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

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

Introduction

The use of the adjoint equations is now well established for design-optimization problems in computational fluid dynamics (CFD) for the Euler and Navier-Stokes equations.[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 which 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.

In the continuous adjoint method approach, the original continuous governing equations are first linearized, and then discretized, while 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 straight-forward (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, since 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. Since the adjoint equations are closely related to the governing flow equations (they contain 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] Using this approach, any iterative scheme applied to the linearized governing equations may be modified to produce an adjoint solution scheme which 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 non-linear 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 in order to maintain good mesh cell qualities. The mesh deformation technique

*Professor, Dept. of Mechanical Engineering

must be robust, in order 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 mesh sensitivities, i.e. 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 $L(w^*(X(D)), X(D))$ to be minimized in an optimization problem, where D represents the design variables, X represents the grid point positions, and w^* denotes the (converged) flow-field variables which result from the solution of the discretized flow equations, given in residual form as $R(w^*(X(D)), X(D)) = 0$. 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 straight-forward differentiation:

$$\frac{dL}{dD} = \frac{\partial L}{\partial X}\frac{\partial X}{\partial D} + \frac{\partial L}{\partial w}\frac{\partial w}{\partial X}\frac{\partial X}{\partial D} \tag{1}$$

where the $*$ superscript which 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 $R(w^*(X(D)), X(D)) = 0$, which yields:

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

Upon substituting this expression into equation (1), we obtain:

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

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. While the Jacobian matrix is generally sparse, its inverse is dense, and it is not feasible in general to compute and store $\left[\frac{\partial R}{\partial w}\right]^{-1}$. Thus, for each design variable, the product

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

may be obtained by solving the system of equations given by:

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

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

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

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

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

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 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 Newton scheme is used to solve the non-linear flow equations as:

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

then the adjoint equations may be obtained by simply transposing the Jacobian matrix $\left[\frac{\partial R}{\partial w}\right]$. 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 $w_k - w_i$ by differences in undivided Laplacians i.e. $q_i = \sum_{k=1}^{neighbors} w_k - w_i$. Therefore, rewriting the Jacobian using the chain rule,

$$\left[\frac{dR}{dw}\right] = \frac{\partial R}{\partial w} + \frac{\partial R}{\partial q}\frac{\partial q}{\partial w} \tag{9}$$

the discrete adjoint can be obtained as

$$\left[\frac{dR}{dw}\right]^T = \left[\frac{\partial R}{\partial w}\right]^T + \left[\frac{\partial q}{\partial w}\right]^T \left[\frac{\partial R}{\partial q}\right]^T \tag{10}$$

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[13]), while 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. Assuming this matrix need not be stored due to its simplicity, the evaluation of $\left[\frac{dR}{dw}\right]^T \Lambda$ makes use of existing (edge-based) flow-solver data-structures, and requires the storage of two matrices: $\left[\frac{\partial R}{\partial w}\right]^T$ which is equivalent to a nearest-neighbor stencil-based sparse matrix, and $\left[\frac{\partial R}{\partial q}\right]^T$ 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 $\mathbf{R}(w)$ now depends on the flow variables w, as previously, 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 $\frac{\partial q}{\partial w}$ represents the linearization of the gradient formulation, which is again a matrix which need not be stored, since the matrix entries are grid metrics which are already stored for the gradient computations in the residual evaluation.[14]

Duality Preserving Iterative Strategy

The primal approach (c.f. equation (5)) and the dual approach (c.f. equation (7)) represent two techniques for evaluating the second term in equation (1). In the case of a single design variable and a single objective function, the two approaches yield the duality relation:

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

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

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

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

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

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

The preconditioning matrix $[P]$ should closely approximate the full Jacobian $\left[\frac{\partial R}{\partial w}\right]$, 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 non-linear problems in reference.[11] In this case, the non-linear flow equations are solved using the approximate Newton scheme:

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

where $R(w)$ represents the residual of the non-linear 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 non-linear flow equation solver described in reference,[11] a small number of linear multigrid cycles (2 to 5) was found to yield the optimal convergence efficiency. Similarly, for equations (12) and (13), a small number of multigrid cycles can be used to compute the approximate inverse matrix vector product:

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

for each outer iteration n, where $\left[\tilde{P}\right]^{-T}$ represents the approximate inverse of $[P]$. 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:

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

in the case of the adjoint problem, where k denotes 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 sub-matrices 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 sub-matrices 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 Figure 1. Each line is solved implicitly by taking the $[D]$ components as the union of all the $[P]$ matrix entries which correspond to points and edges within the lines, while 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 which employs latest available information.

