AIAA JOURNAL
Vol. 48, No. 2, February 2010

Aerodynamic Optimization Algorithm with Integrated Geometry Parameterization and Mesh Movement

Jason E. Hicken* and David W. Zingg†
University of Toronto, Toronto, Ontario M3H 5T6, Canada

DOI: 10.2514/1.44033

An efficient gradient-based algorithm for aerodynamic shape optimization is presented. The algorithm consists of several components, including a novel integrated geometry parameterization and mesh movement, a parallel Newton–Krylov flow solver, and an adjoint-based gradient evaluation. To integrate geometry parameterization and mesh movement, generalized B-spline volumes are used to parameterize both the surface and volume mesh. The volume mesh of B-spline control points mimics a coarse mesh; a linear elasticity mesh-movement algorithm is applied directly to this coarse mesh and the fine mesh is regenerated algebraically. Using this approach, mesh-movement time is reduced by two to three orders of magnitude relative to a node-based movement. The mesh-adjoint system also becomes smaller and is thus amenable to complex-step derivative approximations. When solving the flow-adjoint equations using restarted Krylov-subspace methods, a nested-subspace strategy is shown to be more robust than truncating the entire subspace. Optimization is accomplished using a sequential-quadratic-programming algorithm. The effectiveness of the complete algorithm is demonstrated using a lift-constrained induced-drag minimization that involves large changes in geometry.

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033

I. Introduction

THE aircraft industry faces two critical challenges in the 21st century: climate change [1] and peak oil production [2,3]. These problems may eventually be solved by alternative fuels such as hydrogen and bio-kerosene. However, when production emissions are included, these alternative fuels presently produce more greenhouse gases than traditional kerosene [4]. Alternative fuels must therefore be considered a long-term solution.

In the near term, unconventional aircraft configurations offer the potential for reduced emissions and improved fuel efficiency. For example, design studies of the blended-wing body suggest a 27% reduction in fuel burn per seat mile compared with a conventional composite aircraft [5]. But can we do better than this? In particular, can we use numerical optimization to discover radically new concepts in aircraft design?

Using numerical optimization to uncover novel configurations is an exciting prospect, but there are significant challenges. The required optimization algorithm will involve multiple disciplines and high-fidelity analysis codes. The present work is focused on the aerodynamic discipline with a view to incorporating the resulting modules with other disciplines in subsequent work. Moreover, we consider only clean aerodynamic configurations of fixed topology; holes cannot be created or removed during the optimization.

Even if we limit ourselves to aerodynamic optimization, the questions posed above remain very difficult. For example, any hope of finding novel drag-reduction concepts requires a highly flexible and efficient geometry parameterization. Without a flexible parameterization, the optimization algorithm may not reveal new concepts. If the parameterization is inefficient, the algorithm may use more design variables than necessary and converge slowly.

No method of parameterization is clearly superior to all others. Nevertheless, for *clean* aerodynamic configurations with fixed topologies, there are several arguments in favor of patched B-spline surfaces. From approximation theory, we know that the space of $(p-1)$-degree splines converges asymptotically to any function $f \in C^{p-1}[a,b]$ with order p [6]. In particular, cubic B-splines will converge at a fourth-order rate to the smooth geometries typically used in aerodynamics (indeed, many geometries used in the aerospace industry are B-splines). Although approximations based on Fourier continuation [7] or Chebyshev partial sums may converge faster, B-splines offer other advantages, such as local control of the geometry. More complicated geometries, involving a finite number of piecewise-smooth surfaces, can be readily approximated by joining the individual B-spline patches along their edges. For the above reasons, we have chosen to parameterize geometries using B-spline surfaces.

To address our motivating question, we also need a robust and fast mesh-movement algorithm. Algebraic mesh movement can be fast [8–10], but is typically limited to small shape changes. Liu et al. [11] developed an algebraic mesh movement based on mapping the mesh to a Delaunay graph. They demonstrate that the Delaunay graph approach is robust for large shape changes, provided multiple increments are used; however, analysis of their method suggests that, in general, using multiple increments produces a discontinuous objective function. This may limit the approach to gradient-free optimization methods.

Batina [12] introduced spring-analogy mesh movement, which models the mesh edges as springs. Although more robust than many algebraic algorithms, the spring-analogy method can produce negative cell volumes [13]. To address this problem, torsional [14,15] and semitorsional [16,17] springs have been incorporated into the spring-analogy method. These extensions greatly enhance the robustness of the spring analogy, but they also increase the computational cost.

Johnson and Tezduyar [18] demonstrated that the equations of linear elasticity can be used to achieve robust mesh movement, even for large shape changes. This approach has been used successfully for aeroelastic problems [19] and aerodynamic optimization [13,20]. Unfortunately, the equations of linear elasticity are typically less diagonally dominant than the spring-analogy equations, so the elasticity approach tends to be more computationally expensive.

Jakobsson and Amoignon [21] developed a promising mesh-movement algorithm based on radial basis functions (RBFs). In RBF

Presented as Paper 6079 at the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia, Canada, 10–12 September 2008; received 26 February 2009; revision received 11 September 2009; accepted for publication 15 October 2009. Copyright © 2009 by J. E. Hicken and D. W. Zingg. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 0001-1452/10 and $10.00 in correspondence with the CCC.

*Postdoctoral Fellow, Institute for Aerospace Studies. Student Member AIAA.

†Professor and Director, Canada Research Chair in Computational Aerodynamics, J. Armand Bombardier Foundation Chair in Aerospace Flight. Associate Fellow AIAA.

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033

mesh movement, the surface displacements are interpolated into the interior. This interpolation problem is relatively fast, although the resulting mesh may not interpolate the geometry unless the geometry is also parameterized using radial basis functions [21]. Indeed, such a parameterization is proposed by Allen and Rendall [22] and Morris et al. [23], who implemented an integrated approach based on RBFs for both mesh-movement and free-form deformation of the geometry.

Our proposed mesh-movement algorithm also uses ideas from free-form deformation [24] and integrates geometry parameterization with mesh movement. Integration is achieved by parameterizing the mesh using the control points of B-spline volumes. The control points corresponding to the surface nodes are adopted as the design variables; hence, the geometry is parameterized as a B-spline surface, as desired. Mesh movement is accomplished by applying *any* standard movement algorithm to the coarse B-spline grid. Thus, the proposed integrated approach provides an entire class of efficient mesh-movement algorithms while simultaneously representing the geometry as a B-spline surface. Details regarding the integrated method can be found in Sec. II.

The flow solver plays a critical role in an aerodynamic optimization algorithm. The solver must provide accurate aerodynamic analyses for the range of geometries encountered during an optimization. Moreover, these analyses must be performed rapidly if the optimization is to be practical. To meet these requirements, we use an efficient parallel Newton–Krylov flow solver [25], which we briefly review in Sec. III.

Finally, we must choose an optimization algorithm. Often, this is a choice between fast local convergence (gradient-based methods) and increased probability of finding the global optimum (stochastic methods). Our target application, which will involve hundreds or even thousands of design variables, will likely lead to a highly multimodal design space. We say likely, because the complexity of the design space is not obvious a priori. Clearly there will be local optima (consider the different ways to achieve elliptical loading), but will there be a few or many? If the former, then a gradient-based algorithm with a simple multistart procedure will be sufficient. If the latter, then a hybrid approach can be developed that couples a stochastic global search with a gradient-based local search (see, for example, Gage et al. [26] and Vicini and Quagliarella [27]). In either case, gradients will provide invaluable information about the design space.

To compute the gradient, an adjoint approach is adopted. By introducing adjoint variables, Pironneau [28] showed that the gradient can be calculated at a cost that is (virtually) independent of the number of design variables. This adjoint-based gradient calculation was later pioneered by Jameson [29] within computational aerodynamics and is now well established. A particular variation, called the discrete adjoint method [30,31], ensures that the gradient is exact with respect to the discrete objective function and compatible with sophisticated nonlinear optimization algorithms.

The flow-adjoint variables are governed by a sparse linear system. This system can be solved, for example, using Krylov iterative methods [10,32–35] or by time-marching the system to steady state [29,36,37]. Among Krylov iterative methods, the restarted generalized minimal residual (GMRES) method [38] is popular for solving the adjoint equations. However, restarted GMRES may exhibit degraded and, in some cases, stalled convergence. In Sec. IV.A, we demonstrate improved convergence of the adjoint problem by applying a nested Krylov subspace method.

Nielsen and Park [39] and Truong et al. [20] included mesh-movement adjoint variables in the gradient evaluation and demonstrated improved efficiency. Unfortunately, forming the mesh-adjoint equations can be tedious, depending on the complexity of the mesh-movement algorithm. Our proposed mesh-movement algorithm produces a relatively small mesh-adjoint system, which simplifies code development, since its small size makes complex-step derivative approximation [40–42] practical in the formation of the mesh-adjoint equations (see Sec. IV.B).

The algorithm components are demonstrated by coupling them with the SNOPT optimization software [43]. SNOPT is based on the sequential-quadratic-programming paradigm and is designed for nonlinear optimization problems with general (nonlinear) constraints. The complete algorithm, incorporating the individual components, is described and verified in Sec. V. Subsequently, we present an optimization example to illustrate the algorithm in Sec. VI and provide some concluding remarks in Sec. VII.

II. Integrated Geometry Parameterization and Mesh Movement

A. B-Spline Volume Meshes and Modified Basis Functions

A B-spline tensor-product volume is a mapping from the cubic domain

$$D = \{\boldsymbol{\xi} = (\xi, \eta, \zeta) \in [0, 1]^3\}$$

to $P \subset \mathbb{R}^3$ and is defined by

$$\mathbf{x}(\boldsymbol{\xi}) = \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} \sum_{k=1}^{N_k} \mathbf{B}_{ijk} \mathcal{N}_i^{(p)}(\xi) \mathcal{N}_j^{(p)}(\eta) \mathcal{N}_k^{(p)}(\zeta) \tag{1}$$

The points $\mathbf{B}_{ijk}$ are the de Boor control points, or simply control points. The functions $\mathcal{N}_i^{(p)}(\xi)$ are the B-spline basis functions of order p; they are $(p-1)$-degree spline polynomials joined at nondecreasing knot locations. The first p and last p knots are equal to 0 and 1, respectively (i.e., we use open knot vectors [44]).

A B-spline volume mesh [45,46] is produced from Eq. (1) by discretizing the domain D. The parameters ξ, η, and ζ do not need to be discretized in a uniform way. Indeed, nonuniform parameter spacing is usually necessary for precise control of mesh spacing in physical space. If a flow solver requires uniform mesh spacing in parameter space, an intermediate mapping is implied.[‡] A B-spline volume and a corresponding mesh are illustrated in Fig. 1. Note that B-spline volumes can also be generated by generalizing triangular Bézier patches, which may be of interest to unstructured grid users.

The basis functions appearing in Eq. (1) are generalized to permit curved knot lines [47]. For example, the basis functions in the ξ direction are given by

$$\mathcal{N}_i^{(1)}(\xi; \eta, \zeta) = \begin{cases} 1 & \text{if } T_i(\eta, \zeta) \le \xi < T_{i+1}(\eta, \zeta) \\ 0 & \text{otherwise} \end{cases}$$

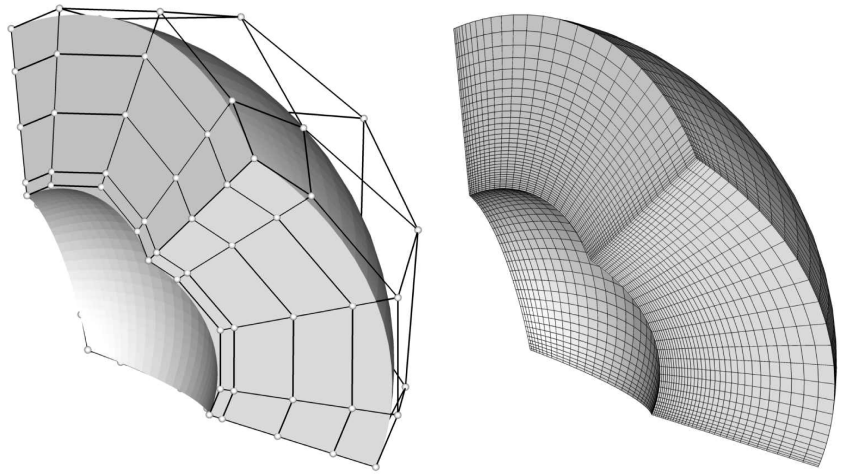
$$\mathcal{N}_i^{(p)}(\xi; \eta, \zeta) = \left(\frac{\xi - T_i(\eta, \zeta)}{T_{i+p-1}(\eta, \zeta) - T_i(\eta, \zeta)} \right) \mathcal{N}_i^{(p-1)}(\xi; \eta, \zeta) + \left(\frac{T_{i+p}(\eta, \zeta) - \xi}{T_{i+p}(\eta, \zeta) - T_{i+1}(\eta, \zeta)} \right) \mathcal{N}_{i+1}^{(p-1)}(\xi; \eta, \zeta) \tag{2}$$

where $T_i(\eta, \zeta)$ are the knot values. Analogous definitions generalize the basis functions $\mathcal{N}_j^{(p)}(\eta; \zeta, \xi)$ and $\mathcal{N}_k^{(p)}(\zeta; \xi, \eta)$. Readers familiar with B-splines will recognize that the above definition differs only in its use of spatially varying knots. The spatially varying knots allow the modified B-spline basis to be tailored to different edges of a geometry.

