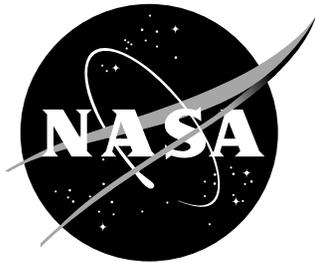


NASA/TM-2013-Seedling Phase 1 Final Report



Adaptive Shape Parameterization for Aerodynamic Design

George R. Anderson
Stanford University, Stanford, CA

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

September 2013

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collection of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- **TECHNICAL PUBLICATION.**
Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.
- **TECHNICAL MEMORANDUM.**
Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- **CONTRACTOR REPORT.**
Scientific and technical findings by NASA-sponsored contractors and grantees.

- **CONFERENCE PUBLICATION.**
Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- **SPECIAL PUBLICATION.**
Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- **TECHNICAL TRANSLATION.**
English- language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, and organizing and publishing research results.

For more information about the NASA STI Program, see the following:

- Access the NASA STI program home page at [*http://www.sti.nasa.gov*](http://www.sti.nasa.gov)
- E-mail your question via the Internet to help@sti.nasa.gov
- Fax your question to the NASA STI Help Desk at 443-757-5803
- Phone the NASA STI Help Desk at 443-757-5802
- Write to:
NASA STI Help Desk
NASA Center for Aerospace Information
7115 Standard Drive
Hanover, MD 21076-1320

NASA/TM-2013-Seedling Phase 1 Final Report



Adaptive Shape Parameterization for Aerodynamic Design

George R. Anderson
Stanford University, Stanford, CA

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

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, CA 94035

September 2013

Acknowledgments

Funding for this project was fully provided by a NASA ARMD Seedling Fund grant. The authors also thank Marian Nemeč for many helpful discussions and for his development of the static-parameterization design framework used in this study.

The use of trademarks or names of manufacturers in this report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.

Available from:

NASA Center for AeroSpace Information
7115 Standard Drive
Hanover, MD 21076-1320
443-757-5802

Abstract

This report concerns research performed in fulfillment of Phase 1 of an ongoing NASA Seedling Fund grant. The overall goal is to develop an aerodynamic shape optimization framework that supports automated shape parameterization. The four objectives for the work were to mature a constraint-based deformation technique, to develop the basic framework necessary to perform automated parameter refinement, to determine an importance indicator that prioritizes candidate design variables, and to develop an efficient refinement strategy. All of these objectives have been met to the degree appropriate for Phase 1. We discuss each in detail in this report.

Nomenclature

P	Vector of design variable definitions/locations
X	Vector of design variable values
\mathcal{J}	Objective function
$\frac{\partial \mathcal{J}}{\partial \mathbf{X}}$	Objective gradients to design variables
C	Computational cost
β	Ratio of cost of gradient computation to cost of PDE solution.
d	Optimizer search direction
I	Importance indicator
r	Trigger reduction factor
w	Averaging window

PDE Solution

α	Angle of attack
M	Discrete PDE mesh
Q	Discrete PDE solution
S	Discrete tessellated surface
M	Mach number
ψ	Adjoint solution

Subscripts

<i>cand</i>	Candidate design variables
<i>dv</i>	Design variables
<i>grad</i>	Objective gradient computation
<i>iter</i>	Design iteration
<i>opt</i>	Optimization
<i>ref</i>	Refinement

1 Introduction

AERODYNAMIC design is gradually transitioning from point analysis towards simulation-guided *shape optimization*, which seeks to improve an initial design, using numerical analysis to intelligently drive the shape changes. The key to effective aerospace design optimization is the *shape parameterization*, the geometric design variables that control the vehicle’s shape and govern the range of possible configurations. This work addresses the pressing need for shape parameterization techniques that give the designer more flexibility and control, while also offloading and automating as much work as possible.

1.1 Traditional Shape Optimization

From the beginning, shape optimization has made great promises: faster design improvement (in wall-clock time), more automation, reduced user-in-the-loop time, and minimal bias while exploring unfamiliar designs. Unfortunately, current approaches fall short of the mark in each of these areas.

An aerospace designer’s task ought to be to “ask the right questions”, which entails choosing an objective and imposing design constraints. In most current approaches, however, the designer spends excessive time on the mechanics of the tools: selecting shape design variables, establishing robust bounds on them, and other tasks which are not directly relevant to design. Despite the promise of reduced bias, it remains strongly present in the designer’s choice of design variables. The optimizer can work only within the range of reachable shapes. The design space may be excessively restrictive (or excessively open-ended) and is highly unlikely to include the optimal continuous shape.

Regarding the promise of accelerated design improvement, current methods meet with mixed success. While at first, the design can be quickly improved, the static design space prevents further improvement. When shape parameters are chosen before design begins, they can restrict the design space in irrelevant ways, needlessly hindering the discovery of optimal designs. A fixed parameterization cannot adapt to capitalize on new insights as they emerge during design.

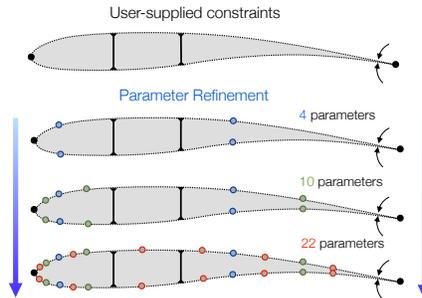


Figure 1: Progressive shape parameterization

1.2 Variable Shape Control

Most current methods use a static design space that does not evolve as the design progresses. If these parameters are too coarse, design improvement is severely restricted. If they are too refined while far from optimal, then navigation of the nonlinear design space is inefficient. In this work we examine the use of a *variable* design space that can be adapted as necessary, as illustrated in Figure 1. In practice, this typically means that a coarse parameterization is used when far from the optimal design, and that the resolution is gradually increased when approaching the optimum.

The primary benefit of a variable shape control approach is that the designer is no longer burdened with predicting how many and which design variables will be appropriate for a particular design problem. It also radically reduces user setup time, as design variables are automatically created and bounded. Furthermore, it constitutes a single hierarchical process that progressively drives the shape towards the true continuous optimal shape, instead of an approximation in an arbitrary fixed design space.