In order 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 adjoint residual as:

$$\left[\frac{\partial R}{\partial w}\right]^T_{first-order} = \left[\frac{\partial R}{\partial w}\right]^T + \left[\frac{\partial R}{\partial q}\right]^T \quad (17)$$

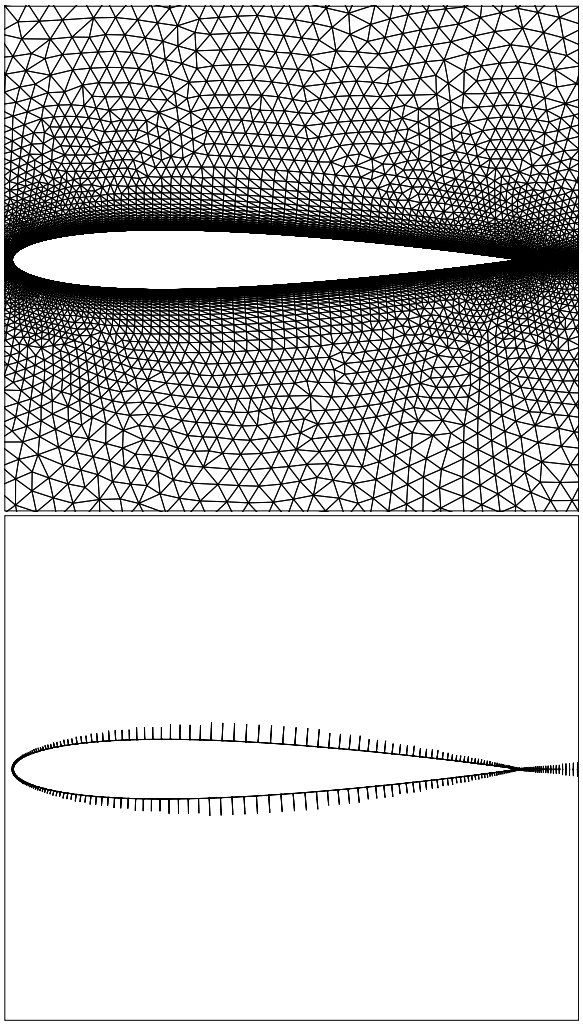


Fig. 1 Unstructured mesh for viscous flow over NACA0012 airfoil and set of lines constructed by graph algorithm for implicit line solver.

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

In practice, a separate copy of the $[D]^T$ block diagonals are factorized and stored in LU form, but the off-diagonal blocks $[O]^T$ are reconstructed as per equation (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 non-liner 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 which employs one pre-smoothing

and no post-smoothing iterations must be replaced by a cycle which uses no pre-smoothing and one post-smoothing iteration for the adjoint problem. Furthermore, the restriction and prolongation operators must be exact transposes of one-another.

The convergence of the linearized sensitivity equations (c.f. equation (5)) and the adjoint equations (c.f. equation (7)) for the inviscid flow over a NACA 0012 airfoil at a Mach number of 0.6 and an incidence of 1.25 degrees is shown in Figure 2. The unstructured mesh for this case contains 7884 vertices and is comprised of approximately isotropic triangular elements. Five multigrid levels are used, with four smoothing iterations on each grid level (two pre-smoothing, and two post-smoothing), 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 approximately invert the first-order Jacobian, prior to re-evaluating the right-hand-side of equations (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 non-linear solution of the flow equations. The convergence of the sensitivity derivative of the lift coefficient with respect to a vertical displacement of a surface grid-point (at x=63% chord on the lower surface), calculated by equation (5) and equation (7) is illustrated in Figures 3 and 4.

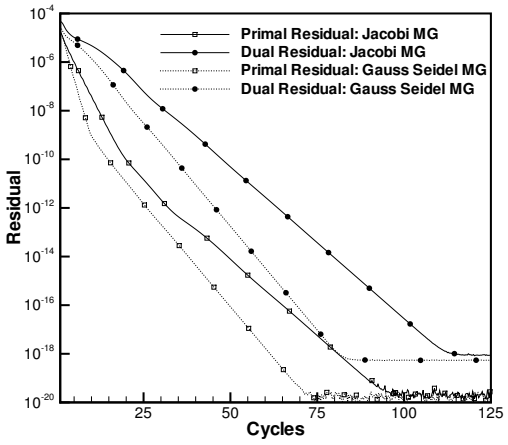


Fig. 2 Convergence of primal (linearized flow sensitivity) equations, and dual (adjoint) equations using Jacobi and Gauss-Seidel agglomeration multigrid driven defect-correction scheme.

