AIAA JOURNAL
Vol. 41, No. 2, February 2003

Algorithm Developments for Discrete Adjoint Methods

Michael B. Giles* and Mihai C. Duta†
Oxford University, Oxford, England OX1 3QD, United Kingdom
Jens-Dominik Müller‡
Queen's University, Belfast, Northern Ireland BT9 5AG, United Kingdom
and
Niles A. Pierce§
California Institute of Technology, Pasadena, California 91125

A number of algorithm developments are presented for adjoint methods using the "discrete" approach in which the discretization of the nonlinear equations is linearized and the resulting matrix is then transposed. With a new iterative procedure for solving the adjoint equations, exact numerical equivalence is maintained between the linear and adjoint discretizations. The incorporation of strong boundary conditions within the discrete approach is discussed, and difficulties associated with the use of linear perturbation and adjoint methods for applications with strong shocks are also examined.

Nomenclature

B	=	boundary condition projection operator
$\mathcal{I}(z)$	=	imaginary part of complex variable z
i	=	$\sqrt{-1}$
J	=	nonlinear output functional, for example, lift
$\tilde{J}$	=	linear perturbation to functional
L	=	linearized discrete operator
$\boldsymbol{R}$	=	nonlinear discrete residual operator
$\boldsymbol{U}$	=	nonlinear variables
$\boldsymbol{u}$	=	linear perturbation variables
v	=	adjoint variables
α	=	design/sensitivity variable

Subscripts

$\perp$	=	orthogonal to boundary conditions
$\|$	=	parallel to boundary conditions

Superscripts

(m)	=	subiteration index
n	=	iteration index
T	=	transpose

Introduction

THERE is a long history of the use of adjoint equations in optimal control theory.[1] In fluid dynamics, the first use of adjoint equations for design was by Pironneau,[2] but within the field of aeronautical computational fluid dynamics (CFD), the use of adjoint equations for design optimization has been pioneered by Jameson[3,4] and Jameson et al.[5] for the potential flow, Euler, and Navier–Stokes equations. The complexity of the applications within these papers has also progressed from two-dimensional airfoil optimization, to three-dimensional wing design, and finally to complete aircraft configurations.[6–8] A number of other research groups have also developed adjoint CFD codes[9–13] using the same "continuous" approach in which the first step is to linearize the original partial differential equations. The adjoint partial differential equation (PDE) and appropriate boundary conditions are then formulated, and finally the equations are discretized. Although this minimizes the memory requirements and the CPU cost per iteration, it requires one to develop an appropriate iterative solution procedure, and this may not give as good a convergence rate as the original nonlinear code. In addition, the debugging and validation of the adjoint code is complicated by the lack of a suite of benchmark test cases.

The alternative "discrete" approach, which we use, takes a discretization of the Navier–Stokes equations, linearizes the discrete equations, and then uses the transpose of the linear operator to form the adjoint problem. This approach has been developed by Elliott and Peraire,[14] Elliott,[15] Nielsen and Anderson,[16] Anderson and Bonhaus,[17] Mohammadi and Pironneau,[18] and Kim et al.[19] The main advantage of this approach, in our opinion, is that the code development becomes a more straightforward process. The linearization of the nonlinear discrete equations can be performed either manually or by automatic differentiation software, and the linear code can be validated by direct comparison with the nonlinear code. Similarly, because the adjoint code is obtained by transposing the linear operator, it must yield exactly the same values for the linearized objective function and so can be validated against the linear code. For an excellent review of research on both continuous and discrete adjoint design methods, see the paper by Newman et al.[20]

In this paper we contribute to the development and understanding of discrete adjoint methods in four respects:

1) We discuss the implementation of the adjoint code in a way that minimizes the memory and CPU requirements and that can be automated using automatic differentiation tools.

2) We develop an adjoint multigrid iteration procedure with preconditioned time stepping that maintains exact equivalence between the linear and adjoint codes at all times during the evolution of their respective solutions.

3) We present a detailed discussion of the imposition of strong boundary conditions and the inclusion of viscous stresses in objective functions and the consequence for the formulation of the adjoint code.

4) We present a numerical investigation indicating the potential for problems with strong shocks.

This research forms part of the development of the HYDRA suite of codes. The foundation is a nonlinear code that approximates the Reynolds averaged Navier–Stokes equations on unstructured hybrid grids, using an edge-based discretization. The solution procedure uses Runge–Kutta time-marching accelerated by Jacobi

Received 23 August 2001; revision received 27 September 2002; accepted for publication 27 September 2002.

*Professor, Oxford University Computing Laboratory; giles@comlab.ox.ac.uk. Member AIAA.

†Research Officer, Oxford University Computing Laboratory; mcd@comlab.ox.ac.uk.

‡Lecturer, School of Aeronautical Engineering; j.mueller@qub.ac.uk. Member AIAA.

§Assistant Professor, Applied and Computational Mathematics; niles@caltech.edu. Member AIAA.

preconditioning and multigrid (see Ref. 21), with dual time stepping for unsteady flows.

The second code in the suite is the steady adjoint code, which is based on a linearization of the flow equations around the nonlinear steady-state flow conditions. It is the development of this code that is the primary subject of this paper.

The third code is for the linear analysis of unsteady flows. This is also based on a linearization of the unsteady flow equations around the steady-state flow conditions calculated by the nonlinear code. Because of linearity, unsteady periodic flows can be decomposed into a sum of harmonic terms, each of which can be computed independently. Thus, the linear harmonic code considers just one particular frequency of unsteadiness, resulting in a formulation in which the purpose is to compute a complex flow solution that represents the amplitude and phase of the unsteady flow.

The fourth code, which is an extension of the second, is the adjoint counterpart of the linear harmonic code. The development and application of these harmonic analysis and adjoint codes is discussed elsewhere,[22–24] but in this paper the harmonic analysis code is used with zero frequency to obtain steady linearized flow results.

Discrete Adjoint Formulation

We start by considering the discrete nonlinear Euler equations with a weak imposition of boundary conditions on solid walls through the specification of zero mass flux through faces on the surface. If the far-field boundary conditions are also imposed through far-field fluxes, then the discrete system of equations that is solved is of the form

$$\boldsymbol{R}(\boldsymbol{U}, \alpha) = 0$$

where $\boldsymbol{U}$ is the vector of flowfield variables, α represents one or more design variables that control the geometry of the airfoil or wing (and, hence, the grid coordinates), and $\boldsymbol{R}(\boldsymbol{U})$ represents the discrete flux residuals that are driven to zero by the iterative solution process.

