AIAA JOURNAL
Vol. 44, No. 4, April 2006

Efficient Construction of Discrete Adjoint Operators on Unstructured Grids Using Complex Variables

Eric J. Nielsen* and William L. Kleb†
NASA Langley Research Center, Hampton, Virginia 23681

A methodology is developed and implemented to mitigate the lengthy software development cycle typically associated with constructing a discrete adjoint solver for aerodynamic simulations. The approach is based on a complex-variable formulation that enables straightforward differentiation of complicated real-valued functions. An automated scripting process is used to create the complex-variable form of the set of discrete equations. An efficient method for assembling the residual and cost function linearizations is developed. The accuracy of the implementation is verified through comparisons with a discrete direct method as well as a previously developed handcoded discrete adjoint approach. Comparisons are also shown for a large-scale configuration to establish the computational efficiency of the present scheme. To ultimately demonstrate the power of the approach, the implementation is extended to high-temperature gas flows in chemical nonequilibrium. Finally, several fruitful research and development avenues enabled by the current work are suggested.

Nomenclature

$\boldsymbol{D}$	=	vector of design variables
D_{ij}	=	elements of diagonal subblock
E	=	total energy per unit volume
e	=	internal energy per unit mass
f	=	objective function
h	=	step size
$\boldsymbol{I}$	=	identity matrix
i	=	$\sqrt{-1}$
$\boldsymbol{K}$	=	mesh movement coefficient matrix
n	=	time level, number of grid points
$\boldsymbol{Q}$	=	flowfield-dependent variables
q	=	heat flux
$\boldsymbol{R}$	=	discretized residual vector
T	=	temperature
t	=	time
u, v, w	=	velocity components
V	=	local cell volume
$\boldsymbol{X}$	=	computational mesh
γ	=	ratio of specific heats
Λ	=	vector of adjoint variables
$\tilde{\nu}$	=	turbulence variable
ρ	=	density

Introduction

IN the field of gradient-based aerodynamic design optimization, there are a number of options available to obtain sensitivity information from computational fluid dynamics (CFD) solvers, and the burden associated with implementing these methods varies widely. Moreover, the efficiency and accuracy of the results depend highly on the method chosen.

Perhaps the most straightforward of these schemes is a simple finite difference algorithm.[1–3] In this approach, the solver can be treated as a "black box," and sensitivities are generated by merely differencing neighboring solutions. The advantage of this technique is its ease of implementation; however, its accuracy can vary widely with the perturbation size. Central differencing is theoretically second-order accurate, but subtractive cancellation error caused by finite precision arithmetic limits the effective step size that can be used. In addition, the cost of the method scales linearly with the number of design variables.

Direct differentiation[4–10] and adjoint[11–35] approaches provide alternative, more elaborate means for obtaining sensitivity information. Whether to use a direct or adjoint approach is usually determined by the parameters of the problem. For cases involving many objectives or constraints and relatively few design variables, the direct approach is appropriate. In this case the solution of an additional linear system of equations for each design variable yields sensitivity information for all of the dependent variables in the flowfield. Conversely, the adjoint approach yields sensitivity information for a single function with respect to many design variables at the cost of solving a single linear system of equations. For typical aerodynamic design problems where the number of variables is large and there are relatively few objectives and constraints, the adjoint approach is generally preferred. Moreover, the adjoint approach can also be used to obtain mathematically rigorous mesh adaptation information that is often nonintuitive and can be used to efficiently guide output-based computational simulations to grid-converged results.[36,37]

Both the direct and adjoint techniques can be applied in either a continuous or discrete setting, depending on the order in which the differentiation and discretization processes are performed; the current work focuses on the discrete variant of the adjoint approach. One major advantage of the discrete approach is that the system of auxiliary equations is uniquely determined by the baseline discretization of the governing equations. Although perhaps difficult to achieve in practice, this property implies that the implementation of the adjoint system can be automated. This is not true for a continuous approach, where changes to the baseline equation set require new derivations of the associated adjoint operators prior to implementation. These operators might prove prohibitively difficult to obtain for complicated functions such as turbulence models and finite-rate chemistry. Another advantage of the discrete approach is that the results can be rigorously verified using the baseline code because the linearizations take place at the discrete level. In a continuous adjoint context, the "correct" answer is generally not known, and verification of the adjoint discretization for even moderately complex problems can be extremely difficult, if not impossible. The linearization of the discrete system also ensures that the design optimization framework uses gradients that are discretely consistent with the analysis problem.

Presented as Paper 2005-0324 at the AIAA 43rd Aerospace Sciences Meeting, Reno, NV, 10–13 January 2005; received 28 January 2005; revision received 5 July 2005; accepted for publication 31 August 2005.

*Research Scientist, MS 128, Computational Modeling and Simulation Branch. Senior Member AIAA.
†Research Scientist, MS 408A, Aerothermodynamics Branch. Lifetime Member AIAA.

Regardless of whether a direct or an adjoint method is used, the discrete form of both approaches ultimately requires an exact linearization of the discrete residual vector and the cost function of interest with respect to both the flowfield variables and the grid. Obtaining these linearizations by hand is time consuming and error prone; manually differentiating a CFD solver for large-scale turbulent flow applications is a monumental undertaking. For example, the discrete adjoint implementation described in Refs. 28–31 has taken more than five years to mature into a robust and accurate tool. Furthermore, any changes to the fundamental discretization, boundary conditions, physical models, or objective function require new linearizations. This lengthy software development cycle has been the primary impediment to widespread use of either approach in conjunction with Euler- and Navier–Stokes-based simulation tools. Tools aimed at automating this process have been under development for some time[7,9,25,38,39]; however, these applications are seldom "hands off" and frequently fail to produce code that rivals the speed and low storage requirements of hand-developed implementations.

In Ref. 40, a technique based on the use of complex variables was introduced that allows derivatives of a real-valued function to be computed with minimal changes to the analysis code. Recently reviewed in Ref. 41, this approach yields sensitivity information equivalent to a discrete direct method and has been applied in Ref. 42 to a Reynolds-averaged Navier–Stokes solver for three-dimensional turbulent flow on unstructured grids. Furthermore, an automated form of this capability is described in Ref. 43, where a scripting approach is used to automatically convert the entire baseline solver to a complex-variable formulation, including such constructs as the file I/O and parallel communication. Unfortunately, the cost of the complex-variable approach, like that of direct differentiation, scales with the number of design variables. However, its automatable implementation, readability, and discrete consistency with the analysis problem are major advantages of the method.

In the current work, a hybrid approach to sensitivity analysis is developed and implemented. To retain the ability to scale to large numbers of design variables, the overall scheme is fundamentally equivalent to the discrete adjoint approach taken in Refs. 28–31. However, an alternative means for forming the residual and cost function linearizations is utilized where automated complex-variable forms of the discrete residual and cost function routines are used to compute the required Jacobians for both the dependent variables and the grid. This new approach requires detailed knowledge of the baseline discretization to achieve efficiency comparable to the previously developed hand-coded implementation; however, no manual code differentiation is required. The resulting scheme is discretely consistent with the baseline solver, provides an adjoint capability for complex equation sets, and requires substantially less code development effort than previous methods.

