AIAA JOURNAL
Vol. 45, No. 4, April 2007

Discrete Adjoint-Based Approach for Optimization Problems on Three-Dimensional Unstructured Meshes

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

DOI: 10.2514/1.22743

A comprehensive strategy for developing and implementing discrete adjoint methods for aerodynamic shape optimization problems is presented. By linearizing each procedure in the entire optimization problem, transposing each linearization, and reversing the sequential order of operations, the adjoint of the complete optimization problem, including flow equations and mesh motion equations is constructed in a modular and verifiable fashion. This construction is also shown to produce minimal memory overheads and retain the same convergence characteristics of the original analysis problem in the sensitivity analysis. These techniques are implemented in a three-dimensional unstructured multigrid Navier–Stokes solver and demonstrated on a transonic drag reduction problem for a wing body configuration.

Nomenclature

C_D	=	drag coefficient
C_L	=	lift coefficient
$\mathbf{D}$	=	vector of design variables
$[D]$	=	diagonal matrix elements
F	=	generic function
$[K]$	=	stiffness matrix for mesh motion equations
k_{ij}	=	edge-based spring coefficient
$\mathbf{L}$	=	objective function
$[O]$	=	off-diagonal matrix elements
$[P]$	=	preconditioning matrix
q	=	undivided Laplacian of flow variables w
$\mathbf{R}$	=	residual vector for flow equations
$\mathbf{w}$	=	vector of flow variables
$\mathbf{X}$	=	vector for grid point coordinates
$\mathbf{x}_{\text{int}}$	=	interior grid point coordinates
$\mathbf{x}_{\text{surf}}$	=	surface grid point coordinates
$\delta\mathbf{x}$	=	vector of mesh point displacements
$\mathbf{\Lambda}_w$	=	flow adjoint or costate variables
$\mathbf{\Lambda}_x$	=	mesh motion adjoint or costate variables
λ	=	step size for steepest descent algorithm

I. 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 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]. The continuous approach also provides a framework for dealing with discontinuous behavior in the discretization scheme [6,7]. On the other hand, the discrete adjoint 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 [3]. The discrete adjoint approach also benefits from a relative simplicity of implementation. This begins with the formulation of the primal or tangent problem, in which the discretized governing equations are linearized. The solution of these linearized equations constitutes the tangent problem for calculating sensitivity derivatives. The discrete adjoint formulation is then obtained by transposing each matrix produced in the tangent problem and performing all operations in reverse order. The derivation of the tangent problem is generally more intuitive than that of the adjoint problem, especially concerning the application of boundary conditions. In fact, often a linearization is required in the formulation of an implicit solution scheme and is thus readily available. The transposition of these operations constitutes a straightforward procedure, which is easily verifiable, as will be shown in this paper. Furthermore, as the formulation of the discrete problem is often constructed in a modular fashion with various sequentially invoked functions or subroutines, the tangent linearization and adjoint models may also be constructed in the same modular fashion, with a one-to-one correspondence between each original function and the corresponding tangent/adjoint function, which can then be verified individually at the component level before proceeding to the full simulation.

Although the full design optimization problem involves various operations including flow solution, surface deformation, and interior mesh deformation, most applications of adjoint methods have concentrated only on developing and solving the adjoint problem for the governing flow equations. The sensitivities resulting from other operations such as surface deformation and interior mesh deformation are generally either bypassed using simplified formulations (particularly in the case of structured grids) [2], or by finite differencing these components of the overall model [8], or by recomputing each sensitivity through a forward linearization [3]. All of these approaches either lack generality or result in a cost which is a strong function of the number of design variables. Recently, Nielsen and Park [9] have shown how the adjoint problem of the mesh motion equations can be used to obtain the mesh sensitivities at a cost which is independent of the overall number of design variables. This approach has also been implemented in two dimensions in previous work by the author [10].

In this paper, we attempt to generalize the idea of using the adjoint model for operations other than simply the solution of the flow equations. Rather, we formulate the adjoint for the complete design

Presented as Paper 0050 at the Aerospace Sciences Meeting and Exhibit, Reno, NV, 9–12 January 2006; received 25 January 2006; revision received 10 September 2006; accepted for publication 20 September 2006.

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

optimization problem, involving all phases from the prescription of design variables to the computation of the sensitivities. This is done by first forming the tangent (or forward differentiation) model of the complete design optimization procedure, by specifically linearizing each procedure. The adjoint model is then composed by transposing each operation and performing all operations in reverse order. Exact duality is preserved between tangent and adjoint models for the entire design optimization procedure, and this is built up by first demonstrating duality for each individual operation or component and then assembling the components in the appropriate order to formulate the complete tangent and adjoint optimization problems.

The construction of the tangent and adjoint models is designed to mimic the construction of the original problem at each individual stage. Thus, second-order accurate tangent and adjoint flow discretizations are constructed using a two-pass approach, similar to the approach used for the nonlinear flow equations. A single modular solution algorithm, based on agglomeration multigrid with line preconditioning [10–12], is used to solve all problems arising in the optimization procedure, including the nonlinear flow equations, the mesh motion equations, and the tangent and adjoint models of both of these sets of equations, when required.

The proposed approach provides a mechanism for systematically obtaining sensitivities for complicated procedures such as those involved in multidisciplinary design optimization processes. Furthermore, the one-to-one correspondence of the sensitivity analysis with the original problem formulation ensures that no additional data structures are required in the formulation of the sensitivity analysis over and above those used in the original problem, and the use of equivalent solvers guarantees similar convergence rates and overall efficiency.

In this manner, the current approach parallels what could be achieved using an automatic differentiation procedure such as ADIFOR or ADJIFOR [13], but the manual implementation affords extra flexibility in terms of data structures and solution techniques and results in a more thorough understanding of the implementation.

II. General Sensitivity Formulation

Consider a simulation which produces various outputs $\mathbf{L}$ each of which constitutes a cost function which may consist of objectives and constraints, which we wish to minimize by varying certain parameters or design variables $\mathbf{D}$ in the simulation. The entire simulation begins with the specification of the design variables $\mathbf{D}$ and may involve the evaluation of multiple functions F_i or sequential steps to finally obtain the values of the outputs $\mathbf{L}$. Thus, the entire procedure may be written as

$$\mathbf{L}(\mathbf{D}) = \mathbf{L}(F_{n-1}(F_{n-2}(\cdots F_2(F_1(\mathbf{D}))\cdots))) \tag{1}$$

where n denotes the number of consecutive functional evaluations required in the construction of $\mathbf{L}$. A variation in the design variables $\delta\mathbf{D}$ produces a corresponding variation of the outputs or objectives $\delta\mathbf{L}$ as

$$\delta\mathbf{L} = \frac{d\mathbf{L}}{d\mathbf{D}}\delta\mathbf{D} \tag{2}$$

where the sensitivity derivative may be calculated as

$$\frac{d\mathbf{L}}{d\mathbf{D}} = \frac{\partial\mathbf{L}}{\partial F_{n-1}}\cdot\frac{\partial F_{n-1}}{\partial F_{n-2}}\cdots\frac{\partial F_2}{\partial F_1}\cdot\frac{\partial F_1}{\partial\mathbf{D}} \tag{3}$$

For an arbitrary number of design variables and objective functions, these sensitivity derivatives constitute a rectangular matrix which will generally be costly to evaluate. However, we consider two special cases, first the case where we have a single design variable and an arbitrary number of objective functions, and secondly the case where we have a single objective function and an arbitrary number of design variables. In the first case, the derivative $\partial F_1/\partial\mathbf{D}$ constitutes a vector which is either given, or may be assumed to be easily computable. Because only the first derivative on the right-hand side of Eq. (3) depends on $\mathbf{L}$, it proves economical to precompute the product of all the other derivatives as

$$\frac{d\mathbf{L}}{d\mathbf{D}} = \frac{\partial\mathbf{L}}{\partial F_{n-1}}\cdot\left[\frac{\partial F_{n-1}}{\partial F_{n-2}}\left[\cdots\left[\frac{\partial F_2}{\partial F_1}\left[\frac{\partial F_1}{\partial\mathbf{D}}\right]\right]\right]\right] \tag{4}$$

Thus, the final result in brackets is a vector which is obtained as a series of matrix-vector products. This vector may be stored and then used to compute the entire vector $\frac{d\mathbf{L}}{d\mathbf{D}}$ (one element for each individual objective function L) in a single matrix-vector multiplication. This constitutes the forward differentiation model or the tangent model.

In the case where a single objective function is specified but multiple design variables are present, the adjoint model provides the most economical approach for the calculation of $\frac{d\mathbf{L}}{d\mathbf{D}}$. For this purpose, we transpose Eq. (3), obtaining

$$\frac{d\mathbf{L}^T}{d\mathbf{D}} = \frac{\partial F_1^T}{\partial\mathbf{D}}\cdot\frac{\partial F_2^T}{\partial F_1}\cdot\frac{\partial F_{n-1}^T}{\partial F_{n-2}}\cdots\frac{\partial\mathbf{L}^T}{\partial F_{n-1}} \tag{5}$$

Noting that $\partial\mathbf{L}^T/\partial F_{n-1}$ is a simple vector which is either given or easily computable, and $\partial F_1^T/\partial\mathbf{D}$ is the only term which depends on the multitude of design variables, the corresponding strategy is to precompute the rightmost derivatives as

$$\frac{d\mathbf{L}^T}{d\mathbf{D}} = \frac{\partial F_1^T}{\partial\mathbf{D}}\cdot\left[\frac{\partial F_2^T}{\partial F_1}\cdot\left[\frac{\partial F_{n-1}^T}{\partial F_{n-2}}\cdot\left[\cdots\left[\frac{\partial\mathbf{L}^T}{\partial F_{n-1}}\right]\right]\right]\right] \tag{6}$$

The vector of sensitivities for all design variables $d\mathbf{L}^T/d\mathbf{D}$ may then be obtained with a single matrix-vector multiplication involving the matrix $\partial F_1^T/\partial\mathbf{D}$ with the precomputed vector obtained from the sequence of bracketed operations.