For fixed η and ζ, the modified basis function (2) reduces to the standard definition, so the modified basis retains C^{p-2} continuity at the knots in the ξ direction. Although less obvious, the modified basis is also C^{p-2} in the η and ζ directions, provided $T_i(\eta, \zeta) \in C^{p-2}$ and the internal knots have a multiplicity of, at most, one. This result is a consequence of the chain rule and the smoothness of the derivative of a B-spline with respect to its knots [48–50].

The internal knots of the modified basis functions must be strictly increasing and sufficiently smooth, but the user is otherwise free to choose the functional form. For simplicity, we use bilinear knots of the form

[‡]In practice, finite difference and other mapping-based discretizations use the grid coordinates only, so the details of the B-spline parameters are not important; however, the smoothness of the mapping must be consistent with the order of the discretization being used. The fourth-order splines used here are suitable for discretizations up to third-order.



a) b)

Fig. 1 Example of a B-spline volume mesh: a) B-spline control grid and b) corresponding volume mesh.

$$T_i(\eta,\zeta) = [(1-\eta)(1-\zeta)]T_{i,(0,0)} + [\eta(1-\zeta)]T_{i,(1,0)} + [(1-\eta)\zeta]T_{i,(0,1)} + [\eta\zeta]T_{i,(1,1)} \quad (3)$$

The four constants $T_{i,(0,0)}$, $T_{i,(1,0)}$, $T_{i,(0,1)}$, and $T_{i,(1,1)}$ denote the ith knot value at the η and ζ edges of the parameter space. Again, similar knot definitions are used for $T_i(\zeta,\xi)$ and $T_i(\xi,\eta)$. Figure 2 shows an example modified basis function. The salient feature is the changing basis location.

B. Approximating Grids Using B-Spline Volume Meshes

We have reviewed how B-spline volumes can be used, in theory, to define mappings from computational space to physical space. To use these mappings in practice, the optimization algorithm could be coupled with a B-spline-based mesh generator that provides access to the control points and their sensitivities. Alternatively, since mesh generators do not generally provide this functionality, we fit existing grids to determine the initial control-point positions and the ξ, η, and ζ parameter values.

A least-squares fit with parameter correction [51] is often used to find spline curve and surface approximations to data points. We use a similar least-squares fit to obtain B-spline volume approximations of multiblock structured grids. Each block of the structured grid is associated with a B-spline volume. For each B-spline volume, the user chooses the number of control points and the B-spline order in the parameter directions ξ, η, and ζ. At interfaces where blocks meet, the number of control points and order must be consistent to ensure continuity.

The ξ, η, and ζ values associated with each grid point are determined using a chord-length parameterization. For example, the ξ values, along a curve of constant η and ζ, are determined by the normalized arc length. The knots along each edge of the B-spline volume are located such that an equal number of nodes are in each knot interval. Thus, the knots inherit a chord-length parameterization from the ξ, η, and ζ values. We have found that chord-length-based knots produce control grids that approximate the spacing of a coarse mesh, a feature that is important for our chosen mesh-movement algorithm. Once the knots along the edges of the volume are calculated, the bilinear interpolation (3) is used to find knot values in the interior.

Finally, the fitting algorithm solves least-squares systems to determine the control points that best approximate the mesh (for further details, see [44]). Least-squares problems are solved sequentially for the edge, face, and internal control points. This ordering ensures that adjacent blocks have consistent grid-point locations. If necessary, the iterative parameter-correction algorithm of Hoschek [51] can be applied to improve the fit by adjusting the parameter values.

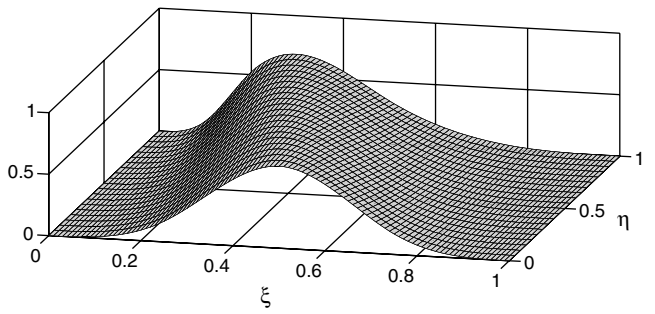


Fig. 2 Example two-dimensional modified basis or, equivalently, a three-dimensional modified basis for fixed ζ.

C. Applications of B-Spline Volumes

The control grid is designed to mimic a coarse mesh, so the surface control points provide a low dimensional approximation of the surface. This makes the surface control points a suitable choice for the design variables in shape optimization. However, B-spline volumes may be useful in applications other than optimization.

1) The mesh-movement algorithm can be used to generate high-quality grids. A simple canonical grid, with the same surface patch topology as a target geometry, can be morphed into a grid for the target geometry (see [52]).

2) The surface control points can be used to control the shape of the wing in studies of aeroelasticity.

3) The volume control points can be used for mesh adaptation, rather than the individual grid points.

4) The parameter space can be refined in a smooth way to achieve rigorous mesh convergence studies (see Sec. VI).

5) Hierarchical grids can easily be constructed for multigrid.

6) Grids and geometries suitable for high-order discretizations can be obtained with a suitable choice of B-spline basis order.

D. Linear Elasticity Model for Control-Point Movement

Once fitted using B-spline volumes, the flow-analysis mesh can be manipulated using the control grid of points $\{\mathbf{B}_{ijk}\}$. In particular, we can apply any mesh-movement algorithm of our choice to the control grid and regenerate the flow-analysis mesh algebraically using Eq. (1).

A very simple algebraic mesh-movement algorithm consists of moving only those control points associated with design variables. Although this fixes the internal control points, the mesh points near the surface geometry will still move, because B-spline basis functions at the boundary extend into the interior (imagine moving the end of a spline curve while fixing the remaining control points). Such a mesh-movement strategy may be useful if only small shape changes are necessary.

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033

If larger shape changes are expected, then an algorithm is needed that moves the internal control points based on the surface control points. Essentially, any grid movement algorithm can be applied to the control points: algebraic, spring analogy, linear elasticity, radial basis functions, etc. For B-spline volume meshes, there are typically two orders of magnitude fewer control points than grid points, so the CPU time of the mesh movement becomes insignificant relative to the flow solution. For this reason, we have chosen to use a robust, albeit expensive, linear-elasticity-based mesh movement.

We model the control mesh as a linear elastic solid with stiffness controlled using a nonconstant Young's modulus E. The algorithm is very similar to the one used in Truong et al. [20]. The equations of linear elasticity are discretized on the B-spline control mesh using the finite element method with trilinear elements.

For large shape changes, the problem can be broken into a sequence of m mesh-movement problems by moving the surface in increments. When increments are used, the stiffness matrix becomes a function of the control-point coordinates at the previous level through the spatially varying Young's modulus (explained below). Thus, the linear equation for the control points at increment i has the form

$$\mathcal{M}^{(i)}(\mathbf{b}^{(i-1)}, \mathbf{b}^{(i)}) = K^{(i)}(\mathbf{b}^{(i-1)})[\mathbf{b}^{(i)} - \mathbf{b}^{(i-1)}] - \mathbf{f}^{(i)} = 0, \quad i = 1, \ldots, m \tag{4}$$

where $\mathcal{M}^{(i)}$ is the mesh-movement residual, $\mathbf{b}^{(i)}$ is a block-column vector of control-point coordinates, and $K^{(i)}$ is the symmetric-positive-definite stiffness matrix. Given the properties of the stiffness matrix, we solve Eq. (4) using the conjugate gradient method preconditioned with an incomplete lower-upper (ILU) factorization: specifically, ILU(p) [53] and a fill level of $p = 1$.

Element stiffness is controlled using a spatially varying Young's modulus. Young's modulus is calculated at the beginning of the mesh movement, or at the beginning of each increment if the movement is broken into smaller steps. The goal is to vary the element stiffness in such a way that mesh quality is maintained in critical regions of the grid (e.g., the boundary layer). Young's modulus at increment i is given by

$$E_{\mathcal{E}}^{(i)} = \frac{\Phi_{\mathcal{E}}^{(i-1)}}{\Phi_{\mathcal{E}}^{(0)} V_{\mathcal{E}}^{(i-1)}}, \qquad i = 1, 2, \ldots, m \tag{5}$$

where $V_{\mathcal{E}}$ is the element volume and

$$\Phi_{\mathcal{E}} = \left(\prod_{v=1}^{8} \frac{1}{(\mathbf{u}_v \times \mathbf{v}_v) \cdot \mathbf{w}_v} \right)^2 \tag{6}$$

is an orthogonality measure. The vectors $\mathbf{u}_v$, $\mathbf{v}_v$, and $\mathbf{w}_v$ are unit vectors parallel to the element edges that meet at vertex v, and they form a right-handed system. Hence, the triple product $(\mathbf{u}_v \times \mathbf{v}_v) \cdot \mathbf{w}_v$ is positive for valid elements, equal to 1 for orthogonal vectors, and tends to zero as the vectors become coplanar. Following Bar-Yoseph et al. [54], $\Phi_{\mathcal{E}}^{(i)}$ is normalized in $E_{\mathcal{E}}^{(i)}$ by its value on the initial mesh.

E. Mesh-Movement Examples

We demonstrate the integrated geometry parameterization and mesh movement using two examples. In the first example, we parameterize and morph an existing shape (the ONERA M6 wing [55]) and test the algorithm using modest shape changes that are typical of traditional aerodynamic shape optimization problems. The second example involves morphing a flat plate into a blended-wing body (BWB) with winglets and demonstrates the algorithm's ability to handle large geometric changes.

In both examples, we use a node-based mesh movement as a benchmark for the B-spline mesh movement. The node-based algorithm consists of applying the methodology described in Sec. II.D to the individual nodes rather than the control points. In both algorithms, the movement is broken into five increments ($m = 5$), and Poisson's ratio is fixed at $\nu = -0.2$.

The grids produced by the two mesh-movement algorithms are evaluated using an orthogonality measure based on Eq. (6). Specifically, the quality of an element $\mathcal{E}$ is defined by

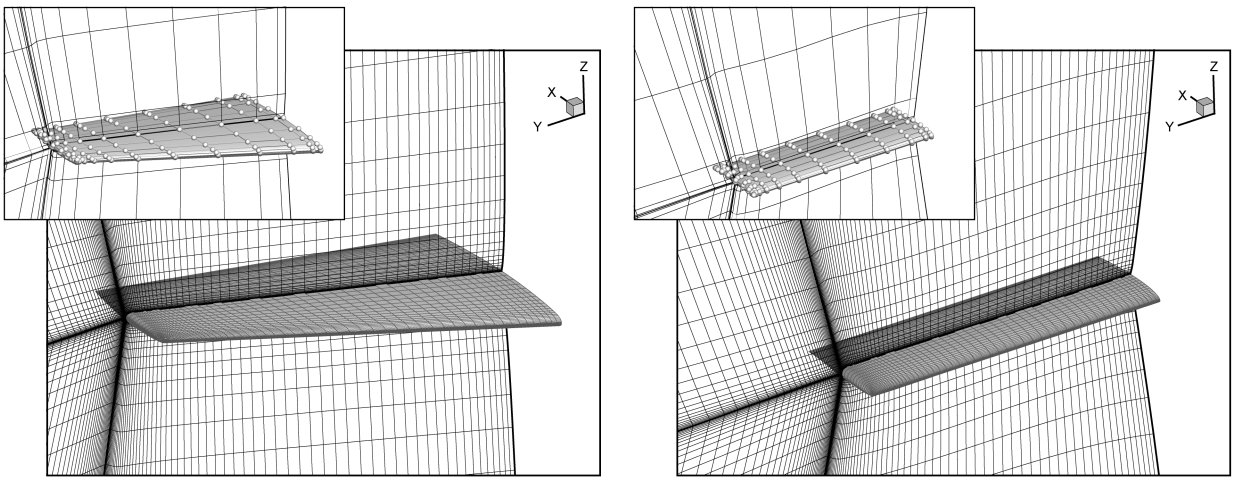
$$Q_{\mathcal{E}} = \sqrt{\frac{1}{\Phi_{\mathcal{E}}}} = \prod_{v=1}^{8} (\mathbf{u}_v \times \mathbf{v}_v) \cdot \mathbf{w}_v \tag{7}$$

It follows from this definition that elements with perfect orthogonality have $Q_{\mathcal{E}} = 1$, and highly skewed elements have $Q_{\mathcal{E}} \approx 0$. To provide a fair comparison, in the case of the B-spline mesh-movement algorithm, $Q_{\mathcal{E}}$ is measured for elements on the interpolated mesh and not the control grid. The measure is calculated for each element and then grouped into 50 bins that uniformly divide the range of possible values: namely, [0, 1]. These bins are then used to produce a distribution of orthogonality. Integrating the distribution over the orthogonality range $[a, b]$ gives the ratio of elements that lie in this range. In particular, integrating the distribution over [0, 1] gives 1.