If there is just one design variable, then linearizing the steady-state equations with respect to a change in that design variable yields

$$L\boldsymbol{u} = \boldsymbol{f}$$

where

$$L \equiv \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}, \qquad \boldsymbol{u} \equiv \frac{\mathrm{d}\boldsymbol{U}}{\mathrm{d}\alpha}, \qquad \boldsymbol{f} \equiv -\frac{\partial \boldsymbol{R}}{\partial \alpha}$$

The corresponding perturbation in a nonlinear objective function $J(\boldsymbol{U}, \alpha)$ is

$$\tilde{J} = \boldsymbol{g}^T \boldsymbol{u} + \frac{\partial J}{\partial \alpha}$$

where

$$\boldsymbol{g}^T \equiv \frac{\partial J}{\partial \boldsymbol{U}}$$

In the adjoint approach, this same quantity can be obtained by evaluating

$$\tilde{J} = \boldsymbol{v}^T \boldsymbol{f} + \frac{\partial J}{\partial \alpha}$$

where the adjoint solution $\boldsymbol{v}$ satisfies the equation

$$L^T \boldsymbol{v} = \boldsymbol{g}$$

The equivalence of this formulation comes from the identity

$$\boldsymbol{v}^T \boldsymbol{f} = \boldsymbol{v}^T L\boldsymbol{u} = (L^T \boldsymbol{v})^T \boldsymbol{u} = \boldsymbol{g}^T \boldsymbol{u}$$

If there are many design variables (each giving rise to a different vector $\boldsymbol{f}$) and only one objective (yielding a single vector $\boldsymbol{g}$), then the benefit of the adjoint approach is that the objective sensitivity $\tilde{J}$ can be obtained following a single evaluation of $\boldsymbol{v}$ instead of separate evaluations of $\boldsymbol{u}$ for each $\boldsymbol{f}$.

Implementation of Adjoint Discretization

In the implementation, the linear operator L is split into two parts,

$$L\boldsymbol{u} = C\boldsymbol{u} + D\boldsymbol{u} \tag{1}$$

The first part represents the convective fluxes due to a Galerkin finite element discretization. The second part represents the smoothing fluxes (to which the viscous fluxes are added later for the Navier–Stokes equations). The operator D can be further broken down into the product of two operators,

$$D\boldsymbol{u} = VG\boldsymbol{u}$$

where G computes the gradient and a pseudo-Laplacian of $\boldsymbol{u}$ at each node, in addition to $\boldsymbol{u}$ itself.

The corresponding adjoint operator is

$$L^T \boldsymbol{v} = C^T \boldsymbol{v} + D^T \boldsymbol{v}$$

with

$$D^T \boldsymbol{v} = G^T V^T \boldsymbol{v}$$

indicating that the adjoint gradient routine is applied after the adjoint smoothing routine, which at first seems counterintuitive.

At an even more detailed level, the action of each of the operators C, V, and G is computed by a loop over all edges in the unstructured grid. Therefore, taking $C\boldsymbol{u}$ as an example, we can express it as a sum of elemental edge matrices whose only nonzero entries corresponds to the two nodes at either end of the edge,

$$C\boldsymbol{u} = \sum_e C_e \boldsymbol{u}$$

The adjoint version of this is simply

$$C^T \boldsymbol{v} = \sum_e C_e^T \boldsymbol{v}$$

corresponding to a similar loop over all edges.

For the convective fluxes, it is easy to compute the edge product $C_e^T \boldsymbol{v}$ directly without explicitly forming the matrix C_e. The transposed gradient operator G^T is also easily formulated. The product $V^T \boldsymbol{v}$ presents greater difficulties. Elliott and Peraire[14] and Elliott[15] precomputed and stored the nonzero entries in the elemental matrices V_e and then evaluated the matrix–vector products $V_e^T \boldsymbol{v}$. However, the storage of these matrices for each edge requires a substantial amount of memory. Anderson and Bonhaus[17] avoided the memory cost by recomputing the matrices during each iteration, but this greatly increases the CPU cost.

To minimize both the memory and CPU requirements, it is necessary to calculate the edge product $V_e^T \boldsymbol{v}$ directly, as with $C_e^T \boldsymbol{v}$. The difficulty is in working out how best to do this. One approach is to use automatic differentiation (AD) software such as Odyssée,[25] ADIFOR,[26,27] or TAMC.[28] In forward mode, AD software takes the original nonlinear code and then uses the basic rules of linearization to construct the code to evaluate $V_e \boldsymbol{u}$. In reverse mode, it produces the code to calculate $V_e^T \boldsymbol{v}$; it may seem that this is a much harder task, but in fact it is not. Furthermore, there are theoretical results that guarantee that the number of floating point operations is no more than three times that of the original nonlinear code.[29]

Mohammadi and Pironneau used Odyssée to generate much of his adjoint code,[18] but a lot of hand-coding was still required. In our work, we have written the adjoint code manually, but following many of the techniques of AD. To simplify the expressions for the partial derivatives, we chose to use the primitive variables (density, velocity, and pressure) as our working variables, rather than the usual conservative variables. The equations are still in conservative form so that this choice of working variables has no effect on the final solution.

The memory requirements for the adjoint code are 20–30% greater than for the nonlinear code and depend on the grid that is used. The CPU cost per iteration is only 10–20% greater than for the nonlinear code, with the increased cost of evaluating the adjoint residuals partially offset by that the Jacobian for the preconditioning remains fixed.

Another important point concerns the evaluation of the term $\boldsymbol{f}$, which is the source term for the linear perturbation equations, and also appears in the linearized objective function in the adjoint approach. Again, forward mode AD software could be used, but a very much simpler alternative is to use the complex Taylor series expansion method (see Ref. 30) used by Anderson and Nielsen.[31] The essence of the idea is that

$$\lim_{\epsilon \to 0} \frac{\mathcal{I}\{\boldsymbol{R}(\boldsymbol{U}, \alpha + i\epsilon)\}}{\epsilon} = \frac{\partial \boldsymbol{R}}{\partial \alpha}$$

In this equation, $\boldsymbol{R}(\boldsymbol{U}, \alpha)$ has been taken to be a complex analytic function, and the notation $\mathcal{I}\{\,\}$ denotes the imaginary part of a complex quantity. The equation itself is an immediate consequence of a Taylor series expansion. The convergence to the limiting value is second order in ϵ so that numerical evaluation with $\epsilon < 10^{-8}$ yields double-precision accuracy. In practice, we use $\epsilon = 10^{-20}$. Unlike the usual finite difference approximation of a linear sensitivity, there is no cancellation effect from the subtraction of two quantities of similar magnitude and, therefore, no unacceptable loss of accuracy due to machine rounding error. Applying this technique to a FORTRAN code requires little more than replacing all REAL*8 declarations by COMPLEX*16, and defining appropriate complex analytic versions of the intrinsic functions min, max, abs.