The remainder of this paper is divided into the following sections. First, the discrete adjoint approach for aerodynamic sensitivity analysis is reviewed to motivate the need for a method by which linearizations of complicated algorithms can be obtained with minimal effort. A brief overview of the complex-variable approach to function differentiation is given, including the relevant advantages and disadvantages of the method. Following this, the formulation of the proposed hybrid complex-variable/adjoint approach is described, which includes an in-depth discussion of the automated generation of complex-valued source code for the residual and objective function evaluations as well as important implementation details critical to the efficiency of the new scheme. Verification of the method is shown for fully turbulent flow by using comparisons with a direct discrete approach as well as the previously developed hand-coded discrete adjoint capability. A large-scale test case is used to demonstrate the computational efficiency of the new scheme relative to the hand-coded implementation. Finally, the generality of the new method is explored by applying it to high-temperature gas equation sets required for hypersonic aerothermodynamic analysis. Conclusions and opportunities for future research are given.

Discrete Adjoint Approach for Aerodynamic Sensitivity Analysis

The governing equations are the compressible and incompressible[44] Euler and Reynolds-averaged Navier–Stokes equations. The system is closed using the perfect-gas equation of state. For turbulent flows, the one-equation turbulence model of Ref. 45 is used. The derivation of the discrete adjoint system is widely available in the literature and is not repeated here. Using the approach outlined in Ref. 31, the final set of discrete equations takes the following form:

$$\left[\frac{V}{\Delta t}\boldsymbol{I} + \left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{Q}^*}\right)^T\right]\Delta\Lambda_f^n = \left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{Q}^*}\right)^T \Lambda_f^n - \frac{\partial f}{\partial \boldsymbol{Q}^*} \tag{1}$$

with $\Lambda_f^{n+1} = \Lambda_f^n - \Delta\Lambda_f^n$. Here, V and Δt are the local cell volume and time step, respectively, $\boldsymbol{Q}^*$ is the vector of steady-state dependent variables, and Λ_f is the vector of flowfield adjoint variables. As discussed in Ref. 31, any convenient linearization of $\boldsymbol{R}$ can be used on the left-hand side, provided it is sufficient to converge the problem. However, the linearizations of the residual and cost function appearing on the right-hand side must be exact. These terms can be extremely cumbersome to implement and often involve linearizations of complex algorithms such as reconstruction operators, flux limiters, boundary conditions, and turbulence models.

Once Eq. (1) has been solved for the flowfield adjoint variable Λ_f, the sensitivity vector ∇f can be computed as

$$\nabla f = \frac{\partial f}{\partial \boldsymbol{D}} + \Lambda_f^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{D}} - \Lambda_g^T \left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right]_{\text{surface}} \tag{2}$$

where Λ_g is an additional adjoint variable, which satisfies a grid adjoint equation[46]:

$$\boldsymbol{K}^T \Lambda_g = -\left[\frac{\partial f}{\partial \boldsymbol{X}} + \left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right)^T \Lambda_f\right] \tag{3}$$

Here, a mesh movement scheme of the form $\boldsymbol{KX} = \boldsymbol{X}_{\text{surface}}$ such as that adopted in Ref. 30 is used during the design procedure. Note that, in general, the linearizations of $\boldsymbol{R}$ and f with respect to the grid that appear in the right-hand side of Eq. (3) are just as cumbersome to obtain as those for Eq. (1).

Differentiation of Real-Valued Functions by Using Complex Variables

In Ref. 40, a Taylor series with a complex step size ih has been used to derive an expression for the first derivative of a real-valued function $f(x)$:

$$f'(x) = \text{Im}[\bar{f}(x + ih)]/h + \text{O}(h)^2 \tag{4}$$

where $\bar{f}$ denotes the complex-analytic form of f. Several observations can be made about Eq. (4). As with real-valued central differencing, the expression is second-order accurate; however, there is no subtraction of neighboring terms involved. This analytical extension allows true second-order accuracy to be realized, where two additional digits of accuracy are obtained for each order-of-magnitude reduction in the step size h. Moreover, implementation of the method is straightforward: declare all floating point variables complex; extend intrinsic functions such as min( ), max( ), and abs( ) to accommodate complex-valued arguments; and apply a complex perturbation to the design variable of interest. Execute the simulation, and, upon completion, the imaginary part of the output is the partial derivative with respect to the perturbed variable multiplied by the step size h. The drawbacks to this technique are the need to recompute f for each perturbation and the additional cost of performing complex arithmetic.

Using Complex Variables to Form Discrete Adjoint Operators

As just discussed, the exact linearizations of $\boldsymbol{R}$ and f with respect to $\boldsymbol{Q}$ and $\boldsymbol{X}$ as required by Eqs. (1) and (3) can be very difficult to obtain by hand. To circumvent these difficulties, the current work uses the complex-variable approach to obtain these linearizations. Because the complex-variable method is a direct mode of differentiation, the cost scales directly with the number of perturbations and, therefore, must be carefully implemented to be of practical use. For example, consider a residual computation on an unstructured grid using a node-based scheme. Unlike a structured-grid solver, the neighbors contributing to the residual at a node are usually not directly known; rather, an edge-based data structure is commonly used. In this manner, a residual computation typically involves a series of global gather operations to form the discrete residual vector across the entire field. To form the complete $[\partial \boldsymbol{R}/\partial \boldsymbol{Q}]^T$ operator using complex variables, each component of $\boldsymbol{Q}$ at every grid point in the field must be perturbed independently, after which a complex-valued residual must be evaluated to construct the corresponding row of the Jacobian $[\partial \boldsymbol{R}/\partial \boldsymbol{Q}]^T$. If n denotes the number of points in the grid, then this relationship implies $6n$ complex residual evaluations to form the complete Jacobian matrix for three-dimensional turbulent perfect-gas flows. Clearly, this cost would be prohibitively expensive if the implementation were performed in an ad hoc manner.

Implementation

References 28, 47, and 48 describe the flow solver used in the current work. The code uses an implicit, upwind, finite volume discretization in which the dependent variables are stored at the mesh vertices. Scalable parallelization is achieved through domain decomposition and message-passing communication. Inviscid fluxes at cell interfaces are computed using the upwind schemes of Roe[49] or Van Leer.[50] Viscous fluxes are formed using an approach equivalent to a central difference Galerkin procedure. For steady-state flows, temporal discretization is performed by using a backward-Euler time-stepping scheme.

An approximate solution of the linear system of equations formed at each time step is obtained through several iterations of a point-iterative scheme in which the nodes are updated in an even–odd fashion, resulting in a Gauss–Seidel-type method. For viscous flows, this scheme is augmented with a line-relaxation algorithm in boundary-layer regions as described in Ref. 31.

The turbulence model is integrated all of the way to the wall without the use of wall functions and can be solved in a tightly coupled fashion[31] or separately from the mean flow equations at each time step with an identical time-integration scheme. The resulting linear system is then solved with the same iterative schemes employed for the flow equations.

In Refs. 28–31, a discrete adjoint capability has been developed for the solver through hand differentiation, and the resulting adjoint system of equations is solved using an exact dual algorithm. This solver framework is employed in the current work; however, the required linearizations are formed using complex variables as described next.

Automated Generation of Complex-Valued Source Code

The capabilities just described have been implemented in a suite of Fortran95 modules that conform to a coding standard[43] that facilitates an automated conversion to complex variables. A code written in the Ruby programming language was developed in Ref. 43 that automates the conversion of the baseline real-valued solver to a complex-variable formulation. This operation yields a capability equivalent to discrete direct differentiation. Because this process is fully automated, the maintenance associated with debugging and synchronizing the complex-valued solver with the baseline solver is eliminated.

In the current work, a similar automated conversion is developed. However, many of the routines required by the residual and objective functions are shared by other parts of the adjoint solver. For this reason, it is necessary not only to create the complex-variable forms of the source code components that form $\boldsymbol{R}$ and f, but also to maintain their original real-valued counterparts and ensure that they can safely coexist. These pieces include not only subroutines and functions but also module variables and derived-type definitions.

