AIAA JOURNAL
Vol. 44, No. 1, January 2006

Multigrid Solution of the Discrete Adjoint for Optimization Problems on Unstructured Meshes

Dimitri J. Mavriplis*
University of Wyoming, Laramie, Wyoming 82071

Techniques for the efficient formulation and solution of the discrete adjoint problem for the Reynolds-averaged Navier–Stokes equations and for the mesh deformation equations are developed for use on unstructured grids in two dimensions. An explicit two-pass approach is used to construct the second-order-accurate flow adjoint equations, and a defect-correction scheme is used to solve the flow adjoint equations, employing a line-implicit agglomeration multigrid strategy to drive the defect-correction scheme. A similar line-implicit multigrid approach is used to solve the mesh deformation equations, as well as the adjoint system of these equations. For the flow sensitivity and mesh motion equations, the multigrid solver is constructed to be duality preserving, delivering similar convergence rates for the primal and dual (adjoint) problems in both cases. These techniques are demonstrated for a viscous turbulent airfoil shape optimization problem.

Nomenclature

C_D	=	drag coefficient
C_L	=	lift coefficient
$\boldsymbol{D}$	=	vector of design variables
$[D]$	=	diagonal matrix elements
$[K]$	=	stifness matrix for mesh motion equations
k_{ij}	=	edge-based spring coefficient
L	=	objective function
$[O]$	=	off-diagonal matrix elements
$[P]$	=	preconditioning matrix
q	=	undivided Laplacian of flow variables w
$\boldsymbol{R}$	=	residual vector for flow equations
$\boldsymbol{w}$	=	vector of flow variables
$\boldsymbol{X}$	=	vector for grid point coordinates
$\boldsymbol{\delta x}$	=	vector of mesh point displacements
Λ	=	flow adjoint or costate variables
Λ_x	=	mesh motion adjoint or costate variables
λ	=	step size for steepest descent algorithm

Introduction

THE use of the adjoint equations is now well established for design-optimization problems in computational fluid dynamics for the Euler and Navier–Stokes equations (see Refs. 1–5). The advantage of adjoint formulations is that they enable the computation of sensitivity derivatives of a given objective function or output functional at a cost that is essentially independent of the number of design variables, requiring a single flow solution and a single adjoint solution, for any number of design variables. On the other hand, adjoint formulations become more expensive when multiple objective functions and constraints are involved. However, for problems involving large numbers of design variables, adjoint methods are most often the methods of choice.

In the continuous adjoint method approach, the original continuous governing equations are first linearized and then discretized, whereas in the discrete adjoint approach, these two steps are performed in reverse order. Because the continuous approach affords more flexibility at the discretization stage, formulations involving reduced memory and CPU overheads have generally been achieved with this approach.[2] On the other hand, the discrete approach reproduces the exact sensitivity derivatives of the original discretization of the governing equations, which provides a verifiable consistency check on the final gradients produced by the adjoint solution.[6] The discrete adjoint approach also benefits from a relative simplicity of implementation, requiring a straightforward (but tedious) linearization of the governing flow equations. In fact, if a linearization of the governing equations is already available in the form of a Jacobian matrix for a fully implicit scheme, the adjoint equations (including boundary conditions) are obtained by simply transposing the Jacobian matrix. However, for second-order finite volume discretizations, storage of the exact Jacobian is generally not practical because the discretization involves an extended stencil, which includes nearest neighbors, as well as neighbors of neighbors. Therefore, an efficient technique is required for constructing the discrete adjoint in such cases. In the current paper, we propose a simple two-pass construction for the discrete adjoint residual.

An efficient solution strategy is also required for converging the adjoint problem. Because the adjoint equations are closely related to the governing flow equations (containing the same eigenvalues as the linearized flow equations), techniques for converging the flow equations have often been used successfully to solve the adjoint equations.[2–4] More recently, an exact duality-preserving correspondence between iterative primal (governing equations) and dual- (adjoint)-equation solution schemes has been demonstrated.[7,8] When this approach is used, any iterative scheme applied to the linearized governing equations may be modified to produce an adjoint solution scheme that delivers similar convergence characteristics for the primal and dual problems. In this work, we develop a line-preconditioned agglomeration multigrid scheme for solving the adjoint problem, based on previous experience with this approach for the nonlinear flow equations.[9–11]

Shape optimization problems involve modification of the surface geometry, which requires deformation of a computational mesh, for boundary-conforming mesh approaches. The surface displacements generated by the design optimization procedure must be propagated smoothly into the domain to maintain good mesh cell qualities. The mesh deformation technique must be robust, to prevent negative mesh cell volumes under large deformations, and efficient solution techniques are required to rapidly compute the new mesh point positions. In this work, we use a linear spring analogy for constructing the governing equations of the mesh deformation and employ a line-preconditioned multigrid technique for efficiently solving these equations.

Finally, for shape optimization problems with deforming computational meshes, the objective function sensitivities depend on the

Presented as Paper 2005-0139 at the 43rd Aerospace Sciences Meeting, Reno, NV, 10–13 January 2005; received 21 January 2005; revision received 17 August 2005; accepted for publication 23 August 2005.

*Professor, Department of Mechanical Engineering. Associate Fellow AIAA.

mesh sensitivities, that is, the gradients of mesh deformation with respect to the design variables for each grid point, for each design variable. Recent work has shown how the effect of the mesh sensitivities on the final objective functional sensitivities can be obtained by solving the adjoint problem of the mesh deformation equations.[12] We make use of this approach in this work and develop an efficient line-preconditioned multigrid solver for the adjoint mesh-motion equations, using the same duality-preserving iterative solver construction developed for the flow equations.

Formulation of the Discrete Adjoint Problem

Consider an objective function $L\{\boldsymbol{w}^*(\boldsymbol{X}(\boldsymbol{D})), \boldsymbol{X}(\boldsymbol{D})\}$ to be minimized in an optimization problem, where $\boldsymbol{D}$ represents the design variables, X represents the grid point positions, and $\boldsymbol{w}^*$ denotes the (converged) flowfield variables that result from the solution of the discretized flow equations, given in residual form as $\boldsymbol{R}\{\boldsymbol{w}^*(\boldsymbol{X}(\boldsymbol{D})), \boldsymbol{X}(\boldsymbol{D})\} = 0$. Note that the design variables do not appear explicitly in these equations. Rather, the objective function and the flow residuals depend on the converged flow variables and grid point positions, which in turn depend on the shape of the surface geometry and, thus, the design variables. The gradient of the objective function with respect to the design variables is obtained by straightforward differentiation,

$$\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{D}} = \frac{\partial L}{\partial \boldsymbol{X}}\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}} + \frac{\partial L}{\partial \boldsymbol{w}}\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{X}}\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}} \tag{1}$$

where the asterisk superscript that denotes converged flow variables has been dropped for clarity. The variation in the flow variables with grid coordinates can be obtained by differentiating the statement $\boldsymbol{R}\{\boldsymbol{w}^*(\boldsymbol{X}(\boldsymbol{D})), \boldsymbol{X}(\boldsymbol{D})\} = 0$, which yields

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right]\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{X}} = -\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}} \tag{2}$$

On substituting this expression into Eq. (1), we obtain

$$\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{D}} = \left[\frac{\partial L}{\partial \boldsymbol{X}} - \frac{\partial L}{\partial \boldsymbol{w}}\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right]^{-1}\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right]\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}} \tag{3}$$

Thus, if the grid sensitivities are known, and the inverse of the Jacobian of the governing flow equations has been computed and stored, the sensitivity of the objective function with respect to each design variable can be obtained through a series of matrix–vector multiplications. Whereas the Jacobian matrix is generally sparse, its inverse is dense, and it is not feasible in general to compute and store $[\partial \boldsymbol{R}/\partial \boldsymbol{w}]^{-1}$. Thus, for each design variable $\boldsymbol{D}_i$, the product

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right]^{-1}\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}_i}\left(= -\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{D}_i}\right) \tag{4}$$

may be obtained by solving the system of equations given by

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right]\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{D}_i} = -\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{D}_i} \tag{5}$$

which is equivalent to solving a linearized flow equation problem for each individual design variable.

