Improved Dynamic Geometry Control Algorithms for Efficient Aerodynamic Shape Optimization

Gregg M. Streuber * and David. W. Zingg †
Institute for Aerospace Studies, University of Toronto, Toronto, Ontario, M3H 5T6, Canada

In aerodynamic shape optimization, traditional static geometry control methods can produce suboptimal performance by introducing performance trade-offs at various stages of optimization, enforcing arbitrary constraints on open-ended optimization, and necessitating foreknowledge of problem behaviour to design an effective control scheme. These shortcomings can be mitigated through dynamic geometry control, which partly automates the geometry control design process by refining the geometry control topology throughout optimization. Such refinement can occur in a pre-determined fashion (as in progressive control) or more automatically using sensitivity information to guide refinement (as in adaptive control). Both progressive and adaptive control are implemented in the context of axial and free-form deformation geometry control, and novel contributions are made to the adaptive algorithm, including the treatment of active constraints and several novel "potential indicators" to rank candidate refinements. Application to a wide suite of aerodynamic shape optimization problems demonstrates that dynamic geometry control is effective, producing lower final drag than well-designed static schemes while reducing required iterations to convergence by 50% or more, and simultaneously reducing labour requirements on the user. These benefits are demonstrated across a wide variety of problems, representative of detailed and exploratory problems often encountered in both academia and industry.

I. Introduction

In aerodynamic shape optimization, geometry control is the system by which the optimizer manipulates the aerodynamic surface which is being optimized. CAD-based approaches have been used previously [1], but many authors have, in more recent years, switched to CAD-free geometry control due its superior flexibility and customizability. CAD-free geometry control in the literature encompasses many different methods, including traditional [2] and axial-curve augmented [3] Free-Form Deformation (FFD) [4], B-Spline patches [5], Hicks-Henne bump functions [6], component-based control [7], radial-basis functions [8], and numerous other specialized geometry control methods

*Ph.D., gregg.streuber@utoronto.ca
†University of Toronto Distinguished Professor of Computational Aerodynamics and Sustainable Aviation, Director, Centre for Research in Sustainable Aviation, and Associate Fellow AIAA, dwz@oddjob.utias.utoronto.ca

tailored to the requirements of their users [9–12]. The geometry control, which allows the optimizer to manipulate the geometry, should be contrasted with the geometry parameterization, which parameterizes the aerodynamic surface itself. In some methods such as B-Spline patches, the two systems are identical; however, in approaches such as FFD these represent two related but distinct systems. The wide variety of available approaches to parameterizing and controlling geometry has been noted previously [13], and different approaches can offer quantitative and qualitative advantages in different applications [14, 15].

While the details of these approaches differ, they are similar in that they are overwhelmingly applied in a static fashion. That is, the geometry control, whatever its specifics, is designed by a user prior to optimization and while design variables will change in value, the number, type, and fundamental characteristics of those design variables will remain fixed. Such static systems are simple to implement and have been successfully utilized in a vast array of different settings, but they do have some inherent drawbacks. First, it can be observed that in many optimization problems the optimizer tends to begin with coarse, large-scale deformations and then steadily moves to smaller and smaller shape refinements as optimization proceeds. Having too few design variables will limit objective function improvement, while starting with too many can lead to a poorly conditioned optimization problem and poor convergence [16]. A second potential issue with static control, particularly in the context of preliminary or exploratory design, where large geometric changes are expected, is that it can be overly restrictive. Any geometry control scheme can be thought of as a second set of constraints on optimization: an optimizer can only manipulate the design variables it is given, and fitting a static geometry control system over a continuous problem inherently invalidates certain search directions and biases the optimizer towards others. In theory we desire a design space as close to continuous as possible, but given the issues with very large search spaces discussed above we are in practice forced to balance the contradictory goals of exploration and speed. Both of these issues impede automation and efficiency, first by requiring the user to be deeply involved in the design of an effective geometry control scheme and then leading to sacrifices in convergence speed or geometry quality if the scheme is sub-optimal.

The alternative to static geometry control is dynamic geometry control (DGC), in which the geometry control system is modified throughout optimization, usually through refinement. Such a system is more complicated to implement, but can address the above issues by starting the optimization in a coarse search space and refining periodically throughout. This allows the optimization to maximize initial speed with a coarse geometry control, while adding finer control later to exploit the design space. Additionally, by leveraging sensitivity information from a gradient-based optimization, it is possible to guide the refinement process and focus new control in regions of greatest potential, automating the geometry control design process and potentially leading to better performing geometries than can be achieved by even a well designed static scheme.

Two particular types of DGC are considered in this work: progressive and adaptive control. Progressive geometry control is relatively straightforward: starting with a coarse geometry control scheme the optimizer steadily moves

through a sequence of progressively finer geometry control schemes. Some early work in this field [17, 18] examined both uniform refinement schemes and multigrid-like v-cycles to improve convergence of aerodynamic shape optimization problems while others such as Andreoli et al. [19] solely consider uniform refinement. While these methods have been successfully applied in gradient-free optimization [20], they have also been shown to be effective in gradient-based optimization [21–23] where the costs of increased dimensionality are often less explicit. In general, a common trait among progressive schemes is that the geometry control systems are designed beforehand by the user, or generated at runtime according to a fixed pattern.

Adaptive control, on the other hand, leaves the design of the new geometry control system largely up to automatic processes. Duvigneau et al. [20] developed an adaptive scheme built on the idea that a Bezier curve parameterization with a more regularized control polygon will be more effective than a less-regular alternative. A more advanced approach in the context of gradient-based optimization is to leverage readily available gradient data to optimize the expected performance of the new geometry control scheme. This approach was examined by Han and Zingg [24], who assessed candidate control schemes based on the value of the objective gradient, a higher gradient suggesting a greater potential for objective improvement. Masters et al. [25] produce good results using a similar method for 2-D airfoil optimization with subdivision curves for parameterization. He et al. [26] built further on the gradient-ranking concept in 2-D by using constraint gradients to penalize the objective gradient in the ranking. The current state of the art in the field, however, is represented by the work of Anderson and colleagues [22, 27, 28], who approximated the objective function in each candidate design space as a quadratic function subject to linearized constraints and took the constrained minimizer of this quadratic as an estimate of its maximum objective improvement, the first attempt in the literature to directly approximate the shape of a candidate design space for the purposes of adaptive refinement. This approach was adapted by Sinsay and Alonso [29], who used a genetic algorithm to determine the optimal refinement strategy.

In this paper we advance the state of the art of adaptive geometry control by building on the quadratic fit approach of Anderson and colleagues with improvements to the treatment of constraints and a novel approach to the approximation of candidate Hessians. Moreover, a novel intermediate step between the pure gradient ranking of earlier works and the quadratic fit approach of Anderson and colleagues is also developed, using a linear fit of the objective and constraints to formulate the indicator calculation as a linear programming problem, and various approaches to ranking candidate design spaces are compared. Finally, a significant effort is made to study and quantify the benefits and applicability of DGC through application of the developed progressive and adaptive algorithms to a variety of problems including several simple verification problems and a pair of cases based on the Reynolds-averaged Navier-Stokes equations representative of detailed-design and exploratory optimization problems commonly encountered in the literature.

II. Jetstream Optimization Framework

This study is undertaken using the Jetstream optimization framework, which is described below, with particular attention given to the FFD geometry control system.

A. Optimization Overview

Flow solutions in Jetstream are obtained through Diablo, a three-dimensional structured multiblock CFD solver capable of solving both the Euler [30] and RANS [31] equations; for the RANS equations, the Spalart-Allmaras one-equation turbulence model is used. Second-order summation-by-parts operators and scalar or matrix numerical dissipation are used for spatial discretization with simultaneous approximation terms weakly enforcing boundary and block interface conditions. The steady-state solution is obtained with a parallel Newton-Krylov-Schur algorithm utilizing an approximate-Newton phase for the initial iterate of a subsequent inexact-Newton phase. Linear systems are solved using the Generalized Minimum Residual (GMRES) method [32].

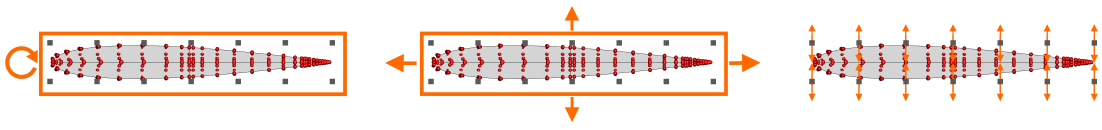
Gradients are obtained using an implementation [5, 33] of the discrete adjoint method [34]. A flexible variant [5] of the GCROT (generalized conjugate residual with inner orthogonalization and outer truncation) Krylov method [35] is leveraged to obtain adjoint solutions. These gradients are used to obtain the optimal geometry using the sequential-quadratic-programming algorithm SNOpt [36], with Hessians obtained via the Broyden–Fletcher–Goldfarb–Shanno (BFGS) [37–40] update method.

A linear elasticity method adapted by Hicken and Zingg [5] from the work of Truong et al. [41] handles mesh movement. Computational performance, both of the mesh adjoint problem and the mesh deformation itself, is improved by fitting the computational mesh with a coarse B-spline control mesh, the mesh deformation being applied to this control mesh, which is then used to generate a deformed computational mesh.

B. Static Free Form Deformation

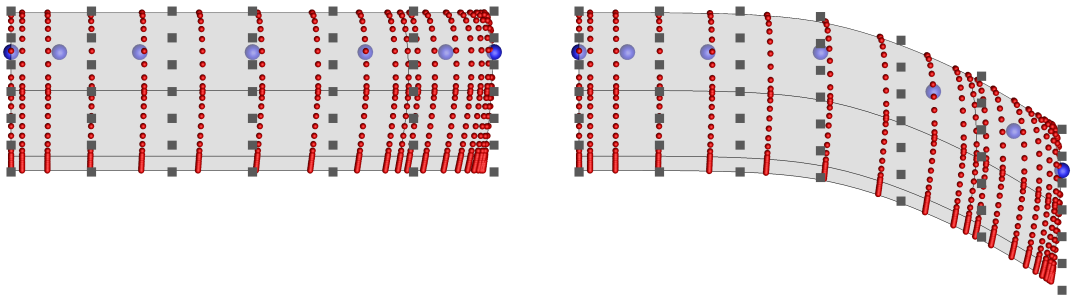
Free-Form Deformation (FFD) [4], is a geometry control scheme which introduces an additional layer of abstraction between the surface and the design variables and enables consistent, intuitive design variables to be used for almost any type of surface geometry. FFD works by embedding the surface to be deformed within an FFD volume; the optimizer then manipulates the FFD volume without needing to have any knowledge of what lies within it. As the FFD volume is deformed, the underlying geometry is deformed in turn - akin to manipulating a rubber block with a shape embedded inside. While it first gained traction with the animation community, the first publicly referenced use of FFD in aerodynamic shape optimization dates to 2004 [42].

In this work, the embedded geometry is parameterized with B-spline surface patch control points and the FFD volumes they are embedded within are themselves B-spline volumes, deformed by displacing the control points on their surface. The control points are arranged into cross-sections which provide local control through three degrees of



(a) Twist control (b) Taper control (c) Section control

Fig. 1 Cross-sectional design variables

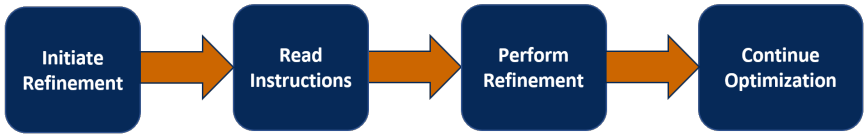


(a) Undeformed (b) Deformed

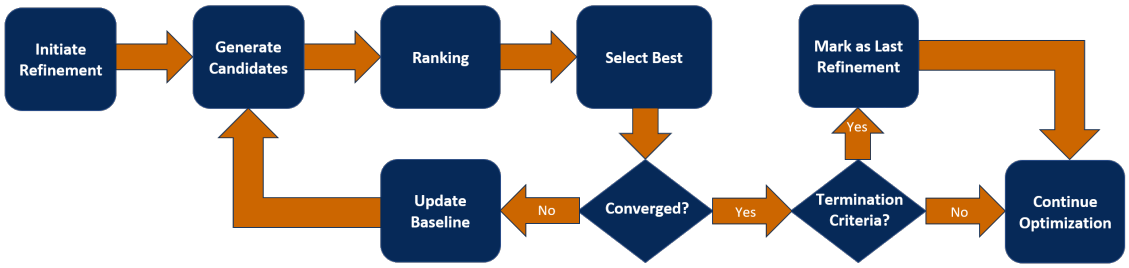
Fig. 2 Axial deformation

freedom (DOFs) illustrated in Figure 1: twist, a bulk rotation of the cross-section about the local origin; taper, a bulk scaling of the chord with fixed t/c; and section, individual movement of cross-sectional control points in the local vertical direction. Global control is provided by an axial curve [43], a B-spline curve which determines the orientation of the FFD cross-sections. The axial curve itself is controlled with its own set of control points, each having up to three DOFs: sweep (translation in the flow axis), span (translation in the span axis), and dihedral (translation in the vertical direction). Figure 2 demonstrates global deformation for a simple rectangular wing. Red spheres represent the embedded surface patch control points, black cubes are the FFD control points, and blue spheres are the axial control points. Starting from the undeformed geometry in Figure 2a, sweep is added by pulling the outboard control points back as per Figure 2b; as the axial curve is deformed, each cross-section is translated or rotated to maintain its position and orientation relative to the curve, in turn deforming the FFD volume, embedded surface control points, and then the surface itself.

Finally, we clarify the nomenclature employed in this work. The FFD system consists of three modes of deformation corresponding to the three types of structure that the optimizer may manipulate: axial control points, cross-sectional control points, and cross-sections (which, while composed of cross-sectional control points, can be deformed independently of the individual points). Each mode of deformation may have one or more DOFs: cross-sectional control points may only have a single DOF (vertical position relative to the cross-section), cross-sections may have up to two (twist and taper), and axial control points may have up to three (sweep, span, and dihedral). Design variables (DVs) refer to the individual values the optimizer may manipulate during optimization; a single DOF, for instance section shape for a cross-section, may encompass multiple DVs.



(a) Progressive algorithm



(b) Adaptive algorithm

Fig. 3 Dynamic geometry control algorithms

III. Dynamic Geometry Control

Figure 3 demonstrates the structure of the DGC algorithms. Optimization begins in a baseline geometry control scheme of minimal resolution. Refinement is initiated by a refinement criterion (Section III.A) which is designed to detect when the optimizer has achieved a significant percentage of the possible objective improvement in the current design space. From there the progressive algorithm is straightforward: the user-defined instructions are read in from a file, the desired points are added to the geometry control scheme, and optimization proceeds. This process is repeated a pre-determined number of times, and once the final requested refinement has been completed, the optimizer will iterate in the current design space until it exits or the user requests termination.