Many of the basic elements described in Ref. 43 are leveraged for the current work; however, the need to simultaneously support real and complex variants of the various components requires considerable additional effort. For example, within complex-valued routines, Fortran95 use statements importing variables, routines, and type definitions from other modules must be modified to use the appropriate complex-valued versions. Moreover, to handle layered call stacks, this capability must be recursive. The full procedure is accomplished in three passes:

Read-only pass. Find all modules on which the residual and objective functions depend, and gather information about what they contain. Specifically, record module name, subroutines, functions, derived-type definitions, and module-level real and derived-type variable declarations.

Main code generation pass.

1) Create complex versions of type definitions that contain real-valued variables.

2) Insert complex versions of the real and derived-type module variables and their associated public declarations.

3) Create complex copies of all functions and subroutines and, within each, change variable references and calls appropriately.

4) Insert a subroutine within each module that can be called to allocate and synchronize the complex- and real-valued module variables.

Driver code generation pass. Create a main synchronization routine that calls all of the individual module-based synchronization routines. This layered approach is necessary to avoid name–space collisions of module variable names.

The autogenerated complex version of the code is joined to the existing adjoint solver framework by means of a standard Makefile. The complex version is first generated from the baseline flow solver, appropriate Makefile dependencies are generated, and finally, all code is compiled and linked to form the composite adjoint solver. A similar approach is taken for the source code used to evaluate the right-hand side of Eq. (3).

Coloring Scheme for Complex Residual Evaluations

Consider the formation of the Jacobian matrix $\boldsymbol{A} \equiv [\partial \boldsymbol{R}/\partial \boldsymbol{Q}]^T$ using complex variables, where the perturbation size in Eq. (4) is taken to be the square root of the Fortran95 intrinsic tiny( ) applied to a standard double-precision real variable. After applying the complex perturbation $i\,\Delta Q$ to an element of $\boldsymbol{Q}$ at grid point j, the entry A_{jk} can be determined by performing a complex residual evaluation and mining the imaginary parts of the residual at node k. In this manner, the rows of $\boldsymbol{A}$ can be constructed in a sequential fashion by successively perturbing the elements of $\boldsymbol{Q}$ at every grid point in the field. As already noted, this would require a complex residual evaluation for every grid point and every dependent variable in the field. However, note that upon applying a perturbation $i\,\Delta Q$ and evaluating the complex-valued residual, the imaginary part of $\boldsymbol{R}$ will be largely zero. The only nonzero terms will lie within the stencil width of the residual operator. For the discretization used in the current work, these terms correspond to the nearest and next-nearest neighbors of the perturbed grid point. A significant speedup can be realized by taking advantage of this property.

Prior to applying any complex perturbations to the field, the grid is preprocessed to establish node colorings. The nodes in each color represent nodes that do not lie within a stencil width of another, and, therefore, can be simultaneously perturbed and processed by the complex residual routine. In this manner, a much larger number of elements in $\boldsymbol{A}$ can be computed during a single complex residual evaluation across the domain.

Consider the one-dimensional structured grid shown in Fig. 1 and a five-point discretization. The first node is placed into the first color, and the neighboring nodes within a stencil width are tagged. The rest of the field is then searched for nodes that do not depend on any tagged nodes. If a node is found, it is added to the current

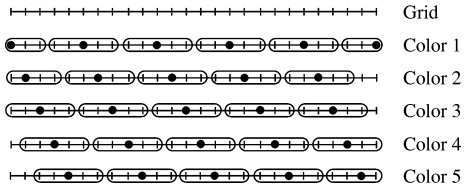

Fig. 1 Perturbation coloring scheme on one-dimensional grid: ●, a perturbation.

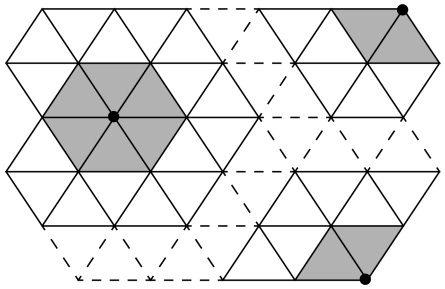

Fig. 2 Local elements provided to complex-valued residual routines: ●, a perturbation. Solid edges are needed for second-order-accurate inviscid terms; shaded elements are necessary for viscous contributions.

color, and the neighbors within its stencil are also tagged. This process continues until no more nodes can be found. At that time, the tags are reset, and a new color is initiated. This algorithm is repeated until every node in the field is placed in a color. For the five-point stencil used in Fig. 1, this results in five colors. Rather than a separate complex residual evaluation for each dependent variable at each of the 26 grid points, the coloring scheme requires just five complex residual evaluations for each dependent variable. For a similar discretization on a three-dimensional structured grid, 125 colors would be expected. For the three-dimensional unstructured grids used in the current work, there are typically between 150 and 200 colors.

Parallelizing the coloring scheme just described is straightforward. In the event that candidate points for perturbations on neighboring processors exhibit overlapping stencils at the partition boundaries, the higher-numbered processor is allowed to place its candidate into the current color, whereas the other processor must place its candidate node into a later color. This strategy is simplistic and by no means optimal; the partitioned color groups are generally not load balanced. However, this drawback has not been serious enough to warrant a more elaborate algorithm.

The mesh linearization $\partial \boldsymbol{R}/\partial \boldsymbol{X}$ required by Eq. (3) is formed in an analogous fashion by applying complex perturbations to the grid coordinates. Here, a complex evaluation of the grid metrics is required prior to the residual computation, as these underlying terms are also affected by such perturbations.

Localized Residual Computations

The scheme just described can yield colors containing widely varying numbers of grid points. The initial colors may contain several hundred grid points; however, the final color groups may each contain only a single grid point. With fewer grid points per color, evaluating a complex-valued residual across the entire field becomes increasingly inefficient. To further reduce the overall computational cost, the routines used to evaluate $\boldsymbol{R}$ have been modified to accept an optional list of elements over which to operate. As the nodes in the current color are perturbed, a temporary collection of edges and cells within their stencils is gathered as shown in Fig. 2. By supplying the residual routines with only those elements required to compute residuals within a stencil width of the nodes in the current color, the overall cost is reduced substantially. In the case of a color containing a single grid point, the residual evaluation now takes place over several dozen edges and cells, as opposed to the millions that might be present in the entire grid.

Strong Boundary Conditions

The backward-Euler time-integration strategy used in the flow solver results in a linear system of equations at each time step n that takes the following general form:

$$\left(\frac{V}{\Delta t_n}\boldsymbol{I} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{Q}^n}\right)\Delta \boldsymbol{Q}^n = -\boldsymbol{R}(\boldsymbol{Q}^n) \tag{5}$$

with $\boldsymbol{Q}^{n+1} = \boldsymbol{Q}^n + \Delta \boldsymbol{Q}^n$. For viscous flows, no-slip and prescribed wall-temperature boundary conditions are imposed using a strong enforcement at solid surfaces.[47] Whereas the continuity equation at the boundary is formed and solved in the same manner as in the interior, the energy at the wall is directly related to the density through the following expression, where a w subscript indicates a wall quantity:

$$E_w = [T_w/\gamma(\gamma-1)]\rho_w \tag{6}$$

Along with the no-slip condition, this relationship is used to modify the diagonal block for rows of the linear system in Eq. (5) corresponding to grid points on viscous walls (associated off-diagonal entries are set to zero):

$$\begin{bmatrix} D_{11} & D_{12} & D_{13} & D_{14} & D_{15} & D_{16} \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -T_w/\gamma(\gamma-1) & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta\rho_w \\ \Delta(\rho u)_w \\ \Delta(\rho v)_w \\ \Delta(\rho w)_w \\ \Delta E_w \\ \Delta\tilde{\nu}_w \end{bmatrix} = -\begin{bmatrix} R_1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{7}$$