Although our primary goal is to streamline and automate design tools, using a variable approach is also motivated by a growing body of evidence that substantial design acceleration can be achieved by using a hierarchical parameterization approach.¹⁻⁴ Figure 2 illustrates the potential computational acceleration. Each curve shows the evolution of the objective using a different parameterization technique (steeper slopes indicate more efficient optimization). Using the coarse 14-DV parameterization initially achieves the fastest design improvement, as the space is simpler to navigate, but the improvement quickly bottoms out. The finer parameterizations can reach superior designs, but do so inefficiently because of the large number of design variables.

In *progressive* parameterization we start with a coarse design space that can be quickly evolved, and then add new degrees of freedom when necessary. This captures the best of both worlds, allowing efficient design improvement early on, while still retaining the ability to ultimately reach a superior design that is accessible only in more refined design spaces.

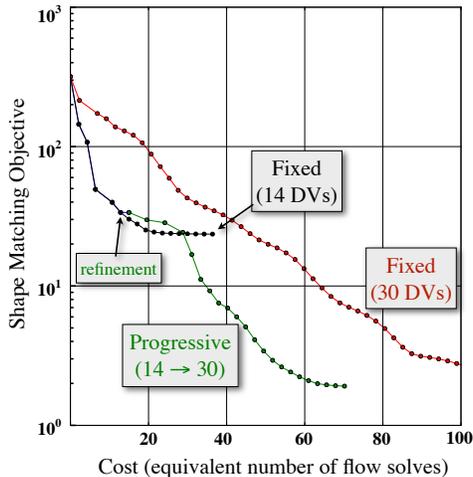


Figure 2: Objective convergence for static vs. progressive parameterizations

1.3 Constraint-Based Deformation

Using adaptive parameterization places greater emphasis on the *structure* of the shape parameterization method. One key objective of this work was to develop and mature a tool that allows design variables to be placed anywhere, and automatically. Our approach is called *constraint-based deformation*.

Geometric constraints are truly the only thing known *a priori* in design. Constraint-based deformation treats constraints exactly like design variables, so that only shapes satisfying the geometric constraints are considered during optimization. This view represents a fundamentally different approach to shape parameterization that unifies the concepts of design variables and geometric constraints. This differs from the conventional approach where constraints are an afterthought, enforced approximately via penalty functions or by awkwardly linking design variables together to preclude undesirable shape changes. This new technique determines both shape parameters and constraints at design time and guarantees exact satisfaction of geometric constraints. Flexibility can be added to the model as the design evolves, enhancing its ability to achieve the designers objective. Some examples of constraint-based deformation are shown in Figure 3.

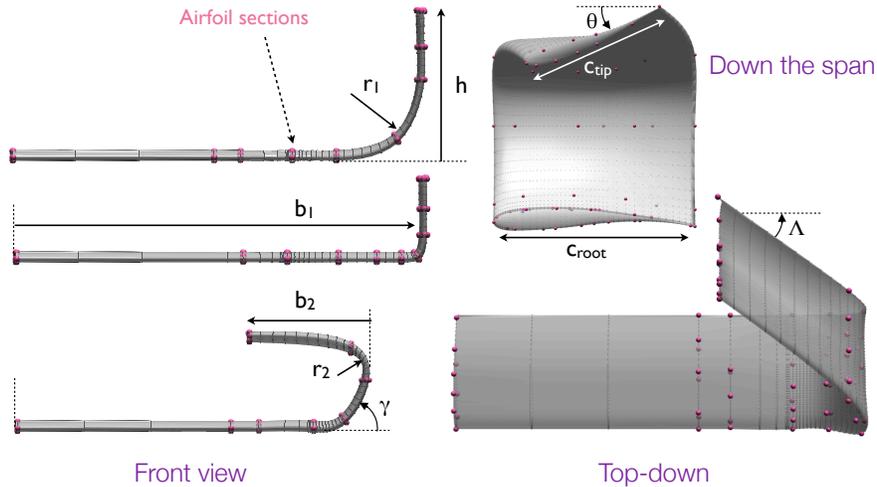


Figure 3: Parametric wingtip deformation using constraint-based deformation.

1.4 A New Role for the Adjoint

The second key enabling technology for our approach is the incorporation of sensitivity information provided by the adjoint equations. Since its introduction to the aerodynamic community 25 years ago,⁵ the adjoint method has revolutionized gradient-based shape optimization, rendering it computationally feasible to optimize on very large numbers of design

variables. The adjoint approach allows *all* of the objective gradients to be computed for a fixed cost of roughly two PDE solutions, instead of the $2N$ solutions required under a finite difference formulation.

In this work, we use the adjoint solution for a novel purpose. The adjoint in fact encodes much *more* information than traditional parametric shape optimization makes use of, and this information can be extracted at trivial cost. If used correctly, it can accelerate the rate of design improvement.

Specifically, we use the adjoint to compute gradients not only of existing shape design variables, but also of *potential* design variables. We can then determine whether the potential design variables would drive the design forward more effectively, and if so, inject them into the active set of design variables. Different design problems may call for different shape control. Our goal is to discover the necessary shape control in the process of optimization. Thus we expand the traditional role of the adjoint from efficiently computing objective gradients to *designing the design space itself*.

1.5 Organization

The following section provides an algorithm for optimization with adaptive shape parameterization. In Section 3 we discuss the shape parameterization methods used for this study. We then discuss a mixed modeling/observational approach to cost-benefit analysis to assess the efficiency of different refinement strategies. In Sections 6-8 we examine each of the essential components of the refinement strategy in detail, drawing conclusions from randomized numerical experiments.

2 Adaptive Parameterization Algorithm

2.1 Standard Shape Optimization Framework

Adjoint-based parametric shape optimization frameworks typically follow the iterative loop outlined in Function 1. First, a discrete tessellated surface \mathbf{S} is generated by a geometry modeler or deformer, based on the initial shape parameter values \mathbf{X} . Next, the solution domain is meshed and the PDE is solved (for our purposes, the fluid flow equations), which enables evaluation of the objective function \mathcal{J} (e.g. drag or sonic boom). Next, the adjoint equations of the PDE 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.