III. Design Optimization Problem

Although design optimization problems may involve various design variables including nongeometric variables (i.e., mass flow rates, etc.) in this paper we will restrict ourselves to shape optimization problems, where the considered shape is defined by a set of geometric design variables. In general, the shape design optimization process consists of at least four sequential steps. Assuming an existing geometry and computational mesh already exist, and the design variables and objective functions are specified, the initial step consists of using the design variables to define a new deformed surface mesh geometry, producing specific values for the coordinates of the surface mesh. The surface mesh coordinates are then used to compute the values of the interior mesh coordinates, usually through a mesh deformation technique which aims to provide a smooth interior mesh with no overlapping cells. The third step involves the computation of the flowfield on this new deformed mesh, and the fourth (final) step involves the computation of the objective function using the newly computed flowfield. Following the notation used in Eq. (1), the design optimization problem can be written as

$$\mathbf{L}(\mathbf{D}) = \mathbf{L}(F_3(F_2(F_1(\mathbf{D})))) \tag{7}$$

with

$$\mathbf{x}_{\text{surf}} = F_1(\mathbf{D}) \tag{8}$$

$$\mathbf{x}_{\text{int}} = F_2(\mathbf{x}_{\text{surf}}) \tag{9}$$

$$\mathbf{w} = F_3(\mathbf{x}_{\text{int}}) \tag{10}$$

$$\mathbf{L} = \mathbf{L}(\mathbf{w}, \mathbf{x}_{\text{int}}) \tag{11}$$

where $\mathbf{D}$ and $\mathbf{L}$ represent the design variables and objectives as previously, $\mathbf{x}_{\text{surf}}$ represents the surface grid point coordinates, $\mathbf{x}_{\text{int}}$ denotes the interior grid point coordinates (which may include the

surface points as well), and $\mathbf{w}$ corresponds to the flow variables computed on the deformed grid. Actually, this equation is slightly more complicated than the formulation given in Eq. (1) due to the fact that the objective functions considered in this work, based on lift and drag coefficients, are functions of both the flow variables and the grid point coordinates as stated in Eq. (11). Nevertheless, a straightforward differentiation of Eq. (11) leads to the following expression for the sensitivity derivatives:

$$\frac{d\mathbf{L}}{d\mathbf{D}} = \frac{\partial \mathbf{L}}{\partial \mathbf{w}} \cdot \frac{\partial \mathbf{w}}{\partial \mathbf{x}_{\text{int}}} \cdot \frac{\partial \mathbf{x}_{\text{int}}}{\partial \mathbf{x}_{\text{surf}}} \cdot \frac{\partial \mathbf{x}_{\text{surf}}}{\partial \mathbf{D}} + \frac{\partial \mathbf{L}}{\partial \mathbf{x}_{\text{int}}} \cdot \frac{\partial \mathbf{x}_{\text{int}}}{\partial \mathbf{x}_{\text{surf}}} \cdot \frac{\partial \mathbf{x}_{\text{surf}}}{\partial \mathbf{D}} \tag{12}$$

We next examine each term in the above equation. The $\partial \mathbf{x}_{\text{surf}}/\partial \mathbf{D}$ term corresponds to the sensitivity of the surface mesh with respect to the design variables. For each design variable, this can be obtained through a linearization of the design variable definition, as may be possible in parametric modeling packages. Alternatively, these vectors may be obtained by finite differencing CAD-based design variable definitions. In any case we will assume these are known vectors for each design variable.

The $\partial \mathbf{x}_{\text{int}}/\partial \mathbf{x}_{\text{surf}}$ term corresponds to the sensitivity of the interior mesh point positions with respect to surface point displacements. In general, interior mesh deformation is governed by a discretized set of equations, such as those arising from a system of springs based on the mesh edges [14–16], or by invoking linear elasticity analogies [3,17,18]. In such cases, the interior mesh displacements are computed by solving the equation

$$[K]\delta \mathbf{x}_{\text{int}} = \delta \mathrm{x}_{\text{surf}} \tag{13}$$

where $[K]$ represents the stiffness matrix obtained from the discrete mesh motion equations. The sensitivities can thus be written conceptually as

$$\frac{\partial \mathbf{x}_{\text{int}}}{\partial \mathbf{x}_{\text{surf}}} = [K]^{-1} \tag{14}$$

This notation is used to imply that the product of the sensitivities with an arbitrary surface mesh vector f is equivalent to $[K]^{-1} f$. This is also consistent with the derivation given in [9,10].

An expression for the $\partial \mathbf{w}/\partial \mathbf{x}_{\text{int}}$ term is obtained by considering the constraint imposed by the flow equations. When the mesh is deformed, the discrete flow equations are no longer satisfied and must be recomputed. If $\mathbf{R}$ represents the residual of the discretized flow equations, linearization about the converged state $\mathbf{R}(\mathbf{w}(\mathbf{x}_{\text{int}}), \mathbf{x}_{\text{int}}) = 0$ in the absence of any other variations in the flow conditions results in the following relation:

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}\right] \frac{\partial \mathbf{w}}{\partial \mathbf{x}_{\text{int}}} = -\frac{\partial \mathbf{R}}{\partial \mathbf{x}_{\text{int}}} \tag{15}$$

which may be rewritten as

$$\frac{\partial \mathbf{w}}{\partial \mathbf{x}_{\text{int}}} = -\left[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}\right]^{-1} \frac{\partial \mathbf{R}}{\partial \mathbf{x}_{\text{int}}} \tag{16}$$

Substituting the above expressions into Eq. (12) we obtain the final expression:

$$\frac{d\mathbf{L}}{d\mathbf{D}} = -\frac{\partial \mathbf{L}}{\partial \mathbf{w}} \cdot \left[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}\right]^{-1} \frac{\partial \mathbf{R}}{\partial \mathbf{x}_{\text{int}}} [K]^{-1} \frac{\partial \mathbf{x}_{\text{surf}}}{\partial \mathbf{D}} + \frac{\partial \mathbf{L}}{\partial \mathbf{x}_{\text{int}}} \cdot [K]^{-1} \frac{\partial \mathbf{x}_{\text{surf}}}{\partial \mathbf{D}} \tag{17}$$

For a single design variable, the tangent model can be implemented in the following steps:

1) Obtain surface mesh sensitivity $\partial \mathbf{x}_{\text{surf}}/\partial \mathbf{D}$ from design variable definition.

2) Obtain interior mesh sensitivity by iteratively solving

$$[K] \frac{\partial \mathbf{x}_{\text{int}}}{\partial \mathbf{D}} = \frac{\partial \mathbf{x}_{\text{surf}}}{\partial \mathbf{D}} \quad \text{or} \quad \frac{\partial \mathbf{x}_{\text{int}}}{\partial \mathbf{D}} = [K]^{-1} \frac{\partial \mathbf{x}_{\text{surf}}}{\partial \mathbf{D}} \tag{18}$$

3) Get flow *residual* sensitivities by performing the matrix-vector product:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{D}} = \frac{\partial \mathbf{R}}{\partial \mathbf{x}_{\text{int}}} \cdot \frac{\partial \mathbf{x}_{\text{int}}}{\partial \mathbf{D}} \tag{19}$$

Get the flow *variable* sensitivities by iteratively solving the equation:

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}\right] \frac{\partial \mathbf{w}}{\partial \mathbf{D}} = -\frac{\partial \mathbf{R}}{\partial \mathbf{D}} \tag{20}$$

5) Compute final sensitivity as

$$\frac{d\mathbf{L}}{d\mathbf{D}} = \frac{\partial \mathbf{L}}{\partial \mathbf{w}} \cdot \frac{\partial \mathbf{w}}{\partial \mathbf{D}} + \frac{\partial \mathbf{L}}{\partial \mathbf{x}_{\text{int}}} \cdot \frac{\partial \mathbf{x}_{\text{int}}}{\partial \mathbf{D}} \tag{21}$$

The adjoint model is obtained by transposing Eq. (17) which yields

$$\frac{d\mathbf{L}^T}{d\mathbf{D}} = \frac{\partial \mathbf{x}_{\text{surf}}^T}{\partial \mathbf{D}} [K]^{-T} \left[\frac{\partial \mathbf{L}^T}{\partial \mathbf{x}_{\text{int}}} - \frac{\partial \mathbf{R}^T}{\partial \mathbf{x}_{\text{int}}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}\right]^{-T} \frac{\partial \mathbf{L}^T}{\partial \mathbf{w}}\right] \tag{22}$$

where the superscript $-T$ denotes the inverse of the transposed matrix, and factorization has also been used for further simplification. This corresponds to the formulation derived in [9] and implemented for two-dimensional problems in [10]. For a single objective function, the adjoint model consists of the following steps:

1) Obtain objective flow sensitivity vector $\frac{\partial \mathbf{L}}{\partial \mathbf{w}}$.