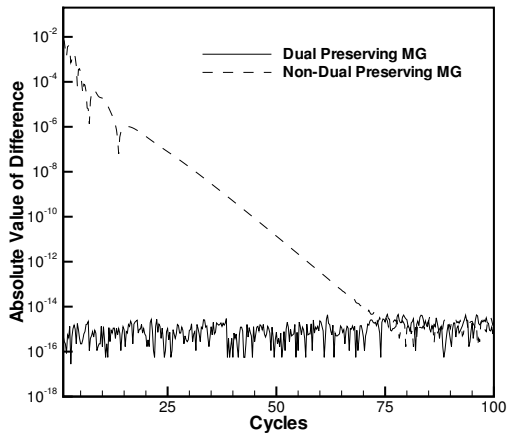


Fig. 3 Convergence history of difference between sensitivity derivatives calculated with primal and dual equations for duality preserving iterative scheme, and non-preserving scheme.

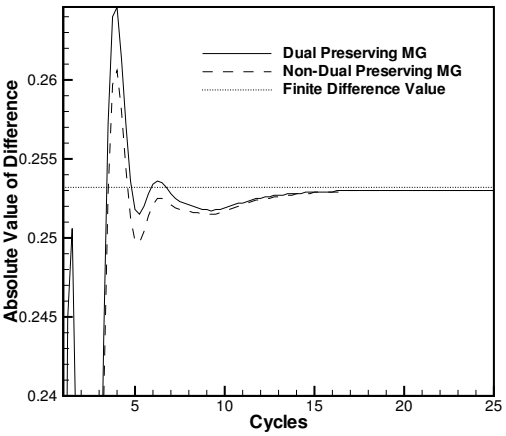


Fig. 4 Convergence history of sensitivity derivatives calculated with duality preserving iterative scheme, and non-preserving scheme compared with finite difference value.

In Figure 3, 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 non-duality 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 1.e-02, and the difference decreases to machine zero as the respective problems converge, while the difference remains at machine zero for the

duration of the iteration procedure for the duality preserving schemes. In Figure 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 non-preserving schemes, where these are also compared with the value obtained by finite-difference. 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 depicts 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 degrees, and a Reynolds number of 6.5 million. The mesh contains a total of 18,466 points, with a grid spacing at the wall in the normal direction of 1.e-06 chords, thus producing highly stretched mesh cells in these regions. The fully coupled flow and turbulence equations are solved using the approximate Newton method (c.f. equation (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 non-linear solution step. Four multigrid W-cycles (with 4 line Gauss Seidel smoothing iterations on each grid level), are employed at each non-linear step, achieving simultaneous convergence of the flow and turbulence equations over 100 non-linear iterations, as depicted in Figure 5. In Figure 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 Figure 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. 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 (i.e. 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. While each defect-correction cycle involves four multigrid cycles, it is noted that these are relatively inexpensive cycles, as the full second-order Jacobian is only evaluated once for each defect-correction cycle.

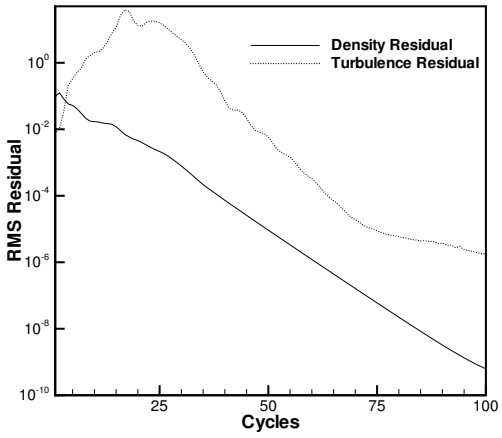


Fig. 5 Convergence of non-linear flow and turbulence equations for viscous turbulent NACA0012 airfoil as a function of the number of non-linear cycles, using four line-Gauss-Seidel driven agglomeration multigrid iterations for each non-linear cycle.

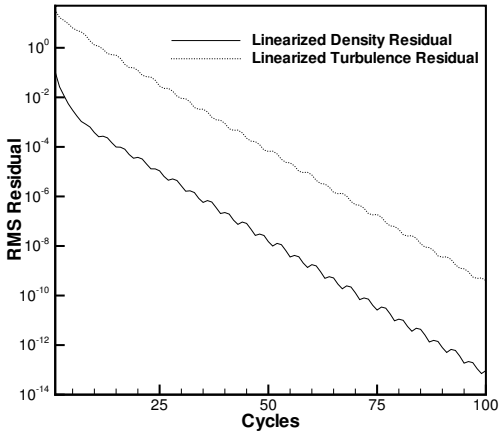


Fig. 6 Convergence of primal (linearized flow sensitivity) equations for viscous turbulent NACA0012 airfoil as a function of the number of defect-correction cycles, using four line-Gauss-Seidel driven agglomeration multigrid iterations for each defect-correction cycle.