In the adaptive algorithm, once refinement is initiated the algorithm generates a number of candidate design spaces (Section III.B), each one a particular refinement of the current geometry control scheme. The candidates are then ranked based on an estimate of their potential and the best candidate is accepted. We define a design space's "potential" as the maximum feasible objective reduction that it permits in practice; each candidate design space, regardless of how many additional design variables it includes, will have a single potential value and several methods of obtaining this estimate are discussed in Section III.C. This includes two indicators based on a novel approach to the quadratic approximation method of Anderson[28] and an entirely novel approach utilizing the simplex method applied to a linear model of the candidate design space. Two termination criteria, described in Section III.A, are used to control the behaviour of the adaptive algorithm. The local termination criterion is checked after each candidate is accepted and determines whether to continue refining or proceed with optimization. The global criterion is checked after the local criterion is satisfied and controls whether this will be the final design space to explore. Optimization proceeds in this cycle of optimization-exhaustion-refinement until the global termination criterion is met, at which point optimization proceeds in the final design space until the user is satisfied that all optimization convergence criteria have been satisfied.

A. Initiating and Terminating Refinement

Several different approaches to initiating refinement are present in the literature. Han and Zingg [24] converge the problem fully within each design space, whereas Anderson and Aftosmis [27] initiate refinement when it appears that objective improvement has become asymptotic. Masters et al. [21] use a similar method, but employ several rolling averages to smooth out the refinement switch and include both the objective and constraints in the formulation. Anderson and Aftosmis show that fully converging, as in Han and Zingg, tends to be less efficient than their method, and so we employ similar ideas in constructing our criterion. The refinement criterion implemented here initiates refinement when:

$$\begin{aligned}
&\frac{\Delta\overline{\mathcal{M}}^{(i)}}{\Delta\overline{\mathcal{M}}_{\text{locmax}}} \leq r_{\text{loc}} \text{ for } n_{\text{loc}} \text{ consecutive iterations, or} \\
&\frac{\Delta\overline{\mathcal{M}}^{(i)}}{\Delta\overline{\mathcal{M}}_{\text{glbmax}}} \leq r_{\text{glb}} \text{ for } n_{\text{glb}} \text{ consecutive iterations,} \\
&\Delta\overline{\mathcal{M}}^{(i)} = \frac{\mathcal{M}^{\left(i-n_{\text{avg}}\right)} - \mathcal{M}^{(i)}}{n_{\text{avg}}},\ \Delta\mathcal{M}^{(i)} = \mathcal{M}^{(i-1)} - \mathcal{M}^{(i)},
\end{aligned} \tag{1}$$

where $\mathcal{M}^{(i)}$ is the merit function (the value of a Lagrangian function that includes the objective function and nonlinear constraints) at the i^{th} iteration, $\Delta\overline{\mathcal{M}}^{(i)}$ is the rolling average of the merit reductions over the last n_{avg} iterations, $\Delta\overline{\mathcal{M}}_{\text{max}}$ is the maximum value of $\Delta\mathcal{M}^{(i)}$ seen since the previous refinement (locmax) or the beginning of optimization (glbmax) and $0 < r < 1$ and $n > 0$ are user-defined parameters. Using the merit function rather than the objective function to initiate refinement simplifies the implementation as it is normally monotonically reducing.

We have augmented Anderson's criterion by separating it into local and global components. The local criterion checks for exhaustion of the current design space, while the global criterion checks whether the optimizer has become stuck in a design space with minimal potential. In practise, the former criterion dominates in early design spaces while the latter criterion is often more active later in the design process when merit gains are more likely to be marginal.

A third condition is also enforced to prevent premature refinement. This condition states that regardless of the status of the criteria in equation 1, refinement will not be initiated unless

$$\frac{1}{n_{\text{avg}}} \sum_{j=0}^{n_{\text{avg}}-1} \frac{\Delta\mathcal{M}^{(i-j)}}{\mathcal{M}^{(i-j-1)}} \leq r_{\text{cut}}, \tag{2}$$

where r_{cut} is the cut-off threshold. This equation is identical to the rolling average merit reduction calculation for the i^{th} iteration, but with each merit drop scaled by the merit at the previous iteration. This then expresses by what percent the merit function is dropping each iteration, and r_{cut} is the minimum per-iteration percent reduction above which the design space will be considered non-asymptotic, even if either or both of the previous criteria are satisfied.

For adaptive geometry control, refinement is terminated when:

$$\begin{aligned} &\frac{\Delta\mathcal{A}_i}{\Delta\mathcal{A}_{\max}} \leq a_{\text{loc}} \text{ for } m_{\text{loc}} \text{ consecutive candidates,} \\ &\Delta\mathcal{A}_i = \mathcal{A}_i - \mathcal{A}_{\text{BL}} \end{aligned} \tag{3}$$

where $\mathcal{A}_i$ is the potential of the i^{th} candidate to be added to the design space and $\mathcal{A}_{\text{BL}}$ is the potential of the design space without i. As a reminder we define the potential of a candidate design space to be the maximum reduction in objective that it enables beyond the current point. We define $\Delta\mathcal{A}$ as the "marginal utility" of a candidate, $\Delta\mathcal{A}_{\max}$ is the largest such utility found so far during the current refinement, and $0 < a_{\text{loc}} < 1$ and $m_{\text{loc}} > 0$ are user-defined parameters. The values of $\mathcal{A}$ are estimates of the objective reduction that a given design space would permit beyond the current point; this is the metric by which the adaptive candidates are ranked and its calculation is discussed in Section III.C. This criterion then ends refinement when the expected benefits of further refinement drop below some user-defined threshold. When the termination criterion in equation 3 is met, a second "global" refinement termination criterion is checked as well. This determines whether the optimization problem has converged with respect to refinement. The global criterion is tripped if

$$\frac{\mathcal{M}^{(0)} - \mathcal{M}^{(\text{r})}}{\mathcal{M}^{(0)}} \leq a_{\text{glb}} \text{ for } m_{\text{glb}} \text{ consecutive design spaces} \tag{4}$$

where $\mathcal{M}^{(0)}$ is the merit function at the first iteration in the design space, $\mathcal{M}^{(\text{r})}$ is the merit function at refinement, and $0 < a_{\text{glb}} < 1$ and $m_{\text{glb}} > 0$ are user-defined parameters. This criterion states that if the total reduction in merit across the last few design spaces is below some user-defined percentage, then the problem has converged. If satisfied, optimization is not terminated, but rather a note is made to prevent any further refinements and the optimizer is permitted to run in the current design space until convergence criteria are satisfied, SNOPT exits of its own accord, or the user terminates the job.

Typical values for the parameters in equations 1 to 4 are given in Table 1. Overall, performance of the progressive geometry control was found to be largely insensitive to tested values of r_{loc} and r_{glb} within the posted ranges. However, for high dimensional problems and at the higher or lower ends of the provided ranges, performance can become more variable. In unknown design spaces, to obtain consistent results we suggest using values near the mean of the posted limits. Similarly, performance was fairly uniform for differing values of a_{loc}, but we suggest a conservative approach is to select values near the midpoint of the posted range.

B. Candidate Generation

For adaptive geometry control we require a list of candidate refinements from which to choose. We again adapt from the work of Anderson and colleagues by achieving this through the use of a binary search tree, as illustrated in Figure 4. The baseline level is composed of a list of all existing control points, and each lower level is then formed

Table 1 Typical values for DGC parameters

Parameter	Typical Range
r_{loc}	0.01 - 0.20
r_{glb}	0.001 - 0.020
r_{cut}	0.0005
n_{avg}	5
n_{loc}	5
n_{glb}	25
a_{loc}	0.05 - 0.50
a_{glb}	0.005
m_{loc}	2
m_{glb}	3

by inserting a new control point halfway between the control points at the previous level (while retaining all existing points). In this way a finite number of candidates can be produced at any level, while still approaching a continuous geometry control scheme as the depth goes to infinity. In our refinement scheme, no candidate can be considered until at least one predecessor control point has been added, for instance in Figure 4, candidates 2(1) and 2(2) could not be considered until candidate 1(1) was added.

Once the candidates are generated, each one is ranked by its indicator, discussed in Section III.C, the best candidate is added to the baseline, and the process is repeated with the updated baseline until the termination criterion, Section III.A, is satisfied. By default, the algorithm may only examine a single control point at a time, in which case for the scenario in Figure 4 the first stage of refinement would see only two candidates: 1(1) and 1(2). These two candidates would then be ranked and the best one (say 1(2)) is added to the baseline. Assuming that the exit criterion is not satisfied, a second stage of refinement is started, in which the algorithm would now see three candidates: 1(1), 2(3), and 2(4), with the latter two now being available by virtue of the addition of 1(2). As a final note, the children of a given candidate are geometrically constrained by the position of that candidate's parents. For instance, every child of 2(1), no matter how deep, must lie between B(1) and 1(1). If there is a desirable geometry requiring control between 1(1) and B(2), then 2(2) and its children must be added to generate it.

C. Adaptive Indicator

Each adaptive candidate consists of one or more design variables that has been added to the existing design space, producing an enriched design space whose DV vector we denote as $\widehat{\mathbf{X}}$. These candidates are generated using the method in Section III.B and then must be ranked. This ranking is accomplished using an "adaptive indicator", broadly speaking a measure of the expected objective function reduction enabled by each candidate. While each takes a different approach to finding this value, all of the indicators discussed here can be loosely categorized as "non-fitting", "linear",

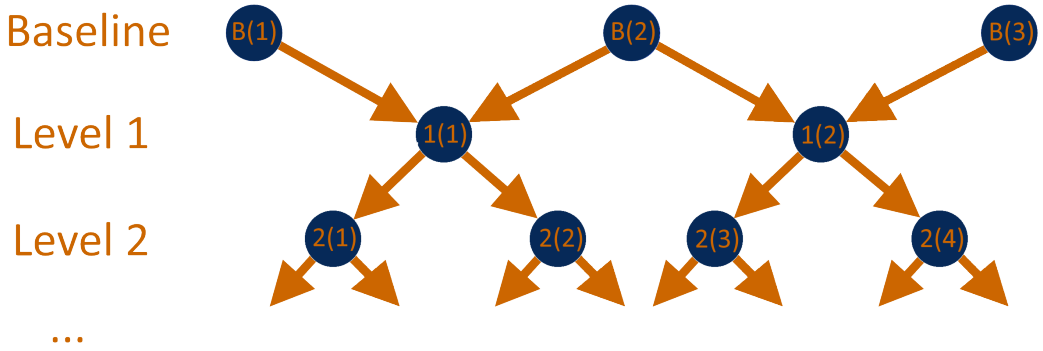


Fig. 4 Candidate generation binary search tree

or “quadratic" based on how they model the design space. In every case we are seeking a single scalar value that encapsulates the projected ability of the optimizer to reduce the objective in the new design space.

One commonality is that all indicators discussed here require gradients within the candidate design space. In gradient-based optimization, the gradient of any function $\frac{\partial \mathcal{F}}{\partial \mathbf{X}}$ will be readily available from the previous function evaluation, but this is the gradient in the unrefined design space and what is required is $\frac{\partial \mathcal{F}}{\partial \widehat{\mathbf{X}}}$, the gradient in the candidate design space. Fortunately, the latter can be obtained accurately and for negligible cost. Within Jetstream, function gradients are calculated as

$$\frac{\partial \mathcal{F}}{\partial \mathbf{X}} = \frac{\partial \mathcal{F}}{\partial \mathbf{b}} \frac{\partial \mathbf{b}}{\partial \mathbf{X}}, \tag{5}$$

where $\mathbf{b}$ are the embedded surface control points, $\mathbf{X}$ are the FFD design variables, and $\mathcal{F}$ is any functional. The expensive components of the gradient calculation, including the adjoint solutions, are contained within $\frac{\partial \mathcal{F}}{\partial \mathbf{b}}$, while $\frac{\partial \mathbf{b}}{\partial \mathbf{X}}$ is purely geometric and may be calculated analytically or estimated cheaply and to arbitrary accuracy with the complex step method [44]. Refining the FFD invalidates the latter term as $\frac{\partial \mathbf{b}}{\partial \mathbf{X}} \neq \frac{\partial \mathbf{b}}{\partial \widehat{\mathbf{X}}}$; however, $\frac{\partial \mathcal{F}}{\partial \mathbf{b}}$ is a function of the surface parameterization, not the FFD, and so will remain constant if the geometry does not deform during refinement. Therefore, since our refinement method does not modify the surface parameterization, we can save the embedded gradients from the current design space and for each candidate calculate the candidate gradients as

$$\frac{\partial \mathcal{F}}{\partial \widehat{\mathbf{X}}} = \frac{\partial \mathcal{F}}{\partial \mathbf{b}} \frac{\partial \mathbf{b}}{\partial \widehat{\mathbf{X}}}, \tag{6}$$

for only the negligible cost of evaluating $\frac{\partial \mathbf{b}}{\partial \widehat{\mathbf{X}}}$.

1. Linear Indicator

The linear indicator approximates potential using a linear approximation of each candidate design space, and to the authors’ knowledge, such an approach to adaptive control has not been previously attempted in the literature. The

benefit of such a linear model is that it permits a natural treatment of the relationships between objective function and constraints in each candidate design space, while requiring only easily obtainable gradient data. To find a linear estimate of potential for each candidate, we form a linear programming problem from linear fits of the objective and each constraint:

$$\begin{aligned} \text{minimize} \quad & \Delta\mathcal{J} = \left(\frac{\partial \mathcal{J}}{\partial \widehat{\mathbf{X}}} \cdot \Delta\widehat{\mathbf{X}}\right) \\ \text{Subject to} \quad & C_{a,i}^{(0)} + \left(\frac{\partial C_{a,i}}{\partial \widehat{\mathbf{X}}} \cdot \Delta\widehat{\mathbf{X}}\right) = 0, \quad i = 1, 2, \ldots \end{aligned} \tag{7}$$

where $\mathcal{J}$ is the objective, $\widehat{\mathbf{X}}$ are the design variables in the candidate design space, and C_a are the active constraints. If the candidate permits a further decrease in objective, we expect $\Delta\mathcal{J}$ to be a negative value and so we define our potential $\mathcal{A} = -\Delta\mathcal{J}$. The only requirement here is that every design variable be bounded in both directions; otherwise a lower bound on the objective may not exist. In such cases a quadratic indicator (discussed in Section III.C.2) is necessary. In practise this rarely occurs, so assuming that the number of constraints and design variable bounds is equal to or greater than the number of design variables, we define the linear indicator as:

$$-\mathcal{A}_{\text{LI}} := \begin{cases} \text{minimum} \ \left(\frac{\partial \mathcal{J}}{\partial \widehat{\mathbf{X}}} \cdot \Delta\widehat{\mathbf{X}}\right) \\ \text{subject to} \ C_{a,i}^{(0)} + \left(\frac{\partial C_{a,i}}{\partial \widehat{\mathbf{X}}} \cdot \Delta\widehat{\mathbf{X}}\right) = 0, \quad i = 1, 2, \ldots \end{cases} \tag{8}$$

Equation 8 is a classical example of a linear programming problem, and can be reliably solved with several different methods. For this work we choose the simplex method [45] due to its simplicity and reliability. Some pre-processing is necessary to ensure the problem is in standard form prior to solving, including introducing slack variables to rewrite all constraints as equality constraints, and a change of variable to ensure all variables have a lower bound of $x_i \geq 0$, but this is not computationally intensive. A detailed description of the simplex method is beyond the scope of this work, but a full discussion of the implementation used here can be found in Nocedal and Wright [46].