2. Iteratively solve adjoint flow problem given as

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}\right]^T \mathbf{\Lambda}_w = \frac{\partial \mathbf{L}^T}{\partial \mathbf{w}} \quad \text{or} \quad \mathbf{\Lambda}_w = \left[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}\right]^{-T} \frac{\partial \mathbf{L}^T}{\partial \mathbf{w}} \tag{23}$$

3) Obtain objective sensitivities with respect to interior mesh deformation with

$$\frac{d\mathbf{L}^T}{d\mathbf{x}_{\text{int}}^*} = \frac{\partial \mathbf{L}^T}{\partial \mathbf{x}_{\text{int}}} - \frac{\partial \mathbf{R}^T}{\partial \mathbf{x}_{\text{int}}} \mathbf{\Lambda}_w \tag{24}$$

which requires evaluating the matrix-vector product $(\partial \mathbf{R}^T/\partial \mathbf{x}_{\text{int}})\mathbf{\Lambda}_w$.

4) Iteratively solve the adjoint mesh deformation problem given as

$$[K]^T \mathbf{\Lambda}_x = \frac{d\mathbf{L}^T}{d\mathbf{x}_{\text{int}}^*} \quad \text{or} \quad \mathbf{\Lambda}_x = [K]^{-T} \frac{d\mathbf{L}^T}{d\mathbf{x}_{\text{int}}^*} \tag{25}$$

5) Compute final sensitivity as

$$\frac{d\mathbf{L}^T}{d\mathbf{D}} = \frac{\partial \mathbf{x}_{\text{surf}}^T}{\partial \mathbf{D}} \mathbf{\Lambda}_x \quad \text{or} \quad \frac{d\mathbf{L}}{d\mathbf{D}} = \mathbf{\Lambda}_x^T \frac{\partial \mathbf{x}_{\text{surf}}}{\partial \mathbf{D}} \tag{26}$$

where the second form may prove to be simpler to implement for cases where the transpose of the design variable to surface mesh sensitivities is not readily available.

Although the steps described above for both the tangent and the adjoint models appear to be relatively straightforward, steps 2 and 4 in both cases require the solution of a set of field equations, while step 3 requires the evaluation of a complicated matrix-vector product. Additionally, for the adjoint model, the term $\partial \mathbf{L}/\partial \mathbf{x}_{\text{int}}$ must be constructed explicitly. Techniques for implementing each of these steps effectively with no additional data structures other than those present in the implicit flow and mesh motion solvers are described in the next section.

IV. Flow Tangent and Flow Adjoint Problem

Step 4 in the tangent formulation and step 2 in the adjoint formulation both require the solution of a linearized flow problem given by Eqs. (20) and (23) respectively. $[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}]$ in these equations represents the Jacobian of the discretized flow equations, which for second-order accurate discretizations on unstructured meshes, involves an extended neighbors-of-neighbors stencil. The storage and inversion of this full Jacobian is generally considered impractical. On the other hand, the nonlinear second-order accurate residual for the flow equations is generally constructed in a two-pass

approach, with each pass operating on a nearest-neighbor stencil. 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, that is,

$$q_i = \sum_{k=1}^{\text{neighbors}} w_k - w_i$$

while a reconstruction scheme first constructs gradients $q(w)$, which are used to extrapolate reconstructed flow variables to the control-volume face where the flux is computed. Therefore, the full Jacobian may be evaluated using the chain rule as

$$\left[\frac{dR}{dw}\right] = \frac{\partial R}{\partial w} + \frac{\partial R}{\partial q}\,\frac{\partial q}{\partial w} \tag{27}$$

whereas the discrete flow adjoint can be obtained as

$$\left[\frac{dR}{dw}\right]^T = \left[\frac{\partial R}{\partial w}\right]^T + \left[\frac{\partial q}{\partial w}\right]^T \left[\frac{\partial R}{\partial q}\right]^T \tag{28}$$

The action of this Jacobian on a field vector can now be evaluated in a two-pass approach as

$$\Delta q = \frac{\partial q}{\partial w} \cdot \Delta w \tag{29}$$

$$\Delta R = \frac{\partial R}{\partial w} \cdot \Delta w + \frac{\partial R}{\partial q} \cdot \Delta q \tag{30}$$

or, for the adjoint problem,

$$\Delta q = \left[\frac{\partial R}{\partial q}\right]^T \Delta R \tag{31}$$

$$\Delta w = \left[\frac{\partial R}{\partial w}\right]^T \Delta R + \left[\frac{\partial q}{\partial w}\right]^T \Delta q \tag{32}$$

For the tangent model [cf. Eqs. (29) and (30)], this procedure corresponds to a straightforward linearization of each stage in the nonlinear residual construction, whereas for the adjoint model [cf. Eqs. (31) and (32)] this procedure corresponds to using the transpose of each individual operation in the tangent model, but applied in reverse order. (Note that in general the adjoint problem involves the use of the costate variables denoted as Λ_w, although in this case we have used the symbol ΔR to emphasize the relationship between the above sets of equations.) Although these procedures do not necessarily simplify the construction and storage of the full Jacobian, they enable a simple procedure for computing the (linear) residuals of Eqs. (20) and (23), which can be used to drive an iterative solution strategy. A standard preconditioning strategy for solving Eq. (20) can be written as

$$[P]\left(\frac{\partial \mathbf{w}^{n+1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{w}^{n}}{\partial \mathbf{D}}\right) = -\frac{\partial \mathbf{R}}{\partial \mathbf{D}} - \left[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}\right]\left(\frac{\partial \mathbf{w}}{\partial \mathbf{D}}\right)^n \tag{33}$$

where $[P]$ is a preconditioning matrix to be inverted. The corresponding duality preserving scheme for solving the adjoint equations [cf. Eq. (23)] can be written as [10,19]

$$[P]^T(\mathbf{\Lambda}_{\mathbf{w}}^{n+1} - \mathbf{\Lambda}_{\mathbf{w}}^{n}) = \frac{\partial \mathbf{L}}{\partial \mathbf{w}} - \left[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}\right]^T \mathbf{\Lambda}_{\mathbf{w}}^{n} \tag{34}$$

The right-hand sides of Eqs. (33) and (34) represent the linear flow residual and adjoint residual, respectively. Because these contain a matrix-vector product involving the full second-order accurate Jacobian $[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}]$, they must be evaluated using the two-pass construction given by Eqs. (29–32) for the primal and dual problems, respectively.

The preconditioning matrix $[P]$ should closely approximate the full Jacobian $[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}]$, 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 [20]. In this case, the nonlinear flow equations are solved using the approximate Newton scheme:

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}\right]_{\text{first order}} \Delta \mathbf{w} = -\mathbf{R}(\mathbf{w}) \tag{35}$$

where $\mathbf{R}(\mathbf{w})$ represents 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, provided the linear system is fully converged at each nonlinear step. Similarly, for the linearized equations (20) and (23), 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 [20], a small number of linear multigrid cycles (2–5) was found to yield the optimal convergence efficiency. Similarly, for Eqs. (20) and (23), a small number of multigrid cycles can be used to compute the approximate inverse matrix-vector product (given here in the case of the adjoint problem):

$$\left(\mathbf{\Lambda}_{\mathbf{w}}^{n+1} - \mathbf{\Lambda}_{\mathbf{w}}^{n}\right) = [\tilde{P}]^{-T}\left[\frac{\partial \mathbf{L}}{\partial \mathbf{w}} - \left[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}\right]^T \mathbf{\Lambda}_{\mathbf{w}}^{n}\right] \tag{36}$$

for each outer iteration n, where $[\tilde{P}]^{-T}$ represents the approximate inverse of $[P]^T$, with a corresponding expression for the tangent problem. In this work we use a line-preconditioned block-Jacobi 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

$$[D]^T\left(\mathbf{\Lambda}_{\mathbf{w}}^{n+1} - \mathbf{\Lambda}_{\mathbf{w}}^{n}\right)^{k+1} = \frac{\partial \mathbf{L}}{\partial \mathbf{w}} - \left[\frac{\partial \mathbf{R}}{\partial \mathbf{w}}\right]^T \mathbf{\Lambda}_{\mathbf{w}}^{n} - [O]^T\left(\mathbf{\Lambda}_{\mathbf{w}}^{n+1} - \mathbf{\Lambda}_{\mathbf{w}}^{n}\right)^{k} \tag{37}$$

in the case of the adjoint problem, where k denotes the iterative smoothing counter. For viscous turbulent flows, the linearization must include the full coupling between the flow equations and the turbulence model. The block submatrices are thus 6×6 blocks, using a single equation turbulence model in three dimensions. Furthermore, a line-implicit scheme is employed to relieve the stiffness associated with high mesh stretching in boundary-layer and wake regions [11,21]. 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. 1a. 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 strategy in highly stretched regions of the mesh, which reverts to a point Jacobi scheme in isotropic regions of the mesh. This solver is used as a smoother on each grid level in the agglomeration multigrid algorithm, where the coarse grid levels are formed by grouping together neighboring control volumes to form sets of larger but fewer control volumes on each level, as depicted in Fig. 1b.