The adjoint approach circumvents the requirement of computing numerous linearized flow solutions for large numbers of design variables by first evaluating the product

$$\frac{\partial L}{\partial \boldsymbol{w}}\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right]^{-1}(=\Lambda^T) \tag{6}$$

and substituting this vector result into Eq. (3). Rearranging Eq. (6), we obtain the adjoint flow equations

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right]^T\Lambda = \left(\frac{\partial L}{\partial \boldsymbol{w}}\right)^T \tag{7}$$

The solution of the adjoint equations is similar in cost to the solution of a single linearized flow equation problem. Thus, the adjoint approach enables the calculation of all of the sensitivity derivatives at a cost of one flow solution and one adjoint solution problem, with additional matrix–vector multiplications, assuming the grid sensitivities are known.

For the discrete adjoint problem, construction of the adjoint matrix is based on the linearization of the discretized flow equations. If a fully implicit or (more precisely) a Newton scheme is used to solve the nonlinear flow equations as

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right]\Delta \boldsymbol{w} = -\boldsymbol{R}(\boldsymbol{w}) \tag{8}$$

then the adjoint equations may be obtained by simply transposing the Jacobian matrix $[\partial \boldsymbol{R}/\partial \boldsymbol{w}]$. In practice, storage of the exact Jacobian for a second-order-accurate discretization involves an extended stencil and is, therefore, not practical. However, second-order-accurate discretizations are most often constructed in two nearest-neighbor passes, suggesting a similar approach for the construction of the discrete adjoint. For example, the artificial dissipation approach achieves higher-order accuracy by replacing differences of neighboring flow variables $\boldsymbol{w}_k - \boldsymbol{w}_i$ by differences in undivided Laplacians that is,

$$q_i = \sum_{k=1}^{\text{neighbors}} \boldsymbol{w}_k - \boldsymbol{w}_i$$

Therefore, rewriting the Jacobian using the chain rule,

$$\left[\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}\boldsymbol{w}}\right] = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}} + \frac{\partial \boldsymbol{R}}{\partial q}\frac{\partial q}{\partial \boldsymbol{w}} \tag{9}$$

the discrete adjoint can be obtained as

$$\left[\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}\boldsymbol{w}}\right]^T = \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right]^T + \left[\frac{\partial q}{\partial \boldsymbol{w}}\right]^T\left[\frac{\partial \boldsymbol{R}}{\partial q}\right]^T \tag{10}$$

The first term represents contributions from convective terms and, in the case of the Navier–Stokes equations, physical dissipation terms (which operate on a nearest-neighbor stencil for triangular elements). The last term is equivalent to contributions from a first-order artificial dissipation (applied to the q variables), thus constituting a symmetric matrix [neglecting variations in the interface coefficient matrix, which is evaluated at the Roe state (see Ref. 13)], whereas the third term represents the linearization (transposed) of the undivided Laplacian, which is a trivial matrix in this case containing unity entries for each edge of the mesh. When it is assumed that this matrix need not be stored due to its simplicity, the evaluation of $[\mathrm{d}\boldsymbol{R}/\mathrm{d}\boldsymbol{w}]^T\Lambda$ makes use of existing (edge-based) flow-solver data structures and requires the storage of two matrices: $[\partial \boldsymbol{R}/\partial \boldsymbol{w}]^T$, which is equivalent to a nearest-neighbor stencil-based sparse matrix, and $[\partial \boldsymbol{R}/\partial q]^T$ of which only half the matrix need be stored, thus, resulting in 1.5 times the storage of a nearest-neighbor first-order Jacobian. For a MUSCL-type reconstruction scheme, the linearization can be cast in a similar form, where the residual $\boldsymbol{R}(\boldsymbol{w})$ now depends on the flow variables $\boldsymbol{w}$, as earlier, and on the computed gradients of these variables, which take the place of the q variables. The discrete adjoint equations are obtained in a similar fashion, where $\partial q/\partial \boldsymbol{w}$ represents the linearization of the gradient formulation, which is again a matrix that need not be stored because the matrix entries are grid metrics that are already stored for the gradient computations in the residual evaluation.[14]

Duality-Preserving Iterative Strategy

The primal approach [cf. Eq. (5)] and the dual approach [cf. Eq. (7)] represent two techniques for evaluating the second term

in Eq. (1). In the case of a single design variable and a single objective function, the two approaches yield the duality relation

$$\frac{\partial L}{\partial \boldsymbol{w}}\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{D}} = -\Lambda^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{D}} \left(= \frac{\partial L}{\partial \boldsymbol{w}}\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{X}}\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}} \right) \tag{11}$$

which must hold for the finally converged values of $\partial \boldsymbol{w}/\partial \boldsymbol{D}$ and Λ. However, iterative solution strategies for Eqs. (5) and (7) that preserve relation (11) at each iteration for partially converged values of $\partial \boldsymbol{w}/\partial \boldsymbol{D}$ and Λ have been demonstrated.[7,8] If Eq. (5) is solved as

$$[P]\left(\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{D}}^{n+1} - \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{D}}^{n} \right) = -\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{D}} - \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}} \right] \left(\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{D}} \right)^n \tag{12}$$

where $[P]$ is a preconditioning matrix to be inverted, the corresponding duality-preserving iteration strategy for Eq. (7) is given by

$$[P]^T(\Lambda^{n+1} - \Lambda^n) = \frac{\partial L}{\partial \boldsymbol{w}} - \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}} \right]^T \Lambda^n \tag{13}$$

The right-hand sides of Eqs. (12) and (13) represent the linear flow residual and adjoint residual, respectively. Because these contain a matrix–vector product involving the full second-order-accurate Jacobian $[\partial \boldsymbol{R}/\partial \boldsymbol{w}]$, they must be evaluated using the two-pass construction given by Eqs. (9) and (10) for the primal and dual problems, respectively.

The preconditioning matrix $[P]$ should closely approximate the full Jacobian $[\partial \boldsymbol{R}/\partial \boldsymbol{w}]$ while being simple to invert. A common strategy is to take $[P]$ as the first-order Jacobian of the flow residual. This results in a defect-correction approach, which is analogous to the solution strategy proposed for nonlinear problems in Ref. 11. In this case, the nonlinear flow equations are solved using the approximate Newton scheme,

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}} \right]_{\text{first-order}} \Delta \boldsymbol{w} = -\boldsymbol{R}(\boldsymbol{w}) \tag{14}$$

where $\boldsymbol{R}(w)$ is the residual of the nonlinear flow equations. The use of a full second-order Jacobian on the left-hand side of this equation would correspond to an exact Newton scheme, which would converge quadratically. Similarly, for the linearized equations (12) and (13), this would produce the exact solution in a single step (matrix inversion). In the defect-correction approach, the simplified form of the left-hand-side matrix results in a more tractable inversion problem at each step, but also results in slower overall convergence. Furthermore, because of the approximate nature of the left-hand-side matrix in the defect-correction approach, a more computationally efficient approximate inversion of this matrix is generally sufficient to achieve the same overall convergence rate of the outer iteration procedure. For example, for the nonlinear flow equation solver described in Ref. 11, a small number of linear multigrid cycles (2–5) was found to yield the optimal convergence efficiency. Similarly, for Eqs. (12) and (13), a small number of multigrid cycles can be used to compute the approximate inverse matrix–vector product,

$$(\Lambda^{n+1} - \Lambda^n) = [\tilde{P}]^{-T} \left[\frac{\partial L}{\partial \boldsymbol{w}} - \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}} \right]^T \Lambda^n \right] \tag{15}$$