2. Quadratic Indicators

Quadratic indicators use a quadratic fit of the objective subject to a linear fit of each constraint. This defines a quadratic programming problem, the solution of which is an estimate of the maximum feasible objective reduction in the candidate design space. The operation of a quadratic indicator is illustrated in Figure 5 for a simple case with two design variables. Refinement occurs at the point $\mathcal{J}_{\text{refine}}$, about which the quadratic fit (shown in blue) is built. A linearized constraint is shown as the orange surface and together with the quadratic fit defines $\mathcal{J}_{\text{min}}$, an estimate of the minimum feasible objective value achievable in this design space. The lower $\mathcal{J}_{\text{min}}$, the more promising a candidate. This approach theoretically improves on the linear fit by retaining a more realistic treatment of constraints while dispensing with the

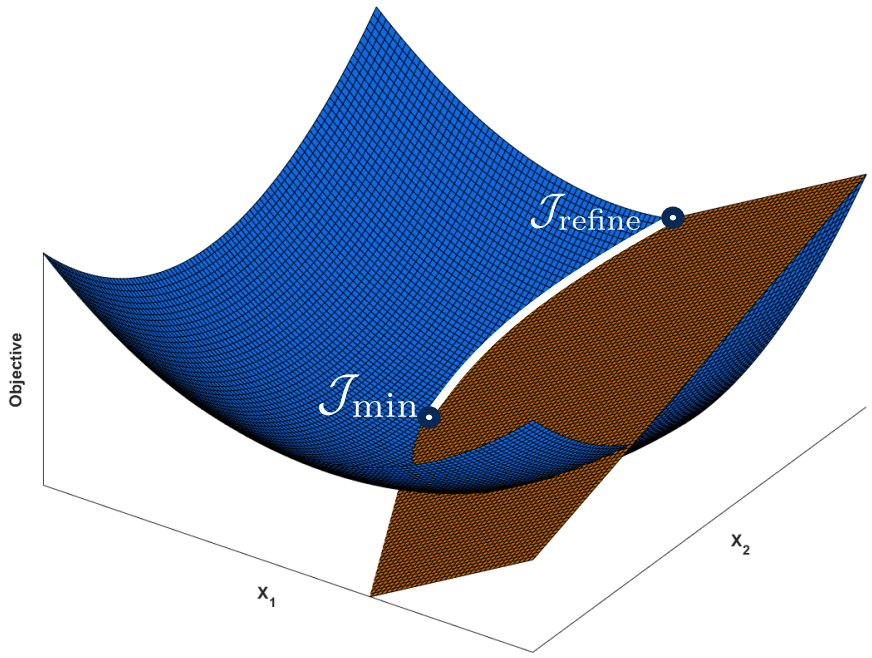


Fig. 5 Quadratic fit of candidate design space. Quadratic objective in blue, linearized constraints in orange

unrealistic assumption that the design space is linear. However, the necessity of acquiring second order sensitivity information to build the model in each candidate design space introduces additional complexity.

The use of quadratic indicators was pioneered and utilized by Anderson and colleagues [22, 27, 28] and we build upon their work here. Concretely, the quadratic indicator forms a quadratic programming problem of the form:

$$\begin{aligned} \text{minimize} \quad & \Delta\mathcal{J} = \left(\frac{\partial\mathcal{J}}{\partial\widehat{\mathbf{X}}}\cdot\Delta\widehat{\mathbf{X}}\right) + \frac{1}{2}\left(\Delta\widehat{\mathbf{X}}\cdot\widehat{H}\Delta\widehat{\mathbf{X}}\right) \\ \text{Subject to} \quad & C_{a,i}^{(0)} + \left(\frac{\partial C_{a,i}}{\partial\widehat{\mathbf{X}}}\cdot\Delta\widehat{\mathbf{X}}\right) = 0, \quad i = 1, 2, \ldots \end{aligned} \tag{9}$$

where all variables are defined as in equation 7 except for the new variable $\widehat{H}$, which is the Hessian of the candidate design space, and as before we define our potential as $\mathcal{A} = -\min(\Delta\mathcal{J})$. The greatest cost in solving equation 9 is obtaining $\widehat{H}$, or more often some approximation $\widehat{B} \approx \widehat{H}$. The selection of approximation has a large impact on the behaviour of the resulting model, and we will examine several different approaches, each defining a distinct quadratic indicator.

The simplest approach to the Hessian, which was examined by Anderson and Aftosmis [27], is to assume that it is the identity matrix. This has the advantage of solely using cheap and accurate first-order sensitivity data, while retaining a second-order fit fit of the objective. We denote this indicator as $\mathcal{A}_{\text{QI}}$ and it is obtained by simplifying equation 9 to

$$-\mathcal{A}_{\text{QI}} := \begin{cases} \text{minimum} \quad \left(\frac{\partial \mathcal{J}}{\partial \widehat{\mathbf{X}}} \cdot \Delta\widehat{\mathbf{X}}\right) + \frac{1}{2}\left(\Delta\widehat{\mathbf{X}} \cdot \Delta\widehat{\mathbf{X}}\right) \\ \text{subject to} \quad C_{a,i}^{(0)} + \left(\frac{\partial C_{a,i}}{\partial \widehat{\mathbf{X}}} \cdot \Delta\widehat{\mathbf{X}}\right) = 0, \quad i = 1, 2, ... \end{cases} \tag{10}$$

This must be done with caution, as assuming that $\widehat{B} = I$ is not removing the Hessian from the equation, but merely assuming a specific set of values for it; depending on the problem scaling an identity Hessian can have very different implications for the shape of the resulting quadratic. Therefore, it is advisable to scale the identity matrix to more closely match it to the scale of the problem at hand. If refinement is occurring after the i^{th} design iteration, we propose a novel approach which scales the identity Hessian as

$$\begin{aligned} \widehat{B} &= \nu I \\ \nu &= \sqrt{\frac{\sum_j \left(\frac{\partial \mathcal{J}^{(i)}}{\partial \mathbf{X}_j} - \frac{\partial \mathcal{J}^{(i-1)}}{\partial \mathbf{X}_j}\right)^2}{\sum_j \left(\mathbf{X}_j^{(i)} - \mathbf{X}_j^{(i-1)}\right)^2}} \end{aligned} \tag{11}$$

where I is the identity matrix, $\mathbf{X}_j$ is the j^{th} entry of the design variable vector, and the superscript $^{(i)}$ denotes the i^{th} design iteration. This scaling factor is commonly used to initialize iterative Hessian approximations such as the BFGS method [46] and is often sufficient to ensure the Hessian is of the correct order of magnitude. Substituting this into equation 9 and simplifying, we find the equation for the scaled identity indicator, denoted as $\mathcal{A}_{\text{QIS}}$, is:

$$-\mathcal{A}_{\text{QIS}} := \begin{cases} \text{minimum} \quad \left(\frac{\partial \mathcal{J}}{\partial \widehat{\mathbf{X}}} \cdot \Delta\widehat{\mathbf{X}}\right) + \frac{\nu}{2}\left(\Delta\widehat{\mathbf{X}} \cdot \Delta\widehat{\mathbf{X}}\right) \\ \text{subject to} \quad C_{a,i}^{(0)} + \left(\frac{\partial C_{a,i}}{\partial \widehat{\mathbf{X}}} \cdot \Delta\widehat{\mathbf{X}}\right) = 0, \quad i = 1, 2, ... \end{cases} \tag{12}$$

The scaled identity matrix approach is simple and inexpensive, but of low accuracy. Many gradient-based optimization algorithms build a Hessian approximation as part of their operation which could be leveraged for the quadratic fit. However, not all algorithms permit the user to extract the Hessian and so a more general approach to building higher-accuracy Hessian approximations is desirable. Later work by Anderson [28] suggests the use of a BFGS approximation of the Hessian, achieved by first obtaining a BFGS approximation of the Hessian in the current design space (B) and then prolonging the diagonal of this approximation into each candidate design space to locate $\widehat{B}$. This method has minimal overhead, but only transfers the Hessian diagonals and relies on objective-specific prolongation operators. While this method is shown to be effective in Anderson's work, we would like to examine whether an objective-agnostic approach requiring fewer assumptions is possible.

For this reason we have developed a novel approach to obtaining the Hessian in each candidate design space which does not require the objective-specific operators employed by previous authors, and which also permits the entire candidate Hessian to be obtained, instead of just the diagonal. This method can be applied to any iterative Hessian

approximation, and we provide two variants here, the first based on the BFGS approximation and the other the Symmetric Rank-One (SR1) [46] approximation. The BFGS approach is ideal for optimization, as when implemented as part of a quasi-Newton optimization algorithm it guarantees a symmetric positive-definite (SPD) Hessian matrix. The SR1 method has no such guarantee, but as there is no guarantee that the exact Hessian is SPD at an arbitrary point in the design space, this may result in a more accurate approximation. Substituting these Hessian approximations into equation 9 yields the equations for the last two second-order indicators examined in this work:

$$-\mathcal{A}_{\mathrm{QB}} := \begin{cases} \text{minimum } \Delta\mathcal{J} = \left(\frac{\partial\mathcal{J}}{\partial\widehat{\mathbf{X}}} \cdot \Delta\widehat{\mathbf{X}}\right) + \frac{1}{2}\left(\Delta\widehat{\mathbf{X}} \cdot \widehat{B}^{\mathrm{B}}\Delta\widehat{\mathbf{X}}\right) \\ \text{subject to} \quad C_{a,i}^{(0)} + \left(\frac{\partial C_{a,i}}{\partial\widehat{\mathbf{X}}} \cdot \Delta\widehat{\mathbf{X}}\right) = 0, \quad i = 1, 2, \ldots \end{cases} \tag{13}$$

$$-\mathcal{A}_{\mathrm{QR}} := \begin{cases} \text{minimum } \Delta\mathcal{J} = \left(\frac{\partial\mathcal{J}}{\partial\widehat{\mathbf{X}}} \cdot \Delta\widehat{\mathbf{X}}\right) + \frac{1}{2}\left(\Delta\widehat{\mathbf{X}} \cdot \widehat{B}^{\mathrm{R}}\Delta\widehat{\mathbf{X}}\right) \\ \text{subject to} \quad C_{a,i}^{(0)} + \left(\frac{\partial C_{a,i}}{\partial\widehat{\mathbf{X}}} \cdot \Delta\widehat{\mathbf{X}}\right) = 0, \quad i = 1, 2, \ldots \end{cases} \tag{14}$$

where $\widehat{B}^{\mathrm{B}}$ denotes the BFGS Hessian approximation and $\widehat{B}^{\mathrm{R}}$ the SR1 approximation.

Our approach differs from Anderson's [28] in that, rather than applying the algorithms in the current design space and then attempting to prolong the Hessian into each candidate space, we instead seek to directly obtain an approximate Hessian in each candidate design space by applying the algorithm within it. To accomplish this, during optimization the surface gradient of each function $\frac{\partial\mathcal{F}}{\partial\mathbf{b}_i}$ at each iteration i is saved. Once refinement has been initiated and a given candidate created by refining the geometry control, at each of these iterations the algorithm solves the geometric shape-matching subproblem

$$\underset{\widehat{\mathbf{X}}^{(i)}}{\text{minimize}} \quad \sum_j \left[b_j(\mathbf{X}^{(i)}) - b_j(\widehat{\mathbf{X}}^{(i)})\right]^2, \tag{15}$$

where b are the coordinates of the embedded surface patch control points, $\mathbf{X}^{(i)}$ is the DV vector in the unrefined design space at the i^{th} iteration, and $\widehat{\mathbf{X}}^{(i)}$ is the corresponding vector in the candidate design space at the same iteration. If the design spaces are perfectly embedded (that is, every surface in the original design space is contained within the refined design space) then the theoretical minimum of this problem is zero. Due to the requirements of the refinement process, perfect embedding cannot be mathematically guaranteed and finite precision makes such deep convergence impossible to achieve regardless. However in practice, the optimizer shows little difficulty in converging to an average point deviation of 10^{-7} or less. The result of this solution is a design variable vector $\widehat{\mathbf{X}}_i^*$ in the candidate design space which produces the same surface, and therefore same aerodynamic solution, as was produced at the i^{th} iteration in the unrefined design space. Once this has been located, equation 6 is leveraged to obtain $\frac{\partial\mathcal{J}}{\partial\widehat{\mathbf{X}}_i}$ and the resulting design variable vector and gradient pair are fed into the BFGS or SR1 approximations, producing an approximation of the Hessian in the candidate design space. This is in effect retroactively applying the BFGS and SR1 Hessian approximations to the candidate design

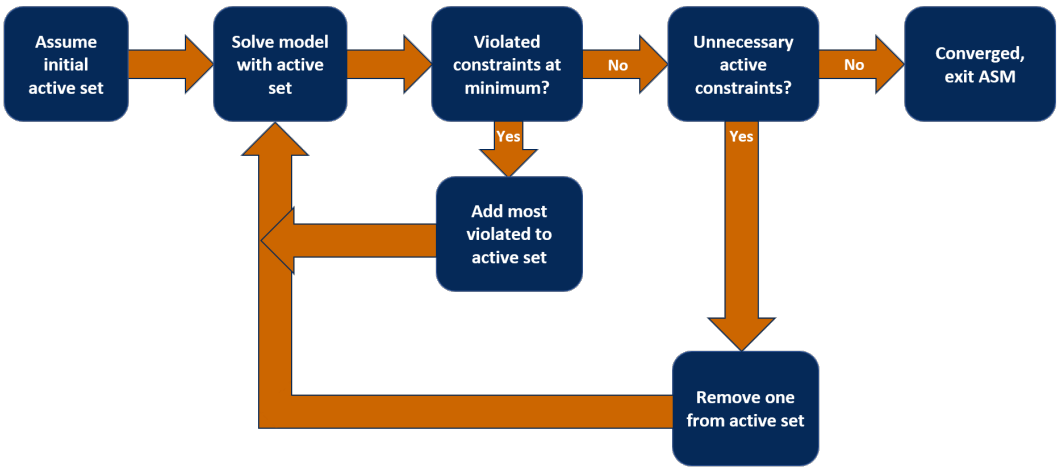


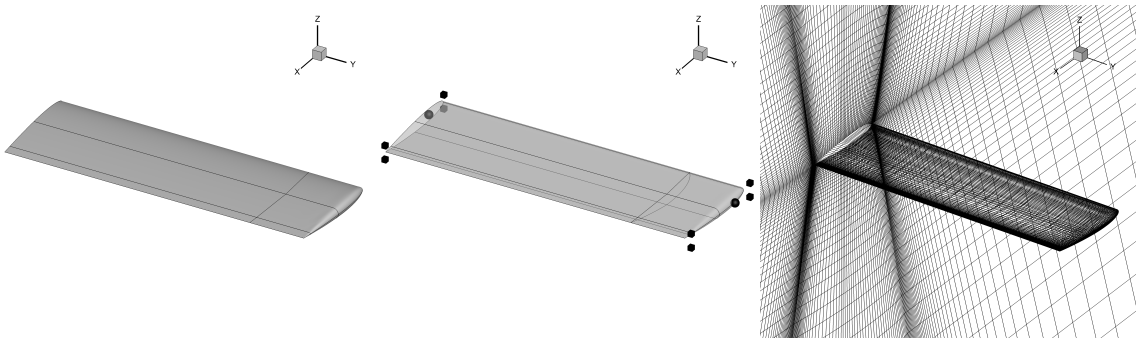
Fig. 6 Active set method

space as if they had always been in use, and for this reason we refer to this approach as the “retroactive Hessian" method.

