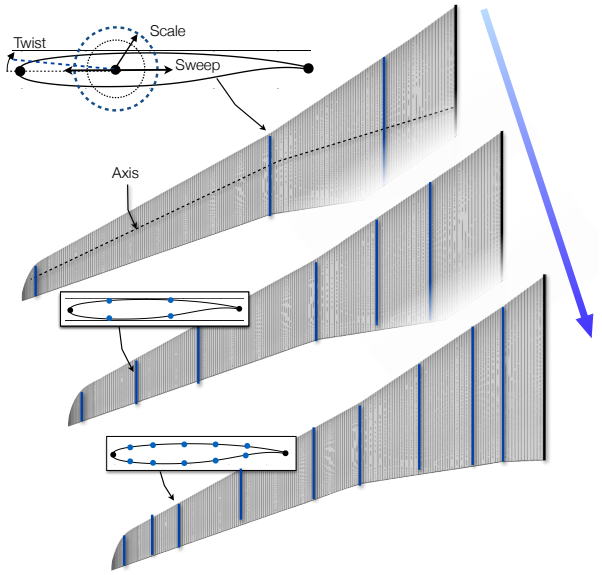


Figure 1: Search space refinement for wing design. Airfoil control is refined independently on each section.

Progressive shape parameterization for aerodynamic optimization was first proposed in 1993 both by Beux and Dervieux[7] and by Kohli and Carey.[8] A series of subsequent papers from INRIA demonstrated that substantial design acceleration can be achieved with nested parameterizations.[9–15] Their approach is well-documented in a detailed report by Duvigneau,[1] 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 fixed fine parameterization. They further argue that optimization is an inherently "anti-smoothing" process, and that design space sequencing is analogous to a preconditioner for the entire optimization process. Their results generally suggest that redistribution of existing design variables is at least as effective as enriching the parameterization. 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.*,[3] who performed genetic optimization in adaptively refined design spaces. Because they used a gradient-free approach, they by evolved 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.[4] 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*.

The most functionally similar approach to ours is that of Han and Zingg,[2,16] who most notably introduced an adaptation criterion based on the objective gradients to the candidate design variables, making the adaptation sensitive to the particular design problem. In this work we develop two new problem-specific adaptation criteria based on constraint gradients and Hessian information, 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.

II. Optimization with Progressive Shape Control

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 irrelevant 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 2, while finer parameterizations can reach superior designs, they take longer to converge, even with gradient-based optimizers, whose running time scales roughly as $\mathcal{O}(N_{DV})$. In practice, a designer will typically perform an optimization and then manually refine the parameterization if necessary — a time-consuming task.

Using a *progressive* parameterization approach, we start in the 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.

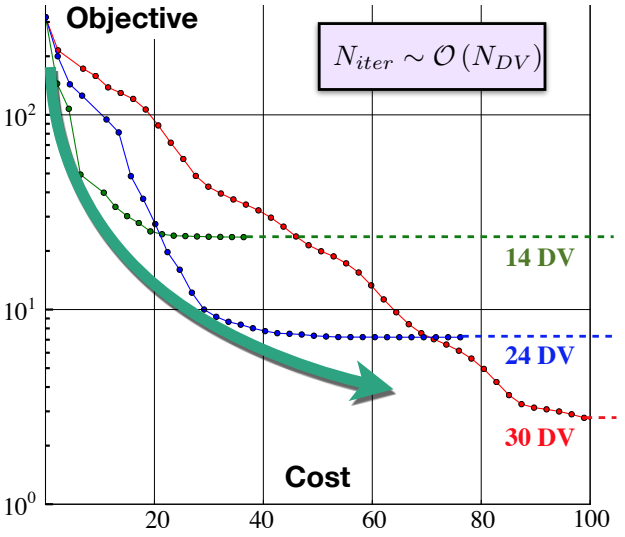


Figure 2: BFGS-style optimization converges in $\mathcal{O}(N_{DV})$ search directions. A progressive parameterization can follow the "inside track", making rapid gains early, while still approaching the continuous optimal shape.

A. Optimization Formulation

The aerodynamic shape optimization problem we consider consists of finding a shape $\mathcal{S}$ that minimizes an objective function

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

where $\mathcal{J}$ is a scalar functional that is evaluated after solving for the flow variables $\mathbf{Q}$. There may also be design constraints of the form $a \leq \mathcal{C}_j \leq b$. $\mathcal{J}$ and $\mathcal{C}_j$ typically involve performance metrics such as lift, drag, range, stability margins, maneuver loads, or operating costs. They may also include more specialized concerns such as reducing sonic boom ground signatures or environmental impact.

B. 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 typically parameterized. A shape parameterization technique, P, is a map from a vector $\mathbf{C}$ describing the shape control to a search space (a subset of the full design space), consisting of a deformation function, D, and a set of shape parameters $\mathbf{X}$. This deformation function takes the shape parameters and generates a new surface $\mathcal{S}$:

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

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

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. The shape parameters $\mathbf{X}$, or a subset thereof, serve as the design variables for optimization. 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.

EXAMPLE: WING PLANFORM DESIGN

Consider the simple wing parameterization scheme, P_{wing}, illustrated in Figure 3. Twist, sweep and chord are interpolated between the spanwise stations. Here, the shape control $\mathbf{C}$ is the spanwise coordinates of the control stations (blue lines), indicating *where* twist, sweep or chord can be manipulated. P_{wing} interprets this terse definition, expanding it into precise geometric descriptions of each deformation mode, encoded by the deformation function D, which takes the twist, sweep and chord values $\mathbf{X}$ and generates a new surface.

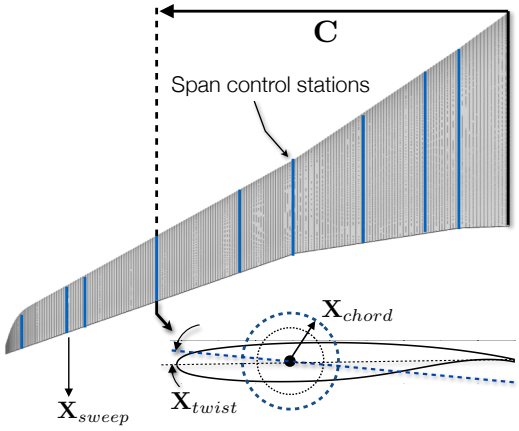


Figure 3: Wing planform parameterization

It is important to draw the distinction between the concept of the shape *control* $\mathbf{C}$ and the concept of shape *parameters* $\mathbf{X}$. In terms of our work, each optimization level involves a search in the space of values $\mathbf{X}$. When we transition to a finer search space, we are modifying the shape control $\mathbf{C}$. 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.

C. Progressive Parameterization

In standard shape optimization approaches, the shape control $\mathbf{C}_\times$ is pre-determined by the designer. This induces a static search space $D_\times$, 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.

This approach aims to automate the generation of nested search spaces of increasing resolution, with little manual labor. However, the designer is still responsible for establishing a basic framework for this process. Specifically, the designer selects a type of shape control (e.g. twist vs. airfoil deformation), specifies an initial coarse parameterization, and indicates how the shape control may be refined. We do not consider the much less tractable problem of determining the best such shape control framework. All refinement is performed within the framework defined by the initial shape control and the refinement mechanics.

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 more 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 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 parameterization to be viewed in terms of binary trees, with deeper levels corresponding to higher resolution shape control. Later, we show how this approach extends to wing parameterization.