for each outer iteration n, where $[\tilde{P}]^{-T}$ is the approximate inverse of $[P]^T$. In this work, we use a line-preconditioned Jacobi or Gauss–Seidel driven agglomeration multigrid method at each iteration n of the defect-correction scheme. On each grid level of the multigrid algorithm, an iterative smoothing strategy may be constructed by decomposing the $[P]$ matrix into implicitly treated components denoted as $[D]$ and explicitly treated components $[O]$ and writing the iteration as

$$[D]^T(\Lambda^{n+1} - \Lambda^n)^{k+1} = \frac{\partial L}{\partial \boldsymbol{w}} - \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}} \right]^T \Lambda^n - [O]^T(\Lambda^{n+1} - \Lambda^n)^k \tag{16}$$

in the case of the adjoint problem, where k is the iterative smoothing counter. For inviscid flows on isotropic triangular meshes, a point-implicit solution strategy is adopted, where the implicit components $[D]$ correspond to the 4×4 block diagonal submatrices of $[P]$ for two-dimensional cases, and the explicit components $[O]$ correspond to the off-diagonal blocks. For viscous turbulent flows, the linearization must include the full coupling between the flow equations and the turbulence model. The block submatrices thus become 5×5 blocks, using a single-equation turbulence model in two dimensions. Furthermore, a line-implicit scheme is employed to relieve the stiffness associated with high mesh stretching in boundary-layer and wake regions.[9,15] This is achieved by constructing lines in the mesh by grouping together sets of edges in the mesh, using a graph algorithm, producing line sets as shown in Fig. 1. Each line is solved implicitly by taking the $[D]$ components as the union of all of the $[P]$ matrix entries, which correspond to points and edges within the lines, whereas the $[O]$ components are composed of the remaining entries. In this approach, the constructed lines have variable length and reduce to a single grid point in isotropic regions of the mesh. Therefore, all grid points are contained in the set of lines. This solution scheme corresponds to a line-Jacobi or line-Gauss–Seidel strategy in highly stretched regions of the mesh, which reverts to a point Jacobi or Gauss–Seidel scheme in isotropic regions of the mesh. In both cases, the Gauss–Seidel scheme provides faster convergence due to the ordered sweep across the mesh that employs latest available information.

To conserve memory space, the individual blocks of the $[P]^T$ matrix need not be stored separately. Rather, they can be quickly reconstructed from the matrices used to evaluate the full second-order

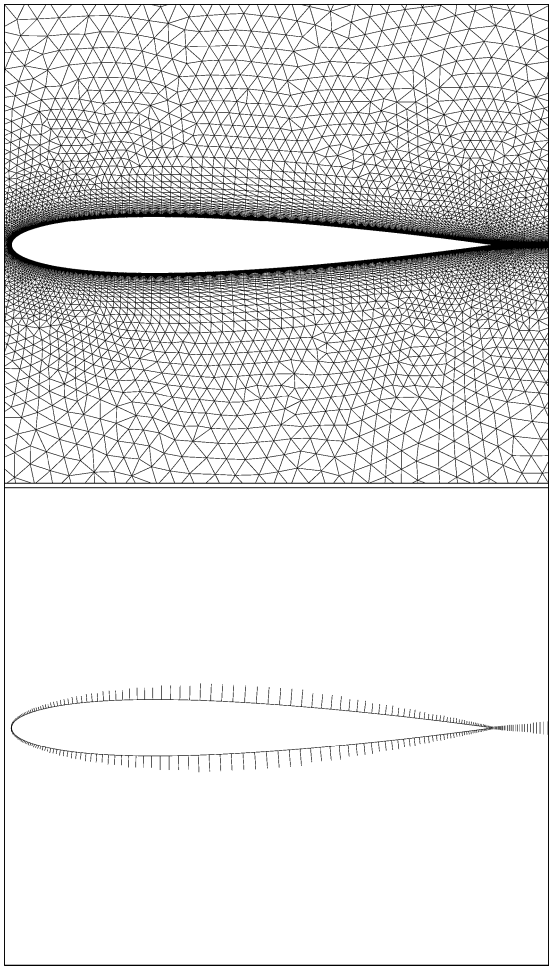


Fig. 1 Unstructured mesh for viscous flow over NACA 0012 airfoil and set of lines constructed by graph algorithm for implicit line solver.

adjoint residual as

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right]^T_{\text{first-order}} = \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right]^T + \left[\frac{\partial \boldsymbol{R}}{\partial q}\right]^T \tag{17}$$

where the matrices on the right-hand side of Eq. (17) correspond to those defined in Eq. (10).

In practice, a separate copy of the $[D]^T$ block diagonals are factorized and stored in lower–upper (LU) form, but the off-diagonal blocks $[O]^T$ are reconstructed as per Eq. (17) at each iteration.

These iterative schemes are then used as the driver for the linear agglomeration multigrid strategy to accelerate the solution of the adjoint problem. The use of agglomeration multigrid for solving the Euler and Navier–Stokes equations on unstructured meshes either directly as a nonlinear solver, or as a linear solver, is now well established.[11,16–18] A duality-preserving agglomeration multigrid algorithm has been developed for solving the discrete adjoint equations on unstructured meshes. To preserve exact duality, the order of operations in the multigrid cycle must be reversed when applied to the adjoint equations.[8] For example, a multigrid cycle that employs one presmoothing and no postsmoothing iterations must be replaced by a cycle that uses no presmoothing and one postsmoothing iteration for the adjoint problem. Furthermore, the restriction and prolongation operators must be exact transposes of one another.

Adjoint Test Problems

The convergence of the linearized sensitivity equations [cf. Eq. (5)] and the adjoint equations [cf. Eq. (7)] for the inviscid flow over a NACA 0012 airfoil at a Mach number of 0.6 and an incidence of 1.25 deg is shown in Fig. 2. The objective function in this case is the lift coefficient, and the design variable is the vertical displacement of a surface grid point at 63% chord on the lower surface of the airfoil. The unstructured mesh for this case contains 7884 vertices and is composed of approximately isotropic triangular elements. Five multigrid levels are used, with four smoothing iterations on each grid level (two presmoothing and two postsmoothing), using either a Jacobi or Gauss–Seidel iteration for the smoother. Four multigrid W cycles are used at each iteration of the defect-correction scheme to invert approximately the first-order Jacobian, before reevaluating the right-hand sides of Eqs. (12) and (13) using the two-pass approach. Rapid convergence to machine accuracy is observed, and the Gauss–Seidel smoother is seen to deliver better overall multigrid convergence than the Jacobi smoother. In both cases, primal and dual problems converge at similar rates, which mirror the convergence of the agglomeration multigrid algorithm for the nonlinear solution of the flow equations. However, even in the presence of the duality-preserving solution scheme, there is no reason to expect the residual histories of the primal and dual problem to overlap exactly because these two residuals depend on different quantities. For example, different objective functions produce different dual residuals, but have no effect on the primal residual. On the other hand, the sensitivity computed by both methods is identical at each iteration when strict duality is preserved, as shown in Fig. 3. In this case, the difference between the sensitivity derivative calculated by the primal and dual approaches is plotted as a function of the iteration number, for the duality-preserving algorithm, and for a nonduality-preserving algorithm, where three smoothing passes were performed in the multigrid coarsening phase, and one pass in the refinement phase, for both primal and dual solvers. In this latter case, the values computed by the two formulations initially differ by approximately 10^{-2}, and the difference decreases to machine zero as the respective problems converge, whereas the difference remains at machine zero for the duration of the iteration procedure for the duality-preserving schemes.