We have also found this complex variable method to be extremely helpful during program development. Because we have also written a linear perturbation code, we have used it to verify that each of the linear flux subroutines is consistent with the original nonlinear flux subroutines, by checking the identity

$$L\boldsymbol{u} = \lim_{\epsilon \to 0} \frac{\mathcal{I}\{\boldsymbol{R}(\boldsymbol{U} + i\epsilon \boldsymbol{u}, \alpha)\}}{\epsilon}$$

for arbitrary choices of $\boldsymbol{u}$. The left-hand side is computed by the linear flux routines, and the right-hand side is computed by applying the complex variable method to the nonlinear flux routines. Having performed these checks, we then verified that the adjoint flux routines were consistent with the linear routines by checking that the identity $\boldsymbol{u}^T(L^T\boldsymbol{v}) = \boldsymbol{v}^T(L\boldsymbol{u})$ holds for any $\boldsymbol{u}$ and $\boldsymbol{v}$.

If one were developing an adjoint code without first writing a linear perturbation code, then these two steps could be combined into one to compare the adjoint routines to the nonlinear flux routines to check for consistency.

Adjoint Solution Procedure

An important issue is how best to solve the adjoint equations. The eigenvalues of the adjoint matrix L^T are the same as those of the linear matrix L, and therefore, one is guaranteed to get the same convergence rate when using Krylov subspace iteration methods such as GMRES, as used by Nielsen and Anderson[16] and Anderson and Bonhaus.[17] On the other hand, if one uses standard time-marching methods with multigrid, as are commonly used to solve the nonlinear equations, it is not necessarily the case that the iterative convergence rate for the adjoint solver will match that of the linear solver.

We have analyzed this for our time-marching method, which uses Jacobi preconditioning with partial updates of the numerical smoothing fluxes (and the viscous fluxes for the Navier–Stokes equations) at selected stages in the Runge-Kutta iteration.[3] One full step of the M-stage procedure for the linear equations can be expressed as

$$\begin{aligned}
\boldsymbol{u}^{(0)} &= \boldsymbol{u}^n \\
\boldsymbol{d}^{(m)} &= \beta_m D\boldsymbol{u}^{(m-1)} + (1-\beta_m)\boldsymbol{d}^{(m-1)} \\
\boldsymbol{u}^{(m)} &= \boldsymbol{u}^{(0)} + \alpha_m P\left(\boldsymbol{f} - C\boldsymbol{u}^{(m-1)} - \boldsymbol{d}^{(m)}\right) \\
\boldsymbol{u}^{n+1} &= \boldsymbol{u}^{(M)}
\end{aligned}$$

where $\beta_1 = \alpha_5 = 1$, P is the Jacobi preconditioning matrix, and C and D are again the convective and diffusive matrices whose sum is the linear matrix L, as in Eq. (1).

The outcome of this analysis[32] is that if the adjoint equations are solved using the following M-stage iterative procedure:

$$\begin{aligned}
\tilde{\boldsymbol{v}}^{(M)} &= P^T(\boldsymbol{g} - L^T\boldsymbol{v}^n) \\
\tilde{\boldsymbol{d}}^{(M)} &= -\alpha_M \tilde{\boldsymbol{v}}^{(M)} \\
\tilde{\boldsymbol{v}}^{(m)} &= P^T\left(-\alpha_{m+1}C^T\tilde{\boldsymbol{v}}^{(m+1)} + \beta_{m+1}D^T\tilde{\boldsymbol{d}}^{(m+1)}\right) \\
\tilde{\boldsymbol{d}}^{(m)} &= -\alpha_m\tilde{\boldsymbol{v}}^{(m)} + (1-\beta_{m+1})\tilde{\boldsymbol{d}}^{(m+1)} \\
\boldsymbol{v}^{n+1} &= \boldsymbol{v}^n + \sum_{m=1}^{M} \alpha_m \tilde{\boldsymbol{v}}^{(m)}
\end{aligned}$$

Then the value of the linearized objective function from the linear and adjoint codes is not only identical once they have each converged to the final steady state, but it is also identical after each Runge–Kutta timestep. Note that this iteration uses the transpose of the Jacobi preconditioning matrix and works "backward" from $m = M$ to $m = 1$. If partial updating of the dissipative fluxes is not used, then it can be shown that this reduces to the standard Runge–Kutta method but with the transposed preconditioner. However, with the use of partial updating, which is commonly employed to lower the CPU cost, it requires quite a lengthy analysis to determine this form for the adjoint iteration.

Furthermore, the analysis also extends to the use of multigrid and shows that the key here is that the restriction operator for the adjoint code must be the transpose of the prolongation operator for the linear code, and vice versa, and the number of presmoothing iterations for the adjoint code must equal the number of postsmoothing iterations for the linear code, and vice versa. Provided that these two conditions are satisfied, the linear and adjoint codes produce identical values for the functional after the same number of multigrid cycles.

This result is important for two reasons. The first is that it guarantees that the adjoint code converges and that it does so with the same rate of convergence as the linear code, which is itself equal to the asymptotic rate of convergence of the nonlinear code. Thus, the adjoint code benefits from the wealth of experience and fine tuning of iterative procedures for nonlinear codes. The second reason is that it provides another validation check on the correct implementation of the adjoint code. If the linear and adjoint codes do not produce identical values for the functional after one time step, it indicates a programming error.

Strong Boundary Conditions

Although it is possible to solve the Euler equations with solid wall boundary conditions imposed weakly by specifying zero mass flux through the wall faces, it is more common when there are grid nodes on the wall to use strong boundary conditions and force the normal component of the velocity at surface nodes to be zero. In doing so, the normal component of the momentum equation flux residual is discarded. Similarly, when the Navier–Stokes equations are discretized, the entire velocity at the surface nodes is set to zero, and all components of the momentum residual are discarded. Thus, in both cases the equations that are solved are actually of the form

$$(I - B)\boldsymbol{R}(\boldsymbol{U}) = 0, \qquad B\boldsymbol{U} = 0$$

where I is the identity matrix and B is a projection matrix, which in the case of the Euler equations extracts the normal component of the boundary velocity and in the case of the Navier–Stokes equations extracts the entire boundary velocity. The presence of the term $(I - B)$ reflects the discarding of the appropriate flux residual components, to be replaced by the strong boundary conditions $B\boldsymbol{U} = 0$.

When considering linear perturbations to these equations, we obtain

$$(I - B)(L\boldsymbol{u} - \boldsymbol{f}) = 0, \qquad B\boldsymbol{u} = \boldsymbol{b}$$

where $\boldsymbol{b}$ is a boundary velocity that is zero for the Navier–Stokes equations but nonzero for the Euler equations due to a rotation in the surface normal.