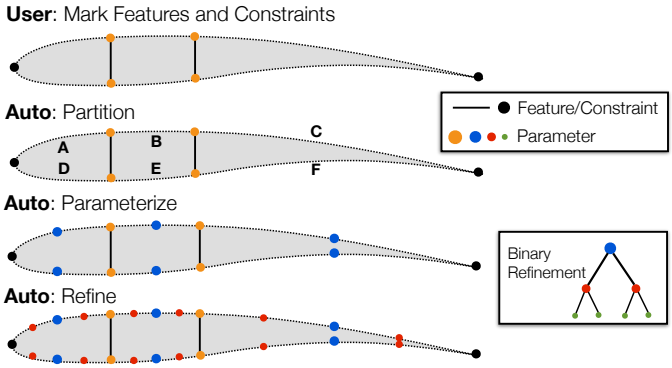


Figure 4: Progressive parameterization with discrete, hierarchical shape control refinement

III. Implementation

The design loop now consists of an alternating sequence of optimization within the current search space followed by a parameterization refinement, as given in Algorithm A. The function *Optimize*$(\cdot)$ represents a standard shape optimization package, which for reference is outlined in Appendix A. *Parameterize*$(\cdot)$ is the modeler-dependent implementation of Equation 2, which generates a search space 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, must ensure that the new shape is identical to the final previous shape. Unlike in static optimization approaches, where this is a user-driven pre-processing step, here it is automated.

Algorithm A also introduces three new functions that govern the refinement strategy. The *Trigger*$(\cdot)$ monitors the optimization progress to determine *when* to refine the shape control. Next, the modeler-dependent *GetCandidateShapeControl*$(\cdot)$ generates a list of possible locations $\mathbf{C}_c$ *where* the shape control may be refined. For example, in the wing planform example, the possible refinement locations might be new stations at the midpoints between existing ones. 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 to add[a], based on objective gradients from a linearization about the current design. We will discuss each of these functions in detail shortly. The ultimate convergence criterion is based on convergence of the objective as the discrete shape control $\mathbf{C}$ approaches continuous shape control (direct optimization of $\mathbf{S}$ or even $\mathcal{S}$).

Algorithm A requires integration of three basic components: (1) a geometry modeler, (2) a gradient-based 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.

Algorithm A: Optimization with Adaptive Shape Control
Parametric Geometry Modeler
Refinement Strategy (modeler independent)

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
  $D, \mathbf{X}_0 \longleftarrow \mathit{Parameterize}(\mathbf{S}, \mathbf{C})$
  $\mathbf{S} \longleftarrow \mathit{Optimize}(D, \mathbf{X}_0, \mathcal{J}, \mathcal{C}_j)$ **until** $\mathit{Trigger}(\cdot)$
  $\mathbf{C}_c \longleftarrow \mathit{GetCandidateShapeControl}(\mathbf{C})$
  if adaptive **then**
    $\mathbf{C} \longleftarrow \mathit{AdaptShapeControl}(\mathbf{C}, \mathbf{C}_c, \psi, \mathbf{S}, \mathbf{g})$
  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.* $\mathbf{C}$

A. Shape Optimization Framework

For the function *Optimize*$(\cdot)$ in Algorithm A, we use a gradient-based aerodynamic shape design framework[17] that uses an embedded-boundary Cartesian mesh method for inviscid flow solutions. Objective and constraint gradients are computed using an adjoint formulation. We leverage this same adjoint solution to prioritize candidate design variables when refining the search space. Optimization can be handled with any black-box gradient-based optimizer; for this study an SQP optimizer was used,[b] enabling proper treatment of constraints.

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.[19] 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 V. 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.[17]

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 shape control). By contrast, most constructive modelers (with a few notable exceptions[2,12]) require an approximate re-fitting procedure when re-parameterizing, 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 at these transfers.

We view each parameterization as a binary tree, restricting refinement to the midpoints between existing parameters or stations, although we can search several levels deep from the current parameterization. Additionally, we prohibit large discrepancies between the refinement depth of adjacent regions on the surface. This is essentially a parameter smoothing step, which was found to be important for robustness in certain cases.

[a] Although we do not consider it in this work, *removing* design parameters that are no longer useful is also possibility.
[b] Stanford's SNOPT optimization package,[18] version 7.

Each technique supports theoretically 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 it well out of the resolution where it could begin to spuriously take advantage of surface and flow mesh discretization effects.

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 a simple shape-matching 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[1] and optimization with progressively refined PDE meshes.[20] 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.

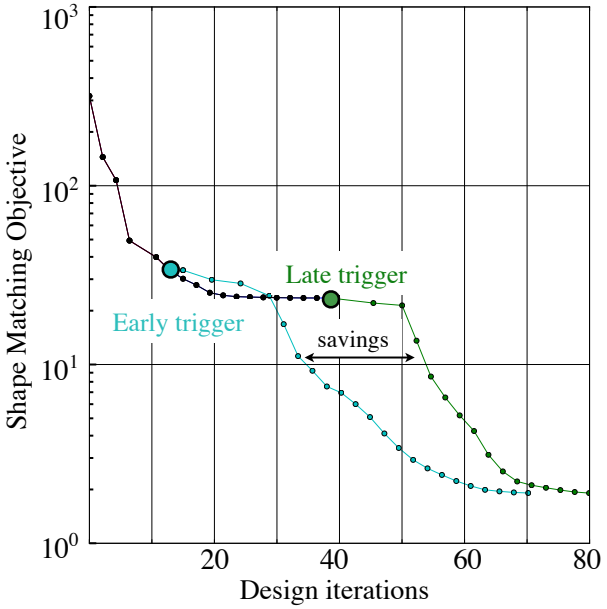


Figure 5: Late triggering leads to slow design improvement. Here and early transition from a 14-DV search space to a 30-DV search space yields much faster results.

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 frequently delayed refinement, substantially slowing progress. Additionally, as the magnitude of the gradients are problem- and scaling-dependent, it is difficult to establish efficient cutoffs without prior experience with a particular problem.

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 $\mathcal{O}\left(10^{-2}\right)$ while a functional based on operating range may be $\mathcal{O}\left(10^{5}\right)$.

The slope is evaluated at major search iterations, which is monotonically decreasing.[c] 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.

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

For relatively simple design problems, such as unconstrained drag mininimization or geometric shape-matching, we observe that it is best to use a fairly aggressive trigger (as high as $r = 0.25$, with window $w = 1$). For more complex problems, especially ones with many competing constraints, we find that it is more effective to allow deeper convergence on the coarser parameterizations before proceeding. An aggressive trigger can introduce shape parameters earlier than strictly necessary. However, under an adjoint formulation, the cost of computing gradients is quite low. The cost associated with having some extra design variables is negligible compared to the cost of over-converging in a coarse search space.[d]

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. Systematic discernment between these two cases is difficult without prior knowledge that a superior design is attainable. As it is infeasible to encode sophisticated problem-dependent logic in a simple trigger, 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.

IV. Adaptive Shape Control Refinement

At this point, we have described an automated, nested but uniform (non-adaptive) shape control strategy. 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 N_{DV} than under a finite difference approach, there are still costs that scale with N_{DV}, including computation of geometric surface derivatives and subsequent gradient projections. Minimizing the number of design variables is generally highly desirable.

To enable selective adaptation, we develop a systematic method for choosing an effective combination of refinement locations from among the myriad candidates. To do so, we compute various indicator metrics for each candidate parameterization, and select the one that appears the best, when ranked by these metrics. The expectation is not to find the truly ideal parameterization, but to substantially improve navigational efficiency, while introducing fewer dimensions than uniform refinement.

A. Effectiveness Indicator

To predict the effectiveness of the various possible shape control refinements we use problem-aware metrics, namely the local objective and constraint gradients to the candidate design variables, and possibly also an approximation of the local Hessian matrix. Like any local metric in a nonlinear design landscape[e], this is only an approximation, but in the absence of *a priori* information, it is the best information available.