In Fig. 4, the convergence of the sensitivity derivative calculated using the adjoint approach is shown as a function of the iteration number for the duality-preserving and nonpreserving schemes, where these are also compared with the value obtained by finite difference. The finite difference result was calculated by displacing the surface grid point in the vertical direction using a step size of 10^{-5} (at $x = 63\%$ chord on the lower surface), and recomputing the flow solution, corresponding to a forward (finite) difference. The flow equations were converged to machine accuracy for both cases required in the finite difference evaluation. The chosen step size, although perhaps not optimal, was found by trial and error to represent a region where the finite difference result was insensitive to the

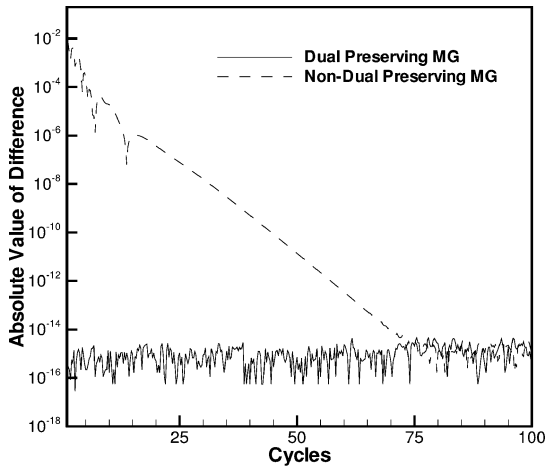


Fig. 3 Convergence history of difference between sensitivity derivatives calculated with primal and dual equations for duality-preserving iterative scheme and nonpreserving scheme.

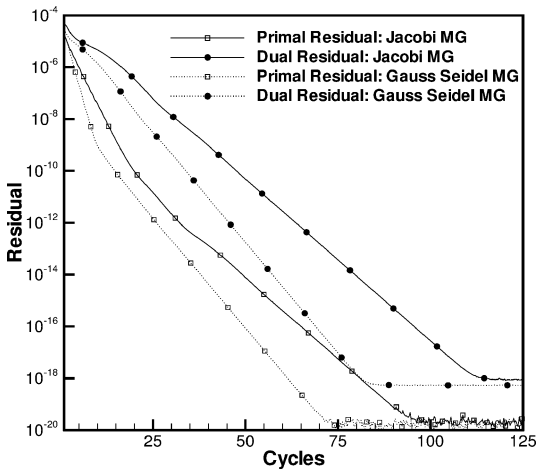


Fig. 2 Convergence of primal (linearized flow sensitivity) equations and dual (adjoint) equations using Jacobi and Gauss–Seidel agglomeration multigrid-driven defect-correction scheme.

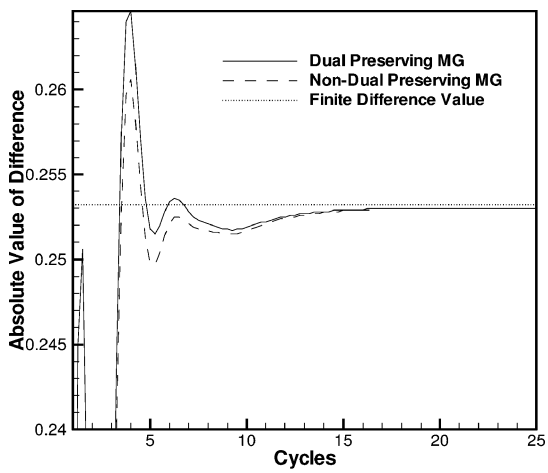


Fig. 4 Convergence history of sensitivity derivatives calculated with duality-preserving iterative scheme and nonpreserving scheme compared with finite difference value.

step size. Both schemes result in rapid convergence of the sensitivity derivative, underlining the fact that exact duality preservation is not required for efficient solution of the adjoint equations. The final value of the converged sensitivity derivative is 0.2530, which is close to the value of 0.2534 computed using a finite difference approach.

Figure 1 shows the mesh employed for the solution of viscous turbulent flow over a NACA 0012 airfoil at a Mach number of 0.729, an incidence of 2.31 deg, and a Reynolds number of 6.5×10^6. The mesh contains a total of 18,466 points, with a grid spacing at the wall in the normal direction of 10^{-6} chords, thus producing highly stretched mesh cells in these regions. The fully coupled flow and turbulence equations are solved using the approximate Newton method [cf. Eq. (14)], using the Spalart–Allmaras turbulence model,[19] and employing a linear agglomeration multigrid method with line preconditioning to solve the linear problem arising at each nonlinear solution step. Four multigrid W cycles (with four line Gauss–Seidel smoothing iterations on each grid level) are employed at each nonlinear step, achieving simultaneous convergence of the flow and turbulence equations over 100 nonlinear iterations, as shown in Fig. 5. In Fig. 6, the convergence of the defect-correction scheme for the linearized flow sensitivity equations (at the converged state) is shown as a function of the number of defect-correction iterations. The defect-correction iterations are analogous to the flow solution iterations, consisting of four W multigrid cycles with four-block-line Gauss–Seidel sweeps on each level, where the flow and turbulence equations are treated in a fully coupled manner. In Fig. 7, the equivalent scheme is also used to solve the adjoint problem for the coupled flow–turbulence equations, and the convergence of these quantities is plotted as a function of the number of defect-correction cycles. In this case, the objective function corresponds to the lift coefficient, and the design variable corresponds to the vertical displacement of a surface grid point at the midchord location on the lower surface of the airfoil. The absolute magnitudes of the primal- and dual-equation residuals depend on the particular right-hand-side vectors for a given problem and are, thus, not important in the current comparison, that is, these would change for a different choice of design variable or objective function. However, the asymptotic convergence rates of both the primal- and dual-equation solution strategies are seen to be almost identical, thus validating the duality-preserving approach of this solution strategy. In both cases, the primal and dual flow and turbulence residuals are reduced by 10 orders of magnitude in 100 cycles. Whereas each defect-correction cycle involves four multigrid cycles, note that these are relatively inexpensive cycles because the full second-order Jacobian is only evaluated once for each defect-correction cycle.

Figure 8 provides a comparison of various smoothers in the multigrid-driven defect-correction scheme used to solve the adjoint problem for the viscous airfoil case. In all cases, four W multigrid cycles are used at each defect-correction step, with four smoothing passes on each grid level. Whereas the line Jacobi solver delivers

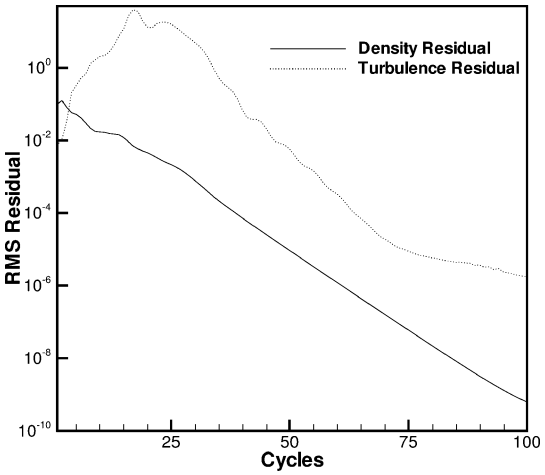


Fig. 5 Convergence of nonlinear flow and turbulence equations for viscous turbulent NACA 0012 airfoil as function of number of nonlinear cycles, using four line Gauss–Seidel driven agglomeration multigrid iterations for each nonlinear cycle.

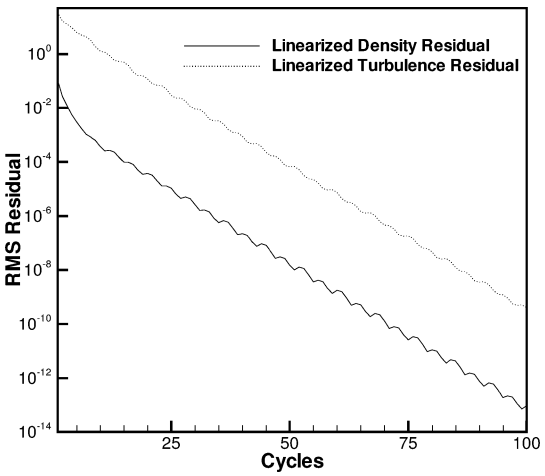


Fig. 6 Convergence of primal (linearized flow sensitivity) equations for viscous turbulent NACA 0012 airfoil as function of number of defect-correction cycles, using four line Gauss–Seidel driven agglomeration multigrid iterations for each defect-correction cycle.