1. *ONERA M6 Wing Morphed to Unswept Wing*

We parameterize the ONERA M6 wing and transform it into an unswept wing with NACA 0012 airfoil sections. The root chord of the M6 wing is normalized to 1.0, and the unswept wing has a root chord of 0.49664. The shape transformation involves sweep, scaling, and section modifications.

The parameterization is obtained from the surface control points of a B-spline mesh fitted to an initial 12-block H-H-topology grid. Each



a) Initial mesh and control grid (inset) **b) Final mesh and control grid (inset)**

Fig. 3 Control grids and flow-analysis meshes for the initial ONERA M6 wing geometry (left) and the final unswept-wing geometry (right).

of the 12 blocks in the grid consists of $45 \times 65 \times 33$ nodes, leading to approximately 1.158×10^6 nodes in total. The mesh spacing is typical for an inviscid flow analysis with the offwall, leading-edge, trailing-edge, and tip spacings set at 0.001 chord-length units. The B-spline volumes for each block use $N_i \times N_j \times N_k = 13 \times 13 \times 9$ control points; hence, the B-spline grid is approximately 60 times smaller than the computational grid. Figure 3 shows the initial and final B-spline control grids and their corresponding flow-analysis meshes.

Figure 4 plots the orthogonality distribution for the initial ONERA M6 grid, the unswept-wing grid obtained using the B-spline mesh movement, and the unswept-wing grid obtained using the node-based mesh movement. The final grids have distributions that are qualitatively similar to the initial distribution, which we would expect for mesh-movement algorithms based on linear elasticity. More notable are the similarities between the B-spline and node-based mesh-quality distributions. Indeed, in some cases, the B-spline distribution is better; consider the first peak, near the low end of the quality range, which is smaller for the B-spline mesh. This can be attributed to the smoothing properties of the B-spline volumes.

For this problem, the B-spline and node-based mesh movement required 227 s and 27.79 h, respectively, on a single 1500 MHz Itanium 2 processor. For the B-spline mesh movement, we found that a fill level of 1 was optimal in the ILU(p) preconditioner, and a fill level of 2 was better for the larger node-based problem. Although better preconditioners may exist, the relative performance of the two approaches is ultimately bounded by the relative size of their linear systems.

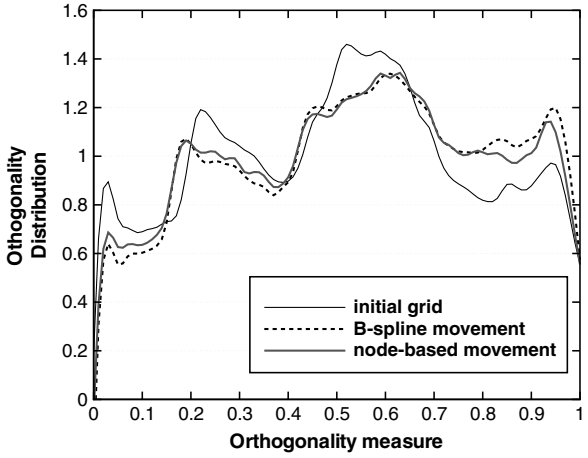


Fig. 4 Orthogonality distribution for the initial ONERA M6 grid and grids for the unswept wing.

2. Flat Plate Morphed to Blended-Wing Body

In this example, the initial shape is a flat plate with unit chord and a semispan of 2. The mesh consists of 12 blocks in an H-H topology, and each block has $45 \times 45 \times 45$ nodes. The offwall spacing is 0.001 chord-length units, and the leading-edge, trailing-edge, and tip spacing are 0.005 chord lengths.

The initial mesh is fit using 12 B-spline volumes, with $9 \times 9 \times 9$ control points per volume; the control grid reduces the number of degrees of freedom by a factor of 125 relative to the fine mesh. The fitted mesh for the flat plate is shown in Fig. 5a, together with its control grid (inset).

The control points on the surface of the plate are chosen as design variables. In this example, we set the design variables to obtain a generic blended-wing geometry with winglets. The final mesh for the blended-wing-body shape is shown in Fig. 5b. The perturbed-surface control points and control grid are shown in the inset.

The orthogonality distribution for the flat-plate grid is plotted in Fig. 6, together with the distributions for the BWB grids obtained using the B-spline and node-based mesh-movement algorithms. The initial grid is almost Cartesian and its distribution reflects this: all the elements have orthogonality measures greater than 0.74. In transforming from the flat plate to the blended-wing body, some loss of element orthogonality is unavoidable. As in the ONERA M6 example, the important observation is that the B-spline and node-based mesh movements produce very similar quality distributions, despite the significant difference in CPU time: the B-spline mesh movement required 128 s, and the node-based mesh movement required 32.4 h.

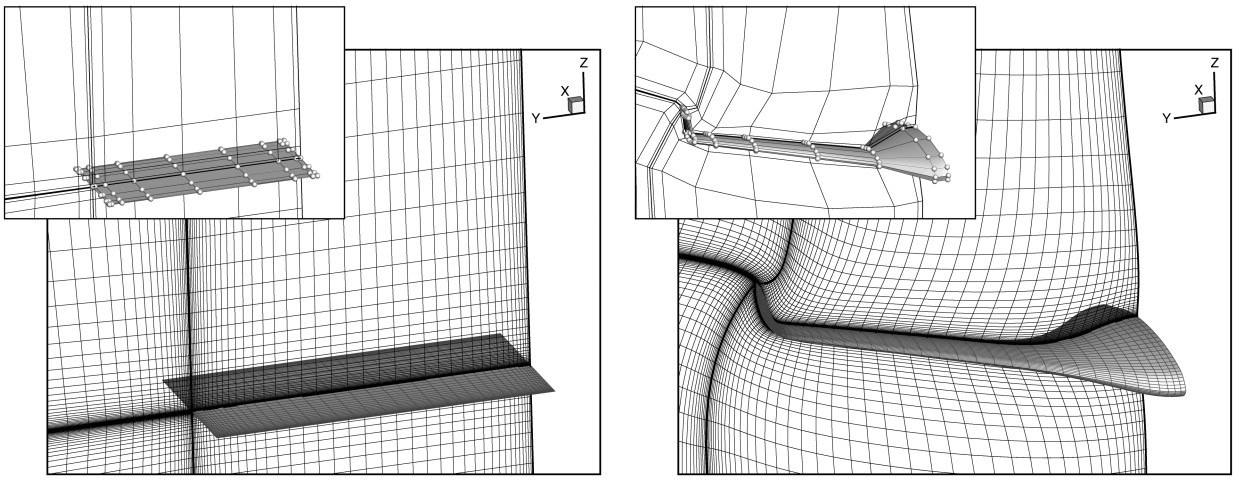
The two examples presented here share the same surface and block topology. To handle geometries with complicated surface features (e.g., wing-body junctions), it would be necessary to consider more general topologies. The issues posed by complex geometries are similar to those encountered in multiblock grid generation of the same geometries and can be accommodated using related blocking strategies. For example, geometries with kinks or junctions can be handled in a straightforward manner by joining B-spline surfaces along their edges.

III. Flow Analysis

The flow solver incorporated into the optimization algorithm uses a second-order-accurate finite difference discretization and a Newton–Krylov solution strategy. The solver is described briefly below and in detail by Hicken and Zingg [25].

A. Governing Equations and Discretization

We consider the three-dimensional Euler equations on multiblock structured grids. Applying a diffeomorphism from physical to computational space, the Euler equations become



a) Initial mesh and control grid (inset) **b) Final mesh and control grid (inset)**

Fig. 5 Control grids and flow-analysis meshes for the initial plate geometry (left) and final blended-wing body (right).

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033

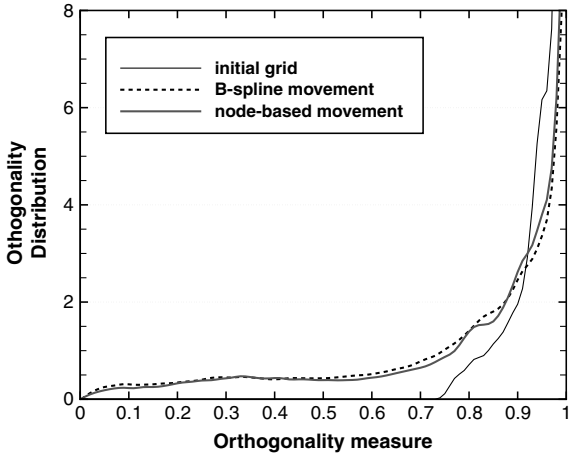


Fig. 6 Orthogonality distribution for the initial flat-plate grid and BWB grids.

$$\partial_t \hat{\mathbf{Q}} + \partial_{\xi_i} \hat{\mathbf{E}}_i = \mathbf{0} \tag{8}$$

where $\hat{\mathbf{Q}}$ is the vector of conservative flow variables scaled by the Jacobian of the mapping, and $\hat{\mathbf{E}}_i$ is the inviscid flux vector in the coordinate direction ξ_i.

The spatial derivatives in Eq. (8) are discretized using second-order-accurate summation-by-parts (SBP) operators [56]. Boundary conditions are imposed and blocks are coupled using simultaneous approximation terms (SATs) [57]. The SBP-SAT discretization is linearly time-stable, requires only C^0 mesh continuity at block interfaces, accommodates arbitrary block topologies, and has low interblock communication overhead. To suppress high-frequency modes, the discretization is augmented with combined second- and fourth-difference scalar dissipation [58,59] or matrix dissipation [60]. The discretization produces a set of nonlinear algebraic equations, which is represented by the vector equation

$$\mathcal{R}\left(\mathbf{q}, \mathbf{b}^{(m)}\right) = \mathbf{0} \tag{9}$$

where $\mathbf{q}$ is a block-column vector of the conservative flow variables. The flow residual depends on the B-spline control points $\mathbf{b}^{(m)}$, since these determine the nodal coordinates. When solving Eq. (9) for the flow variables, the control points are fixed.

B. Newton–Krylov Solution Algorithm

We solve the discretized Euler equations using a Newton–Krylov strategy. This strategy involves, for each major iteration n, a sparse linear system of the form

$$A^{(n)} \Delta \mathbf{q}^{(n)} = -\mathcal{R}^{(n)} \tag{10}$$

where $\mathcal{R}^{(n)} = \mathcal{R}(\mathbf{q}^{(n)})$, $\Delta \mathbf{q}^{(n)} = \mathbf{q}^{(n+1)} - \mathbf{q}^{(n)}$, and

$$A_{ij}^{(n)} = \frac{\partial \mathcal{R}_i^{(n)}}{\partial q_j} + E_{ij}^{(n)}$$

The Jacobian matrix $A^{(n)}$ is exact if the error $E^{(n)}$ is zero, and it is approximate otherwise.