1. Gradient-Maximizing Indicator

In a convex, properly-scaled problem, and in the absence of active constraints or design variable bounds, a higher objective gradient generally indicates that a parameter is more effective. This suggests taking the effectiveness indicator I to be a norm of the vector of objective gradients with respect to the *candidate* design variables $\mathbf{X}_c$:

$$I_G(\mathbf{C}_c) = \left\| \frac{\partial \mathcal{J}}{\partial \mathbf{X}_c} \right\| \tag{4}$$

Each gradient $\frac{\partial \mathcal{J}}{\partial \mathbf{X}_c}$ gives a local forecast of the rate at which that individual candidate parameter will help improve the design. Consistent with the fact that the adjoint is a linearization about the local state, our experiments show that higher gradients are strongly correlated with short-term design improvements. After a few design iterations, the accuracy of that correlation depends on the degree of nonlinearity and the scaling of the problem.

[d]However, under a finite-difference optimization framework (i.e. without the adjoint), where the cost of each extra gradient is two flow solutions, allowing more convergence on fewer design variables could prove more efficient.

[e]Stemming from nonlinearities in the geometry, shape deformation, flow mesh discretization, and the flow equations.

Function 2 shows how I_G is assembled. Importantly, the objective gradients have modest additional cost. During optimization, the adjoint is used to compute gradients with respect to existing shape design variables. However, 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 new candidate design variables. This 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 launching point for the next search space. This is generally true for all discrete modelers.

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

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

$D, \mathbf{X} \longleftarrow Parameterize(\mathbf{S}, \mathbf{C})$
foreach X_i *in* $\mathbf{X}$ **do**
  $\frac{\partial \mathbf{S}}{\partial X_i} \longleftarrow ShapeDerivative(D, X_i)$
  $\frac{\partial \mathcal{J}}{\partial X_i} \longleftarrow ProjectGradient(\psi, \frac{\partial \mathbf{S}}{\partial X_i})$
end
$I \longleftarrow \sqrt{\sum_i (\frac{\partial \mathcal{J}}{\partial X_i})^2}$

2. *Orthogonality of Objective and Constraints*

If there are design constraints, it is desirable to prioritize parameterizations that have high objective gradients *in a feasible direction*. 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 aerodynamic constraints such as lift or pitching moment, or to bulk geometric constraints such as on wing volume.

To handle general constraints, we propose an approach based on the KKT conditions for optimality of a constrained problem. 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. Assuming the final design in the current search space satisfies the constraints, the KKT conditions for a candidate re-parameterization simplify to

$$\frac{\partial \mathcal{J}}{\partial \mathbf{X}_c} = \sum_{i=1}^{n_\mathcal{C}} \lambda_i \frac{\partial \mathcal{C}_i}{\partial \mathbf{X}_c} \tag{5}$$

where the Lagrange multipliers satisfy $\lambda_i > 0$ or $\lambda_i < 0$ for constraints at their maximum and minimum bounds respectively, or $\lambda_i \neq 0$ for equality constraints. This is a non-square system of equations with dimensions $(n_{DV} \times n_\mathcal{C})$, which is usually highly overdetermined. At the final design of the current search space, the system should be nearly satisfied. Adding new shape parameters, however, introduces new equations, which should make it no longer satisfied. Solving Equation 5 using a bounded least-squares solver yields a set of best-fit λ_i. The least-squares residuals of this fit indicate distance from optimality and provide a convenient indicator:

$$I_{KKT}(\mathbf{C}_c) = \left\| \frac{\partial \mathcal{J}}{\partial \mathbf{X}_c} - \sum_{i=1}^{n_\mathcal{C}} \lambda_i \frac{\partial \mathcal{C}_i}{\partial \mathbf{X}_c} \right\| \tag{6}$$

Under this indicator, high objective gradients alone are insufficient. Roughly speaking, it prioritizes parameterizations where the objective gradients as orthogonal as possible to a linear transformation of the constraint gradients.

Note that in the absence of constraints, Equation 6 simplifies to Equation 4. In our experiments I_{KKT} and I_G typically result in very similar rankings, but in some cases, I_{KKT} may avoid adding ineffectual parameters. Like the objective gradients, the constraint gradients, $\frac{\partial \mathcal{C}_i}{\partial \mathbf{X}}$, can be readily computed by reusing the existing constraint adjoint solution(s) by a process similar to Function 2.

3. *Maximal Design Improvement (Hessian) Indicator*

Optimization approaches using approximations of the Hessian typically generate superior search directions to a steepest-descent approach. Similarly, we can surmise that with an approximation of the Hessian of the *candidate* search space, we can favor shape parameters that have longer-term usefulness than simply those with the highest gradients.

Consider Figure 6, which illustrates the local quadratic fit of 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 a Hessian approximation $\frac{\partial^2 \mathcal{J}}{\partial \mathbf{X}_c^2}$. 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. This leads to a very natural indicator

$$I_H(\mathbf{C}_c) \equiv -\Delta\mathcal{J}_{exp}(\mathbf{C}_c) = \frac{1}{2}\frac{\partial \mathcal{J}}{\partial \mathbf{X}_c}^T \left(\frac{\partial^2 \mathcal{J}}{\partial \mathbf{X}_c^2}\right)^{-1} \frac{\partial \mathcal{J}}{\partial \mathbf{X}_c} \quad (7)$$

which prioritizes search spaces that have high capacity for design improvement.

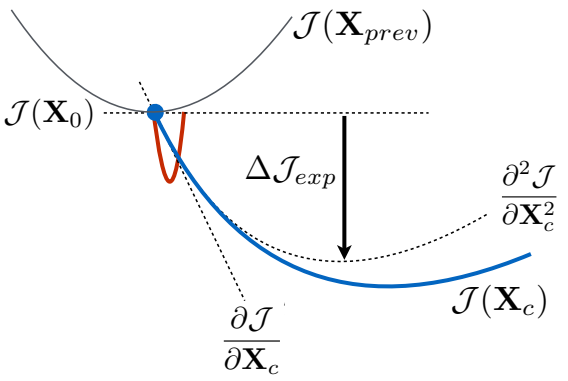


Figure 6: Local first- and second-order fits (dotted lines) of a candidate search space's actual behavior (blue). With second-derivative information, the expected improvement $\mathcal{J}_{exp}$ can be estimated. A second candidate search space with higher gradients (red) may actually offer less potential design improvement if its second derivatives are also high.

This indicator can be expected to perform exceptionally well at poorly-scaled problems, which are common in aerodynamic design. For the analytical problem of geometric shape-matching, we will show that the Hessian indicator I_H is radically superior to the gradient indicator I_G. Unfortunately, for aerodynamic problems, no estimate of the Hessian for the candidate design space is currently readily available, without the prohibitive cost of $2N_{DV}$ finite-differenced flow and adjoint solutions. This is a target for future work.

B. Ranking Candidate Refinements

We now present a search algorithm for finding an effective parameterization. Recall from Equations 4, 6 and 7 that the indicator is defined for each candidate ensemble of parameters $\mathbf{C}_c$, not simply for each individual parameter. This is a critical point, and can have important consequences for efficiency. In general, the deformation mode shape (and therefore the effectiveness) 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.

For certain special types of deformers (notably, Hicks-Henne bump functions[21]) we can make a simplification. If each deformation mode, described by $\frac{\partial \mathbf{S}}{\partial \mathbf{X}}$, is a function of only one element of the shape control $\mathbf{C}$, then we can consider the effect of each parameter in isolation, which greatly reduces the expense of ranking the parameters. Unfortunately, such deformers are the exception rather than the rule. 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.

An important consequence of this is that multiple similar shape control candidates may have "redundant potential", in the sense that adding any one is useful, but adding a second would not help. Thus, when searching for an effective shape control refinement, it is generally important to examine *ensembles* of parameters.