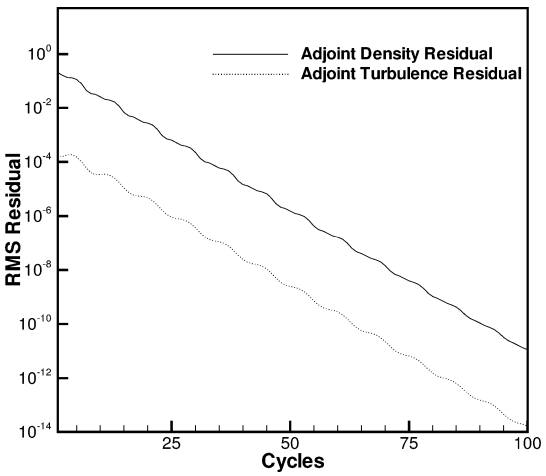


Fig. 7 Convergence of dual problem for viscous turbulent NACA 0012 airfoil as function of number of defect-correction cycles, using four line Gauss–Seidel driven agglomeration multigrid iterations for each defect-correction cycle.

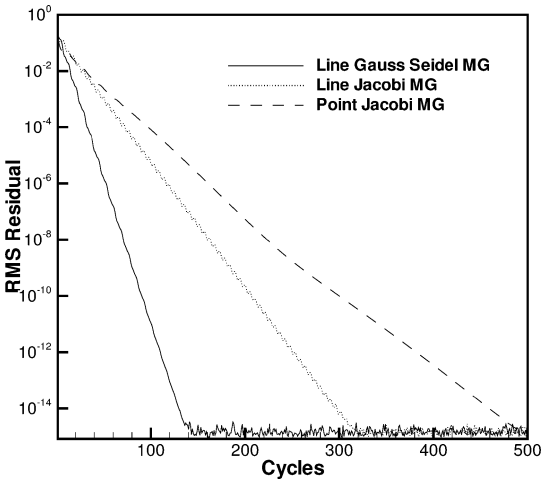
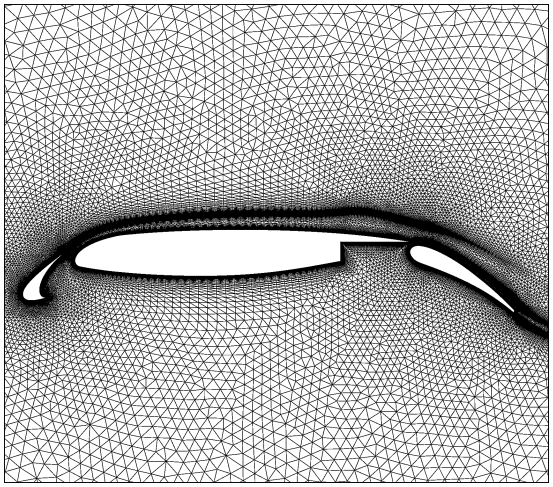


Fig. 8 Comparison of various smoothers for solution of adjoint problem using multigrid driven defect-correction scheme for viscous airfoil case.



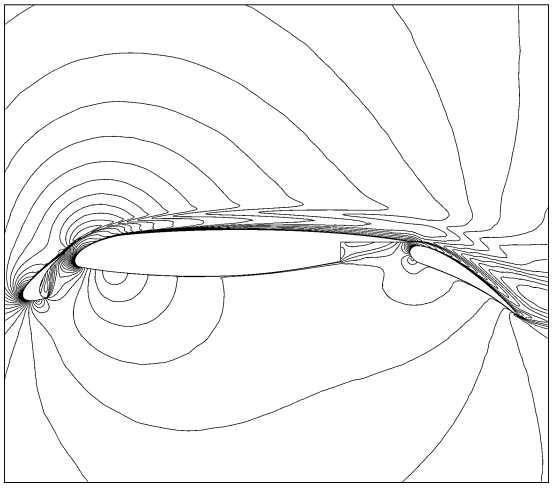


Fig. 9 Unstructured grid and computed Mach contours for flow over three-element airfoil configuration; Mach = 0.2, incidence = 16 deg, and $Re = 5 \times 10^6$, with, 61,104 grid points.

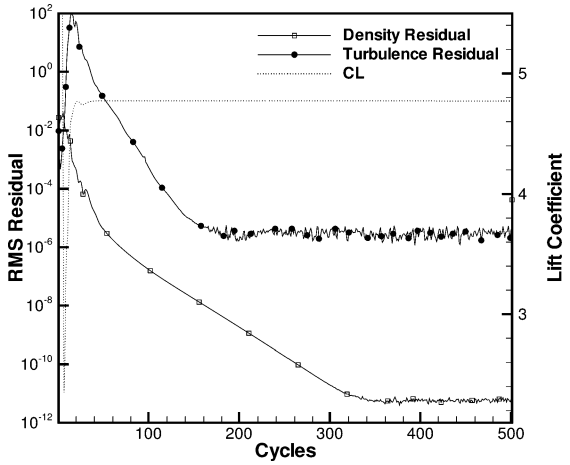


Fig. 10 Convergence of nonlinear flow solution in terms of nonlinear updates, using linear agglomeration multigrid algorithm, with four multigrid W cycles per nonlinear update for three-element airfoil case.

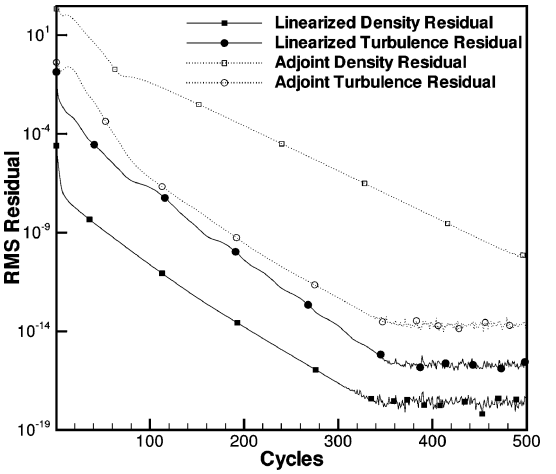


Fig. 11 Convergence of linearized sensitivity (primal) and adjoint (dual) equations using defect-correction scheme with four multigrid W cycles per defect-correction update for three-element airfoil case.

almost twice the convergence of the point Jacobi solver, the line Gauss–Seidel approach delivers another factor of two for this case, enabling convergence to machine zero just over 100 defect-correction cycles. Although Fig. 8 shows only the convergence of the adjoint density residual for clarity, the adjoint turbulence residual (and the primal equation residuals) behave in a similar fashion.

Figure 9 shows an unstructured grid about a three-element airfoil high-lift configuration and the computed Mach contours on this grid for a freestream Mach number of 0.2, an incidence of 16 deg, and a Reynolds number of 5×10^6. This grid contains a total of 61,104 vertices, with a spacing at the airfoil surfaces of 10^{-6} chords. The solution was computed using the linear multigrid method described in Ref. 11 with five multigrid levels, four line Gauss-Seidel sweeps on each level, and four multigrid W cycles between each nonlinear update. The convergence, plotted in terms of nonlinear cycles, is shown in Fig. 10. The overall convergence is slower for this complex configuration than for the simple airfoil case discussed earlier, as had been previously shown in Ref. 11. However, the flow residuals are converged to machine accuracy in approximately 300 cycles, whereas acceptable lift coefficient values are obtained in less than 100 cycles. The convergence of the linearized sensitivity equations (primal) and the adjoint (dual) equations, using the analogous defect-correction scheme (four line Gauss–Seidel sweeps, four multigrid cycles per defect-correction update) is shown in Fig. 11. The primal and dual problems are seen to converge at similar rates and mirror the convergence rate of the nonlinear flow problem shown in Fig. 10. For this case, the objective function corresponds to the lift coefficient, and the design variable was taken as the vertical displacement of a surface grid point at the midchord location on the lower surface of the main airfoil.