Note that the components for the momentum, energy, and turbulence equations in the right-hand side of Eq. (7) are set to zero at the end of a residual evaluation, whereas the residual at the wall is formally given by

$$\boldsymbol{R}_w = \begin{bmatrix} R_1 \\ (\rho u)_w \\ (\rho v)_w \\ (\rho w)_w \\ E_w - [T_w/\gamma(\gamma-1)]\rho_w \\ \tilde{\nu}_w \end{bmatrix} \tag{8}$$

This has important ramifications in forming the Jacobians by using the complex-variable technique. Evaluating a complex form of the residual will result in identically zero elements for the linearizations of the momentum, energy, and turbulence equations at the wall. To remedy this, the Jacobian elements corresponding to these equations are explicitly set according to Eq. (7) once the entire matrix has been assembled. Although this extra step is straightforward, it would not be necessary if the residual vector was formed according to Eq. (8). This detail will be addressed again in a subsequent section.

Cost Function Linearizations

Unlike a residual computation where the output is a vector of quantities associated with each grid point, the cost functions used in the current work are composed of boundary integrals that yield scalar quantities such as lift and drag. This implies that only a single contribution to $\partial f/\partial \boldsymbol{Q}$ or $\partial f/\partial \boldsymbol{X}$ can be determined by a complex force evaluation. For this reason, multiple perturbations cannot be performed simultaneously as with the residual contributions. However,

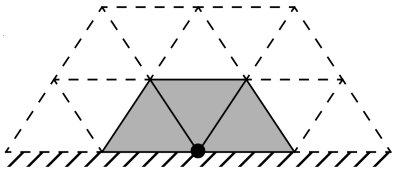

Fig. 3 Local boundary elements provided to complex-valued force routines: ●, a perturbation.

similar to the strategy used for residual computations, the complex-valued force routines are restricted to a subset of boundary elements that are affected by a perturbation as shown in Fig. 3. The need to perform boundary perturbations in a sequential fashion does not have a considerable impact on the overall efficiency of the scheme, as the boundary integrals are generally inexpensive and the boundaries are typically much smaller than the domain as a whole.

For parallel computations, the grid is partitioned without knowledge of surface information. For this reason, the surfaces contributing to the cost function are in general not evenly distributed across processors, and the construction of the cost function linearizations is not load balanced. This has not been found to cause a serious performance penalty.

Distance Function Linearizations

For turbulent flows, the one-equation model used in the current work contains a source term that depends on the distance to the nearest solid wall. This dependency enters the linearization $\partial \boldsymbol{R}/\partial \boldsymbol{X}$ and is the only quantity in the entire solver that depends on values outside the next-nearest neighbor stencil. For this reason, the coloring scheme described earlier cannot be used for simultaneous perturbations of grid coordinates to construct the distance function contribution to $\partial \boldsymbol{R}/\partial \boldsymbol{X}$. Moreover, coloring the stencil pattern associated with the distance function would be cumbersome and likely result in a very inefficient scheme.

Similar to the cost function linearizations, the derivatives of the residual contributions involving the distance function were initially constructed by applying perturbations in a sequential fashion. However, unlike the cost function, which depends at most on the surface values and their nearest neighbors, the distance function linearizations depend on every grid point in the field. For large-scale problems, it has been found that constructing these contributions in a sequential manner is prohibitively expensive. For this reason, the hand-coded implementation of these terms developed in Refs. 28–31 is used, wherein the nearest element on the surface is stored for every grid point in the field, so that the distance function at each grid point can be differentiated very efficiently. A similar scheme could certainly be constructed for the complex-variable approach; however, this has not been pursued. Other limitations of large stencil widths will be discussed in a subsequent section.

Computing the Adjoint Residual

Because $\boldsymbol{Q}$ is fixed for the adjoint problem, the terms $[\partial \boldsymbol{R}/\partial \boldsymbol{Q}]^T$ and $\partial f/\partial \boldsymbol{Q}$ are formed and stored at the start of a computation using the strategies already outlined. As a consequence, the adjoint residual on the right-hand side of Eq. (1) becomes an explicit matrix–vector product at each time step. This is in contrast to the hand-coded method presented in Refs. 28–31, where only the nearest-neighbor terms were stored and the higher-order pieces were recomputed at each time step in order to save memory. A similar strategy could be used for the complex-variable implementation; however, the computational cost associated with recomputing terms at each time step would be prohibitive. As compared to the hand-coded implementation, the current approach requires considerably more CPU time and memory to form and store the linearizations required for Eq. (1); however, the subsequent performance of the adjoint residual computation yields an overall computational savings that will be demonstrated next.

Consistency of Linearization

To verify the accuracy of the implementation, a comparison is made using three discrete methods for obtaining sensitivity derivatives as listed in Table 1. The first method is a direct form of differentiation obtained by converting the entire flow solver to a complex-variable formulation. This process has also been fully automated and is described in Refs. 42 and 43. The second approach used to compute the linearizations is the hand-coded discrete adjoint technique of Refs. 28–31. Finally, the third method is the hybrid approach of the current work where a complex-variable formulation is used to form the discrete adjoint system. All equation sets have been converged to machine precision.

Table 1 Schemes used to obtain sensitivities

Method	Linearization algorithm
1	Direct differentiation via automated complex variables
2	Hand-coded discrete adjoint
3	Automated complex-variable discrete adjoint

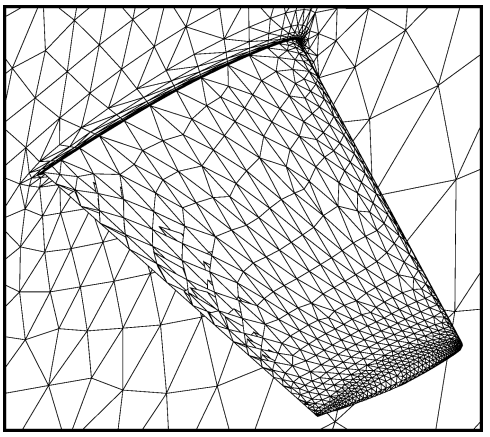

Fig. 4 Surface grid for ONERA M6 wing configuration.

Sensitivity derivatives of the lift and drag coefficients for the ONERA M6 wing[51] shown in Fig. 4 are computed for fully turbulent flow using each of the methods just described. The mesh contains 16,391 nodes and 90,892 tetrahedra. The freestream Mach number is 0.84, the angle of attack is 3.06 deg, and the Reynolds number is 10^6 based on the mean aerodynamic chord. The surface grid has been parameterized using the method of Ref. 52. All of the computations have been performed using 12 processors.

Sensitivity derivatives of the lift and drag coefficients for several shape parameters located at the midspan of the wing are listed in Table 2. The results of the three approaches are in excellent agreement, with discrepancies present in only the 11th decimal place or better. For turbulent flows, the last several digits are often still fluctuating despite machine precision convergence.

Large-Scale Performance

To evaluate the current scheme on a large-scale problem, fully turbulent flow over the transport wing-body shown in Fig. 5 is computed using 64 processors. The grid for this case contains 1,731,262 nodes and 10,197,838 tetrahedra. The freestream Mach number is 0.84, the angle of attack is 2.25 deg, and the Reynolds number is 3×10^6 based on the mean aerodynamic chord. For this test, the objective function is the drag coefficient. Although this case was previously shown in Ref. 31, the performance trends shown here cannot be directly compared with the prior results, as the mean flow and turbulence equations have been solved in a loosely coupled fashion in the current work, as opposed to the tightly coupled solution procedure utilized in Ref. 31. The extra Jacobian terms required for a tightly coupled flow solution can have a considerable impact on the relative performance of the flow and adjoint solvers.