1. *Search Algorithm*

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 (here indicator evaluations).

For this work we developed a "constructive" search procedure, illustrated in Figure 7. 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 IV.A. In the second phase, we make N_{add} passes over the priority queue, reanalyzing only a sliding window, w, of the top few

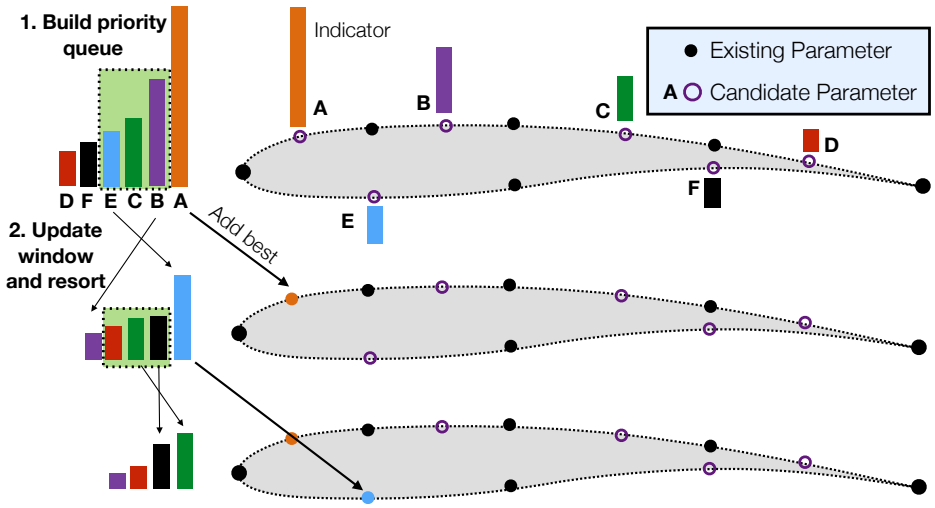


Figure 7: Constructive search algorithm for refining the shape parameterization.

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 $\mathcal{O}(N_{cand})$.[f]

This constructive procedure is most effective when the initial priority queue remains a fairly accurate ranking throughout the search. For many problems this is strongly true, and we observe that the procedure often returns the same result as an exhaustive search, but at a fraction of the cost. However, in cases with high "redundancy" among the candidate shape parameters (an example of which is given in Section V), it can yield far less optimal results. Alternately, if the initial priority queue is perfectly trustworthy (as in the special case of Hicks-Henne bump functions or other linear parameterizations), one can use a window size of $w = 0$, which is equivalent to immediately accepting the top N_{add} members of the queue.

The running time of the search depends on the speed of the geometry modeler and gradient projection tools, which are invoked frequently, and on the number of candidates being considered, and on the window size. In our environment, we observe highly practical running times, with cost usually equivalent to no more than a few design iterations. Naturally, this involves a tradeoff between spending longer to find a more efficient search space vs. immediately making design progress, but in a suboptimal search space.

Function 3: *AdaptShapeControl*$(\cdot)$
(Constructive search algorithm)

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

$N_{add} \longleftarrow int(len(\mathbf{C}) \cdot \mathbf{g}[i])$
$queue \longleftarrow \emptyset$
foreach P_{cand} *in* $\mathbf{C}_c$ **do**
  $I \longleftarrow ComputeIndicator(\mathbf{S}, \mathbf{C} \cup P_{cand}, \psi_i)$
  $queue.Add(P_{cand}, priority = I)$
end
for i=1..N_{add} **do**
  foreach P_{cand} *in* $queue.Best(w)$ **do**
    $I \longleftarrow ComputeIndicator(\mathbf{S}, \mathbf{C} \cup P_{cand}, \psi_i)$
    $queue.Update(P_{cand}, priority = I)$
  end
  $P_{best} \longleftarrow queue.pop()$
  $\mathbf{C} \longleftarrow \mathbf{C} \cup P_{best}$
end

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

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 new design variables introduced) will quickly stagnate, requiring additional shape control adaptation. Likewise, with too many design variables, navigation is slow. The growth rate can have an important impact on efficiency.

Consider manual specification of a relative growth rate (e.g. "increase the number of design variables by 50%"). Figure 8 compares the performance of various growth factors from $1.25 - 2\times$ 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.

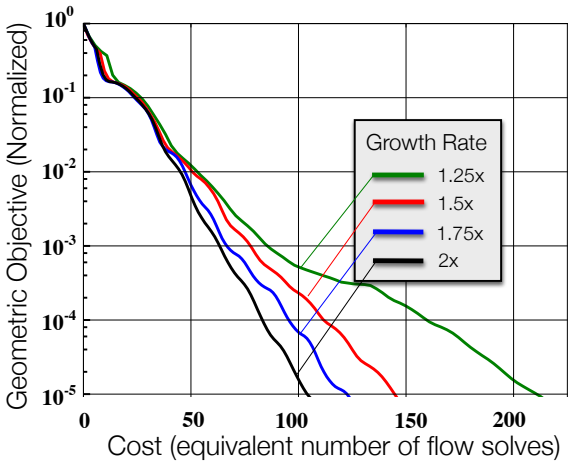


Figure 8: Performance of different growth rates on a geometric shape-matching objective. Each curve shows mean behavior over 10 randomized trials (trigger $r = 0.25$).

An additional consideration is how many shape parameters to start with. As we show in the first example, 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.

V. Results

In a concurrently published applications paper,[22] we provide results for solving four benchmarks posed by the AIAA Aerodynamic Design Optimization Discussion Group. The benchmarks include two airfoil design problems, a twist optimization for minimum induced drag, and a transonic wing design case. In these benchmarks, we extensively demonstrate our progressive and adaptive shape parameterization approach on practical design problems. We also discuss the importance of controlling discretization error in achieving robust design improvement with progressive parameterization.

In this section, we demonstrate the approach on two additional cases that are deliberately challenging for an adaptive approach. 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 involving geometric shape-matching that is particularly challenging for 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 both smooths the design trajectory and accelerates the optimization.

1. Problem Statement

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 γ at the trailing edge (see Figure 9) to prevent excessive cambering that would result in poor viscous performance. We also specify a minimum and maximum geometric closing angle ϕ 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

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

Figure 9: Geometric constraints at the trailing edge

where $(\mathbf{V})$ 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.

2. Geometry and Parameterization

The baseline geometry is a unit-chord RAE 2822 airfoil, shown in Figure 10, which 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.[15,23–25] We choose the cubic basis function $\phi = r^3$, primarily because it requires no local tuning parameters, making it more amenable to automation.

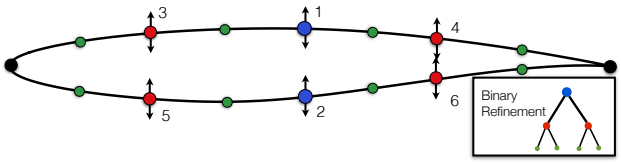


Figure 10: Baseline geometry and first three levels of shape control (2-DV, 6-DV and 14-DV)

Shape control refinement is binary, with adjacency and midpoints defined in terms of arclength along the curve. In the language of Section II.B, 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 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. In other words, the two progressive approaches will ultimately arrive at the static 62-DV search space. This refinement limit prevents the shape control from becoming unreasonably closely spaced.

3. Adaptive Strategy

The trigger for both progressive 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 to avoid early triggering while the constraints are being driven to satisfaction. Subsequently, we reduced the window to $w = 2$ for efficiency. For the adaptive approach, we used a target growth rate[g] of 1.75×, and used the constructive search algorithm (Function 7), with $w = 3$. As there are many constraints in this problem, we used the KKT-based indicator I_{KKT} (Equation 6) to rank candidate refinements.