Successful convergence of Newton's method depends on the initial iterate $\mathbf{q}^{(0)}$, which must be sufficiently close to the solution of Eq. (9). For this reason, our algorithm is broken into two phases: 1) an approximate-Newton startup phase to find a suitable initial iterate and 2) an inexact-Newton phase. The startup phase uses an approximate Jacobian based on a nearest-neighbor stencil and is similar to the implicit Euler time-marching method. The inexact-Newton phase gets its name from solving the Newton update (10) inexactly to a relative tolerance of 10^{-2} using a Krylov linear solver. The Krylov solver permits a Jacobian-free approach, since only Jacobian-vector products are needed and these are approximated using forward differences.

Although the matrix $A^{(n)}$ is different during the startup and inexact-Newton phases, the solution method for the linear equation remains the same. Specifically, we solve Eq. (10) in parallel using a preconditioned Krylov iterative method. Experience suggests that the generalized minimal residual method (GMRES) [38] is an efficient Krylov method for aerodynamic applications. We use flexible GMRES (FGMRES) [61] to accommodate iterative preconditioners.

Preconditioning is necessary when Krylov methods are used to solve ill-conditioned problems. Two parallel preconditioners are implemented in the solver: an additive-Schwarz preconditioner [62–64] with no overlap (block Jacobi) and an approximate-Schur preconditioner [65]. Both preconditioners require an ILU factorization of the nearest-neighbor approximate Jacobian. This matrix approximates the Jacobian, because it lumps the fourth-difference dissipation into the second-difference dissipation [66]. ILU(p) [53] with a fill level of 1 is applied locally to each processor's block of the approximate Jacobian to obtain the incomplete factorizations (i.e., the factorization involves no communication).

IV. Gradient Evaluation

Let $\mathcal{J}$ denote an objective function to be minimized: for example, drag or C_D/C_L. Let the vector of design variables be $\mathbf{v} = [\mathbf{v}_{\text{geo}}^T, \alpha]^T$, where $\mathbf{v}_{\text{geo}}$ are the geometric design variables and α is the angle of attack. Recall that the geometric design variables are the coordinates of the B-spline control points on the aerodynamic surface.

To find the gradient of $\mathcal{J}$ with respect to $\mathbf{v}$, we regard the mesh-movement and flow equations as constraints and introduce the Lagrangian function:

$$\mathcal{L} = \mathcal{J}(\mathbf{v}, \mathbf{b}^{(m)}, \mathbf{q}) + \sum_{i=1}^{m} \boldsymbol{\lambda}^{(i)T} \mathcal{M}^{(i)}(\mathbf{v}, \mathbf{b}^{(i-1)}, \mathbf{b}^{(i)}) + \boldsymbol{\psi}^T \mathcal{R}(\mathbf{v}, \mathbf{b}^{(m)}, \mathbf{q}) \tag{11}$$

The Lagrange multipliers $\{\boldsymbol{\lambda}^{(i)}\}_{i=1}^m$ and $\boldsymbol{\psi}$ are the mesh- and flow-adjoint variables, respectively. The first-order (necessary) optimality conditions are obtained by setting the partial derivatives of $\mathcal{L}$ to zero [67]. The partial derivatives with respect to the adjoint variables recover the mesh-movement and flow equations. Setting the partial derivatives with respect to $\mathbf{q}$ and $\{\mathbf{b}^{(i)}\}_{i=1}^m$ to zero yields

$$\left(\frac{\partial \mathcal{R}}{\partial \mathbf{q}}\right)^T \boldsymbol{\psi} = -\left(\frac{\partial \mathcal{J}}{\partial \mathbf{q}}\right)^T \tag{12}$$

$$\left(\frac{\partial \mathcal{M}^{(m)}}{\partial \mathbf{b}^{(m)}}\right)^T \boldsymbol{\lambda}^{(m)} = -\left(\frac{\partial \mathcal{J}}{\partial \mathbf{b}^{(m)}}\right)^T - \left(\frac{\partial \mathcal{R}}{\partial \mathbf{b}^{(m)}}\right)^T \boldsymbol{\psi} \tag{13}$$

$$\left(\frac{\partial \mathcal{M}^{(i)}}{\partial \mathbf{b}^{(i)}}\right)^T \boldsymbol{\lambda}^{(i)} = -\left(\frac{\partial \mathcal{M}^{(i+1)}}{\partial \mathbf{b}^{(i)}}\right)^T \boldsymbol{\lambda}^{(i+1)}, \quad i \in \{m-1, m-2, \ldots, 1\} \tag{14}$$

We follow the approach outlined in Truong et al. [20], and we drive the conditions (12–14) to zero using a sequential approach. The following subsections provide further details on these equations and the methods used to solve them.

Finally, the gradient of the objective function is given by the partial derivative of $\mathcal{L}$ with respect to $\mathbf{v}$:

$$\mathcal{G} \equiv \frac{\partial \mathcal{L}}{\partial \mathbf{v}} = \frac{\partial \mathcal{J}}{\partial \mathbf{v}} + \sum_{i=1}^{m} \left(\boldsymbol{\lambda}^{(i)T} \frac{\partial \mathcal{M}^{(i)}}{\partial \mathbf{v}}\right) + \boldsymbol{\psi}^T \frac{\partial \mathcal{R}}{\partial \mathbf{v}} \tag{15}$$

where $\mathcal{G}$ denotes the gradient of the objective. Note that the mesh-movement residuals depend on the design variables, since the variables determine the position of the surface control points. Unlike the partial derivatives in Eqs. (12–14), setting $\mathcal{G}$ to zero does not lead to a linear system we can use to solve for $\mathbf{v}$. Instead, we must drive $\mathcal{G}$ to zero using a nonlinear optimizer (see Sec. V).

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033

A. Flow-Adjoint Equation

The flow-adjoint variables are governed by the linear equation

$$A^T\boldsymbol{\psi} = -\left(\frac{\partial \mathcal{J}}{\partial \mathbf{q}}\right)^T \tag{12\prime}$$

If we choose to use a Krylov iterative method to solve Eq. (12′), then we need only evaluate generic transposed-Jacobian products of the form $A^T\mathbf{z}$, since Krylov methods need these products and not necessarily the matrix itself.

Unlike the Jacobian-vector products used in the flow solver, the transposed-Jacobian-vector products cannot be approximated using finite differences. However, the transposed-Jacobian products can be evaluated on-the-fly [68] or using reverse-mode automatic differentiation of the residual [69]. These approaches offer reduced memory requirements, since the Jacobian does not need to be stored. Nevertheless, we prefer evaluating and storing the Jacobian and subsequently performing the transposed products explicitly. We have found that this approach is more efficient than the methods above, because the explicit products are faster and the cost of computing the Jacobian is amortized over the total time needed to converge the adjoint equations.

There are several methods available for computing the Jacobian matrix. Nielsen and Kleb [70] used the complex-step method [40] with coloring to efficiently and accurately evaluate the entries of the Jacobian matrix. Mader et al. [71] constructed the residual on a node-by-node basis and then evaluated each row of the transposed Jacobian by applying the reverse mode of automatic differentiation.

We use a combination of analytical and complex-step differentiation to evaluate the Jacobian matrix. The Euler fluxes and numerical dissipation are differentiated analytically. Much of this linearization is already available from the approximate Jacobian, which is used to build the preconditioner for both the primal and dual-flow problems. This reusability is one of the advantages of the Newton–Krylov approach when calculating the adjoint.

The SAT operators that couple blocks and impose boundary conditions have the following form:

$$\boldsymbol{\Delta}\,\mathbf{f}^{\pm} = \tfrac{1}{2}(|A| \pm A)(\mathbf{q} - \mathbf{q}_{\text{bc}}) \tag{16}$$

where $\mathbf{q}_{\text{bc}}$ contains either boundary data or neighboring block variables. We differentiate the SAT operators using the complex-step method [40,42], which simplifies their linearization. SAT terms appear only in the equations at block interfaces and boundaries, so the application of the complex-step method has a negligible effect on CPU time. There is no benefit to using the reverse-mode here, since the number of inputs and outputs is equal.

1. *Flow Jacobian Verification*

To verify the accuracy of the Jacobian matrix, we implemented a complex-variable version of the flow residual. When applied to the flow residual, the complex-step method provides a second-order-accurate approximation of the matrix-vector product $A\mathbf{z}$. Unlike finite difference approximations, the complex-step method does not experience subtractive cancellation errors as the step size is reduced. Thus, the truncation error in the complex-step approximation can be reduced to machine accuracy by choosing a sufficiently small step size.

Figure 7 shows the difference between matrix-vector products evaluated using the Jacobian matrix and products evaluated using the complex-step method, for various step sizes. The same random vector $\mathbf{z}$ is used for both products, and the second-difference dissipation coefficient is set to zero. Similar results are produced with distinct $\mathbf{z}$, so we conclude that the Jacobian matrix is accurate to machine error.

2. *Iterative Solution of the Flow-Adjoint Equation*

To obtain a gradient accurate to 10^{-6}, the flow-adjoint system must be solved to a relative tolerance of 10^{-8} [72]. This tolerance requires a considerable number of Krylov iterations, unlike the larger tolerance of 10^{-2} used for the linear systems of the inexact-Newton flow solver.

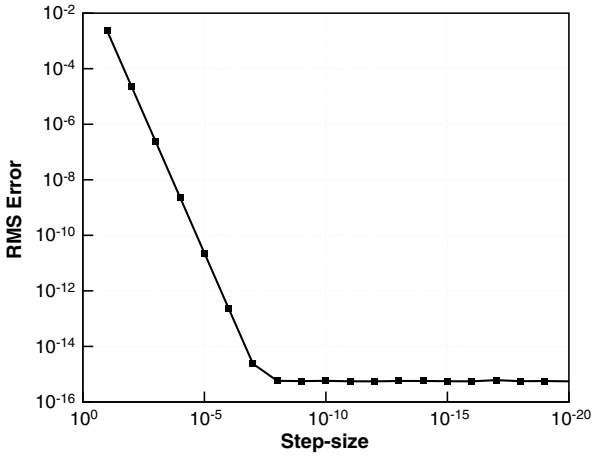


Fig. 7 Verification of the analytical Jacobian matrix using the complex-step method.

The memory requirements of GMRES and its flexible variant FGMRES grow linearly with the number of iterations. This can cause problems when GMRES is applied to the adjoint problem and memory is limited. One way to reduce the memory burden is to use restarted versions of GMRES or FGMRES, denoted as GMRES(m) and FGMRES(m). These solvers simply restart after every m Krylov iterations, which keeps memory requirements proportional to m. Unfortunately, restarted Krylov solvers often exhibit degraded and, in some cases, stalled convergence.

To address this, we have developed a flexible variant of the Krylov method GCROT [73], called GCROT(m, k) [74], which uses the same amount of memory as FGMRES ($m + k$). Unlike restarted FGMRES, GCROT does not discard the entire Krylov subspace each time it restarts but instead maintains a set of vectors from one outer iteration to the next. This nested-subspace strategy has been shown to perform very well with respect to full GMRES while maintaining a Krylov subspace of fixed size [73].

To demonstrate the performance of GCROT(m, k), we consider the solution of the flow-adjoint variables corresponding to $\mathcal{J} = L$, where L is the lift, on a 1-million-node mesh using 12 processors. Figure 8 plots the L^2-norm of the (relative) adjoint system residual versus CPU time in seconds. Results were obtained on a Beowulf-class cluster consisting of Itanium 2 processors with a 6 MB L3 cache and a clock speed of 1500 MHz. Typically, the flow solver requires approximately 30 min to converge 10 orders of magnitude for this size of mesh, number of processors, and architecture. Thus, Fig. 8 indicates that GCROT solves the adjoint system to a relative tolerance of 10^{-8} in 50% of the time needed to converge the flow solution. For the low-memory case, observe that FGMRES(20) stalls while GCROT(10,10) converges. When more memory is available

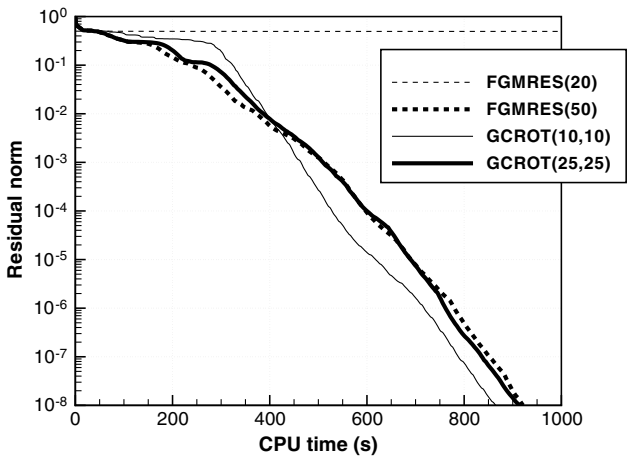


Fig. 8 Comparison of CPU times for FGMRES and GCROT applied to the flow-adjoint problem.