D. Obtaining the Active Set

In all presented indicators, it is necessary to determine which constraints are active at the minimum to properly estimate potential. For the linear indicator this is done automatically as part of the solution of the simplex method; however, for the quadratic indicators this must be done manually. In previous quadratic fit methods [28], it was assumed that the active set at the minimum of the quadratic fit was equal to the active set at the point of refinement, i.e. $C^a_{\text{refine}} = C^a_{\text{min}}$. The assumption of a fixed active set has also been made by other authors who have included constraints in their indicators (though without fitting the design space) [26]. This is convenient as it makes the active set simple to obtain and in the case of quadratic indicators allows the quadratic programming problem in equation 9 to be solved in a single iteration. However, unless the minimum in the refined search space is very close to the point of refinement, it is unlikely that the active sets at both points are equal. Since it can be argued that the purpose of adaptive geometry control is to find a new search space whose minimum is as far as possible from the current point, it is likely that this assumption is invalid.

We propose an alternative solution by noting that the linearized constraints enforced in equation 9 allow us to estimate what the value of any constraint, not merely those already active, would be at a new point in the design space, permitting the use of an active set method (ASM). The ASM, as shown in Figure 6, consists of solving equation 9 subject to a given set of constraints, updating the active set to add violated constraints or remove unnecessary ones, solving the problem again subject to the new active set, and repeating the process until there are no more constraints to add or remove. This requires that equation 9 be solved multiple times, but this is a negligible cost and permits a greater degree of accuracy in the ranking of each candidate.



(a) Baseline geometry **(b) Baseline geometry control** **(c) Baseline mesh**

Fig. 7 Baseline geometry, geometry control and mesh for full scale indicator verification

IV. Results

Four different cases are studied with the developed algorithms: two shape optimization problems in inviscid flow to directly test indicator accuracy and two shape optimization problems in viscous flow representative of problems commonly seen in the literature. The primary focus of these studies is assessing the ability of DGC to improve final geometries and reduce iterations to convergence and seeks to apply these methods in more difficult, representative problems than have previously been attempted. Optimizing the code with respect to CPU time was not a priority and CPU time is not reported. Nevertheless, care was taken to avoid any methods that would be unavoidably prohibitively costly. As is, in most cases a full adaptive refinement cycle is comparable in wall time to a design iteration, and we are confident that with proper parallelization and code optimization the overhead of the presented methods should be consistently minimal in the context of high-fidelity aerodynamic shape optimization.

In every case presented here, the surface parameterization is fixed and the constituent B-splines are 4^{th} order. The B-splines used in the baseline geometry control schemes, including any axial curves, are of the highest order permitted by their initial number of control points up to a maximum order of three. During refinement, any B-spline with an order of less than three which has been enriched will have its order increased to the maximum permissable by the new number of control points, again up to a maximum order of three.

A. Indicator Accuracy Studies

Prior to applying the presented algorithms to shape optimization in viscous flows, we validate their accuracy using less expensive optimization in inviscid flows. Two cases are attempted, each based on optimization of an initially unswept, planar wing with a NACA 0012 cross-section in inviscid, transonic flow. Both cases use the baseline geometry control and mesh depicted in Figure 7, and take drag as the objective function. The baseline geometry control is intentionally very sparse, and the baseline wing in transonic flow begins with a large shock over the upper surface which is difficult to eliminate with such limited control, providing plenty of room for improvement through DGC. For each case, the baseline design space is optimized until the refinement criterion is satisfied and adaptive enrichment initiated.

At this point, each candidate has its potential estimated using all five indicators before being optimized to completion to obtain its true potential, allowing each indicator to be correlated to actual potential; our primary metric of success is the ability of each indicator to accurately predict the relative ranking of each candidate. To test the value of the ASM, the quadratic indicators ($\mathcal{A}_{\text{QI}}$, $\mathcal{A}_{\text{QIS}}$, $\mathcal{A}_{\text{QB}}$, and $\mathcal{A}_{\text{QR}}$) are tested both with and without it; in the latter case, the current active set is used for the solution.

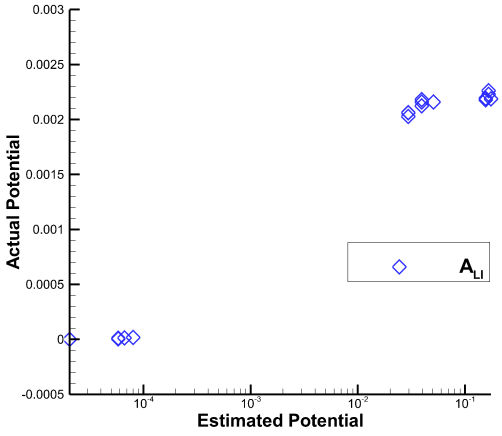
The first of the two cases permits only changes to section shape and is defined as

$$\begin{aligned} \text{minimize} \quad & C_D \\ \text{w.r.t.} \quad & \mathbf{X} \\ \text{subject to} \quad & C_L = 0.1 \\ & V \geq V_{\text{init}} \end{aligned} \tag{16}$$

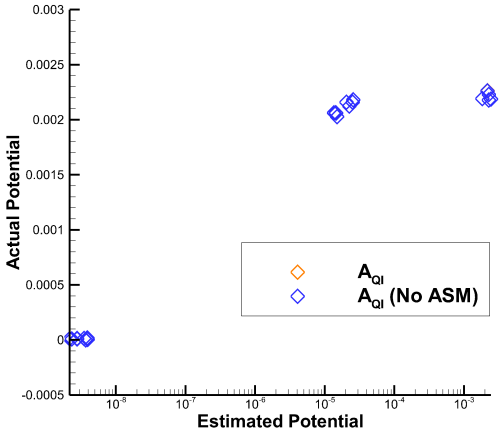
where V is the total volume and an additional linear constraint requires that section thickness at no point be less than 50% of the initial value. Figure 8 shows the resulting correlations.

Examining the vertical distribution of points reveals we have obtained two distinct families of candidates: one with near-zero potential clustered about the origin and another family of competitive candidates with significant actual potential. All five indicators are effective at differentiating between the two families. However, they consistently introduce a third family by splitting the candidates with significant potential into two groups. This tendency to exaggerate the differences between high-potential candidates will be noted in the following case as well and suggests the indicators may be oversensitive to small changes in these design spaces. This does not prevent them from producing accurate rankings, and within the high-potential candidates the indicator values correlate well with actual potential. All of the indicators appear equally capable of identifying high-potential candidates despite the considerable differences in model complexity they represent, ranging from the very simple linear indicator to the much more advanced BFGS or SR1-based quadratic fits. In this particular design space a less advanced Hessian formulation (or an explicitly linear model) does not produce a degradation in indicator performance.

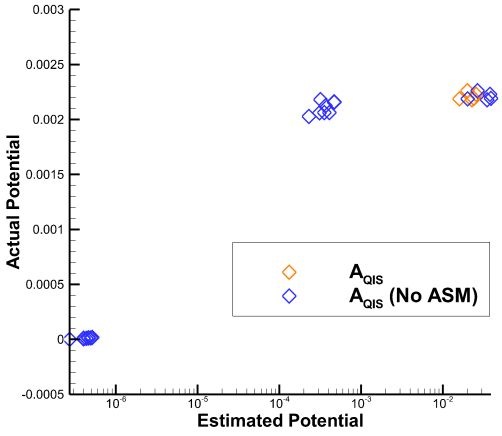
For quadratic cases, results without the ASM (blue) are superimposed on the results with the ASM (orange), and the overall activity of the ASM can be deduced at a glance from the number of visible orange points. For the $\mathcal{A}_{\text{QB}}$ and $\mathcal{A}_{\text{QI}}$ indicators, the ASM has a negligible effect, while for the $\mathcal{A}_{\text{QR}}$ and $\mathcal{A}_{\text{QIS}}$ indicators it has a much greater visibility, in particular for the rightmost candidates (those with greatest potential). This is intuitive as a large potential (all else being equal) suggests a larger step to the minimum, and a greater chance that additional constraints become active. While in this case the ASM does not significantly change candidate rankings, its increased relevance to higher-potential candidates is interesting as these are the candidates the algorithm is most likely to select and those that we most want to accurately rank.



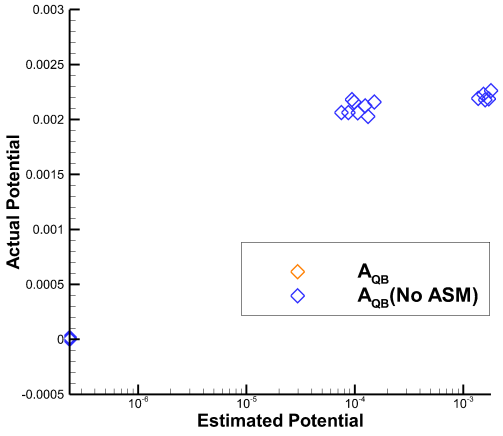
(a) $\mathcal{A}_{\mathbf{LI}}$



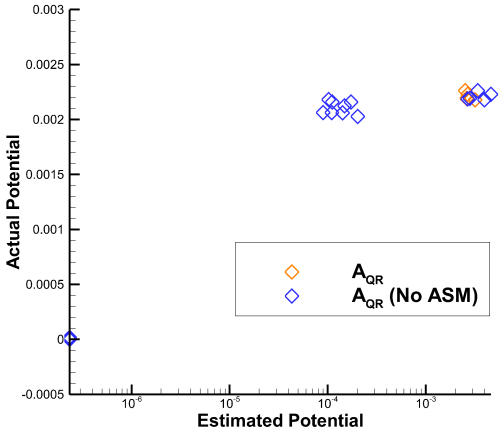
(b) $\mathcal{A}_{\mathbf{QI}}$



(c) $\mathcal{A}_{\mathbf{QIS}}$



(d) $\mathcal{A}_{\mathbf{QB}}$



(e) $\mathcal{A}_{\mathbf{QR}}$

Fig. 8 Indicator/potential correlation for section-only inviscid DGC study

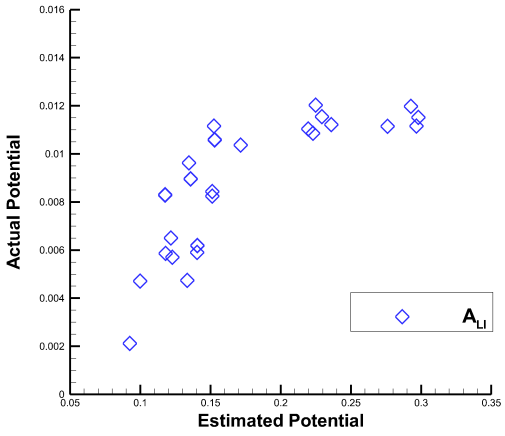
The second optimization problem in inviscid flow is an exploratory lift-constrained drag minimization permitting changes to twist, taper, section shape, and sweep angle meant to test the behaviour of the indicators in a more complicated optimization problem. This case uses the same baseline geometry, geometry control, and mesh as the previous case, but enforces a constant projected area constraint in addition to the minimum thickness and volume constraints from the first case.

The resulting correlations are provided in Figure 9. The correlations are somewhat noisier than previously, not surprisingly for a more complicated problem, but we still observe a good relationship between the indicators and actual potential. The indicators once again appear to become more sensitive for candidates with larger actual potentials, shown by a scatterplot that is predominantly vertical to the left and increasingly horizontal towards the right. The combination of additional constraints and generally larger potentials for each candidate has made the ASM much more visible for the quadratic indicators in this case; in particular for the $\mathcal{A}_{\mathrm{QB}}$ and $\mathcal{A}_{\mathrm{QR}}$ indicators, deactivating the ASM significantly changes both the indicator values and their relative rankings. We also again see no sizeable performance gap between the nominally more accurate $\mathcal{A}_{\mathrm{QB}}$ or $\mathcal{A}_{\mathrm{QR}}$ indicators and the simpler $\mathcal{A}_{\mathrm{QIS}}$, $\mathcal{A}_{\mathrm{QI}}$, or even $\mathcal{A}_{\mathrm{LI}}$ indicators, lending further credence to the theory that the value of accurate Hessian data may be less than expected.

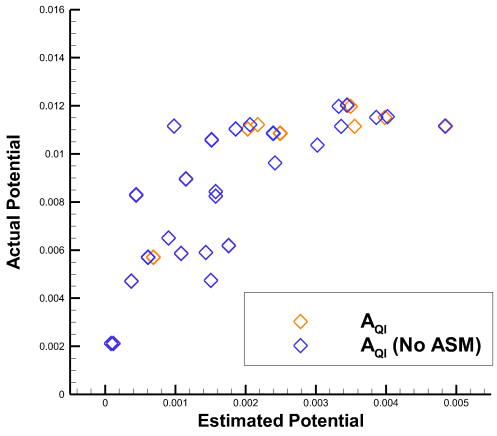
These plots show that good performance can be achieved by assuming a static active set. Nevertheless, we believe the ASM to be advantageous as it is most active in the candidates of greatest interest and automatically vanishes in cases where it is of less importance. Therefore, all further cases are run with the ASM active. This study has also clouded the potential advantages of Hessian information by showing largely similar performance regardless of the accuracy of the Hessian data used in constructing the fit. This raises the question of whether fitting the design space provides any benefits. To test this, we repeat the above tests using a "non-fitting" indicator, that is an indicator which does not construct an explicit model of the design space, but ranks on more direct metrics such as the gradient norm. The indicator chosen for this test was developed by He et al. [26], which we denote as $\mathcal{A}_{\mathrm{GH}}$ and is defined as

$$\mathcal{A}_{\mathrm{GH}} := \sum_{i=1}^{n_y} \left(\frac{\partial \mathcal{J}}{\partial \widehat{X}_i} - \sum_{j=1}^{n_g} \frac{\partial C_j^a}{\partial \widehat{X}_i} \right)^2 . \tag{17}$$

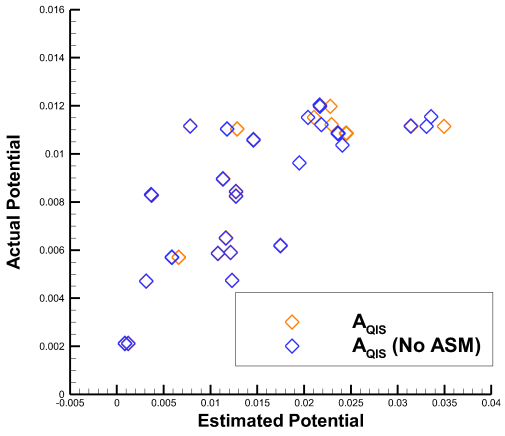
This ranks each candidate based on the value of the objective gradient, penalized by the gradients of any active constraints. This formulation accounts for the fact that while a large $|\partial \mathcal{J}/\partial X_i|$ value suggests high potential, an active constraint with a large $|\partial C/\partial X_i|$ value of the same sign may block any attempt to move a long distance in the optimal direction. However, when this indicator is applied to the two previously discussed cases, as in Figure 10, we see that there is no clear correlation between the indicator value and actual potential. Non-fitting indicators have been used to produce reasonable results before [24, 26], and these methods may still produce performance improvement versus static design spaces; however, they are likely to be less efficient or reliable than fitting indicators. The present results indicate that while using accurate Hessian data does not impart significant advantage to the indicator, the fit itself has value in