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

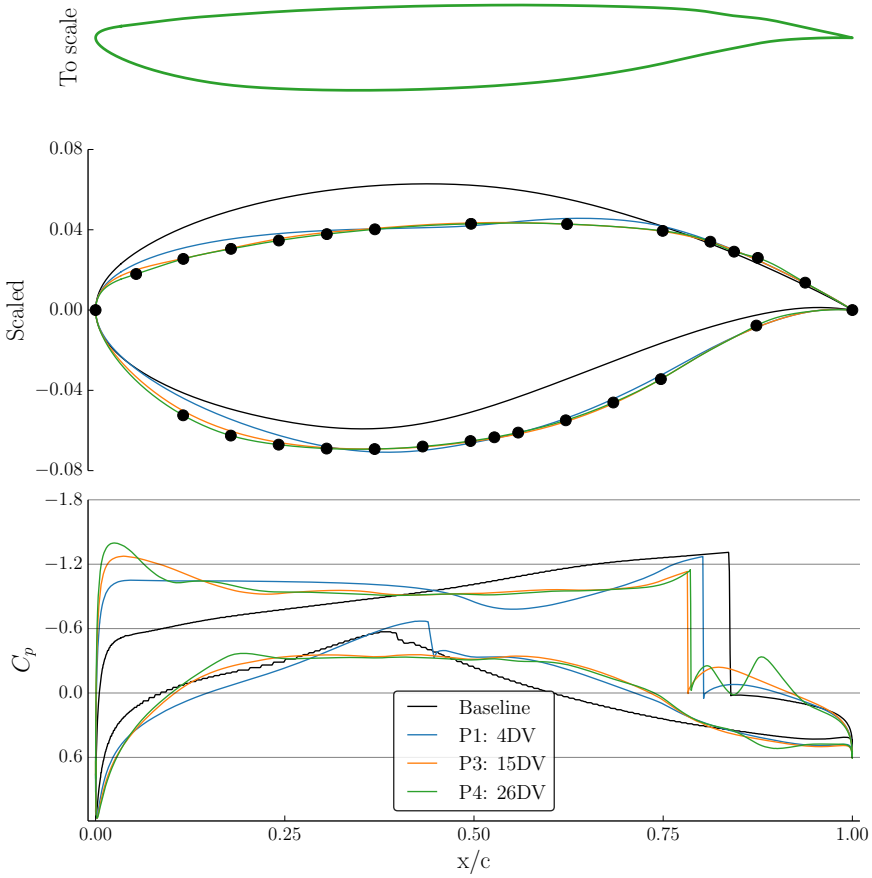


Figure 11: Transonic airfoil optimization results for the adaptively generated 26-DV parameterization. *Top*: Optimized airfoil to scale. *Middle*: Final airfoil and airfoils from two intermediate search spaces, showing final adapted parameterization. *Bottom*: Corresponding pressure profiles for the Mach 0.79 design point.

4. Optimization Results

Figure 11 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, and the added reflex camber at the trailing edge also adds some lift near the trailing edge. 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 11 also shows the non-uniform final parameterization, which is the result of adding design variables over five levels. The sequence of adapted parameterizations is shown in Figure 12.

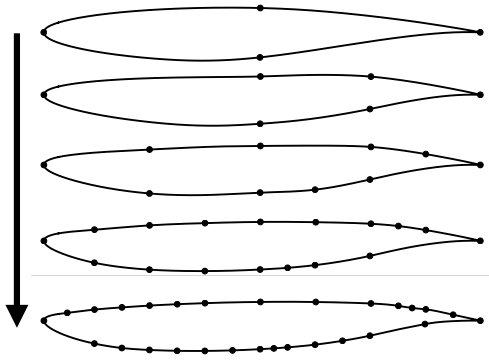
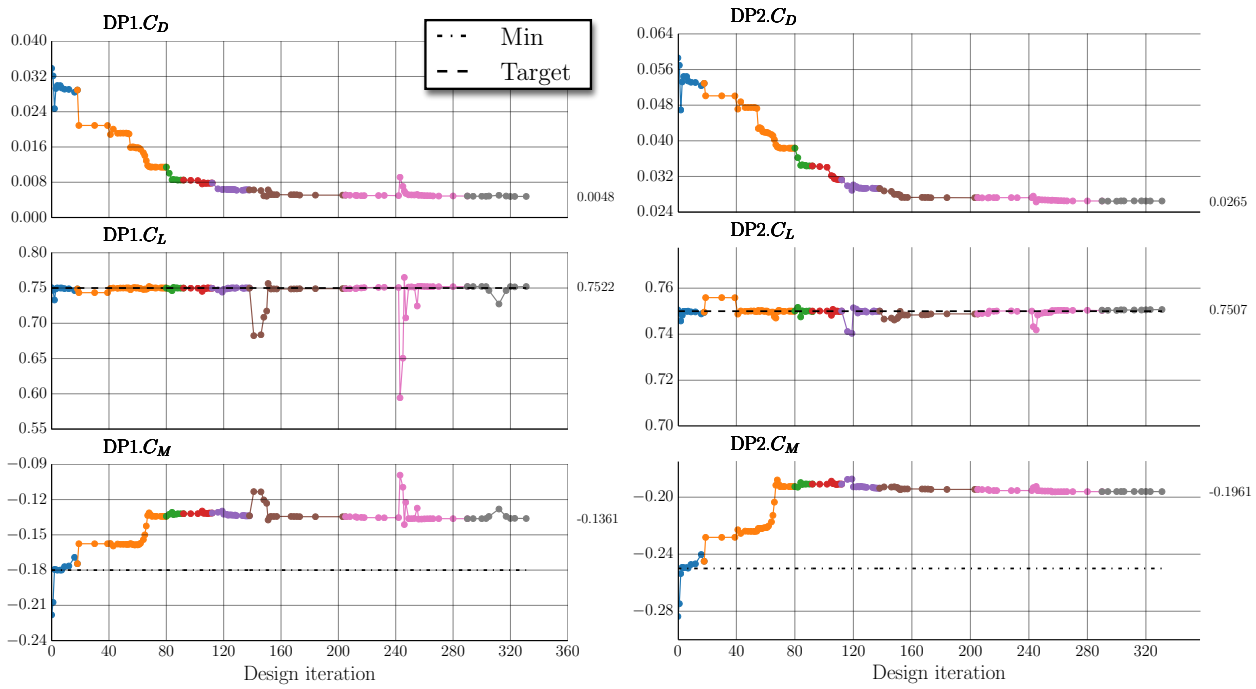


Figure 12: History of adaptive refinement showing best airfoils attained under each parameterization.

Figure 13 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.



(a) Mach 0.79 design point

(b) Mach 0.82 design point

Figure 13: 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.