Because the full Jacobian is only required to evaluate the tangent and adjoint residuals of Eqs. (33) and (34) in the above iterative

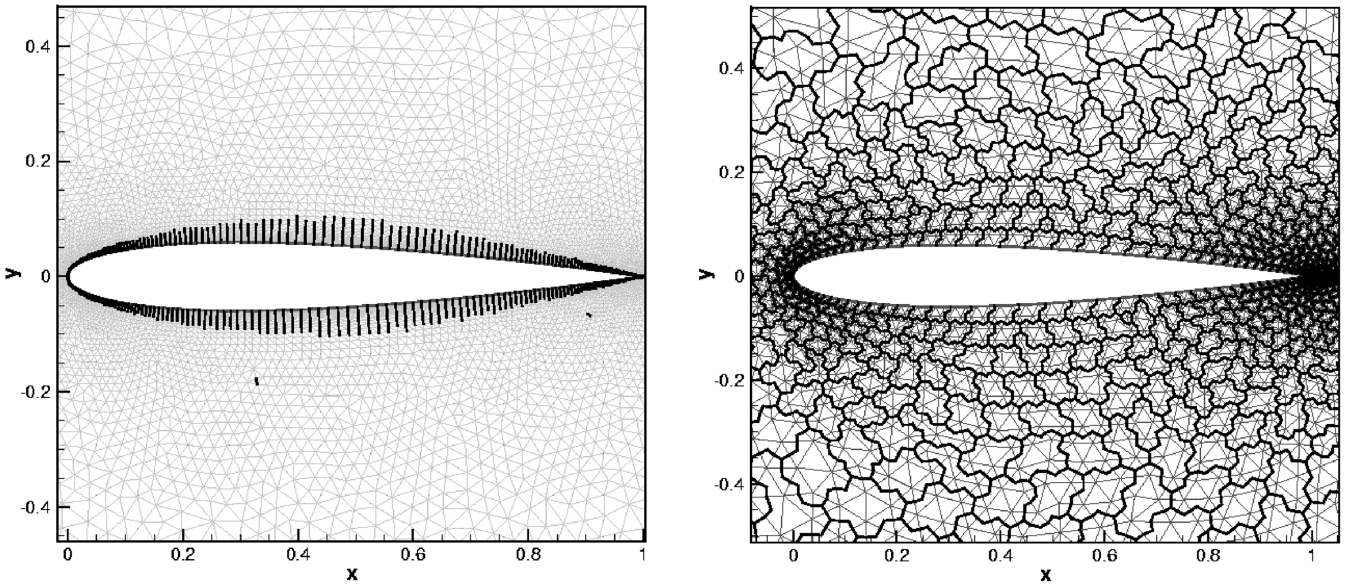


Fig. 1 a) Illustration in two dimensions of line construction for a directional implicit solver and b) coarse agglomerated multigrid level used for flow and mesh analysis, tangent and adjoint problems.

solution strategies, and this can be done using the two-pass approach described above, the question arises as to whether the individual matrices in Eqs. (27) and (28) should be stored and reused for the entire solution process, or reconstructed each time a residual evaluation is performed. The fact that only a small number of (multigrid) subiterations are generally employed in the defect-correction scheme implies that a relatively large number of residual evaluations will be required during the solution process, which favors the storage approach over the recomputation approach. However, memory limitations may make the recomputation approach necessary for large problems. On the other hand, it should be noted that preconditioning matrix $[P]$ is always stored and can be used to reconstruct the full Jacobian component matrices. Because the first-order discretization is obtained by replacing the reconstructed variables q with the original flow variables w (and thus $\frac{\partial q}{\partial w} = 1$ for this case) we have the relation:

$$[P] = \left[\frac{\partial R}{\partial w}\right]_{\text{first order}} = \left[\frac{\partial R}{\partial w}\right] + \alpha\left[\frac{\partial R}{\partial q}\right] \tag{38}$$

where α is a scalar coefficient, which should nominally be unity, but may be required to account for different scalings between first- and second-order dissipation terms. Using this expression, the formulation of the full Jacobian in Eq. (27) can be rewritten as

$$\left[\frac{dR}{dw}\right] = \left[\frac{\partial R}{\partial w}\right]_{\text{first order}} + \left[\frac{\partial R}{\partial q}\right]\left[\frac{\partial q}{\partial w} - [I]\alpha\right] \tag{39}$$

with a corresponding transposed equation for the adjoint formulation. However, recalling that the preconditioning matrix is split into $[D]$ and $[O]$ terms, and that the $[D]$ terms are LU factorized along the implicit lines, and at each grid point, storing a separate unfactorized copy of these terms would negate most of the memory savings sought through the use of Eq. (39). Note that the LU factorized forms of the $[D]$ terms occupy the same memory space as the unfactored version of these terms, but enables the evaluation of the inverse matrix-vector product $x = [D]^{-1} f$ in a two-step forward and backwards substitution process as

$$y = [L]^{-1} f \qquad \text{(forward substitution)} \tag{40}$$

$$x = [U]^{-1} y \qquad \text{(backward substitution)} \tag{41}$$

This two-pass approach requires the same number of operations as multiplying the inverted matrix $[D]^{-1}$ with the vector f. Thus, similarly, the original matrix-vector product $g = [D]x$ can be also be evaluated in the same memory space and number of operations using the factorized form of the matrix as

$$y = [U]x \qquad \text{(forward substitution)} \tag{42}$$

$$g = [L]y \qquad \text{(back substitution)} \tag{43}$$

Thus, for the tangent model, the iterative solution scheme is constructed as

$$[L][U]\left(\frac{\partial \mathbf{w}^{n+1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{w}^{n}}{\partial \mathbf{D}}\right)^{k+1} = -\frac{\partial \mathbf{R}}{\partial \mathbf{D}} - [[L][U] + [O]]\frac{\partial \mathbf{w}^{n}}{\partial \mathbf{D}} - \left[\left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right]\left[\frac{\partial \mathbf{q}}{\partial \mathbf{w}} - [I]\alpha\right]\right]\frac{\partial \mathbf{w}^{n}}{\partial \mathbf{D}} - [O]\left(\frac{\partial \mathbf{w}^{n+1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{w}^{n}}{\partial \mathbf{D}}\right)^{k} \tag{44}$$

and for the adjoint problem, the transposed equivalent scheme becomes

$$[L^*][U^*](\mathbf{\Lambda}^{n+1} - \mathbf{\Lambda}^{n})^{k+1} = \frac{\partial \mathbf{L}^T}{\partial \mathbf{w}} - [[L^*][U^*] + [O]^T]\mathbf{\Lambda}^{n} - \left[\left[\frac{\partial \mathbf{q}^T}{\partial \mathbf{w}} - [I]\alpha\right]\left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right]^T\right]\mathbf{\Lambda}^{n} - [O]^T(\mathbf{\Lambda}^{n+1} - \mathbf{\Lambda}^{n})^{k} \tag{45}$$

where $[L^*]$ and $[U^*]$ correspond to the factorization of the transposed first-order Jacobian, and it is understood that the L and U matrices on the left-hand side are used to solve the linear system through forward and back substitution, as described in Eqs. (40) and (41).

In these formulations, the L, U, and O matrices are obtained for free on the right-hand side at each outer iteration, because they are already stored for the preconditioning matrix. The only additional matrices which are required to be either formed on the fly at each outer iteration, or stored throughout the solution process are the $[\frac{\partial q}{\partial w}]$ and $[\frac{\partial R}{\partial q}]$ matrices.

For an artificial dissipation scheme, the $[\frac{\partial q}{\partial w}]$ matrix represents the linearization of the undivided Laplacian, which is a trivial matrix in this case containing unity entries for each edge of the mesh. For MUSCL-type (monotone upstream centered schemes for conservation laws) reconstruction schemes, the matrix corresponds to the linearization of the gradient construction with respect to the flow variables, which corresponds to the weights used in the gradient reconstruction scheme, which are already stored by the discretization

scheme for the nonlinear flow equations. In either case, this matrix either need not be stored or is readily available. The $[\frac{\partial R}{\partial q}]$ matrix 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 [22]), and thus only half of the matrix needs to be stored. Furthermore, this is only a block 5×5 matrix, because no turbulence modeling quantities are involved in the q variables, and thus the additional storage required by this matrix is approximately 35% of the storage of the first-order accurate Jacobian or preconditioning matrix $[P]$. This is the only additional storage required for the solution of the tangent or adjoint flow models over and above a nearest-neighbor first-order accurate Jacobian, which is present in the flow solver in the first place. Although this additional matrix could easily be recomputed at each outer iteration in the defect-correction scheme, we choose to store it in this work due to the modest size of the matrix and the added CPU time benefits.

V. Mesh Deformation Tangent and Adjoint Problem

Step 2 in the tangent formulation and step 4 in the adjoint formulation both require the solution of the mesh motion or adjoint mesh motion problem given by Eqs. (18) and (25), respectively. In the current work, the mesh deformation process is governed by a system of spring-analogy equations, where each mesh edge is modeled as a spring with a stiffness inversely proportional to the edge length. The spring constants are based on the initial configuration of the mesh and are not updated as the mesh is deformed. Equation (13) is used to compute interior mesh point displacements, subject to given boundary mesh point displacements. These displacements resulting from the solution of Eq. (13) are then added to the mesh point coordinates to obtain the new deformed mesh configuration. Because the spring coefficients are always based on the initial mesh geometry, the governing equations are linear, and the tangent model coincides with the physical mesh motion model. Furthermore, the spring-analogy approach results in a nearest-neighbor stencil, such that the $[K]$ matrix can be represented as a block 3×3 matrix, using an edge-based data structure, similar to the first-order Jacobian for the flow solver described above. Because the entire matrix is stored, the adjoint problem is obtained by simply transposing the entire $[K]$ matrix.