These two equations can be combined to form

$$[(I - B)L + B]\boldsymbol{u} = (I - B)\boldsymbol{f} + \boldsymbol{b} \tag{2}$$

and the appropriate adjoint equation is then found by transposing the linear operator, noting that B is symmetric, to obtain

$$[L^T(I - B) + B]v = g \tag{3}$$

At this point it is convenient to decompose both v and g into orthogonal components as

$$v = (I - B)v + Bv = v_\parallel + v_\perp, \qquad g = (I - B)g + Bg = g_\parallel + g_\perp$$

Premultiplying Eq. (3) by $(I - B)$ shows that $v_\parallel$ satisfies the adjoint equations

$$(I - B)L^T v_\parallel = g_\parallel, \qquad Bv_\parallel = 0$$

These are the equations that are solved iteratively by the adjoint code. Then, once $v_\parallel$ has been computed, $v_\perp$ is calculated in a post-processing step using an equation obtained by premultiplying Eq. (3) by B:

$$v_\perp = g_\perp - BL^T v_\parallel \tag{4}$$

Having computed $v_\parallel$ and $v_\perp$, the linearized functional is given by

$$\tilde{J} = v^T[(I - B)f + b] + \frac{\partial J}{\partial \alpha}$$

$$= v_\parallel^T f + v_\perp^T b + \frac{\partial J}{\partial \alpha}$$

This shows that $v_\perp$ gives the sensitivity of the functional to the boundary condition b that arises from the rotation of the boundary normal in the case of inviscid flows.

Note that $v_\perp$ does not correspond to the normal momentum component of the analytic adjoint solution at the boundary. Hence for visualization, purposes, it is desirable to replace $v_\perp$ by the analytic boundary condition

$$v_\perp^{\text{analytic}} = h$$

which would normally be employed using a continuous formulation. Here h is zero everywhere except on the solid wall, where it corresponds to the sensitivity of the functional to the addition of momentum on the surface. In the case of a lift functional, for example, the element of h at a surface node n is

$$h_n = \begin{pmatrix} 0 \\ j \\ 0 \end{pmatrix}$$

with j being the unit vector in the lift direction.

Residual Contributions to the Functional

If the functional of interest is a force, such as lift or drag, we have to include the surface momentum residuals, which are discarded in imposing the strong boundary conditions, to have a complete force balance. Indeed, for viscous calculations, it is the tangential component of these residuals that corresponds to the viscous shear stress that is, one defines the surface shear stress to have the value that is necessary to make the tangential momentum residual equal to zero.

The nonlinear functional is, thus, of the form

$$J = J_p(U) + h^T BR(U) \tag{5}$$

where J_p corresponds to the force due to the pressure distribution on the body and h is again the vector that takes the component of the discarded momentum residuals in the selected force direction, for example, the direction normal to the freestream in the case of lift.

The corresponding linearized functional is

$$\tilde{J} = g_p^T u + h^T BLu + \frac{\partial J}{\partial \alpha} \tag{6}$$

where

$$g_p^T \equiv \frac{\partial J_p}{\partial U} \tag{7}$$

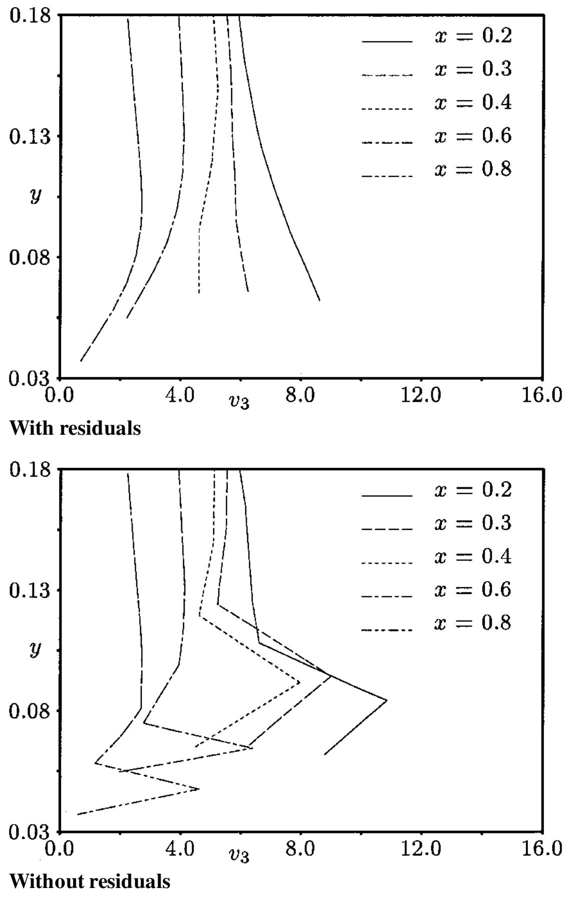

Fig. 1 Variation in third adjoint component in y direction for a subsonic NACA 0012 test case, with and without residual contributions to the functional.

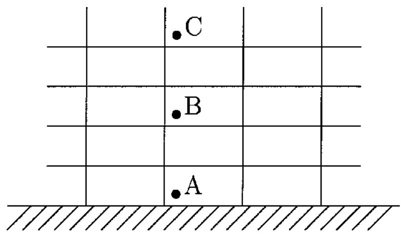

Fig. 2 Three possible locations of momentum injection close to a wall.

and so we obtain

$$g = g_p + L^T Bh \tag{8}$$

Fortunately, the second term in this equation can be computed in a preprocessing step using the adjoint flux routines.

The inclusion of the extra term makes a dramatic improvement to the quality of the adjoint solution near the surface, as illustrated in Fig. 1 for a subsonic NACA 0012 test case to be discussed later in more detail. To understand why it makes such a difference, note that the adjoint variables correspond to the linearized effect of mass, momentum, and energy sources on the functional of interest. Therefore, it is helpful to consider what happens in the linearized flow calculation when normal momentum is added close to a wall, as shown in Fig. 2.

The effect of the momentum addition on the far-field flow solution will be negligible. Therefore, with a conservative treatment, through the inclusion of the discarded momentum residuals, the linear code will correctly predict that the change in the lift is equal and opposite to the addition of normal momentum, regardless of the location of the momentum addition. On the other hand, without the inclusion of the discarded residuals, the addition of momentum at point A, right next to the wall, will have zero effect on the functional because it will contribute solely to the momentum residuals at surface nodes. Similarly, addition at point B will have some effect on the residuals at nearby surface points; if these are not included in the functional, then the influence on the functional must be incorrect. Only at point C, well away from the surface, will the effect on the surface residuals be very small, and so the effect on the functional is correctly captured without the inclusion of the discarded residuals.

Numerical Results

The nonlinear HYDRA code has been validated previously.[21,33] In this section, we are interested in verifying the equality of the linear and adjoint sensitivities and the iterative convergence rates. We also examine an interesting difficulty concerned with the linearization of strong shocks.