The iterative convergence of the flow solver as well as the hand-coded and complex-variable adjoint solvers are shown in Fig. 6. After 3000 time steps, the two histories exhibit similar asymptotic convergence rates for the density and turbulence equations and their adjoint counterparts, as guaranteed by the exact dual nature of the iterative algorithms. Note that the complex-variable adjoint

Table 2 Sensitivity derivatives for lift and drag coefficients using various approaches

Objective function	Method	Design variable			
		Thickness	Shear	Camber	Twist
C_L	1	−0.584383430968430	−0.073891855284066	1.843734584180741	−0.022010251214990
	2	−0.584383430968115	−0.073891855283921	1.843734584180955	−0.022010251214989
	3	−0.584383430968976	−0.073891855294895	1.843734584179641	−0.022010251215037
C_D	1	0.058894900355748	−0.006835640271421	0.064393773359690	−0.001817294278046
	2	0.058894900355780	−0.006835640271392	0.064393773359720	−0.001817294278046
	3	0.058894900355586	−0.006835640272820	0.064393773359577	−0.001817294278054

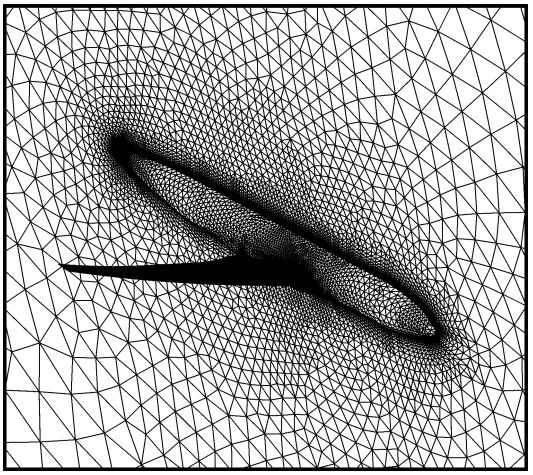

Fig. 5 Surface grid for modern transport configuration.

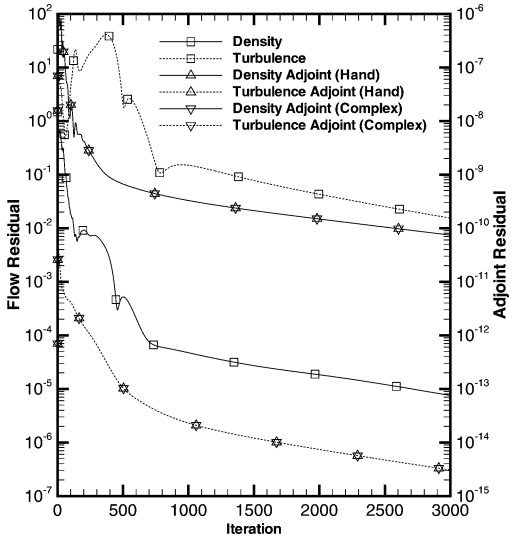

Fig. 6 Residuals vs iteration for modern transport configuration.

scheme lies exactly on top of the hand-coded implementation as would be expected; any discrepancy would indicate an error in the implementation.

On a per-processor basis for the current test case, the flow solver uses 92 MB of memory, the handcoded adjoint solver requires 220 MB, and the complex-variable adjoint solver uses 630 MB. The discrepancy between the two adjoint implementations is because of the linearization storage strategies described earlier and is consistent with the discussion in Ref. 31. The benefit of storing the entire linearization can be seen in Fig. 7, where the convergence is plotted vs CPU time for each solution. The hand-coded adjoint solver requires approximately twice as long as the flow solver to perform 3000 time steps. However, because the matrix–vector product required by the residual in Eq. (1) is performed explicitly in the current approach, the complex-variable adjoint solver requires 60%

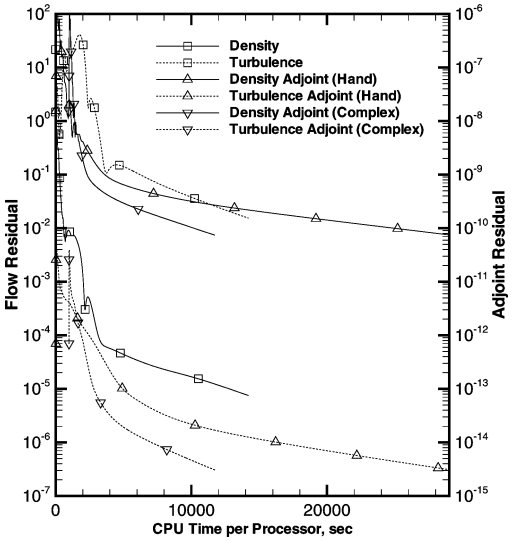

Fig. 7 Residuals vs CPU time for modern transport configuration.

less CPU time than the hand-coded implementation. A subtle feature in Fig. 7 is the y-axis offset for the complex-variable adjoint results (easiest to see for the turbulence equation). This is the initial setup time required to construct the exact linearizations of the residual and cost function using complex variables.

Extension to High-Temperature Gas Equation Sets

There has been a significant effort recently within the CFD community to provide accurate and robust hypersonic aerodynamic and aerothermodynamic capabilities within unstructured grid frameworks, and progress to date has resulted in significantly more elaborate flow solution algorithms. In addition to the nonlinear limiter functions necessary for supersonic flows, high-energy flow solvers typically contain curve fits for transport properties, eigenvalue limiters, a variable number of species and energy equations, and can employ embedded Newton iterations to determine thermodynamic properties or to implement boundary conditions.

Consider the hand-coded discrete adjoint implementation of Refs. 28–31 for the perfect-gas Reynolds-averaged Navier–Stokes equations. This capability has taken over five years to evolve into a mature capability for large-scale problems. Any extension to its basic functionality remains an extremely sobering undertaking, and manually extending it to include the additional complexities required for thermochemical nonequilibrium flows is simply untenable. This issue has instead served as the primary motivation for the current work, wherein an automatable, more efficient, and less error-prone procedure for developing a discrete adjoint solver for increasingly complex sets of governing equations has been sought.

Although obtaining reliable stagnation-point heating on purely tetrahedral grids is proving an elusive goal, the current adjoint formulation has been extended to include the high-temperature gas effects that have been recently added to the baseline solver.[43,53,54] This has been done to demonstrate the power of the current approach to forming discrete adjoint systems for more algorithmically complex equation sets. It is understood that any subsequent adjoint-based

design optimization or solution adaptation would be only as accurate as the underlying discretization; however, the ultimate value of the current approach lies in its ability to provide a discrete adjoint capability in a timely manner for a given discretization.

A detailed overview of the underlying hypersonic algorithms is presented in Ref. 54. An extensive suite of turbulence models has been implemented in the baseline solver; however, the adjoint formulation has not yet been tested in this regime. Only some basic implementation issues are discussed here, and a simple test case is shown to verify the accuracy of the approach and demonstrate its potential for future hypersonic applications. Areas requiring additional research are also identified.