The mesh motion (and tangent problem) and the adjoint problem are solved using a line-preconditioned agglomeration multigrid approach, similar to that described above for the flow equations, although a defect-correction scheme is not required, because the mesh motion equations are linear and operate on a nearest-neighbor stencil, and in this case, the preconditioner $[P]$ corresponds to the $[K]$ matrix. Splitting this matrix into LU factorized components and remaining terms denoted as $[O]$, the iterative solution scheme for the mesh motion or tangent problem can be written as

$$[L][U]\left(\frac{\partial \mathbf{x}_{\text{int}}}{\partial \mathbf{D}}\right)^{k+1} = \frac{\partial \mathbf{x}_{\text{surf}}}{\partial \mathbf{D}} - [O]\left(\frac{\partial \mathbf{x}_{\text{int}}}{\partial \mathbf{D}}\right)^{k} \tag{46}$$

while the corresponding adjoint mesh solution scheme can be written as

$$[L^*][U^*]\mathbf{\Lambda}_x^{k+1} = \frac{d\mathbf{L}}{d\mathbf{x}_{\text{int}}^*} - [O]^T\mathbf{\Lambda}_x^k \tag{47}$$

where, as previously, for the flow adjoint problem, the splitting

$$[K]^T = [L^*][U^*] + [O]^T \tag{48}$$

has been invoked. However, in this case, because a defect-correction scheme is not required, the right-hand sides of Eqs. (18) and (25), or the first terms on the right-hand sides of Eqs. (46) and (47) are held constant throughout the iterative solution process. Thus, these terms need only be computed once for each mesh tangent or adjoint solution, which corresponds to once for each function evaluation (or flow solution) during the optimization process.

VI. Matrix-Vector Product for Grid Sensitivities

The right-hand sides of the mesh tangent and adjoint problems are evaluated as matrix-vector products in step 3 for both the tangent and adjoint formulations. For the tangent problem, step 3 consists of evaluating the product

$$\frac{\partial \mathbf{R}}{\partial \mathbf{x}_{\text{int}}} \cdot \frac{\partial \mathbf{x}_{\text{int}}}{\partial \mathbf{D}} \tag{49}$$

whereas for the adjoint problem, the product

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{x}_{\text{int}}}\right)^T \mathbf{\Lambda}_{\mathbf{w}} \tag{50}$$

must be evaluated. The matrix $\partial \mathbf{R}/\partial \mathbf{x}_{\text{int}}$, which appears in both expressions, corresponds to the linearization of the flow residual with respect to the mesh point coordinates. This constitutes a sparse rectangular matrix which can have a fairly complicated structure. This is due to the fact that the mesh coordinates may appear both directly in the residual construction routines, as well as indirectly, through the dependence of other mesh metrics on the mesh point coordinates. The stencil of this matrix will thus not correspond to a nearest-neighbor stencil and cannot be supported on an edge-based data structure. However, because only the product of this matrix with a field vector is required in the tangent or adjoint model, and because this product need only be evaluated once for each function call (once per flow solution and mesh motion solution) before solving the mesh tangent or adjoint problem, this matrix-vector product is constructed in a multipass approach which computes the required terms. For a typical flow discretization where the residual depends on the mesh point coordinates both explicitly and implicitly through the use of precomputed mesh metrics, the above matrix-vector product may be evaluated as

$$\frac{d\mathbf{R}}{d\mathbf{x}} \cdot \delta\mathbf{x} = \frac{\partial \mathbf{R}}{\partial \mathbf{x}}\delta\mathbf{x} + \frac{\partial \mathbf{R}}{\partial \mathbf{fn}} \cdot \frac{\partial \mathbf{fn}}{\partial \mathbf{x}}\delta\mathbf{x} + \frac{\partial \mathbf{R}}{\partial \mathbf{vol}} \cdot \frac{\partial \mathbf{vol}}{\partial \mathbf{x}}\delta\mathbf{x} \tag{51}$$

where **fn** represents the vector of control-volume face normals over the mesh, and **vol** represents the vector of mesh control volumes. Note that Eq. (51) is used as an illustrative example and is not meant to precisely describe all dependencies of the current flow discretization on the mesh point coordinates. (For example, variations in the distance function used for the single equation turbulence model are not taken into account here.) The evaluation of this matrix-vector product is performed in a multistep procedure, given as

$$\delta\mathbf{fn} = \frac{\partial \mathbf{fn}}{\partial \mathbf{x}}\delta\mathbf{x} \tag{52}$$

$$\delta\mathbf{vol} = \frac{\partial \mathbf{vol}}{\partial \mathbf{x}}\delta\mathbf{x} \tag{53}$$

$$\frac{d\mathbf{R}}{d\mathbf{x}}\delta\mathbf{x} = \frac{\partial \mathbf{R}}{\partial \mathbf{x}}\delta\mathbf{x} + \frac{\partial \mathbf{R}}{\partial \mathbf{fn}}\delta\mathbf{fn} + \frac{\partial \mathbf{R}}{\partial \mathbf{vol}}\delta\mathbf{vol} \tag{54}$$

Note that the first two steps involve matrix-vector products based on a linearization of the mesh metric routines, while the last step involves the linearization of the flow residual with respect to the mesh metrics and coordinates, which appear directly in these routines.

In the case of the adjoint model, the matrix-vector product of Eq. (51) can be evaluated in a similar fashion. Taking the transpose of Eq. (52), we obtain the relation

$$\frac{d\mathbf{R}^T}{d\mathbf{x}}\mathbf{\Lambda}_w = \frac{\partial \mathbf{R}^T}{\partial \mathbf{x}}\mathbf{\Lambda}_w + \frac{\partial \mathbf{fn}^T}{\partial \mathbf{x}}\frac{\partial \mathbf{R}^T}{\partial \mathbf{fn}}\mathbf{\Lambda}_w + \frac{\partial \mathbf{vol}^T}{\partial \mathbf{x}}\frac{\partial \mathbf{R}^T}{\partial \mathbf{vol}}\mathbf{\Lambda}_w \tag{55}$$

which can be evaluated in a multistep procedure as

$$\delta \mathbf{fn} = \frac{\partial \mathbf{R}^T}{\partial \mathbf{fn}} \mathbf{\Lambda}_w \tag{56}$$

$$\delta \mathbf{vol} = \frac{\partial \mathbf{R}^T}{\partial \mathbf{vol}} \mathbf{\Lambda}_w \tag{57}$$

$$\frac{d\mathbf{R}^T}{d\mathbf{x}} \mathbf{\Lambda}_w = \frac{\partial \mathbf{R}^T}{\partial \mathbf{x}} \mathbf{\Lambda}_w + \frac{\partial \mathbf{fn}^T}{\partial \mathbf{x}} \delta \mathbf{fn} + \frac{\partial \mathbf{vol}^T}{\partial \mathbf{x}} \delta \mathbf{vol} \tag{58}$$

As in the previous case, a (transposed) linearization of each individual routine is invoked, but in the reverse order of that used by the original discretization in the tangent model. For example, the first two steps of Eqs. (56) and (57) involve the linearization of the flow residual with respect to the mesh metrics, while the last step [cf. Eq. (58)] involves the linearization of the mesh metrics with respect to the mesh point coordinates.

In both cases, the matrix-vector product involving the matrix $\frac{\partial \mathbf{R}}{\partial \mathbf{x}}$, which may involve a complex dependence of the residual on the mesh point coordinates, can be built up systematically by linearizing each component routine in the residual construction process, and other routines upon which these depend, and then concatenating the results by calling the appropriate routines in the proper order.

VII. Other Required Sensitivities

For the adjoint problem, the construction of the vector $\partial \mathbf{L}^T / \partial \mathbf{x}_{\text{int}}$ is required explicitly. Considering, for example, an objective function based on a pressure force coefficient such as pressure drag, it should be evident that the final form of the above vector is nonzero only at the surface grid points (where pressure forces are integrated), and thus storage of this vector is not in itself problematic. However, the construction of this vector must take into account the fact that surface forces are based on surface element face normals, which in turn depend on the surface mesh point coordinates through the surface element face normals. Thus the evaluation of this vector should be done in the above described modular fashion as

$$\frac{\partial \mathbf{L}}{\partial \mathbf{x}_{\text{int}}} = \frac{\partial \mathbf{L}}{\partial \mathbf{fn}} \frac{\partial \mathbf{fn}}{\partial \mathbf{x}} \tag{59}$$

Because the resulting vector must be obtained explicitly (as opposed to its inner product with another vector as in the previous cases), each of the expressions on the right-hand side of Eq. (59) must be evaluated explicitly as well. Although this is straightforward for the first term, the second term is slightly more complicated due to the fact that the face normal area associated with each grid point control volume is assembled from fractions of surrounding cell face normals, which in turn depend on the cell mesh point coordinates, as depicted in Fig. 2. Although these dependencies can be computed, additional data structures would be required for the storage of the $\frac{\partial \mathbf{fn}}{\partial \mathbf{x}}$ terms.

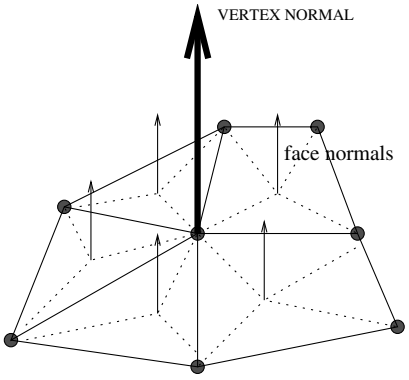


Fig. 2 Illustration of the construction of a surface vertex normal based on surrounding surface face normals and the resulting stencil of surface mesh points which are used in this operation for mixed triangular quadrilateral surface meshes.

An alternative approach is to transpose Eq. (59) as

$$\frac{\partial \mathbf{L}^T}{\partial \mathbf{x}_{\text{int}}} = \frac{\partial \mathbf{fn}^T}{\partial \mathbf{x}} \frac{\partial \mathbf{L}^T}{\partial \mathbf{fn}} \tag{60}$$

In this manner, the required vector can be evaluated as a matrix-vector product. The vector $\partial \mathbf{L}^T / \partial \mathbf{fn}$ has a simple structure and is easily computed and stored in a first step. The matrix-vector product is then built up systematically by forming the equivalent adjoint routine to each phase in the mesh metric routines which compute surface vertex normals. These routines are then called in the appropriate (reverse) order to obtain the final vector $\partial \mathbf{L} / \partial \mathbf{x}_{\text{int}}$ using only the same boundary face data structures used in the mesh metric routines. Note that the construction of the above terms is considerably simplified in the event that a boundary face-based loop is used to integrate the force coefficients directly. However, the goal of the current approach is to construct linearizations of the exact procedures which already exist in the analysis code without requiring modifications to the analysis code itself.