Inviscid Flow over NACA 0012 Airfoil

The first two cases consider steady inviscid flow over a NACA 0012 airfoil. The circles in Fig. 3 show the lift coefficient obtained from the nonlinear code plotted against angle of attack at a freestream Mach number of 0.68. The angle-of-attack variation is achieved by rotating the airfoil as well as the points on and near the airfoil surface. Doing this in a linearized sense gives the geometric perturbations required for the source terms in the linear code and the functional in the adjoint code. The lines in Fig. 3 are the lift slope obtained from the linear and adjoint codes, with the base flow in each case being the nonlinear flow conditions at the angle of attack at the midpoint of the line. The linear and adjoint codes produce values for the lift slope that are identical to machine accuracy, as they should for the fully discrete adjoint approach. They also match well the slope of the nonlinear results. Finite differencing of the nonlinear values yields a slope that agrees to within an error of 10^{-4}.

An interesting situation arises at higher Mach numbers at which there are strong shocks. Figure 4 shows the Mach contours for the

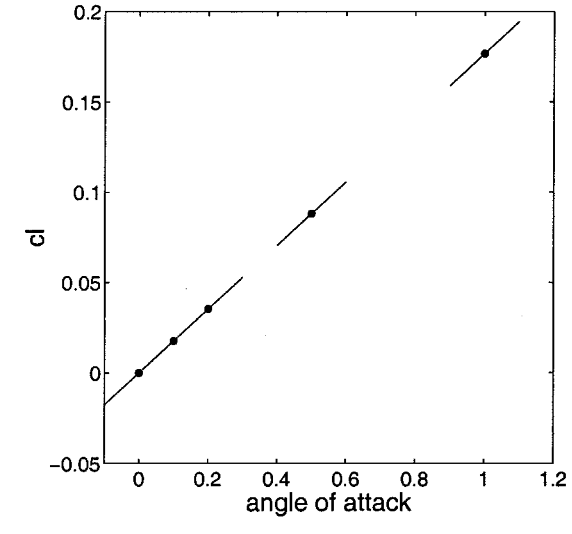

Fig. 3 C_l **vs angle of attack for a NACA 0012 profile at** $M = 0.68$**.**

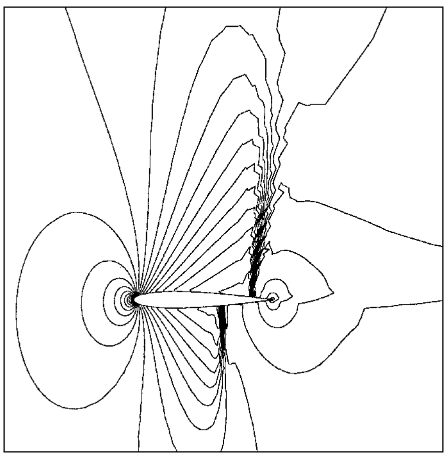

Fig. 4 Mach contours for NACA 0012 at $M = 0.85$**.**

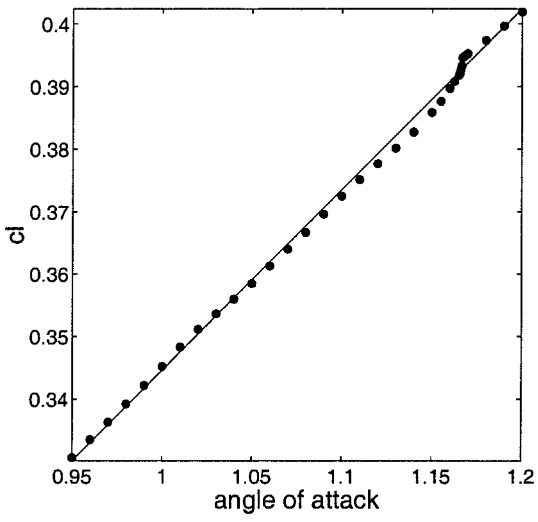

Fig. 5 C_l **vs angle of attack for NACA 0012 at** $M = 0.85$**.**

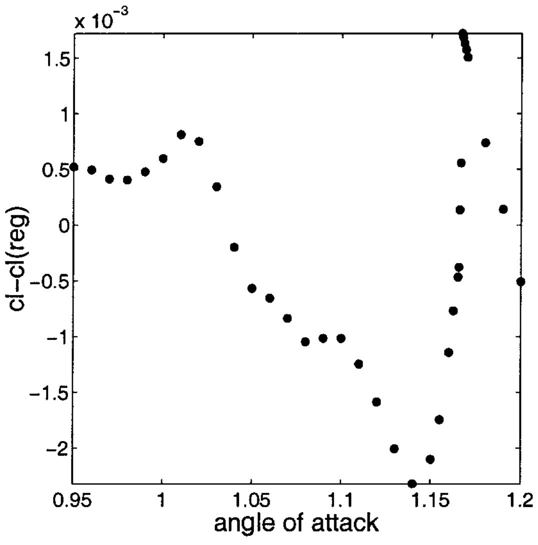

Fig. 6 $C_l - C_l$**(regression) for NACA 0012 at** $M = 0.85$**.**

NACA 0012 at an angle of attack of 1 deg and an increased Mach number of 0.85. There are now two shocks, with the maximum local Mach number reaching approximately 1.45 on the supersonic side of the suction surface shock. The circles in Fig. 5 show the nonlinear lift coefficients over a limited range of angles of attack. The line in Fig. 5 is a linear regression least-square fit of the nonlinear data. The results indicate a peculiar lack of smoothness in the nonlinear data; this is shown more clearly in Fig. 6, which plots the difference between the nonlinear data and the linear regression.

The key point is that there is no physical justification for the loss of smoothness. It appears to be a purely numerical artifact that is probably related to the displacement of the shock as the angle of attack changes. Therefore, the slope of the linear regression line is probably the best representation of the true lift slope. However, the linear/adjoint codes give lift slopes that correspond to the local derivative of the nonlinear data. Figure 7 plots the difference between the linear/adjoint slopes and the slope of the linear regression, showing a large discrepancy around 1.17 deg, where the local derivative of the nonlinear data differs significantly from the linear regression value. Figure 8 plots the number of multigrid cycles required to converge the nonlinear code to a very tight tolerance. Interestingly, the number of cycles increases substantially around 1.17 deg. This suggests that the linearization matrix may be almost singular, which could be related to that small changes in the angle of attack produce larger changes in the lift than one would otherwise expect.