5. Comparison to Static Parameterizations

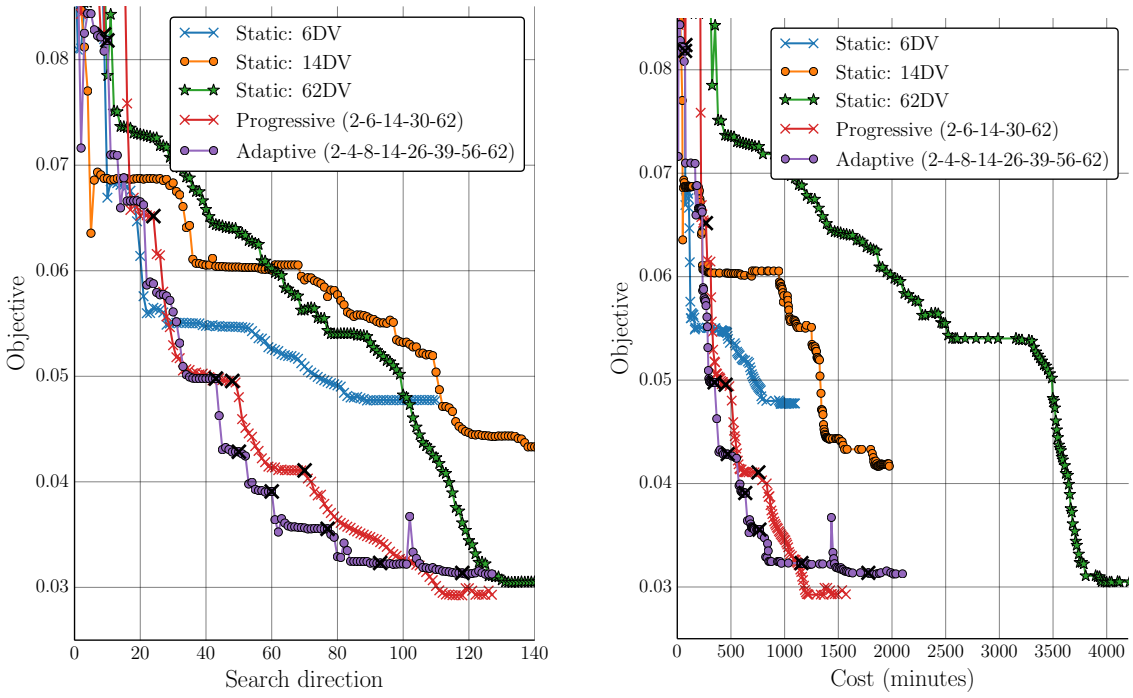
The left frame of Figure 14 compares the convergence of the drag objective for the various parameterizations. Initially, there is a somewhat convoluted startup period of 10-20 search directions, where the initially violated constraints were being driven to satisfaction at the expense of drag. Afterwards, the progressive and adaptive approaches strongly outperform any of the static parameterizations, achieving more consistent progress, converging far faster, and ultimately reaching superior designs. This is a clear confirmation of the predicted behavior, described and illustrated notionally in Figure 2 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 probably 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,[h] 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 14, which shows objective improvement vs. an estimate of wall-clock time[i]. 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,[20,26] 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 14 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.

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

[i] Rough timings on 24 Intel Haswell cores



(a) Search direction history

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

Figure 14: Convergence of combined drag value ($C_{D_1} + C_{D_2}$) (ignoring satisfaction of constraints) for each parameterization method. ×-marks denote search space refinements.

As a final note for this problem, Figure 15 shows several representative airfoils encountered during optimization. We observe that with a progressive or adaptive approach, the entire design trajectory is smoother. This is a desirable characteristic both from robustness standpoint and also because it makes it possible to stop at any point during optimization and have a reasonable design.

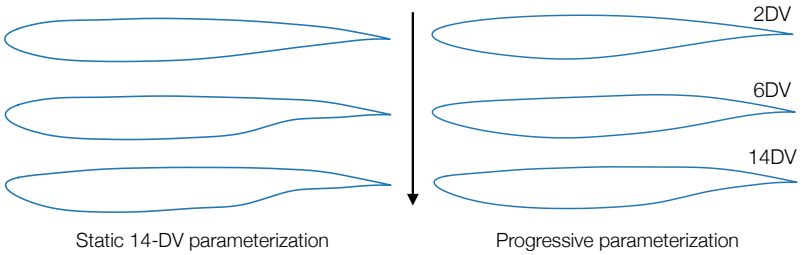


Figure 15: Typical airfoils encountered during optimization under a progressive parameterization (*right*) are consistently much smoother than airfoils encountered under a static parameterization (*left*).

B. Geometric Shape Matching Benchmark

In this two-part example, we establish a benchmark to assess the ability of our approach to discover the parameters necessary to solve a difficult optimization problem with a known solution. In the process, we compare the different indicators and assess the performance of our search procedure.

The problem is geometric shape matching to a swept, twisted wing. In shape matching, we examine the convergence from a baseline geometry to an attainable target shape. The objective function aims to minimize the deviation between the current shape and the target in a least-squares sense:

$$\mathcal{J} = \sum_{i=1}^{N_{verts}} \|\mathbf{v}_i - \mathbf{v}_i^*\|^2 \tag{8}$$

where $\mathbf{v}_i$ are the current vertex coordinates and $\mathbf{v}_i^*$ are the corresponding target vertex coordinates. The wing is represented by a discrete geometry with $N_{verts} \approx 197K$.

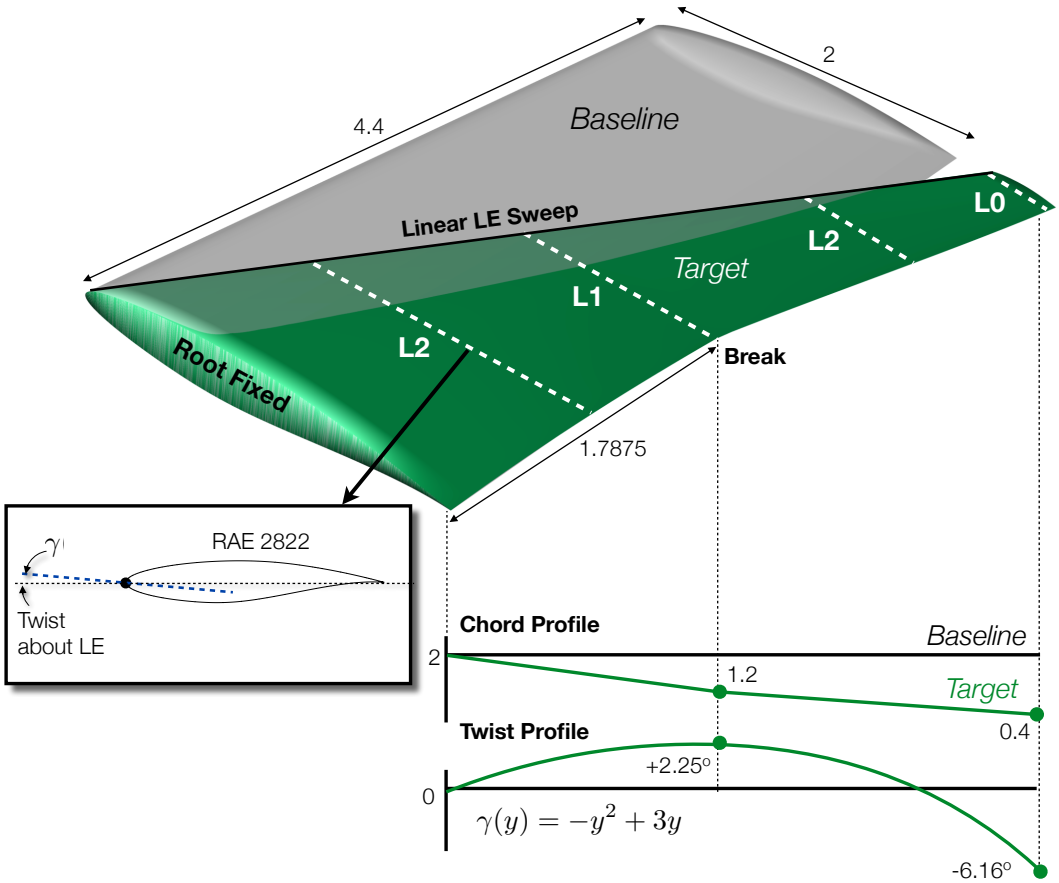


Figure 16: Baseline and target planform profiles. Initial shape control station is labeled L0, after which L1 is added, followed by the L2 stations, etc.