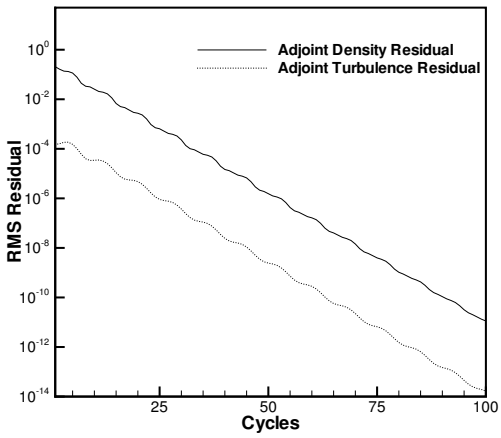


Fig. 7 Convergence of dual problem for viscous turbulent NACA0012 airfoil as a function of the number of defect-correction cycles, using four line-Gauss-Seidel driven agglomeration multigrid iterations for each defect-correction cycle.

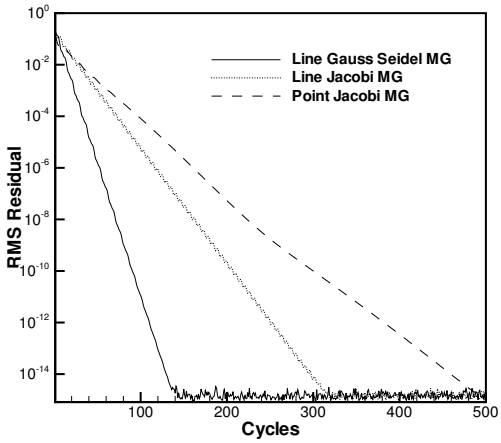
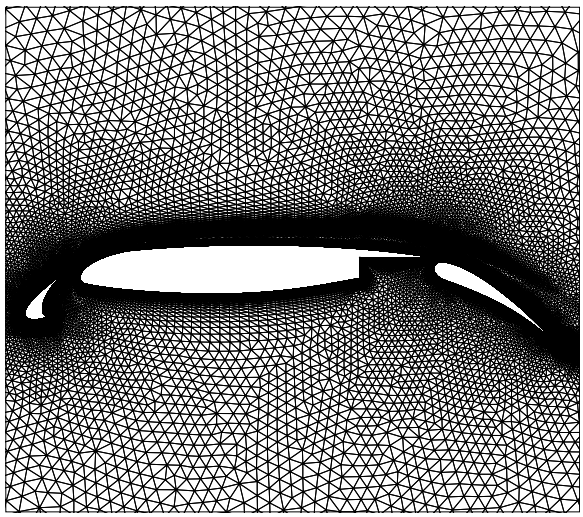


Fig. 8 Comparison of various smoothers for solution of adjoint problem using multigrid-driven defect-correction scheme for viscous airfoil case.

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. While the line-Jacobi solver delivers almost twice the convergence of the point Jacobi solver, the line Gauss Seidel approach delivers another factor of 2, for this case, enabling convergence to machine zero just over 100 defect-correction cycles. Although the figure depicts 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 depicts 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 degrees, and a Reynolds number of 5 million. This grid contains a total of 61,104 vertices, with a spacing at the airfoil surfaces of 1.e-06 chords. The solution was computed using the linear multigrid method described in reference,[11] with five multigrid levels, four line-Gauss-Seidel sweeps on each level, and four multigrid W-cycles between each non-linear update. The convergence, plotted in terms of non-linear cycles is shown in Figure 10. The overall convergence is slower for this complex configuration than for the simple airfoil case discussed previously, as had been previously shown in reference.[11] However, the flow residuals are converged to machine accuracy in approximately 300 cycles, while 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 Figure 11. The primal and dual problems are seen to converge at similar rates, and mirror the convergence rate of the non-linear flow problem depicted in Figure 10.