VIII. Implementation Details

The basic strategy of this paper is to construct a complete tangent and adjoint model for the entire shape optimization problem in a modular fashion, which mimics the implementation of the original physical simulation including the flow solution and mesh motion.

Considering each individual routine required for the construction of the simulation as a function with an input vector x, and an output vector $f(x)$ (which may have different dimensions than the input vector), an equivalent tangent routine is constructed which, given an input vector δx, produces the output vector $\frac{\partial f}{\partial x} \delta x$, and an equivalent adjoint routine is also constructed which, given an input vector δf produces the output $\frac{\partial f^T}{\partial x} \delta f$. For both tangent and adjoint models, the loop structure and data access patterns of the new routines mimic those of the original $f(x)$ routine, and thus no additional data structures or storage are required, and the run-time expense of all three versions of each function are closely related.

The tangent routine is generally constructed first, because its relation to the original simulation routine is more intuitive than the adjoint process, especially concerning the implementation of boundary conditions. Furthermore, the correctness of the tangent routines can be verified by finite differencing the analysis routines. Alternatively, the complex variable approach provides a more accurate technique for verifying these linearizations, because it is not prone to roundoff error as is the finite difference approach [23]. Once a particular tangent routine has been implemented and verified, the corresponding adjoint routine is implemented. The correctness of the adjoint routine is then verified using the duality principle. Consider an analysis routine which, given an input vector x, produces an output vector $f(x)$. The corresponding tangent and adjoint routines can be written as

$$\delta f_1 = \frac{\partial f}{\partial x} \delta x_1 \qquad \text{(tangent routine)} \tag{61}$$

$$\delta x_2 = \frac{\partial f^T}{\partial x} \delta f_2 \text{(adjoint routine)} \tag{62}$$

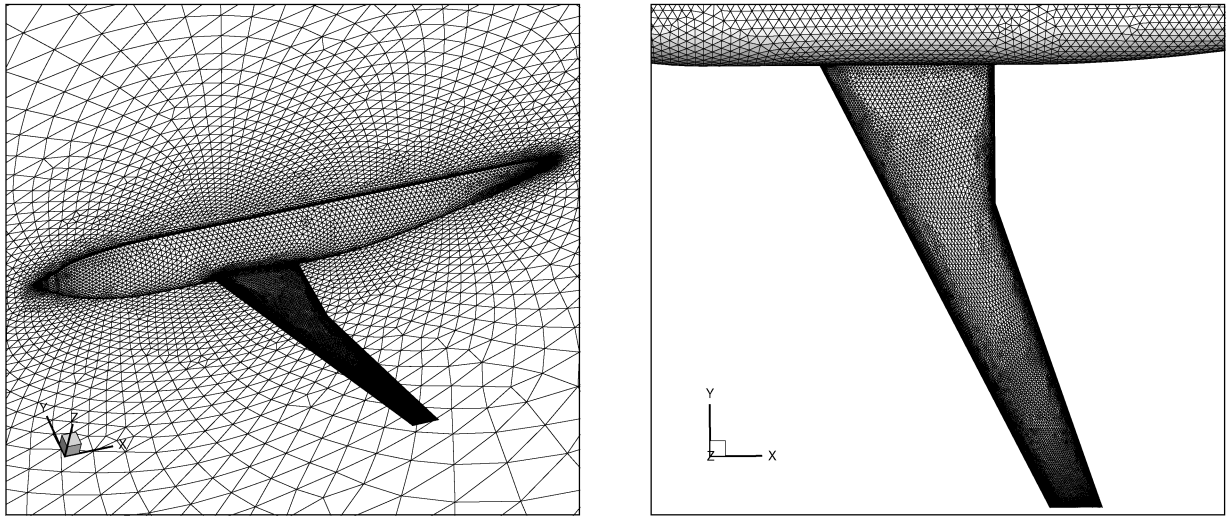
Inverting the second (adjoint) equation above and taking the transpose yields

$$\delta f_2^T = \delta x_2^T \frac{\partial f^{-1}}{\partial x} \tag{63}$$

Combining this equation with the above tangent equation to form the dot product $\delta f_2^T \cdot \delta f_1$ yields the following relation:

$$\delta f_2^T \cdot \delta f_1 = \delta x_2^T \cdot \delta x_1 \tag{64}$$

which is known as the duality relation. The duality principle is well known and has seen widespread use in optimization procedures. For example, duality has been invoked to design efficient solvers for the



a) b)

Fig. 3 Illustration of a DLR-F6 wing body configuration and hybrid unstructured mesh for a wing design optimization problem.

adjoint flow and mesh motion problems in previous work [6,19]. However, the above statement is the most general form of the duality principle and states that given any arbitrary input vectors δx_1 and δf_2, the relation in Eq. (64) should hold. Because this is a necessary test for correctness, but not a sufficient test, we make use of sequences of arbitrary input vectors to test for correctness of our adjoint routine implementations, rather than simply testing these routines in the context of a practical analysis run. For steps 2 and 4 in Sec. III where the tangent and adjoint models involve the solution of system of equations, such as

$$[K]x = f \qquad \text{(tangent model)} \tag{65}$$

$$[K]^T y = g \qquad \text{(adjoint model)} \tag{66}$$

the following duality principle must be verified:

$$x^T g = y^T f \tag{67}$$

given the input vectors f and g, and the final converged solution vectors x and y. This duality principle must hold exactly for the final converged solutions and may be enforced as well at each cycle in the iterative solution process, as advocated in [10,19], although full duality throughout the iterative solution process is usually not strictly enforced in the current work, for practical reasons.

Once the individual tangent and adjoint routines have been implemented and verified, larger simulation components are built up by calling the relevant routines in the appropriate order (in reverse order for the adjoint problem), and checking the results by finite difference and the duality principle at each stage. This is repeated until the full sequence from design variable input to objective function calculation is verified by finite difference and demonstrated to be duality preserving.

IX. Results

The methodology described previously has been used to drive an optimization problem for an aircraft wing design problem, using the Reynolds averaged Navier–Stokes equations with the Spalart–Allmaras turbulence model [24] to simulate the viscous turbulent flow over the aircraft wing body geometry. 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

$$L = (C_L - C_{L_{\text{target}}})^2 + 10(C_D)^2 \tag{68}$$

where $C_{L_{\text{target}}}$ represents the constant lift coefficient value to be maintained, and the weighting factor (10) is used to balance the different magnitudes of lift and drag coefficients. For simplicity, the objective function only considers the pressure lift and drag components. The design variables consist of the normal displacements of each surface grid point on the wing surface between approximately the 10% and 90% span locations and from the leading edge to the 80% chord location within this span region on both the upper and lower surfaces. This choice of design variables results in a simple block diagonal form of the $\partial \mathbf{x}_{\text{surf}}/\partial \mathbf{D}$ sensitivity matrix, where the diagonal elements correspond to the direction cosines of the surface normal for each point. 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

$$\delta \mathbf{D} = -\lambda \frac{d\tilde{\mathbf{L}}}{d\mathbf{D}} \tag{69}$$

where λ represents a small time step, chosen small enough to ensure convergence of the optimization procedure, and $\frac{d\tilde{L}}{dD}$ represents the smoothed gradients $\frac{dL}{dD}$, obtained using an implicit smoothing technique, which is necessary to ensure smooth design shapes, as described in [2]. In actual fact, it is the $\mathbf{\Lambda}_{\mathbf{x}} = \partial \mathbf{L}/\partial \mathbf{x}_{\text{surf}}$ sensitivities