For this study, we use an existing adjoint-based aerodynamic design framework⁶ that uses an embedded-boundary Cartesian mesh method for inviscid flow solutions. We also leverage the framework’s adjoint formulation for computing gradients to prioritize candidate design variables

Function 1: $OptimizeStatic(\mathbf{P}, \mathbf{X}_0, Stop(\cdot))$

Parametric Geometry Engine

PDE Solver Functions

Input: Shape parameters \mathbf{P} with initial values \mathbf{X}_0 ,
stopping condition $Stop(\mathcal{J}, \frac{\partial \mathcal{J}}{\partial \mathbf{X}})$

Result: Optimized surface S , adjoint solution ψ

repeat

$\mathbf{S} \leftarrow GenerateSurface(\mathbf{P}, \mathbf{X})$

$\mathbf{M} \leftarrow GeneratePDEMesh(\mathbf{S})$

$\mathbf{Q} \leftarrow SolvePDE(\mathbf{M})$

$\mathcal{J} \leftarrow ComputeObjective(\mathbf{Q}, \mathbf{S})$

$\psi \leftarrow SolveAdjoint(\mathbf{Q}, \mathbf{M})$

foreach P_i, X_i in \mathbf{P}, \mathbf{X} **do**

$\frac{\partial \mathbf{S}}{\partial X_i} \leftarrow ComputeShapeDerivative(P_i, X_i)$

$\frac{\partial \mathcal{J}}{\partial X_i} \leftarrow ProjectGradient(\psi, \frac{\partial \mathbf{S}}{\partial X_i})$

end

$\mathbf{X} \leftarrow NextDesign(\mathbf{X}, \frac{\partial \mathcal{J}}{\partial \mathbf{X}})$ // Gradient-based optimizer

until $Stop(\mathcal{J}, \frac{\partial \mathcal{J}}{\partial \mathbf{X}})$

when refining the design space. Optimization can be handled with any black-box gradient-based optimizer; for this study we use SNOPT.⁷

2.2 Parametric Geometry Generation

The function $GenerateSurface(\cdot)$ represents an arbitrary geometry modeler (e.g. a CAD engine or surface deformer) that generates a tessellated surface from a set of design parameters. The design framework described above interacts with arbitrary geometry modelers, using an XML-based protocol for communication.⁶ The geometry parameterization components are discussed in more detail in Section 3.

2.3 Adaptive Optimization

Function 1 gives the process for optimizing in a *static* design space defined by a fixed set of design variables. Our *adaptive* optimization algorithm invokes a series of calls to this design framework, refining the parameterization between each call. Algorithm 2 shows a basic strategy for optimization with adaptive shape parameterization. Besides the call to the standard design framework,

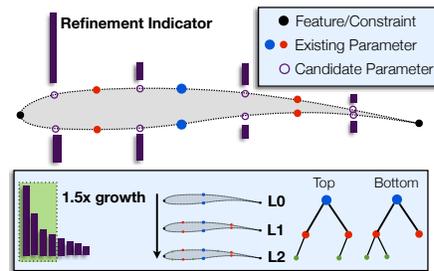


Figure 4: Prioritized candidates for parameterization refinement

there are a four new functions related to parameterization refinement. The first function, *GetCandidateDesignVariables*(\cdot) generates a list of possible new shape parameters. This function depends on the particular parameterization technique being used, and is discussed in more detail in Section 3.

The remaining three adaptation functions are independent of the geometry modeler. The *Trigger*(\cdot) determines when to initiate the next refinement level. *ComputeImportanceIndicator*(\cdot) determines which of the design variables are most important and builds a “priority queue” of candidates as illustrated in Figure 4. Finally, *SetPace*(\cdot) determines how many of these candidates to add, effectively governing the rate of shape parameter growth. These three functions are discussed in detail in Sections 6-8.

Algorithm 2: Adaptive Parameterization Optimization

Parametric Geometry Engine

Refinement Strategy

Input: Initial shape parameters \mathbf{P}_0 with values \mathbf{X}_0
Result: Optimized surface \mathbf{S}
 $\mathbf{P} \leftarrow \mathbf{P}_0, \mathbf{X} \leftarrow \mathbf{X}_0$
repeat
 $\mathbf{X}, \psi \leftarrow \text{OptimizeStatic}(\mathbf{P}, \mathbf{X}, \text{Trigger}())$
 $\mathbf{P}_{cand}, \mathbf{X}_{cand} \leftarrow \text{GetCandidateDesignVariables}(\mathbf{P}, \mathbf{X})$
 foreach P_i, X_i **in** $\mathbf{P}_{cand}, \mathbf{X}_{cand}$ **do**
 $I_i \leftarrow \text{ComputeImportanceIndicator}(P_i, X_i, \psi)$
 end
 $\text{SortByIndicator}(\mathbf{P}, \mathbf{X}, \mathbf{I})$
 $n \leftarrow \text{SetPace}(\mathbf{I})$
 $\mathbf{P} \leftarrow \mathbf{P} \cup \mathbf{P}_{cand}[1 \dots n]$
 $\mathbf{X} \leftarrow \mathbf{X} \cup \mathbf{X}_{cand}[1 \dots n]$
until *overall stopping condition met*

2.4 Uniform Refinement

A *uniform* refinement strategy is a special simplified case of Algorithm 2, where *all* candidate design variables are added, rendering the indicator and pacing irrelevant. To implement a uniform refinement strategy, we simply bypass the computation of indicators in Algorithm 2 and add all of the new candidates. A uniform refinement strategy meets the goals of automation and consistency with the continuous optimal solution, and it is also easier to implement. However, it has efficiency implications, as the number of design variables will grow much faster than in an adaptive scheme.

3 Shape Parameterization

Adaptive parameterization is theoretically compatible with constructive modelers (e.g. CAD or in-house geometry tools) or discrete geometry de-formers (e.g. FFD, bump functions, constraint-based deformation, etc.), and we strive for modularity with respect to the shape parameterization. To function with a standard design framework, a geometry modeler must provide a list of available design variables with minimum and maximum bounds, and subsequently generate surfaces on demand for any set of parameter values requested by the optimizer.^a

To function with an *adaptive* parameterization framework, the modeler must additionally have systematic methods for

- Generating and bounding *new* candidate design variables.
- Transferring constraints between refinement levels.

The method for generating candidates need not be strictly hierarchical (or “nested”), but it should at least allow substantial refinement depth.

Besides these strict requirements, several authors have discussed desirable qualities for a shape parameterization technique, such as automation, consistency, smoothness, compactness, effectiveness, robustness, and intuitiveness.^{8–12} Using an adaptive approach places greater emphasis on the *structure* of the parameterization scheme, which can be summarized in the following desirable characteristics:

1. Hierarchical organization of parameters.
2. Refinement can be localized to particular regions of the shape
3. Unlimited refinement depth
4. Exact shape preservation when transferring between levels.