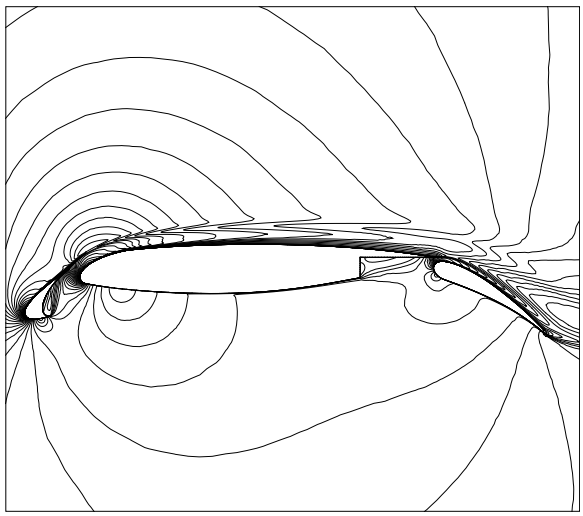


Fig. 9 Unstructured grid and computed Mach contours for flow over three-element airfoil configuration. Mach = 0.2, Incidence = 16 degrees, Re= 5 million. Number of grid points: 61,104

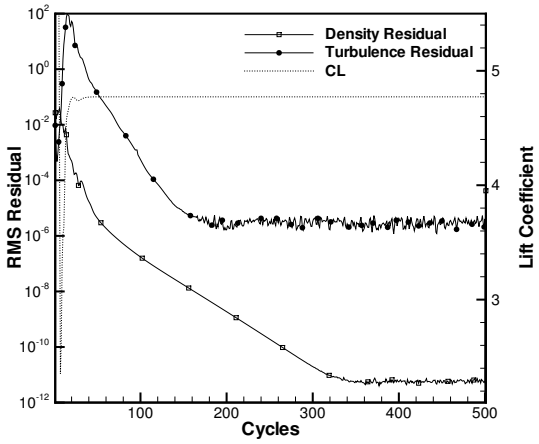


Fig. 10 Convergence of non-linear flow solution plotted in terms of non-linear updates, using linear agglomeration multigrid algorithm, with four multigrid W-cycles per non-linear update for three-element airfoil case.

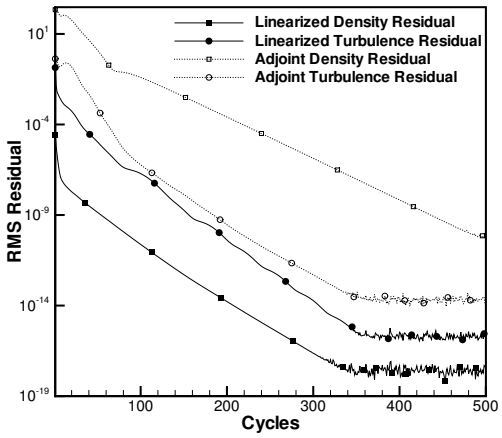


Fig. 11 Convergence of linearized sensitivity (primal) equations and adjoint (dual) equations using defect-correction scheme with four multigrid W-cycles per defect-correction update for three-element airfoil case.

Mesh Deformation Equations

Robust and efficient mesh deformation techniques are required for shape optimization problems in order 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:

$$\delta x_i = \frac{\sum_{j}^{neighbors} k_{ij}\delta x_j}{\sum_{j}^{neighbors} k_{ij}}, \quad \delta y_i = \frac{\sum_{j}^{neighbors} k_{ij}\delta y_j}{\sum_{j}^{neighbors} k_{ij}} \tag{18}$$

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

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:

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

where the vector δx represents the x and y displacements of all grid points, and $\delta x_{surface}$ 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 illustrated in Figure 12, for a highly-stretched mesh around the leading edge of an airfoil, which undergoes an outwards normal surface displacement. The bold line indicates the location of the displaced airfoil surface. The top part of the figure illustrates 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 towards the new airfoil boundary, but still lie close to their original position.