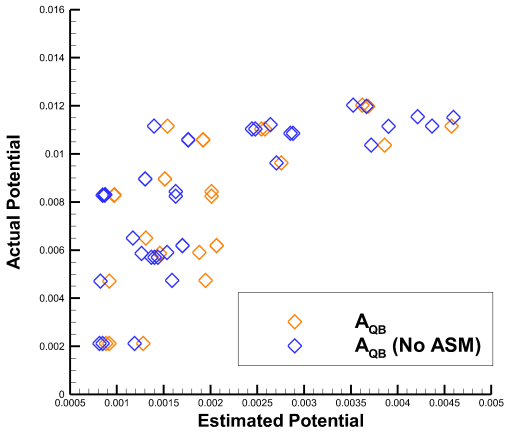
(a) $\mathcal{A}_{\mathbf{LI}}$



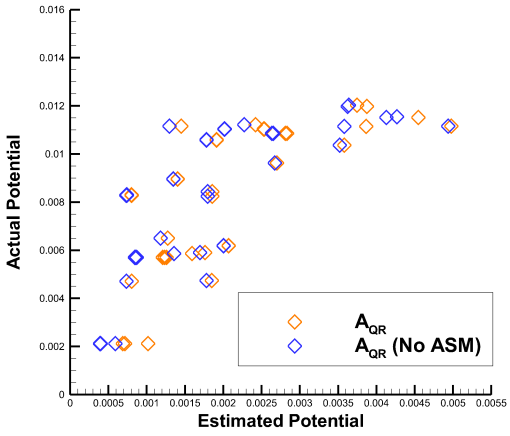
(b) $\mathcal{A}_{\mathbf{QI}}$



(c) $\mathcal{A}_{\mathbf{QIS}}$

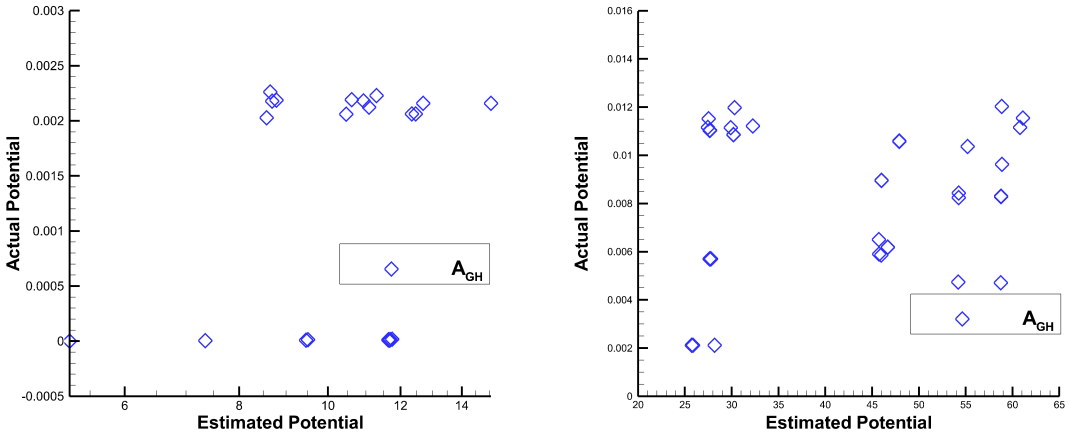


(d) $\mathcal{A}_{\mathbf{QB}}$



(e) $\mathcal{A}_{\mathbf{QR}}$

Fig. 9 Indicator/potential correlation for exploratory inviscid DGC study



(a) Section-only case **(b) Exploratory case**

Fig. 10 $\mathcal{A}_{\mathrm{GH}}$ accuracy

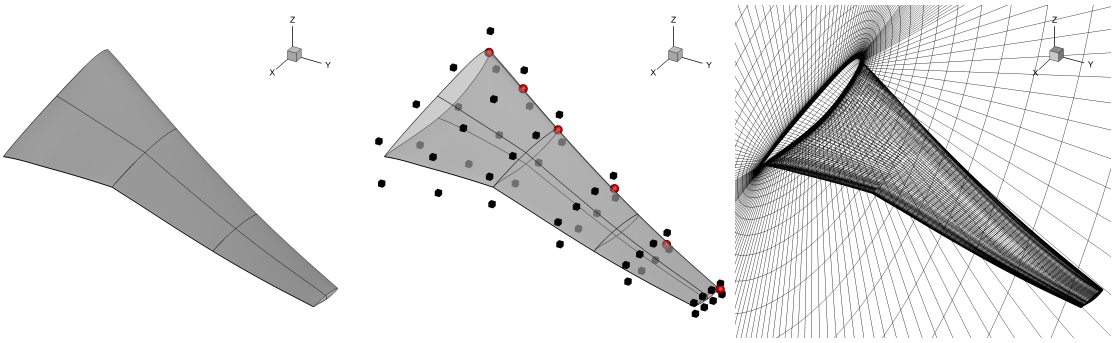
modelling the behaviour of a candidate design space. This advantage is likely at least in part due to the fitting indicators' ability to model the relationship between the active constraints and potential.

B. Detailed Wing Design

Having validated our adaptive algorithm, we now turn to the first of two optimization problems in viscous flow. Based on ADODG Case 4, this study uses the twist and section optimization of the ADODG CRM wing-only geometry illustrated in Figure 11a in transonic flow subject to the RANS equations and is representative of detailed wing-design problems commonly encountered in industry and academia. The problem can be stated as

$$\begin{aligned} \text{minimize} \quad & C_D \\ \text{w.r.t.} \quad & \mathbf{X} \\ \text{subject to} \quad & C_L = 0.5 \\ & V \geq 0.2617\,\mathrm{MAC}^3 \\ & C_M \geq -0.17 \end{aligned} \tag{18}$$

where C_D is the drag coefficient, C_L is the lift coefficient, V the wing volume, and C_M the pitching moment coefficient. A minimum thickness constraint is enforced requiring that the sectional thickness at no point be less than 85% of the initial value. The geometry parameterization is fixed and consists of 1720 surface control points forming a B-spline patch representation of the surface. The baseline geometry control system, shown in Figure 11b, consists of two FFD volumes with three cross-sections along the inboard FFD volume and four along the outboard, each with eight control points per cross-section and the two volumes meeting at the crank. The optimization mesh is depicted in Figure 11c



(a) Baseline geometry **(b) Baseline (S_0) geometry control** **(c) Baseline mesh**

Fig. 11 Baseline geometry, geometry control, and mesh for CRM

Table 2 CRM mesh parameters

Nodes	1,068,856
Blocks	40
Off-Wall Spacing	7.3×10^{-7} MAC
Mean y^+	0.13

Table 3 DGC parameters for detailed wing design case

Parameter	Value
r_{loc}	0.15
r_{glb}	0.015
r_{cut}	0.0005
n_{avg}	5
n_{loc}	5
n_{glb}	25
a_{loc}	0.5
a_{glb}	0.005
m_{loc}	2
m_{glb}	3

with parameters as tabulated in Table 2.

The baseline geometry control was used to create a sequence of five progressively finer static schemes, each scheme created by inserting a new control point halfway between each existing pair. The resulting series of geometry control schemes ranges from S_0 with 54 design variables, to S_4 with 8019 design variables, roughly two orders of magnitude larger than what would generally be used in a case of this nature. The case was then optimized using each of the five static schemes along with the progressive DGC algorithm and the quadratic and linear indicator-based adaptive DGC algorithms. Parameters for the DGC cases are provided in Table 3 and all DGC studies begin in the S_0 design space.

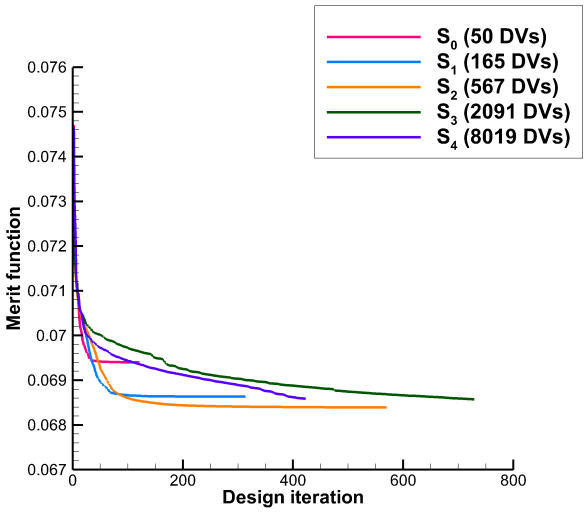
The value of the augmented Lagrangian (merit), maximum nonlinear constraint violation (feasibility) and augmented

Lagrangian gradient norm (optimality) convergence histories for all cases are provided in Figures 12 and 13, while Table 4 summarizes the results for each case, including the number of design iterations until convergence or termination, the maximum number of design variables, and the minimum feasible drag in counts. For unconverged cases, the reported number of design iterations is simply how long it was permitted to run before being terminated; for converged cases this value is the number of iterations until the final drag had been largely achieved and feasibility was satisfied, and may be less than the total number of iterations the case ran for. This is a subjective measure meant to communicate how quickly a given case converged, but the complete convergence histories can be seen in Figures 12 and 13. Note that the drag values reported in Table 4 are not mesh converged; as the purpose here is to compare performance across different geometry control schemes with identical meshes, the optimization mesh is sufficient for our purposes.

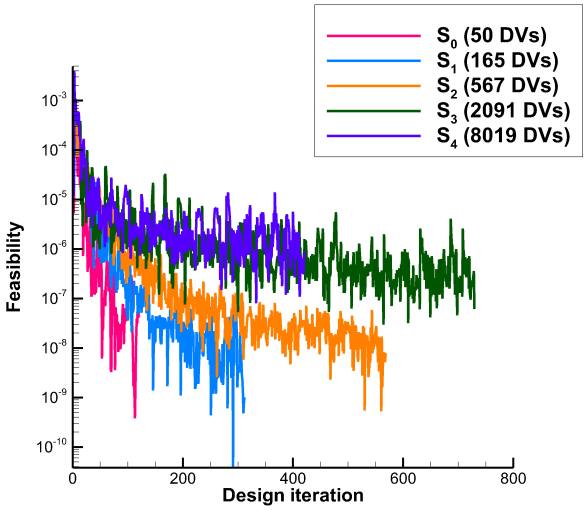
For the static results, the trends are almost exactly as expected. Refining from S_0 through S_2, convergence becomes slower and deeper; as refinement continues past S_2 into S_3 and S_4, performance breaks down and achieving convergence becomes increasingly expensive. One unusual finding in this case is that the finest S_4 level actually converges somewhat faster than the coarser S_3 level. There is no clear reason for this to be the case, but it is likely explained by the optimizer managing to find a sequence of good iterates early in the problem, and this does not change the overall noted trends.

Comparing the various DGC approaches, we note that the progressive geometry control is able to achieve good convergence and Figure 13a and Table 4 illustrate that it performs favourably compared to the static cases. It converges in roughly 50 fewer iterations compared to the best static S_2 case and produces a final drag roughly a half count lower than S_2, and over two counts lower than the comparably fine S_4 is able to achieve in a similar number of iterations. The gains in this case are not extreme, but it is nevertheless notable that even in a well-behaved, straightforward optimization problem starting from a competently-designed baseline geometry control, progressive DGC is still able to produce noticeable gains.

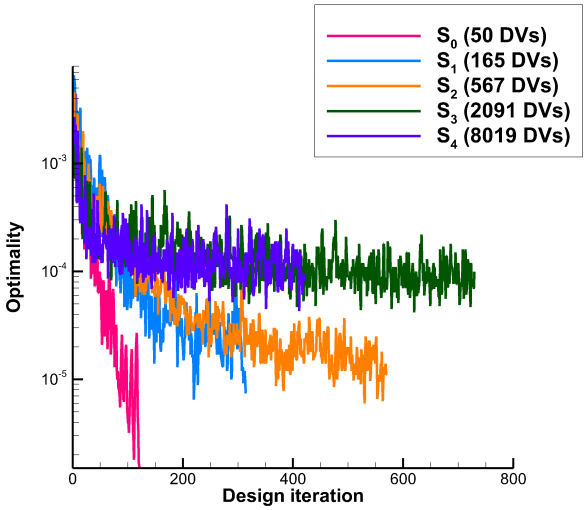
Five variations of the adaptive algorithm are applied to this problem, defined as per Section III.C: $\mathcal{A}_{\mathrm{LI}}$, $\mathcal{A}_{\mathrm{QI}}$, $\mathcal{A}_{\mathrm{QIS}}$, $\mathcal{A}_{\mathrm{QB}}$, and $\mathcal{A}_{\mathrm{QR}}$. All adaptive cases are able to locate geometries with lower drag than the static or progressive cases, and with considerably fewer design variables than either; however the convergence time and exact number of design variables varies. The fastest converging adaptive case was $\mathcal{A}_{\mathrm{QIS}}$, taking 25 fewer design iterations to obtain a half count drag reduction relative to the progressive case (roughly one count lower than S_2 and nearly three counts lower than S_4 in the same number of iterations), doing so using less than a quarter as many design variables as the progressive case. The lowest drag was achieved by the linear indicator, $\mathcal{A}_{\mathrm{LI}}$, which used half as many design variables as $\mathcal{A}_{\mathrm{QIS}}$ to obtain a final drag 1.2 counts lower. Compared to other approaches, the final drag is 2.3 counts lower than S_2 and nearly four counts lower than S_4 in the same number of iterations. However, this is at the cost of a 47% increase in iterations compared to $\mathcal{A}_{\mathrm{QIS}}$. The final three adaptive approaches ($\mathcal{A}_{\mathrm{QI}}$, $\mathcal{A}_{\mathrm{QB}}$, and $\mathcal{A}_{\mathrm{QR}}$) are able to locate lower drags with significantly coarser design spaces than the static geometry control, but converge slowly and were manually terminated after approximately 600 design iterations.