Many geometry modelers that are satisfactory for fixed parameter optimization do not have all these characteristics. For example, with a few exceptions, most constructive modelers (such as CAD packages) do not preserve the shape exactly when changing parameterizations, which means ground (and thus time) would be lost when refining the design space. Discrete geometry deformation techniques, on the other hand, naturally preserve the shape exactly.

In the following sections we discuss the two naturally hierarchical and shape-preserving parametric geometry deformation methods used in this study.

^aIdeally, shape derivatives are also provided, but this is not a strict necessity, as shape derivatives can be computed automatically by finite differencing.

3.1 Rigid Wing Section Transformation

For wing planform design, we use an analytic deformer that applies rigid transformations to arbitrary spanwise stations and lofts the deformation between the stations. Rigid transformations include translation (sweep and span), rotation (twist and dihedral) and scaling (thickness-to-chord ratio). For this study, we restrict the deformation to twisting only and we linearly interpolate the twist between stations, a deliberate choice that simplifies the randomized testing discussed in Section 5.1.

Candidate design variables are generated by subdividing the spanwise stations. If there are N spanwise stations, we can generate $N - 1$ candidates at the midpoints between adjacent sections, as shown in Function 3. However, we could also look “deeper” by generating $2^i - 1$ evenly spaced candidates between each pair of adjacent stations. A yet more advanced approach could allow uneven spacing, but we defer this until a later study.

Function 3: *GetCandidateDesignVariables(\cdot)*

Input: Current shape parameters \mathbf{P} with values \mathbf{X}

Result: Candidate shape parameters \mathbf{P}_{cand} with values \mathbf{X}_{cand}

$\mathbf{P}_{cand} \leftarrow \emptyset, \mathbf{X}_{cand} \leftarrow \emptyset$

SpatiallySortParameters(\mathbf{P}, \mathbf{X})

foreach pair of adjacent parameters (P_i, P_j) in \mathbf{P} **do**

$P_{new} \leftarrow$ midpoint of P_i and P_j

$X_{new} \leftarrow$ *Interpolate* (X_i, X_j)

$\mathbf{P}_{cand} \leftarrow \mathbf{P}_{cand} + P_{new}$

$\mathbf{X}_{cand} \leftarrow \mathbf{X}_{cand} + X_{new}$

end

3.2 Constraint-Based Deformation

In references^{13,14} we developed a constraint-based approach to shape parameterization for arbitrary aerospace configurations. Geometric constraints are managed systematically, ensuring that they are satisfied at every design iteration. The technique works on arbitrary 3D aerospace configurations, as demonstrated in Figure 3, but for the purposes of this study we use it to perform wing section design, as illustrated in Figure 1. Generation of new candidates is handled similarly to the wing planform parameterization, given in Function 3.

4 Designing an Adaptive Strategy

It is worth pausing for a moment to note that by simply using progressive parameterization in *any* form, we have already met several of our goals. Even a uniform refinement strategy without any notion of localized adaptation is automated and has the ability to reach the continuous solution. It also accelerates design improvement by avoiding excessively fine parameterizations early in design.

The remainder of this study is focused on process efficiency, which requires much closer attention to the details of the refinement strategy. Our goal is to find a strategy (consisting of a trigger, an importance indicator, and rate of variable introduction) that maximizes the design improvement $\Delta\mathcal{J}$ relative to the cost of the optimization C_{opt} . Naturally, no single refinement strategy will be perfect for all design problems. We are looking for the essential ingredients of a robust strategy that works consistently and efficiently over a broad spectrum of aerodynamic design problems.

4.1 Measuring Design Improvement

The design improvement is simply the change in the objective function $\Delta\mathcal{J}$. The rate of design improvement cannot be rigorously modeled. It must be measured through numerical experiments, as it strongly depends on the efficacy of the current shape parameters at reducing the arbitrary objective function (e.g. drag vs. sonic boom vs. shape matching).

4.2 Cost-Benefit Trade

The total cost of shape optimization on a static parameterization (or on one level of a progressive parameterization) is

$$C_{opt} = N_{iter}C_{iter} + C_{ref} \quad (1)$$

where N_{iter} is the number of design iterations, C_{iter} is the cost per iteration, and C_{ref} is the cost of refining the parameterization. In the following sections, we model the costs C_{iter} and C_{ref} for typical aerodynamic problems. While we expect that $N_{iter} \sim \mathcal{O}(N_{dv})$ when using a quasi-Newton optimizer such as BFGS, we cannot accurately predict it due to the nonlinearity of the design space. Our approach is therefore to observe how N_{iter} behaves statistically in numerical experiments.

4.2.1 Cost per Design Iteration, C_{iter}

One design iteration is comprised of three significant actions: evaluating the objective (typically by solving a PDE), computing adjoint solution(s), and projecting the surface objective gradient into each of the design variables. We will assume that there is only one adjoint solve

(i.e. one objective and no aerodynamic constraints) and that the cost of the adjoint solution is equal to that of the flow solution, which is a reasonable approximation in most cases. Then the cost \overline{C}_{iter} of a design iteration is

$$\overline{C}_{iter} = 2\overline{C}_{PDE} + N_{dv}\overline{C}_{grad}$$

For simplicity, we assume perfect parallel efficiency. We also assume that the cost of a flow or adjoint solution remains roughly constant throughout the optimization.^b Then we can divide out the constant PDE solution cost to obtain the *non-dimensional* cost per design iteration

$$C_{iter} = 2 + N_{dv}\beta \quad (2)$$

where $\beta = \frac{\overline{C}_{grad}}{\overline{C}_{PDE}}$ indicates the relative cost of a gradient projection to a PDE solution. \overline{C}_{PDE} depends on the PDE solution fidelity (scaling roughly linearly with the number of unknowns in the discrete solution when using a multigrid solver). \overline{C}_{grad} depends on the speed of the geometry modeler and the cost of a gradient projection. In many situations, the PDE solution is at least an order of magnitude more expensive than the geometry generation and gradient projection, so $\beta \ll 1$. Nevertheless, the second term in Equation 2 can still become significant when there are large numbers of design variables.