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033

[i.e., FGMRES(50) and GCROT(25,25)], the performances of the two solvers are similar. Our experience suggests that these results are typical: when FGMRES $(m + k)$ converges, GCROT(m, k) converges with similar CPU time; when FGMRES$(m + k)$ stalls, GCROT(m, k) converges.

B. Mesh-Adjoint Equations

There are two types of B-spline mesh-adjoint equations: namely, Eqs. (13) and (14). The system matrix appearing in these equations can be found by differentiating the control mesh-movement equation (4) with respect to $\mathbf{b}^{(i)}$:

$$\left(\frac{\partial \mathcal{M}^{(i)}}{\partial \mathbf{b}^{(i)}}\right)^T = K^{(i)T} = K^{(i)}, \qquad i \in \{m, m-1, \ldots, 1\}$$

where we have used the symmetry of the stiffness matrix $K^{(i)}$. The symmetry of the stiffness matrix allows the mesh-movement solution algorithm to be reused for the mesh adjoints; hence, we use the conjugate gradient method preconditioned with ILU(1) to solve both Eqs. (13) and (14).

Unlike the left-hand sides, the right-hand sides of Eqs. (13) and (14) are very different. To evaluate the right-hand side (RHS) of the adjoint equation (13), we make liberal use of the chain rule:

$$-\left(\frac{\partial \mathcal{J}}{\partial \mathbf{b}^{(m)}}\right)^T - \left(\frac{\partial \mathcal{R}}{\partial \mathbf{b}^{(m)}}\right)^T \boldsymbol{\psi} = -\left(\frac{\partial \mathbf{g}}{\partial \mathbf{b}^{(m)}}\right)^T \left[\left.\frac{\partial \mathcal{J}}{\partial \mathbf{g}}\right|_{\mathbf{m}} + \left(\left.\frac{\partial \mathcal{J}}{\partial \mathbf{m}}\right|_{\mathbf{g}} + \boldsymbol{\psi}^T \frac{\partial \mathcal{R}}{\partial \mathbf{m}}\right)\frac{\partial \mathbf{m}}{\partial \mathbf{g}}\right]^T \tag{17}$$

where $\mathbf{g}$ and $\mathbf{m}$ are block-column vectors of the grid coordinates and metrics, respectively. The blocks in $\mathbf{g}$ are composed of $\mathbf{x} = (x, y, z)$ at each node, with the coordinates defined by the B-spline volume-mesh equation (1). Similarly, the blocks in $\mathbf{m}$ consist of the nine components of $\nabla\xi_i$ at each node. The term $\partial\mathcal{J}/\partial\mathbf{g}|_{\mathbf{m}}$ denotes the partial derivative of the objective with respect to the grid coordinates while freezing the metric terms (similarly for $\partial\mathcal{J}/\partial\mathbf{m}|_{\mathbf{g}}$). Equation (17) provides a right-hand-side reformulation that is significantly easier to implement. Note that none of the matrices appearing in Eq. (17) are stored, since only the resulting vector is needed.

When the number of increments is greater than one, we must solve the additional adjoint equations (14). Again, the difficulty presented by these equations is evaluating their right-hand sides. The movement residual $\mathcal{M}^{(i+1)}$ has a complicated nonlinear dependence on the control points $\mathbf{b}^{(i)}$ through Young's modulus (5); therefore, in the present work, we evaluate the right-hand sides of Eq. (14) using the complex-step method. Evaluating the right-hand sides in this way typically requires more CPU time than solving the mesh-adjoint systems. However, the relatively small control grid ensures that the impact on the total CPU time is small, as we demonstrate in Sec. IV.D.

C. Verification of Gradient Accuracy

Given the complexity of the present algorithm and the use of hand differentiation, verifying the gradient accuracy is essential. Our goal in this section is to demonstrate that the gradient is sufficiently accurate for gradient-based optimization.

Consider a 12-block mesh around a generic wing with no sweep (see Fig. 9). Each block consists of $23 \times 33 \times 17$ nodes and is fit with B-spline volumes. The wing is parameterized using the B-spline control points corresponding to the surface. These control points are depicted as white spheres in Fig. 9. In total, there are 298 design variables: 297 geometric variables plus the angle of attack.

Each component of an objective-function gradient can be verified using a finite difference approximation; however, this would be time-consuming for the number of design variables considered here. Instead, we use a directional derivative to check all the gradient components simultaneously. For a given direction $\mathbf{d}$, the directional derivative is given by

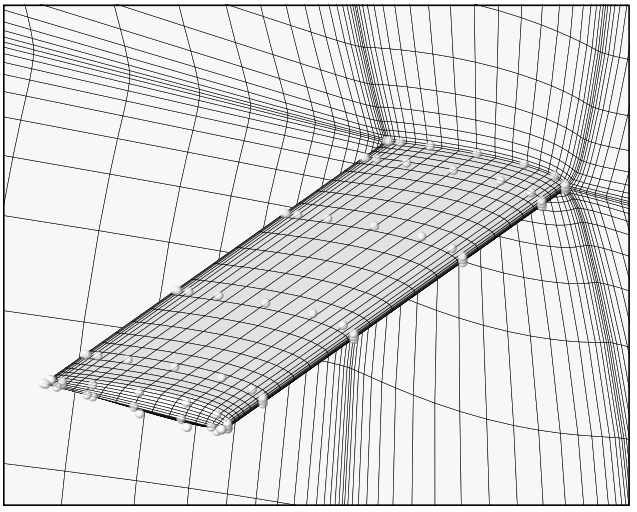


Fig. 9 Generic wing with parameterizing control points (white spheres) used for the gradient accuracy verification.

$$D_{\mathbf{d}}\mathcal{J} = \mathcal{G} \cdot \mathbf{d}$$

and a second-order finite difference approximation is

$$D_{\mathbf{d}}\mathcal{J} = \frac{\mathcal{J}(\mathbf{v} + \epsilon\mathbf{d}) - \mathcal{J}(\mathbf{v} - \epsilon\mathbf{d})}{2\epsilon} + \mathcal{O}(\epsilon^2)$$

where ϵ is a perturbation parameter. In the finite difference approximation, the perturbation to the design variables is propagated through the entire algorithm; that is, this is an approximation to the total derivative of $\mathcal{J}$ and not the partial derivative.

In principle, one could use a complex-step-based directional derivative to test the gradient accuracy; however, the additional accuracy of such a test is not justified, given the costs of implementing a complex-variable version of the Newton–Krylov solution algorithm.

Individual components of the gradient can differ in magnitude by 2–4 orders; therefore, it is tempting to choose a direction $\mathbf{d}$ such that each element of the gradient makes an equal contribution to $D_{\mathbf{d}}\mathcal{J}$. However, this tends to increase the step-size range over which round-off errors affect the finite difference approximation. Instead, we use the direction

$$(\mathbf{d})_i = \text{sign}[(\mathcal{G})_i]$$

which gives a directional derivative equal to the L^1 norm of the gradient. This direction does not eliminate the possibility that large gradient components will overwhelm small gradient components, but it does prevent subtractive cancellation between gradient components.

Figure 10 plots the relative error between the adjoint-based and finite difference values of $D_{\mathbf{d}}\mathcal{J}$, for both $\mathcal{J} = L$ and $\mathcal{J} = D$. For each objective, the freestream Mach number is fixed at 0.5 and the angle of attack is fixed at 4 deg. The plot shows the expected second-order convergence of the finite difference approximation and its eventual contamination by round-off errors. These results suggest that the adjoint-based gradients are at least as accurate as finite difference approximations with optimal step sizes.

D. CPU-Time Breakdown

Table 1 lists a CPU-time breakdown of the objective function and gradient evaluations. The times are normalized by the total time required to calculate the objective function (the first row in bold). To ensure the gradient is accurate to 10^{-6}, the flow and flow-adjoint equations are converged 10 and 8 orders, respectively. The mesh-movement and mesh-adjoint equations are converged 12 orders; while this tolerance may be unnecessary for the desired gradient accuracy, the additional CPU time is insignificant. Times are

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033

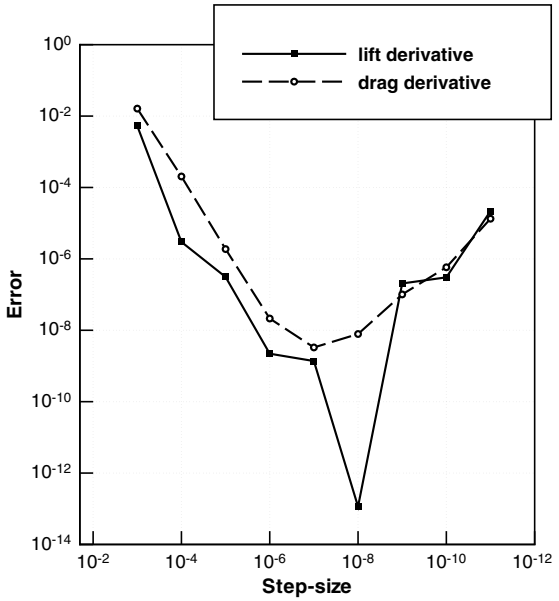


Fig. 10 Relative error between the adjoint-based and second-order-accurate finite difference values of the directional derivative $D_{\mathbf{d}}\mathcal{J}$.

provided for two grid sizes to illustrate how the components of the algorithm scale.

The two examples in Table 1 use B-spline volumes with the same number of control points. Consequently, the mesh-movement and mesh-adjoint routines have a very weak dependence on grid size. This is reflected in the normalized CPU times, which differ by an order of magnitude between the two grids. While the mesh-movement and mesh-adjoint CPU times are relatively significant for the coarse grid, they represent only 3.4% of the total computational effort on the fine grid. Thus, B-spline mesh movement is very efficient for the fine grid, which is representative of the grids used in practice. On the fine grid, the total time needed to compute the gradient is less than one-half of that needed for the objective.

V. Optimization Algorithm

We use the software package SNOPT [43] to solve nonlinear optimization problems with general constraints. SNOPT uses a sequential-quadratic-programing algorithm that is capable of handling both linear and nonlinear constraints.

SNOPT measures convergence using the first-order optimality conditions, also known as the Karush–Kuhn–Tucker (KKT) conditions. Let $\mathbf{c}$ and $\boldsymbol{\pi}$ denote the SNOPT constraints and Lagrange multipliers, respectively. An element of $\mathbf{c}$ may be, for example, a lift constraint, an area constraint, a span constraint, etc. The KKT conditions are given by [75]

$$\begin{aligned}
&\nabla_{\mathbf{v}}(\mathcal{J} + \boldsymbol{\pi}^T\mathbf{c}) = 0\\
&c_i(\mathbf{v}) = 0 \qquad \forall\ i \in E\\
&c_i(\mathbf{v}) \geq 0 \qquad \forall\ i \in I\\
&\pi_i \geq 0 \qquad \forall\ i \in I\\
&\pi_i c_i(\mathbf{v}) = 0 \quad \forall\ i \in E \cup I
\end{aligned}$$

where E and I denote the set of equality and inequality constraints, respectively. For an optimization problem to be considered converged, SNOPT requires the KKT conditions to be satisfied to within a specified tolerance ε. We use a tolerance of $\varepsilon = 10^{-7}$ for the cases presented here.

The Hessian of the SNOPT Lagrangian, $\mathcal{J} + \boldsymbol{\pi}^T\mathbf{c}$, is approximated using the quasi-Newton method of Broyden–Fletcher–Goldfarb–Shanno (BFGS). We use the full-memory BFGS update rather than the limited-memory option, since the storage requirements are modest relative to the flow solver.

If a lift constraint is imposed, then its gradient must be calculated, which requires additional flow- and mesh-adjoint solutions for lift. As shown above, the adjoint solution process is very efficient, and adding this constraint represents an approximately 30% increase in CPU time per optimization cycle. An alternative, not explored in this work, is to impose the lift constraint as an equation in the flow solver and to add the angle of attack as a variable to the vector $\mathbf{q}$. Billing [76] and Zingg and Billing [77] have shown that this approach increases the flow solver time between 20–50%, so the total CPU time per optimization cycle is comparable to solving the lift-adjoint problem.