Implementation Issues

Extension of the current automated adjoint formulation to include high-temperature gas effects is largely straightforward because the discretization stencil for the various terms is identical to those in the perfect-gas implementation. Therefore, the infrastructure developed to assemble the various linearizations using complex variables can be readily applied without modification. The additional thermodynamic and transport routines, as well as the source terms required for chemically reacting flows, have been modified to optionally operate on a localized subset of elements in the same manner as the basic flux and force routines, so that computations are performed only for contributions with nonzero imaginary parts. Strong boundary conditions are handled automatically, as the residual computation on boundaries for the hypersonic portion of the solver is implemented in a general manner analogous to Eq. (8).

The flow solver includes options to use temperature or energy as a fundamental variable. The use of energy (as in the perfect-gas case) requires a Newton subiteration to evaluate $T(e, \rho_i)$. When a complex perturbation is applied to an element of $\boldsymbol{Q}$ and this iterative strategy is invoked, the convergence criterion for this procedure is identically satisfied because the real part of the temperature has not changed from its baseline value, which presumably corresponds to the current value of the energy. However, the imaginary part of the temperature will be incorrect, as the iterations required to determine this component have been terminated immediately. Therefore, when this procedure is invoked in a complex-valued context the Newton algorithm is forced to perform 10 iterations, the maximum allowed for the real-valued case, to allow the imaginary part of the temperature to develop correctly.

Demonstration Case

A five-species air laminar flow cylinder test case is performed. Shown in Fig. 8, the grid used for this case contains 4040 nodes and 11,520 tetrahedra and has been derived from a structured grid similar to those used in Refs. 53 and 54. The grid contains a single layer of cells in the spanwise direction. Note that the structured grid cells have been diagonalized in a uniform manner so that a severe spanwise bias is present in the computation. This grid topology is being heavily relied upon in related work for "stress testing" the accuracy of various discretization schemes on tetrahedra.

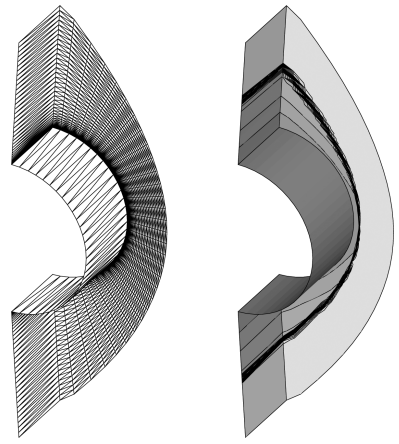

Fig. 8 Surface grid and Mach-number contours for cylinder computation. Darker shades indicate lower Mach numbers.

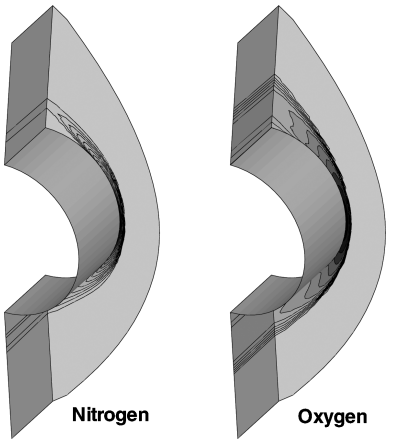

Fig. 9 Atomic nitrogen and oxygen species concentrations for cylinder computation. Darker shades indicate higher concentrations.

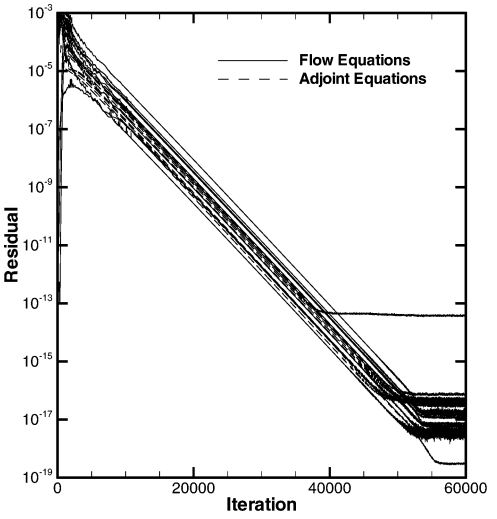

Fig. 10 Iterative convergence history of the flow and adjoint equations for cylinder computation.

For this test, the freestream velocity is 5000 m/s, the freestream density is 0.001 kg/m^3, and the temperature is 200 K. These conditions give a Reynolds number of approximately 4.25×10^5 based on the cylinder diameter and a freestream Mach number of 17.6. The flowfield is governed by nine conservation equations: five species equations (N_2, O_2, N, O, and NO) and the usual momentum and energy equations. The convective terms are only first-order accurate for this demonstration; a more detailed discussion on the reasons for this limitation will follow in a subsequent section. The computation has been performed on eight processors, and contours of the Mach number are shown in Fig. 8, where a strong bow shock can be seen upstream of the cylinder. The jagged shock capture is typical of a first-order scheme and is exacerbated by the lack of grid alignment in these regions. The temperature downstream of the shock in the leading-edge region exceeds 6300 K, causing the freestream molecules to dissociate; contours of atomic nitrogen and oxygen are shown in Fig. 9.

The convergence history of the nine flow equations and their adjoint counterparts for a drag-based objective function is plotted in Fig. 10. No attempt has been made to optimize the solution parameters; a constant Courant–Friedrichs–Lewy number of 1 is used with two point-implicit sweeps through the linearized problem at each time step. The equations exhibit similar asymptotic convergence rates as guaranteed by the exact dual implementation outlined in Ref. 31. The residual that seems to stall slightly earlier than the other equations corresponds to the adjoint variable for the species

Table 3 Sensitivity derivatives for lift, drag, and heating using various approaches

Objective function	Method	Design variable 1	2	3
C_L	1	0.079063607430172	0.040542677709286	0.036254858359808
	3	0.079063607430244	0.040542677709607	0.036254858359736
C_D	1	−0.090184472046108	0.052986957502726	0.048939766511917
	3	−0.090184472039446	0.052986957503082	0.048939766511939
q	1	0.092618219139863	0.052427889309767	−0.144725668575523
	3	0.092618219167176	0.052427889311309	−0.144725668580864

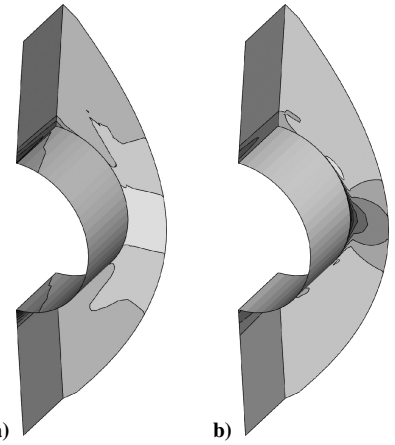

Fig. 11 Contours of the a) streamwise momentum adjoint variable for drag and b) O_2 species conservation adjoint variable for heating.

conservation of atomic nitrogen. The reason for this is unknown. Figure 11 shows contours of the streamwise momentum adjoint solution for drag, as well as the adjoint variable for the O_2 species conservation equation for an objective function based on the net heat flux to the cylinder surface.

To quantify the accuracy of the discrete adjoint implementation, sensitivity derivatives of the lift, drag, and surface heating with respect to the streamwise coordinates of three randomly chosen grid points on the cylinder surface are computed and shown in Table 3. Similar to the perfect-gas test case shown earlier, these derivatives are computed using two discrete methods already outlined in Table 1. The first method is the direct differentiation approach attained by converting the entire baseline solver to a complex-variable formulation. The second is the current discrete adjoint approach, using complex variables to obtain the required linearizations. The agreement is similar to that obtained for the perfect-gas results.