4.2.2 Cost of Refining the Parameterization, C_{ref}

Function *ComputeImportanceIndicator*(\cdot) in Algorithm 2 dominates the cost of design space refinement. In Section 7, we show that this process involves the projection of the adjoint solution onto the shape sensitivities with respect to the design variable, which has a nondimensional cost of roughly β , making the total refinement cost for all N_{cand} candidates

$$C_{ref} = N_{cand}\beta \quad (3)$$

Equation 3 highlights the (intuitive) cost tradeoff. Considering more candidate design variables naturally increases the chances of finding effective design variables, but it also linearly increases the overhead associated with refinement.

4.2.3 Total Optimization Cost, C_{opt}

Substituting the cost models (Equations 2 and 3) into Equation 1, we obtain a model for the total nondimensional cost of an optimization on a fixed set of parameters:

$$C_{opt} = N_{iter}(2 + N_{dv}\beta) + N_{cand}\beta \quad (4)$$

^bThis would be inaccurate when using progressively refined flow meshes, where coarse mesh solutions are used early in design and fine mesh solutions only near the optimum.

By choosing a particular pacing strategy we are implicitly prescribing N_{dv} and N_{cand} , leaving N_{iter} to be determined experimentally.

Finally, our goal is to maximize the design improvement relative to the cost, at every parameter refinement level:

$$\frac{\Delta \mathcal{J}}{N_{iter}(2 + N_{dv}\beta) + N_{cand}\beta} \quad (5)$$

Equation 5 allows us to make the following general statements. For relatively expensive PDE solutions (where $\beta \ll 1$), the most efficient strategy is the one that maximizes $\frac{\Delta \mathcal{J}}{N_{iter}}$. This suggests that considering more candidates and carefully deciding which to add is time well-spent. For relatively *inexpensive* PDE solutions ($\beta \not\ll 1$), the most efficient strategy is the one that maximizes $\frac{\Delta \mathcal{J}}{N_{iter}N_{dv} + N_{cand}}$. In this case, the refinement overhead is more significant, as is the total number of design variables.

5 Numerical Experiments

Our objective for the remainder of this study is to evaluate various approaches to each component of the refinement strategy (trigger, indicator and pacing), drawing conclusions from numerical experiments. In most previous studies on adaptive refinement, broad conclusions are drawn from a handful of specific model problems, which may not be representative even of that particular type of problem (and much less of a broad class of problems). Single cases like these are more likely to lead to conclusions that do not hold in general. In this work we examine large sets of randomized cases to reduce the possibility of error. The goal is to draw statistically significant conclusions about mean performance. While random cases are not precisely representative of real design problems, they do provide a notion of *expected* performance on aerodynamic problems.

5.1 Test Case

Our test problem is wing twist matching. The geometry is the swept and tapered transport wing in Figure 5. Twisting is performed by the rigid wing deformer described in Section 3.1. The goal is to add twist stations where necessary and gradually match a known target design. For each case we randomize the initial and target shapes, as well as the flight conditions, in order to evaluate the abil-

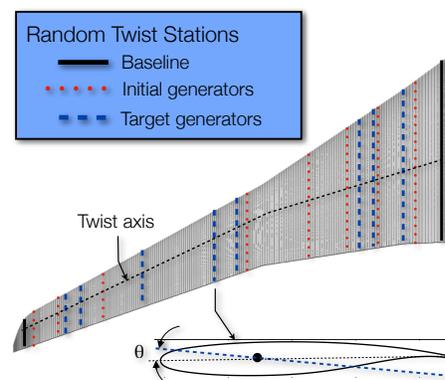


Figure 5: Example of a random set of twist stations

ity of various refinement strategies to drive an arbitrary starting shape to an arbitrary target shape.^c

Twist stations are initially placed at the root and tip of the wing. For each case, 8 more twist stations are then placed at randomly chosen spanwise locations and randomly perturbed, which generates an initial shape. This process is repeated to generate a target shape, using a *different* set of 8 random stations and perturbations. Because the spanwise interpolation for the rigid deformer is linear, the target is exactly attainable, but only if the adaptation process manages to capture all 16 parameters (8 each) used to generate the initial and target shapes.

5.2 Objective Functionals

We consider two inverse design functionals. The first is a *geometric* shape matching functional defined as the sum of squared distances between corresponding vertices:

$$\mathcal{J}_{geom} = \sum_{i=1}^{n_{verts}} (\mathbf{v}_i - \mathbf{v}_i^*)^2 \quad (6)$$

where \mathbf{v}_i are the current vertex coordinates and \mathbf{v}_i^* are the corresponding target vertex coordinates. Second, we consider an *aerodynamic* inverse design functional defined as the squared deviation from a target pressure profile:

$$\mathcal{J}_{aero} = \sum_{i=1}^{n_{verts}} (C_p - C_p^*)_i^2 \quad (7)$$

where the target pressure coefficient C_p^* is specified at each vertex on the discrete surface.

Both functionals have a minimum value of zero when the matching is exact. The purely geometric functional is a much simpler objective, as it eliminates the severe nonlinearities associated with the flow equations and mesh discretization. The aerodynamic functional is more representative of realistic aerodynamic shape optimization problems. However, it is much more expensive to compute, as it involves the solution of the flow and adjoint equations.

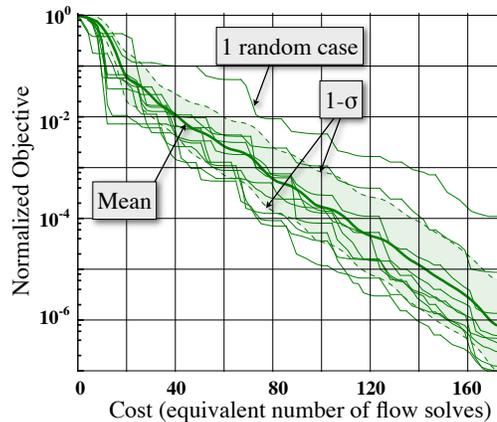
5.3 Flight Conditions

For the aerodynamic objective function, in addition to randomizing the parameterization, we also impose random subsonic flight conditions within the Mach range $0.3 \leq M \leq 0.4$ and angle of attack range $0^\circ \leq \alpha \leq 2^\circ$. The flow and adjoint solver settings are identical for all cases, including generating the target pressure profiles.

^cNaturally, arbitrary only *within* this class of shape parameterizations.

5.4 Statistics

Each random study produces a series of independent objective histories, like the example shown in Figure 6. Each thin line corresponds to a different design problem. The objective functions are independently normalized so that each history starts at a value of 1. We do this to highlight the order of magnitude decrease in the objective.