(a) Merit

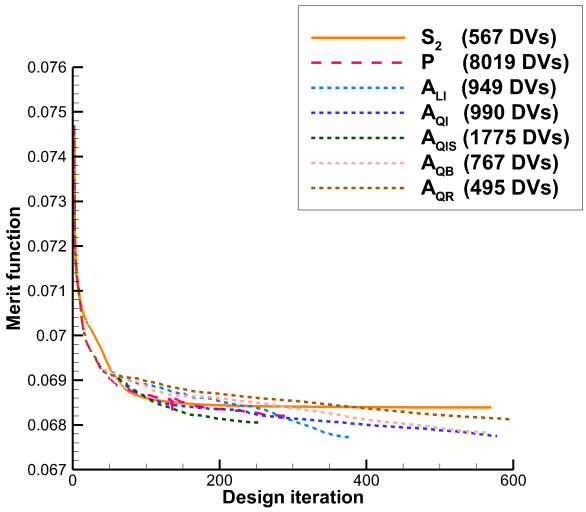


(b) Feasibility

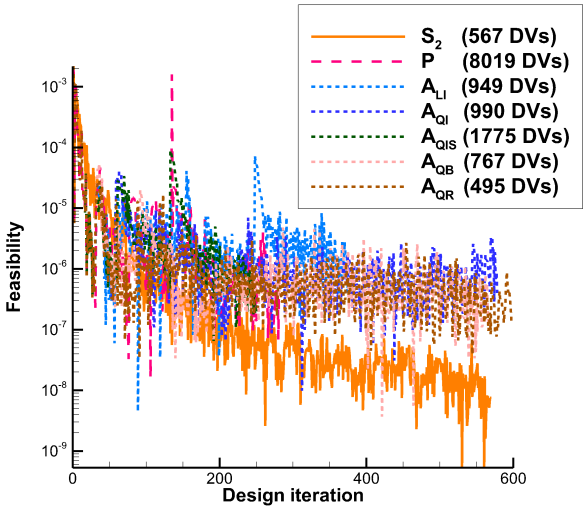


(c) Optimality

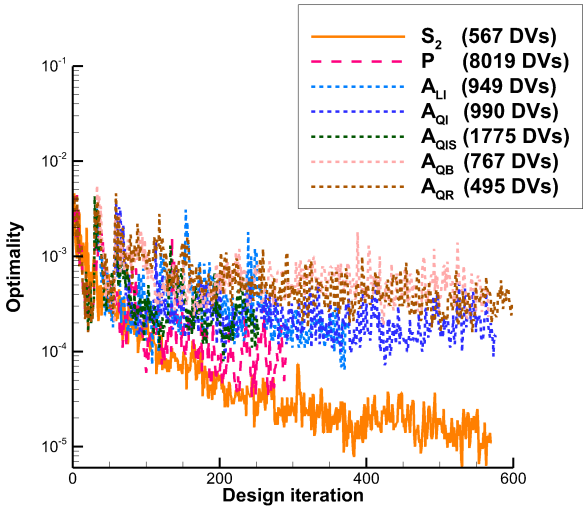
Fig. 12 Detailed wing design static convergence



(a) Merit

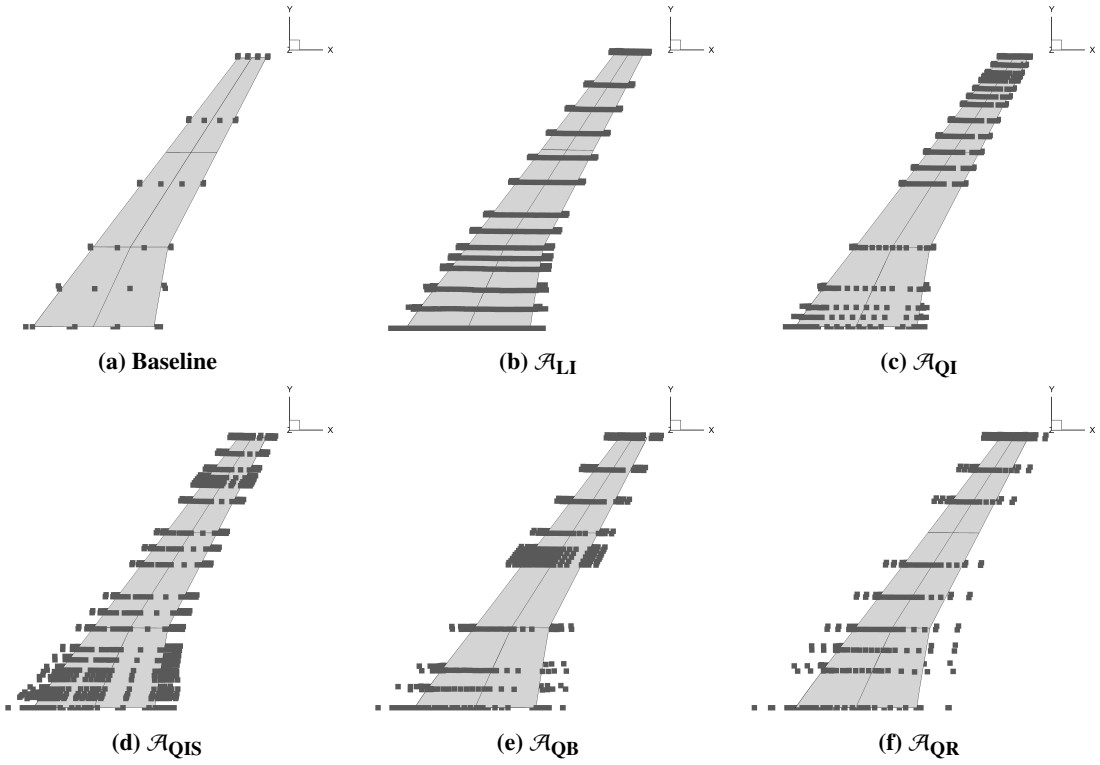


(b) Feasibility



(c) Optimality

Fig. 13 Detailed wing design dynamic convergence



(a) Baseline **(b) $\mathcal{A}_{\mathrm{LI}}$** **(c) $\mathcal{A}_{\mathrm{QI}}$**

(d) $\mathcal{A}_{\mathrm{QIS}}$ **(e) $\mathcal{A}_{\mathrm{QB}}$** **(f) $\mathcal{A}_{\mathrm{QR}}$**

Fig. 14 Final geometry control systems for adaptive geometry control in detailed wing design

Figure 14 compares the final geometry control generated by each adaptive DGC approach, along with the baseline for reference. The adaptive geometry control schemes produce similar distributions of control, though the resolutions vary noticeably. The most notable exception to this is the linear indicator which produces a nearly uniform chordwise distribution of cross-sectional points (though this does not mean that no clustering occurred at intermediate stages of optimization). The spanwise distributions of cross-sections are similarly consistent, though here the variations in resolution are more apparent. Here again the linear indicator is an outlier, but so is the $\mathcal{A}_{\mathrm{QR}}$ indicator, with both producing fairly uniform spacing of cross-sections along the wingspan. The relative consistency of the structures the adaptive indicators are generating would tend to support the theory that they are, for the most part, identifying the same regions as critical, though wide variability in performance and geometry control resolution suggests that the degree to which each indicator exploits these regions varies considerably.

The variability in the performance of the final geometries themselves warrants further attention. The final geometry control scheme for the progressive case is extremely fine and the case converges fully, so we should be able to assume that it achieves the lowest possible drag in the design space, but it is significantly outperformed by the linear adaptive case, which has just over 10% as many design variables. Equally surprising are the drag differences between the adaptive cases, where notable differences in drag persist, for instance $\mathcal{A}_{\mathrm{LI}}$ and $\mathcal{A}_{\mathrm{QIS}}$, which both fully converge and whose final drag coefficients differ by more than a drag count - sufficient to be distinct local optima by the criteria of Streuber

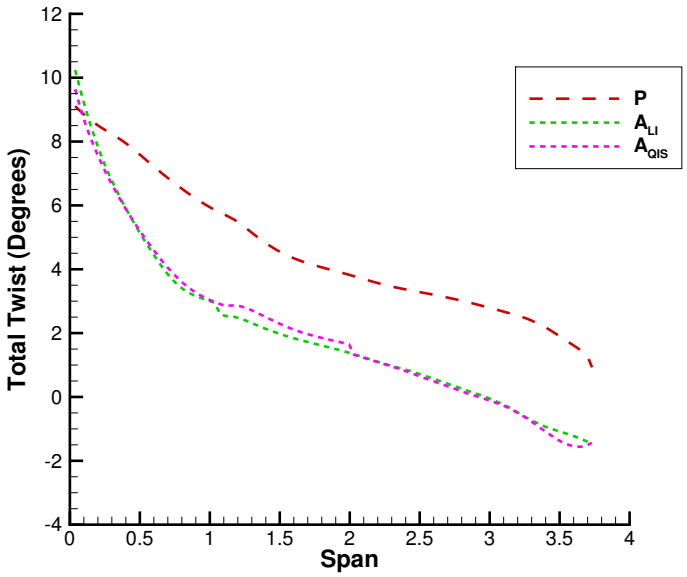


Fig. 15 Selected DGC twist distributions for detailed wing design

and Zingg [47]. Applying these criteria to each of the three fully converged DGC cases (P, $\mathcal{A}_{\text{LI}}$ and $\mathcal{A}_{\text{QIS}}$) reveals three distinct optima, illustrated by the twist distributions plotted in Figure 15. The first two optima are represented by the clearly distinct P and $\mathcal{A}_{\text{LI}}$ cases. $\mathcal{A}_{\text{QIS}}$, despite being quite similar in twist distribution to $\mathcal{A}_{\text{LI}}$ and fairly similar to P in final drag, is sufficiently different in both that it can be considered a distinct third optimum. This degree of multimodality is somewhat larger than that depicted by the similar problem studied in Streuber and Zingg [47], so there is the possibility that the introduction of DGC has had some effect on multimodality, but any further speculation must await a more detailed future study. Future work could also refine the multimodality methodology of Streuber and Zingg by assessing variations in span load and the breakdown of viscous versus induced drag in various final geometries.

These results are positive for adaptive geometry control, despite the presence of multimodality somewhat complicating our analysis. We can observe that DGC consistently offers better performance, with fewer design variables, and less user intervention than traditional static geometry control, and does so even in a relatively well-behaved, well studied problem such as this. These results also reinforce our earlier suspicions that high-fidelity Hessian data may be of little utility. The two best performing cases use only simplistic second-order data ($\mathcal{A}_{\text{QIS}}$) or none whatsoever ($\mathcal{A}_{\text{LI}}$).

C. Exploratory Winglet Design

This case is derived from the nonplanar studies of Koo and Zingg [48], and examines the performance of DGC in an exploratory optimization problem with a considerable number of DOFs and minimal constraints. Once again this is a lift-constrained drag minimization in transonic flow subject to the RANS equations. The baseline geometry is shown in

Table 4 Detailed wing design DGC study results. * denotes unconverged cases

Case	Design Iterations (Convergence)	Max DVs	Drag (Counts)
S_0	50	54	203.6
S_1	150	165	201.2
S_2	300	567	200.5
S_3*	728	2091	201.1
S_4*	420	8019	201.3
P	250	8019	199.9
$\mathcal{A}_{\mathrm{LI}}$	330	949	198.2
$\mathcal{A}_{\mathrm{QI}}$*	578	990	198.6
$\mathcal{A}_{\mathrm{QIS}}$	225	1775	199.4
$\mathcal{A}_{\mathrm{QB}}$*	566	767	198.8
$\mathcal{A}_{\mathrm{QR}}$*	596	495	199.7

Table 5 Winglet mesh parameters

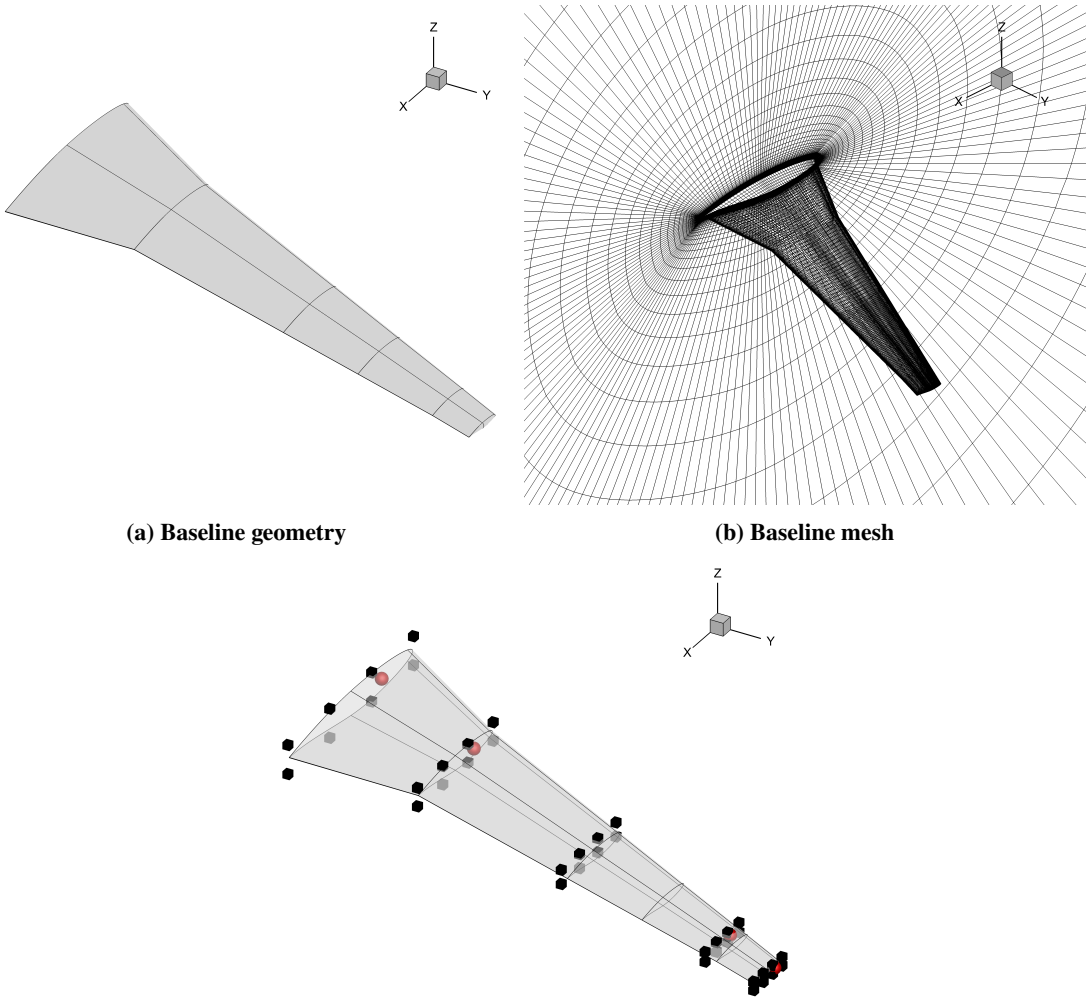
Nodes	1,631,000
Blocks	64
Off-Wall Spacing	7.8×10^{-7}
y^+	0.45

Figure 16a, but the primary interest is in the winglet region comprising the last 10% of the semi-span. For the inner 90% of the semi-span, the only active DOFs are twist and section, sufficient to allow the optimizer to eliminate any shocks that may be present over the wing. Within the last 10% of the semi-span, these two DOFs are combined with sweep and dihedral, to permit the optimizer to design a winglet. The optimization problem can be summarized as

$$\begin{aligned} \text{minimize} \quad & C_D S \\ \text{w.r.t.} \quad & \mathbf{X} \\ \text{subject to} \quad & C_L S = 1.45\,\mathrm{MAC}^2 \\ & V_w \geq 0.1859\,\mathrm{MAC}^3 \end{aligned} \tag{19}$$

where S is the projected area, V_w is the volume of the non-winglet region alone, and optimization is performed at a Mach number of 0.78 and a Reynolds number of 20 million. As this case contains axial freedom we expect the DGC algorithms to refine cross-sections, cross-sectional control points, and axial control points. The optimization mesh is shown in Figure 16b and its parameters are listed in Table 5.

As in the previous cases, a series of static geometry control schemes ranging from S_0 to S_4 were generated and optimized. The baseline geometry control system in Figure 16c contains just two cross-sections and two axial control



(a) Baseline geometry **(b) Baseline mesh**

(c) S_0 geometry control

Fig. 16 Geometry, mesh and control for exploratory winglet baseline design

points in the winglet region, limiting it to a linear B-spline. This is sufficient for a simple linear winglet, but finer (and higher order) control is necessary to create nonlinear geometries. The geometry parameterization consists of 36,992 surface control points and, as in all previous cases, is kept fixed.