Further Research and Development Areas

One issue associated with using the complex-variable approach to form the discrete adjoint system for hypersonic flowfields is the memory required to store the complete linearization of the residual. For the next-nearest neighbor stencil used in the current implementation, the residual at a grid point generally depends on information from roughly 50 neighboring points. A typical hypersonic computation might be performed on a grid consisting of 10 million points; therefore, approximately 500 million nonzero subblocks will be present in the Jacobian matrix. For five-species air with a single energy equation model, nine governing equations are required, so that each subblock in the Jacobian matrix will contain 81 entries. If standard eight-byte double-precision variables are used to store each of these values, approximately 320 GB of memory will be required to store the complete linearization. This represents a substantial amount of memory, even on the largest computing systems currently available, and opportunities to alleviate this memory requirement should be investigated. Moreover, this issue lends further motivation to the pursuit of adjoint-based grid adaptation,[36,37] where grid points are concentrated only in areas that have the highest impact on the output of interest, thereby avoiding unnecessary grid resolution in irrelevant regions of the flowfield.

Another concern in applying the adjoint technique is the requirement that the flowfield solution be linearly stable. At the conclusion of a flowfield computation, the solution might appear satisfactory in an engineering sense; forces have converged to some tolerance, and the residuals of the nonlinear system have been reduced to some acceptable level. However, the flowfield can, in fact, contain some linearly unstable modes. These modes can often be bounded or stabilized by nonlinearities present in the flowfield computation; however, the adjoint system has no such control mechanisms, and any instabilities will amplify and cause the solution to diverge. The need for flux-limiting strategies in second-order accurate hypersonic computations can contribute to this problem; it is for this reason that no second-order-accurate results have been presented here for hypersonic flows. An ability to monitor diagnostics of the linearized system of equations and address such instabilities would be a valuable capability and should be a focus of future work. This requirement for linear stability has occasionally been a problem for turbulent perfect-gas flows, and similar issues have also been reported in Refs. 55 and 56.

The presence of trace species has been found to cause sporadic problems in solving the adjoint system of equations. For example, in a five-species air computation the freestream concentrations of N, O, and NO are set to 1×10^{-10} in the current implementation. Adjoint computations for such flowfields have occasionally shown a tendency to diverge, and increasing the species concentrations in the freestream has been found to overcome this difficulty. The exact cause of this breakdown has not yet been investigated in detail.

Finally, the efficiency problems associated with linearizing the distance function as a result of its inherently large stencil may foreshadow similar difficulties to be encountered in forming discrete adjoint systems for problems governed by integrodifferential equations such as magnetohydrodynamic applications and flowfields involving radiation. Discretizations of these types of systems generally involve noncompact stencils, and, therefore, their linearizations may also be prohibitively expensive to construct using a complex-variable formulation.

Despite these technical challenges, the current approach is an enabling technology for pursuing rigorous design optimization and adaptation for high-energy flows. The perfect-gas implementation has proved invaluable for a wide range of vehicle concepts. Aerodynamic optimizations for full aircraft configurations using large numbers of design variables have been performed with minimal expense,[46] and adjoint-based mesh adaptation and error estimation have been used to efficiently obtain grid-converged solutions for high Reynolds number, geometrically complex flowfields.[36,37] Furthermore, ongoing work is aimed at coupling these capabilities to enable simultaneous design and adaptation. This coupling will not only drastically reduce design cycle time but, perhaps more importantly, provide error bounds on the result. The extension of these technologies to high-temperature gas equation sets will allow these computations to span the speed range, eventually encompassing scramjets, interplanetary probes, and manned space exploration vehicles.

Summary

A new technique for obtaining exact linearizations of complicated real-valued residual operators and cost functions necessary for discrete adjoint computations has been described. The method has been implemented for turbulent flows within a three-dimensional unstructured grid framework, in which the complex-valued source code is

generated by using an automated scripting procedure. A number of efficiency issues have been addressed as well as implementation details. Sensitivity derivatives computed using the new scheme are in excellent agreement with results from a discrete direct approach as well as a previous hand-coded discrete adjoint implementation. Because the new scheme stores the complete linearization of the residual, the method requires considerably more memory than the existing hand-coded approach. However, this reduces the adjoint residual computation to an explicit matrix–vector product so that the overall computational cost for large-scale problems is reduced.

To demonstrate the power of the new approach, the method has also been extended to include finite-rate chemistry models necessary for hypersonic flows. Sensitivity derivatives for five-species reacting air have been computed using the new scheme, and agreement with a discrete direct approach has been demonstrated.

The impact of the new approach on the software development cycle necessary to achieve a discrete adjoint capability is difficult to overstate. The previous hand-coded implementation required on the order of five years to mature into a robust tool suitable for everyday large-scale perfect-gas turbulent flow applications. By using the new complex-variable approach to achieve the required linearizations, an equivalent capability for turbulent flows was achieved with only six weeks of development effort. The experience and software infrastructure gained through the lengthy hand differentiation process certainly provided an excellent foundation for the new effort; however, despite this head start the new method has the potential to reduce the software development cycle by an order of magnitude or more. Although a number of issues warrant further research, the current scheme has opened the door to rigorous adjoint-based hypersonic aerodynamic and aerothermodynamic design optimization and solution adaptation, which up until recently were mere visions.

Acknowledgments

The authors thank Kyle Anderson of the University of Tennessee at Chattanooga; Peter Gnoffo, Michael Park, and Bill Wood of NASA; and David Darmofal of the Massachusetts Institute of Technology for helpful discussions pertaining to the current work. Jamshid Samareh of NASA is also acknowledged for providing geometric parameterizations for this study.

References

[1]Hicks, R. M., and Henne, P. A., "Wing Design by Numerical Optimization," *Journal of Aircraft*, Vol. 15, No. 7, 1978, pp. 407–412.

[2]Joh, C.-Y., Grossman, B., and Haftka, R. T., "Design Optimization of Transonic Airfoils," *Engineering Optimization*, Vol. 21, No. 1, 1993, pp. 1–20.

[3]Vanderplaats, G. N., Hicks, R. N., and Murman, E. M., "Application of Numerical Optimization Techniques to Airfoil Design," NASA SP-347, Pt. 2, March 1975.

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

[5]Borggaard, J. T., Burns, J., Cliff, E. M., and Gunzburger, M. D., "Sensitivity Calculations for a 2-D Inviscid Supersonic Forebody Problem," *Identification and Control Systems Governed by Partial Differential Equations*, Society for Industrial and Applied Mathematics, Philadelphia, 1993, pp. 14–24.

[6]Burgreen, G. W., and Baysal, O., "Aerodynamic Shape Optimization Using Preconditioned Conjugate Gradient Methods," *AIAA Journal*, Vol. 32, No. 11, 1994, pp. 2145–2152.

[7]Hou, G. J.-W., Maroju, V., Taylor, A. C., and Korivi, V. M., "Transonic Turbulent Airfoil Design Optimization with Automatic Differentiation in Incremental Iterative Forms," AIAA Paper 95-1692, June 1995.

[8]Newman, J. C., and Taylor, A. C., "Three-Dimensional Aerodynamic Shape Sensitivity Analysis and Design Optimization Using the Euler Equations on Unstructured Grids," AIAA Paper 96-2464, June 1996.

[9]Sherman, L. L., Taylor, A. C., Green, L. L., Newman, P. A., Hou, G. J.-W., and Korivi, V. M., "First- and Second-Order Aerodynamic Sensitivity Derivatives via Automatic Differentiation with Incremental Iterative Methods," AIAA Paper 94-4262, Sept. 1994.