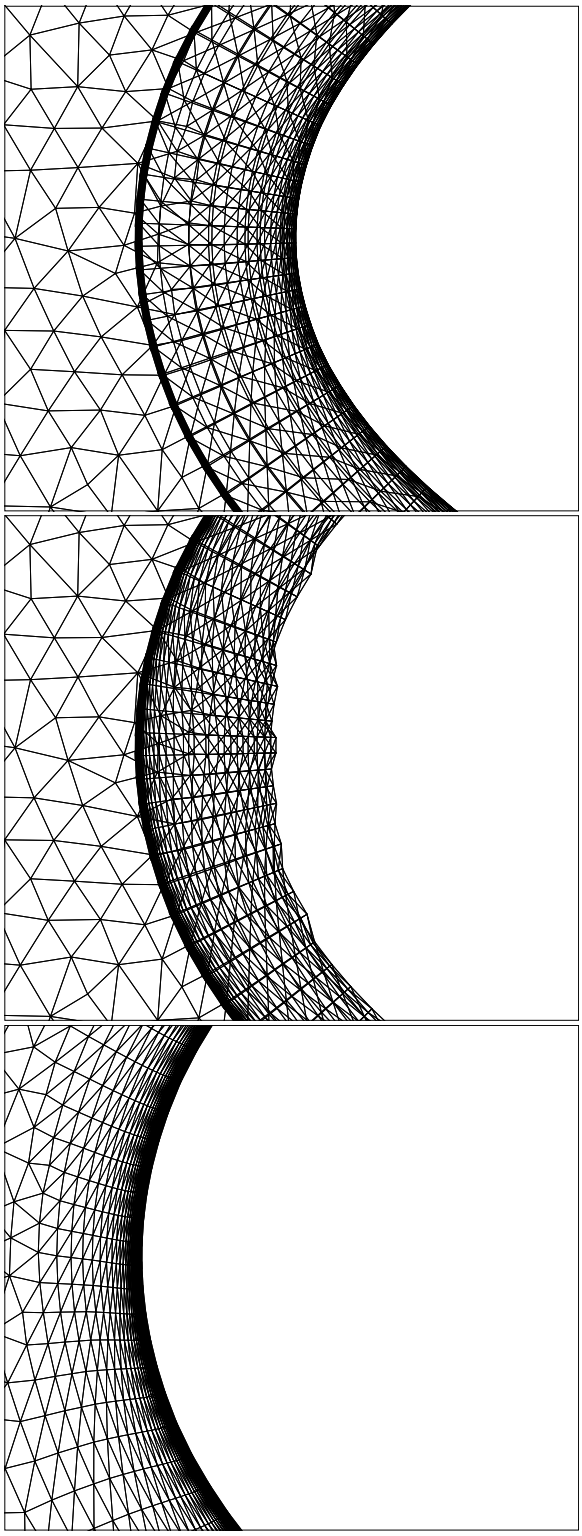


Fig. 12 Illustration of partially converged displaced viscous mesh after 10 point Jacobi single grid iterations (top), after a single line-Jacobi iteration (without multigrid), and after 6 line-Gauss-Seidel multigrid W-cycles.

In the second part of the figure, 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, as 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 Figure 12. Point or line Gauss-Seidel smoothers are especially effective for driving multigrid solvers for the mesh motion equations, as over-relaxation can be used to further accelerate convergence. Figure 13 depicts the convergence of the mesh motion equations for the viscous flow grid of Figure 1, using a simple Jacobi solver, a point Jacobi driven multigrid approach, a line Jacobi multigrid approach, and an over-relaxed 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.

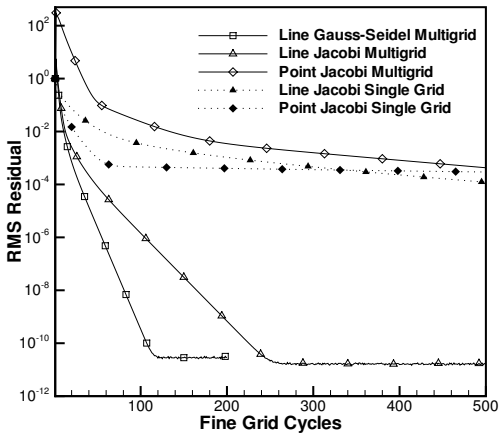


Fig. 13 Convergence of spring-analogy mesh displacement equations using point-Jacobi, line-Jacobi and line-Gauss-Seidel smoothers with agglomeration multigrid.

Mesh Sensitivities

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

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

which may be re-written as:

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

Since evaluating and storing the $[K]^{-1}$ matrix is generally not feasible, the evaluation of the grid sensitivities in the above manner 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] This approach requires the solution of the adjoint mesh equations of the form

$$[K]^T \Lambda_x = f \tag{22}$$

where Λ_x are the unknowns from which the grid sensitivities can be computed, and the construction of f is given in.[12] 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 over-relaxed Gauss Seidel scheme for solving the mesh adjoint equations. The convergence rates for the mesh adjoint equations on the grid of Figure 1 are depicted in Figure 14 showing similar convergence rates to the original mesh motion equations.