Importantly, 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 16 shows the the baseline and target shapes. The baseline is a straight wing with no twist, taper or sweep. 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 ("L1", "L2", etc.), opening up new degrees of freedom. Control over twist, sweep and chord can happen at different stations, allowing for "anisotropic" shape control. The target geometry is a wing with the same airfoil section, but substantial twist, chord-length and sweep profiles, as shown in Figure 16. 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 target shape is unattainable under the 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 our system can recover or approximate this "ideal" parameterization.

This seemingly simple problem is actually, by construction, pathological in several ways. First, it exhibits highly anisotropic parameter refinement requirements. Second, it is scaled in precisely the *converse* way: the objective gradients with respect to chord and sweep control are orders of magnitude higher than the gradients with respect to twist, even though the twist variables are the most needed. Third, the candidate parameters exhibit a high degree of "redundant potential", making it challenging for the search procedure to find an effective combination of parameters. Many aerodynamic problems share these same features, making this an excellent preliminary benchmark for an adaptive approach.

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, and one using Hessian information, I_H, which is accurately computable for this analytic objective.[j]

Figure 17 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 III.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

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

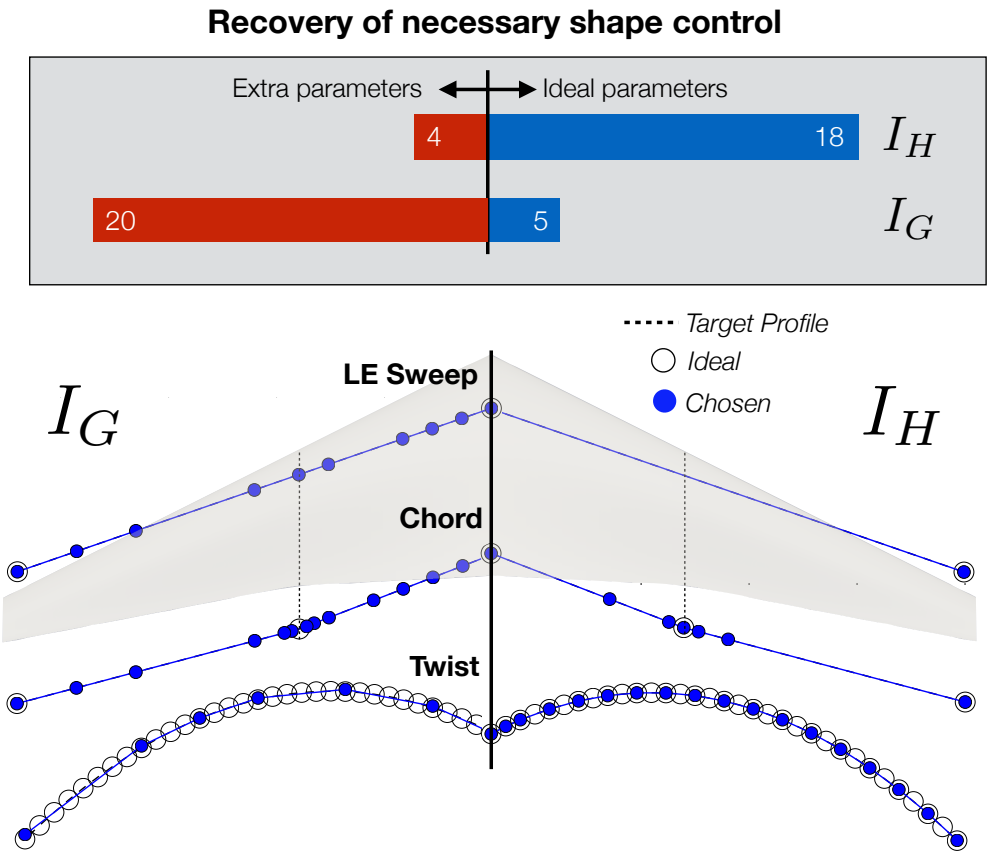


Figure 17: Performance of gradient indicator I_G and Hessian indicator I_H. *Top*: I_H recovers the expected parameters with few extras, while I_G mostly adds extraneous parameters. *Bottom:* Refinement patterns and optimized planform distributions with I_G (*left*) and I_H (*right*).

adaptation procedure did precisely this, and correctly identified the necessary parameters 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 segments. It has also begun to add the next nested level of control near the root.

Now compare the left half of Figure 17, which shows the results using the gradient-norm indicator I_G after 25 adaptation cycles.[k] Although the overall shape recovery is still quite decent, the refinement pattern is inaccurate and fails to efficiently capture the important design variables, and therefore results in an inferior match, especially in the twist profile.

For this objective function, the chord and sweep objective gradients were much higher than the twist gradients. Thus chord and sweep were favored, even though they offered only extremely short-term potential. On a well-scaled problem, I_G should perform much better. This suggests a simple correction to the gradient indicator to soften the impact of poor scaling

$$I_G^*(\mathbf{C}_c) = \left\| s_i \frac{\partial \mathcal{J}}{\partial X_{c_i}} \right\| \tag{9}$$

where s_i are scaling factors, as used to improve the conditioning of an optimization. However, determining appropriate values s_i is far from straightforward. 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 tremendous gains that could be made with even an approximation of second derivative information.

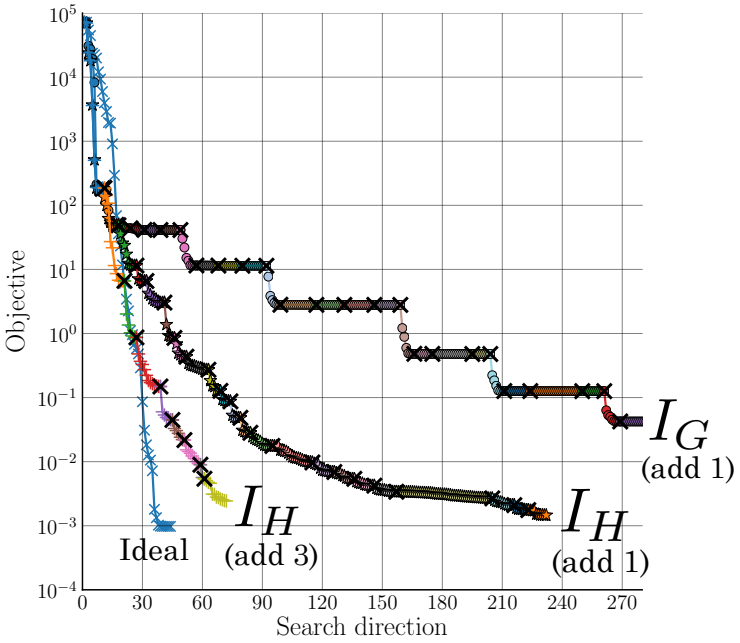


Figure 18: Shape-matching objective convergence for different indicators and search strategies. Solid blue line shows the "best possible" convergence, using the *a priori* known best possible 35-DV parameterization. Each color represents a different parameterization and ×-marks denote search space refinements.

Figure 18 compares the objective convergence under the two indicators. The gradient indicator frequently adds parameters with almost no potential, leading it to stall for several adaptation cycles. Despite the relatively poor performance of the gradient indicator, it still managed to reduce the objective by over 6 orders of magnitude, indicating reasonable shape matching. The Hessian indicator, however, achieves good progress at every cycle and reaches a superior design.

Performance is still relatively slow with respect to the "ideal" parameterization, shown in Figure 18. In this exercise, it took many adaptation cycles to drive towards the target shape, because only one parameter could be added at a time and we searched only one level deep in the parameter tree. As mentioned in section IV.C, much higher growth rates generally 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 adaptation). For this test, we no longer exhaustively evaluate all combinations, as this becomes prohibitive. Instead, we use the search procedure (Function 7) to seek a good, if not perfect, ensemble of shape parameters, by evaluating a small number of candidates.