The x-axis indicates the total computational cost estimated by Equation 4. We compute the mean (bold line) and standard deviation (shaded region) of the objective histories at regular cost intervals along the x-axis^d. In all subsequent plots, the individual objective histories are not shown, except when directly relevant.

Figure 6: Example of a randomized study with computed statistics.

6 Calibrating the Trigger

The *Trigger*(\cdot) function in Algorithm 2 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 7. The two branches show the performance impact of triggering at different times, for the same setup used in Figure 2. Over-optimizing on an immature parameterization leads to sluggish design improvement and is evidently a poor investment of resources. This observation has also

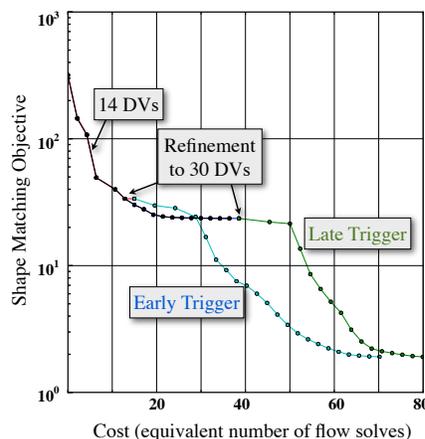


Figure 7: Effects of triggering at different times.

^dFor attainable inverse design problems like this one, the statistics are computed in log-space.

been made by other authors¹ and similar inefficiencies have been demonstrated when over-optimizing on progressive PDE meshes.¹⁵

In their work on adaptive refinement, Han and Zingg² advocate the opposite approach of allowing the optimization to converge at each level, in order to maximize design improvement under the current set of design variables. In this work, however, our goal is rapid design improvement, not to find a “minimal set” of design variables. As the cost of computing gradients for additional design variables is quite low under an adjoint formulation^e, it makes more sense to terminate the optimization as soon as the rate of design improvement starts to significantly taper off.

6.1 Detecting Tapering Design Improvement

The most direct approach to detecting diminishing design improvement is to continuously monitor the cost-slope of the objective convergence, given in Equation 5. The cost-slope must be evaluated only at major search iterations, which is monotonically decreasing. Line searches are generally non-monotone and cannot be interpreted as reasonable information about slopes.

As shown in Function 4, we terminate the optimization when the cost-slope^f falls below some fraction of the maximum cost-slope that has occurred so far. This normalization by the maximum cost-slope is essential, as it accounts for the widely differing scales that can be present in different objective functions. (For example, a drag functional may be $\mathcal{O}(10^{-2})$ while a functional based on operating range may be $\mathcal{O}(10^5)$.)

Function 4: *CostSlopeTrigger*(\cdot)

Input: Objective history $\mathcal{J}(d)$ w.r.t. search direction d , window w , slope reduction factor $r < 1$

Result: *True* = “Terminate” or *False* = “Continue”

$s = \text{ComputeAveragedSlopes}(\mathcal{J}(d), w)$

if $\left(\frac{s[-1]}{\max(s)}\right) < r$ **then** return *True*

else return *False*

Function 5 outlines an alternative approach: monitoring the objective *gradients* to the design variables, $\|\frac{\partial \mathcal{J}}{\partial \mathbf{X}}\|$, which converge to zero as the design approaches a local optimum. In convex regions of the design space, the gradient roughly indicates distance from optimality. Better

^eUnder a finite-difference optimization framework (i.e. without the adjoint), the cost of each extra gradient is two flow solutions. In that case, allowing more convergence on fewer design variables might be more efficient.

^fFor attainable inverse design problems, we measure the cost-slope in log-space to better reflect the problem.

yet, low gradient values typically *precede* flattening design improvement, allowing pre-emptive termination.

Function 5: *GradientTrigger*(\cdot)

Input: Objective gradient history $\frac{\partial \mathcal{J}}{\partial \mathbf{X}_i}(d)$ w.r.t. search direction d and design variable i , window w , reduction factor $r < 1$
Result: *True* = “Terminate” or *False* = “Continue”

$\bar{G}(d) \leftarrow \text{ComputeAveragedGradientNorms}(\frac{\partial \mathcal{J}}{\partial \mathbf{X}_i}(d), w)$

if $(\frac{\bar{G}[-1]}{\bar{G}[0]} < r)$ **then** return True

else return False

More aggressive triggering strategies could also be adopted, wherein refinement is initiated well *before* the convergence begins to taper. This could be done by refining after a predetermined number of search directions, perhaps by linking the number of search directions to the number of design variables to reflect the expected rate of convergence. This approach, however, would require prior, problem-specific knowledge about the rate of design improvement, which would defeat the purpose of a general and automated optimization algorithm.

6.2 Handling Non-smoothness

The objective-cost slopes in Function 4 and the radients in Function 5 can be non-smooth, which can cause false triggering. We alleviate this problem by using running averages over a small window. This helps prevent premature triggering, but it has the side effect of causing a lag the size of the window, which can delay the trigger for a few search directions. Therefore the window should be as small as possible. A window of width 2 proved robust enough for our studies so far.

6.3 Experimental Results

Figure 8 shows the performance of the two triggering strategies (Functions 4 and 5) on the geometric objective functional given by Equation 6. Three reduction factors are examined for each approach. There

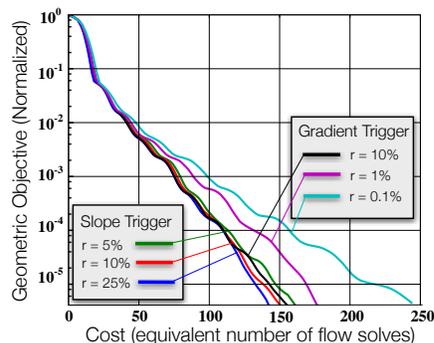


Figure 8: Performance of different triggering strategies on the geometric objective function. Each curve shows mean behavior over 10 random trials. ($\beta = 0.01, 1.5\times$ growth)

are ten individual randomly generated cases (not shown), as explained in Section 5. The same ten cases were used to test each triggering strategy. The slope-based triggering strategies show very little sensitivity to the reduction factor r in Function 4, indicating a fairly robust approach. This is probably because the slope reduction tends to be sudden for the geometric objective function. The gradient triggers are much more sensitive to the reduction factor, but still seemed to perform reasonably well for a good choice of r . Performance of the triggers on the aerodynamic inverse design functional is currently under investigation.