The same DGC algorithms were applied here as previously, with the parameters detailed in Table 6. One difference between this and the previous studies is that two of the DGC cases (the progressive and scaled identity Hessian adaptive approach) were repeated with two sets of initial conditions, starting in both the S_0 and S_1 design spaces. Where the parameters for these sets of cases differed, they are denoted in Table 6 with a (0) and (1), respectively. Initially, these two DGC algorithms were initialized solely in S_0, as in all previous cases, but when their performance was compared to the static cases, as in Figure 17a, it was discovered that they were producing final drag values consistently higher than a well-designed static case. The explanation was found by comparing the final geometries for the S_0-initialized

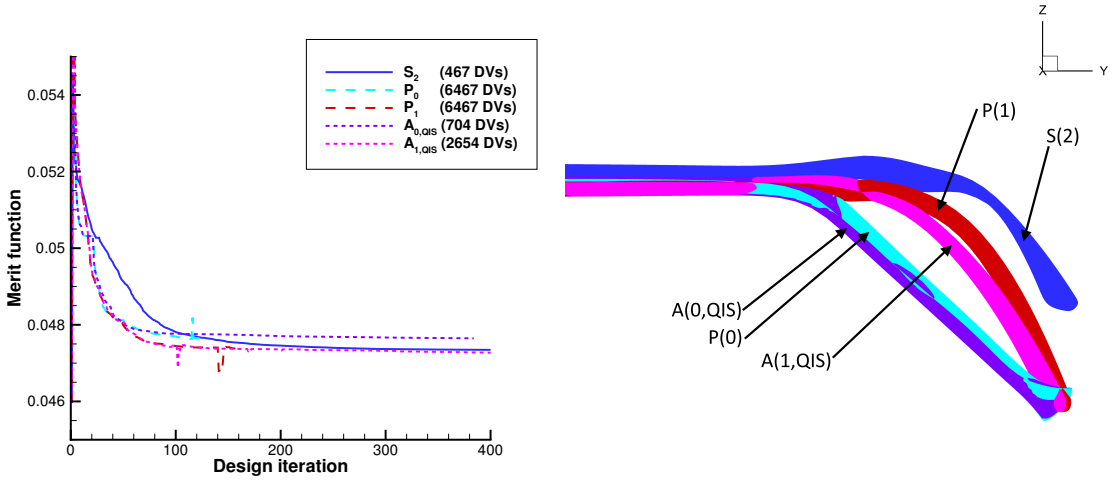
Table 6 DGC parameters for exploratory winglet design case

Parameter	Value
r_{loc}	0.050
r_{glb}	0.005
r_{cut}	0.0005
n_{avg}	5
n_{loc}	5
n_{glb}	25
a_{loc}	0.300 (0), 0.050 (1)
a_{glb}	0.005
m_{loc}	2
m_{glb}	3

DGC cases with the best static case, which we have done in Figure 17b. This revealed that the S_0 initialized DGC cases have consistently designed linear winglets, despite having refined into higher-dimension design spaces capable of nonlinear deformations. Starting in a design space capable of developing only linear winglets, the S_0-initialized DGC cases quickly converge to such a final geometry. A variant of this geometry persists as a local optimum within higher-dimensional design spaces; as the DGC cases refined, they continued to improve upon this existing linear winglet, rather than discovering any nonlinear designs. As a result, while the S_0-initialized DGC algorithms are able to converge in considerably fewer iterations, their final geometries generally underperform compared to the nonlinear designs obtained by the higher-order static design spaces. This was confirmed by repeating these studies with the DGC algorithms initialized in the S_1 design space, which is able to produce nonlinear deformations in the winglet from the beginning. The results of this study are also shown in Figure 17 and illustrate that the S_1-initialized DGC cases are able to converge just as quickly as the S_0-initialized cases, but to superior, nonlinear winglet designs.

These results are intriguing for two reasons: first, as a clear demonstration of the potential interactions between multimodality and DGC, which has already been identified as an area deserving of future study, but also as a potential answer to a question that has so far not been addressed in this work: how best to design the initial geometry control for DGC. For best results it is clear that the baseline geometry control should be sufficiently fine in each direction to permit nonlinear deformation of the geometry, wherever such deformations are desired. For this reason, going forward in this study, all further DGC cases are initialized in the S_1 design space.

Full static and DGC results are provided in Figures 18, 19, and Table 7. Only the S_1-initialized DGC cases are plotted in Figure 19, and in Figure 19a we have slightly changed the y-axis bounds to more clearly show the performance of each DGC algorithm. The static results illustrate a familiar pattern in which initially quick but shallow convergence gradually transitions to slower but deeper convergence with increased design space dimensionality, until optimization begins to break down for extremely fine geometry control schemes. In the DGC cases, the progressive method along with the

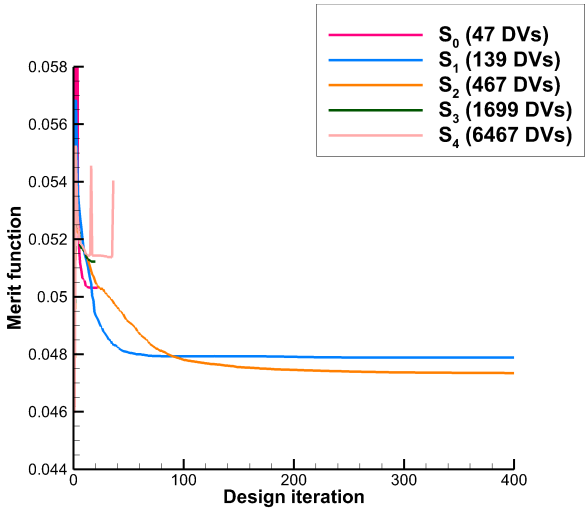


(a) Merit convergence **(b) Final winglet designs**

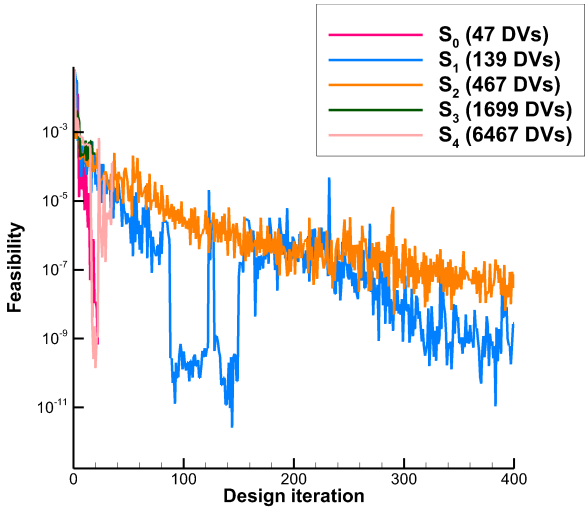
Fig. 17 Effect of initial geometry control on DGC performance. S_2 in blue, P_0 in teal, P_1 in red, $\mathcal{A}_{0,\text{QIS}}$ in purple, and $\mathcal{A}_{1,\text{QIS}}$ in pink

scaled identity and retroactive SR1 Hessian adaptive approaches have converged fairly well, while the unscaled identity and retroactive BFGS adaptive approaches have struggled to reduce optimality and so are considered unconverged. The linear adaptive case is also unconverged, but this was manually terminated after just 100 design iterations. Its failure to converge is not due any numerical difficulties, but rather because the CPU time necessary to solve the linear programming problem scaled very aggressively as problem size increased, ultimately rendering the solve time for the simplex method prohibitive. This is not due to any inherent problems with linear programming, but bottlenecks within the currently implemented simplex method. The simplex method in general, and this implementation in particular, is a fairly simple approach meant as a proof of concept for linear indicator-based adaptive geometry control, for instance lacking any parallelization in the solver itself. More advanced linear programming methods are designed to scale better with problem size, and therefore we strongly recommend that any future implementation of this method invest in such improvements. At the time it was terminated the linear adaptive case was outperforming all other approaches, so while it could not be adequately converged in this case, these results nevertheless reinforce both the potential of a linear adaptive indicator and the need for a more efficient LP solver to make such an approach practical.

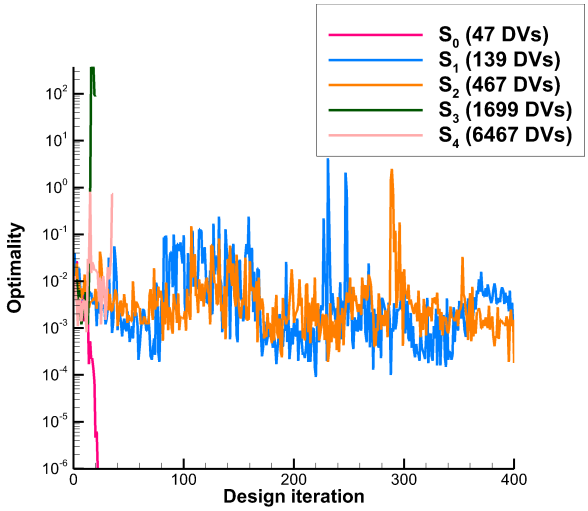
Turning to the successfully converged DGC cases, we note that the progressive algorithm performs quite well. Like many of the DGC algorithms, it encounters some numerical difficulties later in the optimization (indicated by large, temporary spikes or dips in the merit function) but is able to recover and converges to a drag slightly lower than the best static case in roughly half as many design iterations. The nonlinear adaptive algorithms also perform consistently well. While $\mathcal{A}_{1,\text{QIS}}$ performs the best (converging to a drag slightly lower than the progressive algorithm in a similar number of iterations), total variation in final drag between the adaptive approaches is just over 1% and convergence speedup is broadly similar, with most cases converging in roughly 50% as many design iterations as the best static case (S_2) and



(a) Merit

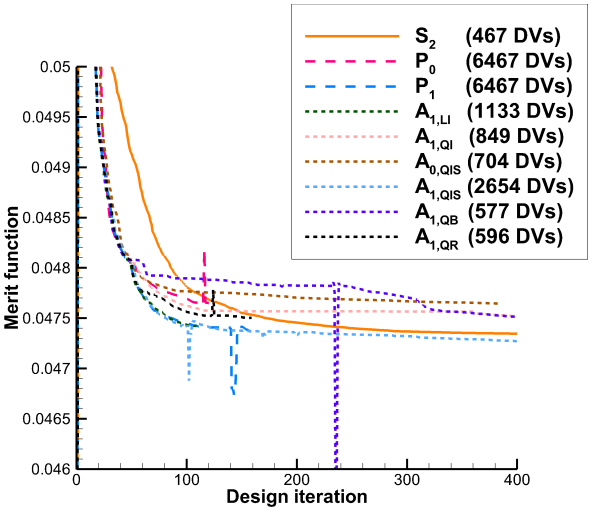


(b) Feasibility

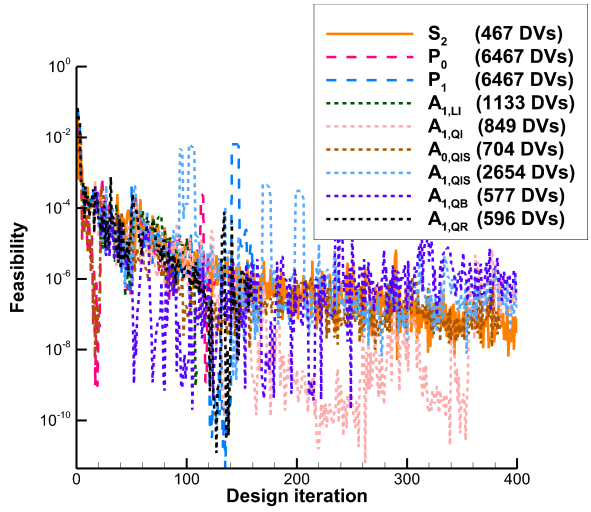


(c) Optimality

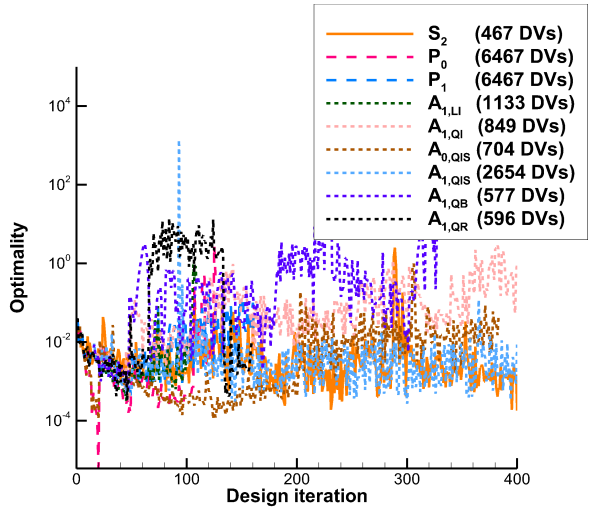
Fig. 18 Exploratory winglet static convergence



(a) Merit



(b) Feasibility



(c) Optimality

Fig. 19 Exploratory winglet dynamic convergence

Table 7 Exploratory winglet design DGC study results. Unconverged cases denoted with *

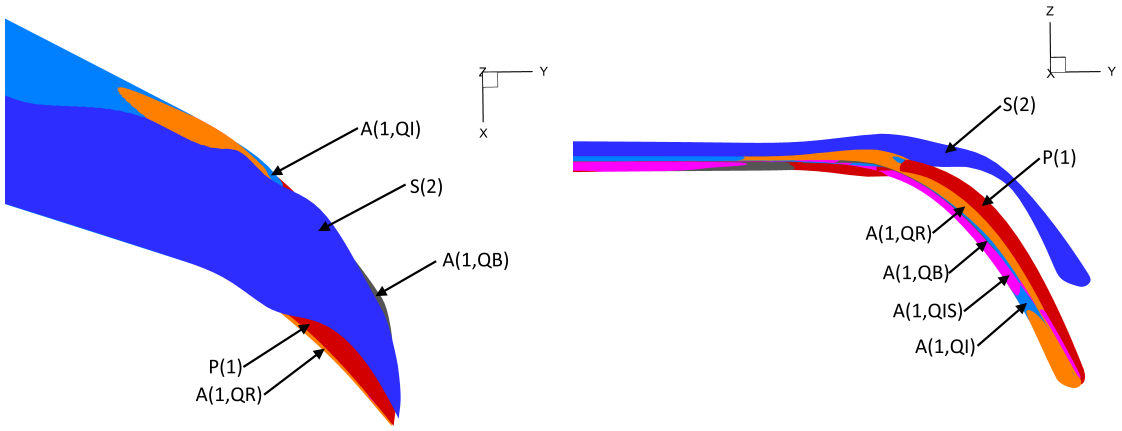
Case	Design Iterations (Convergence)	Max DVs	$C_D S$
S_0	21	47	5.03×10^{-2}
S_1	100	139	4.79×10^{-2}
S_2	300	467	4.74×10^{-2}
S_3*	20	1699	5.12×10^{-2}
S_4*	36	6467	5.14×10^{-2}
P_0	120	6467	4.76×10^{-2}
P_1	150	6467	4.73×10^{-2}
$\mathcal{A}_{1,\mathrm{LI}}$*	109	1133	4.74×10^{-2}
$\mathcal{A}_{1,\mathrm{QI}}$	150	849	4.75×10^{-2}
$\mathcal{A}_{0,\mathrm{QIS}}$	150	704	4.76×10^{-2}
$\mathcal{A}_{1,\mathrm{QIS}}$	150	2654	4.71×10^{-2}
$\mathcal{A}_{1,\mathrm{QB}}$*	340	577	4.75×10^{-2}
$\mathcal{A}_{1,\mathrm{QR}}$	140	596	4.75×10^{-2}

with significantly fewer design variables than the progressive algorithm requires. Reflecting the modest variation in final drag, Figure 20 shows that there is relatively little diversity in the final geometric shapes obtained by the S_1-initialized DGC cases.