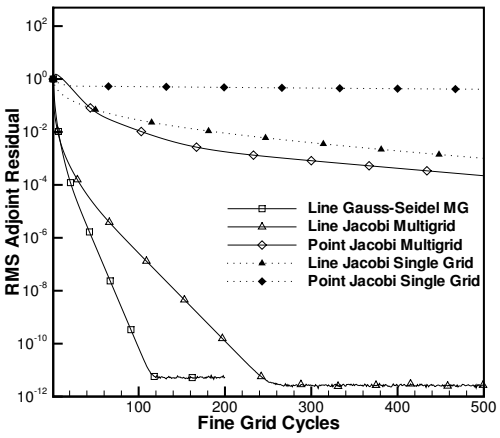


Fig. 14 Convergence of mesh displacement adjoint equations using point-Jacobi, line-Jacobi and line-Gauss-Seidel smoothers with agglomeration multigrid.

Optimization Example

The methodology described above 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.

For simplicity, the objective function only considers the pressure drag component. The optimization procedure follows the steepest descent approach described by Jameson.[1,2] Once the objective function sensitivities $\frac{dL}{dD}$ have been computed using the method described above, an increment in the design variables is prescribed as:

$$\delta D = -\lambda \frac{\tilde{dL}}{dD} \tag{23}$$

where λ represents a small time step, chosen small enough to ensure convergence of the optimization procedure, and $\frac{\tilde{dL}}{dD}$ represents the smoothed gradients $\frac{dL}{dD}$, obtained using an implicit smoothing technique, which is necessary to ensure smooth design shapes, as described in reference.[2] 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 depicted in Figure 1. This mesh contains a total of 18,466 grid points, with a normal grid spacing at the airfoil surface of 1.e-06 chords. The freestream Mach number is 0.729, the incidence is 2.31 degrees, and the Reynolds number if 6.5 million. Four multigrid levels were used, with four line-Gauss-Seidel smoothing passes on each mesh level, and the flow, and flow adjoint equations were run 50 defect-correction cycles at each design iteration, while 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 om the initial configuration, to 0.3798 and 0.0049 on the redesigned configuration, after 25 design cycles. The initial and final flow solutions are plotted in Figures 15 and 16, illustrating the reduction in the shock strength achieved by the design optimization process, while the optimization history of the drag and lift coefficients is shown in Figure 17.

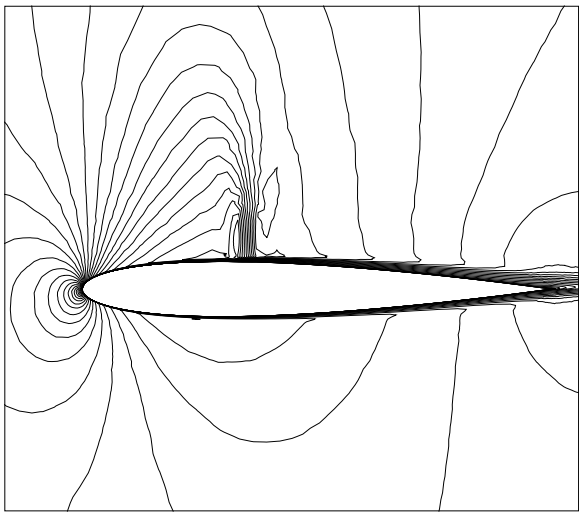


Fig. 15 Computed Mach contours for viscous turbulent flow over NACA0012 airfoil. Mach=0.729, Incidence=2.31°, Re=6.5 million.

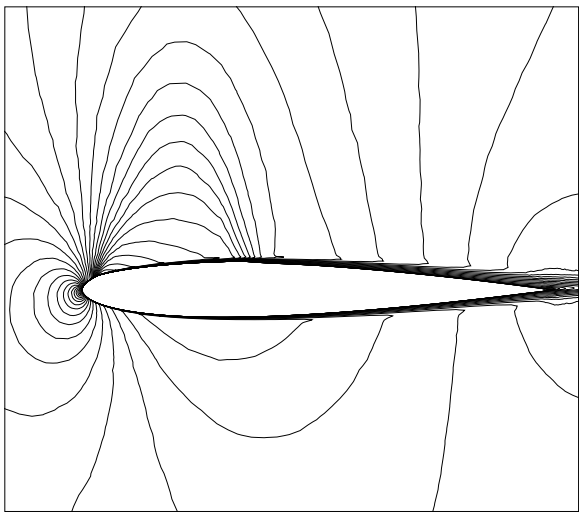


Fig. 16 Computed Mach contours for viscous turbulent flow over redesigned NACA0012 airfoil. Mach=0.729, Incidence=2.31°, Re= 6.5 million.