6.4 False Triggers

Both triggers assume that diminishing design improvement indicates that the optimizer has nearly fully exploited the design space, which should now be expanded. But this assumption is not always correct: the optimizer could be simply navigating a particularly difficult region of the design space, after which faster design improvement would continue. It is impossible to distinguish between these two cases without additional information, such as prior knowledge that a superior design is attainable. But an automated system cannot depend on problem-dependent *a priori* information. We therefore err on the side of occasionally triggering early in order to avoid the much more significant penalties associated with consistently delaying refinement.

7 The Importance Indicator

The refinement indicator is an estimate of the relative *importance* of each candidate design variable. Importance is difficult to properly estimate, because of the inherent nonlinearity of aerodynamic design.^g The importance estimate is necessarily only *locally* meaningful. Nevertheless, as there is no globally valid importance indicator until the problem is solved, local information is the best cue available.

The most significant information at hand is the local objective gradient to each candidate design variable $\frac{\partial J}{\partial \mathbf{X}}$, which we can compute for negligible additional cost as we have already computed an adjoint solution (see Section 4.2.2). Function 6 gives a method for computing the refinement indicator that values high objective gradients. To compute gradients to candidate design variables, we leverage the *ProjectGradient*(\cdot) function, which is part of the static-parameterization design framework.

^gStemming from nonlinearities in the geometry, in the shape deformation, in the flow mesh discretization, and in the flow equations themselves.

Function 6: *ComputeImportanceIndicator*(P, X, ψ)

Input: Candidate shape parameter P with value X ,
adjoint solution ψ

Result: Importance indicator I

$\frac{\partial \mathbf{S}}{\partial X} \leftarrow \text{ComputeShapeDerivative}(P, X)$

$\frac{\partial \mathcal{J}}{\partial X} \leftarrow \text{ProjectGradient}(\psi, \frac{\partial \mathbf{S}}{\partial X})$

$I \leftarrow \|\frac{\partial \mathcal{J}}{\partial X}\|$

8 Pacing Introduction of New Design Variables

After computing the importance indicators of the candidate parameters and sorting them, we have a prioritized list of design variables to add, as illustrated in Figure 4. The *SetPace*(\cdot) function in Algorithm 2 then determines how many of the candidates to add to the optimization.

There are two factors that can slow down an optimization. If the design space is insufficiently flexible, optimization will quickly stagnate. But if the design space has too much freedom, it is more difficult to navigate. Both situations cause inefficient design improvement. Selecting a pacing involves finding an effective balance between flexibility and navigability.

At each refinement, growth in the number of design variables can be specified either in absolute terms (e.g. “add 10 design variables”) or in relative terms (“increase the number of design variables by 50%”). For this initial study, we use preset growth ratios. We defer for future studies the evaluation of more sophisticated strategies, such as performing a cost-benefit estimation, or even removing some design variables from the active set for efficiency.

8.1 Experimental Results

Figure 9 compares the performance of various growth factors from 1.25–2 \times using the geometric objective functional (Equation 6). The results clearly demonstrate that the growth rate has a critical impact on efficiency. There is a factor of two difference in performance between growth rates of 1.25 \times and 2 \times . There is also a clear trend favoring faster introduction of design variables, with 2 \times refinement performing the best. Before drawing sweeping conclusions, however, we note that at least three factors could impact this result.

The first is that, although this is a 3D geometry, the shape parameterization is only 1D: the spanwise direction is the only dimension for refinement. In future work, we plan to consider 2D parameterizations that combine spanwise refinement of planform design variables with streamwise refinement of airfoil design variables. In that case, the uniform growth rate would be approximately 4 \times instead of 2 \times .

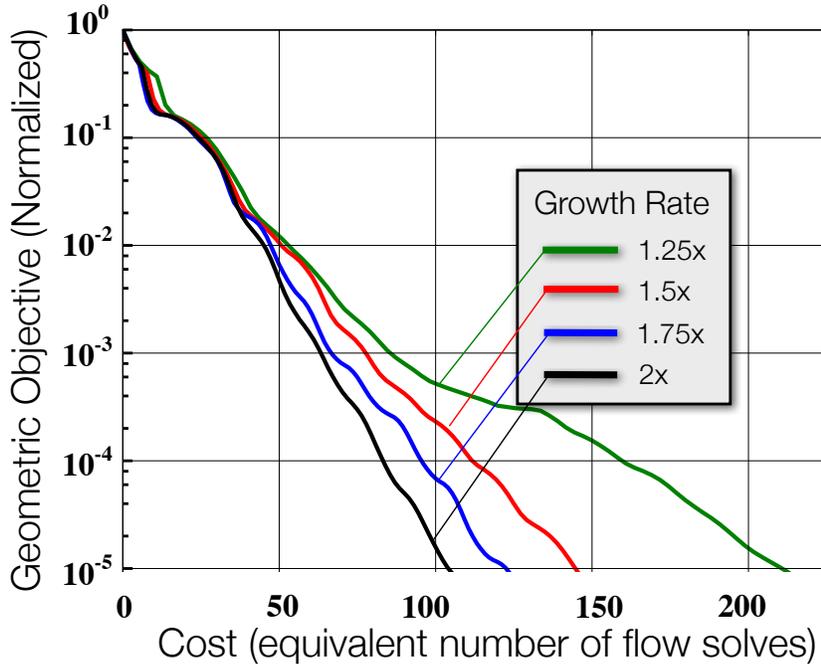


Figure 9: Performance of different growth rates on the geometric objective function. Each curve shows mean behavior over 10 random trials. ($\beta = 0.01$, slope-based trigger, $r = 0.25$)

Another possible factor is the relative simplicity of the geometric objective functional, which allows rapid and reliable design improvement, regardless of the number of design variables. We can simulate this by assuming that N_{iter} in Equation 5 is proportional to the number of design variables, which is the expected behavior for most problems. In this case, the design improvement slope would penalize the introduction of more design variables. A third consideration is that we are starting with a minimal number of design variables (1 in this case).

On the other hand, if fast growth rates do indeed prove to be generally superior, then we are not limited even by uniform refinement. If multiple candidate design variables are generated per interval (recall the discussion in Section 3.1), then we can consider growth rates in excess of $2\times$ (in excess of $4\times$ in 2D). Experimental studies to better elucidate the tradeoff are still ongoing.