Overall we find that DGC performs quite well in this problem, offering minor gains in final drag but significant reductions in the required iterations to converge compared to conventional static geometry control, with adaptive methods offering the added benefit of considerably improved automation. As previously, we find that the $\mathcal{A}_{\mathrm{LI}}$ and $\mathcal{A}_{\mathrm{QIS}}$ adaptive schemes work particularly well (assuming that a more efficient linear solver is used) and the retroactive Hessian approximations in $\mathcal{A}_{\mathrm{QB}}$ and $\mathcal{A}_{\mathrm{QR}}$, despite their nominally higher accuracy, produce performance that is at best comparable to and sometimes demonstrably worse than the simpler approaches.

V. Conclusions

We have presented several novel contributions to DGC, including several new approaches to the ranking of candidate design spaces and the implementation of an active set method for the quadratic indicators. We have also investigated DGC by applying these methods to a varied set of representative problems. Based on the results obtained, we conclude that DGC offers material benefits to aerodynamic shape optimization in a wide range of settings representative of both detailed and exploratory design problems. These benefits are in the form of improved automation, fewer required design iterations, and better final geometries. DGC is consistently able to obtain tangible reductions in final drag, along with reduction in design iterations of more than 50% compared to static methods in some cases. For the adaptive algorithms, these gains are realized alongside a significantly automated geometry control design process, reducing demands on the



(a) Planform view **(b) Dihedral view**

Fig. 20 Selected final geometries from exploratory winglet study. S_2 in dark blue, P_1 in red, $\mathcal{A}_{1,\mathrm{QI}}$ in light blue, $\mathcal{A}_{1,\mathrm{QIS}}$ in pink, $\mathcal{A}_{1,\mathrm{QB}}$ in grey, and $\mathcal{A}_{1,\mathrm{QR}}$ in orange.

end-user, and are achievable with large reductions in the number of required design variables. Progressive geometry control performs consistently well in every tested setting, while adaptive geometry control is often able to offer superior benefits, but performance can vary based on the selected algorithm. In general, we find particularly good performance is obtained from the linear and scaled identity Hessian approaches, respectively denoted as $\mathcal{A}_{\mathrm{LI}}$ and $\mathcal{A}_{\mathrm{QIS}}$, while an approach using an unscaled identity Hessian, $\mathcal{A}_{\mathrm{QI}}$, produces inconsistent results and two different approaches based on the BFGS and SR1 Hessian approximations, $\mathcal{A}_{\mathrm{QB}}$ and $\mathcal{A}_{\mathrm{QR}}$, offer middling to poor performance relative to the other adaptive algorithms, despite utilizing a nominally more accurate approximation of the candidate design spaces. These results are supported by preliminary studies that suggest the existence of hard theoretical limits on the benefits of second-order data in estimating potential. Nevertheless, we also demonstrate that while high-accuracy second-order data may be of limited utility, fitting the design space, whether using a linear or quadratic model, confers considerable advantages over more simplistic "non-fitting" indicators. We recommend that further studies be performed focused on examining the relationship between multimodality and DGC, as well as implementing more efficient linear programming methods for the linear indicator. Finally, applying DGC in the context of more problem types, such multipoint design or much more heavily constrained problems, would also yield valuable data on the performance of DGC as a whole and its various constituent components.

Acknowledgements

The authors wish to gratefully acknowledge the financial support of Bombardier Aerospace and of the Government of Ontario through the Ontario Graduate Scholarship. All cases were performed using computational resources generously provided by Compute Canada.

References

[1] Fudge, D., Zingg, D., and Haimes, R., "A CAD-free and a CAD-based geometry control system for aerodynamic shape optimization," *43rd AIAA Aerospace Sciences Meeting and Exhibit*, No. 2005-451, 2005, doi:https://doi.org/10.2514/6.2005-451.

[2] Kenway, G., Kennedy, G., and Martins, J., "A CAD-free approach to high-fidelity aerostructural optimization," *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, No. 2010-9231, 2010, doi:https://doi.org/10.2514/6.2010-9231.

[3] Gagnon, H. and Zingg, D. W., "Two-Level Free-Form and Axial Deformation for Exploratory Aerodynamic Shape Optimization," *AIAA Journal*, Vol. 53, No. 7, 2015, pp. 2015–2026, doi:https://doi.org/10.2514/1.j053575.

[4] Sederberg, T. and Parry, S., "Free-form deformation of solid geometric models," *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, 1986, pp. 151–160, doi:https://doi.org/10.1145/15922.15903.

[5] Hicken, J. and Zingg, D., "Aerodynamic optimization algorithm with integrated geometry parameterization and mesh movement," *AIAA Journal*, Vol. 48, No. 2, 2010, pp. 400–413, doi:https://doi.org/10.2514/1.44033.

[6] Hicks, R. and Henne, P., "Wing design by numerical optimization," *Journal of Aircraft*, Vol. 15, No. 7, 1978, pp. 407–412, doi:https://doi.org/10.2514/6.1977-1247.

[7] Secco, N., Jasa, J., Kenway, G., and Martins, J., "Component-based geometry manipulation for aerodynamic shape optimization with overset meshes," *AIAA Journal*, Vol. 56, No. 9, 2018, pp. 3667–3679, doi:https://doi.org/10.2514/1.j056550.

[8] Jakobsson, S. and Amoignon, O., "Mesh Deformation using Radial Basis Functions for Gradient-Based Aerodynamic Shape Optimization," *Computers and Fluids*, Vol. 36, 2007, pp. 1119–1136, 10.1016/j.compfluid.2006.11.002.

[9] da Silva, J., Giraldi, G., and Apolinário Jr, A., "Data-driven optimization approach for mass-spring models parametrization based on isogeometric analysis," *Journal of Computational Science*, Vol. 23, 2017, pp. 1–19, doi:https://doi.org/10.1016/j.jocs.2017.09.010.

[10] Lu, X., Huang, J., Song, L., and Li, J., "An improved geometric parameter airfoil parameterization method," *Aerospace Science and Technology*, Vol. 78, 2018, pp. 241–247, doi:https://doi.org/10.1016/j.ast.2018.04.025.

[11] Limkilde, A., Evgrafov, A., Gravesen, J., and Mantzaflaris, A., "Practical isogeometric shape optimization: parametrization by means of regularization," *Journal of Computational Design and Engineering*, 2020, doi:https://doi.org/10.1093/jcde/qwaa093.

[12] Agromayor, R., Anand, N., Müller, J.-D., Pini, M., and Nord, L., "A Unified Geometry Parametrization Method for Turbomachinery Blades," *Computer-Aided Design*, Vol. 133, 2021, pp. 102987, doi:https://doi.org/10.1016/j.cad.2020.102987.

[13] Samareh, J., "Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization," *AIAA journal*, Vol. 39, No. 5, 2001, pp. 877–884, doi:https://doi.org/10.2514/3.14814.

[14] Castonguay, P. and Nadarajah, S., "Effect of shape parameterization on aerodynamic shape optimization," *45th AIAA Aerospace Sciences Meeting and Exhibit*, 2007, p. 59, doi:https://doi.org/10.2514/6.2007-59.

[15] Lee, C., Koo, D., and Zingg, D., "Comparison of B-Spline Surface and Fee-form Deformation Geometry Control for Aerodynamic Optimization," *AIAA Journal*, Vol. 55, No. 1, 2017, pp. 228–240, doi:https://doi.org/10.2514/1.j055102.

[16] Kedward, L., Allen, C., and Rendall, T., "Gradient-Limiting Shape Control for Efficient Aerodynamic Optimization," *AIAA Journal*, Vol. 58, No. 9, 2020, pp. 3748–3764, doi:https://doi.org/https://doi.org/10.2514/1.j058977.

[17] Beux, F. and Dervieux, A., "A hierarchical approach for shape optimization," *Engineering Computations*, Vol. 11, No. 1, 1994, pp. 25–48, doi:https://doi.org/10.1108/02644409410799191.

[18] Désidéri, J., "Hierarchical optimum-shape algorithms using embedded Bézier parameterizations," *Numerical Methods for Scientific Computing, Variational Problems and Applications*, 2003, pp. 45–56.

[19] Andreoli, M., Ales, J., and Désidéri, J.-A., "Free-form-deformation parameterization for multilevel 3D shape optimization in aerodynamics," Tech. Rep. 5019, INRIA, 2003.

[20] Duvigneau, R., Chaigne, B., and Désidéri, J.-A., "Multi-level parameterization for shape optimization in aerodynamics and electromagnetics using a particle swarm optimization algorithm," Tech. Rep. 6003, INRIA, 2006.

[21] Masters, D., Taylor, N., Rendall, T., and Allen, C., "Multilevel subdivision parameterization scheme for aerodynamic shape optimization," *AIAA Journal*, Vol. 55, No. 10, 2017, pp. 3288–3303, doi:https://doi.org/10.2514/1.j055785.

[22] Anderson, G., Nemec, M., and Aftosmis, M., "Aerodynamic shape optimization benchmarks with error control and automatic parameterization," *53rd AIAA Aerospace Sciences Meeting*, No. 2015-1719, 2015, doi:https://doi.org/10.2514/6.2015-1719.

[23] Masters, D., Taylor, N., Rendall, T., and Allen, C., "Progressive subdivision curves for aerodynamic shape optimisation," *54th AIAA Aerospace Sciences Meeting*, No. 2016-0559, 2016, doi:https://doi.org/10.2514/6.2016-0559.

[24] Han, X. and Zingg, D., "An adaptive geometry parametrization for aerodynamic shape optimization," *Optimization and Engineering*, Vol. 15, No. 1, 2014, pp. 69–91, doi:https://doi.org/10.1007/s11081-013-9213-y.

[25] Masters, D., Taylor, N., Rendall, T., and Allen, C., "A Locally Adaptive Subdivision Parameterisation Scheme for Aerodynamic Shape Optimisation," *34th AIAA Applied Aerodynamics Conference*, No. 2016-3866, 2016, doi:https://doi.org/doi.org/10.2514/6.2016-3866.

[26] He, X., Li, J., Mader, C., Yildirim, A., and Martins, J., "Robust aerodynamic shape optimization - from a circle to an airfoil," *Aerospace Science and Technology*, Vol. 87, 2019, pp. 48–61, doi:https://doi.org/10.1016/j.ast.2019.01.051.

[27] Anderson, G. and Aftosmis, M., "Adaptive Shape Control for Aerodynamic Design," *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, No. 2015-398, 2015, doi:https://doi.org/10.2514/6.2015-0398.

[28] Anderson, G., *Shape Optimization in Adaptive Search Spaces*, Ph.D. thesis, Stanford University, 2016.

[29] Sinsay, J. and Alonso, J., "A Heuristic Approach to Finding the Preferred Design Variable Parameterization for Optimization," *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, No. 2016-0415, 2016, doi:https://doi.org/10.2514/6.2016-0415.

[30] Hicken, J. and Zingg, D., "Parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms," *AIAA Journal*, Vol. 46, No. 11, 2008, pp. 2773–2786, doi:https://doi.org/10.2514/1.34810.

[31] Osusky, M. and Zingg, D., "Parallel Newton–Krylov–Schur Flow Solver for the Navier–Stokes Equations," *AIAA Journal*, Vol. 51, No. 12, 2013, pp. 2833–2851, doi:https://doi.org/10.2514/1.j052487.

[32] Saad, Y. and Schultz, M. H., "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, 1986, pp. 856–869, doi:https://doi.org/10.1137/0907058.

[33] Osusky, L., Buckley, H., Reist, T., and Zingg, D., "Drag minimization based on the Navier–Stokes equations using a Newton–Krylov approach," *AIAA Journal*, Vol. 53, No. 6, 2015, pp. 1555–1577, doi:https://doi.org/10.2514/1.j053457.

[34] Jameson, A., "Aerodynamic design via control theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233–260, doi:https://doi.org/10.1007/BF01061285.

[35] De Sturler, E., "Truncation strategies for optimal Krylov subspace methods," *SIAM Journal on Numerical Analysis*, Vol. 36, No. 3, 1999, pp. 864–889, doi:https://doi.org/10.1137/s0036142997315950.

[36] Gill, P., Murray, W., and Saunders, M., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Review*, Vol. 47, No. 1, 2005, pp. 99–131, doi:https://doi.org/10.1137/s1052623499350013.

[37] Broyden, C., "The convergence of a class of double-rank minimization algorithms 1.general considerations," *IMA Journal of Applied Mathematics*, Vol. 6, No. 1, 1970, pp. 76–90, doi:https://doi.org/10.1093/imamat/6.1.76.

[38] Fletcher, R., "A new approach to variable metric algorithms," *The Computer Journal*, Vol. 13, No. 3, 1970, pp. 317–322, doi:https://doi.org/10.1093/comjnl/13.3.317.

[39] Goldfarb, D., "A family of variable-metric methods derived by variational means," *Mathematics of Computation*, Vol. 24, No. 109, 1970, pp. 23–26, doi:https://doi.org/10.1090/s0025-5718-1970-0258249-6.

[40] Shanno, D., "Conditioning of quasi-Newton methods for function minimization," *Mathematics of Computation*, Vol. 24, No. 111, 1970, pp. 647–656, doi:https://doi.org/10.1090/s0025-5718-1970-0274029-x.

[41] Truong, A., Oldfield, C., and Zingg, D., "Mesh movement for a discrete-adjoint Newton-Krylov algorithm for aerodynamic optimization," *AIAA Journal*, Vol. 46, No. 7, 2008, pp. 1695–1704, doi:https://doi.org/10.2514/1.33836.

[42] Samareh, J., "Aerodynamic shape optimization based on free-form deformation," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, No. 2004-4630, 2004, pp. 3672–3683, doi:https://doi.org/10.2514/6.2004-4630.

[43] Gagnon, H. and Zingg, D., "Aerodynamic optimization trade study of a box-wing aircraft configuration," *Journal of Aircraft*, Vol. 53, No. 4, 2016, pp. 971–981, doi:https://doi.org/10.2514/1.C033592.

[44] Squire, W. and Trapp, G., "Using complex variables to estimate derivatives of real functions," *SIAM Review*, Vol. 40, No. 1, 1998, pp. 110–112, doi:https://doi.org/10.1137/s003614459631241x.

[45] Dantzig, G., Orden, A., and Wolfe, P., "The generalized simplex method for minimizing a linear form under linear inequality restraints," *Pacific Journal of Mathematics*, Vol. 5, No. 2, 1955, pp. 183–195, doi:https://doi.org/10.2140/pjm.1955.5.183.

[46] Nocedal, J. and Wright, S., *Numerical Optimization*, Vol. 35, Springer Science, 1999.

[47] Streuber, G. and Zingg, D., "Evaluating the Risk of Local Optima in Aerodynamic Shape Optimization," *AIAA Journal*, Vol. 59, No. 1, 2021, pp. 75–87, doi:https://doi.org/10.2514/1.j059826.

[48] Koo, D. and Zingg, D., "Investigation into Aerodynamic Shape Optimization of Planar and Nonplanar Wings," *AIAA Journal*, Vol. 56, No. 1, 2018, pp. 1–14, doi:https://doi.org/10.2514/1.j055978.