Mesh Deformation Equations

Robust and efficient mesh deformation techniques are required for shape optimization problems to maintain a suitable computational boundary-conforming mesh as the geometry is modified throughout the optimization process. In concurrent work, we have investigated various strategies for mesh deformation, including spring analogies, and linear elastic analogies, as well as efficient multigrid strategies, for solving these mesh deformation equations.[20] In the present work, a linear spring analogy mesh deformation technique is adopted for simplicity, although other approaches are currently under development.[3,20–22] In this approach, each mesh edge is modeled as a spring with an elastic constant inversely proportional to the edge length squared. Given a set of boundary displacements, the displacements of the interior grid points must obey the force balance equations, which result in the following system of equations for mesh point displacements:

$$\delta x_i = \frac{\sum_j^{\text{neighbors}} k_{ij} \delta x_j}{\sum_j^{\text{neighbors}} k_{ij}}, \qquad \delta y_i = \frac{\sum_j^{\text{neighbors}} k_{ij} \delta y_j}{\sum_j^{\text{neighbors}} k_{ij}}$$

$$k_{ij} = \left[(x_j - x_i)^2 + (y_j - y_i)^2 \right]^{-\frac{1}{2}} \tag{18}$$

where the summations are over all edges ij, which link a neighboring grid point j to the current point i. Note that these equations are linear, in the sense that the spring constants or edge lengths are based on the initial mesh configuration, and are not updated as the mesh is deformed. The governing equations for mesh deformation may, thus, be written as

$$[K]\delta x = \delta x_{\text{surface}} \tag{19}$$

where the vector δx represents the x and y displacements of all grid points and $\delta x_{\text{surface}}$ represents the prescribed surface grid point displacements, generated by the optimization procedure. For the linear spring analogy approach, the resulting mesh motion equations correspond to a (scaled) Poisson equation for the displacement in each coordinate direction.

For fine meshes, with large degrees of stretching, simple approaches for solving the mesh motion equations can become very time consuming. In this work, the linear agglomeration multigrid method is used to solve the mesh motion equations. For inviscid isotropic meshes, using point Jacobi or Gauss–Seidel smoothers on each grid level results in an efficient solution strategy. However, for viscous flow meshes with large degrees of stretching in the boundary-layer and wake regions, line solvers are required to efficiently displace the boundary-layer portions of the grid. This is shown in Fig. 12, for a highly stretched mesh around the leading edge of an airfoil, which undergoes an outward normal surface displacement. The bold line indicates the location of the displaced airfoil surface. Figure 12a shows the mesh configuration after 10-point Jacobi iterations (without multigrid), where it is seen that the majority of the boundary-layer points have not been displaced toward the new airfoil boundary, but still lie close to their original position.

In Fig. 12b, after just a single-line Jacobi iteration, the entire boundary-layer portion of the grid has been displaced close to the new boundary position. In fact, a single-line Jacobi iteration often results in a valid boundary-layer mesh in regions with negligible surface curvature because the surface displacements are propagated instantaneously through the boundary-layer region by the line solver. For regions with appreciable surface curvature, the remaining invalid mesh cells outside the boundary-layer regions are rapidly smoothed out with several multigrid steps, as shown in Fig. 12. Point or line Gauss–Seidel smoothers are especially effective for driving multigrid solvers for the mesh motion equations because overrelaxation can be used to further accelerate convergence. Figure 13 shows the convergence of the mesh motion equations for the viscous flow grid of Fig. 1, using a simple Jacobi solver, a point Jacobi-driven multigrid approach, a line Jacobi multigrid approach, and an overrelaxed line Gauss–Seidel approach. This latter method reduces the residuals of the mesh motion equations to machine zero in 100 multigrid steps. At this rate, the solution of the mesh motion equations corresponds to a small fraction of the cost of a flow or adjoint solution.

Mesh Sensitivities

The complete evaluation of the objective function sensitivities requires the evaluation of the mesh sensitivities $\partial \boldsymbol{X}/\partial \boldsymbol{D}$, as per Eq. (3). For linear mesh deformation equations, and for design variables that only produce surface displacements, the equations for grid sensitivities are obtained by differentiating the mesh motion equations as

$$[K]\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}} = \left(\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right)_{\text{surface}} \tag{20}$$

which may be rewritten as

$$\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}} = [K]^{-1}\left(\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right)_{\text{surface}} \tag{21}$$

Because evaluating and storing the $[K]^{-1}$ matrix is generally not feasible, the evaluation of the grid sensitivities in the manner just shown requires the solution of a mesh motion problem for each design variable. This can be avoided using the mesh motion adjoint procedure developed by Nielsen and Park.[12] Substituting the expression for the grid sensitivities of Eq. (21) into Eq. (3), one obtains the following expression for the objective function sensitivities:

$$\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{D}} = \left[\frac{\partial L}{\partial \boldsymbol{X}} - \frac{\partial L}{\partial \boldsymbol{w}}\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right]^{-1}\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right][K]^{-1}\left(\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right)_{\text{surface}} \tag{22}$$

Setting the product of the first bracketed term with the K^{-1} matrix equal to the vector $\boldsymbol{\Lambda}_x^T$, we obtain the mesh adjoint equation

$$[K]^T \boldsymbol{\Lambda}_x = \left[\frac{\partial L}{\partial \boldsymbol{X}} - \frac{\partial L}{\partial \boldsymbol{w}}\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right]^{-1}\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right]^T \tag{23}$$

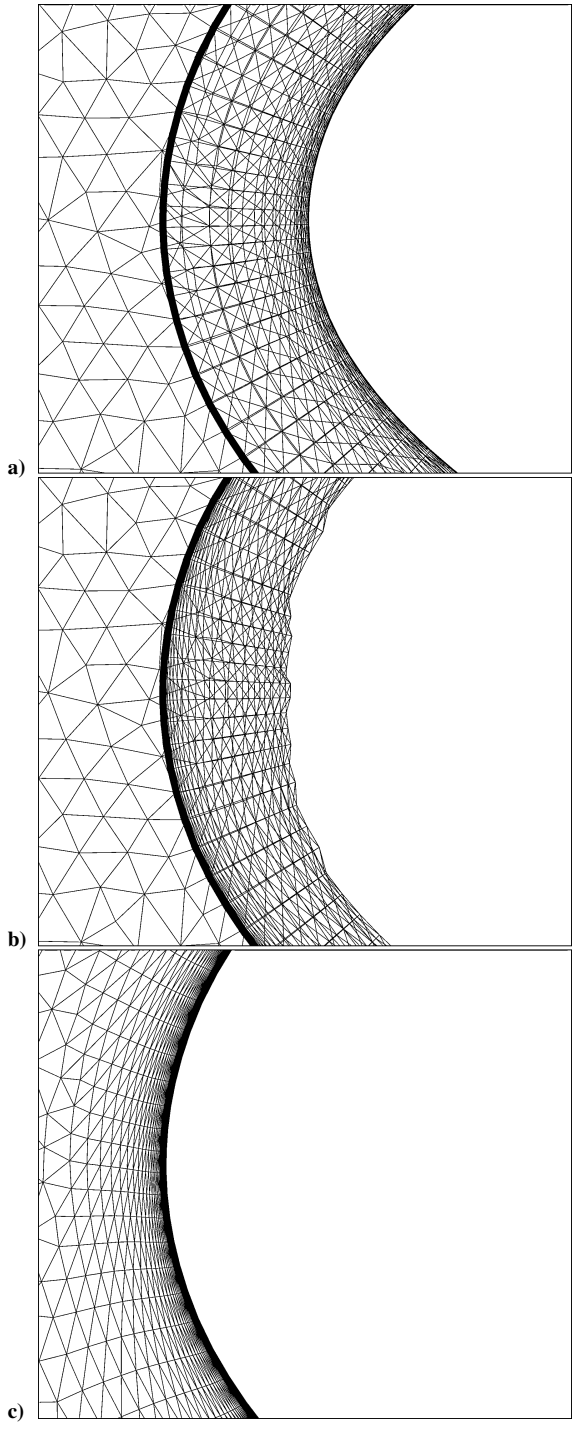


Fig. 12 Partially converged displaced viscous mesh:a) after 10-point Jacobi single-grid iterations, b) after single-line Jacobi iteration (without multigrid), and c) after 6-line Gauss–Seidel multigrid W cycles.