The optimization process may be curtailed due to time limits imposed by a queuing system. In such cases, it is desirable to warm-start the optimization using the previously calculated objectives and gradients. Although SNOPT does not provide a warm-start capability,[§] we have implemented a simple strategy that is transparent to the optimization. SNOPT, like most gradient-based algorithms, is deterministic and will follow the same path if given the same information. Therefore, it is sufficient to store the objective, constraint, and gradient values in the order they are produced and read them into SNOPT in the same order during a warm start. Once all information in the file has been exhausted, new objectives and gradients are evaluated and stored.

As a simple verification of the optimizer, we consider an inverse design based on surface pressure. The design variables consist of the angle of attack and the three coordinates of a control point on the upper surface of the wing shown in Fig. 9. The initial angle of attack is 4 deg. A target pressure distribution is obtained by randomly perturbing the four design variables and solving for the flow. The optimizer is given the unperturbed wing and angle of attack and attempts to recover the perturbed variables using the objective

$$\mathcal{J} = \frac{1}{2}\sum_{i=1}^{N_{\text{surf}}}(p_i - p_{i,\text{targ}})^2\Delta A_i$$

where N_{surf} is the total number of surface nodes, and ΔA_i is the surface area element at node i. The pressure and target pressure at node i are denoted by p_i and $p_{i,\text{targ}}$, respectively.

Figure 11 shows the convergence history for the inverse design problem. The gradient converges 10 orders and the objective converges 20 orders in 25 objective function and gradient evaluations. These convergence tolerances are smaller than those typically used in aerodynamic shape optimization, but they help demonstrate the accuracy of the gradient. Note, in particular, the superlinear convergence of the objective function.

VI. Lift-Constrained Induced-Drag Minimization

Ultimately, we wish to explore the possibility of using high-fidelity optimization to uncover novel configurations. However, any algorithm designed to find novel configurations should, if suitably constrained, recover known results. We can use this idea to demonstrate the capabilities of the optimization algorithm. In particular, we can begin with a low-aspect-ratio wing and minimize induced drag under lift and bound constraints, and the algorithm should maximize the span and the vertical extent of the wing tip.

The initial geometry is a rectangular wing with a root chord of 1, a span of $b = 3$, and NACA 0012 sections. The aspect ratio based on the surface area is 2.917. A five-block mesh surrounds the geometry with 4.05×10^5 nodes. The surface mesh consists of 5850 nodes, and the initial tip, leading-edge, trailing-edge, and offwall mesh spacing is 10^{-3}.

The blocks are associated with B-spline volumes, and the volume mesh is parameterized with 1404 control points. The semispan wing is parameterized using six spanwise stations of control points, with nine control points in the streamwise direction on both the upper and lower surfaces. The parameterization is similar to the one shown in Fig. 9. There are 251 geometric degrees of freedom, after accounting

[§]SNOPT does have a limited version of warm-starting that restarts the optimization with the current design variables and estimates for the Lagrange multipliers; however, the BFGS Hessian approximation is discarded.

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033

Table 1 Breakdown of objective function and gradient CPU times for two grids[a]

	Grid size (nodes)			
	1.55×10^5 CPU time (12 proc.)		1.158×10^6 CPU time (12 proc.)	
	Relative	Absolute	Relative	Absolute
Objective function components				
Mesh movement	0.121	——	0.014	——
Flow solution	0.877	——	0.986	——
Total	1.000	(107.0 s)	1.000	(1278.1 s)
Gradient components				
Flow Jacobian assembly	0.005	——	0.003	——
ILU(2) factorization	0.032	——	0.020	——
GCROT flow-adjoint solution	0.366	——	0.397	——
Mesh Jacobian assembly	0.007	——	0.001	——
ILU(1) factorization	0.020	——	0.002	——
PCG mesh-adjoint solution	0.083	——	0.008	——
Complex-step RHS assembly	0.294	——	0.025	——
Total	0.812	(86.8 s)	0.457	(584.0 s)

[a]Relative times are normalized by the total objective-function evaluation time.

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033

for coincident control points and ignoring the y coordinate of control points on the symmetry plane (i.e., the geometry cannot move in the spanwise direction at the symmetry plane). Including the angle of attack gives 252 degrees of freedom in total.

When the flow is subsonic and the trailing wake is planar, the optimal (elliptical) loading can be achieved by varying the spanwise distribution of twist, chord, or zero-lift angle. Therefore, the number of degrees of freedom must be reduced to avoid nonunique inviscid designs. For this example, the following constraints are imposed.

For this example, the following constraints are imposed:

1) Each spanwise station of control points is limited to rotations and scalings defined by the leading edge (LE) and trailing edge (TE). For each spanwise station, $k = 1, \ldots, 6$, we have

$$\begin{pmatrix} x - x_{\mathrm{LE}} \\ z - z_{\mathrm{LE}} \end{pmatrix}_k = r_k \begin{bmatrix} \cos(\beta_k) & -\sin(\beta_k) \\ \sin(\beta_k) & \cos(\beta_k) \end{bmatrix} \begin{pmatrix} x_{\mathrm{TE}} - x_{\mathrm{LE}} \\ z_{\mathrm{TE}} - z_{\mathrm{LE}} \end{pmatrix}_k$$

The parameters

$$r_k = \| \mathbf{x} - \mathbf{x}_{\mathrm{LE}} \| \; / \; \| \mathbf{x}_{\mathrm{TE}} - \mathbf{x}_{\mathrm{LE}} \|$$

and

$$\beta_k = \arctan[(z - z_{\mathrm{LE}})/(x - x_{\mathrm{LE}})]$$

are based on the initial values of the control-point coordinates.

2) The y coordinate of the spanwise stations $k = 2, \ldots, 6$ maintains the same ratio with the wing span according to

$$y_k = \lambda y_{\max}$$

where $\lambda = (y_k / y_{\max})$ is the initial ratio.

3) To prevent streamwise translation and sweep, the quarter-chord of each spanwise station is fixed to its initial value $x_{\mathrm{qc},k}$:

$$\frac{3}{4} x_{\mathrm{LE},k} + \frac{1}{4} x_{\mathrm{TE},k} = x_{\mathrm{qc},k}$$

4) The distance between the leading and trailing edges of the spanwise sections ($k = 2, \ldots, 6$) is equal to the root chord ($k = 1$):

$$\| \mathbf{x}_{\mathrm{LE},k} - \mathbf{x}_{\mathrm{TE},k} \| = \| \mathbf{x}_{\mathrm{LE},1} - \mathbf{x}_{\mathrm{TE},1} \|$$

This constraint ensures that chord-length changes cannot be used to optimally load the design.

5) The z coordinates of the leading and trailing edges at the wing tip are extrapolated from the two neighboring stations: that is,

$$z_{\mathrm{LE},k} - z_{\mathrm{LE},k-1} = \sigma(z_{\mathrm{LE},k-1} - z_{\mathrm{LE},k-2})$$

where

$$\sigma = (y_{\mathrm{LE},k} - y_{\mathrm{LE},k-1}) / (y_{\mathrm{LE},k-1} - y_{\mathrm{LE},k-2})$$

An analogous constraint is applied at the trailing edge. These constraints prevent excessive stretching of the surface mesh, which may result if the tip station is displaced independently in the vertical direction.

With the exception of the chord-length constraint 4, the above constraints are linear. Linear constraints are satisfied exactly by SNOPT at each design iteration and effectively redefine the design variables. Consequently, the effective design variables for this example are 1) the twist at the five inboard semispanwise stations, 2) the vertical position of the five inboard stations, 3) the wing span, and 4) the root chord.

The vertical position and twist of the tip are not design variables, because they are determined implicitly by the inboard stations through the constraints. The chord-length constraint is nonlinear, so it is not satisfied exactly at each design iteration. Nevertheless, it is satisfied at convergence, so chord-length can be considered an effective design variable.

Bounds on the spanwise and vertical coordinates are necessary to prevent wings of infinite span or winglets of infinite height. We arbitrarily use the box constraints $|y| \leq 2.5$ and $|z| \leq 0.25$ for this example. Constraints are also imposed on the lift and reference area, which implies that the lift coefficient is constrained. The reference area is the surface area of the geometry and is constrained by its initial value of $S = 3.08554$. The target lift is chosen such that $C_L = 0.325$ when the lift constraint is satisfied. The optimization begins with an angle of attack that produces the target lift.

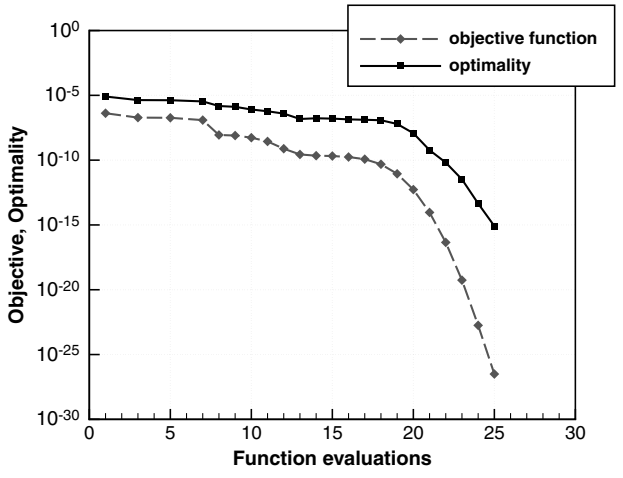


Fig. 11 Convergence history for the inverse design verification.

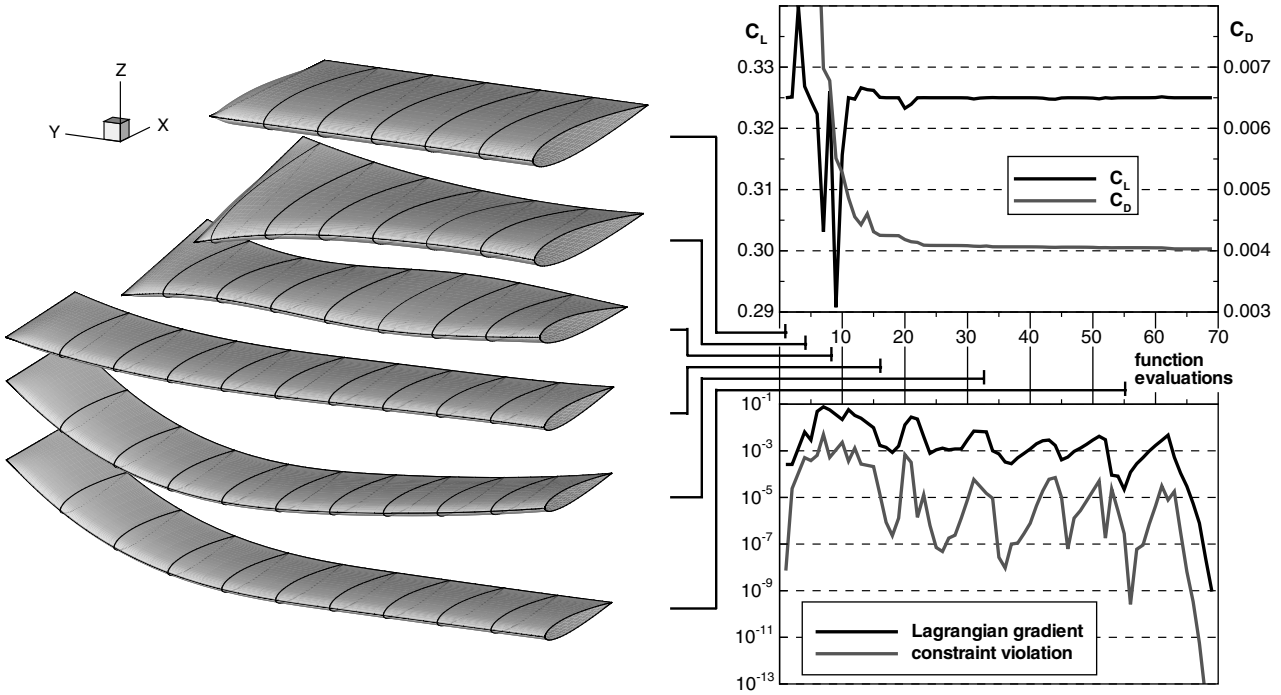


Fig. 12 Convergence history of the lift-constrained induced-drag minimization, including the semispan geometry at various points in the history.

We consider a single-point optimization with the Mach number fixed at 0.5. Practical aerodynamic optimizations must consider multiple operating conditions (see [78] for an example), and so we reiterate that the present problem is intended to demonstrate the algorithm by recovering expected theoretical results.