A similar phenomenon has been observed by Elliott, who reported problems with instabilities in the iterative solution of the adjoint Euler equations when the underlying nonlinear flow solver failed to

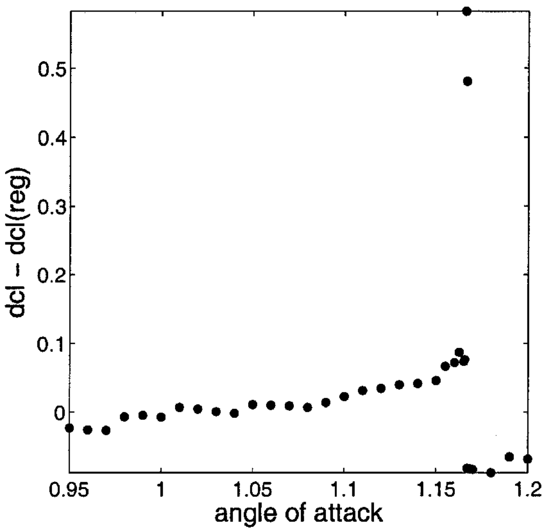

Fig. 7 NACA 0012 at $M = 0.85$, $dC_l/d\alpha$(linear) — $dC_l/d\alpha$(regression).

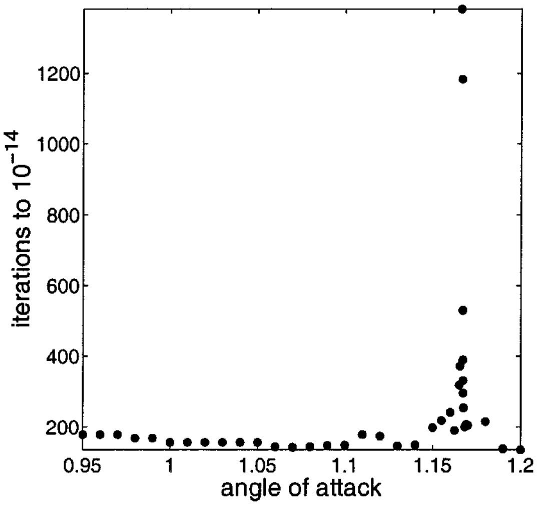

Fig. 8 Number of multigrid cycles for nonlinear calculations for NACA 0012 at $M = 0.85$.

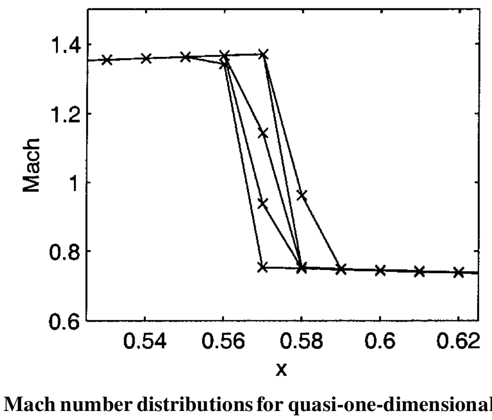

Fig. 9 Mach number distributions for quasi-one-dimensional test case for a range of exit pressures with uniform grid spacing $\Delta x = 1/100$.

converge to machine accuracy.[15] These problems were avoided by increasing the level of smoothing, at the expense of losing shock resolution.

Another perspective on this issue is provided by Figs. 9–11, which show results for a quasi-one-dimensional test case with transonic flow in a diverging duct. The quasi-one-dimensional flow equations in the form

$$\frac{dF}{dx} + A^{-1}\frac{dA}{dx}(F - P) = 0$$

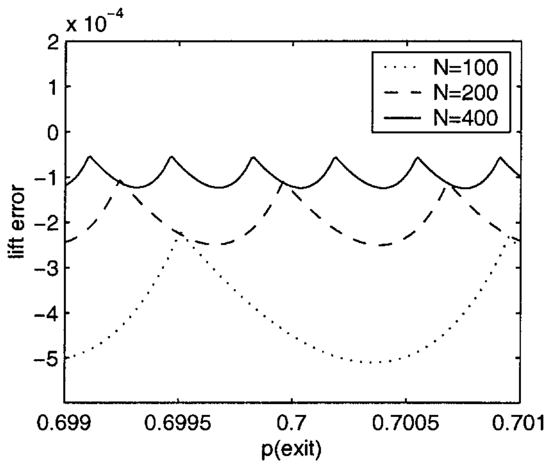

Fig. 10 Lift error for quasi-one-dimensional test case with uniform grid spacing $\Delta x = 1/N$.

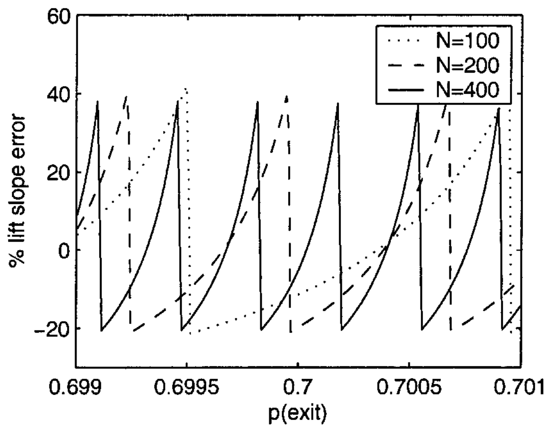

Fig. 11 Lift slope error for quasi-one-dimensional test case with uniform grid spacing $\Delta x = 1/N$.

where $A(x)$ is the duct area and

$$F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u H \end{pmatrix}, \qquad P = \begin{pmatrix} 0 \\ p \\ 0 \end{pmatrix}$$

are approximated using Roe's first-order flux difference splitting[34] and solved by a Newton iteration. Results are obtained for three different grid resolutions and over a range of exit pressures. Figure 9 shows the Mach number distributions on the central part of the coarsest grid for five different values of the exit pressure. The very sharp nature of the shock capturing is evident. Figure 10 shows the error in the computed value of the lift, the integral of the pressure along the duct. There is clear first-order convergence in the maximum error as the grid is refined. However, the variation in lift with exit pressure is not smooth. Because of the dependence of the Roe flux on the absolute magnitude of the characteristic speed $u - c$ at the faces between computational cells, when the sign of this quantity changes there is a corresponding discontinuity in the slope. This leads to the "scalloped" appearance of the lift error, with the spacing between peaks corresponding to the change in exit pressure required to move the shock through one cell. This results in the error in the lift slope failing to converge as the grid is refined, as shown in Fig. 10. Modifications to the absolute magnitude, such as the use of a smooth "entropy fix," to ensure the continuity of the slope, would not alter the basic problem that, as the shock moves a distance Δx due to changes in the exit pressure, the variation in the lift error is $\mathcal{O}(\Delta x)$, and hence, the lift slope error (which is equal to the slope of the lift error) is $\mathcal{O}(1)$, that is, it does not converge as $\Delta x \to 0$.