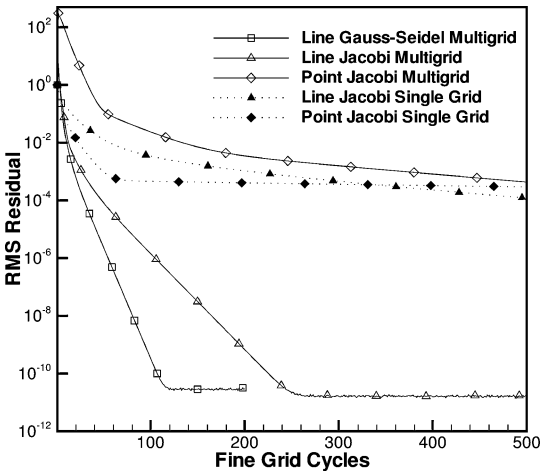


Fig. 13 Convergence of spring-analogy mesh displacement equations using point Jacobi, line Jacobi, and line Gauss–Seidel smoothers with agglomeration multigrid.

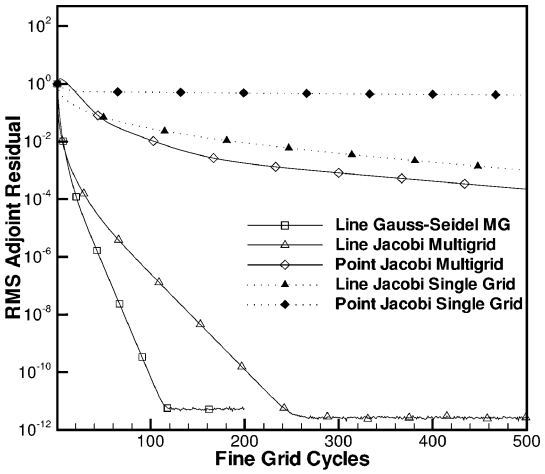


Fig. 14 Convergence of mesh displacement adjoint equations using point Jacobi, line Jacobi, and line Gauss–Seidel smoothers with agglomeration multigrid.

or, rewriting in terms of the flow adjoint solution,

$$[K]^T \boldsymbol{\Lambda}_x = \left[\frac{\partial L}{\partial \boldsymbol{X}}\right]^T - \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right]^T \boldsymbol{\Lambda} \tag{24}$$

The cost of solving these mesh motion adjoint equations is equivalent to the cost of solving a single mesh deformation problem. Once the $\boldsymbol{\Lambda}_x$ variables have been obtained, the complete set of objective function sensitivity derivatives is obtained by the simple vector product:

$$\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{D}} = \boldsymbol{\Lambda}_x \left(\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right)_{\text{surface}} \tag{25}$$

Because the spring analogy mesh deformation equations are linear and based on a nearest-neighbor stencil, the matrix K is easily stored and the matrix K^T is obtained by simply transposing the original mesh deformation stencil coefficient matrix. Following the procedure developed for the flow adjoint equations, we employ a duality-preserving point or line Jacobi or overrelaxed Gauss–Seidel scheme for solving the mesh adjoint equations. The convergence rates for the mesh adjoint equations on the grid of Fig. 1 are shown in Fig. 14 demonstrating similar convergence rates to the original mesh motion equations.

Note that the preceding approach (as well as any sensitivity-based approach involving mesh motion) requires the evaluation of $\partial \boldsymbol{R}/\partial \boldsymbol{X}$. This matrix is evaluated directly using handwritten code based on the residual evaluation routine and the mesh metric routines. However, because the mesh adjoint equations are linear, the product $[\partial \boldsymbol{R}/\partial \boldsymbol{X}]^T \boldsymbol{\Lambda}$ need only be evaluated once for each design step, making it feasible to recompute these terms at each step rather than storing the entire matrix.

Optimization Example

The methodology just described has been used to drive an optimization problem for a viscous turbulent airfoil drag reduction problem. The design problem consists of minimizing the drag coefficient while holding the lift coefficient at a constant value. The objective function used for this purpose is given as

$$L = (C_L - C_{L_{\text{target}}})^2 + 10(C_D)^2 \tag{26}$$

where $C_{L_{\text{target}}}$ is the constant lift coefficient value to be maintained and the weighting factor (10) is required to balance the different magnitudes of lift and drag coefficients. For simplicity, the objective function only considers the pressure drag component. The design variables consist of the normal displacements of each surface grid point on the airfoil. The optimization procedure follows the steepest descent approach described by Jameson[1] and Jameson et al.[2] Once the objective function sensitivities $\mathrm{d}\boldsymbol{L}/\mathrm{d}\boldsymbol{D}$ have been computed using the method described, an increment in the design variables is prescribed as

$$\boldsymbol{\delta D} = -\lambda \frac{\mathrm{d}\tilde{L}}{\mathrm{d}\boldsymbol{D}} \tag{27}$$

where λ is a small time step, chosen small enough to ensure convergence of the optimization procedure and $\mathrm{d}\tilde{L}/\mathrm{d}\boldsymbol{D}$ is the smoothed gradient $\mathrm{d}L/\mathrm{d}\boldsymbol{D}$ obtained using an implicit smoothing technique, which is necessary to ensure smooth design shapes, as described in Ref. 20. The current implementation is not optimal, in that λ is determined empirically, but is sufficient for demonstrating the utility of the adjoint solution techniques described herein.

The computational mesh for this case is shown in Fig. 1. This mesh contains a total of 18,466 grid points, with a normal grid spacing at the airfoil surface of 10^{-6} chords. The freestream Mach number is 0.729, the incidence is 2.31 deg, and the Reynolds number is 6.5×10^6. Four multigrid levels were used, with four line Gauss–Seidel smoothing passes on each mesh level, and the flow adjoint equations were run 50 defect-correction cycles at each design iteration, whereas the mesh motion and mesh adjoint equations were run 25 multigrid W cycles at each design cycle. The computed lift and drag coefficients vary from the values of 0.3790 and 0.0092 computed on the initial configuration to 0.3798 and 0.0049 on the redesigned configuration, after 25 design cycles. The initial and

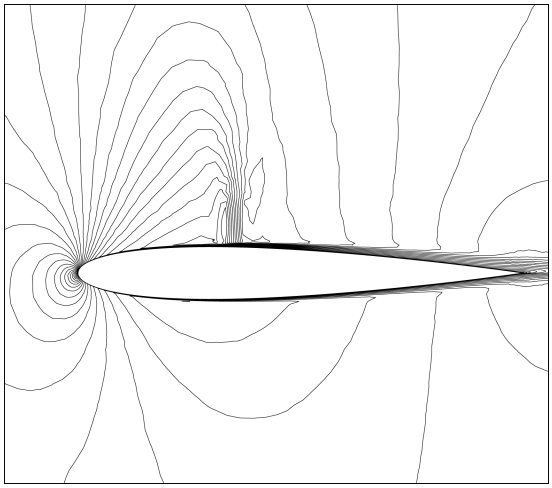


Fig. 15 Computed Mach contours for viscous turbulent flow over NACA 0012 airfoil; Mach = 0.729, incidence = 2.31 deg, and $Re = 6.5 \times 10^6$.