9 Summary and Conclusions

We discussed how a uniform refinement strategy is an excellent stepping-stone towards fully adaptive parameterization. Uniform refinement provides automation, consistency and substantial design acceleration, and implementation is simpler than a fully adaptive framework. To highlight the savings in user time, consider the following changes to a typical designer’s workflow in setting up an optimization:

1. Provide initial design.
2. Define objective function, constraints, PDE solver settings.
3. Provide ~~detailed~~
minimal parameterization.
4. Set bounds on ~~all detailed~~
minimal set of parameters
5. ~~Repeat steps 3 and 4 iteratively to incorporate new insights.~~

What was previously a *user*-driven iterative loop of optimizations can be handled automatically using an progressive parameterization process, freeing the designer to focus on the essentials of posing the right problem.

We then focused on improving the process efficiency by adopting an adaptive approach to parameterization. Although studies are still ongoing, the experimental results above enable us to make several statements about the efficiency of adaptive schemes:

- A trigger is essential for efficiency, and earlier triggers outperform late triggers (see Figure 7).
- The growth rate is also critical for efficiency, and faster growth rates tend to outperform slow ones (see Figure 9).
- In practice, we have found that it is more efficient to start with a moderate degree of shape control, rather than the bare minimum number of design variables.^h

We gave examples of using two different geometry modelers with the adaptive framework. In Section 3 we discussed the development necessary to integrate arbitrary geometry modelers with the framework:

- A system for generating candidate design variables
- A method for automatically setting bounds on parameters

^hThis can still be handled automatically, by uniformly refining the parameterization one or more times before beginning optimization.

9.1 Ongoing Research

Our next major goal is to apply a progressive approach to parameterizations of higher dimensionality, such as simultaneous design (and refinement) of both planform design variables and airfoil design variables.

Regarding efficiency, we plan to examine several modifications to the strategy:

- Generating more candidate design variables by searching “deeper”, using multiple levels of binary refinement.
- Trigger based on cost-benefit estimate.
- Importance indicator:
 - Examining candidate *parameterizations*, instead of single candidate *parameters*.
 - Using Hessian approximation to guide selection.
- Growth rate:
 - Setting growth rate based on cost-benefit analysis
 - Removing design parameters that have low continuing importance.
- Transferring Hessian approximation between refinement levels to jumpstart next optimization.

In this study we exclusively focused on *discrete* parameter refinement, analogous to *h*-refinement in PDE solvers. In future work, we hope to examine a continuous approach, where we seek to optimally set the location of each design parameter, paralleling the PDE solver technique of adaptation through re-distribution or *r*-refinement.

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,” 2013, pp. 1–23.
- 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,” Vol. AIAA Paper 2012-0962, Nashville, TN, January 2012.
- ⁵ Jameson, A., “Aerodynamic Design via Control Theory,” *Journal of Scientific Computing*, Vol. 3, No. 3, 1988.
- ⁶ Nemeč, M. and Aftosmis, M. J., “Parallel Adjoint Framework for Aerodynamic Shape Optimization of Component-Based Geometry,” Vol. AIAA Paper 2011-1249, Orlando, FL, January 2011.

- ⁷ Gill, P. E., Murray, W., and Saunders, M. A., “SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization,” *SIAM Journal on Optimization*, Vol. 12, 1997, pp. 979–1006.
- ⁸ Samareh, J. A., “A Survey of Shape Parameterization Techniques,” *CEAS/AIAA/ICASE/NASA Langley International Forum on Aeroelasticity and Structural Dynamics*, Williamsburg, VA, June 1999.
- ⁹ Samareh, J. A., “Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization,” *AIAA Journal*, Vol. 39, No. 5, May 2001.
- ¹⁰ Kulfan, B. M., “Universal Parametric Geometry Representation Method,” *J. Aircraft*, Vol. 45, No. 1, January 2008, pp. 142–158.
- ¹¹ Berkenstock, D. C. and Aftosmis, M. J., “Structure-Preserving Parametric Deformation of Legacy Geometry,” *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, No. 2008-6026, Victoria, BC, September 2008.
- ¹² Straathof, M. H., *Shape Parameterization in Aircraft Design: A Novel Method, Based on B-Splines*, Ph.D. thesis, Technische Universiteit Delft, Netherlands, February 2012.
- ¹³ Anderson, G. R., Aftosmis, M. J., and Nemec, M., “Constraint-Based Shape Parameterization for Aerodynamic Design,” 7th International Conference on Computational Fluid Dynamics, Big Island, Hawaii, July 2012.
- ¹⁴ Anderson, G. R., Aftosmis, M. J., and Nemec, M., “Parametric Deformation of Discrete Geometry for Aerodynamic Shape Design,” Vol. AIAA Paper 2012-0965, Nashville, TN, January 2012.
- ¹⁵ Nemec, M. and Aftosmis, M. J., “Output Error Estimates and Mesh Refinement in Aerodynamic Shape Optimization,” Vol. AIAA Paper 2013-0865, Grapevine, TX, January 2013.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YYYY) 01-09-2013		2. REPORT TYPE Technical Memorandum		3. DATES COVERED (From - To)	
4. TITLE AND SUBTITLE Adaptive Shape Parameterization for Aerodynamic Design				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) George R. Anderson and Michael J. Aftosmis				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Ames Research Center Moffett Field, CA 94035				8. PERFORMING ORGANIZATION REPORT NUMBER L-	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001				10. SPONSOR/MONITOR'S ACRONYM(S) NASA	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) NASA/TM-2013-Seedling Phase 1 Final Report	
12. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified-Unlimited Subject Category 02 Availability: NASA CASI (443) 757-5802					
13. SUPPLEMENTARY NOTES An electronic version can be found at http://ntrs.nasa.gov .					
14. ABSTRACT This report concerns research performed in fulfillment of Phase 1 of an ongoing NASA Seedling Fund grant. The overall goal is to develop an aerodynamic shape optimization framework that supports automated shape parameterization. The four objectives for the work were to mature a constraint-based deformation technique, to develop the basic framework necessary to perform automated parameter refinement, to determine an importance indicator that prioritizes candidate design variables, and to develop an efficient refinement strategy. All of these objectives have been met to the degree appropriate for Phase 1. We discuss each in detail in this report.					
15. SUBJECT TERMS design,optimization,parametric,adaptive,refinement					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
U	U	U	UU		19b. TELEPHONE NUMBER (Include area code) (443) 757-5802