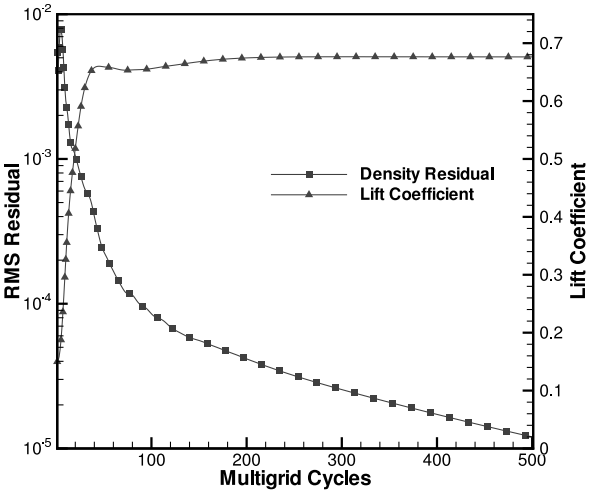
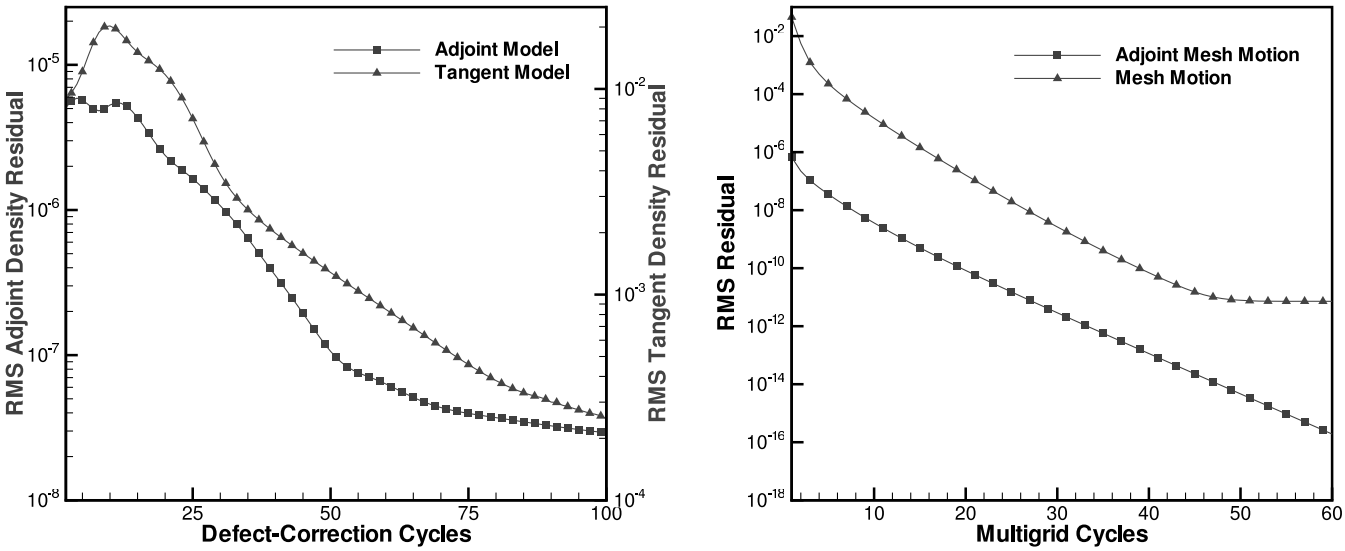


Fig. 4 Convergence of the flow analysis problem using a nonlinear multigrid solver as measured by the density residual and lift coefficient in terms of the number of multigrid cycles.



a) b)

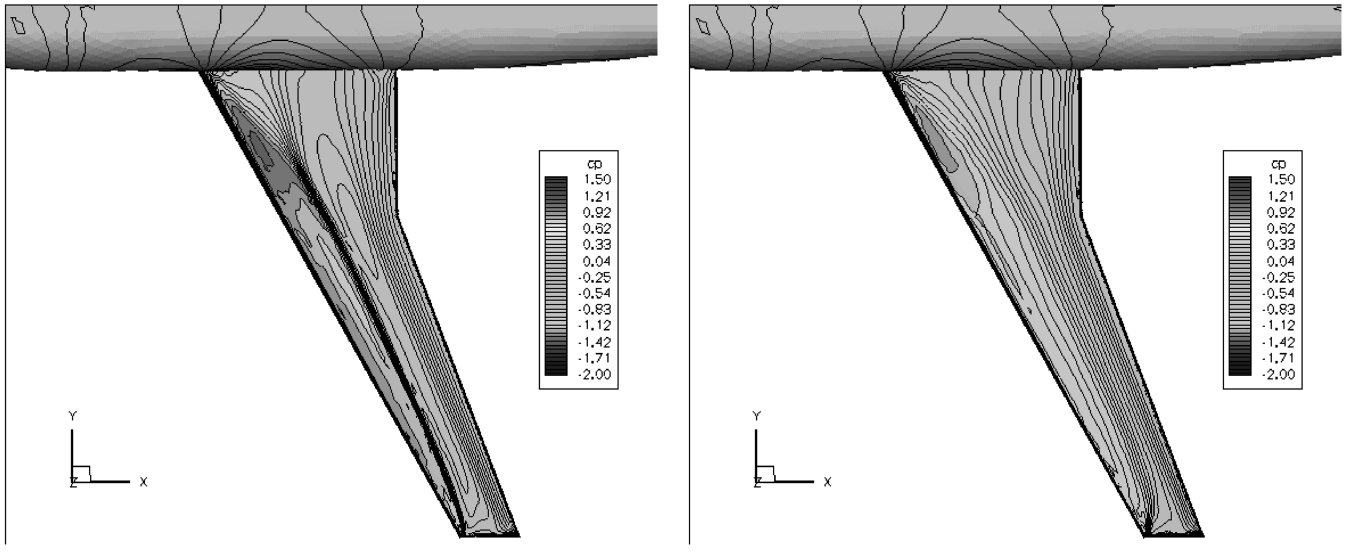
Fig. 5 a) Convergence of flow adjoint and corresponding flow tangent problems using four linear multigrid cycles for each defect-correction cycle in terms of the number of (outer) defect-correction cycles. b) Convergence of mesh motion and corresponding mesh adjoint problem in terms of the number of multigrid cycles for a wing body configuration.

which are smoothed, before the evaluation of Eq. (26), because the smoothing must take place in geometric space rather than in design space. The current implementation of the steepest descent optimization procedure is not optimal, in that λ is determined empirically, but is sufficient for demonstrating the utility of the adjoint solution techniques described herein.

The initial geometry and computational mesh are depicted in Fig. 3. The geometry corresponds to the DLR-F6 wing body configuration which constituted one of the test cases for the recent AIAA Drag Prediction Workshop series [25,26]. The unstructured mesh was generated using the VGRID grid generation program [27] and was originally supplied as one of the standard grids for the workshop. This grid contains a total of 1.12×10^6 points or 4.2×10^6 cells and contains principally prismatic elements in the boundary-layer regions and tetrahedral elements in the inviscid regions of flow away from the aircraft surfaces. The Mach number for this case is $M = 0.75$, the Reynolds number based on the mean aerodynamic chord is 3×10^6, and the incidence is 1 deg. The computed lift and drag coefficients at these conditions are $C_L = 0.6767$ and $C_D = 0.0302$, respectively, and a relatively strong shock is observed on the upper surface of the wing.

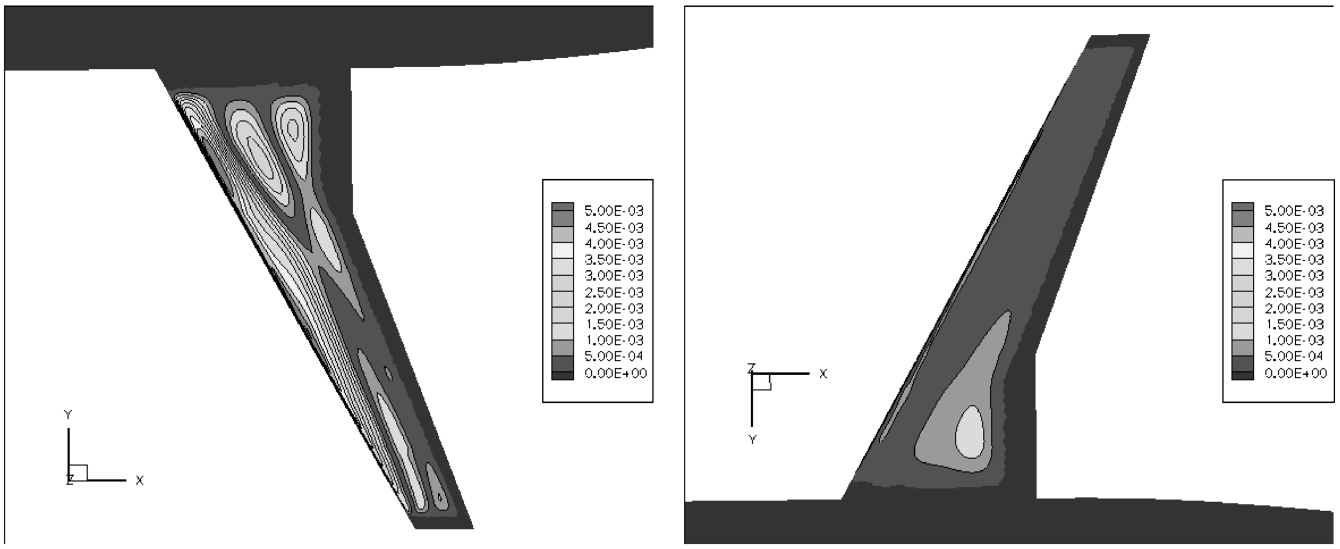
The convergence of the flow solver for this case is shown in Fig. 4, using the nonlinear agglomeration multigrid solver with four grid levels and implicit line preconditioning on each level as described previously. The residuals are reduced by 3 orders of magnitude over 500 multigrid cycles, and the lift coefficient is seen to attain a constant value well within the 500 cycle limit. This calculation is performed on a cluster of PCs using 16 CPUs, and requires approximately 40 minutes of wall clock time.

Figure 5a illustrates the convergence of the flow adjoint problem at a particular design iteration during the optimization procedure. This is compared in the same figure with the convergence of the corresponding tangent flow problem. The defect-correction scheme is used to converge both problems, using the linear agglomeration multigrid scheme with four levels and line-implicit preconditioning on each level to drive the inner problem of the defect-correction



a) b)

Fig. 6 a) Surface pressure coefficient contours on an initial DLR-F6 configuration at Mach = 0.75 and 1 deg incidence showing the upper surface shock wave. b) Surface pressure coefficient contours on a DLR-F6 configuration after 15 design cycles at Mach = 0.75 and 1 deg incidence showing reduced strength of the upper surface shock wave.