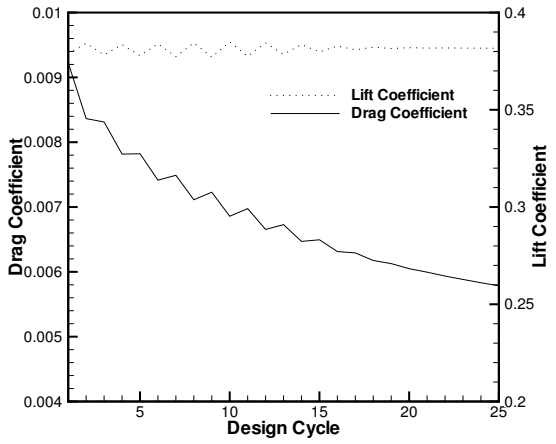


Fig. 17 Evolution of lift and drag coefficients as a function of the design cycle for drag minimization at fixed lift condition for viscous turbulent flow over NACA0012 airfoil.

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 Wyoming EPSCoR program, NSF grant number EPS-9983278.

References

[1]A. Jameson. Aerodynamic shape optimization using the adjoint method. VKI Lecture Series on Aerodynamic Drag Prediction and Reduction, von Karman Institute of Fluid Dynamics, Rhode St Genese, Belgium, February 2003.

[2]A. Jameson, J. J. Alonso, J. J. Reuther, L. Martinelli, and J. C. Vassberg. Aerodynamic shape optimization techniques based on control theory. AIAA Paper 98-2538, June 1998.

[3]E. J. Nielsen and W. K. Anderson. Recent improvements in aerodynamic optimization on unstructured meshes. *AIAA Journal*, 40(6):1155–1163, June 2002.

[4]J. Elliot and J. Peraire. Practical three-dimensional aerodynamic design by optimization. *AIAA Journal*, 35(9):1479–1485, 1997.

[5]R. Löhner O. Soto and C. Yang. An adjoint-based design methodology for CFD optimization problems. AIAA-Paper 2003-0299, 2003.

[6]E. J. Nielsen and W. K. Anderson. Recent improvements in aerodynamic design optimization on unstructured meshes. AIAA Paper 2001-0596, January 2001.

[7]M. B. Giles. Adjoint code developments using the exact discrete approach. AIAA Paper 2001-2596, June 2001.

[8]E. J. Nielsen, J. Lu, M. A. Park, and D. L. Darmofal. An exact dual adjoint solution method for turbulent flows on unstructured grids. AIAA Paper 2003-0272, January 2003.

[9]D. J. Mavriplis. Directional agglomeration multigrid techniques for high-Reynolds number viscous flows. AIAA paper 98-0612, January 1998.

[10]D. J. Mavriplis and S. Pirzadeh. Large-scale parallel unstructured mesh computations for 3D high-lift analysis. *AIAA Journal of Aircraft*, 36(6):987–998, December 1999.

[11]D. J. Mavriplis. An assessment of linear versus non-linear multigrid methods for unstructured mesh solvers. *Journal of Computational Physics*, 175:302–325, January 2002.

[12]E. J. Nielsen and M. Park. Using an adjoint approach to eliminate mesh sensitivities in computational design. AIAA-Paper 2005-0491, January 2005.

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

[14]T. J. Barth and S. W. Linton. An unstructured mesh Newton solver for compressible fluid flow and its parallel implementation. AIAA paper 95-0221, January 1995.

[15]D. J. Mavriplis. Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. *Journal of Computational Physics*, 145(1):141–165, September 1998.

[16]M. Lallemand, H. Steve, and A. Dervieux. Unstructured multigridding by volume agglomeration: Current status. *Computers and Fluids*, 21(3):397–433, 1992.

[17]W. A. Smith. Multigrid solution of transonic flow on unstructured grids. In *Recent Advances and Applications in Computational Fluid Dynamics*, November 1990. Proceedings of the ASME Winter Annual Meeting, Ed. O. Baysal.

[18]D. J. Mavriplis and V. Venkatakrishnan. A unified multigrid solver for the Navier-Stokes equations on mixed element meshes. *International Journal for Computational Fluid Dynamics*, 8:247–263, 1997.

[19]P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. *La Recherche Aérospatiale*, 1:5–21, 1994.

[20]Z. Yang and D. J. Mavriplis. Unstructured dynamic meshes with higher-order time integration schemes for the unsteady Navier-Stokes equations. AIAA-Paper 2005-1222, January 2005.

[21]C. Degand and C. Farhat. A three-dimensional torsional spring analogy method for unstructured dynamic meshes. *Computers and Structures*, 80:305–316, 2002.

[22]T. J. Baker. Mesh movement and metamorphosis. *Engineering with Computers*, 18(3):188–198, 2002.