Figure 12 shows the optimization convergence history using several metrics. The SNOPT optimality conditions are satisfied to 10^{-7} after 68 function evaluations (a function evaluation includes computation of the objective, the constraints, and all gradients). The total optimization requires approximately 42 h using five Itanium 2 processors. After 32 function evaluations, or approximately 20 h, the coefficient of lift and drag are accurate to three and two digits, respectively.

Intermediate designs from the optimization process are also shown in Fig. 12. As expected, the algorithm maximizes the span and height of the configuration within the given box constraints (i.e., $b = 5$ and $h = 0.5$). The aspect ratio b^2/S of the final geometry is 8.102.

The intermediate designs demonstrate the range of geometries handled by the geometry parameterization and mesh movement. They also reveal the relative importance of various variables on the design. Although twist is the first variable to be exploited (iterations 1 to 4), we see that most of the reduction in induced drag is achieved by increasing span (reflected in the geometries from iterations 8 and 16). The remaining improvements are accomplished by changing the vertical position and twist of the sections. Of these, the vertical extent at the tip provides the most reduction, which is consistent with the remarks in [79]. The vertical position of the wing root is established toward the end of the optimization, and the convergence history indicates that the drag is relatively insensitive to this variable.

Lifting-line theory predicts that an elliptical spanwise lift distribution produces the minimum induced drag for a wing with a planar wake. The coefficient of drag for an elliptically loaded planar wing is given by

$$C_{D,\text{ellip}} = \frac{C_L^2 S}{\pi b^2} \tag{18}$$

where b is the span and S is the reference area. Recall that both C_L and S are constrained here. The coefficient of drag for the initial geometry is $C_D = 0.0120$. This is approximately 4% higher than the drag coefficient predicted by Eq. (18) for $b = 3$, which indicates that the initial design is not optimally loaded. In contrast, the coefficient of drag at the end of the optimization is $C_D = 0.00403$, which is 3% lower than $C_{D,\text{ellip}} = 0.00415$ (the induced-drag coefficient of an elliptically loaded planar wing with an equivalent span of $b = 5$). The optimal geometry achieves a lower induced drag, because the winglets produce a nonplanar wake. As mentioned above, most of the reduction in drag from the initial value of $C_D = 0.0120$ to the final value of $C_D = 0.00403$ is due to the increase in span [i.e., the b^{-2} factor in Eq. (18)].

Finally, we conducted a grid refinement study to confirm the performance of the final design. We refined the grid by a factor of 4 in each coordinate direction and broke each block into 64 individual blocks. Hence, the refined grid consists of 320 blocks and approximately 25.9×10^6 nodes. The refined-grid node locations were determined using the B-spline volumes. One of the advantages of the present approach is the ease with which refinement studies can be carried out after an optimization.

Using the refined grid, the coefficients of lift and drag were found to be $C_L = 0.3255$ and $C_D = 0.003935$, respectively. Based on this coefficient of lift, and the refined-surface-area value of 3.08630, the coefficient of drag for a planar wing with elliptical loading is predicted to be $C_{D,\text{ellip}} = 0.004163$. Thus, rather than 3%, the optimized design produces closer to 5% lower induced drag than predicted by

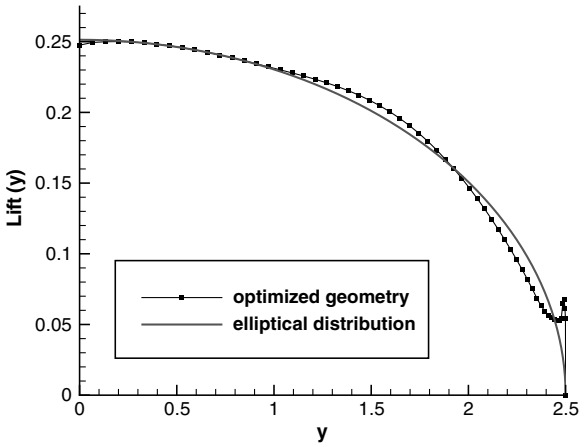


Fig. 13 Lift distribution of the optimized geometry (from the refined grid) and an elliptical distribution with the same total lift. Note that the data points do not reflect the grid resolution.

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033

lifting-line theory for an optimally loaded planar wing. Figure 13 compares the spanwise lift distribution of the optimized geometry, obtained on the refined grid, with an elliptical lift distribution. There is a marked difference in the two distributions, as expected from the nonplanar optimized geometry.

VII. Conclusions

To what extent can numerical optimization be used to find novel and efficient aerodynamic configurations? As a first step toward answering this question, we have developed a robust and efficient optimizer for the Euler equations.

An important aspect of the present work is an integrated parameterization and mesh movement that is flexible enough to explore a wide range of designs. The integrated approach uses B-spline volumes to parameterize the mesh, substantially reducing the number of degrees of freedom used in the mesh movement. B-spline mesh movement produces grids with quality comparable to those obtained using node-based linear-elasticity mesh movement, while requiring two to three orders less CPU time. The integrated approach also reduces the cost of the mesh-adjoint solution, despite considerable use of complex-step differentiation. For typical grid densities, the mesh-movement and mesh-adjoint solutions represent less than 5% of the total computational work.

For the solution of the flow-adjoint equations, we have shown that a Krylov iterative solver with a nested subspace (GCROT) is more robust than restarted flexible GMRES with equivalent memory requirements. Using the mesh-movement and flow adjoints, the gradient of an objective is evaluated in less than 50% of the time needed for a flow solution on typical meshes.

The algorithm components are integrated with the optimizer SNOPT and demonstrated on a lift-constrained induced-drag minimization of a rectangular wing. The optimizer correctly maximizes the span of the wing and the vertical extent of the wing tip. Partially converged results are available in 20 h with three- and two-digit convergence in the lift and drag, respectively, using five processors. Full convergence of the KKT conditions is achieved in 42 h. This case establishes that the algorithm can efficiently solve difficult optimization problems that include large changes in the initial geometry. Future work will include incorporating the Reynolds-averaged Navier–Stokes equations, a turbulence model, and multipoint optimization.

Acknowledgments

The authors gratefully acknowledge financial assistance from the Natural Sciences and Engineering Research Council (NSERC), the Canada Research Chairs program, Mathematics of Information Technology and Complex Systems (MITACS), and the University of Toronto. In addition, the authors acknowledge the SciNet Consortium for providing high-performance computing resources used for the refinement study reported within this paper.

References

[1] "Climate Change 2007: IPCC Synthesis Report," Intergovernmental Panel on Climate Change, World Meteorological Organization, Geneva, Nov. 2007.

[2] Deffeyes, K. S., *Hubbert's Peak—The Impending World Oil Shortage*, Princeton Univ. Press, Princeton, NJ, 2001.

[3] Goodstein, D., *Out of Gas: The End of the Age of Oil*, W. W. Norton, New York, 2004.

[4] *The Potential Use of Alternative Fuels for Aviation*, Seventh Meeting of the Committee on Aviation Environmental Protection (CAEP), International Civil Aviation Organization, Rept. CAEP/7-IP/28, Montréal, Feb. 2007.

[5] Liebeck, R., "Design of the blended wing body subsonic transport," *Journal of Aircraft*, Vol. 41, No. 1, 2004, pp. 10–25. doi:10.2514/1.9084

[6] de Boor, C., *A Practical Guide to Splines*, revised ed., Springer–Verlag, Berlin, 2001.

[7] Brunoa, O. P., Hana, Y., and Pohlmanb, M. M., "Accurate, High-Order Representation of Complex Three-Dimensional Surfaces via Fourier Continuation Analysis," *Journal of Computational Physics*, Vol. 227, No. 2, 2007, pp. 1094–1125. doi:10.1016/j.jcp.2007.08.029

[8] Jones, W. T., "A Grid Generation System for Multi-Disciplinary Design Optimization," 12th AIAA Computational Fluid Dynamics Conference, AIAA Paper 1995-1689, San Diego, CA, June 1995.

[9] Reuther, J., Jameson, A., Farmer, J., Martinelli, L., and Saunders, D., "Aerodynamic Shape Optimization of Complex Aircraft Configurations via an Adjoint Formulation," 34rd AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 1996-0094, Reno, NV, 1996.

[10] Nemec, M., Zingg, D. W., and Pulliam, T. H., "Multipoint and Multi-Objective Aerodynamic Shape Optimization," *AIAA Journal*, Vol. 42, No. 6, 2004, pp. 1057–1065. doi:10.2514/1.10415

[11] Liu, X., Qin, N., and Xia, H., "Fast Dynamic Grid Deformation Based on Delaunay Graph Mapping," *Journal of Computational Physics*, Vol. 211, No. 2, 2006, pp. 405–423. doi:10.1016/j.jcp.2005.05.025

[12] Batina, J. T., "Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes," *AIAA Journal*, Vol. 28, No. 8, Aug. 1990, pp. 1381–1388. doi:10.2514/3.25229

[13] Nielsen, E. J., and Anderson, W. K., "Recent Improvements in Aerodynamic Design Optimization on Unstructured Meshes," *AIAA Journal*, Vol. 40, No. 6, June 2002, pp. 1155–1163. doi:10.2514/2.1765

[14] Farhat, C., Degand, C., Koobus, B., and Lesoinne, M., "Torsional Springs for Two-Dimensional Dynamic Unstructured Fluid Meshes," *Computer Methods in Applied Mechanics and Engineering*, Vol. 163, Nso. 1–4, 1998, pp. 231–245. doi:10.1016/S0045-7825(98)00016-4

[15] Degand, C., and Farhat, C., "A Three-Dimensional Torsional Spring Analogy Method for Unstructured Dynamic Meshes," *Computers and Structures*, Vol. 80, No. 3–4, Feb. 2002, pp. 305–316. doi:10.1016/S0045-7949(02)00002-0

[16] Blom, F. J., "Considerations on the Spring Analogy," *International Journal for Numerical Methods in Fluids*, Vol. 32, No. 6, March 2000, pp. 647–688. doi:10.1002/(SICI)1097-0363(20000330)32:6<647::AID-FLD979>3.0.CO;2-K

[17] Zeng, D., and Ethier, C. R., "A Semi-Torsional Spring Analogy Model for Updating Unstructured Meshes in 3D Moving Domains," *Finite Elements in Analysis and Design*, Vol. 41, Nos. 11–12, June 2005, pp. 1118–1139. doi:10.1016/j.finel.2005.01.003

[18] Johnson, A. A., and Tezduyar, T. E., "Mesh Update Strategies in Parallel Finite Element Computations of Flow Problems with Moving Boundaries and Interfaces," *Computer Methods in Applied Mechanics and Engineering*, Vol. 119, Nos. 1–2, 1994, pp. 73–94. doi:10.1016/0045-7825(94)00077-8

[19] Yang, Z., and Mavriplis, D. J., "Unstructured Dynamic Meshes with Higher-Order Time Integration Schemes for the Unsteady Navier-Stokes Equations," 43rd AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2005-1222, Reno, NV, 2005.

[20] Truong, A. H., Oldfield, C. A., and Zingg, D. W., "Mesh Movement for a Discrete-Adjoint Newton–Krylov Algorithm for Aerodynamic Optimization," *AIAA Journal*, Vol. 46, No. 7, July 2008, pp. 1695–1704. doi:10.2514/1.33836

[21] Jakobsson, S., and Amoignon, O., "Mesh Deformation Using Radial Basis Functions for Gradient-Based Aerodynamic Shape Optimization," *Computers and Fluids*, Vol. 36, No. 6, 2007, pp. 1119–1136. doi:10.1016/j.compfluid.2006.11.002

[22] Allen, C. B., and Rendall, T. C. S., "Unified Approach to CFD-CSD Interpolation and Mesh Motion Using Radial Basis Functions," 25th AIAA Applied Aerodynamics Conference, AIAA Paper 2007-3804, Miami, FL, June 2007.

[23] Morris, A. M., Allen, C. B., and Rendall, T. C. S., "Domain-Element Method for Aerodynamic Shape Optimization Applied to a Modern Transport Wing," *AIAA Journal*, Vol. 47, No. 7, July 2009, pp. 1647–1659. doi:10.2514/1.39382

[24] Sederberg, T. W., and Parry, S. R., "Free-Form Deformation of Solid Geometric Models," *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86)*, Association for Computing Machinery, New York, 1986, pp. 151–160.