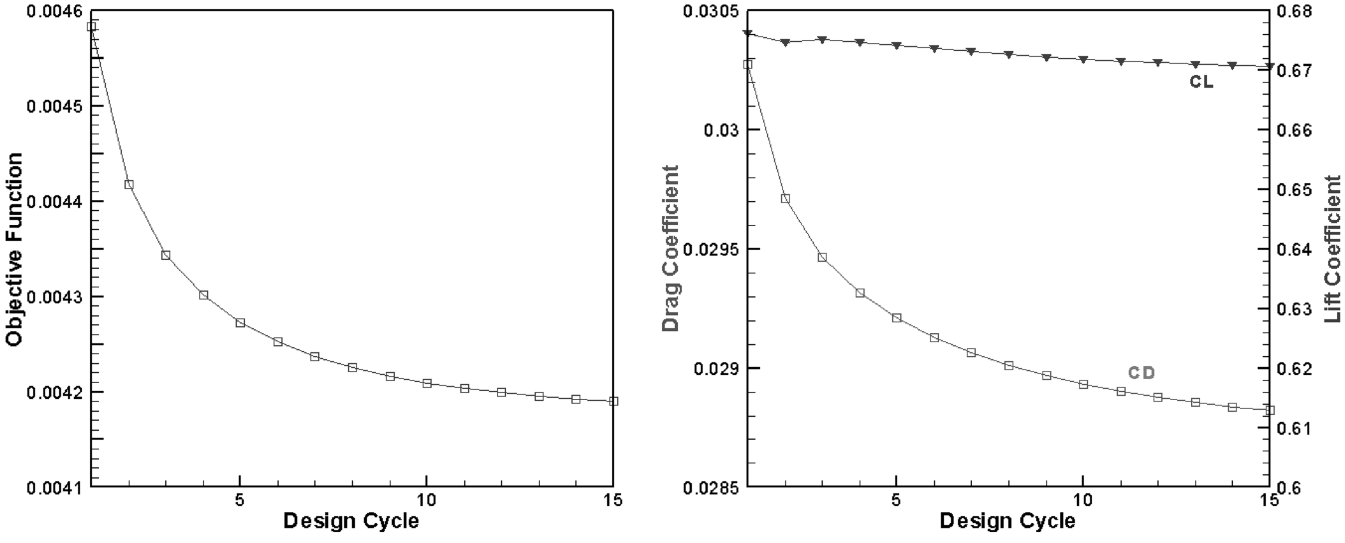
a) b)

Fig. 7 Surface normal displacements or design variable magnitudes on top a) and bottom b) of a final redesigned DLR-F6 configuration.

scheme. A total of four linear multigrid cycles are used within each defect-correction cycle, and the convergence is plotted in terms of the number of (outer) defect-correction cycles. Similar convergence rates are observed for both problems, resulting in a decrease of the residuals by just under 3 orders of magnitude over 100 defect-correction cycles, which corresponds to 400 linear multigrid cycles. Note that in this particular implementation, the two methods do not preserve exact duality at each iteration, due to practical considerations [i.e., we use a multigrid W(1,0) cycle in both tangent and adjoint problems], although the final converged results preserve exact duality.

Figure 5b depicts the convergence of the mesh motion and mesh adjoint equations at a particular cycle in the design procedure. The same solver (line preconditioned four-level agglomeration multigrid) developed for the flow and flow adjoint equations is used to solve these two problems as well. The mesh motion equations obtained using the spring-analogy approach correspond to a set of three uncoupled Poisson equations and therefore converge very rapidly with the multigrid algorithm, reaching machine precision in approximately 50 multigrid cycles. This rapid convergence, combined with the smaller block size of the mesh motion equations (3×3 blocks versus 6×6 blocks for the flow and turbulence equations), ensures that the solution of the mesh motion and adjoint equations comprises a small fraction (less than 10%) of the overall solution time within a design optimization run.

Figure 6 illustrates the flow solution in terms of pressure contours on the wing upper surface for the initial and modified geometry of the final design, showing a considerably reduced shock strength compared to the initial solution. This suggests most of the drag reduction has come from the effect of wave drag. The normal displacements of the surface points on the wing (i.e., the values of the design variables) are plotted in Fig. 7, showing a relatively smooth deformation near the leading edge and mostly on the upper surface of the wing. Figure 8 depicts the convergence of the optimization procedure, in terms of the objective function and in terms of the computed lift and drag coefficients over a total of 15 design cycles. The objective function is seen to decrease monotonically at each design cycle. The lift coefficient is held relatively constant, whereas the drag coefficient is reduced from $C_D = 0.0302$ to $C_D = 0.0288$, for a total drag reduction of 14 counts. The complete sequence of 15 design iterations required approximately 6 hours of wall clock time running on 16 CPUs of the PC cluster.



a) b)

Fig. 8 a) Convergence of objective function with a number of design cycles, and b) evolution of the lift and drag coefficients as a function of design cycles.

X. Conclusions

The discrete adjoint approach developed in this work results in an efficient scheme for computing sensitivity derivatives for large numbers of design variables. The procedure requires minimal memory overheads over and above those already required by an implicit flow solver and retains the efficiency of the underlying flow and mesh motion solvers by using the same solvers for the corresponding adjoint problems. The notion of applying the adjoint technique to the entire optimization process, rather than only to the flow solution process, and the development of a systematic procedure for implementing this approach, results in a verifiable correct scheme, which should also be extendable to more complicated simulations such as unsteady simulations and coupled multidisciplinary problems. In addition to these areas, future work will also concentrate on the development of more effective optimization strategies and more sophisticated techniques for defining design parameters and coupling these to the proposed framework.

References

[1] Jameson, A., "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.

[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, 1998.

[3] Nielsen, E. J., and Anderson, W. K., "Recent Improvements in Aerodynamic Optimization on Unstructured Meshes," *AIAA Journal*, Vol. 40, No. 6, June 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] Soto, R. L. O., and Yang, C., "An Adjoint-Based Design Methodology for CFD Optimization Problems," AIAA Paper 2003-0299, 2003.

[6] Giles, M. B., Duta, M. C., Muller, J. D., and Pierce, N. A., "Algorithm Developments for Discrete Adjoint Methods," *AIAA Journal*, Vol. 41, No. 2, Feb. 2003, pp. 198–205.

[7] Li, S., and Petzold, L., "Adjoint Sensitivity Analysis for Time-Dependent Partial Differential Equations with Adaptive Mesh Refinement," *Journal of Computational Physics*, Vol. 198, No. 1, 2004, pp. 310–325.

[8] Thomas, J. P., Hall, K. C., and Dowell, E. H., "Discrete Adjoint Approach for Modeling Unsteady Aerodynamic Design Sensitivities," *AIAA Journal*, Vol. 43, No. 9, 2005, pp. 1931–1936.

[9] Nielsen, E. J., and Park, M., "Using an Adjoint Approach to Eliminate Mesh Sensitivities in Computational Design," *AIAA Journal*, Vol. 44, No. 5, May 2006, pp. 948–953.

[10] Mavriplis, D. J., "Formulation and Solution of the Discrete Adjoint for Optimization Problems on Unstructured Meshes," *AIAA Journal*, Vol. 44, No. 1, Jan. 2006, pp. 42–50.

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

[12] Mavriplis, D. J., and Pirzadeh, S., "Large-Scale Parallel Unstructured Mesh Computations for 3D High-Lift Analysis," *Journal of Aircraft*, Vol. 36, No. 6, Dec. 1999, pp. 987–998.

[13] Bischof, C., Carle, A., Khademi, P., and Mauer, A., "The ADIFOR 2.0 System for the Automatic Differentiation of Fortran 77 Programs," Rice University, TR CRPC-TR94491, Center for Research on Parallel Computation, 1994.

[14] Batina, J. T., "Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes," *AIAA Journal*, Vol. 28, No. 8, 1990, pp. 1381–1288.

[15] Venkatakrishnan, V., and Mavriplis, D. J., "Implicit Method for the Computation of Unsteady Flows on Unstructured Grids," *Journal of Computational Physics*, Vol. 127, No. 2, Sept. 1996, pp. 380–397.

[16] Degand, C., and Farhat, C., "A Three-Dimensional Torsional Spring Analogy Method for Unstructured Dynamic Meshes," *Computers and Structures*, Vol. 80, Nos. 3–4, Feb. 2002, pp. 305–316.

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

[18] 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, 2005.

[19] Nielsen, E. J., Lu, J., Park, M. A., and Darmofal, D. L., "An Implicit Exact Dual Adjoint Solution Method for Turbulent Flows on Unstructured Grids," *Computers and Fluids*, Vol. 33, No. 9, Nov. 2004, pp. 1131–1155.

[20] 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.

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

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

[23] Nielsen, E. J., and Kleb, W., "Efficient Construction of Discrete Adjoint Operators on Unstructured Grids Using Complex Variables," *AIAA Journal*, Vol. 44, No. 4, April 2006, pp. 827–836.

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

[25] Laflin, K., Brodersen, O., Rakowitz, M., Vassberg, J., Wahls, R., and Morrison, J., "Summary of Data from the Second AIAA CFD Drag Prediction Workshop," AIAA Paper 2004-0555, 2004.

[26] Laflin, K. R., Klausmeyer, S. M., Zickuhr, T., Vassberg, J. C., Wahls, R. A., Morrison, J. H., Brodersen, O. P., Rakowitz, M. E., Tinoco, E. N., and Godard, J.-L., "Data Summary from the Second AIAA Computational Fluid Dynamics Drag Prediction Workshop," *Journal of Aircraft*, Vol. 42, No. 5, 2005, pp. 1165–1178.

[27] Pirzadeh, S., "Three-Dimensional Unstructured Viscous Grids by the Advancing-Layers Method," *AIAA Journal*, Vol. 34, No. 1, 1996, pp. 43–49.

N. Alexandrov
Associate Editor