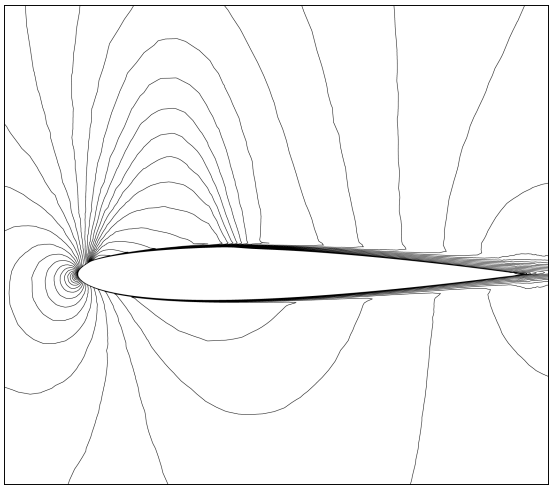


Fig. 16 Computed Mach contours for viscous turbulent flow over redesigned NACA 0012 airfoil; Mach = 0.729, incidence = 2.31 deg, and $Re = 6.5 \times 10^6$.

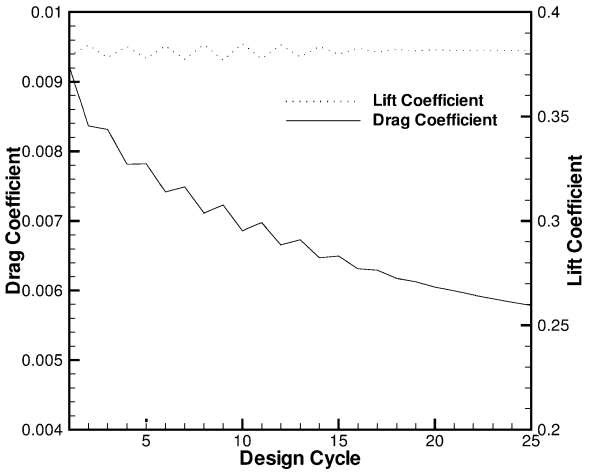


Fig. 17 Evolution of lift-and-drag coefficients as a function of the design cycle for drag minimization at fixed lift condition for viscous turbulent flow over NACA 0012 airfoil.

final flow solutions are plotted in Figs. 15 and 16, illustrating the reduction in the shock strength achieved by the design optimization process, whereas the optimization history of the drag and lift coefficients is shown in Fig. 17.

Conclusions

Efficient formulation and solution techniques for the discrete flow adjoint, mesh motion, and mesh adjoint problems have been demonstrated for two-dimensional unstructured mesh problems. These algorithms have the potential to produce efficient design optimization procedures based on the discrete adjoint principle. Future work will include the implementation of these techniques in the three-dimensional setting, as well as investigations into more sophisticated optimization procedures for minimizing the number of design cycles required to complete an optimization procedure.

Acknowledgments

This work was partly supported by NASA Grant NNL04AA63G from the NASA Langley Research Center and by the National Science Foundation (NSF) Wyoming Experimental Program to Stimulate Competitive Research, NSF Grant EPS-9983278.

References

[1]Jameson, A., "Aerodynamic Shape Optimization Using the Adjoint Method," *VKI Lecture Series on Aerodynamic Drag Prediction and Reduction*, von Karman Inst. of Fluid Dynamics, Rhode St. Genese, Belgium, Feb. 2003.

[2]Jameson, A., Alonso, J. J., Reuther, J. J., Martinelli, L., and Vassberg, J. C., "Aerodynamic Shape Optimization Techniques Based on Control Theory," AIAA Paper 98-2538, June 1998.

[3]Nielsen, E. J., and Anderson, W. K., "Recent Improvements in Aerodynamic Optimization on Unstructured Meshes," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1155–1163.

[4]Elliot, J., and Peraire, J., "Practical Three-Dimensional Aerodynamic Design by Optimization," *AIAA Journal*, Vol. 35, No. 9, 1997, pp. 1479–1485.

[5]Löhner, R., Soto, O., and Yang, C., "An Adjoint-Based Design Methodology for CFD Optimization Problems," AIAA-Paper 2003-0299, Jan. 2003.

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

[7]Giles, M. B., Duta, M. C., and Muller, J.-D., "Adjoint Code Developments Using the Exact Discrete Approach," AIAA Paper 2001-2596, June 2001.

[8]Nielsen, E., Lu, J., Park, M., and Darmofal, D., "Exact Dual Adjoint Solution Method for Turbulent Flows on Unstructured Grids," *Computers and Fluids*, Vol. 33, No. 9, 2004, pp. 1131–1155; also AIAA Paper 2003-0272, Jan. 2003.

[9]Mavriplis, D. J., "Directional Agglomeration Multigrid Techniques for High-Reynolds-Number Viscous Flows," *AIAA Journal*, Vol. 37, No. 10, 1999, pp. 1222–1230.

[10]Mavriplis, D. J., and Pirzadeh, S., "Large-Scale Parallel Unstructured Mesh Computations for Three-Dimensional High-Lift analysis," *Journal of Aircraft*, Vol. 36, No. 6, 1999, pp. 987–998.

[11]Mavriplis, D. J., "An Assessment of Linear Versus Non-Linear Multigrid Methods for Unstructured Mesh Solvers," *Journal of Computational Physics*, Vol. 175, Jan. 2002, pp. 302–325.

[12]Nielsen, E. J., and Park, M., "Using an Adjoint Approach to Eliminate Mesh Sensitivities in Computational Design," AIAA Paper 2005-0491, Jan. 2005.

[13]Hirsch, C., *Numerical Computation of Internal and External Flows, Volume II: Computational Methods for Inviscid and Viscous Flows*, Wiley, New York, 1988, p. 32.

[14]Barth, T. J., and Linton, S. W., "An Unstructured Mesh Newton Solver for Compressible Fluid Flow and Its Parallel Implementation," AIAA Paper 95-0221, Jan. 1995.

[15]Mavriplis, D. J., "Multigrid Strategies for Viscous Flow Solvers on Anisotropic Unstructured Meshes," *Journal of Computational Physics*, Vol. 145, No. 1, 1998, pp. 141–165.

[16]Lallemand, M., Steve, H., and Dervieux, A., "Unstructured Multigridding by Volume Agglomeration: Current Status," *Computers and Fluids*, Vol. 21, No. 3, 1992, pp. 397–433.

[17]Smith, W. A., "Multigrid Solution of Transonic Flow on Unstructured Grids," *Recent Advances and Applications in Computational Fluid Dynamics*, edited by O. Baysal, American Society of Mechanical Engineers, New York, 1990, pp. 93–103.

[18]Mavriplis, D. J., and Venkatakrishnan, V., "A Unified Multigrid Solver for the Navier–Stokes Equations on Mixed Element Meshes," *International Journal for Computational Fluid Dynamics*, Vol. 8, 1997, pp. 247–263.

[19]Spalart, P. R., and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," *La Recherche Aérospatiale*, Vol. 1, 1994, pp. 5–21.

[20]Yang, Z., and Mavriplis, D. J., "Unstructured Dynamic Meshes with Higher-Order Time Integration Schemes for the Unsteady Navier–Stokes Equations," AIAA Paper 2005-1222, Jan. 2005.

[21]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, No. 1, 1998, pp. 231–245.

[22]Baker, T. J., "Mesh Movement and Metamorphosis," *Engineering with Computers*, Vol. 18, No. 3, 2002, pp. 188–198.

C. Bailly
Associate Editor