[25] Hicken, J. E., and Zingg, D. W., "A Parallel Newton-Krylov Solver for the Euler Equations Discretized Using Simultaneous Approximation

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033

Terms," *AIAA Journal*, Vol. 46, No. 11, Nov. 2008, pp. 2773–2786. doi:10.2514/1.34810

[26] Gage, P. J., Kroo, I. M., and Sobieski, I. P., "Variable-Complexity Genetic Algorithm for Topological Design," *AIAA Journal*, Vol. 33, No. 11, Nov. 1995, pp. 2212–2217. doi:10.2514/3.12969

[27] Vicini, A., and Quagliarella, D., "Airfoil and Wing Design Through Hybrid Optimization Strategies," *AIAA Journal*, Vol. 37, No. 5, May 1999, pp. 634–641. doi:10.2514/2.764

[28] Pironneau, O., "On Optimum Design in Fluid Mechanics," *Journal of Fluid Mechanics*, Vol. 64, No. 1, 1974, pp. 97–110. doi:10.1017/S0022112074002023

[29] Jameson, A., "Aerodynamic Design via Control Theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233–260. doi:10.1007/BF01061285

[30] Baysal, O., and Eleshaky, M. E., "Aerodynamic Sensitivity Analysis Methods for the Compressible Euler Equations," *Journal of Fluids Engineering*, Vol. 113, No. 4, 1991, pp. 681–688. doi:10.1115/1.2926534

[31] Frank, P. D., and Shubin, G. R., "A Comparison of Optimization-Based Approaches for a Model Computational Aerodynamics Design Problem," *Journal of Computational Physics*, Vol. 98, No. 1, Jan. 1992, pp. 74–89. doi:10.1016/0021-9991(92)90174-W

[32] Burgreen, G. W., and Baysal, O., "Three-Dimensional Aerodynamic Shape Optimization Using Discrete Sensitivity Analysis," *AIAA Journal*, Vol. 34, No. 9, Sept. 1996, pp. 1761–1770. doi:10.2514/3.13305

[33] Nielsen, E. J., and Anderson, W. K., "Aerodynamic Design Optimization on Unstructured Meshes Using the Navier-Stokes Equations," *AIAA Journal*, Vol. 37, No. 11, Nov. 1999, pp. 1411–1419. doi:10.2514/2.640

[34] Anderson, W. K., and Bonhaus, D. L., "Airfoil Design on Unstructured Grids for Turbulent Flows," *AIAA Journal*, Vol. 37, No. 2, Feb. 1999, pp. 185–191. doi:10.2514/2.712

[35] Rumpfkeil, M. P., and Zingg, D. W., "The Optimal Control of Unsteady Flows with a Discrete Adjoint Method," *Optimization and Engineering*, 2008. doi:10.1007/s11081-008-9035-5

[36] Giles, M. B., Duta, M. C., Müller, J.-D., and Pierce, N. A., "Algorithm Developments for Discrete Adjoint Methods," *AIAA Journal*, Vol. 41, No. 2, Feb. 2003, pp. 198–205. doi:10.2514/2.1961

[37] Nemec, M., Aftosmis, M. J., Murman, S. M., and Pulliam, T. H., "Adjoint Formulation for an Embedded-Boundary Cartesian Method," 43rd AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2005-0877, Reno, NV, 2005; also NASA Advanced Supercomputing Div., TR NAS-05-008, 2005.

[38] 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, July 1986, pp. 856–869. doi:10.1137/0907058

[39] Nielsen, E. J., and Park, M. A., "Using an Adjoint Approach to Eliminate Mesh Sensitivities in Computational Design," 43rd AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2005-0491, Jan. 2005.

[40] 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:10.1137/S003614459631241X

[41] Anderson, W. K., Newman, J. C., Whitfield, D. L., and Nielsen, E. J., "Sensitivity Analysis for Navier-Stokes Equations on Unstructured Meshes Using Complex Variables," *AIAA Journal*, Vol. 39, No. 1, Jan. 2001, pp. 56–63. doi:10.2514/2.1270

[42] Martins, J. R. R. A., Sturdza, P., and Alonso, J. J., "The Complex-Step Derivative Approximation," *ACM Transactions on Mathematical Software*, Vol. 29, No. 3, Sept. 2003, pp. 245–262. doi:10.1145/838250.838251

[43] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006. doi:10.1137/S1052623499350013

[44] Farin, G. E., *Curves And Surfaces for Computed Aided Geometric Design: A Practical Guide*, 4th ed., Academic Press, London, 1997.

[45] Yu, T.-Y., and Soni, B. K., "Application of NURBS in Numerical Grid Generation," *Computer-Aided Design*, Vol. 27, No. 2, Feb. 1995, pp. 147–157. doi:10.1016/0010-4485(95)92154-K

[46] Yu, T.-Y., and Soni, B. K., "NURBS in Structured Grid Generation," *Handbook of Grid Generation*, edited by J. F. Thompson, B. K. Soni, and N. P. Weatherill, CRC Press, Boca Raton, FL, 1999, Chap. 30.

[47] Hayes, J. G., *Numerical Analysis*,Springer, Berlin, 1982, pp. 140–156.

[48] Schumaker, L. L., *Spline Functions*, Wiley, New York, 1981.

[49] Walz, C., "A Unified Approach to B-Spline Recursions and Knot Insertion, with Application to New Recursion Formulas," *Advances in Computational Mathematics*, Vol. 3, No. 1, 1995, pp. 89–100. doi:10.1007/BF02431997

[50] Piegl, L. A., and Tiller, W., "Computing the Derivative of NURBS with Respect to a Knot," *Computer Aided Geometric Design*, Vol. 15, No. 9, 1998, pp. 925–934. doi:10.1016/S0167-8396(98)00028-4

[51] Hoschek, J., "Intrinsic Parametrization for Approximation," *Computer Aided Geometric Design*, Vol. 5, No. 1, 1988, pp. 27–31. doi:10.1016/0167-8396(88)90017-9

[52] Truong, A. H., Hicken, J. E., and Zingg, D. W., "A Pseudo-Nonlinear Elasticity Mesh Movement Algorithm Applied to Mesh Generation: A Hands-Off Approach," *15 Annual Conference of the CFD Society of Canada* [CD-ROM], CFD Society of Canada, May 2007.

[53] Meijerink, J. A., and van der Vorst, H. A., "An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix," *Mathematics of Computation*, Vol. 31, No. 137, Jan. 1977, pp. 148–162. doi:10.2307/2005786

[54] Bar-Yoseph, P. Z., Mereu, S., Chippada, S., and Kalro, V. J., "Automatic Monitoring of Element Shape Quality in 2-D and 3-D Computational Mesh Dynamics," *Computational Mechanics*, Vol. 27, No. 5, 2001, pp. 378–395. doi:10.1007/s004660100250

[55] Schmitt, V., and Charpin, F., "Pressure Distributions on the ONERA M6 Wing at Transonic Mach Numbers," Office National d'Etudes et Recherches Aerospatiales, Rept. 92320, Chatillon, France, 1979.

[56] Strand, B., "Summation by Parts for Finite Difference Approximations for d/dx," *Journal of Computational Physics*, Vol. 110, No. 1, 1994, pp. 47–67. doi:10.1006/jcph.1994.1005

[57] Carpenter, M. H., Gottlieb, D., and Abarbanel, S., "Time-Stable Boundary Conditions for finite Difference Schemes Solving Hyperbolic Systems: Methodology and Application to High-Order Compact Schemes," *Journal of Computational Physics*, Vol. 111, No. 2, 1994, pp. 220–236. doi:10.1006/jcph.1994.1057

[58] Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," 14th Fluid and Plasma Dynamics Conference, Palo Alto, CA, AIAA Paper 81-1259, 1981.

[59] Pulliam, T. H., "Efficient Solution Methods for the Navier-Stokes Equations," *Numerical Techniques for Viscous Flow Computation in Turbomachinery Bladings*, VKI Lecture Series, von Kármán Inst. for Fluid Dynamics, Rhode-Saint-Genèse, Belgium, Jan. 1986.

[60] Swanson, R. C., and Turkel, E., "On Central-Difference and Upwind Schemes," *Journal of Computational Physics*, Vol. 101, No. 2, 1992, pp. 292–306. doi:10.1016/0021-9991(92)90007-L

[61] Saad, Y., "A Flexible Inner-Outer Preconditioned GMRES Algorithm," *SIAM Journal on Scientific and Statistical Computing*, Vol. 14, No. 2, 1993, pp. 461–469. doi:10.1137/0914028

[62] Cai, X.-C., Gropp, W. D., Keyes, D. E., and Tidriri, M. D., "Newton–Krylov-Schwarz methods in CFD," *International Workshop on Numerical Methods for the Navier-Stokes Equations*, Notes in Numerical Fluid Mechanics, edited by F. Hebeker, and R. Rannacher, Vieweg Verlag, Braunschweig, Germany, 1993, pp. 17–30.

[63] Nielsen, E. J., Walters, R. W., Anderson, W. K., and Keyes, D. E., "Application of Newton-Krylov Methodology to a Three-Dimensional Unstructured Euler Code," 12th AIAA Computational Fluid Dynamics Conference, San Diego, CA, AIAA Paper 95-1733, 1995.

[64] Gropp, W. D., Kaushik, D. K., Keyes, D. E., and Smith, B. F., "High-Performance Parallel Implicit CFD," *Parallel Computing*, Vol. 27, No. 4, 2001, pp. 337–362. doi:10.1016/S0167-8191(00)00075-2

[65] Saad, Y., and Sosonkina, M., "Distributed Schur Complement Techniques for General Sparse Linear Systems," *SIAM Journal on Scientific*

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033

Computing, Vol. 21, No. 4, 1999, pp. 1337–1357.
doi:10.1137/S1064827597328996

[66] Pueyo, A., and Zingg, D. W., "Efficient Newton-Krylov Solver for Aerodynamic Computations," *AIAA Journal*, Vol. 36, No. 11, Nov. 1998, pp. 1991–1997.
doi:10.2514/2.326

[67] Nocedal, J., and Wright, S. J., *Numerical Optimization*, Springer–Verlag, Berlin, 1999.

[68] Barth, T. J., and Linton, S. W., "An Unstructured Mesh Newton Solver for Compressible Fluid Flow and Its Parallel Implementation," 33rd AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 95-0221, Reno, NV, 1995.

[69] Griewank, A., *Evaluating Derivatives*, Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[70] Nielsen, E. J., and Kleb, B., "Efficient Construction of Discrete Adjoint Operators on Unstructured Grids by Using Complex Variables," 43rd AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2005-0324, Reno, NV, 2005.

[71] Mader, C. A., Martins, J. R. R. A., Alonso, J. J., and van der Weide, E., "ADjoint: An Approach for Rapid Development of Discrete Adjoint Solvers," *AIAA Journal*, Vol. 46, No. 4, April 2008, pp. 863–873.
doi:10.2514/1.29123

[72] Hicken, J. E., "Efficient Algorithms for Future Aircraft Design: Contributions to Aerodynamic Shape Optimization," Ph.D. Thesis, Univ. of Toronto, Toronto, Ontario, Canada, 2009.

[73] de Sturler, E., "Truncation Strategies for Optimal Krylov Subspace Methods," *SIAM Journal on Numerical Analysis*, Vol. 36, No. 3, 1999, pp. 864–889.
doi:10.1137/S0036142997315950

[74] Hicken, J. E., and Zingg, D. W., "A Simplified and Flexible Variant of GCROT," *SIAM Journal on Scientific Computing* (to be published).

[75] Nocedal, J., and Wright, S. J., *Numerical Optimization*, 2nd ed., Springer–Verlag, Berlin, 2006.

[76] Billing, L. K., "On the Development of an Improved Lift-Constrained Aerodynamic Optimization Algorithm," M.S. Thesis, Univ. of Toronto, Toronto, Ontario, Canada, 2006.

[77] Zingg, D. W., and Billing, L., "Toward Practical Aerodynamic Design Through Numerical Optimization," 18th AIAA Computational Fluid Dynamics Conference, AIAA Paper 2007-3950, Miami, FL, June 2007.

[78] Buckley, H., and Zingg, D. W., "Airfoil Optimization Using Practical Aerodynamic Design Requirements," 19th AIAA Computational Fluid Dynamics Conference, AIAA Paper 2009-3516, San Antonio, TX, 2009.

[79] Kroo, I., "Drag Due to Lift: Concepts for Prediction and Reduction," *Annual Review of Fluid Mechanics*, Vol. 33, 2001, pp. 587–617.
doi:10.1146/annurev.fluid.33.1.587

Z. Wang
Associate Editor

Downloaded by UNIVERSITY OF OKLAHOMA on January 23, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.44033