[10]Young, D. P., Huffman, W. P., Melvin, R. G., Bieterman, M. B., Hilmes, C. L., and Johnson, F. T., "Inexactness and Global Convergence in Design Optimization," AIAA Paper 94-4386, Sept. 1994.

[11]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.

[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, 1999, pp. 443–480.

[13]Elliott, J., "Discrete Adjoint Analysis and Optimization with Overset Grid Modelling of the Compressible High-Re Navier–Stokes Equations," 6th Overset Grid and Solution Technology Symposium, Oct. 2002.

[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]Giles, M. B., and Pierce, N. A., "An Introduction to the Adjoint Approach to Design," *Flow, Turbulence and Combustion*, Vol. 65, Nos. 3 and 4, 2000, pp. 393–415.

[16]Giles, M. B., "On the Use of Runge–Kutta Time-Marching and Multigrid for the Solution of Steady Adjoint Equations," AD2000 Conf. in Nice, June 2000.

[17]Giles, M. B., "Adjoint Code Developments Using the Exact Discrete Approach," AIAA Paper 2001-2596, June 2001.

[18]Giles, M. B., "On the Iterative Solution of Adjoint Equations," *Automatic Differentiation: From Simulation to Optimization*, edited by G. Corliss, C. Faure, A. Griewank, L. Hascoet, and U. Naumann, Springer-Verlag, New York, 2001, pp. 145–152.

[19]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, 2003, pp. 198–205.

[20]Iollo, A., Salas, M. D., and Ta'asan, S., "Shape Optimization Governed by the Euler Equations Using an Adjoint Method," ICASE, Rept. 93-78, Nov. 1993.

[21]Jameson, A., "Aerodynamic Design Via Control Theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233–260.

[22]Jameson, A., Pierce, N. A., and Martinelli, L., "Optimum Aerodynamic Design Using the Navier–Stokes Equations," AIAA Paper 97-0101, Jan. 1997.

[23]Kim, C. S., Kim, C., and Rho, O. H., "Sensitivity Analysis for the Navier–Stokes Equations with Two-Equation Turbulence Models," *AIAA Journal*, Vol. 39, No. 5, 2001, pp. 838–845.

[24]Kim, H.-J., Sasaki, D., Obayashi, S., and Nakahashi, K., "Aerodynamic Optimization of Supersonic Transport Wing Using Unstructured Adjoint Method," *AIAA Journal*, Vol. 39, No. 6, 2001, pp. 1011–1020.

[25]Mohammadi, B., "Optimal Shape Design, Reverse Mode of Automatic Differentiation and Turbulence," AIAA Paper 97-0099, Jan. 1997.

[26]Nemec, M., and Zingg, D. W., "Towards Efficient Aerodynamic Shape Optimization Based on the Navier–Stokes Equations," AIAA Paper 2001-2532, June 2001.

[27]Newman, J. C., III, Taylor, A. C., III, and Burgreen, G. W., "An Unstructured Grid Approach to Sensitivity Analysis and Shape Optimization Using the Euler Equations," AIAA Paper 95-1646, June 1995.

[28]Nielsen, E. J., "Aerodynamic Design Sensitivities on an Unstructured Mesh Using the Navier–Stokes Equations and a Discrete Adjoint Formulation," Ph.D. Dissertation, Dept. of Aerospace and Ocean Engineering, Virginia Polytechnic Inst. and State Univ., Blacksburg, VA, Dec. 1998.

[29]Nielsen, E. J., and Anderson, W. K., "Aerodynamic Design Optimization on Unstructured Meshes Using the Navier–Stokes Equations," *AIAA Journal*, Vol. 37, No. 11, 1999, pp. 1411–1419.

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

[31]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, 2004, pp. 1131–1155.

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

[33]Soemarwoto, B., "Multipoint Aerodynamic Design by Optimization," Ph.D. Dissertation, Dept. of Theoretical Aerodynamics, Delft Univ. of Technology, Delft, The Netherlands, Dec. 1996.

[34]Soto, O., and Lohner, R., "A Mixed Adjoint Formulation for Incompressible Turbulent Problems," AIAA Paper 2002-0451, Jan. 2002.

[35]Sung, C., and Kwon, J. H., "Aerodynamic Design Optimization Using the Navier–Stokes and Adjoint Equations," AIAA Paper 2001-0266, Jan. 2001.

[36]Venditti, D., and Darmofal, D., "Anisotropic Grid Adaptation for Functional Outputs: Application to Two-Dimensional Viscous Flows," *Journal of Computational Physics*, Vol. 187, No. 1, 2003, pp. 22–46.

[37]Park, M. A., "Three-Dimensional Turbulent RANS Adjoint-Based Error Correction," AIAA Paper 2003-3849, June 2003.

[38]Biedron, R. T., Samareh, J. A., and Green, L. L., "Parallel Computation of Sensitivity Derivatives with Application to Aerodynamic Optimization of a Wing," *1998 Computational Aerosciences Workshop*, NASA CP-20857, 1999, pp. 219–224.

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

[40]Lyness, J. N., "Numerical Algorithms Based on the Theory of Complex Variables," *Proceedings of the ACM 22nd National Conference*, edited by S. Rosenthal, Vol. 1, Thomas Book Co., Washington, DC, 1967, pp. 124–134.

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

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

[43]Kleb, W. L., Nielsen, E. J., Gnoffo, P. A., Park, M. A., and Wood, W. A., "Collaborative Software Development in Support of Fast Adaptive Aerospace Tools (FAAST)," AIAA Paper 2003-3978, June 2003.

[44]Chorin, A. J., "A Numerical Method for Solving Incompressible Viscous Flow Problems," *Journal of Computational Physics*, Vol. 2, No. 1, 1967, pp. 12–26.

[45]Spalart, P. R., and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92-0439, Jan. 1991.

[46]Nielsen, E. J., and Park, M. A., "Using an Adjoint Approach to Eliminate Mesh Sensitivities in Computational Design," AIAA Paper 2005-0491, Jan. 2005; also *AIAA Journal* (to be published).

[47]Anderson, W. K., and Bonhaus, D. L., "An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids," *Computers and Fluids*, Vol. 23, No. 1, 1994, pp. 1–21.

[48]Anderson, W. K., Rausch, R. D., and Bonhaus, D. L., "Implicit/Multigrid Algorithms for Incompressible Turbulent Flows on Unstructured Grids," *Journal of Computational Physics*, Vol. 128, No. 2, 1996, pp. 391–408.

[49]Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[50]Van Leer, B., "Flux Vector Splitting for the Euler Equations," *Lecture Notes in Physics*, Vol. 170, No. 507, 1982, pp. 501–512.

[51]Schmitt, V., and Charpin, F., "Pressure Distributions on the ONERA M6 Wing at Transonic Mach Numbers," *Experimental Database for Computer Program Assessment*, AGARD-AR-138, 1979, pp. B1-1–B1-44.

[52]Samareh, J. A., "A Novel Shape Parameterization Approach," NASA TM-1999-209116, May 1999.

[53]Gnoffo, P. A., "Computational Fluid Dynamics Technology for Hypersonic Applications," AIAA Paper 2003-3259, July 2003.

[54]Gnoffo, P. A., and White, J. A., "Computational Aerothermodynamic Simulation Issues on Unstructured Grids," AIAA Paper 2004-2371, June 2004.

[55]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, June 1998.

[56]Campobasso, M., and Giles, M., "Effects of Flow Instabilities on the Linear Analysis of Turbomachinery Aeroelasticity," *Journal of Propulsion and Power*, Vol. 19, No. 2, 2003, pp. 250–259.

E. Livne
Associate Editor