[k] Although we used the L_2 norm of the gradients here, other norms provided nearly identical results.

Figure 19 shows the adapted refinement pattern. Although there are some more unnecessary parameters than before, the shape recovery is excellent. Moreover, as shown in Figure 18, the convergence rate for this strategy is much faster, starting to approach the performance of the ideal parameterization.

As a final note, if there is an extremely high degree of redundancy among candidates, the performance of this search algorithm can suffer. As a final experiment, we look five levels deep, and request 32 design variables immediately, without any prior optimization. There are a total of 93 candidates. An exhaustive search would involve evaluating all $\sim 8 \cdot 10^{24}$ possible combinations of the parameters, making an efficient search procedure essential.

Figure 20 shows the initial priority queue formed by analyzing each candidate shape control element independently, using the Hessian indicator. 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 does work on this problem, but it adds many extraneous variables. Function 7 must work its way through all of the now-useless parameters, one window (of size w) at a time, until finally discovering the still-useful 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 transformational 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 one or two levels deeper than the current parameterization, which eliminates most of the redundancy and results in excellent performance from the basic search procedure.

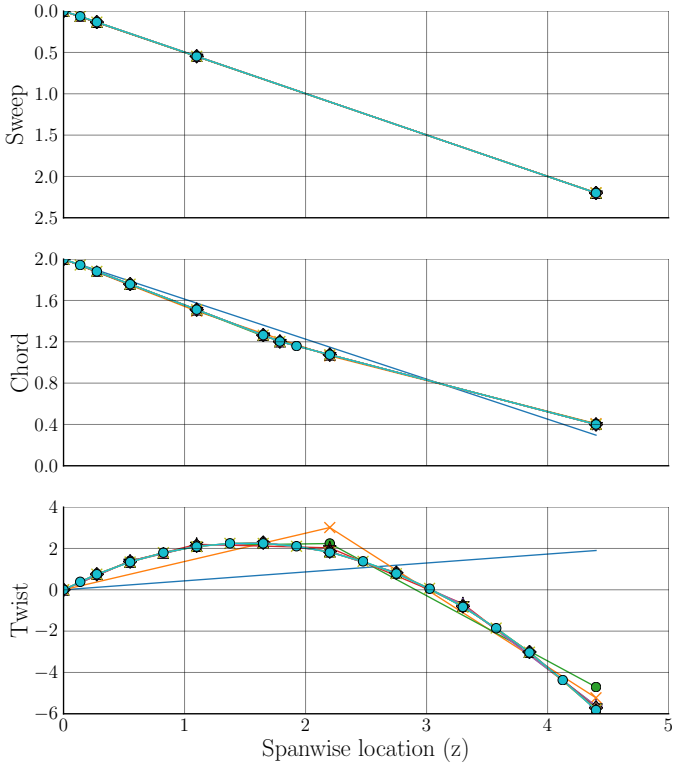


Figure 19: Final recovered planform distribution (search depth of 2, adding 3 parameters at a time, Hessian-based indicator).

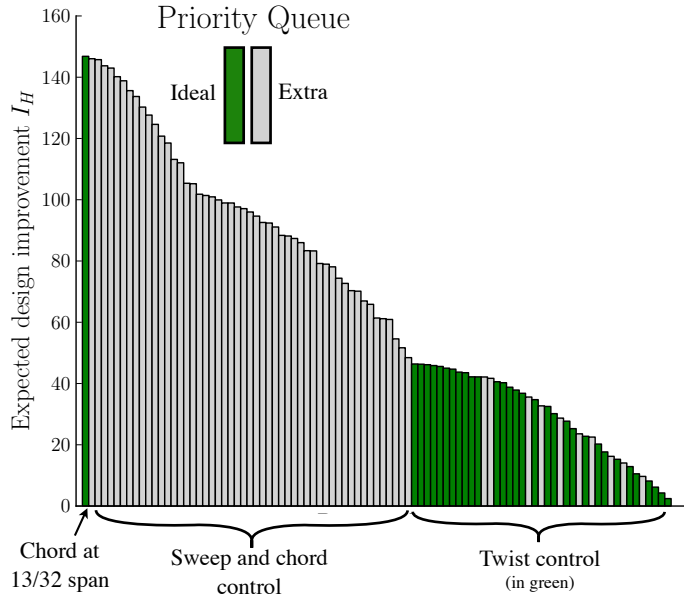


Figure 20: The priority queue after Phase I of the greedy algorithm (search depth of 5, adding 32 parameters at once, Hessian-based indicator). The 32 parameters that would best recover the target shape are highlighted in green. After adding the first parameter in the queue, all of the subsequent gray parameters (chord and sweep controllers) become redundant.

VI. 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 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× in some cases) over using all the design variables up front. Additional important benefits of this approach include:

- **Completeness**: The full design space can be explored more thoroughly, as it is not restricted by the initial parameterization.
- **Feedback**: The refinement pattern conveys information about the design problem.

An important feature of our implementation is that it is architected to work with arbitrary geometry modelers. Some additional development work is required to prepare a modeler for adaptive use. However, the amount of setup time eliminated from each optimization strongly justifies this expenditure. With modest tailoring, it can also invoke different aerodynamic design frameworks.

VII. Future Work

As the designer no longer specifies 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, limits on excessive curvature, etc.) can in theory be codified, which we hope to address in the future.

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 aerodynamic objectives can take advantage of an approximated Hessian-based indicator. 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 Nemec 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 and Joshua Leffell and with the manuscript reviewers, Tom Pulliam and Marco Ceze (all NASA Ames). Support for this work was provided by a NASA Seedling award.

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]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.

[6]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.

[7]Beux, F. and Dervieux, A., "A Hierarchical approach for shape optimisation," Research Report RR-1868, INRIA, 1993.

[8]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.

[9]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.

[10]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.

[11]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.

[12]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.

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

[14]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.

[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]Han, X. and Zingg, D. W., "An Evolutionary Geometry Parametrization for Aerodynamic Shape Optimization," *AIAA Paper 2011-3536*, Honolulu, HI, June 2011.

[17]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.

[18]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.

[19]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.

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

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

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

[23]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.

[24]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.

[25]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.

[26]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 tesselated 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.

Function 4: $Optimize(\cdot)$
Parametric Geometry Engine
PDE Solver Functions

Input: Shape deformation function D with initial design variable values $\mathbf{X}_0$, objective function $\mathcal{J}$, constraints C_j
Result: Optimized surface $\mathbf{S}$, adjoint solution ψ
$\mathbf{X} \longleftarrow \mathbf{X}_0$
repeat
  $\mathbf{S} \longleftarrow GenerateSurface(D, \mathbf{X})$
  $\mathbf{M} \longleftarrow DiscretizePDE(\mathbf{S})$
  $\mathbf{Q} \longleftarrow SolvePDE(\mathbf{M})$
  $\mathcal{J} \longleftarrow ComputeObjective(\mathbf{Q}, \mathbf{S})$
  $\psi \longleftarrow SolveAdjoint(\mathbf{M}, \mathbf{Q})$
  foreach X_i *in* $\mathbf{X}$ **do**
    $\frac{\partial \mathbf{S}}{\partial X_i} \longleftarrow ShapeDerivative(D, X_i)$
    $\frac{\partial \mathcal{J}}{\partial X_i} \longleftarrow ProjectGradient(\psi, \frac{\partial \mathbf{S}}{\partial X_i})$
  end
  $\mathbf{X} \longleftarrow NextDesign(\mathbf{X}, \frac{\partial \mathcal{J}}{\partial \mathbf{X}})$ `// Optimizer`
until $Stop(\mathcal{J}, \frac{\partial \mathcal{J}}{\partial \mathbf{X}})$
return $\mathbf{S}, \psi$