The fact that grid convergence of nonlinear flow calculations does not guarantee convergence of linear sensitivities is a fundamental problem for the discrete approach to adjoint calculations. On the other hand, the mathematical formulation of the adjoint PDE for the continuous approach requires the imposition of an adjoint boundary condition along the shock,[35,36] and if this is not specified, then again one is not guaranteed to converge to the correct value as the grid is refined.

However, this observation of limitations with the linearization of flows with strong shocks may be primarily of academic interest and not of engineering concern. Most aeronautical applications do not have such strong normal shocks, and with weak shocks we have not observed a similar phenomenon. If one is interested in an application with a strong shock, then it may be possible to use more numerical smoothing at shocks to obtain a convergent linear sensitivity.[36,37]

Turbulent Flow over RAE 2822 Airfoil

Figure 12 presents the Mach contours for the Reynolds averaged flow over the Royal Aircraft Establishment (RAE) 2822 airfoil at angle of attack $\alpha = 2.4$ deg, freestream Mach number $M = 0.725$, and Reynolds number $Re = 6.5 \times 10^6$. The turbulence is modeled using a Spalart–Allmaras single-equation model.[38] The circles in Fig. 13 show the sensitivity of the variation in the lift coefficient with changes in the angle of attack. The lines correspond to the lift slopes computed by the linear and adjoint codes, which are again in perfect agreement with each other. There is no evidence of any

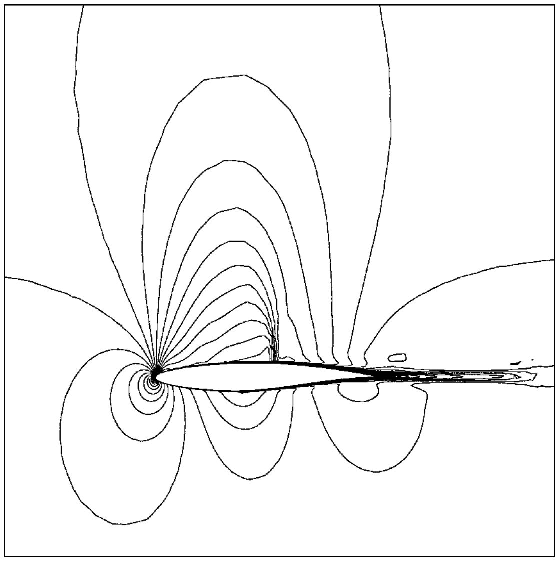

Fig. 12 Mach contours for an RAE 2822 profile at $M = 0.725$ and $Re = 6.5 \times 10^6$.

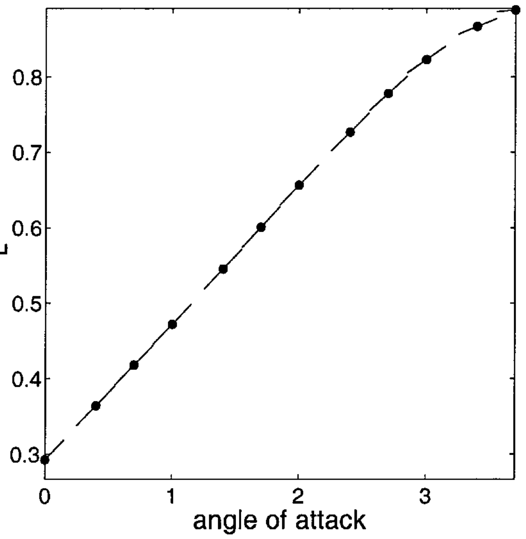

Fig. 13 Lift vs angle of attack for an RAE 2822 profile at $M = 0.725$ and $Re = 6.5 \times 10^6$.

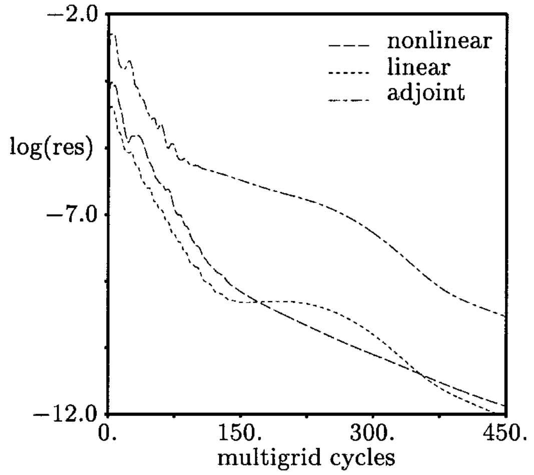

Fig. 14 Convergence histories for the nonlinear, linear and adjoint codes for an RAE 2822 profile at $M = 0.725$ and $Re = 6.5 \times 10^6$.

lack of smoothness in the nonlinear lift predictions, and the linear and adjoint codes again give lift slopes that are identical to machine accuracy and are in very good agreement with the nonlinear results.

Figure 14 shows the convergence histories for the nonlinear, linear, and adjoint codes for the RAE 2822 testcase at $\alpha = 2.4$ deg. As expected, they all exhibit the same asymptotic convergence rate.

Conclusions

We have presented a number of algorithm developments concerned with the formulation and solution of adjoint Euler and Navier–Stokes equations using the discrete approach. These include the treatment of strong boundary conditions and the associated adjoint boundary conditions for lift and drag functionals, as well as a Runge–Kutta time-marching scheme that ensures exact equivalence with a linear perturbation code throughout the convergence process. This property guarantees the same asymptotic convergence rate for nonlinear, linear, and adjoint solvers, as well as being very useful during code validation.

We have also discussed a potential problem with adjoint methods applied to flows with strong shocks. In practice, however, we think this is unlikely to cause problems in design applications with very weak shocks.

Acknowledgments

This research has been supported by the Engineering and Physical Sciences Research Council under Grants GR/K91149 and GR/L95700 and by Rolls–Royce, plc, Technical Monitor, Leigh Lapworth; the Defence Evaluation and Research Agency, Technical Monitor, John Calvert; and BAESystems, plc, Technical Monitor, David Standingford. We also acknowledge the contributions of P. Moinier and M. S. Campobasso at Oxford University and L. Lapworth and M. West at Rolls–Royce, plc, to the development of the HYDRA codes.

References

[1]Lions, J. L., *Optimal Control of Systems Governed by Partial Differential Equations*, translated by S. K. Mitter, Springer-Verlag, Berlin, 1971.

[2]Pironneau, O., "On Optimum Design in Fluid Mechanics," *Journal of Fluid Mechanics*, Vol. 64, 1974, pp. 97–110.

[3]Jameson, A., "Aerodynamic Design via Control Theory," *Journal of Scientific Computing*, Vol. 3, 1988, pp. 233–260.

[4]Jameson, A., "Optimum Aerodynamic Design Using Control Theory," *Computational Fluid Dynamics Review 1995*, edited by M. Hafez and K. Oshima, Wiley, New York, 1995, pp. 495–528.

[5]Jameson, A., Pierce, N., and Martinelli, L., "Optimum Aerodynamic Design Using the Navier–Stokes Equations," *Journal of Theoretical and Computational Fluid Mechanics*, Vol. 10, 1998, pp. 213–237.

[6]Jameson, A., "Reengineering the Design Process Through Computation," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 36–50.

[7]Reuther, J., Jameson, A., Alonso, J. J., Remlinger, M. J., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, Part 1," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 51–60.

[8]Reuther, J., Jameson, A., Alonso, J. J., Remlinger, M. J., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, Part 2," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 61–74.

[9]Taylor, A. C., III, Hou, G. W., and Korivi, V. M., "Methodology for Calculating Aerodynamic Sensitivity Derivatives," *AIAA Journal*, Vol. 30, No. 10, 1992, pp. 2411–2419.

[10]Ta'asan, S., Kuruvila, G., and Salas, M. D., "Aerodynamic Design and Optimization in One Shot," AIAA Paper 92-0025, Jan. 1992.

[11]Baysal, O., and Eleshaky, M., "Aerodynamic Design Optimization Using Sensitivity Analysis and Computational Fluid Dynamics," *AIAA Journal*, Vol. 30, No. 3, 1992, pp. 718–725.

[12]Anderson, W. K., and Venkatakrishnan, V., "Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation," *Computers and Fluids*, Vol. 28, No. 4–5, 1999, pp. 443–480.

[13]Dadone, A., and Grossman, B., "Progressive Optimization of Inverse Fluid Dynamic Design Problems," *Computers and Fluids*, Vol. 29, No. 1, 2000, pp. 1–32.

[14]Elliott, J., and Peraire, J., "Practical Three-Dimensional Aerodynamic Design and Optimization Using Unstructured Meshes," *AIAA Journal*, Vol. 35, No. 9, 1997, pp. 1479–1485.

[15]Elliott, J., "Aerodynamic Optimization Based on the Euler and Navier–Stokes Equations Using Unstructured Grids," Ph.D. Dissertation, Dept. of Aeronautics and Astronautics, Massachusetts Inst. of Technology, Cambridge, MA, 1998.

[16]Nielsen, E., and Anderson, W. K., "Aerodynamic Design Optimization on Unstructured Meshes Using the Navier–Stokes Equations," *AIAA Journal*, Vol. 37, No. 11, 1999, pp. 957–964.

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

[18]Mohammadi, B., and Pironneau, O., "Mesh Adaption and Automatic Differentiation in a CAD-Free Framework for Optimal Shape Design," *International Journal for Numerical Methods in Fluids*, Vol. 30, No. 2, 1999, pp. 127–136.

[19]Kim, H.-J., Sasaki, D., Obayashi, S., and Nakahashi, K., "Aerodynamic Optimization of Supersonic Transport Wing Using Unstructured Adjoint Method," *Computational Fluid Dynamics 2000*, edited by N. Satofuka, Springer, 2001, pp. 581–588.

[20]Newman, J. C., Taylor, A. C., Barnwell, R. W., Newman, P. A., and Hou, G. J.-W., "Overview of Sensitivity Analysis and Shape Optimization for Complex Aerodynamic Configurations," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 87–96.

[21]Moinier, P., Müller, J.-D., and Giles, M. B., "Edge-Based Multigrid and Preconditioning for Hybrid Grids," *AIAA Journal*, Vol. 40, No. 10, 2002.

[22]Duta, M. C., Giles, M. B., and Campobasso, M. S., "The Harmonic Adjoint Approach to Unsteady Turbomachinery Design," *International Journal for Numerical Methods in Fluids*, Vol. 40, 2002, pp. 323–332.

[23]Campobasso, M. S., Duta, M. C., and Giles, M. B., "Adjoint Methods for Turbomachinery Design," International Symposium on Air Breathing Engines, June 2001.

[24]Duta, M. C., "The Use of the Adjoint Method for the Minimisation of Forced Vibration in Turbomachinery," Ph.D. Dissertation, Oxford Univ., Oxford, March 2002.

[25]Faure, C., and Papegay, Y., "Odyssée User's Guide," Ver. 1.7, TR RT-0224, INRIA, Sophia-Antipolis, 1998.

[26]Bischof, C. H., Carle, A., Hovland, P. D., Khademi, P., and Mauer, A., "ADIFOR 2.0 User's Guide," Rev. D, TR 192, Mathematics and Computer Science Div., Argonne National Lab., Argonne, France, 1998.

[27]Carle, A., Fagan, M., and Green, L. L., "Preliminary Results from the Application of Automated Code Generation to CFL3D," AIAA Paper 98-4807, June 1998.

[28]Giering, R., and Kaminski, T., "Recipes for Adjoint Code Construction," *ACM Transactions on Mathematical Software*, Vol. 24, No. 4, 1998, pp. 437–474.

[29]Griewank, A., *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[30]Squire, W., and Trapp, G., "Using Complex Variables to Estimate Derivatives of Real Functions," *SIAM Review*, Vol. 10, No. 1, 1998, pp. 110–112.

[31]Anderson, W. K., and Nielsen, E., "Sensitivity Analysis for Navier–Stokes Equations on Unstructured Grids Using Complex Variables," *AIAA Journal*, Vol. 39, No. 1, 2001, pp. 56–63.

[32]Giles, M. B., "On the Use of Runge–Kutta Time-Marching and Multigrid for the Solution of Steady Adjoint Equations," Oxford Univ. Computing Lab., TR NA00/10, Oxford Univ., Oxford, June 2000.

[33]Moinier, P., "Algorithm Developments for an Unstructured Viscous Flow Solver," Ph.D. Dissertation, Oxford Univ., Oxford, June 1999.

[34]Roe, P. L., "Discrete Models for the Numerical Analysis of Time-Dependent Multidimensional Gas Dynamics," *Journal of Computational Physics*, Vol. 63, No. 2, 1986, pp. 458–476.

[35]Giles, M. B., and Pierce, N. A., "Analytic Adjoint Solutions for the Quasi-One-Dimensional Euler Equations," *Journal of Fluid Mechanics*, Vol. 426, 2001, pp. 327–345.

[36]Giles, M. B., "Discrete Adjoint Approximations with Shocks," *Proceedings of HYP2002*, edited by T. Hou and E. Tadmor, Springer-Verlag, Berlin (to be published).

[37]Lindquist, D. R., and Giles, M. B., "Validity of Linearized Unsteady Euler Equations with Shock Capturing," *AIAA Journal*, Vol. 32, No. 1, 1994, pp. 46–53.

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

P. Givi
Associate Editor